# CSI 3540

Structures, techniques et normes du Web

## Séparer la programmation de la présentation à l’aide de JSP

**Objectifs :**

1. Concevoir des documents JSP
2. Se familiariser ave le cycle de développement d’une application Web
3. Déployer une application Web

**Lectures :**

- Web Technologies (2007) § 8
  Pages 463-472

## Plan

1. Bibliothèques de balises JSTL
2. MVC et JSP

## master.jspx

jsp:include

# main.jspx

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:core="http://java.sun.com/jsp/jstl/core"
      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>jsp:include</title>
  </head>
  <body>
    <jsp:include page="${ param.page }" />
  </body>
</html>
```

# GetDate.jspx

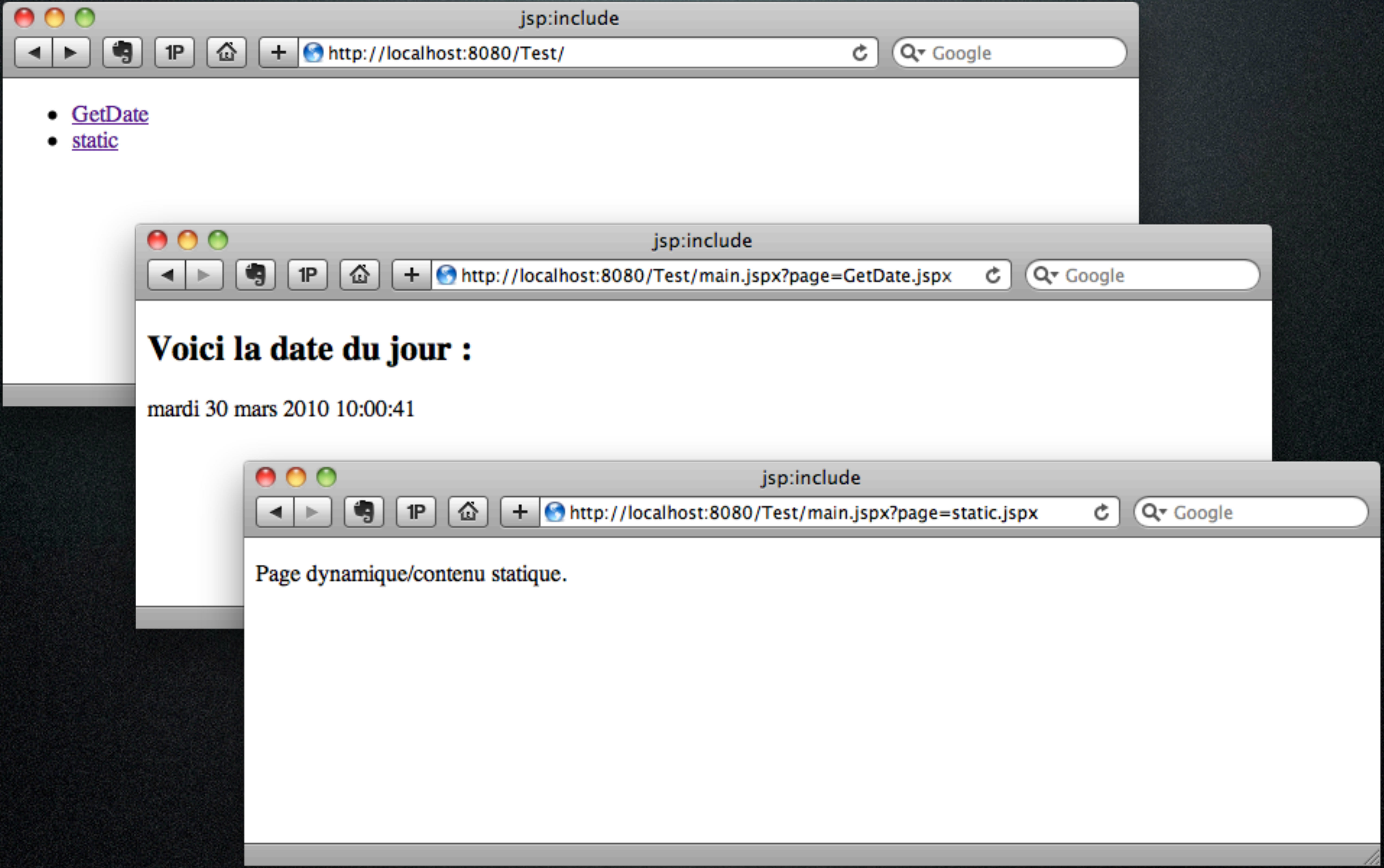
```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns:core="http://java.sun.com/jsp/jstl/core"
          xmlns:fmt="http://java.sun.com/jsp/jstl/fmt"
          xmlns="http://www.w3.org/1999/xhtml"
          version="2.0" >

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

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

</jsp:root>
```

## static.jspx

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns="http://www.w3.org/1999/xhtml"
          version="2.0">

  <p>
    Page dynamique/contenu statique.
  </p>

</jsp:root>
```

## index.html

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>jsp:include</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <ul>
      <li><a href="main.jspx?page=GetDate.jspx">GetDate</a></li>
      <li><a href="main.jspx?page=static.jspx">static</a></li>
    </ul>
  </body>
</html>
```



jsp:include

http://localhost:8080/Test/

- GetDate
- static

jsp:include

http://localhost:8080/Test/main.jspx?page=GetDate.jspx

## Voici la date du jour :

mardi 30 mars 2010 10:00:41

jsp:include

http://localhost:8080/Test/main.jspx?page=static.jspx

Page dynamique/contenu statique.

```
$ telnet localhost 8080
Trying ::1...
Connected to localhost.
Escape character is '^]'.
GET /Test/GetDate.jspx HTTP/1.1
HOST: localhost

HTTP/1.1 200 OK
X-Powered-By: JSP/2.1
Server: GlassFish v3
Set-Cookie: JSESSIONID=f7b2322d1b0d655af0215c3a2c0a; Path=/Test
Content-Type: text/xml;charset=UTF-8
Content-Language: fr-CA
Content-Length: 66
Date: Tue, 30 Mar 2010 14:30:35 GMT

<h2>Voici la date du jour :</h2><p>mardi 30 mars 2010 10:30:35</p>
```

# jsp:directive.include

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

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

  <jsp:directive.include file="copyright.jspf" />

  <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>jsp:include</title>
  </head>
  <body>
    <jsp:include page="${ param.page }" />
  </body>
</html>
```

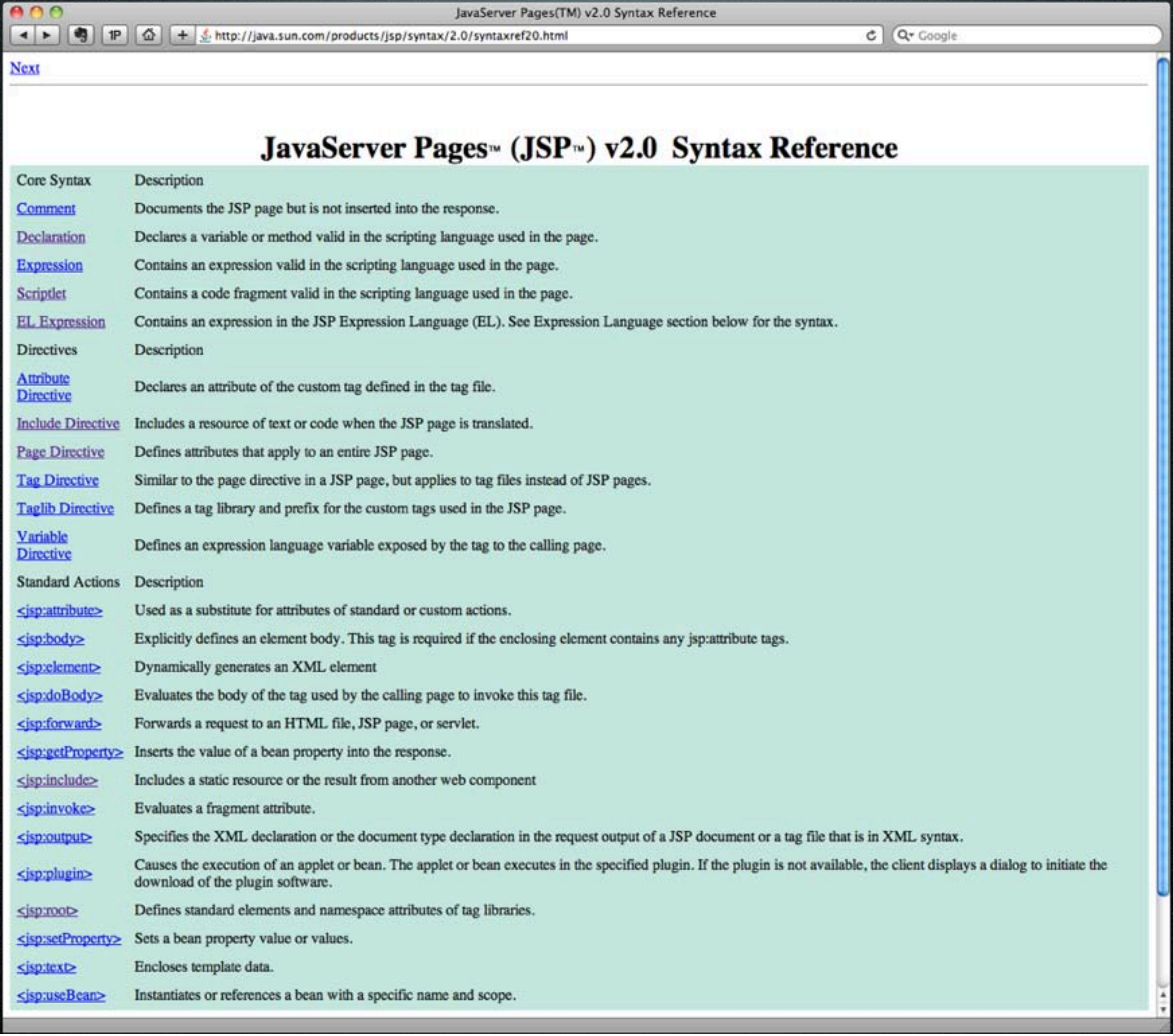
copyright.jspf :

```
<!-- (C) Marcel Turcotte 2010 -->
```

```
<html xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:core="http://java.sun.com/jsp/jstl/core"
      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>jsp:include</title>
  </head>
  <body>

    <core:choose>
      <core:when test="${ param.page eq 'GetDate.jspx' }" >
         <jsp:include page="${ param.page }" />
      </core:when>
      <core:when test="${ param.page eq 'static.jspx' }" >
         <jsp:include page="${ param.page }" />
      </core:when>
      <core:otherwise>
         Erreur!
      </core:otherwise>
    </core:choose>

  </body>
</html>
```

## jsp:include

```
<jsp:include page="scripts/login.jsp">
   <jsp:param name="username" value="jsmith" />
</jsp:include>
```


JavaServer Pages(TM) v2.0 Syntax Reference
http://java.sun.com/products/jsp/syntax/2.0/syntaxref20.html
Google
Next

# JavaServer Pages™ (JSP™) v2.0 Syntax Reference

| Core Syntax | Description |
|---|---|
| Comment | Documents the JSP page but is not inserted into the response. |
| Declaration | Declares a variable or method valid in the scripting language used in the page. |
| Expression | Contains an expression valid in the scripting language used in the page. |
| Scriptlet | Contains a code fragment valid in the scripting language used in the page. |
| EL Expression | Contains an expression in the JSP Expression Language (EL). See Expression Language section below for the syntax. |
| Directives | Description |
| Attribute Directive | Declares an attribute of the custom tag defined in the tag file. |
| Include Directive | Includes a resource of text or code when the JSP page is translated. |
| Page Directive | Defines attributes that apply to an entire JSP page. |
| Tag Directive | Similar to the page directive in a JSP page, but applies to tag files instead of JSP pages. |
| Taglib Directive | Defines a tag library and prefix for the custom tags used in the JSP page. |
| Variable Directive | Defines an expression language variable exposed by the tag to the calling page. |
| Standard Actions | Description |
| <jsp:attribute> | Used as a substitute for attributes of standard or custom actions. |
| <jsp:body> | Explicitly defines an element body. This tag is required if the enclosing element contains any jsp:attribute tags. |
| <jsp:element> | Dynamically generates an XML element |
| <jsp:doBody> | Evaluates the body of the tag used by the calling page to invoke this tag file. |
| <jsp:forward> | Forwards a request to an HTML file, JSP page, or servlet. |
| <jsp:getProperty> | Inserts the value of a bean property into the response. |
| <jsp:include> | Includes a static resource or the result from another web component |
| <jsp:invoke> | Evaluates a fragment attribute. |
| <jsp:output> | Specifies the XML declaration or the document type declaration in the request output of a JSP document or a tag file that is in XML syntax. |
| <jsp:plugin> | Causes the execution of an applet or bean. The applet or bean executes in the specified plugin. If the plugin is not available, the client displays a dialog to initiate the download of the plugin software. |
| <jsp:root> | Defines standard elements and namespace attributes of tag libraries. |
| <jsp:setProperty> | Sets a bean property value or values. |
| <jsp:text> | Encloses template data. |
| <jsp:useBean> | Instantiates or references a bean with a specific name and scope. |

## JSTL

JSP Standard Tag Library

## JSP Standard Tag Library (JSTL)

- **JSTL** fournit un **ensemble de balises** implémentant plusieurs traitements communs aux applications Web
- **JSTL** permet la **création de nouvelles balises**
- **JSTL** est utilisé en conjonction avec **EL** (JSP Expression Language)
- **Ainsi, tout le code Java est éliminé des documents JSP**

## JSTL

- Défini 5 espaces de nommages :
  - Core (**core**) : variables, flot, etc.
  - Traitements XML/XSLT (**xml**)
  - Fonctions (**functions**)
  - Banque de données (**sql**)
  - Internationalisation (**fmt**)
- **http://java.sun.com/jsp/jstl/namespace**

# JSTL et Tomcat 6

- JSTL est maintenant inclu dans la version «**Enterprise**» de Java (Java EE 5 platform) ainsi que **GlassFish**
- Sinon,
  - Télécharger la distribution : http://jakarta.apache.org/taglibs/
  - Copier jstl.jar et standard.jar dans le répertoire web/WEB-INF/lib de chaque application!

C'est plus prudent d'inclure ces librairies dans son application afin qu'on puisse déployer l'application sur tout serveur

## JSTL XML

```
<c:import var="rssFeed" url="http://slashdot.org/slashdot.rdf"/>
<c:import var="rssToHtml" url="/WEB-INF/xslt/rss2html.xsl"/>
<x:transform xml="${rssFeed}" xslt="${rssToHtml}"/>
```

## JSTL:core

- Il faut déclarer cet espace de nommage :

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

# JSTL:core

- **set** : assigne une valeur à une variable
- `<core:set var="v" scope="application" var="${ v+1 }" />`
- **Portée** : page, request, session, application
- `<core:set target="obj" property="key" var="${ expr }" />`
- ou **obj** est un objet de type **Map** ou **JavaBeans**
- 
```
<core:set var="v" >
   <core:out value="${ expr }" >
</core:set>
```

La portée (scope) est optionelle, page par défaut

var et scope sont des littéraux pas des expressions

## JSTL:core

- **remove** : détruit une variable
- Conditionel : **if**

```
<core:if test="${ expr }" >
   ...
</core:if>
```

- Exemple :

```
<core:if test="${ visits eq 1000000 }" >
   You win a price!
</core:if>
```

## JSTL:core

- Conditionel : <u>**choose**</u>

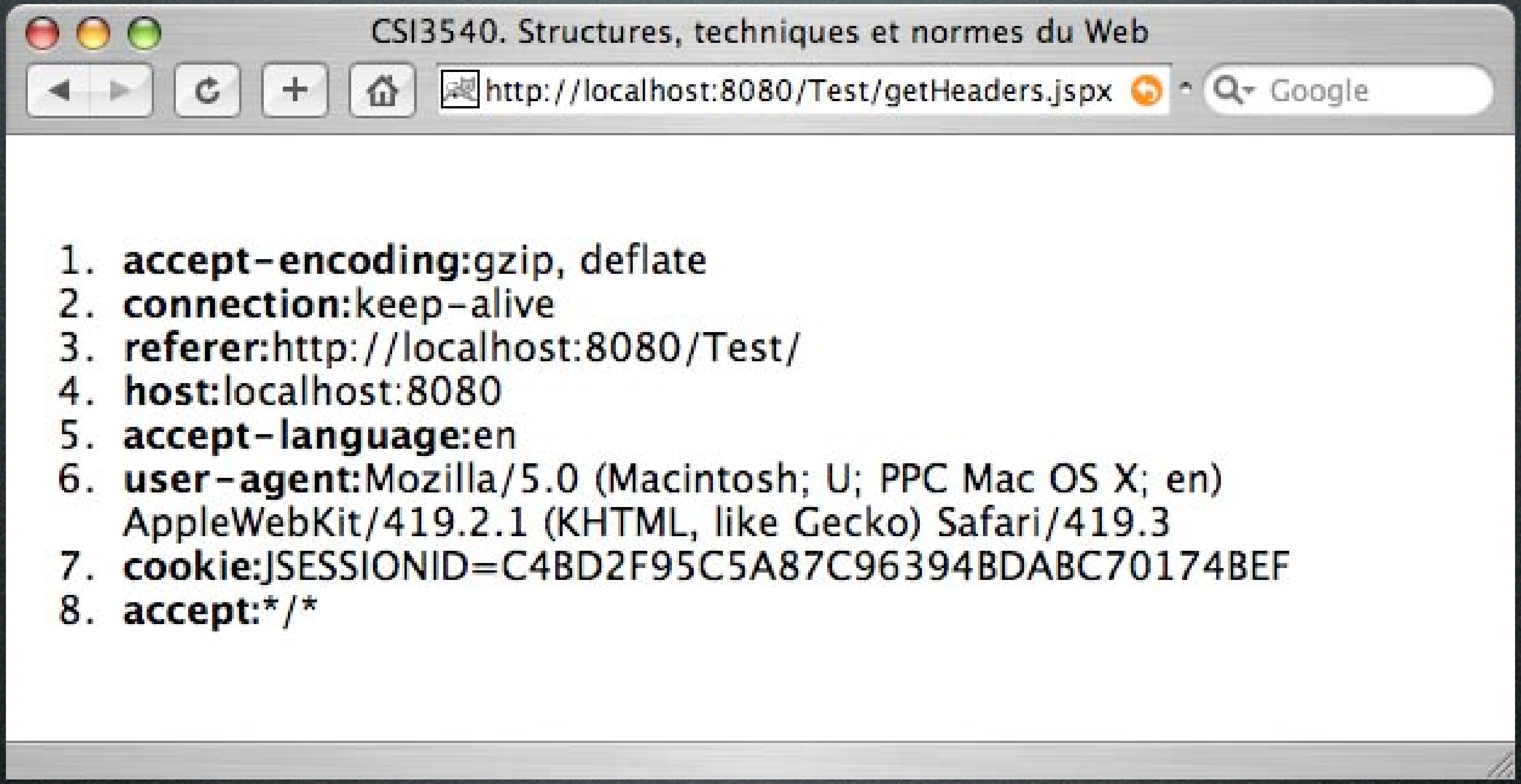
```
<core:choose>
  <core:when test="${ visits eq 1 }" >
    Thank you for selecting brand X!
  </core:when>
 <core:when test="${ visits eq 2 }" >
   Welcome back!
 </core:when>
 <core:otherwise>
   You are a valued customer!
 </core:otherwise>
</core:choose>
```

# forEach

- Afin de voir le contenu des objets implicites ...

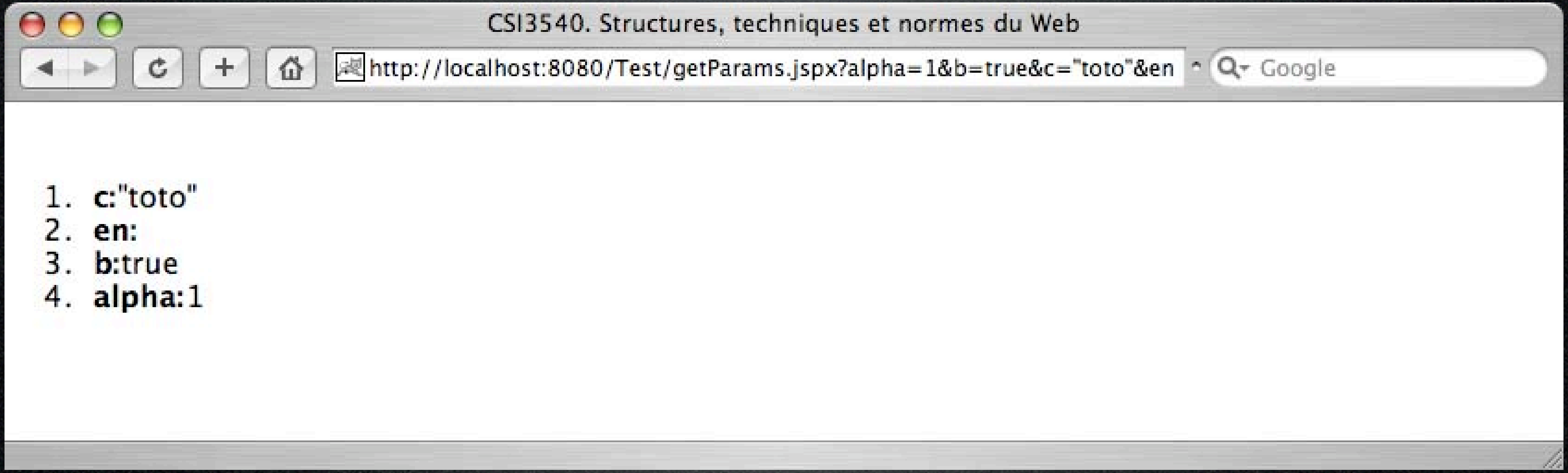
```
<ol>
  <core:forEach var="h" items="${ header }">
  <li>
    <strong>
      <core:out value="${ h.key }:"/>
    </strong>
    <core:out value="${ h.value }"/>
  </li>
  </core:forEach>
</ol>
```

## Exemple


CSI3540. Structures, techniques et normes du Web

http://localhost:8080/Test/getHeaders.jspx

1. **accept-encoding:**gzip, deflate
2. **connection:**keep-alive
3. **referer:**http://localhost:8080/Test/
4. **host:**localhost:8080
5. **accept-language:**en
6. **user-agent:**Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/419.2.1 (KHTML, like Gecko) Safari/419.3
7. **cookie:**JSESSIONID=C4BD2F95C5A87C96394BDABC70174BEF
8. **accept:***/*


CSI3540. Structures, techniques et normes du Web

http://localhost:8080/Test/getParams.jspx?alpha=1&b=true&c="toto"&en

1. **c:**"toto"
2. **en:**
3. **b:**true
4. **alpha:**1

## forEach

- S'utlise pour des objets de type **Map**, **Collection**, **Iterator**, ou **Enumeration**

```
<core:forEach var="v" items="${ expr }">
      ...
</core:forEach>

<core:forEach var="v" items="${ expr }" start="1" items="5" >
      ...
</core:forEach>
```

## forEach

- Deuxième forme de l'élément forEach

```
<core:forEach var="i" begin="1" end="10" step="2" >
  ${ i }
</core:forEach>
```

## core:out

- Lorque l'on génère dynamiquement du code XML (XHTML), il faut s'assurer que le code est valide, en particulier, il faut remplacer < et & par les entités équivalentes (&lt; et &amp;)
- C'est justement ce que **out** fait pour nous
- 
```
<core:set var="v" >
  <core:out value="${ expr }" >
</core:set>
```

## core:url

- Création d'URLs lors de l'exécution

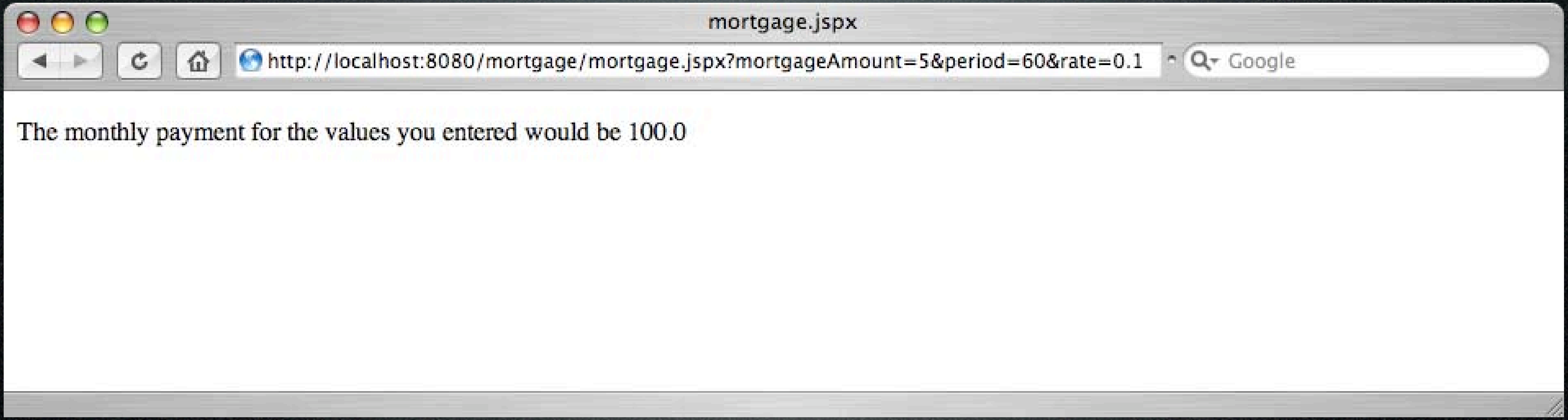
-
  ```
  <core:url value="/path" >
    <core:param name="user" value="Marcel Turcotte" >
  </core:ulr>
  ```
- /myapp/path?user=Marcel+Turcotte

## Remarques

- On devrait utiliser **JSTL** pour de simples calculs liés à la présentation
- Les autres calculs devraient être relégués aux **JavaBeans** et à d’autres Servlets (à suivre)

```
package csi3540;

public class Mortgage {

    private double amount;
    private int nMonths;
    private double intRate;

    public void setAmount( double amount ) {
        this.amount = amount;
    }
    public void setMonths( int nMonths ) {
        this.nMonths = nMonths;
    }
    public void setRate( double intRate ) {
        this.intRate = intRate;
    }
    public double getPayment() {
        ...
    }
}
```

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

 <head>
  <title>mortgage.jspx</title>
 </head>
 <body>
    <p>The monthly payment for the values you entered would be
    <jsp:useBean id="calc" class="csi3540.Mortgage" scope="page" />

    <c:set target="${calc}" property="amount" value="${param.MotgageAmount}" />
    <c:set target="${calc}" property="months" value="${param.period}" />
    <c:set target="${calc}" property="rate" value="${param.rate}" />

    ${calc.payment}
    </p>

 </body>
</html>
```



```
http://localhost:8080/mortgage/mortgage.jspx?mortgageAmount=5&period=60&rate=0.1
```

The monthly payment for the values you entered would be 100.0

## Créer de nouvelles balises

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:csi3540="urn:jsptagdir:/WEB-INF/tags"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  ...
  <head>
    <title>mortgage.jspx</title>
  </head>
  <body>
    <p>The monthly payment for the values you entered would be
      <csi3540:mortgage amount="${ param.mortgageAmount }"
                        period="${ param.period }"
                        rate="${ param.rate }" />
    </p>
  </body>
</html>
```

# WEB-INF/tags/mortgage.tagx

```
<jsp:root version="2.0"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">

 <jsp:directive.attribute name="amount" required="true" />
 <jsp:directive.attribute name="period" required="true" />
 <jsp:directive.attribute name="rate" required="true" />

 <jsp:useBean id="calc" class="csi3540.Mortgage" scope="page" />

     <c:set target="${calc}" property="amount" value="${amount}" />
     <c:set target="${calc}" property="months" value="${period}" />
     <c:set target="${calc}" property="rate" value="${rate}" />

     ${calc.payment}

</jsp:root>
```

## Définir de nouvelles fonctions

- Java EE 5 Tutorial page 166

## Model-View-Controller

Modèle de conception

## Model-View-Controller

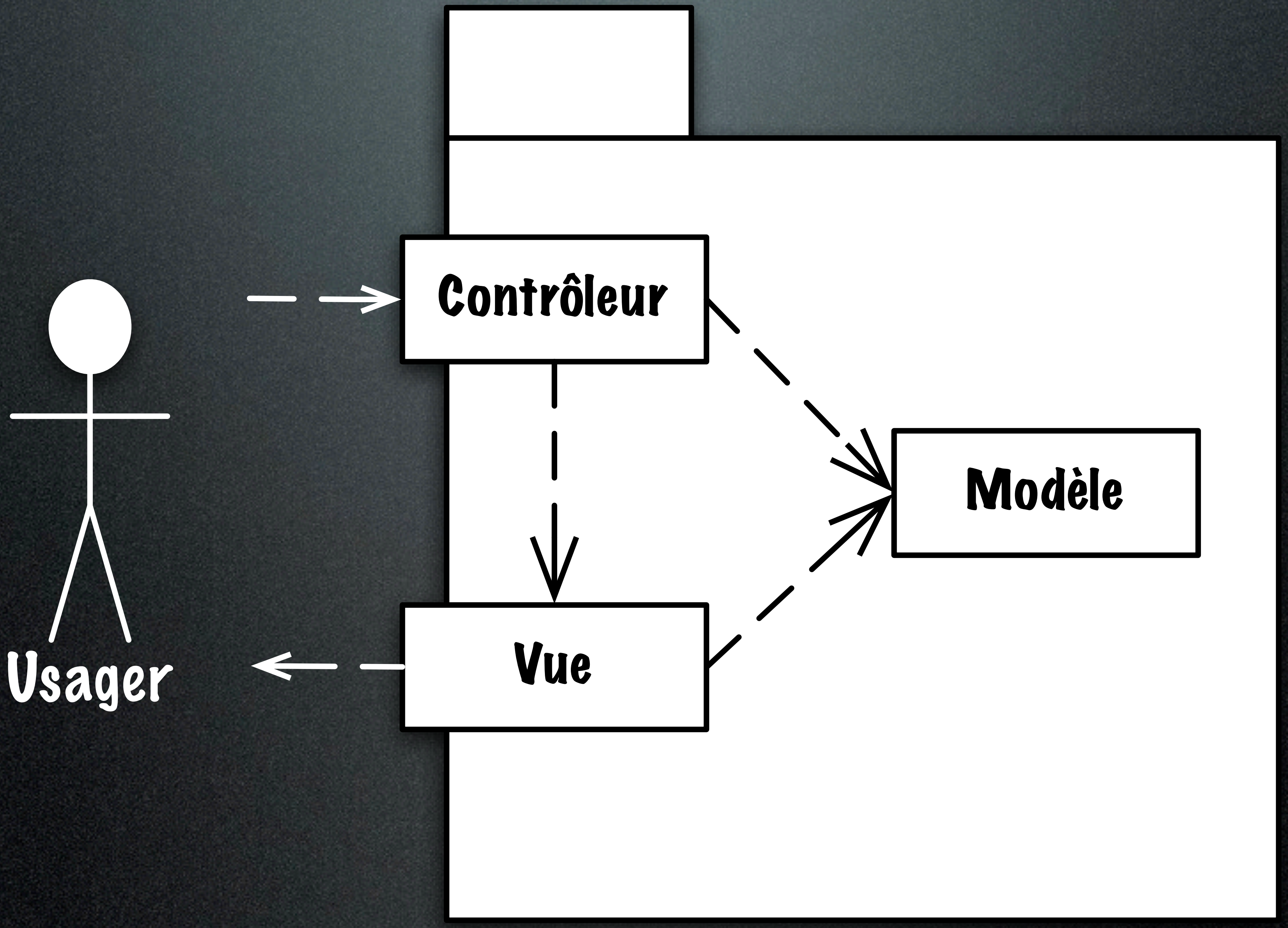
- Les applications concrètes comportent un grand nombre de composants : documents **JSP**, **Servlets**, **JavaBeans**, **package Java**, **pages statiques**, **métadonnées**, etc.

- Le modèle de conception que l’on retrouve souvent pour la réalisation des interfaces usager graphiques, **MVC**, est aussi utilisé pour le développement d’applications Web

## Model-View-Controller

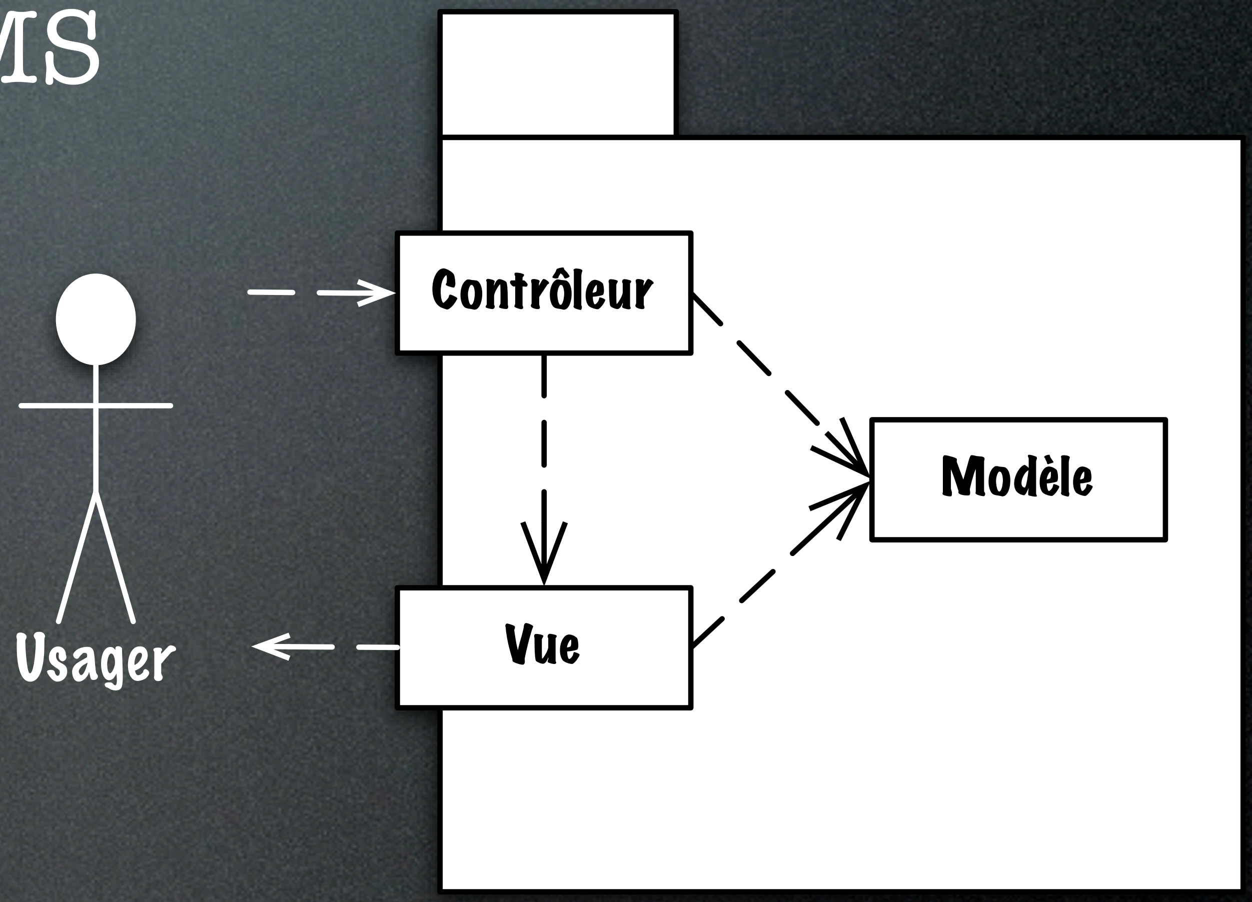
- Le **Contrôleur** reçoit et traite les requêtes HTTP, s'occupe par exemple des initialisations, traces et contrôles les accès
- Le **modèle** représente les données de l'application, possiblement des données persistantes, et s'occupe des calculs
- La troisième partie de MVC s'occupe de la **présentation** des données

## Interactions possibles



## Mariage MVC-technologies

- **Modèle**
  - JavaBeans, DBMS
- **Contrôleur**
  - Servlets
- **Vue**
  - JSP, XML/XSLT



## Servlets et contrôleurs

transmet la requête comme si l'usager avait lui même accédé cette URL

- Afin de supporter le modèle de conception MVC, la technologie des Servlets propose le **répartiteur** (**dispatcher**)
- **1)** Le contrôleur reçoit la requête de l'usager, **2)** interagit avec le modèle (ajout d'un article au panier de l'usager, p.e.), **3)** puis transmet la requête, à l'aide d'un répartiteur, à une vue JSP

# RequestDispatcher

- javax.servlet.RequestDispatcher = dispatcher;
- dispatcher = getServletContext().getRequestDispatcher( "/Login.jspx" );
- dispatcher = getServletContext().getNamedDispatcher("Login");
- web.xml :

```
<servlet>
    <servlet-name>Login</servlet-name>
    <jsp-file>/Login.jspx</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/Login.jspx</url-pattern>
</servlet-mapping>
```

Par URL :
getServletContext().get
RequestDispatcher( URL )

Par nom :

getServletContext().get
NameDispatcher( URL )

# src/ctl/Maestro.java

```
...
public class Maestro extends HttpServlet {

  public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
    throws ServletException, IOException {

    RequestDispatcher dispatcher;

    HttpSession session = requete.getSession();
    String usager = (String) session.getAttribute( "usager" );

    if ( session.isNew() || usager == null ) {
        dispatcher = getServletContext().getNamedDispatcher( "Login" );
    } else {
        dispatcher = getServletContext().getNamedDispatcher(  "GetCount" );
    }
    dispatcher.forward( requete, reponse );
  }
...
```

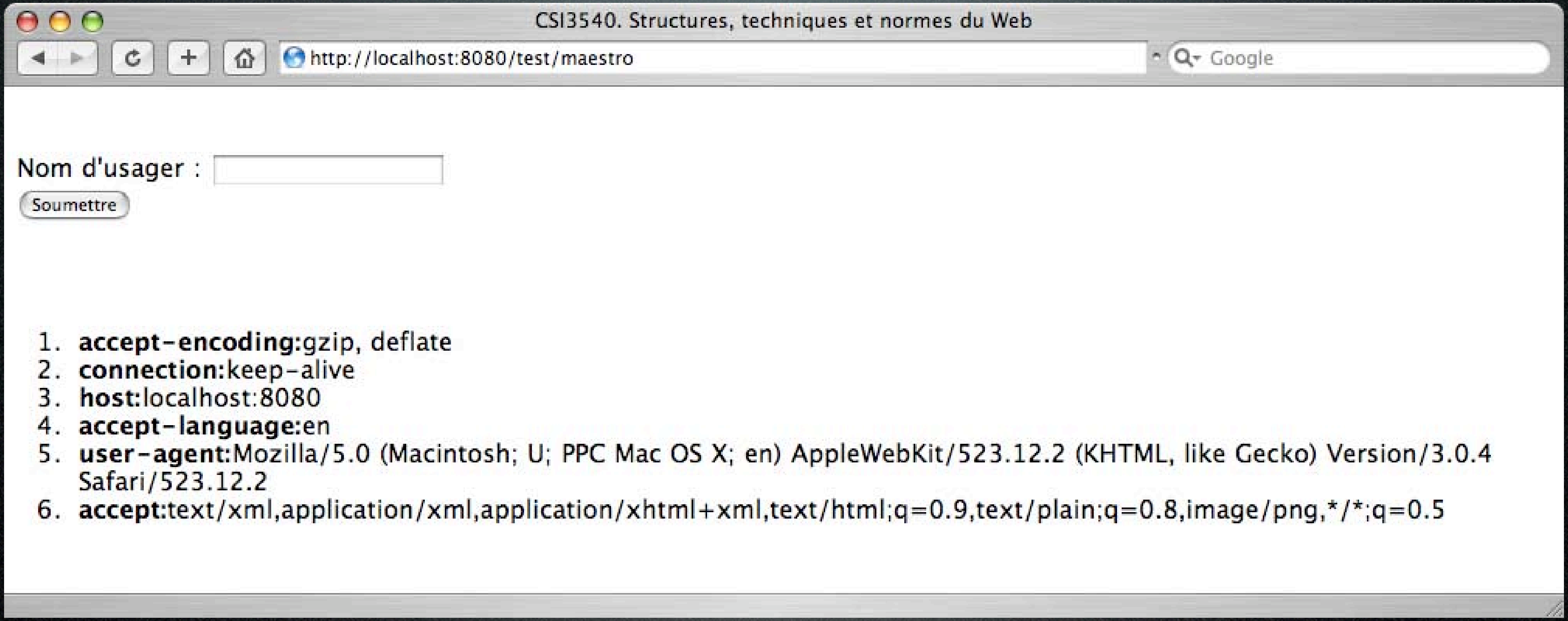
Maestro est un Servlet qui transmet la requête à des documents JSP pour qu'ils affichent les pages XHTML

# Login.jspx

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

  <head>
    <title>CSI3540. Structures, techniques et normes du Web</title>
  </head>

  <body>
    <p>
      <form action="maestro" method="post">
        <div>
          <label>Nom d'usager : <input type="text" size="20" name="user" /></label><br />
          <input type="submit" value="Soumettre" />
        </div>
      </form>
    </p>
  </body>
</html>
```

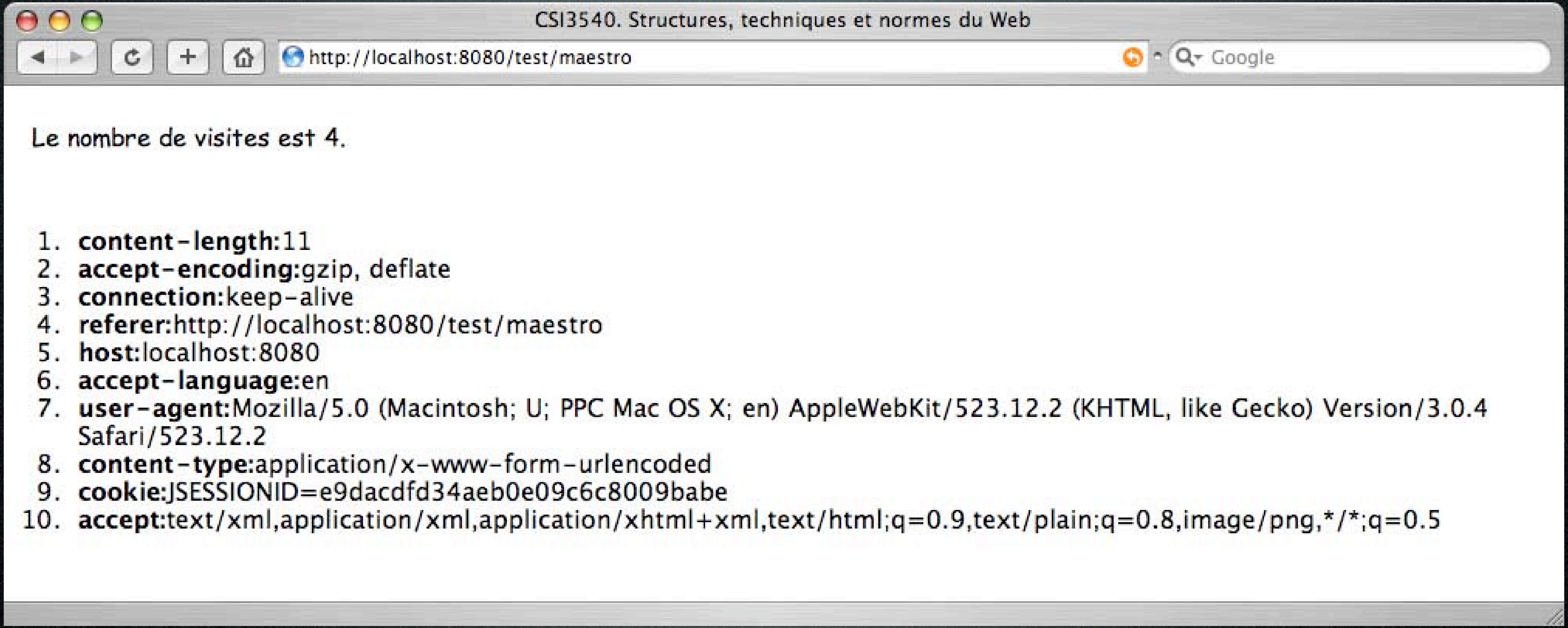
# GetCount.jspx

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

  <core:if test="${ empty visits }">
    <core:set var="visits" scope="application" value="${ initParam.visits }"/>
  </core:if>
  <core:set var="visits" scope="application" value="${ visits+1 }"/>

  <head>
    <title>CSI3540. Structures, techniques et normes du Web</title>
  </head>

  <body>
    <p>
      Le nombre de visites est ${ visits }.
    </p>
  </body>
</html>
```


CSI3540. Structures, techniques et normes du Web
http://localhost:8080/test/maestro
Google
Soumettre

Nom d'usager :

1. **accept-encoding:**gzip, deflate
2. **connection:**keep-alive
3. **host:**localhost:8080
4. **accept-language:**en
5. **user-agent:**Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/523.12.2 (KHTML, like Gecko) Version/3.0.4 Safari/523.12.2
6. **accept:**text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5


CSI3540. Structures, techniques et normes du Web
http://localhost:8080/test/maestro
Google

Le nombre de visites est 4.

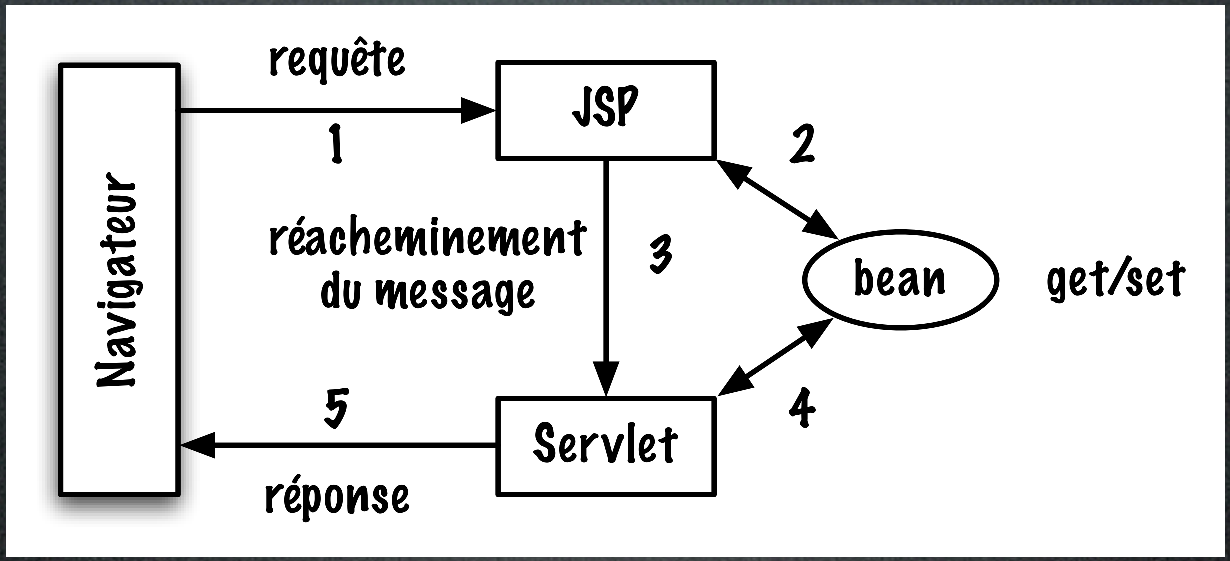
1. **content-length:**11
2. **accept-encoding:**gzip, deflate
3. **connection:**keep-alive
4. **referer:**http://localhost:8080/test/maestro
5. **host:**localhost:8080
6. **accept-language:**en
7. **user-agent:**Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/523.12.2 (KHTML, like Gecko) Version/3.0.4 Safari/523.12.2
8. **content-type:**application/x-www-form-urlencoded
9. **cookie:**JSESSIONID=e9dacdfd34aeb0e09c6c8009babe
10. **accept:**text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

## Remarque

- L’objet **dispatcher** possède aussi une méthode **include( request, response )**
- Les objects **request** et **response** peuvent servir à passer des informations

# <jsp:forward ...>

- De même, JSP possède un élément **<jsp:forward page="..."/>**



## Aiguillage

- Le répartiteur :
  - Consulte les valeurs des paramètres ainsi que les attributs de la session
  - Informations encodées à même l’URL, l’objet **HttpServletRequest** possède une méthode **getPathInfo()** qui retourne la portion chemin de l’URL,
  - **/controller/GetCount**
  - **/controller/Help?topic=login**

# Échange d'informations Servlets/JSP

- Données persistantes (disque, BD)
- Ajout d'attributs (**setAttribute(...)**) des objets **HttpServletRequest**, **HttpSession** ou **ServletContext**
- Qui correspondent aux espaces lexicaux EL **request**, **session** ou **application**
- Wrapper (voir page 468)

## MVC : Exemple

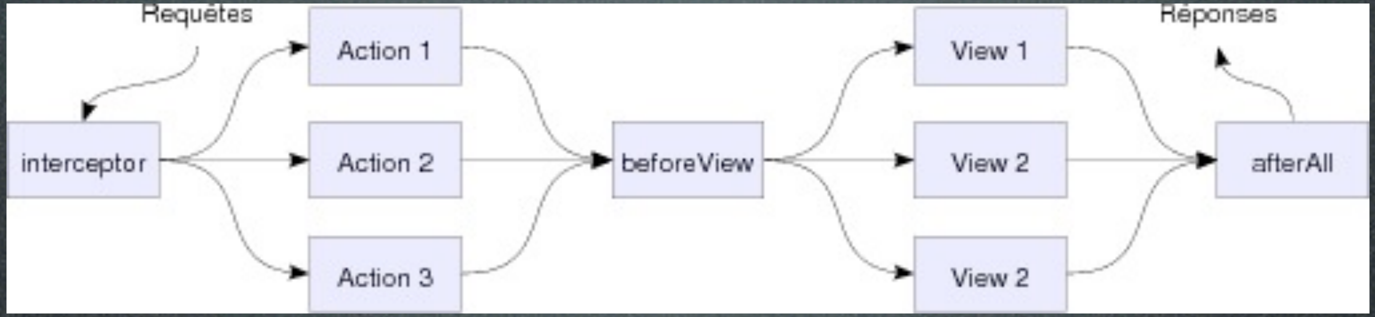
http://www.dil.univ-mrs.fr/~massat/ens/jee/mvc.html

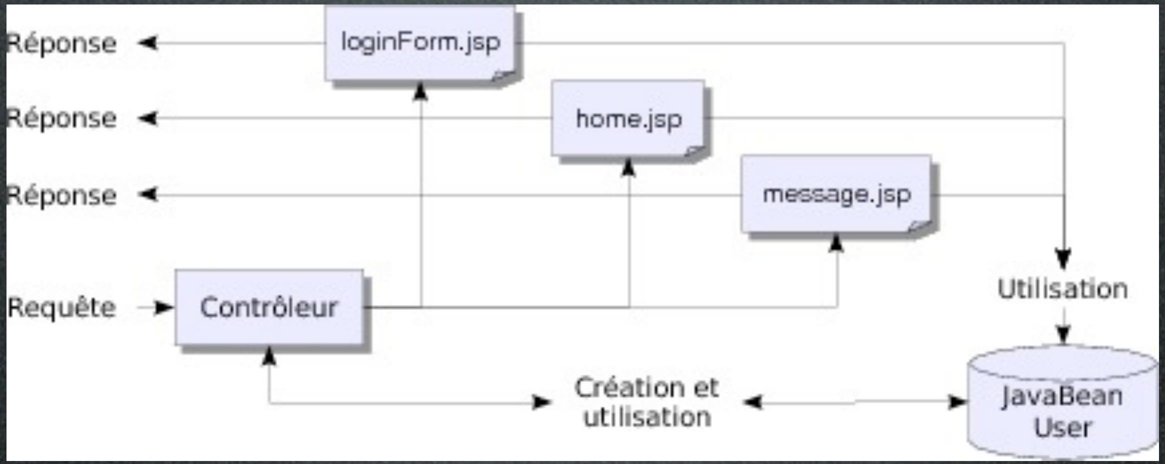
(Jean-Luc Massat, Université de la Méditerranée)

## Modification d’un annuaire

- L’utilisateur se connecte au système, il entre un nom d’usager et mot de passe
- Si l’authentification échoue alors retour au « login »
- Sinon, affiche le formulaire de modification des données usager
- Sauvegarde des données, si valides, sinon recommence

## Enchaînements





- Un seul point d'entrée pour rendre l'application plus sécuritaire

## User (un JavaBean)

```
package fr.exemple.values;

public class User {

    // properties
    private String id = "";
    private String name = "";
    private String password = "";
    private Boolean auth = false; // important !

    public User() {
    }

    // getters and setters
    ...

}
```

# Configurer les pages Web ( web.xml)

```
<jsp-config>
   <jsp-property-group>
      <description>Toutes les pages</description>
      <url-pattern>*.jspx</url-pattern>
      <page-encoding>UTF-8</page-encoding>
      <include-prelude>/copyright.jspf</include-prelude>
   </jsp-property-group>
</jsp-config>
```

## Couche de métier (Business logic)

```
package fr.exemple.business;
import fr.exemple.values.User;

public class Business {

    void authenticate( User p ) throws BusinessException {
        ....
    }

    void updatePerson( User p ) throws BusinessException {
        ....
    }

}
```

## Interface vers les vues

```
package fr.exemple.webapp;

import java.util.Set;
import javax.servlet.http.HttpServletRequest;
import fr.exemple.values.User;

public class Views {

    String loginForm( HttpServletRequest request, User user, String message ) {
        request.setAttribute( "user", user );
        request.setAttribute( "message", message );
        return "/loginForm.jspx";
    }

    String home( HttpServletRequest request, User user, Set<User> users ) {
        request.setAttribute( "user", user );
        request.setAttribute( "users", users );
        return "/home.jspx";
    }

    String message( HttpServletRequest request, String message ) {
        request.setAttribute( "message", message );
        return ( "/message.jsp" );
    }
}
```

## Déclaration du contrôleur (web.xml)

```
<servlet>
  <servlet-name>Cont</servlet-name>
  <servlet-class>fr.exemple.webapp.MyControler</servlet-class>
  <init-param>
    <description>Répertoire de stockage</description>
    <param-name>dataDir</param-name>
    <param-value>/tmp/donnees</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Cont</servlet-name>
  <url-pattern>*.do</url-pattern><url-pattern>/do/*</url-pattern>
</servlet-mapping>
```

## /index.jspx

```
<c:redirect url="/login.do" />
```

## Contrôleur

```
package fr.exemple.webapp;

import ...

public class MyControler extends HttpServlet {

    private Business business = null;
    final private Set<User> users = new HashSet<User>();
    final private Views views = new Views();

    final private String loginUrl = "/login.do";
    final private String logoutUrl = "/logout.do";

    // démarrage et arret du contrôleur
    ...

    // traitement des sessions
    ...

    // traitement des erreurs et des actions
    ...

    }
```

## Ctrl : démarrage

```
// Démarrage du contrôleur : création de la couche métier

@Override
final public void init( ServletConfig c ) throws ServletException {
    Business b = new Business();
    b.setDataDir( c.getInitParameter( "dataDir" ) );
    b.init();
    business = b;
}
```

## Ctrl : arrêt du contrôleur

```
// Arret du contrôleur

@Override
final public void destroy() {
    business.close();
}
```

## Ctrl :méthode auxilliaire

```
private User getUser( HttpServletRequest request ) {

    HttpSession session = request.getSession();
    synchronized ( session ) {
        Object o = session.getAttribute( "user" );
        if ( o instanceof User ) {
            return (User) o;
        }
        User newUser = new User();
        session.setAttribute( "user", newUser );

        // espionner la session
        session.setAttribute( "spy", new Spy( newUser ) );
        return ( newUser );
    }
}
```

## Ctrl :classe auxilliaire

```
private class Spy implements HttpSessionBindingListener {
    final User user;

    public Spy( User user ) {
        this.user = user;
    }

    // traitement d'une nouvelle session
    public void valueBound( HttpSessionBindingEvent ev ) {
        users.add( user );
    }

    // traitement d'une fin de session
    public void valueUnbound( HttpSessionBindingEvent ev ) {
        users.remove( user );
    }

}
```

## Ctrl : traitement des requêtes

```
@Override
final protected void service( HttpServletRequest request,
        HttpServletResponse response ) throws ServletException, IOException {
    try {
        String view = beforeAll( request );
        if ( view == null )
            view = processAction( request, response );
        if ( view != null )
            view = beforeView( view, request );
        if ( view != null )
            processView( view, request, response );
    } catch ( Throwable t ) {
        String view = error( request, response, t );
        if ( view != null )
            try {
                processView( view, request, response );
            } catch ( Throwable e ) {
                throw new ServletException( e );
            }
    } finally {
        afterAll( request );
    }
```

## Ctrl : interception de la requête

```
// Interception d'une requête
public String beforeAll( HttpServletRequest request ) {

    // pour obtenir le temps de calcul
    request.setAttribute( "timer", System.currentTimeMillis() );

    // vérifier l'authentification
    if ( ! request.getServletPath().equals( loginUrl ) ) {
        User user = getUser( request );
        if ( user.getAuth() == false ) {
            String msg = "Vous devez vous authentifier";
            return views.loginForm( request, user, msg );
        }
    }
    return null;
}
```

## Ctrl : aiguillage

```
private String processAction( HttpServletRequest request,
        HttpServletResponse response ) throws Throwable {

    request.setCharacterEncoding( "UTF-8" );

    String action = request.getServletPath();

    if ( action.equals( loginUrl ) )
        return doLogin( request );
    if ( action.equals( logoutUrl ) )
        return doLogout( request );

    // autres actions
    throw new ServletException( "action " + action + " inconnue." );
}
```

## Actions spécifiques du login

```
public String doLogin( HttpServletRequest request ) {
    User user = getUser( request );
    // si l'utilisateur est connu
    if ( user.getAuth() )
        return views.home( request, user, users );

    // récupérer les données du formulaire
    user.setId( request.getParameter("id") );
    user.setPassword( request.getParameter("password") );
    // si aucune données retour au formulaire
    if ( user.getId() == null || user.getPassword() == null )
        return views.loginForm( request, user, null );

    // traiter les données
    if ( user.getPassword().length() == 0 )
        return views.loginForm( request, user, "Mot de passe vide" );
    try {
        business.authenticate( user );
        if ( user.getAuth() )
            return views.home( request, user, users );
        return views.loginForm( request, user, "mot de passe incorrecte" );
    } catch ( BusinessException e ) {}
    return views.loginForm( request, user, "Utilisateur inconnu" );
}
```

## Actions spécifiques du logout

```
public String doLogout( HttpServletRequest request ) {
    User user = getUser( request );
    user.setAuth( false );
    user.setId( "" );
    user.setName( "" );
    user.setPassword( "" );
    return views.message( request, "A bientôt" );
}
```

## Ctrl : interception avant vue

```
public String beforeView( String view, HttpServletRequest request ) {

    return ( view );
}
```

## Ctrl : traitement des vues

```
private void processView( String view, HttpServletRequest request,
        HttpServletResponse response ) throws Throwable {

    if ( view.endsWith(".jsp") || view.endsWith( ".jspx" ) ) {

        // technologie JSP/JSPX
        request.getRequestDispatcher( view ).forward( request, response );

    } else if ( view.endsWith( ".xsl" ) ) {

        // technologie XSLT / JAXP
        processXsltView( view, request, response );

    } else {

        // autres technologies
        throw new IllegalStateException( "no view " + view );

    }
}
```

# Vue : /loginForm.jspx

```
<c:url var="login" value="/login.do" />

<h1 >Authentification</h1 >
<p><c:out value="${message}" default="Identifiez-vous :"/></p>

<form method="POST" action="${login}">
    <label>Login :</label>
    <input name="id" type="text"
        value="<c:out value="${ user.id }"/>"/>
    <label>Password : </label>
    <input name="password" type="password"
        value="<c:out value="${ user.password }"/>"/>
    <br/>
    <label>Validation :</label>
    <input name="ok" type="submit" value="Ok"/>
</form>
```

# Vue : /home.jspx

```
<c:url var="logout" value="/logout.do" />
<c:url var="editForm" value="/editForm.do" />

<h1 >Bienvenue <c:out value="${user.name}"/></h1 >

<p>Vous pouvez <a href="${ logout }">nous quitter</a> ou
<a href="${ editForm }">modifier vos informations</a>.</p>

<p>Utilisateurs :</p>
<ul><c:forEach var="u" items="${ users }">
    <li>${ u.name }</li>
</c:forEach>
</ul>
```

## Vue : /message.jspx

```
<c:url var="index" value="/index.jspx" />

<h1 ><c:out value="${message}"/></h1 >
<p><a href="${ index }">Retour</a>.</p>
```

## Ctrl : interception après vue

```
// Interception après vue

public void afterAll( HttpServletRequest request ) {
    Long start = (Long) request.getAttribute( "timer" );
    Long duration = System.currentTimeMillis() - start;
    debug( "time : ", duration.toString(), "ms" );
}
```

## Autres

- <jsp:include page=“/navbar.jspx”> (p 472)

# Vue : /loginForm.jspx

```
<c:url var="login" value="/login.do" />

<h1 >Authentification</h1 >
<p><c:out value="${message}" default="Identifiez-vous :"/></p>

<form method="POST" action="${login}">
   <label>Login :</label>
   <input name="id" type="text"
      value="<c:out value="${user.id}"/>"/>
   <label>Password : </label>
   <input name="password" type="password"
      value="<c:out value="${user.password}"/>"/>
   <br/>
   <label>Validation :</label>
   <input name="ok" type="submit" value="Ok"/>
</form>
```

# web.xml

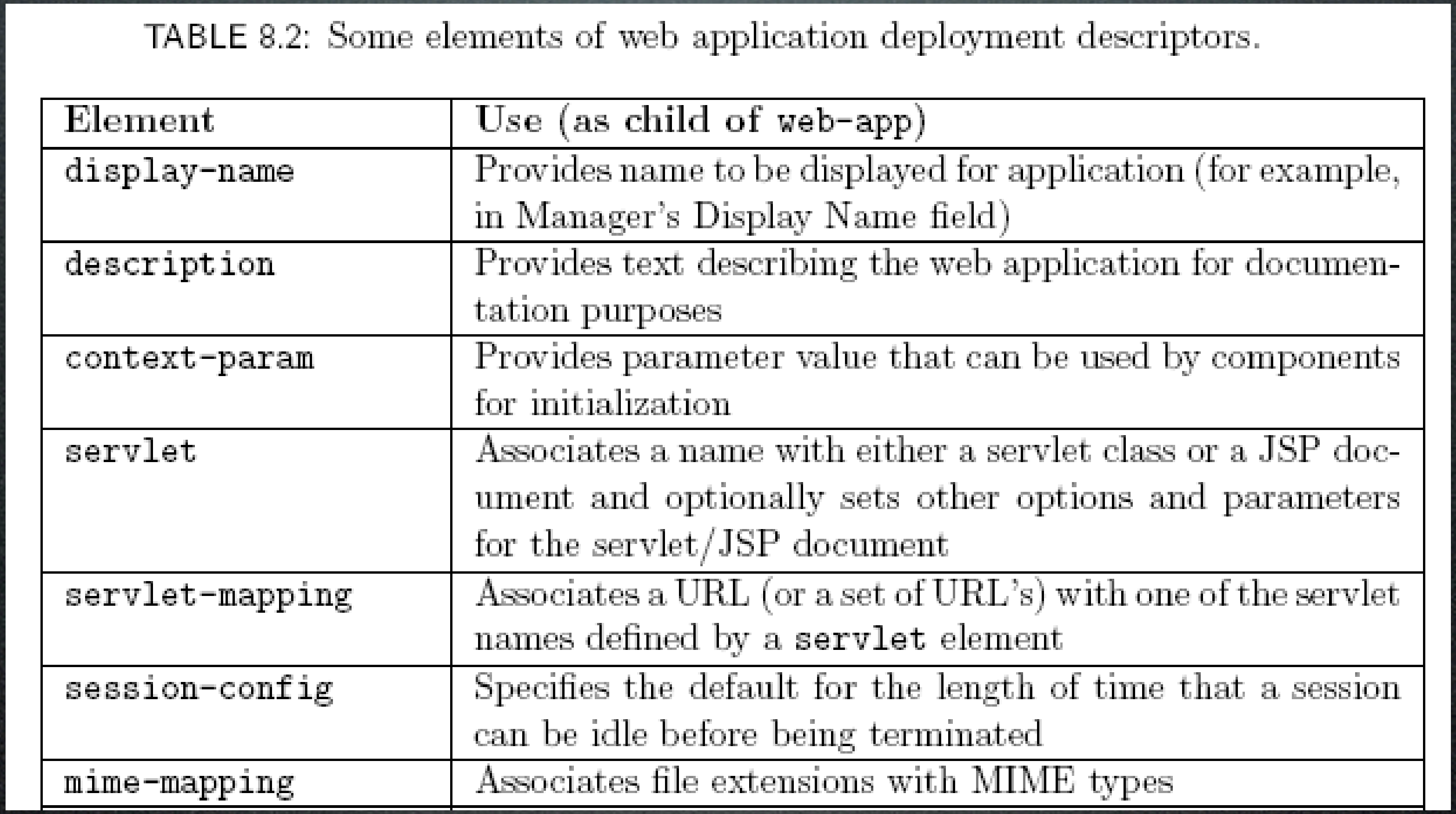
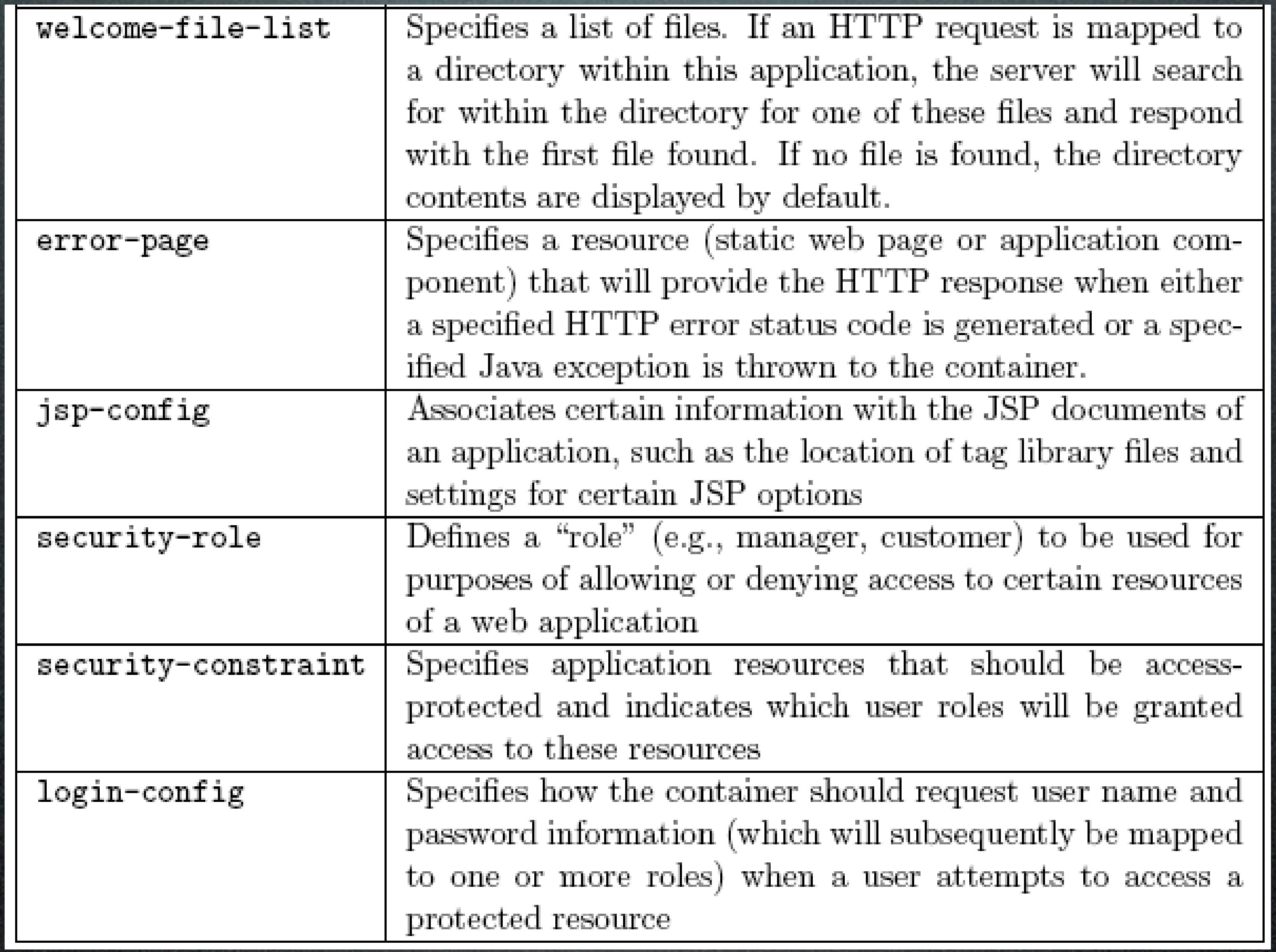


TABLE 8.2: Some elements of web application deployment descriptors.

| Element | Use (as child of web-app) |
|---|---|
| display-name | Provides name to be displayed for application (for example, in Manager's Display Name field) |
| description | Provides text describing the web application for documentation purposes |
| context-param | Provides parameter value that can be used by components for initialization |
| servlet | Associates a name with either a servlet class or a JSP document and optionally sets other options and parameters for the servlet/JSP document |
| servlet-mapping | Associates a URL (or a set of URL's) with one of the servlet names defined by a servlet element |
| session-config | Specifies the default for the length of time that a session can be idle before being terminated |
| mime-mapping | Associates file extensions with MIME types |

# web.xml



| | |
|---|---|
| `welcome-file-list` | Specifies a list of files. If an HTTP request is mapped to a directory within this application, the server will search for within the directory for one of these files and respond with the first file found. If no file is found, the directory contents are displayed by default. |
| `error-page` | Specifies a resource (static web page or application component) that will provide the HTTP response when either a specified HTTP error status code is generated or a specified Java exception is thrown to the container. |
| `jsp-config` | Associates certain information with the JSP documents of an application, such as the location of tag library files and settings for certain JSP options |
| `security-role` | Defines a “role” (e.g., manager, customer) to be used for purposes of allowing or denying access to certain resources of a web application |
| `security-constraint` | Specifies application resources that should be access-protected and indicates which user roles will be granted access to these resources |
| `login-config` | Specifies how the container should request user name and password information (which will subsequently be mapped to one or more roles) when a user attempts to access a protected resource |

## web.xml

```xml
<session-config>
  <session-timeout>1</session-timeout>
</session-config>
```

```xml
<servlet>
  <servlet-name>visit_count</servlet-name>
  <jsp-file>/HelloCounter.jspx</jsp-file>
</servlet>
<servlet-mapping>
  <servlet-name>visit_count</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>visit_count</servlet-name>
  <url-pattern>/visitor/*</url-pattern>
</servlet-mapping>
```

# web.xml

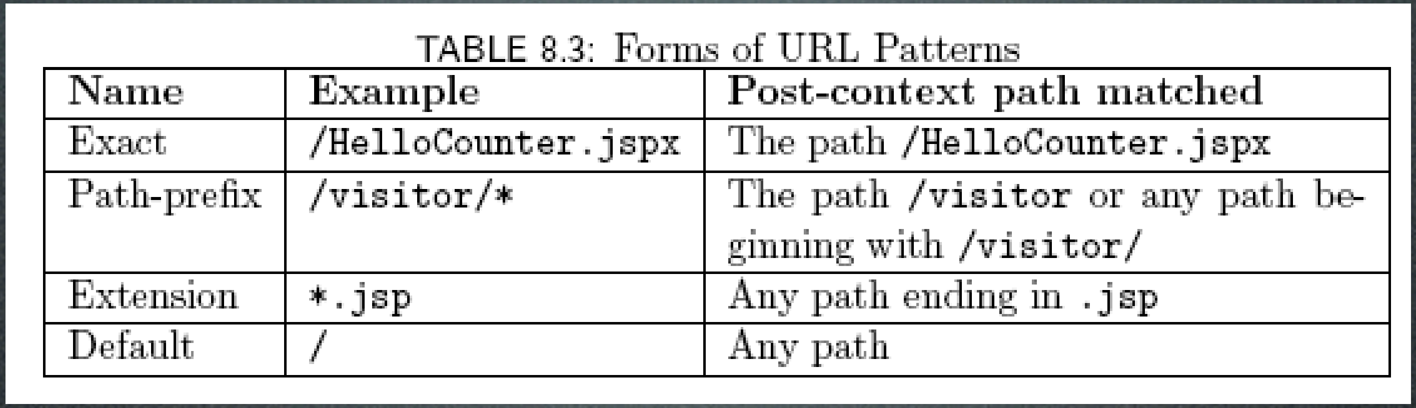


TABLE 8.3: Forms of URL Patterns

| Name | Example | Post-context path matched |
|---|---|---|
| Exact | /HelloCounter.jspx | The path /HelloCounter.jspx |
| Path-prefix | /visitor/* | The path /visitor or any path beginning with /visitor/ |
| Extension | *.jsp | Any path ending in .jsp |
| Default | / | Any path |

- If no URL pattern matches, Tomcat treats path as a relative file name

## Épilogue

## Remarques

- Les remarques faites au sujet des **accès partagés** par rapport aux Servlets s’appliquent aussi aux documents JSP
- Technologies alternatives :
  - Active Server Pages et ASP.NET
  - PHP
- Nouvelles technologies : Faces

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

## Ressources

- 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
- Application Web : Les bonnes pratiques
  - http://www.dil.univ-mrs.fr/~massat/ens/jee/mvc.pdf