Programmation Web

JSP et Taglib

Les TagLibs

- La spécification des JSP prévoit la possibilité de définir ses **propres tags** XML

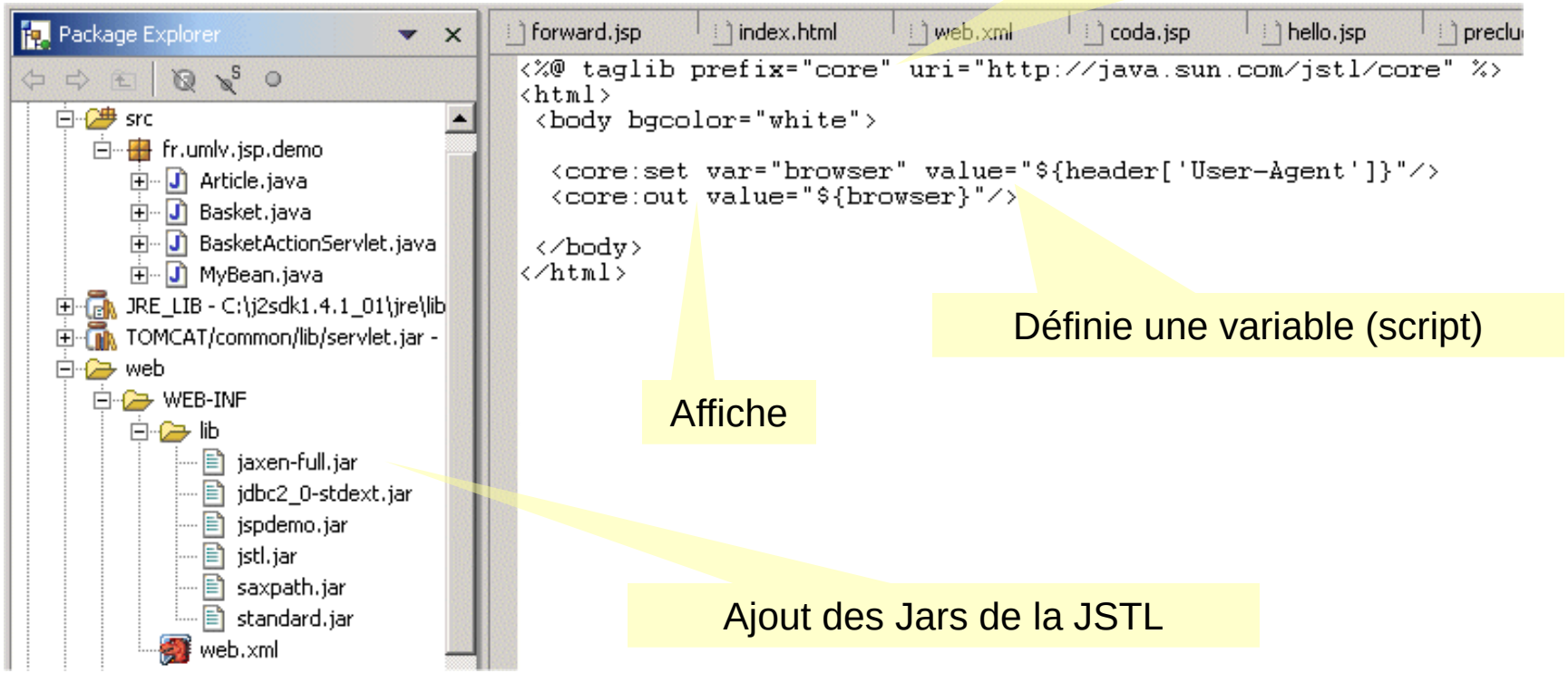
- Un ensemble de tag est regroupé au sein d'une **taglib** qui possède URI comme identifiant unique

JSP Standard Tag Library

- Ensemble de tags pré-programmés permettant d'effectuer les opérations standard
- Initiative Apache spécifiée via la JSR052
- Plusieurs parties :
  Core (http://java.sun.com/jstl/core),
  Internationalisation (http://java.sun.com/jstl/fmt),
  XML (http://java.sun.com/jstl/xml),
  SQL (http://java.sun.com/jstl/sql).

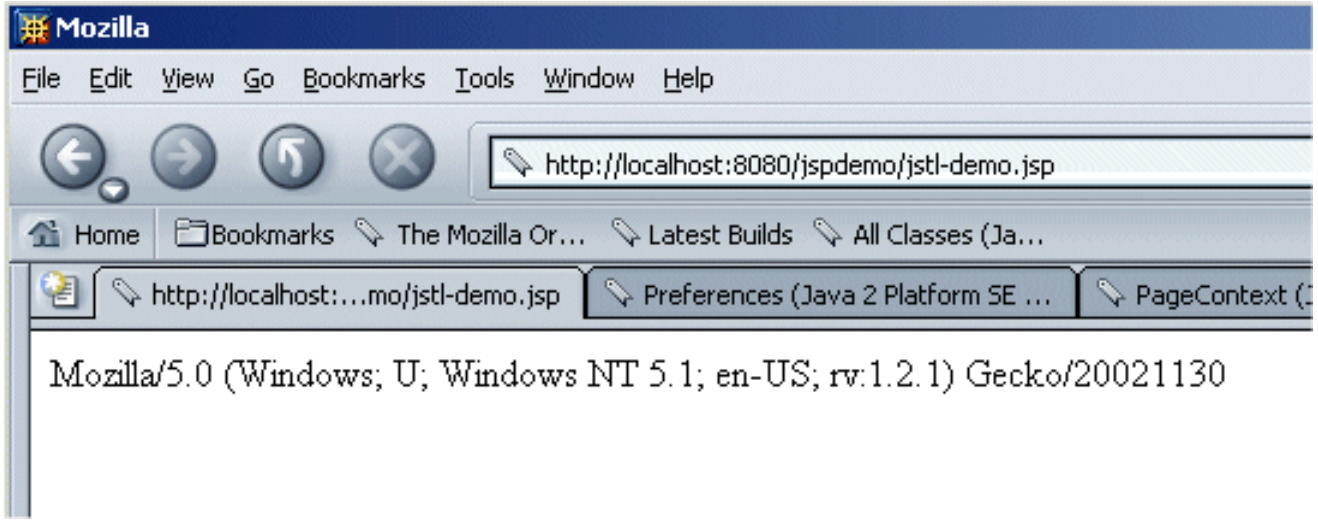
Hello JSTL

Affiche le champs **User-Agent** des entêtes



Hello JSTL

Affiche la version du browser



JSTL Expression Langage

Langage de script à l'intérieur des attributs des tags

```
<%@ taglib prefix="core" uri="http://java.sun.com/jstl/core" %>
<html>
 <body bgcolor="white">
  <core:set var="browser" value="${header['User-Agent']}"/>
  <core:out value="${browser}"/>
 </body>
</html>
```

```
<html>
 <body bgcolor="white">
  <%= request.getHeader("User-Agent") %>
</body>
</html>
```

JSTL Expression Langage (2)

${**name**} permet d'accéder à la variable **name** du scope (parcours les scopes)

Accéder à une propriété ${**name**.**prop**} ou ${**name**['**prop**']}

- appel sur l'objet **name** les méthodes get**Prop**/set**Prop.**
- dans le cas d'une *Map* les méthodes get("**prop**") et put("**prop**",value).
- Dans le cas des *List* ou des *tableaux*, ${**name**[0]) renvoie le premier élément.

JSTL Expression Langage (3)

Variables prédéfinies :

- **applicationScope** ensemble des variables disponible par ordre de visibilité
- **cookie** Collection de tous les cookies
- **header** entête HTTP vue comme une String
- **headerValues** entête vue comme une collection de String
- **initParam** Collection des paramètre d'initialisation
- **pageContext** objet représentant la page courante
- **pageScope** collection des variables de la page
- **param** paramètres sous forme de String
- **paramValues** paramètres sous forme d'une collection de String
- **requestScope** collection des variables de la requête
- **sessionScope** collection des variables de la session

Deux taglibs jumelles

- 2 langages de script :
  Run**T**ime Script:
  <%= article.getName() %>
  Expression **L**anguage Script :
  ${article.name}

- 2 taglibs :
  RT: http://java.sun.com/jstl/core_rt
  EL :http://java.sun.com/jstl/core

JSTL: core

- **Déclaration**
  core:set, core:declare, core:remove
- **Affichage**
  core:out
- **Condition**
  core:if, core:choose/when/otherwise, core:catch
- **Itération**
  core:forEach, core:forTokens

Définie une variable scopée

Déclare une variable locale

Retire une variable scopée

Affiche

Switch

Attrape les exceptions

Itère sur des portions de String découpée par des délimiteurs

Itère sur une collection ou un tableau

JSTL : core (exemple)

Affiche Rebonjour après reload

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

<html>
 <body bgcolor="white">

  <c:choose>
   <c:when test="${loggedIn}">
     <h1>Rebonjour</h1>
   </c:when>
   <c:otherwise>
     <h1>Bonjour</h1>
   </c:otherwise>
  </c:choose>

  <c:set var="loggedIn" scope="session" value="${true}"/>

   Please Reload me !!

 </body>
</html>
```

JSTL : headers
Mélange RT et EL Core taglibs

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="crt" uri="http://java.sun.com/jstl/core_rt" %>

<html>
 <body bgcolor="white">
  <h1>Headers</h1>
  <dl>
   <crt:forEach items="<%= request.getHeaderNames() %>" var="h">
    <dt><c:out value="${h}"/></dt>
    <dd><c:out value="${header[h]}"/></dd>
  </crt:forEach>

 </body>
</html>
```

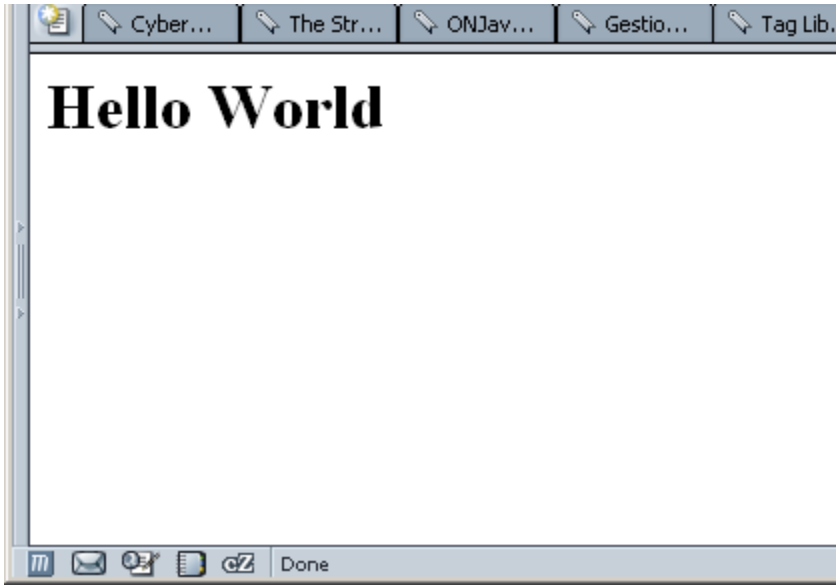
Écrire sa propre Taglib

1. Écrire le code des tags (implantant **Tag, BodyTag**)
2. Définir un Tag Library Descriptor (fichier XML contenant la définition des tags)
3. Référencer le TLD dans le web.xml
4. Déclaration par une directive de taglib lors de l’utilisation

Exemple HelloTag

Création d'un tag affichant « hello world »

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>

<html>
 <head>
  <title>Hello World Tag</title>
 </head>
 <body bgcolor="white">
  <h1><tgd:helloworld/></h1>
 </body>
</html>
```



Création de la classe

La classe implémente **Tag** (**TagSupport** implémente **Tag**)

```
package fr.umlv.tagdemo;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class HelloWorldTag extends TagSupport {
   public int doStartTag() throws JspException {
      try {
         pageContext.getOut().print("Hello World");
      } catch (Exception e) {
         throw new JspTagException(e.getMessage());
      }
      return SKIP_BODY;
   }
   public int doEndTag() {
      return EVAL_PAGE;
   }
}
```

Ne pas évaluer l'intérieur du tag

Évalue la suite de la page

Cycle de vie d’un Tag

- Initialisation
  setPageContext(), fixe le contexte de page
  setParent(), fixe le tag père
  set*(value) initialise les attributs
- **doStartTag**() appelée à la balise ouvrante
- **doEndTag**() appelée à la balise fermante
- **release**() libère les ressources

Cycle de vie d'un Tag (2)

Valeurs de retour possibles pour :

- **doStartTag**()
  EVAL_BODY_INCLUDE le contenu entre les balises est évalué
  SKIP_BODY le contenu n'est pas évalué
- **doEndTag**()
  EVAL_PAGE continue l'évaluation de la page
  SKIP_PAGE stoppe l'évaluation de la page

Tag Library Descriptor

Fichier XML de description des tags

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
        "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
 <tlibversion>1.0</tlibversion>
 <jspversion>1.2</jspversion>
 <shortname>tgd</shortname>
 <uri>http://www.univ-mlv.fr/taglibs/tagdemo</uri>
 <info>Demo Tag Library</info>

 <tag>
  <name>helloworld</name>
  <tagclass>fr.umlv.tagdemo.HelloWorldTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>print helloworld</info>
 </tag>
</taglib>
```

URI associée à la taglib

Nom du tag

Nom de la classe

Le contenu est ignoré
doStart() doit renvoyer
SKIP_BODY

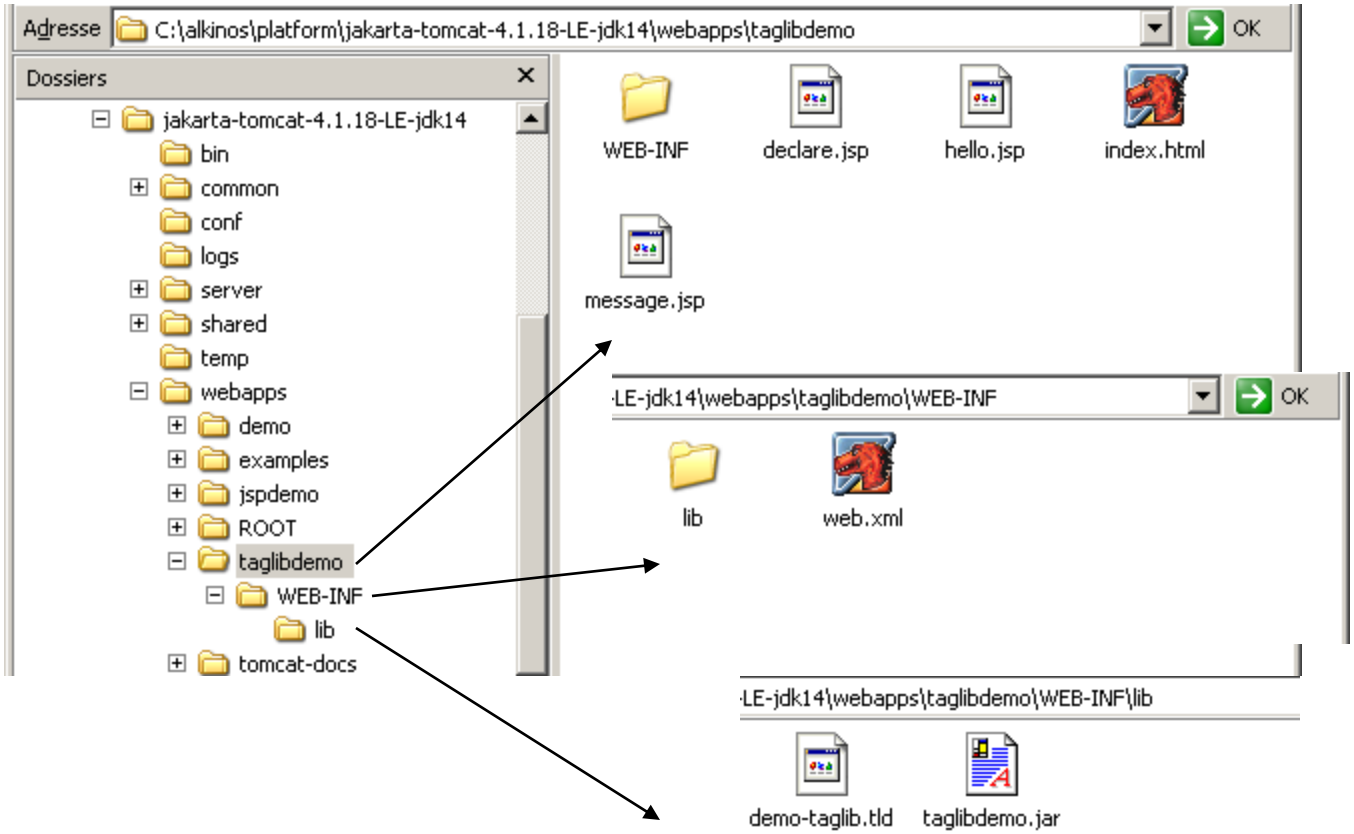
Déclaration du TLD

Déclaration du fichier de la taglib dans le web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>Demo de taglib</display-name>
  <description>Ceci est une serie de tag de demo</description>

  <taglib>
    <taglib-uri>http://www.univ-mlv.fr/taglibs/tagdemo</taglib-uri>
    <taglib-location>
       /WEB-INF/lib/demo-taglib.tld
    </taglib-location>
  </taglib>

</web-app>
```

Fichiers sur le disque

Les différents fichiers sur le disque



Tag avec attributs

Lors de l'initialisation d'un tag pour chaque attribut du tag la méthode setAttributeName(String value) est appelée

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>

<html>
 <head>
  <title>Message Tag</title>
 </head>
 <body bgcolor="white">
  <tgd:message value="helloworld"/><br>
 </body>
</html>
```

Tag avec attributs

Appel la méthode setValue("helloworld")

```
public class MessageTag extends TagSupport {
   public int doStartTag() throws JspException {
      try {
         pageContext.getOut().print(value);
      } catch (Exception e) {
         throw new JspTagException(e.getMessage());
      }
      return SKIP_BODY;
   }
   public int doEndTag() {
      return EVAL_PAGE;
   }

   public void setValue(String value) {
     this.value=value;
   }
   private String value;
}
```

Déclarer les attributs

- Les attributs sont déclarés dans le TLD

```
<tag>
  <name>message</name>
  <tagclass>fr.umlv.tagdemo.MessageTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>print a message passed as attribute</info>
  <attribute>
    <name>value</name>
    <required>true</required>
    <rtexprvalue>false</rtexprvalue>
  </attribute>
 </tag>
```

Nom de l'attribut

Attribut obligatoire ??

La valeur peut être le résultat d'un script

Attribut des Tags & Script

Un attribut peut prendre en paramètre une valeur calculée lors de l'exécution

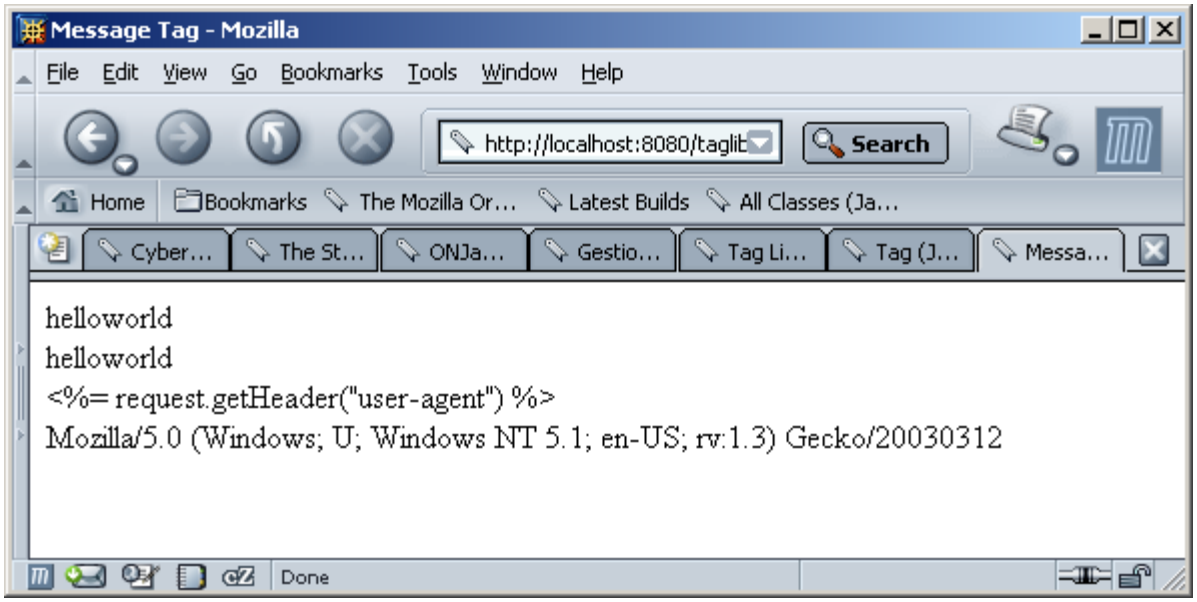
```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>

<html>
 <head>
  <title>Message Tag</title>
 </head>
 <body bgcolor="white">
  <tgd:message value="helloworld"/>
  <br>
  <tgd:rt-message value="helloworld"/>
  <br>
  <tgd:message value="<%= request.getHeader("user-agent") %>"/>
  <br>
  <tgd:rt-message value="<%= request.getHeader("user-agent") %>"/>
 </body>
</html>
```

rtexprvalue=false

rtexprvalue=true

Attribut des Tags & Script (2)

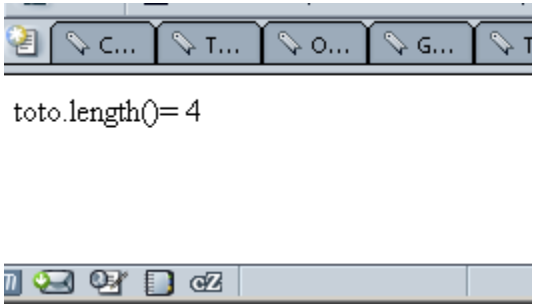
Si **rtexprvalue=false** alors pas d'évaluation des expressions à l'exécution



Déclaration de variables dans un Tag

Comment déclarer un variable locale ?

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>

<html>
 <head>
  <title>Declare Tag</title>
 </head>
 <body bgcolor="white">
  <tgd:declare name="variable" type="java.lang.String" value="toto"/>

  <%= variable %>.length()= <%= variable.length() %>
 </body>
</html>
```



Déclaration de variables dans un tag

La valeur est manipuler en utilisant le **pageContext**

```
public class DeclareTag extends TagSupport {
   public int doStartTag() throws JspException {
     return SKIP_BODY;
   }
   public int doEndTag() {
     pageContext.setAttribute(name, transtype(value,type));
     return EVAL_PAGE;
   }
   private Object transtype(
     String value, Class type) {
     return value;
   }
private String name,value;
private Class type;
```

```
   public void setType(String className) {
     try {
       type=Class.forName(className);
     } catch (ClassNotFoundException e) {
       this.type=null;
     }
   }
   public void setName(String name) { this.name=name; }
   public void setValue(String value) { this.value=value; }
}
```

TagExtraInfo

Permet de définir des variables pendant l'étape de traduction

```
public class DeclareTagExtraInfo extends TagExtraInfo {
  public VariableInfo[] getVariableInfo(TagData data) {
    return new VariableInfo[] {
      new VariableInfo(
        data.getAttributeString("name"),
        data.getAttributeString("type"),
        true,
        VariableInfo.AT_END)
    };
  }
}
```

Info sur le tag

Déclaration de la variable

Nom de la variable

Type de la variable

NESTED : entre les balises ouvrantes et fermantes
AT_BEGIN : de la balise ouvrantes à la fin de fichier
AT_END : de la balise fermante à la fin de fichier

Déclaration du TagExtraInfo

Déclaration dans le fichier TLD

```
<tag>
  <name>declare</name>
  <tagclass>fr.umlv.tagdemo.DeclareTag</tagclass>
  <teiclass>fr.umlv.tagdemo.DeclareTagExtraInfo</teiclass>
  <bodycontent>empty</bodycontent>
  <info>declare a variable with page scope</info>
  <attribute>
    <name>name</name>
    <required>true</required>
    <rtexprvalue>false</rtexprvalue>
  </attribute>
  <attribute>
    <name>value</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
  …
</tag>
```

Traduction d’un Tag
Traduction de la JSP en servlet

```
fr.umlv.tagdemo.DeclareTag _jspx_th_tgd_declare_0 =
   (fr.umlv.tagdemo.DeclareTag) _jspx_tagPool_tgd_declare_value_type_name.get(
      fr.umlv.tagdemo.DeclareTag.class);
_jspx_th_tgd_declare_0.setPageContext(pageContext);
_jspx_th_tgd_declare_0.setParent(null);
_jspx_th_tgd_declare_0.setName("variable");
_jspx_th_tgd_declare_0.setType("java.lang.String");
_jspx_th_tgd_declare_0.setValue("toto");
int _jspx_eval_tgd_declare_0 = _jspx_th_tgd_declare_0.doStartTag();
if (_jspx_th_tgd_declare_0.doEndTag() == javax.servlet.jsp.tagext.Tag.SKIP_PAGE)
  return;
_jspx_tagPool_tgd_declare_value_type_name.reuse(_jspx_th_tgd_declare_0);
java.lang.String variable = null;
variable = (java.lang.String) pageContext.findAttribute("variable");
out.write("\r\n  \r\n  ");
out.print( variable );
out.write(".length()= ");
out.print( variable.length() );
```

Évaluation conditionnelle

On veut évaluer le contenu entre les deux balises **if** si la valeur de l'attribut est vrai

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>
<html>
 <body bgcolor="white">
 <tgd:if value="true">
    test ok
  </tgd:if>
  <tgd:if value="false">
    test wrong
  </tgd:if>
 </body>
</html>
```

Évaluation conditionnelle

doStartTag() renvoie une valeur différentes

```
public class IfTag extends TagSupport {
  public int doStartTag() throws JspException {
    return (condition)?EVAL_BODY_INCLUDE:SKIP_BODY;
  }
  public int doEndTag() {
    return EVAL_PAGE;
  }
  public void setValue(String value) {
    condition=Boolean.valueOf(value).
      booleanValue();
  }
  private boolean condition;
}
```

```
<tag>
  <name>if</name>
  <tagclass>fr.umlv.tagdemo.IfTag</tagclass>
  <teiclass>fr.umlv.tagdemo.IfTagExtraInfo</teiclass>
  <bodycontent>JSP</bodycontent>
  <info>conditional execution</info>
  <attribute>
    <name>value</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

Validation des attributs

Assurer à la traduction que le tag **if** prend bien une valeur booléenne

Est appelée lors de la traduction pour valider les arguments

```
public class IfTagExtraInfo extends TagExtraInfo {
  public boolean isValid(TagData data) {
    Object value = data.getAttribute("value");
    if (value == TagData.REQUEST_TIME_VALUE)
      return true;

    String text=value.toString().toLowerCase();
    return text.equals("true") || text.equals("false");
  }
}
```

Indique une expression calculée à l'exécution

Validation des attributs (2)

Permet uniquement de valider les valeurs lors de la traduction

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>
<html>
 <body bgcolor="white">
  <tgd:if value="false">
    test wrong
  </tgd:if>
  <tgd:if value="hello">
    test wrong
  </tgd:if>
  <% String test="hello"; %>
  <tgd:if value="<%= test %>">
    test ok
  </tgd:if>
 </body>
</html>
```

Traduction ok

Erreur à la traduction

Ok, pas d'erreur de traduction

Les BodyTag

Un BodyTag permet :

- d’effectuer des itérations sur le contenu
- d’effectuer des post-traitement sur le contenu

La valeur renvoyer par **doStartTag()**
permet de choisir

Cycle de vie d'un BodyTag (1)

Itérer sur le contenu (body) :

- Initialisation (**set***)
- Si **doStartTag**() renvoie EVAL_BODY_INCLUDE
- **doInitBody**()
- Évalue le contenu du body
- Boucle sur le **doAfterBody**() si EVAL_BODY_AGAIN
- **doEndTag**();
- **release**();

Cycle de vie d’un BodyTag (2)

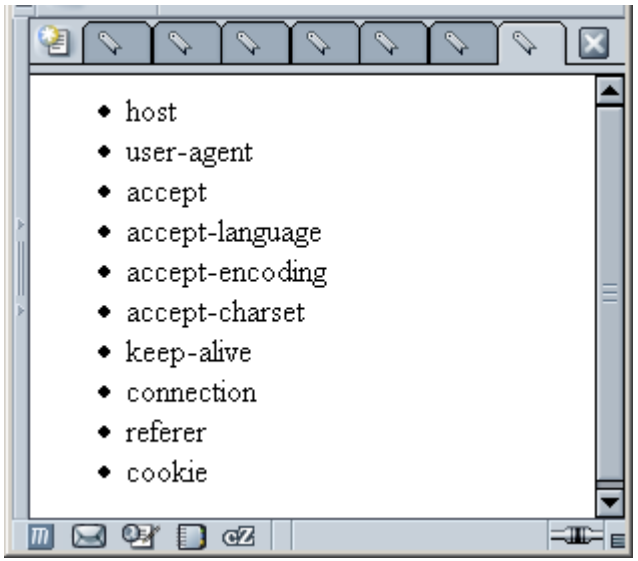
Bufferiser le contenu (body) :

- Initialisation (**set***)
- Si **doStartTag**() renvoie EVAL_BODY_BUFFERED
- out = pageContext.pushBody(), setBodyContent(out);
- **doInitBody**()
- Évalue le contenu du body
- Boucle sur le **doAfterBody**() si EVAL_BODY_AGAIN
- **doEndTag**();
- pageContext.popBody();
- **release**();

Tag d'Itération

On souhaite utiliser un tag pour itérer sur une collection

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>
<html>
 <body bgcolor="white">
  <ul>
    <tgd:iterate collection="<%= request.getHeaderNames() %>">
      <li><tgd:item/></li>
    </tgd:iterate>
  </ul>
 </body>
</html>
```



Tag d'Itération (2)

Le TLD et le TagExtraInfo correspondant

```
<tag>
 <name>iterate</name>
 <tagclass>fr.umlv.tagdemo.IterateTag</tagclass>
 <teiclass>fr.umlv.tagdemo.IterateTagExtraInfo</teiclass>
 <bodycontent>JSP</bodycontent>
 <info>iterate over a collection</info>
 <attribute>
  <name>collection</name>
  <required>true</required>
  <rtexprvalue>true</rtexprvalue>
 </attribute>
</tag>
<tag>
 <name>item</name>
 <tagclass>fr.umlv.tagdemo.ItemTag</tagclass>
 <bodycontent>empty</bodycontent>
 <info>print an item</info>
</tag>
```

Le tag regarde le contenu

```
public class IterateTagExtraInfo extends TagExtraInfo {
  public VariableInfo[] getVariableInfo(TagData data) {
    return new VariableInfo[] {
      new VariableInfo(
        "item",
        "java.lang.Object",
        true,
        VariableInfo.NESTED)
      }; } }
```

Nom de la variable

Type de la variable

Variable locale au body

Tag d'Itération (3)

Code du tag iterate

```
public class IterateTag extends BodyTagSupport {
  public int doStartTag() throws JspException {
    if (!iterator.hasNext()) return SKIP_BODY;
    pageContext.setAttribute("item", iterator.next());
    return EVAL_BODY_INCLUDE;
  }
  public int doAfterBody() throws JspException {
    if (!iterator.hasNext()) return SKIP_BODY;
    pageContext.setAttribute("item", iterator.next());
    return EVAL_BODY_AGAIN;
  }
  public void setCollection(Object value) {
    if (value instanceof Iterator) {
      iterator=(Iterator)value;
    } else throw new RuntimeException("bad collection "+value);
  }
  private Iterator iterator;
}
```

Implémente **BodyTag**

Exécute le contenu du body

Ré-exécute le contenu du body

Tag d'Itération (4)

Code du Tag item

```
public class ItemTag extends TagSupport {
  public int doStartTag() throws JspException {
    try {
      pageContext.getOut().print(pageContext.getAttribute("item"));
    } catch (IOException e) {
      throw new JspException(e.getMessage());
    }
    return SKIP_BODY;
  }

  public int doEndTag() {
    return EVAL_PAGE;
  }
}
```

Tag Library Descriptor

```
<tag>
  <name>item</name>
  <tagclass>fr.umlv.tagdemo.ItemTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>print an item</info>
 </tag>
```

Post traitement du contenu

Appliquer une feuille de style à un contenu

```
<%@ taglib uri="http://www.univ-mlv.fr/taglibs/tagdemo" prefix="tgd" %>

<html>
 <head>
  <title>XSLT Tag</title>
 </head>
 <body bgcolor="white">

  <tgd:xslt stylesheet="helloworld.xsl">
    <helloworld/>
  </tgd:xslt>

 </body>
</html>
```

```
<?xml version="1.0" encoding="latin1"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
  <xsl:output method="xml"
              encoding = "isolatin1"
              omit-xml-declaration = "yes"
              indent = "yes"/>
  <xsl:template match="helloworld">
    hello world !!
  </xsl:template>
</xsl:stylesheet>
```

Post traitement du contenu (2)

EVAL_BODY_BUFFERED permet de bufferiser le contenu

Bufferise le contenu

Il est interdit d'utiliser **out** ici

```
public class XSLTag extends BodyTagSupport {
  public int doStartTag() throws JspException {
    return EVAL_BODY_BUFFERED;
  }
  public int doAfterBody() throws JspException {
    return SKIP_BODY;
  }
  public int doEndTag() throws JspException {
    Reader reader=getBodyContent().getReader();
    try {
      transformer.transform(
        new StreamSource(reader),
        new StreamResult(pageContext.getOut()));
    } catch (TransformerException e) {
      throw new JspException(e.getMessage());
    }
    return EVAL_PAGE;
  }

  public void setStylesheet(String file)
    throws TransformerConfigurationException {
    TransformerFactory factory=
      TransformerFactory.newInstance();

    Templates template=factory.newTemplates(
      new StreamSource(new File(
        pageContext.getServletContext().
          getRealPath(file))));
    transformer=template.newTransformer();
  }

  private Transformer transformer;
}
```

Extensions de JSP 2.0

- Include automatique preclude/coda.
- Script à l'intérieur des attributs des beans.
- Fragment JSP (sorte de templates, tiles de struts)
- Interface **SimpleTag** pour les Tags

Pas de code Java dans les JSP

- On mélange l’interface graphique et le code métier
- Les erreurs de syntaxes sont détectés lors du déploiement
- Le code est dur à lire

Récapitulatif

- Introduites avec la version JSP 1.1
- **Avantages**
  - Étendre les balises JSP standards
  - Balises spécifiques à un cas d'usage
  - Réduire l'utilisation des scriplets
  - Améliorer la lisibilité de la page JSP
  - Libérer les concepteurs de pages du code Java
- **Mise en oeuvre**
  - Classe Java Handle avec la librairie *javax.servlets.jsp.tagext*
  - Fichier file.tld descripteur du tag
  - Page JSP utilisant la nouvelle balise