

Modèle-Vue-Contrôleur

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

Le modèle : 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 contrôleur : 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 1 – Sondages



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.



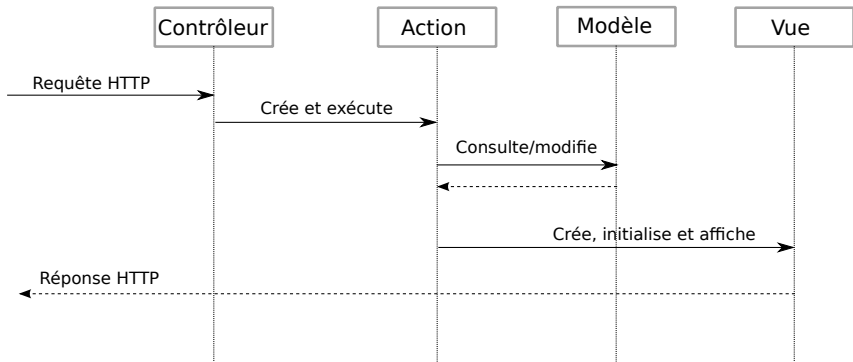
Question 1

Réponse 1 
Réponse 2 

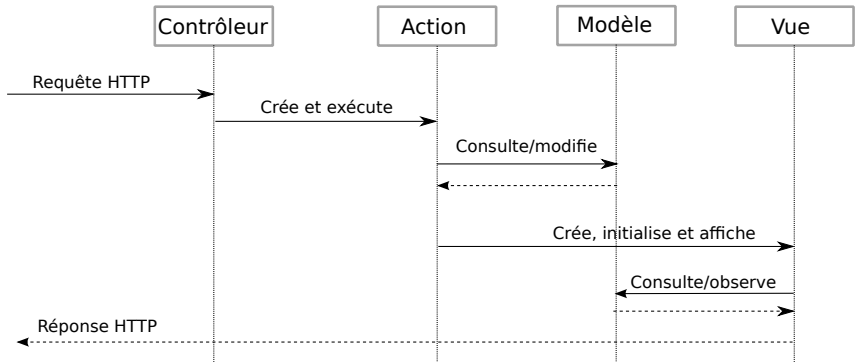
Question 2

Réponse 1 
Réponse 2 

Organisation générale



Organisation générale



Le contrôleur (index.php)

Les fonctions du contrôleur :

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

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

function getAction() { /* Factory */
    if (!isset($_REQUEST['action'])) $action = 'Default';
    else $action = $_REQUEST['action'];
    $actions = array('Default', 'SignUpForm', ...);
    if (!in_array($action, $actions)) $action = 'Default';
    return getActionByName($action);
}
?>
```

Le contrôleur (index.php)

Exécution du contrôleur :

```
<?  
session_start();  
$action = getAction();  
$action->run();  
$view = $action->getView();  
$action->getView()->setLogin($action->getSessionLogin());  
$view->run();  
?>
```

Les actions du projet

Les actions :

- ▶ [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

Toutes les actions sont définies en étendant la classe suivante :

```
<?
abstract class Action {
    private $view;
    protected $database;

    public function __construct(){
        $this->view = null;
        $this->database = new Database();
    }

    protected function setView($view) { $this->view = $view; }

    public function getView() { return $this->view; }

    /* ... */
}
?>
```


La classe Action

Toutes les actions sont définies en étendant la classe suivante :

<?

```
abstract class Action {  
    /* ... */  
  
    public function getSessionLogin() {  
        if (isset($_SESSION['login'])) $login = $_SESSION['login'];  
        else $login = null;  
        return $login;  
    }  
  
    protected function setSessionLogin($login) {  
        $_SESSION['login'] = $login;  
    }  
  
    protected function setMessageView($message, $style="") {  
        $this->setView(getViewByName("Message"));  
        $this->getView()->setMessage($message, $style);  
    }  
  
    abstract public function run();  
}
```

?>

Exemple : l'action updateUser

```
<?
class updateUserAction extends Action {
    private function setUpdateUserFormView($message) {
        $this->setView(getViewByName("UpdateUserForm"));
        $this->getView()->setMessage($message, "alert-error");
    }

    public function run() {
        if ($this->getSessionLogin()===null) {
            $this->setMessageView("Vous devez être authentifié.");
            return;
        }
        $updatePassword = $_POST['updatePassword'];
        $updatePassword2 = $_POST['updatePassword2'];
        if (!isset($updatePassword) || !isset($updatePassword2)) {
            $this->setUpdateUserFormView("Vous devez remplir le formulaire.");
            return;
        }
        /* ... */
    }
}

?>
```

Exemple : l'action updateUser

```
<?
class updateUserAction extends Action {

    public function run() {
        /* ... */

        $res = $this->database->updateUser($this->getSessionLogin(),
                                           $updatePassword);

        if ($res!==true) {
            $this->setUpdateUserFormView($res);
            return;
        }

        $this->setMessageView("Modification enregistrée.", "alert-success");
    }
}
?>
```

Le modèle

Le modèle contient les classes permettant de manipuler les objets “métiers” du site 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 getId() { /*...*/ }
    public function getOwner() { /*...*/ }
    public function getQuestion() { /*...*/ }
    public function &getResponses() { /*...*/ }
    public function setResponses($responses) { /*...*/ }
    public function addResponse($response) { /*...*/ }
    public function computePercentages() { /*...*/ }
}
```

?>

Le modèle

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 *Database* (instanciée dans le constructeur de classe *Action*). Nous détaillerons les fonctionnalités de cette classe plus tard.

La classe View

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

<?

```
abstract class View {  
    protected $message = "";  
    protected $style = "";  
    protected $login = null;  
  
    public function run() { require("templates/page.inc.php"); }  
  
    public function setMessage($message, $style="") {  
        $this->message = $message; $this->style = $style;  
    }  
  
    public function setLogin($login) { $this->login = $login; }  
}
```

?>

La classe View

<?

```
abstract class View {  
  
    /* ... */  
  
    private function displayLoginForm() { require("templates/loginform.inc.php"); }  
    private function displayLogoutForm() { require("templates/logoutform.inc.php"); }  
    private function displayCommands() { require("templates/commands.inc.php"); }  
    private function displaySearchForm() { require("templates/searchform.inc.php"); }  
  
    protected abstract function displayBody();  
}
```

?>

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

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Sondages</title>
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <div class="navbar navbar-inverse navbar-fixed-top">
      <div class="navbar-inner">
        <div class="container">
          <? $this->displaySearchForm(); ?>
          <? if ($this->login===null) $this->displayLoginForm();
            else $this->displayLogoutForm(); ?>
        </div>
      </div>
    </div>
    <? $this->displayBody(); ?>
  </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

Exemple : la vue SurveysView

Par exemple, la vue suivante permet d'afficher une liste de sondages :

```
<?
class SurveysView extends View {
    private $surveys;

    public function displayBody() {
        if (count($this->surveys)===0) { ?>
            <div class="container"><br><br><br><br>
                <div style="text-align:center" class="alert">
                    Aucun sondage ne correspond à votre demande.
                </div>
            </div>';
            <?
            return;
        }
        require("templates/surveys.inc.php");
    }

    public function setSurveys($surveys) { $this->surveys = $surveys; }
}
?>
```

Exemple : la vue SurveysView

La vue précédente inclue le code du fichier *surveys.inc.php* :

```
<?
<div class="container"><br><br><br>
<div class="span7 offset2">
  <ul class="media-list">
    <?
      foreach ($this->surveys as $survey) {
        $survey->computePercentages();
        require("survey.inc.php");
      }
    ?>
  </ul>
</div>
</div>
?>
```

Exemple : la vue SurveysView

La vue précédente inclue le code du fichier *surveys.inc.php* :

```
<?
<div class="container"><br><br><br>
<div class="span7 offset2">
  <ul class="media-list">
    <?
      foreach ($this->surveys as $survey) {
        $survey->computePercentages();
        require("survey.inc.php");
      }
    ?>
  </ul>
</div>
</div>
?>
```

Exemple : la vue SurveysView

La page *survey.inc.php* doit générer un code HTML de la forme suivante :

```
<li class="media well">
  <div class="media-body">
    <h4 class="media-heading">Question</h4>
    <div class="fluid-row">
      <div class="span2">Réponse 1</div>
      <div class="span2 progress progress-striped active">
        <div class="bar" style="width: 20%"></div>
      </div>
      <span class="span1">(20%)</span>
      <form class="span1" method="post" action="index.php?action=Vote">
        <input type="hidden" name="responseId" value="1">
        <input type="submit" style="margin-left:5px"
          class="span1 btn btn-small btn-danger" value="Voter">
      </form>
    </div>
    <!-- Les autres réponses possibles -->
  </div>
</li>
```

Exemple : déroulement d'une inscription

- ▶ Le client demande la page *index.php*;
- ▶ Le serveur affiche la vue par défaut (vide);
- ▶ Le client clique sur *Inscription*;
- ▶ 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);
- ▶ ...