

---

# **Documentazione di Plone**

*Release 4*

**La comunità Italiana di Plone**

July 31, 2015



<b>1</b>	<b>Plone 4: Manuale utente</b>	<b>1</b>
1.1	Introduzione . . . . .	1
1.2	Aggiungere contenuti . . . . .	20
1.3	Gestione dei contenuti . . . . .	49
1.4	Usare TinyMCE come visual editor . . . . .	79
1.5	Usare Kupu come visual editor . . . . .	92
1.6	Collaborazione e flusso di lavoro . . . . .	103
1.7	Utilizzo delle collezioni . . . . .	115
1.8	Gestione delle Portlet . . . . .	128
<b>2</b>	<b>Altri manuali</b>	<b>135</b>
2.1	Creare un tema con Diazo . . . . .	135
2.2	ZODB - un database nativo ad oggetti per Python . . . . .	151
2.3	La guida completa alla Zope Component Architecture . . . . .	162
<b>3</b>	<b>Credits e ringraziamenti</b>	<b>241</b>



---

## Plone 4: Manuale utente

---

Il manuale per utenti, redattori e amministratori.

### 1.1 Introduzione

Una panoramica dei concetti di Plone

#### 1.1.1 Panoramica

**Una spiegazione di Plone come content management system.**

##### Cos'è Plone?

Plone è un sistema di gestione dei contenuti (CMS) che permette di costruire un sito web. Con Plone, anche chi ha poca esperienza può contribuire alla creazione dei contenuti di un sito senza l'aiuto di un mago del computer. Inoltre Plone “gira sul Web”, quindi non c'è bisogno di installare alcun software speciale sul proprio computer. La parola *contenuto* vuole essere generale, in quanto è possibile pubblicare molti tipi di informazioni, tra cui:

Un sito web Plone contiene diversi tipi di contenuto, compresi testi, foto e immagini. Questi possono esistere in molte forme: documenti, notizie, eventi, video, file audio e tutti i tipi di file e dati che possono essere caricati o creati su un sito web. I contenuti possono anche essere caricati dal proprio computer. In un sito Plone puoi creare delle *cartelle* per raccogliere i contenuti e per definire una struttura di navigazione:

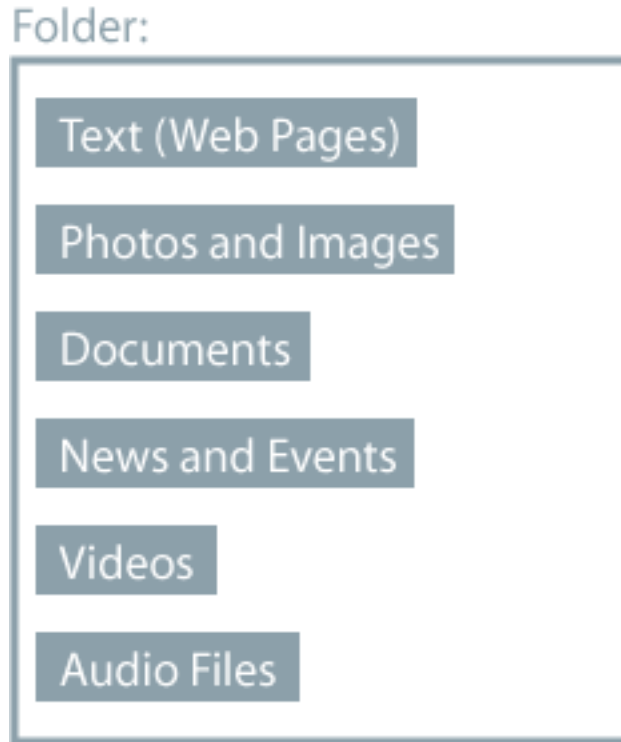
##### Ti piacciono le farfalle

Ad esempio, per pubblicare un contenuto sulle farfalle, potresti creare una cartella denominata “Farfalle” e poi aggiungere del testo a una pagina Web nella cartella:

Poi potresti caricare alcune foto di farfalle nella cartella:

In una cartella puoi aggiungere vari tipi di contenuto, comprese delle sotto-cartelle. Dopo aver inserito alcune notizie e video nella cartella Farfalle, il contenuto potrebbe essere organizzato in questo modo, con due sottocartelle all'interno della cartella Farfalle:





### Cosa succede dietro le quinte

Ci si potrebbe chiedere come funziona tutto questo. Un tipico sito web Plone esiste come installazione del software Plone su un server web. Il web server può essere ovunque, spesso si trova su un server di società specializzate all'interno di un "rack" di computer dedicati al compito:

Il diagramma mostra i cavi che collegano i singoli server a Internet, attraverso connessioni di rete veloci. Il sito Plone è prodotto dal software e da un database installati su uno dei server. Quando digiti o clicchi sul tuo computer, i dati vengono inviati su e giù per i cavi di rete e dei canali di comunicazione di Internet per interagire con il software Plone installato sul server.

Ora semplifichiamo un pò il diagramma che mostra come interagire con Plone:

Puoi utilizzare il tuo browser web - Firefox, Safari, Internet Explorer, ecc - per visualizzare e modificare il tuo sito web Plone, e le modifiche vengono memorizzate dal software Plone nel suo sistema di archiviazione.

Per esempio, immagina che il tuo sito web Plone sulle farfalle si trovi su [mysite.com](http://mysite.com). In questo caso dovresti digitare [www.mysite.com](http://www.mysite.com) nel tuo web browser. Dopo aver premuto Invio, inizia la seguente sequenza di eventi quando il tuo browser "parla" con il server web su [mysite.com](http://mysite.com):

e il sito Plone risponde con:

Plone legge il suo database per cercare informazioni memorizzate in [mysite.com](http://mysite.com). Quindi restituisce la pagina web al tuo computer, in un codice chiamato HTML. HTML è un linguaggio per computer che descrive come una pagina web appare. Include testo, grafica, font, il colore dello sfondo, e tutto il resto. Ci sono molte risorse online che possono insegnarti i dettagli di HTML, ma uno dei vantaggi di Plone è che non c'è bisogno di sapere (molto) di HTML. Questo è uno dei motivi per cui esistono Plone e altri software web simili, perchè ti permettono di concentrarti sui tuoi contenuti, come testo e grafica delle farfalle, invece di imparare un nuovo linguaggio del computer.

Ma torniamo alla nostra panoramica. Il tuo browser "renderizza" (traduce) questo HTML, e viene visualizzata la seguente pagina web:

## Introduction

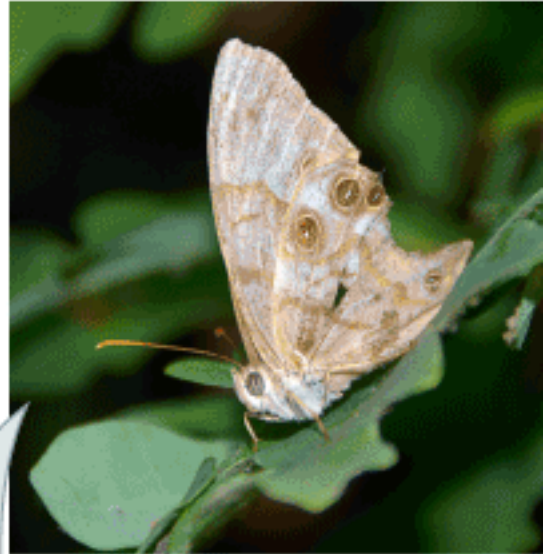
A butterfly is an insect of the order Lepidoptera. The scope of the term depends on how far the concept of "butterfly" is extended. Currently, most experts include the superfamilies Hedyloidea (the American moth-butterflies), Hesperioidea (the skippers) and Papilionoidea (the so-called 'true' butterflies). This concept of butterflies including the Hedyloidea is an only recently extended one, but it makes the group a natural clade, the Rhopalocera. Like all Lepidoptera, butterflies are notable for their unusual life cycle with a larval caterpillar stage, an inactive pupal stage and a spectacular metamorphosis into a familiar and colourful winged adult form, and most species being day-flying, they regularly attract attention. The diverse patterns formed by their brightly coloured wings and their erratic-yet graceful-flight have made butterfly watching a popular hobby.

### Butterflies

- Text (Web Page)
- Photos and Images
- Documents
- News and Events
- Videos
- Audio Files



Butterfly 1



Butterfly 2



## Butterflies

Text (Web Pages)

Photos and Images

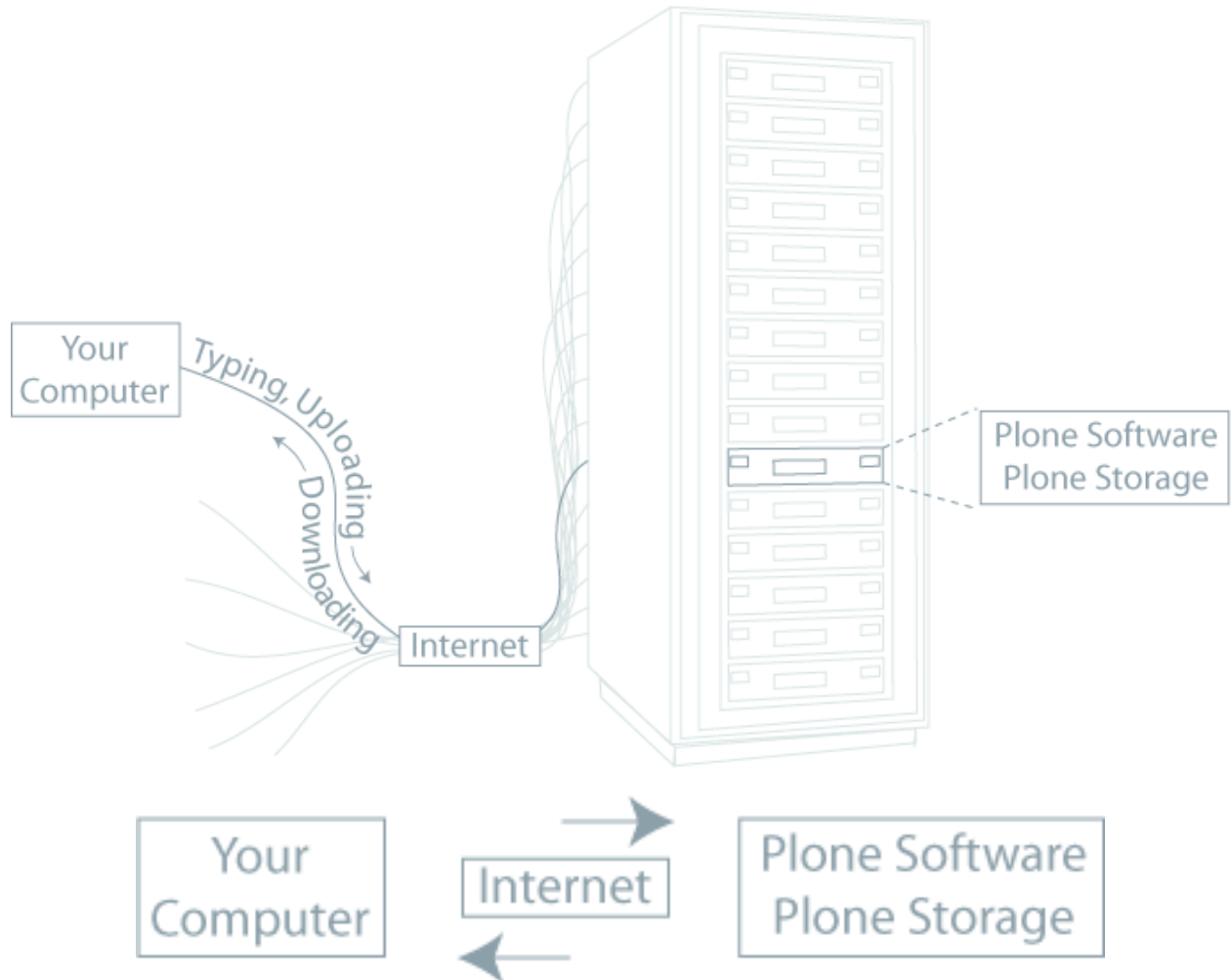
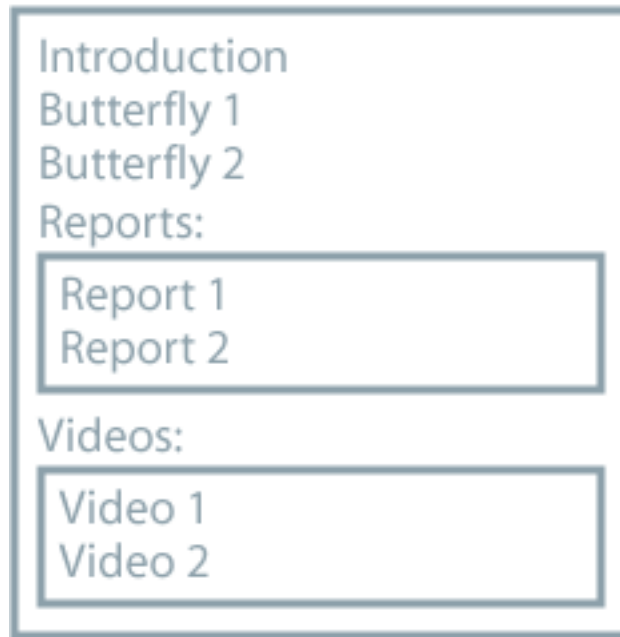
Documents

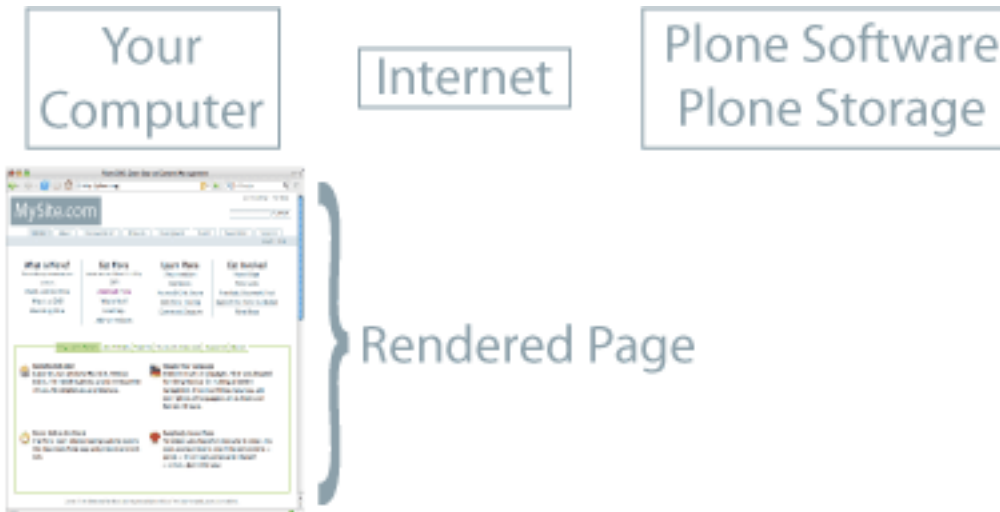
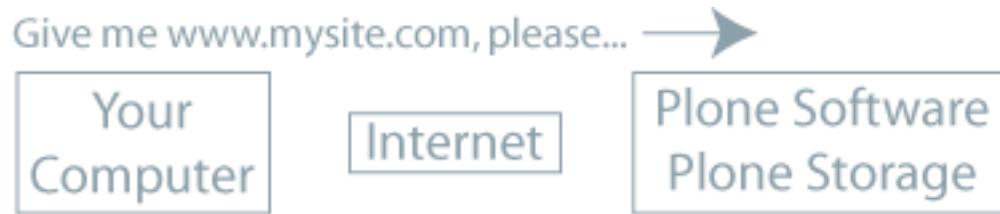
News and Events

Videos

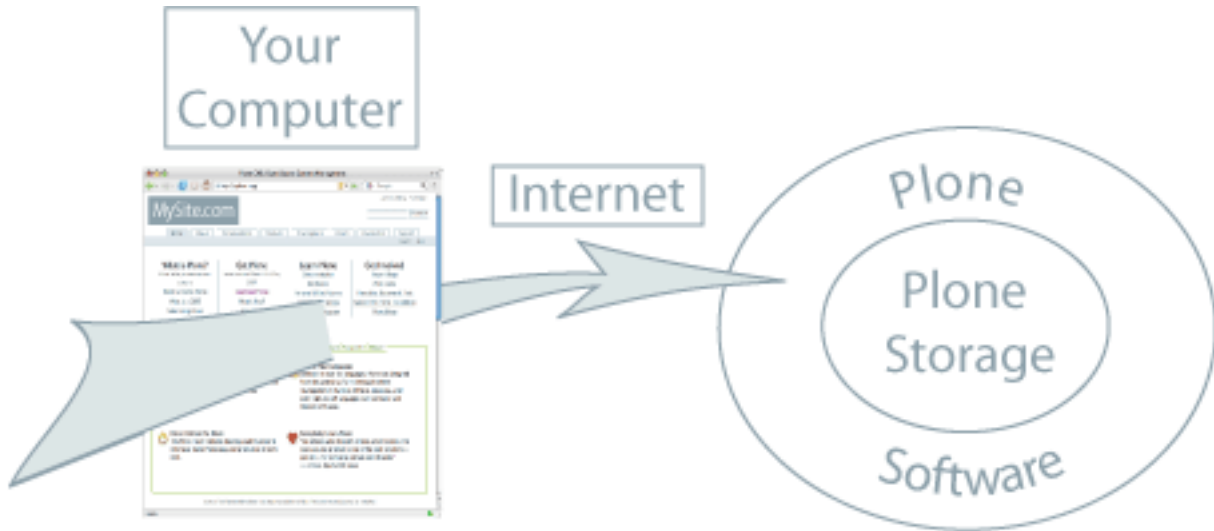
Audio Files

Butterflies:





Mentre stai guardando la pagina web della tua farfalla, puoi decidere di cambiare o aggiungere nuovo testo. È inoltre possibile caricare foto, documenti, ecc., in qualsiasi momento:

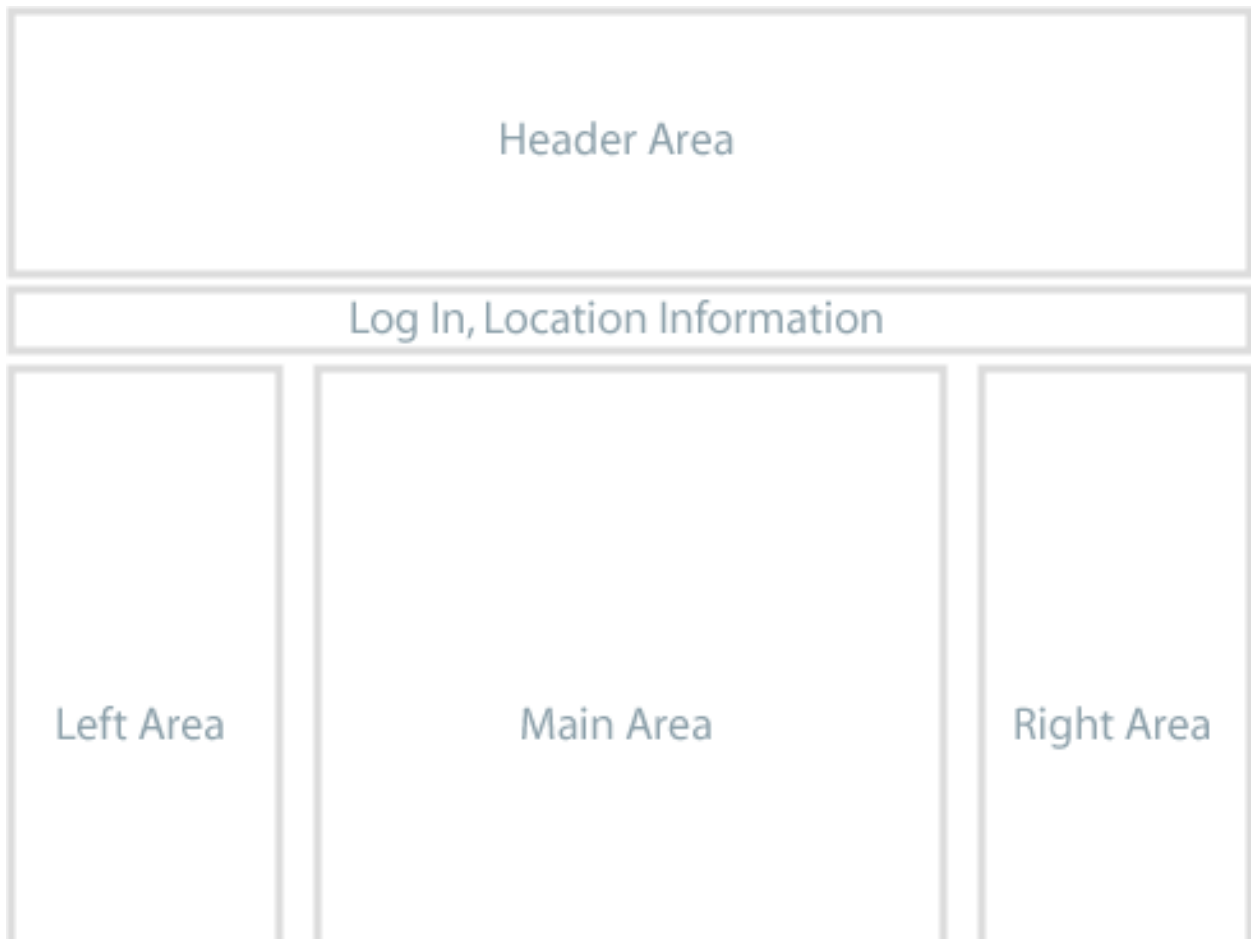
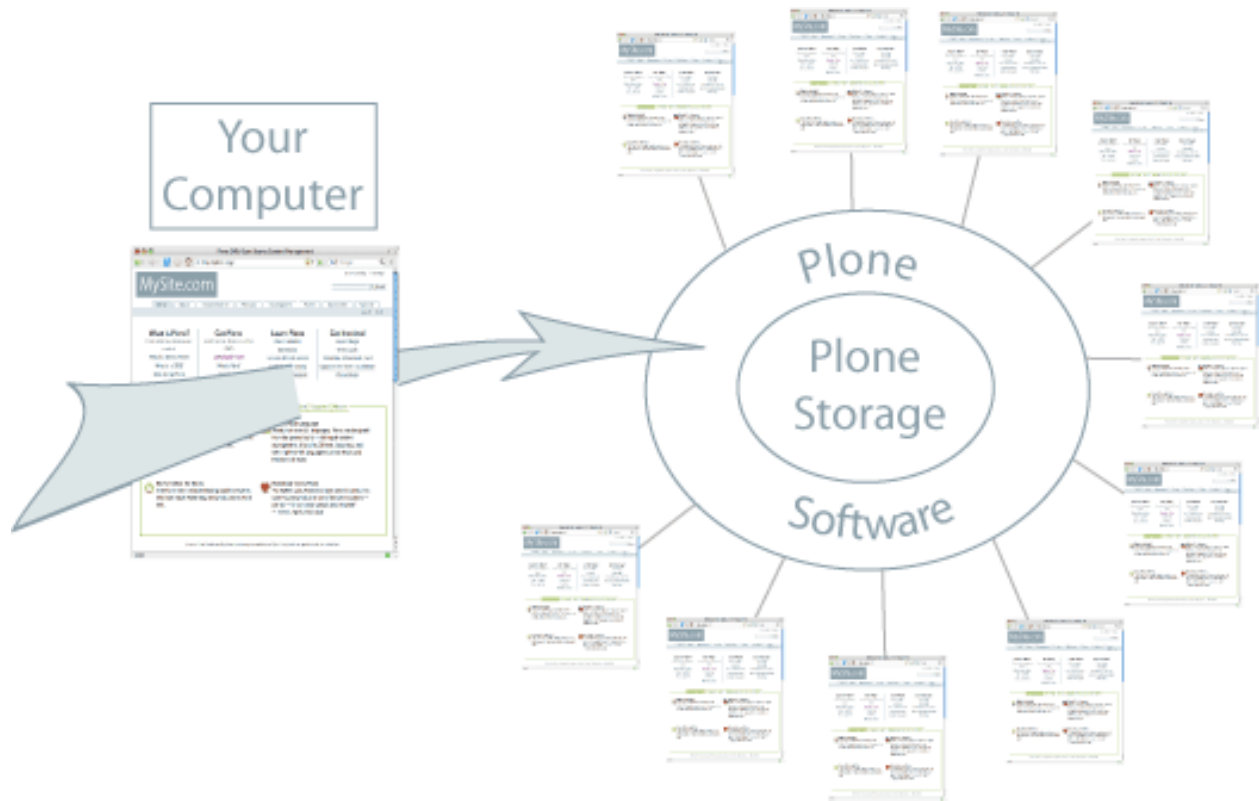


Dopo aver effettuato le modifiche e premuto su “salva modifiche”, la nuova versione della pagina web sarà immediatamente disponibile per chiunque navighi sul tuo sito:

### 1.1.2 Design Grafico dei Siti Web Plone

**Plone permette agli amministratori e ai designer dei siti web di creare design unici. Ecco una panoramica del layout Plone, e alcuni esempi di design.**

Come dovrebbe apparire un sito web Plone? Per anni c'è stato un design coerente per l'aspetto predefinito di Plone. Il design predefinito appare generalmente così:



Il tuo sito web Plone potrebbe avere un design radicalmente diverso da questo, ma dovresti essere in grado di trovare elementi comuni, come il link al log-in e un pannello o menu di navigazione. Nel design di default, il menu di navigazione si trova nella zona a sinistra, e di solito appare come un elenco indentato delle cartelle del sito. Ci può anche essere un insieme di schede nella striscia *Log In, Location Information* in testata.

Possiamo fare una distinzione tra il *design* e la *funzionalità* di un sito web. Per quanto riguarda i contenuti, concentrati sulla funzionalità e non preoccuparti tanto dell'aspetto e del layout del sito web. Un punto di forza del sistema di gestione dei contenuti Plone è che un sito web può essere radicalmente riprogettato con un nuovo look, senza incidere sul contenuto sottostante e sulle funzionalità. Il menu di navigazione potrebbe essere spostato da sinistra a destra, ma funzionerebbe lo stesso. L'area di destra potrebbe essere cancellata se le funzionalità che normalmente contiene non sono necessarie. Le aree sinistra, centrale e destra, come illustrato sopra e sotto, potrebbero essere spostate in alto, al centro e in basso, ma continuerebbe comunque a essere un sito web Plone.

Useremo il design del layout di default di Plone come esempio di tipica divisione dello schermo:

The screenshot shows the Plone 4 default layout with several red annotations. At the top left is the Plone logo. To its right is the word "TESTATA" in red. Further right is a search box with the text "Cerca nel sito" and a "Cerca" button, and a checkbox for "solo nella sezione corrente". Below this is a horizontal navigation menu with items: Home, Notizie (highlighted in blue), Eventi, and Collaboratori. The main content area is divided into three columns. The left column, labeled "COLONNA SINISTRA" in red, contains a sidebar with "Home", "Notizie" (with a sub-item "Una notizia di esempio"), "Eventi", and "Collaboratori", and a "Gestione portlet" button. The middle column, labeled "AREA PRINCIPALE" in red, shows a news article titled "Una notizia di esempio" with a "Visualizza" button and a "Stato: Pubblicato" dropdown. The right column, labeled "COLONNA DESTRA" in red, contains a calendar for November 2012 and a "Gestione portlet" button. At the bottom is a footer area labeled "FOOTER" in red, containing copyright information and links for "Mappa del sito", "Accessibilità", and "Contatti".

Potrebbe essere necessario adattare queste parti, se servono per il design del tuo sito web Plone. Ti potresti imbattere in diversi termini usati per descrivere le varie parti dello schermo, come ad esempio "slot" sinistro e destro, per le colonne di sinistra e destra, "portlet", o "viewlet", per zone o box specifici, e molti altri termini.

Per esempio, selezioniamo tre siti web dalla [lista di siti web Plone](#) per fare un confronto:

Questo è il sito web per Akamai, un fornitore leader di strumenti web online e acceleration technology. L'area di intestazione ha un semplice menu testuale per cinque aree di contenuto principali disposte orizzontalmente nella parte inferiore dell'area di intestazione. A destra, l'intestazione contiene un altro menù orizzontale e una casella di ricerca. La parte inferiore dell'area di intestazione conterrebbe dati di accesso, per l'uso da parte dei manutentori del sito. La grafica principale in alto a sinistra è una zona di messa a fuoco per la grafica accattivante e gli argomenti attuali. C'è un'area principale al centro sinistra dove si trova il testo più importante. La colonna di destra contiene una serie di "portlet". Il piè di pagina contiene un menu orizzontale, ripetendo le scelte di menu nell'intestazione, per comodità. C'è una colonna più a destra che contiene le impostazioni di zoom.

Questo è il sito web per Discover Magazine. L'area di intestazione contiene un menu orizzontale di grandi dimensioni, il "menu principale", se si vuole chiamarlo così, un menu orizzontale più piccolo in alto a destra, e una casella di ricerca. Questo sito è ricco di "portlet" testuali che coprono molte aree tematiche, divise in tre colonne, sinistra, centro, e destra. La parte superiore della colonna centrale contiene una zona focus con un video. Ci sono grandi

Header Logo, Menus, Search

Header Focus

Main Area

Footer

Right Column with "Portlets"

"Zoom" Column



Header Logo, Menus, Search

Header Focus

The header features the 'DISCOVER' logo in large, bold, black letters. To the right of the logo is a search bar with a 'SEARCH' button. Below the logo is a horizontal navigation menu with categories: Health & Medicine, Mind & Brain, Technology, Space, Human Origins, Living World, Environment, Physics & Math, Video, Photos, Podcasts, and RSS. The text 'Science, Technology, and The Future' is written in smaller font below the logo.

The 'Daily Dose' section features a large video player showing a microscopic view of a red, branching biological structure. To the left of the video are three small text snippets with dates and headlines. To the right are four smaller video thumbnails with titles and dates.

The 'Recent Features' section contains several article snippets. Each snippet includes a title, a short introductory paragraph, and a date. The titles include 'What Are the Best Scientists Working On?', 'Fetus Fight Club', 'The Rabbit's Cube Solution That Could Improve Your Life', 'How Fast is a Space Shuttle?', 'The Biggest Thing in Physics', 'What Breaks Down the Atom?', '13 Unsolved Mysteries of the Brain', and 'How to Sleep 4 Hours per Night'.

This section is divided into two columns: 'Departments' and 'News'. Each column lists several articles with their titles and authors. The 'Departments' list includes 'Vital Signs', 'Janet's World', '28 Things You Didn't Know About...', 'Peer Review', 'Shocked By Science', and 'Health Trends'. The 'News' list includes 'Really Obvious Science Headline of the Day?', 'Attack of the Roboquippers', 'No-F! Dead... But It Came Back in a Different Form!', and 'Don't Worry About CO2: Just Slow Down Population Growth'.

The 'Podcast' section features a red button labeled 'PODCAST' and a small image of a podcast cover. The 'On the Web' section has a red button labeled 'ON THE WEB' and a small image of a website. To the right is a 'Subscribe Now' section with a 'DISCOVER' logo and a 'Subscribe' button.

A grid of article snippets organized by category. The categories are: Health & Medicine, Mind & Brain, Technology, Space, Human Origins, Living World, Environment, and Physics & Math. Each category has a list of article titles and authors.

An advertisement for 'What's YOUR RealAge?'. It features a woman's face and text indicating 'Calendar age 53.5', 'RealAge 48.2', and 'Difference -7.3 years'. A red button at the bottom says 'Take the FREE RealAge Test'.

A sign-up form for 'DISCOVER NEWSLETTERS'. It includes the text 'The Leading Stories in Science and Technology, Delivered Monthly'. Below this is a list of checkboxes for various categories: Health & Medicine, Mind & Brain, Technology, Space, Human Origins, Living World, Environment, and Physics & Math. There is also a 'General Newsletter' checkbox. At the bottom is an email input field and a 'Subscribe' button.

Left Column

Right Column

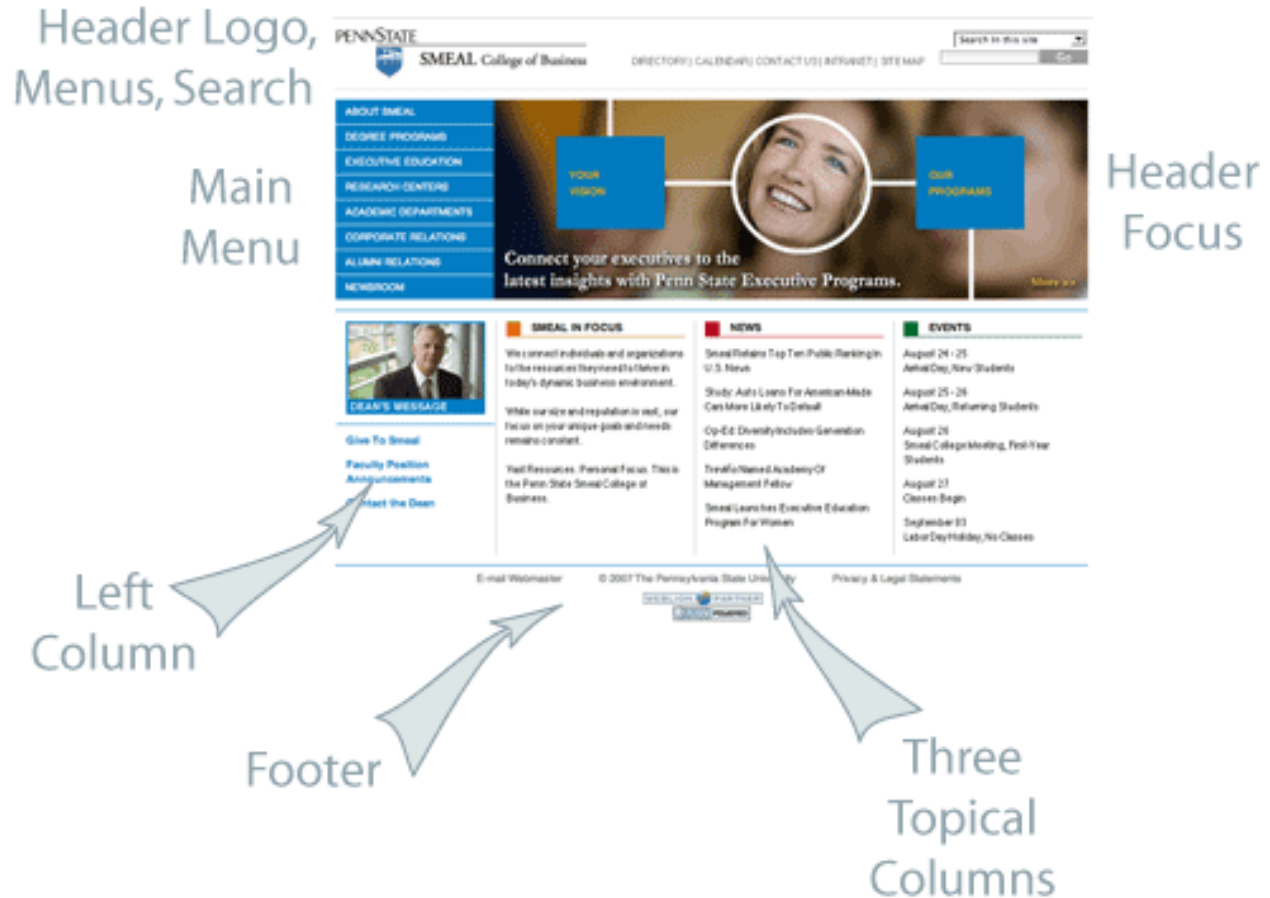
Center Area

Footer

A subscription form for 'DISCOVER' magazine. It features a large image of the magazine cover with the text 'THE YEAR IN SCIENCE'. Below the image is a form with fields for Name, Address 1, Address 2, City, State, Zip Code, and Email. A 'SUBSCRIBE' button is at the bottom right. A note at the bottom left says 'If you live outside of the US & Canada, click here'.



box interattivi in diversi punti della pagina. Il piè di pagina contiene le informazioni di identificazione di base del sito e un link al “chi siamo”. Per un grande sito web come Discover, i manutentori del sito effettuano il log-in per accedere a funzioni di editing personalizzate, e c’è molta automazione nei flussi informativi - Plone utilizza Zope, un sofisticato sistema di archiviazione, e Python, un celebre linguaggio di programmazione, che facilita un intelligente “collegamento” del flusso di testo e grafica nel sito web.



L'ultimo dei tre siti da esaminare è il sito web per lo Smeal College of Business della Penn State University. L'intestazione contiene un logo, un menu orizzontale per le aree tematiche principali, e una casella di ricerca a destra. C'è un menu principale per questo sito a sinistra, il che è più tradizionale per un sito web Plone. Una vasta area grafica contiene un'animazione “rolling focus”. C'è un altro piccolo focus grafico nella colonna di sinistra. Tre colonne testuali completano il design, al di sopra dell'identificazione di base a piè di pagina. I manutentori di questo sito accedono per mezzo di una pagina di log-in personalizzata, con il log-in e le informazioni utente che appaiono lungo la parte inferiore dell'area più in alto in testata.

Allora, come dovrebbe apparire un sito web Plone? Tradizionalmente, l'aspetto out-of-the-box è simile a quello mostrato nella parte superiore di questa pagina, con intestazione, menu, colonne, e un piè di pagina. Questi tre siti illustrano come i designer tipicamente combinano le aree di interesse, i menu verticali e orizzontali, “portlet” e contenuti testuali, di solito disposti in diverse colonne. La struttura di base della pagina è generata da Plone, Zope e Python, ma il “tema” o “skin” di design può essere fatto risultare in qualunque modo il designer preferisca.

### 1.1.3 Account e Ruoli di un utente Plone

In questo capitolo vedremo le basi di utilizzo di un account utente su un sito Plone, la distinzione fra navigazione anonima e quella autenticata e una descrizione dei ruoli degli utenti.

I siti Plone possono essere di molti tipi, dal sito personale con un solo utente ai portali di comunità ed organizzazioni con centinaia di utenti. Ogni persona che vuole aggiungere dei contenuti al sito deve avere un proprio account, definito da un nome utente e una password. Alcuni siti Plone consentono di auto-isciversi visitando il collegamento “Accedi” e compilando un form con le proprie informazioni di base. In altri siti invece gli account utente vengono creati solo dagli amministratori del sito, nel qual caso le persone normalmente ricevono un messaggio di posta elettronica con i dettagli del loro nuovo account.

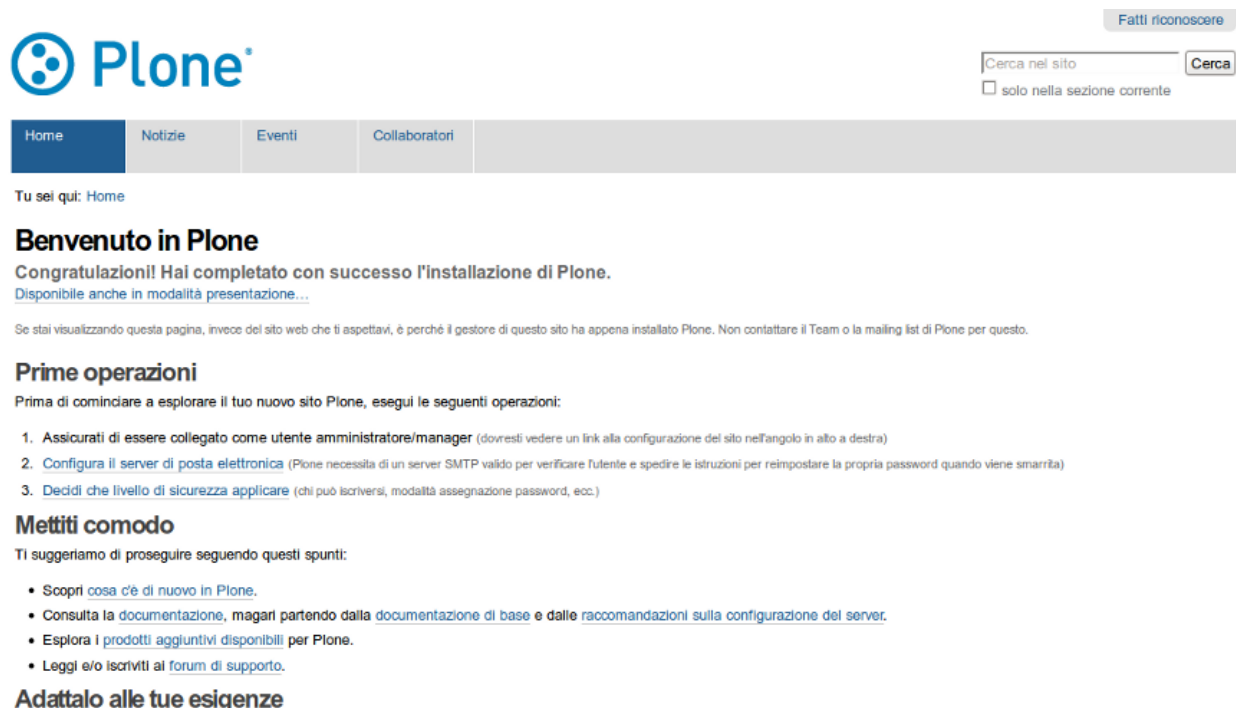
In qualsiasi modo sia stato creato, un account utente Plone permette sempre ad una persona di autenticarsi inserendo il proprio username e password. Le password sono case-sensitive, cioè la stessa lettera viene considerata diversa se scritta in maiuscolo o in minuscolo. Ad esempio, se la password è xCFGt6v, l’utente deve scrivere esattamente questa password per potersi autenticare. Le password con una buona variabilità di caratteri sono preferibili a parole troppo semplici come “cane” o “giallo”, poichè sono più difficili da indovinare e quindi sono più sicure.

### Differenze tra navigazione anonima e autenticata

La distinzione tra *navigazione anonima* e *navigazione autenticata* è molto importante:

#### Navigazione anonima

La navigazione anonima identifica la normale esperienza di un utente che naviga un sito web. Si digita l’indirizzo web di un sito nel proprio browser e si visualizza la pagina, si guardano video e immagini, ma non è necessario autenticarsi. Ecco perchè questa viene chiamata navigazione anonima: chiunque è anonimo prima dell’autenticazione. Da notare che la presenza del link *Fatti riconoscere* (n.d.t. “Log in” in Inglese) nell’angolo in alto a destra dell’immagine qui sotto. Se c’è un link “Fatti riconoscere” sulla pagina, significa che non hai effettuato l’accesso e stai visitando il sito come utente anonimo:



The screenshot shows the Plone website interface. At the top right, there is a button labeled "Fatti riconoscere". Below it is a search bar with the text "Cerca nel sito" and a "Cerca" button, with a checkbox option "solo nella sezione corrente". A navigation menu is visible with items: Home, Notizie, Eventi, Collaboratori. Below the menu, it says "Tu sei qui: Home". The main heading is "Benvenuto in Plone" followed by "Congratulazioni! Hai completato con successo l'installazione di Plone." and a link "Disponibile anche in modalità presentazione...". A message states: "Se stai visualizzando questa pagina, invece del sito web che ti aspettavi, è perché il gestore di questo sito ha appena installato Plone. Non contattare il Team o la mailing list di Plone per questo." Under "Prime operazioni", it lists three steps: 1. Assicurati di essere collegato come utente amministratore/manager, 2. Configura il server di posta elettronica, 3. Decidi che livello di sicurezza applicare. Under "Mettili comodo", it suggests following four points: 1. Scopri cosa c'è di nuovo in Plone, 2. Consulta la documentazione, magari partendo dalla documentazione di base e dalle raccomandazioni sulla configurazione del server, 3. Esplora i prodotti aggiuntivi disponibili per Plone, 4. Leggi e/o iscriviti al forum di supporto. The final section is "Adattalo alle tue esigenze".

## Navigazione autenticata

Se hai utilizzato il sito di una banca, o qualsiasi altro sito che preveda l'uso di un account, allora hai già avuto esperienza di *navigazione autenticata*. Il sito di una banca ad esempio ti permette di vedere le informazioni del tuo account, di riempire dei form, di trasferire dei fondi e altri tipi di operazioni, ma tutto questo solo dopo aver effettuato l'accesso. Un sito Plone non è molto differente, ad eccezione del fatto che si possono fare cose più complesse. Dai un'occhiata all'immagine qui sotto, catturata dopo che un utente "Mario Rossi" ha effettuato l'accesso. Vicino all'angolo in alto a destra, puoi vedere il link con il nome di Mario Rossi e un link di uscita. Un'altra differenza importante che si nota quando si è autenticati è che nell'area principale al centro c'è una barra verde con dei tab (o schede). Questa specie di striscia di testa è presente quando un utente ha i permessi per modificare l'area del sito che sta visitando. I tab nella striscia verde potrebbero variare, ma avranno sempre questo aspetto e questo caratteristico colore. Nella seguente immagine, l'utente Mario Rossi si è autenticato in un nuovo sito Plone:

The screenshot shows the Plone user interface. At the top right, the user name "Mario Rossi" is displayed with a dropdown arrow. Below it is a search bar with the text "Cerca nel sito" and a "Cerca" button. A checkbox option "solo nella sezione corrente" is visible below the search bar. A navigation menu contains "Home", "Notizie", "Eventi", and "Collaboratori". Below the menu, it says "Tu sei qui: Home". A green bar contains tabs for "Contenuti", "Visualizza", "Modifica", and "Condivisione", along with "Vista", "Aggiungi...", and "Stato: Pubblicato". The main content area displays "Benvenuto in Plone" with a creation date of 14/11/2012 23:04. A congratulatory message states: "Congratulazioni! Hai completato con successo l'installazione di Plone." Below this, there are sections for "Prime operazioni" and "Mettilti comodo" with a list of suggestions.

## Ruoli utente

In un sito Plone è molto importante la distinzione dei diversi ruoli degli utenti. Per illustrare il caso più semplice, consideriamo due ruoli utente: *collaboratore* e *manager*. Vediamo i diversi permessi o "poteri" di questi due ruoli:

### Collaboratore

- ha un account utente, quindi può autenticarsi;
- può aggiungere contenuti, ma solo in aree specifiche, e non può modificare niente al di fuori di queste aree; spesso agli utenti viene assegnata un'area "home" da utilizzare come uno spazio personale dove possono aggiungere contenuti;

- non può pubblicare un contenuto per renderlo visibile nella navigazione anonima, nemmeno nel caso dei contenuti che ha creato direttamente; un utente con il ruolo di manager dovrà approvare il contenuto per la pubblicazione.

### Manager

- ha un account utente, quindi può autenticarsi;
- può aggiungere contenuti ovunque e ha il potere di modificare qualunque cosa;
- può pubblicare qualsiasi contenuto.

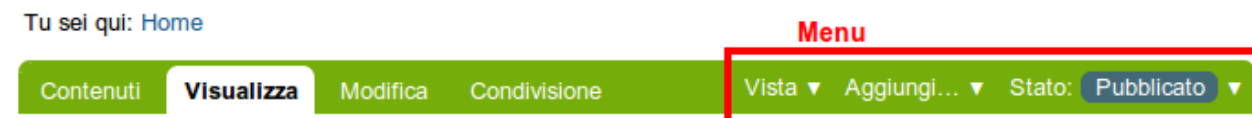
Quando ottieni il tuo nuovo account su un sito Plone, ti dovrebbero fornire tutte le informazioni che indicano dove hai il diritto di aggiungere contenuti. Dopo aver effettuato l'accesso, se vai in una cartella in cui hai i permessi adeguati, vedrai la striscia di intestazione con il tipico colore verde e le schede *Contenuti*, *Visualizza*, *Modifica*, *Regole*, e *Condivisione*:



Potrai navigare per scoprire di persona le differenze tra questi tab, ma ecco qualche indicazione per aiutarti a cominciare:

- *Contenuti* - mostra la lista dei contenuti in una cartella;
- *Visualizza* - mostra come un utente anonimo vede il contenuto corrente;
- *Modifica* - mostra un pannello per modificare il contenuto;
- *Condivisione* - mostra un pannello per assegnare ad altri utenti i permessi per vedere e modificare il contenuto.

Puoi inoltre vedere i menu nella parte finale della barra verde, *Vista*, *Aggiungi...* e *Stato*:



Esplora anche questi menu. Ecco qualche indicazione per partire:

- *Vista* - mostra il menu per scegliere il tipo di visualizzazione (vista tabellare, vista riassuntiva, etc.);
- *Aggiungi...* - mostra il menu per aggiungere nuovi contenuti (immagini, pagine, cartelle, etc...);
- *Stato* - mostra il menu per modificare lo stato di pubblicazione (privato, bozza, pubblicato, etc..).

Questi menu e tab sono il modo principale per interagire con Plone. Ti saranno molto familiari quando imparerai di più su come gestire un sito Plone.

### 1.1.4 Autenticazione

#### Cosa aspettarsi quando ci si autentica in un sito Plone

Quando visiti un sito Plone come anonimo, oppure ti viene dato un indirizzo web per manutenzione del sito, potrai vedere un bottone “Fatti riconoscere” in alto a destra come questo:

### Fatti riconoscere

Una volta cliccato il link *Fatti riconoscere*, vedrai un pannello di autenticazione dove inserire il tuo nome utente e la tua password:

Dopo l'autenticazione ad un sito web Plone, potrai vedere il tuo nome, solitamente in alto nell'angolo a destra del tuo schermo. Puoi cliccare sul tuo nome per effettuare alcune azioni relative al tuo utente, come spiegato nelle sezioni successive.

Da Plone 4 in poi, tu (o l'amministratore del sito) puoi permettere agli utenti di utilizzare il loro indirizzo di posta elettronica come nome utente per effettuare l'autenticazione. Questa funzionalità può essere attivata nelle impostazioni di sicurezza nel pannello di controllo. L'effetto è tale per cui:

- nel modulo di registrazione non viene richiesto uno specifico nome utente
- nel modulo di autenticazione viene chiesto all'utente di inserire l'email.

Vedi [E-mail address based login in the Upgrade Guide](#) per maggiori informazioni su questa funzionalità.

## 1.1.5 Impostare il tuo profilo

**Una volta autenticato in un sito web Plone, puoi cambiare il tuo profilo personale indicando informazioni circa la tua identità e scegliere le impostazioni del sito web.**

Il tuo nome completo viene mostrato nell'angolo in alto a destra dello schermo. Clicca sul tuo nome per aprire il menù a discesa, quindi clicca il link *Dashboard* per entrare nella tua area personale:

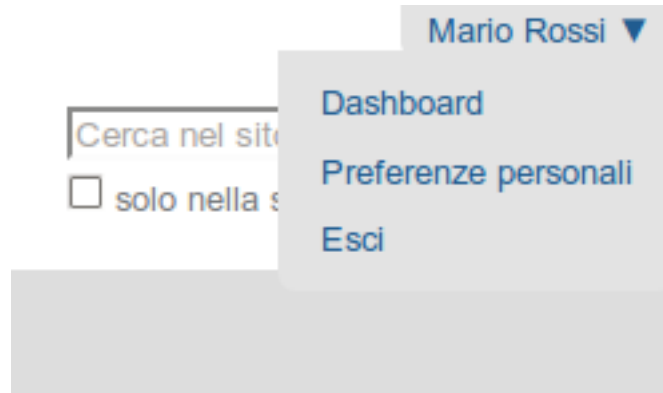
Vedrai la **dashboard** (o scrivania personale):

La prima volta che ti autentichi, la dashboard sarà vuota come indica il messaggio di Info. Le **portlet** sono specifiche "viste" di vari tipi di contenuto. Puoi scegliere quali vedere nella tua dashboard cliccando sul tab *modifica*, ma ci arriveremo in un secondo.

Prima di tutto, diamo un'occhiata al link *Preferenze personali*, nel menù di cui parlavamo prima, che ti porterà alla modifica del tuo profilo:

I campi disponibili sono:

- *Nome e cognome* - Indica il tuo nome completo.



**Bacheca** Modifica

Informazioni La tua bacheca è vuota. Seleziona la scheda di *modifica* per personalizzarla aggiungendovi qualche portlet.

## Dashboard di Mario Rossi

Informazioni personali Preferenze personali Password

### Informazioni personali

Modifica le tue informazioni personali

#### Nome e cognome

Inserisci il tuo nome completo, ad esempio Mario Rossi.

Mario Rossi

#### Indirizzo posta elettronica

mario.rossi@plone.it

#### Pagina personale

L'indirizzo della tua pagina personale esterna, se ne hai una.

#### Biografia

Una breve panoramica su chi sei e cosa fai. Verrà visualizzata nella tua pagina autore, collegata agli elementi creati da te.

#### Località:

La località in cui vivi (paese e città) oppure, in ambito aziendale, il luogo in cui si trova il tuo ufficio.

#### Ritratto



- *Indirizzo e-mail* - OBBLIGATORIO - Puoi ricevere email dall'amministratore del sito web, o da un forum, se installato, ecc. Quando un campo è obbligatorio, un piccolo quadrato rosso è presente a fianco del nome del campo.
- *Località* - Questo è il nome della tua città, stato, provincia o qualsiasi altra informazione vorrai fornire.
- Selezione della *lingua* - Plone eccelle nell'offrire un supporto multilingua.
- *Biografia* - Inserisci una breve descrizione di te stesso in questo campo, un paragrafo o poco più.
- *Pagina personale* - Se hai un tuo web site personale o, per esempio, un'area dove condividi foto, se vuoi puoi inserire qui l'indirizzo web. In questo modo altre persone potranno trovare più informazioni su di te.
- *Editor* - Puoi scegliere di utilizzare TinyMCE o Kupu, che ti permettono di modificare le pagine web con una interfaccia grafica oppure una normale area di testo, che è adatta se sei abituato a scrivere pagine web in HTML (il "codice" base delle pagine web). L'impostazione di default per i siti appena creati è di utilizzare TinyMCE, e, in questo manuale, si assume che sia questa l'impostazione.
- *Abilita la modifica con l'editor esterno* - Questa impostazione abilita e disabilita l'uso di un editor "esterno", se questo è stato impostato dall'amministratore del sito web. L'uso di un editor esterno è principalmente inteso per web designer e programmatori che modificano il codice sorgente, ma può essere utile per la creazione di pagine quando si usa un linguaggio di markup specializzato. (Non ti preoccupare di questa impostazione se il tuo amministratore non te ne ha parlato esplicitamente).
- *Ritratto* - Il tuo ritratto apparirà come una piccola immagine, quindi è consigliata una foto del viso o del busto.

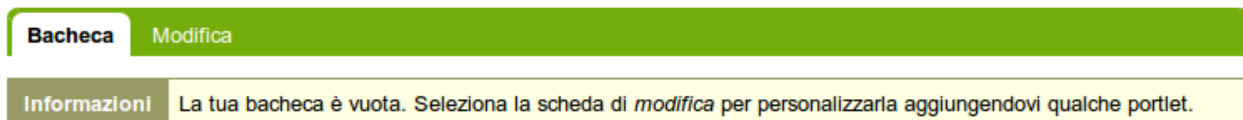
Puoi cambiare le tue preferenze ogni volta che vuoi.

## 1.1.6 La tua Dashboard

**Ogni utente Plone ha una sua "dashboard" da personalizzare.**

Plone ha diversi "viste" predefinite per le notizie, gli eventi, i documenti modificati recentemente, ecc. Queste liste sono raggruppate in aree rettangolari chiamate portlet. Pensa ad una portlet come ad una finestra su un dato tipo di contenuti. Per esempio, la portlet "notizie" offre una vista delle notizie pubblicate recentemente.

Tu controlli quali portlet vedi nella tua dashboard e dove sono disposte. Il seguente screenshot mostra cosa vedrebbe l'utente Mario Rossi una volta che si fosse autenticato e che avesse cliccato sul suo nome, posto in alto a destra, per andare alla sua area personale:



## Dashboard di Mario Rossi

La dashboard appare vuota per un nuovo utente.

Un click sul tab di modifica per la dashboard mostrerà che ci sono portlet già assegnate alla dashboard – la dashboard mostrata sopra è vuota perchè non ci sono contenuti disponibili da presentare nelle portlet di questo nuovo sito web. Ecco le portlet di default:

Vedi le portlet *Notizie* ed *Eventi* nella colonna più a sinistra, i *Contenuti recenti* nella seconda colonna e l'*Elenco di revisione* nella colonna di destra. La terza colonna non ha portlet assegnate.

L'account di un nuovo utente, in un sito web Plone base, avrà una dashboard come quella mostrata, ma per un sito web che è stato personalizzato con funzionalità aggiuntive potrebbero esserci più portlet tra cui scegliere e la dashboard

potrebbe partire con diverse portlet già posizionate nelle colonne. Per esempio, potrebbero esserci portlet per “il meteo”, “quotazioni di borsa”, “frase del giorno”, ecc. a seconda di cosa è stato installato sul sito web (queste opzioni richiederebbero software personalizzato). L’utente può personalizzare le portlet che vuole vedere e la loro posizione tra le quattro colonne.

Quindi, per l’account Plone tipico, la dashboard parte con le portlet mostrate sopra che verrebbero popolate con le news, gli eventi e gli altri contenuti creati nel sito web.

## 1.2 Aggiungere contenuti

Come aggiungere contenuti base ai siti web Plone

### 1.2.1 Aggiungere nuovi contenuti

Una panoramica generale su come aggiungere nuovi contenuti in Plone e definizione dei tipi di contenuto standard.

Per aggiungere nuovi contenuti si utilizza il menu a discesa **Aggiungi...**:

In Plone, i contenuti vengono aggiunti *localmente* quindi devi navigare fino alla sezione dove desideri aggiungere il contenuto prima di usare la voce del menu **Aggiungi...** E’ possibile ovviamente tagliare, copiare e incollare contenuti da una sezione ad un’altra se necessario.

#### Tipi di contenuti

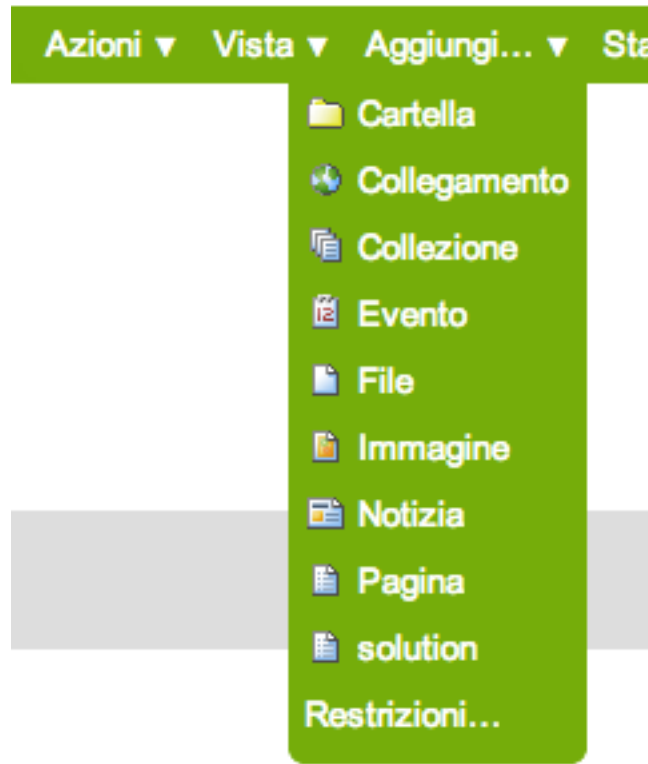
In Plone, hai a disposizione un certo numero di **Tipi di contenuto** che corrispondono ai diversi tipi di contenuto che puoi pubblicare. Ad esempio, per caricare un’immagine devi utilizzare il tipo di contenuto **Immagine**. Di seguito la lista dei tipi di contenuti disponibili nell’ordine in cui appaiono ed una breve descrizione:

**Collezione** Le Collezioni sono utilizzate per raggruppare e visualizzare contenuti in base a dei **criteri** configurabili. Il funzionamento delle Collezioni è molto simile a quello delle query in un normale database.

**Evento** Un Evento è un tipo di pagina speciale specifico per la pubblicazione di un evento (ad esempio una raccolta fondi, un barbecue, etc). Questo tipo di contenuto ha una funzione che permette ai visitatori del sito di aggiungere l’evento al proprio calendario personale utilizzando gli standard iCal e vCal. Questi standard sono compatibili con: Google Calendar, Outlook, Sunbird e altri. Per aggiungere un singolo evento al tuo calendario personale, fai click sui link vCal o iCal accanto al testo “Aggiungi l’evento al calendario” nella pagina principale dell’Evento.

A partire da Plone 3.3 è anche possibile scaricare tutti gli Eventi di una cartella in una sola volta (al momento è disponibile solo in formato iCal). Per scaricare il file iCal, appendi `@@ics_view` alla fine dell’URL della cartella che contiene gli eventi. Ad esempio, se si desidera ottenere tutti gli eventi della cartella *eventi* posizionata nella





## European Plone Symposium 2010

<b>When</b>	<u>May 26, 2010 11:00 AM to</u> <u>May 30, 2010 07:00 PM</u>
<b>Where</b>	Sorrento, Italy
<b>Attendees</b>	Alexander Limi, Martin Aspeli
<b>Add event to calendar</b>	 <a href="#">vCal</a>  <a href="#">iCal</a>

radice del tuo sito, vai all' indirizzo `http://tuodominio.tld/events/@@ics_view`. In un futuro rilascio di Plone, è in programma l'inserimento di questo indirizzo direttamente nell'interfaccia utente.

**File** Un File in Plone è un file binario caricabile sul sito con l'intento di farlo scaricare dai visitatori. Gli esempi più comuni di file sono PDF, Documenti di testo e fogli di calcolo.

**Cartella** Le Cartelle in Plone funzionano come le cartelle del tuo computer. Puoi utilizzare le cartelle per organizzare i contenuti e per dare al tuo sito Plone una struttura di navigazione.

**Immagine** Il tipo di contenuto Immagine è utilizzato per caricare file di immagini (JPG, GIF, PNG) in modo tale che tu possa inserirli all'interno di pagine o di contenuti simili.

**Collegamento** Indicato anche come 'Oggetto Link', non è da confondere con i collegamenti che vengono creati tramite TinyMCE o Kupu, gli editor visuali per le pagine di Plone. Il tipo di contenuto Collegamento è spesso usato per includere un collegamento ad un sito web esterno nell'albero di navigazione o per altri usi specifici.

**Notizia** Questo tipo di contenuto è molto simile agli Eventi, anche se una Notizia si utilizza appositamente per la pubblicazione di notizie. È inoltre possibile allegare un'immagine ad una Notizia, la miniatura apparirà nella vista riassuntiva della cartella accanto alla descrizione della stessa.

**Pagina** Una Pagina in Plone è uno dei tipi di contenuto più semplici disponibili. Utilizza questo oggetto per scrivere la maggior parte delle pagine web del tuo sito Plone.

Nota: a seconda di quali prodotti aggiuntivi hai installato, potresti vedere più opzioni sotto la voce **Aggiungi...** del tuo menu. Per informazioni su questi tipi di contenuto, fai riferimento alla documentazione dei vari prodotti installati.

### Titolo

Quasi tutti i tipi di contenuto in Plone hanno due campi in comune: **Titolo** e **Descrizione**.

Il campo **Titolo** delle cartelle, delle immagini, delle pagine, etc., può contenere tutto quello che vuoi – puoi usare qualsiasi carattere della tastiera, inclusi gli spazi. I **Titoli** vanno a comporre l'indirizzo web dei contenuti creati. Gli indirizzi web, noti come URLs, sono quello che digiti in un browser per passare in una specifica posizione di un sito (o il percorso del link selezionato), come ad esempio:

`www.mysite.com/about/personnel/sally/bio`

o

`www.mysite.com/images/butterflies/skippers/long-tailed-skippers`

Gli indirizzi web, al contrario dei titoli, *sono* soggetti a restrizioni. Alcuni caratteri della tastiera non sono consentiti come, ad esempio, gli spazi. Plone fa un buon lavoro nel mantenere gli indirizzi web molto simili ai **Titoli** forniti, convertendoli in lettere minuscole, sostituendo gli spazi con i trattini e sostituendo altri segni di punteggiatura.

In Plone l'indirizzo web di un certo elemento è denominato **nome breve**. Quando si utilizza la funzione **Rinomina**, verrà visualizzato sia il nome breve sia il titolo.

I campi variano a seconda del tipo di contenuto. Per esempio, il tipo di contenuto Collegamento ha il campo URL. Il tipo di contenuto File ha il campo File e così via.

### Descrizione

La **Descrizione** appare nella parte superiore delle pagine, appena sotto il titolo. Sono spesso visualizzate in molte viste assegnate a Cartelle e Collezioni (come in quella Standard e in quella Sintetica). La descrizione appare anche nei risultati delle ricerche eseguite con il motore di ricerca nativo di Plone.

## 1.2.2 Aggiungere una Cartella

Aggiungere cartelle ad un sito web Plone è il passo fondamentale per controllare l'organizzazione dei contenuti

**Traduzione** Giampiero Lago (27/11/2012)

**Impaginazione** Giacomo Spettoli (27/11/2012)

**Revisione** Giacomo Spettoli (19/05/2013)

I pc utilizzano una struttura gerarchica per organizzare i programmi e i files all'interno del disco rigido. In passato avrai sicuramente creato delle cartelle (o directory) sul tuo computer per organizzare i tuoi documenti. In Plone le cartelle sono utilizzate praticamente nello stesso modo, l'unica differenza è che sono create in un sito web al fine di dare una struttura al contenuto.

Le cartelle si aggiungono cliccando sul menu *Aggiungi...* e selezionando **Cartella**:

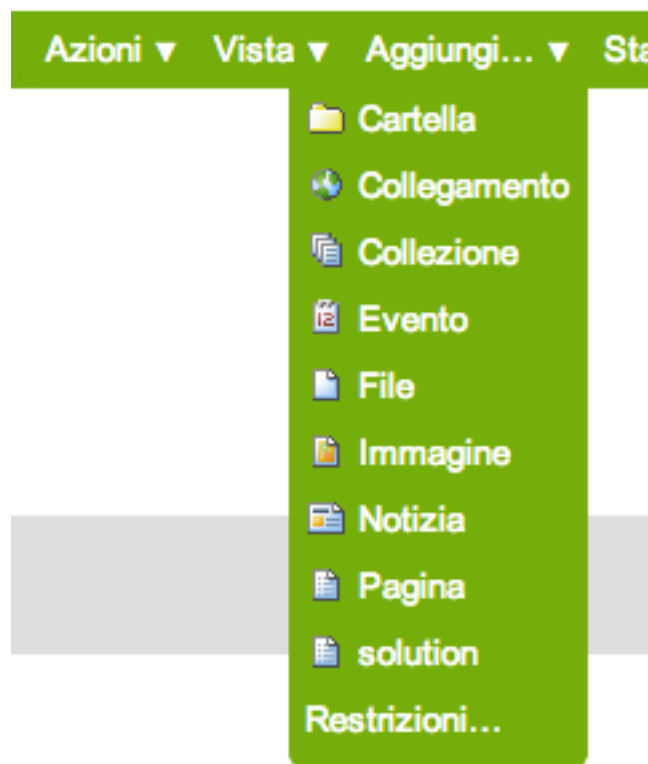


Fig. 1.1: add-item-menu-folder.png

Ora dovresti vedere il pannello *Aggiungi Cartella*:

E' necessario compilare il campo Titolo, perchè si tratta di un campo obbligatorio (come è indicato dal quadratino rosso). Il campo Descrizione è invece opzionale; potrai sempre tornare indietro al pannello di modifica se hai necessità di aggiungere una descrizione alla cartella. Le descrizioni sono utili quando un visitatore utilizza la ricerca di Plone – nei risultati saranno visualizzati sia il Titolo sia la Descrizione del contenuto.

Potrete notare altri tab nella parte superiore:

- **Default**, per inserire i campi Titolo e Descrizione,

## Add Folder

A folder which can contain other items.

The screenshot shows the 'Add Folder' form in Plone. At the top, there are five tabs: 'Default', 'Categorization', 'Dates', 'Ownership', and 'Settings'. The 'Default' tab is active. Below the tabs, there is a 'Title' field with a red asterisk indicating it is required. The text 'Butterflies' is entered in this field. Below the title field is a 'Description' field with the text 'Used in item listings and search results.' Below the description field are two buttons: 'Save' and 'Cancel'.

- **Categorizzazione**, per specificare le categorie a cui appartiene la cartella (conosciute anche come keywords o tag),
- **Date**, per settare il periodo di tempo durante il quale la cartella sarà visibile nel sito,
- **Proprietario**, per specificare l'autore e/o i collaboratori dell'elemento in questione,
- **Impostazioni**, per abilitare i commenti, abilitare la navigazione precedente/successivo e scegliere se visualizzare il contenuto nel menu di navigazione del sito web.

Queste schede sono standard e si trovano anche su altri tipi di contenuto. Vedremo più nel dettaglio queste schede più avanti in questo manuale.

Assicurati di cliccare sul bottone **Salva** in basso alla pagina quando hai finito di inserire le informazioni. Questo completerà il processo di creazione di una cartella

Guarda un video su [come aggiungere una cartella](#).

### 1.2.3 Cosa c'è in un nome Web?

**Ogni contenuto di un sito Plone ha un indirizzo web univoco. Plone crea gli indirizzi automaticamente, in base al Titolo che avete fornito.**

Il **Titolo** di un contenuto, incluse cartelle, immagini, pagine, etc. può essere tutto ciò che vuoi – puoi usare tutti i caratteri della tastiera, inclusi gli spazi bianchi. I titoli diventano parte dell'indirizzo web per ogni elemento che crei in Plone. Gli indirizzi Web, conosciuti anche come URL, sono quello che scrivete in un browser web per andare ad una posizione specifica in un sito web (In alternativa, come ad esempio:

www.mysite.com/about/personnel/sally/bio

o

[www.mysite.com/images/butterflies/skippers/long-tailed-skippers](http://www.mysite.com/images/butterflies/skippers/long-tailed-skippers)

Al contrario dei titoli, gli indirizzi web *hanno* restrizioni sui caratteri consentiti, come gli spazi bianchi. Plone fa un ottimo lavoro per mantenere gli indirizzi web corretti utilizzando una struttura quasi equivalente al titolo che avete fornito, convertendo tutte le lettere in minuscolo e sostituendo i trattini agli spazi bianchi e alla punteggiatura.

Per capire meglio, prendiamo ciascuno di questi due indirizzi web e dividiamoli nei vari componenti:

```
www.mysite.com/about/personnel/sally/bio
^
website name
      ^
      a folder named About
            ^
            a folder named Personnel
                  ^
                  a folder named Sally
                          ^
                          a folder named Bio
```

In questo caso Plone ha cambiato ogni titolo della cartella in lettere minuscole, ad esempio da Personnel a personnel. Ma non dovete preoccuparvi di questo perchè Plone gestisce l'indirizzamento web; vi basterà digitare nei titoli quello che volete.

E, per il secondo esempio:

```
www.mysite.com/images/butterflies/skippers/long-tailed-skippers
^
website name
      ^
      a folder named Images
            ^
            a folder named Butterflies
                  ^
                  a folder named Skippers
                          ^
                          a folder named Long-Tailed Skippers
```

Questo esempio è simile al primo ed illustra come avviene la conversione in lettere minuscole del titolo di ciascuna cartella alla corrispondente parte dell'indirizzo web. Da notare il caso della cartella nominata "Long-tailed Skippers": Plone conserva il trattino, in quanto carattere consentito, sia nel titolo che nella parte dell'indirizzo web, ma converte in un trattino, nell'indirizzo web, lo spazio bianco tra le parole Tailed e Skippers oltre che le lettere da maiuscole a minuscole.

L'indirizzo web di un certo contenuto è indicato in Plone come nome breve. Quando usate la funzione Rinomina verrà visualizzato il nome breve insieme al titolo.

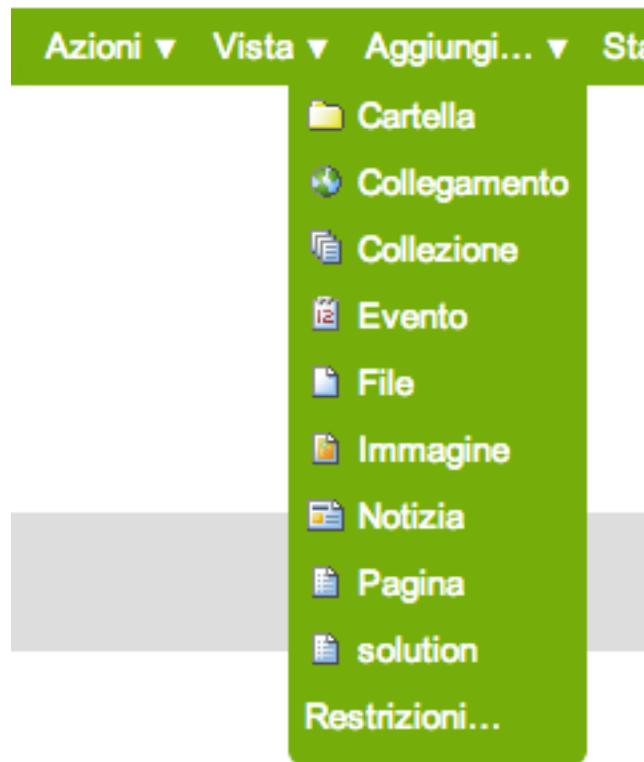
## 1.2.4 Aggiungere un'Immagine

**Aggiungere immagini in un sito web Plone è un'attività di base che può comportare un po' di lavoro sul tuo computer locale, ma è essenziale perché fotografie, mappe e grafica personalizzata sono molto importanti nei siti web.**

[Preparare un'immagine per il web](#)

**Note:** Ricorda di utilizzare per tutte le immagini i formati di file comunemente accettati come standard sul web. I formati accettabili includono: JPG, JPEG, GIF e PNG. Non usare i formati BMP e TIFF, poiché non sono supportati da tutti i browsers.

Quando vuoi caricare una immagine, utilizza il menu *Aggiungi...* (vedrai il menu *Aggiungi...* solo dopo aver effettuato l'accesso):



Dopo aver cliccato per aggiungere una **Immagine**, vedrai il pannello *Aggiungi Immagine*:

Ci sono i campi Titolo e Descrizione (campi intesi come “campi di immissione dati”), così come visto nel caso della creazione di una cartella. In fondo al pannello c'è un campo per caricare un'immagine. Analizziamo i tre campi di immissione dei dati:

- *Titolo* - Inserisci il testo che preferisci, compresi spazi bianchi e punteggiatura (Plone gestisce automaticamente l'adattamento del titolo per generare l'URL dell'immagine)
- *Descrizione* - È sempre una buona idea valorizzare questo campo, anche se non è obbligatorio
- *Immagine* - Il campo Immagine è una casella di testo seguita da un bottone *Sfoglia...*. Nella casella di testo, tuttavia, non devi scrivere niente: clicca sul bottone *Sfoglia* e potrai scegliere il file da caricare selezionandolo direttamente dalle cartelle presenti sul tuo computer (per fare questo, è utile tenere bene a mente in quale cartella del proprio computer è stata salvata l'immagine da caricare).

Poichè Titolo e Descrizione non sono campi obbligatori, per caricare un'immagine sul tuo sito Plone tutto quello che serve è selezionare l'immagine stessa sul tuo computer tramite il bottone *Sfoglia* e cliccare sul bottone *Salva* presente nella parte inferiore del pannello. Dovrai aspettare qualche secondo per il completamento del caricamento (qualche minuto, se hai una connessione internet lenta). Un'anteprima dell'immagine verrà visualizzata al termine del caricamento.

A partire da Plone 4, le immagini e i file che vengono caricati sul sito hanno un loro ID (URL), basato sul Titolo inserito tramite il campo visto in precedenza. Se il campo Titolo non viene valorizzato (non è obbligatorio), l'URL assegnato all'immagine (o al file) sarà generato utilizzando il nome del file.

# Add Image

An image, which can be referenced in documents or displayed in an album.

**Default**

**Categorization**

**Dates**

**Ownership**

**Settings**

**Title**

**Description**

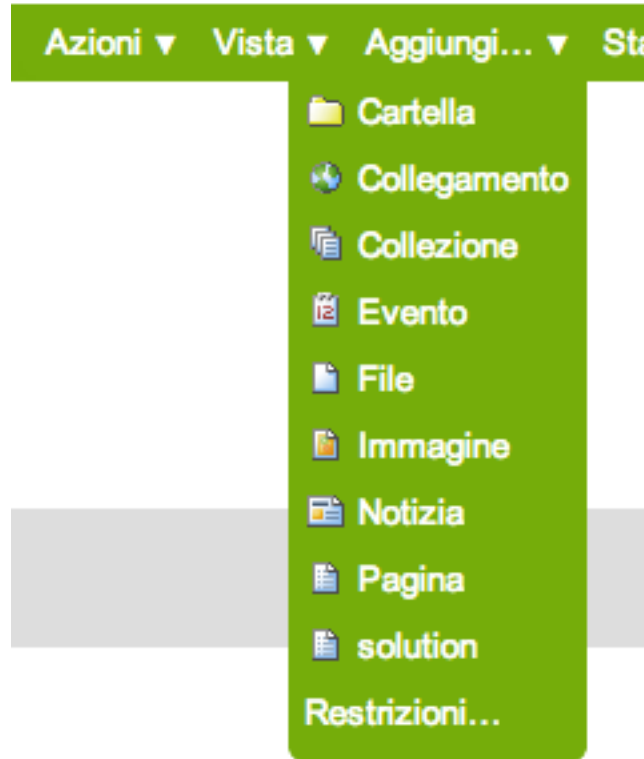
A short summary of the content.

**Image** ■

## 1.2.5 Aggiungere una Pagina

Le pagine in Plone possono variare molto, ma ad ogni modo sono sempre “pagine web”.

Per aggiungere una pagina, utilizza il menu *Aggiungi...* presente in Plone a livello di cartella:



Seleziona la voce **Pagina** dal menu a discesa e vedrai il pannello *Aggiungi pagina*:

I campi **Titolo** e **Descrizione** sono i primi in alto, riempili in maniera appropriata. C'è un campo *Commento alle modifiche* in fondo, un normale campo di testo utile per memorizzare eventuali annotazioni che descrivano le modifiche fatte al documento. Ciò è utile per le pagine sulle quali potresti dover collaborare con altri.

Al centro del pannello c'è il campo **Testo del documento**. Il software utilizzato per la composizione delle pagine è comunemente detto *editor di testo visuale* e nello specifico in Plone si utilizza TinyMCE. Un editor di tipo testuale ti permette di comporre le pagine in maniera WYSIWYG. WYSIWYG (*What You See Is What You Get*, quello che vedi è quello che avrai) è un termine che descrive il modo in cui l'editor funziona: se, ad esempio, applichi il grassetto ad una parola, vedrai immediatamente il risultato del nuovo stile applicato.

Le persone normalmente si trovano subito a proprio agio con l'approccio utilizzato dagli editor WYSIWYG. Vedremo in maniera più approfondita questo argomento più avanti in questo manuale.

### Linguaggi di markup

Se preferite scrivere il testo delle pagine utilizzando i formati di markup, è possibile disabilitare l'editor di testo visuale nel pannello delle preferenze personali, e rimpiazzare così TinyMCE con un campo di testo semplificato. I formati di markup disponibili in Plone sono:

- [Markdown](#)



# Add Page

A page in the site. Can contain rich text.

**Default** Categorization Dates Ownership Settings

## Title

**Writing for the web**

## Summary

Used in item listings and search results.

**Things to remember when writing content for the web**

## Body Text

Style... **B** *I*          

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam et elementum leo. In vel risus enim, eget eleifend erat. Fusce metus ipsum, pretium eget pretium in, tincidunt quis quam. Pellentesque ac tellus at orci elementum adipiscing. Morbi nulla lorem, ultricies quis dictum vel, commodo in libero. Mauris lacinia magna eu ligula gravida congue. Donec pulvinar, lorem quis semper congue, diam dolor rutrum turpis, ac placerat ipsum arcu non orci. Pellentesque sit amet ipsum et nibh semper vehicula. Vivamus mauris libero, vulputate sed malesuada a, venenatis nec turpis. Cras augue nulla, volutpat non vulputate quis, dignissim at magna. In hac habitasse platea dictumst. Pellentesque at libero nisi, in fringilla mi. Maecenas eget ipsum magna, sit

## Change note

Enter a comment that describes the changes you made.

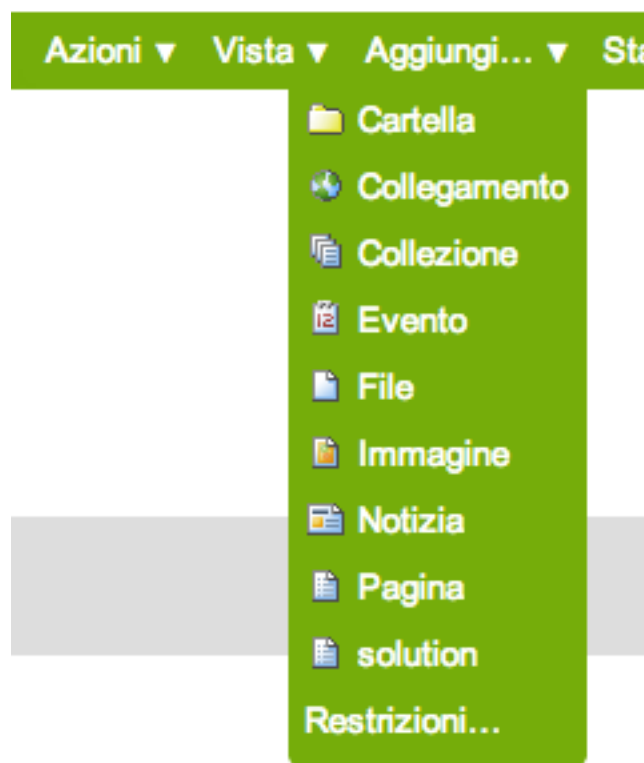
- Textile
- Structured Text
- Restructured Text

Ognuno di questi formati si basa sull'utilizzo di speciali codici di formattazione all'interno del testo. Ad esempio, nel formato di markup Structured Text, mettere tra doppi asterischi una parola o una frase renderà la parola o la frase in grassetto, così **Questo testo sarebbe in grassetto**. Può essere utile imparare ad utilizzare questi formati di markup per aumentare la velocità di input (soprattutto se si creano molte pagine) o se si preferisce un approccio leggermente più tecnico nell'inserimento del testo. Alcune persone preferiscono questi formati non solo per la velocità in sé ma anche per la fluidità di espressione.

### 1.2.6 Aggiungere un File

**File di vari tipi possono essere caricati su un sito Plone.**

Per aggiungere un file, utilizza il menu *Aggiungi...* presente a livello di cartella:



Seleziona la voce **File** dal menu a discesa e vedrai il pannello *Aggiungi file*:

Fai click sul pulsante *Scegli file* per cercare nelle cartelle del tuo computer locale il file da caricare. Inserisci un titolo (puoi utilizzare lo stesso nome del file se vuoi). Inserisci anche una *descrizione*, se vuoi. Quando fai click sul bottone *Salva* il file verrà caricato nella cartella Plone.

Puoi caricare file PDF, documenti Word, file di database, archivi zip...- praticamente qualunque cosa. All'interno di un sito Plone, i file vengono trattati semplicemente come file e verranno mostrati nelle liste di contenuti delle cartelle Plone all'interno delle quali sono caricati, ma non ci sarà nessuna visualizzazione speciale per loro. Appariranno con il loro nome nelle liste e sarà possibile scaricarli cliccando sul link costituito dal loro nome nella lista.

## Add File

An external file uploaded to the site.

Default Categorization Dates Ownership Settings

Title

Description

Used in item listings and search results.

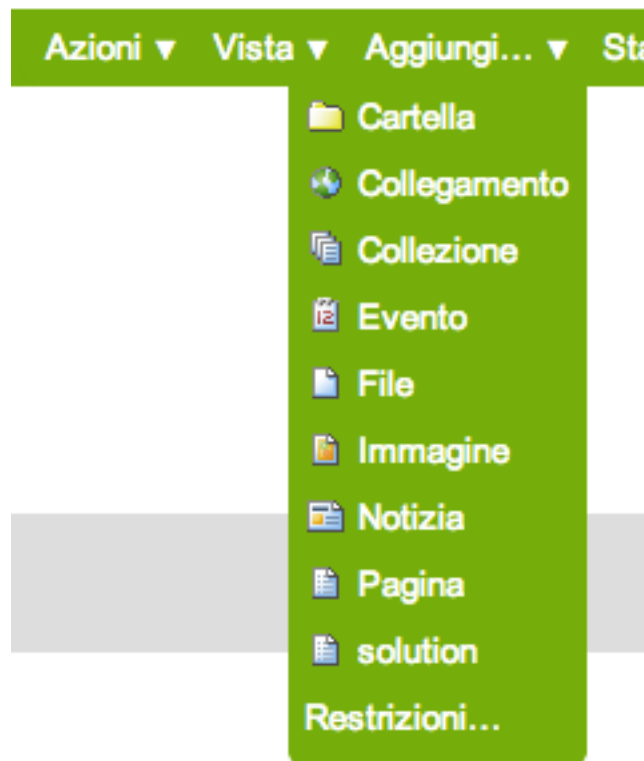
File ■

Esistono add-on specializzati per Plone, che permettono la ricerca di contenuti all'interno dei file. Chiedi all'amministratore del tuo sito Plone se hai bisogno di queste funzionalità.

### 1.2.7 Aggiungere un Collegamento

Oltre ad inserire collegamenti nel testo delle pagine, è possibile in Plone creare collegamenti anche come tipi di contenuto autonomi. Avere collegamenti come tipi di contenuto ti permette ad esempio di organizzarli in cartelle, di impostare delle parole chiave ad essi associate per facilitarne il raggruppamento negli elenchi e nei risultati di ricerca, o di inserirli nei menù di navigazione.

Per aggiungere un oggetto di tipo Collegamento seleziona la voce corrispondente dal menu *Aggiungi...* presente a livello di cartella Plone:



Avrai accesso al pannello *Aggiungi Collegamento*:

Scegliere dei buoni titoli è importante, perchè sono proprio i titoli ad essere visualizzati nella lista di tutti i collegamenti presenti all'interno di una cartella Plone. Immagina cosa significhi questo se il numero di collegamenti nella cartella tende a crescere...

Incolla l'indirizzo web nel campo URL oppure digitalo. Poichè non c'è alcuna funzionalità di anteprima dell'URL inserito, è meglio copiare quest'ultimo direttamente dalla finestra del browser nella quale lo si sta vedendo, in modo da essere sicuri della sua correttezza.

#### Comportamento dell'oggetto di tipo Collegamento

Un oggetto di tipo Collegamento si comporterà nei seguenti modi, a seconda delle autorizzazioni di cui si dispone.

## Add Link

A link to an internal or external resource.

Default Categorization Dates Ownership Settings

Title ■

Description

Used in item listings and search results.

URL ■

Change note

Enter a comment that describes the changes you made.

Save

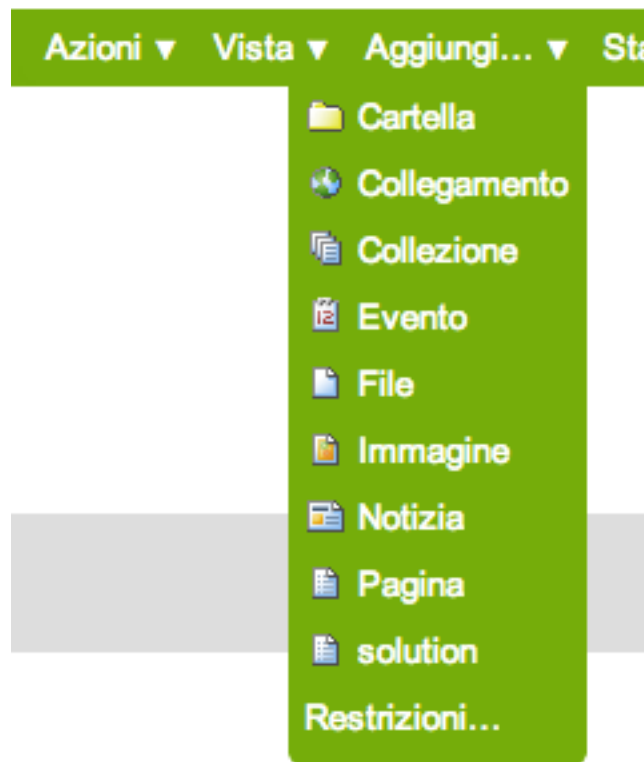
Cancel

- **Se hai il permesso di modificare l'oggetto Collegamento**, quando clicchi sull'oggetto vieni rimandato al pannello di editazione del contenuto stesso, per poterlo modificare (se così non fosse, verresti indirizzato all'URL associato all'oggetto, e non avresti modo di modificarlo)
- **Se non hai il permesso di modificare l'oggetto Collegamento**, quando clicchi sull'oggetto vieni indirizzato direttamente all'URL associato all'oggetto. Il comportamento in questo caso è lo stesso che si avrebbe inserendo direttamente l'indirizzo nel browser. L'oggetto collegamento in questo caso si comporta come un *redirect*

### 1.2.8 Aggiungere un Evento

I siti Plone hanno un sistema integrato per la gestione e la visualizzazione di eventi in un calendario.

In una cartella, utilizza la voce del menu *Aggiungi...* per aggiungere un evento:



Comparirà un pannello abbastanza grande *Aggiungi Evento*:

# Add Event

Information about an upcoming event, which can be displayed in the calendar.

Default   Categorization   Dates   Ownership   Settings

Title ■

Description

Used in item listings and search results.

Event Location

Event Starts ■

2010 / March / 22  02 : 10 PM

Event Ends ■

2010 / March / 22  02 : 10 PM

Event body text

Style... **B** *I*          

Attendees

Dall'alto, si hanno i seguenti campi:

- *Titolo* - **OBBLIGATORIO**
- *Descrizione*
- *Luogo dell'evento*
- *Inizio dell'evento* - **OBBLIGATORIO**
- *Termine dell'evento* - **OBBLIGATORIO**
- *Testo dell'evento* (editor visuale)
- *Partecipanti*
- *Tipo(i) di evento*
- *URL dell'evento*
- *Nome del contatto*
- *Indirizzo e-mail per contatti*
- *Telefono del contatto*
- *Commento alle modifiche*

Nota che solo tre campi sono obbligatori: titolo, inizio e termine dell'evento. Anche se si tratta di una pannello con molte informazioni da inserire, se hai fretta, ti basta inserire questi tre campi e salvare per creare l'evento. Ovviamente, se hai altre informazioni, puoi inserirle.

Una parte del pannello richiede qualche informazione aggiuntiva: l'inizio e il termine dell'evento. L'anno, il mese, il giorno ed altri campi sono semplicemente menu a discesa. Spesso però non è semplice ricordare esattamente il giorno da inserire e si ha la necessità di consultare un calendario. C'è un comodo calendario pop-up che offre un modo alternativo per selezionare il giorno. Se clicchi una volta sull'icona del calendario accanto al selettore a discesa del giorno, :

The image shows a user interface for selecting event dates. It consists of two rows of date pickers. The first row is labeled 'Event Starts' and the second row is labeled 'Event Ends'. Each row contains a series of dropdown menus: year (2007), month (July), day (5), a small calendar icon, hour (10), and minute (--). A dashed rectangular box highlights the calendar icons in both rows, indicating that clicking them will open a calendar pop-up.

comparirà questo calendario pop-up:

E' sufficiente cliccare sul giorno di interesse e questo verrà automaticamente impostato nel pannello. Compila i campi per i quali hai informazioni e salva, ma ricorda :

**IMPORTANTE:** L'evento non verrà visualizzato nel calendario principale del sito fino a quando non sarà **pubblicato**.

### 1.2.9 Aggiungere una Notizia

**Plone integra un sistema nativo per la pubblicazione di notizie.**

Per aggiungere una nuova notizia, utilizza la voce corrispondente del menù *Aggiungi* presente a livello di cartella Plone:

Avrai accesso al pannello *Aggiungi notizia*:



**Event Starts** ■

2010 / March / 22 02 : 10 PM

**Event Ends** ■

2010 / March / PM

March, 2010							
Today							
Mon	Tue	Wed	Thu	Fri	Sat	Sun	
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14
11	15	16	17	18	19	20	21
12	22	23	24	25	26	27	28
13	29	30	31				

Select date

Azioni ▼ Vista ▼ Aggiungi... ▼ Sta

- Cartella
- Collegamento
- Collezione
- Evento
- File
- Immagine
- Notizia
- Pagina
- solution
- Restrizioni...

## Add News Item

An announcement that will show up on the news portlet and in the news listing.

Default   Categorization   Dates   Ownership   Settings

Title ■

Summary

Used in item listings and search results.

Body Text

Style... **B** *I* [List icons] [Link icon] [Image icon] [Table icon] [HTML icon]

Image

Will be shown in the news listing, and in the news item itself. Image will be scaled to a sensible size.

Image Caption

Change note

38 Enter a comment that describes the changes you made.

Nel pannello, ci sono i campi standard Titolo, Descrizione e Commento alle modifiche, insieme ad un editor visuale per inserire il corpo della notizia (Testo del documento) ed ai campi per l'upload dell'immagine e per la sua didascalia. Nel area Testo del documento puoi inserire qualsiasi tipo di testo, con la formattazione di cui hai bisogno, e tramite la funzionalità Inserisci/modifica immagine dell'editor, puoi aggiungere al testo della notizia tutte le immagini che desideri. Le immagini caricate verranno aggiunte alla cartella nella quale stai creando la notizia.

I campi *Immagine* e *Didascalia immagine* servono ad aggiungere un'immagine che verrà utilizzata come elemento grafico rappresentativo della notizia stessa all'interno degli elenchi di notizie pubblicate sul sito Plone. L'immagine verrà automaticamente ridimensionata e posizionata in ciascun elenco. Se devi inserire un'immagine nel corpo del testo della notizia, pertanto, non devi utilizzare il campo Immagine, ma la funzionalità dell'editor visuale presente per il campo Testo del documento.

**IMPORTANTE:** le notizie inserite non appariranno nell'elenco principale o nella portlet utilizzata per pubblicare le notizie sul tuo sito Plone finchè non saranno nello stato **“Pubblicato”**.

## 1.2.10 Impostazione delle proprietà di base

**I tab disponibili per ogni tipo di contenuto Plone dispongono di campi per l'immissione delle informazioni di base. Fornire tali dati è importante, significa fornire combustibile per il motore di Plone.**

Ogni tipo di contenuto, se editato da un utente con diritti di modifica su di esso, mostrerà una serie di tab nella parte superiore, per l'impostazione delle proprietà di base:



Questi tab per le proprietà di base sono:

- *Default* - mostra il form di inserimento dei dati principali per il contenuto
- *Categorizzazione* - mostra un pannello per la creazione e l'impostazione delle categorie (parole chiave) per il contenuto
- *Date* - mostra la data di pubblicazione e la data di scadenza per il contenuto
- *Possessore* - mostra un pannello per l'impostazione dei creatori del contenuto e di tutti coloro che vi hanno contribuito, nonché di tutte le informazioni sul copyright
- *Impostazioni* - mostra un piccolo pannello per stabilire se l'elemento apparirà nel menu di navigazione e se sono ammessi i commenti sul contenuto

I campi di inserimento in queste schede coprono le informazioni descrittive di base chiamate **\*metadati\***. I metadati vengono a volte chiamati “dati sui dati”. Plone può utilizzare questi metadati in moltissimi modi.

Ecco il pannello di *Categorizzazione*, mostrato per un tipo di contenuto “Pagina” (sarebbe lo stesso per altri tipi di contenuto):

*Nota: In Plone 3, i tag erano chiamati categorie. Nelle versioni precedenti la 3.0, essi erano invece chiamati Parole Chiave*

Il campo principale di inserimento del pannello serve a specificare le *categorie* associate al contenuto che si sta editando. Per crearne di nuove, basta semplicemente digitare parole o frasi, una per riga, nel box **Nuovi tag**. Quando si salva il contenuto, i nuovi tag saranno creati all'interno dell'elenco di tag del sito web, e il contenuto stesso sarà archiviato sotto di essi. Se si ri-modifica questo contenuto, o si modifica qualsiasi altro contenuto, i tag creati saranno automaticamente disponibili come **Tag esistenti**.

Default Categorization Dates Ownership Settings

### Tags

Tags are commonly used for ad-hoc organization of content.

#### New tags

#### Related Items

Add...

#### Location

The geographical location associated with the item, if applicable.

#### Language

English

#### Change note

Enter a comment that describes the changes you made.

Save Cancel

Il campo *Elementi Correlati* permette di impostare collegamenti tra i vari contenuti. Quando un contenuto viene visualizzato, i contenuti correlati vengono mostrati come link a fondo pagina. Ciò è utile quando non si desidera utilizzare le categorie esplicite (tag) per la correlazione di contenuti diversi.

Il campo *Posizione* fa riferimento ad una posizione geografica associabile al contenuto. E' adatto per l'uso con sistemi di mappatura, ma utilizzabile per l'archiviazione del contenuto in generale.

La *Lingua* scelta normalmente è quella di default del sito, ma su siti web multilingue, lingue diverse potrebbero essere utilizzate in un mix di contenuti.

Il pannello *Date* presenta campi per impostare la data di pubblicazione e quella di scadenza del contenuto. Se impostate, esse definiscono in concreto le date di inizio e fine della validità del contenuto:

**Default** **Categorization** **Dates** **Ownership** **Settings**

**Publishing Date**  
The date when the item will be published. If no date is selected the item will be published immediately.

-- / -- / -- -- : -- --

**Expiration Date**  
The date when the item expires. This will automatically make the item invisible for others at the given date. If no date is chosen, it will never expire.

-- / -- / -- -- : -- --

**Change note**  
Enter a comment that describes the changes you made.

Le date di pubblicazione e di scadenza funzionano in questo modo:

- Se visualizzato, ogni contenuto che ha una data di scadenza già trascorsa viene contrassegnato come “scaduto” in rosso nel suo sottotitolo.
- Un oggetto la cui data di pubblicazione è posteriore alla data attuale non presenta testo aggiuntivo nel suo sottotitolo.
- In entrambi i casi, l'elemento è “non pubblicato”, definizione che non deve essere confusa con uno stato del suo workflow.
- Vuol dire semplicemente che l'elemento non compare negli elenchi e nelle ricerche.
- Questi elenchi includono gli elenchi di contenuti presenti in una cartella.
- Tuttavia, il proprietario dell'elemento continuerà a vederlo, questo perchè è desiderabile sapere quali documenti giacenti ci sono nel nostro sito.
- Il permesso che controlla tutto questo si chiama “Access inactive portal content”.
- Gli elementi scaduti in una cartella sono contrassegnati come tali durante la visualizzazione `folder_contents`.

- Non c'è un modo rapido di vedere se gli elementi in un elenco di cartelle sono non ancora pubblicati.
- Quando si imposta un elemento non pubblicato come visualizzazione predefinita per una cartella, tale elemento verrà visualizzato.
- L'annullamento della pubblicazione di un elemento non ha alcun effetto per gli amministratori. Essi potranno sempre vedere gli oggetti non pubblicati nei loro elenchi e nelle ricerche.
- Anche se si assegnano permessi sul contenuto ad utenti non amministratori ("può aggiungere", "può modificare", "può revisionare"), per questi utenti il contenuto resterà sempre "non pubblicato".
- Un modo pratico per un utente non amministratore per accedere a un elemento non pubblicato è direttamente attraverso il suo URL.

Il pannello *Possessore* dispone di tre campi liberi per assegnare i creatori del contenuto, coloro che vi hanno contribuito, e le informazioni in merito ai diritti d'autore e di proprietà:

Il pannello *Impostazioni* ha campi che possono variare un po' da un tipo di contenuto all'altro, ma in generale ci sono campi di input per stabilire se l'elemento debba apparire o meno nella navigazione, o se sono autorizzati i commenti, e altri controlli simili:

### Raccomandazioni

Non vi è alcun obbligo di inserire le informazioni specificate attraverso questi pannelli, ma farlo è una buona idea. Per il pannello *Possessore*, fornire i dati è importante per situazioni dove ci sono diverse persone coinvolte nella creazione di contenuti, soprattutto se ci sono più creatori e collaboratori che lavorano in gruppo. Non sempre è necessario compilare campi quali la data di pubblicazione e di scadenza, lingua e diritti d'autore, ma questi dati devono essere specificati al momento opportuno. Un sistema di gestione dei contenuti è tanto buono quanta completezza nella gestione dei dati permette.

Specificare le categorie richiede attenzione, ma se si prende l'abitudine, e se ci si impegna a creare un insieme significativo di categorie, vi è un grande ritorno dallo sforzo fatto. Tale ritorno si concretizza nella maggiore efficacia delle funzionalità di ricerca e di altre funzionalità Plone che si basano sulla categorizzazione. Lo stesso vale per l'impostazione degli elementi correlati. Sarai in grado di trovare rapidamente quello che ti serve, e potresti diventare abile nello scoprire e sfruttare le relazioni fra i contenuti.

### Esposizione delle proprietà dei metadati come meta tag nel codice HTML

Da Plone 4 in poi, in *Configurazioni del sito, Sito*, c'è una check box che permette di esporre le proprietà di base dei metadati Dublin Core. Selezionando questa casella verranno aggiunti il titolo, la descrizione, ecc... e altri metadati come meta tag all'interno dell'HTML <head>. Per esempio:

```
<meta content="short description" name="DC.description" />
<meta content="short description" name="description" />
<meta content="text/html" name="DC.format" />
<meta content="Page" name="DC.type" />
<meta content="admin" name="DC.creator" />
<meta content="2009-11-27 17:04:03" name="DC.date.modified" />
<meta content="2009-11-27 17:04:02" name="DC.date.created" />
<meta content="en" name="DC.language" />
```

Le proprietà Dublin Core *Creator*, *Contributors* e *Publisher* saranno visualizzate solamente se è stata abilitata la visualizzazione di queste informazioni per agli utenti anonimi. La configurazione si trova in *Configurazioni del sito*, al link *Sicurezza*

Per saperne di più su [Dublin Core](#) e [HTML Metatags](#).

Default Categorization Dates **Ownership** Settings

#### Creators

Persons responsible for creating the content of this item. Please enter a list of user names, one per line. The principal creator should come first.

jsmith

#### Contributors

The names of people that have contributed to this item. Each contributor should be on a separate line.

#### Rights

Copyright statement or other rights information on this item.

#### Change note

Enter a comment that describes the changes you made.

Save

Cancel

Default   Categorization   Dates   Ownership   **Settings**

Allow comments  
If selected, users can add comments to this item.

Exclude from navigation  
If selected, this item will not appear in the navigation tree

Presentation mode  
If selected, this will give users the ability to view the contents as presentation slides.

Table of contents  
If selected, this will show a table of contents at the top of the page.

Change note  
Enter a comment that describes the changes you made.

### 1.2.11 Restrizioni sui tipi di contenuto in una cartella

**Il menu “Aggiungi nuovo” ha la possibilità di limitare i tipi di contenuto che possono essere aggiunti alla cartella.**

Limitare i tipi di contenuti che possono essere aggiunti ad una cartella è il modo più semplice per controllare la creazione di contenuti in un sito web Plone. Puoi scegliere di utilizzare restrizioni sui tipi di contenuti se il tuo sito viene gestito da numerose persone. In questo modo puoi forzare buone pratiche, come ad esempio inserire solo immagini in una cartella cui hai dato nome “cartella immagini”.

Prima di tutto, seleziona l’ultima opzione nel menu “aggiungi” chiamata “Restrizioni...”

Ci sono tre scelte possibili per aggiungere restrizioni ai tipi di contenuto creabili in una cartella:

La scelta di default è di utilizzare le impostazioni della cartella-padre. Avere questa impostazione come default significa che se crei una cartella e crei restrizioni sui tipi che possono essere aggiunti ad essa, ogni sottocartella creata erediterà automaticamente tali restrizioni.

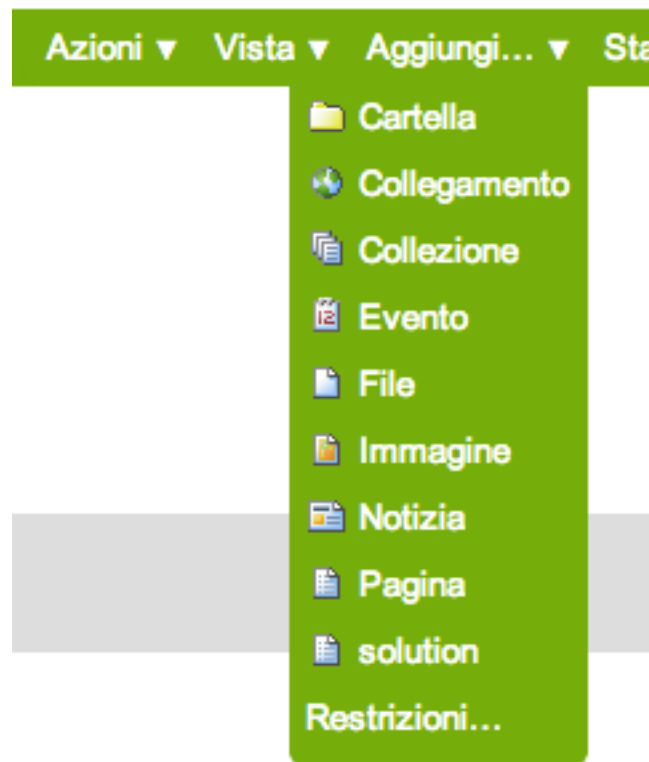
La seconda scelta, permettere la sola aggiunta di tipi di contenuti standard, è il modo di ritornare alle impostazioni iniziali, senza restrizioni.

L’ultima scelta permette di selezionare da una lista di tipi di contenuti disponibili:

I tipi di contenuti elencati sotto la voce *Tipi consentiti* sono quelli disponibili all’interno del sito web. Il default, come mostrato, è di permettere l’aggiunta di tutti i tipi di contenuti. A partire dalle impostazioni di default, i vari tipi di contenuti disponibili possono essere abilitati o disabilitati, per permettere di aggiungerli o meno alla cartella.

L’uso di *tipi supplementari* permette un controllo ancora più di dettaglio. Per esempio, se è deciso di salvare tutte le immagini in una sola cartella, invece di spargerle in varie cartelle sul sito web – uno schema che in molti





## Restrict what types of content can be added

Type restrictions

Select the restriction policy in this location.

- Allow the standard types to be added
- Specify types manually

## Restrict what types of content can be added

### Type restrictions

Select the restriction policy in this location.

- Allow the standard types to be added  
 Specify types manually

### Allowed types

Controls what types are addable in this location.

- |  |  |   |
|--|--|---|
| <input checked="" type="checkbox"/> Collection | <input checked="" type="checkbox"/> Folder | <input checked="" type="checkbox"/> News Item |
| <input checked="" type="checkbox"/> Event      | <input checked="" type="checkbox"/> Image  | <input checked="" type="checkbox"/> Page      |
| <input checked="" type="checkbox"/> File       | <input checked="" type="checkbox"/> Link   |   |

### Secondary types

Select which types should be available in the 'More...' submenu *instead* of in the main pulldown. This is useful to indicate that these are not the preferred types in this location, but are allowed if you really need them.

- |                                     |                                 |                                    |
|-------------------------------------|---------------------------------|------------------------------------|
| <input type="checkbox"/> Collection | <input type="checkbox"/> Folder | <input type="checkbox"/> News Item |
| <input type="checkbox"/> Event      | <input type="checkbox"/> Image  | <input type="checkbox"/> Page      |
| <input type="checkbox"/> File       | <input type="checkbox"/> Link   |                                    |

Save

Cancel

preferiscono – una cartella “immagini” può essere creata, impostando le immagini come unici tipi di contenuti creabili al suo interno.

Allo stesso modo una cartella “Eventi aziendali” potrebbe essere creata per contenere solo contenuti di tipo Evento. Se si impostassero le cose in questo modo, i creatori di contenuti sarebbero forzati a seguire questo schema restrittivo.

Tuttavia, un po’ più di flessibilità è spesso desiderabile per le immagini. Selezionando il contenuto “Immagine” nella voce *tipi supplementari* per la cartella “Eventi aziendali”, le immagini possono essere aggiunte, se è effettivamente necessario, utilizzando il sotto menù “Altri...” che appare quando viene attivato questo meccanismo.

Alcune persone preferiscono un mix eterogeneo di contenuti sul sito web, senza restrizioni. Altri preferiscono un approccio più regolamentato, restringendo tipi secondo un dato schema organizzativo. Plone ha la flessibilità necessaria per accettare un ampio spettro di impostazioni.

## 1.2.12 Preparare le immagini per il web

**Preparare le immagini per il web è una parte essenziale per utilizzare le immagini in Plone, come in qualsiasi contesto online. Come vedrai, le dimensioni contano.**

Molte fotografie utilizzate dagli utenti sono scattate con una fotocamera digitale, ma possono anche essere immagini acquisite da scanner, illustrazioni grafiche realizzate con software specifici e altri tipi di immagini particolari. Prendiamo il caso di una foto di una farfalla scattata con una fotocamera digitale.

Le fotografie digitali scattate con macchine fotografiche moderne sono di solito troppo grandi per essere inserite direttamente su un sito web, quindi hanno bisogno di essere ridimensionate. Un tipico design di un sito web potrebbe avere una larghezza di circa 1000 pixel. Quando una foto viene scattata con una moderna macchina fotografica, può avere diverse migliaia di pixel di larghezza e altezza, e quindi risultare di diversi megabyte di dimensione come file. Dovrai quindi utilizzare software appositi che ridimensionino l’immagine in qualcosa di meno di 1000 x 1000 pixel, molto spesso serviranno anche dimensioni più piccole.

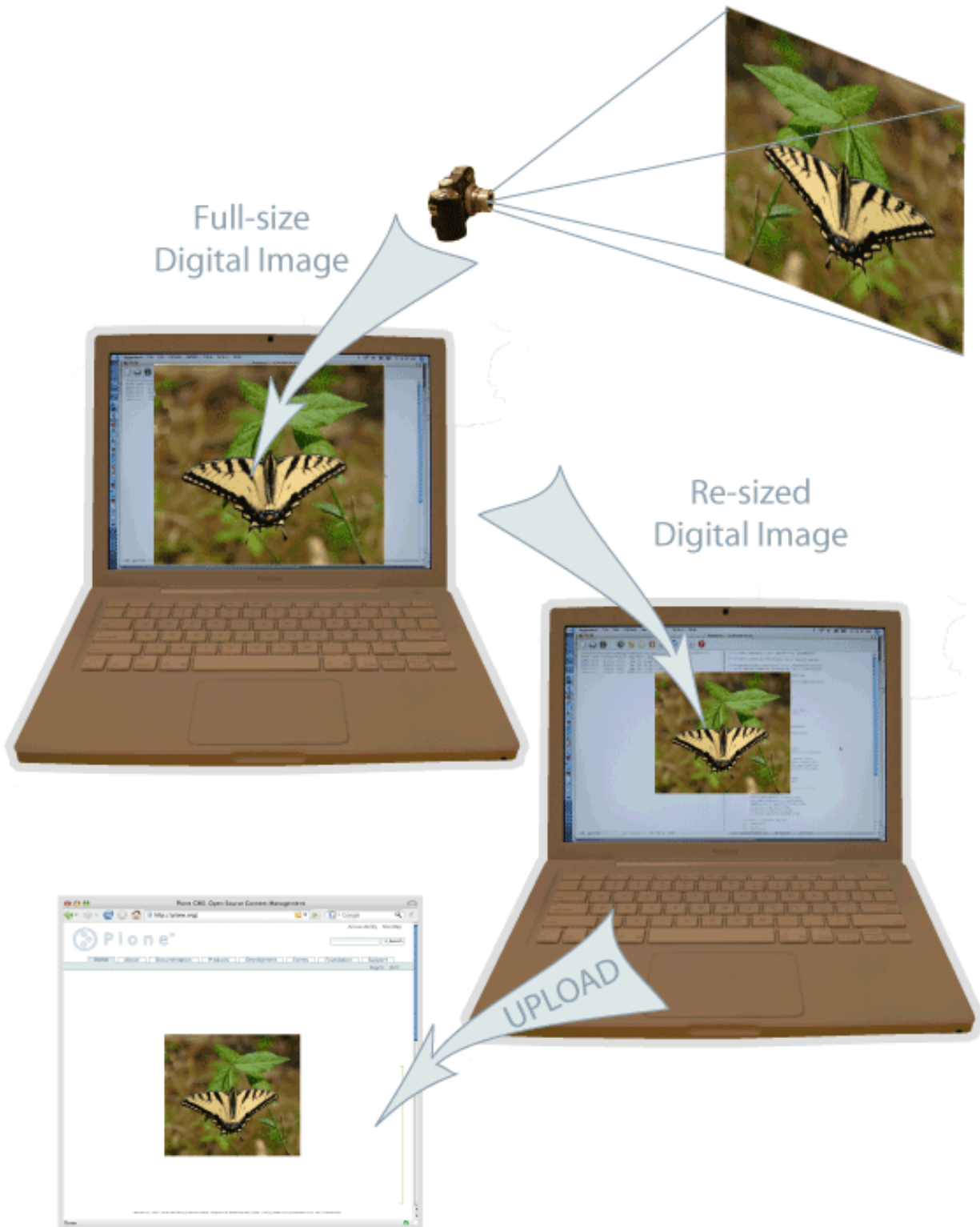
I software che si utilizzano per visualizzare o stampare le foto digitali hanno spesso questa funzionalità di ridimensionamento, in alternativa si potrebbero utilizzare software di grafica come Corel Draw, Adobe Photoshop, Irfanview, o Gimp. Il ridimensionamento di un’immagine, a volte chiamato ricampionamento, è una funzione standard che spesso si trova nei software di fotoritocco sotto la voce di menu Immagine.

Come facciamo a sapere di che dimensione di larghezza in pixel abbiamo bisogno per ridimensionare la nostra immagine? Dipende. Per un “head shot”, una fotografia da inserire in una biografia, forse 200 pixel di larghezza potrebbero bastare. Per una foto di gruppo, 200 pixel risulterebbero troppo poco per consentire l’identificazione delle persone nella fotografia, quindi magari si potrebbe aver bisogno di una larghezza di almeno 400 pixel. Per una immagine di una mappa presa da scanner, forse la larghezza dovrebbe essere di 1000 pixel per permettere di visualizzare i dettagli della mappa.

Dopo aver salvato l’immagine ridimensionata, diamogli un nome che indichi il nuovo formato (ad esempio, butterflyfly-resized-300px.jpg). Il formato del file è di solito jpg (o. jpeg). Altri formati comuni per le immagini sono png e gif. Prendi nota dove salvi le immagini sul tuo computer, in modo da trovarle facilmente quando le carichi sul tuo sito web Plone.

**Per riassumere:**

1. Scatta la fotografia con la tua fotocamera, o trova un’immagine esistente che desideri utilizzare
2. Trasferiscila sul tuo computer
3. Utilizza software di fotoritocco sul tuo computer per ridimensionare la fotografia o l’immagine
4. Carica la fotografia o l’immagine sul tuo sito Plone



### 1.2.13 Aggiungere collezioni

Le collezioni (precedentemente chiamate Smart Folders) sono contenitori virtuali di liste di contenuti trovati attraverso specifiche ricerche.

Consulta a questo proposito la sezione del manuale Utilizzo delle collezioni.

## 1.3 Gestione dei contenuti

La scheda contenuti è il posto dove gli oggetti posso essere copiati, tagliati, incollati, spostati, rinominati, etc.

### 1.3.1 Tagliare, Copiare e Incollare contenuti

Le operazioni taglia, copia e incolla comportano lo spostamento di uno o più contenuti da una cartella ad un'altra.

#### Taglia/Incolla

Spostare contenuti da una area ad un'altra in un sito web è un operazione comune. Spesso si ha necessità di questa operazione quando alcuni contenuti sono posizionati nella cartella sbagliata. Ad esempio, se il creatore della cartella sulle farfalle Skippers mostrata nella figura che segue (che contiene a sua volta le cartelle relative alla singole farfalle) si rende conto che la cartella Eastern Tiger Swallowtail evidenziata in figura è stata erroneamente creata nella cartella padre Skippers, può semplicemente spostarla con una operazione di *taglia/incolla*:

Nota che la cartella Eastern Tiger Swallowtail è stata selezionata, e che il pulsante *Taglia* sta per essere cliccato. Dopo aver fatto clic sul pulsante *Taglia*, lo schermo mostrerà un nuovo pulsante *Incolla*. La cartella Eastern Tiger Swallowtail e tutto il suo contenuto sono ora nella "memoria" del sito web. La cartella Eastern Tiger Swallowtail non scompare subito in quanto è in attesa della relativa operazione *Incolla*. Il pulsante *Incolla* viene ora evidenziato per mostrare che l'operazione *taglia/incolla* è in corso:

Il pulsante *Incolla* ora è attivo. Il passo successivo è quello di selezionare la cartella di destinazione, in questo caso la cartella Swallowtails:

Dopo aver cliccato ed essere entrati nella cartella Swallowtails, il pulsante *Incolla* continuerà a vedersi, perché l'operazione Incolla non è ancora stata completata:

Per ultimo, facendo clic sul pulsante *Incolla* all'interno della cartella di destinazione Swallowtails, la cartella Eastern Tiger Swallowtail viene infine aggiunta nel giusto posto, e viene quindi tagliata dalla posizione originale, la cartella Skippers. L'operazione di *Copia/Incolla* è ora completata:

Il pulsante *Incolla* rimane attivo, perché è possibile continuare ad incollare la cartella in altri posti, se si vuole. Ciò potrebbe accadere in diverse situazioni, quando magari è necessario copiare una pagina, ad esempio una sorta di modello o documento standard, in diverse cartelle.

#### Copia/Incolla

L'operazione di *Copia/Incolla* è identica all'operazione di *Taglia/Incolla*, tranne che non c'è rimozione del contenuto dalla cartella originale. Esso funziona come ci si aspetta che funzioni.

Contents View Edit Rules Sharing Add new... ▼

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

[Up one level](#)

Select: [All](#)

	Title	Size	Modified	State
 <input type="checkbox"/>	<a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
 <input type="checkbox"/>	<a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	<a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	<a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input checked="" type="checkbox"/>	<a href="#">Eastern Tiger Swallowtail</a>	1 kB	May 07, 2010 05:19 AM	Private
 <input type="checkbox"/>	<a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	<a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	<a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private

Copy Cut Rename Delete Change State

Contents **View** Edit Rules Sharing Add new... ▼

Info 1 item(s) cut.

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)  
[Up one level](#)

Select: All

	Title	Size	Modified	State
☰ <input type="checkbox"/>	<a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
☰ <input type="checkbox"/>	<a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
☰ <input type="checkbox"/>	<a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
☰ <input type="checkbox"/>	<a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰ <input type="checkbox"/>	<a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰ <input type="checkbox"/>	<a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰ <input type="checkbox"/>	<a href="#">Eastern Tiger Swallowtail</a>	1 kB	May 07, 2010 05:19 AM	Private
☰ <input type="checkbox"/>	<a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private

Copy Cut Rename Paste Delete Change State

nts Users Butterflies Swallowtails

Contents View Edit Rules Sharing Add new... ▼

Info 1 item(s) cut.

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

[Up one level](#)

Select: All

	Title	Size	Modified	State
 <input type="checkbox"/>	<a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
 <input type="checkbox"/>	<a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	<a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	<a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	<a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	<a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private

Home News Events Users Butterflies Swallowtails

Contents View Edit Rules Sharing Add new... ▼

## Swallowtails

by admin — last modified May 07, 2010 05:32 AM — [History](#)

[Up one level](#)

This folder has no visible items. To add content, press the add button, or paste content from another location.

Paste

 Eastern Tiger Swallowtail

[Manage portlets](#)

Contents View Edit Rules Sharing Add new... ▼

Info Item(s) pasted.

## Swallowtails

by admin — last modified May 07, 2010 05:32 AM — [History](#)

[Up one level](#)

Select: All

	Title	Size	Modified	State
 <input type="checkbox"/>	<a href="#">Eastern Tiger Swallowtail</a>	1 kB	May 07, 2010 05:19 AM	Private

Copy Cut Rename Paste Delete Change State



### 1.3.2 Modificare i contenuti

**La modifica dei contenuti in Plone funziona allo stesso modo dell'aggiunta - solitamente i pannelli per l'immissione dei dati e per la modifica dei contenuti sono gli stessi.**

Naturalmente, quando si modifica un elemento, l'oggetto esiste già. Fare clic sulla scheda Modifica di un contenuto per vedere il pannello di inserimento dati per quel contenuto, insieme con i valori già esistenti per quell'elemento.

Per un esempio molto semplice di come modificare un contenuto sia molto simile ad aggiungerlo, possiamo rivedere come si modifica una cartella.

Il pannello *Modifica* di una cartella mostra semplicemente le aree di input per il titolo e la descrizione. Spesso la descrizione non è prevista per una cartella, quindi l'unica cosa da cambiare è il titolo. Se si desidera dare una descrizione, che è una buona idea per distinguere le cartelle in un elenco, la descrizione può essere inserita solo in formato testo – non c'è alcuna possibilità di impostare lo stile di testo, come grassetto, corsivo, o altre formattazioni. Ciò mantiene le descrizioni degli elementi Plone il più semplice possibile.

Ecco il pannello *Modifica* di una cartella, in questo caso, una cartella chiamata “Butterflies”:

The screenshot shows the 'Edit Folder' interface in Plone. At the top, there is a green navigation bar with tabs for 'Contents', 'View', 'Edit', 'Rules', and 'Sharing'. Below this, the main heading is 'Edit Folder'. Underneath, there is a grey navigation bar with tabs for 'Default', 'Categorization', 'Dates', 'Ownership', and 'Settings'. The 'Title' field is highlighted in yellow and contains the text 'Butterflies'. Below the title field, there is a 'Description' field with the placeholder text 'Used in item listings and search results.' At the bottom of the form, there are two buttons: 'Save' and 'Cancel'.

Tutto qui. Cambia ciò che si desidera e salva, ed il contenuto dell'elemento sarà aggiornato nel sistema Plone. Puoi modificare ripetutamente il contenuto degli elementi, proprio come puoi farlo con i file presenti sul tuo PC. Ormai avrai apprezzato il fatto che Plone memorizza gli elementi come entità separate, simili a “file” su un computer locale, ma non c'è bisogno di pensarla necessariamente in questo modo. Plone è un CMS (sistema di gestione dei contenuti), in cui il contenuto viene fornito sotto forma di numerosi elementi separati, che possono essere modificati singolarmente a piacimento.

Per fare un esempio di modifica di un contenuto che è un pò diverso dal suo inserimento iniziale, possiamo esaminare la modifica di un'immagine. La modifica di una immagine può essere fatta navigando fino a trovare la singola immagine e facendo clic sul pannello Modifica. Facendo clic sul pannello *Modifica*, verrà visualizzato il seguente pannello *Modifica immagine*:

Nell'esempio in figura, un'immagine chiamata “Eastern Tiger Swallowtail Butterfly” sta per essere modificata. Puoi modificare il titolo e la descrizione, come al solito, e in questo caso potresti lasciare l'impostazione “Mantieni

Eastern Tiger Swallowtail >

View Edit Transform Sharing

## Edit Image

Images can be referenced in documents or displayed in an album.

Default Categorization Dates Ownership Settings

Title

Eastern Tiger Swallowtail Butterfly

Description

Used in item listings and search results.

Image ▀



 [Current image](#) PNG image — 13KB

Keep existing image

Replace with new image:


l'immagine corrente". È anche possibile modificare l'immagine stessa scegliendo "Sostituisci con la nuova immagine". In alternativa, cliccando sul pulsante "Elimina immagine corrente" l'immagine sarà eliminata del tutto.

Si noti anche sulla parte superiore la presenza del tab *Trasforma*, che è pertinente alle immagini, e che offre la possibilità di effettuare diverse trasformazioni dell'immagine:

Eastern Tiger Swallowtail >

View Edit **Transform** Sharing

## Eastern Tiger Swallowtail Butterfly



Original size: 13.5 kB

**Transform**

Flip around vertical axis

Select method

- Flip around vertical axis
- Flip around horizontal axis
- Rotate 90 counterclockwise
- Rotate 180
- Rotate 90 clockwise

Powered by Plone & Python

Quindi, la modifica di un'immagine è un'operazione leggermente diversa rispetto alla sua aggiunta, anche se non di molto.

I pannelli di modifica per gli altri tipi di contenuto sono solitamente simili ai pannelli per l'aggiunta.

### Modifica in linea (opzionale)

La modifica in linea è disabilitata di default nelle ultime versioni di Plone (3.3+). Può essere abilitata tramite il pannello di controllo da un Amministratore del Sito (Configurazione del sito -> Modifica -> Spuntare la checkbox Abilita modifica in linea).

La normale procedura per modificare un contenuto è quello di fare clic sul pannello *Modifica* e utilizzare i relativi campi di input del contenuto. Per i campi di testo, ad esempio Titolo, Descrizione, Testo del documento, ecc, c'è un modo più rapido per farlo, ed è chiamato modifica in linea. La modifica in linea è utilizzata durante la visualizzazione dell'elemento stesso (il pannello *Visualizza* è attivo).

Quando il mouse passa sopra parti di testo modificabili, un piccolo box evidenzierà il testo modificabile. Nella seguente schermata, il cursore del mouse *non* si trova sopra un testo da modificare: titolo della pagina e testo del documento vengono pertanto mostrati come di consueto:

Ma quando il mouse viene spostato sopra il testo del documento, un box lo metterà in evidenza permettendo la modifica:

View Edit Sharing History Actions State: Private

## Eastern Tiger Swallowtail

by [John Smith](#) — last modified Sep 05, 2007 03:59 AM

This boldly colored swallowtail is one of our most spectacular and familiar butterflies. Bright yellow wings with black stripes make this usually very large swallowtail immediately identifiable. Females are dimorphic (have two forms) with some having black wings. The percentage of black form females increases as one moves into areas of warmer climate. These forms are easy to distinguish from other "black" swallowtails, because they lack the median orange spot-band on the hindwing, seen on Black and Spicebush swallowtails, and the iridescent blue of Pipevine swallowtails. The black-form females usually retain a shadow of the typical "tiger" pattern.

This species occupies deciduous woodlands, especially woodland edges and wooded stream systems. It is often present in suburban settings, soaring high among the trees.

— Send this — Print this —

[Eastern Tiger Swallowtail](#) >

View Edit Sharing Actions State: Private

## Eastern Tiger Swallowtail

by admin — last modified May 07, 2010 06:22 AM — [History](#)

The boldly colore swallowtail is one of out most spectacular and familiar butterflies. Bright yellow wings with black stripes make this usually very large swallowtail identifiable. Females are dimorphic (have two forms) with some having black wings. This percentage of black form females increases as one moves into areas of warmer climate. These forms are easy to distinguish from other "black" swallowtails, because they lack the median orange spot-band on the hindwing, seen on Black and Spicebush swallotails, and the iridescent blue of Pipevine swallowtails. The black-form females usually retain a shadow of the typical "tiger" pattern.

This species occupies deciduous woodlands, specially woodland edges and wooded stream systems. It is often present in suburban settings, soaring high among the trees.

Facendo clic all'interno del testo del documento dopo che il box della modifica in linea è apparso, si attiverà l'editor di testo:

Eastern Tiger Swallowtail >

View Edit Sharing Actions ▼ State: Private ▼

## Eastern Tiger Swallowtail

by admin — last modified May 07, 2010 06:22 AM — [History](#)

Style... **B** *I* [List icons] [Table icon] [HTML icon]

The boldly colore swallowtail is one of our most spectacular and familiar butterflies. Bright yellow wings with black stripes make this usually very large swallowtail identifiable. Females are dimorphic (have two forms) with some having black wings. This percentage of black form females increases as one moves into areas of warmer climate. These forms are easy to distinguish from other "black" swallowtails, because they lack the median orange spot-band on the hindwing, seen on Black and Spicebush swallowtails, and the iridescent blue of Pipevine swallowtails. The black-form females usually retain a shadow of the typical "tiger" pattern.

This species occupies deciduous woodlands, specially woodland edges and wooded stream systems. It is often present in suburban settings, soaring high among the trees.

Save Cancel

Puoi cambiare o aggiungere del testo e salvare, e tornare quindi alla visualizzazione normale. Questa procedura è notevolmente più veloce (in termini di numero di click e tempo di attesa) rispetto a quella che prevede di fare clic sul pannello *Modifica* ed attivare l'intero pannello di modifica per tutta la pagina.

Se il mouse viene spostato sopra il titolo, anch'esso editabile, appare un box di modifica in linea:

Facendo clic sul titolo dopo che compare il box, si attiva un campo di editing molto semplice, con due bottoni di scelta Salva e Annulla:

Puoi cambiare il titolo e salvare. Il vantaggio della velocità della modifica in linea si percepisce soprattutto quando si deve modificare qualcosa di molto semplice, come ad esempio un titolo.

### 1.3.3 Viste per una cartella

**Le cartelle hanno il tab Visualizza, che permette di impostare i vari modi in cui può essere mostrato il contenuto della cartella stessa.**

Per la maggior parte dei contenuti, puoi editare il contenuto stesso per cambiare il modo in cui esso appare. Ma le cartelle sono tipi di contenuto particolari. In quanto contenitori di altri elementi, le cartelle possono mostrare il loro

[Eastern Tiger Swallowtail >](#)

**View** Edit Sharing Actions ▼ State: **Private** ▼

## Eastern Tiger Swallowtail

by admin — last modified May 07, 2010 06:22 AM — [History](#)

The boldly colore swallowtail is one of out most spectacular and familiar butterflies. Bright yellow wings with black stripes make this usually very large swallowtail identifiable. Females are dimorphic (have two forms) with some having black wings. This percentage of black form females increases as one moves into areas of warmer climate. These forms are easy to distinguish from other "black" swallowtails, because they lack the median orange spot-band on the hindwing, seen on Black and Spicebush swallotails, and the iridescent blue of Pipevine swallowtails. The black-form females usually retain a shadow of the typical "tiger" pattern.

This species occupies deciduous woodlands, specially woodland edges and wooded stream systemas. It is often present in suburban settings, soaring high among the trees.

[Eastern Tiger Swallowtail >](#)

**View** Edit Sharing Actions ▼ State: **Private** ▼



## Eastern Tiger Swallowtail

Save Cancel

by adm in — last m odified M ay 07, 2010 06:22 AM — [History](#)

The bddly colore swallowtail is one of out most spectacular and familiar butterflies. Bright yellow wings with black stripes make this usually very large swallowtail identifiable. Females are dimorphic (have two forms) with some having black wings. This percentage of black form females increases as one moves into areas of warmer climate. These forms are easy to distinguish from other "black" swallowtails, because they lack the median orange spot-band on the hindwing, seen on Black and Spicebush swallotails, and the iridescent blue of Pipevine swallowtails. The black form females usually retain a shadow of the typical "tiger" pattern.

This species occupies deciduous woodlands, specially woodland edges and wooded stream systemas. It is often present in suburban settings, soaring high among the trees.

contenuto in vari modi, tra loro diversi. Spiegheremo in questa sezione le varie opzioni di visualizzazione del contenuto di una cartella.

Ipotizza uno scenario in cui un appassionato di farfalle, John Smith, ha effettuato l'accesso al suo sito web per lavorare su una sezione dedicata alle farfalle Skipper. Egli naviga fino alla cartella "Skipper" tramite click sui tab principali del sito web, cioè tramite il menù di navigazione, che è posto a sinistra nel layout di default del suo sito Plone. Quando clicca sulla cartella "Skipper", viene mostrato il pannello di visualizzazione standard della cartella, più semplicemente la "vista standard":

The screenshot shows the Plone web interface. At the top right, the user 'John Smith' is logged in. A search bar is present with the text 'Search Site' and a 'Search' button. Below the search bar is a checkbox labeled 'only in current section'. The navigation menu includes 'Home', 'News', 'Events', 'Users', and 'Butterflies' (which is highlighted). The main content area shows a folder named 'Skippers' with a 'Manage portlets' button below it. A green toolbar contains 'Contents', 'View', 'Edit', 'Rules', 'Sharing', 'Actions', 'Display', 'Add new...', and 'State: Published'. The main content area displays a list of items under the heading 'Skippers' by admin, last modified May 03, 2010 10:46 PM. The items listed are: 'Spread-winged Skippers' (by admin, last modified May 03, 2010 05:41 PM), 'Cloudywings' (by admin, last modified May 03, 2010 05:43 PM), 'Duskywings' (by admin, last modified May 03, 2010 05:43 PM), 'Grass-Skippers' (by admin, last modified May 03, 2010 05:44 PM), 'Hesperia Skippers' (by admin, last modified May 03, 2010 05:44 PM), 'Roadside-Skippers' (by admin, last modified May 03, 2010 05:44 PM), and 'Giant-Skippers' (by admin, last modified May 03, 2010 05:45 PM). At the bottom, there is a footer with copyright information: 'The Plone® Open Source CMS/WCM is © 2000-2010 by the Plone Foundation and friends. Plone® and the Plone logo are registered trademarks of the Plone Foundation. Distributed under the GNU GPL license.' Below the footer, it says 'Powered by Plone & Python' and provides links for 'Site Map', 'Accessibility', and 'Contact'.

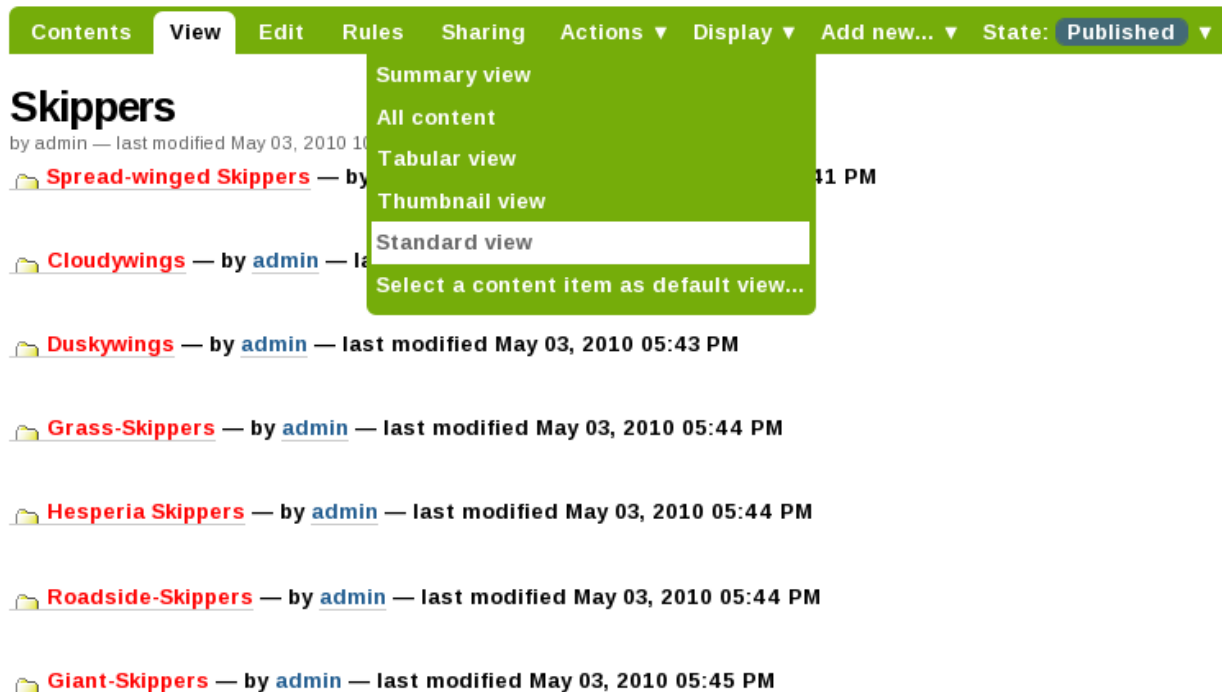
Il tab *Visualizza* mostra sempre il modo in cui un qualsiasi contenuto appare al navigatore anonimo del sito web. Fai click sul tab *Visualizza*, perciò, ogni volta che vuoi vedere come un contenuto che hai modificato viene visualizzato dagli utenti del sito. Nel caso delle cartelle, vedrai la lista dei contenuti in essa compresi, in una delle diverse possibilità di presentazione selezionabili tramite il menù a tendina *Vista*. La vista di default è chiamata *Vista standard*:

Di seguito, invece, come appare la *Vista riassuntiva*:

E ancora, la *Vista tabellare*:

E, infine, la *Vista provini*, che è particolarmente utile per le immagini, ma funziona anche per gli altri tipi di contenuto:

Creare un album fotografico in Plone è molto semplice. Devi solo aggiungere le foto (immagini, tipicamente in formato .jpg) in una cartella ed impostare per la cartella stessa la **Vista provini**. La vista si aggiornerà automaticamente man mano che aggiungi nuove immagini alla cartella, mostrandole in maniera raggruppata all'interno della pagina, se necessario in ragione del numero crescente.



Se stai caricando immagini fotografiche da una macchina fotografica digitale o da uno scanner, ti convertirà probabilmente ridimensionarle sul tuo PC prima di caricarle, perchè spesso esse sono troppo grandi.

### Impostare un singolo contenuto come vista per una cartella

La funzionalità appena descritta che permette di impostare la vista di una cartella come un elenco di contenuti ben si adatta al modo in cui noi pensiamo alle cartelle – come contenitori di contenuti, appunto –. Plone, tuttavia, offre anche un modo facile di impostare come vista di una cartella anche un qualsiasi singolo contenuto della cartella stessa. Questa possibilità massimizza il vantaggio che deriva dal fatto che il menù di navigazione di un sito web Plone si compone in maniera automatica, mappando dinamicamente le sue voci sulla struttura delle cartelle man mano che queste vengono create.

Puoi ad esempio impostare una singola pagina come vista di una cartella, e ciò può tornare utile nel caso volessi mostrare il documento più recente tra quelli presenti all'interno della cartella stessa. Oppure, puoi impostare come vista una collezione, che di per se è già un potente strumento di filtro di contenuti. Le impostazioni della vista di una cartella dovrebbero essere usate con attenzione, poichè cambiano il modo in cui una cartella si comporta, dall'essere un semplice contenitore all'essere un collegamento diretto ad un contenuto. Invece, puoi spesso ottenere ciò che desideri semplicemente usando le collezioni, che saranno descritte più avanti in questo manuale.

Di seguito, proseguiremo analizzando il tab *Contenuti*, per descrivere altre importanti funzioni per l'accesso ai contenuti presenti nella lista all'interno di una cartella.

### 1.3.4 Contenuti delle cartelle

**Il tab Contenuti mostra la lista degli elementi in una cartella. E' il posto dove eseguire semplici operazioni sugli elementi e dove eseguire azioni come copiare, tagliare, incollare, spostare, riordinare, etc.**

Il tab Contenuti delle cartelle è come il "Gestione file" o "Risorse del Computer" dei PC con Windows e Linux o il "Cerca" nei Mac OS X, con funzionalità simili.



[Contents](#) **View** [Edit](#) [Rules](#) [Sharing](#) [Actions](#) ▼ [Display](#) ▼ [Add new...](#) ▼ State: **Published** ▼

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

### [Spread-winged Skippers](#)

[Read More...](#)

### [Cloudywings](#)

[Read More...](#)

### [Duskywings](#)

[Read More...](#)

### [Grass-Skippers](#)

[Read More...](#)

### [Hesperia Skippers](#)

[Read More...](#)

### [Roadside-Skippers](#)

[Read More...](#)

### [Giant-Skippers](#)

[Read More...](#)

Contents **View** Edit Rules Sharing Actions ▼ Display ▼ Add new... ▼ State: **Published** ▼

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

Title	Author	Type	Modified
<a href="#">Spread-winged Skippers</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:41 PM
<a href="#">Cloudywings</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:43 PM
<a href="#">Duskywings</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:43 PM
<a href="#">Grass-Skippers</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:44 PM
<a href="#">Hesperia Skippers</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:44 PM
<a href="#">Roadside-Skippers</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:44 PM
<a href="#">Giant-Skippers</a>	<a href="#">admin</a>	Folder	May 03, 2010 05:45 PM

Contents **View** Edit Rules Sharing Actions ▼ Display ▼ Add new... ▼ State: **Published** ▼

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

Spread-winged Skippers (0)

Cloudywings (0)

Duskywings (0)

Grass-Skippers (0)

Hesperia Skippers (0)

Roadside-Skippers (0)

Giant-Skippers (0)

Cliccando sul tab *Contenuti* di una cartella, come ad esempio la cartella “Skippers” qui sotto, verrà mostrato la pannello *Contenuti*:

The screenshot shows the Plone CMS interface for the 'Skippers' folder. The navigation bar includes 'Home', 'News', 'Events', 'Users', and 'Butterflies'. The 'Skippers' folder is selected, and the 'Contents' tab is active. A table lists seven sub-folders: Spread-winged Skippers, Cloudywings, Duskywings, Grass-Skippers, Hesperia Skippers, Roadside-Skippers, and Giant-Skippers. Each row has a checkbox, a title, size (1 kB), modified date, and state (Private). Below the table are buttons for Copy, Cut, Rename, Delete, and Change State. The footer contains copyright information for Plone Foundation and mentions 'Powered by Plone & Python'.

La scheda *Contenuti* è immediatamente riconoscibile per la presenza delle caselle di spunta (check boxes) accanto alle voci della lista. Spunta le caselle per selezionare più elementi ed eseguire su di essi le funzioni *copia*, *taglia*, *rinomina*, *elimina* o *cambia lo stato*.

Plone ha una “area appunti” interna per la gestione delle operazioni di *copia* e *taglia*. Se selezioni uno o più elementi e premi taglia o copia, sarà aggiunto un pulsante incolla in fondo alla scheda nella stessa riga dove si trovano gli altri pulsanti. Se a questo punto vai in un’altra cartella, vi potrai incollare l’elemento. Utilizzando la funzione taglia, gli elementi rimangono nella cartella di origine – non scompariranno – finchè non saranno incollati da un’altra parte.

Quando si *Rinomina* i contenuti, verrà mostrata una scheda dove inserire un nuovo valore per il *nome breve* (o *id*) dell’elemento, così come per il *titolo*. La differenza tra il *nome breve* ed il *titolo* diventa evidente solo quando si utilizza la funzione *rinomina*, perchè di solito Plone crea automaticamente il *nome breve* dal *titolo* (senza che sia necessario impostarlo). Ma se utilizzi la funzione *rinomina*, allora ti verranno mostrati sia il *nome breve* sia il *titolo*, perchè tipicamente se modifichi uno vorrai modificare anche l’altro. Considera il seguente esempio:

Se vuoi modificare il titolo in “Long-tailed Skippers,” vorrai cambiare anche il nome breve in “long-tailed-skippers.” In questo modo i due valori saranno entrambi corretti ed allineati, così che l’URL dell’elemento (basato sul nome breve), l’indirizzo web, sarà aggiornato rispetto all’elemento stesso. Nota che il nome breve non deve contenere spazi. Utilizza i trattini al posto degli spazi e, se non ce ne sono, fai una copia precisa del titolo. Inoltre, usa solo lettere minuscole per il nome breve. Guarda la pagina Cosa c’è in un nome web? per una descrizione di come Plone gestisce gli indirizzi web e i nomi brevi. Il seguente video include anche la funzione *rinomina*:

## Rename item

Each item has a **Short Name** and a **Title**, which you can change by entering the new details below.

— Giant-Skippers (giant-skippers) —

**New Short Name**  
Short name is the part that shows up in the URL of the item.

**New Title**

L'operazione *cancella* è lineare. Clicca per selezionare uno o più elementi, in seguito premi il pulsante cancella e gli elementi saranno cancellati.

L'operazione *cambia stato* offre un ottimo modo per cambiare lo stato di pubblicazione delle cartelle selezionate (e delle relative sotto-cartelle, se hai selezionato questa opzione). Nel seguente esempio, lo stato di pubblicazione della cartella “Long-tailed Skippers” sarà modificato. Selezionando “Includi gli elementi contenuti” il cambiamento dello stato avrà effetto anche su tutto il contenuto della cartella (incluse eventuali sotto-cartelle). Non dimenticare che questa operazione può essere fatta, ad esempio, per tre cartelle alla volta (con tutti i loro contenuti, comprese le sotto-cartelle), cosicché in un colpo solo puoi velocemente pubblicare, rimuovere dalla pubblicazione ecc..

Utilizza *Shift-click* per selezionare un intervallo di elementi. Questo è molto utile in una cartella con più di una dozzina di elementi e risulta indispensabile in cartelle con centinaia di oggetti.

In aggiunta a queste operazioni, il riordinamento può essere fatto in maniera naturale con il mouse, come descritto nella sezione successiva.

### 1.3.5 Ordinamento elementi

**Il tab dei contenuti contiene una funzione per l'ordinamento veloce e preciso degli elementi di una cartella.**

Considera la seguente cartella, chiamata “Skippers,” che contiene informazioni su questo tipo di farfalle. Spesso, quando aggiungiamo contenuti, non li inseriamo nell'ordine finale che vorremmo ottenere. L'ordine desiderato non è sempre quello alfabetico, ma in questo esempio possiamo presumere di volere proprio questo tipo di ordinamento. Sotto puoi vedere che le sottocartelle di Skipper non sono in ordine alfabetico:

Per muovere l'elemento più in alto chiamato “Spread-winged Skippers” in fondo alla lista, dovrai cliccare nella colonna di sinistra per l'Ordinamento (quella con il simbolo dei due punti ripetuti) e trascinare la riga nella posizione desiderata:

Il trascinamento si esegue tenendo premuto il pulsante del mouse mentre sposti l'elemento. L'oggetto che stai spostando diventerà giallo e inizierà a muoversi:

Quando rilascerai il pulsante del mouse, l'elemento si posizionerà in quel punto:

### 1.3.6 Link Precedente - Successivo

**La visualizzazione dei link automatici Precedente-Successivo per i contenuti presenti in una cartella può essere abilitata nel tab Impostazioni della cartella stessa.**

## Publishing process

An item's status (also called its review state) determines who can see it. Another way to control the visibility of an item is with its *Publishing Date*. An item is not publicly searchable before its publishing date. This will prevent the item from showing up in portlets and folder listings, although the item will still be available if accessed directly via its URL.

**Affected content**

	Title	Size	Modified	State
<input checked="" type="checkbox"/>	History	1 kB	Apr 01, 2010 12:46 PM	Published

**Publishing Date**  
The date when the item will be published. If no date is selected the item will be published immediately.

2010 / March / 29 02 : 35 PM

**Expiration Date**  
The date when the item expires. This will automatically make the item invisible for others at the given date. If no date is chosen, it will never expire.

-- / -- / -- -- : -- --

**Comments**  
Will be added to the publishing history. If multiple items are selected, this comment will be attached to all of them.

**Change State**  
Select the transition to be used for modifying the items state.

No change  
 Retract  
 Send back

Contents
View
Edit
Rules
Sharing
Add new... ▼

## Skippers

by admin — last modified May 03, 2010 10:46 PM — [History](#)

[Up one level](#)

Select: **All**

		Title	Size	Modified	State
☰	<input type="checkbox"/>	<a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
☰	<input type="checkbox"/>	<a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
☰	<input type="checkbox"/>	<a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
☰	<input type="checkbox"/>	<a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰	<input type="checkbox"/>	<a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰	<input type="checkbox"/>	<a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
☰	<input type="checkbox"/>	<a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private

Il tab *Impostazioni* viene visualizzato al click del tab *Modifica* della cartella. C'è una casella di spunta per abilitare i link Precedente-Successivo per i vari elementi contenuti nella cartella:

Una volta abilitati, i link Precedente-Successivo compariranno automaticamente, se necessario, man mano che i vari contenuti verranno aggiunti alla cartella:

Tre pagine sono state create all'interno della cartella Cloudywings, ed è stata selezionata la "Pagina Due" (che in questo esempio non ha testo). Alla fine della "Pagina Due" sono presenti i link "Precedente: Pagina Uno" e "Successivo: Pagina Tre"

Questa è una funzione *veramente* utile!

### 1.3.7 Cancellare contenuti da una cartella

**I vari contenuti possono essere cancellati facilmente da una cartella.**












Alcune volte è necessario cancellare un contenuto, spesso per rimpiazzarlo con una versione aggiornata. Oppure è semplicemente necessario cancellare quel contenuto, per diverse necessità. Nell'esempio del contenuto relativo alla farfalla swallowtail aggiunto per errore alla cartella Skippers, esso può essere semplicemente cancellato, invece che tagliato ed incollato in qualche altra cartella:

Nell'esempio mostrato sopra, la cartella *Eastern Tiger Swallowtail* sarà cancellata al click del bottone *Cancella*.

Interi cartelle possono essere cancellate con un solo click, perciò è bene fare molta attenzione, anche se questa è una regola che vale in generale quando si lavora ad un PC. Tutti noi abbiamo imparato a nostre spese che è sempre meglio fare un ultimo controllo prima della cancellazione, per essere sicuri che cancellare è proprio quello che vogliamo fare.

Click and drag a row by grabbing it in the order column to change the order of rows

Click on the title to go to the item

	Title	Size	Modified	State
	 <a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
	 <a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
	 <a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
	 <a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
	 <a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
	 <a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
	 <a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private

Click a checkbox to perform one of the operations along the bottom

Copy Cut Rename Delete Change State

Fig. 1.2: Esempio di Ordinamento

	Title	Size	Modified	State
 <input type="checkbox"/>	 <a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private
 <input type="checkbox"/>	 <a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	 <a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	 <a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private

	Title	Size	Modified	State
 <input type="checkbox"/>	 <a href="#">Cloudywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	 <a href="#">Duskywings</a>	1 kB	May 03, 2010 05:43 PM	Private
 <input type="checkbox"/>	 <a href="#">Grass-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Hesperia Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Roadside-Skippers</a>	1 kB	May 03, 2010 05:44 PM	Private
 <input type="checkbox"/>	 <a href="#">Giant-Skippers</a>	1 kB	May 03, 2010 05:45 PM	Private
 <input type="checkbox"/>	 <a href="#">Spread-winged Skippers</a>	1 kB	May 03, 2010 05:41 PM	Private



Contents View **Edit** Rules Sharing

# Edit Folder

A folder which can contain other items.

Default Categorization Dates Ownership **Settings**

**Allow comments**

If selected, users can add comments to this item.

**Exclude from navigation**

If selected, this item will not appear in the navigation tree

**Enable next previous navigation**

This enables next/previous widget on content items contained in this folder.

Save Cancel

- Cloudywings
  - Page One
  - Page Two
  - Page Three

---

- Duskywings

---

- Giant-Skippers

---

- Grass-Skippers

View Edit Sharing Actions ▼ State: **Published** ▼

Cloudywings >

## Page Two

by John Smith — last modified May 05, 2010 10:15 AM — [History](#)

◀ Previous: Page One    Next: Page Three ▶

Contents View Edit Rules Sharing Add new... ▼

## Skippers

by Manager — last modified May 05, 2010 10:13 AM — [History](#)

[Up one level](#)

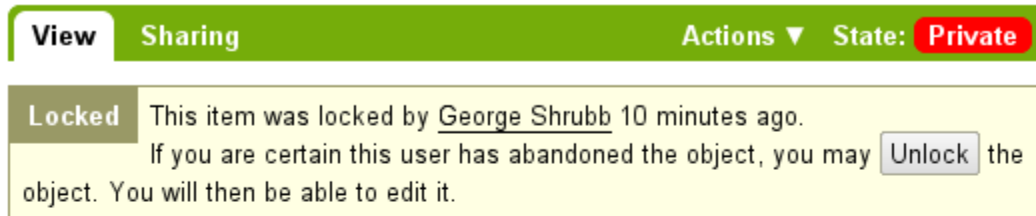
Select: [All](#)

	Title	Size	Modified	State	Order
<input type="checkbox"/>	<a href="#">Cloudywings</a>	1 kB	May 05, 2010 10:17 AM	Published	::
<input type="checkbox"/>	<a href="#">Duskywings</a>	1 kB	May 05, 2010 10:14 AM	Published	::
<input type="checkbox"/>	<a href="#">Giant-Skippers</a>	1 kB	May 05, 2010 10:14 AM	Published	::
<input type="checkbox"/>	<a href="#">Grass-Skippers</a>	1 kB	May 05, 2010 10:14 AM	Published	::
<input checked="" type="checkbox"/>	<a href="#">Eastern Tiger Swallowtail</a>	1 kB	May 05, 2010 10:26 AM	Published	::
<input type="checkbox"/>	<a href="#">Hesperia Skippers</a>	1 kB	May 05, 2010 10:28 AM	Published	::
<input type="checkbox"/>	<a href="#">Roadside Skippers</a>	1 kB	May 05, 2010 10:29 AM	Published	::
<input type="checkbox"/>	<a href="#">Spread-winged Skippers</a>	1 kB	May 05, 2010 10:29 AM	Published	::


### 1.3.8 Blocco e sblocco automatico dei contenuti

Plone visualizza un messaggio che riporta se un contenuto è bloccato, chi l'ha bloccato e da quanto tempo è stato bloccato - in tal modo, non potrai interferire con i cambiamenti apportati a quel contenuto da un altro utente.

Quando qualcuno fa un click sul tab Modifica di un contenuto, quest'ultimo viene immediatamente bloccato. Questa funzione previene che due persone editino contemporaneamente lo stesso contenuto, e che un utente sovrasciva accidentalmente con le proprie modifiche quelle fatte da un altro utente. In questo esempio, George Schrubb ha iniziato ad editare il contenuto "Widget Installation". Quando Jane Smythe (anche lei con i permessi di modifica dell'oggetto) visualizza lo stesso contenuto, vedrà quanto mostrato nella figura che segue:



## Widget Installation

 by [George Schrubb](#) — last modified May 02, 2010 11:51 AM

How to install and use Widget.

### Step One

1. Proin bibendum metus at odio.
2. Aenean mollis egestas odio.

Una volta che George ha finito di editare il contenuto e fatto click sul bottone Salva, il contenuto viene automaticamente sbloccato e reso disponibile agli altri editori (sempre che, come ovvio, abbiano i permessi di modifica sul contenuto in questione).

In ogni caso, se è evidente che George non sta lavorando in quel momento sul contenuto (ad esempio, il messaggio riporta che il contenuto è stato bloccato diversi giorni prima e non pochi minuti prima), allora è la stessa Jane che può "sbloccarlo" e renderlo disponibile per nuove modifiche.

Nelle versioni a partire da Plone 3.3:

Se un utente abbandona la pagina di modifica di un contenuto senza un click sui bottoni Salva o Cancella, il blocco del contenuto rimane attivo per i successivi dieci minuti, trascorsi i quali, il contenuto viene automaticamente sbloccato. Questa funzione "timeout" è importante soprattutto per tutti quei browser, come ad esempio Safari, che non eseguono correttamente l'azione javascript "on-unload".

Se desideri disabilitare le funzioni di blocco dei contenuti, devi accedere al pannello di controllo di Plone (Configurazione Sito -> Sito) e deselezionare la casella di spunta *Enable locking for through-the-web edits*

### 1.3.9 Controllo di versione

Una panoramica su come visualizzare la cronologia delle versioni di un elemento, confrontare le versioni, visualizzare in anteprima le versioni precedenti e ripristinare versioni precedenti.

### Creare una nuova versione

Plone include una funzione per gestire le versioni. Per impostazione predefinita, i seguenti tipi di contenuti hanno il controllo di versione abilitato:

- Pagina
- Notizia
- Eventi
- Collegamento

Si noti che tutti gli altri tipi di contenuto mantengono la storia del flusso del workflow associato.

I tipi di contenuto possono essere configurati per avere il controllo di versione abilitato/disabilitato attraverso il pannello di Configurazione del Sito, alla voce “Tipi di contenuto”.

Quando modifichi un elemento, puoi utilizzare il campo **commento alle modifiche** in fondo; il commento alle modifiche verrà memorizzato nella cronologia delle versioni dell’elemento. Se il commento alle modifiche viene lasciato vuoto, Plone includerà una nota standard: “Revisione iniziale”.

Una nuova versione viene creata ogni volta che un elemento viene salvato. Il controllo di versione tiene traccia di qualsiasi modifica effettuata: contenuti, metadata, impostazioni, etc.

### Visualizzazione della cronologia delle versioni

Una volta salvato un oggetto, è possibile utilizzare il link **Cronologia** situato nella parte superiore della pagina. Con un semplice click sul link è possibile visualizzare la Cronologia in una finestra sovrapposta alla pagina:



La versione più recente è la prima voce dell’elenco. La viewlet della Cronologia fornisce le seguenti informazioni:

- Il tipo di modifica (al contenuto o al workflow)
- quale utente ha fatto la modifica
- in che data e ora è stata fatta la modifica

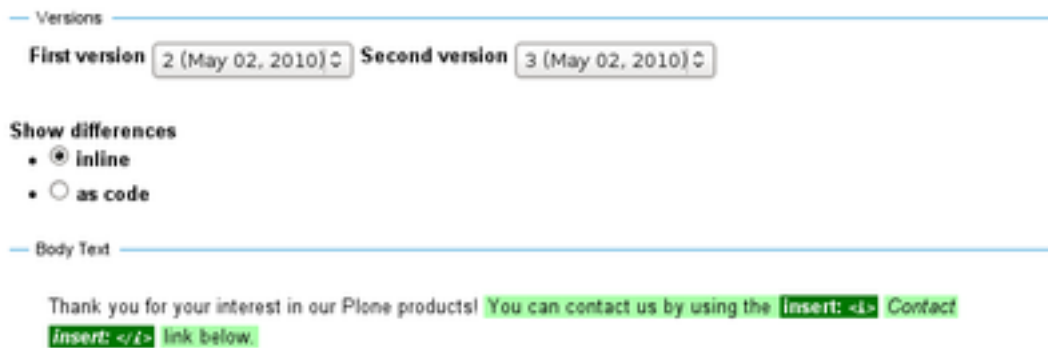
## Confrontare le versioni

Dalla viewlet della Cronologia puoi confrontare qualsiasi versione precedente con quella corrente o qualsiasi altra versione con quella appena prima.

Per confrontare qualsiasi versione precedente con quella appena prima, cliccare sul link *Confronta* collocato tra le due versioni nella finestra della Cronologia.



Cliccando su questo link, vedrai una schermata come questa, in cui è possibile vedere le differenze fra le due versioni:



In questo esempio, il testo in rosso è quello che è stato cancellato e il testo in verde è quello che è stato aggiunto alla versione più recente. È possibile visualizzare le differenze tra le versioni in modalità **in linea** o **come codice**.



È inoltre possibile confrontare qualsiasi versione precedente con la versione *corrente* cliccando sul link *Confronta con versione attuale* nella finestra della Cronologia, situato all'estrema destra di ogni versione elencata.

### Visualizzare e tornare alle versioni precedenti

**Puoi fare una anteprima di qualsiasi versione precedente** di un documento cliccando il link *Visualizza* alla destra di ogni versione elencata.

**Per tornare ad una versione precedente**, clicca sul pulsante *Ripristina questa versione* alla destra di ogni versione elencata.

## 1.3.10 Modalità di presentazione

**Plone viene fornito con la possibilità di creare semplici presentazioni di diapositive.**

La Modalità di Presentazione è una funzione speciale del tipo di contenuto Pagina. Puoi abilitare la Modalità di Presentazione modificando la pagina, entrando nella linguetta **Impostazioni**. Nota che lì sarà presente la checkbox Modalità di Presentazione. Una volta selezionata, un link apparirà nella vista della pagina per dare la possibilità ad un utente di visualizzarla nella Modalità di Presentazione.

### Come creare una diapositiva

Tutto il contenuto di una presentazione rimane in una sola pagina. Non devi creare una pagina per ogni diapositiva. Una diapositiva viene creata quando vedi la classe Intestazione (h1) nella pagina - queste intestazioni indicano effettivamente a Plone dove si vuole far iniziare una diapositiva.

Non ci sono limiti al numero di diapositive che puoi aggiungere in una presentazione. Ti basta inserire più tags Intestazione (h1) nella tua pagina ed il contenuto tra quel tag h1 e quello successivo diverrà il contenuto della tua diapositiva.

### Come Formattare una Diapositiva

E' molto importante notare che i contenuti con associato lo **Stile Paragrafo Normale non vengono visualizzati nelle diapositive**. Le diapositive sono pensate per visualizzare informazioni di riepilogo, non blocchi di testo. Per questo, devi dare uno stile diverso dal Paragrafo Normale al contenuto di ogni diapositiva. Esempi di questi stili sono:

- Intestazione (h1)
- Sottotitolo (h3)
- Definizioni di liste
- Liste non ordinate
- Liste ordinate
- Literal
- Pull-quote
- Call out
- Evidenziato

### 1.3.11 Copia di lavoro

**La Copia di Lavoro ti permette di avere due versioni del tuo contenuto in parallelo.**

Quando un sito Plone viene creato per la prima volta, ci sono diverse funzioni aggiuntive che possono essere abilitate, tra cui la “Copia di Lavoro”. Se il sito Plone che stai usando non presenta l’opzione “Estrai versione” nel menu Azioni, devi contattare l’amministratore del sito e richiedere che il “Supporto alla Copia di Lavoro” venga installato.

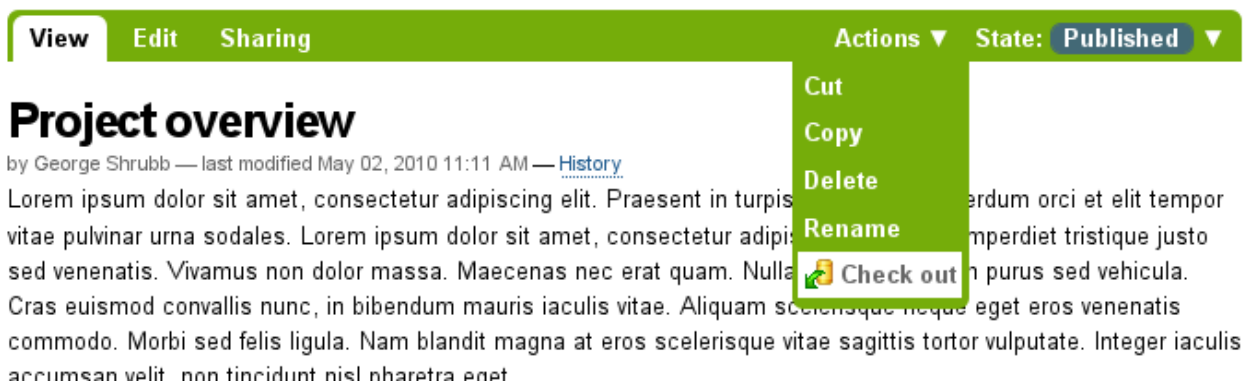
#### Panoramica

In precedenza potresti esserti trovato in una situazione come questa: hai pubblicato un contenuto e lo devi aggiornare con frequenza, ma vuoi che la vecchia versione continui ad esistere sul sito web finchè non hai quella nuova da pubblicare. Vuoi anche che il nuovo contenuto sostituisca quello attuale, ma ti piacerebbe mantenere la storia di quello vecchio. La Copia di Lavoro rende tutto questo possibile.

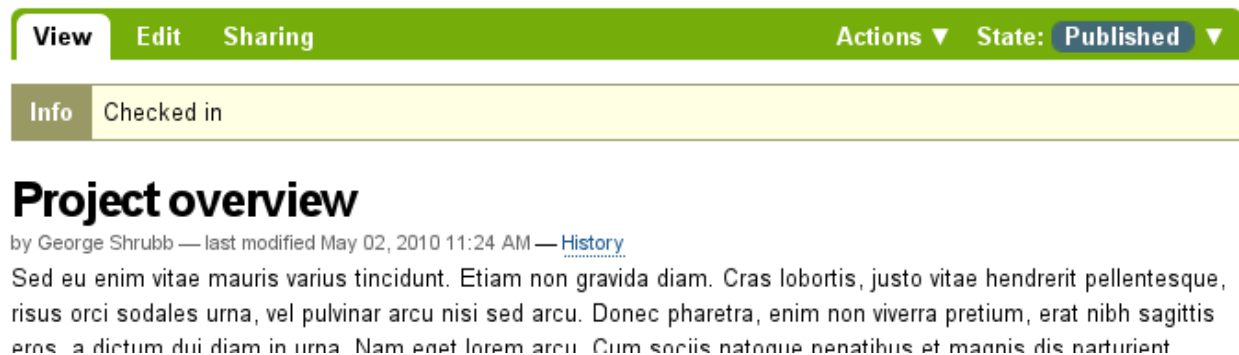
In sostanza, “estrai” la versione attualmente pubblicata del contenuto, creandone così una “copia di lavoro”. A questo punto, potrai modificare la copia di lavoro (mettendoci tutto il tempo che ti serve) e, quando la nuova versione sarà pronta per essere pubblicata, utilizzando l’azione “crea versione” la tua copia di lavoro sostituirà quella online. Dietro le quinte, Plone sostituirà il contenuto originale con quello nuovo, nell’esatta posizione e con lo stesso indirizzo web e archiverà la vecchia versione come parte della storia nel controllo di versione del contenuto nuovo.

#### Utilizzare la funzione “Estrai”

In primo luogo, raggiungi la pagina che intendi rivedere. Poi dal menu “Azioni”, seleziona “Estrai”:



Apparirà un messaggio per informarti che da quel momento stai lavorando su una copia di lavoro:



Ora puoi liberamente modificare la copia locale del contenuto pubblicato. Il contenuto originale risulterà “bloccato” – ovvero, nessun altro potrà modificare la versione pubblicata finché avrai una copia di lavoro estratta. Questo impedirà che, mentre stai modificando la tua copia di lavoro, altre modifiche vengano apportate (e conseguentemente perse) sulla versione pubblicata.

**Locked** This item is being edited by [George Shrubbs](#) in a [working copy](#) created on 11:14 AM. ([View changes](#))

**Locked** This item was locked by [George Shrubbs](#) 3 minutes ago.

**View** **Sharing** **Actions** ▼ **State:** **Published**

### Project overview

 by [George Shrubbs](#) — last modified May 02, 2010 11:11 AM — [History](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent in turpis neque. Cras interdum orci et elit tempor vitae pulvinar urna sodales. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla imperdiet tristique justo sed venenatis. Vivamus non dolor massa. Maecenas nec erat quam. Nulla tristique aliquam purus sed vehicula. Cras euismod convallis nunc, in bibendum mauris iaculis vitae. Aliquam scelerisque neque eget eros venenatis commodo. Morbi sed felis ligula. Nam blandit magna at eros scelerisque vitae sagittis tortor vulputate. Integer iaculis

### Utilizzare la funzione “Crea versione”

Quando sei pronto a sostituire la tua copia locale con quella pubblicata, ti basta semplicemente selezionare “Crea versione” dal menu “Azioni”:

**Locked** This is a working copy of [Project overview](#), made by [George Shrubbs](#) on 11:14 AM. ([View changes](#))

**View** **Edit** **Sharing** **Actions** ▼ **State:** **Private** ▼

**Info** Changes saved.

**Project overview**  
by [George Shrubbs](#) — last modified May 02, 2010 11:19 AM — [History](#)

Sed eu enim vitae mauris varius tincidunt. Etiam non gravida diam. Cras interdum orci et elit tempor vitae pulvinar urna sodales. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla imperdiet tristique justo sed venenatis. Vivamus non dolor massa. Maecenas nec erat quam. Nulla tristique aliquam purus sed vehicula. Cras euismod convallis nunc, in bibendum mauris iaculis vitae. Aliquam scelerisque neque eget eros venenatis commodo. Morbi sed felis ligula. Nam blandit magna at eros scelerisque vitae sagittis tortor vulputate. Integer iaculis

- Cut
- Copy
- Delete
- Rename
- Check in**
- Cancel check-out**

Ti verrà richiesto di inserire un commento legato alla creazione della nuova versione. Compilalo e clicca su “Crea versione”:

Il tuo contenuto aggiornato diventerà la nuova copia pubblicata.

Verrai inoltre informato che non esisterà più una copia di lavoro del documento nella tua cartella personale.

Nota che non è necessario (ed infatti non è consigliata) l’uso del menu “Stato” con una copia di lavoro. Se tuttavia lo utilizzi senza volere, non farti prendere dal panico. Ti basta tornare nella tua copia di lavoro e utilizzare la funzione “Crea versione” dal menu “Azioni”.



View Edit Sharing

## Check in Project overview

Checking in this working copy will replace the existing item with the working copy.

### Check-in Message

Enter a message to be saved alongside the check-in. This should explain what was changed, for audit purposes.

Lorem Ipsum text updated to last version.

Check in Cancel

View Edit Sharing

Actions ▼ State: **Published** ▼

Info Checked in

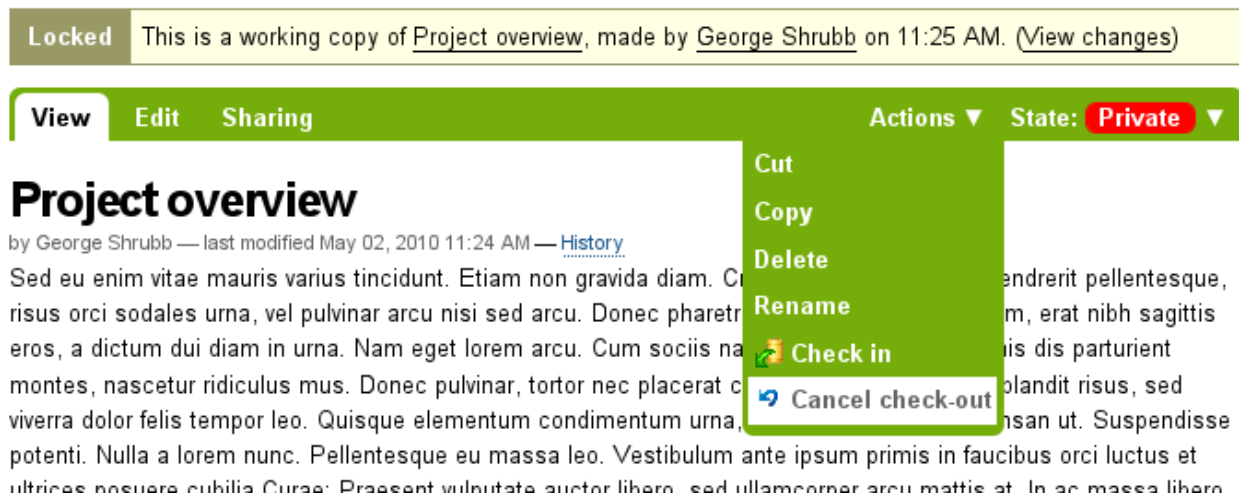
## Project overview

by George Shrubbs — last modified May 02, 2010 11:24 AM — [History](#)

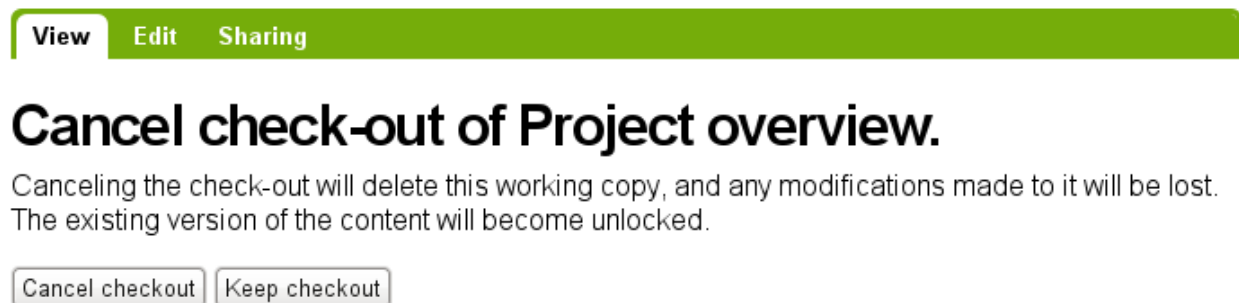
Sed eu enim vitae mauris varius tincidunt. Etiam non gravida diam. Cras lobortis, justo vitae hendrerit pellentesque, risus orci sodales urna, vel pulvinar arcu nisi sed arcu. Donec pharetra, enim non viverra pretium, erat nibh sagittis eros. a dictum dui diam in urna. Nam eget lorem arcu. Cum sociis natoque penatibus et magnis dis parturient

## Cancellare una “Copia di lavoro”

Se, per qualsiasi motivo, devi cancellare una copia di lavoro e **non vuoi salvare le tue modifiche**, vai semplicemente nella copia di lavoro e clicca su “Annulla il check-out”:



Ti verrà chiesto di confermare il comando “Annulla il check-out” o di “Mantenere il checkout”:



Nota che se un utente che ha estratto una copia di lavoro non è disponibile per effettuare la pubblicazione della copia di lavoro o annullarla, gli utenti con il ruolo di Manager possono accedere alla copia di lavoro ed effettuare sia la creazione della versione che l’annullamento della copia di lavoro. Questo perché non tutti i collaboratori hanno il privilegio di eseguire la funzione “Crea versione”. Se tale opzione non è presente dal menu Azioni:

1. Utilizza il menu “Stato”.
2. Sottoponi per pubblicazione.
3. Chiedi ad un revisore di **non** cambiare lo stato.
4. Chiedi invece al revisore di effettuare il “Crea versione” per tuo conto.

La procedura “Crea versione” si occuperà della gestione dello stato.

## 1.4 Usare TinyMCE come visual editor

### 1.4.1 Introduzione

#### Introduzione a TinyMCE.

TinyMCE è un editor WYSIWYG (What You See Is What You Got) in Javascript basato su una piattaforma web indipendente, con il quale è possibile creare contenuti HTML sul proprio sito web. TinyMCE supporta molti Sistemi Operativi e browsers. Alcuni esempi sono: Mozilla, Internet Explorer, Firefox, Opera, Safari e Chrome. TinyMCE ha alle spalle una consistente base di utenti e una community di sviluppo molto attiva.

TinyMCE è l'editor visuale di default a partire da Plone 4.0, sebbene Kupu sia comunque disponibile per gli utenti che lo preferiscono. Si è deciso di fornire TinyMCE come editor di default perchè Kupu non è un componente adeguatamente mantenuto, laddove TinyMCE può vantare sia un utilizzo molto più diffuso in diverse community, sia una più ampia disponibilità di plugin, oltre a funzioni native molto interessanti come, ad esempio, la possibilità di aggiungere link sia interni sia esterni utilizzando lo stesso pulsante.

### 1.4.2 Nozioni di base

#### Opzioni base di TinyMCE.

L'editor TinyMCE di default ha il seguente aspetto:

Nella parte alta puoi vedere la barra degli strumenti, sotto l'area di testo ed in fondo una barra per il ridimensionamento. Se trascini l'angolo in basso a destra puoi allargare o ridurre la finestra dell'editor.

#### Barra degli strumenti

La seguente tabella descrive le funzioni ed il risultato di ogni pulsante.

### 1.4.3 Inserire delle immagini

#### Una panoramica delle opzioni disponibili per l'inserimento di immagini in TinyMCE.

L'editor TinyMCE ti permette di inserire delle immagini caricate in Plone nella tua pagina, utilizzando il bottone sulla barra degli strumenti di TinyMCE:

Cliccando su questo bottone, si apre la finestra per inserire un immagine:

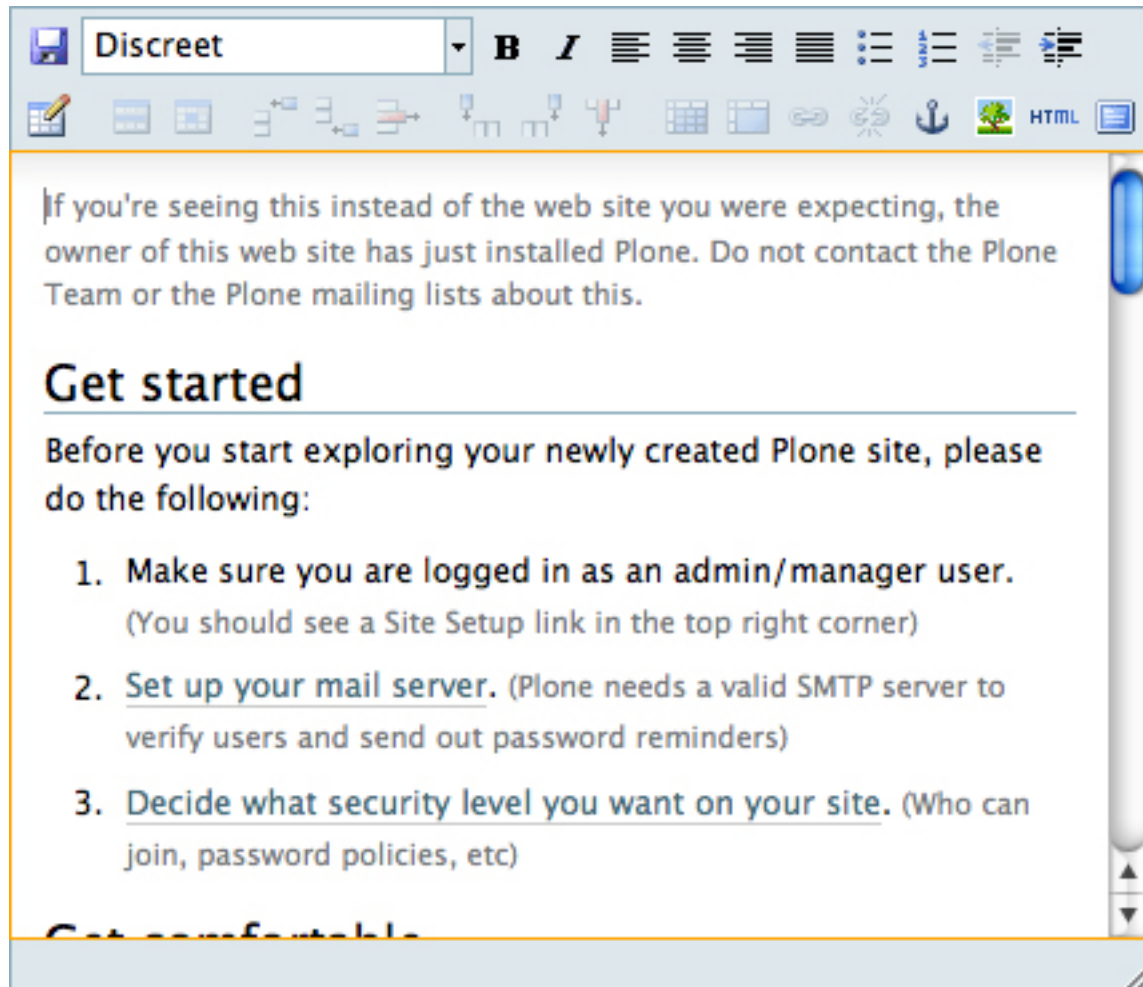
Le tre colonne della finestra sono:


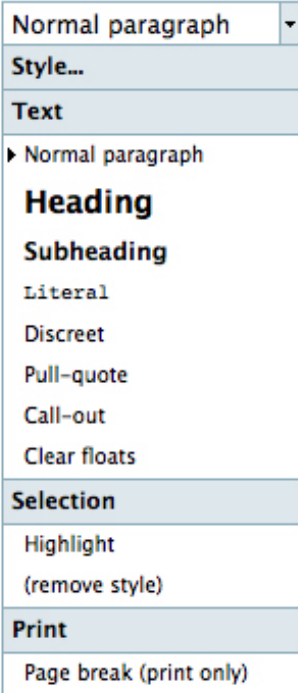







- nella prima colonna c'è la lista di navigazione delle cartelle
- nella seconda colonna c'è la lista dei contenuti della cartella corrente
- nella terza colonna c'è l'anteprima dell'immagine, le opzioni di allineamento, le dimensioni disponibili per mostrare l'immagine e la didascalia.

Nell'esempio sopra, è stata selezionata l'immagine di una rosa - rose.png ( l'immagine originale è piuttosto grande, 600\*450 pixel).

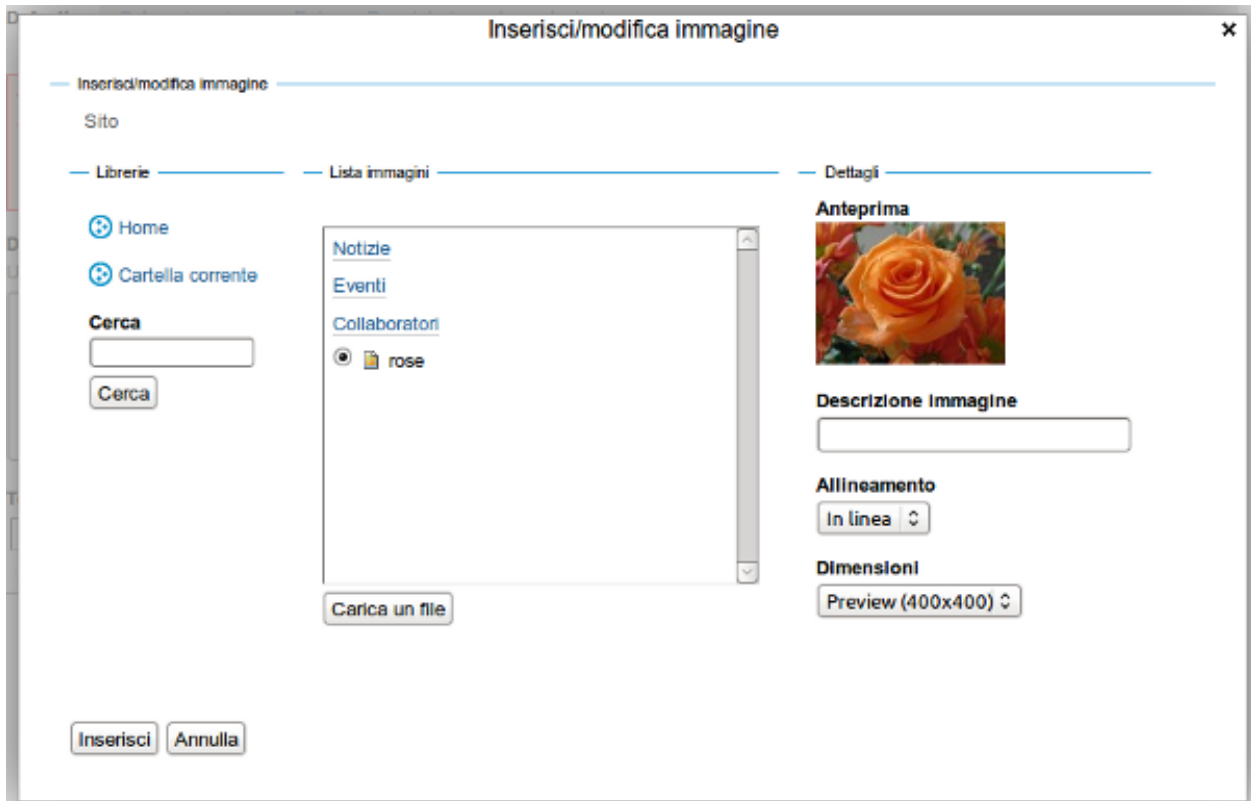
L'immagine verrà posizionata nella pagina in accordo al "Allineamento" scelto e verrà generato il seguente codice HTML:

- A sinistra ()
- A destra ()



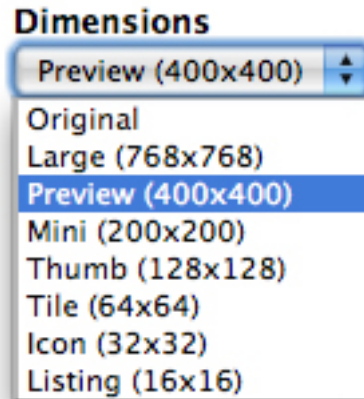
Icon	function	description	example	
	save	Saves changes		
	text style	Normal paragraph	text	
		Style...		
		Text	<b>text</b>	
		▶ Normal paragraph		
		<b>Heading</b>	Subheading	text
		<b>Subheading</b>	Literal	<b>text</b>
		Literal		
		Discreet	Discreet	text
		Pull-quote		
		Call-out		
		Clear floats	Pull-quote	Pull-quote
		<b>Selection</b>		
		Highlight	Call-out	
(remove style)	Clear floats			
<b>Print</b>				
Page break (print only)	Highlight			
	(remove style)			
	Page break (print only)			
	bold	Bolds selected text	<b>test</b>	
	italic	Italicizes selected text	<i>test</i>	
	justify left	Aligns the selected text to the left	text	
	justify center	Aligns the selected text to the center of the screen	text	
	justify right	Aligns the selected text to the right	text	
	justify full	Aligns the selected text to both left and right	text	
	bulleted list	Creates a bulleted list	<ul style="list-style-type: none"> <li>• item 1</li> <li>• item 2</li> <li>• item 3</li> </ul>	

1.4. Usare TinyMCE come visualizzatore	Creates a numbered list	1. item 1	81
		2. item	



- In linea (``)

Puoi anche scegliere la dimensione dell'immagine di cui hai bisogno. Questo è particolarmente utile quando hai a che fare con immagini di grandi dimensioni, poichè non c'è bisogno di utilizzare Photoshop o altre applicazioni esterne per ritagliare o modificare l'immagine. Il menu a tendina "Dimensioni" ti consente di scegliere tra diverse dimensioni e formati:



- Large (``);
- Preview (``);
- Mini (``) - questa è la dimensione minima per la visualizzazione dell'immagine;
- Thumb (``) - a thumb(inch)- dall'immagine verrà estratta una miniatura; (un pò più piccola di 2,5cm);
- Tile (``) - dall'immagine viene ricavata una 'mattonella';
- Icon (``) - dall'immagine verrà ricavata un'icona;
- Listing (``) - dall'immagine verrà ricavata una piccola immagine in stile 'elenco';

### Didascalia dell'immagine

In TinyMCE è possibile inserire una didascalia sotto l'immagine. La didascalia è presa dalla descrizione dell'immagine. Il testo alternativo è tratto dal titolo dell'immagine. Il testo alternativo e la didascalia si aggiornano automaticamente se l'immagine viene aggiornata.

Per abilitare questa funzione accedi a **Configurazione del sito -> Editor TinyMCE**. Assicurati di selezionare **Consenti la 'titolazione' delle immagini** nel pannello **Tipi di risorse**.

Quando aggiungi un'immagine sul sito, puoi inserire una breve descrizione e questa verrà utilizzata come didascalia.

Ora, quando crei una pagina e inserisci un'immagine in essa, seleziona il box Didascalia:

Salvando le modifiche, il risultato dovrebbe essere simile a questo, con la descrizione dell'immagine inserita come didascalia all'interno di una cornice:

## 1.4.4 Inserire collegamenti (links)

Inserire collegamenti interni, esterni ed ancore

[Layout](#)[Barra degli strumenti](#)**Tipi di risorse**[Librerie](#) **Collega utilizzando gli UID**

Il collegamenti agli oggetti di questo sito possono essere realizzati utilizzando un id univoco. In questo modo i collegamenti rimarranno validi anche se l'oggetto cui si riferiscono verrà rinominato o spostato all'interno del sito.

 **Consenti la 'titolazione' delle immagini**

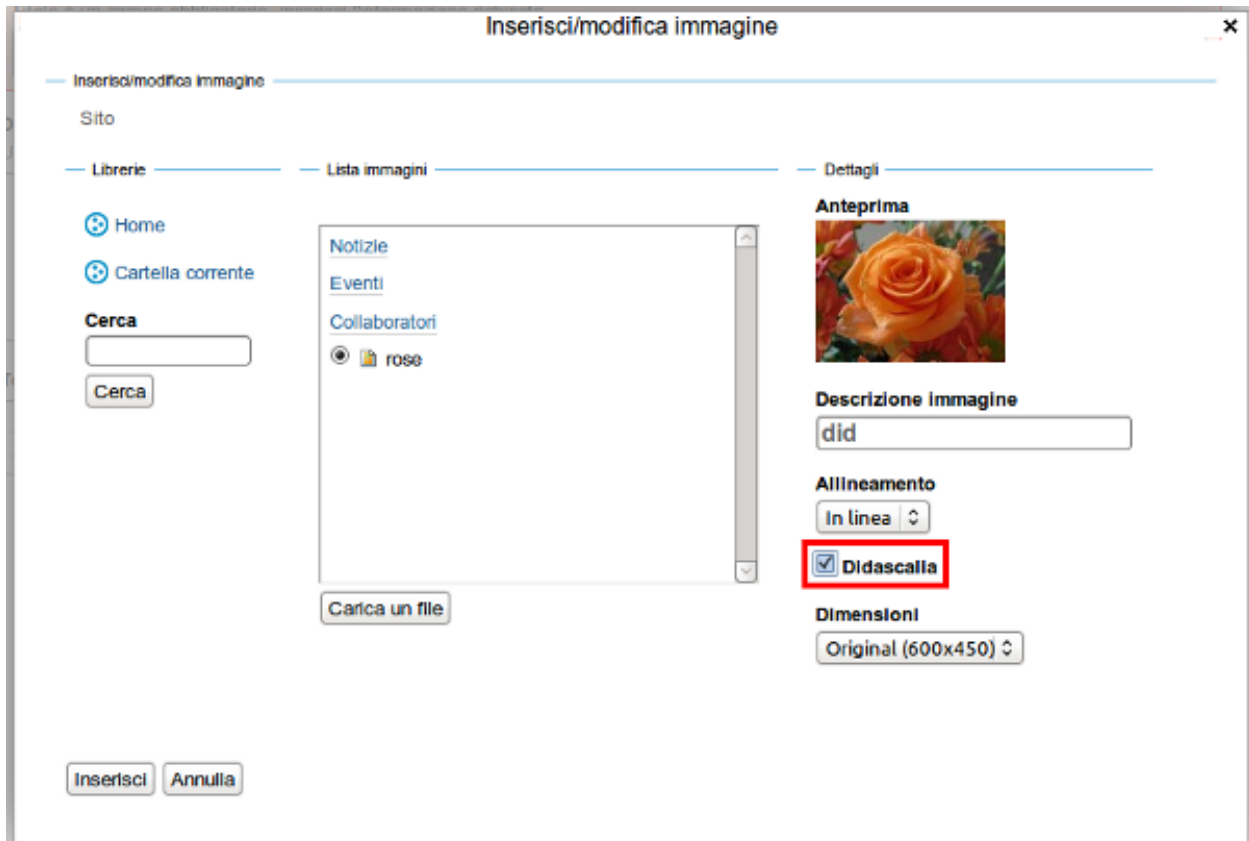
Verrà aggiunta in automatico una didascalia alle immagini.

**Default** ■[Categorizzazione](#)[Date](#)[Proprietario](#)[Impostazioni](#)**Titolo****Descrizione**

Usato nell'elenco degli elementi e nei risultati delle ricerche.

**Immagine** ■





## pagina di esempio

creato da [admin](#) — ultima modifica 11/12/2012 23:40 — [Cronologia](#)



didascalia

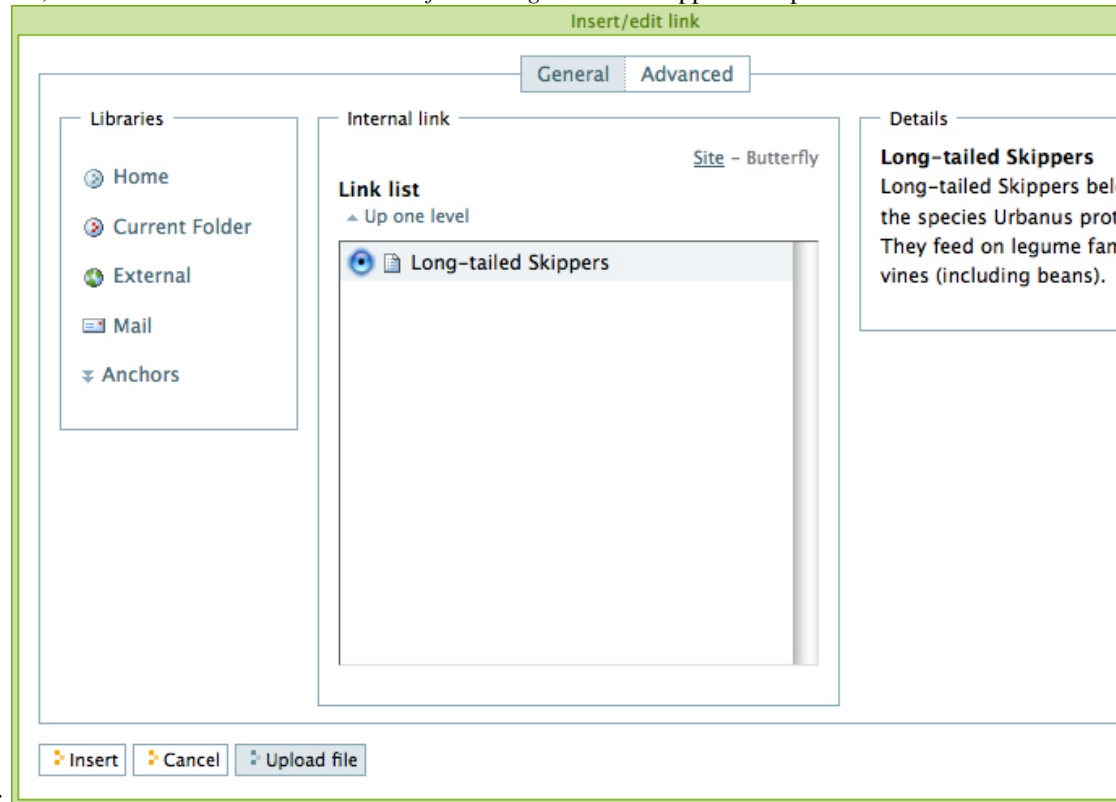
Lorem ipsum dolor sit amet, consectetur adipiscing elit. In a massa velit. In scelerisque, lectus ac mattis porttitor, odio magna molestie lorem, in hendrerit velit purus sit amet lorem. Mauris nisi nibh, volutpat ut commodo vel, faucibus eu metus. Suspendisse eu eros ut mi dignissim sagittis. Morbi vestibulum felis vitae diam viverra tempor. Nunc in convallis sapien. Pellentesque a velit at erat lacinia adipiscing. Praesent nisi sem, mollis ultricies iaculis a, iaculis et leo. Nunc eget lectus ac purus ultrices porttitor vel non arcu. Maecenas nisi risus, imperdiet vitae suscipit vulputate, fringilla a eros. Vestibulum eu sapien justo. Aenean porta, est quis congue feugiat, nunc enim cursus diam, ut gravida neque velit ut arcu. Proin auctor,

mauris quis elementum suscipit, mauris mauris vulputate purus, id vulputate sem erat quis diam.

Sed quam ante, adipiscing in facilisis quis, pretium at purus. Donec consequat ipsum eget dolor dignissim ultricies tincidunt nibh adipiscing. Proin at felis nunc. Pellentesque sagittis luctus fermentum. Quisque luctus turpis quis nulla rutrum ut tristique odio laoreet. Etiam id dui elit. Suspendisse gravida aliquet consequat. Vestibulum ut ligula sit amet lorem lacinia auctor et id sem. Duis in ligula quis turpis faucibus semper in vitae mauris. Aliquam convallis varius tempor. Aliquam a velit velit, non dictum risus. Suspendisse tincidunt consectetur fringilla.

## Collegamenti interni

Seleziona una parola o una frase, fai click sull'icona *Inserisci/modifica collegamento* ed apparirà il pannello *In-*



*serisci/modifica collegamento* :

Il pannello va usato selezionando la cartella Home o quella corrente, per iniziare a navigare all'interno del sito Plone alla ricerca della cartella, della pagina o dell'immagine cui far puntare il collegamento. Nell'esempio della figura sopra, la pagina "Long-tailed Skippers" è stata scelta come destinazione del link. Una volta chiuso il pannello, un collegamento alla pagina "Long-tailed Skippers" verrà creato per la parola o la frase scelta come testo del collegamento.

## Collegamenti esterni

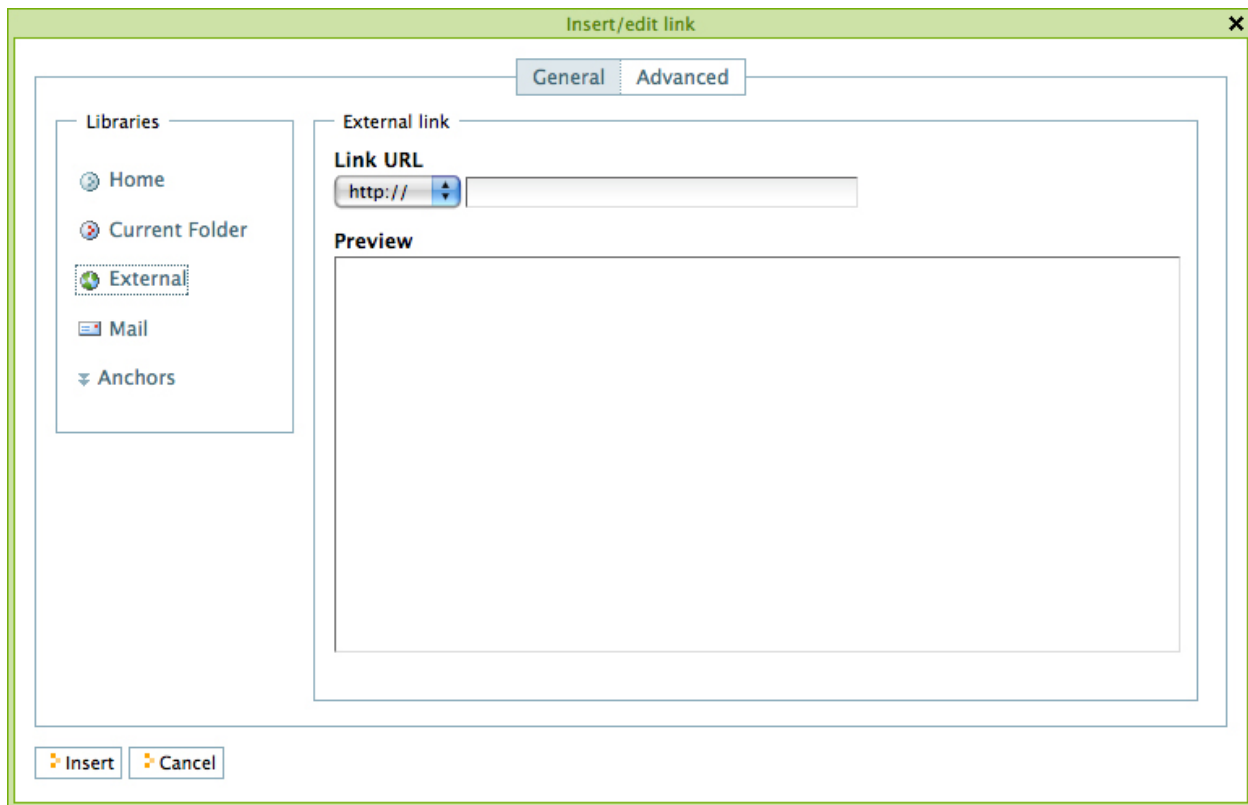
Seleziona una parola o una frase, fai click sull'icona *Inserisci/modifica collegamento*, seleziona "Barra degli strumenti esterna" nella colonna Librerie; si aprirà un pannello simile a questo:

Digita l'indirizzo del sito web esterno cui vuoi far puntare il collegamento nella casella dopo <http://>. Quando premi Invio sulla tastiera o clicchi in un altro punto del pannello diverso dal campo di input, apparirà una preview, per permetterti di verificare che il sito web scelto sia quello giusto. Se incolli l'indirizzo web, assicurati che la stringa <http://> non venga duplicata. Quindi clicca *Ok*. Il collegamento esterno verrà impostato per la parola o la frase che hai selezionato.

## Ancore

Le ancore sono 'segnaposti' all'interno di un documento, legate a titoli, sottotitoli o altri stili impostati per il documento stesso. Ad esempio, per la pagina "Eastern Tiger Swallowtail, con sottotitoli Description," "Habitat," "Behavior," "Conservation Status," e "Literature," è possibile impostare un insieme di collegamenti ai vari sottotitoli utilizzando le ancore.

Per prima cosa, crea il documento impostando i vari sottotitoli nel corpo, e riscrivi i sottotitoli all'inizio del documento:



Ora, crea le ancore per ciascun sottotitolo. Per creare un'ancora, muovi il cursore all'inizio del sottotitolo e fai click sull'icona "Inserisci/modifica ancora". Inserisci il nome dell'ancora nell'apposito campo. Quindi, fai click su *Ok*

Poi, seleziona uno dei sottotitoli che hai riscritto all'inizio del documento e fai click sull'icona *Inserisci/modifica collegamento*

Selezionando *Ancore* dalla colonna "Librerie\*", apparirà un pannello che ti permette di selezionare il sottotitolo cui far puntare il collegamento:

Il tab "Collegamento ad un'ancora" apparirà. La parte destra del pannello mostra le ancore che sono state impostate per il documento. Nell'esempio, l'ancora *Description* è stata scelta come destinazione del collegamento (ed impostata per la parola *Description* all'inizio del documento).

Ci si può sbizzarrire con questa potente funzionalità, impostando ancore per i vari stili del documento ed inserendo i relativi collegamenti all'interno delle parti narrative di un documento. Ciò può rivelarsi particolarmente utile nel caso di documenti di grosse dimensioni.

## 1.4.5 Inserire Tabelle

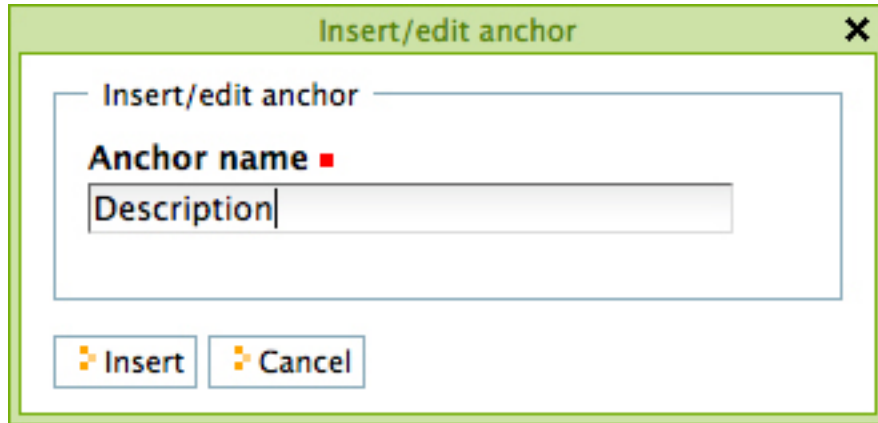
### Inserire, aggiornare e cancellare tabelle, colonne, righe e celle.

Le tabelle sono ideali per mostrare dati in formato tabulare e liste. Per aggiungere una tabella, posiziona il cursore nel punto in cui vuoi che la tabella sia creata e fai click sull'icona *Inserisci una nuova tabella*. Avrai accesso al pannello *Inserisci/Modifica Tabella*:

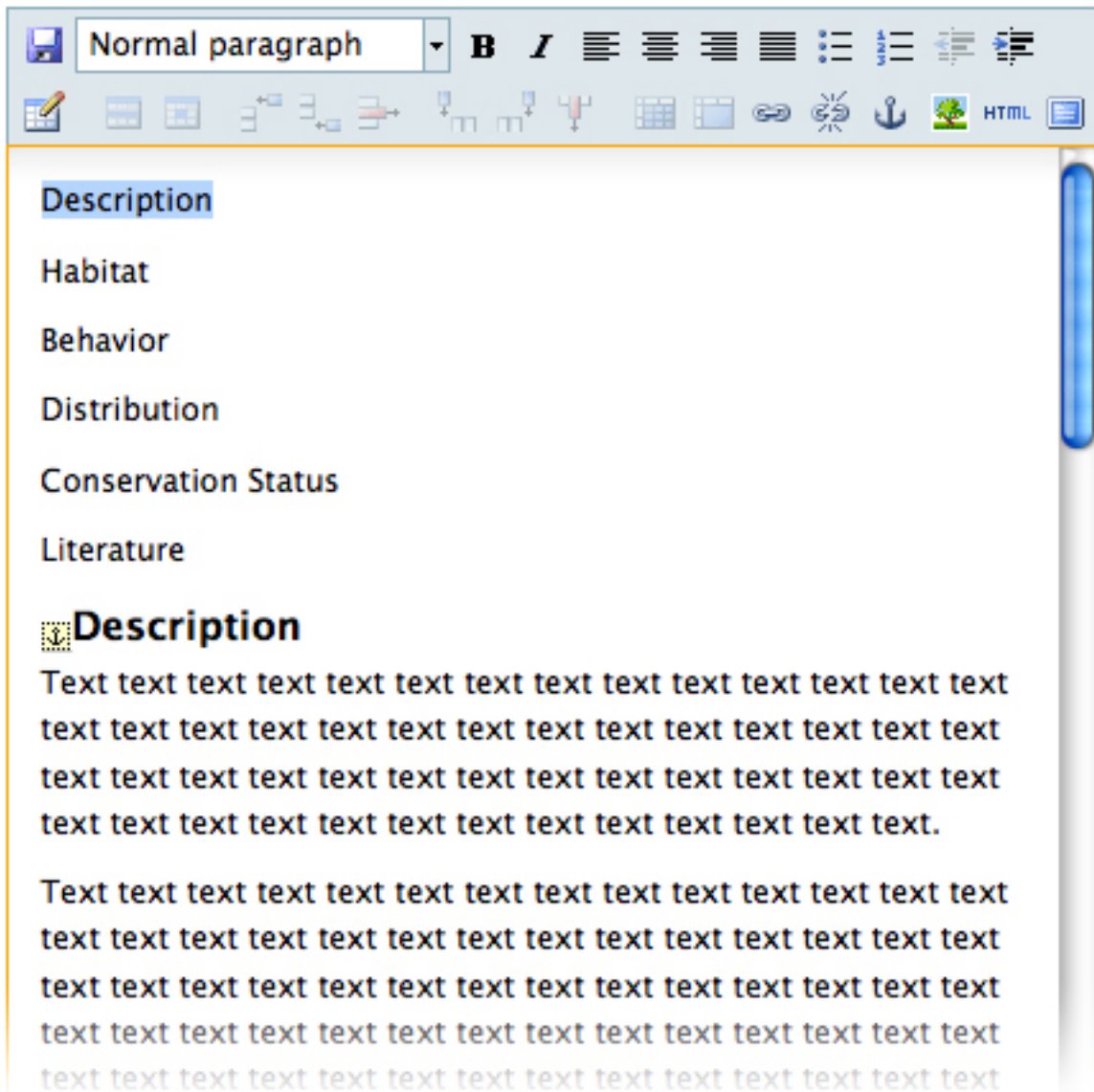
Impostare il numero di colonne e righe è intuitivo. Se lo desideri, puoi aggiungere anche un sommario della tabella tramite l'apposito campo. Le classi assegnabili alla tabella ti permettono di decidere lo stile da applicare. Puoi scegliere tra opzioni come quelle mostrate nella figura che segue:

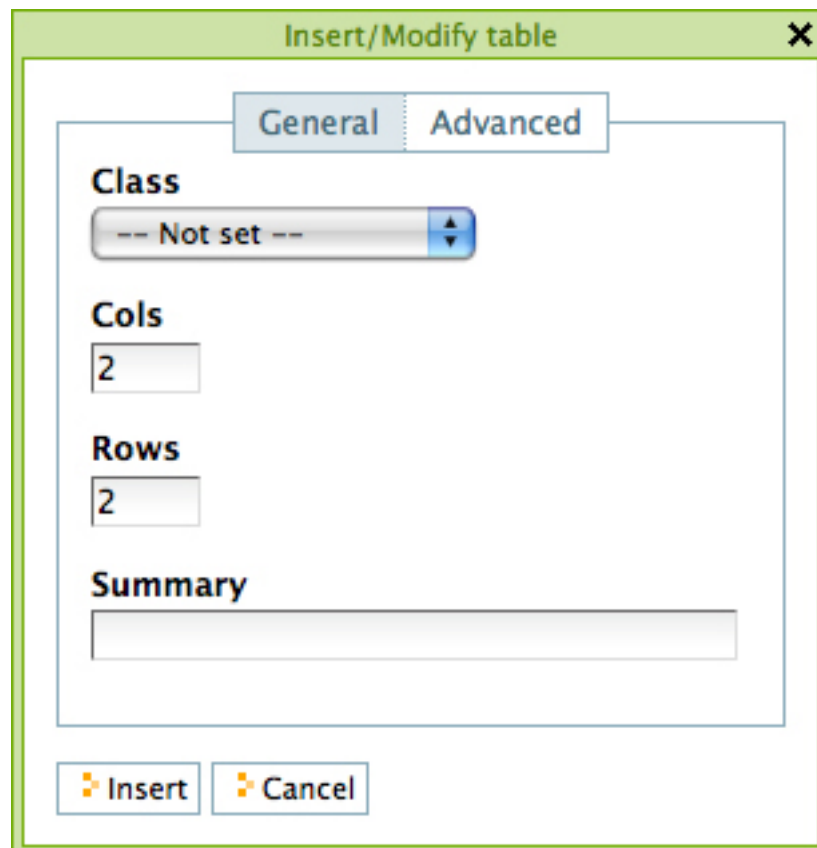
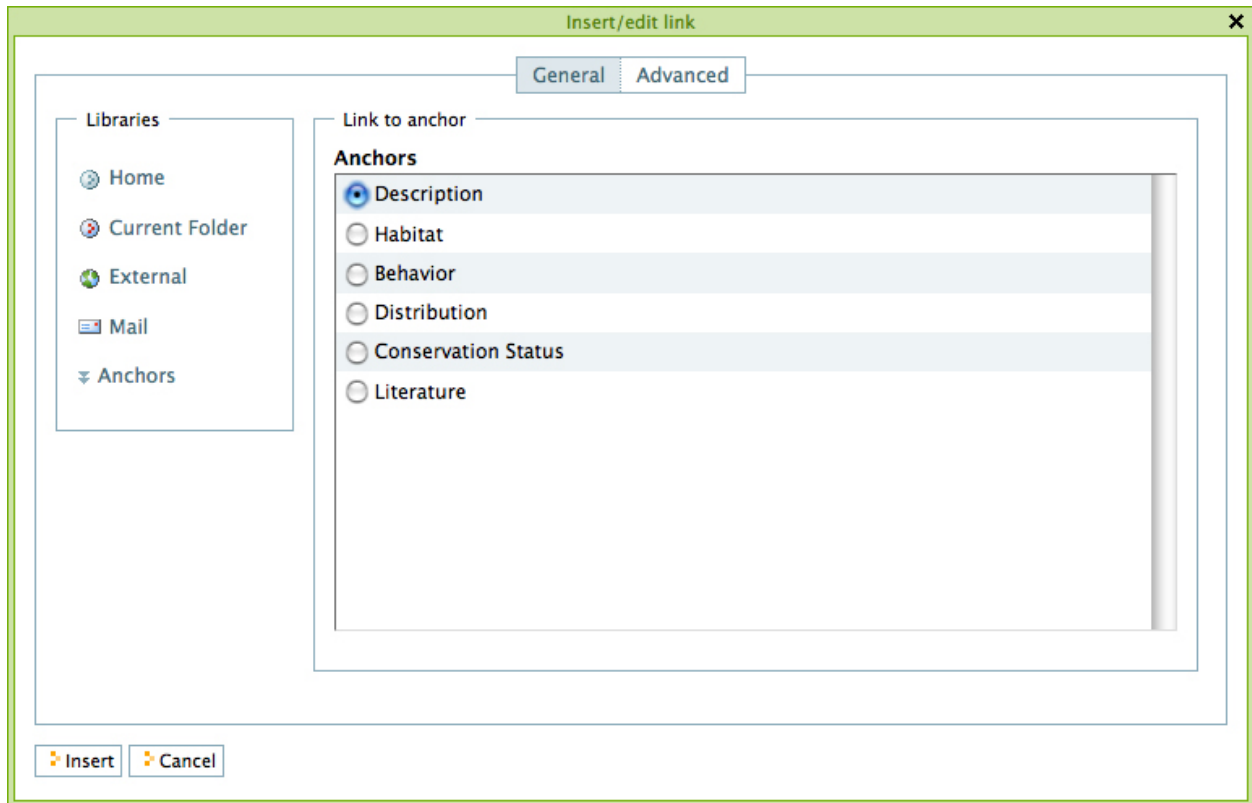
Di seguito alcuni esempi degli stili disponibili:

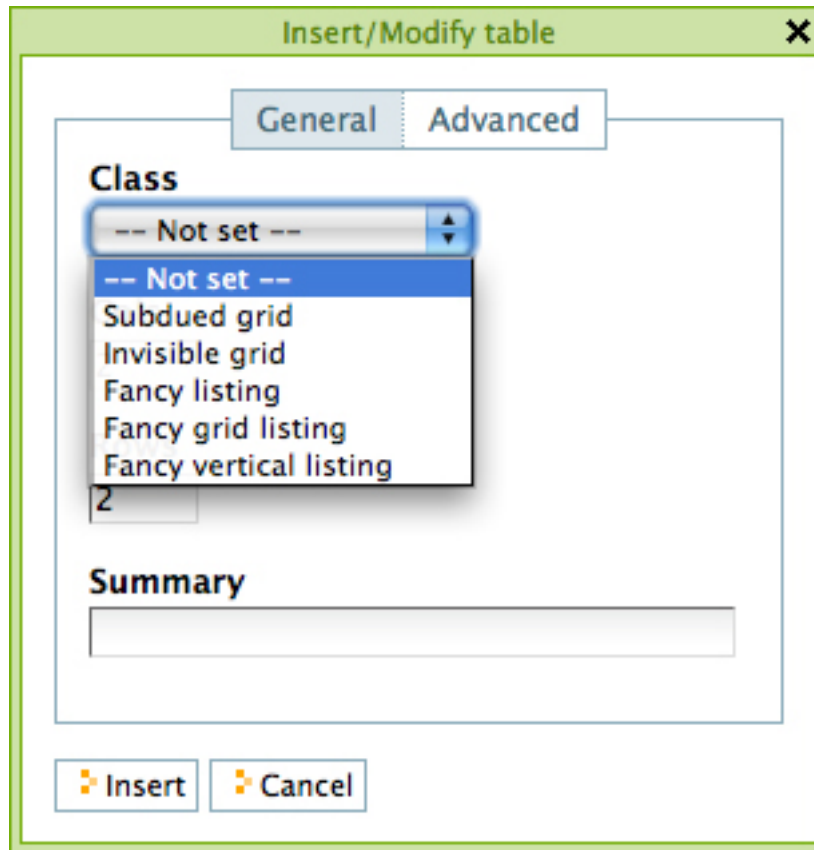




### Body Text







Subdued grid:

Invisible grid:

Fancy listing:

Fancy grid listing:

Fancy vertical listing:

Una volta creata la tabella, è possibile far apparire i comandi di ridimensionamento manuale, cliccando in una cella qualsiasi:

Nell'esempio mostrato dalla figura sopra, il cursore viene posizionato nella prima cella in alto a sinistra: ciò fa apparire dei piccoli quadratini lungo i bordi della tabella, che possono essere utilizzati per modificare le dimensioni della tabella stessa. Cliccando su una cella qualsiasi compariranno inoltre nella barra degli strumenti i comandi specifici della tabella: potremo così editare le proprietà di una riga o di una cella, aggiungere o eliminare righe o colonne, dividere o unire celle.

## 1.5 Usare Kupu come visual editor

**Kupu è una piattaforma indipendente, un editor Javascript HTML WYSIWYG, web based. Questo significa che ti consente di creare contenuti HTML sul tuo sito web.**

Da Plone 4 in poi, TinyMCE è il visual editor predefinito per i nuovi siti. Tuttavia, Kupu è ancora disponibile se lo preferisci. Controlla la sezione [Impostare il tuo profilo](#) per imparare a impostare Kupu come il tuo editor predefinito.

Una tipica barra degli strumenti Kupu assomiglia a questa:



Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet





Il formato testo viene normalmente lasciato con l'impostazione HTML, ma alcuni siti offrono testo strutturato o altri linguaggi di markup per la modifica delle pagine.

Le icone sono:

- grassetto,
- italico,
- giustificato a sinistra,
- giustificato centrato,
- giustificato a destra,
- elenco numerato,
- elenco puntato,
- elenco di definizioni,
- tab a sinistra (blocco),
- tab a destra (blocco),
- immagine (l'icona "albero"),
- link interno (l'icona "anello della catena", crea un link a un'altra pagina nel sito),
- link esterno (l'icona "mondo", crea un link a una pagina web altrove),
- ancora (l'icona "ancora", crea un link ad una sezione specifica di una pagina),
- tabella (aggiunge una tabella con righe e colonne),
- HTML editing diretto (l'icona "HTML"; se si conosce HTML, modifica direttamente il codice HTML per la pagina),
- menu a tendina per lo stile del testo.

### 1.5.1 Immagini

Posizionare il cursore all'interno del testo di una pagina, fare click sull'icona "albero". Si aprirà questo pannello:

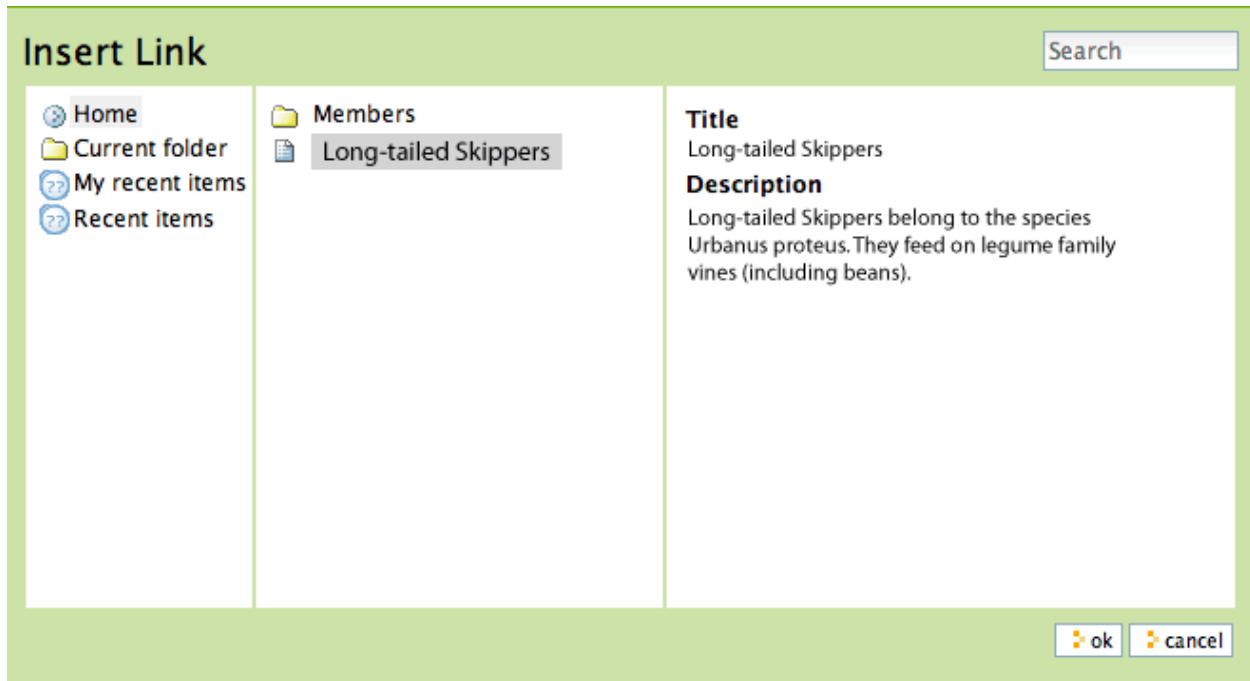


Fare click su “Cartella Corrente” nella parte sinistra del pannello, se non è già evidenziata. La cartella corrente è la cartella che contiene la pagina che si sta modificando - tutte le pagine sono contenute in cartelle. Ci sono molti modi per gestire la memorizzazione di immagini, tra cui avere una cartella centrale di immagini, ma un metodo comune è quello di memorizzare le immagini mostrate su una pagina nella cartella che contiene la pagina stessa (la cartella corrente). In questo modo, le pagine e le immagini ad esse associate sono memorizzate insieme all’interno della struttura delle cartelle. Se fai click sul pulsante Carica, ti verrà chiesto di selezionare un’immagine sul tuo computer e caricarla. Dopo aver selezionato l’immagine da caricare, il pannello di destra ti permetterà di dare all’immagine un titolo per l’uso sul sito web, e diverse opzioni per la posizione e il dimensionamento dell’immagine. Facendo click su OK viene caricata l’immagine e collocata nella pagina. Lo stesso pannello apparirà se si seleziona un’immagine nella pagina e si fa click sulla stessa icona “albero”, per modificare le opzioni dell’immagine selezionata o sostituirla con un’altra immagine. Sei responsabile per il dimensionamento e l’editing delle immagini sul tuo computer prima di caricarle, ma un modo semplice per gestire le immagini da usare sulla maggior parte delle pagine web è quello di fare una copia di un’immagine sul computer, quindi ridimensionarla a qualcosa come 1000 pixel nella dimensione più grande. Questo è una dimensione ragionevole per il caricamento - non è necessario caricare le tue immagini gigantesche provenienti dalla fotocamera digitale. Plone creerà automaticamente diversi formati di un’immagine caricata, tra cui “large”, “mini”, e altre dimensioni. Si sceglie la dimensione che si desidera utilizzare quando si carica o modifica l’immagine con l’icona “albero”. È anche possibile impostare la dimensione dell’immagine modificando direttamente il codice HTML.

### 1.5.2 Collegamenti Interni

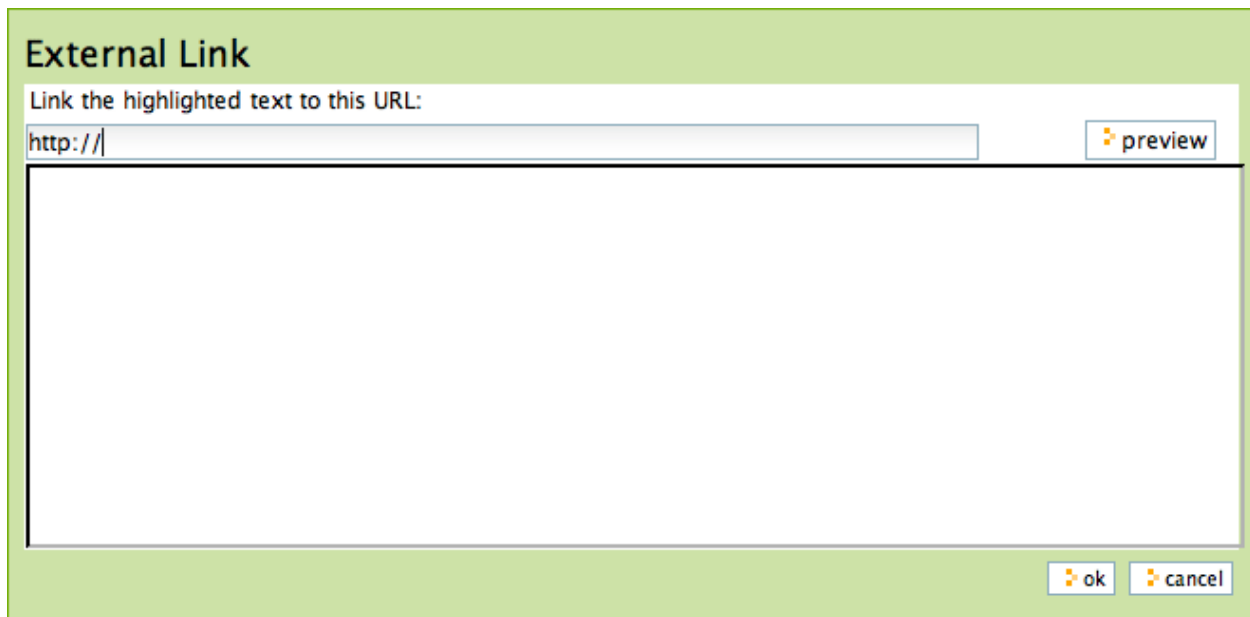
Selezionare una parola o una frase, fare click sull’icona *Collegamento interno*, e apparirà il pannello *Inserisci collegamento*:

Puoi utilizzare questo pannello facendo click sulla cartella Home o sulla cartella Corrente, per iniziare la navigazione nel sito web Plone e per trovare una cartella, una pagina, o l’immagine verso cui creare un collegamento. Nel precedente esempio, è stata scelta per il collegamento una pagina denominata “Long-tailed Skippers”. Dopo che il pannello si è chiuso, verrà impostato un collegamento alla pagina “Long-tailed Skippers” per la parola o per la frase selezionata per il collegamento.



### 1.5.3 Collegamenti Esterni

Selezionare una parola o una frase, fare click sull'icona *Collegamento esterno*, e apparirà il pannello *Collegamento Esterno*:



Digitare l'indirizzo web del sito esterno nel box che inizia con `http://`. È possibile fare click su *anteprima* se hai bisogno di controllare l'indirizzo. Se incolli l'indirizzo web, assicurati di non duplicare `http://` all'inizio dell'indirizzo. Poi fai click su *OK*. Il collegamento esterno verrà impostato per la parola o la frase selezionata.

### 1.5.4 Le Ancore

Le Ancore sono dei link in un documento che puntano direttamente ad una sezione del documento stesso, sulla base di titoli, sottotitoli o altri stili definiti all'interno del documento stesso. Ad esempio, in una pagina chiamata "Eastern Tiger Swallowtail", in cui sono definite delle sottosezioni intitolate "Description", "Habitat", "Behavior", "Conservation Status" e "Literature", è possibile creare un semplice elenco di link a queste sottovoci (che puntano direttamente alla sottovoce all'interno del documento) utilizzando le ancore.

Per prima cosa, crea il documento con i sottotitoli e riscrivi i sottotitoli all'inizio del documento come per un indice:

Quindi seleziona ogni sottotitolo riscritto all'inizio del documento e clicca l'icona a forma di ancora:

Apparirà una maschera che permetterà di selezionare quale sottotitolo deve essere collegato all'ancora:

Verrà visualizzata la scheda *Collegamento all'ancora*. La sezione di sinistra mostrerà una lista di stili che posso essere utilizzati nel documento. Nel nostro esempio, i sottotitoli verranno usati per ogni sezione, ed è il caso più comune. La parte destra mostrerà i sottotitoli che sono stati creati nel documento. Nel nostro caso verrà selezionato il sottotitolo *Description* (per creare il collegamento con lo stesso riscritto all'inizio del documento).

Puoi essere molto creativo con questa potente funzione, tessendo un testo dinamico, intelligente, con diversi riferimenti interni alle varie sezioni della narrazione. Questa funzionalità è particolarmente importante per i documenti lunghi.

### 1.5.5 Tabelle

Le tabelle sono ideali per la visualizzazione di dati tabulari e liste. Per aggiungere una tabella posiziona il cursore nel punto desiderato e fai click sull'icona *Aggiungi tabella*. Vedrai il pannello *Aggiungi tabella*:

L'impostazione delle righe e delle colonne è semplice. Se selezioni il box *Crea Intestazioni* avrai un posto dove digitare le intestazioni della colonna per la tabella. La classe della tabella si riferisce al suo stile. Hai scelte come queste:

Ecco alcuni esempi di questi stili per la tabella:

**plain:**

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

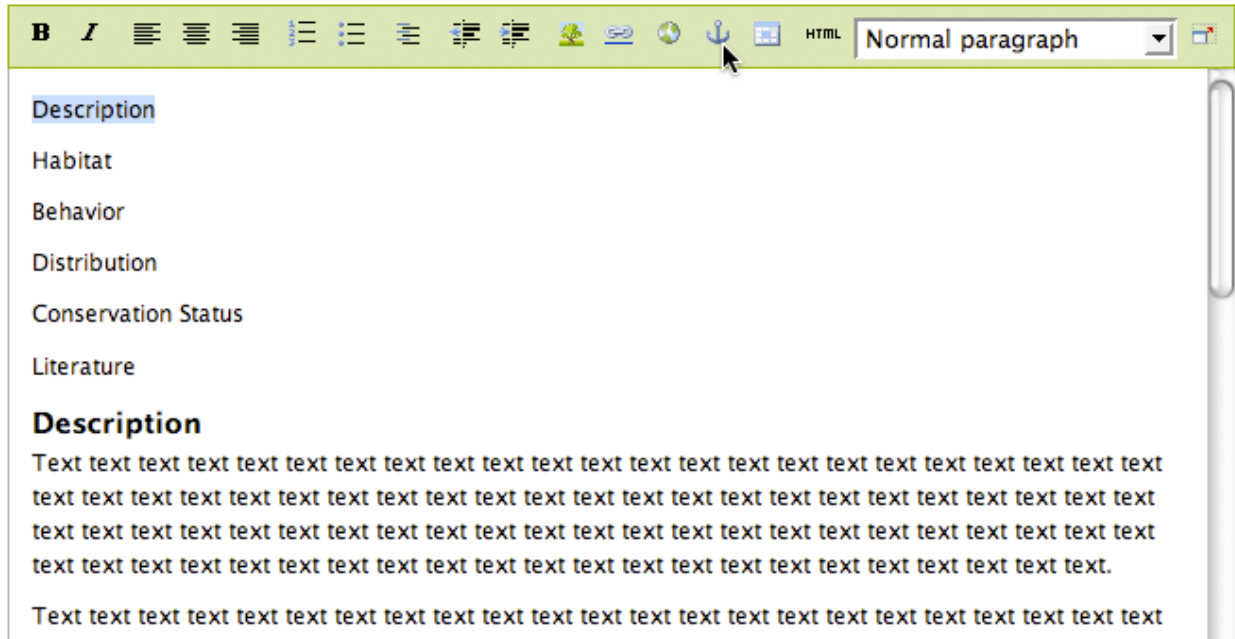
**listing:**

Thoroughbred Champions	Quarter Horse Champions
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

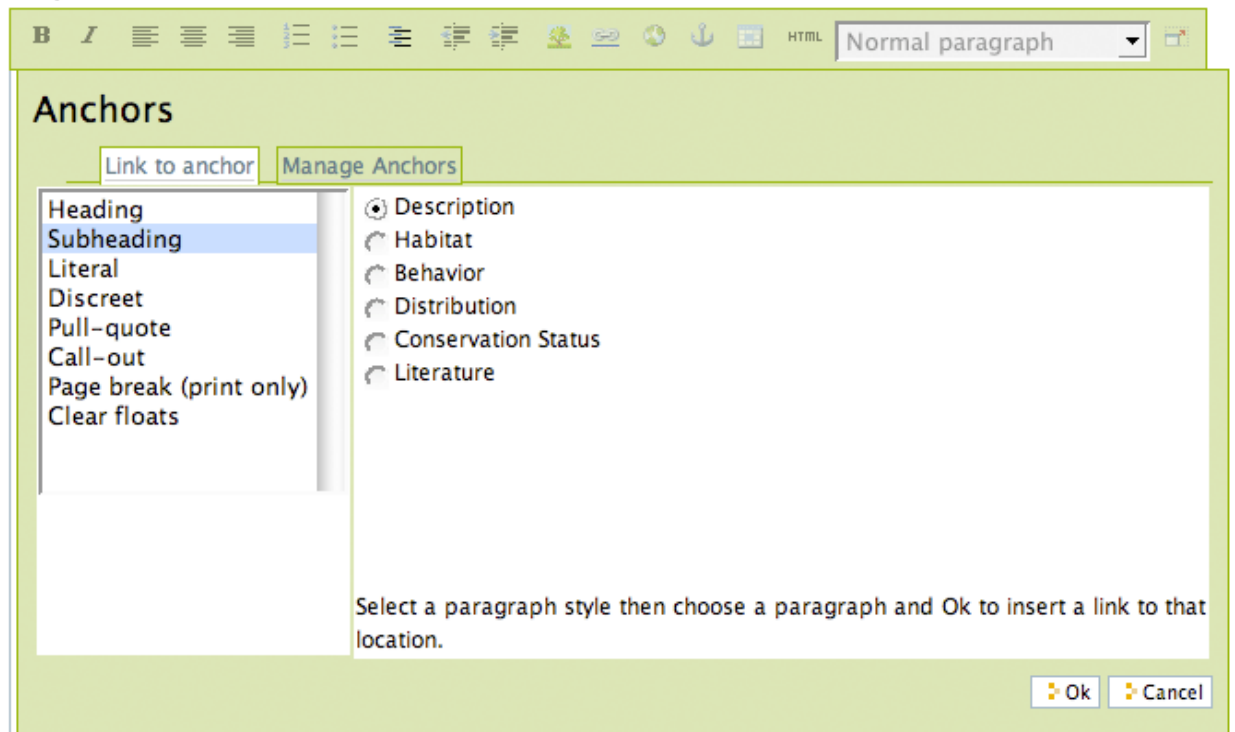
Dopo che la tabella è stata creata puoi fare click in una cella per far apparire i comandi necessari al ridimensionamento della tabella e le icone per aggiungere/eliminare righe e colonne:



Body Text



Body Text





## Table

Table Class

Rows

Columns

Create Headings

## Table

Table Class

Rows

Columns

Create Headings

Thoroughbred Champions Quarter Horse Champions	
Man O' War	First Down Dash
Secretariat	Dashing Folly
Citation	Special Leader
Kelso	Gold Coast Express
Count Fleet	Easy Jet

Nella tabella sopra, il cursore è stato posizionato nella cella “Special Leader”, esso attiva i quadratini di gestione intorno ai bordi per ridimensionare l’intera tabella. Attiva anche le icone aggiungere/eliminare per la cella corrente, la cella “Special Leader”. Cliccando sulla piccola x nel cerchio si elimina l’intera riga o colonna che contiene l’attuale cella. Cliccando le piccole icone a punta di freccia si aggiunge una riga sopra o al di sotto, o una colonna a sinistra o a destra della cella corrente.

### 1.5.6 Stile del Testo

L’impostazione dello stile del testo è fatta con un menu a tendina. Ecco le scelte:

Description	Example
Normal	text
Heading	<b>text</b>
Subheading	<b>text</b>
Literal	text
Discreet	text
Pull-quote	text
Call-out	text
Page break (print only)	
Clear floats (remove style)	
Highlight	text

Come in un normale editor di testi, seleziona una parola, una frase, o paragrafo con il mouse, quindi scegli una delle

opzioni di stile del menu a tendina e vedrai la modifica immediatamente.

## 1.5.7 Salvare

Fare click sul pulsante Salva in fondo e le modifiche della pagina saranno memorizzate.

---

## 1.5.8 Note a piè di pagina

### Linguaggi di mark-up

Se sei il tipo di persona che ama inserire il testo utilizzando i cosiddetti formati mark-up, è possibile disattivare l'editor visuale sotto le tue preferenze personali, e un pannello semplificato di inserimento testo andrà a sostituire Kupu. I formati mark-up disponibili in Plone sono:

- Markdown
- Textile
- Structured Text
- Restructured Text

Ognuno di questi funziona incorporando speciali codici di formattazione all'interno del testo. Ad esempio, con la formattazione Structured Text, circondando una parola o una frase da un asterisco doppio si otterrà quella parola o frase in grassetto, come in **Questo testo sarà in grassetto.** Vale la pena di imparare questi formati di mark-up per la velocità di inserimento se si creano molte pagine, o se si preferiscono approcci per l'inserimento di testo leggermente più tecnici. Alcune persone preferiscono questi formati non solo per la velocità in sé, ma per fluidità di espressione.

## 1.6 Collaborazione e flusso di lavoro

**Imparare a condividere e controllare l'accesso al contenuto utilizzando la Scheda Condivisione e il menu Stato.**

### 1.6.1 Stati di pubblicazione di base

**Il sistema di controllo della pubblicazione di Plone è molto flessibile, a partire dalle impostazioni di base per la creazione di un elemento privato o pubblico.**

Nell'angolo in alto a destra del riquadro di modifica di qualsiasi tipo di contenuto – cartelle, immagini, pagine, etc. e tutti i tipi di contenuto specializzati – c'è il menu per gestire lo stato di pubblicazione. Questo menu degli *stati* gestisce lo stato della pubblicazione:

L'intestazione del menu mostrerà lo stato di pubblicazione attuale del contenuto, ad esempio *Stato: Privato*, come mostrato sopra. Lo stato Privato è lo stato iniziale al momento della creazione di un elemento – un'immagine caricata, una pagina, una notizia – ed in questo stato, come indica il nome, l'elemento non sarà generalmente fruibile dai visitatori del sito web. Scegliendo dal menu lo stato *Pubblica*, il contenuto diverrà fruibile agli utenti anonimi del sito. L'opzione *Sottoporti per pubblicazione* viene utilizzata nei siti dove ci sono dei revisori di contenuti che devono approvare l'elemento per la pubblicazione, come descritto in seguito.

Inoltre, e questo sarà molto importante, certi tipi di contenuto, come ad esempio le notizie e gli eventi, non appariranno sul sito web come ti aspetteresti, fino a quando non verranno esplicitamente *pubblicati*.

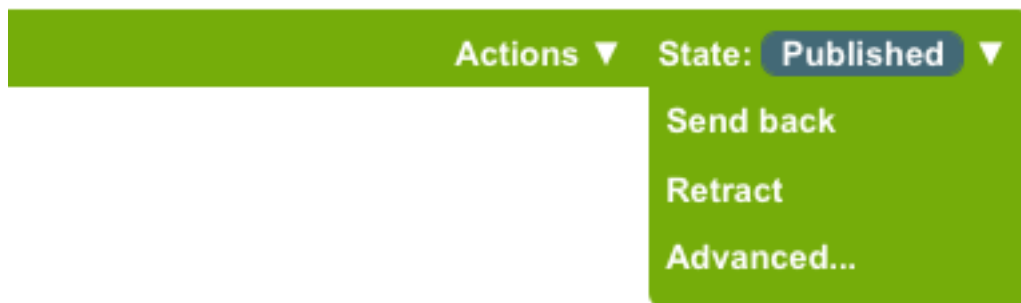
Imprimiti nella memoria che: **Lo stato di pubblicazione è importante!**



Lo stato di pubblicazione può essere modificato solo dagli utenti che dispongono delle autorizzazioni necessarie. Le scelte nel menu rispecchieranno le autorizzazioni possedute. Ad esempio, in un sito web di un grande giornale, un giornalista potrebbe aggiungere delle pagine come se fossero degli articoli, ma il menu di pubblicazione non mostrerà la scelta *Pubblica* ma solo *Sottoponi per pubblicazione*. Questo perchè il giornalista, prima della pubblicazione, deve inviare l'articolo alla redazione per l'approvazione. Se tuttavia il tuo account ha i permessi, la scelta *Pubblica* sarà disponibile e potrai semplicemente pubblicarlo in un solo passaggio.

Per un editore, un contenuto che è stato sottoposto per la pubblicazione può essere *pubblicato* o *revocato*, revocato nei casi in cui l'invio sia inappropriato rispetto alla situazione o per la ragione più comune ovvero che il contenuto ha bisogno di essere revisionato.

Dopo che un elemento è stato *pubblicato*, può essere *revocata la pubblicazione*, per reimpostare lo stato alla *bozza pubblica*, o *mandarlo indietro* allo stato privato. Le scelte del menu di pubblicazione cambieranno di conseguenza:



Occorre prendere in considerazione di revocare (“annullamento della pubblicazione”), o di rendere *privato*, qualsiasi contenuto che è diventato obsoleto o indesiderato per qualsiasi ragione. L'impostazione *privato* renderà l'elemento invisibile al pubblico e nei risultati delle ricerche, ma rimarrà nel caso in cui il formato o i contenuti (testo, immagini, etc.) servano in futuro. Questo è utile soprattutto per i contenuti relativi a eventi che possono ripresentarsi o a contenuti che fanno parte di una serie di contenuti simili. La decisione di cancellare un contenuto a semplicemente impostarlo come *privato* dipende dalla disponibilità di una copia del contenuto stesso sul proprio PC. Se il contenuto è di grandi dimensioni, nel senso di spazio su disco occupato, è probabilmente opportuno farsene una copia locale prima di cancellarlo, se lo spazio sul server è un problema.

## 1.6.2 Controllo avanzato

**Il sistema di controllo della pubblicazione, alla voce del menu avanzate, ha caratteristiche sofisticate per impostare la disponibilità per data e per contesto.**

Il menu *Stato* ha una voce *Avanzate...*:

che porta al pannello per la gestione *avanzata* della pubblicazione:



## Publishing process

An item's status (also called its review state) determines who can see it. Another way to control the visibility of an item is with its *Publishing Date*. An item is not publicly searchable before its publishing date. This will prevent the item from showing up in portlets and folder listings, although the item will still be available if accessed directly via its URL.

**Affected content**

	Title	Size	Modified	State
<input checked="" type="checkbox"/>	<b>Long-tailed Skipper</b>	1 kB	Apr 21, 2010 11:41 AM	Private

**Include contained items**  
If checked, this will attempt to modify the status of all content in any selected folders and their subfolders.

**Publishing Date**  
The date when the item will be published. If no date is selected the item will be published immediately.

/  /    :

**Expiration Date**  
The date when the item expires. This will automatically make the item invisible for others at the given date. If no date is chosen, it will never expire.

/  /    :

**Comments**  
Will be added to the publishing history. If multiple items are selected, this comment will be attached to all of them.

**Change State**  
Select the transition to be used for modifying the items state.

**No change**  
 **Submit for publication**  
 **Publish**

Di seguito una spiegazione dell'immagine: partendo dall'inizio del pannello c'è una casella che mostra il contenuto che sarà interessato dalla modifica dello stato di pubblicazione. Nel caso mostrato nella figura, la cartella "Long-tailed Skipper" sarà interessata da questo cambiamento.

Il campo successivo, *Includi gli elementi contenuti*, è una casella per controllare se il cambiamento di stato debba avere effetto solo sull'oggetto selezionato (la cartella "Long-tailed Skipper") o anche sugli elementi contenuti, includendo sia eventuali sottocartelle e relativi sottocontenuti sia altri tipi di elementi. Questa è una casella importante. Ti permette di cambiare semplicemente lo stato di un'intera sezione del sito internet. Per esempio, la cartella "Long-tailed Skipper" potrebbe contenere quattro sottocartelle: una per le fotografie, una per le occorrenze speciali, una per la tassonomia e una per delle descrizioni, tutte lasciate *private* durante il lavoro iniziale per la creazione della sezione. Tutte possono essere rese pubbliche – possono essere *pubblicate* – selezionando questo controllo e selezionando *Pubblica* in basso prima di salvare. Analogamente, la scelta *\*Sottoporti per pubblicazione\** può essere usata nei siti web dove gli editori revisionano cosa pubblicare.

Allo stesso modo, una intera sezione potrebbe essere immediatamente resa *privata*. Per esempio, se un'agenzia di noleggio auto ha deciso di rimuovere un modello di auto dalla sua flotta, un'intera sezione del loro sito web dedicato a questa vettura, con diverse sottocartelle piene di pagine, immagini e file, potrebbe essere impostata su *privato*.

I successivi due campi data sono per impostare la *data di pubblicazione* e la *data di scadenza*. Il loro significato è semplice. Se un elemento o un insieme di elementi devono essere pubblicati per un certo lasso di tempo, si possono impostare questi campi.

Puoi lasciare un commento con la spiegazione legato a tutti i contenuti interessati dal cambiamento di stato. Ciò è particolarmente utile quando più persone lavorano sullo stesso sito web: una persona che ha meno familiarità con un'area del sito web si chiederebbe il motivo per cui certi elementi non sono stati pubblicati. Si domanderebbero: "Questa informazione sembra buona. Perché non è già stata pubblicata?". In seguito però potrebbero leggere un commento che dice qualcosa del tipo: "Non pubblicare fino a che Richard non abbia fatto dei controlli su possibili problemi di copyright per quanto riguarda elementi descritti qui". L'uso dei commenti è molto utile per annotare informazioni sensibili, anche se si è l'unica persona che lavora sul sito web, in quanto si potrebbe dimenticare il motivo di una decisione presa sullo stato di pubblicazione.

Infine, nella parte finale, si può scegliere lo stato da applicare tra quelli disponibili per questa azione. Varierà, a seconda dello stato attuale dell'elemento. Ad esempio, se l'articolo è attualmente nello stato pubblicato, non vi sarà la scelta *pubblica*, se si trova nello stato *privato*, non sarà presente la scelta per renderlo *privato*, ecc. Se un elemento è già pubblicato, in questa parte inferiore del pannello saranno presenti le scelte per *revocare* e *mandare indietro*, che "annulleranno la pubblicazione" dell'elemento, reimpostandolo allo stato *bozza* o allo stato *privato*.

### 1.6.3 Politiche dei workflow

**Le politiche dei workflow consentono ad un amministratore del sito di creare un sistema formalizzato per controllare la pubblicazione e la gestione dei contenuti come in un flusso che, passo a passo, coinvolge utenti diversi con ruoli prestabiliti.**

I workflow sono un argomento avanzato. Implicano la creazione di un controllo più rigido nell'aggiunta, revisione e pubblicazione di contenuti. Se disponi di un account su un tipico sito Plone di piccole dimensioni, probabilmente non utilizzerai le politiche di workflow personalizzate perché non c'è bisogno di questo controllo sofisticato. Ma, la potenzialità di questo strumento è presente, in quanto è parte di Plone.

Per una introduzione al concetto di workflow, considera un esempio che coinvolge il sito web di un quotidiano, dove lavorano questi differenti gruppi di persone:

**Reporters** Possono creare articoli ma solo inviarli per essere revisionati.

**Redattori** Possono revisionare articoli, ma non possono pubblicarli direttamente. Mandano la revisione positiva e fanno avanzare il flusso alla successiva approvazione.

**Editori** Fanno il controllo finale, le correzioni, la revisione e possono pubblicare gli articoli.

Una *politica di workflow*, spesso abbreviata in *workflow*, descrive i vincoli che esistono per i diversi gruppi di persone durante i cambiamenti di stato dei contenuti. Una volta che la politica di workflow è stata creata, deve essere applicata ad un'area del sito web in modo tale che le regole abbiano effetto. Nell'esempio del sito del quotidiano, una politica di workflow dovrà essere impostata e applicata alla cartella dove i reporters aggiungono nuovi articoli. In seguito, i reporters vi potranno creare gli articoli ed inviarli per la revisione e l'approvazione:



I reporter dovrebbero aggiungere articoli e dovrebbero solo *sottoporli per la pubblicazione* (l'opzione *pubblica* del menu non è disponibile per loro). Allo stesso modo, i redattori possono *rifutare* l'articolo da revisionare o possono, a loro volta, *sottoporre* l'articolo agli editori per la correzione finale e la pubblicazione. Nell'esempio del sito web del quotidiano, questa politica potrebbe essere chiamata "Politica di revisione editoriale". Nella configurazione di una politica di workflow è fondamentale assegnare la stessa ad un'area del sito web – per definire il campo di applicazione del workflow. Questo è compito dell'amministratore del sito. L'amministratore deve usare il pannello di controllo di Plone per specificare dove la "Politica di revisione editoriale" si applica, se a livello globale o in una sottosezione.

Plone è dotato di diverse politiche di workflow utili - quella di default è un semplice politica di pubblicazione web. Il tuo amministratore del sito potrebbe impiegare una politica più specifica, ad esempio configurata per una comunità web o configurata per una Intranet aziendale (sistema web interno). In tal caso, potrebbe essere necessario imparare alcuni passi procedurali per la pubblicazione, ma queste sono solo varianti della politica base di default.

## 1.6.4 Collaborazione attraverso la condivisione

**La scheda Condivisione consente di collaborare con altri utenti attraverso l'uso di diversi ruoli integrati.**

### Esempio 1: Consentire ad altri di aggiungere contenuti in una cartella che hai creato

In questo esempio, Jane Smythe ha pieno accesso al suo sito Plone. Lei è in grado di aggiungere, modificare, cancellare e pubblicare contenuti in qualsiasi parte del sito. Per ora, ha creato una cartella denominata "Documentazione" e vi ha aggiunto una pagina, "Presentazione Progetto". Non ha pubblicato né la cartella né il documento. Il workflow di default per questo sito Plone non è stato modificato. Ora vuole dare il permesso al suo collega, George Shrubbs, di aggiungere contenuti alla cartella Documentazione. Lui ha il permesso di modificare qualsiasi contenuto esistente, ma lei ha bisogno che lui inizi ad aggiungere contenuti. Prima di proseguire insieme a Jane, diamo uno sguardo a quello che George vede quando si autentica in questo sito Plone:

Nota che in questo momento George non può nemmeno vedere la cartella Documentazione perché quando Jane l'ha creata, l'ha lasciata nello stato *Privato*. Tutte le autorizzazioni predefinite sono attualmente in atto e funzionano come previsto.

Jane conferisce a George le autorizzazioni necessarie per aggiungere contenuti alla cartella Documentazione.

Jane passa alla cartella Documentazione e fa clic sul tab Condivisione:

Una delle prime cose da notare è che Jane ha già tutte le autorizzazioni disponibili per questa cartella. Queste autorizzazioni erano in realtà state concesse in una sezione superiore del sito, come indicato dal simbolo di spunta verde.



George Shrubb ▼


 only in current section

- Home
- News
- Events
- Users
- 

- Contents
- View
- Edit
- Rules
- Sharing

## Sharing for “Documentation”

You can control who can view and edit your item using the list below.

Name ▲	Can add	Can edit	Can view	Can review
Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jayne Smythe (smythe)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



**Inherit permissions from higher levels**

By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol indicates an inherited value. Similarly, the symbol indicates a global role, which is managed by the site administrator.



Dando un'occhiata più da vicino, le autorizzazioni disponibili sono:

- **Può aggiungere** - Questo significa che quando questa autorizzazione viene concessa ad un particolare utente (o gruppo di utenti), egli può aggiungere nuovi contenuti. Dal momento che l'utente è stato anche il creatore di quel contenuto, sarà in grado di modificarlo a suo piacimento.
- **Può modificare** - Quando questa autorizzazione viene concessa per una cartella, l'utente può non solo modificare la Cartella (il titolo e la descrizione), ma può anche modificare uno qualsiasi degli elementi contenuti. Nota, tuttavia, che l'utente non è autorizzato ad eliminare il contenuto. Se questa autorizzazione viene concessa su una pagina, per esempio, l'utente può modificare solo quella pagina e nessuno degli altri elementi presenti nella cartella.
- **Può vedere** - Quando questa autorizzazione viene utilizzata su una cartella o un altro elemento, l'utente può visualizzare il contenuto ma non apportare modifiche.
- **Possono revisionare** - Quando questa autorizzazione viene concessa, l'utente può pubblicare i contenuti.

Nota: queste autorizzazioni sovrascrivono i permessi definiti nel workflow! Ad esempio, se si concede ad un utente l'autorizzazione "Può vedere" in una pagina che è nello stato privato, l'utente potrà vederla.

In questo esempio, Jane concede a George l'autorizzazione "Può aggiungere" nella Cartella "Documentazione" in modo che possa aggiungere contenuti in essa. Per fare questo, come primo passo, lo cerca utilizzando il suo nome:

**Sharing for "Documentation"**

You can control who can view and edit your item using the list below.

shrubb  ←

Name	Can add	Can edit	Can view	Can review
Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
George Shrub (shrubb)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jayne Smythe (smythe)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Inherit permissions from higher levels**

By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol  indicates an inherited value. Similarly, the symbol  indicates a global role, which is managed by the site administrator.

Jane ora può aggiungere le autorizzazioni necessarie a George per la cartella "Documentazione". Deve selezionare il permesso "Può aggiungere" e premere "Salva":

Questo è tutto ciò che deve fare! Vediamo come George vede il sito ora.

Nota: George NON ha bisogno di disconnettersi e riconnettersi. Le autorizzazioni sono sempre aggiornate perché sono controllate ogni volta che un utente accede a qualsiasi cosa (ad esempio cliccando su un link) su un sito Plone.

George clicca sulla scheda *Home* (per esempio) per aggiornare la sua visione del sito e visualizzerà la cartella "Documentazione":

Contents View Edit Rules **Sharing**

## Sharing for “Documentation”


You can control who can view and edit your item using the list below.

shrub

Name	Can add	Can edit	Can view	Can review
Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
George Shrubb (shrub)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jayne Smythe (smythe)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Inherit permissions from higher levels**


By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol indicates an inherited value. Similarly, the symbol indicates a global role, which is managed by the site administrator.



George Shrubb ▾

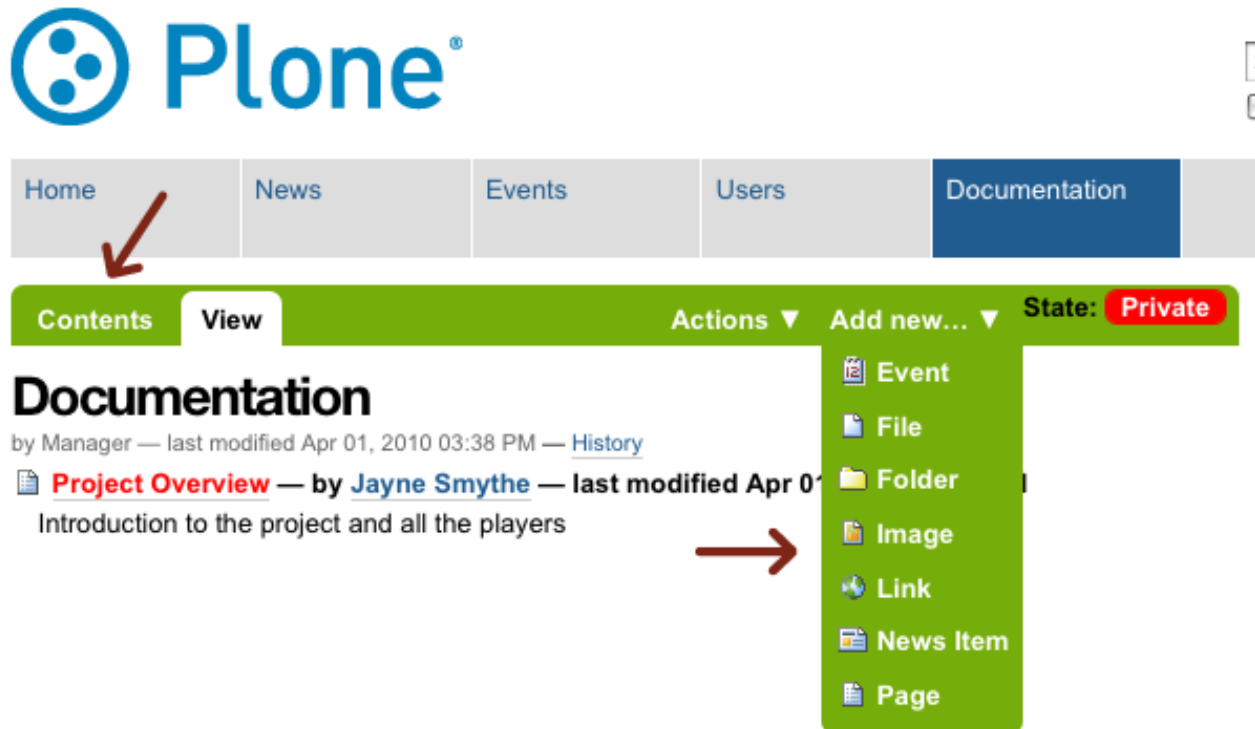
Search Site

only in current section



Home	News	Events	Users	Documentation	
------	------	--------	-------	---------------	--

Quando George fa click sulla scheda “Documentazione”, si accorge che può visualizzare tutto il contenuto e che è ora in grado di aggiungere i tipi di contenuto disponibili nella cartella, come mostrato nel menu *Aggiungi...*:



George vuole visionare ciò che Jane ha già creato, quindi seleziona il link Project Overview e vede:



Anche se George può visualizzare il documento, le sue autorizzazioni limitate non gli consentono di modificarlo o di cambiare il suo stato. L'unica cosa che può fare, al di là di visualizzare il documento, è di farne una sua copia.

George aggiunge una pagina intitolata “Widget Installation” e ne crea il contenuto. Quando ha terminato, la salva:

Jane vede il lavoro fatto da George. Seleziona la scheda “Documentazione” e vede che George si è dato da fare. Clicca



Home	News	Events	Users	Documentation
------	------	--------	-------	---------------

View Edit Sharing Actions ▼ State: **Private** ▼

## Widget Installation

by George Shrubbs — last modified Apr 01, 2010 03:46 PM — [History](#)

### How to install your widget

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Aliquam tincidunt mauris eu risus.
- Vestibulum auctor dapibus neque.
- Nunc dignissim risus id metus.
- Cras ornare tristique elit.

sulla pagina “Widget Installation” per dare un’occhiata più da vicino:

Si noti che Jane ha pieno accesso alla pagina che ha creato George. Lei può modificarla così come tagliarla/copiarla/incollarla. In realtà lei attenderà che George invii la pagina per la revisione prima di modificarla eventualmente.

### Esempio 2: Permettere ad altri di modificare contenuti creati da te

Sia Jane sia George hanno lavorato duramente per creare le pagine della cartella Documentazione. **Jane ha pubblicato la cartella e diverse pagine:**

Jane ha deciso che vuole consegnare a George i permessi di modifica (ma non di pubblicazione) per tutti gli elementi cartella “Documentazione”. Per fare questo Jane deve tornare nella cartella “Documentazione” e cliccare sulla scheda *Condivisione*:

Da qui deve solo selezionare la casella di controllo “Può modificare” e George sarà in grado di modificare tutto il contenuto nella cartella “Documentazione” – compresa la cartella “Documentazione” stessa. Quando successivamente George visiterà la cartella e cliccherà su “Presentazione del progetto” (che è una pagina che Jane ha creato), questo è quello che vedrà:

Ora George può modificare qualsiasi elemento nella cartella “Documentazione”, indipendentemente da chi lo ha creato o da quando è stato creato.

Nel frattempo, Molly si è unita a George come nuovo membro del team. Molly aiuterà George nell’aggiornamento del documento “Widget Installation”. George va nella scheda condivisione dell’elemento “Widget Installation”, cerca il nome completo di Molly (non il nome utente) e seleziona “Può modificare” per darle l’autorizzazione su questo documento.

Quando Molly entrerà nella cartella “Documentazione”, potrà vedere i due articoli pubblicati e l’elemento privato che ora è autorizzata a modificare:



## Widget Installation

by George Shrubbs — last modified Apr 01, 2010 03:46 PM — [History](#)

### How to install your widget

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Aliquam tincidunt mauris eu risus.
- Vestibulum auctor dapibus neque.
- Nunc dignissim risus id metus.
- Cras ornare tristique elit.

Cut  
Copy  
Delete  
Rename



## Documentation

by Manager — last modified Apr 01, 2010 03:38 PM — [History](#)

[Project Overview](#) — by [Jayne Smythe](#) — last modified Apr 21, 2010 02:13 PM

Introduction to the project and all the players

[Widget Installation](#) — by [George Shrubbs](#) — last modified Apr 21, 2010 02:09 PM

[Customer Support](#) — by [Jayne Smythe](#) — last modified Apr 21, 2010 02:11 PM



[FAQs](#) — by [Jayne Smythe](#) — last modified Apr 21, 2010 02:12 PM

Contents View Edit Rules **Sharing**


## Sharing for “Documentation”

You can control who can view and edit your item using the list below.

Search for user or group

Name	Can add	Can edit	Can view	Can review
 Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 George Shrubbs (shrubb)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 Jayne Smythe (smythe)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

### Inherit permissions from higher levels

By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol  indicates an inherited value. Similarly, the symbol  indicates a global role, which is managed by the site administrator.

Contents
View
Edit
Rules
Sharing

## Sharing for “Documentation”

You can control who can view and edit your item using the list below.

Name	Can add	Can edit	Can view	Can review
👤 Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
👤 George Shrubbs (shrubbs)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
👤 Jayne Smythe (smythe)	✔	✔	✔	✔

✔ **Inherit permissions from higher levels**

By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol ✔ indicates an inherited value. Similarly, the symbol 🌐 indicates a global role, which is managed by the site administrator.

E, infatti, quando farà un click sul documento “Widget Installation”, sarà in grado di modificarlo:

Si noti, tuttavia, che quando Molly selezionerà uno dei due elementi dove non ha il permesso di modifica, non avrà alcun ulteriore accesso. Può visualizzare questi due elementi perché sono pubblicati come definito nel workflow di default di Plone (il che significa che chiunque può vederli).

Una nota finale su questo esempio: se la cartella “Documentazione” non fosse stata nello stato di pubblicazione O Molly non avesse avuto delle autorizzazioni particolari (per esempio, “Può visualizzare” nella cartella Documentazione), Molly avrebbe avuto bisogno dell’URL completo per raggiungere il documento a cui le era stato dato l’accesso per la modifica. Le autorizzazioni sono molto specifiche in Plone!

## 1.7 Utilizzo delle collezioni

Le collezioni sfruttano l’intelligenza di Plone.

### 1.7.1 Introduzione alle Collezioni

**Una Collezione in Plone funziona come un report o una query fatta in un database. Utilizza le Collezioni per ordinare e visualizzare in modo dinamico il tuo contenuto.**

Una **Collezione** in Plone funziona come un report o una query fatta in un database. L’idea è di utilizzare una collezione per cercare nel tuo sito web in base ad un insieme di **Criteri** quali: il tipo di contenuto (pagina, notizia, immagine), la data di pubblicazione o parole chiave contenute nel titolo, nella descrizione o nel corpo.

Contents View Edit Rules **Sharing**

## Sharing for “Documentation”

You can control who can view and edit your item using the list below.

Name	Can add	Can edit	Can view	Can review
Logged-in users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
George Shrub (shrub)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jayne Smythe (smythe)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Molly Jones (jones)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



**Inherit permissions from higher levels**

By default, permissions from the container of this item are inherited. If you disable this, only the explicitly defined sharing permissions will be valid. In the overview, the symbol  indicates an inherited value. Similarly, the symbol  indicates a global role, which is managed by the site administrator.

Molly Jones ▼



only in current section

Home News Events Users **Documentation**

## Documentation

by Manager — last modified Apr 21, 2010 03:14 PM — [History](#)

[Project Overview](#) — by [Jayne Smythe](#) — last modified Apr 21, 2010 02:13 PM  
 Introduction to the project and all the players

[Widget Installation](#) — by [George Shrub](#) — last modified Apr 21, 2010 02:09 PM

[FAQs](#) — by [Jayne Smythe](#) — last modified Apr 21, 2010 02:12 PM





Molly Jones ▼

Search Site

Search

 only in current section

Home

News

Events

Users

Documentation

View

Edit

Sharing

Actions ▼

State:

Private ▼

## Widget Installation

by George Shrubbs — last modified Apr 21, 2010 03:16 PM — [History](#)

### How to install the Widget

- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Aliquam tincidunt mauris eu risus.
- Vestibulum auctor dapibus neque.
- Nunc dignissim risus id metus.
- Cras ornare tristique elit.

Diciamo che sul tuo sito web hai un ampio catalogo di foto e mappe. Si possono facilmente visualizzare tutte in una sola volta creando un collegamento ipertestuale alla cartella dove sono archiviate. Potresti persino creare collegamenti differenti per differente sotto-cartelle, se hai organizzato le cose in questo modo. Tuttavia, se le immagini e le mappe fossero inserite in più cartelle sparse nel sito, questa operazione potrebbe diventare macchinosa. Inoltre, non c'è modo con le cartelle normali di visualizzare contenuti diversi, provenienti da diverse parti del tuo sito, basandosi su criteri come:

- parole chiave nel titolo
- data di creazione
- autore
- tipo di contenuto

La necessità di visualizzazione i contenuti in una varietà di modi dinamici viene soddisfatta dalle **Collezioni** (precedentemente note come **Smart Folders** o **Rich Topic** nelle versioni più vecchie di Plone). Le Collezioni di fatto non contengono elementi come accade in una cartella. Al contrario sono i **Criteri** stabiliti che determinano quali contenuti far apparire nella pagina dove è definita la Collezione.

I casi più comuni nei quali viene utilizzata una Collezione sono:

- Archivio di Notizie
- Archivio di Eventi
- Visualizzazione di Foto dato un intervallo di date
- Visualizzazione di contenuti data una parola chiave



Molly Jones ▼

Search Site

only in current section

Home

News

Events

Users

Documentation

## Project Overview

by Jayne Smythe — last modified Apr 21, 2010 03:30 PM — [History](#)

### Introduction to the project and all the players

#### Lorem ipsum dolor sit amet

consectetuer adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede.

#### Donec nec justo eget felis facilisis fermentum.

Aliquam porttitor mauris sit amet orci. Aenean dignissim pellentesque felis.

#### Morbi in sem quis dui placerat ornare.

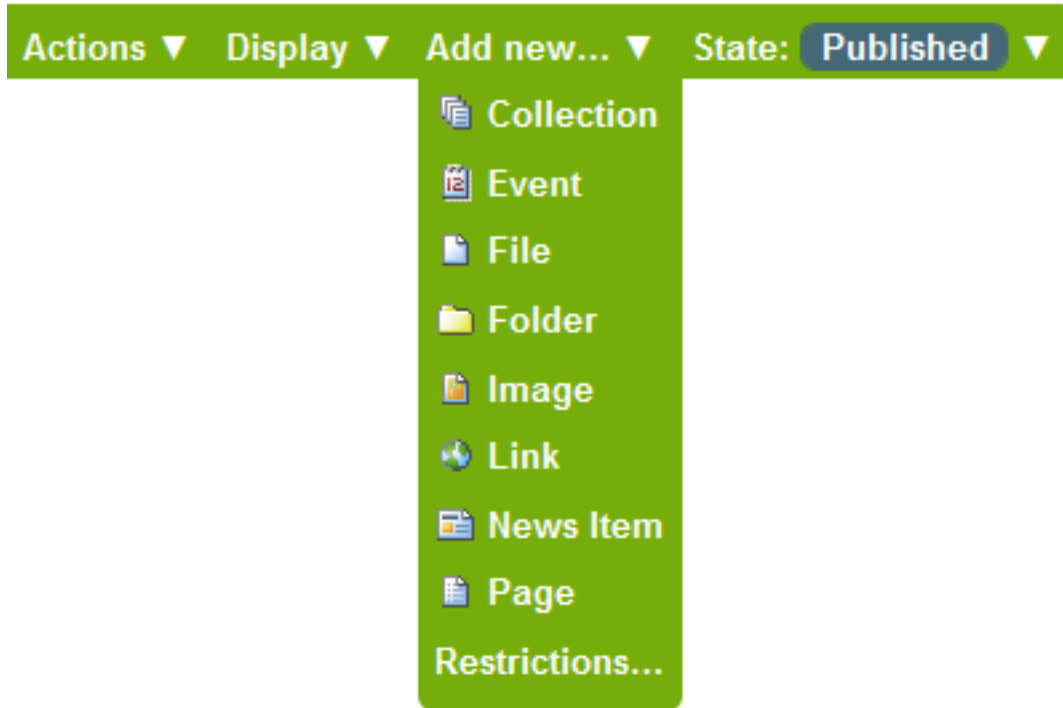
- Pellentesque odio nisi, euismod in, pharetra a, ultricies in, diam.
- Sed arcu. Cras consequat.

## 1.7.2 Aggiungere Collezioni

Le Collezioni (una volta chiamate Smart Folders) sono contenitori virtuali con liste di elementi trovati utilizzando ricerche specifiche.

Comprendere che i contenuti possono essere memorizzati ovunque in un sito Plone ma possono essere recuperati con collezioni personalizzate che creano “viste” sui contenuti stessi, è un passo importante per poter utilizzare Plone in modo efficace. E’ un sistema intelligente.

Per aggiungere una collezione, utilizza il menu “Aggiungi...”, nello stesso modo in cui aggiungi altri tipi di contenuto:



Ti comparirà il riquadro per aggiungere una *Collezione*:

Sotto i campi titolo e descrizione ci sono un insieme di campi per specificare il formato dei risultati restituiti dal criterio di ricerca della nuova collezione. I quattro campi nel riquadro sopra sono a coppie. I primi due in alto consentono di limitare i risultati della ricerca a un certo numero di elementi. Gli ultimi due consentono di controllare l’ordinamento dei risultati.

### Impostare il criterio di ricerca

Dopo aver impostato le configurazioni di visualizzazione nel riquadro di modifica sopra indicato, fai click sulla scheda criteri per visualizzare il pannello per impostare i criteri di ricerca:

L’area superiore del pannello, *Aggiungi nuovo Criterio*, consente di impostare un campo e un criterio di corrispondenza. L’area inferiore, *Ordinamento*, permette semplicemente di selezionare un campo per l’ordinamento:

I tipi di criteri per la ricerca dipendono da quale campo viene selezionato.

Dopo aver salvato la collezione, i criteri di ricerca saranno applicati ed il risultato mostrato quando la collezione viene selezionata. È possibile creare un numero qualsiasi di collezioni per ogni visualizzazione che si vuole personalizzare.

## Add Collection

An automatically updated stored search that can be used to display items matching criteria you specify.

Default Categorization Dates Ownership Settings

### Title ■ (Required)

Title is required, please correct.

### Description

Used in item listings and search results.

### Body Text

Style...

**B** *I*

**Limit Search Results**

If selected, only the 'Number of Items' indicated below will be displayed.

### Number of Items

**Display as Table**

Columns in the table are controlled by 'Table Columns' below.

### Table Columns

Select which fields to display when 'Display as Table' is checked.

Creation Date	>>	Title
Creator	<<	
Description		
Effective Date		
End Date		
Expiration Date		

Save Cancel

Contents View Edit **Criteria** Sharing

## Criteria for News and Events

No criteria defined yet. The search will not show any results. Please add criteria below.

—Add New Search Criteria—

### Field name

The time and date an item was created

Creation Date ▼

- Creation Date
- Creator
- Description
- Effective Date
- End Date
- Expiration Date
- Item Type
- Location
- Modification Date
- Related To
- Search Text
- Short Name
- Start Date
- State
- Tags
- Title
- Title

Save

Contents View Edit **Criteria** Sharing

## Criteria for News and Events

No criteria defined yet. The search will not show any results. Please add criteria below.

---

—Add New Search Criteria —

### Field name

The time and date an item was created

Creation Date ▾

### Criteria type

Criteria does match

Relative date ▾

Relative date  
Date range

---

—Set Sort Order —

### Field name

List Available Fields

No sort order ▾

### Reverse

Reverse display order

Save

Per l'esempio delle farfalle sopra descritto, oltre ad un vincolo sulla data per trovare gli elementi recenti, il campo categorie potrebbe essere utilizzato per abbinare il colore alle farfalle e ottenere una serie di collezioni di "Farfalle blu", "Farfalle Bianche," ecc.

Criteri multipli possono essere utilizzati nella stessa collezione. Per esempio, una collezione chiamata "Farfalle fotografate nel mese scorso" potrebbe essere creata impostando un criterio relativo alla data di creazione ed uno relativo al tipo di elemento, che dovrà essere un'immagine. Le collezioni con criteri basati sulle date sono veramente efficaci per mostrare viste di contenuti sempre aggiornate, che non richiedono alcun lavoro da parte dell'amministratore del sito - una volta che la collezione è stata creata, essa mostrerà automaticamente i contenuti più recenti.

*Nota:* Una collezione non si comporta come una normale cartella; non puoi aggiungere elementi tramite la voce del menu aggiungi, come è possibile fare in una normale cartella.

### 1.7.3 Regolazione delle Impostazioni di Visualizzazione

**Scopri come le impostazioni di visualizzazione possono modificare l'aspetto della pagina Collezione.**

Mentre il punto di forza delle Collezioni sta nei Criteri, le impostazioni di visualizzazione possono fare una grande differenza nel modo in cui la vostra Collezione sarà mostrata. Tutte e tre le impostazioni che tratteremo in questa sezione possono essere trovate facendo click sulla scheda **Modifica** di una Collezione.

#### Eredita Criteri

Selezionando l'opzione **Eredita Criteri**, la Collezione erediterà i Criteri da una Collezione padre. Questo è utile solo quando si utilizzano Collezioni Subordinate. Se questa opzione è selezionata, è possibile creare un'altra Collezione che è più specifica rispetto alla Collezione Padre, pur mantenendo i Criteri di base della Collezione Padre. Un semplice esempio potrebbe essere una Collezione Padre per la visualizzazione di tutti gli Eventi in un sito, e una Collezione Subordinata che visualizza Eventi (ereditando i Criteri) *ma solo* gli Eventi con una particolare parola chiave.

#### Limita i Risultati della Ricerca

Possiamo usare Limita i Risultati della Ricerca per limitare il numero di risultati che verranno visualizzati *per pagina*. In questo modo, se abbiamo una Collezione che sta mostrando Notizie, siamo in grado di limitare i risultati a cinque o dieci, invece di mostrare tutte le Notizie in un singolo elenco di grandi dimensioni.

#### Visualizza come Tabella

Visualizza come Tabella è semplicemente un altro modo per visualizzare i risultati di una Collezione. Invece di avere una Collezione che mostra i risultati in una lista, possiamo **generare una tabella** con i risultati, e impostare esattamente le informazioni che desideriamo visualizzare nel risultato. Si può personalizzare la tabella selezionando le **Colonne della Tabella** nell'elenco a sinistra e facendo clic sul pulsante freccia destra per spostarle nell'elenco a destra. Nell'esempio precedente abbiamo scelto di includere il Titolo dell'oggetto, i suoi Creatori e la Data di Accessibilità. È possibile utilizzare qualsiasi numero di colonne, o tutte se lo si vuole.

Quando si considera cosa scegliere, bisogna tenere presente che non tutti gli oggetti disporranno delle informazioni per ogni colonna disponibile. Per esempio, la **Data di Inizio** e la **Data di Fine** valgono solo per gli Eventi. Pertanto, se si aggiungono queste colonne e la tabella include Pagine oltre agli Eventi allora le righe per le pagine non avrebbero le date di inizio e di fine popolate. L'altra cosa da considerare è che più colonne stai mostrando, più affollata diventerà la tabella. La migliore regola è quella di visualizzare solo ciò che è assolutamente necessario.

Alcune note ulteriori sulla selezione delle colonne: è possibile selezionarne più di una alla volta tenendo premuto il tasto (Ctrl) mentre si fa click. Se si desidera rimuovere una colonna, selezionarla a destra e fare clic sul pulsante freccia sinistra. Inoltre è possibile aggiungere e rimuovere le colonne con un doppio click sul loro nome.

### 1.7.4 Definizione dei Criteri

**Definizioni ed esempi dei vari campi criteri disponibili**

Il potere delle Collezioni dipende certamente dai criteri che si possono impostare per esse. Imparare come utilizzare i diversi Criteri vi permetterà di creare Collezioni molto utili. In questa sezione, useremo esempi per illustrare i molti modi di utilizzare i Criteri.

### Categorie

Il criterio Categoria consente di ricercare il **campo Categoria** degli oggetti. Perché funzioni, è necessario aver specificato prima le Categorie per i contenuti (questo è fatto tramite la scheda Categorizzazione sugli oggetti contenuto). Un esempio di come sia possibile utilizzare questo campo è creare una Collezione che riporti tutti gli oggetti relativi alla Categoria *Organizzazione*. Siamo in grado di selezionare il valore *Organizzazione* per il nostro criterio. Quindi, salvando questo criterio e visualizzando la nostra Collezione, i risultati saranno tutti gli oggetti di contenuto che avevamo marcato con la Categoria *Organizzazione*.

Ancora una volta, i valori disponibili sono completamente dipendenti da ciò che abbiamo specificato sui nostri oggetti nella scheda Categorizzazione.

### Creatore

Quando utilizziamo il criterio Creatore, stiamo **creando un filtro sugli oggetti basato su chi li ha creati**. Ciò potrebbe essere utile se si vuole creare una sezione autore, in cui si desidera visualizzare solo i contenuti sul tuo sito che sono stati creati da un certo autore.

Abbiamo diverse opzioni per questo tipo di criterio. Essi ci permettono di limitare il creatore alla persona attualmente connessa, immettere manualmente il nome di un altro utente, oppure selezionare gli utenti da un elenco.

Se si vogliono visualizzare i risultati di più utenti, è necessario utilizzare l'opzione **Elenco dei Valori**. In caso contrario, si usa normalmente l'opzione Testo, a meno che il creatore che si vuole selezionare non siate voi stesso, nel qual caso si utilizza l'opzione Limita a Utente Attuale.

### Descrizione

Il campo Descrizione è essenzialmente un criterio che funziona come le ricerche testuali sul sito, fatte con un campo di tipo **search box** dove immettere i termini da cercare. Tuttavia, invece di cercare nel titolo e nel corpo di una pagina, la collezione **effettuerà una ricerca solo per il testo nel campo Descrizione** di un contenuto. Questo criterio è realmente utile solo se si compila il campo Descrizione in modo coerente per tutti i tipi di contenuto.

### Posizione

Utilizzare il criterio Posizione è molto simile a specificare una posizione nella ricerca di un documento sul disco rigido del tuo PC. Specificando un criterio Posizione, **i risultati che vengono visualizzati nella tua Collezione provengono solo da quella posizione**, tipicamente una Cartella. Ciò può essere utile se si desidera visualizzare solo il contenuto che si trova nella sezione Chi siamo del sito, per esempio. Il criterio Posizione è anche utile per restringere risultati delle Collezioni quando è combinato con altri criteri.

Per specificare una Posizione, è sufficiente fare click sul pulsante **Aggiungi**, si apre quindi una nuova finestra che mostra una directory del tuo sito. Se seguiamo il nostro esempio e vogliamo cercare la sezione Chi siamo del nostro sito, dobbiamo fare click sul pulsante Inserisci accanto alla cartella Chi siamo.

È possibile aprire le cartelle per visualizzare i contenuti in essi presenti sia facendo clic sul pulsante Sfoglia, sia facendo click direttamente sul titolo della cartella che si desidera aprire. È inoltre possibile utilizzare la casella di ricerca per cercare il Titolo di un oggetto.



## Testo Ricercabile

Il Testo Ricercabile è un criterio molto utile. È simile al box di ricerca sul tuo sito o un motore di ricerca Internet. Prende il testo che hai indicato, cerca il Titolo, la Descrizione, e il Corpo di tutti gli oggetti e restituisce **quelli che hanno la parola o la frase specificata**. Ciò è utile quando si desidera trovare oggetti che hanno a che fare con una certa cosa, soprattutto se la parola o la frase appare in molti tipi di contenuto. Utilizzando LearnPlone.Org come esempio, se voglio creare una Collezione che consente di visualizzare tutti gli oggetti che fanno riferimento alla parola Collezioni, lo devo fare utilizzando il criterio Testo Ricercabile, specificando *collezioni* come valore del criterio. Tutti i Tutorial, Video, voci di Glossario, ecc con *Collezioni* nel Titolo, nella Descrizione, o nel Corpo del Testo dovrebbero quindi comparire come risultati delle Collezione.

## Contenuti Correlati

Il campo Contenuti Correlati è un altro campo, come Categoria, che **deve essere specificato su un oggetto contenuto prima di essere utilizzato per una Collezione**. Il campo Contenuti Correlati su un oggetto consente di specificare quali altri oggetti nel tuo sito sono simili o sono rilevanti per l'oggetto creato. Specificando questo campo quando si crea un oggetto, è possibile creare una rete di contenuti correlati che faranno riferimento a vicenda (si pensi a un tipo di funzione "vedi anche"). Una volta fatto questo, è possibile utilizzare il criterio Contenuti Correlati in una Collezione, per visualizzare tutto ciò che è collegato a un specifico oggetto.

Ad esempio, se abbiamo creato contenuti che hanno come Contenuti correlati le pagine Nostro Staff, Storia, e la Homepage Chi siamo, possiamo selezionare questi valori per il criterio della Collezione, e la nostra collezione visualizzerà tutti i contenuti che hanno quei valori come contenuti correlati.

Se avessimo scelto la pagina Storia come valore dell'opzione del criterio Contenuti correlati, la nostra Collezione mostrerebbe tutto ciò che è legato alla pagina Storia.

Tenete a mente che, se ad esempio scelgo la pagina Storia come valore dell'opzione del criterio Contenuti correlati, la collezione non tirerà fuori tutti i contenuti impostati come Contenuti Correlati della pagina Storia, ma tutti i contenuti che hanno la Pagina Storia come Contenuto correlato.

## Stato

Utilizzare il criterio Stato è molto semplice. Ci permette di **fare una selezione in base allo stato pubblicato o privato**. È una buona idea limitare le Collezioni agli elementi visibili pubblicamente **impostando il filtro sullo stato pubblicato**, in modo che i contenuti privati non appaiano nei risultati della Collezione. Può essere utile anche impostare il filtro sullo stato Privato. Per esempio, un amministratore del sito potrebbe desiderare di vedere rapidamente i contenuti privati, in modo da poter determinare quale lavoro deve essere ancora fatto e che cosa potrebbe essere cancellato.

## Date

Avrete notato che sono **disponibili parecchie date** da utilizzare come Criteri. Poiché ci sono un grande numero di date, esse avranno una propria sezione nel manuale.

### 1.7.5 Impostazione del criterio di Ordinamento

**Scopri come utilizzare la funzione di Ordinamento per personalizzare l'ordine in cui i risultati vengono visualizzati**

L'Ordinamento **determina l'ordine dei risultati della Collezione**. L'Ordinamento consente di ordinare su tre principali categorie: testo, proprietà degli oggetti, e date. Quando si ordina in base al testo, gli oggetti saranno ordinati in ordine alfabetico. Quando si ordina in base alle proprietà degli oggetti, stiamo effettivamente raggruppando gli oggetti

attraverso le proprietà specificate. Quando si ordina per data i risultati saranno visualizzati con la data più recente per prima (anche se ci sono molte 'date' in Plone). Tutti gli Ordinamenti sono in Ordine Crescente a meno che il checkbox Ordine Inverso sia selezionato. Selezionandolo è possibile visualizzare in ordine inverso, o visualizzare prima le date più recenti, ecc.

### Date

Ci sono numerose opzioni Data che saranno descritte nella prossima sezione del manuale.

### Proprietà degli oggetti

#### Tipo

Quando si ordina per Tipo, si ottiene una Collezione che presenta i risultati raggruppati per Tipo. Possiamo utilizzare questo Ordinamento se abbiamo una Collezione che deve restituire molti Tipi diversi di elementi. In questo modo possiamo rendere la Collezione molto facile da navigare per il visitatore del sito.

#### Stato

L'Ordinamento per Stato visualizzerà i risultati raggruppandoli per lo stato di pubblicazione. Dal momento che ci sono solo due Stati nella configurazione di default di Plone, ci saranno solo le voci Pubblicato e Privato. Possiamo usare questo Ordinamento per separare tutte le pagine del nostro sito e vedere facilmente quello che è pubblico (Pubblicato) e ciò che si nasconde agli occhi del pubblico (Privato).

#### Categoria

L'Ordinamento Categoria è utile quando si desidera visualizzare gli oggetti del nostro sito raggruppati in base alla Categoria nella quale li abbiamo posti. Tenete a mente che è necessario aver specificato la Categoria sulla maggior parte degli oggetti perché l'ordinamento per Categoria sia utile. Se non avete specificato alcuna Categoria, l'ordinamento per categorie non farà nulla.

#### Correlato con

L'Ordinamento Correlato applica di fatto un criterio alla tua Collezione. Esso, infatti, limita i risultati unicamente ai contenuti quelli che hanno l'informazione Correlato con specificata nelle loro proprietà.

### Testo

#### Nome breve

L'Ordinamento per il Nome Breve restituisce i risultati in ordine alfabetico. Di default Plone imposta il Titolo come Nome Breve di un oggetto. La differenza tra i due è che il Nome breve è tutto in minuscolo e ha trattini tra tutte le parole. Per esempio il Nome breve per la pagina dal titolo Chi siamo è *chi-siamo*. Il Nome breve è quello che Plone utilizza anche nell'URL della pagina ([www.myploonesite.org/chi-siamo](http://www.myploonesite.org/chi-siamo)). È possibile specificare un diverso Nome Breve per un oggetto utilizzando il pulsante Rinomina nella scheda Contenuti.

#### Creatore

L'Ordinamento Creatore raggrupperà tutti i risultati in ordine alfabetico sul loro autore. Per esempio, diciamo che abbiamo diversi documenti pubblicati da Bob Baker e molti altri documenti pubblicati da Jane Smith. L'Ordinamento Creatore si tradurrebbe in tutti i documenti creati da Bob Baker elencati per primi, seguiti da quelli di Jane Smith.

#### Titolo

L'Ordinamento per Titolo visualizzerà i risultati in ordine alfabetico, sui Titoli.

Nella prossima sezione tratteremo le date che abbiamo saltato in questa sezione e in quella sui Criteri.

## 1.7.6 Uso e comprensione delle Date

### Spiegazione delle Date associate alle Collezioni ed il loro uso

Ci sono diversi tipi di date che possiamo scegliere, molti di essi sembrano simili. Per questo motivo è molto facile confondersi su quale data utilizzare. Di seguito è definita ogni opzione data.

### Definizione delle Date

#### Data di Creazione

La Data di Creazione è la data in cui è stato fatto il documento. Si può pensare questa data come il suo compleanno, il giorno in cui è nato. Non è possibile modificare la Data di Creazione di un oggetto.

#### Data di Accessibilità

La Data di Accessibilità è la data in cui un oggetto viene pubblicato. Questo data è personalizzabile attraverso il tab **Modifica** presente sul tab **Data** di un contenuto. Tuttavia, in quella scheda essa è indicata come Data di Pubblicazione (un discrepanza nella nomenclatura di Plone).

La **Data di Creazione** e la **Data di Accessibilità** sono molto simili. Entrambe rappresentano il punto di inizio di un oggetto. Un importante punto da tenere a mente quando si sceglie la data da utilizzare, è che un oggetto può essere creato molto prima che diventi pubblico. Una pagina potrebbe venire modificata per diverse settimane prima che sia effettivamente pubblicata. Quindi si dovrebbero avere risultati diversi in un Collezione a seconda di quale data è stato utilizzata. Si consiglia di utilizzare la **Data di Accessibilità**, invece della Data di Creazione per Collezioni basate sulle date. In questo modo la tua Collezione mostra i risultati sulla base di quando sono diventati visibili, il che è più rilevante per il pubblico della tua collezione. Inoltre, è possibile modificare la Data di Accessibilità per controllare l'ordinamento, cosa che non si può fare con la Data di Creazione.

#### Data di Scadenza

La Data di Scadenza si riferisce al giorno in cui il contenuto non sarà più pubblico. Questa data è anche personalizzabile attraverso il tab Modifica (indicata sopra), come la Data di Accessibilità. Per impostazione predefinita, gli oggetti non hanno la Data di Scadenza.

#### Data di Modifica

La Data di Modifica è la data dell'ultima modifica fatta sull'oggetto. Notare che questa data è inizialmente impostata al giorno in cui l'oggetto viene creato e sarà cambiata automaticamente ogni volta che l'oggetto viene modificato. Non vi è alcun modo per personalizzarla. Per esempio, è possibile utilizzare questa data come Ordinamento insieme ad un criterio Tipo impostato su Pagina, per visualizzare tutte le pagine modificate di recente entro la settimana scorsa. L'elenco What's New sulla homepage di LearnPlone.Org usa la Data di Modifica come criterio data. In questo modo i documenti appena creati e quelli che sono stati aggiornati appaiono nell'elenco.

#### Date specifiche degli Eventi

Le due seguenti date si applicano **solo** agli oggetti Eventi. Queste due date sono molto efficaci per la creazione di Collezioni Eventi Recenti e Prossimi Eventi che permetterà al tuo pubblico di conoscere ciò che la tua organizzazione sta facendo e farà in futuro.

#### Data di Inizio

La Data di Inizio è semplicemente la data da cui parte un evento.

#### Data di Fine

La Data di Fine è semplicemente la data in cui l'evento si conclude.

#### Data di Pubblicazione

La Data di Pubblicazione è la data in cui un oggetto è stato pubblicato l'ultima volta. Può essere impostata manualmente per mezzo del campo Data di Accessibilità o, se quest'ultima non è stata impostata, può essere calcolata in base alla data in cui l'oggetto è stato pubblicato l'ultima volta.

Per visualizzare la Data di Pubblicazione sulle proprie pagine è necessario attivare l'opzione “*Visualizza la data di pubblicazione nelle informazioni personali*” nel **\*\*Pannello di Configurazione del Sito\***. La Data di Pubblicazione sarà mostrata prima della Data di Modifica dell'oggetto all'interno dell'area informazioni personali. Per essere sicuri che tutte le funzioni attivino anche l'opzione “*Consenti a chiunque di vedere le informazioni personali*” all'interno del **Pannello di Impostazioni sicurezza**.

### Impostazione Date

Una cosa che può causare confusione sulle date è come impostare i loro Criteri. Essi hanno una configurazione che non è come quella degli altri. Prima di tutto, devi scegliere se desideri una Data Relativa o un Intervallo di Date.

La Data Relativa permette di costruire un'**istruzione condizionale**. Come ad esempio: gli articoli modificati da meno di 5 giorni nel passato. L'Intervallo di Date consente di specificare un **determinato range di date**, ad esempio dal 01/02/08 al 02/02/08. L'Intervallo di Date è utile quando si desidera creare una Collezione con una data statica che non cambierà. La Data Relativa può essere molto utile in quanto vi permetterà di creare Collezioni che sono automaticamente auto-aggiornate, come una Collezione News Recenti o una Sezione Prossimo Evento.

### Data Relativa

Osservando per prima l'opzione Data Relativa, si nota che abbiamo tre opzioni da compilare.

La prima opzione è **Quale giorno**. Questo ci permette di selezionare il numero di giorni che il nostro criterio comprenderà. Una delle opzioni è chiamata *Adesso*. L'utilizzo di questa opzione imposterà l'intervallo di date al giorno corrente. Le altre due opzioni non hanno importanza e possono essere ignorate quando si utilizza *Adesso*.

La seconda opzione è **Nel passato o nel futuro**. Questo ci permette di scegliere se stiamo cercando in avanti o indietro nel tempo.

L'ultima opzione è **Prima o dopo**. Qui si può scegliere tra tre opzioni. *Minore di* ci permette di includere tutto da ora a un periodo di tempo pari o inferiore all'impostazione **Quale giorno**, sia in passato che nel futuro. *Maggiore di* includerà tutto ciò oltre il nostro numero di giorni specificato, pari o superiore di **Quale giorno**. Infine *In Giornata* comprenderà solo gli oggetti del giorno che abbiamo specificato in **Quale giorno**. Utilizzando l'esempio nell'immagine qui sopra se avessimo selezionato *In Giornata* invece di *Minore di* la nostra Collezione mostrerebbe solo gli oggetti che sono stati modificati 5 giorni fa (stiamo usando il criterio Data di Modifica).

Se questo per te è fonte di confusione, prova a leggerlo come una frase sostituendo nei campi le opzioni che hai scelto. “Voglio i risultati per includere oggetti **Prima o dopo di Quale giorno, Nel passato o nel Futuro**. Il nostro esempio diventerebbe “Voglio i risultati che includono gli oggetti **Minore di 5 giorni nel passato**”.

### Intervallo di Date

L'**Intervallo di Date** è molto più facile da capire. Sono obbligatori sia una Data di Inizio che una Data di Fine (non confondere questi termini con le date specifiche dell'Evento!). L'Intervallo di Date ci permette di inserire un inizio e una fine e viene mostrato tutto ciò che è entro la suddetta finestra. Si noti anche che ci permette di specificare una determinata ora del giorno.

## 1.8 Gestione delle Portlet

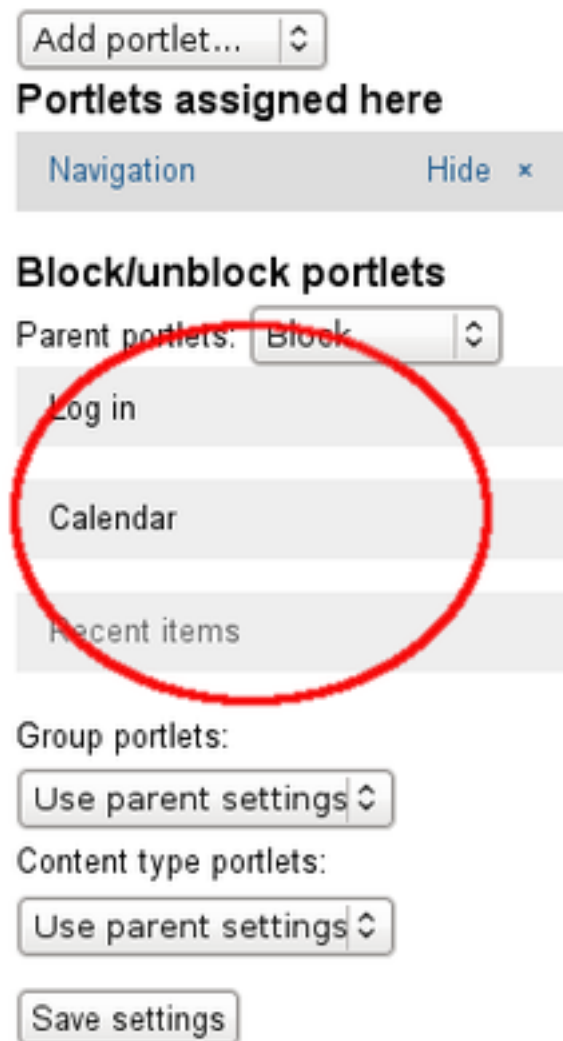
Una introduzione all'uso e alla gestione delle portlets.

### 1.8.1 Gerarchia delle Portlet

**Le Portlet utilizzano un approccio gerarchico che determina come e se devono apparire in ogni sezione del sito.**

Le Portlet utilizzano un approccio basato sulla gerarchia. Per impostazione predefinita, le portlet che assegni alla radice (home page) del sito si propagano verso tutte le sottosezioni dello stesso. Se desideri un diverso insieme di portlet o un ordinamento differente per una particolare sotto-sezione, dovrai utilizzare il controllo **Blocca/sblocca portlets** per “bloccare” le portlet ereditate dalla pagina superiore. Quando blocchi le Portlet, è necessario aggiungere esplicitamente tutte quelle che desideri vedere sulla pagina figlia.

La schermata di gestione delle portlet è stata aggiornata in Plone 4 per mostrare tutte le portlet, incluse quelle bloccate. Gli utenti ora possono vedere ciò che è stato bloccato e ciò che è stato ereditato. Quando una portlet è bloccata, si noterà un sottile cambiamento di colore nella schermata di gestione portlet:



In questo schema, le nostre Portlets sono rappresentate in blu sotto il titolo della Pagina:

Come puoi vedere abbiamo due Portlet nella nostra pagina iniziale (navigation and recent items). Entrambe appariranno nella pagina About a causa della gerarchia delle portlet.

Tuttavia, nella pagina Documentation abbiamo aggiunto una terza portlet - la Collection Portlet. Qui stiamo ancora permettendo la visualizzazione delle Portlet della pagina genitore ma in più abbiamo espressamente aggiunto la Collection Portlet.

Su **entrambe** le pagine Tutorials e Videos dobbiamo bloccato le portlet ereditate dai genitori, perché non vogliamo che la Collection Portlet che si trova nella pagina Documentation venga mostrata. Quando blocchiamo le Portlet ereditate dai Genitori dobbiamo ri-aggiungere le portlet a **ogni** pagina figlia. In questo caso ri-aggiungiamo la Navigation Portlet ad entrambi e successivamente la Search Portlet a tutti e due.

Ricorda che le pagine figlie ereditano solo dalla loro pagina padre superiore. Nel nostro esempio, se aggiungessimo una pagina chiamata *Staff* sotto About, senza altre portlet se non quelle ereditate dal genitore, essa mostrerebbe le stesse portlet mostrate sia nella Home Page che nella pagina About. Tuttavia, le sue portlet non sarebbero ereditate dalla Home page, ma dalla pagina About. Se dovessimo cambiare la pagina About e aggiungere una Search Portlet, la nostra Pagina Staff rispecchierebbe le portlet nella pagina About e non più quelle nella Home Page.

### 1.8.2 Gestione delle Portlets


**Come aggiungere, rimuovere e riordinare le portlets.**

Per iniziare a manipolare le portlet, è necessario trovare il link **Gestione Portlet**, solitamente posizionato nella parte inferiore di ogni colonna laterale. In Gestione Portlet è possibile crearne di nuove, rimuoverle, rinominarle e riordinarle.

[All recent changes...](#)

Manage portlets


Cliccando su questo link verremo portati ad una nuova pagina che ci permetterà di modificare le portlet. L'altro metodo per arrivare a questa schermata è quello di aggiungendo `/@@manage-portlets` alla fine dell'URL della pagina dove si vuole modificare le portlet. Ad esempio, per modificare le portlet della pagina About Us, l'URL deve diventare `www.myplonesite.org/about/@@manage-portlets`.


Add portlet... 

**Portlets assigned here**


Log in	↓ Hide ×
Calendar	↑ ↓ Hide ×
Recent items	↑ Hide ×

### Block/unblock portlets

Parent portlets:  

Group portlets:  

Content type portlets:



### Aggiungere una Portlet

Per aggiungere una Portlet basta semplicemente selezionare **Aggiungi Portlet** dalla casella a discesa e cliccare sul tipo che si desidera aggiungere. Le diverse opzioni disponibili saranno spiegate nella sezione successiva.

### Modificare una Portlet Esistente

Per modificare le proprietà di una Portlet esistente è sufficiente fare click sul suo nome. Nell'esempio a sinistra, se volessimo modificare le proprietà della portlet di navigazione, si dovrebbe Cliccare su *Navigazione*. Ogni tipo di portlet avrà diverse opzioni di configurazione disponibili.

### Riordinare le Portlet

Per modificare l'ordine delle Portlet è sufficiente fare click sulle **frece blu**. Questo influenzerà l'ordine di visualizzazione delle portlet nella pagina.

### Rimuovere le Portlets

Per rimuovere una Portlet, cliccare sulla **“X” rossa** di fianco al nome della stessa.

### Nascondere le Portlets

Da Plone 4 puoi visualizzare/nascondere le portlets utilizzando il rispettivo link visualizza/nascondi.

Come avrai notato nella schermata “Gestisci Portlet”, puoi modificare le portlets sia sul lato destro sia su quello sinistro della pagina. Questo perché ci sono due colonne per le portlet, una a sinistra e una a destra. **Le Portlet appariranno solo sul lato in cui vengono aggiunte.**

E’ possibile aggiungere più di una portlet dello stesso tipo in una pagina. Non c’è nessun limite rispetto a quante volte possa essere utilizzata la stessa portlet o al numero di portlet per pagina.

### 1.8.3 I tipi di Portlet

#### Descrizione dei tipi di Portlet disponibili

Ci sono diversi tipi di Portlet da scegliere. A volte, il nome dei vari tipi può essere ambiguo. Inoltre, alcuni possono essere configurati attraverso la Gestione Portlet e altri richiedono configurazioni attraverso la ZMI o la preventiva creazione di un oggetto. Di seguito è riportato un elenco con la descrizione base d’uso e le funzionalità di ogni tipo di portlet disponibile.

#### Navigazione

La portlet di Navigazione **permette agli utenti di navigare il sito** con facilità fornendo una “mappa del sito” strutturata, o albero di navigazione. Hai la possibilità di configurare la portlet in modo tale che la navigazione mostri l’intero sito o scegliere solo di visualizzare il contenuto della cartella corrente. In LearnPlone.Org, è possibile vedere un esempio della Portlet di Navigazione nella colonna di sinistra. Addentrandosi nel sito, l’albero continuerà ad espandersi. Ci sono diverse opzioni disponibili per modificare il comportamento della Portlet di Navigazione.

#### Calendario

La portlet Calendario è molto semplice e permette di visualizzare un Calendario sul proprio sito. Questa portlet non ha opzioni personalizzabili. Se hai pubblicato degli oggetti eventi sul tuo sito, i giorni in cui si verificheranno saranno in grassetto e cliccandoci sopra si verrà indirizzati all’ evento corrispondente sul sito.

#### Classico

La Portlet Classico si riferisce al modo in cui le portlet sono state utilizzate nelle vecchie versioni di Plone, prima di Plone 3. È necessario creare un Page Template nella ZMI ed impostare correttamente il percorso e le macro per attivare la portlet. Questo richiede una conoscenza tecnica sia di TALES sia della ZMI.

#### Collezione

La Portlet Collezione ti permette **di visualizzare i risultati di una Collezione**. È necessario creare precedentemente una collezione e dopo aver aggiunto questa Portlet, si potrà specificare la Collezione da utilizzare. Questo è un ottimo modo per riassumere i risultati di una importante raccolta in modo che sia facilmente visibile al pubblico.

#### Eventi

La Portlet Eventi **mostrerà gli Eventi Futuri**, a condizione che ci siano Eventi sul proprio sito. È possibile personalizzare il numero di eventi che si desidera visualizzare e specificare un filtro in base allo stato di pubblicazione.



## Autenticazione

La Portlet di Autenticazione è un'altra portlet non configurabile che semplicemente **visualizza la Form di Log in**, per consentire agli utenti di autenticarsi. Una volta che un utente è loggato sul sito, questa Portlet verrà nascosta automaticamente.

## Notizie

La Portlet Notizie funziona esattamente come la Portlet Eventi. Tuttavia invece che visualizzare Eventi, **mostrerà le ultime Notizie**. Nuovamente potrai personalizzare il numero di Notizie da visualizzare e filtrarle in base allo stato di pubblicazione.

## Flusso RSS

La Portlet Flusso RSS ti permette di fare un link ad un Flusso RSS, di scegliere quante notizie visualizzare e di specificare il tempo di aggiornamento.

## Contenuti Recenti

La Portlet Contenuti Recenti visualizza un numero personalizzabile **di Contenuti Recenti**, elencati per Titolo. Un Contenuto viene classificato Recente in base alla Data dell'ultima Modifica fatta.

## Elenco di Revisione

La Portlet Elenco di Revisione visualizzerà un elenco di oggetti **da revisionare**. Se si utilizza un workflow dove è presente uno stato di revisione (e sono impostati correttamente i ruoli globali per gli utenti), questa portlet è molto comoda ai revisori per tenere sott'occhio quando un oggetto viene inviato per essere sottoposto a revisione. Questa Portlet appare solo ai revisori, poichè i contenuti in stato sottoposto a revisione non sono visibile al pubblico.

## Ricerca

La Portlet Ricerca visualizzerà una casella di ricerca nella colonna dove viene aggiunta. La ricerca del testo specificato avverrà nel titolo, nella descrizione e nel corpo degli oggetti del sito. C'è la possibilità di abilitare la Ricerca Istantanea che mostra i risultati durante la digitazione del testo da ricercare, se il browser supporta JavaScript.

## Testo Statico

La Portlet Testo Statico permette di inserire del testo come in una normale Pagina. E' utile per aggiungere collegamenti ad altri siti o qualsiasi informazione statica. Un esempio è la Portlet "Ancora Perplexi?" sulla destra di questo sito.



## **2.1 Creare un tema con Diazo**

**Questa guida fornisce una panoramica sull'uso di DIAZO per creare un tema in Plone.**

## Contents

- *Creare un tema con Diazo*
  - *Che cos'è un tema Diazo?*
  - *Uso del pannello di controllo*
    - \* *Selezionare un tema*
    - \* *Creare un nuovo tema*
    - \* *Caricamento di un tema esistente*
    - \* *Modifica del tema*
    - \* *Ispezione del tema*
    - \* *Il generatore delle regole*
    - \* *Impostazioni avanzate*
  - *Riferimenti*
    - \* *Sviluppo e test dei temi*
      - *Installazione sul filesystem*
      - *Installazione attraverso il web*
      - *Installazione come file zip*
      - *Installazione tramite un pacchetto Python (solo per programmatori)*
    - \* *Il file 'manifesto'*
    - \* *Sintassi delle regole*
      - *Selettori*
      - *Condizioni*
      - *Regole disponibili*
      - *rules*
      - *theme*
      - *replace*
      - *before*
      - *drop*
      - *merge*
      - *Modifiche avanzate*
    - \* *Parametri del tema*
    - \* *Debug del tema*
    - \* *Regole di uso comune*
    - \* *Uso avanzato di portal\_css per la gestione del proprio CSS*

### 2.1.1 Che cos'è un tema Diazo?

Un “tema” definisce l'aspetto grafico e le modalità di interazione per un sito web (in questo caso un sito basato su Plone).

Diazo (già conosciuto come XDV) è una tecnologia utilizzabile per creare il tema di un sito web. Non è specifico per Plone, ma è stato creato dalla comunità Plone e, a partire dalla versione Plone 4.3, fornisce la modalità predefinita per l'applicazione di un tema ad un sito Plone. Per saperne di più: <http://diazo.org>.

I temi Diazo possono essere un po' differenti da quelli creati in altri sistemi, ed anche dai temi creati per le precedenti versioni di Plone. Un tema Diazo è in realtà la trasformazione di contenuti – in questo caso l'output di un Plone “base” – in un diverso insieme di modelli HTML mediante l'applicazione di regole che combinano il modello HTML statico del risultato finale che si vuole ottenere con il contenuto dinamico proveniente da Plone.

La precedente modalità per la creazione di un tema in un sito Plone (come pure la modalità presente in molti altri sistemi di gestione dei contenuti) si basa sulla sovrascrittura (overriding) selettiva di template e script che Plone utilizza per costruire una pagina con le versioni personalizzate che producono un diverso markup HTML. Quest'ultima strategia può risultare sicuramente più potente, ma richiede però una profonda conoscenza dei meccanismi interni di

Plone e dei comandi di tecnologie usate lato server come Zope Page Templates ed anche Python. I temi Diazo sono invece facili da capire per i progettisti web ed anche per chi non è sviluppatore.

Un tema Diazo si compone di tre elementi:

1. Uno o più modelli HTML, indicati anche come file del tema, che rappresentano l'aspetto e l'interfaccia desiderati.

Essi conterranno segnaposti per il contenuto che deve essere fornito dal sistema di gestione dei contenuti di Plone. I modelli fanno di solito riferimento a file CSS, JavaScript e di immagini con i relativi percorsi. La modalità più comune usata per creare un tema prevede l'utilizzo di software come Dreamweaver o di un editor di testo per impostare i relativi markup, stili e script, e testare poi localmente il tema in un browser web.

2. Il contenuto a cui il tema deve essere applicato. In questo caso si tratta dell'output da Plone.
3. Un file di regole, che definisce il modo in cui i segnaposti nel tema (cioè il modello HTML) dovranno essere sostituiti dal markup pertinente nel contenuto.

Il file delle regole usa la sintassi XML (simile all'HTML). Questo è un semplice esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
<rules
  xmlns="http://namespaces.plone.org/diazo"
  xmlns:css="http://namespaces.plone.org/diazo/css"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <theme href="theme.html" />
  <replace css:content-children="#content" css:theme-children="#main"
  />
</rules>
```

Nell'esempio si sostituiscono i contenuti (nodi figli) di un elemento segnaposto con id HTML main nel file del tema (theme.html, che si trova nella stessa directory del file rules.xml, come indicato nel riferimento della regola <theme />) con i contenuti (figli) dell'elemento con l'id HTML content nel markup generato da Plone.

Quando viene applicato questo tema, il risultato avrà un aspetto molto simile a quello del file HTML statico theme.html (ed ai suoi file di riferimento CSS, JavaScript ed immagini), eccezion fatta per il segnaposto identificato nel tema dal nodo con id main che sarà riempito dall'area di contenuto principale di Plone.

Plone viene fornito con un tema di esempio chiamato appunto Example theme, che usa il venerabile [Twitter Bootstrap](#) per costruire un tema semplice ma funzionale che espone la maggior parte delle funzionalità di Plone "base". Si consiglia di studiarlo - in particolare il file rules.xml - per capire meglio come lavorano i temi Diazo.

## 2.1.2 Uso del pannello di controllo

Dopo l'installazione del package 'Diazo theme support' in un sito Plone, nella pagina di configurazione del sito Plone comparirà il pannello di controllo Theming.

La scheda principale, Themes, di questo pannello di controllo mostrerà tutti i temi disponibili, con i tasti comando per attivare/disattivare, modificare, copiare o cancellare ciascun tema, come pure i tasti comando per creare nuovi temi o far apparire il contenuto di questo documento. .

Con un click sull'immagine con l'anteprima del tema si apre l'anteprima del tema in una nuova scheda o in una nuova finestra. L'anteprima è navigabile, ma l'invio di un form ed alcune funzioni avanzate non funzionano.

### Selezionare un tema

Per applicare un tema esistente basta un click sul tasto comando **Activate** posizionato sotto l'anteprima del tema. Il tema attualmente attivo sarà evidenziato in giallo. Se il tema attivo viene disattivato, non risulterà applicato alcun tema. Di conseguenza, pertanto verrà applicato il tema "base" di Plone.

n.b.: Al pannello di controllo **Theming** non si applica mai il tema, assicurando in tal modo che si potrà sempre disattivare un tema che genera errore e che potrebbe rendere inutilizzabile lo stesso pannello di controllo. Non si vedrà pertanto alcuna differenza immediatamente dopo l'abilitazione di un tema. Basta però passare a un'altra pagina del sito Plone e si dovrebbe vedere il tema applicato.

### Creare un nuovo tema

I nuovi temi possono essere creati in due modi:

- Nel pannello di controllo **Theming**, Click sul tasto comando **New theme** nella parte superiore della scheda **Themes** ed immettere un titolo e una descrizione nel form visualizzato. Verrà creata la struttura essenziale del tema, e verrà visualizzata la pagina **Modify theme** dove si potranno modificare o creare i file del tema e delle regole.
- Click sul tasto comando **Copy** presente sotto ad ogni tema esistente e, nel form visualizzato, inserire il titolo e la descrizione del tema. Verrà creato un nuovo tema copia del tema esistente e verrà visualizzata la pagina **Modify theme** dove si potranno modificare o creare i file del tema e delle regole.

### Caricamento di un tema esistente

I temi possono essere distribuiti come file Zip contenenti i file del modello HTML e delle regole. Per caricare un file esistente basta un click sul tasto comando **Download** presente sotto al tema nella scheda **Themes** del pannello di controllo di **Theming**.

Per caricare un file di questo tipo in un altro sito si usa il tasto comando **Upload Zip file** nella scheda **Themes** del pannello di controllo di **Theming**. Si può scegliere se sostituire o meno un tema esistente ed avere lo stesso nome (in base al nome della directory di livello superiore contenuta all'interno del file Zip).

Si può anche caricare il file di un modello statico HTML che non contiene il file delle regole, quale può essere per esempio un progetto fornito da un progettista che non è un praticante di Plone.

In questo caso verrà aggiunto automaticamente un file di base (`rules.xml`) per permettere di iniziare a costruire un tema utilizzando la schermata **Modify theme**. Il file di regole generato assume che il file principale del modello HTML abbia nome `index.html`, che potrà comunque essere cambiato in `rules.xml`.

Una volta caricato con successo un file Zip del tema, verrà presentata la schermata **Modify theme** dove si potrà modificare il file del tema o creare un nuovo file.

Suggerimento: Se si riceve un messaggio di errore del tipo "Il file caricato non contiene un archivio valido di tema", questo di solito significa che è stato caricato un file zip che contiene più file e cartelle, piuttosto che una singola cartella di livello superiore contenente tutte le risorse del tema. Ciò potrebbe accadere se è stato compresso un tema o un modello HTML aggiungendo i relativi file e cartelle direttamente in un archivio Zip, piuttosto che comprimere la directory in cui sono stati trovati. Per risolvere questo problema, è sufficiente decomprimere l'archivio in una nuova directory sul computer locale, salire di un livello, e comprimere questa directory da sola in un nuovo file Zip, che è poi possibile caricare.

### Modifica del tema

Si accede alla modifica di un tema con un click sul tasto comando **Modify theme** posto sotto al tema nella scheda **Themes** del pannello di controllo di **Theming**. Questa schermata viene aperta automaticamente quando si crea o si

carica un nuovo tema.

n.b.: Da Plone si possono modificare solo i temi creati o caricati dal pannello di controllo di Theming. Non possono invece essere modificati i temi installati dagli add-on di terze parti, anche se le modifiche apportate sul file system si rifletteranno immediatamente se Zope viene eseguito in modalità di debug. Per modificare un tema presente sul filesystem, si può copiarlo in un nuovo tema Plone con il tasto comando Copy presente sotto il tema nel pannello di controllo di Theming

La schermata Modify theme mostra inizialmente un gestore di file con l'albero dei file sulla sinistra ed un editor sulla destra. Un Click su un file nell'albero dei file apre un editor o un'anteprima: file HTML, CSS, JavaScript ed altri file di testo possono essere visualizzati direttamente nell'editor. Altri file (p.es. immagini) saranno aperti in anteprima.

N.b.: Nel browser Internet Explorer di Microsoft non è disponibile l'editor avanzato con la sintassi evidenziata.

Un click su New folder per creare una nuova cartella. Questo si può ottenere anche con un click destro su una cartella dell'albero dei file.

Un click su New file per creare un nuovo file. Questo si può ottenere anche con un click destro su una cartella dell'albero dei file.

Un click su Upload file per caricare un file dal computer locale. Questo si può ottenere anche con un click destro su una cartella dell'albero dei file.

Un click su Preview theme per visualizzare in anteprima il tema secondo il modello e le regole attualmente salvate. L'anteprima è navigabile ma i form ed alcune funzionalità avanzate non funzionano.

Per salvare le modifiche fatte nel file corrente, click sul tasto comando Save file oppure utilizzare i tasti di scelta rapida Ctrl+S (Windows/Linux) o Cmd+S (Mac).

Per rinominare o cancellare un file o una cartella basta un click destro sull'elemento di interesse nell'albero dei file e si seleziona poi l'azione desiderata.

## Ispezione del tema

Lo strumento di ispezione di un tema fornisce un'interfaccia avanzata per scoprire e costruire le regole di un tema Diazo. Può essere lanciato con il tasto comando Show inspectors presente nella schermata Modify theme per i temi propri di Plone, o con il tasto comando Inspect theme presente sotto ad un tema del filesystem nella scheda Themes del pannello di controllo di Theming.

Lo strumento di ispezione di un tema è costituito da due pannelli:

- Il mockup HTML. Se ci sono diversi file HTML in un tema, è possibile passare da uno all'altro utilizzando la lista a discesa posizionata sotto il pannello del modello HTML.
- Il Unthemed content. Mostra Plone senza alcun tema applicato.

La dimensione di entrambi i pannelli possono essere massimizzate con un click sulle icone delle frecce presenti in alto a destra in ciascun pannello.

I pannelli HTML mockups ed Unthemed content possono passare alla vista sorgente e mostrare il codice HTML sottostante con un click sulle icone tag presenti in alto a destra in ciascun pannello.

Posizionando il mouse sopra gli elementi nei pannelli del mockup HTML o del Unthemed content, si vedrà:

- Un contorno che mostra l'elemento sotto il cursore.
- Un selettore CSS o XPath nella barra di stato nella parte inferiore del pannello; il selettore identifica univocamente l'elemento in una regola Diazo.

Click su un elemento o premere Enter quando il mouse è posizionato sopra un elemento per selezionarlo. L'elemento selezionato più di recente in ciascun pannello viene mostrato nella barra di stato presente nella parte inferiore di ciascun pannello.

Premendo Esc quando il mouse è posizionato sopra un elemento per selezionare il suo genitore. Ciò è utile quando si cerca di selezionare elementi contenitori “non visibili”. Premere Enter per salvare la selezione.

I contenuti del pannello del mockup HTML o (più comunemente) di quello del Unthemed content sono navigabili, per ottenere per esempio una pagina di contenuto che richiede regole del tema specifiche disabilitando lo strumento di ispezione. Utilizzare i commutatori in basso a destra del pannello in questione per attivare o disattivare il selettore.

### Il generatore delle regole

Usare il tasto comando Build rule nella parte superiore della schermata Modify theme o Inspect theme per lanciare la procedura guidata per la costruzione interattiva delle regole. Verrà richiesto il tipo di regola da costruire e quindi di selezionare, come richiesto, i relativi elementi nei pannelli del mockup HTML e/o di Unthemed content. Per impostazione predefinita, vengono utilizzate le selezioni salvate, a meno che non si deselezioni la casella Use selected elements nella prima pagina della procedura guidata.

Al termine della procedura guidata, verrà mostrata la regola generata. Se si vuole, la regola può essere modificata. Con un click su Insert, la nuova regola generata viene inserita nell’editor di rules.xml in corrispondenza o vicino all’attuale posizione del cursore. È possibile spostare o modificare ulteriormente la regola a proprio piacimento.

Click Preview theme per l’anteprima del tema in una nuova scheda o finestra. Se sono state fatte modifiche, ricordarsi di salvare il file rules.xml.

N.b.: In modalità di solo lettura, si possono costruire regole ed ispezionare il modello HTML ed il tema ma non cambiare il file rules.xml file. In questo caso, anche il tasto comando Insert del generatore di regole non sarà disponibile.

N.b.: Nel browser Internet Explorer di Microsoft non è disponibile la possibilità di inserire regole con la procedura guidata Build rule, anche se sarà data la possibilità di copiare la regola negli appunti quando si utilizza questo browser.

### Impostazioni avanzate

Il pannello di controllo di Theming contiene anche una scheda con nome Advanced settings. E qui comincia l’avventura.

La scheda Advanced settings è divisa in due aree. La prima, Theme details, contiene le impostazioni che vengono modificate quando viene applicato un tema dal pannello di controllo Themes.

Queste sono:

- Abilitazione dei temi Diazo.
- Il percorso del file di regole, chiamato convenzionalmente rules.xml, sia relativo alla root del sito Plone o come percorso assoluto verso un server esterno.
- Il prefisso da applicare per passare nei temi da percorsi relativi (p. es. i riferimenti ad immagini nell’attributo src del tag <img />) a percorsi assoluti in fase di visualizzazione dei contenuti.
- Il DOCTYPE HTML da applicare all’output generato, se diverso dal valore predefinito XHTML 1.0 Transitional.
- Se permettere o meno la lettura dalla rete delle risorse del tema (come rules.xml). Disattivare questa voce porta ad un modesto miglioramento delle prestazioni.
- Una lista di nomi di host ai quali non viene mai applicato un tema. Spesso contiene 127.0.0.1, che consente di vedere, per esempio nella fase di sviluppo, un sito senza tema in <http://127.0.0.1:8080> ed il sito con tema in <http://localhost:8080>.
- Una lista di parametri del tema e le espressioni TALES che li generano (vedi di seguito).



Il secondo, Theme base, controlla la presentazioni dei contenuti senza l'applicazione di alcun tema, utilizzabile anche se non viene applicato alcun tema Diazo. Queste sono le impostazioni che si trovavano nel pannello di controlli di Themes nelle precedenti versioni di Plone.

### 2.1.3 Riferimenti

Il resto di questa guida contiene materiale di riferimento utile per i realizzatori di temi.

#### Sviluppo e test dei temi

Per costruire e testare un tema, si deve prima creare un modello statico HTML con l'aspetto grafico e le modalità di interazione che si desiderano, e realizzare poi un file di regole per descrivere come il contenuto di Plone viene mappato nei segnaposto di questo modello.

Il modello può essere creato ovunque con l'utilizzo dello strumento che si ritiene più adatto per la realizzazione di pagine web. Per semplificare l'integrazione con Plone, si raccomanda di essere certi che vengano usati i collegamenti relativi per le risorse quali file CSS, JavaScript ed immagini, in modo che siano visualizzati correttamente quando vengono aperti in un browser Web da un file locale. Plone convertirà automaticamente questi collegamenti relativi negli appropriati percorsi assoluti, assicurando così il corretto funzionamento del tema indipendentemente dall'URL visualizzato dall'utente quando il tema è applicato ad un sito Plone.

Ci sono diversi modi per rendere disponibile il tema in Plone:

#### Installazione sul filesystem

Se si usa un'installatore o un "buildout" standard per allestire un sito Plone, dovrebbe allora essere presente una directory con nome resources nella root dell'installazione Plone (questa directory viene creata se si usa l'opzione resources nella ricetta del buildout plone.recipe.zope2instance. Vedi <http://pypi.python.org/pypi/plone.recipe.zope2instance> per maggiori dettagli)

Dentro questa directory si può trovare (o creare) una directory theme che viene usata per contenere temi. Ciascun tema richiede una propria directory con un nome univoco. Se ne crea una (p. es. resources/theme/mytheme) e si inseriscono al suo interno i file HTML e ogni risorsa di riferimento. Se lo si desidera, si possono usare subdirectory, ma si consiglia di conservare i file HTML di base del tema nella parte superiore della cartella del tema.

Sarà necessario anche un file di regole chiamato rules.xml all'interno della directory. Se non è già disponibile se ne crea uno vuoto:

```
<?xml version="1.0" encoding="UTF-8"?>
<rules
  xmlns="http://namespaces.plone.org/diazo"
  xmlns:css="http://namespaces.plone.org/diazo/css"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<theme href="theme.html" />

<replace css:content-children="#content" css:theme-children="#main" />

</rules>
```

Se si esegue Zope in modalità debug (p. es.. è stato avviato con bin/instance fg), le modifiche fatte al tema e alle regole hanno effetto immediato. Si può avere un'anteprima o abilitare il tema attraverso il pannello di controllo Themes, e quindi modificare come si desidera ed in modo interattivo il file rules.xml o il modello del tema.

### Installazione attraverso il web

Se lo si preferisce (o non si ha l'accesso al filesystem), si può creare completamente il tema dal pannello di controllo di Plone, sia per duplicazione di un tema esistente, sia partendo da zero con un tema quasi vuoto.

Per maggiori dettagli si rimanda alle istruzioni sull'uso del pannello di controllo descritte precedentemente.

Una volta creato, il tema può essere modificato dal pannello di controllo di Theming. Per maggiori dettagli si rimanda alle istruzioni descritte precedentemente.

### Installazione come file zip

I temi possono essere scaricati da Plone come file Zip; questi file possono essere poi caricati in altri siti web.

Per maggiori dettagli si rimanda alle istruzioni sull'uso del pannello di controllo descritte precedentemente.

E' infatti possibile creare archivi zip del tema validi, comprimendo la cartella di un tema presente su filesystem utilizzando uno strumento standard di compressione come 7-Zip o Winzip (per Windows) o l'azione Compress incorporata nel Mac OS X Finder. Bisogna solo essere certi di comprimere esattamente la cartella che contiene tutti i file del tema ed il file rules.xml. (Non comprimere direttamente i contenuti della cartella: il file zip quando viene scompattato deve produrre esattamente una cartella che a sua volta contiene i relativi file).

### Installazione tramite un pacchetto Python (solo per programmatori)

Se si sta creando un pacchetto Python che contiene le personalizzazioni di Plone che si intendono installare nel sito, si può usarlo per registrare un tema da installare nel sito.

Per fare questo si posiziona una directory, p. es. di nome. Theme, all'inizio del pacchetto, accanto al file Zope configure.zcml, ed si aggiunge una dichiarazione `<plone:static />` nel file configure.zcml:

```
<configure
  xmlns:plone="http://namespaces.plone.org/plone"
  xmlns="http://namespaces.zope.org/zope">

  ...

  <plone:static name="mytheme" directory="theme" type="theme" />

  ...

</configure>
```

Si noti la dichiarazione del namespace plone nell'elemento radice `<configure />`. I file del tema ed il file rules.xml vanno posizionati nella directory theme.

Se il pacchetto ha un GenericSetup profile, si può abilitare dopo l'installazione di questo profilo aggiungendo nella directory profiles/default un file theme.xml contenente p. es.:

```
<theme>
  <name>mytheme</name>
  <enabled>true</enabled>
</theme>
```

## Il file ‘manifesto’

E’ possibile dare ulteriori informazioni sul tema inserendo all’inizio della directory di un tema un file con nome manifest.cfg accanto al file rules.xml.

Il file ha un aspetto di questo tipo:

```
[theme]

title = My theme

description = A test theme
```

Come si vede, il file ‘manifesto’ può essere utilizzato per fornire un titolo del tema più comprensibile ed una descrizione più lunga da usare poi nel pannello di controllo. E’ richiesta solo l’instestazione [theme] – tutte le altre chiavi sono opzionali.

Si può anche impostare:

```
rules = http://example.org/myrules.xml
```

per usare un nome per il file delle regole diverso da rules.xml (si deve fornire un URL o un percorso relativo).

Per cambiare il prefisso per il percorso assoluto (vedi Impostazioni avanzate), si usa:

```
prefix = /some/prefix
```

Per impiegare un DOCTYPE diverso da XHTML 1.0 Transitional per il contenuto a cui viene applicato il tema, aggiungere p. es.:

```
doctype = html
```

Per visualizzare nel pannello di controllo Theming un’anteprima user-friendly del tema, aggiungere:

```
preview = preview.png
```

preview.png è il file di un’immagine relative to the location del file manifest.cfg.

Estensioni del motore di Diazo possono aggiungere il supporto per ulteriori blocchi di parametri configurabili.

## Sintassi delle regole

Nel seguito un breve sommario della sintassi delle regole di Diazo. Vedi <http://diazo.org> per maggiori dettagli ed altri esempi.

### Selettori

Ciascuna regola è composta da un tag XML che opera su uno o più elementi HTML nel contenuto e /o sul tema. Gli elementi su cui operare sono indicati da attributi delle regole noti come selettori.

Il modo più semplice per selezionare gli elementi è quello di utilizzare una espressione selettore CSS, come ad esempio `css:content=#content` o `css:theme=#main .content`. Si può utilizzare una qualsiasi espressione CSS3 valida (inclusi pseudo-selettori quali: `first-child`).

I selettori standard, `css:theme` e `css:content`, operano sull’elemento/i che soddisfano la selezione. Se invece si vuole operare sui figli degli elementi selezionati si deve usare `css:theme-children=“...”` o `css:content-children=“...”`.

Se non è possibile costruire una espressione CSS 3 adeguata, è possibile utilizzare espressioni XPath come `content=/head/link` o `theme=/div[@id=’main’]` (si noti la mancanza di un prefisso `css:` quando si usano le espressioni

XPath). I due approcci sono equivalenti, e si possono combinare liberamente, ma non si può avere ad esempio sia un `css:theme` ed un attributo `theme` nella stessa regola. Per operare sui figli di un nodo selezionato con un'espressione XPath si può usare `theme-children="..."` o `content-children="..."`.

Per approfondire XPath vedi <http://www.w3schools.com/xpath/default.asp>.

### Condizioni

Per impostazione predefinita, ogni regola viene eseguita, anche se le regole a cui non corrispondono elementi non modificano nulla nella pagina attuale. Si può creare una regola, un'insieme di regole o un riferimento al tema (vedi sotto) a condizione che un elemento sia presente nel contenuto, aggiungendo un attributo alla regole del tipo `css:if-content="#some-element"` (per usare invece un'espressione XPath, eliminare il prefisso `css:`). La regola viene ignorata se nessun elemento soddisfa l'espressione.

Suggerimento: se una regola `<replace />` corrisponde a un elemento nel tema, ma non nel contenuto, il nodo tema sarà eliminato e non sostituito. Se non si desidera questo comportamento e non si è sicuri se il contenuto conterrà l'elemento/i in questione, è possibile utilizzare la regola condizionale `css:if-content`. Poiché questa è una situazione comune, è disponibile una scorciatoia: `css:if-content=""` che significa "usare l'espressione dall'attributo `css:content`".

Allo stesso modo è possibile creare una condizione in base al percorso della richiesta corrente utilizzando un attributo del tipo `if-path="/news"` (si noti l'assenza di `css:if-path`). Se questo percorso inizia con una barra (/), l'eventuale corrispondenza sarà con la fine dell'URL. Si può impostare un percorso assoluto usando un barra iniziale ed una finale (`/.../`).

Si possono infine usare espressioni XPath arbitrarie invece di una variabile definita, con un attributo del tipo `if="$host = 'localhost'"`. Per impostazione predefinita, sono disponibili le variabili `url`, `scheme`, `host` e `base` che rappresentano l'URL attuale. I temi possono definire ulteriori variabili nei rispettivi manifesti.

### Regole disponibili

Di seguito il riassunto dei vari tipi di regole.

#### rules

```
<rules>
...
</rules>
```

Racchiude un insieme di regole. Deve essere utilizzato come elemento radice del file delle regole. `<rules />` nidificato può essere utilizzato assieme ad una `condition` per applicare una singola condizione ad un'insieme di regole.

Quando viene utilizzato come elemento radice del file delle regole, debbono essere dichiarati i vari namespace XML:

```
<rules
  xmlns="http://namespaces.plone.org/diazo"
  xmlns:css="http://namespaces.plone.org/diazo/css"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ...
</rules>
```

**theme**

```
<theme href="theme.html" />
<theme href="news.html" if-path="/news" />
<notheme if="$host = 'admin.example.org'" />
```

Sceglie il file del tema da utilizzare. L'attributo href è un percorso relativo a file di regole. Se sono presenti più elementi <theme />, solo per uno di essi può essere assente una condizione. Verrà utilizzato il primo tema con una condizione che sia vera, con il tema senza condizioni utilizzato come riserva.

<notheme /> può essere usato per specificare una condizione che non prevede l'uso di alcun tema. <notheme /> ha la precedenza su <theme />.

Suggerimento: Per essere sicuri di non applicare gli stili a pagine non Plone, aggiungere all'ultimo tema della lista una condizione del tipo `css:if-condition="#visual-portal-wrapper"`, e non inserire alcun tema senza condizione.

**replace**

```
<replace
  css:content="#content"
  css:theme="#main"
/>
```

Sostituisce gli elementi che soddisfano la regola nel tema con i corrispondenti che soddisfano la regola nel contenuto.

**before**

```
<before
  css:content-children="#portal-column-one"
  css:theme-children="#portlets"
/>

<after
  css:content-children="#portal-column-two"
  css:theme-children="#portlets"
/>
```

Inserisce gli elementi che soddisfano la regola nel contenuto prima o dopo i corrispondenti nel tema. Utilizzando theme-children , si possono inserire gli elementi del contenuto selezionati all'inizio (prepend) o alla fine (append) all'interno dei corrispondenti elementi che soddisfano la regola nel tema.

**drop**

```
<drop css:content=".documentByLine" />
<drop theme="/head/link" />
<drop css:theme="#content \*" attributes="onclick onmouseup" />
<strip css:content="#parent-fieldname-text" />
```

Rimuove gli elementi dal tema o dal contenuto. Si noti che a differenza di altre regole, una regola <drop /> o <strip /> può operare sul theme o sul content , ma non su entrambi. <drop /> rimuove gli elementi corrispondenti ed i relativi figli, mentre <strip /> rimuove gli elementi corrispondenti ma non i relativi figli.

A <drop /> può essere data una lista di attributes da rimuovere separati da spazi bianchi. In questo caso gli elementi corrispondenti non saranno rimossi. Usando `attributes="*"` si rimuovono tutti gli attributi.

**merge**

```
<merge
  attributes="class"
  css:content="body"
  css:theme="body"
/>

<copy
  attributes="class"
  css:content="#content"
  css:theme="#main"
/>
```

Queste regole operano sugli attributi. `<merge />` aggiungerà i contenuti letti nel tema per gli attributi indicati, ai valori di ogni attributo esistente nel contenuto avente lo stesso nome; i valori sono separati da spazi bianchi. E' principalmente usato per aggiungere classi CSS.

`<copy />` copia gli attributi dagli elementi che soddisfano la regola nel contenuto, nei corrispondenti elementi nel tema; gli attributi con lo stesso nome eventualmente già presenti nel tema, vengono completamente sostituiti.

L'attributo `attributes` può contenere una lista di attributi separati da spazi bianchi, oppure il valore speciale `*` per operare su tutti gli attributi degli elementi che soddisfano la regola.

### Modifiche avanzate

Invece di selezionare il markup da inserire nel tema dal contenuto, è possibile inserire il markup direttamente nel file delle regole, come nodi figlio dell'elemento della relativa regola:

```
<after css:theme="head">
  <style type="text/css">
    body > h1 { color: red; }
  </style>
</after>
```

Nello stesso modo si può operare sul contenuto. E' così possibile modificarlo prima dell'applicazione delle regole:

```
<replace css:content="#portal-searchbox input.searchButton">
  <button type="submit">
    
  </button>
</replace>
```

Oltre ad aggiungere in questo modo HTML statico, si possono usare le istruzioni XSLT che operano sul contenuto. In XSLT, si possono anche usare direttamente i selettori css:

```
<replace css:theme="#details">
  <dl id="details">
    <xsl:for-each css:select="table#details > tr">
      <dt><xsl:copy-of select="td[1]/text()" /></dt>
      <dd><xsl:copy-of select="td[2]/node()" /></dd>
    </xsl:for-each>
  </dl>
</replace>
```

Utilizzando l'attributo `href` per specificare il percorso di una risorsa relativamente alla root del sito Plone, le regole possono operare su contenuti provenienti da sorgenti che non siano l'attuale pagina restituita da Plone:

```
<after
  css:theme-children="#leftnav"
```

```
css:content=".navitem"
href="/@@extra-nav"
/>
```

## Parametri del tema

E' possibile passare al tema parametri arbitrari a cui si può far riferimento come a variabili nelle espressioni di XPath. I parametri possono essere impostati nel pannello di controllo Theming di Plone e possono anche venire importati da un file manifest.cfg.

Si potrebbe avere, per esempio, un parametro mode impostabile con la stringa live o test. Nelle proprie regole si potrebbe fare qualcosa del genere per inserire un avviso visualizzato quando si lavora sul server di prova:

```
<before css:theme-children="body" if="$mode = 'test'">
  <span class="warning">Attenzione: questo è il server di prova</span>
</before>
```

Si può usare anche direttamente il valore del parametro, p. es.:

```
<before css:theme-children="body">
  <span class="info">Questo è il server di <xsl:value-of select="$mode" />
</span>
</before>
```

I seguenti parametri sono sempre disponibili per i temi Plone:

**scheme** Il nome dello schema dell'URL in entrata (la parte che precede i due punti), normalmente http o https.

**host** Il nome, nell'URL in entrata, del server che ha inviato i dati.

**path** Il segmento dell'URL in entrata relativo al percorso. Non include alcun virtual hosting tokens, è cioè il percorso visto dall'utente finale.

**base** Il Zope base url (la variabile BASE1 di una request a Zope).

Si possono aggiungere ulteriori parametri dal pannello di controllo utilizzando espressioni TALES. I parametri sono elencati, uno per riga, nella scheda Advanced nel formato <name> = <expression>.

Se, per esempio, si vuole evitare di applicare il tema ad ogni pagina caricata dai diversi livelli (overlays) di Plone, si può far uso del parametro ajax\_load della request parameter impostato dai livelli (overlays). In questo caso il file delle regole includerebbe:

```
<notheme if="$ajax\_load" />
```

Per aggiungere questo parametro come pure il parametro mode descritto in precedenza, è possibile aggiungere quanto segue nel pannello di controllo:

```
ajax\_load = python: request.form.get('ajax\_load')
mode = string: test
```

La parte destra presenta un'espressione TALES. Deve restituire un valore di tipo string, integer, float, boolean o None: le liste, i dizionari e gli oggetti non sono supportati. python:, string: ed espressioni di percorso funzionano come nei Zope Page Templates.

Sono disponibili le seguenti variabili quando si costruiscono queste espressioni TALES:

**context** Il contesto dell'attuale request, normalmente un oggetto contenuto.

**request** L'attuale request.

**portal** L'oggetto radice del portale.

**context\_state** La vista `@@plone_context_state`, da cui è possibile cercare altri valori come l'URL del contesto o la vista predefinita.

**portal\_state** La vista `@@plone_portal_state`, da cui è possibile cercare altri valori come la root dell'URL di navigazione o se l'utente attuale è collegato (autenticato?) o meno.

Vedi `plone.app.layout` per i dettagli circa le viste `@@plone_context_state` e `@@plone_portal_state`.

I parametri del tema sono normalmente parte integrante di un `themeTheme` e saranno pertanto impostati in base al manifesto del tema quando il tema viene importato od abilitato. Questo è fatto utilizzando la sezione `[theme:parameters]` nel file `manifest.cfg`. Per esempio:

```
[theme]
title = My theme
description = A test theme
[theme:parameters]
ajax\_load = python: request.form.get('ajax\_load')
mode = string: test
```

### Debug del tema

Quando Zope è in modalità sviluppo (cioè esecuzione in foreground in una console con `bin/instance fg`), il tema sarà ricompilato ad ogni request. Se la modalità non è di sviluppo viene compilato al primo accesso, poi ricompilato solo se vengono cambiati i valori del pannello di controllo.

Anche nella fase di sviluppo è possibile disabilitare temporaneamente il tema aggiungendo alla request una query string con il parametro `diazo.off=1`. Per esempio:

<http://localhost:8080/Plone/some-page?diazo.off=1>

Il parametro viene ignorato se la modalità non è di sviluppo.

### Regole di uso comune

Le ricette che seguono mostrano le regole di uso comune nella costruzione di temi per Plone:

Per copiare il titolo della pagina:

```
<replace css:theme="title" css:content="title" />
```

Per copiare il tag `<base />` (necessario perchè funzionino i link di Plone):

```
<replace css:theme="base" css:content="base" />
```

Se non è presente nel tema il tag `<base />`, si può procedere così:

```
<before css:theme-children="head" css:content="base" />
```

Per eliminare dal tema tutte le risorse relative agli stili ed a JavaScript e copiarle invece dallo strumento di Plone `portal_css`:



```

<!-- elimina gli stili in head - questi vengono aggiunti nuovamente
includendo quelli di Plone -->

<drop theme="/html/head/link" />

<drop theme="/html/head/style" />

<!-- inserimento dei CSS di Plone -->

<after theme-children="/html/head" content="/html/head/link \
/html/head/style" />

```

Per copiare le risorse JavaScript di Plone:

```

<!-- inserimento degli script di Plone -->

<after theme-children="/html/head" content="/html/head/script" />

```

Per copiare la classe del tag <body /> (necessaria per il corretto funzionamento di alcune funzioni e di alcuni stili JavaScript di Plone):

```

<!-- Body -->

<merge attributes="class" css:theme="body" css:content="body" />

```

## Uso avanzato di portal\_css per la gestione del proprio CSS

I “registri delle risorse” di Plone, incluso lo strumento portal\_css, possono essere utilizzati per gestire i fogli di stile CSS. Questa opportunità offre diversi vantaggi rispetto al semplice collegamento ai propri fogli di stile nel template, come:

- Controllo dettagliato sull’ordine dei fogli di stile
- L’unione dei fogli di stile per ridurre il numero di download necessari per la presentazione di una pagina
- Compressione On-the-fly del foglio di stile (ad esempio con rimozione degli spazi bianchi)
- La possibilità di includere od escludere un foglio di stile in base ad un’espressione

E’ spesso desiderabile (e qualche volta assolutamente necessario) lasciare intatto il file del tema, ma è comunque possibile utilizzare portal\_css per gestire i fogli di stile. Il trucco consiste in:

- Registrare i propri stili del tema con lo strumento di Plone portal\_css (questo è normalmente meglio farlo quando si inserisce un tema in un pacchetto di Python - attualmente non esiste un modo per fare questo automaticamente per un tema importato da un file Zip o creato attraverso il web)
- Eliminare gli stili del tema con una regola e quindi
  - Includere tutti gli stili da Plone

Si potrebbero, per esempio, aggiungere le seguenti regole:

```

<drop theme="/html/head/link" />

<drop theme="/html/head/style" />

<!-- Pull in Plone CSS -->

<after theme-children="/html/head" content="/html/head/link \
/html/head/style" />

```

L'uso per il contenuto di un'espressione "or" nella regola <after /> indica che viene mantenuto l'ordine relativo degli elementi link e style.

Per registrare i fogli di stile al momento dell'installazione del prodotto mediante GenericSetup, bisogna usare il passo di importazione di cssregistry.xml nella directory del proprio GenericSetup profiles/default:

```
<?xml version="1.0"?>

<object name="portal\_css">

  <!-- Imposta le condizioni relative ai fogli di stile che non si
  vogliono includere -->

  <stylesheet
    expression="not:request/HTTP\_X\_THEME\_ENABLED \|| nothing"
    id="public.css"
  />

  <!-- aggiunge i nuovi fogli di stile -->

  <stylesheet title="" authenticated="False" cacheable="True"
    compression="safe" conditionalcomment="" cookable="True" enabled="on"
    expression="request/HTTP\_X\_THEME\_ENABLED \|| nothing"
    id="++theme++my.theme/css/styles.css" media="" rel="stylesheet"
    rendering="link"
    applyPrefix="True"
  />

</object>
```

C'è però una cosa importante da cui stare in guardia. I propri fogli di stile possono includere dei riferimenti ad URL relativi nella forma seguente:

```
background-image: url(../images/bg.jpg);
```

Se il foglio di stile è posizionato in una directory di risorse (ad esempio è registrato in portal\_css con l'id ++theme++my.theme/css/styles.css), questo funziona bene fino a quando il registro (e Zope) è in modalità di debug. L'URL relativo sarà tradotto dal browser in ++theme++my.theme/images/bg.jpg.

Tuttavia, è possibile che l'URL relativo non funzioni quando il Registro di sistema viene messo in modalità di produzione. Questo perché l'unione delle risorse cambia anche l'URL del foglio di stile in qualcosa del tipo:

```
/plone-site/portal_css/Suburst+Theme/merged-cachekey-1234.css
```

1. Per correggere questo, si deve impostare in cssregistry.xml il flag applyPrefix a true quando si installano le proprie risorse CSS. Esiste un flag corrispondente nell'interfaccia utente di portal\_css.

Qualche volta è utile mostrare alcuni CSS di Plone nel sito. Questo si può ottenere usando una regola Diazo <after /> o in modo simile copiare nel tema i CSS dall'<head /> generato da Plone. Si può utilizzare lo strumento portal\_css per disattivare i fogli di stile indesiderati.

Però, se si vuole che il sito sia usabile anche in modalità senza tema (per esempio in un URL separato), si potrebbe voler abilitare un insieme più ampio di stili quando Diazo non viene utilizzato. Per facilitare questa operazione, è possibile utilizzare le seguenti espressioni come condizioni nello strumento portal\_css (ed eventualmente in portal\_javascripts), in portal\_actions, nei page template, ed in altri posti che usano la sintassi delle espressioni TAL:

```
request/HTTP_X_THEME_ENABLED | nothing
```

L'espressione restituisce True se Diazo è attualmente abilitato, nel qual caso sarà impostato un header HTTP "X-Theme-Enabled".

Se in seguito si distribuisce il tema ad un server Web frontale come per esempio nginx, si può impostare lì lo stesso header della request per ottenere un egual risultato anche se plone.app.theming non è installato.

Utilizzare:

```
not: request/HTTP_X_THEME_ENABLED | nothing
```

per ‘nascondere’ un foglio di stile dal sito a cui è applicato il tema.

## 2.2 ZODB - un database nativo ad oggetti per Python

**Non schiacciare i tuoi oggetti nelle tabelle: memorizzali in un database ad oggetti.**

### 2.2.1 Panoramica

I programmi Python sono scritti seguendo il paradigma orientato agli oggetti. Si utilizzano gli oggetti che fanno riferimento l’un l’altro liberamente e possono essere di qualsiasi forma e dimensione: nessun oggetto deve aderire ad uno schema specifico e può contenere informazioni arbitrarie.

Memorizzare quegli oggetti nei database relazionali richiede di rinunciare alla libertà dei riferimenti e dello schema. I vincoli del modello relazionale riducono la capacità di scrivere codice orientato agli oggetti.

Lo ZODB è un database nativo ad oggetti, che memorizza i vostri oggetti e consente di lavorare con qualsiasi paradigma che si può esprimere in Python. In tal modo il vostro codice diventa più semplice, più affidabile e facile da capire.

Inoltre, non esiste un divario tra il database e il programma: nessun codice-colla per scrivere, nessuna mappatura da configurare. Date un’occhiata al tutorial per vedere, come è facile.

Alcune delle funzionalità che lo ZODB ti dà sono:

- persistenza trasparente degli oggetti Python
- supporto alle transazioni pienamente compatibile ACID (inclusi i savepoints)
- abilità di avere uno storico e la possibilità di annullare
- supporto efficiente per oggetti binari di grandi dimensioni (BLOB)
- sistemi di storage innestabili
- architettura scalabile

### 2.2.2 Documentazione

#### Tutorial

Questo tutorial ha lo scopo di guidare gli sviluppatori con un’introduzione passo-passo allo sviluppo un’applicazione che memorizza i dati nel ZODB.

#### Introduzione

Diamo un’occhiata a un semplice pezzo di codice che vogliamo modificare per poter utilizzare lo ZODB:

```
class Account(object):
    def __init__(self):
        self.balance = 0.0

    def deposit(self, amount):
        self.balance += amount

    def cash(self, amount):
        assert amount < self.balance
        self.balance -= amount
```

Questo codice definisce una semplice classe che mantiene il saldo di un conto bancario e fornisce due metodi per manipolare il saldo: deposito e prelievo di contanti.

### Installazione

Prima di poter utilizzare lo ZODB dobbiamo installarlo, usando `easy_install`. Notare che il vero nome del pacchetto è “ZODB3”:

```
$ easy_install ZODB3
...
$ python
>>> import ZODB
```

Ora lo ZODB è ora installato e può essere importato dalla vostra installazione Python.

Se non si ha a disposizione `easy_install` sul proprio sistema, seguire le [Istruzioni per l'installazione di EasyInstall](#).

Ci sono altri meccanismi di installazione disponibili per gestire l'installazione dei pacchetti Python. Questo tutorial assume che si stia utilizzando un'installazione base di Python e che lo ZODB sia installato globalmente.

### Configurazione

Quando un programma vuole utilizzare lo ZODB deve stabilire una connessione, come per qualsiasi altro database. Per lo ZODB abbiamo bisogno di 3 diverse parti: uno storage, un database e infine una connessione:

```
>>> from ZODB.FileStorage import FileStorage
>>> from ZODB.DB import DB
>>> storage = FileStorage('Data.fs')
>>> db = DB(storage)
>>> connection = db.open()
>>> root = connection.root()
```

Creiamo un storage chiamato `FileStorage`, che è l'attuale standard di memorizzazione usato praticamente da tutti. Esso tiene traccia di tutti i dati in un singolo file, come dichiarato dal primo parametro. Da questo storage creiamo un database e poi apriamo una connessione. Infine, recuperiamo l'oggetto `root` del database attraverso la connessione che abbiamo aperto.

### Memorizzare gli oggetti

Per memorizzare un oggetto nello ZODB, lo colleghiamo semplicemente a qualsiasi altro oggetto che è già presente nel database. Quindi, l'oggetto `root` funziona come un punto di partenza. L'oggetto `root` è un dizionario e si può iniziare a memorizzare gli oggetti direttamente da lì:

```
>>> root['account-1'] = Account()
>>> root['account-2'] = Account()
```

I framework come Zope creano solamente un singolo oggetto nella radice dello ZODB che rappresenta l'applicazione stessa e poi referenziano tutti gli altri oggetti da lì. Essi scelgono nomi come 'app' per il primo oggetto che posizionano nello ZODB.

## Transazioni

Ora abbiamo due oggetti posizionati nel oggetto root e nel nostro database. Tuttavia, essi non sono ancora memorizzati in modo persistente. Lo ZODB utilizza le transazioni e per rendere permanenti le modifiche, è quindi necessario il commit della transazione:

```
>>> import transaction
>>> transaction.commit()
>>> root.keys()
['account-1', 'account-2']
```

Ora possiamo stoppare e riavviare l'applicazione e guardare di nuovo all'oggetto root. Vedremo che le voci 'account-1' e 'account-2' sono ancora presenti e sono gli oggetti che abbiamo creato.

Gli oggetti che non sono ancora stati memorizzati nello ZODB non vengono rimossi da un abort.

Se l'applicazione apporta delle modifiche durante una transazione ma scopre che non vuole fare il commit di quelle modifiche, allora si può annullare la transazione e le modifiche vengono annullate per noi:

```
>>> del root['account-1']
>>> root.keys()
['account-2']
>>> transaction.abort()
>>> root.keys()
['account-1', 'account-2']
```

## Oggetti persistenti

Un ultimo aspetto che dobbiamo coprire sono gli stessi oggetti persistenti. Lo ZODB sarà lieto di memorizzare quasi qualsiasi oggetto Python che gli viene passato (ma non memorizzerà i file per esempio). Ma per capire quali oggetti sono stati modificati, lo ZODB ha bisogno che quegli oggetti collaborino con il database. In generale, per fare ciò basta ereditare da *persistent.Persistent*. Quindi la nostra classe di esempio sopra andrebbe modificata così:

```
import persistent

class Account(persistent.Persistent):
    # ... same code as above ...
```

Date un'occhiata alla documentazione di riferimento per saperne di più sulle regole di persistenza e sugli oggetti specializzati come i BTree.

## Sommario

Abbiamo visto come installare lo ZODB, come aprire un database nella nostra applicazione e come iniziare a memorizzare gli oggetti al suo interno. Abbiamo anche accennato ai due semplici comandi per le transazioni: commit e abort. La documentazione di riferimento contiene le sezione con maggiori informazioni sugli specifici argomenti.

### Guida alla programmazione dello ZODB

Questa guida si basa in gran parte sul lavoro di AM Kuchling che scrisse la guida originale nel 2002 e che fu pubblicata sotto la GNU Free Documentation License, Versione 1.1. Vedere l'appendice intitolata "GNU Free Documentation License" per ulteriori informazioni.

#### Contenuti

**Introduzione** Questa guida spiega come scrivere programmi Python che utilizzano Z Object Database (ZODB) e Zope Enterprise Objects (ZEO). L'ultima versione di questa guida è sempre disponibile su <http://www.zope.org/Wikis/ZODB/guide/index.html>.

**Cos'è lo ZODB?** Lo ZODB è un sistema di persistenza di oggetti Python. I linguaggi di programmazione persistenti forniscono strutture che scrivono automaticamente gli oggetti sul disco e li rileggono quando sono richiesti durante l'esecuzione del programma. Installando lo ZODB abbiamo aggiunto queste strutture a Python.

Certamente è possibile costruire un proprio sistema per rendere gli oggetti Python persistenti. Il punto iniziale di solito è il modulo `pickle`, per convertire gli oggetti in una rappresentazione di stringa, e vari moduli per i database, come i moduli `gdbm` o `bsddb` che forniscono dei modi per scrivere queste stringe sul disco e rileggerle. È semplice combinare il modulo `pickle` e un modulo per i database per memorizzare e recuperare gli oggetti, e in effetti il modulo `shelve`, incluso nella libreria standard di Python, fa proprio questo.

Lo svantaggio è che il programmatore deve gestire in modo esplicito gli oggetti, la loro lettura quando è necessario e la loro scrittura su disco quando non sono più richiesti. Lo ZODB gestisce gli oggetti per noi, scrivendoli su disco quando vengono modificati, e rimuovendoli dalla cache quando non vengono utilizzati per qualche tempo.

**OODBs vs. DB Relazionali** Un altro modo di vedere le cose è che lo ZODB è un database orientato agli oggetti specifico per Python (OODB). I database ad oggetti commerciali per C++ e Java spesso richiedono di saltare attraverso alcuni cerchi, come dover utilizzare uno speciale preprocessor o essere costretti ad evitare certi tipi di dati. Come vedremo, anche lo ZODB ha alcuni cerchi da saltare ma in confronto la naturalezza dello ZODB è stupefacente.

I database relazionali (RDB) sono molto più diffusi degli OODBs. I database relazionali memorizzano le informazioni in tabelle; una tabella è costituita da un numero qualsiasi di righe e ogni riga contiene diverse colonne di informazioni. (Le righe sono più formalmente chiamate relazioni, che è dove il termine "database relazionale" ha origine.)

Diamo un'occhiata a un esempio concreto. L'esempio viene dal mio lavoro di giorno per la Borsa MEMS, in una versione molto semplificata. Il lavoro è quello di tracciare le esecuzioni di processi, che sono liste di fasi di produzione da eseguire in un semiconduttore fab. Un'esecuzione appartiene ad un particolare utente, e ha un nome e un numero ID assegnato. Le esecuzioni consistono in un numero di operazioni; un'operazione è un singolo passo da eseguire, come depositare qualcosa su un wafer o incidere qualcosa su di esso.

Le operazioni possono avere dei parametri, i quali sono le informazioni aggiuntive richieste per eseguire un'operazione. Per esempio, se si sta depositando qualcosa su un wafer, avrete bisogno di sapere due cose: 1) cosa si sta depositando, e 2) quanto se ne dovrebbe depositare. Si potrebbe depositare 100 micron di ossido di silicio, o 1 micron di rame.

Mappare queste strutture in un database relazionale è semplice:

```
CREATE TABLE runs (
  int      run_id,
  varchar  owner,
  varchar  title,
  int      acct_num,
  primary key(run_id)
);
```

```

CREATE TABLE operations (
  int      run_id,
  int      step_num,
  varchar  process_id,
  PRIMARY KEY(run_id, step_num),
  FOREIGN KEY(run_id) REFERENCES runs(run_id),
);

CREATE TABLE parameters (
  int      run_id,
  int      step_num,
  varchar  param_name,
  varchar  param_value,
  PRIMARY KEY(run_id, step_num, param_name)
  FOREIGN KEY(run_id, step_num)
    REFERENCES operations(run_id, step_num),
);

```

In Python, si dovrebbero scrivere tre classi denominate `Run`, `Operation`, e `Parameter`. Non illustrerò il codice per definire queste classi, poiché non sarebbe interessante a questo punto. Ogni classe dovrebbe avere un singolo metodo con cui inizializzarle, un metodo `__init__()` che assegna i valori di default, come 0 o `None` ad ogni attributo della classe.

Non è difficile scrivere codice Python che crea una istanza `Run` e la valorizza con i dati presi dalle tabelle relazionali; con poco sforzo in più si potrebbe costruire un semplice tool, normalmente chiamato object-relational mapper (mappatore oggetto-relazione), per svolgere questo compito automaticamente. (Vedere <http://www.amk.ca/python/unmaintained/ordb.html> per un truccetto veloce sui Python object-relational mapper e vedere <http://www.python.org/workshops/1997-10/proceedings/shprentz.html> per l'implementazione più efficace di Joel Shprentz della stessa idea; A differenza del mio, il sistema di Shprentz è stato utilizzato realmente per un lavoro.)

Tuttavia è difficile rendere un object-relational mapper ragionevolmente veloce; un'implementazione da semplicità come la mia è abbastanza lenta perché deve fare molte query per accedere a tutti i dati di un oggetto. Gli object-relational mappers a maggiori prestazioni utilizzano delle cache di oggetti per migliorare le performance, eseguendo le query SQL solo quando veramente necessario.

Questo è utile se si vuole accedere all'improvviso all'operazione 123. Ma cosa succede se si vuole trovare tutte le operazioni dove uno step ha un parametro chiamato 'thickness' con valore uguale a 2.0? Nella versione relazionale, si hanno due scelte poco attraenti:

1. scrivere una query SQL specializzata per questo caso: `SELECT run_id FROM operations WHERE param_name = 'thickness' AND param_value = 2.0`

Se tali query sono comuni, si potrebbe finire per avere moltissime query specializzate. Se le tabelle del database dovessero venire modificate tutte queste query andrebbero riscritte.

2. un object-relational mapper non aiuta molto. Scansionare attraverso le operazioni significa che il mapper deve eseguire le query SQL richieste per leggere l'operazione #1, e poi un semplice ciclo Python dovrebbe verificare se qualcuno dei suoi step ha il parametro che stiamo cercando. Ripetere il tutto per l'operazione #2, #3 e così via. Questo comporta un enorme numero di query SQL, e quindi è incredibilmente lento.

Un database ad oggetti come lo ZODB semplicemente memorizza dei puntatori interni da oggetto a oggetto, per cui la lettura in un unico oggetto è molto più veloce che fare un mucchio di query SQL e assemblare i risultati. Quindi scansionare tutte le operazioni è ancora inefficiente ma non esageratamente inefficiente.

**Cos'è lo ZEO?** Lo ZODB viene fornito con diverse classi che implementano l'interfaccia `Storage`. Tali classi sono incaricate di gestire il lavoro di scrittura degli oggetti Python in un supporto fisico di archiviazione, che può essere un file sul disco (la classe `FileStorage`), un file BerkeleyDB (`BDBFullStorage`), un database relazionale (`DCOracleStorage`) o qualche altro tipo di supporto. ZEO aggiunge `ClientStorage`, un nuovo `Storage` che

non scrive su un supporto fisico ma semplicemente inoltra tutte le richieste attraverso la rete ad un server. Il server, che sta eseguendo un'istanza della classe `StorageServer`, semplicemente si comporta come un front-end per qualche classe fisica `Storage`. L'idea è abbastanza semplice, ma come vedremo in seguito in questo documento, apre molte possibilità.

**A proposito di questa guida** L'autore principale di questa guida lavora su un progetto che utilizza lo ZODB e ZEO come sua tecnologia principale di storage. Usiamo il ZODB per memorizzare le esecuzioni di processi e le operazioni, un catalogo di processi disponibili, informazioni sugli utenti, informazioni di contabilità e altri dati. Parte dell'obbiettivo di scrivere questo documento è rendere la nostra esperienza più ampiamente disponibile. Qualche volta abbiamo speso ore e persino giorni cercando di capire un problema e questa guida è un tentativo di raccogliere la conoscenza che abbiamo acquisito in modo che altri non debbano rifare gli stessi errori che abbiamo fatto noi durante l'apprendimento.

Il progetto ZODB dell'autore è descritto in un articolo disponibile qui: <http://www.amk.ca/python/writing/mx-architecture/>

Questo documento sarà sempre un work in progress. Se volete suggerire chiarimenti o altri argomenti, si prega di inviare i commenti a ZODB- [dev@zope.org](mailto:dev@zope.org).

**Riconoscimenti** Andrew Kuchling ha scritto la versione originale di questa guida, che ha fornito una tra le prime documentazioni sullo ZODB ai programmatori Python. La sua versione iniziale, è stato aggiornato nel tempo da Jeremy Hylton e Tim Peters.

Vorrei ringraziare le persone che hanno segnalato imprecisioni e bug, che hanno offerto suggerimenti sul testo, o proposto nuovi argomenti da coprire: Jeff Bauer, Willem Broekema, Thomas Guettler, Chris McDonough, George Runyan.

### Programmazione dello ZODB

**Installare lo ZODB** Lo ZODB è pacchettizzato utilizzando gli strumenti standard `distutils`.

**Requisiti** Avrete bisogno di Python 2.3 o superiore. Dal momento che il codice è pacchettizzato utilizzando `distutils`, è semplicemente questione di scompattare il pacchetto rilasciato, e poi lanciare il comando `python setup.py install`.

Avrete bisogno di un compilatore C per compilare i pacchetti, perché ci sono vari moduli di estensione scritti in C. Per gli utenti Windows sono disponibili gli installer dei binari.

**Installare i pacchetti** Scaricate il tarball ZODB contenente tutti i pacchetti sia per ZODB sia per ZEO da <http://www.zope.org/Products/ZODB3.3>. Vedere il file `README.txt` nel livello superiore delle directory per i dettagli su come compilare, testare e installare.

È possibile trovare informazioni su ZODB e le versioni più recenti dello ZODB nel Wiki su <http://www.zope.org/Wikis/ZODB>.

**Come funziona lo ZODB** Lo ZODB è concettualmente semplice. Le classi Python ereditano da una classe `persistent.Persistent` per diventare ZODB-compatibili. Le istanze di oggetti persistenti vengono caricate da un supporto di memorizzazione permanente, come un file su disco, quando il programma ha bisogno di loro, e rimangono nella cache in RAM. Lo ZODB intercetta le modifiche agli oggetti, in modo che quando uno statement come `obj.size = 1` viene eseguito, l'oggetto modificato viene segnato come "dirty" (sporco). A richiesta, ogni oggetto



sporco viene scritto sullo storage permanente; questo è chiamato committing di una transazione. Le transazioni possono anche essere annullate e ripristinate, il che scarta tutte le modifiche e gli oggetti sporchi vengono ripristinati al loro stato iniziale prima dell'inizio della transazione.

Il termine “transazione” ha uno specifico significato tecnico nella computer science. È estremamente importante che i contenuti di un database non vengano corrotti da fallimenti hardware o software, e la maggior parte dei database offrono protezione contro tali tipi di corruzioni supportando 4 utili proprietà: Atomicità, Consistenza, Isolamento e Durabilità. Nel gergo della computer science questi quattro termini sono chiamate collettivamente proprietà ACID, formando un acronimo con i loro nomi.

Lo ZODB fornisce tutte le proprietà ACID. Le definizioni delle proprietà ACID sono:

**Atomicità** significa che qualsiasi cambiamento ai dati fatto durante una transazione segue la regola del tutto-o-niente. O vengono applicate tutte le modifiche o nessuna di esse. Se un programma fa un sacco di modifiche e poi va in crash, il database non rimarrà parzialmente modificato, lasciando potenzialmente i dati in uno stato inconsistente; al contrario tutte le modifiche verranno rimosse. Ovviamente questo è un male, ma è meglio che avere delle modifiche parzialmente applicate che mettano il database in uno stato inconsistente.

**Consistenza** significa che ogni transazione esegue una trasformazione valida dello stato del database. Alcuni database, ma non lo ZODB, forniscono una varietà di controlli di consistenza nel database o nel linguaggio; per esempio, un database relazionale costringe le colonne ad essere di un particolare tipo e può forzare delle relazioni tra le tabelle. Più in generale, l'atomicità e l'isolamento rendono possibile alle applicazioni di fornire la consistenza.

**Isolamento** significa che due programmi o thread in esecuzione in due diverse transazioni non possono vedere le modifiche dell'altro fino a che non eseguono il commit delle loro transazioni

**Durabilità** significa che una volta che una transazione è stata committata, un fallimento successivo non causerà nessuna perdita o corruzione dei dati.

**Apriamo uno ZODB** There are 3 main interfaces supplied by the ZODB: `Storage`, `DB`, and `Connection` classes. The `DB` and `Connection` interfaces both have single implementations, but there are several different classes that implement the `Storage` interface.

- `Storage` classes are the lowest layer, and handle storing and retrieving objects from some form of long-term storage. A few different types of `Storage` have been written, such as `FileStorage`, which uses regular disk files, and `BDBFullStorage`, which uses Sleepycat Software's BerkeleyDB database. You could write a new `Storage` that stored objects in a relational database, for example, if that would better suit your application. Two example storages, `DemoStorage` and `MappingStorage`, are available to use as models if you want to write a new `Storage`.
- The `DB` class sits on top of a storage, and mediates the interaction between several connections. One `DB` instance is created per process.
- Finally, the `Connection` class caches objects, and moves them into and out of object storage. A multi-threaded program should open a separate `Connection` instance for each thread. Different threads can then modify objects and commit their modifications independently.

Preparing to use a ZODB requires 3 steps: you have to open the `Storage`, then create a `DB` instance that uses the `Storage`, and then get a `Connection` from the `DB` instance. All this is only a few lines of code:

```
from ZODB import FileStorage, DB

storage = FileStorage.FileStorage('/tmp/test-filestorage.fs')
db = DB(storage)
conn = db.open()
```

Note that you can use a completely different data storage mechanism by changing the first line that opens a `Storage`; the above example uses a `FileStorage`. In section zero, “How ZEO Works”, you'll see how ZEO uses this flexibility

to good effect.

**Using a ZODB Configuration File** ZODB also supports configuration files written in the ZConfig format. A configuration file can be used to separate the configuration logic from the application logic. The storages classes and the DB class support a variety of keyword arguments; all these options can be specified in a config file.

The configuration file is simple. The example in the previous section could use the following example:

```
<zodb>
  <filestorage>
    path /tmp/test-filestorage.fs
  </filestorage>
</zodb>
```

The `ZODB.config` module includes several functions for opening database and storages from configuration files.

```
import ZODB.config

db = ZODB.config.databaseFromURL('/tmp/test.conf')
conn = db.open()
```

The ZConfig documentation, included in the ZODB3 release, explains the format in detail. Each configuration file is described by a schema, by convention stored in a `component.xml` file. ZODB, ZEO, zLOG, and zdaemon all have schemas.

**Writing a Persistent Class** Making a Python class persistent is quite simple; it simply needs to subclass from the `Persistent` class, as shown in this example:

```
from persistent import Persistent

class User(Persistent):
    pass
```

The `Persistent` base class is a new-style class implemented in C.

For simplicity, in the examples the `User` class will simply be used as a holder for a bunch of attributes. Normally the class would define various methods that add functionality, but that has no impact on the ZODB's treatment of the class.

The ZODB uses persistence by reachability; starting from a set of root objects, all the attributes of those objects are made persistent, whether they're simple Python data types or class instances. There's no method to explicitly store objects in a ZODB database; simply assign them as an attribute of an object, or store them in a mapping, that's already in the database. This chain of containment must eventually reach back to the root object of the database.

As an example, we'll create a simple database of users that allows retrieving a `User` object given the user's ID. First, we retrieve the primary root object of the ZODB using the `root()` method of the `Connection` instance. The root object behaves like a Python dictionary, so you can just add a new key/value pair for your application's root object. We'll insert an `OOBTree` object that will contain all the `User` objects. (The `BTree` module is also included as part of Zope.)

```
dbroot = conn.root()

# Ensure that a 'userdb' key is present
# in the root
if not dbroot.has_key('userdb'):
    from BTrees.OOBTree import OOBTree
    dbroot['userdb'] = OOBTree()
```

```
userdb = dbroot['userdb']
```

Inserting a new user is simple: create the `User` object, fill it with data, insert it into the `BTree` instance, and commit this transaction.

```
# Create new User instance
import transaction

newuser = User()

# Add whatever attributes you want to track
newuser.id = 'amk'
newuser.first_name = 'Andrew' ; newuser.last_name = 'Kuchling'
...

# Add object to the BTree, keyed on the ID
userdb[newuser.id] = newuser

# Commit the change
transaction.commit()
```

The `transaction` module defines a few top-level functions for working with transactions. `commit()` writes any modified objects to disk, making the changes permanent. `abort()` rolls back any changes that have been made, restoring the original state of the objects. If you're familiar with database transactional semantics, this is all what you'd expect. `get()` returns a `Transaction` object that has additional methods like `note()`, to add a note to the transaction metadata.

More precisely, the `transaction` module exposes an instance of the `ThreadTransactionManager` transaction manager class as `transaction.manager`, and the `transaction` functions `get()` and `begin()` redirect to the same-named methods of `transaction.manager`. The `commit()` and `abort()` functions apply the methods of the same names to the `Transaction` object returned by `transaction.manager.get()`. This is for convenience. It's also possible to create your own transaction manager instances, and to tell `DB.open()` to use your transaction manager instead.

Because the integration with Python is so complete, it's a lot like having transactional semantics for your program's variables, and you can experiment with transactions at the Python interpreter's prompt:

```
>>> newuser
<User instance at 81b1f40>
>>> newuser.first_name           # Print initial value
'Andrew'
>>> newuser.first_name = 'Bob'   # Change first name
>>> newuser.first_name           # Verify the change
'Bob'
>>> transaction.abort()         # Abort transaction
>>> newuser.first_name           # The value has changed back
'Andrew'
```

**Rules for Writing Persistent Classes** Practically all persistent languages impose some restrictions on programming style, warning against constructs they can't handle or adding subtle semantic changes, and the ZODB is no exception. Happily, the ZODB's restrictions are fairly simple to understand, and in practice it isn't too painful to work around them.

The summary of rules is as follows:

- If you modify a mutable object that's the value of an object's attribute, the ZODB can't catch that, and won't mark the object as dirty. The solution is to either set the dirty bit yourself when you modify mutable objects, or

use a wrapper for Python's lists and dictionaries (`PersistentList`, `PersistentMapping`) that will set the dirty bit properly.

- Recent versions of the ZODB allow writing a class with `__setattr__()`, `__getattr__()`, or `__delattr__()` methods. (Older versions didn't support this at all.) If you write such a `__setattr__()` or `__delattr__()` method, its code has to set the dirty bit manually.
- A persistent class should not have a `__del__()` method. The database moves objects freely between memory and storage. If an object has not been used in a while, it may be released and its contents loaded from storage the next time it is used. Since the Python interpreter is unaware of persistence, it would call `__del__()` each time the object was freed.

Let's look at each of these rules in detail.

**Modifying Mutable Objects** The ZODB uses various Python hooks to catch attribute accesses, and can trap most of the ways of modifying an object, but not all of them. If you modify a `User` object by assigning to one of its attributes, as in `userobj.first_name = 'Andrew'`, the ZODB will mark the object as having been changed, and it'll be written out on the following `commit()`.

The most common idiom that *isn't* caught by the ZODB is mutating a list or dictionary. If `User` objects have a attribute named `friends` containing a list, calling `userobj.friends.append(otherUser)` doesn't mark `userobj` as modified; from the ZODB's point of view, `userobj.friends` was only read, and its value, which happened to be an ordinary Python list, was returned. The ZODB isn't aware that the object returned was subsequently modified.

This is one of the few quirks you'll have to remember when using the ZODB; if you modify a mutable attribute of an object in place, you have to manually mark the object as having been modified by setting its dirty bit to true. This is done by setting the `_p_changed` attribute of the object to true:

```
userobj.friends.append(otherUser)
userobj._p_changed = True
```

You can hide the implementation detail of having to mark objects as dirty by designing your class's API to not use direct attribute access; instead, you can use the Java-style approach of accessor methods for everything, and then set the dirty bit within the accessor method. For example, you might forbid accessing the `friends` attribute directly, and add a `get_friend_list()` accessor and an `add_friend()` modifier method to the class. `add_friend()` would then look like this:

```
def add_friend(self, friend):
    self.friends.append(otherUser)
    self._p_changed = True
```

Alternatively, you could use a ZODB-aware list or mapping type that handles the dirty bit for you. The ZODB comes with a `PersistentMapping` class, and I've contributed a `PersistentList` class that's included in my ZODB distribution, and may make it into a future upstream release of Zope.

**`__getattr__()`, `__delattr__()`, and `__setattr__()`** ZODB allows persistent classes to have hook methods like `__getattr__()` and `__setattr__()`. There are four special methods that control attribute access; the rules for each are a little different.

The `__getattr__()` method works pretty much the same for persistent classes as it does for other classes. No special handling is needed. If an object is a ghost, then it will be activated before `__getattr__()` is called.

The other methods are more delicate. They will override the hooks provided by `Persistent`, so user code must call special methods to invoke those hooks anyway.

The `__getattribute__()` method will be called for all attribute access; it overrides the attribute access support inherited from `Persistent`. A user-defined `__getattribute__()` must always give the `Persistent` base class a chance to handle special attribute, as well as `__dict__` or `__class__`. The user code should call

`_p_getattr()`, passing the name of the attribute as the only argument. If it returns `True`, the user code should call `Persistent`'s `__getattribute__()` to get the value. If not, the custom user code can run.

A `__setattr__()` hook will also override the `Persistent` `__setattr__()` hook. User code must treat it much like `__getattribute__()`. The user-defined code must call `_p_setattr()` first to all `Persistent` to handle special attributes; `_p_setattr()` takes the attribute name and value. If it returns `True`, `Persistent` handled the attribute. If not, the user code can run. If the user code modifies the object's state, it must assigned to `_p_changed`.

A `__delattr__()` hooks must be implemented the same was as a the last two hooks. The user code must call `_p_delattr()`, passing the name of the attribute as an argument. If the call returns `True`, `Persistent` handled the attribute; if not, the user code can run.

**`__del__()` methods** A `__del__()` method is invoked just before the memory occupied by an unreferenced Python object is freed. Because ZODB may materialize, and dematerialize, a given persistent object in memory any number of times, there isn't a meaningful relationship between when a persistent object's `__del__()` method gets invoked and any natural aspect of a persistent object's life cycle. For example, it is emphatically not the case that a persistent object's `__del__()` method gets invoked only when the object is no longer referenced by other objects in the database. `__del__()` is only concerned with reachability from objects in memory.

Worse, a `__del__()` method can interfere with the persistence machinery's goals. For example, some number of persistent objects reside in a `Connection`'s memory cache. At various times, to reduce memory burden, objects that haven't been referenced recently are removed from the cache. If a persistent object with a `__del__()` method is so removed, and the cache was holding the last memory reference to the object, the object's `__del__()` method will be invoked. If the `__del__()` method then references any attribute of the object, ZODB needs to load the object from the database again, in order to satisfy the attribute reference. This puts the object back into the cache again: such an object is effectively immortal, occupying space in the memory cache forever, as every attempt to remove it from cache puts it back into the cache. In ZODB versions prior to 3.2.2, this could even cause the cache reduction code to fall into an infinite loop. The infinite loop no longer occurs, but such objects continue to live in the memory cache forever.

Because `__del__()` methods don't make good sense for persistent objects, and can create problems, persistent classes should not define `__del__()` methods.

**Writing Persistent Classes** Now that we've looked at the basics of programming using the ZODB, we'll turn to some more subtle tasks that are likely to come up for anyone using the ZODB in a production system.

**Changing Instance Attributes** Ideally, before making a class persistent you would get its interface right the first time, so that no attributes would ever need to be added, removed, or have their interpretation change over time. It's a worthy goal, but also an impractical one unless you're gifted with perfect knowledge of the future. Such unnatural foresight can't be required of any person, so you therefore have to be prepared to handle such structural changes gracefully. In object-oriented database terminology, this is a schema update. The ZODB doesn't have an actual schema specification, but you're changing the software's expectations of the data contained by an object, so you're implicitly changing the schema.

One way to handle such a change is to write a one-time conversion program that will loop over every single object in the database and update them to match the new schema. This can be easy if your network of object references is quite structured, making it easy to find all the instances of the class being modified. For example, if all `User` objects can be found inside a single dictionary or `BTree`, then it would be a simple matter to loop over every `User` instance with a `for` statement. This is more difficult if your object graph is less structured; if `User` objects can be found as attributes of any number of different class instances, then there's no longer any easy way to find them all, short of writing a generalized object traversal function that would walk over every single object in a ZODB, checking each one to see if it's an instance of `User`.

Some OODBs support a feature called extents, which allow quickly finding all the instances of a given class, no matter where they are in the object graph; unfortunately the ZODB doesn't offer extents as a feature.

Missing parts: zeo.rst transactions.rst modules.rst links.rst gfdl.rst

### Bugs

Per riportare bug generici utilizzare [Launchpad bug tracker](#).

Tuttavia, lo ZODB è stato in giro molto più a lungo e quindi ci sono alcune risorse storiche per informazioni relative ai bug:

- il collettore Zope (con l'argomento *database*) su <http://collector.zope.org/Collectors/Zope>
- il tracciatore di bug di ZODB su sourceforge (molto più vecchio) su [http://sourceforge.net/tracker/?group\\_id=15628&atid=115628](http://sourceforge.net/tracker/?group_id=15628&atid=115628)

### Feature requests

Le richieste di funzionalità sono attualmente gestite un pò ad-hoc. Sentitevi liberi di scrivere un post sulla mailing list chiedendo di una funzionalità e - se non si vuole venire dimenticati - aggiungere un [blueprint in Launchpad](#)

Inoltre, in precedenza eravamo soliti gestire le proposte sul nostro vecchio wiki su <http://wiki.zope.org/ZODB/ListOfProposals/contents#ListOfProposals>

- Lo ZODB Book (in corso di stesura)

## 2.2.3 Downloads

Lo ZODB è distribuito come egg Python attraverso il [Python Package Index](#).

È possibile installare l'egg con il comando `easy_install` di `setuptools`:

```
$ easy_install ZODB3
```

## 2.2.4 La comunità e i contributi

Le discussioni avvengono sulla [mailing list degli sviluppatori ZODB](#).

Le segnalazioni di bug, le richieste di funzionalità, e i piani di rilascio sono fatti su [Launchpad](#).

Se desideri contribuire, saremo lieti di accettare qualsiasi lavoro di documentazione, aiuto agli altri sviluppatori e agli utenti della mailing list, segnalazione di bug, proposta o scrittura di codice.

ZODB è un progetto gestito dalla Fondazione Zope in modo da poter ottenere l'accesso in scrittura per contribuire direttamente - leggi le [informazioni per gli sviluppatori della fondazione Zope](#).

## 2.3 La guida completa alla Zope Component Architecture

**Autore** Baiju M

**Versione** 0.5.8

**Libro stampato** <http://www.lulu.com/content/1561045>

**Online PDF (en)** <http://www.muthukadan.net/docs/zca.pdf>

**Traduttore** Giacomo Spettoli <[giacomo.spettoli@gmail.com](mailto:giacomo.spettoli@gmail.com)>

Copyright (C) 2007,2008,2009 Baiju M <baiju.m.mail AT gmail.com>.

È permessa la copia, la redistribuzione e/o la modifica di questo documento secondo i termini della «GNU Free Documentation Licence», versione 1.3 o versioni successive pubblicate dalla Free Software Foundation.

Il codice presente in questo documento è soggetto alle condizioni della «Zope Public Licence», versione 2.1 (ZPL).

THE SOURCE CODE IN THIS DOCUMENT AND THE DOCUMENT ITSELF IS PROVIDED “AS IS” AND ANY AND ALL EXPRESS OR IMPLIED WARRANTIES ARE DISCLAIMED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF TITLE, MERCHANTABILITY, AGAINST INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

### Ringraziamenti

Molte persone mi hanno aiutato nella stesura di questo libro. La bozza iniziale fu revisionata dal mio collega Brad Allen. Quando annunciavi questo libro attraverso il mio blog, ricevevi molti commenti di incoraggiamento a procedere con questo lavoro. Kent Tenney modificò numerose parti di questo libro e riscrisse anche l'applicazione di esempio. Molti altri mi hanno inviato correzioni e commenti, inclusi Lorenzo Gil Sanchez, Michael Haubenwallner, Nando Quintana, Stephane Klein, Tim Cook, Kamal Gill and Thomas Herve. Lorenzo ha tradotto questo lavoro in Spagnolo e Stephane in Francese. Grazie a tutti!

## 2.3.1 Come iniziare

### Introduzione

Sviluppare un sistema software di grandi dimensioni è sempre molto complicato. È stato dimostrato che quando si tratta di grandi sistemi un buon approccio all'analisi, al design e alla programmazione è dato dalla programmazione orientata agli oggetti. Il design basato sui componenti e la programmazione a componenti stanno diventando molto popolari in questi giorni. L'approccio basato sui componenti aiuta a scrivere e a mantenere facilmente testabili con unit-test i sistemi software. Ci sono molti framework per il supporto al design a componenti in diversi linguaggi, alcuni sono persino indipendenti dal linguaggio. Due esempi sono il COM della Microsoft e XPCOM di Mozilla.

La **Zope Component Architecture (ZCA)** è un framework Python per il supporto al design e alla programmazione basati sui componenti. Essa è molto adatta allo sviluppo di sistemi software di grandi dimensioni scritti in Python. La ZCA non è specifica per il web application server Zope: può essere utilizzata per qualsiasi applicazione Python. Forse dovrebbe essere chiamata *Python Component Architecture*.

La ZCA tratta principalmente l'utilizzo efficace degli oggetti Python. I componenti sono oggetti riutilizzabili con interfacce introspezionabili. Un interfaccia è un oggetto che descrive come interagire con un particolare componente. In altre parole: un componente fornisce un interfaccia implementata in una classe (o in qualsiasi altro oggetto chiamabile). Non è tanto importante come un oggetto venga implementato, l'importante è che esso aderisca al contratto della sua interfaccia. Utilizzando la ZCA, è possibile suddividere la complessità di un sistema su molteplici componenti che cooperano tra loro. Essa aiuta a creare due principali tipi di componenti: gli *adapter* e le *utility*.

I tre pacchetti principali che compongono la ZCA sono:

- `zope.interface`: viene utilizzato per definire l'interfaccia di un componente
- `zope.event`: fornisce un semplice sistema di eventi
- `zope.component`: si occupa della creazione, della registrazione e del recupero dei componenti.

Notare che la ZCA non è un insieme di componenti, ma più propriamente serve a creare, registrare e recuperare i componenti. È bene ricordare inoltre che un *adapter* è una normale classe Python (o più in generale una factory) e una *utility* è un normale oggetto chiamabile Python.

Il framework ZCA fu sviluppato come parte del progetto Zope3. Come già anticipato, è un framework scritto esclusivamente in Python, così da poter essere utilizzato da qualsiasi tipo di applicazione Python. Attualmente i progetti Zope3, Zope2 e Grok utilizzano questo framework in maniera massiccia. Ci sono molti altri progetti che la utilizzano, inclusi progetti non legati al web <sup>1</sup>.

### Breve storia

Il progetto del framework ZCA iniziò nel 2011 come parte del progetto Zope3. Venne sviluppato basandosi sulle lezioni imparate durante lo sviluppo di grandi sistemi software utilizzando Zope2. Jim Fulton fu il project leader di questo progetto. Molte persone contribuirono al design e all'implementazione, inclusi ma non limitati a, Stephan Richter, Philipp von Weitershausen, Guido van Rossum (aka. Python BDFL), Tres Seaver, Phillip J Eby and Martijn Faassen.

Inizialmente la ZCA definiva dei componenti aggiuntivi, *services* e *views*, ma gli sviluppatori arrivarono alla conclusione che le utility potevano rimpiazzare i *service* e i multi-adapter potevano rimpiazzare le *view*. Oggi la ZCA ha un numero molto ridotto di tipi di componenti principali: *utility*, *adapter*, *subscriber*, e *handler*. In effetti, i *subscriber* e gli *handler* sono due particolari tipi di *adapter*.

Durante il ciclo di sviluppo di Zope3.2, Jim Fulton propose una grande semplificazione della ZCA <sup>2</sup>. Con questa semplificazione, fu creata una nuova singola interfaccia (*IComponentRegistry*) per la registrazione di componenti sia locali sia globali.

Il pacchetto `zope.component` ha una lunga lista di dipendenze, molte delle quali non erano necessarie per applicazioni non basate su Zope3. Durante il PyCon2007, Jim Fulton aggiunse a `setuptools` la funzionalità *extras\_require* per permettere di separare il nucleo della ZCA dalle funzionalità aggiuntive <sup>3</sup>.

Nel marzo del 2009, Tres Seaver rimosse poi le dipendenze da `zope.deferredimport` e `zope.proxy`.

Oggi, il progetto ZCA è un progetto indipendente con il proprio ciclo di rilasci e il proprio repository Subversion. Questo progetto sta diventando parte del più grande progetto del framework Zope <sup>4</sup>. In ogni caso le segnalazioni e i bug sono ancora tracciati come parte del progetto Zope3 <sup>5</sup>, e la mailing list principale `zope-dev` viene utilizzata per le discussioni sullo sviluppo <sup>6</sup>. C'è anche un'altra user-list generica per Zope3 (`zope3-users`) che può essere utilizzata per qualsiasi domanda sulla ZCA <sup>7</sup>.

### Installazione

Il pacchetto `zope.component` insieme ai pacchetti `zope.interface` e `zope.event` costituiscono il nucleo della Zope Component architecture. Essi forniscono le strutture per definire, registrare e recuperare i componenti. Il pacchetto `zope.component` e le sue dipendenze sono disponibili in formato egg sul Python Package Index (PyPI) <sup>8</sup>.

È possibile installare `zope.component` e le sue dipendenze utilizzando *easy\_install* <sup>9</sup>

```
$ easy_install zope.component
```

Questo comando scarica `zope.component` e le sue dipendenze da PyPI e installa il tutto nel vostro *Python path*.

---

<sup>1</sup> <http://wiki.zope.org/zope3/ComponentArchitecture>

<sup>2</sup> <http://wiki.zope.org/zope3/LocalComponentManagementSimplification>

<sup>3</sup> <http://peak.telecommunity.com/DevCenter/setuptools#declaring-dependencies>

<sup>4</sup> <http://docs.zope.org/zopeframework/>

<sup>5</sup> <https://bugs.launchpad.net/zope3>

<sup>6</sup> <http://mail.zope.org/mailman/listinfo/zope-dev>

<sup>7</sup> <http://mail.zope.org/mailman/listinfo/zope3-users>

<sup>8</sup> Repository dei pacchetti Python: <http://pypi.python.org/pypi>

<sup>9</sup> <http://peak.telecommunity.com/DevCenter/EasyInstall>



In alternativa, è possibile scaricare `zope.component` e le sue dipendenze da PyPI e poi installarle. Installare i pacchetti nell'ordine indicato sotto. Su sistemi Windows, potrebbero essere necessari i pacchetti binari di `zope.interface`:

1. `zope.interface`
2. `zope.event`
3. `zope.component`

Per installare questi pacchetti, dopo averli scaricati, è possibile usare il comando `easy_install` con gli eggs come argomento. (è possibile passare tutti gli egg come argomenti sulla stessa linea):

```
$ easy_install /path/to/zope.interface-3.x.x.tar.gz
$ easy_install /path/to/zope.event-3.x.x.tar.gz
$ easy_install /path/to/zope.component-3.x.x.tar.gz
```

È anche possibile installare questi pacchetti dopo averli estratti singolarmente. Ad esempio:

```
$ tar zxvf /path/to/zope.interface-3.x.x.tar.gz
$ cd zope.interface-3.x.x
$ python setup.py build
$ python setup.py install
```

Questi metodi installano la ZCA sul Python di *sistema*, nella cartella `site-packages`, ma questo potrebbe creare problemi. In un post sulla mailing list di Zope3, Jim Fulton sconsiglia l'utilizzo del Python di sistema<sup>10</sup>. In alternativa si può utilizzare `virtualenv` e/o `zc.buildout` per installare qualsiasi pacchetto Python. Questo metodo è adatto anche per il deploy.

## Come provare il codice

In Python ci sono due approcci per la configurazione di ambienti di lavoro isolati per lo sviluppo di applicazioni. Il primo è `virtualenv` creato da Ian Biking e l'altro è `zc.buildout` creato da Jim Fulton. È anche possibile utilizzare questi due pacchetti insieme. Con questi pacchetti è possibile installare `zope.component` e le altre dipendenze in un ambiente di lavoro isolato. Queste sono le buone pratiche per la sperimentazione di codice Python, e familiarizzare con questi strumenti tornerà utile quando si vorrà sviluppare e fare il deploy delle proprie applicazioni.

### virtualenv

Si può installare `virtualenv` utilizzando `easy_install`:

```
$ easy_install virtualenv
```

Poi si può creare un nuovo ambiente in questo modo:

```
$ virtualenv --no-site-packages myve
```

Questo comando crea un nuovo ambiente virtuale nella cartella `myve`. Ora, dall'interno della cartella `myve`, è possibile installare `zope.component` e le sue dipendenze utilizzando il comando `easy_install` che si trova dentro alla cartella `myve/bin`:

```
$ cd myve
$ ./bin/easy_install zope.component
```

Ora è possibile importare `zope.interface` e `zope.component` dal vostro nuovo interprete `python` disponibile dentro alla cartella `myve/bin`:

<sup>10</sup> <http://article.gmane.org/gmane.comp.web.zope.zope3/21045>

```
$ ./bin/python
```

Questo comando fornisce un prompt Python che può essere utilizzato per eseguire il codice di questo libro.

### zc.buildout

Utilizzando `zc.buildout` con la ricetta `zc.recipe.egg`, è possibile creare un interprete Python che ha a disposizione gli eggs specificati. Per prima cosa, installare `zc.buildout` utilizzando il comando `easy_install`. (è possibile farlo anche dentro all'ambiente virtuale). Per creare un nuovo buildout per fare esperimenti con gli egg Python, per prima cosa creare una cartella e inicializzarla utilizzando il comando `buildout init` :

```
$ mkdir mybuildout
$ cd mybuildout
$ buildout init
```

Ora la nuova cartella `buildout` è un buildout. Il file di configurazione di default per il buildout è `buildout.cfg`. Dopo l'inizializzazione, avrà questo contenuto:

```
[buildout]
parts =
```

Cambiamolo così:

```
[buildout]
parts = py

[py]
recipe = zc.recipe.egg
interpreter = python
eggs = zope.component
```

Ora lanciamo il comando `buildout` disponibile dentro alla cartella `mybuildout/bin` senza argomenti. Questo crea un nuovo interprete Python dentro alla cartella `mybuildout/bin`:

```
$ ./bin/buildout
$ ./bin/python
```

Questo comando farà apparire un prompt Python che può essere utilizzato per eseguire il codice di questo libro.

## 2.3.2 Un esempio

### Introduzione

Consideriamo come esempio un'applicazione commerciale per la registrazione degli ospiti di un hotel. Python può implementare questa applicazione in vari modi. Inizieremo dando una breve occhiata ad una possibile implementazione procedurale, e poi ci sposteremo verso un semplice approccio orientato agli oggetti. Mentre esamineremo l'approccio orientato agli oggetti, vedremo come potremo trarre beneficio dai pattern di design classici, *adapter* e *interface*. Questo ci porterà nel mondo della Zope Component Architecture.

### Approccio procedurale

In qualsiasi applicazione commerciale, una delle parti principali è la conservazione dei dati. Per semplicità, in questo esempio utilizzeremo un dizionario Python come sistema di storage. Creeremo degli id univoci per il dizionario e il valore associato ad ogni chiave sarà a sua volta un dizionario con i dettagli della prenotazione.

```
>>> bookings_db = {} #key: unique Id, value: details in a dictionary
```

Un'implementazione minimale richiede una funzione che verifichi i dettagli della prenotazione e una funzione di supporto che fornisca gli id univoci per le chiavi del dizionario di storage.

Possiamo generare un id univoco in questo modo:

```
>>> def get_next_id():
...     db_keys = bookings_db.keys()
...     if db_keys == []:
...         next_id = 1
...     else:
...         next_id = max(db_keys) + 1
...     return next_id
```

Come si può notare, l'implementazione della funzione *get\_next\_id* è molto semplice. La funzione prende una lista di chiavi e controlla una lista vuota. Se la lista è vuota questa è la nostra prima prenotazione, quindi restituiamo *1*. Se la lista non è vuota, aggiungiamo *1* al valore massimo della lista e lo restituiamo.

Ora utilizzeremo la funzione sopra per inserire degli elementi nel dizionario *bookings\_db*:

```
>>> def book_room(name, place):
...     next_id = get_next_id()
...     bookings_db[next_id] = {
...         'name': name,
...         'room': place
...     }
```

Un'applicazione per la gestione delle prenotazioni di un hotel ha bisogno di dati supplementari:

- numero di telefono
- opzioni della camera
- metodo di pagamento
- ...

e ha bisogno di codice per la gestione dei dati:

- cancellare una prenotazione
- aggiornare una prenotazione
- pagare una stanza
- rendere i dati persistenti
- assicurare la sicurezza dei dati
- ...

Se dovessimo continuare con l'esempio procedurale, dovremmo creare molte funzioni e dovremmo passare i dati avanti e indietro tra di loro. Man mano che i requisiti cambiano o aumentano, il codice diventa sempre più difficile da mantenere e i bug diventano più difficili da correggere.

Possiamo terminare qui la nostra discussione sull'approccio procedurale poichè sarà molto più facile fornire la persistenza dei dati, la flessibilità di design e la testabilità del codice utilizzando gli oggetti.

### Approccio orientato agli oggetti

La nostra discussione sul design orientato agli oggetti ci porta a introdurre la *classe*. La *classe* serve ad incapsulare i dati e il codice per gestirli.

La classe principale sarà il *FrontDesk*. La classe *FrontDesk* o verso cui delegherà la gestione, saprà come gestire i dati dell'hotel. Andremo a creare delle *istanze* di *FrontDesk* per applicare questa conoscenza al mestiere di gestire un hotel.

L'esperienza ha mostrato che incapsulare il codice e i dati attraverso gli oggetti porta ad un design più facile da comprendere, testare e modificare.

Vediamo i dettagli dell'implementazione di una classe *FrontDesk*:

```
>>> class FrontDesk(object):
...
...     def book_room(self, name, place):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': name,
...             'place': place
...         }
... 
```

In questa implementazione, l'oggetto *frontdesk* (istanza della classe *FrontDesk*) è in grado di gestire le prenotazioni. Possiamo usare questa classe così:

```
>>> frontdesk = FrontDesk()
>>> frontdesk.book_room("Jack", "Bangalore")
```

Qualsiasi progetto reale sarà soggetto a cambiamenti nei requisiti. In questo caso la gestione dell'hotel ha deciso che ogni ospite deve fornire anche un numero di telefono, quindi siamo costretti a cambiare il codice.

Possiamo raggiungere questo requisito aggiungendo un argomento al metodo *book\_room* che verrà aggiunto al dizionario dei valori:

```
>>> class FrontDesk(object):
...
...     def book_room(self, name, place, phone):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': name,
...             'place': place,
...             'phone': phone
...         }
... 
```

Oltre a migrare i dati verso il nuovo schema, ora dobbiamo anche cambiare le chiamate a *FrontDesk*. Se però noi astraiamo i dettagli dell'ospite in un oggetto e lo usiamo per la registrazione, i cambiamenti al codice vengono minimizzati. Così possiamo applicare i cambiamenti ai dettagli dell'ospite e le chiamate a *FrontDesk* non avranno bisogno di cambiamenti.

Così abbiamo:

```
>>> class FrontDesk(object):
...
...     def book_room(self, guest):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }
... 
```

Dobbiamo ancora cambiare il codice per rispondere ai cambiamenti dei requisiti. Sebbene questo sia inevitabile, il nostro obiettivo è quello di minimizzare questi cambiamenti, in modo da aumentare la manutenibilità.

---

**Note:** Quando si aggiunge del codice, è importante sentirsi liberi di apportare i cambiamenti senza paura di rompere l'applicazione. Il modo per avere i riscontri richiesti immediatamente è usare i test automatizzati. Con dei test ben scritti (e un buon sistema di controllo di versione) è possibile fare cambiamenti piccoli o grandi senza conseguenze. Una buona fonte di informazioni sulla filosofia della programmazione è il libro *Extreme Programming Explained* di Kent Beck.

Con l'introduzione dell'oggetto *ospite*, abbiamo risparmiato un pò di scrittura di codice e, cosa ancora più importante, l'astrazione fornita dall'oggetto *ospite* ha reso il sistema più semplice e più comprensibile. Come risultato, il codice è più facile da ri-fattorizzare e da mantenere.

## Il pattern adapter

Nelle applicazioni reali, l'oggetto *frontdesk* dovrebbe eseguire compiti come la cancellazione e l'aggiornamento delle prenotazioni. Nel design attuale dobbiamo passare l'oggetto *ospite* al *frontdesk* ogni volta che chiamiamo metodi come *cancel\_booking* e *update\_booking*.

Possiamo evitare facilmente questo vincolo se passiamo l'oggetto *ospite* al metodo `FrontDesk.__init__()`, rendendolo così un attributo dell'istanza:

```
>>> class FrontDeskNG(object):
...     def __init__(self, guest):
...         self.guest = guest
...
...     def book_room(self):
...         guest = self.guest
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }
```

In effetti la soluzione che abbiamo raggiunto è un pattern molto conosciuto, l'*adapter* (adattatore). In generale, un *adapter* contiene un oggetto *adattato*:

```
>>> class Adapter(object):
...     def __init__(self, adaptee):
...         self.adaptee = adaptee
```

Questo pattern sarà utile quando si avrà a che fare con i dettagli implementativi che dipendono da considerazioni riguardanti:

- il cambio dei requisiti del cliente
- requisiti di persistenza dei dati (ZODB, RDBMS, XML...)
- requisiti di output (HTML, PDF, testo semplice...)
- il linguaggio di markup usato per il rendering (ReST, Markdown, Textile...)

Grazie agli *adapters* e al *component registry* (registro dei componenti), la ZCA permette di cambiare i dettagli implementativi del codice attraverso la *configurazione*.

Come vedremo in questa sezione sugli *adapter* della ZCA, la possibilità di configurare i dettagli implementativi fornisce utili abilità:

- l'abilità di passare da una implementazione all'altra

- l'abilità di aggiungere implementazioni quando necessario
- aumenta il riutilizzo sia del codice precedente sia del codice della ZCA

Queste capacità portano il codice ad essere più flessibile, scalabile e riutilizzabile. Tuttavia c'è un costo per tutto ciò poiché il mantenimento del *component registry* aggiunge un livello di complessità all'applicazione. Se è noto a priori che un'applicazione non avrà mai bisogno di queste funzionalità, la ZCA non è necessaria.

Ora siamo pronti per iniziare il nostro studio della *Zope Component Architecture*, iniziando dalle interfacce.

### 2.3.3 Interfacce

#### Introduzione

Il file `README.txt`<sup>11</sup> nel percorso `path/to/zope/interface` definisce le interfacce in questo modo

```
Le interfacce sono oggetti che specificano (documentano) il comportamento verso l'esterno degli oggetti che le "forniscono". Un'interfaccia specifica il suo comportamento attraverso:
```

- la documentazione informale in una doc string.
- la definizione degli attributi
- le Invariants (invarianti), sono condizioni che devono essere verificate per un oggetto che fornisce l'interfaccia.

Il libro classico dell'ingegneria del software «Design Patterns»<sup>12</sup> della *Gang of Four* raccomanda di “Programmare per interfacce, non per implementazione”. Definire un'interfaccia formale è utile per la comprensione del sistema. In più, le interfacce portano a tutti i benefici della ZCA.

Un'interfaccia specifica le caratteristiche di un oggetto, il suo comportamento, le sue capacità. L'interfaccia descrive *cosa* può fare un oggetto, mentre per capire *come* lo fa, si dovrà guardare l'implementazione.

Due metafore usate comunemente per spiegare le interfacce sono i *contratti* e le *cianografie*, termini dei dizionari legale e architeturale per indicare un insieme di specifiche.

In alcuni linguaggi moderni come il Java, C#, VB.NET etc. le interfacce sono un aspetto esplicito del linguaggio. Siccome in Python mancano le interfacce, la ZCA le implementa con delle meta-classi da cui ereditare.

Di seguito un classico esempio di *hello world*:

```
>>> class Host(object):
...
...     def goodmorning(self, name):
...         """Say good morning to guests"""
...
...         return "Good morning, %s!" % name
```

Nel classe qui sopra abbiamo definito un metodo *goodmorning*. Se chiamiamo il metodo *goodmorning* su un oggetto istanza di questa classe, esso resituirà *Good morning, ...!*

```
>>> host = Host()
>>> host.goodmorning('Jack')
'Good morning, Jack!'
```

<sup>11</sup> L'albero del codice di Zope è pieno di file `README.txt` che offrono una meravigliosa documentazione.

<sup>12</sup> [http://en.wikipedia.org/wiki/Design\\_Patterns](http://en.wikipedia.org/wiki/Design_Patterns)

Qui *host* indica l'oggetto attuale utilizzato dal codice. Se si volesse esaminare i dettagli implementativi si dovrebbe accedere alla classe *Host*, o attraverso il codice sorgente o con uno strumento di documentazione delle API <sup>13</sup>.

Ora inizieremo ad utilizzare le interfacce della ZCA. Per la classe sopra si può specificare l'interfaccia così:

```
>>> from zope.interface import Interface
>>> class IHost(Interface):
...     def goodmorning(guest):
...         """Say good morning to guest"""
```

Come si può notare, l'interfaccia eredita da `zope.interface.Interface`. Questo utilizzo (abuso?) dello statement `class` del Python è come la ZCA definisce le interfacce. Il prefisso "I" per i nomi delle interfacce non è altro che un'utile convenzione.

### Dichiarazione delle interfacce

Abbiamo già visto come dichiarare un'interfaccia utilizzando `zope.interface` nella sezione precedente. Questa sezione spiegherà il concetto più nel dettaglio.

Si consideri questa interfaccia di esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> class IHost(Interface):
...     """A host object"""
...     name = Attribute("""Name of host""")
...     def goodmorning(guest):
...         """Say good morning to guest"""
```

L'interfaccia *IHost* ha due attributi, *name* e *goodmorning*. Si ricordi che, in Python, i metodi sono anche attributi delle classi. L'attributo *name* è definito utilizzando la classe `zope.interface.Attribute`. Quando si aggiunge un attributo *name* all'interfaccia *IHost*, non viene impostato un valore iniziale. Lo scopo di definire l'attributo *name* qui è puramente per indicare che qualsiasi implementazione di questa interfaccia dovrà fornire un attributo chiamato *name*. In questo caso, non viene nemmeno indicato di che tipo deve essere l'attributo! Si può passare una stringa di documentazione come primo argomento di *Attribute*.

L'altro attributo, *goodmorning*, è un metodo definito utilizzando la definizione di funzione. Si noti che *self* non è richiesto nelle interfacce, perché *self* è un dettaglio implementativo della classe. Ad esempio, un modulo potrebbe implementare questa interfaccia. Se un modulo implementa questa interfaccia, saranno definiti al suo interno un attributo *name* e una funzione *goodmorning* e la funzione *goodmorning* accetterà un argomento.

Ora vedremo come fare la connessione interfaccia-classe-oggetto. Gli oggetti sono la vera parte attiva e sono istanze delle classi. L'interfaccia è la vera definizione dell'oggetto, quindi la classe è solo un dettaglio implementativo. Ecco perché si dovrebbe sempre programmare un'interfaccia e non un'implementazione.

Ora si dovrebbe prendere familiarità con due ulteriori termini per comprendere altri concetti. Il primo è *provide* (fornisce) e l'altro è *implement* (implementa). Gli oggetti **forniscono** le interfacce e le classi **implementano** le interfacce. In altre parole, gli oggetti forniscono le interfacce che le loro classi implementano. Nel esempio sopra *host* (l'oggetto) fornisce *IHost* (l'interfaccia) e *Host* (la classe) implementa *IHost* (l'interfaccia). Un oggetto può fornire più di una interfaccia e anche una classe può implementare più di una interfaccia. Gli oggetti possono anche fornire delle interfacce direttamente in aggiunta alle interfacce implementate dalle loro classi.

<sup>13</sup> [http://en.wikipedia.org/wiki/Application\\_programming\\_interface](http://en.wikipedia.org/wiki/Application_programming_interface)

**Note:** Le classi sono i dettagli implementativi degli oggetti. In Python, le classi sono oggetti chiamabili, quindi perché altri oggetti chiamabili non possono implementare un'interfaccia? In effetti possono. Per qualsiasi *oggetto chiamabile* è possibile dichiarare che esso produce oggetti che forniscono una qualche interfaccia dichiarando che l'*oggetto chiamabile* implementa le interfacce. Gli *oggetti chiamabili* sono generalmente chiamati *factories* (fabbriche). Dato che le funzioni sono oggetti chiamabili, una funzione può essere un *implementatore* di una interfaccia.

---

### Implementare le interfacce

Per dichiarare che una classe implementa una particolare interfaccia, si utilizza la funzione `zope.interface.implements` nella definizione della classe.

Si consideri questo esempio, qui *Host* implementa *IHost*:

```
>>> from zope.interface import implements
>>> class Host(object):
...     implements(IHost)
...     name = u''
...     def goodmorning(self, guest):
...         """Say good morning to guest"""
...         return "Good morning, %s!" % guest
```

---

**Note:** se ci si chiede come lavori la funzione *implements*, si faccia riferimento al post del blog di James Henstridge (<http://blogs.gnome.org/jamesh/2005/09/08/python-class-advisors/>). Nella sezione degli adapter, si potrà vedere la funzione *adapts*, che lavora in maniera simile.

---

Siccome *Host* implementa *IHost*, le istanze di *Host* forniscono *IHost*. C'è qualche metodo di utilità per introspezionare le dichiarazioni. La dichiarazione può essere fatta anche fuori dalla classe. Se si omette *interface.implements(IHost)* nel esempio sopra, una volta che la classe è già stata definita, è possibile scrivere:

```
>>> from zope.interface import classImplements
>>> classImplements(Host, IHost)
```

### Esempio rivisitato

Ora, ritorniamo all'applicazione di esempio. Qui si vedrà come definire l'interfaccia dell'oggetto *frontdesk*:

```
>>> from zope.interface import Interface
>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...     def register():
...         """Register object's details"""
...     """
```

Per prima cosa abbiamo importato la classe *Interface* dal modulo `zope.interface`. Se si definisce una sottoclasse della classe *Interface* essa sarà una interfaccia dal punto di vista della Zope component architecture. Un'interfaccia può essere implementata, come abbiamo già visto, in una classe o in qualsiasi oggetto chiamabile.



L'interfaccia `frontdesk` definita qui è `IDesk`. La stringa di documentazione dell'interfaccia fornisce un'idea di un possibile oggetto. Nella definizione di un metodo in un'interfaccia, il primo argomento **non** deve essere `self`, poiché un'interfaccia non verrà mai istanziata e i suoi metodi non saranno mai chiamati. Al contrario, la classe interfaccia documenta semplicemente come dovrebbero apparire i metodi e gli attributi in qualsiasi classe normale che dichiari di implementarla, e il parametro `self` è un dettaglio implementativo che non ha bisogno di essere documentato.

Come sappiamo, un'interfaccia può anche specificare normali attributi:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute

>>> class IGuest(Interface):
...     ...
...     name = Attribute("Name of guest")
...     place = Attribute("Place of guest")
```

In questa interfaccia, l'oggetto ospite ha due attributi specificati con la documentazione. Un'interfaccia può anche specificare attributi e metodi insieme. Un'interfaccia può essere implementata da una classe, da un modulo o qualsiasi altro oggetto. Per esempio una funzione può creare dinamicamente un componente e restituirlo; in questo caso la funzione è un implementatore dell'interfaccia.

Ora sappiamo cos'è un'interfaccia e come definirla e usarla. Nel prossimo capitolo vedremo come utilizzare un'interfaccia per definire un componente adapter.

### Interfacce marker

Un'interfaccia può essere utilizzata per dichiarare che un particolare oggetto appartiene ad uno speciale tipo. Un'interfaccia senza attributi o metodi è chiamata *interfaccia marker*.

Ecco un esempio di *interfaccia marker*:

```
>>> from zope.interface import Interface

>>> class ISpecialGuest(Interface):
...     """A special guest"""
```

Questa interfaccia può essere utilizzata per indicare che un oggetto è uno speciale tipo di ospite.

### Invarianti

A volte c'è la necessità di utilizzare alcune regole per un componente che coinvolgono uno o più normali attributi. Questo tipo di regole sono chiamate *invariants* (invarianti). Si può utilizzare `zope.interface.invariant` per impostare delle *invarianti* sulle interfacce degli oggetti.

Si consideri un semplice esempio: c'è un oggetto *persona* con gli attributi *name*, *email* e *phone*. Come si potrebbe implementare una regola di validazione che imponga che almeno uno fra gli attributi *email* e *phone* debba esistere ma non necessariamente entrambi?

Per prima cosa bisogna costruire un oggetto chiamabile, o una semplice funzione o una istanza chiamabile di una classe come questa:

```
>>> def contacts_invariant(obj):
...     ...
...     if not (obj.email or obj.phone):
...         raise Exception(
...             "At least one contact info is required")
```

Poi si deve definire l'interfaccia dell'oggetto *person* in questo modo. Utilizzare la funzione `zope.interface.invariant` per definire l'invariante:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import invariant

>>> class IPerson(Interface):
...     name = Attribute("Name")
...     email = Attribute("Email Address")
...     phone = Attribute("Phone Number")
...
...     invariant(contacts_invariant)
```

Ora utilizzare il metodo `validateInvariants` dell'interfaccia per la validazione:

```
>>> from zope.interface import implements

>>> class Person(object):
...     implements(IPerson)
...
...     name = None
...     email = None
...     phone = None

>>> jack = Person()
>>> jack.email = u"jack@some.address.com"
>>> IPerson.validateInvariants(jack)
>>> jill = Person()
>>> IPerson.validateInvariants(jill)
Traceback (most recent call last):
...
Exception: At least one contact info is required
```

Come si può vedere l'oggetto *jack* è validato senza alcuna eccezione mentre l'oggetto *jill* non è stato validato dal vincolo invariante, così viene sollevata un'eccezione.

### 2.3.4 Adapters

#### Implementazione

In questa sezione verranno descritti gli adapter in dettaglio. La Zope Component Architecture, come abbiamo già visto, aiuta ad utilizzare efficacemente gli oggetti Python. I componenti adapter sono uno dei componenti di base utilizzati dalla ZCA. Gli adapter sono oggetti Python, ma con interfacce ben definite.

Per dichiarare che una classe è un adapter si utilizza la funzione `adapts` definita nel pacchetto `zope.component`. Ecco il nuovo adattatore *FrontDeskNG* con una dichiarazione esplicita di interfaccia:

```
>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
```

```

...     self.guest = guest
...
...     def register(self):
...         guest = self.guest
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

```

Quello che abbiamo definito qui è un *adapter* per *IDesk*, che adatta gli oggetti *IGuest*. L'interfaccia *IDesk* è implementata dalla classe *FrontDeskNG*. Quindi un'istanza di questa classe fornirà l'interfaccia *IDesk*

```

>>> class Guest(object):
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

```

Il *FrontDeskNG* è solo uno dei possibili adattatori. È possibile creare anche altri adapter che permettano di gestire le registrazioni degli ospiti diversamente.

## Registration

Per utilizzare questo componente adapter, bisogna registrarlo nel *component registry* anche conosciuto come *site manager*. Un site manager normalmente risiede in un sito. Il sito e il suo site manager saranno più importanti quando si svilupperanno applicazioni Zope3. Per ora ci interesseremo solo del *global site* e del *global site manager* (o *component registry*). Il global site manager risiede in memoria mentre un local site manager è persistente.

Per registrare il nostro componente, per prima cosa recuperiamo il global site manager:

```

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                     (IGuest,), IDesk, 'ng')

```

Per recuperare il global site manager, bisogna chiamare la funzione `getGlobalSiteManager` disponibile nel pacchetto `zope.component`. In effetti, il global site manager è disponibile anche come un attributo (*globalSiteManager*) del pacchetto `zope.component`. Quindi è anche possibile utilizzare direttamente l'attributo `zope.component.globalSiteManager`. Per registrare l'adapter nei componenti, come si può vedere sopra, si utilizza il metodo `registerAdapter` del *component registry*. Il primo argomento deve essere un classe/factory adapter. Il secondo argomento è una tupla di oggetti *adattati*, ad esempio l'oggetto che stiamo adattando. In questo esempio, stiamo adattando solo l'oggetto *IGuest*. Il terzo argomento è l'interfaccia implementata dal componente adapter. Il quarto argomento è opzionale ed è il nome di quel particolare adapter. Dato che abbiamo dato un nome a questo adapter, questo è un *named adapter*. Se non viene passato alcun nome allora questo sarà automaticamente una stringa vuota ('').

Nella registrazione sopra abbiamo passato l'interfaccia adattata e l'interfaccia fornita dall'adapter. Dato che questi

dettagli sono già stati specificati nella implementazione dell'adapter, non è necessario specificarli ancora. Infatti, avremmo potuto fare la registrazione così

```
>>> gsm.registerAdapter(FrontDeskNG, name='ng')
```

Ci sono alcune vecchie API per fare la registrazione che però andrebbero evitate. Le funzioni delle vecchie API iniziano con *provide*, ad es. *provideAdapter*, *provideUtility*, etc. Durante lo sviluppo di un'applicazione Zope3 è possibile utilizzare lo Zope configuration markup language (ZCML) per la registrazione dei componenti. In Zope3, i *local component* (o componenti persistenti) possono essere registrati dalla Zope Management Interface (ZMI) o anche in maniera programmatica.

**Note:** I local component sono componenti persistenti mentre i global component risiedono in memoria. I global component saranno registrati in base alla configurazione dell'applicazione. I local component sono caricati in memoria dal database all'avvio dell'applicazione.

---

### Recuperare un adapter

Il recupero dei componenti registrati dal component registry può essere effettuato con due funzioni disponibili nel pacchetto `zope.component`. Una di esse è *getAdapter* e l'altra è *queryAdapter*. Entrambe le funzioni accettano gli stessi parametri. Il metodo *getAdapter* solleva *ComponentLookupError* se la ricerca del componente fallisce, mentre *queryAdapter* restituirà *None*.

Si possono importare i due metodi in questo modo:

```
>>> from zope.component import getAdapter
>>> from zope.component import queryAdapter
```

Nella sezione precedente abbiamo registrato un componente per l'oggetto ospite (l'oggetto adattato) che fornisce l'interfaccia *IDesk* con nome 'ng'. Nella prima sezione di questo capitolo, abbiamo creato un oggetto ospite di nome *jack*.

Ecco come recuperare un componente che adatta l'interfaccia dell'oggetto *jack* (*IGuest*) e fornisce l'interfaccia *IDesk* e con il nome 'ng'. Qui sia *getAdapter* sia *queryAdapter* lavorano in maniera simile:

```
>>> getAdapter(jack, IDesk, 'ng')
<FrontDeskNG object at ...>
>>> queryAdapter(jack, IDesk, 'ng')
<FrontDeskNG object at ...>
```

Come si può vedere il primo argomento è l'oggetto da adattare, poi l'interfaccia che dovrebbe essere fornita dal componente e per ultimo il nome del componente adapter.

Se si prova a cercare un componente con un nome non registrato ma per lo stesso oggetto adattato e la stessa interfaccia, la ricerca fallirà. Ecco come si comportano i due metodi in questo caso:

Come si può vedere sopra, *getAdapter* ha sollevato un'eccezione *ComponentLookupError* mentre *queryAdapter* ha restituito *None* quando la ricerca è fallita.

Il terzo argomento, il nome di registrazione, è opzionale e se non viene passato il suo valore predefinito sarà una stringa vuota (''). Dal momento che non ci sono componenti registrati con una stringa vuota, *getAdapter* solleva *ComponentLookupError* e 'queryAdapter' restituirà *None*:

```
>>> getAdapter(jack, IDesk)
Traceback (most recent call last):
...
ComponentLookupError: ...
>>> reg = queryAdapter(jack, IDesk)
```

```
>>> reg is None
True
```

In questa sezione abbiamo imparato come registrare un semplice adapter e come recuperarlo dal component registry. Questo tipo di adapter sono chiamati single adapter (adattatore singolo) perché adattano solo un oggetto. Se un adapter adatta più di un oggetto, allora si chiama multi-adapter (multi-adattatore).

### Recuperare gli adapter tramite le interfacce

Gli adapter possono essere recuperati direttamente utilizzando le interfacce, ma questo funziona solo per gli adapter senza nome. Il primo argomento è l'oggetto adattato e il secondo è un argomento keyword. Se la ricerca dell'adapter fallisce, viene restituito il secondo argomento.

```
>>> IDesk(jack, alternate='default-output')
'default-output'
```

Il nome della keyword può anche essere ommesso:

```
>>> IDesk(jack, 'default-output')
'default-output'
```

Se il secondo argomento non viene passato allora viene sollevata *TypeError*:

```
>>> IDesk(jack)
Traceback (most recent call last):
...
TypeError: ('Could not adapt',
<Guest object at ...>,
<InterfaceClass __builtin__.IDesk>)
```

Qui *FrontDeskNG* viene registrato senza nome:

```
>>> gsm.registerAdapter(FrontDeskNG)
```

Ora la ricerca dell'adapter dovrebbe andare a buon fine:

```
>>> IDesk(jack, 'default-output')
<FrontDeskNG object at ...>
```

Quindi, per casi semplici, si può utilizzare l'interfaccia per recuperare il componente adapter.

### Il pattern adapter

Il concetto di adapter nella Zope Component Architecture è molto simile al classico *pattern adapter* che viene descritto nel libro «Design Pattern». L'intento degli adapter della ZCA è però più ampio di quello del *pattern adapter*. L'intento del *pattern adapter* è quello di convertire l'interfaccia di una classe in un'altra interfaccia che il client si aspetta. Questo permette di poter far lavorare insieme le classi che altrimenti sarebbero incompatibili a causa delle loro interfacce. Ma nella sezione *Motivation* del libro «Design Pattern», GoF dice: “Spesso l'adapter fornisce delle funzionalità che le classi adattate non forniscono”. L'adapter della ZCA è più incentrato sull'aggiunta di funzionalità che sulla creazione di una nuova interfaccia per un oggetto adattato. L'adapter della ZCA permette alle classi adapter di estendere le funzionalità aggiungendo nuovi metodi. (sarebbe interessante notare che l'*Adapter* era conosciuto come *Feature* nelle prime fasi del design della ZCA.)<sup>14</sup>

Nel paragrafo sopra c'è una citazione dal libro della “Gang of Four”, che finisce così “...che le **classi** adattate non forniscono”. Ma nella frase successiva io ho utilizzato “oggetto adattato” invece di “classe adattata”, poiché GoF descrive due varianti di adapter basati sull'implementazione. La prima è chiamata *class adapter* e l'altra *object adapter*.

<sup>14</sup> Discussione sulla rinomina delle Feature in Adapter: <http://mail.zope.org/pipermail/zope3-dev/2001-December/000008.html>

Un class adapter utilizza l'ereditarietà multipla per adattare un'interfaccia all'altra, mentre un object adapter fa affidamento sulla composizione degli oggetti. L'adapter della ZCA segue il pattern object adapter, il quale usa la delega come meccanismo di composizione. Il secondo principio di GoF a proposito del design orientato agli oggetti dice: "Favorite la composizione degli oggetti rispetto all'ereditarietà di classe". Per maggiori dettagli su questo argomento vi invito a leggere il libro «Design Pattern».

La cosa più interessante degli adapter della ZCA sono le interfacce esplicite per i componenti e il component registry. I componenti adapter della ZCA vengono registrati nel component registry e recuperati dagli oggetti client utilizzando le interfacce e il nome quando richiesto.

### 2.3.5 Utility

#### Introduzione

Ora conosciamo i concetti di interfaccia, adapter e component registry. A volte però sarebbe utile poter registrare un oggetto che non adatta nulla. Connessioni a database, parse XML, oggetti che restituiscono Id univoci, etc. sono tutti esempi di questo tipo di oggetti. Questo tipo di componenti forniti dalla ZCA sono chiamati *utility*.

Le utility sono solo oggetti che forniscono un'interfaccia e che vengono ricercati per interfaccia e per nome. Questo approccio crea un *global registry* attraverso il quale le interfacce possono essere registrate e accedute da diverse parti della nostra applicazione, senza bisogno di passare le istanze avanti e indietro come parametri.

Non è però consigliabile registrare tutte le istanze di componenti in questo modo. Si dovrebbero registrare solo i componenti che si vuole rendere rimpiazzabili.

#### Semplici utility

Una utility può essere registrata con un nome o senza nome. Una utility registrata con un nome è chiamata *named utility*, e la vedremo nella prossima sezione. Prima di implementare l'utility, come solito, definiamo la sua interfaccia. Ecco un'interfaccia `IGreeter` ("salutatore"):

```
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         """Say hello"""
```

Come anche un adapter, una utility può avere più di una implementazione. Ecco una possibile implementazione della interfaccia sopra:

```
>>> class Greeter(object):
...     implements(IGreeter)
...     def greet(self, name):
...         return "Hello " + name
```

La vera utility sarà un'istanza di questa classe. Per utilizzare questa utility dobbiamo registrarla per poterla richiedere in seguito utilizzando l'API della ZCA. Possiamo registrare un'istanza di questa classe (*utility*) utilizzando `registerUtility`:

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
```

```
>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter)
```

In questo esempio abbiamo registrato l'utility che fornisce l'interfaccia `IGreeter`. Si può ricercare l'interfaccia sia con `queryUtility` sia con `getUtility`:

```
>>> from zope.component import queryUtility
>>> from zope.component import getUtility

>>> queryUtility(IGreeter).greet('Jack')
'Hello Jack'

>>> getUtility(IGreeter).greet('Jack')
'Hello Jack'
```

Come si può vedere, gli adapter normalmente sono delle classi mentre le utility normalmente sono istanze di classi. L'istanza della classe utility viene creata solo una volta mentre le istanze dell'adapter vengono create dinamicamente quando vengono richieste.

### Named utility

Quando si registra un componente, come ad esempio un adapter, è possibile assegnargli un nome. Come detto nella precedente sezione, una utility registrata con un particolare nome è chiamata *named utility*.

Ecco come registrare l'utility `greeter` con un nome:

```
>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter, 'new')
```

In questo esempio abbiamo registrato l'utility con un nome fornendo l'interfaccia `IGreeter`. Ecco come ricercare l'interfaccia con `queryUtility` o con `getUtility`:

```
>>> from zope.component import queryUtility
>>> from zope.component import getUtility

>>> queryUtility(IGreeter, 'new').greet('Jill')
'Hello Jill'

>>> getUtility(IGreeter, 'new').greet('Jill')
'Hello Jill'
```

Come si può vedere qui, quando si fa un'interrogazione è necessario utilizzare il *name* come secondo argomento.

Chiamare la funzione `getUtility` senza un nome (come secondo argomento) è uguale a chiamare a chiamarla con una stringa vuota come nome poichè il valore predefinito del secondo argomento (keyword) è una stringa vuota. Poi il meccanismo di ricerca dei componenti proverà a trovare il componente il nome uguale alla stringa vuota e fallirà. Quando la ricerca del componente fallisce solleva l'eccezione `ComponentLookupError`. Si ricordi che non ritornerà un componente a caso con un'altro nome. Le funzioni di ricerca degli adapter, `getAdapter` e `queryAdapter` lavorano in maniera simile.

### Factory

Una *factory* è un componente utility che fornisce l'interfaccia `IFactory`.

Per creare una *factory*, per prima cosa definiamo l'interfaccia dell'oggetto:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IDatabase(Interface):
...     ...
...     def getConnection():
...         """Return connection object"""
```

Ecco una finta implementazione dell'interfaccia IDatabase:

```
>>> class FakeDb(object):
...     ...
...     implements(IDatabase)
...     ...
...     def getConnection(self):
...         return "connection"
```

Possiamo creare una factory utilizzando `zope.component.factory.Factory`:

```
>>> from zope.component.factory import Factory

>>> factory = Factory(FakeDb, 'FakeDb')
```

Ora possiamo registrarla in questo modo:

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> from zope.component.interfaces import IFactory
>>> gsm.registerUtility(factory, IFactory, 'fakedb')
```

Per utilizzare la factory, possiamo fare così:

```
>>> from zope.component import queryUtility
>>> queryUtility(IFactory, 'fakedb')()
<FakeDb object at ...>
```

C'è una scorciatoia per utilizzare una factory:

```
>>> from zope.component import createObject
>>> createObject('fakedb')
<FakeDb object at ...>
```

### 2.3.6 Adapter avanzati

In questo capitolo discuteremo di adapter avanzati come i multi-adapter, i subscription adapter e gli handler.

#### Multi adapter

Un semplice adapter normalmente adatta solo un oggetto, ma un adapter può adattare più di un oggetto. Se un adapter adatta più di un oggetto, è chiamato *multi-adapter*.

```
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.component import adapts
```



```

>>> class IAdapteeOne(Interface):
...     pass

>>> class IAdapteeTwo(Interface):
...     pass

>>> class IFunctionality(Interface):
...     pass

>>> class MyFunctionality(object):
...     implements(IFunctionality)
...     adapts(IAdapteeOne, IAdapteeTwo)
...
...     def __init__(self, one, two):
...         self.one = one
...         self.two = two

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerAdapter(MyFunctionality)

>>> class One(object):
...     implements(IAdapteeOne)

>>> class Two(object):
...     implements(IAdapteeTwo)

>>> one = One()
>>> two = Two()

>>> from zope.component import getMultiAdapter

>>> getMultiAdapter((one,two), IFunctionality)
<MyFunctionality object at ...>

>>> myfunctionality = getMultiAdapter((one,two), IFunctionality)
>>> myfunctionality.one
<One object at ...>
>>> myfunctionality.two
<Two object at ...>

```

### Subscription adapter

A differenza dei normali adapter, i *subscription adapter* vengono utilizzati quando vogliamo recuperare tutti gli adapter che adattano un oggetto a una particolare interfaccia. Un subscription adapter è anche conosciuto come *subscriber*.

Consideriamo un problema di validazione. Abbiamo degli oggetti e vogliamo verificare se essi aderiscono a qualche tipo di standard. Si definisce un'interfaccia di validazione:

```

>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IValidate(Interface):
...
...     def validate(ob):

```

```
...         """Determine whether the object is valid
...
...         Return a string describing a validation problem.
...         An empty string is returned to indicate that the
...         object is valid.
...         """
```

Magari abbiamo dei documenti:

```
>>> class IDocument(Interface):
...     summary = Attribute("Document summary")
...     body = Attribute("Document text")
>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body
```

Ora, potremmo voler specificare diverse regole di validazione per questi documenti. Per esempio, potremmo richiedere che la descrizione sia una linea singola:

```
>>> from zope.component import adapts
>>> class SingleLineSummary:
...     adapts(IDocument)
...     implements(IValidate)
...     def __init__(self, doc):
...         self.doc = doc
...     def validate(self):
...         if '\n' in self.doc.summary:
...             return 'Summary should only have one line'
...         else:
...             return ''
```

Oppure potremmo richiedere che il corpo del testo sia lungo al massimo 1000 caratteri:

```
>>> class AdequateLength(object):
...     adapts(IDocument)
...     implements(IValidate)
...     def __init__(self, doc):
...         self.doc = doc
...     def validate(self):
...         if len(self.doc.body) < 1000:
...             return 'too short'
...         else:
...             return ''
```

Possiamo registrare queste regole come subscription adapter:

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerSubscriptionAdapter(SingleLineSummary)
>>> gsm.registerSubscriptionAdapter(AdequateLength)
```

In seguito possiamo utilizzare i subscriber per validare gli oggetti:

```
>>> from zope.component import subscribers

>>> doc = Document("A\nDocument", "blah")
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['Summary should only have one line', 'too short']

>>> doc = Document("A\nDocument", "blah" * 1000)
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['Summary should only have one line']

>>> doc = Document("A Document", "blah")
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['too short']
```

## Handler

Gli handler sono delle fabbriche di subscription adapter che non restituiscono nulla. Essi infatti eseguono tutto il loro lavoro quando vengono chiamati. Gli handler tipicamente sono utilizzati per la gestione degli eventi e sono anche conosciuti come *event subscribers* o *event subscription adapter*.

Gli event subscriber sono diversi dagli altri subscription adapter per il fatto che il chiamante dell'event subscriber non si aspetta di interagire con loro in nessun modo diretto. Per esempio, un generatore di eventi non si aspetta di ricevere alcun valore di ritorno. Poiché i subscribers non hanno bisogno di fornire alcuna API ai loro chiamanti, è più naturale definirli con delle funzioni, piuttosto che con delle classi. Per esempio, in un sistema di gestione documentale potremmo voler registrare le date di creazione dei documenti:

```
>>> import datetime

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()
```

In questo esempio, abbiamo una funzione che prende un evento e svolge qualche operazione e in effetti non restituisce nulla. Questo è un caso speciale di subscription adapter che adatta un evento verso nulla. Tutto il lavoro è svolto quando la “factory” dell'adapter viene chiamata. I subscriber che non restituiscono niente sono chiamati “handler” e per registrarli ci sono delle API specifiche:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IDocumentCreated(Interface):
...
...     doc = Attribute("The document that was created")
```

```
>>> class DocumentCreated(object):
...     implements(IDocumentCreated)
...
...     def __init__(self, doc):
...         self.doc = doc
```

Dovremo anche cambiare la definizione del nostro handler:

Questo identifica l'handler come un adapter di eventi di tipo *IDocumentCreated*.

Andiamo a registrare l'handler:

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerHandler(documentCreated)
```

Ora possiamo creare un evento e utilizzare la funzione *handle* per chiamare gli handler registrati per l'evento:

```
>>> from zope.component import handle
>>> handle(DocumentCreated(doc))
>>> doc.created.__class__.__name__
'datetime'
```

### 2.3.7 Utilizzo della ZCA in Zope

La *Zope Component Architecture* viene utilizzata sia in Zope3 sia in Zope2. Questo capitolo tratterà l'utilizzo della ZCA in Zope.

#### ZCML

Lo **Zope Configuration Markup Language (ZCML)** è un sistema di configurazione basato su XML per la registrazione dei componenti. Così, invece di utilizzare le API Python per la registrazione, è possibile utilizzare lo ZCML. Sfortunatamente però l'utilizzo dello ZCML richiederà l'installazione di più pacchetti di dipendenze.

Per installare questi pacchetti lanciare:

```
$ easy_install "zope.component [zcml]"
```

Ecco come registrare un componente:

```
<configure xmlns="http://namespaces.zope.org/zope">
<adapter
    factory=".company.EmployeeSalary"
    provides=".interfaces.ISalary"
    for=".interfaces.IEmployee"
/>
```

Gli attributi *provides* e *for* sono opzionali, a patto che siano già stati dichiarati nell'implementazione del componente:

```
<configure xmlns="http://namespaces.zope.org/zope">
<adapter
    factory=".company.EmployeeSalary"
/>
```

Se si vuole registrare il componente come un named adapter, si può fornire un attributo *name*:

```
<configure xmlns="http://namespaces.zope.org/zope">
<adapter
  factory=".company.EmployeeSalary"
  name="salary"
/>
```

Anche le utility sono registrate in maniera simile:

```
<configure xmlns="http://namespaces.zope.org/zope">
<utility
  component=".database.connection"
  provides=".interfaces.IConnection"
/>
```

l'attributo *provides* è opzionale, a patto che sia stato dichiarato nell'implementazione:

```
<configure xmlns="http://namespaces.zope.org/zope">
<utility
  component=".database.connection"
/>
```

Se si vuole registrare il componente come named utility, si può fornire l'attributo *name*:

```
<configure xmlns="http://namespaces.zope.org/zope">
<utility
  component=".database.connection"
  name="Database Connection"
/>
```

Invece di utilizzare direttamente il componente, è possibile anche fornire la factory:

```
<configure xmlns="http://namespaces.zope.org/zope">
<utility
  factory=".database.Connection"
/>
```

## Overrides

Quando registriamo un componente utilizzando le API Python (i metodi *register\**), l'ultimo componente registrato rimpiazzerà il componente registrato in precedenza se entrambi sono registrati con gli stessi componenti. Per esempio, consideriamo l'esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IA(Interface):
...     pass

>>> class IP(Interface):
...     pass

>>> from zope.interface import implements
```

```
>>> from zope.component import adapts

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> class AP(object):
...     implements(IP)
...     adapts(IA)
...
...     def __init__(self, context):
...         self.context = context

>>> class AP2(object):
...     implements(IP)
...     adapts(IA)
...
...     def __init__(self, context):
...         self.context = context

>>> class A(object):
...     implements(IA)

>>> a = A()
>>> ap = AP(a)

>>> gsm.registerAdapter(AP)

>>> getAdapter(a, IP)
<AP object at ...>
```

Se registriamo un'altro adapter quello esistente viene rimpiazzato:

```
>>> gsm.registerAdapter(AP2)

>>> getAdapter(a, IP)
<AP2 object at ...>
```

Ma quando si registrano i componenti utilizzando ZCML, la seconda registrazione solleva un errore di conflitto. Questo è un suggerimento per noi, altrimenti ci sarebbe la possibilità di sovrascrivere le registrazioni per sbaglio e questo potrebbe portare a una maggiore difficoltà nel tracciare i bug nel sistema. Quindi l'utilizzo dello ZCML è una buona cosa per l'applicazione.

A volte avremo la necessità di sovrascrivere una registrazione esistente. Per questa evenienza lo ZCML fornisce la direttiva *includeOverrides*. Con questa direttiva possiamo scrivere le nostre sostituzioni in un file separato:

```
<includeOverrides file="overrides.zcml" />
```

### NameChooser

Posizione: *zope.app.container.contained.NameChooser*

Questo è un adapter che permette di scegliere un nome univoco per un oggetto all'interno di un contenitore.

La registrazione dell'adapter è simile a questa:

```
<adapter
  provides=".interfaces.INameChooser"
  for="zope.app.container.interfaces.IWriteContainer"
  factory=".contained.NameChooser"
/>
```

Dalla registrazione possiamo vedere che l'oggetto adattato è un *IWriteContainer* e che l'adapter fornisce *INameChooser*.

Questo adapter fornisce una funzionalità molto comoda per i programmatori Zope. La principale implementazione di *IWriteContainer* in Zope3 sono *zope.app.container.BTreeContainer* e *zope.app.folder.Folder*. Normalmente ereditiamo da queste implementazioni per creare le nostre classi contenitori. Se che non ci fosse nessuna interfaccia chiamata *INameChooser* e il relativo adapter, allora dovremmo implementare questa funzionalità per ogni implementazione separatamente.

### LocationPhysicallyLocatable

Posizione: *zope.location.traversing.LocationPhysicallyLocatable*

Questo adapter viene utilizzato frequentemente nelle applicazioni Zope3 ma normalmente viene chiamato attraverso un API in *zope.traversing.api*. (Qualche vecchio codice utilizza le funzioni di *zope.app.zapi* che è solo una redirectione aggiuntiva)

La registrazione dell'adapter è simile a questa:

```
<adapter
  factory="zope.location.traversing.LocationPhysicallyLocatable"
/>
```

L'interfaccia fornita e l'interfaccia adattata sono specificate nell'implementazione.

Ecco qui l'inizio dell'implementazione:

```
class LocationPhysicallyLocatable(object):
    """Provide location information for location objects
    """
    zope.component.adapts(ILocation)
    zope.interface.implements(IPhysicallyLocatable)
    ...
```

Normalmente, quasi tutti gli oggetti persistenti nell'applicazione Zope3 forniranno l'interfaccia *ILocation*. Questa interfaccia ha solo due attributi, `__parent__` e `__name__`. Il `__parent__` è il contenitore nella gerarchia degli oggetti e `__name__` è il nome dell'oggetto all'interno del contenitore.

L'interfaccia *IPhysicallyLocatable* ha 4 metodi: `getRoot`, `getPath`, `getName`, e `getNearestSite`.

- `getRoot` restituisce l'oggetto radice fisica
- `getPath` restituisce il percorso fisico verso l'oggetto in formato stringa
- `getName` restituisce l'ultimo segmento del percorso fisico
- `getNearestSite` restituisce il sito in cui è contenuto l'oggetto. Se l'oggetto è un sito, viene restituito l'oggetto stesso.

Quando si studia Zope3, si capisce che queste sono le cose importanti e quelle che vengono richieste più spesso. Per comprendere la bellezza di questo sistema bisogna vedere come Zope2 recupera l'oggetto radice fisica e come questo è implementato. Esiste un metodo chiamato `getPhysicalRoot` virtualmente per ogni oggetto contenitore.

### DefaultSized

Posizione: `zope.size.DefaultSized`

Questo adapter non è che l'implementazione di default dell'interfaccia `ISized`. Esso è registrato per tutti i tipi di oggetti. Se si vuole registrare questo adapter per una particolare interfaccia si dovrà sovrascrivere questa registrazione nella propria implementazione.

La registrazione dell'adapter è simile a questa:

```
<adapter
  for="*"
  factory="zope.size.DefaultSized"
  provides="zope.size.interfaces.ISized"
  permission="zope.View"
/>
```

Come si può vedere, l'interfaccia adattata è "\*" quindi può adattare qualsiasi tipo di oggetto.

`ISized` è una semplice interfaccia con due contratti di metodi:

```
class ISized(Interface):

    def sizeForSorting():
        """Returns a tuple (basic_unit, amount)

        Used for sorting among different kinds of sized objects.
        'amount' need only be sortable among things that share the
        same basic unit."""

    def sizeForDisplay():
        """Returns a string giving the size.
        """
```

Si può trovare un'altro adapter `ISized` registrato per `IZPTPage` nel pacchetto `zope.app.zptpage`.

### ZopeVersionUtility

Posizione: `zope.app.applicationcontrol.ZopeVersionUtility`

La registrazione è questa:

```
<utility
  component=".zopeversion.ZopeVersionUtility"
  provides=".interfaces.IZopeVersion" />
```

L'interfaccia fornita, `IZopeVersion`, ha solo un metodo chiamato `getZopeVersion`. Questo metodo restituisce una stringa contenente la versione di Zope (con eventualmente le informazione di SVN). L'implementazione di default, `ZopeVersionUtility`, prende le informazioni sulla versione da un file `version.txt` nella cartella `zope/app`. Se Zope è in esecuzione a partire da un checkout di Subversion, esso mostra l'ultimo numero di revisione. Se nessuno dei metodi sopra funziona allora restituisce *Development/Unknown*.

## 2.3.8 Caso di studio

---

**Note:** Questo capitolo non è ancora completo. Ogni suggerimento è benvenuto!

---



## Introduzione

Questo capitolo è un esempio di creazione di un'applicazione desktop utilizzando la libreria PyGTK per le GUI e la ZCA. Quest'applicazione utilizza anche due diversi tipi di meccanismi per la persistenza dei dati, un database ad oggetti (ZODB) e un altro database relazionale (SQLite). In ogni caso nella pratica solo uno storage può essere utilizzato per una particolare installazione. La ragione di utilizzare due diversi meccanismi di persistenza è la dimostrazione di come usare la ZCA per incollare tra loro i componenti. La maggior parte del codice di questa applicazione è legato a PyGTK.

Man mano che l'applicazione cresce si potranno utilizzare i componenti ZCA dovunque si desideri avere modularità e estensibilità. Si utilizzino invece direttamente oggetti Python dove non sono richieste queste due proprietà.

Non c'è differenza nell'utilizzo della ZCA per il web o per il desktop o per qualsiasi altro tipo di applicazione o framework. È preferibile seguire una convenzione per posizione dalla quale registrare i componenti. Questa applicazione utilizza una convenzione che permette di essere estesa posizionando delle registrazioni di componenti simili in moduli separati e in seguito importarli dal modulo di registrazione principale. In questa applicazione il modulo principale per la registrazione dei componenti è `register.py`.

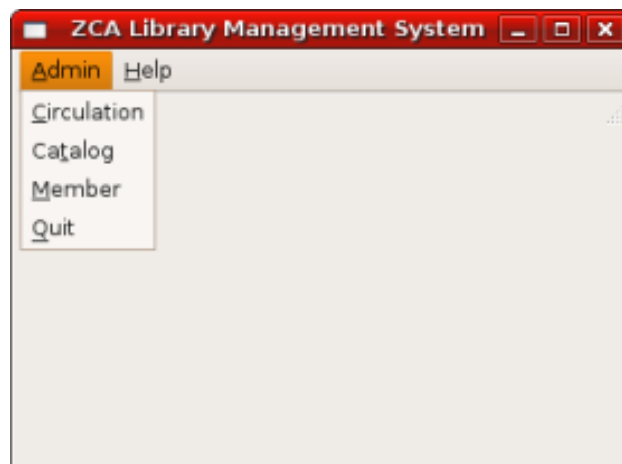
Il codice sorgente di questa applicazione può essere scaricato su <http://www.muthukadan.net/downloads/zcalib.tar.bz2>

## Casi d'uso

L'applicazione che ora andiamo a discutere è un sistema per la gestione di una biblioteca con funzionalità minimali. I requisiti possono essere riassunti così:

- aggiunta dei membri con un numero univoco e un nome
- aggiunta dei libri con il codice a barre, autore e titolo
- prestito dei libri
- restituzione dei libri

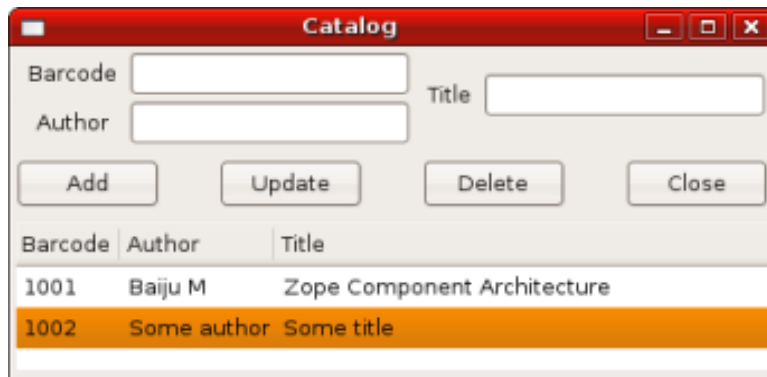
L'applicazione può essere disegnata in modo che le funzionalità principali possano essere utilizzate da una singola finestra. La finestra principale per accedere a tutte queste funzionalità potrebbe avere questo aspetto:



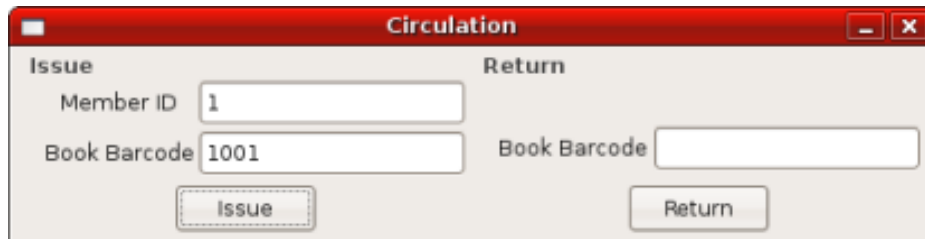
Dalla finestra Member, l'utente dovrebbe poter gestire i membri. Quindi dovrebbe essere possibile *aggiungere, modificare e eliminare* i membri come in figura sotto



Simile alla finestra dei membri, la finestra del catalogo permette all'utente di *aggiungere, modificare e eliminare* i libri:



La finestra dei movimenti dovrebbe gestire i prestiti e le restituzioni dei libri:



## Panoramica del codice PyGTK

Come si può vedere dal codice, la maggior parte del codice è legato a PyGTK e la sua struttura è molto simile per le diverse finestre. Le finestre di questa applicazione sono disegnate utilizzando il costruttore di GUI Glade. Si dovrebbero assegnare nomi sensati ai widget che si andrà ad utilizzare nel codice. Nella finestra principale, tutte le voci del menu hanno nomi come *circulation, catalog, member, quit* e *about*.

La classe `gtk.glade.XML` è utilizzata analizzare il file Glade e quindi creare gli oggetti widget dell'interfaccia grafica. Ecco come analizzare e accedere agli oggetti:

```
import gtk.glade
xmlobj = gtk.glade.XML('/path/to/file.glade')
widget = xmlobj.get_widget('widget_name')
```

Nel file `mainwindow.py`, si può vedere il codice:

```
curdir = os.path.abspath(os.path.dirname(__file__))
xml = os.path.join(curdir, 'glade', 'mainwindow.glade')
```

```
xmlobj = gtk.glade.XML(xml)

self.mainwindow = xmlobj.get_widget('mainwindow')
```

Il nome del widget della finestra principale è `mainwindow`. In maniera simile, gli altri widget vengono recuperati così:

```
circulation = xmlobj.get_widget('circulation')
member = xmlobj.get_widget('member')
quit = xmlobj.get_widget('quit')
catalog = xmlobj.get_widget('catalog')
about = xmlobj.get_widget('about')
```

Poi questi widget vengono connessi a certi eventi:

```
self.mainwindow.connect('delete_event', self.delete_event)
quit.connect('activate', self.delete_event)
circulation.connect('activate', self.on_circulation_activate)
member.connect('activate', self.on_member_activate)
catalog.connect('activate', self.on_catalog_activate)
about.connect('activate', self.on_about_activate)
```

Il `delete_event` è l'evento scatenato durante la chiusura della finestra utilizzando l'apposito bottone. L'evento `activate` è lanciato quando il menu viene selezionato. I widget sono connessi a certe funzioni di callback per certi eventi.

Possiamo vedere dal codice sopra che la finestra principale è connessa al metodo `on_delete_event` per il `delete_event`. Il widget `quit` è anche connesso allo stesso metodo per l'evento `activate`:

```
def on_delete_event(self, *args):
    gtk.main_quit()
```

La funzione di callback chiama semplicemente la funzione `main_quit`.

## Il codice

Ecco il file `zcalib.py`:

```
import registry
import mainwindow

if __name__ == '__main__':
    registry.initialize()
    try:
        mainwindow.main()
    except KeyboardInterrupt:
        import sys
        sys.exit(1)
```

Qui vengono importati due moduli: `registry` e `mainwindow`. Poi il registro viene analizzato e viene chiamata la funzione `main` di `mainwindow`. Se l'utente sta cercando di uscire dall'applicazione usando `Ctrl+C`, il sistema uscirà normalmente poiché abbiamo intercettato l'eccezione `KeyboardInterrupt`.

Questo è il modulo `registry.py`:

```
import sys
from zope.component import getGlobalSiteManager

from interfaces import IMember
```

```
from interfaces import IBook
from interfaces import ICirculation
from interfaces import IDbOperation

def initialize_rdb():
    from interfaces import IRelationalDatabase
    from relationaldatabase import RelationalDatabase
    from member import MemberRDbOperation
    from catalog import BookRDbOperation
    from circulation import CirculationRDbOperation

    gsm = getGlobalSiteManager()
    db = RelationalDatabase()
    gsm.registerUtility(db, IRelationalDatabase)

    gsm.registerAdapter(MemberRDbOperation,
                        (IMember,),
                        IDbOperation)

    gsm.registerAdapter(BookRDbOperation,
                        (IBook,),
                        IDbOperation)

    gsm.registerAdapter(CirculationRDbOperation,
                        (ICirculation,),
                        IDbOperation)

def initialize_odb():
    from interfaces import IObjectDatabase
    from objectdatabase import ObjectDatabase
    from member import MemberODbOperation
    from catalog import BookODbOperation
    from circulation import CirculationODbOperation

    gsm = getGlobalSiteManager()
    db = ObjectDatabase()
    gsm.registerUtility(db, IObjectDatabase)

    gsm.registerAdapter(MemberODbOperation,
                        (IMember,),
                        IDbOperation)

    gsm.registerAdapter(BookODbOperation,
                        (IBook,),
                        IDbOperation)

    gsm.registerAdapter(CirculationODbOperation,
                        (ICirculation,),
                        IDbOperation)

def check_use_relational_db():
    use_rdb = False
    try:
        arg = sys.argv[1]
        if arg == '-r':
            return True
    except IndexError:
```

```

        pass
    return use_rdb

def initialize():
    use_rdb = check_use_relational_db()
    if use_rdb:
        initialize_rdb()
    else:
        initialize_odb()

```

Diamo uno sguardo alla funzione `initialize` che stiamo chiamando dal modulo principale, `zcalib.py`. La funzione `initialize` per prima cosa controlla quale db è in uso, il database relazionale (RDB) o il database ad oggetti (ODB) e questo controllo è fatto nella funzione `check_use_relational_db`. Se è stata passata dalla linea di comando l'opzione `-r`, la funzione chiamerà `initialize_rdb` altrimenti `initialize_odb`. Se la funzione RDB viene chiamata, essa configurerà tutti i componenti legati a RDB altrimenti se viene chiamata la funzione ODB, verranno configurati tutti i componenti legati a ODB.

Ecco il file `mainwindow.py`:

```

import os
import gtk
import gtk.glade

from circulationwindow import circulationwindow
from catalogwindow import catalogwindow
from memberwindow import memberwindow

class MainWindow(object):

    def __init__(self):
        curdir = os.path.abspath(os.path.dirname(__file__))
        xml = os.path.join(curdir, 'glade', 'mainwindow.glade')
        xmlobj = gtk.glade.XML(xml)

        self.mainwindow = xmlobj.get_widget('mainwindow')
        circulation = xmlobj.get_widget('circulation')
        member = xmlobj.get_widget('member')
        quit = xmlobj.get_widget('quit')
        catalog = xmlobj.get_widget('catalog')
        about = xmlobj.get_widget('about')

        self.mainwindow.connect('delete_event', self.delete_event)
        quit.connect('activate', self.delete_event)

        circulation.connect('activate', self.on_circulation_activate)
        member.connect('activate', self.on_member_activate)
        catalog.connect('activate', self.on_catalog_activate)
        about.connect('activate', self.on_about_activate)

    def delete_event(self, *args):
        gtk.main_quit()

    def on_circulation_activate(self, *args):
        circulationwindow.show_all()

    def on_member_activate(self, *args):
        memberwindow.show_all()

    def on_catalog_activate(self, *args):

```

```
catalogwindow.show_all()

def on_about_activate(self, *args):
    pass

def run(self):
    self.mainwindow.show_all()

def main():
    mainwindow = MainWindow()
    mainwindow.run()
    gtk.main()
```

La funzione `main` crea un'istanza della classe `MainWindow` che inizializza tutti i widget.

Ecco qui `memberwindow.py`:

```
import os
import gtk
import gtk.glade

from zope.component import getAdapter

from components import Member
from interfaces import IDbOperation

class MemberWindow(object):

    def __init__(self):
        curdir = os.path.abspath(os.path.dirname(__file__))
        xml = os.path.join(curdir, 'glade', 'memberwindow.glade')
        xmlobj = gtk.glade.XML(xml)

        self.memberwindow = xmlobj.get_widget('memberwindow')
        self.number = xmlobj.get_widget('number')
        self.name = xmlobj.get_widget('name')
        add = xmlobj.get_widget('add')
        update = xmlobj.get_widget('update')
        delete = xmlobj.get_widget('delete')
        close = xmlobj.get_widget('close')
        self.treeview = xmlobj.get_widget('treeview')

        self.memberwindow.connect('delete_event', self.on_delete_event)
        add.connect('clicked', self.on_add_clicked)
        update.connect('clicked', self.on_update_clicked)
        delete.connect('clicked', self.on_delete_clicked)
        close.connect('clicked', self.on_delete_event)

        self.initialize_list()

    def show_all(self):
        self.populate_list_store()
        self.memberwindow.show_all()

    def populate_list_store(self):
        self.list_store.clear()
        member = Member()
        memberdboperation = getAdapter(member, IDbOperation)
```

```

members = memberdboperation.get()
for member in members:
    number = member.number
    name = member.name
    self.list_store.append((member, number, name,))

def on_delete_event(self, *args):
    self.memberwindow.hide()
    return True

def initialize_list(self):
    self.list_store = gtk.ListStore(object, str, str)
    self.treeview.set_model(self.list_store)
    tvcolumn = gtk.TreeViewColumn('Member Number')
    self.treeview.append_column(tvcolumn)

    cell = gtk.CellRendererText()
    tvcolumn.pack_start(cell, True)
    tvcolumn.add_attribute(cell, 'text', 1)

    tvcolumn = gtk.TreeViewColumn('Member Name')
    self.treeview.append_column(tvcolumn)

    cell = gtk.CellRendererText()
    tvcolumn.pack_start(cell, True)
    tvcolumn.add_attribute(cell, 'text', 2)

def on_add_clicked(self, *args):
    number = self.number.get_text()
    name = self.name.get_text()
    member = Member()
    member.number = number
    member.name = name
    self.add(member)
    self.list_store.append((member, number, name,))

def add(self, member):
    memberdboperation = getAdapter(member, IDbOperation)
    memberdboperation.add()

def on_update_clicked(self, *args):
    number = self.number.get_text()
    name = self.name.get_text()
    treeselection = self.treeview.get_selection()
    model, iter = treeselection.get_selected()
    if not iter:
        return
    member = self.list_store.get_value(iter, 0)
    member.number = number
    member.name = name
    self.update(member)
    self.list_store.set(iter, 1, number, 2, name)

def update(self, member):
    memberdboperation = getAdapter(member, IDbOperation)
    memberdboperation.update()

def on_delete_clicked(self, *args):

```

```
treeselection = self.treeview.get_selection()
model, iter = treeselection.get_selected()
if not iter:
    return
member = self.list_store.get_value(iter, 0)
self.delete(member)
self.list_store.remove(iter)

def delete(self, member):
    memberdboperation = getAdapter(member, IDbOperation)
    memberdboperation.delete()

memberwindow = MemberWindow()
```

Ecco qui *components.py*:

```
from zope.interface import implements

from interfaces import IBook
from interfaces import IMember
from interfaces import ICirculation

class Book(object):

    implements(IBook)

    barcode = ""
    title = ""
    author = ""

class Member(object):

    implements(IMember)

    number = ""
    name = ""

class Circulation(object):

    implements(ICirculation)

    book = Book()
    member = Member()
```

Ecco qui *interfaces.py*:

```
from zope.interface import Interface
from zope.interface import Attribute

class IBook(Interface):

    barcode = Attribute("Barcode")
    author = Attribute("Author of book")
    title = Attribute("Title of book")

class IMember(Interface):
```



```
number = Attribute("ID number")
name = Attribute("Name of member")

class ICirculation(Interface):

    book = Attribute("A book")
    member = Attribute("A member")

class IRelationalDatabase(Interface):

    def commit():
        pass

    def rollback():
        pass

    def cursor():
        pass

    def get_next_id():
        pass

class IObjectDatabase(Interface):

    def commit():
        pass

    def rollback():
        pass

    def container():
        pass

    def get_next_id():
        pass

class IDbOperation(Interface):

    def get():
        pass

    def add():
        pass

    def update():
        pass

    def delete():
        pass
```

Ecco qui *member.py*:

```
from zope.interface import implements
from zope.component import getUtility
```

```
from zope.component import adapts

from components import Member

from interfaces import IRelationalDatabase
from interfaces import IObjectDatabase
from interfaces import IMember
from interfaces import IDbOperation

class MemberRDbOperation(object):

    implements(IDbOperation)
    adapts(IMember)

    def __init__(self, member):
        self.member = member

    def get(self):
        db = getUtility(IRelationalDatabase)
        cr = db.cursor()
        number = self.member.number
        if number:
            cr.execute("""SELECT
                id,
                number,
                name
                FROM members
                WHERE number = ?""",
                (number,))
        else:
            cr.execute("""SELECT
                id,
                number,
                name
                FROM members""")
        rst = cr.fetchall()
        cr.close()
        members = []
        for record in rst:
            id = record['id']
            number = record['number']
            name = record['name']
            member = Member()
            member.id = id
            member.number = number
            member.name = name
            members.append(member)
        return members

    def add(self):
        db = getUtility(IRelationalDatabase)
        cr = db.cursor()
        next_id = db.get_next_id("members")
        number = self.member.number
        name = self.member.name
        cr.execute("""INSERT INTO members
                (id, number, name)
```

```

        VALUES (?, ?, ?)""" ,
        (next_id, number, name))
    cr.close()
    db.commit()
    self.member.id = next_id

def update(self):
    db = getUtility(IRelationalDatabase)
    cr = db.cursor()
    number = self.member.number
    name = self.member.name
    id = self.member.id
    cr.execute("""UPDATE members
                SET
                    number = ?,
                    name = ?
                WHERE id = ?""",
                (number, name, id))
    cr.close()
    db.commit()

def delete(self):
    db = getUtility(IRelationalDatabase)
    cr = db.cursor()
    id = self.member.id
    cr.execute("""DELETE FROM members
                WHERE id = ?""",
                (id,))
    cr.close()
    db.commit()

class MemberODbOperation(object):

    implements(IDbOperation)
    adapts(IMember)

    def __init__(self, member):
        self.member = member

    def get(self):
        db = getUtility(IObjectDatabase)
        zcalibdb = db.container()
        members = zcalibdb['members']
        return members.values()

    def add(self):
        db = getUtility(IObjectDatabase)
        zcalibdb = db.container()
        members = zcalibdb['members']
        number = self.member.number
        if number in [x.number for x in members.values()]:
            db.rollback()
            raise Exception("Duplicate key")
        next_id = db.get_next_id('members')
        self.member.id = next_id
        members[next_id] = self.member
        db.commit()

```

```
def update(self):
    db = getUtility(IObjectDatabase)
    zcalibdb = db.container()
    members = zcalibdb['members']
    id = self.member.id
    members[id] = self.member
    db.commit()

def delete(self):
    db = getUtility(IObjectDatabase)
    zcalibdb = db.container()
    members = zcalibdb['members']
    id = self.member.id
    del members[id]
    db.commit()
```

### PySQLite

### ZODB

### Conclusions

## 2.3.9 Riferimenti

### adaptedBy

Questa funzione permette di trovare le interfacce adattate.

- Posizione: `zope.component`
- Firma: `adaptedBy(object)`

Esempio:

```
>>> from zope.interface import implements
>>> from zope.component import adapts
>>> from zope.component import adaptedBy

>>> class FrontDeskNG(object):
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest

>>> adaptedBy(FrontDeskNG)
(<InterfaceClass __builtin__.IGuest>,)
```

### adapter

Qualsiasi tipo di oggetto può essere un adattatore, è possibile utilizzare il decoratore *adapter* per dichiarare che un oggetto chiamabile adatta qualche interfaccia (o classe)

- Posizione: `zope.component`

- Firma: *adapter(\*interfaces)*

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implementer
>>> from zope.component import adapter
>>> from zope.interface import implements

>>> class IJob(Interface):
...     """A job"""

>>> class Job(object):
...     implements(IJob)

>>> class IPerson(Interface):
...
...     name = Attribute("Name")
...     job = Attribute("Job")

>>> class Person(object):
...     implements(IPerson)
...
...     name = None
...     job = None

>>> @implementer(IJob)
... @adapter(IPerson)
... def personJob(person):
...     return person.job

>>> jack = Person()
>>> jack.name = "Jack"
>>> jack.job = Job()
>>> personJob(jack)
<Job object at ...>
```

## adapts

Questa funzione permette di dichiarare le interfacce adattate dall'adapter

- Posizione: `zope.component`
- Firma: *adapts(\*interfaces)*

Esempio:

```
>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
... 
```

```
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }
```

### alsoProvides

Dichiara le interfacce fornite direttamente da un oggetto. Gli argomenti dopo l'oggetto sono una o più interfacce. Le interfacce fornite vengono aggiunte alle interfacce già dichiarate per l'oggetto.

- Posizione: `zope.interface`
- Firma: `alsoProvides(object, *interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.interface import alsoProvides

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...     college = Attribute("Name of college")

>>> class Person(object):
...     implements(IDesk)
...     name = u""

>>> jack = Person()
>>> jack.name = "Jack"
>>> jack.college = "New College"
>>> alsoProvides(jack, IStudent)
```

Si può testare così:

```
>>> from zope.interface import providedBy
>>> IStudent in providedBy(jack)
True
```

### Attribute

Con questa classe è possibile definire i normali attributi di una interfaccia.

- Posizione: `zope.interface`
- Firma: `Attribute(name, doc='')`
- Vedi anche: *Interface*

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IPerson(Interface):
...
...     name = Attribute("Name of person")
...     email = Attribute("Email Address")
```

### classImplements

Dichiara le interfacce aggiuntive implementate dalle istanze di una classe. Gli argomenti dopo la classe sono una o più interfacce. Le interfacce fornite vengono aggiunte alle interfacce già dichiarate.

- Posizione: `zope.interface`
- Firma: `classImplements(cls, *interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.interface import classImplements

>>> class IPerson(Interface):
...
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...
...     college = Attribute("Name of college")

>>> class Person(object):
...
...     implements(IDesk)
...     name = u""
...     college = u""

>>> classImplements(Person, IStudent)
>>> jack = Person()
>>> jack.name = "Jack"
>>> jack.college = "New College"

Si può testare così:

>>> from zope.interface import providedBy
>>> IStudent in providedBy(jack)
True
```

### classImplementsOnly

Dichiara le sole interfacce implementate dalle istanze di una classe. Gli argomenti dopo la classe sono una o più interfacce. Le interfacce fornite rimpiazzano le dichiarazioni precedenti.

- Posizione: `zope.interface`

- Firma: `classImplementsOnly(cls, *interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.interface import classImplementsOnly

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...     college = Attribute("Name of college")

>>> class Person(object):
...     implements(IPerson)
...     college = u""

>>> classImplementsOnly(Person, IStudent)
>>> jack = Person()
>>> jack.college = "New College"
```

Si può testare così:

```
>>> from zope.interface import providedBy
>>> IPerson in providedBy(jack)
False
>>> IStudent in providedBy(jack)
True
```

### classProvides

Normalmente se una classe implementa una particolare interfaccia, l'istanza di questa classe fornirà l'interfaccia implementata da questa classe. Se però si vuole che la classe stessa fornisca un'interfaccia, si può utilizzare questa funzione.

- Posizione: `zope.interface`
- Firma: `classProvides(*interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import classProvides

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class Person(object):
...     classProvides(IPerson)
...     name = u"Jack"
```



Si può testare così:

```
>>> from zope.interface import providedBy
>>> IPerson in providedBy(Person)
True
```

## ComponentLookupError

Questa è l'eccezione che viene sollevata quando una ricerca di un componente fallisce.

Esempio:

```
>>> class IPerson(Interface):
...     name = Attribute("Name of person")
>>> person = object()
>>> getAdapter(person, IPerson, 'not-exists')
Traceback (most recent call last):
...
ComponentLookupError: ...
```

## createObject

Crea un oggetto usando una factory.

Cerca la named factory nel sito corrente e la chiama con i parametri forniti. Se non può essere trovata alcuna factory, viene sollevata l'eccezione `ComponentLookupError` altrimenti restituisce l'oggetto creato.

Può essere fornito come argomento keyword un context per forzare la ricerca della factory in una posizione diversa dal sito corrente. Ovviamente questo significa che è impossibile passare un argomento keyword alla factory chiamato "context".

- Posizione: `zope.component`
- Firma: `createObject(factory_name, *args, **kwargs)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IDatabase(Interface):
...     def getConnection():
...         """Return connection object"""

>>> class FakeDb(object):
...     implements(IDatabase)
...     def getConnection(self):
...         return "connection"

>>> from zope.component.factory import Factory

>>> factory = Factory(FakeDb, 'FakeDb')
```

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> from zope.component.interfaces import IFactory
>>> gsm.registerUtility(factory, IFactory, 'fakedb')

>>> from zope.component import createObject
>>> createObject('fakedb')
<FakeDb object at ...>
```

### Declaration

Non deve essere usata direttamente.

### directlyProvidedBy

Questa funzione restituirà le interfacce fornite direttamente dall'oggetto passato come argomento.

- Posizione: `zope.interface`
- Firma: `directlyProvidedBy(object)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...     college = Attribute("Name of college")

>>> class ISmartPerson(Interface):
...     pass

>>> class Person(object):
...     implements(IPerson)
...     name = u""

>>> jack = Person()
>>> jack.name = u"Jack"
>>> jack.college = "New College"
>>> alsoProvides(jack, ISmartPerson, IStudent)

>>> from zope.interface import directlyProvidedBy

>>> jack_dp = directlyProvidedBy(jack)
>>> IPerson in jack_dp.interfaces()
False
>>> IStudent in jack_dp.interfaces()
True
>>> ISmartPerson in jack_dp.interfaces()
True
```

## directlyProvides

Dichiara le interfacce fornite direttamente da un oggetto. Gli argomenti dopo l'oggetto sono una o più interfacce. Le interfacce fornite rimpiazzano le interfacce già dichiarate in precedenza dall'oggetto.

- Posizione: `zope.interface`
- Firma: `directlyProvides(object, *interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...     college = Attribute("Name of college")

>>> class ISmartPerson(Interface):
...     pass

>>> class Person(object):
...     implements(IPerson)
...     name = u""

>>> jack = Person()
>>> jack.name = u"Jack"
>>> jack.college = "New College"
>>> alsoProvides(jack, ISmartPerson, IStudent)

>>> from zope.interface import directlyProvidedBy

>>> jack_dp = directlyProvidedBy(jack)
>>> ISmartPerson in jack_dp.interfaces()
True
>>> IPerson in jack_dp.interfaces()
False
>>> IStudent in jack_dp.interfaces()
True
>>> from zope.interface import providedBy

>>> ISmartPerson in providedBy(jack)
True

>>> from zope.interface import directlyProvides
>>> directlyProvides(jack, IStudent)

>>> jack_dp = directlyProvidedBy(jack)
>>> ISmartPerson in jack_dp.interfaces()
False
>>> IPerson in jack_dp.interfaces()
False
>>> IStudent in jack_dp.interfaces()
True
```

```
>>> ISmartPerson in providedBy(jack)
False
```

### getAdapter

Recupera un adapter per un oggetto verso una specifica interfaccia. Restituisce un adapter che può adattare l'oggetto all'interfaccia. Se non può essere trovato alcun adapter, solleva `ComponentLookupError`.

- Posizione: `zope.interface`
- Firma: `getAdapter(object, interface=Interface, name=u'', context=None)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
```

```

>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                     (IGuest,), IDesk, 'ng')

>>> getAdapter(jack, IDesk, 'ng')
<FrontDeskNG object at ...>

```

### getAdapterInContext

Al posto di questa funzione, utilizzare l'argomento *context* della funzione *getAdapter*.

- Posizione: `zope.component`
- Firma: `getAdapterInContext(object, interface, context)`
- Vedi anche: `queryAdapterInContext`

Esempio:

```

>>> from zope.component.globalregistry import BaseGlobalComponents
>>> from zope.component import IComponentLookup
>>> sm = BaseGlobalComponents()

>>> class Context(object):
...     def __init__(self, sm):
...         self.sm = sm
...     def __conform__(self, interface):
...         if interface.isOrExtends(IComponentLookup):
...             return self.sm

>>> context = Context(sm)

>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,

```

```
...     'phone': guest.phone
...     }

>>> class Guest(object):
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> sm.registerAdapter(FrontDeskNG,
...                    (IGuest,), IDesk)

>>> from zope.component import getAdapterInContext

>>> getAdapterInContext(jack, IDesk, sm)
<FrontDeskNG object at ...>
```

### getAdapters

Cerca tutti gli adapter corrispondenti per degli oggetti e per una interfaccia fornita. Restituisce una lista di adapter che corrispondono. Se un adapter ha un nome, viene restituito solo l'adapter più specifico.

- Posizione: `zope.component`
- Firma: `getAdapters(objects, provided, context=None)`

Esempio:

```
>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> jack = Guest("Jack", "Bangalore")
```

```

>>> jack_frontdesk = FrontDeskNG(jack)

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerAdapter(FrontDeskNG, name='ng')

>>> from zope.component import getAdapters
>>> list(getAdapters((jack,), IDesk))
[(u'ng', <FrontDeskNG object at ...>)]

```

### getAllUtilitiesRegisteredFor

Restituisce tutte le utility registrate per un'interfaccia. Questo include anche le utility sovrascritte. Il valore di ritorno è un iterabile di istanze di utility.

- Posizione: `zope.component`
- Firma: `getAllUtilitiesRegisteredFor(interface)`

Esempio:

```

>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         print "Hello", name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter)

>>> from zope.component import getAllUtilitiesRegisteredFor

>>> getAllUtilitiesRegisteredFor(IGreeter)
[<Greeter object at ...>]

```

### getFactoriesFor

Restituisce una tupla (nome, factory) delle factory registrate che creano oggetti che implementano l'interfaccia fornita.

- Posizione: `zope.component`
- Firma: `getFactoriesFor(interface, context=None)`

Esempio:

```

>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IDatabase(Interface):
...     def getConnection():
...         """Return connection object"""

>>> class FakeDb(object):
...     implements(IDatabase)
...     def getConnection(self):
...         return "connection"

>>> from zope.component.factory import Factory

>>> factory = Factory(FakeDb, 'FakeDb')

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> from zope.component.interfaces import IFactory
>>> gsm.registerUtility(factory, IFactory, 'fakedb')

>>> from zope.component import getFactoriesFor

>>> list(getFactoriesFor(IDatabase))
[(u'fakedb', <Factory for <class 'FakeDb'>>)]

```

## getFactoryInterfaces

Trova le interfacce implementate da una factory. Trova la factory più vicina al contesto con il nome specificato e restituisce l'interfaccia o la tupla dell'interfaccia che gli oggetti istanza creati forniranno.

- Posizione: `zope.component`
- Firma: `getFactoryInterfaces(name, context=None)`

Esempio:

```

>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IDatabase(Interface):
...     def getConnection():
...         """Return connection object"""

>>> class FakeDb(object):
...     implements(IDatabase)
...     def getConnection(self):
...         return "connection"

```



```
>>> from zope.component.factory import Factory
>>> factory = Factory(FakeDb, 'FakeDb')
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> from zope.component.interfaces import IFactory
>>> gsm.registerUtility(factory, IFactory, 'fakedb')
>>> from zope.component import getFactoryInterfaces
>>> getFactoryInterfaces('fakedb')
<implementedBy __builtin__.FakeDb>
```

### getGlobalSiteManager

Restituisce il global site manager. Questa funzione non dovrebbe mai fallire e dovrebbe sempre restituire un oggetto che fornisce *IGlobalSiteManager*

- Posizione: `zope.component`
- Firma: `getGlobalSiteManager()`

Esempio:

```
>>> from zope.component import getGlobalSiteManager
>>> from zope.component import globalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm is globalSiteManager
True
```

### getMultiAdapter

Cerca e restituisce un multi-adapter che può adattare degli oggetti ad una certa interfaccia. Se non può essere trovato alcun adapter, solleva `ComponentLookupError`. La stringa vuota come nome è riservata per gli adapter senza nome. I metodi per gli adapter senza nome spesso chiamano i metodi per i named adapter con una stringa vuota come nome.

- Posizione: `zope.component`
- Firma: `getMultiAdapter(objects, interface=Interface, name='', context=None)`
- Vedi anche: `queryMultiAdapter`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.component import adapts
>>> class IAdapteeOne(Interface):
...     pass
>>> class IAdapteeTwo(Interface):
...     pass
>>> class IFunctionality(Interface):
```

```

...     pass

>>> class MyFunctionality(object):
...     implements(IFunctionality)
...     adapts(IAdapteeOne, IAdapteeTwo)
...
...     def __init__(self, one, two):
...         self.one = one
...         self.two = two

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerAdapter(MyFunctionality)

>>> class One(object):
...     implements(IAdapteeOne)

>>> class Two(object):
...     implements(IAdapteeTwo)

>>> one = One()
>>> two = Two()

>>> from zope.component import getMultiAdapter

>>> getMultiAdapter((one,two), IFunctionality)
<MyFunctionality object at ...>

>>> myfunctionality = getMultiAdapter((one,two), IFunctionality)
>>> myfunctionality.one
<One object at ...>
>>> myfunctionality.two
<Two object at ...>

```

## getSiteManager

Prende il site manager più vicino al contesto dato. Se il *context* è *None*, restituisce il global site manager. Se il *context* non è *None*, ci si aspetta di poter trovare un adapter dal *context* a *IComponentLookup*. Se non viene trovato alcun adapter, viene sollevato *ComponentLookupError*.

- Posizione: `zope.component`
- Firma: `getSiteManager(context=None)`

Esempio 1:

```

>>> from zope.component.globalregistry import BaseGlobalComponents
>>> from zope.component import IComponentLookup
>>> sm = BaseGlobalComponents()

>>> class Context(object):
...     def __init__(self, sm):
...         self.sm = sm
...     def __conform__(self, interface):
...         if interface.isOrExtends(IComponentLookup):
...             return self.sm

```

```
>>> context = Context(sm)

>>> from zope.component import getSiteManager

>>> lsm = getSiteManager(context)
>>> lsm is sm
True
```

Esempio 2:

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> sm = getSiteManager()
>>> gsm is sm
True
```

### getUtilitiesFor

Ricerca le utility registrate che forniscono un'interfaccia. Restituisce un iterabile delle coppie nome-utility.

- Posizione: `zope.component`
- Firma: `getUtilitiesFor(interface)`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         print "Hello", name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter)

>>> from zope.component import getUtilitiesFor

>>> list(getUtilitiesFor(IGreeter))
[(u'', <Greeter object at ...>)]
```

### getUtility

Recupera l'utility che fornisce l'interfaccia. Restituisce l'utility più vicina al contesto e che implementa una specifica interfaccia. Se non ne vengono trovate, viene sollevata `ComponentLookupError`.

- Posizione: `zope.component`

- Firma: `getUtility(interface, name='', context=None)`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         return "Hello " + name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter)

>>> from zope.component import getUtility

>>> getUtility(IGreeter).greet('Jack')
'Hello Jack'
```

### handle

Chiama tutti gli handler per gli oggetti dati. Gli handler sono fabbriche di subscription adapter che non restituiscono nulla. Essi fanno tutto il loro lavoro quando vengono chiamati. Gli handler sono tipicamente utilizzati per gestire gli eventi.

- Posizione: `zope.component`
- Firma: `handle(*objects)`

Esempio:

```
>>> import datetime

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IDocumentCreated(Interface):
...     doc = Attribute("The document that was created")

>>> class DocumentCreated(object):
...     implements(IDocumentCreated)
...
...     def __init__(self, doc):
...         self.doc = doc
```

```

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import adapter

>>> @adapter(IDocumentCreated)
... def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerHandler(documentCreated)

>>> from zope.component import handle

>>> handle(DocumentCreated(doc))
>>> doc.created.__class__.__name__
'datetime'

```

## implementedBy

Restituisce le interfacce implementate dalle istanze di una certa classe.

- Posizione: `zope.interface`
- Firma: `implementedBy(class_)`

Esempio 1:

```

>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         print "Hello", name

>>> from zope.interface import implementedBy
>>> implementedBy(Greeter)
<implementedBy __builtin__.Greeter>

```

Esempio 2:

```

>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

```

```
>>> class ISpecial (Interface):
...     pass

>>> class Person (object):
...     implements (IPerson)
...     name = u""

>>> from zope.interface import classImplements
>>> classImplements(Person, ISpecial)

>>> from zope.interface import implementedBy

To get a list of all interfaces implemented by that class::

>>> [x.__name__ for x in implementedBy(Person)]
['IPerson', 'ISpecial']
```

### implementer

Crea un decoratore per dichiarare le interfacce implementate da una factory. Viene restituito un oggetto chiamabile che fa una dichiarazione di implementazione sugli oggetti che gli vengono passati.

- Posizione: `zope.interface`
- Firma: `implementer(*interfaces)`

Esempio:

```
>>> from zope.interface import implementer
>>> class IFoo (Interface):
...     pass
>>> class Foo (object):
...     implements (IFoo)

>>> @implementer (IFoo)
... def foocreator():
...     foo = Foo()
...     return foo
>>> list(implementedBy(foocreator))
[<InterfaceClass __builtin__.IFoo>]
```

### implements

Dichiara le interfacce implementate dalle istanze di una classe. Questa funzione è chiamata all'interno di una definizione di una classe. Gli argomenti sono una o più interfacce. Le interfacce fornite sono aggiunte a quelle già dichiarate in precedenza. Le dichiarazioni precedenti, incluse le dichiarazioni delle classi base, vengono preservate, a meno che non sia stata utilizzata `implementsOnly`.

- Posizione: `zope.interface`
- Firma: `implements(*interfaces)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
```

```

>>> class IPerson(Interface):
...
...     name = Attribute("Name of person")

>>> class Person(object):
...
...     implements(IPerson)
...     name = u""

>>> jack = Person()
>>> jack.name = "Jack"

```

Si può testare così:

```

>>> from zope.interface import providedBy
>>> IPerson in providedBy(jack)
True

```

### implementsOnly

Dichiara le sole interfacce implementate dalle istanze di una classe. Questa funzione è chiamata all'interno di una definizione di classe. Gli argomenti sono una o più interfacce. Le dichiarazioni precedenti, incluse le dichiarazioni delle classi base, vengono sovrascritte.

- Posizione: `zope.interface`
- Firma: `implementsOnly(*interfaces)`

Esempio:

```

>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.interface import implementsOnly

>>> class IPerson(Interface):
...
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...
...     college = Attribute("Name of college")

>>> class Person(object):
...
...     implements(IPerson)
...     name = u""

>>> class NewPerson(Person):
...     implementsOnly(IStudent)
...     college = u""

>>> jack = NewPerson()
>>> jack.college = "New College"

```

Si può testare così:

```

>>> from zope.interface import providedBy

```

```
>>> IPerson in providedBy(jack)
False
>>> IStudent in providedBy(jack)
True
```

### Interface

Con questa classe si possono definire le interfacce. Per definire un'interfaccia, basta ereditare dalla classe `Interface`.

- Posizione: `zope.interface`
- Firma: `Interface(name, doc='')`

Esempio 1:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IPerson(Interface):
...
...     name = Attribute("Name of person")
...     email = Attribute("Email Address")
```

Esempio 2:

```
>>> from zope.interface import Interface

>>> class IHost(Interface):
...
...     def goodmorning(guest):
...         """Say good morning to guest"""
```

### moduleProvides

Dichiara le interfacce fornite da un modulo. Questa funzione è utilizzata nella definizione di un modulo. Gli argomenti sono una o più interfacce. Le interfacce fornite vengono utilizzate per creare la definizione di interfaccia degli oggetti diretti del modulo. Verrà sollevato un errore se il modulo ha già una dichiarazione di interfaccia. In altre parole, è un errore chiamare questa funzione più di una volta nella definizione di un modulo.

Questa funzione è fornita per comodità. Essa fornisce un modo più conveniente per chiamare `directlyProvides` su un modulo.

- Posizione: `zope.interface`
- Firma: `moduleProvides(*interfaces)`
- Vedi anche: `directlyProvides`

You can see an example usage in `zope.component` source itself. The `__init__.py` file has a statement like this:

```
moduleProvides(IComponentArchitecture,
               IComponentRegistrationConvenience)
```

So, the `zope.component` provides two interfaces: `IComponentArchitecture` and `IComponentRegistrationConvenience`.



## noLongerProvides

Rimuove un'interfaccia dalla lista delle interfacce fornite direttamente da un oggetto.

- Posizione: `zope.interface`
- Firma: `noLongerProvides(object, interface)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.interface import classImplements

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class IStudent(Interface):
...     college = Attribute("Name of college")

>>> class Person(object):
...     implements(IPerson)
...     name = u""

>>> jack = Person()
>>> jack.name = "Jack"
>>> jack.college = "New College"
>>> directlyProvides(jack, IStudent)

Si può testare così:

>>> from zope.interface import providedBy
>>> IPerson in providedBy(jack)
True
>>> IStudent in providedBy(jack)
True
>>> from zope.interface import noLongerProvides
>>> noLongerProvides(jack, IStudent)
>>> IPerson in providedBy(jack)
True
>>> IStudent in providedBy(jack)
False
```

## provideAdapter

Si raccomanda di utilizzare *registerAdapter* al posto di questa funzione.

## provideHandler

Si raccomanda di utilizzare *registerHandler* al posto di questa funzione.

### provideSubscriptionAdapter

Si raccomanda di utilizzare *registerSubscriptionAdapter* al posto di questa funzione.

### provideUtility

Si raccomanda di utilizzare *registerUtility* al posto di questa funzione.

### providedBy

Verifica se l'interfaccia è fornita dall'oggetto dato. Restituisce true se l'oggetto dichiara di fornire l'interfaccia, anche se dichiara di fornire un'interfaccia che estende l'interfaccia data.

- Posizione: `zope.interface`
- Firma: `providedBy(object)`

Esempio 1:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class Person(object):
...     implements(IPerson)
...     name = u""

>>> jack = Person()
>>> jack.name = "Jack"

Si può testare così:

>>> from zope.interface import providedBy
>>> IPerson in providedBy(jack)
True
```

Esempio 2:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IPerson(Interface):
...     name = Attribute("Name of person")

>>> class ISpecial(Interface):
...     pass

>>> class Person(object):
...     implements(IPerson)
...     name = u""

>>> from zope.interface import classImplements
```

```
>>> classImplements(Person, ISpecial)
>>> from zope.interface import providedBy
>>> jack = Person()
>>> jack.name = "Jack"
```

Ecco come vedere la lista di tutte le interfacce fornite da questo oggetto::

```
>>> [x.__name__ for x in providedBy(jack)]
['IPerson', 'ISpecial']
```

## queryAdapter

Cerca e restituisce un named adapter che può adattare un oggetto ad un'interfaccia. Se non può essere trovato alcun adapter restituisce il default.

- Posizione: `zope.component`
- Firma: `queryAdapter(object, interface=Interface, name=u'', default=None, context=None)`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place
```

```
>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                       (IGuest,), IDesk, 'ng')

>>> queryAdapter(jack, IDesk, 'ng')
<FrontDeskNG object at ...>
```

### queryAdapterInContext

Cerca uno speciale adapter per adattare un oggetto a un'interfaccia.

Nota: Questo metodo dovrebbe essere utilizzato solo se è necessario fornire un context personalizzato per fornire una ricerca personalizzata. Altrimenti, chiamare l'interfaccia come in:

```
interface(object, default)
```

Restituisce un adapter che può adattare un oggetto a un'interfaccia. Se non può essere trovato alcun adapter, restituisce il default.

Il context viene adattato a IServiceService, e viene utilizzato il servizio 'Adapters' di questo adapter.

Se l'oggetto ha un metodo `__conform__`, questo metodo viene chiamato con l'interfaccia richiesta. Se il metodo restituisce un valore diverso da `None`, questo valore viene restituito. Altrimenti, se l'oggetto implementa già l'interfaccia, viene restituito l'oggetto.

- Posizione: `zope.component`
- Firma: `queryAdapterInContext(object, interface, context, default=None)`
- Vedi anche: `getAdapterInContext`

Esempio:

```
>>> from zope.component.globalregistry import BaseGlobalComponents
>>> from zope.component import IComponentLookup
>>> sm = BaseGlobalComponents()

>>> class Context(object):
...     def __init__(self, sm):
...         self.sm = sm
...     def __conform__(self, interface):
...         if interface.isOrExtends(IComponentLookup):
...             return self.sm

>>> context = Context(sm)

>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
... 
```

```

...     def register():
...         """Register object's details"""
...
>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> sm.registerAdapter(FrontDeskNG,
...                    (IGuest,), IDesk)

>>> from zope.component import queryAdapterInContext

>>> queryAdapterInContext(jack, IDesk, sm)
<FrontDeskNG object at ...>

```

### queryMultiAdapter

Cerca e restituisce un multi-adapter per adattare degli oggetti a un'interfaccia. Se non può essere trovato alcun adapter, restituisce il default. Il nome costituito dalla stringa vuota è riservato per gli adapters senza nome. I metodi per gli unnamed adapter spesso chiamano i metodi per i named adapter con una stringa vuota come nome.

- Posizione: `zope.component`
- Firma: `queryMultiAdapter(objects, interface=Interface, name=u'', default=None, context=None)`
- Vedi anche: `getMultiAdapter`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class IAdapteeOne(Interface):
...     pass

>>> class IAdapteeTwo(Interface):
...     pass

>>> class IFunctionality(Interface):
...     pass

>>> class MyFunctionality(object):
...     implements(IFunctionality)
...     adapts(IAdapteeOne, IAdapteeTwo)
...
...     def __init__(self, one, two):
...         self.one = one
...         self.two = two

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerAdapter(MyFunctionality)

>>> class One(object):
...     implements(IAdapteeOne)

>>> class Two(object):
...     implements(IAdapteeTwo)

>>> one = One()
>>> two = Two()

>>> from zope.component import queryMultiAdapter

>>> getMultiAdapter((one,two), IFunctionality)
<MyFunctionality object at ...>

>>> myfunctionality = queryMultiAdapter((one,two), IFunctionality)
>>> myfunctionality.one
<One object at ...>
>>> myfunctionality.two
<Two object at ...>
```

### queryUtility

Questa funzione è utilizzata per cercare una utility che fornisce una certa interfaccia. Se non trova alcuna utility, restituisce il default.

- Posizione: `zope.component`
- Firma: `queryUtility(interface, name='', default=None)`

Esempio:

```

>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         return "Hello " + name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet, IGreeter)

>>> from zope.component import queryUtility

>>> queryUtility(IGreeter).greet('Jack')
'Hello Jack'

```

## registerAdapter

Questa funzione registra una factory di adapter.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registerAdapter(factory, required=None, provided=None, name=u'', info=u'')`
- Vedi anche: `unregisterAdapter`

Esempio:

```

>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):

```

```
...     next_id = get_next_id()
...     bookings_db[next_id] = {
...         'name': guest.name,
...         'place': guest.place,
...         'phone': guest.phone
...     }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                     (IGuest,), IDesk, 'ng')

Si può testare così:

>>> queryAdapter(jack, IDesk, 'ng')
<FrontDeskNG object at ...>
```

### registeredAdapters

Restituisce un iterabile di *IAdapterRegistrations*. Queste registrazioni descrivono le attuali registrazioni degli adapter per l'oggetto.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registeredAdapters()`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
```



```

...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                     (IGuest,), IDesk, 'ng2')

>>> reg_adapter = list(gsm.registeredAdapters())
>>> 'ng2' in [x.name for x in reg_adapter]
True

```

## registeredHandlers

Restituisce un iterabile di *IHandlerRegistrations*. Queste registrazioni descrivono le attuali registrazioni degli handler per l'oggetto.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registeredHandlers()`

Esempio:

```

>>> import datetime

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IDocumentCreated(Interface):

```

```
...     doc = Attribute("The document that was created")

>>> class DocumentCreated(object):
...     implements(IDocumentCreated)
...
...     def __init__(self, doc):
...         self.doc = doc

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import adapter

>>> @adapter(IDocumentCreated)
... def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerHandler(documentCreated, info='ng3')

>>> reg_adapter = list(gsm.registeredHandlers())
>>> 'ng3' in [x.info for x in reg_adapter]
True

>>> gsm.registerHandler(documentCreated, name='ng4')
Traceback (most recent call last):
...
TypeError: Named handlers are not yet supported
```

### registeredSubscriptionAdapters

Restituisce un iterabile di *ISubscriptionAdapterRegistrations*. Queste registrazioni descrivono le attuali registrazioni dei subscription adapter per l'oggetto.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registeredSubscriptionAdapters()`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IValidate(Interface):
...     def validate(ob):
...         """Determine whether the object is valid
...
...         Return a string describing a validation problem.
...         An empty string is returned to indicate that the
...         object is valid.
...         """

>>> class IDocument(Interface):
```

```

...     summary = Attribute("Document summary")
...     body = Attribute("Document text")

>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body

>>> from zope.component import adapts

>>> class AdequateLength(object):
...     adapts(IDocument)
...     implements(IValidate)
...     def __init__(self, doc):
...         self.doc = doc
...     def validate(self):
...         if len(self.doc.body) < 1000:
...             return 'too short'
...         else:
...             return ''

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerSubscriptionAdapter(AdequateLength, info='ng4')

>>> reg_adapter = list(gsm.registeredSubscriptionAdapters())
>>> 'ng4' in [x.info for x in reg_adapter]
True

```

## registeredUtilities

Restituisce un iterabile di *IUtilityRegistrations*. Queste registrazioni descrivono le attuali registrazioni delle utility per l'oggetto.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registeredUtilities()`

Esempio:

```

>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...     implements(IGreeter)
...     def greet(self, name):
...         print "Hello", name

```

```
>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet, info='ng5')

>>> reg_adapter = list(gsm.registeredUtilities())
>>> 'ng5' in [x.info for x in reg_adapter]
True
```

### registerHandler

Questa funzione registra un handler. Un handler è un subscriber che non calcola un adapter ma svolge qualche funzione quando viene chiamato.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registerHandler(handler, required=None, name='u', info='')`
- Vedi anche: `unregisterHandler`

Note: In the current implementation of `zope.component` doesn't support `name` attribute.

Esempio:

```
>>> import datetime

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IDocumentCreated(Interface):
...     doc = Attribute("The document that was created")

>>> class DocumentCreated(object):
...     implements(IDocumentCreated)
...
...     def __init__(self, doc):
...         self.doc = doc

>>> def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import adapter

>>> @adapter(IDocumentCreated)
... def documentCreated(event):
...     event.doc.created = datetime.datetime.utcnow()

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerHandler(documentCreated)
```

```
>>> from zope.component import handle
>>> handle(DocumentCreated(doc))
>>> doc.created.__class__.__name__
'datetime'
```

### registerSubscriptionAdapter

Questa funzione serve a registrare una factory di subscribers.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `registerSubscriptionAdapter(factory, required=None, provides=None, name=u'', info='')`
- Vedi anche: `unregisterSubscriptionAdapter`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IValidate(Interface):
...     def validate(ob):
...         """Determine whether the object is valid
...
...         Return a string describing a validation problem.
...         An empty string is returned to indicate that the
...         object is valid.
...         """

>>> class IDocument(Interface):
...     summary = Attribute("Document summary")
...     body = Attribute("Document text")

>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body

>>> from zope.component import adapts

>>> class AdequateLength(object):
...
...     adapts(IDocument)
...     implements(IValidate)
...
...     def __init__(self, doc):
...         self.doc = doc
...
...     def validate(self):
...         if len(self.doc.body) < 1000:
...             return 'too short'
...         else:
...             return ''

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
```

```
>>> gsm.registerSubscriptionAdapter(AdequateLength)
```

### registerUtility

Questa funzione serve a registrare una utility.

- Posizione: `zope.component` - `IComponentRegistry`
- Firma: `registerUtility(component, provided=None, name='u', info='u')`
- Vedi anche: `unregisterUtility`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         print "Hello", name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet)
```

### subscribers

Questa funzione serve a recuperare i subscribers. Vengono restituiti i subscribers che forniscono l'interfaccia passata e che dipendono e sono calcolati dalla sequenza di oggetti richiesti.

- Posizione: `zope.component` - `IComponentRegistry`
- Firma: `subscribers(required, provided, context=None)`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IValidate(Interface):
...     def validate(ob):
...         """Determine whether the object is valid
...
...         Return a string describing a validation problem.
...         An empty string is returned to indicate that the
...         object is valid.
...         """
```

```

>>> class IDocument(Interface):
...     summary = Attribute("Document summary")
...     body = Attribute("Document text")

>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body

>>> from zope.component import adapts

>>> class SingleLineSummary:
...     adapts(IDocument)
...     implements(IValidate)
...
...     def __init__(self, doc):
...         self.doc = doc
...
...     def validate(self):
...         if '\n' in self.doc.summary:
...             return 'Summary should only have one line'
...         else:
...             return ''

>>> class AdequateLength(object):
...     adapts(IDocument)
...     implements(IValidate)
...
...     def __init__(self, doc):
...         self.doc = doc
...
...     def validate(self):
...         if len(self.doc.body) < 1000:
...             return 'too short'
...         else:
...             return ''

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerSubscriptionAdapter(SingleLineSummary)
>>> gsm.registerSubscriptionAdapter(AdequateLength)

>>> from zope.component import subscribers

>>> doc = Document("A\nDocument", "blah")
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['Summary should only have one line', 'too short']

>>> doc = Document("A\nDocument", "blah" * 1000)
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['Summary should only have one line']

>>> doc = Document("A Document", "blah")

```

```
>>> [adapter.validate()
...   for adapter in subscribers([doc], IValidate)
...   if adapter.validate()]
['too short']
```

### unregisterAdapter

Questa funzione serve a de-registrare una factory di adapter. Viene restituito un booleano che indica se il registro è stato modificato o meno. La funzione restituisce False se il componente dato è None e non ci sono componenti registrati, o se il componente dato non è None e non è registrato, altrimenti restituisce True.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `unregisterAdapter(factory=None, required=None, provided=None, name=u'')`
- Vedi anche: `registerAdapter`

Esempio:

```
>>> from zope.interface import Attribute
>>> from zope.interface import Interface

>>> class IDesk(Interface):
...     """A frontdesk will register object's details"""
...
...     def register():
...         """Register object's details"""
...

>>> from zope.interface import implements
>>> from zope.component import adapts

>>> class FrontDeskNG(object):
...
...     implements(IDesk)
...     adapts(IGuest)
...
...     def __init__(self, guest):
...         self.guest = guest
...
...     def register(self):
...         next_id = get_next_id()
...         bookings_db[next_id] = {
...             'name': guest.name,
...             'place': guest.place,
...             'phone': guest.phone
...         }

>>> class Guest(object):
...
...     implements(IGuest)
...
...     def __init__(self, name, place):
...         self.name = name
...         self.place = place

>>> jack = Guest("Jack", "Bangalore")
>>> jack_frontdesk = FrontDeskNG(jack)
```



```

>>> IDesk.providedBy(jack_frontdesk)
True

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()
>>> gsm.registerAdapter(FrontDeskNG,
...                     (IGuest,), IDesk, 'ng6')

Si può testare così:

>>> queryAdapter(jack, IDesk, 'ng6')
<FrontDeskNG object at ...>

Ora de-registriamo l'adapter:

>>> gsm.unregisterAdapter(FrontDeskNG, name='ng6')
True

Dopo la de-registrazione si ha che:

>>> print queryAdapter(jack, IDesk, 'ng6')
None

```

### unregisterHandler

Questa funzione serve a de-registrare un handler. Un handler è un subscriber che non calcola un adapter ma svolge qualche funzione quando viene chiamato. Viene restituito un booleano che indica se il registro è stato modificato o meno.

- Posizione: `zope.component` - `IComponentRegistry`
- Firma: `unregisterHandler(handler=None, required=None, name=u'')`
- Vedi anche: *registerHandler*

Esempio:

```

>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IDocument(Interface):
...     summary = Attribute("Document summary")
...     body = Attribute("Document text")

>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body

>>> doc = Document("A\nDocument", "blah")

>>> class IDocumentAccessed(Interface):
...     doc = Attribute("The document that was accessed")

>>> class DocumentAccessed(object):

```

```
...     implements(IDocumentAccessed)
...
...     def __init__(self, doc):
...         self.doc = doc
...         self.doc.count = 0

>>> from zope.component import adapter

>>> @adapter(IDocumentAccessed)
... def documentAccessed(event):
...     event.doc.count = event.doc.count + 1

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerHandler(documentAccessed)

>>> from zope.component import handle

>>> handle(DocumentAccessed(doc))
>>> doc.count
1

Ora de-registriamo l'handler:

>>> gsm.unregisterHandler(documentAccessed)
True

Dopo la de-registrazione si ha:

>>> handle(DocumentAccessed(doc))
>>> doc.count
0
```

### unregisterSubscriptionAdapter

Questa funzione serve a de-registrare una factory di subscriber. Viene restituito un booleano che indica se il registro è stato modificato o meno. La funzione restituisce False se il componente dato è None e non ci sono componenti registrati, o se il componente dato non è None e non è registrato, altrimenti restituisce True.

- Posizione: `zope.component - IComponentRegistry`
- Firma: `unregisterSubscriptionAdapter(factory=None, required=None, provides=None, name=u'')`
- Vedi anche: `registerSubscriptionAdapter`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import Attribute
>>> from zope.interface import implements

>>> class IValidate(Interface):
...     def validate(ob):
...         """Determine whether the object is valid
...
...         Return a string describing a validation problem.
...         An empty string is returned to indicate that the
```

```

...     object is valid.
...     """

>>> class IDocument(Interface):
...     summary = Attribute("Document summary")
...     body = Attribute("Document text")

>>> class Document(object):
...     implements(IDocument)
...     def __init__(self, summary, body):
...         self.summary, self.body = summary, body

>>> from zope.component import adapts

>>> class AdequateLength(object):
...
...     adapts(IDocument)
...     implements(IValidate)
...
...     def __init__(self, doc):
...         self.doc = doc
...
...     def validate(self):
...         if len(self.doc.body) < 1000:
...             return 'too short'
...         else:
...             return ''

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> gsm.registerSubscriptionAdapter(AdequateLength)

>>> from zope.component import subscribers

>>> doc = Document("A\nDocument", "blah")
>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
['too short']

Ora de-registriamo il componente:

>>> gsm.unregisterSubscriptionAdapter(AdequateLength)
True

Dopo la de-registrazione si ha:

>>> [adapter.validate()
...  for adapter in subscribers([doc], IValidate)
...  if adapter.validate()]
[]

```

### unregisterUtility

Questa funzione serve a de-registrare una utility. Viene restituito un booleano che indica se il registro è stato modificato o meno. La funzione restituisce False se il componente dato è None e non ci sono componenti registrati, o se il

componente dato non è None e non è registrato, altrimenti restituisce True.

- Posizione: `zope.component` - `IComponentRegistry`
- Firma: `unregisterUtility(component=None, provided=None, name=u')`
- Vedi anche: `registerUtility`

Esempio:

```
>>> from zope.interface import Interface
>>> from zope.interface import implements

>>> class IGreeter(Interface):
...     def greet(name):
...         "say hello"

>>> class Greeter(object):
...
...     implements(IGreeter)
...
...     def greet(self, name):
...         return "Hello " + name

>>> from zope.component import getGlobalSiteManager
>>> gsm = getGlobalSiteManager()

>>> greet = Greeter()
>>> gsm.registerUtility(greet)

>>> queryUtility(IGreeter).greet('Jack')
'Hello Jack'

Now unregister:

>>> gsm.unregisterUtility(greet)
True

After unregistration:

>>> print queryUtility(IGreeter)
None
```

---

## Credits e ringraziamenti

---

Si ringraziano per il loro contributo a questa documentazione:

- Giacomo Spettoli (coordinatore/traduttore/motivatore)
- Maurizio Delmonte (ideatore/motivatore)
- Massimo Azzolini (motivatore/traduttore)
- Federica D'Elia (traduttore)
- Giovanni Giangiobbe (traduttore/revisore)
- Giampiero Lago
- Alex Sani
- Luciano Naldesi
- Giorgio Borelli