CSI 3540

Structures, techniques et normes du Web

Séparer la programmation de la présentation à l'aide de JSP

Objectif:

1. Concevoir des documents JSP

Lectures:

- Web Technologies (2007) § 8
  Pages 432–447

Plan

1. Survol
2. Exécution d'applications JSP
3. JSP 2.0
    1. Expression Language (EL)
    2. Bibliothèques de balises JSTL
    3. JavaBeans et JSP
    4. MVC et JSP

Problématique

- La génération dynamique de documents XHTML est parfois nécessaire (engins de recherche, par exemple)
- Cependant, avec les technologies vues jusqu'à présent, **la logique des applications et la génération du code XHTML étaient entremêlées**

Cons:

- Programmes/documents sont difficiles à lire, à changer...
- Programmation et présentation sont bien souvent traitées par des experts distincts (équipe multidisciplinaire)
- Puisque le document principal est un programme Java (Servlet), il est difficilement manipulable par des applications graphiques (Dreamweaver, FrontPage...)

JavaServer Pages (JSP)

- Dans le cas des Servlets, un programme Java fait la génération d'un document XHTML

- Dans le cas de JSP, c'est **un document XHTML dans lequel des énoncés (Java) sont insérés**

- JSP et les Servlets sont souvent utilisés de concert, comme nous le verrons

JavaServer Pages

- Il y a deux formes syntaxiques :
  - **Classique** : Pre-JSP 2.0 (2003)
  - **Syntaxe XML** : Celle que nous utiliserons

JavaServer Pages

- Un document JSP comprend deux types de contenu :
  - **Statique** : décrit à l'aide de XHTML, SVG, XML, etc. (**modèle**)
  - **Dynamique** : décrit à l'aide d'éléments JSP (**actions JSP**)

GetDate.jspx

(forme déconseillée)

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns="http://www.w3.org/1999/xhtml">
 <head><title>La date du jour</title></head>
 <body style="font-size:x-large">
  <h2>Voici la date du jour :</h2>
   <p>
    <jsp:scriptlet>
      out.write( ( new java.util.Date() ).toString() );
    </jsp:scriptlet>
   </p>
 </body>
</html>
```

Technologie JSP

Traitement des document

La technologie JSP

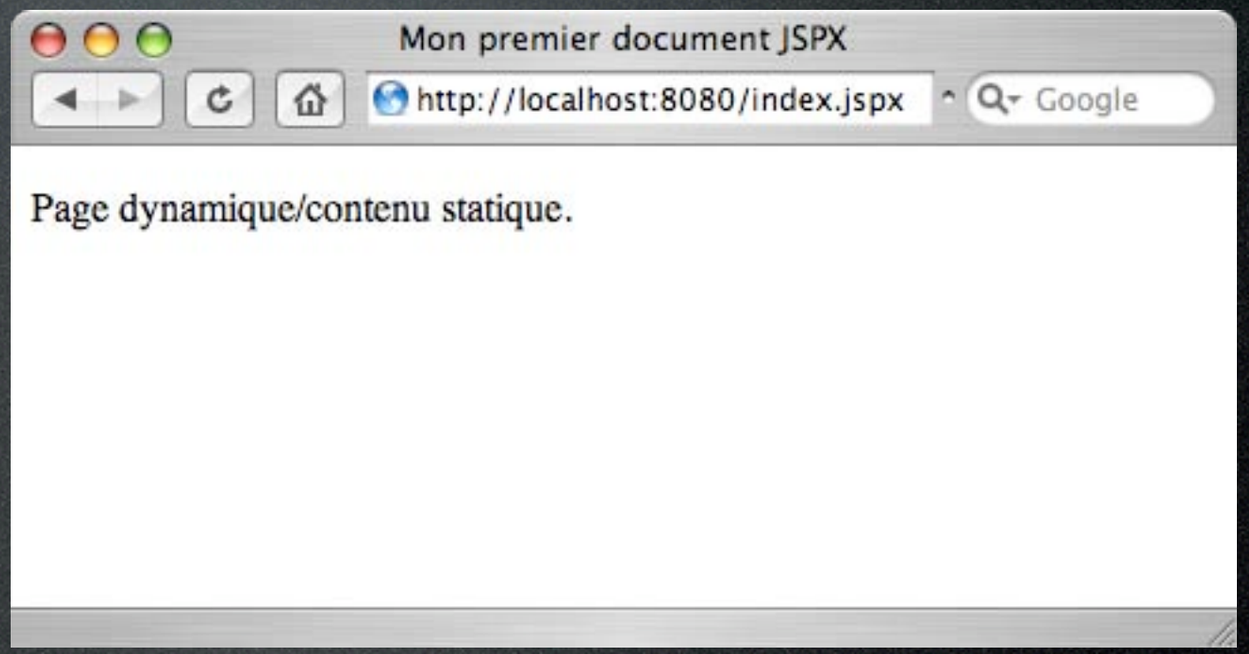
- Un langage, tel que XHTML, sert de **modèle** (template)
- Les éléments JSP sont exécutés pour chaque accès au document
- Comment ça fonctionne?

index.jspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Mon premier document JSPX</title>
  </head>
  <body>
    <p>
      Page dynamique/contenu statique.
    </p>
  </body>
</html>
```

Installation (1/2)
– Glass –

- Copiez index.jspx dans le répertoire domains/domain1/docroot
- Visitez la page :



Installation (2/2)
– Glass –

- **GlassFish produit automatiquement :**

  domains/domain1/generated/jsp/j2ee-modules/__default-web-module-server/org/apache/jsp/**index_jspx.java**

- **et le .class correspondant :**

  domains/domain1/generated/jsp/j2ee-modules/__default-web-module-server/org/apache/jsp/**index_jspx.class**

index_jspx.java

```
public final class index_jspx extends org.apache.jasper.runtime.HttpJspBase
                                implements org.apache.jasper.runtime.JspSourceDependent {
    ...
  public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
    out.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
    out.write("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
    out.write("<head>");
    out.write("<title>");
    out.write("Mon premier document JSPX");
    out.write("</title>");
    out.write("</head>");
    out.write("<body>");
    out.write("<p>");
    out.write("\n");
    out.write("      Page dynamique/contenu statique.\n");
    out.write("    ");
    out.write("</p>");
    out.write("</body>");
    out.write("</html>");
    ...
    }
}
```

Code XHTML transmis

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Mon premier document JSPX</title>
 </head>
 <body>
  <p>
   Page dynamique/contenu statique.
  </p>
 </body>
</html>
```

Pour chaque accès/appel, le même contenu sera généré

index.jspx

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns="http://www.w3.org/1999/xhtml">

  <jsp:directive.page contentType="text/html" />

  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML Basic 1.0//EN"
    doctype-system="http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd" />

    <head>
      <title>Mon premier document JSPX</title>
    </head>
    <body>
      <p>
        Page dynamique/contenu statique.
      </p>
    </body>
  </html>
```

index_jspx.java

```
public final class index_jspx extends org.apache.jasper.runtime.HttpJspBase
                                implements org.apache.jasper.runtime.JspSourceDependent {

 public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
    out.write( "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML Basic 1.0//EN\"" );
    out.write( "\ " http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd\">\n" );
    out.write( "<html xmlns=\"http://www.w3.org/1999/xhtml\">" );
    out.write( "<head>" );
    out.write( "<title>" );
    out.write( "Mon premier document JSPX" );
    out.write( "</title>" );
    out.write( "</head>" );
    out.write( "<body>" );
    out.write( "<p>" );
    out.write( "\n" );
    out.write( "      Page dynamique/contenu statique.\n" );
    out.write( "    " );
    out.write( "</p>" );
    out.write( "</body>" );
    out.write( "</html>" );
    ...
    }
}
```

Code XHTML transmis

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
         "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Mon premier document JSPX</title>
  </head>
  <body>
    <p>
      Page dynamique/contenu statique.
    </p>
  </body>
</html>
```

Remarques

- Les pages JSP sont **traduites** en Java, puis **compilées** par le serveur de conteneur
- Plusieurs extensions existent : EL, JSTL...

Page dynamique/ contenu dynamique

GetDate.jspx

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns="http://www.w3.org/1999/xhtml">
 <jsp:directive.page contentType="text/html" />
 <jsp:output
   omit-xml-declaration="yes"
   doctype-root-element="html"
   doctype-public="-//W3C//DTD XHTML Basic 1.0//EN"
   doctype-system="http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd" />
 <head><title>La date du jour</title></head>
 <body style="font-size:x-large">
  <h2>Voici la date du jour :</h2>
  <p>
   <jsp:scriptlet>
    out.write( ( new java.util.Date() ).toString() );
   </jsp:scriptlet>
  </p>
 </body>
</html>
```

Cette façon de faire (élément jsp:scriptlet) est déconseillée, nous aurons une meilleure solution sous peu

GetDate.jspx

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns="http://www.w3.org/1999/xhtml">
 <jsp:directive.page contentType="text/html" />
 <jsp:output
   omit-xml-declaration="yes"
   doctype-root-element="html"
   doctype-public="-//W3C//DTD XHTML Basic 1.0//EN"
   doctype-system="http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd" />
 <head><title>La date du jour</title></head>
 <body style="font-size:x-large">
  <h2>Voici la date du jour :</h2>
   <p>
    <jsp:scriptlet>
      out.write( ( new java.util.Date() ).toString() );
    </jsp:scriptlet>
   </p>
 </body>
</html>
```

```
public final class GetDate_jspx extends org.apache.jasper.runtime.HttpJspBase
                                implements org.apache.jasper.runtime.JspSourceDependent {
    ...
  public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
      out.write( "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML Basic 1.0//EN\"" );
      out.write( "\ " http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd\">\n" );
      out.write( "<html xmlns=\"http://www.w3.org/1999/xhtml\">" );
      out.write( "<head>" );
      out.write( "<title>" );
      out.write( "La date du jour" );
      out.write( "</title>" );
      out.write( "</head>" );
      out.write( "<body>" );
      out.write("<h2>");
      out.write("Voici la date du jour :");
      out.write("</h2>");
      out.write("<p>");
      out.write( ( new java.util.Date() ).toString() );
      out.write("</p>");
      out.write( "</p>" );
      out.write( "</body>" );
      out.write( "</html>" );
      ...
      }
}
```

JSP et Servlets

```
public final class GetDate_jspx extends org.apache.jasper.runtime.HttpJspBase
                                implements org.apache.jasper.runtime.JspSourceDependent {
    ...
  public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
    }
}
```

- Les documents JSP sont traduites en Java, le résultat est un **Servlet**
- **HttpJspBase** est une sous classe de **javax.servlet.http.HttpServlet**
- Lors d'un accès, le serveur exécute **_jspService** (plutôt que doGet())

public interface
JspSourceDependent
Interface for tracking the source files dependencies, for the purpose of compiling out of date pages. This is used for 1) files that are included by page directives 2) files that are included by include-prelude and include-coda in jsp:config 3) files that are tag files and referenced 4) TLDs referenced

JSP et Servlets

```
public void _jspService( HttpServletRequest request, HttpServletResponse response )
    throws java.io.IOException, ServletException {

  PageContext pageContext = null;
  HttpSession session = null;
  ServletContext application = null;
  ServletConfig config = null;
  JspWriter out = null;
  Object page = this;
  ...
  }
}
```

- **_jspService** définit et initialise plusieurs objets que nous utiliserons (implicitement) dans nos documents JSP, par exemple **out.write(...)** (À suivre...)

JSP et Servlets

```
public final class index_jspx extends org.apache.jasper.runtime.HttpJspBase
                                implements org.apache.jasper.runtime.JspSourceDependent {

  public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
    out.write( "<html xmlns=\"http://www.w3.org/1999/xhtml\">" );
    out.write( "<head>" );
    out.write( "<title>" );
    out.write( "Mon premier document JSPX" );
    ...
    out.write( "</html>" );
    ...
    }
}
```

- Le **modèle** est reproduit sur la sortie

  out.write(“...”);

JSP et Servlets

GetDate.jspx :

```
...
<jsp:scriptlet>
out.write( ( new java.util.Date() ).toString() );
</jsp:scriptlet>
...
```

GetDate_jspx.java :

```
public void _jspService( HttpServletRequest request, HttpServletResponse response )
      throws java.io.IOException, ServletException {
    ...
    out.write( "<html xmlns=\"http://www.w3.org/1999/xhtml\">" );
    out.write( "<head>" );
    out.write( "<title>" );
    out.write( "Mon premier document JSPX" );
    ...
    out.write( ( new java.util.Date() ).toString() );
    ...
    out.write( "</html>" );
    ...
    }
}
```

- Le contenu d'un élément **jsp:scriptlet** est reproduit textuellement dans le code Java généré

Documents JSP

EL, JSTL, JavaBeans, etc.

JSP et Servlets

- L'utilisation des éléments **jsp:scriptlet** nous donne toute la puissance du langage de programmation Java à l'intérieur des documents JSP
- Du même coup, **son utilisation réduit considérablement la séparation entre la programmation et la présentation**
- Comme le démontre l'exemple qui suit

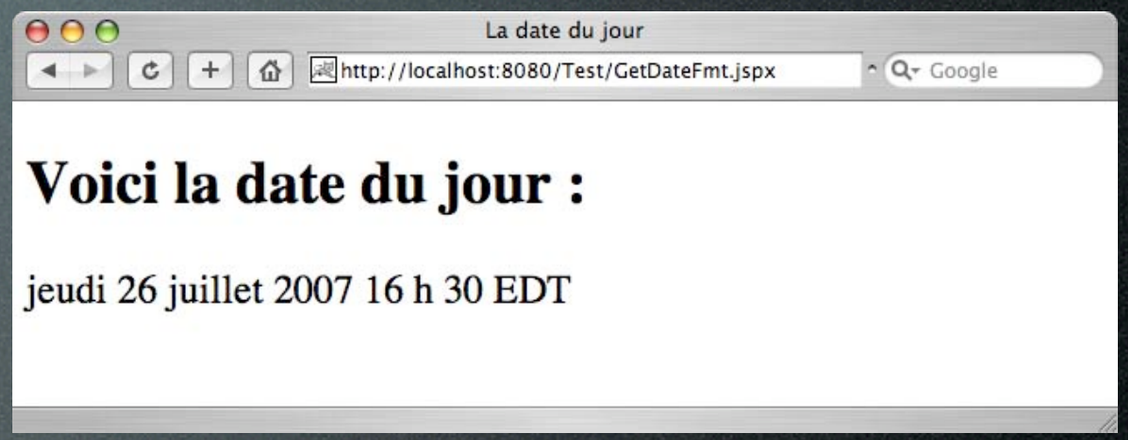
GetDateFmt.jspx

```
<html xmlns:core="http://java.sun.com/jsp/jstl/core"
    xmlns:jsp="http://java.sun.com/JSP/Page"
    xmlns="http://www.w3.org/1999/xhtml">

 <head><title>La date du jour</title></head>
 <body>
  <h2>Voici la date du jour :</h2>
  <p>
   <jsp:scriptlet>

    int fmt = java.text.DateFormat.FULL;
    java.text.DateFormat df;
    df = java.text.DateFormat.getDateTimeInstance( fmt, fmt, java.util.Locale.CANADA_FRENCH );
    out.write( df.format( new java.util.Date() ) );

   </jsp:scriptlet>
  </p>
 </body>
</html>
```



Les proportions code XHTML vs code Java sont sensiblement les mêmes

GetDateFmtJSTL.jspx

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:core="http://java.sun.com/jsp/jstl/core"
      xmlns:fmt="http://java.sun.com/jsp/jstl/fmt"
      xmlns="http://www.w3.org/1999/xhtml">

 <jsp:useBean id="date" class="java.util.Date" />

 <head><title>La date du jour</title></head>
 <body>
  <h2>Voici la date du jour :</h2>
  <p>
   <fmt:setLocale value="fr_CA" />
   <fmt:formatDate value="${date}" type="both" dateStyle="full"/>
  </p>
 </body>

</html>
```

On peut facilement imaginer les discussions entre le concepteur graphiquique et le programmeur

Pourrais-tu me programmer une balise qui fait ...

Forme déconseillée

- La construction suivante est maintenant déconseillée

```
<jsp:scriptlet>
    out.write( ( new java.util.Date() ).toString() );
</jsp:scriptlet>
```

Nouvelle tendance :
JSP Standard Tag Library (JSTL)

```
<html xmlns:core="http://java.sun.com/jsp/jstl/core"
      xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns="http://www.w3.org/1999/xhtml">

 <jsp:useBean id="date" class="java.util.Date" />

  <head>
   <title>La date du jour</title>
  </head>
  <body style="font-size:x-large">
   <h2>Voici la date du jour :</h2>
   <p>
    <core:out value="${date}" />
   </p>
  </body>
 </html>
```

JSP 2.0

- JSP a évolué au fil des années
- La présentation met l'accent sur **JSP 2.0** et les diverses technologies qui, lorsque bien utilisées, permettent une séparation accrue entre la programmation et la présentation
- Comment sortir la programmtion du document de présentation ?

Java Server Page (JSP)

- Afin de séparer présentation et programmation :

  1. **JSP Expression Language (EL)**

  2. **JSP Standard Tag Library (JSTL)**

  3. **JavaBeans**

- Technolgies instrumentales, interface en les documents JSP et Java

EL dans les documents JSP

JSTL + JavaBeans pour l'interface avec des classes externes

JavaServer Pages (JSP)

- Les documents JSP sont maintenant des **applications** (vocabulaires) **XML**
- Ce qui permet l'utilisation d'une foule d'outils, pour la validation (processeurs SAX et DOM) ou la transformation (processeurs XSLT) des documents JSP

En industrie, vous trouverez des exemples de pages dont la syntaxe n'est pas XML

JavaServer Pages (JSP)

- Ces documents sont généralement plus faciles à lire, mais aussi ils peuvent être développés à l'aide de logiciels graphiques de composition tels que **GoLive**, **Dreamweaver**, **Frontpage**, etc.

Remarques

- Dans un document JSP, les éléments du vocabulaire XHTML constituent le modèle
- Plusieurs autres vocabulaires, entre autres JSP et Core, servent à créer le contenu dynamique

Ressources

J2EE 1.4 Tutorial est une excellente source d'information, mais le PDF fait 1542 pages ...

- The J2EE 1.4 Tutorial [http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html] 2007
- The Jakarta Taglibs Project [http://jakarta.apache.org/taglibs/index.html] 2007
- JavaBeans(TM) Specification 1.01 [http://java.sun.com/products/javabeans/docs/spec.html] 2007

Resources

- Une ressource étonnamment brève et très utile
  - JavaServer Pages (JSP) v2.0 Syntax Reference [http://java.sun.com/products/jsp/syntax/2.0/syntaxref20.html] 2008-03-05