Hellish PHP framework

Руководство программиста

# Создание приложения

Hellish использует apache mod\_rewrite, так что создаем в корневом каталоге приложения файл **.htaccess** со следующим содержимым:

RewriteEngine on

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)$ /index.php?route=$1 [L,QSA]

Далее, пишем следующий **index.php**:

<?php

define('ROOT',dirname(\_\_FILE\_\_).'/');

include '/hellish/hellish.php';

class MySite extends HApplication

{

var $someInternalData = "my own data";

}

// Execute

$my\_app = new MySite();

$my\_app->start(true);

?>

Здесь выполняется подключение фреймворка, создание класса приложения на основе HApplication, и его выполнение (функция start принимает параметр true, означающий, что работаем в debug-режиме, где ведутся логи). На боевой схеме следует запускаться с параметром false.

Функция start описана так: function **start** ($isDebugMode = false, $defaultArea = 'app', $defaultController='index', $defaultAction='index'), соответственно, вы можете изменить эти параметры.

## Навигация

Вся навигация построена по принципу поиска контроллера по следующему принципу (в порядке убывания важности):

* area/controller/action/id
* controller/action/id
* controller/id
* area/controller/action
* controller/action
* controller

# НАПИСАНИЕ КОНТРОЛЛЕРА

Создайте следующую структуру каталогов:

[Ваша AREA]

Controller

[имя контроллера].php

layout

[имя layout].tpl

view

[имя контроллера]

[имя view].tpl

Далее, в файле **[имя контроллера].php** опишите класс контроллера с требуемыми методами:

<?php

class IndexController extends \MySite

{

function ActionName ()

{

$this->m\_Logger->write (HLOG\_WARNING, "Method /Index/Index () executed");

return new View('Index', $s\_text);

}

}

?>

Каждый метод контроллера должен возвратить один из вариантов view (представления), см. следующий раздел.

Метод контроллера наследован от класса вашего приложения, и может обращаться к следующим базовым свойствам:

* me() – возвращает объект приложения
* router() – объект роутера путей
* log() – объект логирования
* vars() – объект доступа к переменным и параметрам

# Представления

Метод контроллера должен возвращать одно из следующих представлений (совпадает с именем класса):

* View – шаблон страницы с layout
* PartialView – шаблон страницы без layout

## Возврат view

В случае полной перезагрузки страницы, фреймфорк возьмет текущий layout, отрендерит его, и вставит в указанное место подготовленную по шаблону страницы. В случае ajax-запроса, шаблон будет вставлен в дом-элемент **id=’page’**.

Шаблон ищется по следующему пути: “/**app\_name**/view/**controller\_name**/**view\_name**.tpl”. При этом, если **view\_name** не указан, он совпадает с **action\_name**.

## Возврат ERRORview

TODO

## Возврат CONSOLEview

TODO

# Система логирования

В debug-режиме фреймворк ведет логирование последнего действия пользователя. Лог доступен по ссылке: /hellish/log/index