Nancy-Université

Mise en œuvre des serveurs d'application

UE 203d

Master 1 IST-IE

Printemps 2008

Ces transparents, ainsi que les énoncés des TDs, seront disponibles à l'adresse :

`http://www.loria.fr/~burel/empty_cours.html`

Troisième partie III

Introduction à JSP

Plan

- Rappels des généralités
- Rappels sur HTML
  - Historique
  - Contenu
  - Balises disponibles (liste non exhaustive)
- JSP
  - Servlets et JSP
  - Contenu
- Application : création d'un site

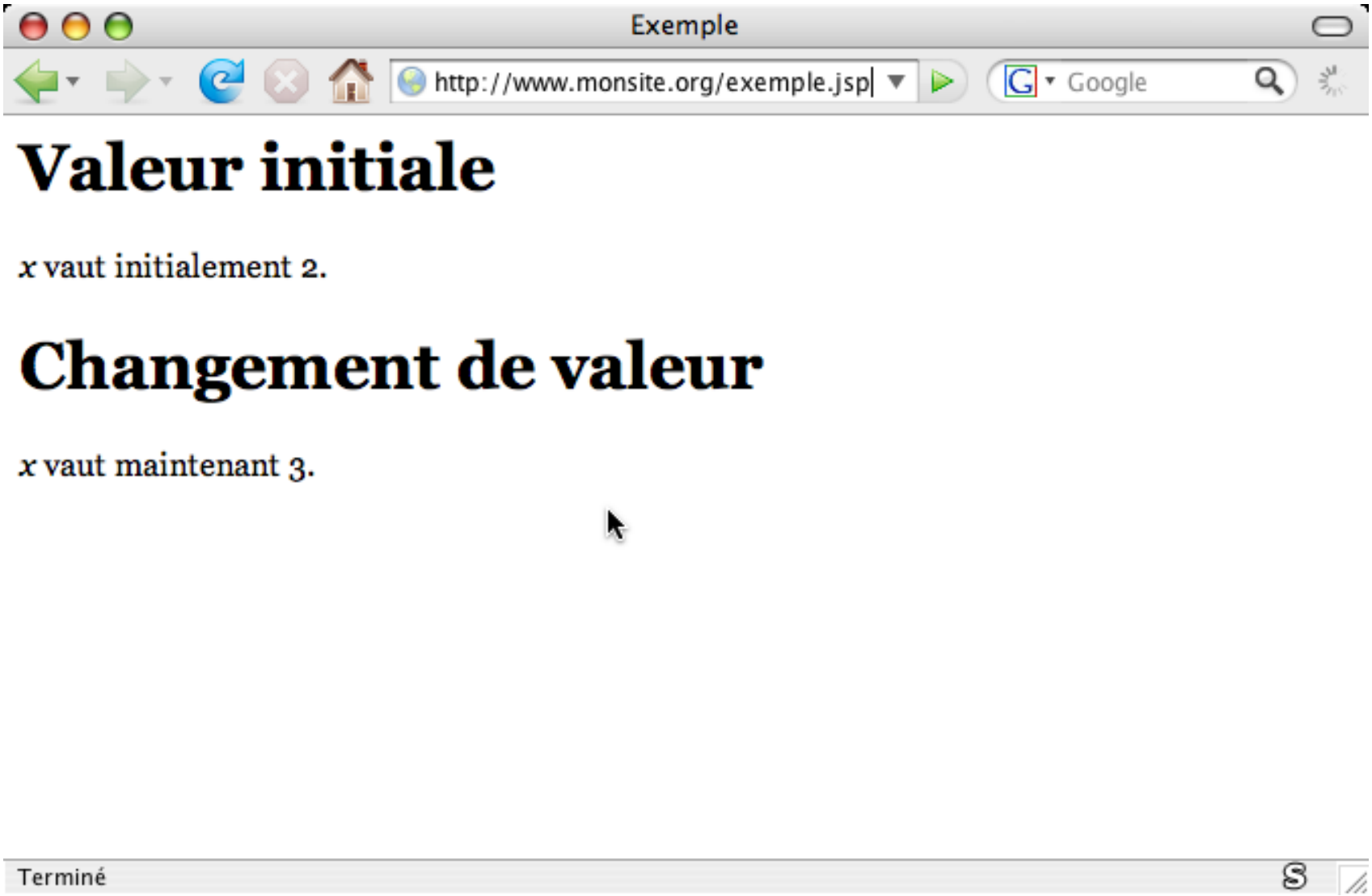
Pages JSP

permet de créer des pages HTML dynamiques
insertion de bouts de code java dans des pages HTML
code exécuté sur le serveur ($\neq$ javascript) : création d'un servlet correspondant

Exemple de page JSP

```
<%@ page language="java" %>
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
    <% int x = 2; %>
    <h1>Valeur initiale</h1>
    <p><var>x</var> vaut initialement <%=x%>.</p>
    <h1>Changement de valeur</h1>
    <% x = x + 1; %>
    <p><var>x</var> vaut maintenant <%=x%>.</p>
  </body>
</html>
```

Résultat


Exemple
http://www.monsite.org/exemple.jsp
Google
Valeur initiale
x vaut initialement 2.
Changement de valeur
x vaut maintenant 3.
Terminé

Plan

- Rappels des généralités
- Rappels sur HTML
  - Historique
  - Contenu
  - Balises disponibles (liste non exhaustive)
- JSP
  - Servlets et JSP
  - Contenu
- Application : création d'un site

Naissance

Langage créé par le CERN en 1990, pour échanger des informations (hyper)textuelles
Compromis entre manipulation par la machine et lisibilité par les humains
Standardisé en 1995 pour la première fois (W3C)

Évolution

À partir de 2000, passage à XML : généralisation de HTML à toute sorte de grammaires
changements notoires: minuscules obligatoires, obligation d'avoir des balises correctement fermées
`<clef argument="chaine">...</clef>` ou
`<clef argument="chaine" />`
Autres exemples de XML : RSS, podcast, SVG, fichiers de configuration des applications J2EE
Avantage : pas besoin d'écrire un parser spécifique

Contenu d'une page (X)HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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"
xml:lang="fr">
  <head>
    <!-- Ceci est un commentaire -->
    <!-- Entête (titre, styles, meta-infos) -->...
  </head>
  <body>
    <!-- Contenu de la page -->...
  </body>
</html>
```

Entête

```
<head>
  <meta name="info" content="contenu" />
  <link rel="relation" type="format" href="place"/>
  <title>Titre</titre>
  <style type="format">
    <!-- Style pour le rendu -->...
  </style>
  <script type="format">
    <!-- Ex: javascript -->...
  </script>
  <base href="adresse" />
</head>
```

Corps

- `<p>...</p>` Paragraphe contenant du texte

```
<p>Ceci est un
paragraphe.</p>
```

Ceci est un paragraphe.

- `<h1>...</h1>` à `<h6>...</h6>` Titres de sections

```
<h1>Grand titre</h1>
<h2>Petit titre</h2>
```

Grand titre
Petit titre

- `<pre>...</pre>` Texte brut

```
<pre>Une phrase
sur  deux lignes.</pre>
```

```
Une phrase
sur  deux lignes.
```

- `<fieldset>...</fieldset>` Cadre

```
<fieldset>
  <legend>Légende</legend>
   <p>Contenu</p>
</fieldset>
```

Légende
Contenu

- `<hr/>` Ligne horizontale

`<ul>...</ul>`, `<ol>...</ol>` Listes sans numéro, avec numéro

```
<ul>
  <li>chou</li>
  <li>carotte</li>
  <li>navet</li>
</ul>
```

- chou
- carotte
- navet

```
<ol>
  <li>métro</li>
  <li>boulot</li>
  <li>dodo</li>
</ol>
```

1. métro
2. boulot
3. dodo

`<table>...</table>` Tableaux

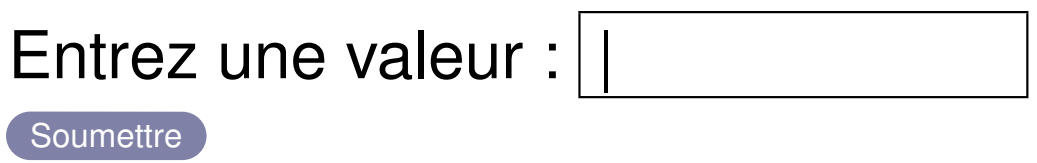
```
<table>
  <caption>Légende</caption>
  <tr> <th> Titre 1 </th> <th> Titre 2 </th> </tr>
 <tr><td>Cellule 1.1</td><td>Cellule 2.1</td></tr>
 <tr><td>Cellule 1.2</td><td>Cellule 2.2</td></tr>
 <tr><td>Cellule 1.3</td><td>Cellule 2.3</td></tr>
</table>
```

Légende

Titre 1	**Titre 2**
Cellule 1.1	Cellule 2.1
Cellule 1.2	Cellule 2.2
Cellule 1.3	Cellule 2.3

`<form>...</form>` Formulaires

```
<form action="reponse.html" method="get">
  <p>
    Entrez une valeur :<input name="param"/>
  </p>
  <button>Soumettre</button>
</form>
```



Appelle la page `reponse.html?param=contenu`

Contenu des paragraphes, titres, cellules, . . .

- `<a>...</a>` Lien hypertexte

```
<p>
  Ceci est un
<a href="page.html">lien</a>.
</p>
```

Ceci est un lien.

- `<br/>` Retour à la ligne

```
<p>
  Une ligne. <br/> Deux
  lignes.
</p>
```

Une ligne.
Deux lignes.

`<b>Gras</b>`	**Gras**
`<i>Italique</i>`	*Italique*
`<tt>Espacement constant</tt>`	`Espacement constant`
`<big>Grand</big>`	Grand
`<small>Petit</small>`	Petit
`Ind<sub>ice</sub>`	Ind$_{ice}$
`Expo<sup>sant</sup>`	Expo^{sant}
`<del>Supprimé</del>`	~~Supprimé~~
`<ins>Inséré</ins>`	Inséré

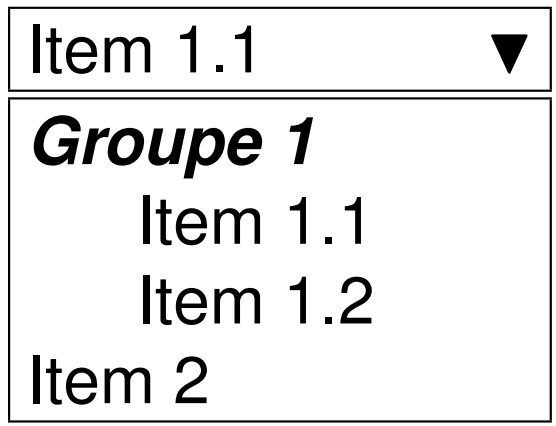
- `<dfn>...</dfn>` Définition
- `<em>...</em>` Emphase
- `<strong>...</strong>` Renforcement
- `<code>...</code>` Code source
- `<q>...</q>` Citation
- `<samp>...</samp>` Exemple
- `<kbd>...</kbd>` Entrée clavier
- `<var>...</var>` Variable
- `<cite>...</cite>` Citation
- `<abbr>...</abbr>` Abréviation
- `<acronym>...</acronym>` Acronyme

- `<img/>` Image

```
<img alt="description"
     src="logo_nu.jpg"/>
```

- `<select>...</select>` Liste à choix multiples

```
<select>
  <optgroup label="Groupe 1">
    <option>Item 1.1</option>
    <option>Item 1.2</option>
  </optgroup>
  <option>Item 2</option>
</select>
```



`Entrée <input value="valeur" type="`*type*`"/>`

type	résultat
`text`	Entrée valeur
`password`	Entrée ******
`checkbox`	Entrée ☒
`radio`	Entrée ⊙
`submit`	Entrée valeur
`reset`	Entrée valeur
`file`	Entrée Parcourir...
`hidden`	Entrée
`image`	Entrée 
`button`	Entrée valeur

Plan

- Rappels des généralités
- Rappels sur HTML
  - Historique
  - Contenu
  - Balises disponibles (liste non exhaustive)
- JSP
  - Servlets et JSP
  - Contenu
- Application : création d'un site

Servlet

Objet java permettant de traiter des requêtes HTTP
Sous-classe de HttpServlet :

```
public class HttpServlet {
  public void doGet (HttpServletRequest request,
                     HttpServletResponse response)
    throws ServletException, IOException;

  public void doPost (HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException;
}
```

```
public class ExempleServlet extends HttpServlet {
  public void doGet (HttpServletRequest request,
                     HttpServletResponse response)
    throws ServletException, IOException {
      String nom = request.getParameter("nom");
      response.setContentType("text/html");
      response.setBufferSize(8192);
      PrintWriter out = response.getWriter();
      out.println("<html><head><title>Titre");
      out.println("</title></head><body><p>");
      out.println("Bonjour " + nom + " !");
      out.println("</p></body></html>");
    } }
```

Désavantages

- Assez éloigné des connaissances des concepteurs de sites
- Besoin de connaître java
- Une grande partie du code toujours pareil

Pages JSP

Écrire une page HTML avec la possibilité de rajouter du java
Est ensuite transformé en un servlet qui gère les requêtes

(En fait plus généralement permet de mélanger des parties statiques et dynamiques quel que soit leur type)

```
<%@ page language="java" contentType="text/html" %>
<html>
  <head>
    <title>Titre</title>
  </head>
  <body>
<% String nom = request.getParameter("nom"); %>
    <p>Bonjour <%=nom %></p>
  </body>
</html>
```

```
public void doGet (HttpServletRequest request,
                   HttpServletResponse response) ... {
  PrintWriter out = response.getWriter();
  response.setContentType("text/html");
  out.println("<html>");
  out.println("  <head>");
  out.println("    <title>Titre</title>");
  out.println("  </head>");
  out.println("  <body>");
  String nom = request.getParameter("nom");
  out.println("    <p>Bonjour " + nom + "</p>");
  out.println("  </body>");
  out.println("</html>");
}
```

Expressions et scripts

- `<%=expression%>` : Calcule `expression` puis l'affiche
  Est transformé en `out.print(expression)`
- `<% code %>` : Exécute le code
  Ajoute `code` au servlet
- `<%! declaration %>` : ajoute le code `declaration` à la classe du servlet
  Exemple :

  ```
  <%! private String fonctionUtile() { ... } %>
  ```

  également redéfinition de `jspInit()` et `jspDestroy()`

Directives JSP

`<%@ ... %>`
trois directives :

- `page` : information sur la page
- `include` : inclusion d'autres pages JSP
- `taglib` : utilisation d'extensions de JSP

Directive `page`

```
<%@ page language="java"
      contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"
      import="java.util.*" %>
```

- `language` : langage utilisé pour les parties dynamiques
- `contentType` : type du résultat, c'est celui des parties statiques
- `pageEncoding` : encodage des caractères dans la page JSP
- `import` : importation de classes java, pour ne pas avoir à donner leur nom complet dans la page (ex: `Vector` au lieu de `java.util.Vector`)

Directive `include`

```
<%@ include file="nomDeFichier" %>
```

Permet d'inclure le fichier `nomDeFichier` dans la page
Améliore maintenance : partie commune à plusieurs pages dans un seul fichier inclus dans ces pages
Exemple : titre des pages, menu, etc.

Ajoute de façon statique : recopie le contenu de `nomDeFichier` tel quel
$\neq$ action `<jsp:include page="nomDeFichier" />` qui ajoute de façon dynamique: `nomDeFichier` est d'abord transformé, puis inclus

Directive `taglib`

```
<%@ taglib prefix="prefixe" uri="adresse" %>
```

- `prefix` : préfixe utilisé dans les balises de l'extension
- `uri` : adresse de la description de l'extension

Exemple :

```
<%@ taglib prefix="c"
      uri="http://java.sun.com/jsp/jstl/core" %>
...
<c:if test="${!empty param.nom}">
  Bonjour ${param.nom}
</c:if>
```

Actions JSP

- `<jsp:include />` : inclusion de page
- `<jsp:forward />` : transfert de page
- `<jsp:param />` : passage de paramètre
- `<jsp:useBean />` : utilisation de classe
- `<jsp:setProperty />` : modification de propriété
- `<jsp:getProperty />` : affichage de propriété

Actions `<jsp:include />`, `<jsp:forward />`, `<jsp:param />`

- `<jsp:include page="nomDePage" />`
  Fait la requête de `nomDePage`, puis l'inclut
- `<jsp:forward page="nomDePage" />`
  Redirige la requête vers `nomDePage`
  Le reste du fichier JSP n'est pas évalué
- `<jsp:param name="nomParam" value="valeur" />`
  Permet de rajouter des paramètres en plus pour les requêtes de `<jsp:include />` et `<jsp:forward />`

Exemple

```
<jsp:include page="reponse.jsp">
  <jsp:param name="titre" value="93" />
</jsp:include>
```

inclut la page `reponse.jsp?titre=93`

Action `<jsp:useBean />`

```
<jsp:useBean id="var" class="nomDeClasse"
scope="portee"/>
```

Permet de créer un objet d'une classe JavaBean

- `id` : nom de la variable où l'objet est affecté
- `class` : nom de la classe
- `scope` : portée de l'objet, (`application`, `session`, `request`, `page`)

JavaBean ($\neq$ EJB !) = classe java avec méthodes de la forme

```
PropClass getPropriete ();
void setPropriete (PropClass nouvelleValeur);
```

qui définissent une propriété

Action `<jsp:setProperty />`

Permet de modifier la propriété d'un bean
Appelle `setPropriete(nouvelleValeur)`

- `<jsp:setProperty name="var" property="propriete" value="chaine"/>`
  convertit `chaine` en `PropClass`
  modifie la propriété `propriete` du bean préalablement défini `var` avec cette valeur
- `<jsp:setProperty name="var" property="propriete" param="parametre"/>`
  idem, mais en utilisant un paramètre de la requête
  si `param` omis, utilise `propriete`

Action `<jsp:getProperty />`

```
<jsp:getProperty name="var" property="propriete"/>
```

Récupère et affiche la propriété `propriete` du bean `var`

Autre syntaxe : `${var.propriete}`

Exemple d'utilisation de JavaBean

```
<jsp:useBean id="date" class="java.util.Date"
    scope="page"/>
<p>Il est
  <jsp:getProperty name="date" property="hours"/>
  heures.</p>
<jsp:setProperty name="date" property="time"
    value="${date.time + 86400000}" />
<p>Demain à la même heure il sera
  ${date.hours} heures.</p>
```

Plan

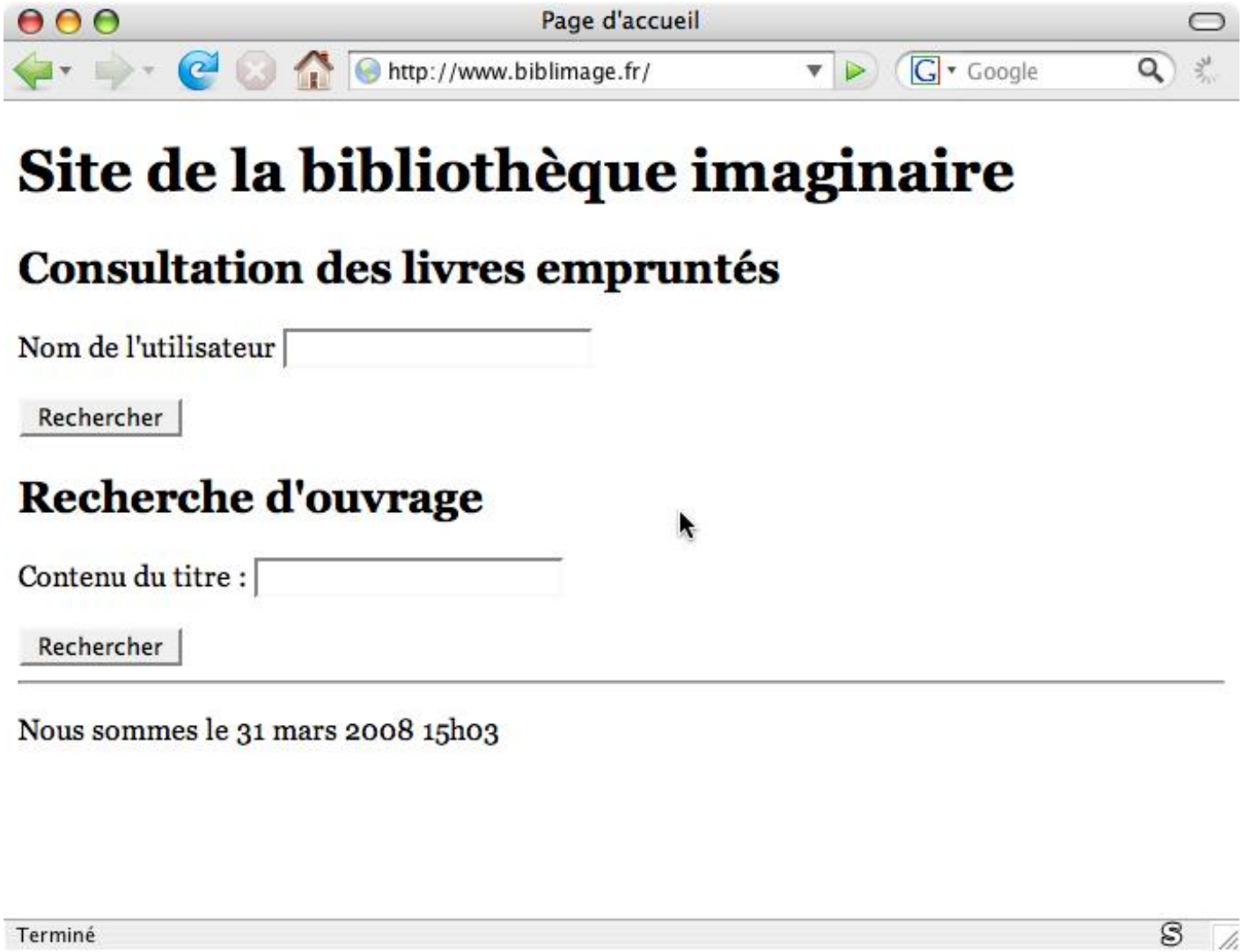
- Rappels des généralités
- Rappels sur HTML
  - Historique
  - Contenu
  - Balises disponibles (liste non exhaustive)
- JSP
  - Servlets et JSP
  - Contenu
- Application : création d'un site

Site de la bibliothèque imaginaire

On veut créer un site de gestion d'une bibliothèque, avec la possibilité de:

- emprunter et rendre un livre
- consulter la liste des livres empruntés par quelqu'un
- rechercher un livre par son titre

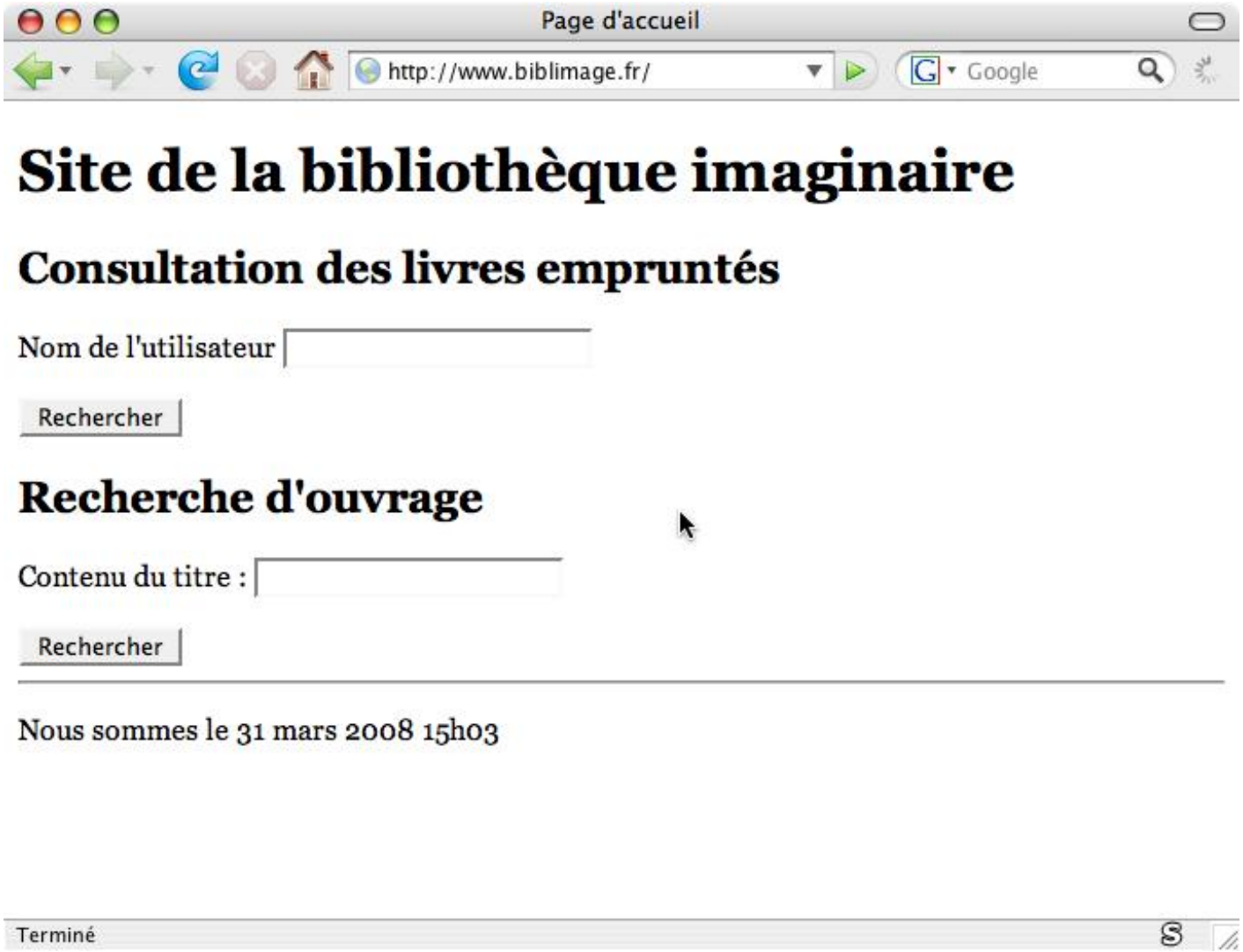
Page d'accueil



Emprunts d'un utilisateur

Retour d'un livre

Page d'accueil



Recherche par titre

Code de la page d'accueil

```
<%@ page language="java" contentType="text/html; charse
    pageEncoding="ISO-8859-1"%>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Page d'accueil</title></head>
  <body>
    <h2>Consultation des livres empruntes</h2>
      <form action="emprunts.jsp">
        <p>Nom de l'utilisateur
          <input name="nom" type="text"/></p>
        <button>Rechercher</button>
      </form>
    <h2>Recherche d'ouvrage</h2>
...
```

Ajout d'entête et de pied de page

Inclusion de fichiers grâce à JSP

Fichier `entete.jspf` :

```
<%@ page language="java"
         pageEncoding="ISO-8859-1"%>
<h1>Site de la bibliothèque imaginaire</h1>
```

Dans `index.jsp` :

```
...
  <body>
  <%@ include file="entete.jspf" %>
...
```

Fichier `piedDePage.jspf` :

```
<%@ page language="java"
         pageEncoding="ISO-8859-1"
         import="java.util.Date" %>
<hr/>
<% Date d = new Date(); %>
<p>Nous sommes le <%=d.toLocaleString()%></p>
```

Dans `index.jsp` :

```
...
  <%@ include file="piedDePage.jspf" %>
  </body></html>
```

Utilisation d'un paramètre

Dans la page `emprunts.jsp`

```
<% String nomUtil = request.getParameter("nom"); %>
...
  <h2>Livres empruntés par <%=nomUtil%></h2>
```

Remarque: possibilité d'utiliser

```
  <h2>Livres empruntés par ${param.nom}</h2>
```

Affichage conditionnel

Utilisation du `if` java

Dans la page `emprunts.jsp`

```
<% String nomUtil = request.getParameter("nom");
   if (nomUtil == null) { %>
     <!-- Cette partie est affichée quand le
          paramètre nom n'est pas donné -->
<% } else { %>
     <!-- Cette partie est affichée quand le
          paramètre nom est donné -->
<% } // fin du if %>
```

Remarque : voir aussi `<c:if ...>` de la bibliothèque de balises (taglib) core

Redirection

Dans la page `emprunts.jsp`

```
...
   if (nomUtil == null) { %>
     <!-- Pas de nom donné
          Retour à la page d'accueil -->
     <jsp:forward page="index.jsp" />
  <% } else { %>
...
```

Rappel: vecteurs et itérateurs

classe java `java.util.Vector` : "sac" d'éléments

- `Vector()` : crée un nouveau vecteur vide
- `void add(Object o)` : ajoute `o` dans le vecteur
- `Iterator iterator()` : retourne un itérateur sur le vecteur

classe java `java.util.Iterator` : permet de parcourir un ensemble d'élément

- `boolean hasNext()` : retourne `true` s'il y a encore des éléments à traiter
- `Object next()` : retourne le prochain élément à traiter

Utilisation

```
Iterator i = vecteur.iterator();
while ( i.hasNext() ) {
  Object o = i.next();
  // traitement de o
};
```

Création d'une liste dynamique

```
<%@ page import="java.util.*" %>
<% /*  code préalable permettant d'obtenir un
    *  vecteur v
    */ %>
<ul>
  <% Iterator i = v.iterator;
     while ( i.hasNext() ) {
       Object o = i.next(); %>
  <li> <%=o%> </li>
  <% }; %>
</ul>
```