



# JavaServer Pages (JSP)

## Cours 5

JSP





## Prérequis pour ce cours

- Ce cours a trait à la programmation Java coté serveur
- Il faut connaître un minimum de technologie J2EE
- Il faut connaître les servlets



- Nécessité d'avoir des pages HTML dynamiques i.e. pages créées lors de la requête (météo, cours de la bourse, vente aux enchères, etc.)
- Technologie des server side include (ssi)
- Des pages HTML contiennent du code à exécuter lors de la requête
- Traiter par le serveur Web car ayant un suffixe caractéristique (.shtml, etc.)



- Le serveur Web, lorsqu'une telle page est demandée, passe la main au programme adéquat qui traite la partie de la page le concernant.
- Ce programme génère la partie dynamique en HTML
- La page HTML créée dans son ensemble est retournée au client Web.



- = JSP = la technique des ssi en Java
- = une page HTML contenant du code Java
- => meilleure division des tâches :
  - présentation générale par les graphistes
  - coté dynamique par des programmeurs (Java)



## Comment ça marche ?

- Concrètement :
  - toute la page HTML est convertie en une servlet
  - cette servlet est traitée par le moteur Java intégré au serveur Web (technologie des servlets) et retourne la page HTML construite



- Servlet = du code Java contenant de l'HTML
- JSP = une page HTML contenant du code Java
- Concrètement avec les JSP :
  - les parties statiques de la page HTML sont écrites en HTML
  - les parties dynamiques de la page HTML sont écrites en Java



- fichier `date.jsp`

```
<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>
```

- Traité quand le client demande l'URL de la JSP :  
`http://serveurWeb:<port>/.../date.jsp`



## Moteurs de JSP (et de servlets)

- Pour exécuter des JSP (resp. des servlets), il faut un moteur de JSP (resp. de servlets) dans le serveur Web.
- Ces moteurs sont des plug-in pour des serveurs Web existants
- Souvent des serveurs Web eux mêmes
- Deux candidats plug-in : JRun  
([www.allaire.com](http://www.allaire.com)), tomcat  
([jakarta.apache.org](http://jakarta.apache.org))



- Il existe des serveurs Web qui traitent les servlets et JSP :
  - IBM WebSphere
  - iPlanet Enterprise 4.x (ex Netscape)
- Voir à `java.sun.com/products/jsp`

- Développé par la communauté qui implémente les spécifs servlets et JSP.
- Téléchargeable (en version d'utilisation élémentaire) gratuitement à <http://jakarta.apache.org/downloads/bin/index.html>
- Plug-in de Apache version 1.3 ou plus, Microsoft IIS version 4.0 ou plus, Netscape Enterprise Server version 3.0 ou plus



## Tomcat (suite)

- Peut être utilisé comme serveur Web (bien qu'industriellement déconseillé)
- Existe pour plusieurs Unix et Win32
- Pour Win32 un `.zip` ou `.exe` de 8 Mo
- Lire `doc\readme` (et les fichiers du répertoire `doc`)
- Nécessite d'avoir un JRE 1.1 ou plus





- Des exemples de JSP (code + liens pour l'exécution) sont disponibles dans  
`REP_INSTALL_TOMCAT/webapps/examples/jsp/index.html`





- Il faut mettre les pages JSP dans un endroit particulier du serveur Web
- Cet endroit dépend du serveur Web et de sa configuration
- Pour tomcat en configuration standard,  
`http://serveurWeb/examples/jsp/date.jsp`  
~  
`REP_INSTALL_TOMCAT\webapps\examples\jsp\  
date.jsp`



- Le résultat de `date.jsp` est :

**Bonjour de ma part**

---

La date courante est : Sun Dec 23 18:04:36 CET 2001

- Une autre exécution donne une autre date => **dynamicité**



## Que s'est il passé ?

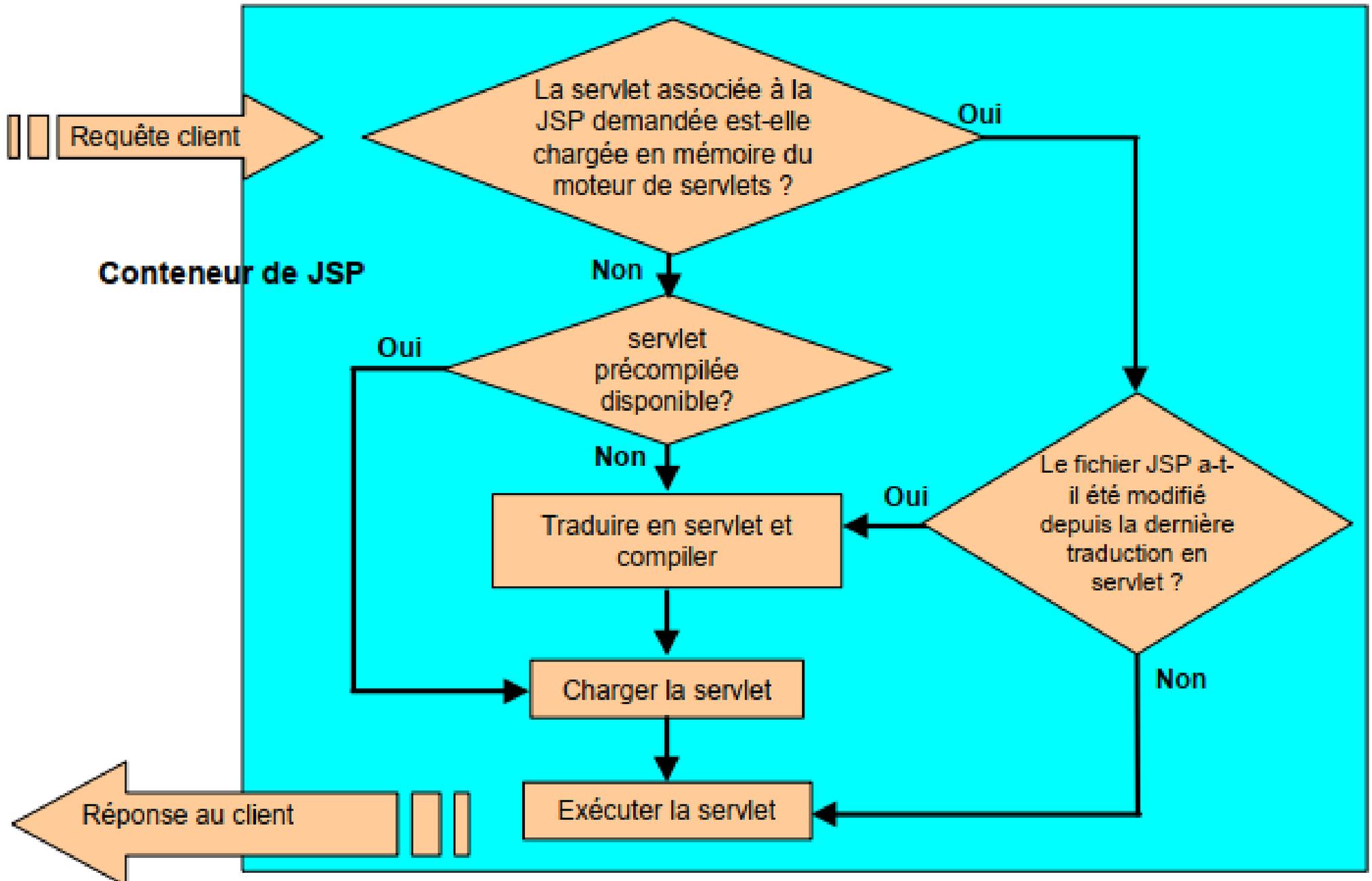
- Le moteur de JSP a construit une servlet  
(`_0002fjsp_0002fdate_0002ejspdate_jsp_0.java`  
dans  
`REP_INSTALL_TOMCAT\work\localhost_8080%2Fexamples`)
- Cette phase est parfois appelée la  
traduction de la JSP
- Puis a compilé et exécuté la servlet



# La servlet construite

```
package jsp;
...
public class _0002fjsp_0002fjsp_0002fdate_0002ejspdate_jsp_1 extends
HttpJspBase {
    ...
    public void _jspService(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
        ...
        _jspx_init();
        ...
        pageContext = _jspxFactory.getPageContext(...);
        session = pageContext.getSession();
        out = pageContext.getOut();
        // HTML
        // begin [file="C:\\...\\examples\\jsp\\date.jsp";from=(0,0);to=(4,24)]
        out.write("<html><head><title>Obtenu par une JSP</title></head>\r\n
<body>\r\n\r\n<h3>Bonjour de ma part</h3> <hr>\r\n
    La date courante est : ");
        // end
        //begin [file="C:\\...\\examples\\jsp\\date.jsp";from=(4,27)to=(4,49)]
            out.print( new java.util.Date() );
            // end
            // HTML
        // begin [file="C:\\...\\examples\\jsp\\date.jsp";from=(4,51);to=(6,7)]
            out.write("\r\n</body>\r\n</html>"); // end
            ...
        }
    }
}
```

# Algorithme d'exécution de la JSP





## 3 parties d'une JSP

- scriptlets  $\langle \% \quad \quad \% \rangle$
- déclarations  $\langle \% ! \quad \quad \% \rangle$
- expressions  $\langle \% = \quad \quad \% \rangle$

## Scriptlets

# Scriptlets `<% %>`

- contient du code Java
- insérer dans `_jspService()` de la servlet, donc peut utiliser `out`, `request`, `response`, etc.
- Exemple :

```
<%  
String[] langages = {"Java", "C++", "Smalltalk", "Simula 67"};  
out.println("<h3>Principaux langages orientés objets : </h3>");  
for (int i=0; i < langages.length; i++) {  
    out.println("<p>" + langages[i] + "</p>");  
}  
%>
```

## Déclarations `<% ! %>`

- Sont des déclarations Java.
- Seront insérées comme des membres de la servlet
- Permet de définir des méthodes ou des données membres
- Exemples :

```
<%!  
    int random4() {  
        return (int)(Math.random() * 4);  
    }  
%>
```

```
<%!  
    int nombreFetiché = 2;  
%>
```

## Expressions

# Expressions `<%= %>`

- En fait expression Java qui renvoie un objet `String` ou un type primitif.
- Un raccourci pour `<% out.println(...); %>`
- `<%= XXX %>` ~ `<% out.println(XXX); %>`
- attention au `;`
- est donc converti en `out.println(...)` dans la méthode `_jspService(...)` de la servlet.

```
La somme est: <%= (195 + 9 + 273) %>
```

```
Je vous réponds à l'adresse : <%= request.getParameter("email_address") %>
```



## Objets prédéfinis dans une JSP

- 3 objets peuvent être immédiatement utilisés dans une expression ou une scriptlet d'une JSP :
  - `out` : le canal de sortie
  - `request` (`HttpServletRequest`) : l'objet requête
  - `response` (`HttpServletResponse`) : l'objet réponse
- Il y en a d'autres
- Cf. ces mêmes objets dans une servlet



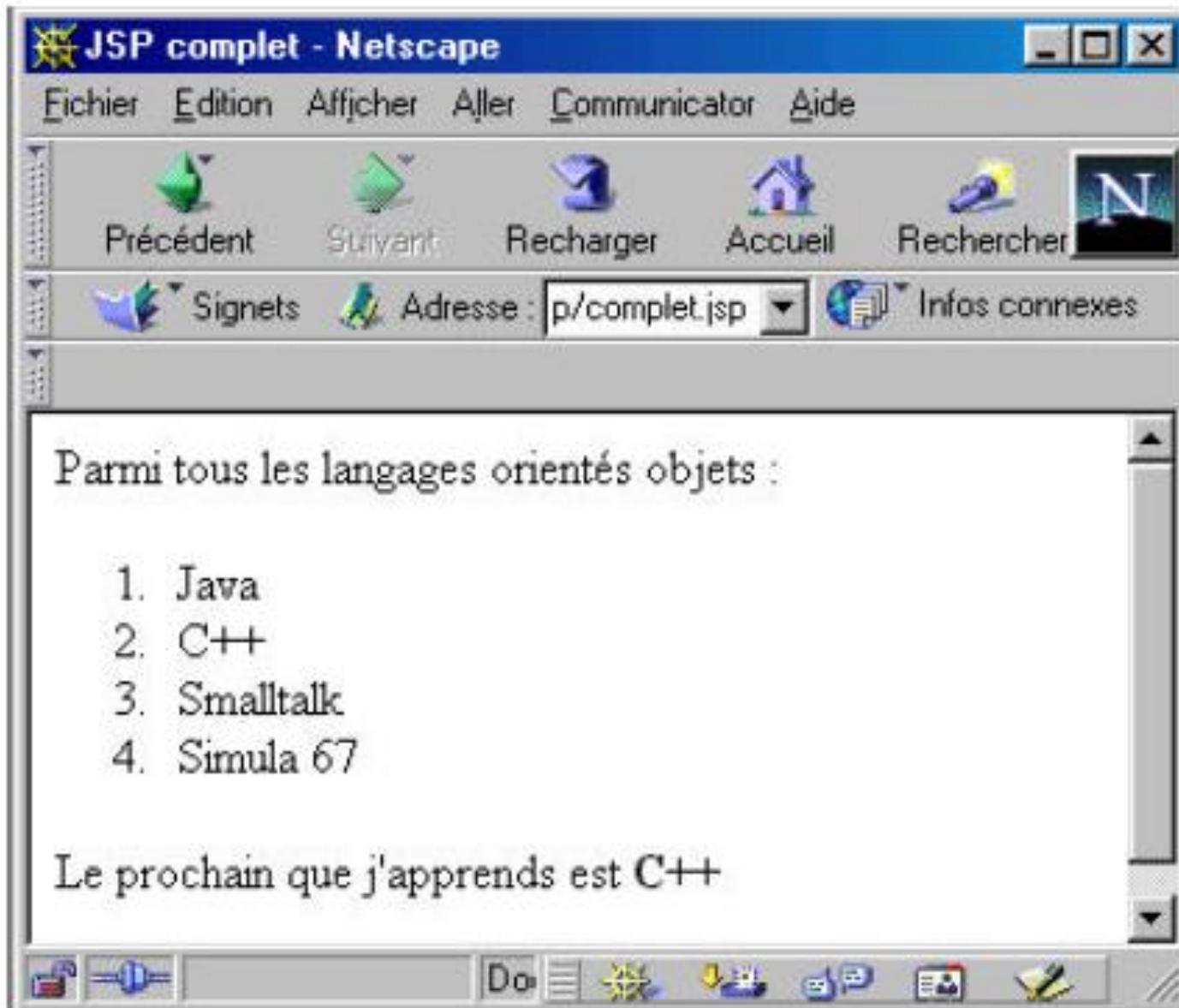
## Un exemple complet

complet.jsp

```
<html><head><title>JSP complet</title></head>
<body>
<%! String[] langages = {"Java", "C++", "Smalltalk", "Simula 67"};
    int random4() {
        return (int) (Math.random() * 4);
    }
%>
<p>Parmi tous les langages orientés objets :</p>
<ol>
<%
    for (int i=0; i < langages.length; i++) {
        out.println("<li>" + langages[i] + "</li>");
    }
%>
</ol>
<p>Le prochain que j'apprends est <b><%= langages[random4()] %> </b></p>
</body>
</html>
```

## Un exemple complet

complet.jsp





- La fenêtre de lancement du serveur Web donne des indications. Suivant les serveurs, une page HTML est retournée avec des indications.
- Ces éléments sont très souvent relatifs à la servlet et pas à la page JSP.
- Directives `<%@ page errorPage= ... %>`  
et  
`<%@ page isErrorPage="true" %>`



## Déboguer les JSP (suite)

- Un page JSP peut référencer une page erreur par `<%@ page errorPage="page.jsp"%>`
- La page erreur est indiquée par l'entête `<%@ page isErrorPage="true"%>`
- Si une exception est levée le traitement est dérouté vers la page erreur qui connaît la référence `exception` qui repère l'exception



## Déboguer les JSP : exemple

langages.jsp

```
<%@ page errorPage="erreur.jsp"%>
<%! String[] langages = {"Java", "C++", "Smalltalk", "Simula 67"};
%>
<p>Parmi tous les langages orientés objets :</p>
<ol>
<%
    // levée d'une ArrayIndexOutOfBoundsException
    for (int i=0; i < 7; i++) {
        out.println("<li>" + langages[i] + "</li>");
    }
%>
```



## Déboguer les JSP : exemple (suite)

erreur.jsp

```
<%@ page isErrorPage="true"%>
<html><body>
exception levée <b> <%= exception %> </b>
<hr>
<h3>trace de la pile</h3>
<pre>
<%
    java.io.PrintWriter myWriter = new java.io.PrintWriter(out);
    exception.printStackTrace(myWriter);
%>
</pre>
</body></html>
```



## Déboguer les JSP : exemple (fin)

Signets Adresse : <http://localhost:8080/examples/jsp/langages.jsp> Infos connexes

exception levée `java.lang.ArrayIndexOutOfBoundsException`

**trace de la pile**

```
java.lang.ArrayIndexOutOfBoundsException
  at jsp._0002fjsp_0002flangages_0002ejsplangages_jsp_0._jspServi
  at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.js
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
  at org.apache.jasper.servlet.JspServlet$JspCountedServlet.servi
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
  at org.apache.jasper.servlet.JspServlet$JspServletWrapper.servi
  at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServi
  at org.apache.jasper.servlet.JspServlet.service(JspServlet.java
```

## Enchaîner les pages

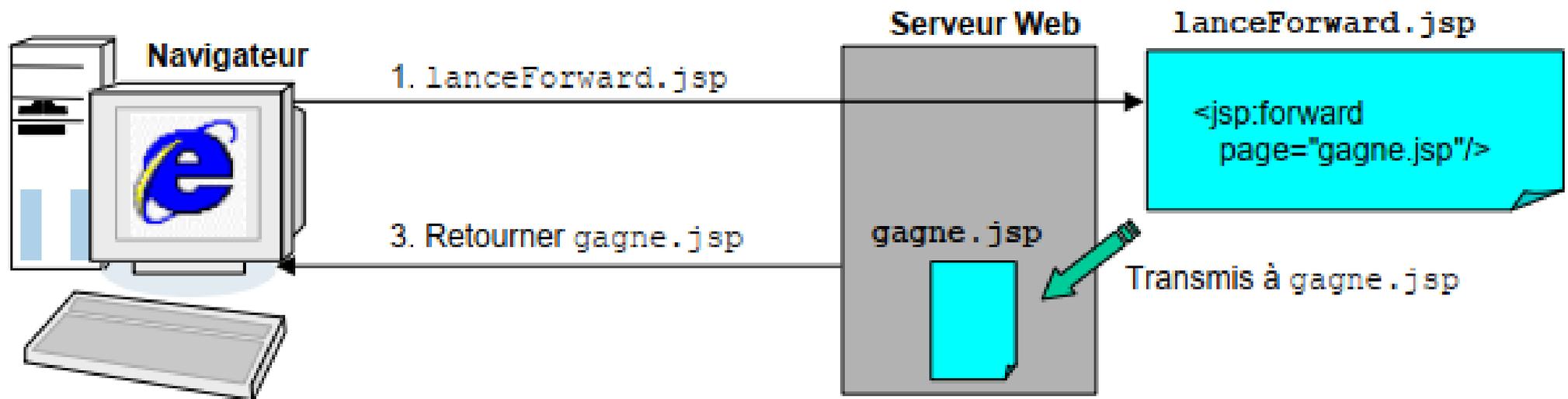
- Un page JSP peut en appeler une autre par la directive : `<jsp:forward>`
- Syntaxe :  
`<jsp:forward page="pageDeRedirection" />`

`lanceForward.jsp`

```
<% String repUtilisateur = request.getParameter("repTextField");
    int rep = Integer.parseInt(repUtilisateur);
        if ((rep % 2) == 0) {
%>
        <jsp:forward page="gagne.jsp"/>
<% } else { %>
        <jsp:forward page="perdu.jsp"/>
<% } %>
On n'affiche jamais cela
```

## Enchaîner les pages (suite)

- Après un `<jsp:forward>`, le traitement est entièrement pris en charge par nouvelle page

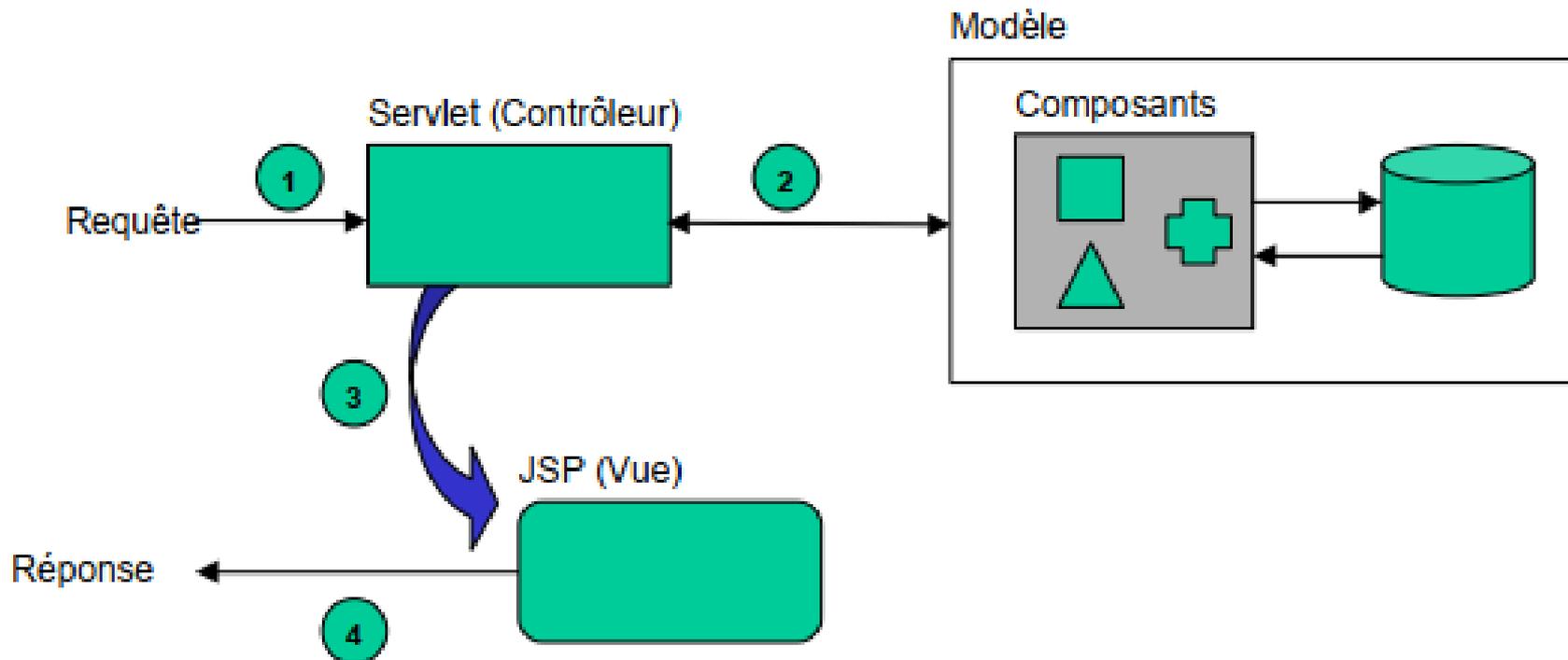




## Architecture MVC

- modèle = les données accédées par un code Java (JDBC, RMI, EJB, etc.)
- vues = JSP
- contrôleur = servlets

## Architecture MVC (suite)



- Syntaxe dans la servlet pour lancer la JSP :

```
public void doPost(HttpServletRequest request, HttpServletResponse response) {  
    ServletContext context = getServletContext(); // héritée de GenericServlet  
    RequestDispatcher dispatcher =  
        context.getRequestDispatcher("/maPageMiseEnForme.jsp");  
    dispatcher.forward(request, response);  
}
```

## Architecture MVC (suite)

- La servlet peut passer des valeurs à la JSP appelé grâce à `setAttribute()`

```
public void doPost(HttpServletRequest request, HttpServletResponse response) {  
    // appelle les méthodes sur les objets métiers  
    ArrayList theList = // un objet à passer  
    // ajoute à la requête  
    request.setAttribute("nomDelObjet", theList);  
    ServletContext context = getServletContext();  
    RequestDispatcher dispatcher = context.getRequestDispatcher("/jspAAppeler.jsp");  
    dispatcher.forward(request, response);  
}
```

- La JSP extrait les objets de `request` grâce à `getAttribute()`

```
<% ArrayList theList = (ArrayList)  
    request.getAttribute("nomDelObjet");  
// maintenant, utiliser l'ArrayList  
%>
```



## Bibliographie

- JavaServer Pages. Hans Bergsten; ed O'Reilly. ISBN 1-56592-746-X
- Technologie Apache/Tomcat à <http://jakarta.apache.org>



**QUESTIONS ?**