Começando com Zend Framework

Versão brasileira da apresentação disponível em
http://www.slideshare.net/baohx2000/zend-framework-19-setup-using-zendtool

Ou:
Como eu aprendi a parar de me preocupar
e amei o framework.

Parte 1: Configurando

Tradutor e adaptador para a versão 1.11 do Zend Framework: Flávio Gomes da Silva Lisboa

www.fgsl.eti.br

Preparando o Zend Framework

- Download ZF1.11

  `http://framework.zend.com/releases/ZendFramework-1.11.0/ZendFramework-1.11.0.zip`

- Descompacte em um diretório da sua preferência
- Adicione o diretório que contém a pasta Zend na diretiva include_path do arquivo php.ini.
- Alternativa 1: Cole a pasta dentro do diretório library do seu projeto
- Alternativa 2: Crie um link simbólico na pasta library do seu projeto para a pasta Zend

Preparando o Zend Framework (Continuação)

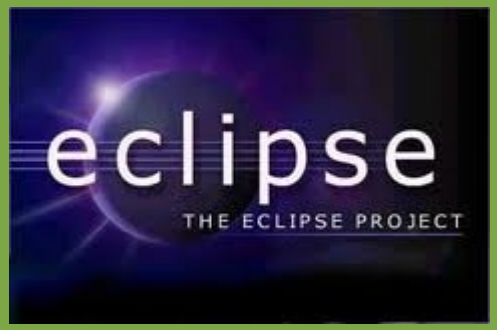
- Copie os arquivos zf.php e zf.sh (ou zf.bat no Windows) para um diretório no caminho de busca do sistema operacional (no Linux, /usr/bin)
- (Só para Linux) Renomeie o zf.sh para zf e torne-o executável (sudo chmod a+x zf)

Usando Zend_Tool para configurar seu projeto

- Crie um diretório para hospedar seus projetos
  - `mkdir projects`
- Entre nesse diretório
  - `cd projects`
- Crie um projeto usando o script zf
  - `zf create project {projectname}`

Integre Zend_Tool com um IDE

- Você pode automatizar a execução dos comandos do Zend_Tool, programando botões ou menus em um IDE, como o Eclipse, para chamá-los.

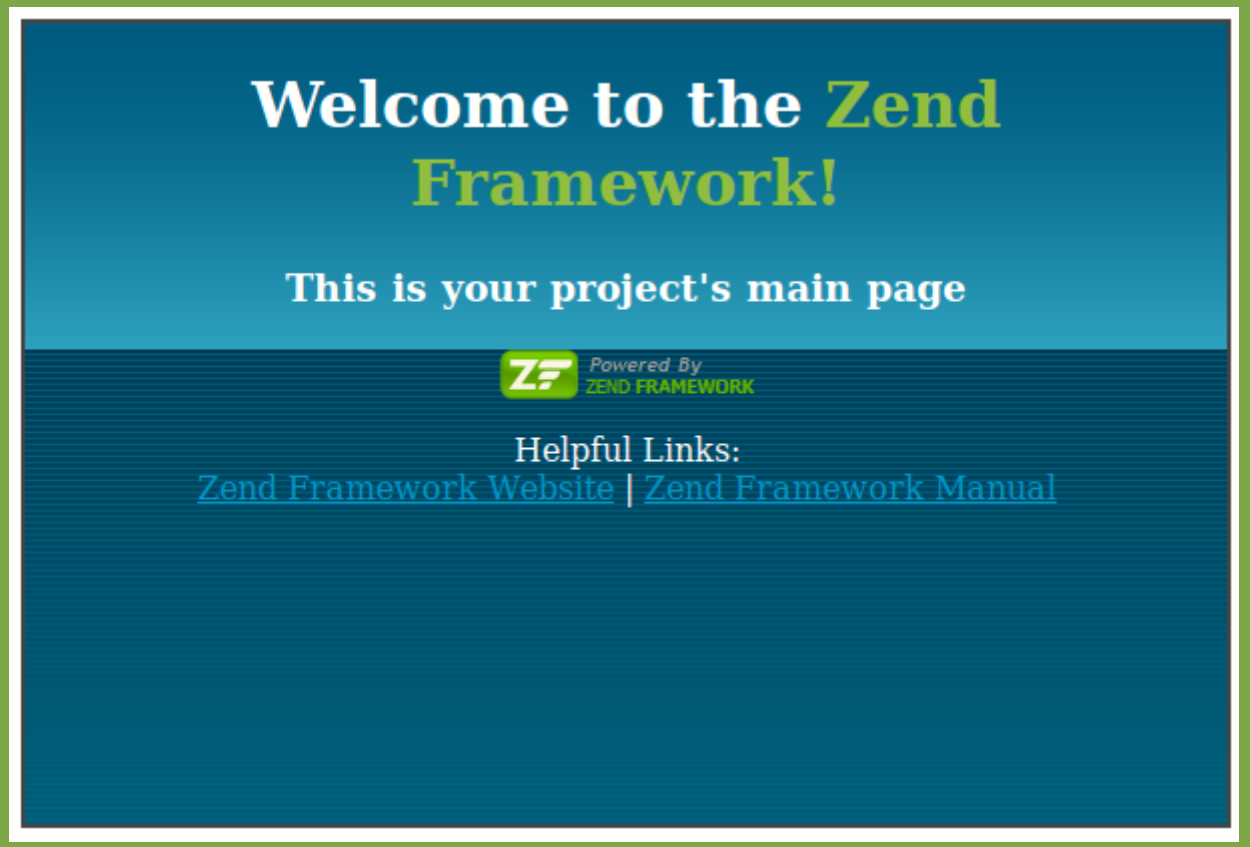


Incluído pelo tradutor

Adicione o projeto ao Apache

- Isso pode variar amplamente para cada sistema operacional e versão, assim você está realmente por conta própria nesta parte.
- Configure o diretório “public” do seu diretório {projectname} como DocumentRoot.
- Depois de reiniciar o Apache, você deve ser capaz de ir ao site que você configurou e ver a página padrão do Zend Framework!

Tcharam!



Incluído pelo tradutor

Alternativa

- Você pode copiar o arquivo .htaccess do diretório “public” para a raiz do projeto.
- Criar uma variável contendo a base URL

```
$baseUrl =
  substr($_SERVER['PHP_SELF'],0,strpos($_SERVER['PHP_SELF'],'/pu
  blic/index.php'));
```

- Definir uma constante com o valor dessa variável

```
define('BASE_URL',$baseUrl);
```

- Configurar o Controlador Frontal no application.ini

```
resources.frontController.baseUrl = BASE_URL
```

Incluído pelo tradutor

Configuração Normal da Aplicação

- Geralmente nós temos o domínio configurado para apontar diretamente para o diretório “public”
  - Isso evita que o código PHP seja acessível diretamente
  - Nenhum arquivo php deve ficar na pasta public a menos que seja simples e o framework peça por isso
    - A exceção é o index.php

Estrutura de Diretórios

- /application
  - arquivos da aplicação
- /library
  - arquivos do framework (/library/Zend)
  - Classes e bibliotecas extras usadas pela aplicação
- /public
  - index.php, html/js/images/etc, qualquer coisa que deva ser diretamente acessível pelo navegador
- /tests
  - testes unitários…

Diretório Application

- Bootstrap.php
  - Configuração do framework específica para a aplicação
- configs
  - Arquivo .ini da aplicação
- controllers
  - Contém as classes controladores de página
- models
  - Contém as classes que modelam os dados
- views
  - Você está convidado a adivinhar o que vai aqui…
  - Errado, não são classes, são simples arquivos php+html.

index.php

- Prepara o include path
- Especifica a configuração da aplicação
- Carrega e roda o bootstrap

Crie uma tabela de banco de dados

- Use qualquer ferramenta com a qual esteja acostumado

```
CREATE TABLE `user` (

`user_id` INT NOT NULL AUTO_INCREMENT ,
  `username` VARCHAR( 255 ) NOT NULL ,
  `email` VARCHAR( 128 ) NOT NULL ,
  `password` VARCHAR( 32 ) NOT NULL ,
  `salt` VARCHAR( 32 ) NOT NULL ,
  `api_key` VARCHAR( 32 ) NOT NULL ,
  `api_secret` VARCHAR( 32 ) NOT NULL ,
  `user_type` TINYINT NOT NULL ,
  PRIMARY KEY ( `user_id` ),

UNIQUE KEY `username` (`username`)

)
```

Adicione o banco de dados à configuração da aplicação

- Edite o arquivo application/configs/application.ini

```
# Database
resources.db.adapter = "pdo_mysql"
resources.db.params.host = "localhost"
resources.db.params.username = "zfclass"
resources.db.params.password = "zfclass"
resources.db.params.dbname =
    "zfclass_{yourname}"
resources.db.isDefaultTableAdapter = true
```

application.ini

- Note que o arquivo application.ini tem múltiplas seções
  - Production, Staging, Development
  - Isso pode ser definido no arquivo .htaccess
  - Essas seções podem ser usadas para especificar diferentes bancos de dados para diferentes ambientes

Configurando o layout

- O que é um layout?
  - Um modo em 2 passos de prover um layout global para sua aplicação
  - A visão resultará em html que será colocado como “conteúdo” no layout
  - Layout pode hospedar links dinâmicos para cabeçalho javascript, css, etc
  - Facilmente acessível e controlável pelo controlador ou visão.

Layout (Continuação)

- Crie os diretórios application/layouts e application/layouts/scripts
- Adicione as seguintes linhas ao seu arquivo application.ini:

```
#layout
resources.layout.layout = "layout"
resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts"
```

Layouts em Módulos

- Estamos fazendo um módulo administrativo
- Vamos carregar um layout específico para ele

application/plugins/ModuleLayout.php

```
class Plugin_ModuleLayout extends Zend_Controller_Plugin_Abstract
{
	public function preDispatch(Zend_Controller_Request_Abstract $request)
	{
		$module = strtolower($request->getParam('module'));
		$layout = Zend_Layout::getMvcInstance();

		if ($layout->getMvcEnabled())
		{
			switch($module) {
				case 'default':
					break;
				default:
					$layout->setLayoutPath(APPLICATION_PATH .
'/modules/' . $module . '/layouts/scripts');
					break;
			}
		}
	}
}
```

Habilite o plugin

- Edite o arquivo application/Bootstrap.php

```
public function _initLayouts() {
    Zend_Layout::startMvc();
    $this->getPluginResource('frontcontroller')
     ->getFrontController()
     ->registerPlugin(new Plugin_ModuleLayout());
}
```

Layout geral da aplicação

- applications/layouts/scripts/layout.phtml

```
<?php echo $this->doctype() ?>
  <html>
  <head>
      <?php echo $this->headTitle() ?>
      <?php echo $this->headLink() ?>
      <?php echo $this->headStyle() ?>
      <?php echo $this->headScript() ?>
  </head>
  <body>
      <?php echo $this->layout()->content ?>
  </body>
  </html>
```

Crie um modelo User

- application/models/DbTable/User.php

```
<?php
class Model_DbTable_User extends Zend_Db_Table
{
    protected $_name = 'user';
    protected $_primary = 'user_id';
```

Modelo User (Continuação)

```
 public function addUser($username, $email, $password, $user_type=1)
 {
     // gera um id único para a senha
     $salt = strtolower(uniqid(rand(), true));
     // e finalmente... algo mais para a chave 'secreta' de sua API
inicial
     $api_secret_key = strtolower(uniqid(rand(), true));
     // cria um hash de senha para gravar no banco
     $hashed_pwd = strtolower(md5($password . $salt));
```

Modelo User...

```
    $data = array(
        'username' => $username,
        'email' => $email,
        'password' => $hashed_pwd,
        'salt' => $salt,
        'user_type' => $user_type
    );
    return $this->insert($data);
}
```

Modelo User...

```
    function updateUser($id, $email, $password=null, $user_type=1)
    {
        $where = array('user_id = ?' => (int)$id);
        $data = array('email' => $email,'user_type'=>$user_type);
        if ($password !== null){
            // generate unique id (again) for the password salt
            $salt = strtolower(uniqid(rand(), true));
            $hashed_pwd = strtolower(md5($password . $salt));
            $data['salt']=$salt;
            $data['password']=$hashed_pwd;
        }
        $this->update($data, $where);
    }
} // Fim da classe (Todo começo tem um fim - Neo)
```

Autoloader

- Permite que chamemos nossas classes modelo sem ter de incluí-las manualmente
  - No arquivo application/Bootstrap.php

```
public function _initAutoload() {
  $autoloader = new Zend_Application_Module_Autoloader(
    array(
     'namespace'=>'',
     'basePath'=>APPLICATION_PATH
    )
  );
  return $autoloader;
}
```

Profiling de Banco de Dados no Firebug!

- Ainda no Bootstrap.php

```
public function _initDbprofile() {
  if($this->getEnvironment() == 'development') {
    $profiler = new Zend_Db_Profiler_Firebug('All DB
  Queries');
    $db = $this->getPluginResource('db');
    $db = $db->getDbAdapter();
    $profiler->setEnabled(true);
    $db->setProfiler($profiler);
  }
}
```

Estrutura da URL

- Padrões para o controlador Index & ação Index
  - /public/{controller}/{action}
  - /public/{module}/{controller}/{action}
  - /public/{controller} (assumes Index action)
  - /public/{controller}/{action}/{key}/{value}/{key2}/{value2}
  - /public/admin/user/edit/user_id/3 (admin = module)

Roteamento de URL

- Você pode criar sua própria estrutura de url usando Zend_Controller_Router_*
  - Muitas opções
    - Static
    - Regex
    - Hostname
    - Chain
  - http://framework.zend.com/manual/en/zend.controller.router.html

Crie o módulo Admin

- Use zf tool
  - `zf create module admin`
- Habilite o módulo no arquivo application.ini
  - Adicione a área de [production]
  - `resources.frontController.moduleDirectory = APPLICATION_PATH "/modules"`
- Copie o arquivo `application/layout/scripts/layout.phtml` para `application/modules/admin/layouts/scripts/layout.phtml`

Crie o controlador User

- `zf create controller user false admin`
- Havia um bug aqui na versão 1.9, com relação ao namespace (emulado) da classe, mas ele foi corrigido na versão 1.10.

Crie a ação

- `zf create action createuser user 1 admin`
- Abra o arquivo do controlador User em um editor... (sinta-se encorajado a usar o Eclipse for PHP Developers)

Crie um formulário de forma programática!

```
public function createuserAction()
   {
       // cria o formulário
       $form = new Zend_Form();
       $form->setMethod('post');
       $form->addElement('text','username', array(
           'label' =>'User name',
           'required'=>true,
           'filters'=>array('StringTrim')
       ));
       $form->addElement('password','password',array(
           'label'=>'Password',
           'required'=>true,
           'filters'=>array('StringTrim')
       ));
       $form->addElement('text','email',array(
           'label'=>'Email address',
           'required'=>true,
           'filters'=>array('StringTrim'),
           'validators'=>array('EmailAddress')
       ));
```

```
    $form->addElement('captcha','captcha',array(
    'label'=>'Enter the letters below',
    'required'=>true,
    'captcha'=>array(
    'captcha'=>'Figlet',
    'wordLen'=>5,
    'timeout'=>300
)
));
    $form->addElement('submit','Save',array(
    'ignore'=>true,
    'label'=>'Save New User'
));
    // Fim do formulário
```

```
    $request = $this->getRequest();

    if($request->isPost()) {
            if($form->isValid($request->getPost())) {
                    $data = $form->getValues();
                    $user = new Model_DbTable_User();
                    if($user->addUser(
                            $data['username'],
                            $data['email'],
                            $data['password'],2)) // 2 for admin
                    {
                            $this->view->message = 'User created';
                    } else {
                            $this->view->message = 'Something bad happened.';
                    }
      }
    }

   $this->view->form = $form;
} // Fim do método
```

Crie a visão

application/modules/admin/views/scripts/user/createuser.php

```
<?php
if(isset($this->message)) {
   ?><h1><?=$this->message ?></h1><?php
}
?>
Cria seu novo usuário.
<?php
$this->form->setAction($this->url());
echo $this->form;
?>
```

Crie seu usuário!

- Você deve ser capaz de acessar http://{yoursite}/admin/user/createuser
- Experimente!

Crie uma ação de login

- Feche o arquivo do controlador de usuário em seu editor
- `zf create action login user 1 admin`
- Abra-o e procure o método loginAction

```
public function loginAction()
{
   $form = new Zend_Form();
$form->setMethod('post');
   $form->addElement('text','username', array(
      'label' =>'User name',
      'required'=>true,
      'filters'=>array('StringTrim')
   ));
   $form->addElement('password','password',array(
      'label'=>'Password',
      'required'=>true,
      'filters'=>array('StringTrim')
   ));
   $form->addElement('submit','submit',array(
      'label'=>'Login',
      'ignore'=>true
   ));
```

```
    $request = $this->getRequest();
    $data = $request->getPost();
    if($request->isPost() && $form->isValid($data)) {
      $u = new Model_DbTable_User();
      $auth = new Zend_Auth_Adapter_DbTable($u->getAdapter()
     ,'user','username','password',
         "MD5(CONCAT(?,salt)) AND user_type=2"
      );
      $auth->setIdentity($data['username'])->setCredential(
     $data['password']
      );
   $mainauth = Zend_Auth::getInstance();
      $result = $mainauth->authenticate($auth);
      if($result->isValid()) {
          $this->view->message = 'Você está logado agora.';
      } else {
         $m = $result->getMessages();
         $this->view->message = $m[0];
         }
      }
      $this->view->form = $form;
  }// Fim do método
```

Crie uma ação de logout

- Tente por si mesmo criar a parte inicial do método de ação.

```
public function logoutAction() {
  $auth=Zend_Auth::getInstance();
  $auth->clearIdentity();
  $this->_redirect('/admin/user/login');
}
```

Crie um “helper” de login

Edite o arquivo application/modules/admin/views/helpers/LoginLink.php

```
<?php
class Admin_View_Helper_LoginLink extends Zend_View_Helper_Abstract {
  public function loginLink() {
    $auth = Zend_Auth::getInstance();
    $front = Zend_Controller_Front::getInstance();
    if($auth->hasIdentity()) {
      $username = $auth->getIdentity();
      return "Olá, $username  [<a href='".$front->getBaseUrl().
         "/admin/user/logout'>Logout</a>]";
    } else {
      return '[<a href="'.$front->getBaseUrl().
        '/admin/user/login">Login</a>]';
    }
  }
}
```

Adicione o helper de login ao layout

- Edite o arquivo application/modules/admin/layouts/scripts/layout.phtml

Adicione antes o conteúdo do comando echo:

```
<div id="login">
<?php echo $this->loginLink();?>
</div>
```

Isso chama o helper LoginLink que criamos antes.

Tente isto agora: http://{yoursite}/admin/user/login

O bastante por hoje

- Se conseguimos ir tão longe neste encontro, estou maravilhado.
- Se não, continuarei no próximo encontro.

Nota do tradutor: Se você quer um exemplo de uma loja virtual, há um projeto criado com Zend Framework 1.10 em meu site, disponível para download: http://www.fgsl.eti.br/site/files/exemplos/booba10.zip