

Développement Web en JavaEE (Servlet & JSP)

Didier Donsez

Université Joseph Fourier (Grenoble 1)

PolyTech Grenoble LIG/ERODS

Didier.Donsez@imag.fr

Didier.Donsez@ieee.org

Au sommaire

- Rappel
- JavaEE et Web Application
- Servlets et Filters
- JSP et JSTL
- MVC et JSF

Répartition Client-Serveur

- Traitement coté client (User Agent)
 - extériorisation de l 'information
 - formats visualisables par le client
 - interactivité non lié au réseau
 - Interactivité requierant le serveur (RIA + Bayeux protocol)
- Traitement coté serveur
 - serveur applicatif
 - conserve les traitements en interne (business logic)
 - adapté le document retourné
au capacité du visualisateur
 - prise en compte des champs accept:
communiqués par le client
 - charge le serveur

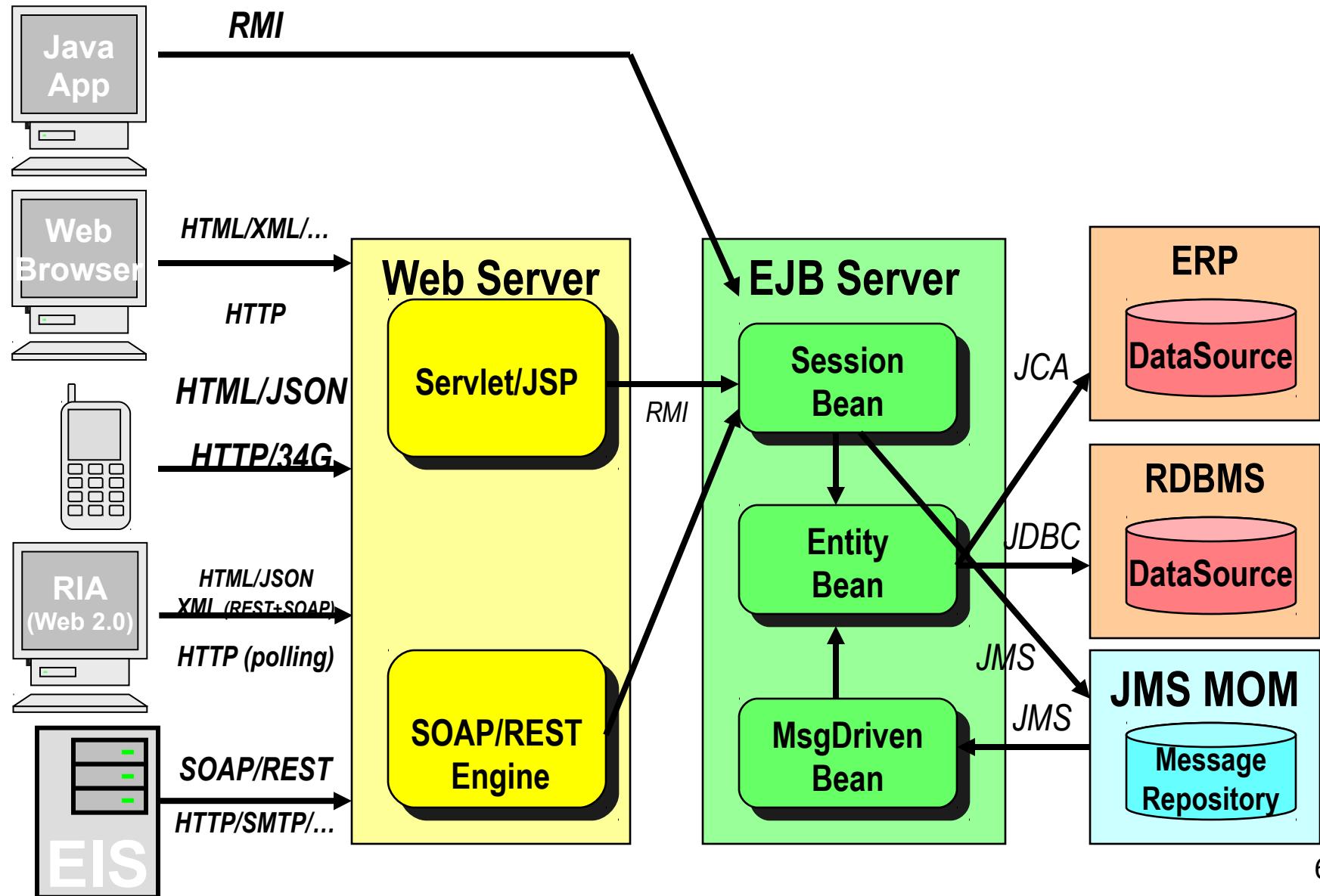
Le Scripting Serveur

- Utilisation
 - Traitement des formulaires par le serveur
 - Génération de pages HTML à la volée
 - requête sur des bases de données, ...
- Les Techniques
 - Script générant du HTML
 - CGI : *Common Gateway Interface*, Fast-CGI
 - NSAPI, ISAPI, Servlets
 - Script embarqué dans des pages HTML
 - SSI : *Server Side Include*
 - SSS : *Server Side Script*
 - Active Server Pages (MicroSoft), SSJ (NS), JSP (Sun), PHP, EJS (Express) ...

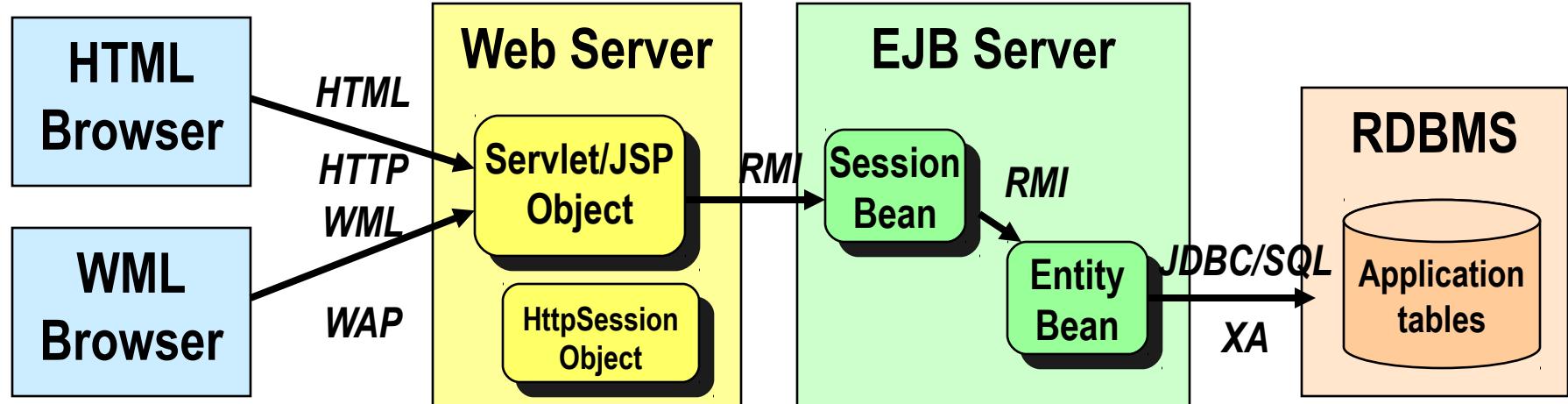
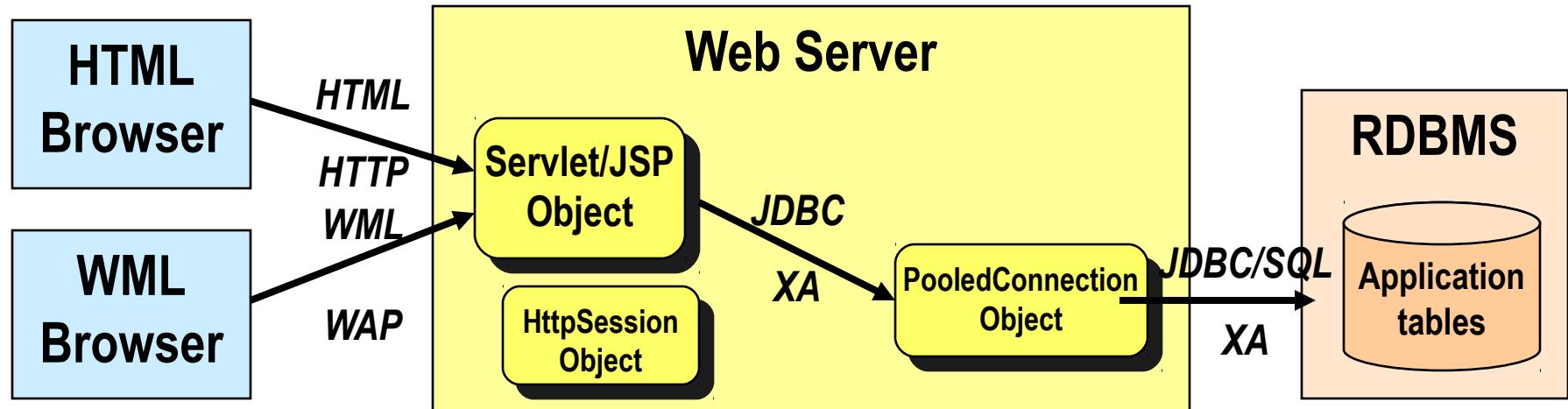
Les canevas Web

- Coccon
- Struts
- JSF
- Wicket
- AJAX
- GWT
- REST & WS
- ...
- Point commun : Servlet +/- JSP +/- JSTL

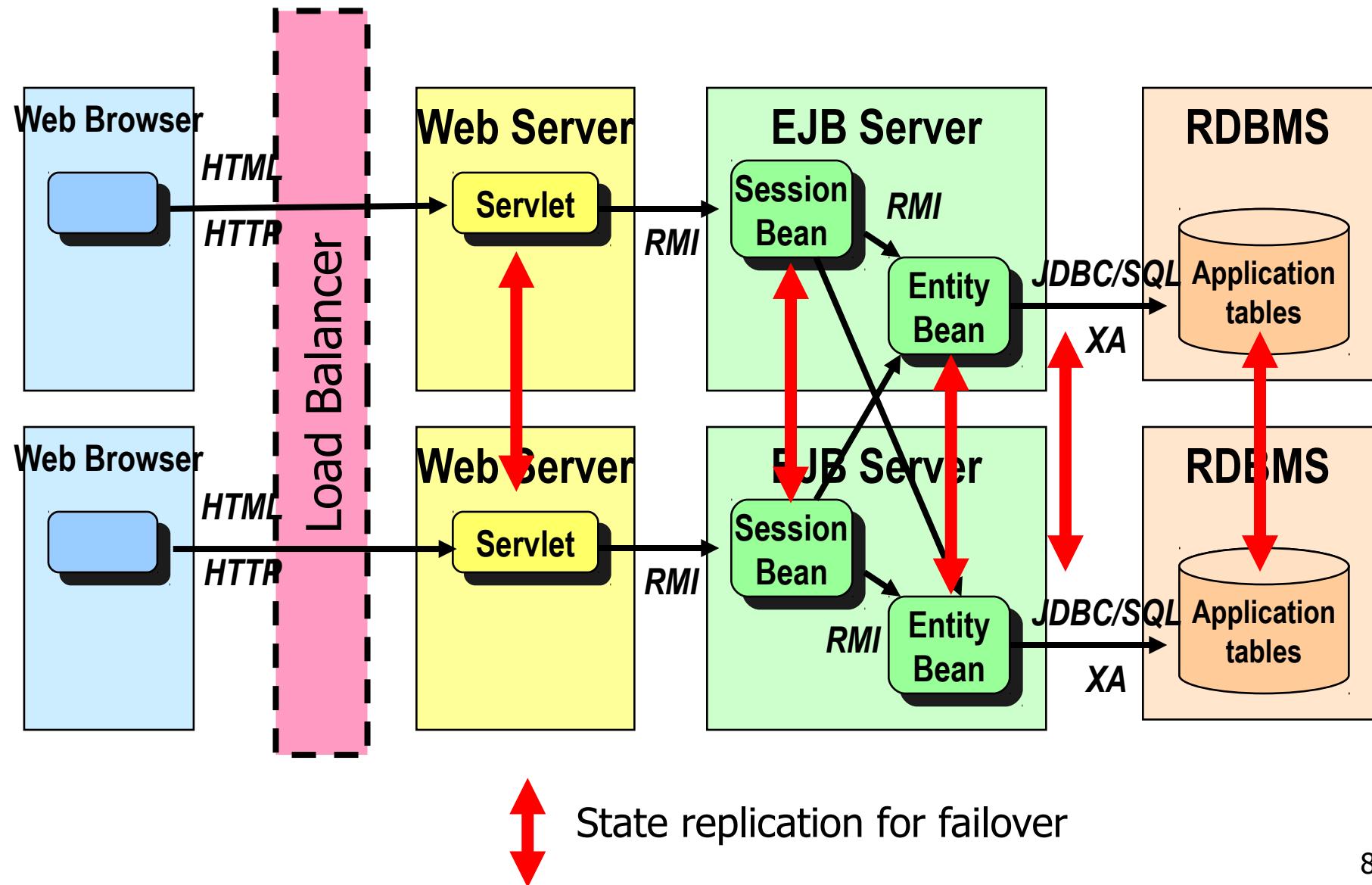
Architecture d'un service JavaEE (i)



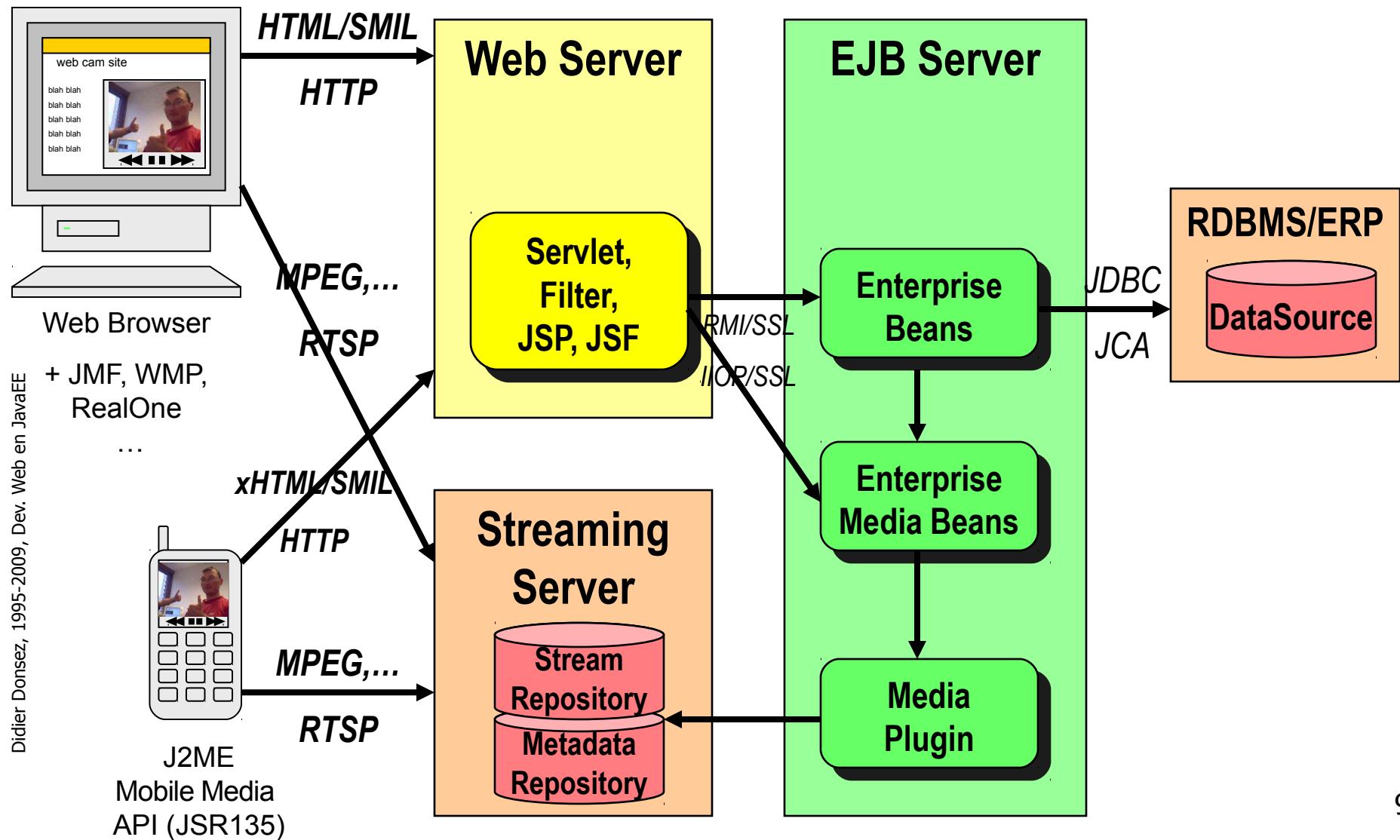
Architecture d'un service JavaEE (ii)



JavaEE en cluster + failover



JavaEE et EMB (Enterprise Media Beans)



Web application

- Partie « présentation » d'une application JavaEE
 - Servlet
 - Filter
 - JSP (Java Server Page)
 - JSTL (Java Server Tag Library)
 - JSF (Java Server Face)
- +
- Classes, bibliothèques (Jar File), ...
- Ressources (Document statiques, Images, Messages internationalisés (i18n), Propriétés ...)

Packaging d'une application Web en JavaEE (Web Component)

- **Web Component**
 - Une application Web (*.html, *.jsp, servlets, ...) packagée dans un .jar (.war) et est paramétrée dans le fichier WEB-INF/web.xml
 - L'application est installée dans le répertoire webapps du serveur web JavaEE
- **Structure d'une Web Application Archive (.war)**
 - * .html, * .png, * .jsp, ..., applets.jar, midlets.jar
 - **WEB-INF/web.xml**
 - Fichier de déploiement
 - Paramétrage des servlets, types MIME additionnels, ...
 - **WEB-INF/classes/**
 - .class des servlets et des classes (JavaBean, ...) associées
 - ressources additionnelles (localstring.properties, ...)
 - **WEB-INF/lib/**
 - jar additionnels provenant de tierce parties (comme des drivers JDBC, TagLib (jsf, ...), ...)
 - **WEB-INF/tlds/**
 - .tld décrivant les Taglibs

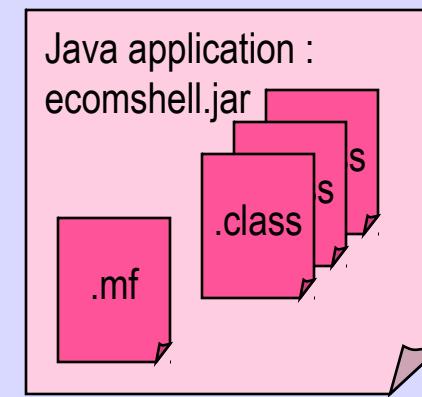
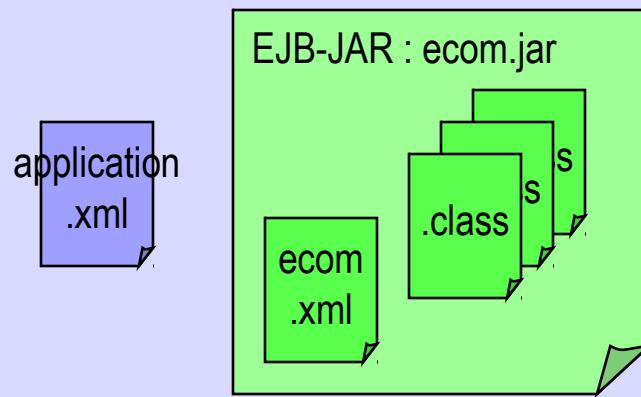
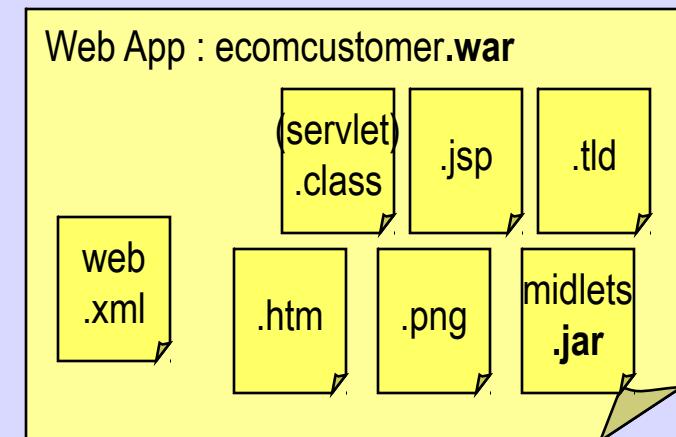
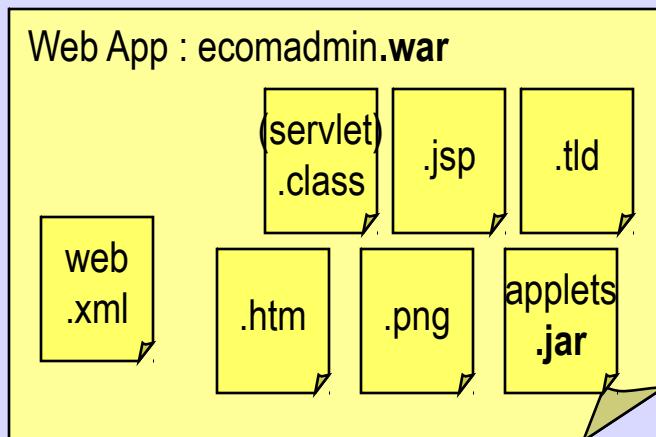
Le fichier *WEB-INF/web.xml*

- Fichier de déploiement de l'application
 - Correspondance URI -> classes Servlets
 - Valeurs d'initialisation des Servlets
 - Valeurs d'environnement
 - Ressources
 - Références vers EB Home, DataSource, Mail Session, ...
 - Types MIME supplémentaires
 - `text/vnd.wap.wml`, `text/vnd.sun.j2me.app-descriptor`, ...
 - Contraintes de sécurité
 - ...

Rappel

Packaging d'une application JavaEE complète

EAR : ecom.ear



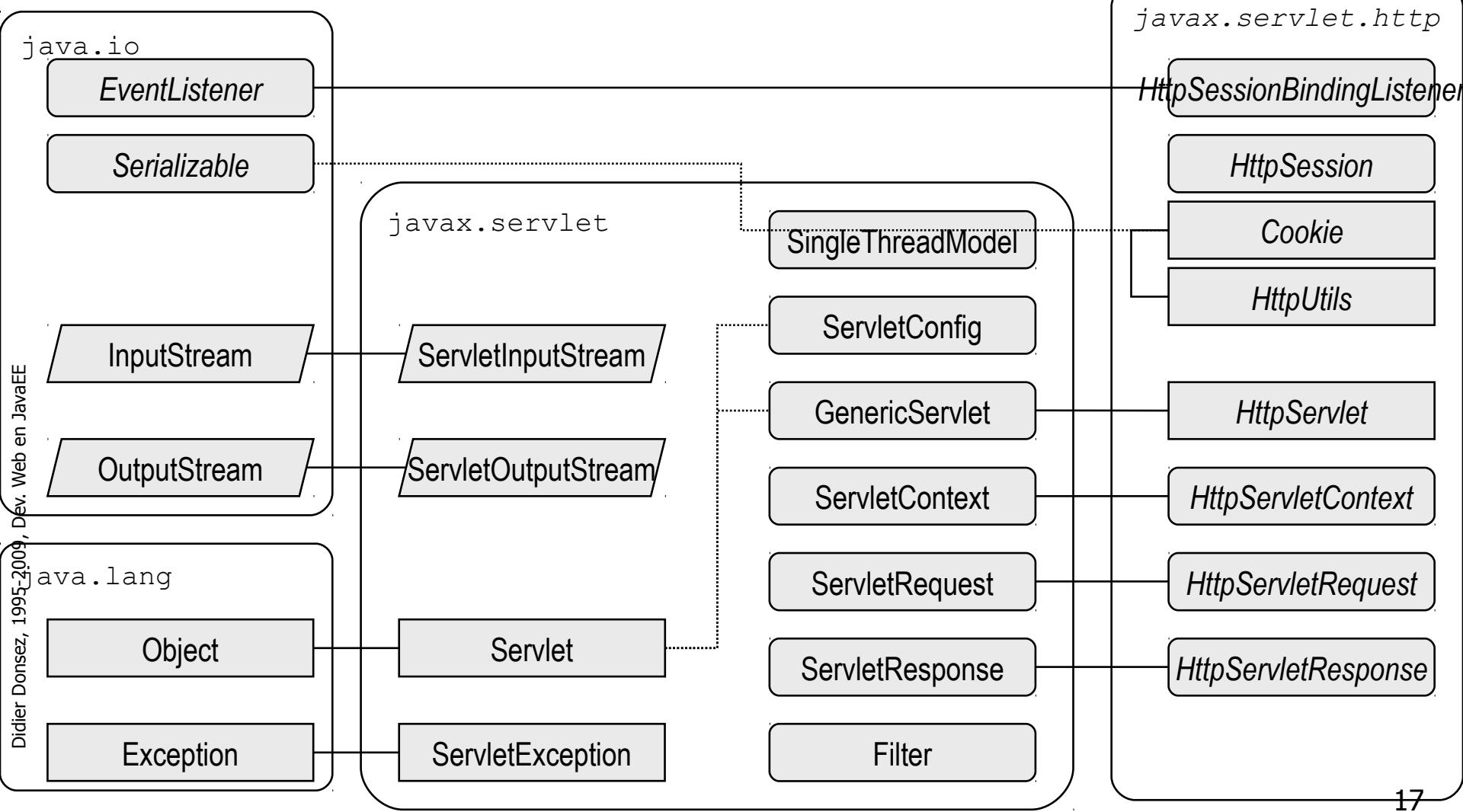
Servlet et Filter

Les Servlets

- Servlets
 - Scripts serveur écrit en Java
 - Servlets de Base : FileServlet, CGIServlet, ...
 - HttpServlet, SIPServlet, ...
 - Exécution dans un espace isolé (Web Application)

L'API Servlet

javax.servlet et javax.servlet.http



Exemple de Servlet *Hello World !*

```
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        String pname = req.getParameter("name"); // Get a parameter
        if(pname==null) pname="World !";
        res.setContentType("text/html"); // Set the Content-Type header
        PrintWriter out = res.getWriter(); // Get the output
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Hello, " + pname + "</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Hello, " + pname);
        out.println("</BODY></HTML>");
        out.flush();
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException { doGet(req, res); }
}
```

Servlets

Cycle de Vie (i)

① Chargement

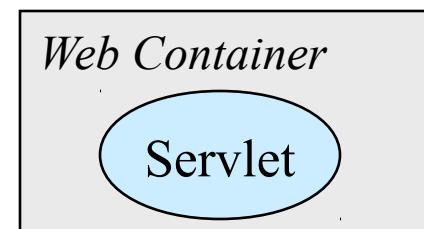
Remarque :
spécifié lors
du déploiement



② Initialisation

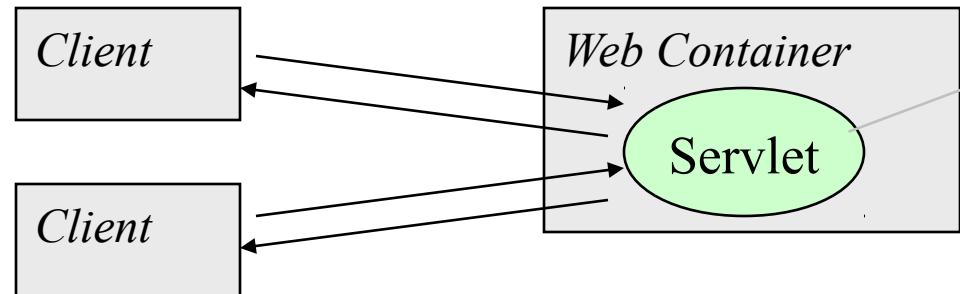
`init()`

Remarque :
spécifié lors
du déploiement



③ Traitement de
1 à N requêtes

`service()`

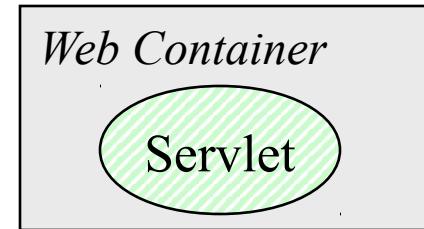


Remarque :
l'état est conservé
entre les requêtes

④ Destruction

`destroy()`

Remarque :
shutdown
du serveur, ...



La Requête

L'exploration du serveur

```
public class ServerSnoop extends GenericServlet {  
    public void service(ServletRequest req, ServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");PrintWriter out = res.getWriter();  
  
        out.println("Server: getServletContext().getServerInfo(): " +  
            this.getServletContext().getServerInfo());  
        Enumeration atts=this.getServletContext().getAttributeNames();  
        while(atts.hasMoreElements()){ // informations supplémentaires du constructeur  
            String att= (String)atts.nextElement();  
            out.println("Server: getServletContext().getAttribute(\""+att+"\")": " +  
                this.getServletContext().getAttribute(att));  
        }  
        out.println("Server: req.getServerName(): " + req.getServerName());  
        out.println("Server: req.getServerPort(): " + req.getServerPort());  
    } }
```

La Requête

L'exploration du client

```
public class ClientSnoop extends GenericServlet {  
    public void service(ServletRequest req, ServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");PrintWriter out = res.getWriter();  
  
        out.println("Client: req.getRemoteHost(" + req.getRemoteHost());  
        out.println("req.getRemoteAddr()" + req.getRemoteAddr());  
    } }
```

La Requête

La récupération des paramètres (POST ou GET)

```
public class ParameterSnoop extends HttpServlet {  
    public void doPost(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException { doGet(req,res); }  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
        PrintWriter out = res.getWriter();  
        out.println("Query String:"); out.println(req.getQueryString()); out.println();  
        out.println("Request Parameters:");  
        Enumeration pnames = req.getParameterNames();  
        while (pnames.hasMoreElements()) {  
            String name = (String) pnames.nextElement();  
            String values[] = req.getParameterValues(name);  
            if (values != null) {  
                for (int i = 0; i < values.length; i++) {  
                    out.println(name + " (" + i + "): " + values[i]);  
                }  
            }  
        }  
    }  
}
```

La Requête

La récupération des entêtes

```
public class HeaderSnoop extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
        PrintWriter out = res.getWriter();  
        out.println("Request Headers:");  
        out.println();  
        Enumeration enum = req.getHeaderNames();  
        while (enum.hasMoreElements()) {  
            String name = (String) enum.nextElement();  
            String value = req.getHeader(name);  
            if (value != null) {  
                out.println(name + ": " + value);  
            } } } }
```

La Requête

La récupération des Cookies

```
public class CookiesSnoop extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain"); PrintWriter out = res.getWriter();  
        out.println("Cookies:");  
        javax.servlet.http.Cookie[] cookies = req.getCookies();  
        for (int c=0;c<cookies.length;c++) {  
            out.print("Name:" + cookies[c].getName());  
            out.print("Value:" + cookies[c].getValue());  
            out.print("Domain:" + cookies[c].getDomain());  
            out.print("Path:" + cookies[c].getPath());  
            out.print("Secure:" + cookies[c].getSecure());  
            out.print("Version:" + cookies[c].getVersion());  
            out.print("MaxAge:" + cookies[c].getMaxAge());  
            out.println("Comment :" + cookies[c].getComment());  
        } } } }
```

La Requête

La récupération du flot d'entrée

```
public class InputStreamSnoop extends GenericServlet {  
    public void service(ServletRequest req, ServletResponse res)
```

```
        throws ServletException, IOException {
```

```
    String contenttype=req.getContentType();
```

```
    int contentlength=req.getContentLength();
```

- 2 flots de données possibles

```
    java.io.BufferedReader req.getReader();
```

```
    ...
```

```
    javax.servlet.ServletInputStream req.getInputStream();
```

```
    ...
```

- Utilisation exclusive sinon IllegalStateException

- Utilisation

- décodage d'un corps en multipart/data

La Réponse

La réponse simple

```
public class Hello extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/html");  
        PrintWriter out = res.getWriter();  
        out.println("<HTML>");  
        out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");  
        out.println("<BODY>");  
        out.println("<BIG>Hello World</BIG>");  
        out.println("</BODY></HTML>");  
    } }
```

La Réponse

Maintien de la connexion (KeepAlive)

```
public class KeepAliveHello extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/html");  
        // Set up a PrintStream built around a special output stream  
        ByteArrayOutputStream bytes = new ByteArrayOutputStream(1024);  
        PrintWriter out = new PrintWriter(bytes, true); // true forces flushing  
        out.println("<HTML>");  
        out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");  
        out.println("<BODY>");  
        out.println("<BIG>Hello World</BIG>");  
        out.println("</BODY></HTML>");  
        res.setContentLength(bytes.size()); // Set the content length  
        bytes.writeTo(res.getOutputStream()); // Send the buffer  
    } }
```

La Réponse

Les codes d 'état et la journalisation

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    ...
    try {// ...
        ...
    } catch (FileNotFoundException e) {
        // utilise le journal de GenericServlet
        this.log("Could not find file: " + e.getMessage());
        // envoi du code 404
        res.sendError(HttpServletResponse.SC_NOT_FOUND);

    } catch (IOException e) {
        // utilise le journal de ServletContext
        this.getServletContext().log(e, "Problem sending file");
        // envoi du code 500
        res.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,e.toString());
    }
}
```

La Réponse

L 'ajout d 'entête

```
public class ClientPull extends HttpServlet {  
    public void doGet(HttpServletRequest res, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain"); PrintWriter out = res.getWriter();  
        res.setHeader("Refresh", "10");  
        out.println(new Date().toString());  
    } }
```

La Session javax.servlet.http.HttpSession

Le suivi de session

- Le serveur maintient une session de 2 manières :
 - Cookie (Name: SESSIONID Value: To1010mC8601021835741167At)
 ⌚ les cookies peuvent être désactivés sur le navigateur
 - Réécriture des URLs
- Ouverture/récupération d 'une session
 - `javax.servlet.http.HttpSession session = req.getSession(false);`
 // la session est récupérée ou null si elle n 'existait pas déjà
 - `javax.servlet.http.HttpSession session = req.getSession(true);`
 // la session est récupérée ou ouverte si elle n 'existait pas déjà
- Invalidation d 'une session
 - `javax.servlet.http.HttpSession session = req.getSession(false);`
 - `session.invalidate(); // la session est invalidée (i.e. fermée)`

La Session javax.servlet.http.HttpSession

Information sur la session

```
javax.servlet.http.HttpSession session = req.getSession(false);
```

- L'identifiant

```
String sessionid= session.getId(); // par exemple: To1010mC8601021835741167At
```

- La date de création

```
long datecreation= session.getCreationTime(); // nb de ms depuis  
1/1/1970:00:00
```

- La date du dernier accès

```
long datelastaccess= session.getLastAccessedTime();
```

- Exemple

```
HttpSession session = req.getSession(true);  
if(session.getLastAccessedTime() - session.getCreationTime() > 5*60*1000 ) {  
    session.invalidate();  
}
```

La Session javax.servlet.http.HttpSession

Liaison d 'objets à une session

```
javax.servlet.http.HttpSession session = req.getSession(true)
```

- Ajout/remplacement d 'une valeur
void HttpSession.setAttribute(String name, Object value)
- Suppression d 'une valeur
void HttpSession.removeAttribute(String name)
- Récupération des valeurs/d 'une valeur
Enumeration<String> HttpSession.getAttributeNames()
Object HttpSession.getAttribute(String name)

La Session javax.servlet.http.HttpSession

Exemple de liaison d 'objets

```
import mycybermarket.Cart; ...  
public void doGet(HttpServletRequest req, HttpServletResponse res) ... {  
    Cart cart;  
    HttpSession session = req.getSession(true);  
    if((cart=(Cart)session.getAttribute("CART"))!=null) {  
        cart=CartFactory.create(); // new Cart( ... ); ou =cartHome.create();  
        session.setAttribute("CART",cart);  
    } ...  
    ...  
    if(action.equals("exit") {  
        cart.releaseProducts();  
        session.removeAttribute("CART");  
    }  
    ...
```

La Session

Observateurs sur la session

- Motivations
 - réagir face à des événements intervenants dans la/les sessions

- 4 interfaces Listeners

HttpSessionActivationListener

la session peut être passivée puis réactivée

HttpSessionListener

changement sur la liste des sessions actives de l'application Web.

HttpSessionAttributeListener

changement sur les attributs d'une des sessions de l'application Web.

HttpSessionBindingListener

un objet peut être notifié de sa liaison/rupture à un session

La Session

Exemple de Observateur de Liaison

```
class CartBindingListener implements HttpSessionBindingListener {  
    Cart cart=null;  
    public CartBindingListener( ... ) { this.cart = new Cart( ... ); }  
    public void valueBound(HttpSessionBindingEvent event) {}  
    public void valueUnbound(HttpSessionBindingEvent event)  
    { cart.releaseProducts();}  
    public void doGet(HttpServletRequest req, HttpServletResponse res) {  
        CartBindingListener cart;  
        HttpSession session = req.getSession(true);  
        CartBindingListener cart=(Cart)session.getValue("CART");  
        if(cart==null) { cart=new CartBindingListener( ... );  
            session.setAttribute("CART",cart); // valueBound est invoqué sur l'objet lié  
        } ...  
        session.removeAttribute ("CART"); // valueUnbound est invoqué sur l'objet lié
```

Accès à des ressources de l'application JavaEE

- Enterprise JavaBeans
- DataSources (Bases de données)
- Mail
- JMS (Queues/Topics)
- ...

Accès aux Enterprise Beans

■ WEB-INF/web.xml

```
<ejb-ref><!-- Remote EJB Reference -->
  <ejb-ref-name>ejb/Cart</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>myapp.beans.CartHome</home>
  <remote>myapp.beans.Cart</remote>
  <ejb-link>myapp.jar#Cart</ejb-link>
</ejb-ref>

<ejb-local-ref><!-- Local EJB Reference -->
  <ejb-ref-name>ejb/CartLocal</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>myapp.beans.CartLocalHome</local-home>
  <local>myapp.beans.CartLocal</local>
  <ejb-link>myapp.jar#Cart</ejb-link>
</ejb-local-ref>
```

Accès aux Enterprise Beans

- Code

```
Context ctx = new InitialContext  
// start transactions from client: get UserTransaction  
UserTransaction utx= (UserTransaction)  
    initialContext.lookup("java:comp/UserTransaction");  
  
// Connecting to CartLocalHome thru JNDI  
ICartLocalHome cartLocalHome =  
    ctx.lookup("java:comp/env/ejb/CartLocal");  
ICartLocal cartLocal = cartLocalHome.create(SessionId);
```

Accès aux Enterprise Beans via les annotations

```
@ManagedBean
public class CartServlet extends HttpServlet {
    @EJB(mappedName = "LocalCart")
    private ICart cart;
    void doGet(...) {
        ...
        cart.add(productId,qte, ... );
        ...
    }
}
```

Request Dispatch

- Redirige la traitement de la requête vers une autre servlet ou JSP
 - Utilisé pour le MVC (Controleur)
- Exemple

```
public class ForwardServlet extends HttpServlet {  
    public void doGet (HttpServletRequest request,  
                      HttpServletResponse response) {  
        // Set the attribute and forward to hello.jsp  
        request.setAttribute ("action", "hello");  
        ServletContext context=getServletConfig().getServletContext().  
            context.getRequestDispatcher("/hello.jsp").forward(request, response);  
    }  
}
```

Filtre (*Filter*)

- Traitement s'interposant entre le client et la ressource (Servlet, JSP, fichier) requise
 - en pré-traitement et/ou et post-Traitement
 - Chaînage de filtres (selon l'ordre de déclaration dans web.xml)
- Exemple de filtres
 - Authentification
 - Conversion (Image), Transformation (XSLT, FOP)
 - Chiffrage/Compression (Image, Données gzip)
 - Audit/Journalisation, Caching
 - Load balancing/Redirection (304Filter)
- API
 - **javax.servlet.Filter**
 - **init(FilterConfig config)**
 - **doFilter(request,response,chain) { ... chain.doFilter(request,response); ... }**
 - **destroy()**
 - Déploiement dans le descripteur web.xml

Exemple de Filtre

```
public class TraceFilter implements javax.servlet.Filter {  
    private String separator;  
    private FilterConfig filterConfig  
    public void init(FilterConfig filterConfig) throws ServletException {  
        this.filterConfig= filterConfig;  
        this.separator = filterConfig.getInitParameter("fieldseparator");  
    }  
    public void doFilter(ServletRequest req,ServletResponse res, FilterChain chain)  
        throws IOException, ServletException {  
        HttpServletRequest hreq = (HttpServletRequest)req;  
        long startTime = System.currentTimeMillis();  
        chain.doFilter(req, res);  
        long stopTime = System.currentTimeMillis();  
        log(hreq.getRemoteAddr()  
            + separator + hreq.getServletPath() + separator + (stopTime-startTime));  
    }  
    private void log(String msg){ filterConfig.getServletContext().log(msg); }  
}
```

Servlet 3.0 (JSR 315)

- Servlet asynchrone
 - Support pour Comet et WebSockets
- Annotations
 - @WebServlet, @WebFilter, @WebListener
 - @ServletSecurity, @MultipartConfig
- Fragment web.xml
- Support Multipart
- ...

Exemple de Servlet par Annotation

```
@WebServlet(name="MyHelloServlet",
    urlPatterns={"/hello*", "/hi"},
    initParams={ @InitParam(name="fr", value="Bonjour"),
@InitParam(name="es", value="Hola") }),
    asyncSupported=true
)
public class HelloServlet extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse res) {
        AsyncContext aCtx = req.startAsync(req, res);
        ...
    }
}
```

JSR 224 JAX-WS

- API pour les Web Services
- Exemple

`@WebService`

```
public class Hello {  
    @WebMethod  
    public String sayHello(String name) { return "Hello "+ name; }  
}
```

```
public class HelloAppClient {
```

`@WebServiceRef(wsdlLocation = "http://localhost:8080/helloservice-war/HelloService?WSDL")`

```
private static HelloService service;  
public static void main(String[] args) {
```

```
    Hello.endpoint.Hello port = service.getHelloPort();
```

```
    System.out.println(port.sayHello("world"));
```

```
}
```

Proxy client généré par
wsimport -p hello `http://...HelloService?WSDL`

JSR-311

Java API for RESTful Web Services

- Motivations
 - API Java for RESTful Services
 - Java 1.5 Annotation for POJO : @Path, @GET, @POST, ...

- Exemple

```
@Path("/library")
@Produces("application/json")
public class Library {
```

```
    @GET @Path("/book")
    public String getBooks() {...}
```

```
    @GET @Path("/book/{isbn}")
    public String getBook(@PathParam("isbn") String id) { ... }
```

```
    @POST @Path("/book/{isbn}")
    @Consumes("application/x-www-form-urlencoded")
    public String getBook(FormURLEncodedProperties formData) { ... }
```

```
    @POST @Path("/book/{isbn}")
    @Consumes("multipart/related")
    public String getBookMulti(MimeMultipart mimeMultipartData) { ... }
```

```
    @PUT @Path("/book/{isbn}")
    public void addBook(@PathParam("isbn") String id, @QueryParam("name") String name) {...}
```

```
    @DELETE @Path("/book/{id}")
    public void removeBook(@PathParam("id") String id {...}
}
```

JSP & JSTL

JSP (Java Server Page)

- Server Side Script
 - Insertion de SSS (syntaxe Java) dans les pages HTML
- Avantage par rapport aux servlets
 - Ecriture moins verbeuse Orientée *Web Designer*
 - Insérable par des outils auteurs dans le code de pages HTML
 - Extensible grâce aux JSTL
- Spécification
 - JSR-52
 - JSR-152 JavaServer Pages 2.0 Specification
- Implémentations
 - J2EE SDK et Jakarta JASPER/TomCat

Insertion des scripts

■ Directives

```
<%@page import="java.util.*" %>  
<%@taglib prefix="c" uri="WEB-INF/tld/core.tld" %>
```

■ Éléments de script

- *Scriptlets* <% code java %>
- *Déclarations* <%! Déclarations %>
- *Expressions* <%= expression %>

■ TagLib

```
<jsp:forward page="forward.jsp" />  
<jsp:include page="result.jsp" />  
<c:if test="${applicationScope:booklist == null}" >  
    <c:import url="/books.xml" var="xml" />  
    <x:parse xml="${xml}" var="booklist" scope="application" />  
</c:if>
```

Exemple de traitement d'un formulaire

<HTML>

<HEAD><TITLE>Hello</TITLE></HEAD>

<BODY>

<H1> Hello

<%

Scriptlet
(source Java)

String pname; // déclaration de variable

pname = request.getParameter("name"); // request : objet implicite

if (pname== null) { out.println("World"); } else {

%>

Mister <%=pname%>

<% } // fin du else %>

</H1>

</BODY></HTML>

Expression (EL)

Exemple de traitement d'un formulaire

```
<%@ method = "doPost" %>
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1> Hello
<%
String pname;      // déclaration de variable
pname = request.getParameter("name"); // request : objet implicite
if (pname== null) { out.println("World"); } else {
%
Mister <%=pname%>
<% } // fin du else %>
</H1>
</BODY></HTML>
```

Directives
<%@ varname="value" %>
content-type, import,
extends, implements,
method, language

JSP : Exemple avec une session

JSP listant un « caddie » virtuel

```
<html>
<jsp:useBean id="cart" scope="session" class="mycybermarket.MyCart" />
<jsp:setProperty name="cart" property="*" />
<%
    cart.processRequest(request);
%>
<br> You have the following items in your cart:
<ol>
<%      String[] items = cart.getItems();
        for (int i=0; i<items.length; i++) { %>
            <li> <%= items[i] %>
            <% } %>
</ol><hr>
<%@ include file ="catalog.html" %>
</html>
```

Classe de « caddie » utilisé par la JSP

```
package mycybermarket;  
import javax.servlet.http.*; import java.util.Vector; import java.util.Enumeration;  
public class MyCart {  
    Vector v = new Vector();  String submit = null;  String item = null;  
    private void addlItem(String name) { v.addElement(name); }  
    private void removeItem(String name) { v.removeElement(name); }  
    public void setItem(String name) { item = name; }  
    public void setSubmit(String s) { submit = s; }  
    public String[] getItems() { String[] s = new String[v.size()]; v.copyInto(s); return s; }  
    public void processRequest(HttpServletRequest request) {  
        // null value for submit - user hit enter instead of clicking on "add" or "remove"  
        if (submit == null) addlItem(item);  
        if (submit.equals("add")) addlItem(item);  
        else if (submit.equals("remove")) removeItem(item);  
        reset(); // reset at the end of the request  
    }  
    private void reset() { submit = null; item = null; }  
}
```

Génération des JSP

- Compilation des JSP en classes Java
 - génération et compilation d 'une classe étendant `HttpJspBase` à la première invocation.
- Au runtime
 - la servlet `JspServlet` invoque le compilateur Jasper puis charge et exécute la méthode `jspService` de la classe `HttpJspBase` générée
- Avant déploiement
 - Les JSP peuvent être aussi générées avant le déploiement (tâche `<jspc>`)

JSP - Exemple de Génération

Parsing du document .jsp par JspServlet

```
<HTML>
```

```
<HEAD><TITLE>Hello</TITLE></HEAD>
```

```
<BODY>
```

```
<H1> Hello
```

```
<%
```

```
String pname; // déclaration de variable
```

```
pname = request.getParameter("name"); // request : objet implicite
```

```
if (pname== null) { out.println("World"); } else {
```

```
%>
```

```
Mister <%=pname%>
```

```
<% } // fin du else %>
```

```
</H1>
```

```
</BODY></HTML>
```

```
_jspx_html_data[0]
```

```
_jspx_html_data[1]
```

```
_jspx_html_data[2]
```

chargement des fragments du document

```
public class jsp_0005chello_0002ejspHello_jsp_1 extends HttpJspBase {  
    static char[][] _jspx_html_data = null;  
    public jsp_0005chello_0002ejspHello_jsp_1( ) {  
    }  
    private static boolean _jspx_initited = false;  
    public final void _jspx_init() throws JspException {  
        ObjectInputStream oin = null;  
        int numStrings = 0;  
        try {  
            FileInputStream fin = new FileInputStream  
("work\\%3A8080%2Fdemo\\C_0003a.test.jspHello.dat");  
            oin = new ObjectInputStream(fin);  
            _jspx_html_data = (char[][]) oin.readObject();  
        } catch (Exception ex) {  
            throw new JspException("Unable to open data file");  
        } finally {  
            if (oin != null)  
                try { oin.close(); } catch (IOException ignore) {}  
        }  
    }  
}
```

JSP - Exemple de Génération la méthode `_jspService` (partie 1)

```
public void _jspService(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {

    JspFactory _jspxFactory = null; PageContext pageContext = null;
    HttpSession session = null;     ServletContext application = null;
    ServletConfig config = null;     JspWriter out = null;
    Object page = this;   String _value = null;
    try {
        if (_jspx_initited == false) { _jspx_init(); _jspx_initited = true; }
        _jspxFactory = JspFactory.getDefaultFactory();
        response.setContentType("text/html");
        pageContext = _jspxFactory.getPageContext
(this, request, response,"", true, 8192, true);
// objets implicites
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
// ...
    }
}
```

JSP - Exemple de Génération la méthode _jspService (partie 2)

```
// ...suite de la méthode jspService
    out.print(_jspx_html_data[0]);
    // begin [file="C:\\test\\jsp\\hello.jsp";from=(4,2);to=(8,0)]
        String pname; // déclaration de variable
        pname = request.getParameter("name"); // request : objet implicite
        if (pname== null) { out.println("World"); } else {
    // end
    out.print(_jspx_html_data[1]);
    // begin [file="C:\\ test \\jsp\\hello.jsp";from=(9,10);to=(9,15)]
        out.print(pname);
    // end
    out.print(_jspx_html_data[2]);
    // begin [file="C:\\ test \\jsp\\hello.jsp";from=(10,2);to=(10,5)]
        } // fin du else
    // end
    out.print(_jspx_html_data[3]);
} catch (Throwable t) {
    if (out.getBufferSize() != 0) out.clear();
    throw new JspException("Unknown exception: ", t);
} finally { out.flush(); _jspxFactory.releasePageContext(pageContext);
}
}
```

JSTL

Java Server Tag Library

- Motivations
 - Extension des JSP par des tiers
 - Développement modulaire
 - Développement sans Java (pour *Web Designer*)
- Principes
 - Le document contient des éléments (*tag*) supplémentaires
 - Le générateur JSP appelle des classes tiers lors de la transformation qui implantent Tag, IterationTag ou BodyTag.
- JSTL usuels

JSTL

Usage

- Directive

```
<%@ taglib uri="/WEB-INF/tlds/my-taglib.tld" prefix="my" %>
```

- Invocation

```
<my:mytag anattribute="avalue" >  
  <my:funcparam name="param1"  
    value="${my:myfunc(sessionScope.cart)}"/>  
  some enclosed text here  
</my:mytag>
```

- Déploiement

- WEB-INF/tlds/my-taglib.tld
 - Déclaration des classes implémentant la Tag Lib
- WEB-INF/lib/my-taglib.jar
 - Jar file contenant les classes d'implémentation

TagLib usuels

- Une liste non exhaustive ...
 - jsp (action standard)
 - c (core)
 - fmt (formattage et internationalisation)
 - xml (xml, xpath et xslt)
 - sql (sql)
 - jsf/core, jsf/html, ...
 - logic (Jakarta Struts)
 - tags (TagLib)
- à déployer dans votre application

TagLib usuels

- Une liste hébergé par Jakarta <http://jakarta.apache.org/taglibs/>
 - Benchmark (1.1)
 - BSF (1.1)
 - Cache (1.2)
 - DateTime (1.1)
 - I18N (1.1)
 - Input (1.1)
 - IO (1.1)
 - JMS (1.1)
 - JNDI (1.1)
 - Log (1.1)
 - Mailer (1.1)
 - Random (1.1)
 - RDC - Reusable Dialog Components (2.0)
 - Regexp (1.1)
 - Scrape (1.1)
 - String (1.1)
 - XTags (1.1)

TagLib usuels

jsp (les actions standards)

- **jsp:useBean** associe une instance d'objet Java(Bean)
- **jsp:setProperty** positionne la valeur d'une propriété d'un bean
- **jsp:getProperty** récupère la valeur de la propriété d'un bean
- **jsp:include** inclut le contenu du page statique ou dynamique
- **jsp:invoke** invoque l'exécution d'un fragment (JSP, ...)
- **jsp:forward** redirige le traitement de la requête à un autre script
- **jsp:body**
- **jsp:doBody** appelle le traitement des sous-éléments (cf: TagFile)
- **jsp:element** insère un élément (XML) dans le document
- **jsp:attribute** attribut d'un élément inséré ou
- **jsp:output** configure le prélude du document XML
- **jsp:param** paramètre (key/value) pour jsp:forward et jsp:include
- **jsp:text** ajoute le texte à la sortie
- **jsp:plugin** insère un objet externe (dépendant de l'UA)
- **jsp:fallback** alternative en cas d'échec du jsp:plugin
- **jsp:params** liste les jsp:param dans un jsp:plugin

Exemple de JSTL JSP

```
<%@ include file="header.jsp" %>
<jsp:useBean id="mydate" scope="session" class="my.Date" />
<jsp:setProperty name="mydate" property="date" param="${param.pdate}" />
<c:if test="${mydate.date eq 0}" >
    <jsp:forward page="error.jsp" />
</c:if>
<jsp:plugin type="applet" code="Clock.class" width="160" height="150" >
    <jsp:params>
        <jsp:param name="adate" value="${mydate.date}" />
    <jsp:params/>
    <jsp:fallback>
        Plugin tag OBJECT or EMBED not supported by browser.
    </jsp:fallback>
</jsp:plugin>
<jsp:include page="footer.jsp" flush="true"/>
```

TagLib usuels c (Core)

- URL="http://java.sun.com/jstl/core"
- Expression Language
 - Implicit objects
 - pageContext, param, paramValues, header, headerValues, cookie, initParam, pageScope, requestScope, sessionScope, applicationScope
- Variables
 - catch, out, set, remove
- Flow Control
 - if, choose (when, otherwise), forEach, forTokens
- URL Management
 - import (paramredirect, param), url (param)

Exemple d'EL (Expression Language)

- Arithmétique
 - \${3/0}Infinity
 - \${10%4} 2
 - \${10 mod 4} 2
 - \${1==2) ? 3 : 4} 4
- Comparaison
 - \${1 < 2} true
 - \${1 lt 2} true
 - \${100.0 == 100}true
 - \${100.0 eq 100}true
 - \${'hip' > 'hit'} false

Exemple d'EL (Expression Language)

■ Objets implicites

- \${param.custname} didier
- \${param["custname"]} didier
- \${header["host"]} mywebserver:8080
- \${header["accept"]} image/gif, image/jpeg, /*
- \${sessionScope.cart.numberOfItems}
- \${pageContext.request.contextPath}

■ Fonctions

- <%@ taglib prefix="my" uri="http://mysite/jsp-taglib"%>
- \${my:reverse(param["custname"])} reidid
- \${my:reverse(my:reverse(param["custname"]))} didier
- \${my:countVowels(param["custname"])} 3

Exemple de JSTL Core

- Variables

```
<c:set var="mycart" value="${sessionScope.cart}" />
<c:set var="bookId" value="${param.Remove}" />
<% mycart.remove(bookId); %>
<c:if test="${mycart.numberOfltems == 0}">
    <c:remove var="cart" scope="session" />
</c:if>
```

- URL

```
<c:url var="url" value="/catalog" ><c:param name="Add" value="${bookId}" /></c:url>
<a href=<c:out value='${url}'/>>Add</a>
```

- Control Flow

```
<c:forEach var="item" items="${sessionScope.cart.items}">
<c:choose>
    <c:when test="${item.category == 'book'}" > ... </c:when>
    <c:when test="${item.quantity == 'dvd'}" > ... </c:when>
    <c:otherwise> ... </c:otherwise>
</c:choose>
</c:forEach>
```

TagLib usuels

fmt (Internationalization)

- URI="http://java.sun.com/jstl/fmt"
- Locale
 - setLocale
- Message formatting
 - bundle, setBundle, message (param)
- Number and dateformatting
 - formatNumber, parseNumber,
 - formatDate, parseDate, setTimeZone, timeZone
- Example

```
<fmt:formatNumber value="${book.price}" type="currency"/>
<jsp:useBean id="now" class="java.util.Date" />
<fmt:message key="ShipDate"/>
<fmt:formatDate value="${now}" type="date" dateStyle="full"/>.
```

TagLib usuels

sql (SQL)

- URI="http://java.sun.com/jstl/sql"
- Exemple

...

```
<sql:query var="customers" dataSource="${dataSource}">
    SELECT * FROM customers
    WHERE country = 'Algeria'
    ORDER BY lastname
</sql:query>
```

```
<table>
    <c:forEach var="row" items="${customers.rows}">
        <tr>
            <td><c:out value="${row.lastName}" /></td>
            <td><c:out value="${row.firstName}" /></td>
            <td><c:out value="${row.address}" /></td>
        </tr>
    </c:forEach>
</table>
```

TagLib usuels x (XML)

- URI="http://java.sun.com/jstl/xml"
- Expression Xpath
 - Objets implicites
 - \$param:, \$header:, \$cookie:, \$initParam:, \$pageScope:, \$requestScope:,
\$sessionScope:, \$applicationScope:
 - Exemple
 - \$applicationScope.booklist/books/book[@id=\$param:bookId]
- Core
 - out, parse, set
- Flow Control
 - if, choose (when,otherwise), forEach
- Transformation
 - transform (param)

Exemple JSTL XML

- Parsing

```
<c:if test="${applicationScope:booklist == null}">  
  <c:import url="/books.xml" var="xml" />  
  <x:parse xml="${xml}" var="booklist" scope="application" />  
</c:if>
```

- Variables et Sorties

```
<x:set var="abook"  
      select="$applicationScope.booklist/books/book[@id=$param:bookId]" />  
<c:set var="price">  
  <x:out select="$abook/price"/>  
</c:set>  
<h2><x:out select="$abook/title"/></h2>  
<h2><x:set var="price" select="string($abook/price)"/></h2>
```

- Transformation

```
<x:transform xslt="book.xslt" xml="booklist.xml">  
  <x:param name="date" value="${now}" />  
</x:transform>
```

Exemple JSTL XML

```
<x:forEach var="book" select="$applicationScope:booklist/books/*">
<tr>
<c:set var="bookId"><x:out select="$book/@id"/></c:set>
<td>
<c:url var="url" value="/bookdetails" >
<c:param name="bookId" value="${bookId}" />
</c:url>
<a href=<c:out value='${url}'/>><x:out select="$book/title"/></a>
</td><td>
<c:set var="price"><x:out select="$book/price"/></c:set>
<fmt:formatNumber value="${price}" type="currency"/>
</td> <td>
<c:url var="url" value="/catalog" ><c:param name="Add" value="${bookId}" /></c:url>
<p><a href=<c:out value='${url}'/>><fmt:message key="CartAdd"/></a>
</td>
</tr>
</x:forEach>
```

JSPX

- Motivation
 - Génération dynamique de documents XML avec les JSP
 - XHTML, SVG, SMIL, ...
 - Utilisation des Namespaces
- xmlns:jsp="http://java.sun.com/JSP/Page"
 - jsp:root, jsp:declaration, jsp:scriptlet, jsp:expression, jsp:directive.page, jsp:directive.include
- Limites
 - Seulement une validation par la DTD
 - Pas de validation avec XML Schema
 - Pas de support du XInclude.

Exemple de JSPX

Génération d'un document HTML (i)

```
<?xml version="1.0"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="1.2">
<jsp:directive.page contentType="text/html"/>
<jsp:directive.page import="java.util.Date, java.util.Locale"/>
<jsp:directive.page import="java.text.*"/>

<jsp:declaration>
String getTimeStr(Locale l) {
    DateFormat df = SimpleDateFormat.getDateInstance(DateFormat.MEDIUM,
        DateFormat.MEDIUM, l);
    return df.format(new Date());
}
</jsp:declaration>

<html><head><title>Example JSP in XML format</title></head>
...
```

Exemple de JSPX

Génération d'un document HTML (i)

```
... <body>
```

```
This is the output of a simple JSP using XML format.
```

```
<br />
```

```
<jsp:scriptlet>
```

```
<![CDATA[ for (int i = 1; i<=10; i++) { out.println(i); if (i < 10) { out.println(", "); } } ]]>
</jsp:scriptlet>
```

```
<![CDATA[ <br><br> ]]>
```

```
<div align="left">
```

```
<jsp:expression>getDateTimeStr(request.getLocale())</jsp:expression>
</div>
```

```
<jsp:text> &lt;p&gt;This sentence is enclosed in a jsp:text element.&lt;/p&gt; </jsp:text>
```

```
</body>
```

```
</html>
```

```
</jsp:root>
```

Exemple de JSPX

Génération d'un document SVG

```
<svg xmlns="http://www.w3.org/2000/svg" width="450" height="500" viewBox="0 0 450 500"
      xmlns:c="http://java.sun.com/jstl/core_rt"
      xmlns:jsp="http://java.sun.com/JSP/Page">
<jsp:directive.page contentType="image/svg+xml" />
<c:set var="name" value='${empty param["name"] ? "JSPX" : param["name"]}'/>
<g id="testContent">
<g opacity="1.0" transform="translate(225, 250)" id="rotatedText">
<c:forEach var="i" begin="1" end="24">
  <jsp:text><![CDATA[<g opacity="0.95" transform="scale(1.05) rotate(15)">]]></jsp:text>
  <text x="0" y="0" transform="scale(1.6, 1.6)" fill="DarkSlateBlue"
        text-anchor="middle" font-size="40" font-family="Serif" id="words">${name}</text>
</c:forEach>
<c:forEach var="i" begin="1" end="24"><jsp:text><![CDATA[</g>]]></jsp:text></c:forEach>
<text style="font-size:75;font-family:Serif;fill:white" text-anchor="middle"> ${name}</text>
</g></g>
</svg>
```

Tag File (JSP 2.0)

- Motivation
 - Allows page authors to author tag extensions using own JSP syntax.

Exemple de Tag File

Directives
taglib, tag, include,
attribute, variable

```
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>
<tags:panel color="#ff8080" bgcolor="#ffc0c0" title="Panel 1">
    Hello<br/>
    ${empty param.custname ? "World" : param.custname} !
</tags:panel>
<br>
<tags:panel color="#8080ff" bgcolor="#c0c0ff" title="Panel 2">
    Second panel.<br/>
    <tags:panel color="#ff80ff" bgcolor="#ffc0ff" title="Inner Panel">
        A panel in a panel.
        <tags:panel color="#80ff80" bgcolor="#c0ffc0" title="Inner Inner Panel">
            A panel<br>in a panel<br>in a panel.
        </tags:panel>
    </tags:panel>
    Second panel.<br/>
</tags:panel>
```

```
<!-- /WEB-INF/tags/panel.tag -->
<%@ attribute name="color" %>
<%@ attribute name="bgcolor" %>
<%@ attribute name="title" %>
<table border="1" bgcolor="\${color}">
    <tr>
        <td><b>\${title}</b></td>
    </tr>
    <tr>
        <td bgcolor="\${bgcolor}">
            <jsp:doBody/>
        </td>
    </tr>
</table>
```

Panel 1
Hello
didier !

Panel 2
Second panel.

Inner Panel
A panel in a panel.

Inner Inner Panel
A panel
in a panel
in a panel.

Second panel.

Exemple de TagLib (i)

Insertion d'une table à partir d'une requête SQL !

```
<%@ taglib uri="/WEB-INF/tlds/sql-taglib.tld" prefix="sql" %>
<HTML><HEAD><TITLE>SQLTagLib
Example</TITLE></HEAD><BODY>
<sql:open
cnxid="cnx1"
driver="com.mckoi.JDBCDriver"
url="jdbc:mckoi://localhost/" user="admin" password="motpasse"
/>
<sql:query cnxid="cnx1" output="text/html">
  select * from accountsample_ where balance_ > ${param["balance"]}
</sql:query>
<sql:close cnxid="cnx1" />
</BODY></HTML>
```

Exemple de TagLib (II)

Insertion d'une table à partir d'une requête SQL !

The screenshot shows a Microsoft Internet Explorer window titled "SQL TagLib - Microsoft Internet Explorer". The address bar contains the URL "http://127.0.0.1:8080/sql/query.jsp?balance=390.0". The main content area displays a table with three columns: "accno_", "customer_", and "balance_". The table has four rows with the following data:

accno_	customer_	balance_
102	Alexandre Dumas fils	400.0
103	Conan Doyle	500.0
106	Alphonse de Lamartine	650.0

Déploiement d'une TagLib (i)

- WEB-INF/web.xml

```
<jsp-config>
```

```
<taglib>
```

```
  <taglib-uri>http://www-adele.imag.fr/j2ee/sql-taglib</taglib-uri>
```

```
  <taglib-location>/WEB-INF/tlds/sql-taglib.tld</taglib-location>
```

```
</taglib>
```

```
<jsp-config>
```

Déploiement d'une TagLib (ii)

- WEB-INF/ /WEB-INF/tlds/sql-taglib.tld

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>1.2</jsp-version>
  <short-name>sql</short-name>
  <uri> http://www-adele.imag.fr/j2ee/sql-taglib </uri>
  <tag>
    <name>open</name>
    <tag-class>fr.imag.adele.taglib.sqltaglib.DBOpenTag</tag-class>
    <attribute>
      <name>jspFile</name>
      <required>true</required>
      <rteprvalue>true</rteprvalue>
    </attribute>
  </tag>
```

...

Outils JSP et JSTL

- IDE
 - Sun Java Studio Creator, IBM WebSphere, Borland Jbuilder, Oracle Jdeveloper, Plusssss for Eclipse...
- Déboggeurs
 - TODO
- Tag Library Documentation Generator
 - Générateur de Doc (JavaDoc) de JSTL
 - <https://taglibrarydoc.dev.java.net/>

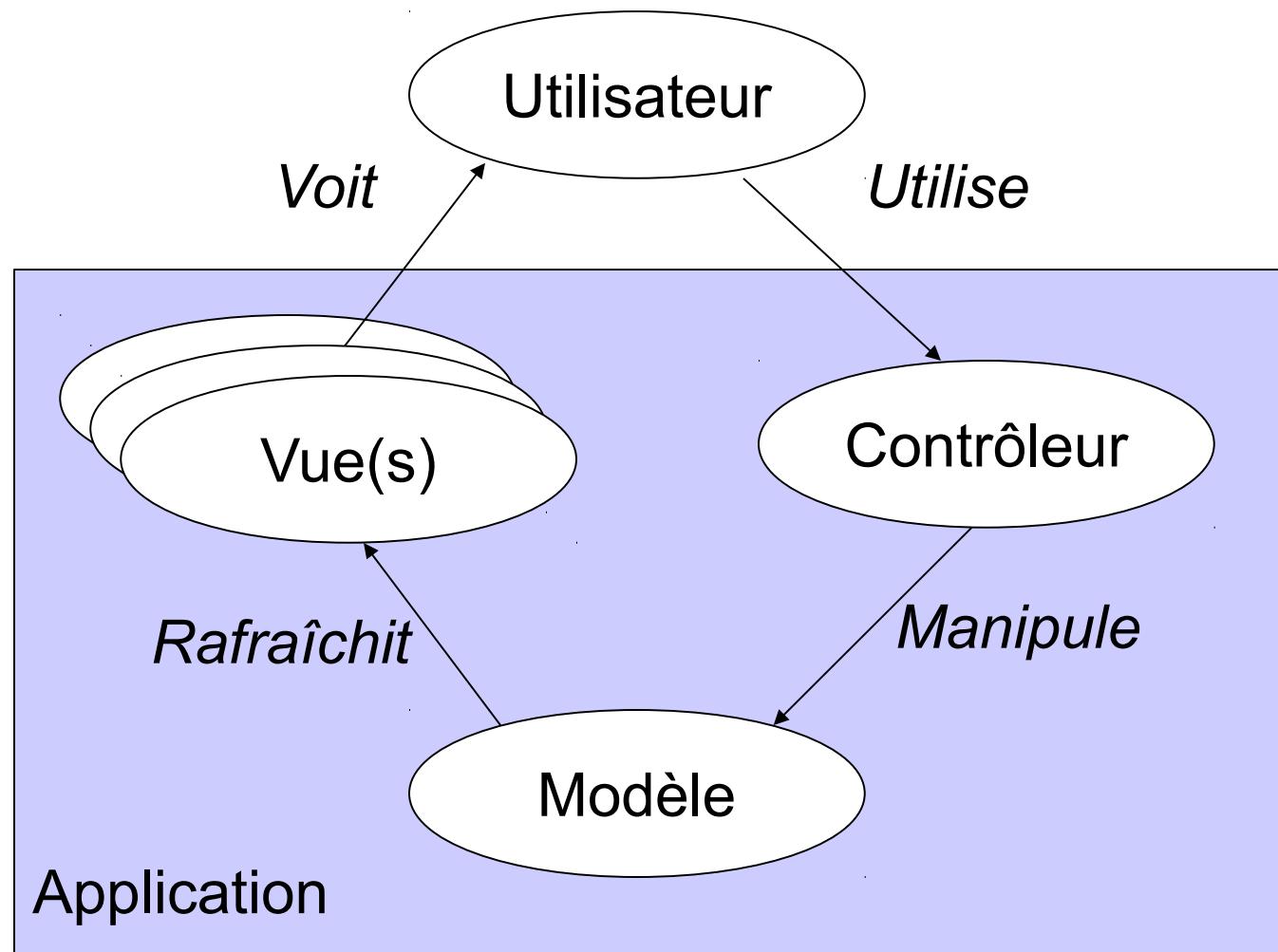
MVC et Web Application

Le Modèle MVC

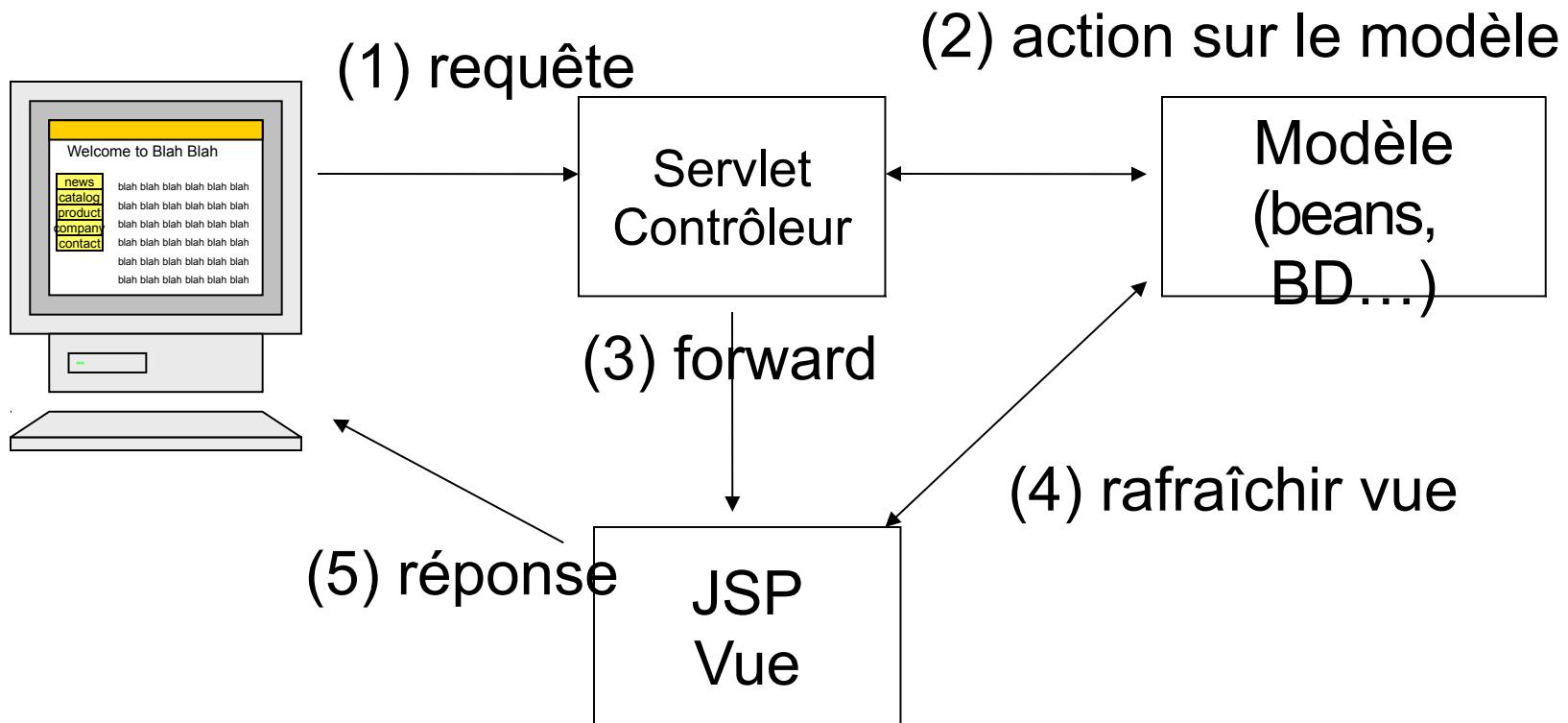
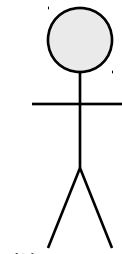
- Model
 - L'application métier, les données... ce qui est manipulé par l'application.
- View
 - présentation de l'état du modèle (généralement à un utilisateur)
- Controller
 - interface d'action sur le modèle
 - (par exemple, pour les mises à jour)
- L'interface utilisateur, composée de vues, intéragit avec le contrôleur.

Le Pattern MVC

Une « boucle de rétroaction »...



Servlets/JSP « Model2 »



Exemple : Servlet Contrôleur

```
public class ControllerServlet extends HttpServlet {  
  
    public void doGet (HttpServletRequest req,  
                      HttpServletResponse res)  
        throws ServletException,IOException {  
  
        String action = req.getParameter("action");  
        // traitement en fonction des actions et de l'état de la session  
        if("consulter".equals(action)) {  
            String nom = req.getParameter("nom");  
            if(nom != null) {  
                String tel = annuaire.chercherContact(nom); // consulte le modèle  
                req.setAttribute("tel", tel); // passe les paramètres à la vue  
                forward("contact.jsp", req, res); // affiche la vue  
            } else  
                forward("error.jsp", req, res); // affiche la vue  
        } else if ("modifier".equals(action)) {  
            // ...  
        } else if ("login".equals(action)) {  
            // ...  
        }  
    }  
}
```

JSR223 Web Scripting Engines

- Motivation
 - Écriture / Récupération de scripts Web dans les langages JavaScript, PHP, Groovy, ...
- API
 - javax.script.servlet.HttpScriptServlet
 - javax.script.servlet.HttpScriptContext