

- SYMFONY

Jhony M. Maseto

Agenda



- O que são Frameworks bla bla bla
- O que é o Symfony
- Conceitos fundamentais
- Requisitos
- Instalação
- Configurações
- Exemplos

- O que é um Framework
 - Reutilização de códigos;
- Quais são seus princípios básicos
 - Reusabilidade;
- O que um Framework deve conter
 - Programas de apoio
 - bibliotecas de código
 - Linguagem de script

Vantagens



- Redução de custos
- Redução do Time-to-Market
 - O desenvolvedor pode se concentrar em desenvolver novos recursos
- Menos Manutenção
 - Uso de Herança
 - Estabilização de código
 - Melhor consistência entre as aplicações

- O que é o Symfony
 - POO, MVC;
- Primeira versão em 2005
 - Fabien Potencier;
 - Chegada do PHP5;
- Em que foi baseado
 - MVC - Mojavi
 - ORM - Propel
 - Helpers – Ruby on Rails

Por que utilizar



- Licença flexível
- Flexibilidade
- Não reinventa a roda
- Gerador de códigos
- Helpers
- Validação de campos
- Interfaces amigáveis
- Suporte a testes

Por que utilizar



- Log's e debugs
- Plug-ins
- Boa documentação
 - Tutoriais;
 - Wikis;
 - Livros;
 - API's;
 - Cases;
- Bibliotecas pré construídas que automatizam as funções mais comuns

Por que utilizar



- Inclui um suite inteiro de efeitos Java Script
- Gera CRUD e Scaffolding
- Bases de dados suportadas
 - ORACLE
 - MYSQL
 - MSSQL
 - POSTGRESSQL
 - SQLITE

- Requisitos:
 - Fácil de instalar e configurar.
 - Simples de utilizar, e ainda flexível o bastante para casos complexos.
 - Código legível, padrão PHPDocumentor.
 - Fácil de estender, integração fácil com bibliotecas de terceiros.
 - Estável para projetos longos.

- Funções para WEB automatizadas
 - Internacionalização.
 - Templates e layouts que podem ser construídos sem ter conhecimento do framework.
 - Formulários com suporte a validação.
 - Gerenciamento de cache.
 - Facilidade na criação de áreas restritas.
 - Roteamento URL inteligentes.

- Ferramentas que automatizam tarefas comuns a engenharia:
 - Gerador de código.
 - Ferramentas para testes.
 - Painel de debug acelerando a depuração.
 - Interface em linha de comando.
 - Alterações de configurações em tempo real.
 - Funcionalidades de log.

Contempla



| PHP4 | PHP5 | MVC | MULTIPLES DB'S | ORM | VALIDATION | AJAX |
|--------------|---------|-----------|----------------|-----------|---------------------|-----------------|
| NÃO | SIM | SIM | SIM | SIM | SIM | SIM |
| AUTH MODULES | MODULES | TABLELESS | DB OBJECTS | TEMPLATES | INTERNACIONALIZAÇÃO | INTEGRAÇÃO PEAR |
| SIM | SIM | SIM | SIM | SIM | SIM | SIM |

Conceitos fundamentais



- PHP5.
- POO.
 - Classes, Heranças, Objetos.
- PEAR
 - Framework de distribuição de componentes.
- ORM
 - Interface de que faz a comunicação entre o banco de dados e a ferramenta orientada objeto.

Conceitos fundamentais



- RAD
 - Linguagem script que possibilitam o desenvolvimento rápido de aplicações.
- YAML
 - Descrição de um XML de forma mais simples.

- MVC - Architecture
 - Model
 - Lógica do negócio.
 - View
 - Interação com o usuário.
 - Controller
 - Responde as ações dos usuários e invoca as mudanças no model ou na View conforme necessário.

Requisitos



- Servidor WEB com as funcionalidades:
 - Sessions
 - Mod_rewrite
- PHP5
- PEAR
- Base de dados

Principais Métodos - Modelos



- `get`
 - Busca dados da base
- `set`
 - Altera dados
- `save`
 - Responsável pelas alterações e inclusões no banco de dados
- `retrieveByPk`
 - Pesquisa no banco via chave primária

- retrieveByPks
 - Retorna um array de objetos da tabela correspondente.
- delete
- doSelect
 - Consultas SQL mais complexas usando um objeto do tipo Criteria, que deve conter os critérios da busca.

- forward
 - repassa a requisição a outro módulo.
- redirect
 - redireciona para a URL especificada.
- forward404
 - Redireciona para a ação que controla o erro.
- forward404Unless
 - Redireciona caso a condição do parâmetro passado não seja atendida.

- `hasParameter`
 - Verifica a existência da variável passada como parâmetro.
- `getParameter`
 - Retorna o valor da variável da requisição.
- `setFlash`
 - guarda uma mensagem a ser mostrada como resultado de alguma operação.

- `getFlash`
 - Utiliza a variável que foi utilizada pelo `setFlash`.
 - Útil para mensagens de erros.

Helper de formulários



- `form_tag`
 - Gera a tag `<form>` com os parâmetros necessários.
- `input_tag`
 - Gera um campo tipo texto.
- `textarea_tag`
 - Gera um campo tipo `textarea`.
- `checkbox_tag`
 - Gera um `checkbox`.

Helper de formulários



- `select_tag`
 - Gera um campo tipo select.
- `input_file_type`
 - Gera um campo do tipo file.
- `input_password_tag`
 - Gera um campo do tipo password.
- `input_hidden_tag`
 - Gera um campo do tipo hidden 'oculto'.

Helper de formulários



- `submit_tag`
 - Gera um botão submit para o formulário.
- `Submit_image_tag`
 - Gera um botão submit com uma imagem no label do botão.

Config's do necessárias



- php.ini
 - magic_quotes_gpc deixar OFF
- Habilitar mod_rewrite.so
 - Reescrita de URL
- Habilitar no DocumentRoot
 - AllowOverride all

Pacotes disponíveis



- Pacote Sandbox
 - É um projeto symfony vazio que já contém as bibliotecas requeridas para seu uso.
- Pacotes PEAR
 - Indicados para projetos maiores.
- Pacotes de códigos SVN
 - Indicado a desenvolvedores avançados de PHP.

Sandbox



```
wget http://www.symfony-project.com/get/sf\_sandbox.tgz
```

```
tar xfvz sf_sandbox.tgz
```

```
sudo mv sf_sandbox /var/www/sf_evento
```

```
sudo chown -R www-data.www-data /var/www/sf_evento/cache
```

Estrutura de diretórios

| | |
|---------|--|
| APP'S | Onde se encontra os diretórios de cada aplicação. |
| BATCH | Arquivos php chamados pela linha de comando ou agendados para rodar em processos de carga. |
| CACHE | È o cache do projeto, cada aplicação terá seu subdiretório contendo HTML pré-processados e arquivos de configuração. |
| CONFIG | Local onde ficam armazenados as configurações gerais do projeto. |
| DATA | Arquivos de dados do projeto, esquema do banco de dados. |
| DOC | Onde ficam armazenada a documentação do projeto. |
| LIB | Onde ficam armazenadas as classes e bibliotecas de terceiros. |
| LOG | Armazena os arquivos de log gerados pelo symfony |
| PLUGINS | Onde ficam os plugins do projeto |
| TEST | Unidades de testes utilizadas pelo framework |
| WEB | Diretório raiz do servidor web. |

Diretórios da aplicação

| | |
|-----------|--|
| CONFIG | Arquivos de configuração específicos da aplicação |
| I18N | Diretório utilizado para os arquivos de internacionalização da aplicação, este pode ser desprezado no caso da utilização de banco de dados |
| LIB | Contém as bibliotecas específicas da aplicação |
| MODULES | Armazena os módulos que contém as funcionalidades da aplicação |
| TEMPLATES | Armazena os templates globais da aplicação, que vão ser utilizados por todos os módulos |

Aplicação Exemplo



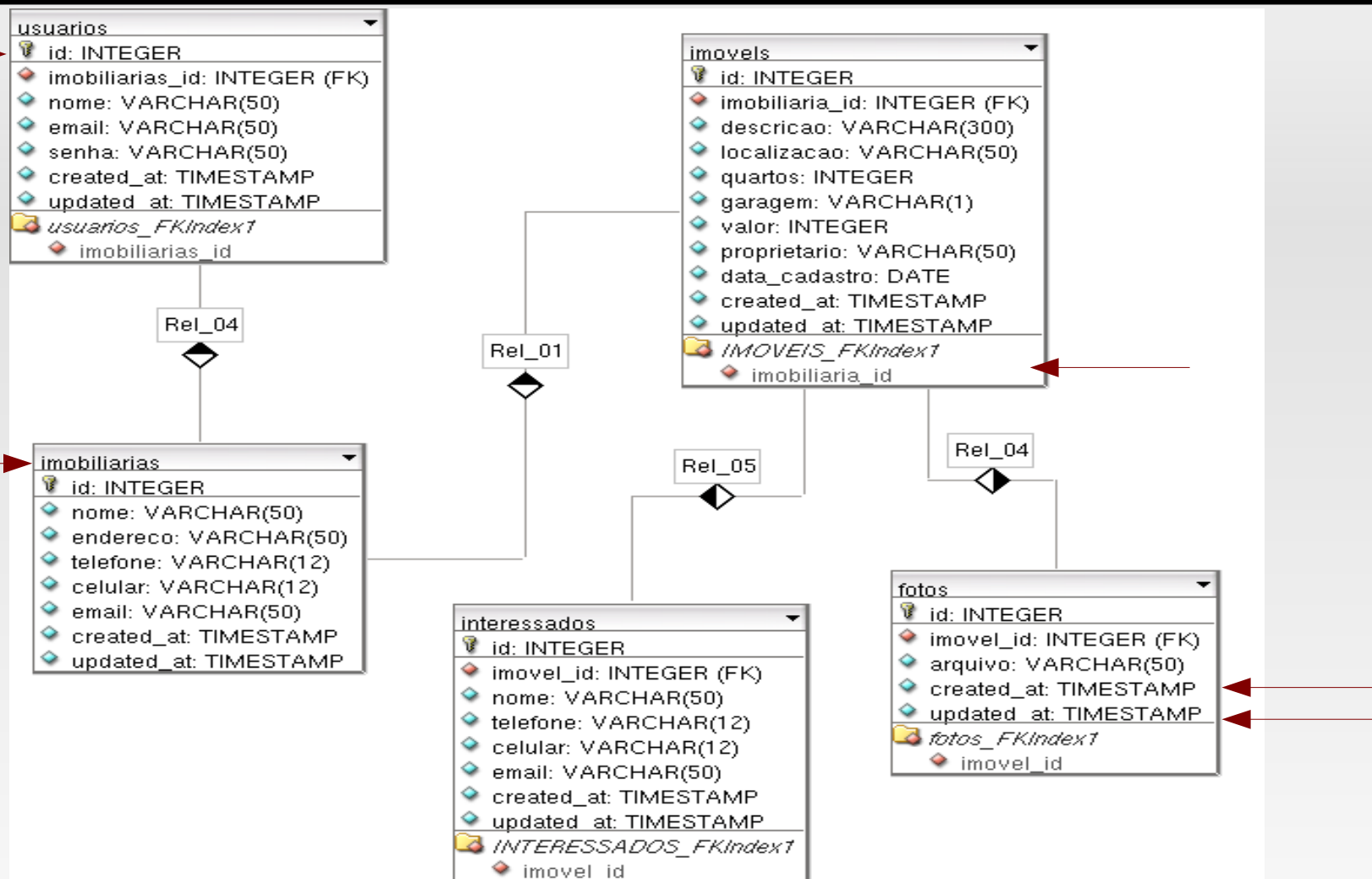
- Site para uma imobiliária
 - Controle de acesso.
 - Upload de arquivo.
 - RSS – Plugins do Symfony.
 - Validação de formulários.
 - Ajax.

Criando a aplicação

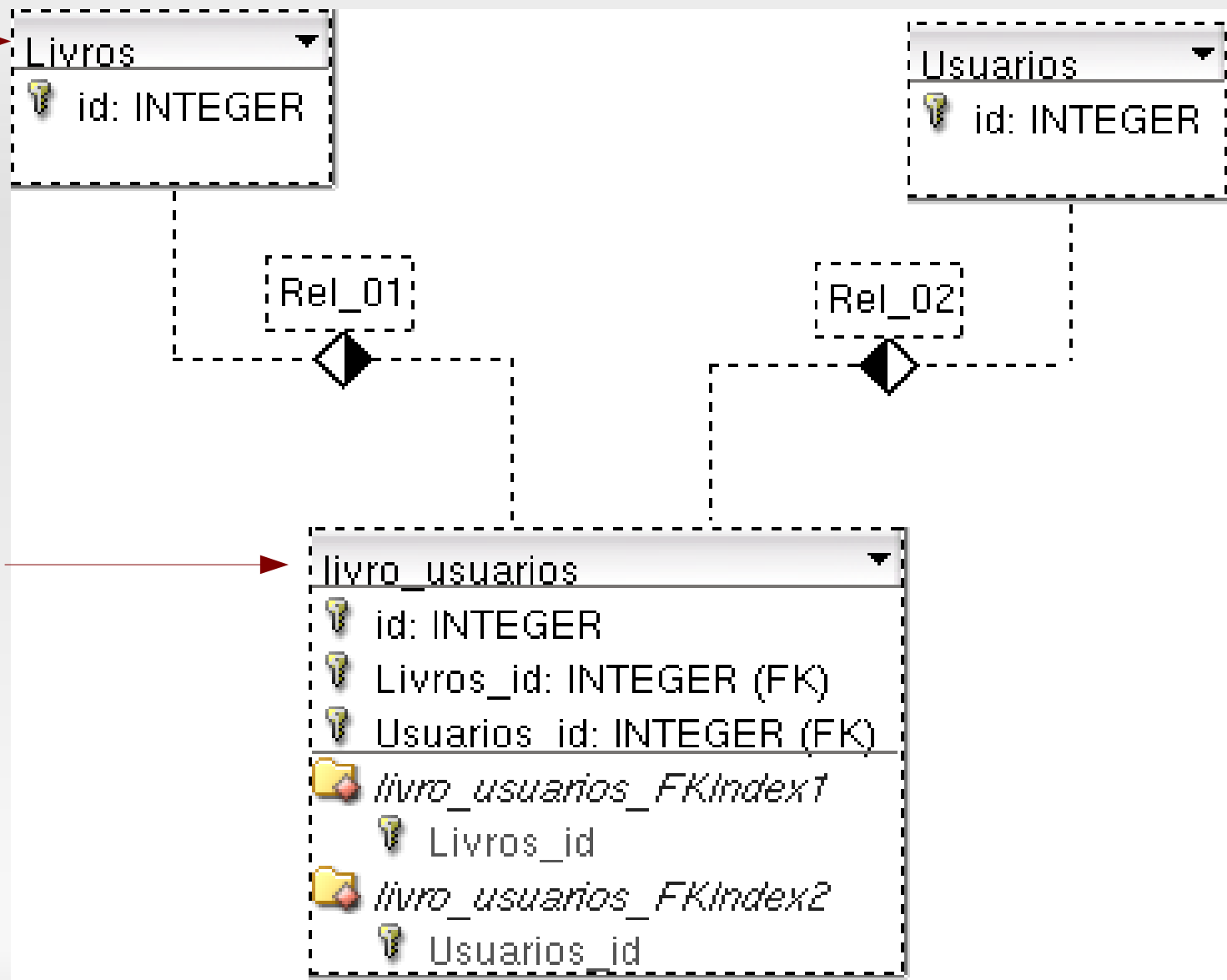


- `cd /var/www/sf_evento`
- `sudo ./symfony init-app imobiliaria`

Base de Dados



Base 2 exemplo N-N



- Configurando a base de dados para gerar o modelo.

```
/var/www/sf_evento/config/propel.ini
```

```
propel.project= sf_evento
```

```
propel.database= mysql
```

```
;propel.database.createUrl=
```

```
propel.database.url=mysql://jhony:teste@localhost/evento
```

Databases.yml



- config da base para gerar o restante.

`/var/www/sf_evento/config/databases.yml`

all: *pode ser produção ou desenvolvimento no nosso caso é all.*

propel:

class: sfPropelDatabase

param: phptype:mysql

host: localhost

database: evento

username: jhony

password: teste

Schema.yml



- `./symfony propel-build-schema.`
 - Será criado um arquivo XML em:
`/var/www/sf_evento/config/schema.yml`

Gerando o modelo



- `./symfony propel-build-model`

`/var/www/sf_imobiliaria/lib/model/Fotos.php`

`/var/www/sf_imobiliaria/lib/model/FotosPeer.php`

`/var/www/sf_imobiliaria/lib/model/map/FotosMapBuilder.php`

`/var/www/sf_imobiliaria/lib/model/om/BaseFotos.php`

`/var/www/sf_imobiliaria/lib/model/om/BaseFotosPeer.php`

- Com estes arquivos são gerados os controles e a visão.

Gerando o CRUD



- `./symfony propel-generate-crud aplicação modulo Modelo`
- `./symfony propel-generate-crud imobiliaria fotos Fotos`
- `./symfony propel-generate-crud imobiliaria usuarios Usuarios`
- `./symfony propel-generate-crud imobiliaria interessados Interessados`
- `./symfony propel-generate-crud imobiliaria imobiliarias Imobiliarias`
- `./symfony propel-generate-crud imobiliaria imovels Imovels`

- `/var/www/sf_evento/apps/imobiliaria/modules/ fotos`
 - actions
 - `action.class.php`
 - templates
 - `editSuccess.php`
 - `listSuccess.php`
 - `showSuccess.php`

Melhorando a interface



- `/var/www/sf_evento/apps/imobiliaria/templates/layout.php`
- `/var/www/sf_evento/web/css/main.css`

Upload de arquivo



- `/var/www/sf_evento/apps/imobiliaria/modules/fotos/actions/actions.class.php`
- `/var/www/sf_evento/apps/imobiliaria/modules/fotos/templates/editSuccess.php`

- Criando o módulo principal
 - `./symfony init-module imobiliaria principal`
- Página inicial
 - `/apps/imobiliaria/config/routing.yml`

http://localhost/sf_evento/web/imobiliaria.php

- `executeIndex()` //mostra o imóveis
- `executeLogin()` //faz login no sistema
- `executeAdmin()` //chama o form de login
- `executeMenu()` //menu de administração
- `executeShow()` //mostra detalhes dos imóveis
- `executeInteressados()` //cria interessados
- `executeUpdate()` //salva o novo interessado
- `executeLogout()` //remove as credenciais do usuário

- `/var/www/sf_evento/apps/imobiliaria/usuarios/config/security.yml`

all:

is_secure: on

credentials: normal

Validação de dados



- `/var/www/sf_evento/apps/imobiliaria/modules/interecidos/validate/update.yml`

fields:

nome:

required:

msg: O campo Nome deve ser preenchido

sfStringValidator:

min:2

min_error: Este nome é muito curto.

max:100

max_error: Este nome é muito longo.

Validação de dados



- apps/imobiliaria/modules/interessados/templates/updateError.php
- Mostra os erros no cadastro.

Plugins - RSS

- `sudo ./symfony plugin-install`
<http://plugins.symfony-project.com/sfFeedPlugin>
- `/var/www/sf_evento/lib/model/om/Baselmovels.php`

```
public function __toString()
```

```
{
```

```
    return $this->getDescricao();
```

```
}
```

- `apps/imobiliaria/modules/principal/config/view.yml`

```
rssSuccess:
```

```
    has_layout: off
```

- `app/imobiliaria/modules/principal/actions/actions.class.php`
- `app/imobiliaria/modules/principal/templates/resultadoSuccess.php`
- `app/imobiliaria/modules/principal/templates/resultadoError.php`
- `app/imobiliaria/modules/principal/templates/indexSuccess.php`

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



- Minetto, Elton Luís . Frameworks para desenvolvimento em PHP. Novatec, 2007
- <http://www.symfony-project.com>
- <http://www.yaml.org/>
- <http://propel.phpdb.org/trac/>
- <http://www.symfony-brasil.com/>

Jhony M. Maseto

jhony.masetto@gmail.com

<http://www.unochapeco.edu.br/~jhony>

<http://jhony.wordpress.com>