Symfony

Elton Luís Minetto

Agenda

- O que é o Symfony
- Conceitos básicos
- Requisitos
- Instalação
- Configuração
- Aplicação de exemplo

O que é o Symfony

Symfony é um framework de desenvolvimento para PHP5. Seu principal objetivo é aumentar a velocidade de desenvolvimento e manutenção de aplicações web. Ele reduz as tarefas de codificação repetitivas, usando os princípios de DRY(Dont Repeat Yourself) e KISS (Keep It Simple Stupid)

Vantagens

- Licensa flexível (MIT)
- Fácil de instalar e configurar
- Simples para aprender
- Altamente configurável: desde a estrutura de diretórios até bibliotecas de terceiros, quase tudo pode ser customizado
- Compatível com grande número de "melhores práticas" e "design patterns" do ambiente web
- Compatível com diversos bancos de dados
- Não re-inventa a roda. Usa outros projetos como Creole (banco de dados), Propel (mapeamento objeto-relacional) e Mojavi (MVC)

Conceitos - MVC

Separação da estrutura da aplicação em três partes distintas: Modelo, Visão e Controle:

- Modelo: gerencia os dados da aplicação
- Visão: gerencia a saída gráfica e textual da parte da aplicação visível ao usuário
- Controle: interpreta as entradas de mouse e teclado do usuário, comandando a Visão e o Modelo para se alterarem de forma apropriada.

Conceitos - Diretórios

config/	Configurações gerais do projeto.
data/	Arquivos de dados do projeto, esquema do banco de dados.
sql/	Arquivos com instruções SQL.
doc/	Documentação do projeto.
api/	Documentos gerados pelo phpdoc
lib/	Classes e bibliotecas de terceiros
model/	Modelo de objetos do projeto
log/	Arquivos de log
test/	Unidades de testes utilizadas pelo framework
web/	Diretório raiz do servidor web
css/	Onde se armazena arquivos de css
images/	Onde se armazena as imagens da aplicação
js/	Onde se armazena os javascripts da aplicação
uploads/	Diretório de upload para a aplicação

Requisitos

- Servidor web com as funcionalidades: sessions, mod_rewrite (não obrigatório mas aconselhável)
- PHP 5
- PEAR (não obrigatório)
- Uma base dados. É suportado MySQL, PostgreSQL , Oracle, MSSQL e outras bases suportadas pelo Creole.

Instalação

A instalação pode ser feita usando-se o PEAR ou usando o "sandbox", que é um "esqueleto" de aplicação que pode ser copiado e usado como modelo para iniciar o desenvolvimento. A instalação usando-se o sandbox é a mais fácil e prática, principalmente em servidores onde o PEAR não está disponível.

```
wget http://www.symfony-project.com/get/sf_sandbox.tgz
tar xfvz sf_sandbox.tgz
sudo mv sf_sandbox /var/www/symfony
sudo chown -R www-data.www-data /var/www/symfony/cache
```

Instalação

Para testar a instalação pode-se entrar no navegador:
http://localhost/symfony/web/index.php/

Caso ocorra algum erro pode ser a configuração do php.ini,
magic_quotes_gpc que deve estar em Off

Configuração Apache

É necessário duas configurações: o mod_rewrite e o AllowOverride precisam estar ativos.
Alterar o httpd.conf (apache2.conf no Ubuntu) e descomentar ou adicionar as linhas :

```
LoadModule rewrite_module modules/mod_rewrite.so
AddModule mod_rewrite.c
```

Adicionar a linha abaixo no DocumentRoot e reiniciar o Apache:

```
AllowOverride all
```

talk is cheap...

...show me the code!

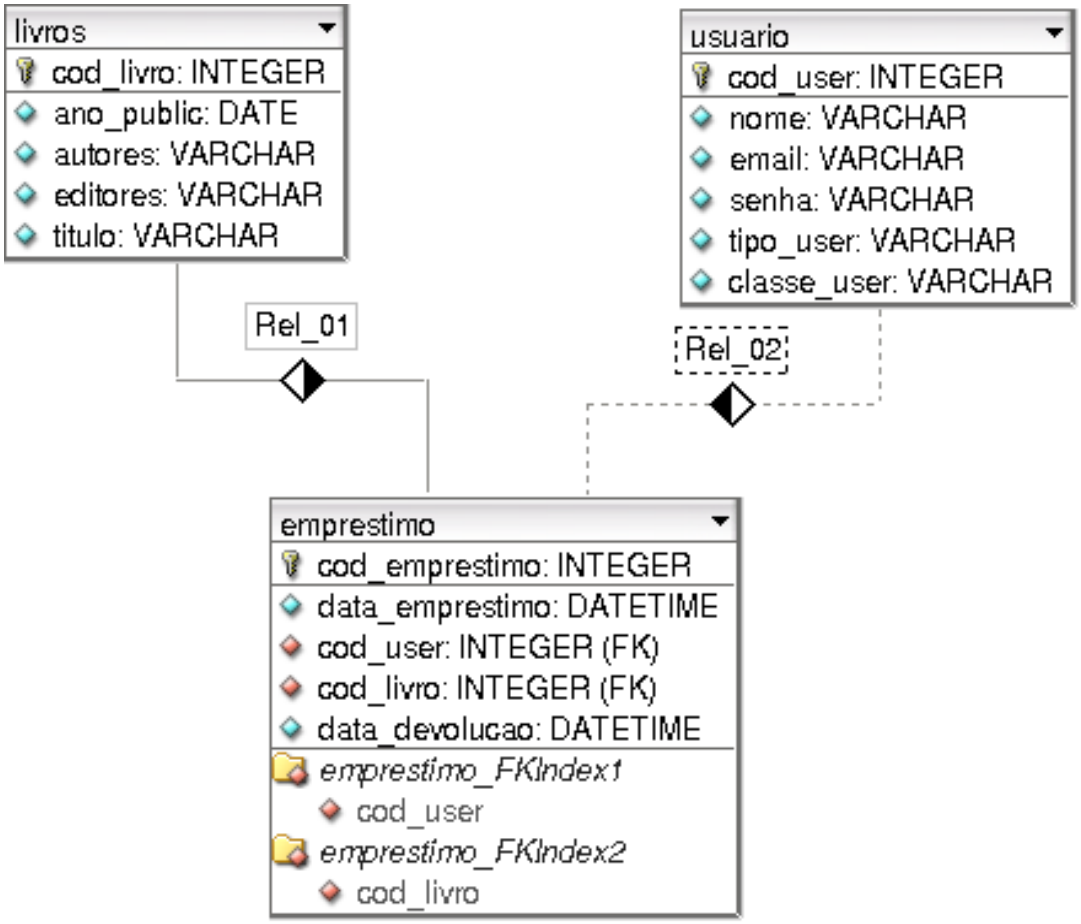
Desenvolvendo

A aplicação

Um professor gostaria de uma aplicação web simples para manter-se informado sobre os livros que empresta a seus alunos. Quanto ele entre no sistema pode visualizar e alterar os alunos cadastrados e também os empréstimos.

Desenvolvendo

A modelagem



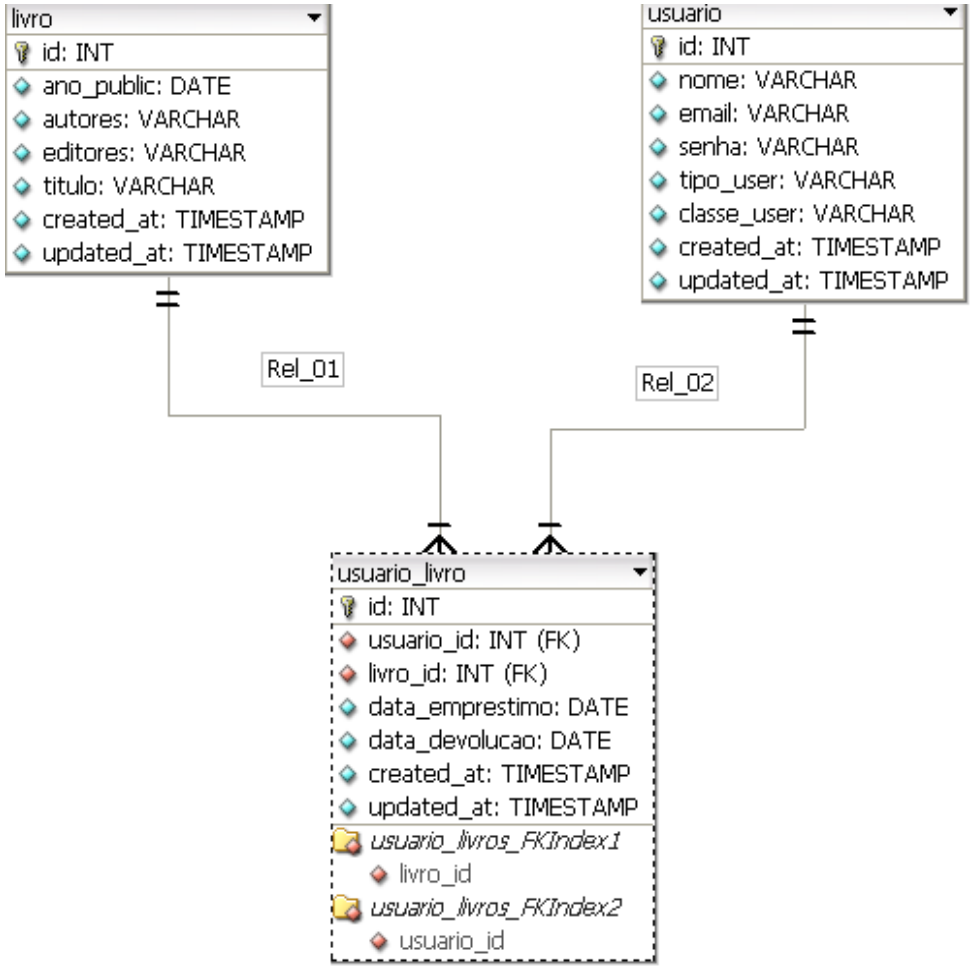
Desenvolvendo

O Symfony segue algumas regras de padronização para a base de dados:

- Colunas com nome id são consideradas chaves primárias
- Colunas com nome finalizando com id são consideradas chaves estrangeiras e relacionadas com outras tabelas de acordo com a primeira parte do nome. Ex: livro_id
- Colunas chamadas created_at e updated_at são automaticamente consideradas do tipo timestamp e são atualizadas automaticamente.

Desenvolvendo

Modelagem adaptada



Desenvolvendo

Criando um projeto

Como estamos usando o sandbox, não é necessário criarmos o projeto, pois o sandbox é um projeto básico. Precisamos somente criar as aplicações:

```
cd /var/www/symfony
sudo ./symfony init-app conference (é necessário o sudo para ele alterar as permissões dos diretórios de cache)
sudo chown -R elm.elm apps/conference/
```

Desenvolvendo

Criando a base de dados.

O Symfony precisa de uma definição da base de dados no arquivo config/schema.xml. Este arquivo pode ser criado manualmente ou apartir de uma base de dados existente. Neste caso criamos as tabelas na base de dados e iremos gerar este arquivo automaticamente. Para gerar o arquivo é preciso configurar a conexão com a base de dados. O arquivo config/propel.ini deve ser configurado:

```
propel.database.url = mysql://usuario:senha@localhost/database
```

Desenvolvendo

Para gerar o arquivo deve-se executar:

```
./symfony.sh propel-build-schema
```

e o arquivo config/schema.xml está criado!

Desenvolvendo

Gerando os modelos

Depois que o schema.xml foi criado podemos gerar os modelos baseados na base de dados. No Symfony o mapeamento objeto-relacional é tratado pelo Propel. Para gerar os modelos pode-se executar:

```
./symfony.sh propel-build-model
```

Serão criados os arquivos dos modelos:

```
lib/model/om/BaseLivrosPeer.php
lib/model/om/BaseLivros.php
lib/model/om/BaseUsuarioLivrosPeer.php
lib/model/om/BaseUsuarioLivros.php
lib/model/om/BaseUsuariosPeer.php
lib/model/om/BaseUsuarios.php
```

Desenvolvendo

Agora podemos gerar as funções básicas de cada tabela para podermos testar.
Antes devemos configurar a base de dados da aplicação, editando o arquivo:
config/databases.yml

O arquivo fica desta maneira:

```
all:
  propel:
    class:      sfPropelDatabase
    param:
      phptype:  mysql
      host:     localhost
      database: database
      username: youruser
      password: yourpasswd
```

Desenvolvendo

CRUD

Gerando o CRUD:

./symfony propel-generate-crud aplicacao modulo Modelo

Neste exemplo:
./symfony propel-generate-crud conference livro Livro
./symfony propel-generate-crud conference usuario Usuario
./symfony propel-generate-crud conference usuario_livro UsuarioLivro

Foram criados diretórios em:
apps/conference/modules/livro
apps/conference/modules/usuario

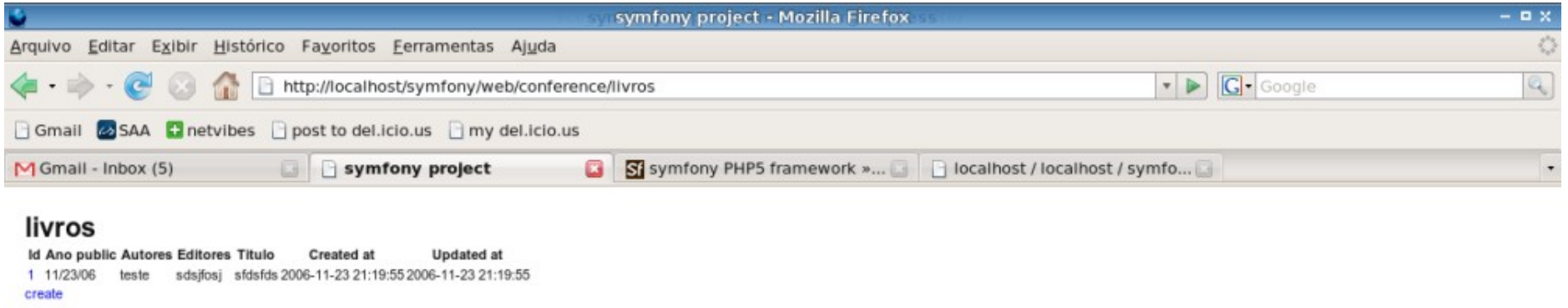
Com os arquivos gerados.

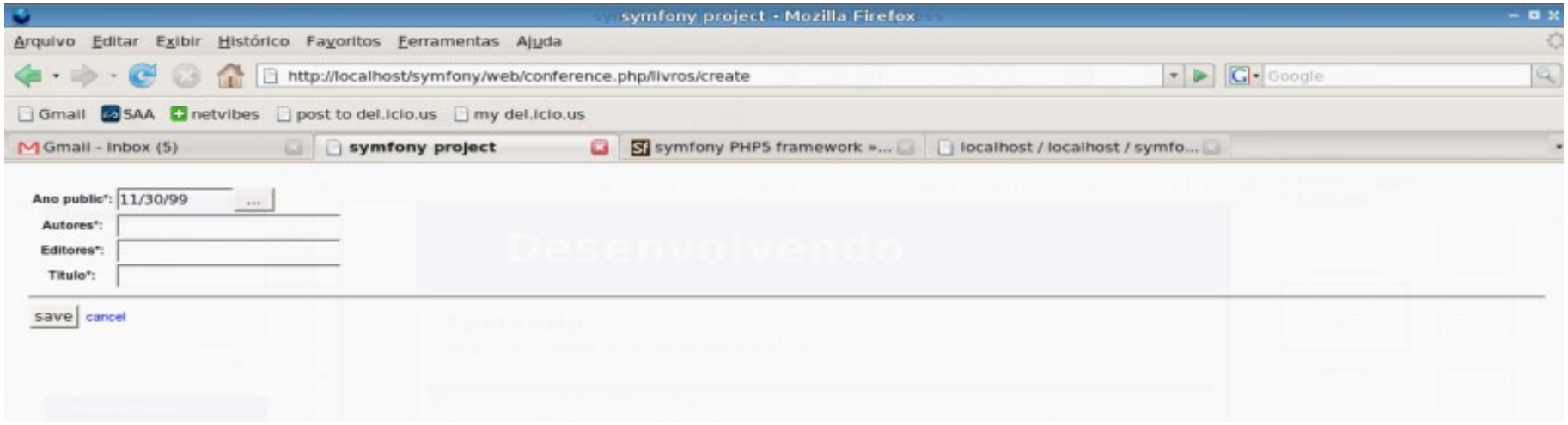
Sempre que um novo módulo é adicionado é preciso limpar o cache de configurações, usando:
./symfony cc conference config

Desenvolvendo

Testando:

http://localhost/symfony/web/conference/livro/index





Customizando

Alterando o Layout geral

O layout geral da aplicação pode ser alterado no arquivo apps/conference/templates/layout.php

Podemos adicionar um cabeçalho e rodapé para as páginas:

```
<body>
  <div id="header">
    <?php echo link_to('Inicial', '@homepage') ?>
  </div>
  <?php echo $sf_data->getRaw('sf_content') ?>
  <div id="content_bar">
        <?php echo link_to(image_tag('http://www.symfony-
project.com/images/symfony_logo.gif', 'alt=Symfony'),
'http://www.symfony-project.com') ?>
    </div>
</body>
```

Customizando

Alterando o Layout de um módulo

Queremos alterar a apresentação do módulo livros. Os arquivos se encontram em

apps/conference/modules/livro/templates/

Customizando

O Symfony automaticamente reconhece os relacionamentos entre as tabelas e cria métodos que facilitam a manipulação dos dados. Por exemplo, podemos descobrir quantos livros um determinado usuário emprestou.

Para isso devemos alterar a visão, já que a parte do modelo e o controlador já contemplam esta informação. Cada módulo criado possui um diretório em

apps/conference/modules

No diretório actions está o controlador, e no diretório templates encontram-se as visões.

Customizando

Iremos alterar a visão listSuccess.php, que mostra a lista de usuários e é ativada pelo método executeList do controlador.
O seguinte código foi adicionado no listSucess.php:

```
<td><?php echo count($usuario->getUsuarioLivros()); ?></td>
```

Desta forma é selecionado quantos livros o usuário tem cadastrado na tabela usuario_livros.

Customizando

Criando uma busca de livros usando Ajax. Precisaremos alterar o controlador e criar as visões para a busca.
No arquivo
apps/conference/modulo/livro/actions/actions.class.php

iremos adicionar dois métodos. O executeBusca() que irá invocar a visão com o formulário e o método executeResultado() que irá fazer a consulta no banco de dados e invocar a visão com os resultados:

Customizando

```
//mostra o formulario
public function executeBusca() {
 }

//faz a consulta
 public function executeResultado() {
     //pega o valor vindo do formulario
     $titulo = $this->getRequestParameter('titulo');
     //cria um novo critério de busca
     $c = new Criteria();
     //usa o campo titulo e o conectivo LIKE do SQL
     $c->add(LivroPeer::TITULO, "$titulo%", Criteria::LIKE);
     //executa a consulta
      $this->livros = LivroPeer::doSelect($c);
     return sfView::SUCCESS;
 }
```

Customizando

O próximo passo é criar os arquivos das visões:

apps/conference/modules/livro/templates/buscaSuccess.php

```
<?//indica que irá usar o helper de javascript para o Ajax
echo use_helper('Javascript') ;
?>
<?//cria um formulário cuja ação será executada usando Ajax
echo form_remote_tag(array(
    'update'   => 'resultado',
    'url'      => 'livro/resultado',
)); ?>
  <label for="titulo">Título:</label>
  <?php echo input_tag('titulo') ?>
  <?php echo submit_tag('Buscar') ?>
</form>

<div id='resultado'></div>
```

Customizando

apps/conference/modules/livro/templates/resultadoSuccess.php

```
<h2>Livros Encontrados</h2>
<?
foreach($livros as $livro)  {
      echo $livro->getTitulo().'<br>';
}
?>
```

Conclusões

- Desenvolver usando frameworks facilita a padronização e trabalho em equipe.
- Acelera o ciclo de desenvolvimento.
- Symfony é altamente flexível.
- Seu aprendizado é um pouco complexo no início mas sua praticidade compensa.

Referências

- http://www.symfony-project.com
- http://www.yaml.org/
- http://propel.phpdb.org/trac/

Contato

Elton Luís Minetto
eminetto@gmail.com
http://www.eltonminetto.net