Génie logiciel avancé - 2016
TD n°5
Système de planification des vols

La présentation rendue les semaines précédentes doit être complétée par toutes les informations nécessaires à la bonne compréhension du projet par un intervenant externe.

L'objectif de ce TD est de compléter ce livrable et de continuer la mise en place votre environnement de développement pour le dimanche 28 février au plus tard.

Un exemple d'implémentation est disponible sur le projet GitHub suivant :

https://github.com/ktekkal/jetty-jersey

Question 1 – Mise en place IHM

Les interfaces graphiques du projet s'appuieront sur des pages web statiques. Les informations venant du serveur seront récupérées via des appels AJAX sur les web-services vus dans les séances précédentes.

Pour des raisons de sécurité, il faut que les pages web appartiennent au même domaine que les web-services. Le navigateur refusera de réaliser les appels AJAX si ce n'est pas le cas. Plus d'informations sur ce mécanisme « Cross-request » sont disponibles sur la page suivante : https://developer.mozilla.org/fr/docs/HTTP/Access_control_CORS.

Pour ne pas être confronté à ce problème, il faut que vous utilisiez Jetty en tant que serveur de page web. Cela se fait en complément de la fourniture des web-services.

Le code suivant est un exemple de configuration de Jetty afin de réaliser cela :

```
// Add a handler for resources (/*)
ResourceHandler handlerPortal = new ResourceHandler();
handlerPortal.setResourceBase("src/main/webapp");
handlerPortal.setDirectoriesListed(false);
handlerPortal.setWelcomeFiles(new String[] { "home.html" });
ContextHandler handlerPortalCtx = new ContextHandler();
handlerPortalCtx.setContextPath("/");
handlerPortalCtx.setHandler(handlerPortal);
```

Cette configuration permet à Jetty :

- de fournir les fichiers présents dans le répertoire « src/main/webapp »
- de fournir par défaut le fichier « home.html »
- et de mettre ces fichiers à la racine du serveur (paramètre « Context-Path »).

Cette configuration vient en conflit avec la déclaration des web-services vu dans le précédent TD. Il faut donc l'adapter pour mettre un contexte différent (« ws ») qui deviendra le préfixe de tous les appels des web-services :

```
// Add a servlet handler for web services (/ws/*)
ServletHolder servletHolder = new ServletHolder(new ServletContainer(rc));
ServletContextHandler handlerWebServices = new ServletContextHandler(Servl
handlerWebServices.setContextPath("/ws");
handlerWebServices.addServlet(servletHolder, "/*");
```

Question 2 – Appel des web-services

Une fois la configuration modifiée, il est nécessaire de réaliser l'appel des web-services pour charger dynamiquement des informations sur les pages web.

La solution proposée est d'utiliser la fonction « ajax » de la librairie « jQuery »[1]. Voici un exemple d'appel :

```
function getServerData(url, success){
    $.ajax({
        dataType: "json",
        url: url
    }).done(success);
}
```

L'appel est réalisé en précisant le type de donnée utilisé (« json » qui ajoutera automatiquement le bon header « Content-Type » durant l'appel) ainsi que la fonction à appeler lorsque la réponse est récupérée depuis le serveur.

La documentation complète de cette fonction ainsi que des exemples d'utilisation et de gestion des erreurs sont disponibles à l'url suivante :

http://api.jquery.com/jQuery.ajax/

Vous devez implémenter les quatre type d'appel aux web-services via cette fonction (GET, POST, PUT et DELETE).

[1] https://jquery.com/

Question 3 – Génération dynamique de code

Pour générer dynamiquement du code html dans vos pages web, vous pouvez utiliser un moteur de template qui facilitera l'écriture du code HTML.

Il est conseillé d'utiliser la librairie « Underscore »[2], en particulier la fonction « template »[3] pour cela.

Vous devez créer un template dans votre page HTML :

```
<script id="templateExample" type="text/template">
    <div>
        <%= attribute %>
    </div>
</script>
```

Puis le compiler dans votre code Javascript :

```
var templateExample = _.template($('#templateExample').html());
```

Une fois compilé, vous pouvez l'utiliser pour instancier votre template et utiliser le code ainsi généré :

```
var html = templateExample({
    "attribute":JSON.stringify(result)
});

$("#result").append(html);
```

[2] http://underscorejs.org/
[3] http://underscorejs.org/#template