11. Expliquer l'objectif, les diverses syntaxes (en les illustrant par un exemple) et le mécanisme des Java Server Pages (que se passe-t-il exactement quand un client accède à un JSP au moyen de son browser ?). Pourquoi le mécanisme des balises à Java Beans est- il vite ressenti comme nécessaire ? Expliquer comment fonctionne ce mécanisme des balises pour Java Beans à partir d'un exemple complet.

Objectif et description :
Définition: Les JSP (Java Server Page) permettent de créer dynamiquement une partie de page HTML stockée sur le serveur, le reste de la page étant statique.

Il s'agit donc de crée une page HTML dynamique (avec une certaine partie statique néanmoins). La zone dynamique est codé en Java et **généré du coté serveur** (contrairement à un langage de script qui est généré du coté client). Nous nous rapprochons donc du fonctionnement du PHP avec en plus, grâce au Java, la portabilité ,sécurité et l'utilisation des JavaBean. Les JSP sont de plus multithread. En effet, plusieurs thread peuvent accéder en même temps à une page JSP.

Un objectif des JSP est également le faite que l'on puisse dissocier le coté graphique de la page et le coté logique métier. En effet, une équipe d'infographie peuvent facilement manipuler la partie HTML de la page sans interferer avec la partie Java tandis que l'équipe de développement peuvent manipuler la partie code JAVA sans que celle-ci interfere avec la partie visuel.

Exemple de JSP avec sa syntaxe :

```
<%@ page language="java" %>
<%@page contentType="text/html"%> <HTML>
<HEAD>
<TITLE>Bonjour a repetition</TITLE>
<BODY>
*** Accueil dans le monde des JSP *** <p>
<% int nbre = Integer.parseInt(request.getParameter("nombre")); for (int i=0; i<nbre; i++)
{ %>
Bonjour ! <BR>
<% }
%> </P>
Cela nous fait <%=nbre %> fois ... </BODY>
</HTML>
```

Nous nous retrouvons donc avec un système de Balise. Nous retrouvons les balises habituels de HTML qui forme une page.
Les différentes zone d'une JSP sont appelées:

- Des **scriptlets** ; si elle contiennent du code Java. C'est scriptes sont donc les zones délimité par les tags <% et %>. Nous retrouverons donc le code métiers entre ces balises. Une scriptlets peut laissé un bloc de code ouvert tant qu'une autre scriptlet la referme plus tard (ce qui est le cas dans le bloc « for » de l'exemple ci dessus).
- Des **directives** ; si elles définissent des attributs constant (pour toute requête des clients) de la futur page, elle sont délimité par les tags <%@ et %>, Nous retrouvons ces directives au début de notre exemple :
  - <%@ page language=« java » %> : qui indique le langage utilisé dans les scriptlets est du Java
  - <%@ page contentType =«text/html» %> : qui indique que le type MIME de retour est une page de type text/html.
- Des commentaires ; si elle sont délimités par les tags <%-- et —%>, qui seront ignorés.

- Les **déclaration** de variable ou de méthode ; si ils sont délimité par le tag <%! et %>, voici un exemple :

```
<%! int cpt=0; %>
<%! public int Fois2(int x)
{
        return x*2;
} %>
```

- les **actions** ; délimité par les tags spécifique comme <%jsp:…> pour gérer par exemple les beans et les plugins.

Le mécanisme des JSP :

Lors de l'appel du JSP, le code JSP va être compilé par le moteur à servlets (= son compilateur de pages) en une servlet et c'est cette servlet qui sera, en définitive, compilée puis exécutée. La page résultante sera envoyé au client sous forme de page statique normal. Les methodes et variables déclaré dans un bloc de déclaration deviendront alors des méthodes de cette servlet.
Un container (ou moteur) à servlets qui prend en charge l'appel du compilateur de page est appelé « **container à JSP** ».

Fonctionnement du mécanisme des balises pour Java Bean :

Le mécanisme des balises à Java Bean et personnalisées permettent de séparer la présentation de la logique fonctionnelle. Les JSP fournissent des balises particulières qui sont des « actions » : il s'agit de balises permettant de spécifier l'utilisation d'un élément extérieur (un bean, un plugin, un autre JSP) et/ ou d'effectuer une tâche selon des informations obtenues au moment où le JSP est accédé par le client.

```
<nom_action attribut=valeur [attribut=valeur]>
Corps_action
</nom_action>
```

Une bibliothèque de base est celle dont le préfixe est « :jsp ». On peut cependant créer nos propres librairies, ce que ne manque pas de faire les constructeurs.
Exemple : Oracle ⇔ <ora : … >

Attention à ne pas confondre « action » et « directive »

- Une directive est simplement utilisée lors de la « compilation de la page », c'est-à-dire lors de la génération de la servlet correspondante au JSP.
- Une action représente une action dynamique qui à lieu lors de l'exécution de cette servlet.

Ainsi, lorsque nous vous voulons utiliser un Bean, on utilisera la balise suivante au sein de notre JSP :

```
<%jsp :useBean id="1" class="2"    scope="3 "/>
[
        <Code d'initialisation>
</jsp :usebean>]
```

Où 1 équivaut au nom de l'identificateur de l'objet instanciant le bean, 2 le nom de la classe du bean complet ou non dépendant des directives d'import, 3 la portée (page,resquest,session,application).

La portée permet de définir si un objet pour la portée indiquée existe déjà ou non. Le cas échéant, on créé un nouvel objet.

Au sein du code d'initialisation, nous pouvons retrouver des balises où l'on initialise ou récupère les propriétés du bean via :

```
<jsp :getProperty name= "nom du bean" property="nom de la propriété" />
<jsp :setProperty name= "nom du bean" property="nom de la propriété" value="valeur"/>
```

Précisons qu'ils subsistent des balises personnalisées visent le même objectif en offrant des balises d'action dont le code est situé au sein d'une classe et non plus au sein d'un bean. Bien entendu, tout un mécanisme existe pour réaliser cela notamment l'utilisation d'une bibliothèque de ces balises qui est décrite par un TLD (Tag Library Descriptor), concrètement, un fichier XML d'extension « tld ». Il suffit de spécifier que l'on va faire référence à une bibliothèque bien précise pour pouvoir ensuite utiliser toutes les nouvelles balises correspondantes.

Exemple :

Le bean :

```
/* PreferencesWeb.java */
package Preferences;
import java.beans.*;
import java.io.Serializable;
import java.awt.*;

public class PreferencesWeb extends Object implements Serializable {
    private PropertyChangeSupport propertySupport;
    public PreferencesWeb() {
        couleurFond = "blue"; couleurTrait = "black";
        langue = "FR"; uniteMonetaire = "EUR"; propertySupport = new PropertyChangeSupport(this);}
    public void addPropertyChangeListener(PropertyChangeListener listener){
        propertySupport.addPropertyChangeListener(listener); }
    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener); }
```

```
<%@page language="java" %>
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="thePreferences" scope="page" class="Preferences.PreferencesWeb" />
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page with bean PreferencesWeb - Yeahh ... </title>
    </head>
 <body bgcolor="<jsp:getProperty name="thePreferences" property="couleurFond" />" >
    You know what ? I'm happy
    <p>(Pierre Rapsat, folder de son dernier album)
    <HR>
    <p>Le caddie virtuel sera calculé en : <jsp:getProperty name="thePreferences" property="uniteMonetaire" />
 </body>
</html>
```

```
    public void setLangue(String langue) {
        this.langue = langue; }
    private String uniteMonetaire;
    public String getUniteMonetaire(){
        return this.uniteMonetaire; }
    public void setUniteMonetaire(String uniteMonetaire) {
        String oldUniteMonetaire = this.uniteMonetaire;
        this.uniteMonetaire = uniteMonetaire;
        propertySupport.firePropertyChange ("uniteMonetaire", oldUniteMonetaire,
        uniteMonetaire); }
}
```

La JSP :

```
<%@page language="java" %>
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<jsp:useBean id="thePreferences" scope="page" class="Preferences.PreferencesWeb" />
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page with bean PreferencesWeb - Yeahh ... </title>
    </head>
 <body bgcolor="<jsp:getProperty name="thePreferences" property="couleurFond" />" >
    You know what ? I'm happy
    <p>(Pierre Rapsat, folder de son dernier album)
    <HR>
    <p>Le caddie virtuel sera calculé en : <jsp:getProperty name="thePreferences" property="uniteMonetaire" />
 </body>
</html>
```