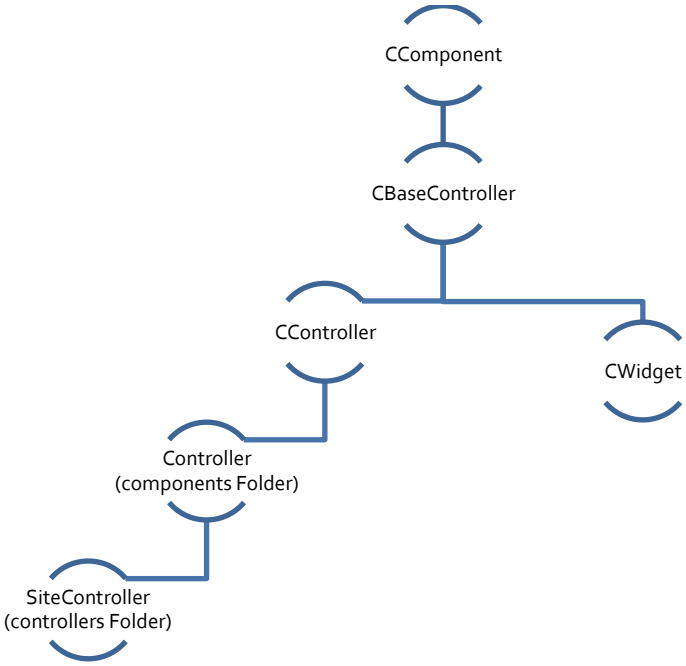
کنترلر

نمودار ارث بری های کنترلرها :



معرفی کنترلر :

فایل کنترلر ها در شاخه protected/controllers قرار می گیرند. در ابتدای شروع برنامه یک نمونه از کنترلر توسط Bootstrap یعنی فایل yiiBase.php ساخته و سپس اجرا می شود. هر کنترلر دارای یک نام منحصر به فرد یا ControllerID می باشد. کنترلر شامل تعدادی از اکشن ها می باشد که بر اساس درخواستهای کاربر کار خاصی را انجام می دهند. کنترلر ارطریق اکشن ها، جریان داده بین مدل ها و ویوها را مدیریت می کند. Action در واقع متد موجود در کنترلر است که حتما این متد با کلمه action شروع میشود.کاربر در فراخوانی هایش کلمه action را ذکر نمی کند مثل :

www.hostname.com/index.php?r=site/view

در این مثال کنترلر site و اکشنview فراخوانی میشوند ولی نام متد در اصل actionView است که کلمه action آن ذکر نمی گردد و این باعث افزایش امنیت نیز می شود. زیرا که کاربر مستقیما نام فایل را متوجه نمی شود. نحوه تعریف یک اکشن جدید :

```
class UpdateAction extends CAction
{
    public function run()
    {
        // place the action logic here
    }
}
```

اکشن می تواند پارامترهایی را هم داشته باشد که این پارامترها خود کار می باشند و مقدار آنها توسط $_GET از کاربر دریافت می شوند.

پارامترهای خودکار اکشن

یک متد اکشن می تواند پارامتهایی را برای آن تعریف نمود که مقادیر آنها را از طریق $_GET از کاربر دریافت نماید.در حالتی که نخواهیم از پارامترها خودکار استفاده نماییم اگر یک اکشن با نام create ایجاد نماییم که در کنترلر PostController تعریف شده باشد و دارای دو پارامتر category و languageباشد که مقادیر آنها را از ورودی بگیرد کد نویسی طولانی زیر را خواهیم داشت :

```
class PostController extends CController
{
    public function actionCreate()
    {
        if(isset($_GET['category']))
            $category=(int)$_GET['category'];
        else
            throw new CHttpException(404,'invalid request');

        if(isset($_GET['language']))
            $language=$_GET['language'];
        else
            $language='en';

        // ... fun code starts here ...
```

```
    }
}
```

حال با استفاده از پارامترهای خودکار مجددا همین کنترلر را تعریف می نماییم. که دارای کد نویسی ساده به شکل زیر می باشد.

```
class PostController extends CController
{
public function actionCreate($category, $language='en')
    {
        $category=(int)$category;

        // ... fun code starts here ...
    }
}
```

تعداد پارامترها و نام آنها باید دقیقا با آن چیزی که کاربر وارد می کند تطابق داشته باشد برای کد بالا درخواستی مثل کد زیر قابل قبول است:

www.hostname.com/index.php?r=Post/categoory=1&language=fa

اگر کاربر درخواستی بدهد که از این الگو یا الگوهای مورد قبول این الگو پیروی نکند یک خطای ۴۰۰ رخ می دهد. پارامتر language بالا دارای یک مقدار پیش فرض است $language='en' و در صورتی که کاربر این پارامتر را ارسال نکند خطایی رخ نمی دهد و مقدار پیش فرض برای آن در نظر گرفته می شود. اما چون category مقدار پیش فرض ندارد درخواستی که پارامتر category در آن نباشد باعث بروز خطا خواهد شد.

```
class PostController extends CController
{
public function actionCreate(array $categories)
    {
        // Yii will make sure $categories be an array
    }
}
```

در این حالت کلمه کلیدی arrayباید نوشته شود. کاربر می تواند با $_GET['categories'] کار کند که یک رشته عادی است ولی میتوان آن را به یک آرایه تبدیل کرد و از عناصر آن آرایه استفاده نمود.

وقتی که کاربر یک اکشن مثل XYZ را درخواست می کند، کنترلر یکی از کارهای زیر را انجام می دهد :

Method-based action : یعنی اینکه متد actionXYZ را در صورت وجود فراخوانی می کند.

Class-based action : یعنی اینکه یک نمونه از کلاس XYZ را در صورت موجود بودن کلاس در action class map ایجاد می کند. و سپس اکشن را فراخوانی می کند.

Call missingAction : یعنی اینکه به طور پیش فرض یک خطای HTTP ۴۰۴ را تولید می کند.

اگر کاربر هیچ اکشن مشخصی را درخواست ندهد اکشن defaultAction اجرا خواهد شد.defaultAction را به عنوان یک متغیر می توان در ابتدای کلاس کنترلر تعیین نمود.

کنترلر ممکن است که بخواهد قبل و یا بعد از اجرای یک اکشن فیلترهایی را اجرا کند.

پارامترهای عمومی قابل تعریف در کنترلر :

تعریف قالب :public $layout

می توان قالب پیش فرض برای استفاده در ویوهای مورد استفاده در این کنترلر را تعیین نمود که به طور پیش فرض main می باشد. اگر مقدار آن false تعریف شود هیچ قالبی استفاده نمی شود. مثال :

```
public $layout='//layouts/mylayout;
```

اگر کنترلر در یک ماژول قرار داشته باشد می توان برای تعیین قالب از دستور CWebModule::layout module layout استفاده می شود.

تعریف اکشن پیش فرض : public $defaultAction='index'

اکشن پیش فرض برای اجرا مشخص می شود. در اینصورت اگر در، درخواست کاربر اکشن مشخص نشود این اکشن پیش فرض اجرا می شود. البته از ابتدا مقدار آن 'index' استو باعث اجرای اکشن actionIndex می شود.مثال :

```
public $defaultAction='myAction';
```

متدهای قابل استفاده در کنترلر :

```
public function init()
```

کنترلر را مقدار دهی اولیه می کند. این متد توسط application و قبل از شروع اجرای کنترلر اجرا می شود. ممکن است شما این متد را override کنید تا بتوانید تنظیمات پیش از اجرای کنترلر را اعمال نمایید.

public function filters()

این متد تنظیمات فیلتر را بر می گرداند. این متد آرایه ای از مقادیر فیلتر را بر می گرداند که هر کدام از عناصر آرایه مربوط به هر یک از فیلتر ها هستند. کنترلر ممکن است که بخواهد قبل و یا بعد از اجرای یک اکشن فیلترهایی را اجرا کند.

فیلتر ها می توانند قبل و یا بعد از درخواست و یا پاسخ به کاربر اجرا شوند و ممکن است جلوی اجرای یک اکشن را بگیرند. در صورت لزوم فیلتر ها می توانند در یک ترتیب خاص اجرا شوند. در اینصورت اگر در هر مرجله ای از اجرا هر یک از فیلتر ها مقدار true را بر گرداند بقیه فیلترها و اکشن ها اجرا نخواهند شد.

فیلترها می توانند به صورت یک شی ساخته شده از روی یک کلاس مجزا باشند و یا به صورت متدهای تعریف شده در کلاس کنترلر. فیلتر ها با override کردن متد filters به شکل زیر اجرا می شوند. مثال :

```
<pre>
  array(
      'accessControl - login',
      'ajaxOnly + search',
      array(
          'COutputCache + list',
          'duration'=>300,
      ),
  )
</pre>
```

در مثال بالا ۳ فیلتر تعریف شده است که عبارتند از accessControl, ajaxOnly, COutputCache دو فیلتر اول یعنی accessControl, ajaxOnly, بر اساس متد ساخته شده اند. که متد آنها در کلاس CController تعریف شده است که به فیلتر کردن متد ها در کلاس کنترلر اشاره می کنند.

در حالی که سومین فیلتر از نوع شی است که کلاس آن system.web.widgets.COutputCache می باشد و پارامتر duration آن برابر مقدار ۳۰۰ قرار گرفته است. مثال دیگری به شکل زیر است :

```
class PostController extends CController
{
    ......
    public function filters()
    {
        return array(
            'postOnly + edit, create',
            array(
```

```
                'application.filters.PerformanceFilter - edit,
create',
                'unit'=>'second',
            ),
        );
    }
}
```

کد بالا دو فیلتر را تعریف می نماید.

متد فیلتر postOnly و کلاس PerformanceFilter– مسیر قرار گیری این کلاس در مسیر application.filters.PerformanceFilter است که کلاس فیلتر مورد نظر در فایل PerformanceFilter.php قرار دارد. که دارای یک property با نام unit است که مقدار second برای آن ارسال می شود.

عملگر + : مشخص می کند که این فیلتر برای کدامیک از اکشن ها اجرا شود. در مثال بالا اکشن postOnly باید برای اکشن های edit و create اجرا شود.

عملگر - : مشخص می کند که این فیلتر باید برای کدامیک از اکشن ها اجرا نشود. در مثال بالا فیلتر PerformanceFilter برای همه اکشن ها به جز edit و create باید اجرا شود.

نکته : اگر – یا + در فیلتر مشخص نشود آن فیلتر برای همه اکشن ها اجرا می شود.

برای فیلترهایی که بر اساس متد ساخته شده اند یک متد با نام filterXYZ به صورت filterXYZ($filterChain) تعریف می شود که نام این فیلتر XYZ است.

نکته : داخل متد فیلتر باید کد filterChain->run$() حتما نوشته شود تا زنجیره اجرایی ادامه پیدا کند وگرنه زنجیره اجرادر همین نقطه پایان می پذیرد.

نکته : اگر مقدار برگشتی یک متد فیلتر false باشد اکشن های مربوطه اجرا نخواهند شد.

فیلترها می توانند به گونه ای تعریف شوند که تنها زمانی اجرا شوند که یک اکشن خاص اجرا می شود. برای فیلترهای بر اساس متد این کار به وسیله عملگرهای + و – در تعریف فیلتر انجام می شود. عملگر + مشخص می کند که فیلتر تنها زمانی اجرا شود که یک فیلتر به خصوص فراخوانی شود در حالی که عملگر – به این معنی است که فیلتر تنها زمانی اجرا می شود که این اکشن در میان اکشن های درخواست قرار ندارد.برای اکشن های بر اساس شی عملگر + و – نام کلاس را دنبال می کنند.

فیلتر ها به دو دسته تقسیم می شوند :

۱- inline filter : که فیلترهای بر اسا متد هستند. و الگوی تعریف آنها به شکل زیر است:

```
FilterName[ +|- Action1, Action2, ...]
```

که عملگرهای + و – مشخص می کنند که کدام اکشن باید/نباید فیلتر گذاری شود.

۲- class-based filter : که فیلتر مربوطه به وسیله یک شی ساخته شده از روی کلاس تعریف می شود. این کلاس از کلاس Cfilter ارث بری می کند. مثال :

```
class PerformanceFilter extends CFilter
{
    protected function preFilter($filterChain)
    {
// logic being applied before the action is executed
return true; // false if the action should not be executed
    }

    protected function postFilter($filterChain)
    {
// logic being applied after the action is executed
    }
}
```

نحوه تعریف اینگونه فیلتر به شکل زیر است :

```
<pre>
 array(
      'FilterClass[ +|- Action1, Action2, ...]',
      'name1'=>'value1',
      'name2'=>'value2',
      ...
  )
</pre>
```

'name1'=>'value1' در این ساختار مقادیر property های فیلتر را مشخص می کنند.

نکته : برای ارث بری فیلترها از یکدیگر یک کلاس فرزند باید با کلاس والد خود ادغام شود که این کار می تواند توسط توابعی مثل array_merge انجام گیرد.

```
public function actions()
```

این متد لیستی از اکشن های موجود در کلاس خارجی را بر می گرداند. این متد شامل آرایه ای است که عناصر آن اکشن ها و کلاسهای آنها را مشخص می کنند. مثال :

```
'edit'=>'application.controllers.article.EditArticle'
```

در این مثال اکشنی با نام edit در کلاسی به مسیر 'application.controllers.article.EditArticle' فراخوانی می شود و از این به بعد در این کلاس قابل فراخوانی و استفاده است.همچنین می توان پارامترهایی را نیز به اکشن فراخوانی شده ارسال نمو. مثال :

```
    <pre>
      return array(
          'action1'=>'path.to.Action1Class',
          'action2'=>array(
              'class'=>'path.to.Action2Class',
              'property1'=>'value1',
              'property2'=>'value2',
          ),
      );
    </pre>
```

در مثال بالا action2 تعریف می شود که مسیر آن 'path.to.Action2Class' است و دو property نیز همراه با فراخوانی ارسال می شود.

یکی دیگر از کاربردهای این متد این است که اگر یک کلاس کنترلر از کلاس کنترلر ما مشتق شود توسط این متد می تواند اکشن های کلاس والد خود را فراخوتنی نماید.

نکته : برای ارث بری اکشن ها از یکدیگر یک کلاس فرزند باید با کلاس والد خود ادغام شود که این کار می تواند توسط توابعی مثل array_merge انجام گیرد.

شما همچنین می توانید اکشن ها را از یک action providerمثل CWidget::actions فراخوانی نمایید. مثال :

```
<pre>
return array(
...other actions...
// import actions declared in ProviderClass::actions()
// the action IDs will be prefixed with 'pro.'
'pro.'=>'path.to.ProviderClass',
// similar as above except that the imported actions are
values
'pro2.'=>array(
'class'=>'path.to.ProviderClass',
```

```
'action1'=>array(
'property1'=>'value1',
),
'action2'=>array(
'property2'=>'value2',
),
),
 )
</pre>
```

در مثال بالا ما action providers را از سایر تعاریف اکشن جدا کرده ایم برای action providers ها باید برای تعریف از یک نقطه استفاده نماییم بنابر این مثلا pro2.action1 به عنوان action1 شناخته می شود که در ProviderClass تعریف شده است.

public function behaviors()

لیستی از رفتارها را که کنترلر باید از خود نشان دهد بر می گرداند. این متد شامل آرایه ای است که در آن نام رفتار و مقدار آن مشخص می شود مثل name=>behavior. هر رفتار می تواند یک رشته معرف نوع رفتار کلاس باشد یا یک آرایه با ساختار زیر داشته باشد :

```
    <pre>
     'behaviorName'=>array(
         'class'=>'path.to.BehaviorClass',
         'property1'=>'value1',
         'property2'=>'value2',
     )
    </pre>
```

در مثال بالا 'behaviorName' یک رفتار است که در کلاسی با آدرس 'path.to.BehaviorClass' معرفی می شود و دو پارامتر با مقادیر مشخص شده را دریافت می کند.

توجه کنید که کلاس behavior بایستی از واسط IBehavior استفاده کند و یا از کلاس CBehavior ارث بری نماید. رفتارهای تعریف شده در کنترلر در زمان ساخت نمونه کنترلر توسط application به کلاس کنترلر ملحق می شوند.

توضیحات تکمیلی در مورد رفتارها را می توان در راهنمای شی گرایی در PHP و یا در کلاس CComponent مشاهده نمود.

public function accessRules()

این متد فیلترهای دسترسی کاربر به اکشن های کنترلر را مشخص می کند. به عنوان مثال مشخص می کند که کدام کاربر مجاز به اجرای کدام اکشن می باشد یا نمی باشد.

برای این که هر کاربری نتواند با وارد کردن هر آدرسی وارد آن صفحه شود از این قسمت استفاده می شود. مثلا کاربری که وارد سایت نشده یعنی login نکرده نمی تواند کنترلر Post را اجرا کند.

برای هر یک یا چند اکشن می توان یک آرایه جدا تشکیل داد و نحوه دسترسی کاربرانی که می توانند از آن استفاده کنند را مشخص کرد.

از کاراکتر * برای معرفی کردن همه کاربران و از کاراکتر @ برای معرفی کاربرانی که وارد سایت شده اند استفاده می کنیم همچنین از علامی ؟ برای کاربران ناشناس استفاده می شود. هر مدخل آرایه برای این متد به شکل زیر تعریف می شود :

```
array('deny or allow',
'actions'=>array('action1','action2',…),
     'users'=>array('@ or *')
),
```

ابزار gii در استفاده از ابزار Crud generator باعث می شود که مدخلی با شکل زیر ایجاد می کند که برای اجرای اکشن های کاربر باید آن را اصلاح نمود وگرنه اکشن های کاربر اجرا نخواهند شد.

```
array('deny',  // deny all users
     'users'=>array('*'),
),
```

در این اکشن همچنین می توان پارامترهای دیگری مثل Ip, Roles, Controllers, Verbs, Expression و غیره را نیز استفاده کرد به عنوان مثال از Ips می توان استفاده کرد تا کاربرانی با Ip مشخص اجازه استفاده از سایت را داشته باشند یا نداشته باشند. برای مدیریت گروهها نیز از Roles استفاده می شود که برای توضیحات بیشتر به ویکی پدیا بخش RBAC مراجعه نمایید.

روش دیگر این کار تعریف یک شرط است مثل:

```
array('allow',
          'actions'=>array('admin'),
          'expression'=> 'Yii::app()->user->group == 3',
     ),
```

public function run($actionID)

این متد نام یک اکشن را گرفته و آن را اجرا می کند. فیلترهای اکشن مورد نظر نیز اجرا خواهند شد. اگر اکشن مورد نظر پیدا نشد و یا نام اکشن صحیح وارد نشده باشد یک CHttpException رخ می دهد.

public function runActionWithFilters($action,$filters)

این متد یک فیلتر را با یک فیلتر مشخص شده اجرا می کند. یک زنجیره فیلتر در این مرحله ایجاد شده و سپس اکشن مورد نظر اجرا می شود.

public function runAction($action)

این متد یک اکشن را پس از اعمال تمامی فیلترها مرتبط با این اکشن اجرا می کند.

public function createAction($actionID)

این متد یک نمونه از اکشن معرفی شده را تولید می کند. این اکشن می تواند یک اکشن inline یعنی تعریف شده در داخل همین کلاس کنترلر باشد و یا یک اکشن object یعنی تعریف شده در کلاس خارجی و سپس نمونه سازی شده باشد.

public function missingAction($actionID)

این متد بررسی می کند که آیا اکشن معرفی شده موجود می باشد یا خیر و در صورتی که موجود نباشد یک خطا صادر می کند.

public function getRoute()

این متد رشته تقاضای جاری به شکل module ID, controller ID and action ID را بر می گرداند.

public function getAction()

این متد نام اکشن فعال را بر می گرداند. در صورتی که هیچ اکشن فعالی وجود نداشته باشد مقدار null را بر می گرداند.