Second tutorial JSP

Pour ce deuxième tutorial, nous allons montrer l'intégration de JSP avec les javabeans comme modèle. Pour cela, nous repartons du tutorial JSP 1. L'objectif est de transformer les scripts précèdent pour faire un mini annuaire non persistant.

Etape 1 : Création des java beans
1.a Créer un java bean (AnnuaireBean) dans un package beans.
Ce java bean va permettre d'enregistrer une liste de Personnes dans l'annuaire. Il doit permettre d'ajouter une personne et de rechercher une personne par son nom. Dans un premier temps, on considère qu'il n'y a pas 2 personne de même nom enregistrer dans l'annuaire.

Le code du bean est le suivant :

```
package beans;

import java.beans.*;
import java.io.Serializable;
import java.util.Hashtable;

public class AnnuaireBean implements Serializable {


    private Hashtable liste = new  Hashtable();
    private PropertyChangeSupport propertySupport;

    public AnnuaireBean() {
        propertySupport = new PropertyChangeSupport(this);
    }

    public void addPersonne(String nom, String prenom){
        PersonneBean tmp =  new PersonneBean();
        tmp.setNom(nom);
        tmp.setPrenom(prenom);
        liste.put(nom,tmp);
    }

     public PersonneBean findPersonne(String nom){
        return (PersonneBean) liste.get(nom);
    }

    public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);
    }
}
```

1.b. Creation du Bean Personne
Créer le bean PersonneBean dans le package beans.
Ce bean permet des maintenir le modèle associé à une personne particulière.
Le code du bean est le suivant :

```
package beans;

import java.beans.*;
import java.io.Serializable;

public class PersonneBean implements Serializable {


    private String nom;
    private String prenom;

    private PropertyChangeSupport propertySupport;

    public PersonneBean() {
        propertySupport = new PropertyChangeSupport(this);
    }


    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }




    public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);
    }
}
```

Etape 2 : Modification du script jsp form1.jsp
Dans le tuto precedent, form1.jsp fait juste un helloworld. Ici sont comportement consiste à ajouter la personne saisit dans la liste des personnes connues.
Le script est donné page suivante. Noter l'extraction des paramètres et l'utilisation du java bean.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%= "Ajout de l'utilisateur"%>
        <%= request.getParameter("lenom") %>
        <%= request.getParameter("leprenom") %>
        <jsp:useBean id="liste" scope="session" class="beans.AnnuaireBean" />
        <% liste.addPersonne(request.getParameter("lenom"),
request.getParameter("leprenom"));
        %>
    </body>
</html>
```

Etape 3 : Construction d'un formulaire de recherche d'une personne.
Créer un nouveau script jsp (index2.jsp), Ce script est une page html qui contient un formulaire pour rechercher une personne pa son nom.
Le script est le suivant :

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Recherche Hello World!</h1>

            <form name="form2" action="form2.jsp">
                Nom <input type="text" name="lenom" value="" size="25" /><br>
                <input type="submit" value="OK" name="bouton_ok" />
                <input type="reset" value="Effacer" name="bouton_reset" />
            </form>
    </body>
</html>
```

Noter que le formulaire référence référence le script jsp form2.jsp.

Etape 4 : Création du script form2.jsp
créer le script jsp form2.jsp qui va rechercher une personne à partir de son nom en utilisation le javabean AnnuaireBean. Le code du script est donné dans la suite. Le code du script à un bug, il plante si la personne recherché n'existe pas dans l'annuaire.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Resultat</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%= "Recherche de l'utilisateur"%>
        <%= request.getParameter("lenom") %><br>
        <jsp:useBean id="liste" scope="session" class="beans.AnnuaireBean" />
        <% PersonneBean p = liste.findPersonne(request.getParameter("lenom"));
        %>
        <%= "Personne trouvee :" %>
        <%= p.getNom()%>
        <%= p.getPrenom()%>
    </body>
</html>
```

Etape 5 : Tester votre programme (run du projet).
Le premier acces est assez long car netbean lance le serveur d'application qui compilera les script jsp en servlet. Les accès ultérieurs sont plus rapides.
Par defaut le navigateur affiche la page correspondant à index.jsp.
Pour rechercher une personne, vous devez vous connecter à index2.jsp. Pour cela taper dans le navigateur l'url suivante : http://localhost:8080/MiniFormulaire/index2.jsp

Noter que les bean sont créés dans la session de l'utilisateur. L'annuaire n'est donc pas globale mais propre pour chaque client (un client ne voit pas les modifications faites par un autre client). Ceci vient du fait de l'utilisation de l'attribut `scope="session"` des balises `jsp:useBean`.

Etape 6 : Ajouter des vérifications qui traite des erreurs.
6.1 On ne doit pas pouvoir ajouter une personne qui existe déjà dans l'annuaire.
6.2 Le programme ne doit pas planter si on recherche une personne inconnu mais afficher un message qui indique que la personne est inconnue.
6.3 L'utilisateur doit saisir forcement un nom, un prénom et un email pour ajouter une personne.

Etape 7 : Modification des formulaires et des beans
Modifier index.jsp existant pour ajouter la saisie d'une adresse email.
Modifier form1.jsp pour enregistrer l'email des personnes.
Modifier le bean PersonneBean pour ajouter une propriété email.
Modifier form2.jsp pour afficher le nom, le prenom et l'email de la personne recherché dans un tableau.

Etape 8 : Programmation d'une contrôleur et utilisation de la directive forward
Dans cette étape nous allons ajouter un script jsp qui va servir de contrôleur.

8.1 Création du contrôleur
Copier index.jsp dans index1.jsp
Modifier index.jsp. Nous allons avoir 1 boutons pour l'ajout d'une personne et un bouton pour la recherche d'une personne.

Le code d'index.jsp est donné dans la suite :

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Gestion des personnes</h1>
        <h2>Choisisser votre action : </h2>
            <form name="ctler" action="ctrler.jsp">
                <input type="submit" value="Ajouter" name="add" />
                <input type="submit" value="Rechercher" name="find" />
            </form>
    </body>
</html>
```

Ajouter un nouveau script jsp (ctrler.jsp) qui va orienter les requêtes vers form1.jsp ou form2.jsp (les vues) en fonction du bouton clické par le client. Le code de ctrler.jsp est donné dans la suite.

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <% if ((request.getParameter("add"))!=null){
       request.getRequestDispatcher("index1.jsp").forward(request, response);
       } else {
          request.getRequestDispatcher("index2.jsp").forward(request, response);
       }
       %>
    </body>
</html>
```

Remarquer la façon de tester le bouton clické et la redirection vers les vues.

8.2 Modifications de form1.jsp et form2.jsp

On modifie ces script pour ajouter un bouton de retour qui permet de revenir à la page principale (index.jsp). Voici le code de form1.jsp et form2.jsp.

Form1.jsp :

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
   <h1>Ajout d'une personne</h1>
   <%= "Ajout de l'utilisateur"%>
   <%= request.getParameter("lenom") %>
   <%= request.getParameter("leprenom") %>
   <jsp:useBean id="liste" scope="session" class="beans.AnnuaireBean" />
   <% liste.addPersonne(request.getParameter("lenom"),
            request.getParameter("leprenom"));
   %>
   <form name="retour" action="index.jsp">
     <input type="submit" value="Retour" name="retour_index" />
   </form>
  </body>
</html>
```

form2.jsp :

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Resultat</title>
    </head>
    <body>
        <h1>Recherche d'une personne</h1>
        <%= "Recherche de l'utilisateur"%>
        <%= request.getParameter("lenom") %><br>
        <jsp:useBean id="liste" scope="session" class="beans.AnnuaireBean" />
        <% PersonneBean p = liste.findPersonne(request.getParameter("lenom"));
        %>
        <%= "Personne trouvee :" %>
        <%= p.getNom()%>
        <%= p.getPrenom()%>
         <form name="retour" action="index.jsp">
            <input type="submit" value="Retour" name="retour_index" />
        </form>
    </body>
</html>
```

Etape 9 : Tester le programme.

Etape 10 : Option pour les courageux, vous pouvez utiliser la persistance des Java Beans pour stocker le modèle dans un fichier et le relire à l'initialisation de la servlet s'il existe.