Comparaison Web Statique et Web Dynamique

En quoi est ce que le web dynamique est-il plus intéressant que le web statique ?

Contrairement à un site statique, un site dynamique utilise un langage de programmation dynamique (en l'occurrence java), et utilise en général une base de données pour stocker les ressources qui constitueront par la suite ses pages HTML.

Un site web dynamique est surtout un site Internet dont les pages sont créées "dynamiquement" en fonction des requêtes des internautes.

Comparé au web statique, le web dynamique (même si plus long pour charger les pages) :

- Est plus rapide à mettre à jour.
- Comporte des fonctionnalités plus faciles à mettre en oeuvre qu'en statique.
- Permet la facilité de transformation de la charte graphique.
- Impose une présentation de contenu normalisée.
- N'a pas de difficulté à gérer un grand nombre de clients en même temps.
- Beaucoup plus pratique et moins cher à maintenir, même si le cout de développement initial est plus important.
- Utilise des bases de données.

En quoi le web statique est-il plus intéressant que le web dynamique ?

Le web statique présente comme avantage d'être beaucoup plus léger pour le serveur. En effet, contrairement au web dynamique, le serveur n'a ici qu'à répondre aux requêtes HTTP effectuées par le web browser du client (ici dans la partie JavaScript).

On pourra également penser à la facilité de programmation du web statique. Cette facilité est cependant nuancée sur le long terme, car comme souligné plus haut, le web dynamique est plus facile à mettre à jour.

En quoi nos prototypes respectent le modèle MVC ?

MVC (Model-View-Controller ou Modèle-Vue-Contrôleur) est un modèle dans la conception de logiciels. Il met l'accent sur la séparation entre la logique métier et l'affichage du logiciel. Cette « séparation des préoccupations » permet une meilleure répartition du travail et une maintenance améliorée.

1. Model (modèle) : gère les données et la logique métier.

2. View (vue) : gère la disposition et l'affichage.

3. Controller (contrôleur) : achemine les commandes des parties "model" et "view".

Dans le premier atelier réalisé au sein de ce module, l'implémentation du modèle MVC a bien été mise en évidence grâce à la création de différents paquets : le **com.sp.controller**, le **com.sp.model** ainsi que les répertoires **src/main/resources/templates** et **src/main/resources/static**

Le controller HTTP (controller qui peut intercepter les requêtes HTTP) avait pour objectif la définition des routes (endpoint) et par suite l'affichage des pages dynamiquement complétées (templating). Le cycle de vie de ce dernier est géré par SpringBoot.

- La définition des routes se fait dans le fichier **RequestCrt.java**

```
...
 @RequestMapping(value = { "/", "/index" }, method = RequestMethod.GET)
 public String index(Model model) {
        model.addAttribute("messageLocal", messageLocal);
        return "index";
 }
...
```

Dans ce cas, une page **index.html** a été créée dans le dossier templates et qui va être appelée par le controller et affichée sur le navigateur.

La création de modèles se fait dans le paquet **com.sp.model**, dans notre cas, on a créé un premier modèle de Poney et son controller **PoneyDao.java** dans le paquet **com.sp.controller** pour qu'il puisse interagir avec le modèle.

Le retour d'une vue affichant un poney est contrôlé par le **RequestCrt.java**

```
    @Autowired
    PoneyDao poneyDao;
  ...
        @RequestMapping(value = { "/view"}, method = RequestMethod.GET)
      public String view(Model model) {
      model.addAttribute("myPoney",poneyDao.getRandomPoney() );
        return "poneyView";
     }
  ...
```

La page **poneyView.html** qui va être appelée lors de l'appel de la route **/view** a ensuite été créée dans le dossier **templates**.

On voit bien la séparation entre les différentes parties, la vue qui est présente dans le répertoire **templates**, les modèles dans le paquet **com.sp.model** et le controller dans le paquet **com.sp.controller**

Dans le cas du prototype réalisé en statique, on respecte moins le modèle MVC :

1. La vue est réalisée par le HTML et le CSS.

2. Le controlleur est représenté par les requêtes HTTP, réalisées par le JavaScript.

3. Le modèle est également réalisé par la réception des données du formulaire et l'affichage des données de la carte par le JavaScript.