

Documentation framework 3WA

Organisation des dossiers et fichiers

/application	Code source de l'application
/config	Configuration de l'application
database.php	Configuration base de données
(identifiants de connexion)	
/controllers	Le C de MVC, les chefs
d'orchestres de l'application	
HomeController.php	Contrôleur de la page d'accueil
/models	Le M de MVC, le coeur de
l'application	
/www	Le V de MVC, les fichiers
statiques (CSS, JS, images, fonts, etc.)	
/css	Feuilles de styles CSS
/fonts	Polices de caractères
/images	Images
/js	Code source JavaScript
/classes	Classes (code source orienté
objet)	
HomeView.phtml	Template interne de la page
d'accueil	
LayoutView.phtml	Template global

/library	Code source générique, réutilisable dans d'autres projets
Database.php	Classe d'accès à la base de données
Form.php	Classe pour gérer les formulaires
FlashBag.php	Classe qui permet de créer des messages flash
Http.php	Classe qui gère toutes les requêtes HTTP
index.php	Point d'entrée de l'application

Afficher une page

Fonctionnement

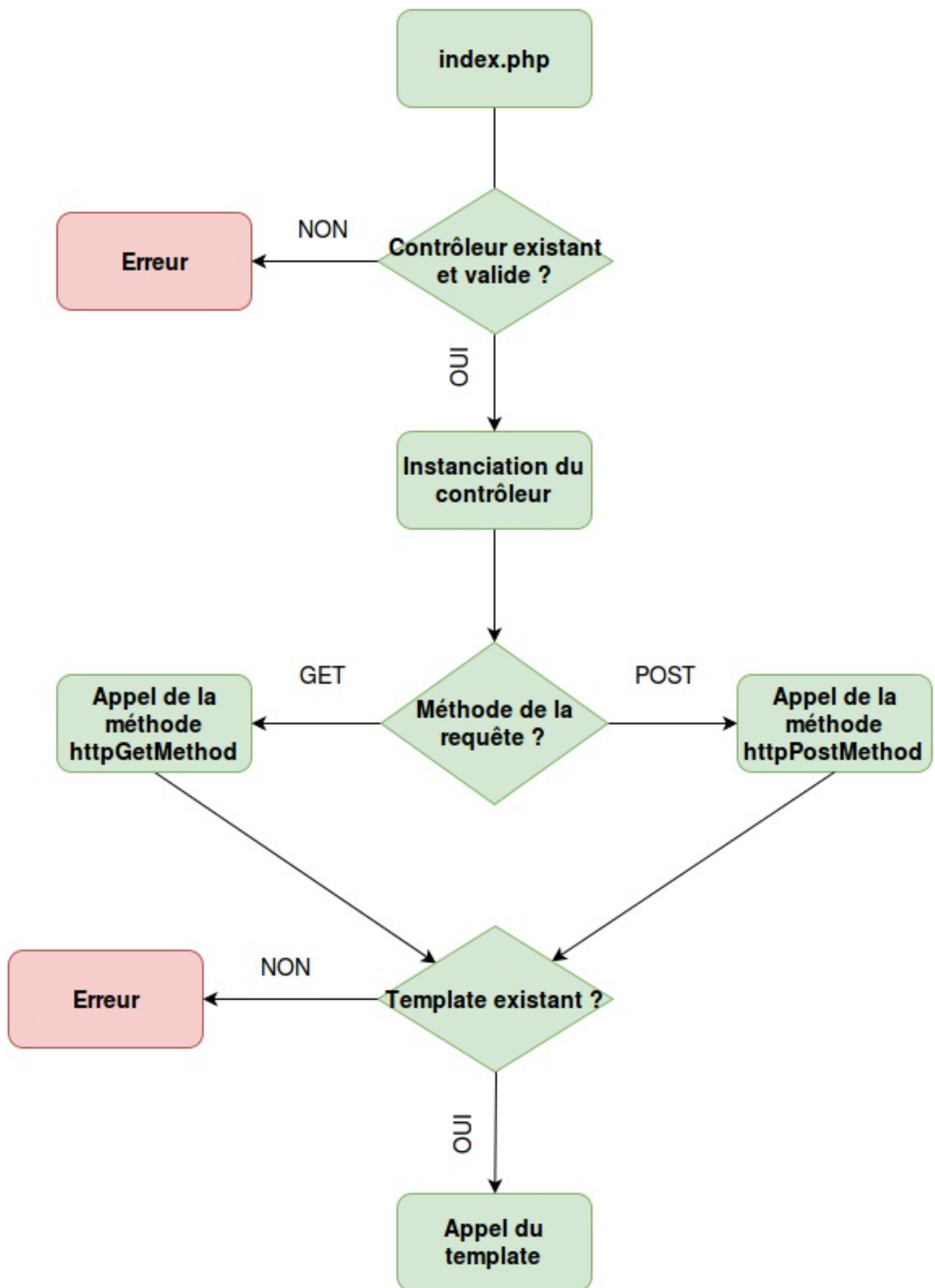
Lorsque vous essayez d'accéder à une page de l'application, le micro-kernel (le moteur du framework) va analyser l'url après index.php et appeler un contrôleur et un template en fonction de cette url. Cette url correspondra au chemin du contrôleur dans le dossier controllers et de la vue dans le dossier www. Le nom de ce contrôleur et de cette vue correspondra à la dernière partie de l'url.

Par exemple si vous visitez l'url <http://www.mon-site.com/index.php/user/login>, le micro-kernel essaiera :

- d'instancier une classe nommée LoginController qui se trouve dans le fichier LoginController.php situé dans le dossier controllers/user/login

- de charger un template nommé LoginView.phtml situé dans le dossier `www/user/login`

Cycle de vie d'une requête HTTP



Création d'une nouvelle page

Création d'une nouvelle page lorsque l'on ira sur l'URL

<http://www.mon-site.com/index.php/dossier/sousdossier> :

- Créer un fichier dans controllers/dossier/sousdossier qui s'appelle SousdossierController.php
- Créer les fonctions suivantes :

```
public function httpGetMethod(Http $http, array $queryFields)
{
    // Code appelé lorsque vous êtes en GET
}

public function httpPostMethod(Http $http, array $formFields)
{
    // Code appelé lorsque vous êtes en POST
}
```

- Créer un fichier dans www/dossier/sousdossier qui s'appelle SousdossierView.phtml

Exemple

Affichage d'une vue à l'url <http://www.mon-site.com/index.php/user/register> :

- Créer un fichier dans controllers/user/register qui s'appelle RegisterController.php
- Créer les fonctions suivantes :

```
public function httpGetMethod(Http $http, array $queryFields)
{
    // Code appelé lorsque vous affichez le formulaire d'inscription
}

public function httpPostMethod(Http $http, array $formFields)
{
    // Code appelé lorsque vous soumettez le formulaire d'inscription
}
```

- Créer un fichier dans www/user/register qui s'appelle RegisterView.phtml

Récupérer des informations dans la vue depuis le contrôleur

Dans le contrôleur on retourne des informations dans un tableau. Les clés de ce tableau deviendront des variables dans la vue

- Dans le fichier

controllers/dossier/sousdossier/SousdossierController.php on retourne un tableau

```
class SousdossierController
{
    public function httpGetMethod(Http $http, array $queryFields)
    {
        return [
            'hello' => 'Hello world' //Création d'une variable $hello dans la vue SousdossierView.phtml
        ];
    }
}
```

- Dans le fichier `www/dossier/sousdossier/SousdossierView.phtml` on affiche la variable utilisée

```
<p><?= $hello ?></p>
```

Variables partagées par toutes les vues

Il existe deux variables partagées par toutes les vues : `$wwwUrl` et `$requestUrl`. Elles sont utilisables dans toutes les vues (mais uniquement les vues).

- `$wwwUrl` : récupère l'url complète vers le dossier `www`

- `$requestUrl` : récupère l'url complète vers le fichier `index.php`

Récupérer des paramètres GET ou POST dans le contrôleur

Les données en GET sont récupérées à l'aide du 2ème paramètre `$queryFields` de la fonction `httpGetMethod`. Les données en POST sont récupérées à l'aide du 2ème paramètre `$formFields` de la fonction `httpPostMethod`.

Par exemple pour récupérer la valeur du paramètre `name` de l'URL `http://mon_site/index.php/dossier/sousdossier?name=John`

```
class SousdossierController
{
    public function httpGetMethod(Http $http, array $queryFields)
    {
        // $queryFields contient les données du tableau $_GET

        $name = $queryFields['name'];
        return [
            'name' => $name //Création d'une variable $name dans la vue SousdossierView.phtml
        ];
    }
}
```


- Dans le fichier

www/dossier/sousdossier/SousdossierView.phtml on affiche la variable utilisée

```
<p>Hello <?= $name ?></p>  
<!-- Affiche Hello John -->
```

Afficher un template sans le layout

Si vous souhaitez uniquement afficher le template sans le layout (pratique pour les appels AJAX), il suffit de retourner la variable `_raw_template` avec la valeur `true` dans le tableau de la fonction `httpGetMethod` ou `httpPostMethod` (en fonction de la méthode http).

Exemple

```
class RegisterController  
{  
    public function httpGetMethod(Http $http, array $queryFields)  
    {  
        return [  
            '_raw_template' => true //Seul le template RegisterView.phtml sera affiché  
        ];  
    }  
}
```

Gestion des formulaires

Afficher les anciennes valeurs du formulaire

Dans le dossier forms, créer une classe qui hérite de la classe Form (qui se trouve dans library/Form.php) et redéfinir la méthode build(). Indiquer dans celle-ci les champs pour lesquelles on veut afficher les anciennes valeurs. Appeler la méthode bind avec les valeurs du formulaire. Enfin retourner le formulaire dans la vue :

```
return [  
    '_form' => new CustomForm() //La variable doit s'appeler _form  
]
```

Exemple

Création de la classe RegisterForm dans le fichier forms/RegisterForm.php

```
class RegisterForm extends Form  
{  
    public function build()  
    {  
        $this->addFormField('lastName');  
        $this->addFormField('firstName');
```

```
$this->addFormField('address');  
$this->addFormField('city');  
$this->addFormField('zipCode');  
$this->addFormField('phone');  
$this->addFormField('email');  
  
}  
}
```

Puis instancier la classe RegisterForm dans le fichier
RegisterController.php

```
public function httpPostMethod(Http $http, array $formFields)  
{  
    //On instancie la classe RegisterForm  
    $form = new RegisterForm();  
    //Et on lie les valeurs du formulaire avec les champs  
    paramétrés dans la classe  
    $form->bind($formFields);  
  
    //On retourne le formulaire à la vue  
    return [  
        '_form' => $form  
    ];  
}
```

Enfin dans la vue RegisterView.phtml on affiche les anciennes valeurs

du formulaire

```
<li>
  <label for="lastName">Nom :</label>
  <input id="lastName" type="text" name="lastName" value=
"<?= $lastName ?>">
</li>
<li>
  <label for="firstName">Prénom :</label>
  <input id="firstName" type="text" name="firstName" valu
e="<?= $firstName ?>">
</li>
<li>
  <label class="textarea" for="address">Adresse :</label>
  <textarea id="address" name="address" rows="3"><?= $add
ress ?></textarea>
</li>
<li>
  <label for="city">Ville :</label>
  <input id="city" type="text" name="city" value="<?= $ci
ty ?>">
</li>
<li>
  <label for="zipCode">Code Postal :</label>
  <input id="zipCode" class="zip-code" type="text" maxlen
gth="5" name="zipCode" value="<?= $zipCode ?>">
</li>
<li>
```

```
<label for="phone">Téléphone :</label>
<input id="phone" class="phone" type="text" maxlength="
10" name="phone" value="<?= $phone ?>">
</li>
```

Requête et réponse HTTP

Utilisation de la classe Http dans library/Http.php

Redirection

```
public function httpPostMethod(Http $http, array $formF
ields)
{
    $http->redirectTo('/'); //Redirige vers la page d'a
ccueil
    $http->redirectTo('/user/login'); //Redirige vers l
a page de connexion
}
```

Envoyer une réponse au format JSON

La méthode sendJsonResponse permet de renvoyer un résultat au format JSON.

```
public function httpGetMethod(Http $http, array $queryF
ields)
```

```
{  
    $model = new PostModel();  
    $posts = $model->findAll();  
    //Envoie les données au format JSON au lieu de renv  
oyer du HTML  
    $http->sendJsonResponse($posts);  
}
```

Enregistrer un fichier envoyé depuis un formulaire

La méthode `moveUploadedFile` permet de sauvegarder dans un fichier dont le nom est passé en paramètre un fichier envoyé depuis un formulaire.

```
public function httpPostMethod(Http $http, array $formF  
ields)  
{  
    //Sauvegarde le fichier envoyé depuis le formulaire  
dans le fichier /images/avatar.png  
    $http->moveUploadedFile('avatar.png', '/images');  
}
```

Récupérer des informations sur la requête

- La méthode `getRequestPath` permet de récupérer la partie d'url située après `index.php`
- La méthode `getRequestMethod` permet de récupérer la méthode utilisée (GET ou POST)
- La Méthode `getRequestFile` permet de récupérer la dernière partie de l'url

Session flash (ou FlashBag)

Session temporaire : permet de stocker des informations et de les effacer tout de suite après les avoir affichées.

Création d'un FlashBag

Création d'un message flash dans un contrôleur

```
$flashBag = new FlashBag();  
$flashBag->add('Connexion réussie !');  
  
return [  
    'flashBag' => $flashBag //N'oubliez pas de le passer  
    à la vue !  
];
```

Afficher un message flash

```
<?php if($flashBag->hasMessages()): ?>
```

```
<p><?= $flashBag->fetchMessage() ?></p>
<?php endif; ?>
```

Afficher tous les messages flash

```
<?php if($flashBag->hasMessages()): ?>
    <?php foreach($flashBag->fetchMessages() as $message)
    : ?>
        <p><?= $message ?></p>
    <?php endforeach ?>
<?php endif; ?>
```

Intercepting Filter

Code qui sera appelé avant le chargement des contrôleurs et des vues. Vous pouvez les utiliser pour créer des variables disponibles dans toutes vos vues. Ou les utiliser pour restreindre l'accès à certaines parties de votre site.

Création d'une classe Intercepting Filter

Créez une classe dans le dossier classes/ qui implémente l'interface InterceptingFilter (library/InterceptingFilter.php). Redéfinissez la méthode run.

Exemple

Ajout d'un flashBag disponible dans toutes les vues :

```
//La classe doit se terminer par Filter !!
class FlashBagFilter implements InterceptingFilter
{
    //Redéfinition de la fonction run
    public function run(Http $http, array $queryFields, array $formFields)
    {
        //Création d'une variable $flashBag dans toutes les vues
        return [
            'flashBag' => new FlashBag()
        ];
    }
}
```

Ajouter ensuite ce nouveau filtre sur votre site en l'ajoutant dans la configuration, dans le fichier config/library.php :

```
// List of all the intercepting filters classes.
$config['intercepting-filters'] = ['FlashBag']; //Nom de la classe sans Filter
```

Database

Classe qui permet de gérer plus facilement la connexion et les requêtes vers la base de données

Configuration

Il faut en premier lieu configurer les informations de connexion dans le fichier database.php du dossier config

```
$config['dsn']      = 'mysql:host=localhost;dbname=nom_de_la_bdd';  
$config['user']     = 'root';  
$config['password'] = 'troisva';
```

Ceci fait, lors de l'instanciation d'un objet Database, la connexion à la base de données se fera avec ces paramètres

```
$db = new Database(); //Connexion à la base de données
```

Récupérer un tableau de résultats

La méthode query permet de récupérer un tableau de résultat

```
$db = new Database();  
$results = $db->query("SELECT * FROM nom_de_la_table");
```

Il est possible de passer un 2ème argument à cette méthode dans le cas où l'on souhaiterait exécuter la requête avec certains paramètres

```
$db = new Database();  
$results = $db->query('SELECT * FROM nom_de_la_table  
                        WHERE nom_du_champ = ?', ['valeur  
_du_champ']);
```

Récupérer un unique résultat

La méthode `queryOne` permet de récupérer un unique résultat de la base de données. Elle prend un 2ème paramètre facultatif correspondant à un tableau de paramètres passés lors de l'exécution de la requête préparée

```
$db = new Database();  
$result = $db->queryOne('SELECT * FROM nom_de_la_table  
                        WHERE id = ?', ['valeur_id']);
```

Insertions, modifications ou suppressions

Pour ajouter, modifier ou supprimer des informations dans la base de données, il faut utiliser la méthode `executeSql`. Elle s'utilise comme `query` et `queryOne`.

```
$db = new Database();
```

```
$db->executeSql('DELETE FROM nom_de_la_table WHERE id = ?  
, ['valeur_id']);
```