TP 3 les JavaBeans, JSTL, JSP et Servlet

1. Les JavaBeans

Un Bean (modèle dans l'architecture MVC) est un simple objet Java qui suit des contraintes et représente des données du monde réel.

Il doit être une classe publique ayant un constructeur par défaut (public et sans paramètres) ; ne doit pas avoir des champs publics ; définit des propriétés (les champs privés) accessibles via les getter et setter (qui sont des méthodes publiques) (exemple de propriétés : `private String nom= null`)

La méthode de lecture des propriétés possède la signature suivante :

```
public TypeDePropriété getNomDeLaPropriété() ;
```

Exemple `public String getNom();`

Le méthode d'écriture des propriétés possède la signature suivante :

```
public void setNomDeLaPropriété (TypeDePropriété
valeurDePropriété)
```

Exemple `public void setNom(String valeur);`

L'utilisation d'un Bean avec la syntaxe Java est comme suite :

```
ExempleBean exemple = new ExempleBean();
exemple.setNom("Mohamed");
String leNom = exemple.getNom();
```

Utilisation d'un Bean avec la syntaxe XML dans une page JSP est comme suite : Création simple d'une variable Bean dans une page JSP :

```
<jsp:useBean
id = "nomDuBean"
class = "nomPackage.NomClasse">
</jsp:useBean>
```

Id : c'est le nom de l'objet Bean.
Class : la classe NomClasse qui représente le Java Bean. Elle se trouve dans le package nomPackage de l'application courante.

Remplir un champ propriété dans un Bean par une constante :

```
<jsp:setProperty name="nomDuBean" property="uneProprieteDeBean"
value="uneValeurConstante" />
```

Remplir un champ propriété dans un Bean par un paramètre d'un formulaire :

```
<jsp:setProperty name="nomDuBean" property="unePropriete"
param="nomduParametre />
```

Récupérer un champ propriété d'un Bean:

```
<jsp:getProperty name="nomDuBean" property="unePropriete"/>
```

2. JSTL (JSP standard Tag Library)

JSTL (JSP standard Tag Library) permet de convertir un code Java en HTML.

JSTL EL (Expression Language) aide à chercher des données stockées dans une classe Java.

Les core Tags :

- **<c :out>** pour sécuriser l'affichage des données saisies
- **<c :url> pour la gestion des liens**
- < c : import> pour inclure une page dans une autre
- Pour la condition, vous pouvez modifier vos servlets pour qu'elles transmettent à vos JSP une information supplémentaire on utilise la balise **<c :choose>** ou **<c :if>**

Exercice 1

Créer un projet Demo. Créer une JSP nomée « display.jsp » et une servlet « Demo.java » (en utilisant un code java)

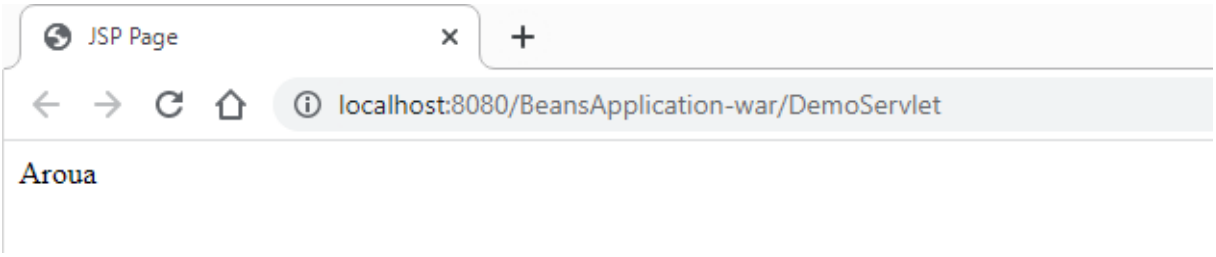
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        String nom= "Aroua";

    request.setAttribute("label", nom);
    RequestDispatcher rd = request.getRequestDispatcher("display.jsp");
    rd.forward(request, response);
```

display.jsp

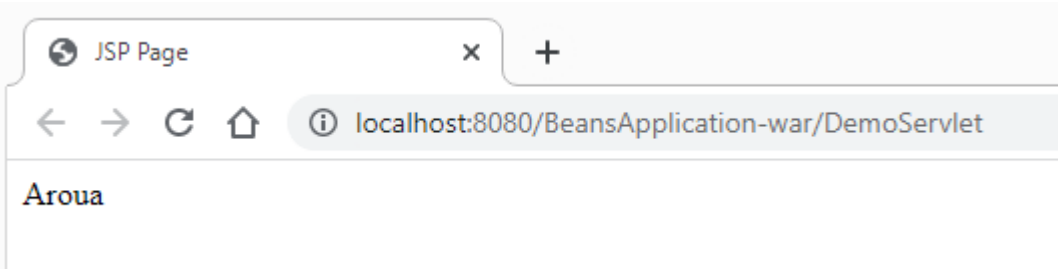
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <% String nom = request.getAttribute("label").toString();
           out.println(nom);
%>
    </body>
</html>
```

On obtient ce résultat



En utilisant EL dans la page JSP

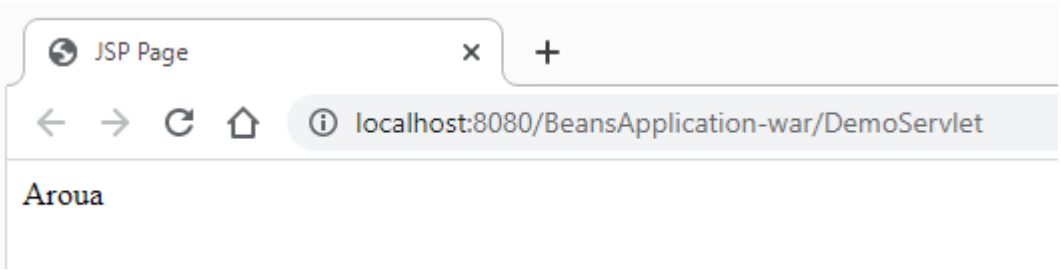
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
      ${label}
    </body>
</html>
```



Si on utilise JSTL tags (ça s'appelle un custom tag et pour l'utiliser on doit ajouter la librairie JSTL à notre projet.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

      <c:out value="${label}" />
    </body>
</html>
```



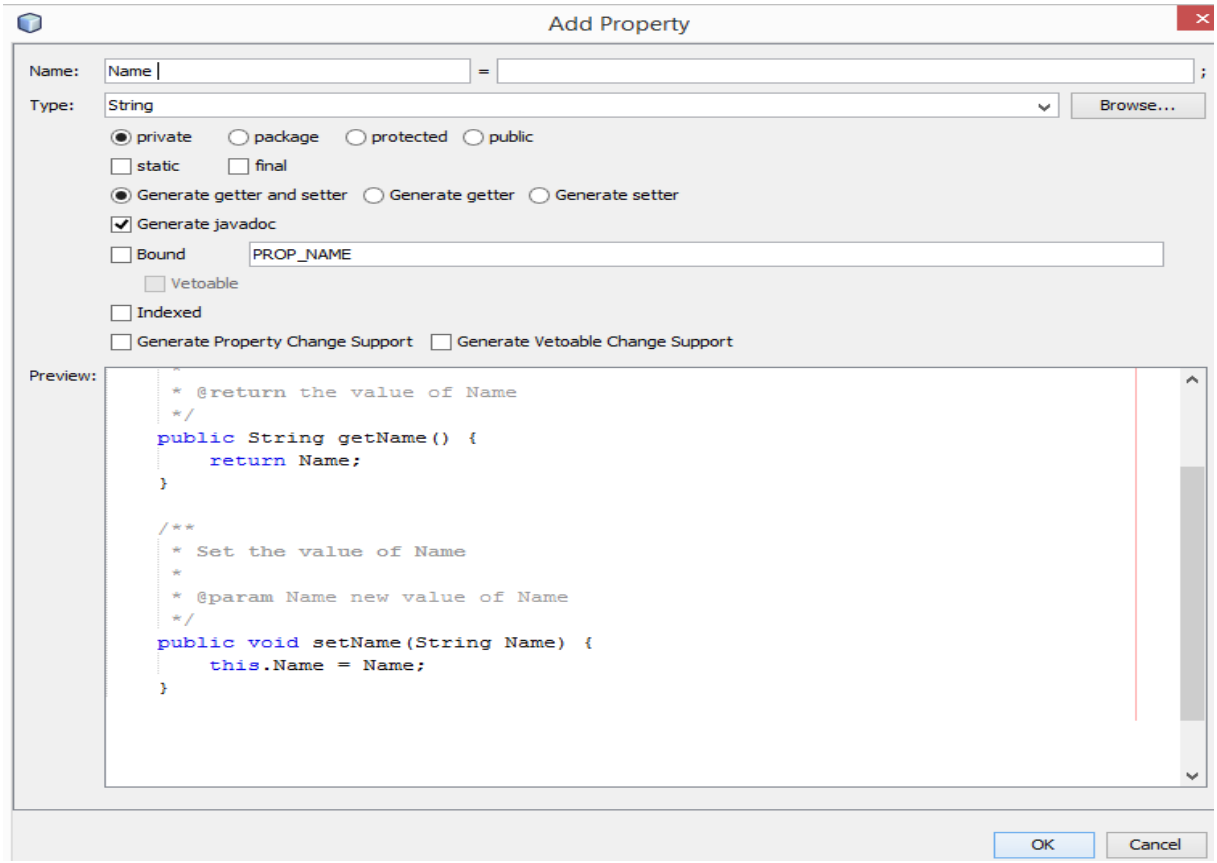
Exemple pour afficher une page web

```
    </head>
    <body>

        <c:import url="https://www.yamli.com/clavier-arabe/" />
```

Exercice2 :

Créer un nouveau java package nommé BeanPackage ; créer une nouvelle classe Java nommée Student, il a un nom de type String et un numéro no de type int on appuie sur le clic droit de la souris et on choisit insert code > constructor ; de même pour add property ; replace the Name by Name (tu peux choisir entre ajouter setter ou getter).



On crée une page JSP nommée BeanUsageJsp

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <jsp:useBean id="Bean" class="Beanpackage.Student">
            <jsp:setProperty name="Bean" property="name" value="Amin"/>
            <h2> Bienvenue <jsp:getProperty name="Bean" property="name" /> </h2>
        </jsp:useBean>
    </body>
</html>
```

Run la page pour voir le résultat

Exercice 3

Manipuler le JavaBean avec la Servlet DemoServlet.java (le but est de renvoyer plusieurs étudiants)

On revient au Bean Student on ajoute un constructeur ayant le no et name comme paramètres et les getters et setters.

```java
public Student(int no, String Name) {
    this.no = no;
    this.Name = Name;
}
```

Puis on click droit sur la souris, insert code > generate toString()

```java
@Override
public String toString() {
    return "Student{" + "no=" + no + ", Name=" + Name + '}';
}
```

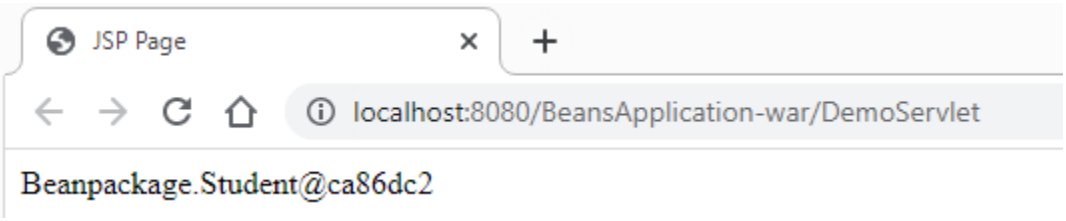
On crée un nouveau étudiant de no 1 et de name Aroua et on le renvoie à la page jsp

```java
    Student s= new Student(1,"Aroua");
request.setAttribute("student", s);
RequestDispatcher rd = request.getRequestDispatcher("display.jsp");
rd.forward(request, response);
```

In the JSP display.jsp, on écrit ce code

```jsp
<c:out value="${student}" />
```

le résultat obtenu est



Ce n'est pas le résultat souhaité on veut afficher le nom donc le code convenable de la JSP est

```
<c:out value="${student.name}" />
```

Si on veut renvoyer une liste d'étudiants à une page jsp

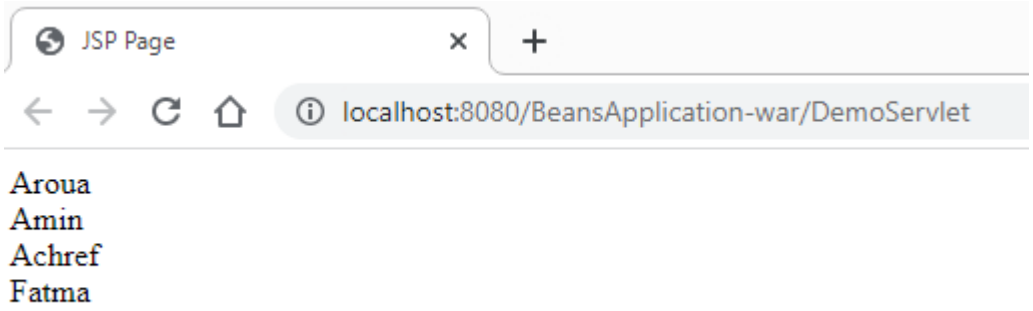
Pour la servlet DemoServlet on ajoute le code suivant :

```
List<Student> std= Arrays.asList(new Student(1,"Aroua"), new Student(2,"Amin"),new Student(3,"Achref"), new Student(4,"Fatma"));

request.setAttribute("students", std);
RequestDispatcher rd = request.getRequestDispatcher("display.jsp");
rd.forward(request, response);
```

On ajoute dans la JSP le code suivant pour afficher les étudiants par leurs noms.

```
<c:forEach items="${students}" var="s">
    ${s.name} <br/>
</c:forEach>
```



Exercice 4

Utilisation des JSTL functions tags.

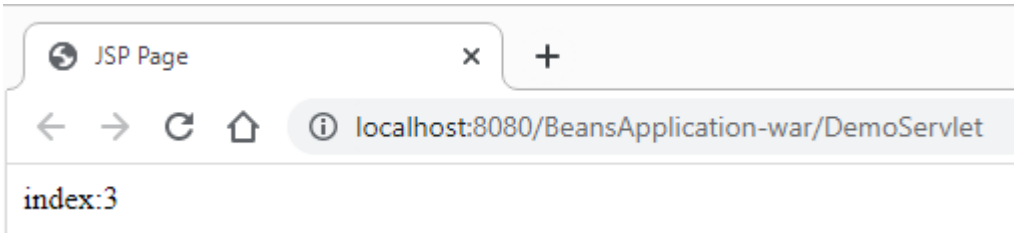
Dans une page JSP on ajoute le préfix core et puis le préfix des fonctions

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

On va utiliser **fn:indexOf() (fn:indexOf()** function return an index of string. It is used for determining the index of string specified in substring).

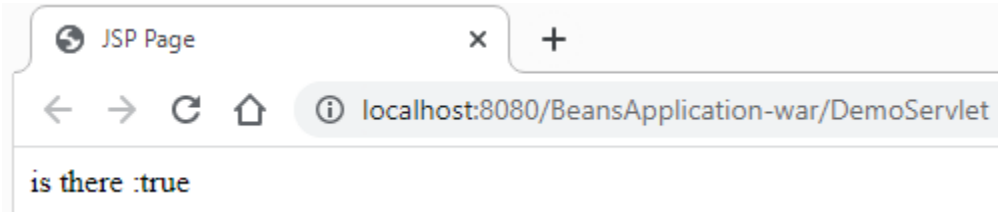
On veut savoir l'index de is dans la chaîne str

```
<c:set var="str" value="He is a Java Programmer"/>
index:${fn:indexOf(str, "is")}
```



On veut vérifier si str contient Java

```
<c:set var="str" value="He is a Java Programmer"/>
is there :${fn:contains(str, "Java")}
```



Autrement : on test si java existe on affiche Java is there

```
<c:set var="str" value="He is a Java Programmer"/>
<c:if test="${fn:contains(str, 'Java')}">
    Java is there
</c:if>
```