IN328 : Servlets - Corrigé SI

http://personnel.supaero.fr/garion-christophe/IN328

Ce TP va vous permettre de manipuler des servlets en utilisant le *web container* Tomcat.

---

1 Objectifs

Les objectifs de ce TP sont les suivants :
- comprendre la notion de servlet ;
- utiliser le *web container* Tomcat.

2 Mise en place de Tomcat

Nous allons dans un premier temps mettre en place un environnement pour utiliser le serveur Tomcat. Pour cela :
- récupérez le fichier `tomcat.tgz` sur le site et décompressez-le dans votre répertoire de travail. Vous obtenez un répertoire `tomcat` ;
- récupérez le script `tomcat.sh` ;
- rendre le script `tomcat.sh` exécutable avec `chmod u+x tomcat.sh` ;
- positionner la variable d'environnement `CATALINA_BASE` avec `export CATALINA_BASE=CHEMIN_VERS_tomcat`.

Pour lancer Tomcat, on utilisera la commande `./tomcat.sh start`.

Ouvrir alors votre navigateur web à l'adresse `http://localhost:8080`. Vous devez arriver sur la page d'accueil de Tomcat (attention au proxy de SUPAERO, il faut que votre machine ne l'utilise pas). Profitez-en pour regarder les exemples de servlets...

Quelques remarques importantes :
- une fois que Tomcat est démarré, si vous changez vos servlets ou votre fichier de configuration `web.xml`, cela ne sera pas pris en compte. Il faut arrêter et redémarrer Tomcat pour cela.
- e pas oublier d'arrêter Tomcat en fin de TP par `./tomcat.sh stop`.

3 Écrire une première servlet

Dans un premier temps, écrire une servlet `MaServlet`. Cette servlet devra utiliser un paramètre de la requête et afficher la date sur une page HTML.

Vous aurez besoin pour cela d'utiliser l'archive `servlet-api.jar` disponible dans le répertoire `lib` de Tomcat (pour avoir le paquetage `javax.servlet` par exemple).

Créer un fichier de configuration `web.xml` et construire une architecture d'application web (vous pourrez utiliser le squelette de fichier `web.xml` disponible sur le site). Vous placerez ensuite cette architecture dans un répertoire `firstQuestion` (qui sera donc votre « Application Root ») dans le répertoire `webapps` de Tomcat. Vous pourrez vérfier que votre servlet fonctionne en utilisant l'URL `http://localhost:8080/firstQuestion/essai` (ou `essai` est l'URL definie par mapping sur la servlet `MaServlet` dans `web.xml`).

Rien de bien particulier sur cette question. Il suffisait de reprendre ce qui avait présenté en cours.

4 Les menus du RU

Nous allons essayer d'afficher les menus de la semaine du RU dans une page Web. Pour cela, un ensemble de classes représentant les menus d'une semaine a déjà été développé.

On suppose que l'application web que nous allons développée sera placée dans le répertoire `secondQuestion` de `webapps` (ce sera donc l'« Application root » de l'application).

- récupérer l'archive contenant les classes `Menu` et `MenusSemaine` sur le site. On vérifiera que tout fonctionne bien en exécutant `java -jar TP4.jar`. Un menu doit s'afficher dans la console.
- développer une première servlet `AffichageMenus` dont le rôle sera d'afficher les menus de la semaine. sous forme d'un tableau HTML. Un objet de type `MenusSemaine` sera créé à l'initialisation de la servlet et que celui-ci sera accessible via le `ServletContext`. On n'affichera dans un premier temps que les entêtes du tableau et pas encore les menus.

  Le source est disponible sur le site. Dans un premier temps, il fallait bien sûr se demander quels allaient être les éventuels attributs de la classe. Ici, il fallait un objet de type `MenusSemaine` qui allait nous servir de modèle.
  Cet attribut n'est pas initialisé par le constructeur de la classe, mais par la méthode `init`. Pour pouvoir le partager avec d'éventuelles autres servlets de l'application, on demandait à utiliser le `ServletContext`. J'ai donc choisi d'écrire une méthode `init` qui :
  - vérifie si un objet de type `MenusSemaine` est disponible dans le `ServletContext` ;
  - si oui, l'assigne à l'attribut de la classe ;
  - si non, crée un objet de type `MenusSemaine`, l'assigne à l'attribut de la classe et l'enregistre dans le `ServletContext` via un nom.

  La méthode `doGet` était quand à elle simple, puisqu'elle ne concernait que la mise en page du tableau.
- développer une seconde servlet `AffichageMenu` (sans s cette fois-ci) dont le rôle est d'afficher la ligne du tableau pour un jour particulier. Cette servlet sera donc « incluse » dans la première via un `RequestDispatcher`. On modifiera `AffichageMenus` en conséquence.

  Cette servlet devait également manipuler les menus de la semaine sous la forme d'un objet de type `MenusSemaine`. J'ai choisi de la stocker en attribut et d'utiliser la même méthode `init` que pour `AffichageMenus`. On aurait également pu passer l'objet via un attribut de la requête de `doGet`, mais cela est plus lourd.
  La méthode `doGet` récupérait le nom du jour à afficher via un attribut stocké dans l'objet `HttpServletRequest` passé en paramètre. C'est donc à `AffichageMenus` de faire une boucle sur les jours de la semaine et d'utiliser un dispatcher pour appeler `AffichageMenu` avec le jour souhaité à chaque fois.
  On remarquera que `AffichageMenu` ne doit pas positionner de propriétés de la réponse renvoyée, car celle-ci est destinée à être incluse dans la réponse construite par `AffichageMenus`.

On rappelle qu'un tableau HTML a la syntaxe suivante :

```
<table>
<caption>Le titre de la table</caption >
<tr><th>Premier titre</th><th>Second titre</th></td></tr>
<tr><td>val1</td><td>val2</td></td></tr>
<tr><td>val3</td><td>val4</td></td></tr>
</table>
```

On pourra utiliser le fichier html proposé sur le site comme modèle. Plus de renseignements sont disponibles sur la page `http://www.w3.org/TR/html401/struct/tables.html`