TP Servlet

Utilisation des cookies, d'une session, accès à une base de données et utilisation de filtres

Gestion d'une session lors d'une commande dans un magasin

1 Objectifs :

Une application sur le WEB se présente comme une succession de pages, en principe il n'y a aucun lien entre toutes ces pages, le serveur ne connaît le client que le temps d'une requête, or dans diverses applications (comptabilité commerce électronique, gestion de comptes bancaires, etc.) il est nécessaire de factoriser un certain nombre de données propres à chaque client entre ces différentes pages.

Ce TP, par l'utilisation des cookies et des sessions HTTP et d'accès à une base de données, va montrer les différentes possibilités.

2 Test d'une application

Nous allons voir comment créer une application web et une servlet avec éclipse. Le contenu de cette servlet se trouve dans les données de ce TP, sous e- nautia, dans l'archive "*donnees TP Magasin vente disque.zip*" sous le répertoire " Test d'une application".

1) sous le menu "**File**" allez sous "**New**" et choisissez ""**Dynamic Web Project",** donnez un nom à votre projet, par exemple "monprojet", verifier que le "**Target Runtime**" est bien "**Apache Tomcat v9.0**" et cliquez sur "**Finish**".

Eclipse crée une application web "*monprojet*".

2) sélectionnez votre projet sous éclipse et avec le bouton droite de la souris faites apparaitre le menu, faites "**New** " et choisissez "**Servlet**", donnez lui un nom de Classe (Majuscule en premier caractère), par exemple "MaServlet", la superclass est **javax.servlet.http.HttpServlet,** faites "**Finish**".

3) Vous pouvez déjà l'exécuter car la génération d'éclipse mais dans sa méthode doGet :

```
response.getWriter().append("Served at: ").append(request.getContextPath());
```

et rajoute l'annotation

```
@WebServlet("/MaServlet")
```

Sélectionnez votre servlet " MaServlet .java" et avec la souris (bouton droit) choisissez

Run as "Run on Server" (sous Eclipse : window-> Web Browser vous pouvez choisir soit Firefox, si vous l'installez, soit Internet Explorer, par defaut c'est un browser interne à Eclipse)

Un navigateur doit s'ouvrir, avec l'adresse de l'appel " http://localhost:8080/monprojet/MaServlet", la servlet doit s'exécuter.

vous devez avoir en résultat " /monprojet " qui est la racine de votre application

4) Ouvrir cette classe, elle se trouve dans "**Java Resources**", sous "**src**" et faire un copier-coller de la méthode doGet dans celle de la servlet (fichier *contenu maServelt.txt*) :

```
public class MaServlet extends HttpServlet {
        public void doGet (HttpServletRequest request,
                           HttpServletResponse response)    throws ServletException, IOException {
            PrintWriter       out;
            String            title = "Simple Servlet Output";
            response.setContentType("text/html");
            out = response.getWriter();
            out.println("<HTML><HEAD><TITLE>");
```

```
        out.println(title);
        out.println("</TITLE></HEAD><BODY>");
        out.println("<H1>" + title + "</H1>");
        out.println("<P>This is output from SimpleServlet.");
        out.println("</BODY></HTML>");
        out.close();
    } }
```

5) Acceptez les "**import**" qu'Eclipse vous propose pour corriger les erreurs.

6) Enregistrez votre servlet.

7) Relancez votre servlet. La modification doit se voir car il est mis (reloadable="true") dans le contexte de "monprojet" que vous trouvez dans le fichier **server.xml** de l'instance serveur que vous avez sous "Servers" à coté de votre projet.

3 Travail demandé sur les servlets : Magasin vente de disques

Vous trouver six types de données pour ce TP dans "*donnees TP Magasin vente disque.zip*" vous avez "*TP magasin.zip*" :

1. le **paquetage mesCommandes** qui contient toutes les classes java utiles pour ce TP (servlets et filtres),
2. un fichier **index.html** qui permet d'accéder à l'application,
3. un fichier **web.xml** qui décrit une partie des servlets et des filtres de ce TP, attention vous devez le compléter au fur et à mesure de l'avancement de votre TP si nécessaire ou mettre les annotations dans vos servlets.
4. le fichier **magasin.sql** qui donne la structure de la base de données à utiliser,
5. le driver **mysql-connector-java-5.1.26.jar** à mettre dans le répertoire **lib** de WEB-INF de votre application WEB
6. et enfin le répertoire **images** qui contient les pochettes des disques

Sous éclipse, créer une autre application web, par exemple **monMagasin**.

3.1 Installation

Copiez le répertoire "*mesCommandes*" (ou glissez) sous "**src**" de **"Java Resources**"

Dans "**WebContent**" :

- Copiez le fichier "index.html"
- Mettez le répertoire "images"
- sous "WEB-INF",
  - copiez "web.xml"
  - sous "**lib**" copiez " mysql-connector-java-5.1.26.jar "

3.2 Accès au magasin (index.html, FormulairesAcces.java, VerificationCompte.java, EntreeMagasinDisque.java)

Certains alias d'appel de ces servlets sont définis dans le fichier web.xml, les autres c'est à vous de les définir, soit par une annotation soit dans le fichier web.xml. Pour commencer, vous devez définir les alias des servlets **FormulairesAcces.java** (*servlet/formulaire*) et **VerificationCompte.java (***servlet/voirCompte*), ces classes se trouvent dans le paquetage "*mesCommandes*".

3.2.1 Fichier : index.html

Le fichier **index.html** permet au client

- soit de demander sa connexion pour faire ses achats (il doit connaitre un identifiant et un mot de passe),
- soit de s'inscrire pour créer ces informations.

Modifiez ce fichier pour qu'il appelle dans votre **propre application web** la servlet "*FormulairesAcces*" pour **l'inscription ou la connexion,** (paramètre demande = *inscription* ou *connexion*) vérifiez aussi si c'est le port que vous avez choisi pour le serveur, par défaut 8080, ou 8443 si vous continuez en HTTPS.

3.2.2 servlet : FormulairesAcces .java alias : servlet/formulaire

Deux cas sont considérés dans cette servlet, le client vient pour s'inscrire (paramètre demande = *inscription*) ou le client vient pour se connecter (paramètre demande = *connexion*).

1. **Cas 1 : inscription**

La servlet demande les informations suivantes : nom, mot de passe, email, téléphone par un formulaire. En utilisant un bouton "*submit*" de nom "**inscrire**", la servlet **VerificationCompte** est appelée.

Si le paramètre " **erreurInscription** " est présent, il rajoute un message disant que les informations précédemment envoyées par ce formulaire n'étaient pas satisfaisantes (nom et mot de passe obligatoires et de plus de 4 caractères)

2. **Cas 2 : connexion**

Un client pourra passer une commande s'il possède un identifiant (nom) et un mot de passe, c'est-à-dire s'il est déjà passé par l'inscription.

La servlet lui demande par un formulaire son "**nom**" et son "**mot de passe**". Un bouton de type "*submit* " de nom "**connecter**" permet d'appeler la servlet **VerificationCompte**, avec comme paramètres le contenu de ce formulaire.

Si le paramètre " **erreurConnexion** " est présent, la vérification s'est mal passée, ce paramètre permet de savoir si le nom est inconnu ou si le mot de passe est mauvais, un message est écrit en conséquence.

Si le paramètre **inscriptionFaite** est présent, il vient de s'inscrire, on rajoute un message comme quoi l'inscription s'est bien réalisée et dans le formulaire on initialise le nom avec le nom reçu en paramètre.

3.2.3 servlet : VerificationCompte.java alias : servlet/ verifCompte

Cette servlet gère le compte des utilisateurs, ici on utilise tout simplement un cookie de nom celui passé par le formulaire et de valeur le « mot de passe » passé par le formulaire.

Cette servlet gère aussi deux cas :

- **Cas 1: paramètre "inscrire"** présent**.**

Si le nom et le mot de passe reçus en paramètre sont corrects (plus de 4 caractères), un cookie est créé de nom celui passé par le formulaire et de valeur le « mot de passe » passé par le formulaire.

Puis un appel à la servlet "FormulairesAcces" est réalisé pour se connecter avec le paramètre **" inscriptionFaite "** à "true" et le paramètre "**nom**" avec la valeur reçue.

Sinon, un rappel de la servlet "**FormulairesAcces**" pour une autre inscription est faite avec un paramètre "**erreurConnexion**" pour signaler le type d'erreur.

- **Cas 2: paramètre "connecter"** présent**.**

On vérifie si le "**nom**" et le "**mot de passe**" reçus en paramètre correspondent à ceux d'un cookie. Pour cela utilisez la méthode "**Util. rechercheCookies"** que vous devez compléter.

static public String rechercheCookies(Cookie[] lescookies, String nom)

- Si c'est correct, on appelle la servlet d'entrée du magasin " EntreeMagasinDisque " avec comme paramètre le nom.

- Sinon, on rappelle la servlet **FormulairesAcces** pour se connecter, avec un paramètre " **erreurConnexion** " qui indique le pourquoi de l'erreur.

3.2.4 EntreeMagasinDisque.java alias servlet/entreClient

- **EntreeMagasinDisque. java**, permet d'appeler la page **AfficherLesDisques,** auparavant elle crée deux variables de session : « nomClient » qui a pour valeur le nom de l'utilisateur et « stockCourant » qui a pour valeur une instance de la classe Stock, cette classe permet la gestion des disques présents pour la vente, elle contient une liste des disques présents " listeDisques".

3.2.5 Sécurisation du site

Faites en sorte que l'accès aux différentes pages passe par le protocole en HTTPS.

3.2.6 Gestion d'une commande

Les servlets de gestion de la commande de disques sont les suivantes :

- **AfficherLesDisques.java** alias **servlet/afficheDisques**
- **CommanderUnDisque.java** alias **servlet/commandeClient**

Au départ, l'accès à ces classes sera libre, par la suite, nous les ferons précéder par un filtre qui rappellera la servlet "**InscriptionClient**" si l'utilisateur courant n'est pas correctement connu.

Trois autres classe java fournissent des services pour ces servlets : **Stock.java, Disque.java**, **Magasin.java**.

- **AfficherLesDisques. java**, cette servlet propose, une liste de disques où le client peut choisir celui qu'il va acheter, elle fournit, entre autre, pour chaque disque son titre et son prix, le chanteur ainsi que la proposition de l'ajouter dans son caddy pour l'achat.
  - La liste des disques et leur prix, pourraient être recherchés dans une base de données, pour l'instant, nous allons tout simplement utiliser la classe **Stock.**
- **Stock.java,** cette classe contient un tableau des disques en vente, Elle offre les méthodes
  - "`getListeDisques`" qui donne la liste des références des disques,
  - "`trouveDisque`" qui à partir d'une référence donne les caractéristiques de ce disque.
- **Disque.java**, cette classe décrit les caractéristiques d'un disque.
- **Magasin.java,** cette classe définit une structure de données qui à partir d'un nom permet de donner le caddy d'un client. Cette structure de données est une « **TreeMap** » avec comme clé le nom du client (String) et comme contenu les listes de références de disques (String) commandés par chaque client, c'est une « **ArrayList<String**>, qui représente le caddy d'un client ». La TreeMap est une variable de classe, " **lesCaddy** ", (static en java).

  Cette classe fournit aussi différentes méthodes d'affichages.
  - `afficherContenuCaddy` affiche un tableau HTML des disques contenu dans un caddy.
  - `afficherDisquesEnVente` affiche un tableau HTML des disques du magasin, et donne la possibilité de les mettre dans un caddy par un lien html.
  - `afficherDisquesDansBase` affiche un tableau HTML des caractéristiques des disques, à partir d'un « **ResultSet** » (ensemble des disques issus d'une requête d'une base de données).

Noter l'utilisation de la méthode **"request.getContextPath()"** qui permet dans cette classe de connaitre le nom que vous avez donné à votre application (répertoire), et donc de retrouver son contexte, lors des appels, par exemple à partir de AfficherLesDisques.
Par la suite il serait possible de remplacer la classe « **Stock.java** » par un accès à une base de données mais ce travail ne vous est pas demandé.

- **CommanderUnDisque.java**, cette servlet, rajoute le disque dont le code est passé en paramètre dans le caddy du client, puis affiche les différents disques en cours de commande dans son caddy**.**
  - **Si le client n'a pas de caddy, il est créé**
  - Puis elle propose :

- de commander un autre disque par un rappel à la servlet **AfficherLesDisques**
- d'enregistrer le contenu du panier dans une base de données par un appel à la servlet **EnregistrerCommande.**

3.3 Enregistrement du caddy dans une base de données EnregistrerCommande.class alias servlet/enregistreCommande

Pour utiliser cette servlet, vous devez avoir accès à une base de donnée "magasin". Pour cela, installez sur votre machine un serveur de base de donnée, si vous n'en n'avez pas (voir le paragraphe **3.6 Remarque** qui suit).

- **EnregistrerCommande.java**, cette servlet,
  - ajoute le client dans la base de données, s'il n'est pas connu,
  - donne le contenu de son panier, *à vous d'écrire cette action*,
  - enregistre le contenu du panier dans une base de données, par la méthode "**AjouteCaddyBase**" *que vous devez compléter*. Pour plus de clarté, utilisez une "PreparedStatement" (JDBC).
  - Affiche l'ensemble des disques que le client a commandé dans la base de données, par la méthode " **MontreCommandeBase**" *que vous devez compléter.* Pour plus de clarté, utilisez une "PreparedStatement" (JDBC).
  - S'il continue ses achats, dans un premier temps son panier n'est pas vidé, le contenu du panier peut donc être enregistré plusieurs fois. **Ce problème sera réglé par la suite par un filtre.**

3.4 Filtres sur l'accès aux pages

1) Lorsque toutes les servlets fonctionnent, vous faites précéder l'exécution de certaines par un filtre qui vérifie si le client est bien inscrit (cookies présents), et que ces pages peuvent connaitre les disques en ventes (variable de session « leStock » présente). Si ce n'est pas le cas ce filtre renvoie à la servlet d'inscription, autrement on continue sur la servlet en appel.

Servlet concernées : toutes sauf (FormulairesAcces.java, VerificationCompte.java, EntreeMagasinDisque.java)

Les filtres sont déclarés dans web.xml, ainsi que les servlets sur lesquels ils agissent.

- **FiltreEntree.java** nom : **verifieClient**

Pour différentier ces classes sur lesquelles le filtre s'exécute, vous leur donnez une "url-pattern" différentes pour l'appel. C'est dans "EntreeMagasinDisque.java", que ce nouveau contexte sera pris en compte. Attention, vous devrez sûrement modifier la création de vos cookies qui doivent alors être connus dans tous le site et non à une url précise.

2) Ecrire le filtre **FiltreSortie.java** qui vide le Panier après l'exécution de la servlet EnregistrerCommande

- **FiltreSortie.java** nom : **effaceCommande**

3.5 Création de la facture

Rajoutez dans les menus la possibilité de demander une facture.

Ce menu doit faire appel à une nouvelle servlet à écrire "Facturation.java" qui présente un tableau avec les disques commandés dans la base de données par l'utilisateur, avec le prix pour chaque disque et le prix total de la commande.

Par la suite, l'utilisateur ne doit pas pouvoir continuer sa commande sans se reconnecter.

3.6 Remarque

Ce TP utilise une base de données, ce sera une base **Mysql,** le code serait le même pour une autre base à part le choix du driver.

Vous devez donc installer un serveur de base de données, dans les logiciel du TP précédent, vous avez xampp (*xampp-win32-7.2.11-0-VC15-installer.exe*) que vous pouvez utilisez, il suffit de cliquer sur l'exécutable pour l'installer.

Pour lancer cette base, vous devez le faire par "Xamp Control Panel", puis dans la fenêtre qui s'ouvre lancer Apache(le serveur) et Mysql (le gestionnaire de base). Puis vous pouvez cliquez sur admin de Mysql, qui ouvre sous votre navigateur "**phpmyadmin**" qui gère les bases.

La base s'appelle "magasin" (a créer), elle contient deux tables : client et commande

Vous trouverez le fichier **"magasin.sql"** qui décrit les tables dans les données de ce TP, à importer dans la base.

La table "client" possède deux attributs une clé "id" qui est un entier et le "nom" qui est une chaîne de caractères.

La table "commande" possède trois attributs une clé "id" qui est un entier et le "nomarticle" qui est une chaîne de caractères et une référence " "client" sur une clé de la table "personnel.

3.7 Accès à la base de données

Cet accès est possible si la librairie gérant le driver est connue de l'application WEB. Sous le répertoire WEB-INF de votre application vous rajoutez donc un répertoire **lib** ce driver :

mysql-connector-java-5.1.26.jar, pour accéder à une base de données mysql. Vous le trouvez sous les données.

3.8 Accès à l'information sur les servlets

Servlets

http://docs.oracle.com/javaee/7/index.html

http://java.developpez.com/cours/servlets/

API servlets

http://tomcat.apache.org/tomcat-8.0-doc/servletapi/

HTML

http://perso.wanadoo.fr/chatinais/index.htm

JDBC

http://docs.oracle.com/cd/B28359_01/java.111/b31224.pdf

http://docs.oracle.com/cd/B28359_01/java.111/b31224/toc.htm

http://www.jmdoudoux.fr/java/dej/chap-jdbc.htm

API java

http://docs.oracle.com/javase/8/docs/api/

Et n'oublier pas les moteurs de recherche

4 Compte rendu

Vous devez rendre l'ensemble des classes (l'application web entière) qui constituent ce TP, dans une archive

nom1ET**nom2**Servlet.zip

nom1 et nom2 sont les deux noms d'un binôme.

Le placer sous l'espace habituel réservé à la réception des TP de ce module,

et me l'envoyer par mail à l'adresse **ogor.robert@orange.fr**

Vous pouvez commenter votre code.

Date limite de remise du TP **Lundi 4 février 2019**