

CONCEPTION ET DEVELOPPEMENT D'APPLICATIONS
INTERNET

CDAI 2 - JAVA SERVER PAGES (JSP)

Université Paris Dauphine

Master M2 MIAGE

Année 2014-2015

Bekhouché Abdesslem

Sobral Diogo



PLAN

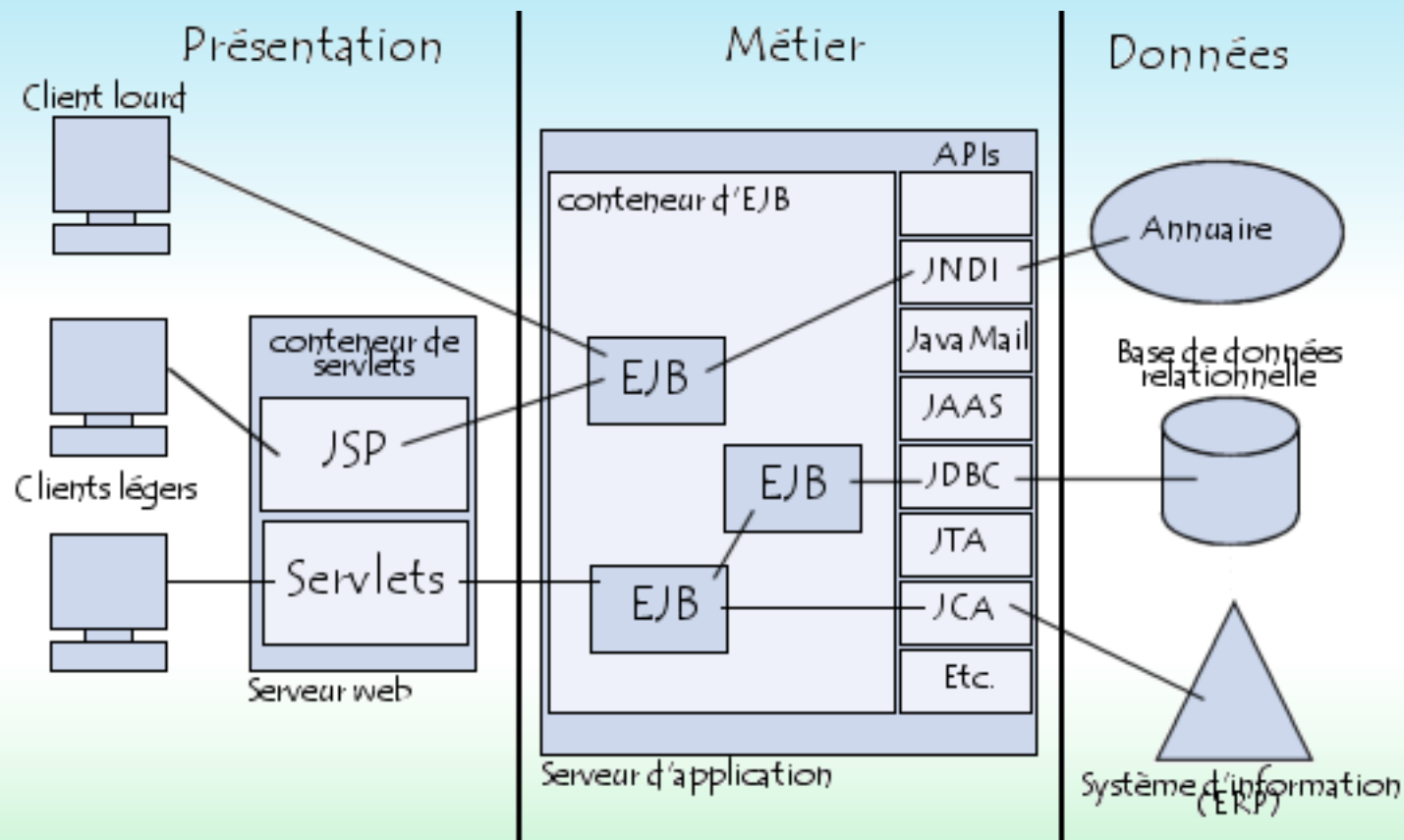
1. INTRODUCTION
2. JSP ET ARCHITECTURE N-TIERS
3. CARACTÉRISTIQUES DES JSP
4. FONCTIONNEMENT DES JSP
5. MISE EN OEUVRE
6. LES FORMULAIRES
7. GESTION D'ERREURS
8. INCLUSION DE JSP
9. DÉLÉGATION DE JSP
10. COMPLÉMENTS API
11. PACKAGING ET DÉPLOIEMENT
12. COMPARATIF JSP/SERVLET
13. CONCLUSION
14. WEBOGRAPHIE

1. INTRODUCTION

JAVA SERVER PAGES (JSP): HTML + JAVA

- Technologie java qui permet la génération de pages web dynamiques:
 - Code Java embarqué dans une page HTML;
- Composant de présentation JEE, comme les servlets:
 - JSP: peu de code java, beaucoup de HTML;
 - Servlet: beaucoup de code java, peu de HTML.

2. JSP ET ARCHITECTURE N-TIERS



3. CARACTÉRISTIQUES DES JSP

JSP est un programme Java qui s'exécute côté serveur Web:

- Servlet : programme "autonome" stocké dans un fichier .class sur le serveur;
- JSP: programmes source Java embarqué dans une page ".html".

Servlet et JSP:

- Exécutable avec tous les serveurs Web (Apache, IIS, ...);
- Auxquels on a ajouté un "moteur" de servlet/JSP (les plus connus : Tomcat, Glassfish);
- Les JSP sont compilées automatiquement en servlet par le conteneur Web.

3. CARACTÉRISTIQUES DES JSP

[Overview](#) [Package](#) **[Class](#)** [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

org.apache.jasper.runtime

Class HttpJspBase

java.lang.Object

└ javax.servlet.GenericServlet

└ javax.servlet.http.HttpServlet

└ org.apache.jasper.runtime.HttpJspBase

All Implemented Interfaces:

javax.servlet.jsp.HttpJspPage, javax.servlet.jsp.JspPage, java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

public abstract class **HttpJspBase**

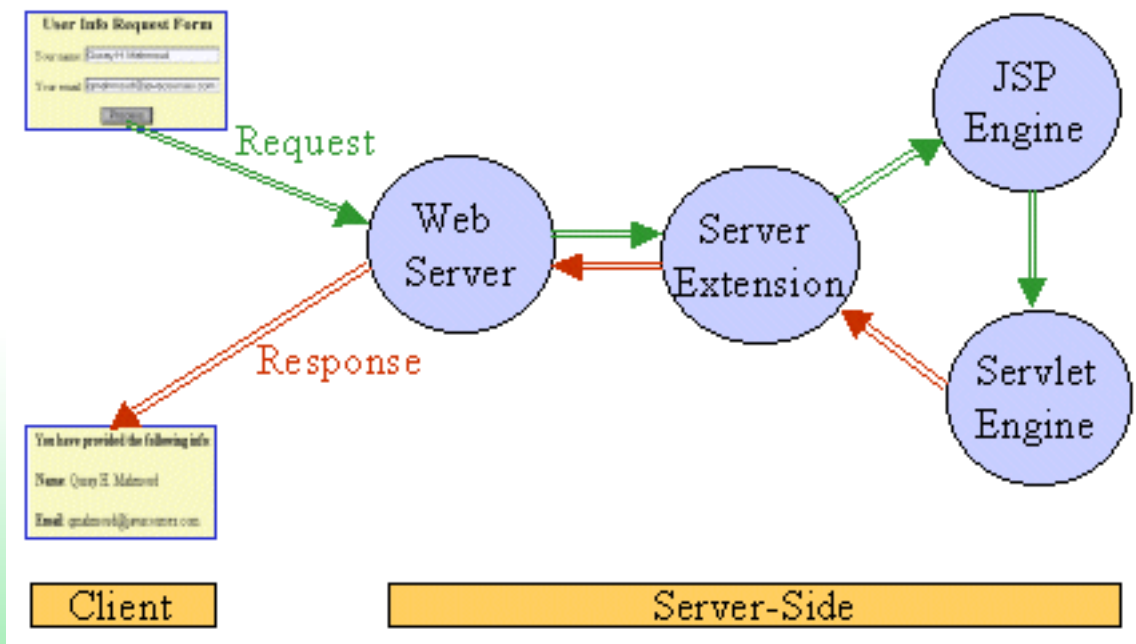
extends javax.servlet.http.HttpServlet

implements javax.servlet.jsp.HttpJspPage

This is the subclass of all JSP-generated servlets.

3. CARACTÉRISTIQUES DES JSP

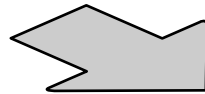
- Code java embarqué dans une page HTML entre les balises `<%` et `%>`;
- Fichiers avec extension ".jsp";
- Fichiers stockés côté serveur (web);
- Désignés par une url:
 - `http://monsite/majsp.jsp`
- Exécutés côté serveur à chaque chargement de l'url.



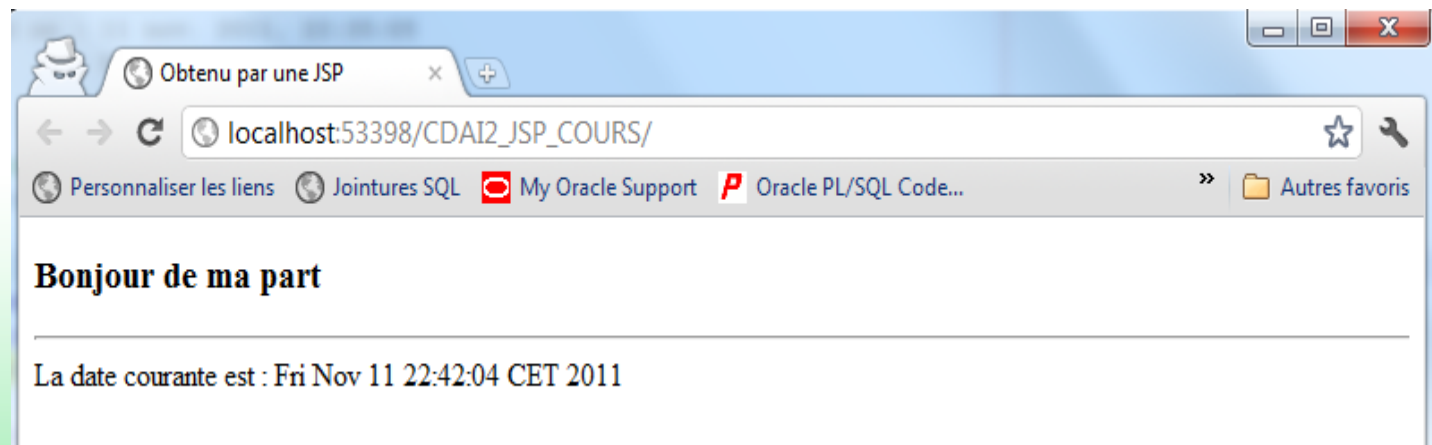
4. FONCTIONNEMENT DES JSP

Exemple:

```
<html>  
  <head>  
    <title>Obtenu par une JSP</title></head>  
  <body>  
    <h3>Bonjour de ma part </h3> <hr>  
    La date courante est : <% out.print(new java.util.Date()); %>  
  </body>  
</html>
```



Invocation => Exécution côté serveur



4. FONCTIONNEMENT DES JSP

Analyse de l'exemple:

```
<html>
```

```
<head>
```

```
<title>Obtenu par une JSP</title></head>
```



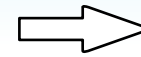
Code html

```
<body>
```

```
<h3>Bonjour de ma part </h3> <hr>
```

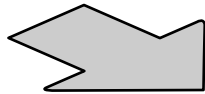
```
La date courante est : <% out.print(new java.util.Date()); %>
```

```
</body>
```



*Code Java
produit du code html*

```
</html>
```



Code renvoyé au client

```
<html>
```

```
<head>
```

```
<title>Obtenu par une JSP</title></head>
```

```
<body>
```

```
<h3>Bonjour de ma part </h3> <hr>
```

```
La date courante est : Fri Nov 11 22:42:04 CET 2011
```

```
</body>
```

```
</html>
```

5. MISE EN OEUVRE

Mécanismes de mise en oeuvre:

- Plusieurs zones `<%...%>` peuvent cohabiter dans une même page;
- Lors du premier chargement d'une JSP:
 - Génération d'une servlet à partir de la jsp;
 - Compilation de la servlet;
 - Instanciation de la servlet;
 - Délai d'attente lors de la première consultation;
 - En cas d'erreur de syntaxe, message envoyé au navigateur web:
 - Erreurs détectées uniquement à l'exécution !
- Lors des chargements suivants:
 - Exécution de la servlet dans un thread

5. MISE EN OEUVRE

Objets implicites pré-déclarés et utilisables dans les JSP:

out	le flux de sortie pour générer le code html
request	la requête qui a provoqué le chargement de la jsp
response	la réponse à la requête de chargement de la jsp
page	l'instance de servlet associée à la jsp courante (=this)
exception	l'exception générée en cas d'erreur sur une page
session	suivi de session pour un même client
application	espace de données partagé entre toutes les JSP

5. MISE EN OEUVRE

Compilation d'une JSP en servlet:

```
<html>
<head>
<title>Obtenu par une JSP</title></head>
<body>
  <h3>Bonjour de ma part </h3> <hr>
  La date courante est : <% out.print(new java.util.Date()); %>
</body>
</html>
```



*Code html généré
avec out.write()*



Code Java reporté

```
public final class ma_jsp /* ... */ {
    public void _jspService(HttpServletRequest request,
        HttpServletResponse response){
        /* ... */ response.setContentType("text/html"); /* ... */
        out.write("<html><body>\n"); /* ... */
        out.print(new java.util.Date());
        /* ... */
    }
}
```

5. MISE EN OEUVRE

Directive `<%= expr %>`:

- Génère l'affichage d'une valeur de l'expression `expr`;
- Raccourci pour `<% out.print(expr); %>`

```
<html>
  <head>
    <title>Obtenu par une JSP</title></head>
  <body>
    <h3>Bonjour de ma part </h3> <hr>
    La date courante est : <%= new java.util.Date() %>
  </body>
</html>
```

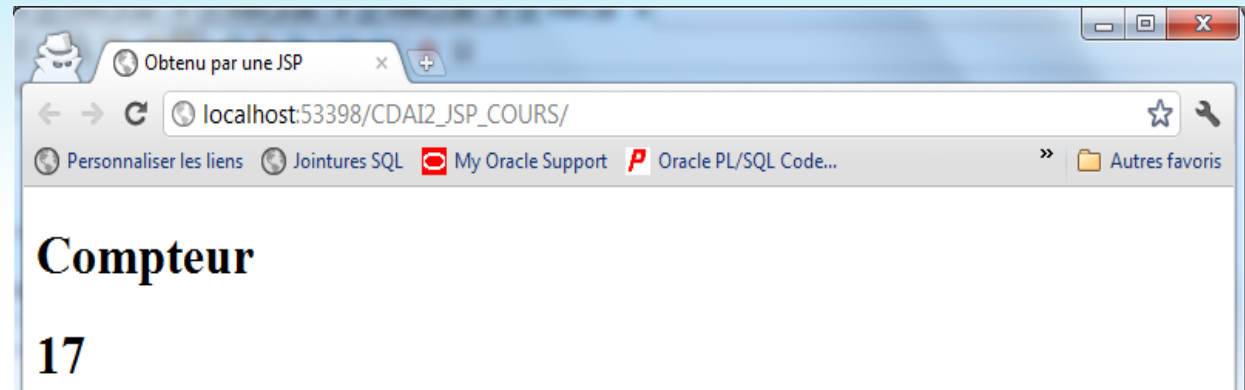
5. MISE EN OEUVRE

Méthodes et variables d'instance peuvent être associées à une page JSP entre les directives `<%! ... %>`

==> Méthodes et variables de la servlet générée

```
<html>
<h1>Compteur </h1>
<%! int cpt = 0;

    int getCpt(){
        return cpt ++;
    }
%>
<h1> <%= getCpt() %> </h1>
</html>
```



Attention: si redémarrage du conteneur web, alors perte de la valeur de la variable d'instance.

5. MISE EN OEUVRE

Méthode d'instance:

- Méthode d'instance de la servlet générée à partir de la jsp
=> pas d'accès aux objets implicites (out, request, etc).

Ce sont des objets définis dans la méthode principale de la servlet `_jspService()`.

Variables d'instance:

Attention `<%! int i = 0; %>` différent de `<% int i = 0; %>`

- `<%! expr %>` définit une variable d'instance (persiste entre 2 invocations de la jsp);
- `<% expr %>` définit une variable locale à la jsp (réinitialisée à chaque invocation de la jsp).

5. MISE EN OEUVRE

Directive `<%@ page ... %>`:

- Donne des informations sur la JSP (non obligatoire, valeurs par défaut).
- `<%@ page import="[nom package]">`
 - ex. `<%@ page import="java.io.*"%>`
 - Les "import" nécessaires au code Java de la JSP
- `<%@ page contentType="[type]"%>`
 - ex. `<%@ page contentType="text/html"%>`
 - Le type MIME du contenu retourné par la JSP
- `<%@ page isThreadSafe="[true (défaut)|false]" %>`
 - true : la JSP peut être exécutée dans plusieurs threads à la fois
- `<%@ page errorPage="[nom jsp]"%>`
 - ex. `<%@ page errorPage="err.jsp"%>`
 - Fournit l'URL de la JSP à charger en cas d'erreur
- `<%@ page isErrorPage="[true|false]" %>`
 - true : la JSP est une page invoquée en cas d'erreur

6. LES FORMULAIRES

Permettent aux clients de saisir des informations qui seront envoyées aux serveur.

Récupération des données d'un formulaire:

```
<html>
```

```
<body>
```

```
<h3>Hello, What's your name?</h3>
```

```
<form method="post" action="url-jsp">
```

```
  Name: <input type="text" name="username" size="25">
```

```
  Password: <input type="password" name="password" size="25">
```

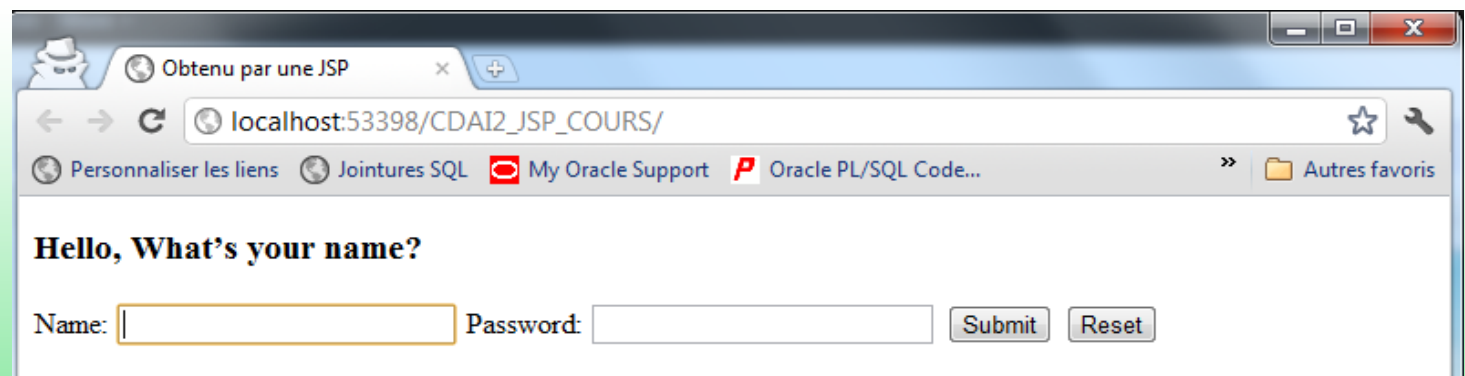
```
  <input type="submit" value="Submit" />
```

```
  <input type="reset" value="Reset" />
```

```
</form>
```

```
</body>
```

```
</html>
```



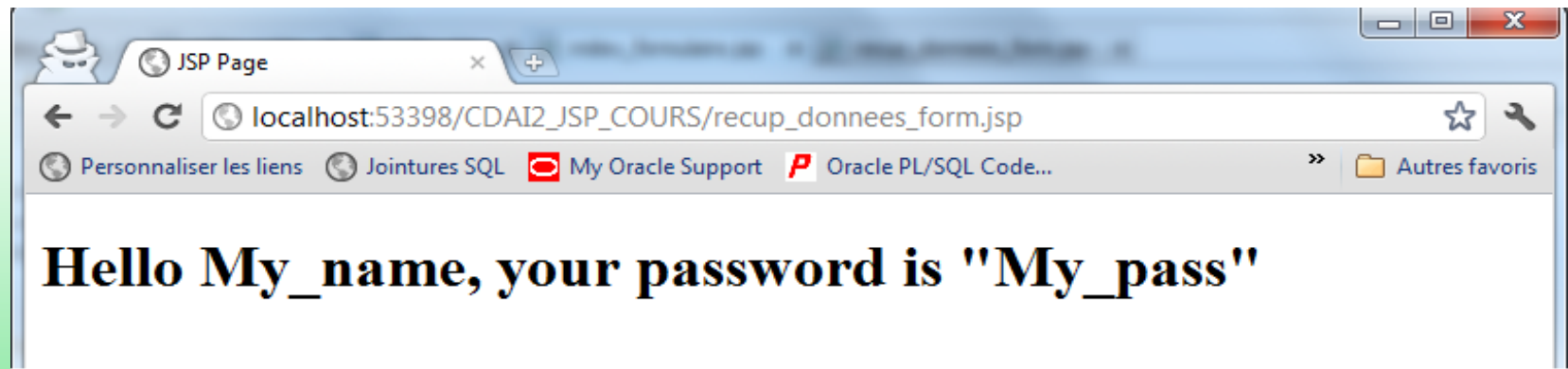
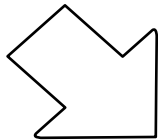
6. LES FORMULAIRES

Récupération des données d'un formulaire:

- Méthode String `getParameter(String)` de l'objet prédéfini `request`;
- Retourne le texte saisi ou null si le nom de paramètre n'existe pas.

```
<html>
<body>
  <h1>Hello <%= request.getParameter("username")
    + ", your password is \"
    + request.getParameter("password")
    + "\" %>

</h1>
</body>
</html>
```



7. GESTION D'ERREURS

1. Erreurs de syntaxe:

- Dans les directives JSP (*ex: oubli d'une directive %>*)
- Dans le code Java (*ex: oubli de ;*)

2. Erreur d'exécution du code Java (ex: NullPointerException)

Dans tous les cas, erreur récupérée dans le navigateur client:

- Conserver la page par défaut construite par le moteur;
 - En concevoir une adaptée aux besoins particuliers de l'application.
-
- Utilisation des directives:
 - `<%@ page errorPage="..." %>`: URL du gestionnaire d'erreurs;
 - `<%@ page isErrorPage="..." %>`: vrai si page est un gestionnaire.

7. GESTION D'ERREURS

EXEMPLE:

ERREUR

Si rand = 0



```
<html>
<body>
  <h1>Test d'une erreur</h1>
  <% int rand = (int) (Math.random() * 2);%>
  <h1> Resultat: <%= 12 / rand%>
</h1>
</body>
</html>
```



PAS D'ERREUR

Si rand <> 0

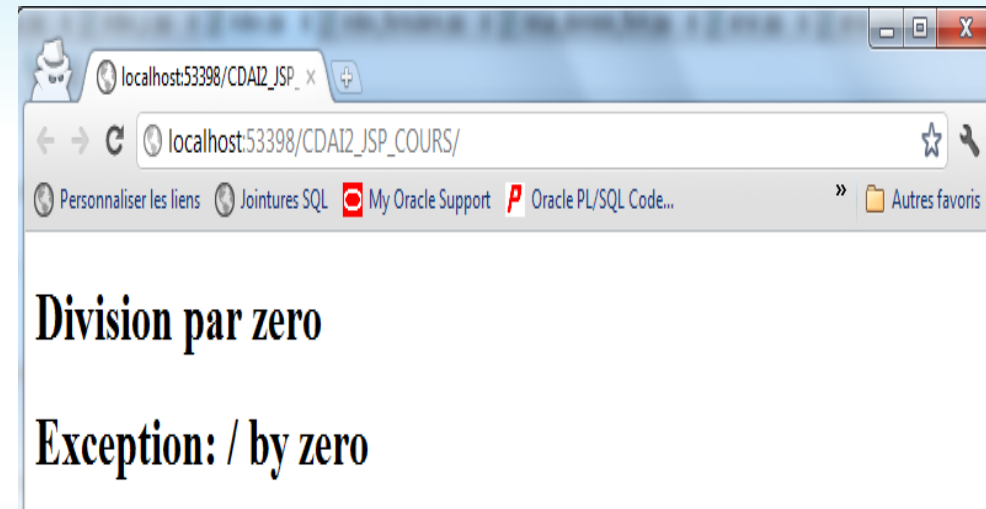


7. GESTION D'ERREURS

```
<html>
<body>
  <h1>Test d'une erreur</h1>
  <%@ page errorPage="error_catch.jsp" %>
  <% int rand = (int) (Math.random() * 2);%>
  <h1>Resultat:<%= 12 / rand%></h1>
</body>
</html>
```

Rattrape l'erreur si $\text{rand} = 0$
via l'objet prédéfini exception

```
<html>
<body>
  <%@ page isErrorPage="true"%>
  <h1>Division par zero</h1>
  <h1>Exception:<%= exception.getMessage()%></h1>
</body>
</html>
```



8. INCLUSION DE JSP

Directive `<jsp:include page="[url jsp]"/>`

Agrégation des résultats fournis par plusieurs JSP:

- Meilleure modularité;
- Meilleure réutilisation;

```
<html>
```

```
<body>
```

```
<h1>JSP principale</h1>
```

```
<jsp:include page="jsp_fille.jsp"></jsp:include>
```

```
</body>
```

```
</html>
```

```
<b>JSP incluse</b>
```

```
<p>
```

```
<%= (int) (Math.random() * 5)%>
```

```
</p>
```

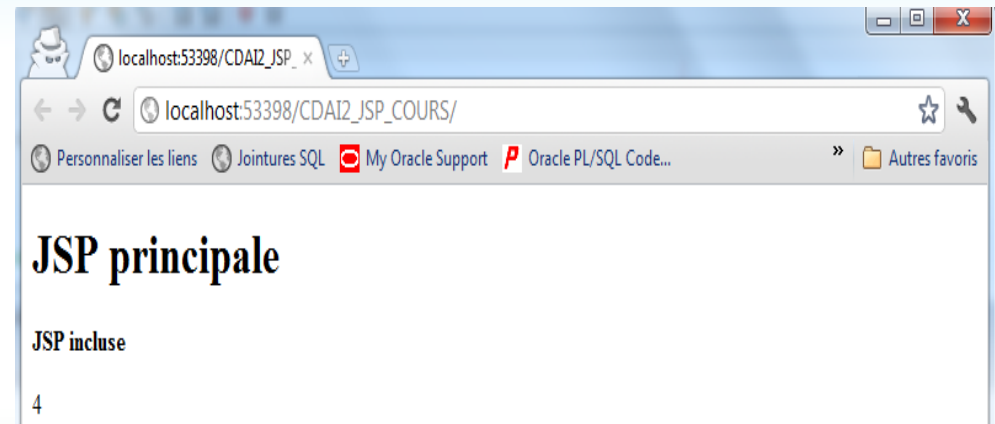
PAS DE `<HTML>` ni de `<BODY>`

8. INCLUSION DE JSP

Deux types d'inclusion de JSP:

1. `<jsp:include page="[url-jsp]"/>` inclusion dynamique (délégation de servlets => deux servlets);
2. `<%@ include file="..." %>` inclusion statique (inclusion au niveau HTML => une seule servlet).

```
<html>
<body>
  <h1>JSP principale</h1>
  <b>JSP incluse</b>
  <p>
    <%= (int) (Math.random() * 5)%>
  </p>
</body>
</html>
```



9. DÉLÉGATION DE JSP

Directives `<jsp:forward page="[url jsp]"/>`

Une JSP peut déléger le traitement d'une requête à une autre JSP: prise en compte complète de la requête par la JSP déléguée.

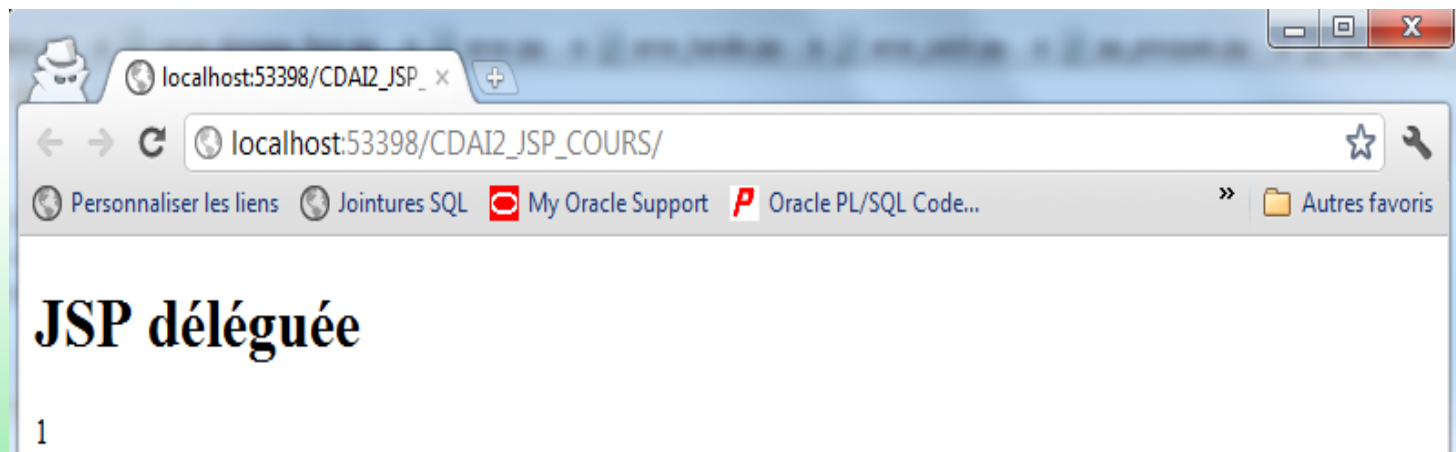
```
<html>
  <body>
    <h1>Ignore</h1>
    <jsp:forward page="jsp_deleguee.jsp" />
    <h1>Ignore</h1>
  </body>
</html>
```

```
<html>
  <body>
    <h1>JSP déléguée</h1>
    <p><%= (int) (Math.random() * 5)%></p>
  </body>
</html> NÉCESSITE <HTML> et <BODY>
```


9. DÉLÉGATION DE JSP

CODE RENVOYÉ AU CLIENT => CODE DE LA PAGE DÉLÉGUÉE:

```
<html>
  <body>
    <h1>JSP déléguée</h1>
    <p><%= (int) (Math.random() * 5)%></p>
  </body>
</html>
```



9. DÉLÉGATION ET INCLUSION DE JSP

Transmission de paramètres aux jsp incluses et déléguées:

- Utilisation de couples (name, value)
- Directive `<jsp:param name="..." value="..." />`
- Récupération des paramètres : comme si transmis via des formulaires

```
<html> <body>
<h1>JSP principale</h1>
<jsp:include page="inc.jsp">
<jsp:param name="nom" value="Bill" />
</jsp:include>
</body> </html>
```

```
<h2>Hello, <% request.getParameter("nom") %></h2>
```

10. COMPLÉMENTS API

L'objet “request” est une instance de `HttpServletRequest`:

- Hérite de `HttpRequest` (voir cours Servlet)
- Toute méthode des Servlets appellable sur request
- `String getProtocol()`
 - retourne le protocole implanté par le serveur (ex: HTTP/1.1)
- `String getServerName()` / `String getServerPort()`
 - retourne le nom/port de la machine serveur
- `String getRemoteAddr()` / `String getRemoteHost()`
 - retourne l'adresse/nom de la machine cliente (ayant invoqué la servlet)
- `String getScheme()`
 - retourne le protocole utilisé (ex. : http ou https) par le client

10. COMPLÉMENTS API

Suivi de session

- Objet prédéfini "*session*" de type HttpSession:
 - La session courante ou une nouvelle session
- Différence avec les servlets : une session est systématiquement associée à un client:
 - void setAttribute(String name, Object value)
 - Object getAttribute(String name)
 - void removeAttribute(String name)
 - java.util.Enumeration getAttributeNames()
 - void setMaxInactiveInterval(int seconds)
 - long getCreationTime() / long getLastAccessedTime()

10. COMPLÉMENTS API

Partage de données entre JSP

Notion de contexte d'exécution:

- Ensemble de couples (name, value) partagées par:
 - Toutes les JSP instanciées
 - Toutes les servlets instanciées
- Objet prédéfini "application" de type ServletContext

Méthodes appelables sur l'objet prédéfini "application":

- void setAttribute(String name, Object value)
- Object getAttribute(String name)
- void removeAttribute(String name)
- java.util.Enumeration getAttributeNames()

11. PACKAGING ET DEPLOIEMENT

Packaging de l'application Web:

- Dans un fichier war (Web Archive, standardisé)

/

- fichiers directement mappés (html et jsp)

- hello.jsp

- WEB-INF/

- web.xml

Pas nécessaire si pas de servlet

- sun-web.xml

Pas nécessaire (context-root=nom du war)

Ou autre fichier de déploiement spécifique au conteneur utilisé

- classes/

11. PACKAGING ET DEPLOIEMENT

TP: Net Beans et GlassFish

12. COMPARATIF JSP SERVLET

- JSP compilé en servlet
- servlet : possibilité de distinguer les requêtes HTTP (doPut, doGet, doPost, ...)
- JSP : beaucoup HTML, peu Java
- servlet : beaucoup Java, peu HTML
- contenu autre que HTML (PDF, GIF, Excel, ...) : servlet oui / JSP oui mais
- session, chaînage, redirection : oui dans les 2 cas : API vs directives
- servlet : pur Java : facilement éditable IDE
- JSP : plutôt éditeur de pages HTML
- servlet compilation avant déploiement / JSP après
- JSP à redéployer si erreur

12. CONCLUSION

- Servlet et Java Server Pages :
 - Permettent d'étendre le comportement des serveurs Web avec des programmes Java
- Résumé des fonctionnalités
 - Code embarqué dans un fichier HTML
 - Portabilité, facilité d'écriture (Java)
 - Notion de session au dessus d'HTTP
 - Persistance des données entre deux appels
 - Pas de persistance si serveur redémarre
 - JSP chargée et instanciée une seule fois
 - JSP exécutée dans des processus légers (threads)

13. WEBOGRAPHIE

<http://java.sun.com/products/jsp/index.html>

<http://jmdoudoux.developpez.com/cours/developpons/java/chap-jsp.php#jsp-1>

<http://www.commentcamarche.net/contents/j2ee/j2ee-intro.php3>

<http://cedric.cnam.fr/~farinone/IAGL/JSP.pdf>

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/50-java-servlet-jsp.pdf>