# 知Yii行易

## 一， 三个方面

1. 快速

2. 安全

3. 专业


## 二，分别概述

### A. 快速

**1，动态加载**

1-1，SPL + autoload + classmap

```php
/**
 * @var array class map for core Yii classes.
 * NOTE, DO NOT MODIFY THIS ARRAY MANUALLY. IF YOU CHANGE OR ADD SOME CORE CLASSES,
 * PLEASE RUN 'build autoload' COMMAND TO UPDATE THIS ARRAY.
 */
private static $_coreClasses=array(
    'CApplication' => '/base/CApplication.php',
    'CApplicationComponent' => '/base/CApplicationComponent.php',
    'CBehavior' => '/base/CBehavior.php',
    'CComponent' => '/base/CComponent.php',
    'CErrorEvent' => '/base/CErrorEvent.php',
    'CErrorHandler' => '/base/CErrorHandler.php',
    'CException' => '/base/CException.php',
    'CExceptionEvent' => '/base/CExceptionEvent.php',
    'CHttpException' => '/base/CHttpException.php',
    'CModel' => '/base/CModel.php',
    'CModelBehavior' => '/base/CModelBehavior.php',
    'CModelEvent' => '/base/CModelEvent.php',
    'CModule' => '/base/CModule.php',
    'CSecurityManager' => '/base/CSecurityManager.php',
    'CStatePersister' => '/base/CStatePersister.php',
```

```php
spl_autoload_register(array('YiiBase','autoload'));
```

1-2，先注册，需要时才加载

如：注册核心组件方法registerCoreComponents，先将组件信息写入数组，用到时再去实例化。

1-3，通过配置预加载

在应用配置文件中配置preload，可预加载组件。

```
return array(
    'basePath'=>dirname(__FILE__).DIRECTORY_SEPARATOR.'..',
    'name'=>'hacked by Yns0ng',

    // preloading 'log' component
    'preload'=>array('log'),
```

## 2，缓存

2-1，static静态缓存

autoload，记录日志，单例模式（Yii::app()）

2-2，应用缓存

- 数据缓存
- 缓存PHP变量到某种缓存媒介中，如：memcache
- 缓存组件基类CCache实现了ArrayAccess，可以像访问数组那样去访问缓存实例

如：

$mem = Yii::app()->memcache;

$mem[‘key’] = value; // 等价于：$mem->set(‘key’, value);

$value = $mem[‘key’]; // 等价于：$value = $mem->get(‘key’);

- 片段缓存
- 缓存网页中的某个片段

```
[php]
...别的HTML内容...
<?php if($this->beginCache($id)) { ?>
...被缓存的内容...
<?php $this->endCache(); } ?>
...别的HTML内容...
```

如：缓存分类页顶部品类导航。

- 多种缓存选项

缓存周期，依赖，请求类型等

- 支持嵌套缓存

- 页面缓存
- render(view, data, true) + 数据缓存(Redis)实现
- file_get_contents + 文件缓存

### 3，开发效率

3-1，开发效率高

既可以使用render方法渲染一个页面，又可以输出json数据与前端ajax交互。

## B. 安全

### 1，诸多安全防御措施

1-1，跨站请求攻击（XSS）

要避免XSS攻击，做到“验证输入，过滤输出”即可。提供CHtmlPurifier组件，封装HTMLPurifier类，可以有效防止XSS攻击。

1-2，SQL注入

Yii使用PDO操作数据库，使用prepare+bind的方式可以有效防止SQL注入。

1-3，跨站请求伪造（CSRF）

内建request组件支持开启CSRF验证机制，比较cookie和表单隐藏域中的token，一致则通过，否则400 Bad Request。需配合CHtml::form使用。

常规思路：比较session和表单隐藏域中的token。

```
<div class="form">
<form id="login-form" action="/index.php?r=site/login" method="post">
<input type="hidden" value="3e43a34513a95ff8522529a1cdc2f5796d324373" name="YII_CSRF_TOKEN" />
        <p class="note">Fields with <span class="required">*</span> are required.</p>

```

| Name | Value | Domain |
| --- | --- | --- |
| PHPSESSID | icu46vdp4nvdlojvb5jdnslp44 | www.yiilab.com |
| YII_CSRF_TOKEN | 3e43a34513a95ff8522529a1cdc2f5796d324373 | www.yiilab.com |

1-4，Cookie攻击

内建request组件支持开启Cookie验证机制，防止Cookie被篡改。使用内置的CHttpRequest组件操作Cookie，不要使用$_COOKIES。如：

```php
<?php

// 检索一个名为$name的cookie值
$cookie=Yii::app()->request->cookies[$name];
$value=$cookie->value;

// 设置一个cookie
$cookie=new CHttpCookie($name,$value);
Yii::app()->request->cookies[$name]=$cookie;

```

**2，诸多安全防御措施**

2-1，基本无漏洞


## C. 专业

**1，OOP**

1-1，使用了OOP的各种特性，如：封装，继承，多态，抽象等。

1-2，没有全局函数的概念，公用方法皆封装在组件里。

1-3，任务操作都是在操作对象，如：创建应用，启动应用，使用memcache组件，连接DB，读取配置文件中配置数据等。

**2，MVC**

2-1，分离逻辑层和表现层，前端开发人员只需掌握简单PHP语法即可，较之于使用Smarty更简单且代码更高效。通过AJAX技术，表现层也可以在前端实现，非常的方便灵活。