一、总览

Yii 应用参照[模型-视图-控制器 （MVC）](http://wikipedia.org/wiki/Model-view-controller) 设计模式来组织。

[模型](http://www.yiichina.com/doc/guide/2.0/structure-models)代表数据、业务逻辑和规则；

[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)展示模型的输出；

[控制器](http://www.yiichina.com/doc/guide/2.0/structure-controllers) 接受出入并将其转换为[模型](http://www.yiichina.com/doc/guide/2.0/structure-models)和[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)命令。

除了 MVC, Yii 应用还有以下部分：

* [入口脚本](http://www.yiichina.com/doc/guide/2.0/structure-entry-scripts)：终端用户能直接访问的 PHP 脚本， 负责启动一个请求处理周期。
* [应用](http://www.yiichina.com/doc/guide/2.0/structure-applications)：能全局范围内访问的对象， 管理协调组件来完成请求.
* [应用组件](http://www.yiichina.com/doc/guide/2.0/structure-application-components)：在应用中注册的对象， 提供不同的功能来完成请求。
* [模块](http://www.yiichina.com/doc/guide/2.0/structure-modules)：包含完整 MVC 结构的独立包， 一个应用可以由多个模块组建。
* [过滤器](http://www.yiichina.com/doc/guide/2.0/structure-filters)：控制器在处理请求之前或之后 需要触发执行的代码。
* [小部件](http://www.yiichina.com/doc/guide/2.0/structure-widgets)：可嵌入到[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)中的对象， 可包含控制器逻辑，可被不同视图重复调用。

二、入口脚本

入口脚本是应用启动流程中的第一环， 一个应用（不管是网页应用还是控制台应用）只有一个入口脚本。 终端用户的请求通过入口脚本实例化应用并将将请求转发到应用。

Web 应用的入口脚本必须放在终端用户能够访问的目录下， 通常命名为 index.php， 也可以使用 Web 服务器能定位到的其他名称。

（控制台应用的入口脚本一般在应用根目录下命名为 yii（后缀为.php）， 该文件需要有执行权限， 这样用户就能通过命令 ./yii <route> [arguments] [options] 来运行控制台应用。）

入口脚本主要完成以下工作：

* 定义全局常量；
* 注册 [Composer 自动加载器](http://getcomposer.org/doc/01-basic-usage.md#autoloading)；
* 包含 [Yii](http://www.yiichina.com/doc/api/2.0/yii) 类文件；
* 加载应用配置；
* 创建一个[应用](http://www.yiichina.com/doc/guide/2.0/structure-applications)实例并配置;
* 调用 [yii\base\Application::run()](http://www.yiichina.com/doc/api/2.0/yii-base-application#run()-detail) 来处理请求。

## Web 应用

以下是[基础应用模版](http://www.yiichina.com/doc/guide/2.0/start-installation)入口脚本的代码：

<?php

defined('YII\_DEBUG') **or** define('YII\_DEBUG', **true**);

defined('YII\_ENV') **or** define('YII\_ENV', 'dev');

// 注册 Composer 自动加载器

**require**(**\_\_DIR\_\_** . '/../vendor/autoload.php');

// 包含 Yii 类文件

**require**(**\_\_DIR\_\_** . '/../vendor/yiisoft/yii2/Yii.php');

// 加载应用配置

$config = **require**(**\_\_DIR\_\_** . '/../config/web.php');

// 创建、配置、运行一个应用，注意这是一个应用

(**new** yii\web\Application($config))->run();

（控制台应用，暂不考虑）

## 定义常量

入口脚本是定义全局常量的最好地方，Yii 支持以下三个常量：

* YII\_DEBUG：标识应用是否运行在调试模式。当在调试模式下，应用会保留更多日志信息， 如果抛出异常，会显示详细的错误调用堆栈。 因此，调试模式主要适合在开发阶段使用，YII\_DEBUG 默认值为 false。
* YII\_ENV：标识应用运行的环境，详情请查阅 [配置](http://www.yiichina.com/doc/guide/2.0/concept-configurations#environment-constants)章节。 YII\_ENV 默认值为 'prod'，表示应用运行在线上产品环境。
* YII\_ENABLE\_ERROR\_HANDLER：标识是否启用 Yii 提供的错误处理， 默认为 true。

当定义一个常量时，通常使用类似如下代码来定义：

defined('YII\_DEBUG') **or** define('YII\_DEBUG', **true**);

常量定义应该在入口脚本的开头，这样包含其他 PHP 文件时， 常量就能生效。

三、应用

1. [应用主体配置](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-configurations)
2. [应用主体属性](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-properties)
3. [应用事件](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-events)
4. [应用主体生命周期](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-lifecycle)

## 应用主体

应用主体是Yii 应用系统整体结构和生命周期的对象。

 每个Yii应用系统只能包含一个应用主体，应用主体在 [入口脚本](http://www.yiichina.com/doc/guide/2.0/structure-entry-scripts) 中创建并能通过表达式 \Yii::$app 全局范围内访问。

Yii有两种应用主体: [网页应用主体](http://www.yiichina.com/doc/api/2.0/yii-web-application) and [控制台应用主体](http://www.yiichina.com/doc/api/2.0/yii-console-application)， 如名称所示，前者主要处理网页请求，后者处理控制台请求。

## 应用主体配置

如下所示，当 [入口脚本](http://www.yiichina.com/doc/guide/2.0/structure-entry-scripts) 创建了一个应用主体， 它会加载一个 [配置](http://www.yiichina.com/doc/guide/2.0/concept-configurations) 文件并传给应用主体。

**require**(**\_\_DIR\_\_** . '/../vendor/autoload.php');

**require**(**\_\_DIR\_\_** . '/../vendor/yiisoft/yii2/Yii.php');

// 加载应用主体配置

$config = **require**(**\_\_DIR\_\_** . '/../config/web.php');

// 实例化应用主体、配置应用主体

(**new** yii\web\Application($config))->run();

类似其他 [配置](http://www.yiichina.com/doc/guide/2.0/concept-configurations) 文件, 应用主体配置文件标明如何设置应用对象初始属性。

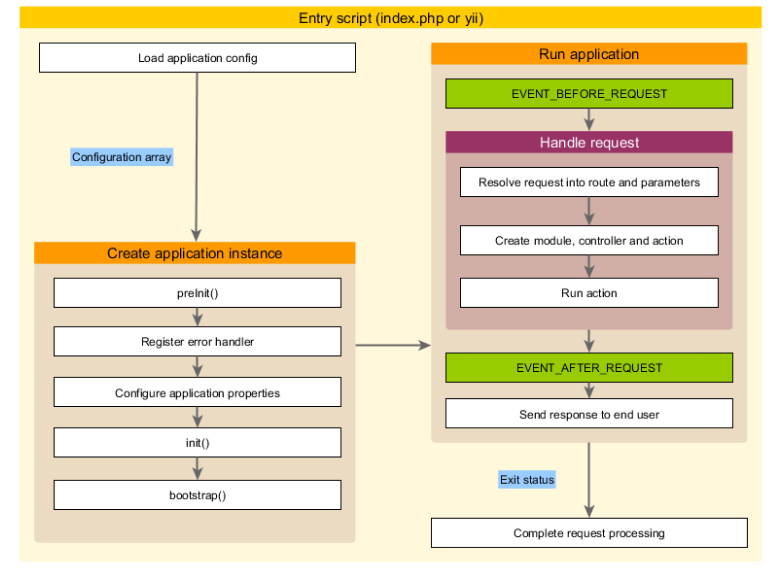
由于应用主体配置比较复杂，一般保存在多个类似如上web.php的[配置文件](http://www.yiichina.com/doc/guide/2.0/concept-configurations#configuration-files) 当中。

应用主体属性：

应用主体配置文件中有许多重要的属性要配置，这些属性指定应用主体的运行环境。 比如，应用主体需要知道如何加载 [控制器](http://www.yiichina.com/doc/guide/2.0/structure-controllers) ， 临时文件保存到哪儿等等。 以下我们简述这些属性。

应用事件：

## 应用主体生命周期



当运行 [入口脚本](http://www.yiichina.com/doc/guide/2.0/structure-entry-scripts) 处理请求时， 应用主体会经历以下生命周期:

1. 入口脚本加载应用主体配置数组。
2. 入口脚本创建一个应用主体实例：
   * 调用 [preInit()](http://www.yiichina.com/doc/api/2.0/yii-base-application#preInit()-detail) 配置几个高级别应用主体属性， 比如 yii\base\Application::basePath。
   * 注册 yii\base\Application::errorHandler 错误处理方法.
   * 配置应用主体属性.
   * 调用 [init()](http://www.yiichina.com/doc/api/2.0/yii-base-application#init()-detail) 初始化，该函数会调用 [bootstrap()](http://www.yiichina.com/doc/api/2.0/yii-base-application#bootstrap()-detail) 运行引导启动组件.
3. 入口脚本调用 [yii\base\Application::run()](http://www.yiichina.com/doc/api/2.0/yii-base-application#run()-detail) 运行应用主体:
   * 触发 [EVENT\_BEFORE\_REQUEST](http://www.yiichina.com/doc/api/2.0/yii-base-application#EVENT_BEFORE_REQUEST-detail) 事件。
   * 处理请求：解析请求 [路由](http://www.yiichina.com/doc/guide/2.0/runtime-routing) 和相关参数； 创建路由指定的模块、控制器和动作对应的类，并运行动作。
   * 触发 [EVENT\_AFTER\_REQUEST](http://www.yiichina.com/doc/api/2.0/yii-base-application#EVENT_AFTER_REQUEST-detail) 事件。
   * 发送响应到终端用户.
4. 入口脚本接收应用主体传来的退出状态并完成请求的处理。

（应用主体是如何使用的，现在还不明白）

四、应用组件

1. [引导启动组件](http://www.yiichina.com/doc/guide/2.0/structure-application-components#bootstrapping-components)
2. [核心应用组件](http://www.yiichina.com/doc/guide/2.0/structure-application-components#core-application-components)

应用主体是[服务定位器](http://www.yiichina.com/doc/guide/2.0/concept-service-locator)， 它部署一组提供各种不同功能的 应用组件 来处理请求。

例如，urlManager组件 负责处理网页请求路由到对应的控制器。 db组件 提供数据库相关服务等等。

在同一个应用中，每个应用组件都有一个独一无二的 ID 用来区分其他应用组件， 你可以通过如下表达式访问应用组件：

\Yii::$app->componentID

如：

可以使用 \Yii::$app->db 来获取到已注册到应用的 [DB connection](http://www.yiichina.com/doc/api/2.0/yii-db-connection)，

使用 \Yii::$app->cache 来获取到已注册到应用的 [primary cache](http://www.yiichina.com/doc/api/2.0/yii-caching-cache)。

应用组件可以是任意对象，可以在 [应用主体配置](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-configurations)配置 yii\base\Application::components 属性 . 例如：

[

'components' => [

// 使用类名注册 "cache" 组件

'cache' => 'yii\caching\ApcCache',

// 使用配置数组注册 "db" 组件

'db' => [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=demo',

'username' => 'root',

'password' => '',

],

// 使用函数注册"search" 组件

'search' => **function** () {

**return** **new** app\components\SolrService;

},

],

]

谨慎注册太多应用组件， 应用组件就像全局变量， 使用太多可能加大测试和维护的难度。 一般情况下可以在需要时再创建本地组件。

## 引导启动组件

一个应用组件只会在第一次访问时实例化， 如果处理请求过程没有访问的话就不实例化。 有时你想在每个请求处理过程都实例化某个组件即便它不会被访问， 可以将该组件ID加入到应用主体的 [bootstrap](http://www.yiichina.com/doc/api/2.0/yii-base-application#bootstrap()-detail) 属性中。

还可以使用闭包来引导启动自定义的组件。不需要直接返回一个实例化的组件。 在应用主体 [yii\base\Application](http://www.yiichina.com/doc/api/2.0/yii-base-application) 实例化后，闭包也会被调用。

例如, 如下的应用主体配置保证了 log 组件一直被加载。

[

'bootstrap' => [

'log',

**function**($app){

**return** **new** ComponentX();

},

**function**($app){

// 可以写自定义的代码

**return**;

}

],

'components' => [

'log' => [

// "log" 组件的配置

],

],

]

## 核心应用组件

Yii 定义了一组固定ID和默认配置的 核心 组件， 例如 yii\web\Application::request 组件 用来收集用户请求并解析 [路由](http://www.yiichina.com/doc/guide/2.0/runtime-routing)；yii\base\Application::db 代表一个可以执行数据库操作的数据库连接。 通过这些组件，Yii应用主体能处理用户请求。

下面是预定义的核心应用组件列表， 可以和普通应用组件一样配置和自定义它们。 当你配置一个核心组件，不指定它的类名的话就会使用Yii默认指定的类。

* 详情请参考 [管理资源](http://www.yiichina.com/doc/guide/2.0/output-assets) 一节。
* 注意配置该组件时必须指定组件类名和其他相关组件属性， 如[yii\db\Connection::$dsn](http://www.yiichina.com/doc/api/2.0/yii-db-connection#$dsn-detail)。 详情请参考 [数据访问对象](http://www.yiichina.com/doc/guide/2.0/db-dao) 一节。
* yii\base\Application::errorHandler: 处理 PHP 错误和异常， 详情请参考 [错误处理](http://www.yiichina.com/doc/guide/2.0/tutorial-handling-errors) 一节。
* 日期使用长格式。 详情请参考 [格式化输出数据](http://www.yiichina.com/doc/guide/2.0/output-formatting) 一节。
* 详情请参考 [国际化](http://www.yiichina.com/doc/guide/2.0/tutorial-i18n) 一节。
* 详情请参考 [日志](http://www.yiichina.com/doc/guide/2.0/tutorial-logging) 一节。
* 详情请参考 [邮件](http://www.yiichina.com/doc/guide/2.0/tutorial-mailing) 一节。
* 详情请参考 [响应](http://www.yiichina.com/doc/guide/2.0/runtime-responses) 一节。
* 详情请参考 [请求](http://www.yiichina.com/doc/guide/2.0/runtime-requests) 一节。
* 仅在[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application) 网页应用中可用， 详情请参考 [Sessions (会话) and Cookies](http://www.yiichina.com/doc/guide/2.0/runtime-sessions-cookies) 一节。
* 详情请参考 [URL 解析和生成](http://www.yiichina.com/doc/guide/2.0/runtime-url-handling) 一节。
* 仅在[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application) 网页应用中可用， 详情请参考 [认证](http://www.yiichina.com/doc/guide/2.0/security-authentication) 一节。
* 详情请参考 [Views](http://www.yiichina.com/doc/guide/2.0/structure-views) 一节。

五、控制器

1. [动作](http://www.yiichina.com/doc/guide/2.0/structure-controllers#actions)
2. [路由](http://www.yiichina.com/doc/guide/2.0/structure-controllers#routes)
3. [创建控制器](http://www.yiichina.com/doc/guide/2.0/structure-controllers#creating-controllers)
4. [创建动作](http://www.yiichina.com/doc/guide/2.0/structure-controllers#creating-actions)
5. [控制器生命周期](http://www.yiichina.com/doc/guide/2.0/structure-controllers#controller-lifecycle)
6. [最佳实践](http://www.yiichina.com/doc/guide/2.0/structure-controllers#best-practices)

控制器是 [MVC](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) 模式中的一部分， 是继承[yii\base\Controller](http://www.yiichina.com/doc/api/2.0/yii-base-controller)类的对象，负责处理请求和生成响应。

具体来说，控制器从[应用主体](http://www.yiichina.com/doc/guide/2.0/structure-applications) 接管控制后会分析请求数据并传送到[模型](http://www.yiichina.com/doc/guide/2.0/structure-models)， 传送模型结果到[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)，最后生成输出响应信息。

## 动作

控制器由 操作 组成，它是执行终端用户请求的最基础的单元， 一个控制器可有一个或多个操作。

如下示例显示包含两个动作view and create 的控制器post：

**namespace** **app**\**controllers**;

**use** **Yii**;

**use** **app**\**models**\**Post**;

**use** **yii**\**web**\**Controller**;

**use** **yii**\**web**\**NotFoundHttpException**;

**class** **PostController** **extends** **Controller**

{

**public** **function** **actionView**($id)

{

$model = Post::findOne($id);

**if** ($model === **null**) {

**throw** **new** NotFoundHttpException;

}

**return** $this->render('view', [

'model' => $model,

]);

}

**public** **function** **actionCreate**()

{

$model = **new** Post;

**if** ($model->load(Yii::$app->request->post()) && $model->save()) {

**return** $this->redirect(['view', 'id' => $model->id]);

} **else** {

**return** $this->render('create', [

'model' => $model,

]);

}

}

}

在操作 view (定义为 actionView() 方法)中， 代码首先根据请求模型ID加载 [模型](http://www.yiichina.com/doc/guide/2.0/structure-models)， 如果加载成功，会渲染名称为view的[视图](http://www.yiichina.com/doc/guide/2.0/structure-views)并显示，否则会抛出一个异常。

在操作 create (定义为 actionCreate() 方法)中, 代码相似. 先将请求数据填入[模型](http://www.yiichina.com/doc/guide/2.0/structure-models)， 然后保存模型，如果两者都成功，会跳转到ID为新创建的模型的view操作， 否则显示提供用户输入的create视图。

## 路由

终端用户通过所谓的 路由 寻找到动作，路由是包含以下部分的字符串：

* 模块ID: 仅存在于控制器属于非应用的[模块](http://www.yiichina.com/doc/guide/2.0/structure-modules);
* 控制器ID: 同应用（或同模块如果为模块下的控制器） 下唯一标识控制器的字符串;
* 操作ID: 同控制器下唯一标识操作的字符串。

路由使用如下格

ControllerID/ActionID

如果属于模块下的控制器，使用如下格式：

ModuleID/ControllerID/ActionID

如果用户的请求地址为 http://hostname/index.php?r=site/index, 会执行site 控制器的index 操作

## 创建控制器

在[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application)网页应用中，控制器应继承[yii\web\Controller](http://www.yiichina.com/doc/api/2.0/yii-web-controller) 或它的子类。

同理在[console applications](http://www.yiichina.com/doc/api/2.0/yii-console-application)控制台应用中，控制器继承[yii\console\Controller](http://www.yiichina.com/doc/api/2.0/yii-console-controller) 或它的子类。 如下代码定义一个 site 控制器:

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**class** **SiteController** **extends** **Controller**

{

}

## 控制器ID

通常情况下，控制器用来处理请求有关的资源类型， 因此控制器ID通常为和资源有关的名词。 例如使用article作为处理文章的控制器ID。

控制器ID应仅包含英文小写字母、数字、下划线、中横杠和正斜杠， 例如 article 和 post-comment 是真是的控制器ID， article?, PostComment, admin\post不是控制器ID。

控制器Id可包含子目录前缀，例如 admin/article 代表 [controller namespace](http://www.yiichina.com/doc/api/2.0/yii-base-application#$controllerNamespace-detail) 控制器命名空间下 admin子目录中 article 控制器。 子目录前缀可为英文大小写字母、数字、下划线、正斜杠，其中正斜杠用来区分多级子目录(如 panels/admin)。

## 控制器类命名

控制器ID遵循以下规则衍生控制器类名：

1. 将用正斜杠区分的每个单词第一个字母转为大写。注意如果控制器ID包含正斜杠， 只将最后的正斜杠后的部分第一个字母转为大写；
2. 去掉中横杠，将正斜杠替换为反斜杠;
3. 增加Controller后缀;
4. 在前面增加[controller namespace](http://www.yiichina.com/doc/api/2.0/yii-base-application#$controllerNamespace-detail)控制器命名空间.

下面为一些示例，假设[controller namespace](http://www.yiichina.com/doc/api/2.0/yii-base-application#$controllerNamespace-detail) 控制器命名空间为 app\controllers:

* article 对应 app\controllers\ArticleController;
* post-comment 对应 app\controllers\PostCommentController;
* admin/post-comment 对应 app\controllers\admin\PostCommentController;
* adminPanels/post-comment 对应 app\controllers\adminPanels\PostCommentController.

控制器类必须能被 [自动加载](http://www.yiichina.com/doc/guide/2.0/concept-autoloading)，所以在上面的例子中， 控制器article 类应在 [别名](http://www.yiichina.com/doc/guide/2.0/concept-aliases) 为@app/controllers/ArticleController.php的文件中定义， 控制器admin/post-comment应在@app/controllers/admin/PostCommentController.php文件中。

## 控制器部署

可通过配置 yii\base\Application::controllerMap 来强制上述的控制器ID和类名对应， 通常用在使用第三方不能掌控类名的控制器上（当不能对应时这样强制配置）。

配置 [应用配置](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-configurations) 中的[application configuration](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-configurations)，如下所示：

[

'controllerMap' => [

// 用类名申明 "account" 控制器

'account' => 'app\controllers\UserController',

// 用配置数组申明 "article" 控制器

'article' => [

'class' => 'app\controllers\PostController',

'enableCsrfValidation' => **false**,

],

],

]

## 默认控制器

每个应用有一个由yii\base\Application::defaultRoute属性指定的默认控制器； 当请求没有指定 [路由](http://www.yiichina.com/doc/guide/2.0/structure-controllers#ids-routes)，该属性值作为路由使用。 对于[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application)网页应用，它的值为 'site'，对于 [console applications](http://www.yiichina.com/doc/api/2.0/yii-console-application) 控制台应用，它的值为 help，所以URL为 http://hostname/index.php 表示由 site 控制器来处理。

可以在 [应用配置](http://www.yiichina.com/doc/guide/2.0/structure-applications#application-configurations) 中修改默认控制器，如下所示：

[

'defaultRoute' => 'main',

]

## 创建动作

创建操作就是在控制器类中定义 操作方法 来完成，操作方法必须是以action开头的公有方法。 操作方法的返回值会作为响应数据发送给终端用户， 如下代码定义了两个操作 index 和 hello-world:

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**class** **SiteController** **extends** **Controller**

{

**public** **function** **actionIndex**()

{

**return** $this->render('index');

}

**public** **function** **actionHelloWorld**()

{

**return** 'Hello World';

}

}

## 操作ID

操作通常是用来执行资源的特定操作，因此， 操作ID通常为动词，如view, update等。

操作ID应仅包含英文小写字母、数字、下划线和中横杠，操作ID中的中横杠用来分隔单词。 例如view, update2, comment-post是真实的操作ID， view?, Update不是操作ID.

可通过两种方式创建操作ID，内联操作和独立操作. An inline action is 内联操作在控制器类中定义为方法；独立操作是继承[yii\base\Action](http://www.yiichina.com/doc/api/2.0/yii-base-action)或它的子类的类。 内联操作容易创建，在无需重用的情况下优先使用； 独立操作相反，主要用于多个控制器重用， 或重构为[扩展](http://www.yiichina.com/doc/guide/2.0/structure-extensions)。

## 内联操作（就是我们自己在控制器中定义的操作）

内联动作指的是根据我们刚描述的操作方法。

动作方法的名字是根据操作ID遵循如下规则衍生：

1. 将每个单词的第一个字母转为大写;
2. 去掉中横杠;
3. 增加action前缀.

例如index 转成 actionIndex, hello-world 转成 actionHelloWorld。

操作方法的名字大小写敏感，如果方法名称为ActionIndex不会认为是操作方法， 所以请求index操作会返回一个异常， 也要注意操作方法必须是公有的， 私有或者受保护的方法不能定义成内联操作。

## 独立动作

独立操作通过继承[yii\base\Action](http://www.yiichina.com/doc/api/2.0/yii-base-action)或它的子类来定义。

例如Yii发布的[yii\web\ViewAction](http://www.yiichina.com/doc/api/2.0/yii-web-viewaction) 和[yii\web\ErrorAction](http://www.yiichina.com/doc/api/2.0/yii-web-erroraction)都是独立操作。

要使用独立操作，需要通过控制器中覆盖[yii\base\Controller::actions()](http://www.yiichina.com/doc/api/2.0/yii-base-controller#actions()-detail)方法在action map中申明， 如下例所示：

**public** **function** **actions**()

{

**return** [

// 用类来申明"error" 动作

'error' => 'yii\web\ErrorAction',

// 用配置数组申明 "view" 动作

'view' => [

'class' => 'yii\web\ViewAction',

'viewPrefix' => '',

],

];

}

如上所示， actions() 方法返回键为操作ID、值为对应操作类名 或数组[configurations](http://www.yiichina.com/doc/guide/2.0/concept-configurations) 的数组。 和内联操作不同，独立操作ID可包含任意字符，只要在actions() 方法中申明.

为创建一个独立操作类，需要继承[yii\base\Action](http://www.yiichina.com/doc/api/2.0/yii-base-action) 或它的子类，并实现公有的名称为run()的方法, run() 方法的角色和操作方法类似，例如：

<?php

**namespace** **app**\**components**;

**use** **yii**\**base**\**Action**;

**class** **HelloWorldAction** **extends** **Action**

{

**public** **function** **run**()

{

**return** "Hello World";

}

}

## 动作结果

操作方法或独立操作的run()方法的返回值非常重要， 它表示对应操作结果。

返回值可为 [响应](http://www.yiichina.com/doc/guide/2.0/runtime-responses) 对象，作为响应发送给终端用户。

* 对于[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application)网页应用，返回值可为任意数据, 它赋值给[yii\web\Response::$data](http://www.yiichina.com/doc/api/2.0/yii-web-response#$data-detail)， 最终转换为字符串来展示响应内容。
* 对于[console applications](http://www.yiichina.com/doc/api/2.0/yii-console-application)控制台应用，返回值可为整数， 表示命令行下执行的 yii\console\Response::exitStatus 退出状态。

在上面的例子中，操作结果都为字符串，作为响应数据发送给终端用户， 下例显示一个操作通过 返回响应对象（因为[redirect()](http://www.yiichina.com/doc/api/2.0/yii-web-controller#redirect()-detail)方法返回一个响应对象） 可将用户浏览器跳转到新的URL。

**public** **function** **actionForward**()

{

// 用户浏览器跳转到 http://example.com

**return** $this->redirect('http://example.com');

}

## 动作参数

内联动作的操作方法和独立动作的 run() 方法可以带参数，称为动作参数。

参数值从请求中获取，对于[Web applications](http://www.yiichina.com/doc/api/2.0/yii-web-application)网页应用， 每个动作参数的值从$\_GET中获得，参数名作为键；

对于[console applications](http://www.yiichina.com/doc/api/2.0/yii-console-application)控制台应用, 动作参数对应命令行参数。

如下例，动作view (内联动作) 申明了两个参数 $id 和 $version。

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**class** **PostController** **extends** **Controller**

{

**public** **function** **actionView**($id, $version = null)

{

// ...

}

}

动作参数会被不同的参数填入，如下所示：

* http://hostname/index.php?r=post/view&id=123: $id 会填入'123'， $version 仍为 null 空因为没有version请求参数;
* http://hostname/index.php?r=post/view&id=123&version=2: $id 和 $version 分别填入 '123' 和 '2'`；
* http://hostname/index.php?r=post/view: 会抛出[yii\web\BadRequestHttpException](http://www.yiichina.com/doc/api/2.0/yii-web-badrequesthttpexception) 异常 因为请求没有提供参数给必须赋值参数$id；
* http://hostname/index.php?r=post/view&id[]=123: 会抛出[yii\web\BadRequestHttpException](http://www.yiichina.com/doc/api/2.0/yii-web-badrequesthttpexception) 异常 因为$id 参数收到数字值 ['123']而不是字符串.

如果想让动作参数接收数组值，需要指定$id为array，如下所示：

**public** **function** **actionView**(array $id, $version = null)

{

// ...

}

现在如果请求为 http://hostname/index.php?r=post/view&id[]=123, 参数 $id 会使用数组值['123']， 如果请求为 http://hostname/index.php?r=post/view&id=123， 参数 $id 会获取相同数组值，因为无类型的'123'会自动转成数组。

## 默认动作

每个控制器都有一个由 [yii\base\Controller::$defaultAction](http://www.yiichina.com/doc/api/2.0/yii-base-controller#$defaultAction-detail) 属性指定的默认操作， 当[路由](http://www.yiichina.com/doc/guide/2.0/structure-controllers#ids-routes) 只包含控制器ID， 会使用所请求的控制器的默认操作。

默认操作默认为 index，如果想修改默认操作，只需简单地在控制器类中覆盖这个属性， 如下所示：

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**class** **SiteController** **extends** **Controller**

{

**public** $defaultAction = 'home';

**public** **function** **actionHome**()

{

**return** $this->render('home');

}

}

## 控制器生命周期

处理一个请求时，[应用主体](http://www.yiichina.com/doc/guide/2.0/structure-applications) 会根据请求 [路由](http://www.yiichina.com/doc/guide/2.0/structure-controllers#routes)创建一个控制器， 控制器经过以下生命周期来完成请求：

1. 在控制器创建和配置后，yii\base\Controller::init() 方法会被调用。
2. 控制器根据请求操作ID创建一个操作对象:
   * 如果操作ID没有指定，会使用[default action ID](http://www.yiichina.com/doc/api/2.0/yii-base-controller#$defaultAction-detail)默认操作ID；
   * 如果在[action map](http://www.yiichina.com/doc/api/2.0/yii-base-controller#actions()-detail)找到操作ID， 会创建一个独立操作；
   * 如果操作ID对应操作方法，会创建一个内联操作；
   * 否则会抛出[yii\base\InvalidRouteException](http://www.yiichina.com/doc/api/2.0/yii-base-invalidrouteexception)异常。
3. 控制器按顺序调用应用主体、模块（如果控制器属于模块）、 控制器的 beforeAction() 方法；
   * 如果任意一个调用返回false，后面未调用的beforeAction()会跳过并且操作执行会被取消； action execution will be cancelled.
   * 默认情况下每个 beforeAction() 方法会触发一个 beforeAction 事件，在事件中你可以追加事件处理操作；
4. 控制器执行操作:
   * 请求数据解析和填入到操作参数；
5. 控制器按顺序调用控制器、模块（如果控制器属于模块）、应用主体的 afterAction() 方法；
   * 默认情况下每个 afterAction() 方法会触发一个 afterAction 事件， 在事件中你可以追加事件处理操作；
6. 应用主体获取操作结果并赋值给[响应](http://www.yiichina.com/doc/guide/2.0/runtime-responses).

## 最佳实践

在设计良好的应用中，控制器很精练，包含的操作代码简短； 如果你的控制器很复杂，通常意味着需要重构， 转移一些代码到其他类中。

归纳起来，控制器

* 可访问 [请求](http://www.yiichina.com/doc/guide/2.0/runtime-requests) 数据;
* 可根据请求数据调用 [模型](http://www.yiichina.com/doc/guide/2.0/structure-models) 的方法和其他服务组件;
* 可使用 [视图](http://www.yiichina.com/doc/guide/2.0/structure-views) 构造响应;
* 不应处理应被[模型](http://www.yiichina.com/doc/guide/2.0/structure-models)处理的请求数据;
* 应避免嵌入HTML或其他展示代码，这些代码最好在 [视图](http://www.yiichina.com/doc/guide/2.0/structure-views)中处理.