Contrôleurs

Un contrôleur Phalcon est une classe héritant de \Phalcon\Mvc\Controller, et dont les méthodes publiques sont qualifiées d'actions, accessibles par l'url. Les actions sont responsables de l'interprétation des requêtes et de la création de la réponse.

-- URLs

Lors de l'accès à l'URL **http://localhost/blog/posts/show/2012/the-post-title**, Phalcon décompose chaque partie de l'url selon le principe suivant :

Root de l'application	blog
Controller	posts
Action	show
Paramètre	2012
Paramètre	the-post-title

Les contrôleurs doivent avoir le suffixe **Controller** et les actions le **suffixe** Action

Exemple :

```
<?php

class PostsController extends \Phalcon\Mvc\Controller{

    public function indexAction(){

    }

    public function showAction($year, $postTitle){

    }

}
```

URL d'accès à l'action :

- /posts/show/2015/elections

Les paramètres de l'action peuvent être facultatifs s'ils ont une valeur par défaut :

```
<?php

class PostsController extends \Phalcon\Mvc\Controller{

    public function indexAction(){

    }

    public function showAction($year=2015, $postTitle='some default title'){
```

```
    }

}
```

URLs d'accès à l'action :

- /posts/show/2015/elections
- /posts/show/2015/
- /posts/show/

-- Redirections

Une action peut-être redirigée vers une autre via la redirection :
Exemple : Redirection vers l'action Inscription du contrôleur **Users** si l'utilisateur n'est pas autorisé à accéder à l'action **show** du contrôleur **Posts**

```
<?php

class PostsController extends \Phalcon\Mvc\Controller{

    public function indexAction(){

    }

    public function showAction($year, $postTitle){
        $this->flash->error("Vous n'avez pas l'autorisation d'accéder à cette
zone");

        // Redirection vers une autre action
        $this->dispatcher->forward(array(
            "controller" => "users",
            "action" => "inscription"
        ));
    }

}
```

```
<?php
class UsersController extends \Phalcon\Mvc\Controller{

    public function indexAction(){

    }

    public function signinAction(){

    }

}
```

-- Initialisation des contrôleurs

La classe Phalcon Phalcon\Mvc\Controller dispose d'une méthode **initialize**, invoquée avant tout appel d'action.

```
<?php

class PostsController extends \Phalcon\Mvc\Controller
{

    public $settings;

    public function initialize()
    {
        $this->settings = array(
            "mySetting" => "value"
        );
    }

    public function saveAction()
    {
        if ($this->settings["mySetting"] == "value") {
            //...
        }
    }

}
```

La méthode **initialize** n'est appelée que si l'événement '**beforeExecuteRoute**' est exécuté avec succès, pour éviter qu'une partie de la logique de l'application ne soit exécutée sans autorisation.

-- Injection de services

Les contrôleurs ont accès par défaut à l'injecteur de services **$di** défini dans le fichier bootstrap (index.php ou services.php) :

```
<?php
...

$di = new Phalcon\DI();

$di->set('storage', function() {
    return new Storage('/some/directory');
}, true);
...
```

Accès au service injecté dans un contrôleur :

```
<?php
class FilesController extends \Phalcon\Mvc\Controller{

    public function saveAction(){

        //Injecting the service by just accessing the property with the same name
        $this->storage->save('/some/file');

        //Accessing the service from the DI
        $this->di->get('storage')->save('/some/file');

        //Another way to access the service using the magic getter
        $this->di->getStorage()->save('/some/file');

        //Another way to access the service using the magic getter
        $this->getDi()->getStorage()->save('/some/file');

        //Using the array-syntax
        $this->di['storage']->save('/some/file');
    }

}
```

-- Request et response

Les classes Phalcon\Http\Request et Phalcon\Http\Response représentent les services permettant d'accéder respectivement à la requête et à la réponse HTTP.

Exemples d'usage des classes request et Response :

```
class PostsController extends Phalcon\Mvc\Controller
{

    public function indexAction()
    {

    }

    public function saveAction()
    {
        // Vérifie si la requête a été postée
        if ($this->request->isPost() == true) {
            // Accède aux données du POST
            $customerName = $this->request->getPost("name");
            $customerBorn = $this->request->getPost("born");
        }
    }

    public function notFoundAction()
    {
        // Envoie le code d'erreur 404 dans les en-têtes de la réponse
        $this->response->setStatusCode(404, "Not Found");
    }
```

```
}
```

-- Persistance des données

Les sessions permettent de gérer la persistance des données entre les requêtes : il est possible d'accéder à la classe Phalcon\Session\Bag depuis n'importe quel contrôleur.

```
<?php

class UserController extends Phalcon\Mvc\Controller{

    public function indexAction()
    {
        $this->persistent->name = "Michael";
    }

    public function welcomeAction()
    {
        echo "Welcome, ", $this->persistent->name;
    }

}
```

Il est également possible d'utiliser $this→session pour assurer la persistance. Les données ajoutés à la session ($this→session) sont disponibles à travers toute l'application, tandis qu'avec $this→persistant, on ne peux y accéder qu'à partir de la portée de la classe courante.

-- Base controller

Moteur de template, gestion du cache, ACL, translation sont souvent communs à plusieurs contrôleurs. Il est intéressant dans ce cas de factoriser le code en créant un contrôleur de base, gérant ces services communs, et d'en faire hériter les autres contrôleurs.

```
<?php

class ControllerBase extends \Phalcon\Mvc\Controller{

  /**
   * Action disponible pour tous les contrôleurs héritant de ControllerBase
   */
  public function someAction(){

  }

}
```

Tout contrôleur héritant de ControllerBase aura maintenant accès aux fonctionnalités de ControllerBase.

```
<?php

class UsersController extends ControllerBase{

}
```

-- Evènements dans les contrôleurs

Les contrôleurs disposent d'écouteurs sur les évènements du dispatcher.
Implémenter les méthodes suivantes permet d'agir avant et après que l'action soit exécutée.

```
<?php

class PostsController extends \Phalcon\Mvc\Controller{

    public function beforeExecuteRoute($dispatcher){
        // This is executed before every found action
        if ($dispatcher->getActionName() == 'save') {

            $this->flash->error("You don't have permission to save posts");

            $this->dispatcher->forward(array(
                'controller' => 'home',
                'action' => 'index'
            ));

            return false;
        }
    }

    public function afterExecuteRoute($dispatcher){
        // Executed after every found action
    }

}
```

Le retour (false/true) permet éventuellement d'annuler l'action.