PHP

Bertrand Estellon

Aix-Marseille Université

13 mars 2012

Modèle-Vue-Contrôleur

Le **Modèle-Vue-Contrôleur** (MVC) est un méthode de conception utilisée pour organiser l'interface homme-machine (IHM) d'une application.

Le modèle : gère le comportement et les données de l'application.
↪ Exemple : gestion des interactions avec une base de données.

La vue : est une interface avec laquelle l'utilisateur interagit. Elle présente des parties du modèle à l'utilisateur et reçoit les actions de l'utilisateur.
↪ Exemple : code HTML/JavaScript présenté aux clients.

Le controleur : analyse les requêtes des clients, décide les actions à effectuer sur le modèle, et choisit les vues à envoyer aux clients.
↪ Exemple : analyse des informations passées dans l'URL

Projet MVC – Réalisation d'un site de sondages

Aix*Marseille université

Présentation du projet : Les utilisateurs postent des sondages constitués d'une question et de plusieurs réponses. Une fois le sondage posté, il est accessible aux visiteurs du site qui pourront voter pour une des réponses proposées. Les nombres de voix obtenues pour chaque réponse sont comptabilisés et affichés sous la forme d'un graphique. Tous les visiteurs peuvent chercher parmi les sondages et voter.

Fichier 'index.php'

```php
session_start();

function getActionByName($name) {
        $name .= 'Action'; require("actions/$name.inc.php");
        return new $name();
}

function getViewByName($name) {
        $name .= 'View'; require("views/$name.inc.php");
        return new $name();
}

function getAction() {
        if (!isset($_REQUEST['action'])) $action = 'Default';
        else $action = $_REQUEST['action'];
        $actions = array('Default', 'SignUp', 'Login', ...);
        if (!in_array($action, $actions)) $action = 'Default';
        return getActionByName($action);
}

$action = getAction(); $action->run(); $view = $action->getView();
$model = $action->getModel(); $view->run($model);
```

Les actions du projet

Les actions possibles d'un visiteur :

- SignUpForm : affichage du formulaire d'inscription
- SignUp : demande d'inscription
- Login : connexion du visiteur
- Logout : déconnexion du visiteur
- UpdateUserForm : affichage du formulaire de modification de profil
- UpdateUser : modification du profil
- AddSurveyForm : affichage du formulaire d'ajout de sondage
- AddSurvey : ajout d'un sondage
- GetMySurveys : affichage des sondages du visiteur
- Search : recherche
- Vote : prise en compte d'un vote

La classe Action

Aix*Marseille université

Toutes les actions sont définies en étendant la classe suivante :

```
abstract class Action {
        private $view;
        private $model;
        protected $database;

        public function __construct(){ ... }

        protected function setView($view) { ... }
        protected function setModel($model) { ... }
        protected function getSessionLogin() { ... }
        protected function setSessionLogin($login) { ... }

        public function getView() { ... }
        public function getModel() { ... }

        abstract public function run();
}
```

Exemple : l'action SignUpForm

```
class SignUpFormAction extends Action {

   /**
    * Dirige l'utilisateur vers le formulaire d'inscription.
    *
    * @see Action::run()
    */
    public function run() {
       $this->setModel(new MessageModel());
       $this->getModel()->setLogin($this->getSessionLogin());
       $this->setView(getViewByName("SignUpForm"));
    }

}
```

L'action commence par construire le modèle (nous verrons par la suite ce que contient les instances de la classe *MessageModel*). Ensuite, elle détermine la vue qui va être utilisée pour afficher le modèle.

Le modèle

Le modèle contient les classes permettant de manipuler les objets “métiers” du site (ici, les sondages) et de modifier la base de données. Aucune fonctionnalité de représentation des données n’est fournie par le modèle.

La classe permettant de représenter un sondage est donnée ci-dessous :

```
class Survey {
        private $id;
        private $owner;
        private $question;
        private $responses;

        public function __construct($owner, $question) { ... }
        public function setId($id) { ... }
        public function addResponse($response) { ... }
        public function getId() { ... }
        public function getOwner() { ... }
        public function getQuestion() { ... }
        public function getResponses() { ... }
        public function computePercentages() { ... }
}
```

Le modèle

Aix*Marseille université

Vous avez également une classe représentant une réponse :

```
class Response {
  private $id;
  private $survey;
  private $title;
  private $count;
  private $percentage;

  public function __construct($survey, $title, $count = 0) { ... }
  public function setId($id) { ... }
  public function computePercentage($total) { ... }
  public function getId() { ... }
  public function getSurvey() { ... }
  public function getTitle() { ... }
  public function getCount() { ... }
  public function getPercentage() { ... }
}
```

Les interaction avec la base de données se feront via une instance de classe *Dababase* (créée dans le constructeur de classe *Action*). Nous détaillerons les fonctionnalités de cette classe plus tard.

La classe Model

Toutes les extensions de la classe *Model* peuvent être passée à des vues pour être affichée. La classe *Model* contient les informations de base nécessaire à l'affichage de la page :

```
class Model {
        private $login;
        private $loginError;

        public function __construct() { ... }
        public function getLogin() { ... }
        public function setLogin($login) { ... }
        public function getLoginError() { ... }
        public function setLoginError($loginError) { ... }
}
```

Dans le projet, nous avons deux classes qui étendent *Model* :

- MessageModel : contient un message à afficher.
- SurveysModel : contient une liste de sondages à afficher.

La classe View

Les différentes vues du projet sont définies en étendant la classe *View* :

```
abstract class View {
        public function run($model) {
                $login = $model->getLogin();
                require("templates/page.inc.php");
        }

        private function displayLoginForm($model) { ... }
        private function displayLogoutForm($model) { ... }
        private function displayCommands($model) { ... }
        private function displaySearchForm($model) { ... }

        protected abstract function displayBody($model);

}
```

Les vues définissent la méthode *displayBody* afin de générer le contenu de la page (en conservant les bordures, le formulaire de connexion, etc.).

La génération de la page – page.inc.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>...</head>
<? $login = $model->getLogin(); ?>
<body>
  <div class="header">
    <div class="title">Site de sondages</div>
      <div class="loginForm"><?
          if ($login===null) $this->displayLoginForm($model);
          else $this->displayLogoutForm($model);
      ?></div>
    </div>
...
    <div class="content">
      <? $this->displayBody($model); ?>
    </div>
...
</body>
</html>
```

Les vues

Les différentes vues du projet :

- DefaultView : page vide
- MessageView : affiche un message à l'utilisateur
- SignUpFormView : formulaire d'inscription
- UpdateUserFormView : formulaire de modification de mot de passe
- AddSurveyFormView : formulaire d'ajout de sondage
- SurveysView : liste de sondages

Remarques :

- *DefaultView* utilise un modèle de type *Model*.
- *MessageView*, *SignUpFormView*, *UpdateUserFormView*, *AddSurveyFormView* utilisent un modèle de type *MessageModel*.
- *SurveysView* utilise un modèle de type *SurveysModel*.

Exemple : la vue SurveysView

La vue suivante permet d'afficher une liste de sondages :

```
class SurveysView extends View {

  public function displayBody($model) {
    if (count($model->getSurveys())===0) {
      echo "Aucun sondage ne correspond a votre recherche.";
      return;
    }

    foreach ($model->getSurveys() as $survey) {
      $survey->computePercentages();
      require("templates/survey.inc.php");
    }
  }

}
```

Elle inclut le fichier *survey.inc.php* pour générer le code HTML représentant le sondage contenu dans la variable `$survey`.

Exemple : déroulement d'une inscription

- Le client demande la page *index.php* ;
- Le serveur affiche la vue par défault (vide) ;
- Le client clique sur *Inscrivez-vous !*
- Le navigateur demande *index.php ?action=SignUpForm* ;
- Le serveur affiche la vue *SignUpFormView* ;
- L'utilisateur remplit le formulaire et l'envoie ;
- Le navigateur poste le formulaire vers *index.php ?action=SignUp* ;
- Le serveur effectue l'inscription si aucune erreur ne s'est produite ;
- Le serveur affiche la vue *MessageView* (avec un message de confirmation) ou *SignUpFormView* (avec un message d'erreur) ;
- ...