Première partie : des servlets simples.

Vous pouvez consulter le cours suivant à partir de la page 16.

http://www.mcours.net/cours/pdf/info/java_cote_serveur_servelet_et_jsp_2.pdf

1 – Ecrire une première servlet qui se contente d'afficher un message simple de salutation sur le navigateur. On accède à la servlet en utilisant la méthode GET.

2 - Utiliser ensuite une page HTML proposant un lien qui accèdera à la servlet.

3 - Faire fonctionner une application cliente Java (client lourd)
Pour cela, on déclare une java.net.URL, construite avec l'URL de la servlet, puis on ouvre une URLConnection avec la méthode openConnection() et enfin on accède au flux d'entrée (de la classe InputStream) de cette connexion avec getInputStream().
Rappel sur les flux en Java : on peut lire avec la méthode readLine() sur un BufferedReader construit avec un InputStreamReader lui-même construit avec un InputStream.

4 - Faire passer des paramètres à la méthode GET et les utiliser dans la servlet : par exemple vous pouvez saluer la personne qui vous transmettra en paramètre son nom.

5 - Utiliser ensuite la méthode POST :
Vous ajouterez côté serveur un formulaire html qui demande le nom de l'utilisateur. Après validation, une servlet est appelée qui affiche un message de bienvenue personnalisé.

6 - Faire afficher le nom de la servlet calculée automatiquement. (getServletName)

7 - Si l'utilisateur n'a pas saisi de nom, utiliser un nom par défaut (getInitParameter) issu d'un fichier de paramètrage (fichier web.xml : balise init-param qui contient les 2 balises param-name et param-value)

8 – Ici, il est demandé d'écrire 2 classes servlets différentes dans la même application.
La première servlet propose une page qui enregistre le nom par défaut, à faire saisir par l'utilisateur. Pour cela, on accèdera au contexte de la servlet (getServletContext) auquel on ajoutera un attribut (setAttribute).
La seconde utilise le même contexte (getServletContext) pour retrouver cet attribut (getAttribute).

Seconde partie : maintenir l'état des clients - Utilisation des cookies

Vous pouvez consulter le cours suivant à partir de la page 27.

http://www.mcours.net/cours/pdf/info/java_cote_serveur_servelet_et_jsp_2.pdf

1 – Ecrire une première servlet qui affiche un formulaire demandant le nom d'un utilisateur, et qui enchaîne sur une deuxième qui répète le nom ou qui affiche une valeur pas défaut si rien n'a été soumis. Cette deuxième page propose de revenir au début.

2 – Ajouter à cette seconde page un lien qui tente de réafficher la page courante.
Corriger en utilisant un champ hidden.

3 – Modifier la seconde servlet pour qu'elle enregistre un cookie du nom si une information a été soumise et qu'elle supprime le cookie sinon.
Modifier la première servlet pour qu'elle lise un cookie si il existe.
Ajouter une troisième servlet, suite de la seconde qui continue à dire qui on est, ou une valeur par défaut quelconque si aucune information n'a été initialement fournie.

Troisième partie : maintenir l'état des clients - Travail avec les sessions

Vous pouvez consulter le cours suivant à partir de la page 32.

http://www.mcours.net/cours/pdf/info/java_cote_serveur_servelet_et_jsp_2.pdf

La documentation Java est accessible à la page http://docs.oracle.com/javaee/7/api/

1 – Ecrire une servlet de connexion qui affiche un formulaire demandant le nom d'un utilisateur. On enchaîne sur un formulaire de saisie d'informations propre à cet utilisateur (par exemple age et code postal). Sur validation, on enchaîne sur une page de visualisation des informations saisies, et qui propose soit de modifier ces informations, soit de terminer la session en cours, soit de continuer sur d'autres pages (dans la même session). Toutes les pages accessibles à partir de cette page de visualisation proposent de revenir voir les informations.
Utilisation de HttpSession, getSession (avec ou sans paramètre) de HttpRequest, invalidate de HttpSession, set et getAttribute de HttpSession

2 – Configurer votre navigateur pour qu'il n'accepte pas les cookies et réutiliser les servlets précédentes. Corriger en utilisant l'encodage d'URL.
Utilisation de encodeURL sur HttpServletResponse.

3 – Modifier votre application de façon à obliger à passer par la page d'authentification
Utilisation de sendRedirect sur HttpServletResponse ou de forward sur un RequestDispatcher obtenu à partir du contexte ou de la requête.