Catalyst

Desenvolver para a Web voltou a ter piada

João Gomes
joao.gomes@log.pt

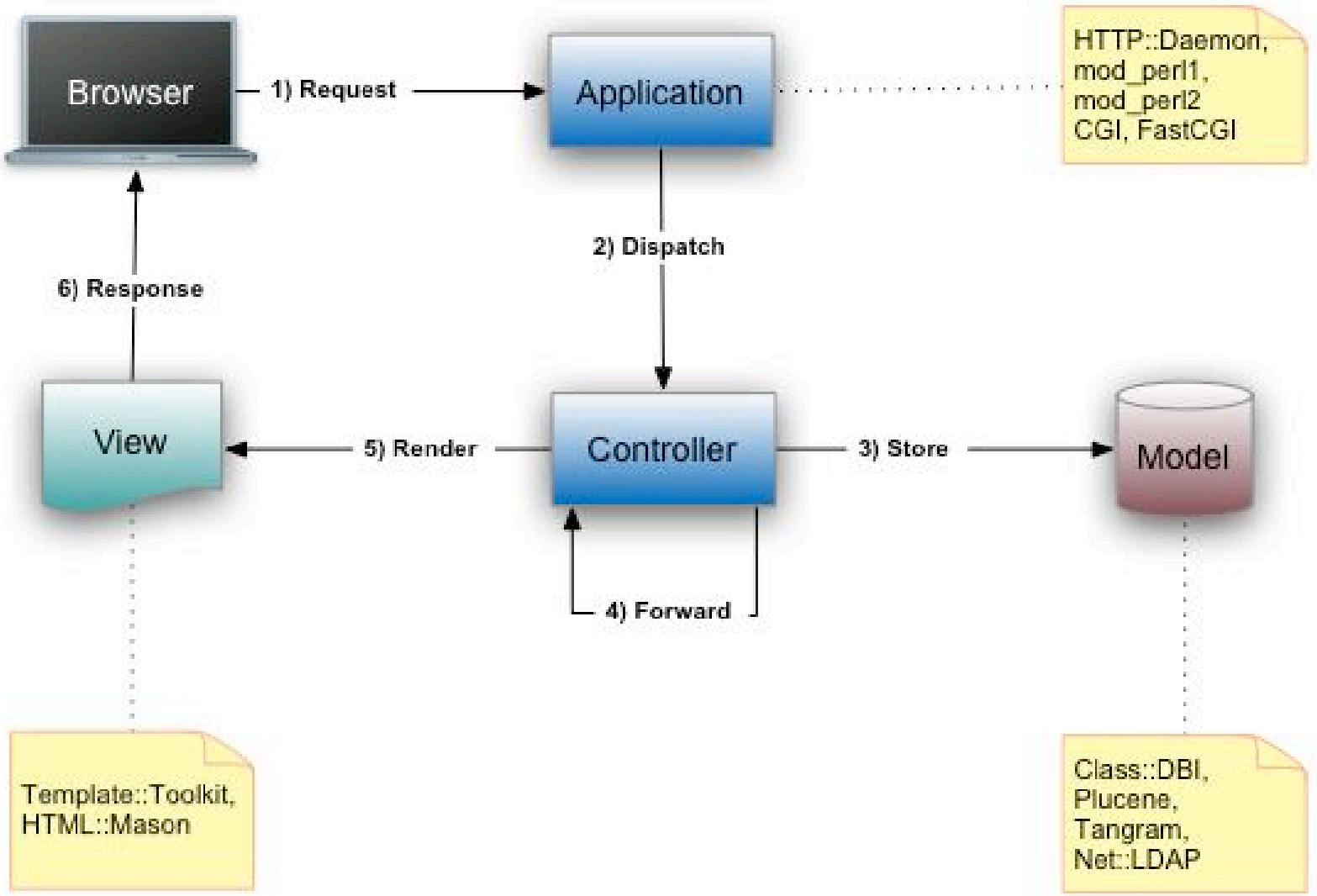
catalyst
Web Framework

O que é o Catalyst?

- Framework MVC de desenvolvimento para web
- Desenvolvimento rápido e flexível
- Inspirado em outras frameworks MVC como o Ruby on Rails, o Spring e o Maypole
- Dezenas de Plugins (sessão, autenticação, tratamento de forms, Ajax, XML-RPC, etc)
- Servidor built-in para desenvolvimento, mod_perl, FastCGI entre outros para produção sem mudar uma linha de código
- Trabalho em equipa
- Desenvolver para a web voltou a ter piada

catalyst
Web Framework

Request Flow



Geração de código

- O Catalyst tem scripts chamados ‘helpers’ para gerar toda a estrutura de uma aplicação
  - `catalyst.pl MyApp`
- A aplicação pode ser testada imediatamente através do servidor de desenvolvimento
  - `script/myapp_server.pl`
  - `http://localhost:3000`

```
created "MyApp"
created "MyApp/script"
created "MyApp/lib"
created "MyApp/root"
created "MyApp/root/static"
created "MyApp/root/static/images"
created "MyApp/t"
created "MyApp/lib/MyApp"
created "MyApp/lib/MyApp/Model"
created "MyApp/lib/MyApp/View"
created "MyApp/lib/MyApp/Controller"
created "MyApp/myapp.yml"
created "MyApp/lib/MyApp.pm"
created "MyApp/README"
created "MyApp/Changes"
created "MyApp/t/01app.t"
created "MyApp/t/02pod.t"
created "MyApp/t/03podcoverage.t"
...
created "MyApp/root/favicon.ico"
created "MyApp/Makefile.PL"
created "MyApp/script/myapp_cgi.pl"
created "MyApp/script/myapp_fastcgi.pl"
created "MyApp/script/myapp_server.pl"
created "MyApp/script/myapp_test.pl"
created "MyApp/script/myapp_create.pl"
```

catalyst
Web Framework

MyApp.pm

- O módulo MyApp.pm é a classe principal da aplicação
- Esta classe é responsável por:
  - Definição de configurações globais à aplicação
  - Carregamento de plugins
  - Definição do método correspondente à raiz do site (’default’)
  - Definição do método que passa o controle para a View (’end’)

MyApp.pm - Exemplo

```
package MyApp;

use strict;
use warnings;

# carregamento dos plugins
use Catalyst qw/-Debug ConfigLoader Static::Simple/;


# definição de configurações globais
MyApp->config( xpto => 'My App is GOOD' );

MyApp->setup;

# raiz do site
sub default : Private {
    my ( $self, $c ) = @_;

    # Hello World
    $c->response->body( $c->welcome_message );
}

sub end : Private {
    my ( $self, $c ) = @_;
    # passa o controle para a view
    $c->forward( $c->view('MyApp::View::TT') ) unless $c->response->body;
}

1;
```

Controllers = URL

- Cada controller é responsável pelo URL correspondente
  - script/myapp_create.pl controller Users
  - lib/MyApp/Controller/Users.pm -> http://localhost:3000/users/
  - Os controllers têm métodos que definem os URL's dentro da classe
  - Existem 5 tipos de métodos:
    - Global
    - Local
    - Path
    - Regex
    - Private

Métodos dos Controllers

- Global
  - Raiz do site
  - http://localhost:3000/login

```
package MyApp::Controller::Login;
sub login : Global {
    my ( $self, $c ) = @_;
    $c->stash->{template} = 'login.xhtml';
    ...
}
```

- Local
  - Relativa ao controller
  - http://localhost:3000/users/view

```
package MyApp::Controller::Users;
sub view : Local {
    my ( $self, $c ) = @_;
    ...
}
```

Métodos dos Controllers

- Path
  - Qualquer caminho
  - http://localhost:3000/qualquer/caminho/que/eu/quiser

```
package MyApp::Controller::Users;

sub xpto : Path('/qualquer/caminho/que/eu/quiser') {
    my ( $self, $c ) = @_;
    ...
}
```

- Regex
  - Faz o match do url com uma expressão regular
  - http://localhost/item23/subitem42

```
package MyApp::Controller::Users;

sub xpto : Regex('^item(\d+)/subitem(\d+)$') {
   my ( $self, $c ) = @_;
   ...
}
```

Métodos dos Controllers

- Private
  - Não está disponível via web
  - Apenas acedida através do método forward
- Parâmetros
  - Os parâmetros adicionais são passados no @_
  - http://localhost/users/view/42

```
package MyApp::Controller::Users;
sub view : Local {
    my ( $self, $c, $user_id ) = @_;
    # $user_id = 42
}
```

Mais sobre controllers

- Os controllers são hierárquicos

  http://localhost/account/billing/edit ->
  MyApp::C::Account::Billing, sub edit : Local { }

  http://localhost/account ->
  MyApp::C::Account, sub default : Private { }

- Métodos especiais dentro dos controllers
  - default - Quando um request a um URL não faz match com nenhum método
  - auto - É chamado antes de qualquer outra acção, hierarquicamente
  - begin - É chamado antes de processar a acção requerida
  - end - É chamado depois de processar a acção requirida

Objecto de Contexto

- Cada método recebe como primeiro parâmetro um objecto de contexto $c
- Este objecto dá acesso à API do Catalyst
  - $c->req - Catalyst::Request
  - $c->req - Catalyst::Response
  - $c->config - configurações globais da aplicação
  - $c->log - logging/debug
  - $c->stash
    - Hash que permite partilhar informação entre os controllers e as views
    - No controller teriamos $c->stash->{titulo} = ‘O meu título’
    - Na template teriamos [% titulo %]

catalyst
Web Framework

Model

- Os Model's são responsáveis por toda a interacção com os dados da aplicação
- Estão disponíveis diversos Model's (Class::DBI, Class::DBI::Sweet, DBIx::Class, Tangram, LDAP, Plucene, etc)
- Podem ser usados em scripts fora do Catalyst

Model - Exemplo

- Bases de dados simples podem ser carregadas através do Catalyst::Model::DBIC
- Cada tabela gera automáticamente uma classe MyApp::Model::Tabela

```
package MyApp::Model::DB;

use strict;
use base 'Catalyst::Model::DBIC';

__PACKAGE__->config(
    dsn           => 'dbi:mysql:database=mydb',
    user          => 'root',
    password      => '',
    options       => {},
    relationships => 1
);
```

View

- Uma view é normalmente chamada no método ‘end’ e vai utilizar como input os dados introduzidos na stash pelos controllers
- Exemplo de uma view:

```
package JobScheduler::View::TToolkit;

use strict;
use base 'Catalyst::View::TT';

__PACKAGE__->config({
  PRE_PROCESS  => 'includes/header.tt',
  POST_PROCESS  => 'includes/footer.tt',
});
```

- As templates têm acesso directo à stash bem como ao objecto de contexto
- A key ‘template’ da stash é utilizada para definir qual a template a mostrar

```
[% IF c.req.params.msg %]
<p class="[% c.req.params.msg_type %]">[% c.req.params.msg %]</p>
[% END %]

[% titulo %]
```

catalyst
Web Framework

Views disponíveis

- Template Toolkit (a mais usada)
- HTML::Template
- Mason
- MicroMason
- Petal
- PHP
- XSLT
- JSON
- PSP

catalyst
Web Framework

Plugins

- O Catalyst utiliza o Next.pm para a sua implementação de plugins
- Extensão do Catalyst em qualquer fase de um request
- Extensão do objecto de contexto
  - Por exemplo um plugin de sessão acrescenta o método $c->sessionid
- Estão disponíveis dezenas de plugins para as mais variadas tarefas
  - Pesquisem por Catalyst::Plugin no CPAN

catalyst
Web Framework

Mais informações

- Trac (wiki, svn) - http://dev.catalyst.perl.org/
- Mailing List - http://lists.rawmode.org/mailman/listinfo/catalyst
- IRC: Join #catalyst on irc.perl.org.
- Não sei se deva dizer isto mas... joao.gomes@log.pt :-)

catalyst
Web Framework

Exemplo Completo

- Job Scheduler