Vues

les classes Phalcon\Mvc\View et Phalcon\Mvc\View\Simple permettent la manipulation des vues dans le cadre du design pattern MVC.

-- Intégration des vues avec les contrôleurs

Phalcon passe automatiquement l'exécution à un composant de type vue dès qu'un contrôleur à terminé son chargement. La vue à charger est recherchée dans le dossier views, dans un sous-dossier du même nom que le dernier contrôleur invoqué, le fichier de vue portant le nom de la dernière action exécutée.
Pour prendre un exemple, si une requête est faite vers l'url **http://127.0.0.1/blog/posts/show/301**, Phalcon va parser l'url de la façon suivante :

Root de l'application	blog
Controller	posts
Action	show
Paramètre	301

Le dispatcher recherche “**PostsController**” et son action “**showAction**”.

Exemple de contrôleur :

```
<?php

class PostsController extends \Phalcon\Mvc\Controller{

    public function indexAction(){

    }

    public function showAction($postId){
        // Passage du paramètre $postId (301 dans l'exemple) à la vue
        $this->view->setVar("postId", $postId);
    }

}
```

La méthode **setVar** permet de créer des variables à la demande de façon à ce qu'elles puissent être utilisées dans la vue. L'exemple montre comment passer des paramètres ($postId) à la vue.

-- Scan hiérarchique

Le composant par défaut pour les vues (Phalcon\Mvc\View) gère l'affichage à partir d'une hiérarchie de fichiers. Cette hiérarchie permet d'utiliser des zones de layout (fréquemment utilisées dans les vues) à condition que les templates soient présents dans les dossiers correspondant au contrôleur et à l'action.

Phalcon\Mvc\View utilise PHP par défaut comme moteur de template, à condition que les vues aient l'extension **.phtml**. Si le dossier des vues est app/views (comme défini dans la configuration) le composant view recherchera automatiquement les 3 fichiers suivants :

Rôle	Fichier	Description
Main Layout	app/views/index.phtml	Affiché pour chaque contrôleur et chaque action de l'application
Controller Layout	app/views/layouts/posts.phtml	Vue relative au contrôleur, visible pour l'exécution de chacune des action du contrôleur “**posts**”. Tout le code implémenté dans ce layout sera utilisé pour toutes les actions du contrôleur
Action View	app/views/posts/show.phtml	Vue relative à l'action, visible uniquement si l'action “**show**” est exécutée.

```
<!-- app/views/index.phtml -->
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <h1>main layout!</h1>

        <?php echo $this->getContent() ?>

    </body>
</html>
```

```
<!-- app/views/layouts/posts.phtml -->

<h2>"posts" controller layout!</h2>

<?php echo $this->getContent() ?>
```

```
<!-- app/views/posts/show.phtml -->

<h3>show view!</h3>

<p>Paramètre passé : <?php echo $postId ?></p>
```

Résultat :


```
<!-- app/views/index.phtml -->
<!DOCTYPE html>
<html>
    <head>
        <title>Phalcon PHP Framework Layout example</title>
    </head>
    <body>
        <h1>Main layout!</h1>
<!-- app/views/layouts/posts.phtml -->

<h2>"posts" controller layout!</h2>

<!-- app/views/posts/show.phtml -->

<h3>show view!</h3>

<p>Paramètre passé : 301</p>    </body>
</html>
```

-- Utilisation de templates

Les templates permettent de factoriser et de partager une partie de l'affichage.

Exemple : Utilisation d'un template pour affichage d'une action

```
class PostsController extends \Phalcon\Mvc\Controller{
    public function initialize(){
        $this->view->setTemplateAfter('common');
    }
...
    public function lastAction(){
        $this->flash->notice("Derniers posts");
    }
}
```

La méthode **setTemplateAfter** de la classe **view** applique le template après le controller layout.

Le template **common** :

```
<!-- app/views/layouts/common.phtml -->

<ul class="menu">
    <li><a href="/">Home</a></li>
    <li><a href="/articles">Articles</a></li>
    <li><a href="/contact">Contact</a></li>
</ul>

<div class="content"><?php echo $this->getContent() ?></div>
```

La vue correspondant à l'action **last** :

```
<!-- app/views/posts/show.phtml -->

<h3>last view!</h3>
```

-- Contrôle hiérarchique du niveau d'affichage

Niveau	**::**	
LEVEL_NO_RENDER	Indicates to avoid generating any kind of presentation.	
LEVEL_ACTION_VIEW	Generates the presentation to the view associated to the action.	1
LEVEL_BEFORE_TEMPLATE	Generates presentation templates prior to the controller layout.	2
LEVEL_LAYOUT	Generates the presentation to the controller layout.	3
LEVEL_AFTER_TEMPLATE		
LEVEL_MAIN_LAYOUT		