SIN6U5 : Développement web 2 (Prog. web coté serveur) Modèle-Vue-Contrôleur

Luigi Santocanale
LIF, Aix-Marseille Université

Transparents basés sur le cours de Bertrand Estellon

8 février 2018

Plan

Plan

Pourquoi (et c'est quoi) MVC

Le code précédent ne peut pas évoluer facilement :

- Comment ajouter une page ? Que modifier ou réutiliser ?
- Sémantique et responsabilité des différents fichiers pas bien définies

Vers un découpage respectant le motif d'architecture logicielle MVC :

- Le [M]odèle permet de manipuler des données
- Les [V]ues présentent des données
- Le [C]ontrôleur analyse les demandes des clients, agit sur ou consulte le modèle et utilise des vues pour présenter des données aux clients

Remarques autour de MVC

- Motif d'architecture logicielle qui remonte à 1978.
- Developpé/utilisé pour les interfaces graphiques.

Modèle :

- Les données, la logique en rapport avec les données : validation, lecture et enregistrement.
- Univers dans lequel s'inscrit l'application.
- Indépendant de la vue et du contrôleur.

Vue :

- Partie visible d'une interface graphique.
- Se sert du modèle, et peut être un diagramme, un formulaire, des boutons, etc.
- Contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant du modèle.
- Dans une application web une vue contient des balises HTML.

Contrôleur :

- traite les actions de l'utilisateur,
- modifie les données du modèle et de la vue.

Frameworks MVCs

Frameworks MVC :

- CodeIgniter, Symfony, Zend Framework,
- Django (Python), Ruby on Rails (Ruby).

Non MVC :

- `http://peej.github.com/tonic/`
- `http://www.recessframework.org/`
- `http://flourishlib.com/`

Plan

Des URL aux méthodes des contrôleurs

`http://www.example.com/index.php/registration/registrations_new`

```php
<?php
// controllers/Registration.php

class Registration extends Controller {
    public function registrations_new() {
        ...
    }
    ...
}
```

Le contrôleur frontal : *index.php*

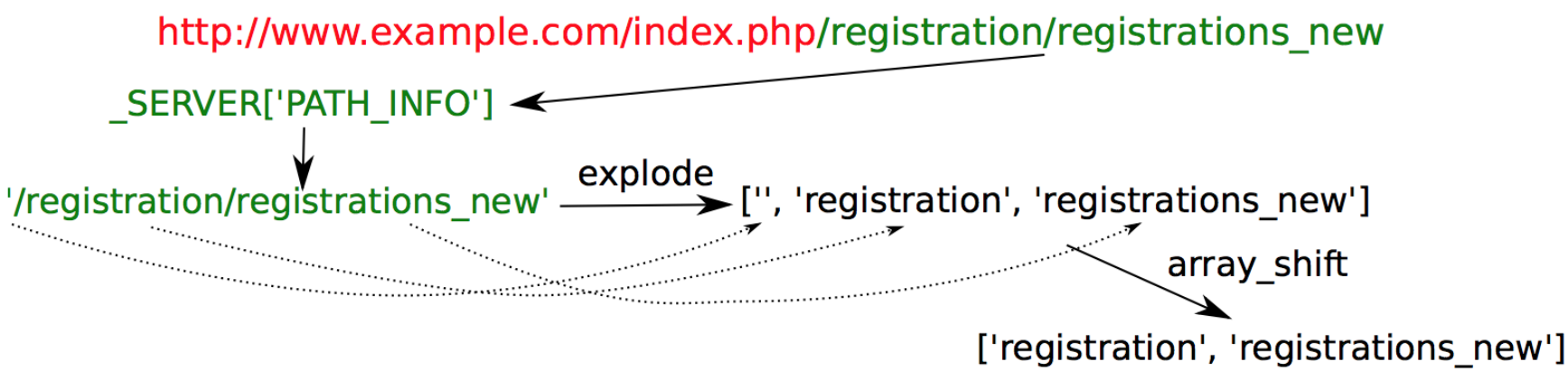
```php
//index.php
include 'config/config.php';

foreach ($config['core_classes'] as $classname){
    require "core/$classname.php";
}


/*
Code principal : les fonctions ci-dessous
get_path_elements, get_controller_name, create_controller,
call_controller_method
vont être définies
*/
$path_elements = get_path_elements();
$controller_name = get_controller_name($path_elements);
$method_name = get_method_name($path_elements);
$controller = create_controller($controller_name);
call_controller_method($controller, $method_name);
```

Les fonctions de *index.php*

```php
function get_path_elements() {
    if (!isset($_SERVER['PATH_INFO'])) return null;
    $path_info = $_SERVER['PATH_INFO'];
    $path_elements = explode('/', $path_info);
    array_shift($path_elements);
    return $path_elements;
}
```



Le tableau superglobal $_SERVER

Contient des informations comme les en-têtes, dossiers et chemins du script :

http://php.net/manual/fr/reserved.variables.server.php

```php
$indicesServerTwo = [
    'PHP_SELF',
    'SERVER_NAME',
    'QUERY_STRING',
    'DOCUMENT_ROOT',
    'SCRIPT_FILENAME',
    'SCRIPT_NAME',
    'REQUEST_URI',
    'PATH_INFO'
];

echo '<table cellpadding="10">' ;
foreach ($indicesServerTwo as $arg) {
    if (isset($_SERVER[$arg])) {
        echo '<tr><td>'.$arg.'</td><td>' . $_SERVER[$arg] .
 ↪   '</td></tr>' ;
    }
    else {
        echo '<tr><td>'.$arg.'</td><td>-</td></tr>' ;
    }
}
echo '</table>' ;
```

Le tableau superglobal $_SERVER II

PHP_SELF	/serverInfo.php/hallo/33/
SERVER_NAME	localhost
QUERY_STRING	hallo=ola&ciao=salut
DOCUMENT_ROOT	/Users/lsantoca/AmuBox/DW/latex/ch05_MVC/code/cours
SCRIPT_FILENAME	/Users/lsantoca/AmuBox/DW/latex/ch05_MVC/code/cours/serverInfo.php
SCRIPT_NAME	/serverInfo.php
REQUEST_URI	/serverInfo.php/hallo/33/?hallo=ola&ciao=salut
PATH_INFO	/hallo/33/

Les fonctions de *index.php*

```php
function path_contains_controller_name($path_elements) {
    return $path_elements!==null && count($path_elements)>=1
                            && ctype_alnum($path_elements[0]);
}
```

```php
function path_contains_method_name($path_elements) {
    return $path_elements!==null && count($path_elements)>=2;
}
```

Remarques : l'extension PHP pour la vérification de types de caractères :

`http://php.net/manual/fr/book.ctype.php`

Les fonctions de *index.php*

```php
<?php
// config/config.php

$config = [
        'default_controller' => 'Registration',
        'default_method' => 'index',
        'core_classes'=> ['Loader', 'Controller', 'Model'],
        'database'=>'db/database.sqlite',
        'models' =>['Registration']
];
```

```php
//index.php
include 'config/config.php';

foreach ($config['core_classes'] as $classname){
    require "core/$classname.php";
}
```

Les fonctions de *index.php*

```php
function generate_error_404() {
    header("HTTP/1.0 404 Not Found");
    exit;
}
```

```php
function get_controller_name($path_elements) {
    global $config;

    return (path_contains_controller_name($path_elements))
        ? $path_elements[0] : $config['default_controller'];
}
```

```php
function get_method_name($path_elements) {
    global $config;

    return (path_contains_method_name($path_elements))
        ? $path_elements[1] : $config['default_method'];
}
```

Les fonctions de *index.php*

```php
function create_controller($controller_name) {
    $controller_classname = ucfirst(strtolower($controller_name));
    $controller_filename =
 ↪   'controllers/'.$controller_classname.'.php';

    if (!file_exists($controller_filename)) generate_error_404();
    require $controller_filename;

    if (!class_exists($controller_classname)) generate_error_404();

    return new $controller_classname();
}
```

```php
function call_controller_method($controller, $method_name) {
    $reflectionObject = new ReflectionObject($controller);
    if (!$reflectionObject->hasMethod($method_name))
 ↪   generate_error_404();
    $reflectionMethod = $reflectionObject->getMethod($method_name);
    $reflectionMethod->invoke($controller);
}
```

API Reflection

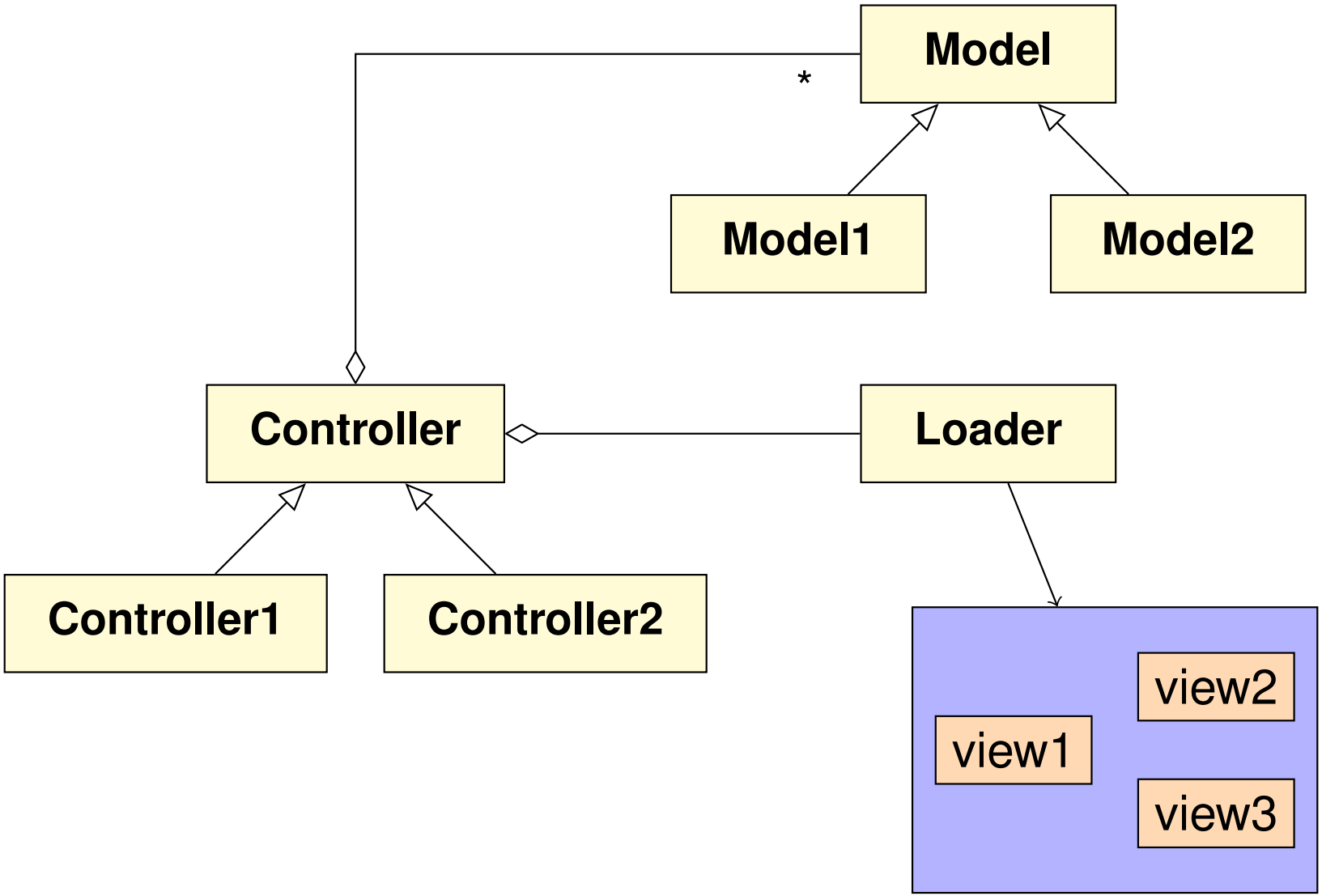
- Permet de faire du *reverse-engineer* sur les classes, les interfaces, les fonctions, les méthodes et les extensions.
- Récupérer les commentaires des fonctions, des classes et des méthodes.

`http://php.net/manual/fr/book.reflection.php`
`http://php.net/manual/fr/class.reflectionobject.php`

Exemple de *reverse-engineer* : PHP2XMI, PHP console script that generates an XMI scheme representing your classes and interfaces . . .
. . . the XMI scheme can be imported into UML modelers like umbrello to browse, print, think about your library/application design.
FEATURES public, protected, private, abstract, static, interface, extends, implements method argument type discovery through php5 type hinting @package @param type @return type @type attribute support . . . Uses PHP5 builtin reflection

Plan

Diagramme de classes



Organisation générale du projet

Les différents répertoires et fichiers de l'application :

- **assets** : `css/bootstrap.min.css`, ...
- **core** : `Controller.php`, `Loader.php`, `Model.php`
- **controllers** : `Registration.php`
- **db** : `database.sqlite`, `create.sql`
- **models** : `Registration_model.php`
- **views** : `header.php`, `footer.php`, `register_form.php`, ...
- **config** : `config.php`
- `index.php`

Le fichier `config.php`

```php
<?php
// config/config.php

$config = [
        'default_controller' => 'Registration',
        'default_method' => 'index',
        'core_classes'=> ['Loader', 'Controller', 'Model'],
        'database'=>'db/database.sqlite',
        'models' =>['Registration']
];
```

Plan

Le fichier core/Model.php

```php
<?php
// core/Model.php

class Model {
    private static $static_db;
    protected $db;

    public function __construct() {
        $this->db = self::$static_db;
    }

    public static function init() {
        global $config;
        self::$static_db = new PDO('sqlite:'.$config['database']);
    }
}

Model::init();
```

Le fichier *model/Registration_model.php*

```php
<?php
// model/Registration_model.php

class Registration_model extends Model {

    public function courses() {

    }

    public function register($firstname, $lastname, $courses) {

    }

    public function registrations() {

    }

}
```

Plan

La classe `Controller`

```php
<?php
// core/Controller.php
class Controller {
    protected $loader;
    public function __construct() {
    }
    private function init_models() {
    }
    private function init_model($model) {
    }
    protected function compute_error($form_data, $filters) {
    }
}
```

La classe `Controller`

```php
public function __construct() {
    $this->loader = new Loader();
    $this->init_models();
}

private function init_models() {
    global $config;
    foreach ($config['models'] as $model) {
        $this->init_model($model);
    }
}


private function init_model($model) {
    $class = $model . '_model';
    require "model/$class.php";
    $variable = strtolower($model);
    $this->$variable = new $class();
}
```

Le contrôleur *Registration*

```php
<?php

// controllers/Registration.php

class Registration extends Controller {

    public function index() {

    }

    public function registrations() {

    }

    public function registrations_new() {

    }

    public function registrations_create() {

    }

    //!-- filters --
    const FORM_FILTERS = [
        'firstname' =>
        [
            'filter' => FILTER_VALIDATE_REGEXP,
            'flags' => FILTER_NULL_ON_FAILURE,
            'options' => ['regexp' => '/^[A-Za-z\' -]{1,20}$/'],
```

Le contrôleur `Registration`

```php
public function index() {
    /* -> index.php/registration/index */
    $this->loader->load();
}

public function registrations() {
    /* -> index.php/registration/registrations */
    $data['registrations'] = $this->registration->registrations();
    $this->loader->load('registration_list', $data);
}

public function registrations_new() {
    /* -> index.php/registration/registrations_new */
    $data = ['courses' => $this->registration->courses()];
    $this->loader->load('registration_form', $data);
}
```

Le contrôleur `Registration`

```
    public function registrations_create() {
        $form_data = filter_input_array(INPUT_POST,
↪   self::FORM_FILTERS, true);
        $error = $this->compute_error($form_data, self::FORM_FILTERS);
        if ($error !== false) {
            $data = ['courses' => $this->registration->courses(),
                'error' => $error];
            $this->loader->load('registration_form', $data);
            return;
        }
        /* sauvegarde dans le modele et choix d'une vue */
        $this->registration->register(
                $form_data['firstname'], $form_data['lastname'],
↪   $form_data['courses']);
        $this->registrations(); // Bonne idée ???
    }
```

Le contrôleur Registration

```
const FORM_FILTERS = [
    'firstname' =>
    [
        'filter' => FILTER_VALIDATE_REGEXP,
        'flags' => FILTER_NULL_ON_FAILURE,
        'options' => ['regexp' => '/^[A-Za-z\' -]{1,20}$/'],
        'message' => 'Le prénom doit etre alphaNum'
    ],
    'lastname' =>
    [
        'filter' => FILTER_VALIDATE_REGEXP,
        'flags' => FILTER_NULL_ON_FAILURE,
        'options' => ['regexp' => '/^[A-Za-z\' -]{1,20}$/'],
        'message' => 'Le prénom doit etre alphaNum'
    ],
    'courses' => [
        'filter' => FILTER_VALIDATE_INT,
        'flags' => FILTER_REQUIRE_ARRAY,
        'options' => ['min_range' => 1, 'max_range' => 3],
        'message' => 'Vous devez choisir au moins un cours'
    ]
];
```

La classe Loader

```php
class Loader {

    public function view($view, $data = []) {
        foreach ($data as $key=>$value){
            $$key = $value;
        }
        include "views/$view.html.php";
    }

    public function load($view=null, $data=[]) {

        $this->define_helper();
        $this->view ('header');
        $this->view ('top_bar');
        if ($view!==null){
            $this->view($view,$data);
        }
        $this->view ('footer');
    }

    /* Les méthodes du slides suivant. */
```

La classe `Loader`

```
    /* Les méthodes du slide précédent. */
    public function site_url($url) {
        return "/index.php/$url";
    }

    public function define_helper() {

        function set_value($name) {
            return filter_input(INPUT_POST, $name,
                                FILTER_SANITIZE_SPECIAL_CHARS,
                                ['options' => ['default' => '']]);
        }
            }

}
```

Plan

Les premières vues

views/header.html.php

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Inscription</title>
    <link href="/assets/css/bootstrap.css"
          rel="stylesheet">
    <link href="/assets/css/style.css"
          rel="stylesheet">
  </head>
  <body>
```

views/footer.html.php

```
</body>
</html>
```

Les premières vues

```
<!-- views/top_bar.html.php -->
<div class="navbar navbar-static-top">
  <div class="container">
    <div class="navbar-header">
      <a class="navbar-brand" href="index.html">Inscription</a>
    </div><!-- navbar-header -->
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav navbar-right">
        <li><a

 ↪   href="<?=$this->site_url('registration/registrations_new') ?>">
            S'inscrire
        </a></li>
        <li><a
              href="<?=$this->site_url('registration/registrations')
 ↪   ?>">
            Liste des inscrits</a>
        </li>
      </ul>
    </div><!-- navbar-collapse collapse -->
  </div><!-- container -->
</div><!-- navbar -->
```

Le contrôleur *Registration*

```php
<?php

// controllers/Registration.php

class Registration extends Controller {

    public function index() {
        /* -> index.php/registration/index */
        $this->loader->load();
    }

    public function registrations() {
        /* -> index.php/registration/registrations */
        $data['registrations'] = $this->registration->registrations();
        $this->loader->load('registration_list', $data);
    }

    public function registrations_new() {
        /* -> index.php/registration/registrations_new */
        $data = ['courses' => $this->registration->courses()];
        $this->loader->load('registration_form', $data);
    }

    public function registrations_create() {
```

La vue registration_list

```
<!-- views/registration_list.html.php -->
<br>
<div class="container">
  <table class="table table-striped table-bordered">
    <tr>
      <th>Prénom</th>
      <th>Nom</th>
      <th>Cours</th>
    </tr>
    <?php foreach ($registrations as $registration): ?>
    <tr>
      <td><?=$registration['firstname'] ?></td>
      <td><?=$registration['lastname'] ?></td>
      <td><?=$registration['courses'] ?></td>
    </tr>
    <?php endforeach; ?>
  </table>
</div>
```

Les actions

Les différentes actions de notre projet :

- `registrations` : liste des inscriptions
- `registrations_new` : formulaire d'ajout
- `registrations_create` : création d'une inscription

Quelques remarques :

- Les noms des actions doivent être cohérents
- Les frameworks permettent de traiter des routes plus complexes :

registrations	$\Rightarrow$	`registrations`
registrations/new	$\Rightarrow$	`registrations_new`
registrations/12	$\Rightarrow$	`registrations_get(12)`
registrations/12/edit	$\Rightarrow$	`registrations_edit(12)`
books/12/chapters	$\Rightarrow$	`books_chapters(12)`
books/12/chapters/2	$\Rightarrow$	`books_chapters_get(2)`

Formulaire d'inscription

L'action qui permet à l'utilisateur d'avoir le formulaire d'inscription :

```
    public function registrations_create() {
        $form_data = filter_input_array(INPUT_POST,
↪   self::FORM_FILTERS, true);
        $error = $this->compute_error($form_data, self::FORM_FILTERS);
        if ($error !== false) {
            $data = ['courses' => $this->registration->courses(),
                'error' => $error];
            $this->loader->load('registration_form', $data);
            return;
        }
        /* sauvegarde dans le modele et choix d'une vue */
        $this->registration->register(
                $form_data['firstname'], $form_data['lastname'],
↪   $form_data['courses']);
        $this->registrations(); // Bonne idée ???
    }
```

Formulaire d'inscription

```
<!-- views/registration_form.html.php -->
<br />
<div class="container">
    <form method="post"
          action="<?=
 ↪   $this->site_url('registration/registrations_create') ?>">

        <?php if (isset($error)): ?>
            <div class="alert"><?= $error ?></div>
        <?php endif; ?>

        <input type="text" name="firstname"
               value="<?= set_value('firstname') ?>"
               placeholder="Votre nom">
        <input type="text" name="lastname"
               value="<?= set_value('lastname') ?>"
               placeholder="Votre prénom">

        <?php foreach ($courses as $course): ?>
            <input type="checkbox" name="courses[]"
                   value="<?= $course['id'] ?>">
                   <?= $course['name'] ?>
                <?php endforeach; ?>
        <br />
        <button type="submit">Valider</button>
    </form>
</div>
```

Traitement de l'inscription

Méthode du contrôleur pour le traitement de l'inscription :

```
    public function registrations_create() {
        $form_data = filter_input_array(INPUT_POST,
↪   self::FORM_FILTERS, true);
        $error = $this->compute_error($form_data, self::FORM_FILTERS);
        if ($error !== false) {
            $data = ['courses' => $this->registration->courses(),
                'error' => $error];
            $this->loader->load('registration_form', $data);
            return;
        }
        /* sauvegarde dans le modele et choix d'une vue */
        $this->registration->register(
                $form_data['firstname'], $form_data['lastname'],
↪   $form_data['courses']);
        $this->registrations(); // Bonne idée ???
    }
```

Traitement de l'inscription

Méthode du contrôleur pour le traitement de l'inscription :

`$this->registrations();` // Bonne idée ???

Mauvaise idée car :

- L'URL du navigateur ne correspondra pas à la vue présentée
- Si l'utilisateur recharge la page, l'action `registrations_create` sera exécutée à nouveau avec le risque d'une nouvelle inscription

Solution : rédiger le navigateur de l'utilisateur vers la bonne URL

```
/* Redirection du navigateur */
header("Location: http://www.monsite/registrations/");
exit;
```

Autres lectures

MVC

- `https://openclassrooms.com/courses/codeigniter-le-framework-au-service-des-zeros/presentation-de-l-architecture-mvc-1`

Contrôleur frontal :

- `https://www.sitepoint.com/front-controller-pattern-1/`
- `https://www.sitepoint.com/front-controller-pattern-2/`