Programmation orientée objet en PHP

Le ServiceLocator

Nous allons modifier notre application pour l'adapter à une architecture orientée service. Chaque classe de notre application sera un "service", c'est-à-dire une unité fonctionnelle possédant un rôle déterminé. Par exemple, la classe PersonneDao assurera la persistance des Personne(s), ou encore, la classe Router assurera le routing de notre application. Ce "service" pourra être sollicité par un autre service (une classe de service pourra posséder des dépendances vers d'autres classes de service), ou être utilisé dans les controllers.
Pour gérer l'instanciation des services, l'injection des dépendances et la récupération des services par les controllers, nous allons utiliser une classe ServiceLocator. Cette classe fournira au composant qui le demande le service prêt à l'emploi. Le ServiceLocator stockera dans un tableau les instances des services, et renverra l'instance du service demandé. Les services ne seront instanciés et stockés dans le tableau qu'à la première demande. Ensuite c'est l'instance stockée dans le tableau qui sera toujours renvoyée. Il n'y aura donc qu'une instance de chaque service.
Pour décrire les services et leurs dépendances, nous utiliserons un fichier service.json au format JSON qui contiendra, pour chaque service, le nom du service, le nom de la classe représentant ce service, les arguments à passer au constructeur :

```
[
    {
        "name": "router",
        "class": "util/Router"
    },
    {
        "name": "personne_dao",
        "class": "dao/PersonneDao"
    },
    {
        "name": "personne_service",
        "class": "business/PersonneService",
        "arguments": "@personne_dao"
    }
]
```

Le caractère @ devant un argument signifie que cet argument est lui-même un service.

Nous allons ajouter au fichier setup.php une constante qui indiquera le fichier de service :

```
<?php

// setup.php
define('SITE_URL', 'http://localhost:8080');
define('ROOT', realpath(__DIR__));
define('CONFIG', ROOT . '/config');
define('CONTROLLER', ROOT . '/controller');
define('ENTITY', ROOT . '/entity');
define('DAO', ROOT . '/dao');
define('VIEW', ROOT . '/view');
define('INI_FILE', CONFIG . '/conf.ini');
define('ROUTING_FILE', CONFIG . '/routing.json');
define('SERVICE_FILE', CONFIG . '/service.json');

spl_autoload_register(function($class) {
    list($dir, $file) = explode('\\', $class);
    include_once ROOT . "/$dir/$file.php";
```

```
});
```

Les attributs d'un service pourront être encapsulés dans un objet de type Service :

```php
<?php

// util/Service.php

namespace util;

class Service {

    private $name;
    private $class;
    private $arguments;

    function __construct($name, $class = '', $arguments = []) {
        $this->name = $name;
        $this->class = $class;
        $this->arguments = $arguments;
    }

    public function getName() {
        return $this->name;
    }

    public function getClass() {
        return $this->class;
    }

    public function getArguments() {
        return $this->arguments;
    }

    public function setName($name) {
        $this->name = $name;
    }

    public function setClass($class) {
        $this->class = $class;
    }

    public function setArguments($arguments) {
        $this->arguments = $arguments;
    }

}
```

La classe ServiceLocator possédera une méthode publique get() qui prendra en argument le nom du service, et qui renverra l'instance de ce service :

```php
<?php

// util/ServiceLocator.php

namespace util;

use ReflectionClass;
use util\Service;

class ServiceLocator {

    private $services;

    public function __construct() {
```

```
        $this->services = [];
    }

    public function get($serviceName) {
        $conf_text = file_get_contents(SERVICE_FILE);
        $conf = json_decode($conf_text, true);
        return $this->getService($conf, $serviceName);
    }

    private function getService($conf, $serviceName) {
        if (array_key_exists($serviceName, $this->services)) {
            return $this->services[$serviceName];
        } else {
            return $this->findService($conf, $serviceName);
        }
    }

    private function findService($conf, $serviceName) {
        foreach ($conf as $service) {
            if ($service['name'] == $serviceName) {
                $serviceObj = $this->createService($conf, $service);
            }
        }
        $class = new ReflectionClass($serviceObj->getClass());
        $instance = $class->newInstanceArgs($serviceObj->getArguments());
        $this->services[$serviceObj->getName()] = $instance;
        return $instance;
    }

    private function createService($conf, $service) {
        $serviceObj = new Service($service['name']);
        $className = str_replace('/', '\\', $service['class']);
        $serviceObj->setClass($className);
        if (array_key_exists('arguments', $service)) {
            $arguments = $this->getArguments($conf, $service);
            $serviceObj->setArguments($arguments);
        }
        return $serviceObj;
    }

    private function getArguments($conf, $service) {
        $arguments = explode(',', $service['arguments']);
        for ($i = 0; $i < count($arguments); $i++) {
            if ($arguments[$i][0] == '@') {
                $arguments[$i] = $this->getService($conf, substr($arguments[$i], 1));
            }
        }
        return $arguments;
    }

}
```

Une fois les services implémentés, nous pourrons tester notre ServiceLocator avec une classe de test ServiceLocatorTest :

```
<?php

include './setup.php';
include './util/ServiceLocator.php';

use util\ServiceLocator;

class ServiceLocatorTest extends PHPUnit_Framework_TestCase {

    private $locator;

```

```
12      protected function setUp() {
13          $this->locator = new ServiceLocator();
14      }
15
16      public function testGet() {
17          $router = $this->locator->get('router');
18          $router2 = $this->locator->get('router');
19          $personneDao = $this->locator->get('personne_dao');
20          $personneService = $this->locator->get('personne_service');
21          $this->assertInstanceOf('util\Router', $router);
22          $this->assertInstanceOf('util\Router', $router2);
23          $this->assertSame($router, $router2);
24          $this->assertInstanceOf('dao\PersonneDao', $personneDao);
25          $this->assertInstanceOf('business\PersonneService', $personneService);
26          $this->assertSame($personneDao, $personneService->getDao());
27      }
28
29  }
```

Fin

NAVIGATION

Accueil
- Ma page
  - Pages du site
  - Mon profil
  - Cours actuel
    - POO PHP
      - Participants
      - Généralités
      - La programmation orientée objet : premiers pas
      - L'héritage
      - Les interfaces
      - Le typage
      - Les namespaces
      - Les exceptions
      - Les bases de données avec PDO
      - Les tests avec PHPUnit
      - Petite application version 2
      - Petitie application version 3
      - **Le ServiceLocator**
      - La dao
      - Les classes de traitement métier
      - Les contrôleurs
  - Mes cours

ADMINISTRATION

Administration du cours

Réglages de mon profil