Framework 3WA

Présentation

Le framework 3WA est un micro-framework PHP basé sur une architecture MVC (Modèle Vue Contrôleur). C'est un framework très léger et simple à utiliser dans un but pédagogique. Il ne sera pas adapté à la réalisation d'un site en production bien que cela soit possible.

Lorsqu'on utilise un framework ou un CMS, ou même lorsqu'on crée une application "from scratch", toutes les requêtes http vers le site seront dirigées vers le fichier index.php. Le code PHP va ensuite orienter la requête vers tel ou tel contrôleur, en fonction de l'url demandée, c'est ce qu'on appelle le routing.

Le routing

Le routing ou routage en français est l'opération qui permet au framework de faire le lien entre une "route" et une "action". La route est la fin de l'URL, ce qu'il y aura après "index.php". Par exemple, dans l'url suivante :
www.mon-site.com/index.php/contact
la route est "/contact". La route de la page d'accueil d'un site sera donc "/".
Le routing est donc l'opération qui va permettre au framework de savoir quelle action effectuer en fonction de la route demandée. Cette action sera généralement l'exécution d'une méthode d'une classe de contrôleur. Par exemple la route
www.mon-site.com/index.php/contact
déclenchera l'exécution de la méthode *showContactForm* de la classe **ContactController**.

Le routing est en général manuel : c'est le développeur qui indique au framework quelle action relier à quelle route.

Framework 3WA : Routing automatique

Le routing avec le framework 3WA est automatique : il va associer automatiquement une classe de contrôleur et une méthode à une route.

Exemple :

La route "/**contact**" déclenchera automatiquement l'appel à la classe **Contact**Controller. Cette classe devra nécessairement se trouver dans un fichier nommé **Contact**Controller.class.php, lui-même devra se trouver dans le dossier ***application/controllers/contact/***.

Chaque niveau supplémentaire dans la route nécessitera un niveau de dossier supplémentaire. Par exemple pour la route "**/blog/posts**" le framework 3WA ira chercher la classe **Posts**Controller dans le fichier **Posts**Controller.class.php dans le dossier ***application/controllers/blog/posts/***.

Méthode appelée

Après avoir instancier la classe de contrôleur adéquate, le framework exécutera l'une des 2 méthodes suivantes :

- **httpGetMethod**() si la méthode de la requête HTTP est **GET**
- **httpPostMethod**() si la méthode de la requête HTTP est **POST**

L'une au moins de ces méthodes devra donc être définie dans les classes de contrôleurs. Chacune pourra le cas échéant recevoir en paramètres 2 choses :

1. Un objet de la classe Http, qui permettra par exemple de faire des redirections
2. Un tableau contenant l'équivalent de $_GET dans httpGetMethod() et $_POST dans httpPostMethod()

Cas particulier : la page d'accueil

La route de la page d'accueil est "*/*". C'est le contrôleur **HomeController**, situé dans le fichier **HomeController.class.php** à la racine du dossier **application/controllers/** qui sera appelé. Ce fichier existe déjà lors du téléchargement du framework.

Système de templating

Le système de template ou de vues est basé sur PHP. Dans d'autres framework il sera possible d'utiliser un autre langage dans la code HTML des vues que le PHP.
Le fichier de template sera inclus automatiquement par le framework, il n'y aura pas d'inclusion manuelle. Le framework se basera comme pour les contrôleurs sur la route pour trouver le fichier de vue correspondant.

Exemples :

Pour la route "/**contact**", le framework inclura le fichier "**Contact**View.phtml" qui devra être situé dans le dossier application/www/**contact**/**Contact**View.phtml

Pour la route "/**blog/posts**", le framework inclura le fichier "**Posts**View.phtml" qui devra être situé dans le dossier application/www/**blog**/**Posts**View.phtml

Cas particulier : la page d'accueil

Le fichier de template responsable de l'affichage de la page d'accueil est **HomeView.phtml**, déjà existant lors du téléchargement du framework dans le dossier **application/www/**.

Transmission de données du contrôleur à la vue

Pour transmettre des données du contrôleur à la vue, celui-ci devra retourner un tableau associatif dont les clés deviendront des variables dans le fichier de vue.

Par exemple si la méthode **httpGetMethod**() de la classe **HomeController** retourne le tableau associatif suivant :

```
return [
   'message' => 'Ceci est un message pour la page ACCUEIL'
];
```

Dans le fichier **HomeView.phtml** il existera une variable **$message** qui contiendra la chaîne de caractères *'Ceci est un message pour la page ACCUEIL'*

Le tableau associatif retourné peut contenir autant de clés que nécessaire, les valeurs associées peuvent être de toutes natures (valeurs scalaires, tableaux, objets).

Remarque :
Il existe 2 clés spéciales dans le tableau associatif retourné par le contrôleur :

- **_form** : permet de transmettre un objet de la classe Form
- **_raw_template** : permet de ne pas inclure le template global

Le template global

Le template global qui contient le code HTML commun à toutes les pages du site (header, footer) est le fichier **LayoutView.phtml** qui se trouve dès le téléchargement du framework dans le dossier application/www. Ce fichier contiendra par exemple le menu de navigation du site, etc.

Variables prédéfinies dans le template global

Dans le fichier de template global **LayoutView.phtml** le framework met à disposition 2 variables prédéfinies :

- **$wwwUrl** : elle contient le chemin local vers le dossier www (application/www) Elle sera utilisée par exemple pour écrire les urls des fichiers de style, fichiers javascript, images, etc.
- **$requestUrl** : elle contient l'url des pages jusqu'à l'index.php, c'est-à-dire par exemple : localhost/[dossiers locaux]/restaurant/index.php
  Cette variable permettra de construire les urls des liens

Transmission de données au template global

Créer une classe de filtre

Pour transmettre des données au template global, c'est-à-dire au fichier **LayoutView.phtml**, le framework propose un système de **filtres**. Un filtre est une classe qui implémente l'interface InterceptingFilter. La méthode run() retournera un tableau associatif dont les clés seront transmises au template global et transformées en variables.

La classe de filtre devra être suffixée par le mot **Filter**.
Exemple : **AnExampleOfFilter**

Cette classe devra se trouver dans le fichier **AnExampleOfFilter.class.php** dans le dossier application/class.

Enregistrer le filtre

Il est nécessaire d'enregistrer le filtre dans le fichier de configuration application/config/**library.php**

Le nom du filtre doit être ajouté au tableau de configuration **$config['intercepting-filters']**. Par exemple pour le filtre **AnExampleOfFilter**, il faut ajouter au tableau le nom **AnExampleOf :**

```
$config['intercepting-filters'] = [ 'AnExampleOf'];
```

Transmission de formulaires au template

Il peut être intéressant de transmettre les données concernant un formulaire au template qui affiche ce formulaire, pour par exemple :

- Pré-remplir les champs du formulaire suite à une erreur de saisie de l'internaute (cela lui évite d'avoir à re-remplir tous les champs)
- Afficher des messages d'erreur concernant le formulaire

Création d'une classe de formulaire

On va créer une classe dans le dossier **forms** qui héritera de la classe **Form** du framework (dossier library) et dont le nom sera suffixé par '**Form**'. Par exemple pour un formulaire de contact, on créera la classe **ContactForm**, dans un fichier **ContactForm.class.php**, lui-même situé dans le dossier ***application/forms***.

La classe **Form** contient la méthode abstraite **build** qu'il faudra donc implémenter dans la classe **ContactForm**. Cette méthode **build** permettra de dire quels sont les champs qui composent le formulaire en appelant la méthode **addFormField** de la classe **Form**.

Transmission d'un objet Form au template

Pour transmettre depuis un contrôleur un objet Form au template du formulaire, on utilisera comme clé dans le tableau retourné par le contrôleur la clé spéciale "**_form**".

Automatiquement on disposera dans le template de variables qui porteront le nom des champs ajoutés à l'objet Form. Si on a ajouté un champ "firstname", on aura une variable $firstname dans le template. Cette variable nous permettra de réécrire les valeurs des champs dans le formulaire.

Ajout d'un message d'erreur à l'objet Form

Il est possible d'associer un message d'erreur à l'objet Form grâce à la méthode **setErrorMessage**. On aura ensuite dans le fichier de template une variable **$errorMessage** contenant le message.