TP06
Utilisation de la bibliothèque JSTL (partie2)

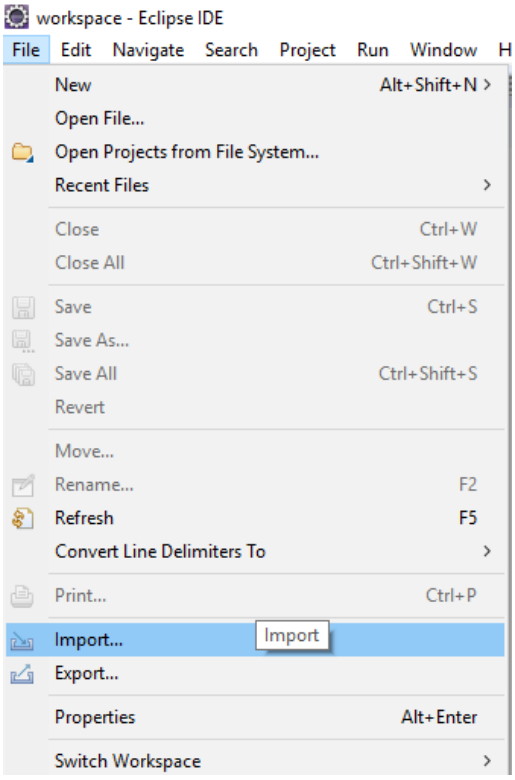
Objectifs

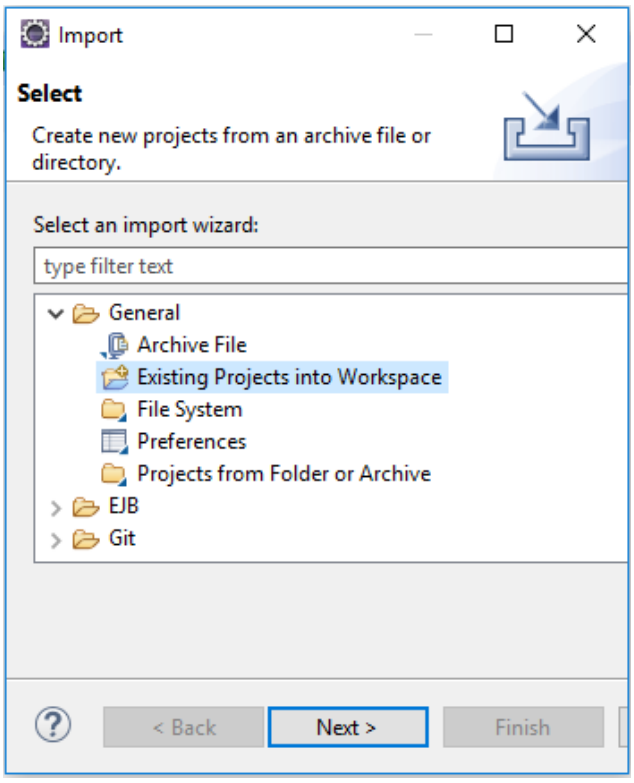
Améliorer la qualité de présentation des pages web en intégrant des éléments de la bibliothèque JSTL (JSTL : Java Standard Tag Library)

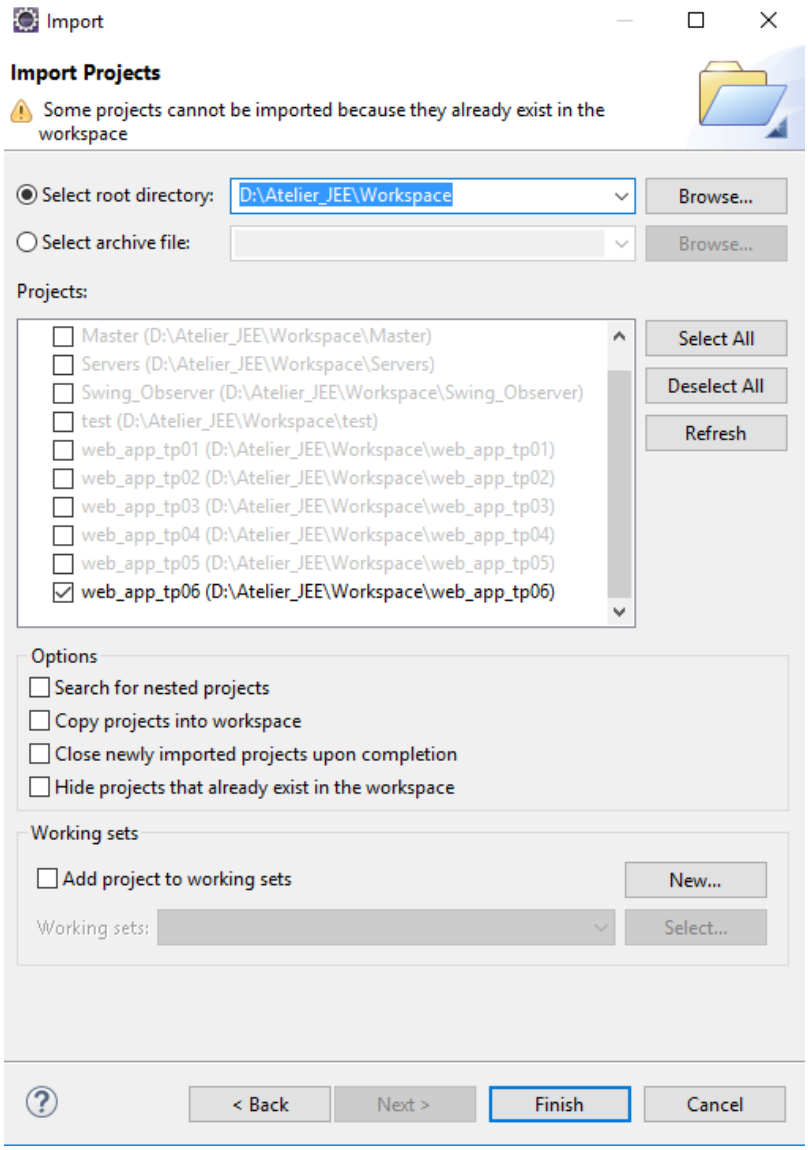
1. **Gestion du flux avec JSTL**
   - Utilisation des conditions
   - Utilisation des structures itératives (les boucles)
2. **Formatage de dates**
   - Utiliser les balises **formatDate** et **parseDate**

Les structures conditionnelles

1. Ouvrir l'EDI «**eclipse**» dans le workspace «**D:\Atelier_JEE\workspace** »
2. Décompresser le fichier « **web_app_tp06.zip** » (donné avec l'énoncé) dans le workspace.
3. Importer le dossier obtenu (projet JAVA) dans « eclipse » :







4. Ce projet répond à l'exercice de l'atelier (05), lancer l'exécution

5. Nous visons à améliorer l'affichage de la vue « **formProduit.jsp** » :

- Le volet d'affichage des erreurs ne doit pas apparaitre en cas d'absence d'erreur. Ajouter, alors, une condition en utilisant la balise « **if** » ayant la syntaxe suivante :

```
<c:if  test ="${condition}" > valeur si condition VRAI </c:if>
```

- Ajouter, dans le fichier « **formProduit.jsp** », le code en gras ci-dessous :

```
<c:if test="${not empty erreur }">
      <div class="erreur">
      <c:out  value="${ erreur }"> </c:out>
     </div>
</c:if>
```

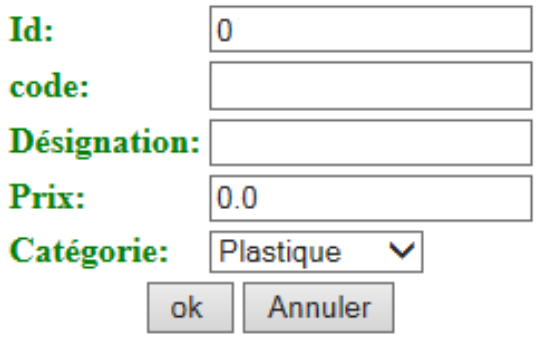
Le block entre les deux balises <c:if> et </ c:if> est exécuté si la condition de test est vrai càd si la la variable erreur est nulle ou vide

- Copier le fichier « **jstl-1.2.jar** » (donné avec l'énoncé) dans le dosier « **lib** » du « **WEB-INF** » du projet.
- Lancer l'exécution.

6. Ajouter, au bean « **Produit** », un nouveau champ « **categorie** » de type « **chaine de caractères** » (**String**). Réaliser les modifications nécessaires :
   - ❖ Ajouter les deux méthodes «**setProduit(..)** » et « **getProduit()** »
   - ❖ Ajouter un constructeur qui reçoit en paramètres tous les attributs (y compris le nouvel argument « **categorie** »).
   - ❖ actualiser la méthode « **toString()** » pour afficher la valeur de l'attribut « **categorie** »

NB : Vous pouvez utiliser les assistants de « eclipse »

7. Modifier la vue « **ficheProduit.jsp** » en ajoutant une liste déroulante qui affiche trois catégories :
   - ❖ Plastique
   - ❖ Alimentaire
   - ❖ Cosmétique



Ajouter le code en gras ci-dessous au fichier « **formProduit.jsp** » :

```
<tr>
      <td class="label">Prix:</td>
      <td><input type="text" name="prix" value="${ bean.prix }" /></td>
</tr>
<tr>
      <td class="label">Catégorie:</td>
      <td><select name="categorie">
            <option value="">--------------</option>
            <option value="Plastique">Plastique</option>
            <option value="Alimentaire" >Alimentaire</option>
            <option value="Cosmétique">Cosmétique</option>
            </select></td>
</tr>
<tr>
      <td align="center" colspan="2">
        <input type="submit" value="ok " />
        <input type="reset" value="Annuler" />
      </td>

</tr>
```

8. Ajouter le test sur la valeur de « **categorie** » dans le contrôleur (servlet « **formProduitAction** ») : Si le paramètre « **categorie** » est **vide** ou **null** alors retourner à la vue « **ficheProduit.jsp** » pour choisir une valeur à partir de la liste déroulante.

9. Lancer l'exécution et remarquer que si une valeur de « **categorie** » est déjà saisie, elle sera effacée en cas de retour au fichier « **formProduit.jsp** »

10. Changer le code de la vue « **formProduit.jsp** » pour conserver la valeur de « **categorie** » déjà choisie. Pour cela, utiliser l'attribut « **selected** » de la balise « **option** » comme suit :

```
<td><select name="categorie">
     <option value="" >--------------</option>
     <option value="Plastique"<c:if test="${param.categorie == 'Plastique'}"
        >selected</c:if> >Plastique</option>
     <option value="Alimentaire"<c:if test="${param.categorie ==
'Alimentaire'}"
         >selected</c:if> >Alimentaire</option>
     <option value="Cosmétique" <c:if test="${param.categorie ==
'Cosmétique'}"
        >selected</c:if>>Cosmétique</option>
</select></td>
```

NB :

- la variable param représente ici l'objet formulaire
- La variable param.categorie représente le champs « categorie » du formulaire

Les structures itératives

11. Maintenant, nous visons à afficher, dans la vue, les erreurs de saisie en détail : dans la zone d'affichage des erreurs, afficher le nom du (ou des) champs vide(s).

- Il suffit d'utiliser une collection (**Exemple** : un objet de type « **ArrayList <String>** » retourné par la servlet « **formProduitAction** » qui contient la liste des erreurs détectées. Ajouter, dans la servlet le code nécessaire pour :
    - Déclarer un objet « **erreurs** » de type « **java.util.ArrayList** »
    - Tester chaque paramètre à part. En cas d'erreur détectée, ajouter un message à l'objet « **erreurs** »
    - Enfin, si l'objet « erreurs » est non vide, le passer comme attribut dans « **request** » pour être envoyé à la vue JSP

- Pour cela, utiliser la balise <**c:forEach**> qui permet de parcourir les différents éléments d'une collection et ainsi d'exécuter de façon répétitive le contenu de son corps.
- Voici le code de la zone d'affichage des erreurs dans la vue JSP :

```
<c:if test="${not empty erreurs }">
        <div class="erreur">
            <ul>
              <c:forEach items="${ erreurs}" var="e"
>
                 <li>${ e }" </li>
             </c:forEach>
            </ul>
        </div>
    </c:if>
```

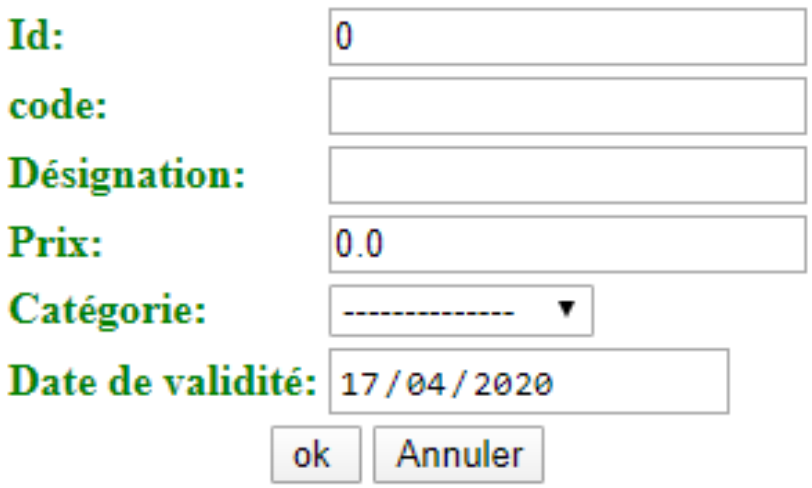
- Lancer l'exécution.

Formatage du type Date

12. Ajoutons maintenant un attribut « **validite** » au bean « **Produit** » qui représente la date de validité d'un produit. Ce nouvel attribut est de type « **java.util.Date** » :

- ❖ Ajouter les deux méthodes «**setValidite(..)** » et « **getValidite()** »
- ❖ Ajouter un constructeur qui reçoit en paramètres tous les attributs (y compris le nouvel argument « **validite** »).
- ❖ actualiser la méthode « **toString()** » pour afficher la valeur de l'attribut « **validite** »

13. Modifier la vue « **ficheProduit.jsp** » en ajoutant une zone de texte de type « date » pour saisir la date de validité :



14. Ajouter le code html suivant dans la définition du formulaire :

```
<tr>
 <td class="label">Date de validité:</td>
 <td>
     <input type="date" name="validite" value="${ bean.validite }"
/>
 </td>
</tr>
```

15. Modifier le test dans la servlet « **formProduitAction** » pour ne pas accepter une date de validité égale ou inférieure de la date d'aujourd'hui

```
        // Récupérer la valeur du paramètre "validite"
           String v = request.getParameter("validite");

         // Afficher la valeur du paramètre "validite"
      System.out.println("La valeur de validite est:" + v);
if(v == null || v.equals(""))
           {
           erreurs.add("Veuillez remplir le champ 'validite' ");
           }
           else
           {
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
                try {
                     java.util.Date dv=sdf.parse(v);
                     java.util.Date now =new Date();
                      System.out.println("date validite : " +
sdf.format(dv));
                    System.out.println("date now : " + sdf.format(now));
                         // passer la valeur de validite au bean
                    produit.setValidite(dv);
                    if (! (now.compareTo(dv) > 0)) {
                     erreurs.add("Date de validité supérieure ou égale à la
date actuelle... ");
                    }

                } catch (ParseException e) {
                     System.out.println("Problème de format de date...");
                }
           }
```

16. Passer maintenant à récupérer la date de validité lorsque la servlet envoie des messages d'erreurs. L'attribut date est envoyé avec le type « **java.util.Date** » et a besoin d'être formaté selon pattern « **yyyy-MM-dd** » pour être affiché dans son champ de formulaire :

- Ajouter la déclaration suivante de la bibliothèque de formatage JSTL dans le fichier « **taglibs.jsp** » :

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/fmt"
prefix = "fmt"  %>
```

- Changer le code de déclaration du champ « **validite** » dans le formulaire comme suit :

```
<tr>
    <td class="label">Date de validité:</td>
    <td><input type="date"
        name="validite"
        value="<fmt:formatDate pattern = 'yyyy-MM-dd'
value = '${bean.validite}' />" /></td>
</tr>
```

- Lancer l'exécution et remarquer le changement.

Exercice

- **Copier le contenu du projet « web_app_tp04 » dans un nouveau projet « mini_projet »**
- **Réaliser les changements nécessaires pour coder les vues avec la bibliothèque « JSTL »**
- **Concevoir le modèle relationnel des données de votre mini-projet « Dashboard » qui :**
  - **Gérer des capteurs par catégorie**
  - **Gérer l'historique des capteurs**
  - **Fournir des statistiques utiles**

- **Implémenter le modèle conceptuel dans une base de données MySQL**
- **Implémenter la couche métier**
- **Implémenter les contrôleurs web et les vues JSP à base de JSTL**