Linux User Group Pordenone

Michele Beltrame
mb@italpro.net
http://www.massainforme.it/mb/

Nordest Perl Mongers
http://nordest.pm.org

Templating XML in Perl

A prova di web designer!

Contenuti

- **Templating per il web**
- **Caratteristiche di Petal**
- **TAL**
  - **Sostituzioni**
  - **Condizioni**
  - **Cicli**
  - **Metal**
  - **Modificatori**
  - **Oggetti**
- **Internazionalizzazione**
- **Futuro di Petal**

Templating per il web

Dovrebbe essere:

- **Leggero => veloce**
- **Portabile (provider...)**
- **A prova di web designer**

Leggero

- **Parser non in Perl (scritto in C e compilato)**
- **Caching in memoria dei template**
- **Caching su disco dei template**

Portabile

Dovrebbe funzionare presso il maggior numero di ISP (PC) possibile.

- **Non deve utilizzare codice compilato (modulo Pure Perl)**
- **Deve dipendere da pochi altri moduli**
- **Deve funzionare anche con versioni meno recenti di Perl**

A prova di web designer

- **(X)HTML compliant**
- **Niente tag proprietari**
- **Attributi XML di tag esistenti**
- **Supporto placeholder**

Petal 1

- **Implementazione Perl di TAL (Template Attribute Language)**
- **TAL è un'idea di ZOPE Corporation (evil Python enemy)**
- **TAL utilizza gli attributi dei normali tag XHTML**

Petal 2

- **Veloce: utilizza cache in memoria e disco**
- **Pure Perl: portabile quasi ovunque (al prezzo di minor velocità di parsing)**
- **Funziona con Perl da 5.005 in su (UTF-8 da 5.6)**

Sostituzioni 1

Sostituzioni semplici

- **È possibile sostituire con una variabile il contenuto (placeholder) di un tag**
- **È possibile sostituire con una variabile un intero tag**
- **Al posto di una variabile si possono utilizzare delle stringhe**

Sostituzioni 2

```
<p tal:content="persona">Nome persona</p>

<p>Astor</p>

<p tal:replace="string:Sei ${persona}!">
  Nome persona
</p>

Sei Astor
```

Sostituzioni 3

```
my $t = new Petal (
   file => './sostituzioni.xhtml'
);

my $output = $t->process (
  persona => 'Astor',
)

print $output;
```

Attributi 1

È possibile impostare un attributo di un tag:

```
<a href="#" tal:attributes="href url">
  Clicca qui
</a>

<td bgcolor="#ffffff" align="left"
  tal:attributes="bgcolor cella/colore;
    align cella/allineamento"
  tal:content="cella/contenuto"
>
  Contenuto
</td>
```

Attributi 2

```
my $t = new Petal (
   file => './attributi.xhtml'
);

my %cella = (
   colore       => '#gg44dd',
   allineamento => 'right',
   contenuto    => 'Regina',
);

print $t->process (
   url   => 'http://www.massainforme.it',
   cella => \%cella,
)
```

Condizioni 1

- **tal:condition permette di visualizzare tutto ciò che si trova all'interno di un tag al verificarsi di una condizione**
- **Questa direttiva può anche essere utilizzata per verificare se una variabile è definita e contiene qualcosa**
- **Ulteriori istruzioni (elsif, else, ...) si trovano in Petal::Utils**

Condizioni 2

```
<b tal:condition="true:asso">
  Hai pescato l'asso!
</b>

<b tal:condition="false:asso; true:carta"
  tal:content="carta"
>Carta
</b>
```

Condizioni 3

```
my $t = new Petal (
   file => './condizioni.xhtml'
);

print $t->process (
   asso          => 0,
   carta         => 'Fante',
)
```

Cicli 1

Con Petal è possibile gestire loop, anche uno dentro l'altro.

```
<table border="1">
  <tr tal:repeat="riga lrighe">
    <td tal:repeat="cella riga/lcelle"
     tal:content="riga/contenuto"
    >Contenuto</td>
  </tr>
</table>
```

Cicli 2

```
my $t = new Petal (   file => './cicli.xhtml');

my @lrighe = (
   { lcolonne => [
      contenuto => 'R1 C1',
      contenuto => 'R1 C2',
   ] },
   { lcolonne => [
      contenuto => 'R2 C1',
   ] }
);

print $t->process (
   lrighe     => \@lrighe,
)
```

METAL

Inclusione di codice da altri file

Definizione (in *com.xhtml*):

```
<span
 metal:define-macro="email"
 tal:content="dati/email"
>E-Mail</span>
```

Inclusione:

```
<span
 metal:use-macro="com.xhtml#email"
>E-mail</span>
```

Modificatori 1

- **È facile estendere facilmente Petal defininendo dei modificatori propri.**
- **Due metodi diversi per creare un modificatore:**
  - **reference a codice**
  - **modulo a sé stante**
- **Il modulo Petal::Utils contiene una serie di modificatori già pronti all'uso.**

Modificatori 2

```
Numero di secondi dall'epoch:
<span tal:replace="stamp">
  Timestamp
</span>
Data ed ora di oggi:
<span tal:replace="formatdate:stamp">
  Data ed ora
</span>
```

```
Numero di secondi dall'epoch: 1079428352
Data ed ora di oggi: 16/03/2004, 09:15:12
```

Modificatori 3

```
# Reference a codice
$Petal::Hash::MODIFIERS->{'formatdate:'} =
sub {
    my $hash = shift;
    my $args = shift;
    my $result = $hash->fetch ($args);
    my @ltime = gmtime($result);
    return sprintf("%02d/%02d/%04d", $ltime
      [3], $ltime[4]+1, $ltime[5]+1900);
};
```

Oggetti

Si possono chiamare i metodi di oggetti, passando anche dei parametri.

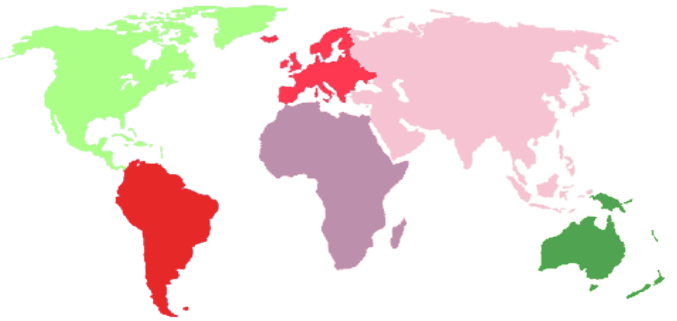
```
Mario: <p tal:content="altmario">alt</p>
Gaia: <p tal:content="altgaia">alt</p>
Lucia: <p tal:content="altlucia">alt</p>

Altezza media:
<p
  tal:content="MiaStatistica/mediaalt mario
altgaia altlucia"
>
  media
</p>
```

Internazionalizzazione

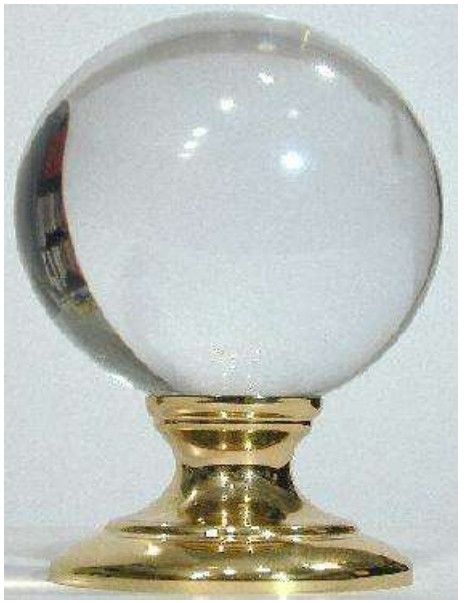
Petal può scegliere in automatico i template della lingua desiderata.

```
# Cerca nella directory indice/
# nell'ordine:
# pt-BR.xhtml
# pt.xhtml
# en.xhtml
My $template = new Petal (
   file             => './indice',
   lang             => 'pt-BR',
   default_language => 'en'
);
```



Futuro

- **Petal fino a poco tempo fa era sviluppato da Jean Michel Hiver (MKDoc), ora è passato in mano ad altri sviluppatori.**
- **Feature previste:**
  - **Integrazione con framework quali HTML::Mason, AxKit, ...**
  - **Miglioramento del supporto al linguaggio TAL**
  - **Nuovo e migliore parser XML**
  - **Potenziamento internazionalizzazione**



Dimmi di più!

- **Mailing list per gli utilizzatori di Petal (in inglese):**
  - **http://lists.webarch.co.uk/mailman/listinfo/petal**
- **Per qualche consiglio in italiano:**
  - **Nordest Perl Mongers (http://nordest.pm.org)**
  - **#nordest.pm su SlashNET**
  - **Perl.It (http://www.perl.it)**

PROGRAMMING REPUBLIC OF
PERL