# Creare **temi** e pagine personalizzate per **Drupal**

*Drupal è un software estremamente potente e flessibile: in questo numero vediamo come creare le basi di un tema ed utilizzare il modulo Views*

Andrea Franceschini
a.franceschini@oltrelinux.com

Laureato in Ingegneria Informatica a Padova, appassionato di informatica a tutto tondo, in particolare di musica, grafica ed elaborazione di immagini e suoni. Programma in C/C++, PHP, varie ed eventuali, fotografa e disegna a tempo perso

Nella scorsa puntata abbiamo fatto conoscenza con Drupal e ne abbiamo visto alcune funzionalità ed alcuni moduli fondamentali. In questa seconda puntata prenderemo confidenza con la gestione dei temi: per prima cosa ne installeremo uno che rimpiazzerà quello di default per le pagine di amministrazione, poi vedremo come aggiungere alcuni moduli essenziali per lo sviluppo di temi e, finalmente, ci tufferemo nella creazione di un nostro tema e parleremo del modulo Views, vero e proprio coltellino svizzero per la realizzazione di elenchi e selezioni dei contenuti del sito Drupal.

## Installiamo Rubik

Rubik è un tema per l'interfaccia di amministrazione che punta a semplificarla e renderla più produttiva. Scarichiamolo da http://drupal.org/project/rubik (versione 3.0-beta2 al momento della scrittura). Ci servirà anche Tao, da http://drupal.org/project/tao, un "tema base" che fornisce uno scheletro su cui costruirne altri ed è necessario per usare Rubik.

Non approfondiremo ulteriormente, ci basti sapere che in generale è una buona idea partire da un "tema base" per sviluppare il proprio tema. In effetti, questa procedura è praticamente la norma in Drupal. Nel caso in cui il nostro tema non disponga di elementi personalizzati per tutte le sfaccettature del sito, sia il framework che i moduli forniscono dei template di default che sono l'estremo fallback.

È inoltre possibile usare come base altri temi pienamente funzionali: per esempio il tema Minnelli è derivato direttamente da Garland (quello di default di Drupal).

Scompattiamo i temi nella directory **themes/** del nostro sito (dovremo crearla nel caso non fosse già presente):

```
$ cd /var/www/localhost/htdocs/drupal-6.20/sites/default
$ mkdir themes
$ tar zxf tao-6.x-3.2.tar.gz -C themes
$ tar zxf rubik-6.x-3.0-beta2.tar.gz -C themes
```

Su http://localhost/drupal-6.20/admin/build/themes troveremo l'elenco dei temi, compresi Tao, Rubik e Cube (un sotto-tema di Rubik). La casella *Attivato* significa che il tema sarà disponibile per i visitatori, mentre *Predefinito* indica che è quello utilizzato di default. In generale, in produzione si avrà attivato un solo tema, che sarà impostato come predefinito.

Su /admin/settings/admin (d'ora in poi ometteremo la parte iniziale dell'URI, http://localhost/drupal-6.20, per brevità: tutti gli URL che iniziano con la / si intendono relativi alla root) potremo scegliere Rubik come tema di amministrazione. È importante spuntare la casella *Usa il tema di amministrazione per le modifiche del contenuto*, altrimenti Drupal userà il tema predefinito del sito in questi casi, riducendo l'efficacia di Rubik.

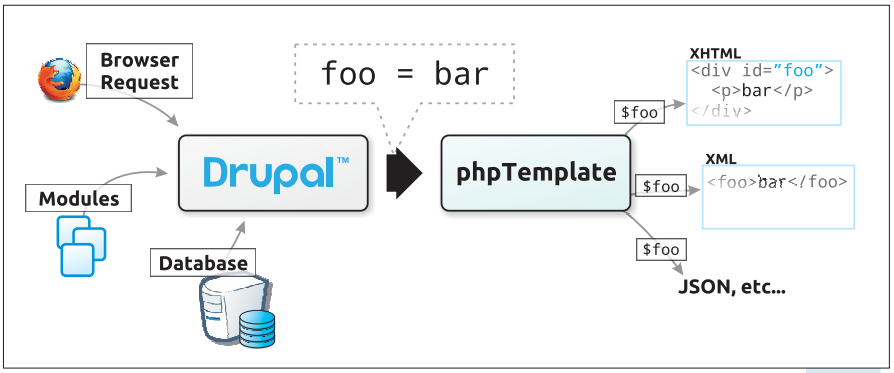
Salviamo la configurazione ed il tema della pagina verrà cambiato. Per confermare che stiamo usando Rubik solo come tema di amministrazione, controlliamo http://localhost/drupal-6.20: il tema della home dovrebbe essere ancora *Garland*.

## Moduli per lo sviluppo

La community Drupal dispone di molti moduli per aiutarci nello

sviluppo di temi, il problema è che spesso tali moduli tendono a "pestarsi i piedi" a vicenda: è consigliabile utilizzarne pochi e scelti con cura, in modo che si sovrappongano il meno possibile, consentendoci di fare tutto quello di cui abbiamo bisogno.

Useremo *Administration menu* (http://drupal.org/project/admin_menu) e due moduli di utilità, *Vertical Tabs* (http://drupal.org/project/vertical_tabs) e *Admin* (http://drupal.org/project/admin) che ci torneranno utili anche una volta messo in produzione il sito.




Schema riassuntivo del funzionamento di phpTemplate

1

Per installare un modulo, dovremo scompattarlo nella directory **modules/** all'interno del percorso del nostro sito, quindi per esempio in **drupal-6.20/sites/default/modules/** e, in seguito, attivato dall'URL /admin/build/modules.

Una nota prima di proseguire: nel tempo passato dalla prima puntata, è probabile che qualche modulo sia stato aggiornato: basta verificare su /admin/reports/updates e, se vediamo qualche richiesta di aggiornamento, sarà sufficiente scaricare la nuova versione, scompattarla sopra alla vecchia e seguire la procedura su /update.php.

## Drush

Drush è uno script che permette di effettuare diverse operazioni di gestione di un sito Drupal direttamente dalla shell. E' particolarmente utile durante la fase di sviluppo, quando dobbiamo azzerare spesso la cache e ricostruire il registro dei temi, ma per utilizzarlo è necessario avere accesso alla shell della macchina su cui gira Drupal, quindi è ideale sul nostro sistema di sviluppo, ma non lo potremo usare su shared hosting che non offrano accesso alla shell.

Drush non richiede una vera e propria installazione, basta scaricarlo dall'URL http://drupal.org/project/drush e, una volta scompattato in una directory a nostro piacimento, potremo lanciarlo da lì con **./drush**. Chi lo desidera, può aggiungere la directory al **$PATH** per avere a disposizione il comando **drush** ovunque nel filesystem.

Come dicevamo, la funzione che ci troveremo ad usare più spesso durante lo sviluppo è l'azzeramento di tutte le cache. Posizioniamoci nel filesystem all'interno della directory del nostro sito, per esempio **drupal-6.20/sites/default**, e lanciamo:

```
$ drush cc all
```

In alternativa, possiamo omettere l'argomento **all** per ottenere una lista delle cache che possiamo eliminare selettivamente.

Drush può tornare utile anche una volta entrati in produzione, ad esempio usandolo all'interno di script lanciati via cron per mantenere in piena efficienza tutti i nostri siti.

## Creare il tema

Drupal dispone di un componente chiamato *template engine*, che si occupa di rendere indipendente lo strato che elabora i dati dal cosiddetto strato di presentazione, cioè quello che deve rappresentare i dati in un qualche formato utile alla consultazione. Nella stragrande maggioranza dei casi, il formato finale sarà l'HTML, ma potrebbe essere anche XML (per i feed RSS) o altri ancora. In pratica, ci permetterà di accedere ai dati tramite delle variabili a cui avremo accesso dai nostri file template. Le variabili conterranno i dati formattati in modo adatto al formato di visualizzazione, che nel nostro caso sarà l'HTML (figura 1).

L'elemento fondamentale di ogni tema è un file **.info** che ne definisce le proprietà principali. Vediamo come crearne uno: per prima cosa, posizioniamoci in **drupal-6.20/sites/default/themes/** (sostituendo eventualmente **default** con il sito su cui stiamo lavorando) e creiamo la directory del nostro tema: **mkdir mytheme**. Al suo interno, creiamo un file **mytheme.info**:

```
$ cd mytheme && touch mytheme.info
```

Apriamolo con il nostro editor di testi preferito e copiamoci il contenuto del listato 1: queste sono le informazioni principali sul tema, ad esempio la prima riga specifica il nome del tema, che comparirà (insieme all'eventuale descrizione), nell'elenco dei temi. Più avanti specifichiamo che la versione del core (di Drupal) con cui questo tema è compatibile è la **6.x** e che il template engine è **phptemplate**. In seguito vanno specificati i file CSS e Javascript (che nel nostro

caso è commentato, dato che non ne useremo); è importante ricordarsi di inserire anche la coppia di parentesi quadrate prima del segno **=**, perché quella sintassi crea una lista (array) di file.

Come vedete, nel codice abbiamo specificato l'utilizzo di un file **960gs.css**: si tratta di un framework CSS per la creazione di siti basati su Grid System, che potete scaricare da http://960.gs.

Nelle righe che elencano i CSS possiamo anche specificare le *media query*, cioè per quale medium (schermo, screen reader, dispositivo mobile...) ogni file dovrà essere usato.

Alla fine del file troviamo una lista di *region*, che sono delle aree che Drupal ci metterà a disposizione attraverso l'interfaccia di amministrazione e in cui potremo inserire *blocchi*, cioè frammenti di pagina che possono contenere semplice codice HTML, ma anche contenuti dinamici, ma ne parleremo meglio più avanti.

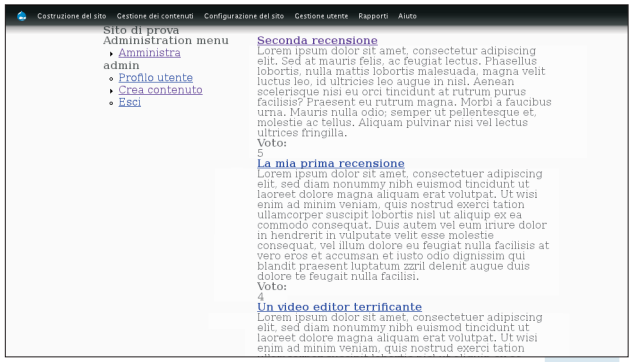
Ora che abbiamo un file di descrizione, possiamo attivare il nostro tema da /admin/build/themes facendo click sulle due caselle *Attivato* e *Predefinito*. Salviamo la configurazione e verifichiamo che il tema sia in uso tornando alla home page http://localhost/drupal-6.20/: per ora non c'è molto da vedere (figura 2), ma ci lavoreremo su!

**L1 – mytheme.info**

```
name = "MyTheme"
description = "Un tema per Drupal"
core = "6.x"
engine = "phptemplate"

stylesheets[screen][] = "css/960gs.css"
stylesheets[screen][] = "css/style.css"
; scripts[] = "js/example.js"

region[header] = Header
region[left] = Left sidebar
region[right] = Right sidebar
region[footer] = Footer
```




Il nostro tema appena creato non dispone di alcuna formattazione, ci lavoreremo su!

**2**

## Lo scheletro HTML del nostro tema

Completata questa fase preliminare, passiamo alla costruzione dell'ossatura HTML del nostro sito: partiremo con un file chiamato **page.tpl.php**, che è il template principale per le nostre pagine. Copiamo al suo interno il listato 2 ed andiamo ad analizzare un po' il codice: alla prima riga, dato che phpTemplate è in grado di produrre XML sintatticamente corretto, abbiamo optato per utilizzare XHTML. Drupal opera già nel primo tag, **<html>**, in cui specifichiamo gli attributi relativi alla lingua accedendo all'oggetto **$language**. Proseguendo, incontriamo il tag **<head>**, in cui usiamo le variabili **$head** (che contiene in genere alcuni tag **<meta>** utili alla fruibilità delle pagine, per esempio il charset); **$head_title**, che contiene il titolo della pagina; **$styles**, che contiene l'HTML necessario ad integrare i fogli di stile che abbiamo specificato nel file **.info** del tema e **$scripts** che fa la stessa cosa per le librerie javascript.

La potenza di phpTemplate engine è proprio questa: abbiamo a disposizione i contenuti già in formato XHTML nelle variabili, per cui ci basterà "stamparne" (con **echo**) il contenuto per ottenere quanto desiderato.

Sebbene nella maggior parte dei casi avere a disposizione i dati già formattati risulti comodo, questo riduce la flessibilità del sistema: la formattazione potrebbe non corrispondere a quanto desideravamo e costringerci a trucchi ed hack per ottenere un determinato risultato. Inoltre, questo approccio richiede una forte interazione tra il grafico che produce il design del sito ed il programmatore che lo deve adattare all'engine, cosa che non si adatta molto bene al più comune workflow di sviluppo web, in cui ognuno ha compiti separati.

Per fortuna, la natura modulare di Drupal permette di modificare le parti che creano problemi e perfino di implementare template engine personalizzati: esistono port di Smarty ed altri motori con una sintassi orientata ai marcatori, anche se la loro flessibilità viene spesso minata dalle scelte progettuali di Drupal, come ad esempio le template suggestion che vedremo tra poco.

Il resto del template dovrebbe essere facilmente comprensibile, ma dobbiamo fare una precisazione: alcune variabili hanno nomi che ricordiamo di aver già visto nel file **mytheme.info**, per esempio **$header** o **$footer**. Queste *region* conterranno il codice dei *blocchi* che specificheremo tra poco attraverso l'apposita interfaccia.

Le variabili che, invece, non sono riconducibili a delle region vengono generate da phpTemplate per conto del core e dei vari moduli che abbiamo installato, oppure per conto del nostro tema personalizzato usando delle speciali funzioni, chiamate *override*, che andranno inserite in un file **template.php** che dovremo creare all'interno della directory principale del nostro tema. Possiamo personalizzare in

## L2 - page.tpl.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="<?php echo $language->language ?>"
    lang="<?php echo $language->language ?>">
<head>
  <?php echo $head ?>
  <title><?php echo $head_title ?></title>
  <?php echo $styles ?>
  <?php echo $scripts ?>
</head>

<body class="<?php echo $body_classes ?> container_12">
  <div id="header" class="grid_12 alpha omega">
    <div class="site_title">
      <h1><?php echo $site_name ?></h1>
      <h2><?php echo $site_slogan ?></h2>
    </div>
    <div class="region"><?php echo $header ?></div>
  </div>

  <div id="primary_links" class="grid_12 alpha omega">
    <?php echo $primary_links ?>
  </div>
  <div id="left_sidebar" class="grid_3 alpha">
    <?php echo $left ?>
  </div>
  <div id="content" class="grid_6">
    <?php echo $content ?>
  </div>
  <div id="right_sidebar" class="grid_3 omega">
    <?php echo $right ?>
  </div>
  <div class="clear"></div>
  <div id="footer">
    <?php echo $footer ?>
  </div>
  <?php echo $closure ?>
</body>
</html>
```

## R2 - Template.php

Drupal sfrutta un meccanismo chiamato *override* per consentire agli utenti che scrivono moduli e temi di modificarne e personalizzarne il comportamento. Attraverso una serie di *hook*, funzioni speciali che possono essere implementate da moduli e temi, è possibile modificare in modo anche piuttosto complesso il comportamento di Drupal e, per questo, è bene essere certi di quello che si sta facendo.

Override è, in realtà, un termine poco felice, dato che non si tratta di una vera *sostituzione* degli hook esistenti, bensì di una sorta di *aggiunta in coda* delle nostre funzioni a quelle già esistenti, in modo che non vengano minate funzionalità del sistema. Così facendo è possibile per moduli e temi diversi agganciarsi agli stessi hook per aggiungere funzionalità indipendenti le une dalle altre.

Nel caso dei temi, queste funzioni vengono raccolte in un file speciale all'interno della directory del tema: **`template.php`**. I nomi delle funzioni saranno sul modello **`nomedeltema_nomedellhook`**: se c'è un hook che si chiama **`theme_preprocess_node`**, potremo creare un override chiamato **`mytheme_preprocess_node`**, facendo attenzione al fatto che **`mytheme`** è il nome del tema usato per il file **`.info`**.

La lista degli hook del core è molto vasta ed ogni modulo può aggiungere i propri. È importante notare, però, che non tutti gli hook sono disponibili a tutti i livelli: per esempio alcuni possono essere accessibili solo dai temi, altri solo dai moduli e, a volte, l'API di alcuni moduli crea qualche confusione mischiando le due cose. Al solito, si può fare affidamento su una vasta (anche se non sempre lineare) documentazione e su una community molto attiva e disponibile.

Tra i sorgenti, disponibili su http://www.oltrelinux.com/risorse/LC74/ è disponibile una buona parte del file **`template.php`** che uso per tutti i miei siti Drupal.

maniera molto precisa tutti questi parametri: in questa puntata non approfondiremo, ma possiamo farci un'idea del meccanismo leggendo l'approfondimento nel riquadro 2.

Il secondo file fondamentale per la personalizzazione del nostro tema è **`node.tpl.php`** (listato 3), che viene usato per il rendering dei *nodi*. Ricordate che tra le variabili di **`page.tpl.php`** c'era anche **`$content`**? Questa variabile conterrà il frammento di HTML generato a partire da **`node.tpl.php`** o altri file, come per esempio quelli del modulo Views di cui parleremo più avanti.

Le prime righe servono ad inserire il titolo del nodo, se questo viene visualizzato in modalità *teaser*: questa è una modalità "troncata" che possiamo scegliere in varie situazioni, per esempio per mostrare i contenuti recenti, come risultato di una ricerca oppure in una *vista* che elenchi solo i contenuti di un certo tipo (vedremo più avanti come creare queste pagine col modulo Views).

Le due righe che seguono inseriscono i link che ci permettono di interagire con il contenuto principale della pagina; nel caso di un nodo avremo le opzioni *Mostra* e *Modifica*, che ci porteranno rispettivamente alla visualizzazione predefinita del nodo (l'URL /node/1, per intenderci) ed alla pagina di modifica del nodo (/node/1/edit). Attorno alle variabili **`$tabs`** e **`$tabs2`** abbiamo inserito i tag che creano una lista non ordinata, perché phpTemplate formatta i link dei tab usando il tag **`<li>`**. Se avessimo voluto inserire i tab in una sola riga, separandoli con un trattino o con una virgola, avremmo dovuto sovrascrivere la funzione che li renderizza.

Infine, le ultime righe del file servono ad inserire il titolo (nel caso non ci trovassimo in modalità teaser) e, finalmente, anche il contenuto vero e proprio del nodo.

Prima di proseguire, ricordiamo che il template **`node.tpl.php`**, opportunamente popolato con i contenuti reali, verrà inserito da phpTemplate nella variabile **`$content`** di **`page.tpl.php`**: questo meccanismo si chiama template suggestion e permette di realizzare

template specifici per pagine particolari, tipi di nodo o singoli blocchi, consentendo di personalizzare in modo estremamente granulare i temi per Drupal.

## Template suggestion

Immaginiamo di voler implementare un layout diverso per la home page: possiamo creare il template **page-front.tpl.php** e Drupal, quando si troverà a renderizzare la home, sceglierà questo file anziché **page.tpl.php**, che verrà usato per tutte le altre pagine.

Come facciamo a sapere quali nomi dare ai nostri template? Ci viene in aiuto il modulo Theme Developer, in grado di elencare in un riquadro non solo i template di alto livello (quelli relativi al layout di pagina o di nodo), ma anche tutti quelli che sono stati utilizzati per renderizzare ogni porzione della pagina.

Purtroppo il modulo ha la tendenza ad inserire il suo markup in posti poco prevedibili, scombinandoci il layout: l'importante è essere consapevoli di questo comportamento ed usarlo con parsimonia, attivandolo solo quando serve.

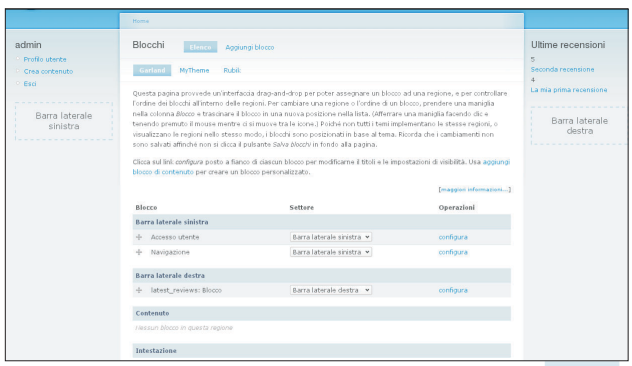
### L3 - node.tpl.php

```
<?php if($teaser) : ?>
   <h4><?php echo l($title, 'node/' . $nid) ?></h4>
<?php endif ?>

<?php if($tabs) : ?><ul class="tabs primary">
                              <?php echo $tabs ?></ul><?php endif ?>
<?php if($tabs2) : ?><ul class="tabs primary">
                             <?php echo $tabs2 ?></ul><?php endif ?>

<?php if(!$teaser) : ?>
   <div class="header">
            <h1><?php echo $title ?></h1>
   </div>
<?php endif ?>

<div class="node_content"><?php echo $content ?></div>
```




L'interfaccia di gestione dei blocchi

3

A questo punto potremmo creare un template **node-review.tpl.php** ed inserirvi un markup speciale per le pagine corrispondenti alle recensioni (*nodo* di tipo **review**). Ogni volta che aggiungiamo una template suggestion dobbiamo rigenerare il *registro dei temi* usando Drush (con il comando **drush cc theme**) oppure con l'apposita funzione, che troviamo nella barra fornita dal modulo *Admin menu* (click sulla prima icona a sinistra, poi sulla voce *Flush all caches* e, infine, *Theme registry*).

## Blocchi

I blocchi sono delle "porzioni" di pagina. Come abbiamo detto, possono essere inseriti ovunque abbiamo previsto una *region* e possono contenere informazioni di vario tipo: dal testo statico alle liste dinamiche più elaborate. Se accediamo all'interfaccia per la gestione (/admin/build/block) ne troveremo già molti, messi a disposizione da Drupal stesso o dai moduli che prevedono questa possibilità.

Esaminando l'interfaccia di gestione dei blocchi (figura 3), l'elemento che ci interessa notare per primo è l'elenco delle *region* che abbiamo specificato nel file del nostro tema; se non sono presenti, sarà bene rigenerare il registro dei temi e ricaricare l'interfaccia.

Possiamo organizzare i blocchi all'interno di ciascuna region trascinandoli (con l'icona a croce posta alla sinistra di ogni blocco). **Attenzione:** ogni volta che riorganizziamo la disposizione dei blocchi dovremo confermare la nostra scelta con un click sul pulsante *Salva blocchi* in fondo alla pagina. Questo vale anche nel caso volessimo modificare la configurazione (link *Configura* a destra) di un blocco che abbiamo appena inserito e spostato in una region.

L'importanza e la flessibilità dei blocchi può non essere chiara a prima vista: come quasi tutte le cose in Drupal, necessita di un certo approfondimento e non c'è modo migliore per approfondire un argomento che metterlo in pratica subito. Nella seconda parte dell'articolo, vedremo come utilizzare il modulo Views per creare un nuovo tipo di blocco contenente elenchi strutturati dei contenuti del sito e come aggiungere tale blocco alle nostre pagine.

## Views

Il modulo Views permette di recuperare e strutturare informazioni in una forma non prevista da Drupal e di renderizzarle in molte forme, dalla pagina vera e propria, al feed RSS, al blocco. In pratica si tratta di un'interfaccia per la creazione di query - anche piuttosto complesse - al database di Drupal.

Un esempio di cosa si può fare con Views? La lista delle *Categorie* di un sito, un archivio delle notizie, una "Top 5" delle recensioni con i voti più alti, un elenco con gli ultimi commenti degli utenti...

Come dicevamo, Views si occupa di fare la query, ma anche di preparare la visualizzazione dei risultati, per esempio potremmo volere una pagina con l'elenco delle *Categorie*, ma ci potrebbe far comodo anche un blocco, da inserire nella barra laterale...

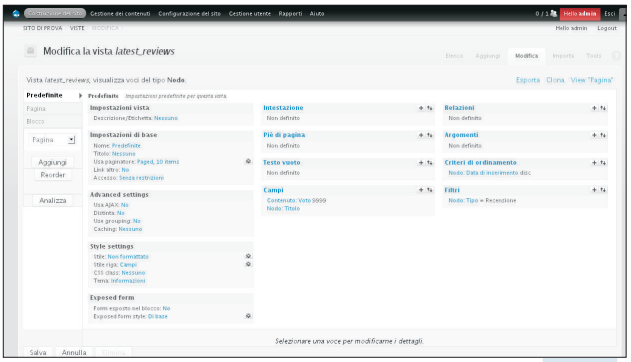
Buttiamoci nella mischia: accediamo all'URL /admin/build/views e, a seconda delle nostre impostazioni e dei moduli che abbiamo installato avremo una certa quantità di viste predefinite, ma noi vogliamo creare qualcosa di nuovo, l'elenco delle ultime recensioni pubblicate, quindi andiamo sul tab *Aggiungi* in alto a destra ed inseriamo il nome "di sistema" per questa vista, **`latest_reviews`**. Trattandosi di un nome che Views userà internamente, potrà contenere solo lettere (per convenzione, solo le minuscole), numeri ed il carattere underscore.

Tralasciamo per il momento i campi *Descrizione* ed *Etichetta*, assicuriamoci che il *Tipo di vista* sia impostato su *Nodo* e facciamo click su *Avanti*. A questo punto ci troveremo davanti all'interfaccia di gestione di una vista (figura 4), estremamente ricca e dettagliata. Il nostro obiettivo è piuttosto semplice: vogliamo creare un elenco contenente solo *titolo* e *voto* delle ultime recensioni, ordinate a partire dalla più recente.

Assicuriamoci che lo *Stile riga* sia impostato su *Campi* e, nell'omonimo pannello situato nella seconda colonna, selezioniamo quali campi recuperare. Partiamo aggiungendone uno con un click sull'icona + sulla destra del pannello, poi scendiamo fino a trovare l'elenco di tutti i possibili campi e selezioniamo dalla casella *Gruppi* la voce *Nodo* e scorriamo la lista fino a trovare la voce *Nodo: Titolo*.

Confermiamo la scelta e concentriamoci sul *Voto* che, essendo fornito dal modulo CCK, si trova nel gruppo *Contenuto*: cambiamo la nostra selezione ed andiamo a cercare *Contenuto: Voto (field_vote)*. Views non è a conoscenza dei vari tipi di nodi disponibili, quindi quando scegliamo il gruppo *Contenuto*, ci elencherà tutti i campi che abbiamo aggiunto attraverso CCK, senza distinzione sul tipo di nodo a cui sono associati. Ad esempio, supponiamo di avere un tipo di nodo *Prodotto*, a cui abbiamo associato un campo *Immagine*, in cui inserire una foto. Ora immaginiamo di voler creare una vista che elenchi **tutti** i contenuti del nostro sito, di qualsiasi tipo siano: le *Recensioni* non contengono il campo *Immagine* e, viceversa, i *Prodotti* non contengono il campo *Voto*, ma Views ci renderà disponibili entrambi i campi. Se li aggiungeremo alla vista, dovremo tenere conto del fatto che uno dei due campi sarà vuoto, a seconda del tipo di nodo, e dovremo gestirlo opportunamente quando scriveremo il template.

Comunque, per noi è tutto facile, abbiamo deciso di lavorare solo con le recensioni, quindi possiamo scegliere senza paura il campo




Interfaccia di gestione di una vista

4

*Voto* e fare click su *Aggiungi*.

Views ci chiederà di configurare i campi che abbiamo selezionato: per il *Voto* lasciamo le impostazioni che ci vengono suggerite e clicchiamo *Aggiorna*, poi cancelliamo la parola Titolo dall'Etichetta per il campo Titolo, in modo che non compaia nell'elenco e deselezioniamo le voci *Trim only on a word boundary* e *Add an ellipsis* (la traduzione di queste voci non era completa, nella nostra installazione). Infine assicuriamoci che la voce *Collega questo campo al suo nodo* sia attivata, per far sì che Drupal si occupi di trasformare il titolo in un link alla pagina della recensione. Confermiamo con l'ennesimo click su *Aggiorna*.

Se scendiamo fino alla sezione *Anteprima in diretta* dovremmo trovare un elenco di nodi... il problema è che saranno di ogni tipo: non solo recensioni, ma anche normali pagine e notizie! Per ottenere solo le recensioni dovremo tornare all'interfaccia di configurazione della vista e cercare il pannello Filtri, per aggiungerne (pulsante +) uno: scendiamo fino a trovare l'elenco di quelli a disposizione e, dato che a noi interessa filtrare i nodi per tipo, dalla casella *Gruppi* selezioniamo *Nodo* e cerchiamo nella lista sottostante la voce *Nodo: Tipo*, facciamo click su *Aggiungi* e, nella pagina di configurazione che apparirà di seguito, selezioniamo l'operatore *"È uno di"* ed il tipo di nodo *Recensione*. Dopo un click su *Aggiorna*, se tutto è andato per il verso giusto, l'elenco nell'anteprima in diretta dovrebbe mostrarci esclusivamente le recensioni che abbiamo creato nella scorsa puntata. **Attenzione:** per verificare tutti i passaggi sarà necessario disporre di più di dieci recensioni, quindi in caso aggiungetene qualcuna.

Tornando alla vista, manca solo l'ordinamento per data. Basta aggiungerlo (col solito pulsante +) nel pannello *Criteri di ordinamento*, scegliendo dall'elenco la voce *Nodo: Data di inserimento*, impostando *Sort descending* e confermando con un click su *Aggiungi*. Aggiornando la pagina, vedremo che la lista è ordinata in base alla data d'inserimento.

## Pagine e blocchi

Vi ricordate che avevamo detto che avremmo creato un blocco? Siamo quasi pronti! Prima istruire Views su come vogliamo che la vista venga presentata ai visitatori del sito, è bene fare clic sul pulsante *Salva* situato appena sopra l'anteprima in diretta, per salvare il lavoro fatto finora. Fatto questo, vediamo di creare una modalità di visualizzazione di tipo *Pagina*. Facciamo click su *Aggiungi* nel pannello delle visualizzazioni (la voce *Pagina* sarà selezionata di default) e vedremo che la prima colonna cambierà: ora possiamo passare tra la visualizzazione *Predefinita* e quella *Pagina* usando i due tab corrispondenti. La modalità *Predefinita* permette di configurare le impostazioni di default per tutte le altre visualizzazioni.

Faremo solo un aggiustamento nel pannello *Impostazioni pagina*, impostando il *Path* su *Nessuno* e scrivendo **`archivio-recensioni`** nel campo che comparirà subito sotto. Confermiamo con un click su *Aggiorna*: in questo modo, i nostri visitatori potranno accedere alla pagina attraverso l'URL /archivio-recensioni: questa pagina mostrerà dieci titoli alla volta insieme al rispettivo voto e conterrà anche i comandi di paginazione, se le recensioni sono più di dieci.

**Attenzione:** quando un parametro può essere sovrascritto e tentiamo di modificarlo da un'altra vista, per default la modifica andrà a cambiare il parametro nella visualizzazione *Predefinita*. Se vogliamo cambiarlo solo per la vista corrente dobbiamo esplicitarlo attivando il pulsante *Sovrascrivi*.

Torniamo al pannello delle visualizzazioni per creare il nostro *Blocco*: selezioniamo la voce dal menu a tendina (dove prima abbiamo selezionato *Pagina*) e facciamo click su *Aggiungi*, poi impostiamo come titolo *Ultime recensioni*, dopo aver spuntato nel pannello *Impostazioni di base* la voce *Sovrascrivi per il titolo*. Clicchiamo *Aggiorna* per confermare e passiamo alla voce *Usa paginatore*: ricordiamoci di spuntare l'opzione *Sovrascrivi* e selezioniamo *Display a specified number of items*. Un click su *Aggiorna* e comparirà un secondo pannello in cui dovremo inserire quante voci visualizzare (**`5`**) e quante saltarne a partire dalla prima (**`0`**). Aggiorniamo di nuovo e, soddisfatti, salviamo la vista: siamo pronti ad inserire il nostro blocco nel sito!

## Aggiungere un blocco al sito

Dirigiamoci all'URL /admin/build/block e cerchiamo il blocco generato da Views, che si chiamerà **`latest_reviews: Blocco`**, poi inseriamolo in

### R2 – Il pannello di configurazione di Views

Il pannello di configurazione di una vista è composto da tre parti fondamentali: la prima colonna all'estrema sinistra ci consente di aggiungere, selezionare e rimuovere le visualizzazioni; la seconda colonna da sinistra, in cui possiamo configurare i parametri di presentazione e, infine, le ultime due colonne a destra, in cui possiamo configurare i parametri della query che Views andrà ad eseguire sul database.

Le visualizzazioni sono i diversi modi in cui Views può renderizzare i dati che gli chiediamo di recuperare: come lista in un blocco, come insieme di pagine, legate da un paginatore, come griglia in una pagina (ad esempio per una galleria d'immagini) o anche come feed RSS.

Selezionando una delle visualizzazioni, i pannelli sulla destra possono subire dei cambiamenti, ma l'impianto generale rimane il medesimo; per esempio, la seconda colonna da sinistra contiene sempre i parametri di presentazione, cioè come vogliamo che Views presenti i dati ai visitatori. In questa sezione, i valori che più probabilmente andremo a modificare sono il Titolo, il Numero di risultati da mostrare e l'eventuale uso di un paginatore, lo Stile di presentazione e, nel caso delle pagine e dei feed, anche l'URL attraverso il quale saranno disponibili.

Le ultime due colonne ci permettono di configurare i parametri della query: abbiamo visto nell'articolo come selezionare solo i campi che ci interessano usando il pannello *Campi*, come filtrare l'elenco dei risultati secondo certi parametri usando il pannello *Filtri* e come specificare un ordinamento usando il pannello *Criteri di ordinamento*.

Andiamo con ordine ad esaminare quelli che restano: il pannello *Intestazione* ci permette di inserire del testo libero in cima alla visualizzazione, utile per aggiungere una spiegazione della lista. Allo stesso modo, il pannello *Piè di pagina* ci permette di inserire del testo libero in fondo alla lista. *Testo vuoto* permette di inserire una risposta predefinita nel caso la query non avesse restituito alcun risultato.

Il pannello *Argomenti* è quello che rende veramente interessante Views: nella prima puntata abbiamo visto come, attraverso la *Tassonomia*, possiamo filtrare i contenuti per categorie e tag, ma se assegniamo lo stesso tag ad una pagina e ad una recensione, queste verranno elencate insieme nella pagina del tag. Tramite questo pannello, possiamo creare una vista filtrata anche in base a categorie e tag: quando impostiamo un URL per accedere ad una visualizzazione pagina (o feed) della vista, possiamo anche impostarne alcuni segmenti variabili usando il carattere **`%`**. Per esempio nel nostro caso avremmo potuto specificare l'URL **`archivio-recensioni/%`** in modo da poter usare un eventuale secondo termine come filtro, trasformando l'URL in /archivio-recensioni/schede-video, per ottenere la lista delle ultime recensioni riguardanti le schede video, oppure /archivio-recensioni/schede-audio per le recensioni relative alle schede audio.

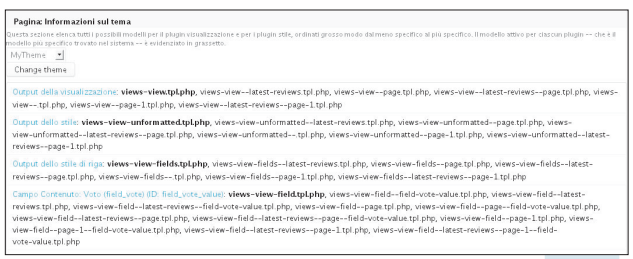
E' possibile ottenere lo stesso effetto anche per i blocchi, ma in quel caso è necessario scrivere il codice per estrarre i parametri dall'URL e non approfondiremo l'argomento.

Ci sarebbe molto da dire anche sul pannello *Relazioni*, che ci permette di recuperare contenuti non direttamente riconducibili alla vista, ma che potrebbe interessarci visualizzare insieme ad essa, ad esempio per poter creare una view che elenchi gli ultimi contenuti del sito, raggruppati per autore e, in corrispondenza di ognuno, un altro elenco di articoli dello stesso autore (o della stessa categoria).

una regione a piacere, per esempio nella barra laterale destra.
Salviamo e torniamo alla home page o in una qualsiasi delle pagine interne del sito: troveremo la nostra bella lista di recensioni! Possiamo inserire un link che conduca alla pagina che mostra tutte le recensioni: per farlo bisogna modificare la visualizzazione (nell'interfaccia di modifica della vista), ma questo lo lasciamo come esercizio per il lettore...
C'è un piccolo bug: il blocco ci verrà mostrato anche quando visitiamo l'archivio delle recensioni (la visualizzazione in stile Pagina di quella stessa view!). E' possibile limitare la visualizzazione di un blocco a certe pagine: torniamo all'URL /admin/build/block e facciamo clic sul link *Configura*, passiamo alla scheda *Impostazioni specifiche di visibilità per pagina*, assicuriamoci che la voce *Mostra su ogni pagina eccetto quelle elencate* sia selezionata ed inseriamo nella casella *Pagine* la voce **`archivio-recensioni*`** (l'asterisco serve per catturare tutti gli URL che iniziano con quella stringa).
Salviamo e dirigiamoci su /archivio-recensioni, dove possiamo verificare che il blocco è effettivamente sparito. Se passiamo alla home page o ad una qualsiasi altra pagina interna, il blocco tornerà visibile.

## Personalizzare il markup

Visitando la pagina dell'archivio ci accorgiamo subito che i campi sono disposti nell'ordine in cui li abbiamo impostati nell'omonimo pannello dell'interfaccia di configurazione della vista. Se questo ordine non ci soddisfa, possiamo modificarlo con un click sul pulsante che si trova accanto a quello per aggiungere un nuovo campo, cioè quello con le due frecce.
Il markup con cui Views ha renderizzato la vista è tutto un susseguirsi di **`div`** annidati che non conferiscono al contenuto alcun tipo di struttura semantica, ma per fortuna Views offre alcune funzionalità per la personalizzazione del rendering e supporta il meccanismo di template suggestion, così da permetterci di adattare alle nostre necessità il rendering dei dati, qualora non trovassimo uno stile predefinito che ci soddisfi.
Torniamo nell'interfaccia di configurazione della nostra vista, selezionando la visualizzazione *Pagina* e concentriamoci sul pannello *Style settings*: qui possiamo impostare lo stile di visualizzazione, per esempio come griglia, come tabella, come lista o senza formato (come nel nostro caso). Selezionando uno di questi formati, aggiornando e facendo poi un click sul pulsante che compare a destra dell'impostazione (quello col piccolo ingranaggio), possiamo modificarne ulteriormente il comportamento. Se selezioniamo lo stile *Non formattato*, ci viene chiesto quale dev'essere lo stile della riga, cioè il modo in cui viene renderizzato ciascun risultato. Per il nostro esempio abbiamo scelto *Campi*, ma è possibile anche scegliere *Nodo* ed ottenere così il rendering dell'intero nodo.
Infine troviamo la voce *Tema*, che è quella in cui entrano in gioco le template suggestion: con un click su *Informazioni* ci verrà proposto un elenco di tutti i template (figura 5) attraverso cui passano i dati per essere renderizzati e, per ciascun passaggio, ci verranno elencate tutte le template suggestion che Views ci mette a disposizione, evidenziando in grassetto quella che sta usando in questo momento. Se volessimo modificare il markup con cui viene renderizzato il titolo del nodo nella visualizzazione *Pagina*, dovremmo creare all'interno della directory del nostro tema un file chiamato **`views-view-field--latest-reviews--page-1--title.tpl.php`**.
Per avere un'idea del markup usato da Views e per sapere quali variabili abbiamo a nostra disposizione, facciamo click sul link all'inizio della riga corrispondente al template che vogliamo sovrascrivere, in questo caso il link è *Campo Nodo: Titolo (ID: title)* e ci comparirà il codice predefinito che possiamo copiare nel nostro template. Al termine delle modifiche ricordiamoci che dobbiamo rigenerare il registro dei temi.
Nella maggior parte dei casi il markup predefinito e gli stili forniti con Views andranno bene, ma dovessimo modificarlo, possiamo creare dei nuovi stili di riga come plugin di Views oppure usare lo stile *Non formattato* e sovrascrivere i template predefiniti. La strada da preferire sarebbe la prima perché è concettualmente la più corretta, tuttavia spesso è prassi creare stili piuttosto complicati e diversi da quelli predefiniti facendo uso esclusivo delle template suggestion.



L'elenco dei template utilizzati nel rendering di una vista **5**

## Tirando le somme

Il miglior modo per imparare ad usare uno strumento è usarlo (oltre che studiarne la documentazione!!). Con questa seconda puntata abbiamo visto che Drupal dispone di strumenti di grandissima flessibilità e l'ideale per comprenderli appieno è sperimentare. Buon divertimento!