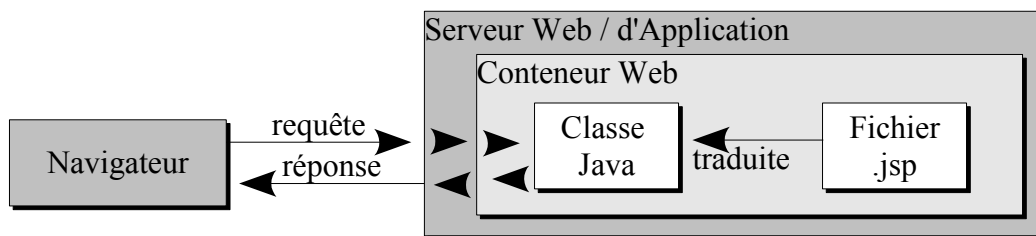


Java Server Page (JSP)

JSP est une technologie qui permet de générer dynamiquement des pages HTML. Nous en examinons ici les éléments principaux.

VUE D'ENSEMBLE

- ✓ Une page JSP est constituée de **code Java inséré dans du code HTML**. Le code Java permet de créer dynamiquement du contenu en fonction de : paramètres envoyés à la page, informations reçues de la session, information lue dans une BD, ...
- ✓ Elle porte l'extension « .jsp » et est demandée au *serveur web* par le protocole HTTP comme une page HTML classique. Elle est d'abord **traduite en une classe Java**, exécutée sur le serveur et **le résultat de son exécution est une page HTML classique** retournée au navigateur (donc pas de différence au niveau du client). Le *serveur web* ne traite pas directement la page mais passe par un *conteneur* qui s'occupe de la traiter.



- ✓ Alors qu'une JSP est du code Java inséré dans du code HTML, une **servlet** est une classe Java qui facilite la création dynamique de page HTML. La classe Java produite à partir d'une JSP est en fait une **servlet**.
- ✓ La compilation se fait lors de la première demande de la page. Un objet est alors instancié et c'est ce même objet qui est utilisé à chaque appel de la page. NetBeans permet de voir la servlet équivalente et de tenter de la compiler ce qui permet de détecter plus rapidement les éventuelles erreurs.
- ✓ Le code Java est délimité par des **balises**. Il y en a de plusieurs sortes.
- ✓ Une **directive** fournit des informations relatives à la page dans son entièreté.
Ex: `<%@ page import java.io.*%>` pour importer un package Java.
- ✓ Une **expression** est une simple expression Java.
Ex: `<%= new Date () %>` pour insérer à cette endroit la date courante.

- ✓ Une **scriptlet** est un fragment de code Java.

Ex: pour afficher 10x « Bonjour ! »

```
<% for(int i=0; i<10; i++) { %>
  Bonjour !
<% } %>
```

- ✓ Un commentaire s'indique par la balise `<!-- commentaire %>`.

- ✓ Rappelons-nous qu'une JSP est traduite en Java (une servlet qui produit le même effet). Lors de cette traduction, tout se retrouve dans une même méthode. Une **déclaration** permet d'obtenir du code Java en dehors de cette méthode (pour déclarer un attribut ou une méthode par exemple). Ne pas en abuser !

Ex: `<%! Date theDate = new Date(); %>`

- ✓ Les « **tags** » JSP (**actions** JSP) sont des éléments offrant un comportement plus complet. Certains sont standard (définis par la spécification) d'autres sont développées dans des librairies (les **TLD**, Tag Library Descriptor) que l'on peut inclure et utiliser.

Ex: `<jsp:useBean>` pour utiliser un JavaBean

- ✓ Certains objets sont prédéfinis; ils permettent d'accéder à l'environnement d'exécution de la page. Ex: L'objet *request* contient les paramètres envoyés à la page (via un HTTP/GET ou un HTTP/POST)

- ✓ Il est possible d'indiquer une page qui doit être affichée si une erreur se produit.

Pour cela, la page d'erreur contiendra la directive

```
<%@ page errorPage="maPageErreur.jsp" %>
```

Et la page d'erreur la directive

```
<%@ page isErrorPage="true" %>
```

- ✓ Une autre possibilité est de l'indiquer de façon globale (pour toutes les pages) via le fichier de configuration web.xml. Dans ce cas, il est possible de définir des pages différentes en fonction de l'exception qui se produit.

Ex: `<error-page>`

```
    <exception-type>java.lang.NumberFormatException</exception-type>
```

```
    <location>/BadNumber.html</location>
```

```
</error-page>
```

- ✓ Il existe pour tous les éléments JSP une **autre syntaxe** qui respecte la norme **XML**.

APPRENTISSAGE

Commencez par le petit tutoriel de NetBeans. (<http://www.netbeans.org/kb/55/quickstart-webapps.html>) Vous apprendrez ainsi :

- A créer un projet Web contenant des pages JSP
- A utiliser un JavaBean dans une page JSP
- A récupérer les informations d'un formulaire HTML dans une JSP

APPLICATION

Reprenez l'application d'interrogation de la BD eVente que vous avez écrite en servlet et refaites la en JSP.

COMPLÉMENTS

Il reste quelques notions de base que l'on voudrait vous voir pratiquer mais qui ne sont pas reprises dans le tutoriel NetBeans. Voyons-les ici.

Inclure d'autres pages

Pour l'uniformité d'un site, il est bon que toutes les pages se terminent par un même texte par exemple (auteur, date, copyright, ...). Tapez ce texte dans chaque page serait fastidieux à maintenir. Une meilleure solution est d'inclure un fichier commun.

Exercice :

- Reprenez votre application.
- Ajoutez un fichier « footer.jspf » (extension standard pour un *segment* de code JSP) reprenant un texte avec votre nom. (lisez et respectez la note donnée par NetBeans)
- Faites-y référence dans les autres pages via la directive
`<%@ include file="footer.jspf"%>`.
- Demandez à voir la servlet équivalente à vos pages JSP.

Exercice :

Avec la technique précédente, l'inclusion doit se faire explicitement dans chaque page. Une inclusion automatique dans tout un groupe de page, peut se définir via le fichier web.xml

```
<jsp-property-group>
  <url-pattern>/*.jsp</url-pattern>
  <include-prelude>/common-jsp/ps/prelude1.jspf</include-prelude>
  <include-coda>/common-jsp/ps/coda1.jspf</include-coda>
</jsp-property-group>
```

Utiliser des expressions

Cela permet d'inclure à un endroit donné, le résultat d'une expression Java.

Exercice :

- Modifiez votre *footer.jspf* en y ajoutant.
Page créée le `<%= new java.util.Date() %>`

Importer une classe Java

A ne pas confondre avec la directive « include ». Il s'agit ici d'inclure une classe Java pour une référence plus aisée dans le code

Exercice :

- Ajoutez en tête du fichier *footer.jspf* la directive suivante :
`<%@ page import="java.util.Date" %>`
- Simplifiez en conséquence le code qui génère la date.

Déclaration

Cette clause permet de donner un code Java qui sera placé en dehors de la méthode principale.

Exercice :

- Ajoutez à la page footer.jspf le code suivant.
`<%! int cpt1 = 0; %>`
`<% int cpt2 = 0; %>`
Compteur 1 = `<%=++cpt1%>`
Compteur 2 = `<%=++cpt2%>`

- Chargez la page et rechargez-la plusieurs fois. Pouvez-vous expliquer ce comportement ?
- Répétez-la même expérience mais à partir d'une autre machine. Que se passe-t-il ?

L'objet request

On peut accoler à une URL des paramètres envoyés à la page (regarder une requête à Google par exemple). La page reçoit ses paramètres via l'objet prédéfini *request*.

Exercice :

- Créez une page JSP (*monGoogle.jsp*) avec le code suivant


```
<%
    String result = request.getParameter("find");
%>
Désolé, aucun résultat trouvé pour <%= result%>
```
- Chargez la page dans le navigateur en donnant une valeur au paramètre find.
- Relisez le code du tutoriel qui construit le Bean avec les valeurs entrées dans le formulaire. Vous devriez mieux comprendre comment cela fonctionne.

Gestion de la session

Les JSP utilisent le protocole HTTP qui est sans état. Dès lors, comment gérer une session ? Grâce à l'objet prédéfini *session*.

Exercice :

- Modifiez la page JSP (*monGoogle.jsp*) qui ressemblera à ceci (supprimez le texte qui suivait)


```
<%
    session.setAttribute("resultat", "aucun");
%>
```
- Créez la page JSP (*resultat.jsp*)


```
<h1>Résultat de la recherche</h1>
<%=session.getAttribute("resultat"%>
```
- Chargez la page monGoogle dans le navigateur comme précédemment
- Chargez à présent la page *resultat*.

Redirection

Dans l'exemple précédent, vous avez du manuellement faire appel à la seconde page. Cette manipulation peut-être automatisée.

Exercice :

- Ajoutez à la page JSP (*monGoogle.jsp*) ce qui suit


```
<jsp:forward page="resultat.jsp" />
```
- Chargez la page monGoogle dans le navigateur comme précédemment.