Desenvolvendo Webapps Com Perl

Eden Cardim

31 October 2010

Outline

O que é Perl

- tipagem fraca
- multi-paradigma
  - procedural
  - orientada a objetos
  - funcional
- compilador/interpretador
  - 30 plataformas
  - API em C
- baseada em linguagens humanas
  - níveis de linguagem
  - idiomas/gírias

sintaxe

```
1 print "hello world\n";
```

```
hello world
```

```
1 perl source.pl
```

```
1 perl -le'print "hello world\n"'
```

tipos de dados

escalar

dado “singular”

```
1 my $text = "hello world\n";
2 print $text;
3 print scalar reverse $text;
```

```
hello world

dlrow olleh
```

tipos de dados

lista

dado “plural” (ordenado)

```
1 my @text = ("hello", "world", "\n");
2 print $text[0], "\n";
3 print reverse(@text), "\n";
4 print $text[-2], "\n";
```

```
hello

worldhello
world
```

tipos de dados

hash

lista associativa

```
1 my %data = ( text => "hello world\n" );
2 print $data{text};
```

```
hello world
```

estruturas de controle

```
1 if($text) {
2     print $text;
3 } else {
4     print "no value"
5 }
6
7 while(my $text = shift @text)) {
8   print $text;
9 }
```

subrotinas

```
1 sub print_text {
2     my($text) = @_;
3     print $text
4 }
5 print_text("hello world");
```

```
hello world
```

orientação a objetos

```
1  package Point;
2  use Moose;
3  has ['x','y'] => (is => 'rw', isa => 'Int');
4  sub clear {
5      my $self = shift;
6      $self->x(0); $self->y(0);
7  }
8  package main;
9  my $point = Point->new({x => 10, y => -20});
10 print $point->x, "\n";
11 $point->clear; print $point->x, "\n";
```

```
10
0
```

Módulos

CPAN

The Comprensive Perl Archive Network

```
1 cpan install Lingua::Romana::Perligata
```

```
1 use Lingua::Romana::Perligata;
2
3 dictum sic hello world cis egresso scribe.
```

```
hello world
```

Moose

```perl
package Point;
use Moose;
```

- declaração automática
  - strict/warnings
  - construtor

```perl
my $point = Point->new;
```

- sintaxe de Orientação a Objetos

Atributos

```
1 has 'x' => (is => 'rw', isa => 'Int');
2 has 'y' => (is => 'rw', isa => 'Int');
```

- Tipos
- Métodos de Acesso
- Inicializadores

```
1 my $point = Point({x => 10, y => -20});
```

Métodos

```
1 sub clear {
2     my $self = shift;
3     $self->x(0);
4     $self->y(0);
5 }
```

Modificadores

```
1  package Point3D;
2  use Moose;
3
4  extends 'Point';
5
6  has 'z' => (is => 'rw', isa => 'Int');
7
8  after 'clear' => sub {
9      my $self = shift;
10     $self->z(0);
11 };
```

Roles

```
1  package Point::Role::Radius;
2  use Moose::Role;
3  has 'radius' => (is => 'rw', isa => 'Int');
4  sub max_x {$_[0]->x + $_[0]->radius}
5  sub min_x {$_[0]->x - $_[0]->radius}
6  sub max_y {$_[0]->y + $_[0]->radius}
7  sub min_y {$_[0]->y - $_[0]->radius}
8  sub intersects {
9      my($self, $other) = @_;
10         ( $self->max_x >= $other->min_x
11       and  $self->max_x <= $other->max_x )
12  or     ( $self->max_y >= $other->min_y
13       and  $self->max_y <= $other->max_y )
14 }
```

Roles

```
1 package Circle;
2 use Moose;
3 extends 'Point';
4 with 'Point::Role::Radius';
```

Exercício

Como transformar a class Point3D num Role?

Lógica

Arquitetura

Lógica pertinente à distribuição, execução e manutenção da aplicação.

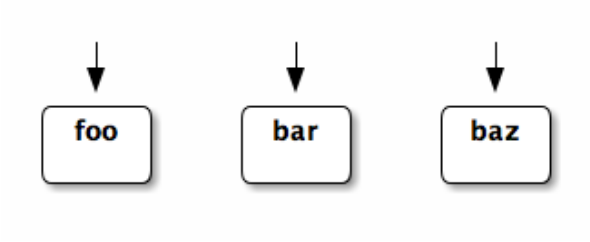
Negócio

Lógica pertinente aos requisitos funcionais da aplicação, que motivam sua existência.

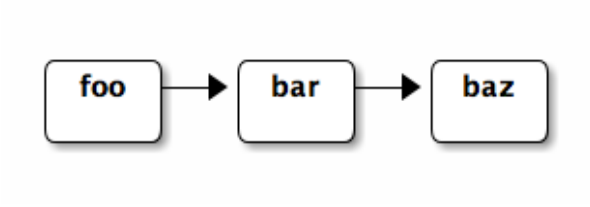
Estado

Uma aplicação web tradicionalmente utiliza o protocolo HTTP, que não tem estado, todos os caminhos lógicos da aplicação são potencialmente executados simultaneamente.

- sem estado



- com estado



Requisição/Resposta

Requisição

Solicitação, por parte de uma usuária, de um recurso da aplicação.

Resposta

Fornecimento do recurso adequado, em resposta à requisição.

Segurança

- Anonimidade
  Usuárias podem "mentir" para a aplicação.
- Interceptação
  Requisições e respostas podem ser interceptadas ao percorrem a rede entre o cliente e o servidor.

Autenticação

Método para verificar se a usuária da aplicação realmente é quem ela é.

Instável

A quantidade de pessoas utilizando uma aplicação web varia aleatoriamente.

Carga

Quantidade de requisições simultâneas à aplicação.

Arquitetura Cliente-Servidor

Client-Side

Parte da aplicação que executa no hardware da usuária.

Server-Side

Parte da aplicação que executa no hardware da aplicação.

Disponibilidade

A aplicação só é útil/rentável enquanto estiver em execução.

Uptime

Período de execução da aplicação, idealmente, maximizado.

Downtime

Período em que a aplicação não está executando.

Catalyst

Framework

Arquitetura “pura”. Aplicação “em branco”, sem lógica de negócio.

Desenvolvimento

Arquitetura de execução adequada à dar suporte ao desenvolvimento da aplicação.

Produção

Arquitetura de execução final da aplicação.

- Engines
  - FastCGI
  - mod_perl
  - `HTTP::Prefork`
  - Plack

Dispatch

Dispatch

Mapeamento requisição/resposta

url	método
/foo	foo()
/bar	bar()

MVC

Model

Lógica

View

Apresentação

Controller

Mapeamento Requisição <-> Resposta <-> Lógica <-> Aplicação

Plugins

Extensões do framework

- Configuração
- Sessões
- Autenticação
- etc. . .

Instalando os Módulos

- cpan
- look local::lib
- perl Makefile.PL –bootstrap
- make test && make install
- echo 'eval $(perl I$HOME/perl5/lib/perl5 Mlocal::lib)' »~/.bashrc
- source ~/.bashrc
- cpan Task::Catalyst

Criando a Aplicação

```
1 catalyst.pl MyApp
```

```
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.conf
```

Estrutura

- lib
  Módulos/Bibliotecas em perl
- t
  Testes
- root
  Arquivos diversos
- myapp.conf
  Configuração

Execução

```
script/myapp_server.pl
[debug] Debug messages enabled
[debug] Statistics enabled
[debug] Loaded plugins:
.----------------------------------------------------------------------------.
| Catalyst::Plugin::ConfigLoader  0.30                                       |
'----------------------------------------------------------------------------'

[debug] Loaded dispatcher "Catalyst::Dispatcher"
[debug] Loaded engine "Catalyst::Engine::HTTP"
[debug] Found home "/Users/edenc/MyApp"
[debug] Loaded Config "/Users/edenc/MyApp/myapp.conf"
```

Execução

- Componentes

```
[debug] Loaded components:
.-----------------------------------------------------------------+----------.
| Class                                                           | Type     |
+-----------------------------------------------------------------+----------+
| MyApp::Controller::Root                                         | instance |
'-----------------------------------------------------------------+----------'
```

Execução

Action

Método de um Controller invocável pelo dispatcher.

```
[debug] Loaded Private actions:
.----------------------+--------------------------------
| Private              | Class
+----------------------+--------------------------------
| /default             | MyApp::Controller::Root
| /end                 | MyApp::Controller::Root
| /index               | MyApp::Controller::Root
'----------------------+--------------------------------
```

Execução

Path Action

Action invocável em resposta a uma URL.

```
[debug] Loaded Path actions:
.-------------------------------------+--------------------
| Path                                | Private
+-------------------------------------+--------------------
| /                                   | /index
| /                                   | /default
'-------------------------------------+--------------------
```

Execução

- Verificando a Página
  http://localhost:3000/

```
[info] *** Request 1 (0.167/s) [37009] [Mon Oct 25 22:57
[debug] "GET" request for "/" from "127.0.0.1"
[debug] Path is "/"
[debug] Response Code: 200; Content-Type: text/html; cha
[info] Request took 0.005863s (170.561/s)
.----------------------------------------------------
| Action
+----------------------------------------------------
| /index
| /end
'----------------------------------------------------
```

Declarando Actions

```
1 sub index :Path :Args(0) {
2     my ( $self, $c ) = @_;
3
4     # Hello World
5     $c->response->body( $c->welcome_message );
6 }
```

Tipos De Dispatch

Path

Action baseada numa url relativa simples

```
1 sub path :Path('welcome') :Args(0) {
2     my ( $self, $c ) = @_;
3
4     # Hello World
5     $c->response->body( $c->welcome_message );
6 }
```

Tipos De Dispatch

Regex

Casa urls através de uma expressão regular

```
1 sub regex :Regex(^item(\d+)/order(\d+)$) {
2     my ( $self, $c ) = @_;
3
4     my($item, $order) = @{$c->req->captures};
5     $c->response->body( "item: $item, order: $order" );
6 }
```

Tipos De Dispatch

Chained

Cadeias de métodos

```
1  sub item :Chained('/item') CaptureArgs(1) {
2    my($self, $c, $item) = @_;
3    $c->stash->{item} = $item;
4  }
5  sub orders :Chained('item') Args(0) {
6    my($self, $c) = @_;
7    $c->stash->{orders} = 'orders';
8  }
9  sub order :Chained('item') Args(1) {
10   my($self, $c, $order) = @_;
11   $c->stash->{order} = $order;
```

Tipos De Dispatch

- Chained

```
1 use Data::Dump;
2 sub end :Action {
3   my($self, $c) = @_;
4   $c->res->body(Data::Dump::dump($c->stash));
5 }
```

Model

```
1  CREATE TABLE user (
2    id INTEGER PRIMARY KEY,
3    name TEXT,
4    password TEXT,
5  );
6  CREATE TABLE tweet (
7    id INTEGER PRIMARY KEY,
8    user_id REFERENCES user (id),
9    data TEXT
10 );
11 CREATE TABLE follow (
12   id INTEGER PRIMARY KEY,
13   user_a_id INTEGER REFERENCES user (id),
14   user_b_id INTEGER REFERENCES user (id)
```

Model

```
1 INSERT INTO user (name, password) values ('foo', 'foosecret
2 INSERT INTO user (name, password) values ('bar', 'barsecret
3 INSERT INTO follow (user_a_id, user_b_id) values (1, 2);
4 INSERT INTO tweet (user_id, data) values (1, "Hello World");
```

Model

- DBIC::Schema

```
1 script/myapp_create.pl model DB DBIC::Schema \\
2   MyApp::Schema create=static dbi:SQLite:myapp.db
```

```
1 Dumping manual schema for MyApp::Schema to directory
2   /Users/edenc/workspace/org/MyApp/script/../lib ...
3 Schema dump completed.
4 ...
```

Model

Result

Objetos que representam um registro.

ResultSet

Objetos que representam um conjunto de registros.

Model

- lib/MyApp/Schema/Result/User.pm

```
1 __PACKAGE__->table("user");
2 __PACKAGE__->add_columns(
3   "id", { data_type => "integer",
4           is_auto_increment => 1, is_nullable => 0 },
5   "name", { data_type => "text", is_nullable => 1 },
6 );
7 __PACKAGE__->set_primary_key("id");
```

Controller

REST

REpresentational State Transfer

```
1 script/myapp_create.pl controller REST::User
```

```
1 extends 'Catalyst::Controller::REST';
```

Model

- Consultas

```
1  sub set :Chained('/') PathPart('user') CaptureArgs(0) {
2    my($self, $c) = @_;
3    $c->stash->{users} = $c->model('DB::User');
4  }
5  sub object :Chained('set') PathPart('') CaptureArgs(1) {
6    my($self, $c, $id) = @_;
7    my $users = $c->stash->{users};
8    $c->stash->{user} = $users->find($id);
9  }
10 sub view :Chained('object') PathPart('') Args(0) :Actior
11 sub view_GET {
12   my($self, $c) = @_;
13   my $user = $c->stash->{user};
```

Model

- Retorna Result
  - new
  - create
  - find
  - next
  - all
- Retorna ResultSet
  - search

Model

- Relacionamentos
  - 1-N

```
1 __PACKAGE__->has_many(
2   "followers",
3   "MyApp::Schema::Result::Follow",
4   { "foreign.user_b_id" => "self.id" },
5   { cascade_copy => 0, cascade_delete => 0 },
6 );
```

- N-1

```
1 __PACKAGE__->belongs_to(
2   "user_b",
3   "MyApp::Schema::Result::User",
4   { "foreign.id => "self.user_b_id" },
5   {
```

View

- Template Toolkit

```
1 script/myapp_create.pl view TT TT
```

- templates

```
1 Name: [% user.name %]
```

Sessões

```
1 $c->session
```

- Catalyst::**Plugin**::Session
  - State
    - URI
    - Cookie
  - Store
    - File
    - Cache::FastMmap
    - Cache::Memcached
    - DBI

Autenticação

- Store
  - Minimal
  - Htpasswd
  - DBIx::Class
  - LDAP

Autenticação

- Credential
  - Password
  - HTTP
  - Twitter

Autenticação

- Realms

```
__PACKAGE__->config->{Plugin::Authentication} =
        {
            default_realm => 'members',
            realms => {
                members => {
                    credential => {
                        class => 'Password',
                        password_field => 'password',
                        password_type => 'clear'
                    },
#...
```

Autenticação

- Realms

```
1                store => {
2                    class => 'DBIx::Class',
3                    user_model => 'MyApp::User',
4                    role_relation => 'roles',
5                    role_field => 'rolename',
6                }
7            }
8        }
9    };
```

Autenticação

```
1 $c->authenticate(
2   $c->req->body_parameters
3 );
```

Configuração

- Default por código

```
1 __PACKAGE__->config(foo => 'bar');
```

- Via Arquivo
  - myapp.conf
  - myapp.xml
  - myapp.yaml
  - myapp.pl

Testes

Test-Anything Protocol

Protocolo de relatório de testes.

- Test::Builder

Testes

Unitários

Teste dos componentes individuais da aplicação.

- Test::Simple

```
1 use Test::Simple qw(no_plan);
2 ok($foo, 'bar');
```

- Test::More

```
1 use Test::More qw(no_plan);
2 is($this, $that);
3 cmp_ok(1, '==', 1);
```

Testes

Integração

Teste da interação entre componentes.

- Test::**WWW**::Mechanize::Catalyst

```
1  use Test::More qw(no_plan)
2  use Test::WWW::Mechanize::Catalyst 'MyApp';
3  my $mech = Test::WWW::Mechanize::Catalyst
4    ->new(catalyst_app => 'MyApp');
5  $mech->get_ok("/");
6  $mech->title_is("Root", "On the root page");
7  $mech->content_contains(
8    "This is the root page", "Correct content"
9  );
10 $mech->follow_link_ok({text => 'Hello'});
```

Implantação

- FastCGI
  - FCGI
  - FCGI::ProcManager

```
1 script/myapp_fastcgi.pl -l /var/myapp/sock
```

```
1 FastCgiExternalServer script/myapp_fastcgi.pl -socket /var/
```

Redeploy

- iniciar o servidor atualizado no mesmo socket
- matar o processo antigo