Построение PSGI-совместимых Web-framework'ов

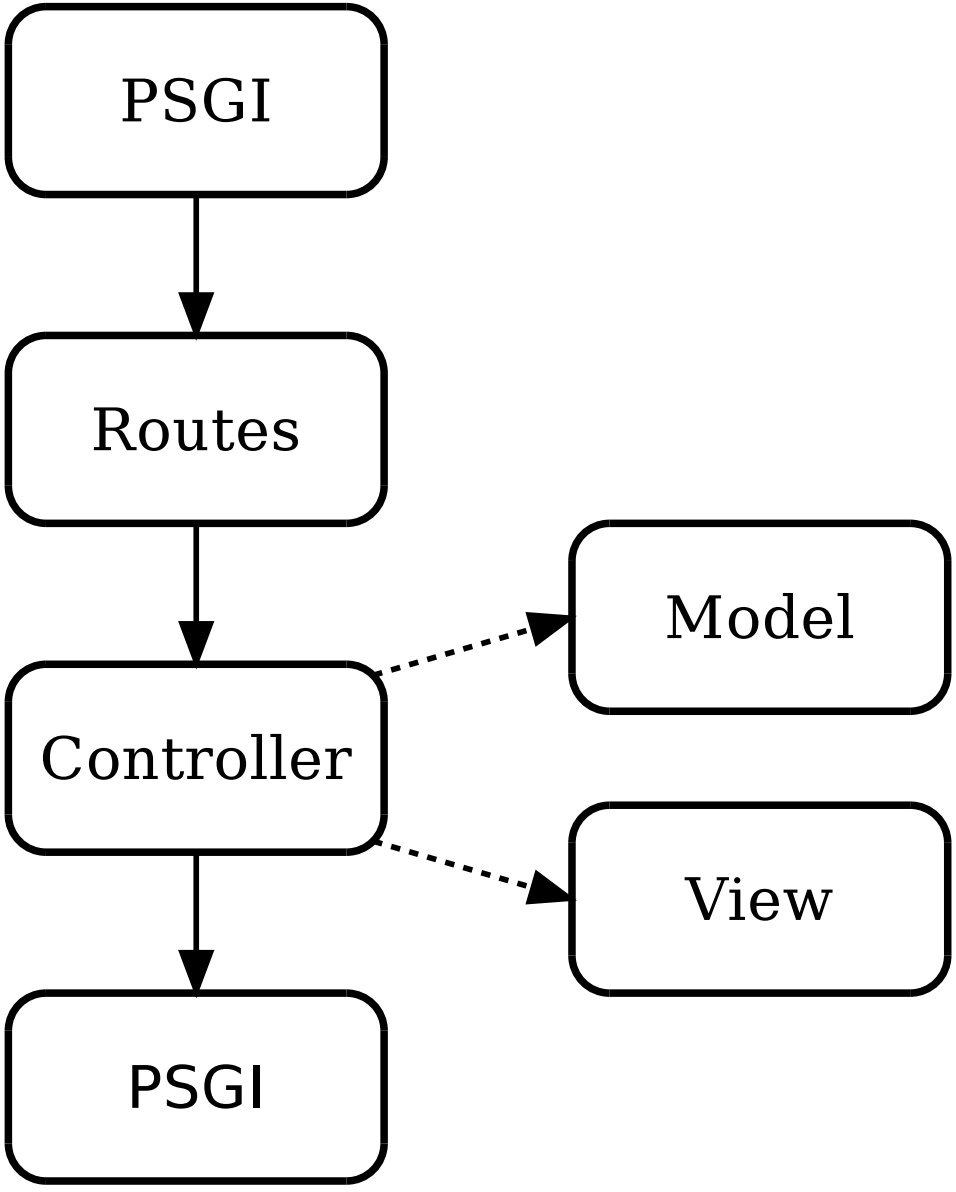
Сергей Засенко (und3f)

1 октября 2011 г.

Преимущества

- Свобода выбора архитектуры приложения.
- Возможность простроения асинхронного приложения на базе любой событийной машины.
- Выбор зависимостей системы.

Составляющие веб-фреймворка



Составляющие веб-фреймворка

Routes — разбор адреса запроса.
Controller — контроль входящих данных и реализация реакции с помощью модели и представления.
View — отображение информации.
Model — данные и методы работы с ними.

Что необходимо выполнить?

Соединить готовые компоненты

Модули разбора маршрутов

- HTTP::Router
- Path::Dispatcher
- Path::Router
- Route::Simple
- Routes::Tiny
- другие.

Шаблонизаторы

- HTML::CTPP2
- HTML::Template
- Template::Toolkit
- Text::Caml
- Text::Xslate
- другие.

Этапы выполнения

1. Разбор адреса запроса и определение обрабатывающего контроллера.
2. Передача управления в соответствующий контроллер.
3. Обработка шаблона с параметрами контроллера.

Реализация

Hello, **Plack**!

Инструменты реализации

- Plack
- Text::Caml
- Routes::Tiny

Инициализация rout'ов

```
my $routes = Routes::Tiny->new;

$routes->add_route('/',
      defaults  => {action => \&root});

$routes->add_route('/welcome/:name' ,
      defaults  => {action => \&welcome});
```

Разбор URL

```
sub dispatch {
    my $env = shift;

    my $path = $env->{PATH_INFO};

    if (my $route = $routes->match($path)) {
        my $action = $route->{params}{action};

        $action->($env, $route->{params});
    }
}
```

View

```
sub render {
    my ($template, $data) = @_;

    my $view = Text::Caml->new;

    my $html = $view->render_file($template, $data);

    [200, ['Content-Type', 'text/html'], [$html]];
}
```

Контроллеры

```
sub root {
    my ($env, $params) = @_;

    render('root.mt');
}

sub welcome {
    my ($env, $params) = @_;

    render('welcome.mt', $params);
}
```

Шаблоны

root.mt

```
<html>
  <body>Hello, Plack!</body>
</html>
```

welcome.mt

```
<html>
  <body>Hello, {{name}}.</body>
</html>
```

Тесты

```
use FindBin '$Bin';
my $app = require "$Bin/../app.psgi";

test_psgi $app, sub {
    my $cb = shift;

    my $res = $cb->(GET '/');
    like  $res->content, qr/Hello, Plack!/;

    $res  = $cb->(GET '/welcome/Sergey');
    like  $res->content, qr/Hello, Sergey./;
}
```

Итог

- Есть набор готовых компонент для построения собственного web-framework'а.
- Соединять компоненты легко.
- Возможно разработать приложение любой конфигурации.
- Разрабатывайте!

Исходный код

https://github.com/und3f/black-perl-2011

Другие примеры

JLogger::Web

`https://github.com/und3f/jlogger-web`

Lamework

`https://github.com/vti/lamework`

Web::Simple

`https://metacpan.org/module/Web::Simple`

Вопросы?