Programmation orientée objet en PHP

Les classes de traitement métier

Nous allons implémenter des classes de service métier dont le rôle va être de contenir les traitements "métiers" de notre application. Pour l'instant, lorsque nous ajoutons une personne, le traitement métier de vérification du nom et du prénom est contenu dans le controller. Ce dernier possède donc, dans cette méthode, une double responsabilité : vérifier les champs de la Personne et rediriger vers la bonne page.
Notre classe de service métier va nous permettre d'extraire ce traitement du controller.

Dans un répertoire business, nous allons créer une classe PersonneService qui contiendra tous les traitements métiers liés à l'entité Personne. Cette classe ne contiendra que deux méthodes :

1. addPersonne() qui vérifiera la validité de la personne et permettra sa persistance
2. getAllPersonnes() qui ne fera qu'appeler la méthode du même nom de la DAO (on appelle cela de la délégation d'appel)

Cette classe possédera une dépendance vers un objet de type IPersonneDao. Cette dépendance lui sera fournie par le constructeur. Notre classe de service implémentera aussi une interface qui déclarera les méthodes qu'un service gérant les Personne(s) doit implémenter :

```php
<?php

// business/IPersonneService.php

namespace business;

interface IPersonneService {

    public function addPersonne($nom, $prenom);

    public function getAllPersonnes();
}
```

La classe PersonneService implémentera cette interface :

```php
<?php

// business/PersonneService.php

namespace business;

use entity\Personne;
use dao\IPersonneDao;
use business\IPersonneService;

class PersonneService implements IPersonneService {

    private $dao;

    public function __construct(IPersonneDao $dao) {
        $this->dao = $dao;
    }

    public function addPersonne($nom, $prenom) {
        $nom = filter_var($nom, FILTER_VALIDATE_REGEXP
                , ['options' => ['regexp' => '/^[A-Za-z- ]{1,20}$/']]);
        $prenom = filter_var($prenom, FILTER_VALIDATE_REGEXP
```

```
                , ['options' => ['regexp' => '/^[A-Za-z0-9]{4,10}$/']]);
        if ($nom && $prenom) {
            $personne = new Personne($nom, $prenom);
            $result = $this->dao->addPersonne($personne);
            return $result;
        } else {
            return false;
        }
    }

    public function getAllPersonnes() {
        $personnes = $this->dao->getAllPersonnes();
        return $personnes;
    }

    public function getDao() {
        return $this->dao;
    }

}
```

Nous allons pouvoir tester la classe de service en créant un mock de sa dépendance, c'est-à-dire la classe implémentant IPersonneDao. Dans le répertoire test/business nous créons une classe PersonneServiceTest :

```
<?php

include './business/IPersonneService.php';
include './business/PersonneService.php';
include './entity/Personne.php';
include './dao/IPersonneDao.php';

use business\PersonneService;

class PersonneServiceTest extends PHPUnit_Framework_TestCase {

    private $service;

    protected function setUp() {
        $dao = $this->getMock('dao\IPersonneDao');
        $dao->expects($this->any())->method('addPersonne')
                ->with($this->isInstanceOf('entity\Personne'))->willReturn(1);
        $this->service = new PersonneService($dao);
    }

    public function testAddPersonne() {
        $nom = "Sparrow";
        $prenom = "Jack";
        $result = $this->service->addPersonne($nom, $prenom);
        $this->assertEquals(1, $result);
        $result = $this->service->addPersonne('R2D2', 'azerty');
        $this->assertFalse($result);
    }

}
```

Fin

NAVIGATION

Accueil
- Ma page

Pages du site
Mon profil
Cours actuel

ADMINISTRATION

Administration du cours

Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP