



ZPT-Zen

L'interfaccia web secondo Zope

m.delmonte@tin.it

Maurizio Delmonte, Paolo Dina

ZPT: una tecnologia di principi

Zope Page Template cerca di risolvere le problematiche di integrazione tra sviluppatori e designer adottando tre principi base:

- permettere l'uso approfondito dei normali strumenti di editing;
- fare in modo che quello che si vede sia molto simile a quello che si ottiene;
- Tenere il codice fuori dai template con l'eccezione delle strutture logiche.

Due piccioni con una fava

TAL (Template Attribute Language) non infastidisce gli editor *wysiwyg*, che i designer potranno utilizzare senza troppi problemi anche dopo il lavoro degli sviluppatori.

Gli editor *wysiwyg* non infastidiscono i designer, mostrando l'HTML valido degli ZPT nella forma che loro si aspettano.

ZPT: pregi e... '*difetti*'

Mediante comandi ZPT si possono rimpiazzare interi tag, il loro contenuto, o solo alcuni attributi. Si può ripetere il tag più volte od ometterlo completamente.

Non si possono creare sotto-procedure o classi, scrivere cicli o test a più vie, o esprimere facilmente algoritmi complessi.

La logica deve stare *fuori* dall'interfaccia!

Fenomenologia di ZPT

TAL (Template Attribute Language) – attributi XML del namespace TAL, applicati ad un documento XML o HTML lo rendono un template.

TALES (TAL Expression Syntax) – espressioni capaci di fornire dati alle istruzioni TAL e METAL.

METAL (Macro Expansion TAL) – strumento di macro preprocessing, consente la modularizzazione dei template.



Il trucco c'è e si vede!

ZPT è un modello della pagina da generare, a sua volta HTML valido:

```
<title tal:content="here/title">Page Title</title>
```

l'editor *wysiwyg* ignorerà la parte dinamica, poiché “*tal:*” viene interpretato come *namespace XML*, inoltre farà comparire “Page Title” nella forma definita dal designer.

Il simpatico TAL

Permette di definire la struttura dei nostri template, semplicemente aggiungendo attributi ai mock-up HTML:

Prima

```
<p>
```

```
  l'url che hai richiesto è:
```

```
  <b>url richiesta</b>.
```

```
</p>
```



L'url che hai chiesto è: **url richiesta**.

Dopo

```
<p>
```

```
  l'url che hai richiesto è:
```

```
  <b tal:content="request/URL">url richiesta</b>.
```

```
</p>
```



L'url che hai chiesto è: **url richiesta**.

L'url che hai chiesto è: **http://...**

L'indispensabile TALES (1)

TALES fornisce le espressioni necessarie a dare vita ai nostri template.

Esistono vari tipi di espressioni: possiamo utilizzare delle canoniche **path expression**...

request/URL

URL della richiesta web corrente

user/getUserName

username dell'utente responsabile della richiesta

Container/objectIds

lista degli Id degli oggetti nella stessa Cartella che contiene il template corrente

L'indispensabile TALEs (2)

... possiamo invocare utili **string expression** per formattare stringhe da utilizzare nei template ...

```
<b tal:content="string:titolo: ${here/title}"/>
```



```
<b>titolo: doc1</b>
```

... o **python expression** per sfruttare nei template la flessibilità del python senza ricorrere a script esterni:

```
<div tal:replace="python:here.getSize() / 100"/>px
```



```
240px
```

L'indispensabile TALES (3)

La colla che fa funzionare il tutto sono delle parole chiave che possiamo sempre utilizzare:

Nothing

rappresenta un non-valore (analogo a None, void, Null, ...)

here

l'oggetto al quale il template viene “*applicato*”

template

il template stesso

request

l'oggetto REQUEST di Zope

user

l'oggetto che rappresenta l'utente registrato

(...)

TAL in pillole: inserire testo

Funzionalità minima richiesta ad un template language, TAL gioca con i tag definendone il contenuto...

<H1 tal:content="here/title"/>



<H1>Homepage</H1>

... oppure direttamente rimpiazzandoli:

<H1 tal:replace="here/title">Titolo</H1>



Homepage

nb: attenzione a cosa chiedete ;) il tag H1 è sparito dalla pagina!

TAL in pillole: strutture cicliche

Il vantaggio di un template è quello di potersi fermare alla forma... TAL offre l'istruzione repeat, semplice e potente:

```
<table border="1" width="100%">  
  <tr tal:repeat="item container/objectValues">  
    <td tal:content="repeat/item/number">#</td>  
    <td tal:content="item/id">Id</td>  
    <td tal:content="item/title">Title</td>  
  </tr>  
</table>
```



Con un po' di fantasia vedrete formarsi una tabella, la cui prima colonna riporta il numero della riga corrente.

TAL in pillole: condizioni

Quando siete ad un bivio, TAL vi permette di gestire le vostre decisioni mediante *tal:condition*

```
<table border="1" width="100%"  
  tal:condition="container/objectValues">  
  <tr tal:repeat="item container/objectValues">  
    <td tal:content="repeat/item/number">#</td>  
    <td tal:content="item/id">Id</td>  
    <td tal:content="item/title">Title</td>  
  </tr>  
</table>
```



In questo caso, la tabella si formerà solo se *container/objectValues* fornisce effettivamente degli oggetti...

TAL in pillole: definire variabili

Per migliorare l'usabilità e la leggibilità dei template non si può fare a meno di utilizzare *tal:define*, ottenendo:

```
<table border="1" width="100%"  
  tal:define="object_list container/objectValues"  
  tal:condition="object_list">  
  <tr tal:repeat="item object_list">  
    <td tal:content="repeat/item/number">#</td>  
    <td tal:content="item/id">Id</td>  
    <td tal:content="item/title">Title</td>  
  </tr>  
</table>
```



object_list potrà essere utilizzato ovunque nel contesto del tag *table*

TAL in pillole: gli attributi

Quando iniziate a chiedervi come formattare una stringa all'interno di un tag HTML è il momento di *tal:attributes...*

```
<td>  
<span tal:replace="item/meta_type">MType</span>  
</td>
```



Il valore dell'attributo *src* sarà sostituito dal risultato dell'espressione *item/icon*, ma l'editor wysiwyg caricherà l'immagine *icon.gif* nella cartella *images*

TAL in pillole: chicche (1)

tal:content e *tal:replace* quotano il testo che inseriscono (< sostituisce < ...), per evitarlo è sufficiente usare *structure*, unito a *tal:omit-tag* se si desidera eliminare il tag contenitore:

```
<div tal:define="ciao string:<b>ciao</b>"  
  tal:omit-tag="nothing">  
  <span tal:replace="ciao"/>  
  <span tal:replace="structure ciao"/>  
</div>
```



```
&lt;b&gt;ciao&lt;b&gt;  
<b>ciao</b>
```

potete facilmente immaginare come un browser web vi mostrerà il risultato :)

TAL in pillole: chicche (2)

Editare un template in un editor *wysiwyg* è poco utile se non si utilizzano dei segnaposto, che scompariranno a run-time:

```
<tr tal:replace="nothing">  
  <td>nome</td> <td>cognome</td>  
</tr>
```



in questo modo, potremo vedere nell'editor il numero di righe che ci si aspetta una tabella abbia durante l'esecuzione...

TAL in pillole: gestire errori

Dopo aver preso confidenza con i rudimenti, vorrete qualcosa di più, e tal può offrirvi *tal:on-error...*

```
<b tal:on-error="string:username non definito"  
  tal:content="here/getUsername">Mauro</b>
```



Se *here/getUsername* restituisce errore:
username non definito

```
<b tal:on-error="nothing"  
  tal:content="here/getUsername">Mauro</b>
```



Se *here/getUsername* restituisce errore
verrà eliminato l'intero tag

TAL: ...e non finisce qui

Molto altro dovrebbe essere detto sulle possibilità offerte da TAL, ma *non è più tempo...*

Lo *ZopeBook* (<http://zope.org/Documentation>) offre due interi capitoli su ZPT oltre a ZPT reference.

Molti validi tutorial sono sparsi nella rete:
Google è il vostro migliore amico!

METAL: quando il gioco si fa duro

Zope è incentrato sugli oggetti e sul concetto di **riusabilità** del codice.

Una applicazione web mostra di solito un'interfaccia piuttosto ricca, ma basata su elementi facilmente individuabili.

METAL offre la tecnologia necessaria a poter definire e richiamare tali elementi nei vari template che costituiscono la nostra applicazione.

METAL: le macro (1)

Una macro è una sezione di una pagina definita in modo da essere richiamata in altre pagine.

```
<p metal:define-macro="copyright">  
  Copyright 2001,  
  tal:content="here/author"  
  Zope.it</em>  
</p>
```

Richiamando la macro in un determinato punto di una pagina, si avrà l'*espansione* della macro in quel punto, come se *in quel punto* fosse definita...

METAL: le macro (2)

La macro precedente potrà essere richiamata dovunque se ne abbia bisogno:

```
<b metal:use-macro="container/master/macros/copyright">
```

Qui sarà espansa la macro

```
</b>
```



```
<p>
```

Copyright 2001,

```
<em>m.delmonte</em>
```

```
</p>
```

Si noti che il tag viene completamente sostituito.

METAL: gli slot (1)

Uno slot è una sezione di una *macro* definita in modo da essere *riempita* in base al contesto.

```
<div metal:define-macro="sidebar">  
  <h1>Links</h1>  
  <ul metal:define-slot="links">  
    <li><a href="/">A Link</a></li>  
  </ul>  
</div>
```

In questo modo è possibile lasciare sezioni vuote da utilizzare all'occorrenza...

METAL: gli slot (2)

Utilizziamo uno slot per riempire all'occorenza un box contenente dei link.

```
<div metal:use-macro="here/doc1/macros/sidebar">
```

```
<h1>Links</h1>
```

```
<ul metal:fill-slot="links">
```

```
<li><a href="http://linux.org">Linux</a></li>
```

```
<li><a href="http://zope.org">Zope</a></li>
```

```
</ul>
```

```
<div>
```



```
<h1>Links</h1>
```

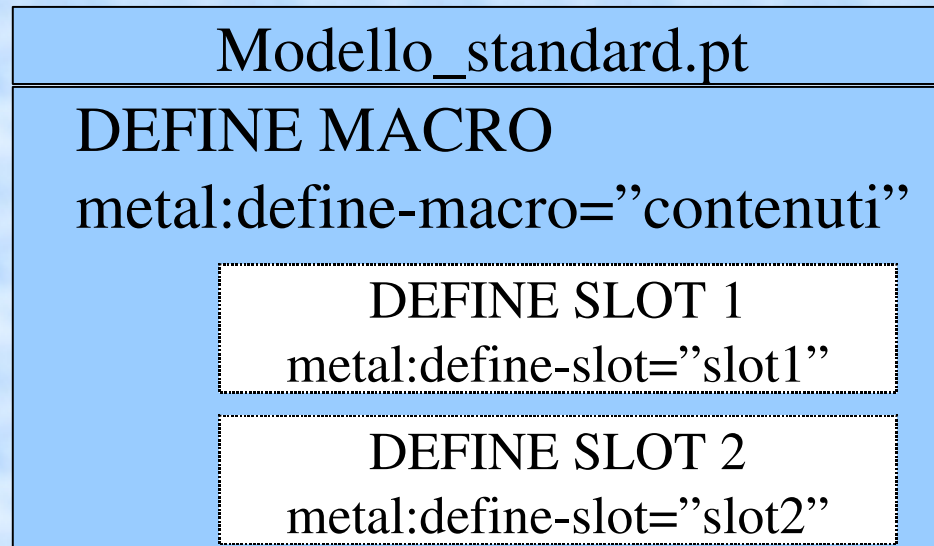
```
<ul>
```

```
<li><a href="http://linux.org">Linux</a></li>
```

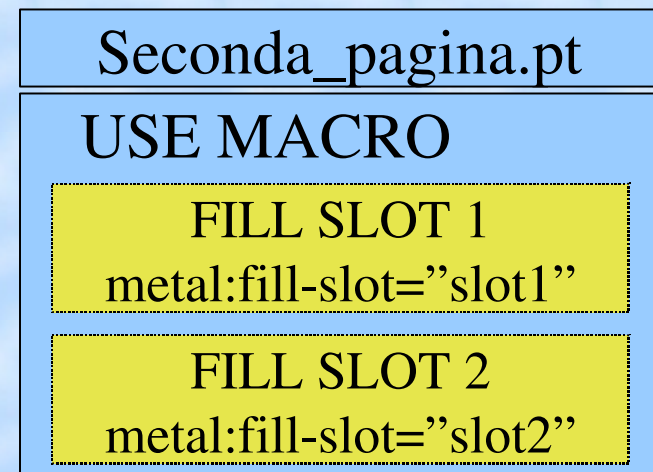
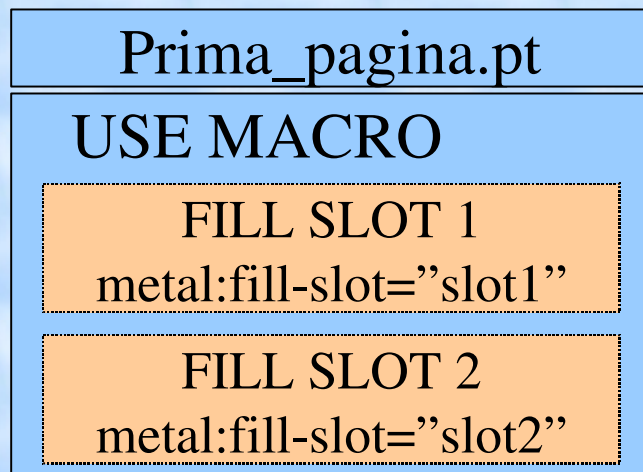
```
<li><a href="http://zope.org">Zope</a></li>
```

```
</ul>
```

METAL in sintesi



Metal:use-macro="here/modello_standard/macros/contenuti"



Grazie per l'attenzione!



Zope

Maurizio Delmonte
m.delmonte@tin.it

Paolo Dina