

Phalcon routage

Le composant de routage [Phalcon\Phalcon\MvcRouter](#) permet de définir des routes mappées vers les contrôleurs ou vers des handlers qui reçoivent la requête. Le routage peut fonctionner en mode MVC ou en mode match-only.

-- Définition de routes

En mode MVC, il est possible de définir des routes et de les mapper vers les actions des contrôleurs.

-- Routes simples

Quelques exemples :

```
<?php

// Instanciation du routeur
$routeur = new \Phalcon\MvcRouter();

//Création d'une route
$routeur->add(
    "/admin/users/my-profile",
    array(
        "controller" => "users",
        "action"      => "profile",
    )
);

//Ajout d'une autre route
$routeur->add(
    "/admin/users/change-password",
    array(
        "controller" => "users",
        "action"      => "changePassword",
    )
);

$routeur->handle();
```

Utilisation de la syntaxe courte pour définir les routes :

```
<?php

//Création d'une route
$routeur->add("/admin/users/my-profile","users::profile");

//Ajout d'une autre route
```

```
$router->add("/admin/users/change-password", "users::changePassword");
```

-- Routes génériques

Il est possible de définir des routes plus génériques en utilisant les expressions régulières (syntaxe [PRCE](#))

Exemple :

```
<?php
// Create the router
$router = new \Phalcon\Mvc\Router();

//Define a route
$router->add(
    "/admin/:controller/a/:action/:params",
    array(
        "controller" => 1,
        "action"      => 2,
        "params"      => 3,
    )
);
```

Les alias **:controller**, **:action** et **:params** simplifient l'expression et évitent l'usage direct d'expressions régulières.

- Le premier paramètre de add définit le pattern de l'url interceptée
- Le second définit comment les parties de l'url se répartissent en controller, action et parameters

Si l'url frappée est **/admin/users/a/delete/dave/301**, elle sera interprétée en :

- contrôleur : users
- action : delete
- param : dave
- param : 301

Alias disponibles :

Alias	Expression régulière associée	Usage
/:module	/([a-zA-Z0-9_-]+)	Correspond à un nom de module valide avec caractères alphanumériques seulement
/:controller	/([a-zA-Z0-9_-]+)	Correspond à un nom de contrôleur valide avec caractères alphanumériques seulement
/:action	/([a-zA-Z0-9_-]+)	Correspond à un nom d'action valide avec caractères alphanumériques seulement
/:params	(/.*)*	Liste de paramètres optionnels séparés par des /. A utiliser uniquement à la fin de l'URL
/:namespace	/([a-zA-Z0-9_-]+)	Correspond à un namespace
/:int	/([0-9]+)	Correspond à un paramètre de type entier

-- Nommage des paramètres

```
<?php
$router->add(
    "/news/([0-9]{4})/([0-9]{2})/([0-9]{2})/:params",
    array(
        "controller" => "posts",
        "action"      => "show",
        "year"        => 1, // ([0-9]{4})
        "month"       => 2, // ([0-9]{2})
        "day"         => 3, // ([0-9]{2})
        "params"     => 4, // :params
    )
);
```

Une URL de la forme /news/2015/1/11/last/desc sera interprétée de la façon suivante :

- controller : posts
- action : show
- year : 2015
- month : 1
- day : 11
- params : last et desc

Version courte avec paramètres nommés :

```
<?php
$router->add(
    "/news/{year:([0-9]{4})}/{month:([0-9]{2})}/{day:([0-9]{2})}/:params",
    array(
        "controller" => "posts",
        "action"      => "show",
        "params"     => 4
    )
);
```

-- Récupération des paramètres nommés dans l'action cible

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

    public function indexAction(){

    }
}
```

```
public function showAction(){

    // Return "year" parameter
    $year = $this->dispatcher->getParam("year");

    // Return "month" parameter
    $month = $this->dispatcher->getParam("month");

    // Return "day" parameter
    $day = $this->dispatcher->getParam("day");

}

}
```

-- Fonctions de vérification

Les routes doivent parfois satisfaire des conditions bien spécifiques qu'il est possible d'inclure avec la méthode **beforeMatch** : si elle retourne false, la route sera considérée comme non correspondante :

```
<?php

$router->add('/login', array(
    'module' => 'admin',
    'controller' => 'session'
))->beforeMatch(function($uri, $route) {
    //Vérifie si la requête est réalisée en Ajax
    if ($_SERVER['HTTP_X_REQUESTED_WITH'] == 'xmlhttprequest') {
        return false;
    }
    return true;
});
```

-- Groupes de routes

Si un ensemble de routes a des chemins identiques, il est possible de les regrouper pour les gérer plus facilement :

```
<?php

$router = new \Phalcon\Mvc\Router();

//Create a group with a common module and controller
$blog = new \Phalcon\Mvc\Router\Group(array(
    'module' => 'blog',
    'controller' => 'index'
));

//All the routes start with /blog
$blog->setPrefix('/blog');
```

```
//Add a route to the group
$blog->add('/save', array(
    'action' => 'save'
));

//Add another route to the group
$blog->add('/edit/{id}', array(
    'action' => 'edit'
));

//This route maps to a controller different than the default
$blog->add('/blog', array(
    'controller' => 'blog',
    'action' => 'index'
));

//Add the group to the router
$router->mount($blog);
```

-- Initialisation du service routage

L'initialisation du service routage se fait par injection de dépendance dans le fichier index.php ou services.php (si le projet a été créé avec webtools).

```
<?php

/**
 * Ajout de la fonctionnalité routage
 */
$di->set('router', function(){
    $router = new \Phalcon\Mvc\Router();
    require __DIR__.'../app/config/routes.php';
    return $router;
});
```

Le fichier **app/config/routes.php** peut ensuite définir les différentes routes :

```
<?php
$router->add("/admin/users/my-profile","users::profile");

//Ajout d'une autre route
$router->add("/admin/users/change-password","users::changePassword");
```

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/slam4/php/phalcon/routes?rev=1421112189>

Last update: **2019/08/31 14:41**



