eZ publish, un CMS Open Source di classe Enterprise

Introduzione di base all'estensibilità del sistema

a cura di **Francesco Trucchia <francesco@cphp.it>**

un **phpBreakfast** offerto dal **GrUSP**

<?php=php_info() ?>

Chi sono:

- Sviluppatore PHP dal 2001
- Laureando in Scienze dell'Informazione

Che faccio:

- Collaboro per Ser.In.Ar. con il Polo Didattico e Scientifico di Cesena per la realizzazione dei nuovi sit dei Corsi di Laurea del Polo di Cesena. Mi occupo della progettazione e implementazione dei siti sul CMS eZ publish.
- Sviluppo applicazioni Web, utilizzando architetture Open Source.

un talk

Cos’è eZ Publish

- ✓ **Un C.M.F. (Content Management Framework)**
- ✓ **Un C.M.S. (Content Management System)**

eZ CMF - Content Management Framework

- ✓Ez come C.M.F. è un sistema che incorpora elementi avanzati
  - librerie proprietarie (A.P.I.)
  - una architettura strutturata sul pattern M.V.C.
- ✓È dotato di un kernel modulare. Ogni modulo si occupa della gestione di una parte di funzionalità e l'ottima divisione lo rende facilmente estendibile, anche grazie alle numerose librerie di cui dispone
- ✓Dispone di un alto livello di estendibilità attraverso i plug-in.
- ✓Il sistema è indipendente dalla piattaforma e dal database.

un talk

Architettura strutturata su pattern MVC

Un framework strutturato su un pattern M.V.C. è un framework che separa completamente la logica di business dalla vista, il tutto gestito da un controller centrale.

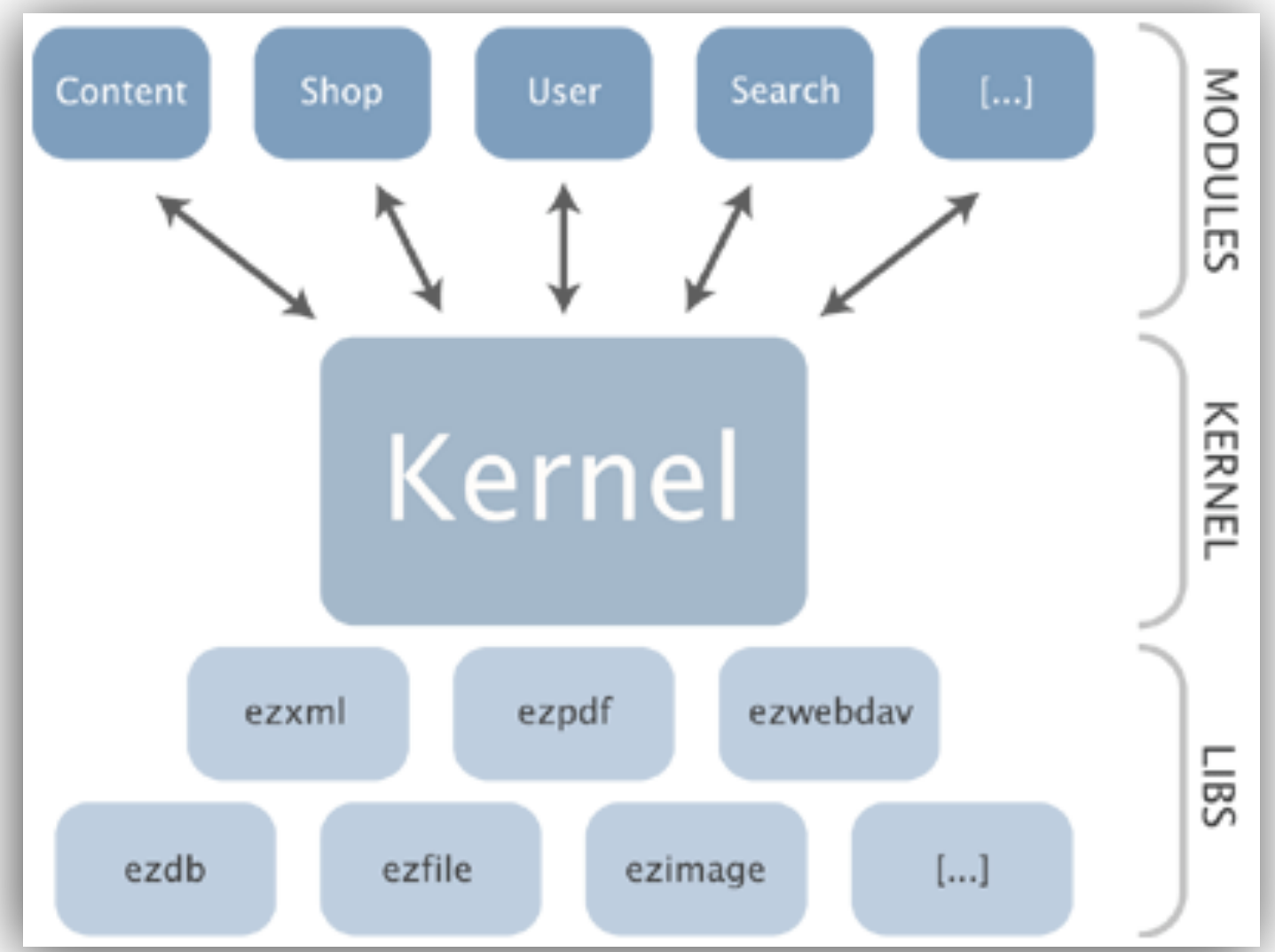
Model = Database + eZPersistentObject()

View = Template Engine + eZTemplate()

Controller = Index.php + Kernel Function

un talk

Architettura d'eZ publish



un talk

eZ publish Extension System

Plug-in system (extension/):

- Actions (dipendente dal Content Module) (actions/)
- Datatypes (dipendente dal Content Module) (datatypes/)
- Design (design/)
- Events (Nuovi eventi per il Workflow Engine) (eventypes)
- Modules (modules/)
- Settings (settings/)
- Translations (translations/)
- Template Engine Operator (autoloads/)

Attivare un’estensione

- Creare la directory extension/<myextension>
- Aggiungere il nome dell’estensione alla direttiva ActiveExtensions[] del blocco di configurazione [ExtensionSettings] del file d’override settings/override/site.ini

```
[ExtensionSettings]
ActiveExtensions[]=<myExtension>
ActiveExtensions[]=<anotherExtension>
```

- Se si vuole attivare un’estensione solo per un certo siteaccess, si deve usare la direttiva ActiveAccessExtensions[] nel file settings/siteaccess/<mydesign>/site.ini.append

```
[ExtensionSettings]
ActiveAcccessExtensions[]=<myExtension>
ActiveAccessExtensions[]=<anotherExtension>
```

Template Operator Extension (1/4)

- Con gli operatori di template è possibile chiamare qualsiasi funzione PHP all'interno dei template dell'applicazione. Creeremo un operatore che prende in input due parametri e un operatore senza parametri.
- Creiamo le directory:
  - extension/<myextension>
  - extension/<myextension>/autoloads
  - extension/<myextension>/settings
- Creiamo il file extension/<myextension>/settings/site.ini e inseriamo la seguente direttiva:

```
[TemplateSettings]
ExtensionAutoloadPath[]=myextension
```

- Questa direttiva dice al sistema di caricare anche gli operatori presenti nella directory della nostra estensione

Template Operator Extension (2/4)

- Creiamo il file extension/<myextension>/autoloads/eztemplateautoloads.php con il seguente contenuto:

```
<?php

// Operator autoloading

$eZTemplateOperatorArray = array();

$eZTemplateOperatorArray[] = array(
'script' => 'extension/<myextension>/autoloads/mystringoperators.php',
'class' => 'MyStringOperators',
'operator_names' => array( 'addstrings', 'helloworld' ) );

?>
```

Template Operator Extension (3/4)

- Creiamo successivamente il file

```
<?php
class MyStringOperators {
   ……
 /*!Executes the needed operator(s).
```

```
<?php
class MyStringOperators {
   ……
    //return the sum of two strings
    function addStrings( $string1, $string2 ){
        return $string1 . $string2;
    }
   //return a famous string
    function helloWorld()
    {
        return 'Hello World!';
    }
   ……
}
?>
```

```
?>
```

Template Operator Extension (4/4)

- Nel template a questo punto è possibile utilizzare i nuovi operatori:

```
<p>{addstrings( 'Forty', 'two' )}</p>

<p>{helloworld()}</p>
```

un talk

Module Extension (1/6)

- Creiamo i seguenti file:
  - extension/<myextension>/settings/module.ini.append
  - extension/<myextension>/modules/mymodule/module.php
  - extension/<myextension>/modules/mymodule/hello.php
  - extension/<myextension>/modules/mymodule/world.php
  - design/standard/templates/mymodule/list.tpl

- Attiviamo l'estensione nel file settings/override/site.ini.append:

```
[ExtensionSettings]
ActiveExtensions[]=<myextension>
```

- Attiviamo il modulo nel file extension/<myextension>/settings/module.ini.append

```
[ModuleSettings]
ExtensionRepositories[]=<myextension>
```

Module Extension (2/6)

- Definiamo il modulo e le sue viste nel file module.php

```
<?php

$Module = array( "name" => "MyModule" );

$ViewList = array();
$ViewList["hello"] = array( "script" => "hello.php");
$ViewList[”world"] = array( "script" => ”world.php");

?>
```

un talk

Module Extension (3/6)

- Scriviamo il contenuto del file hello.php

```php
<?php
// Module return value,
// normally fetched from template
$text = 'Benvenuti al Linux Day';
// Build module result array
$Result = array();
$Result['content'] = $text;
$Result['path'] = array(
          array( 'url' => '/mymodule/hello',
                 'text' => "Hello") );
?>
```

Module Extension (4/6)

- Scriviamo il contenuto del file world.php

```
<?php
$text = 'Hello World!!';
//Include template engine library & set template variable
include_once( 'kernel/common/template.php' );
$tpl =& templateInit();
$tpl->setVariable( 'text', $text );
// Build module result array
$Result = array();
$Result['content'] = $tpl->fetch("design:mymodule/list.tpl");
$Result['path'] = array(
          array( 'url' => '/mymodule/world',
                 'text' => "World") );
?>
```

Module Extension (5/6)

- Se vogliamo limitare l'accesso alle viste create modifichiamo il file module.php:

```
<?php

$Module = array( "name" => "MyModule" );

$ViewList = array();
$ViewList["hello"] = array("script" => "hello.php",
                           "functions" => array('read_hello'));
$ViewList["world"] = array("script" => "world.php",
                           "functions" => array('read_world'));

$FunctionList['read_hello'] = array( );
$FunctionList['read_world'] = array( );

?>
```

Module Extension (6/6)

- A questo punto possiamo accedere alle viste del nostro modulo dall'URL:

```
http://www.example.it/index.php/<mysiteaccess>/<mymodule>/hello
http://www.example.it/index.php/<mysiteaccess>/<mymodule>/world
```

Design Extensio (1/3)

- Vogliamo aggiungere un nuovo ***widget*** per le toolbar che mostri gli oggetti correlati ad un certo oggetto.
- Creiamo il file
  - extension/<myextension>/settings/design.ini.append
  - extension/<myextension>/settings/toolbar.ini.append
  - extension/<myextension>/design/standard/templates/toolbar/full/correlati.tpl
- Attiviamo l'estensione nel file settings/override/site.ini.append

```
[ExtensionSettings]
ActiveExtensions[]=<myextension>
```

- Attiviamo l'estensione del design nel file design.ini.append

```
[ExtensionSettings]
DesignExtensions[]=<myextension>
```

Design Extensio (2/3)

- Definiamo ora il nuovo widget nel file toolbar.ini.append

```
[Tool]
AvailableToolArray[]=correlati

[Tool_correlati]
correlati_classidentifiers=
section=
title=

[Tool_correlati_description]
correlati_classidentifiers=Filtro delle classi
section=Sezione
title=Titolo
```

Design Extensio (3/3)

- Creiamo il widget attravero il template correlati.tpl

```
{let node=fetch('content','node',hash( 'node_id' , $module_result.node_id))
     related=$:node.object.related_contentobject_array}

{set-block name=correlati variable=text}
{section name=Related loop=$related}
{section show=and($sectionleq($:item.section_id),
                  $correlati_classidentifierslexplode( ',' )lcontains($:item.class_identifier))}
<li>{node_view_gui view=listitem content_node=$:item.main_node}</li>
{/section}
{/section}
{/set-block}

{section show=and($related,$allegati:textleq('')lnot)}
<div class="correlati">
<h2>{$title}</h2>
<ul >{$correlati:text}</ul>
</div>
{/section}
{/let}
```

Riferimenti

- **eZ System**

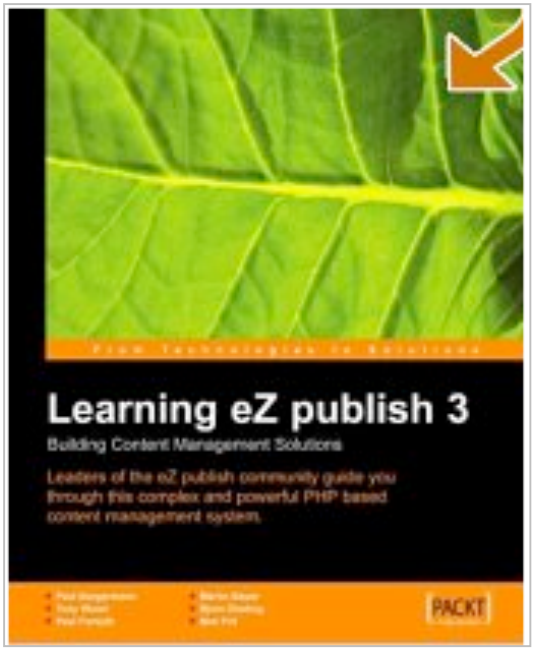
  http://www.ez.no

- **eZ Publish**

  http://ez.no/products/ez_publish_cms

- **eZ documentation**

  http://ez.no/products/ez_publish_cms/documentation



un talk

Contatti

Francesco Trucchia

francesco@cphp.it

trucchia

http://www.cphp.it

http://wiki.grusp.it

un talk