FORMATION JAVA FRAMWORK

Introduction à Java/JEE

Développement Web en J2EE (Servlet & JSP)

- Rappel
- J2EE et Web Application
- Servlets et Filters
- JSP et JSTL

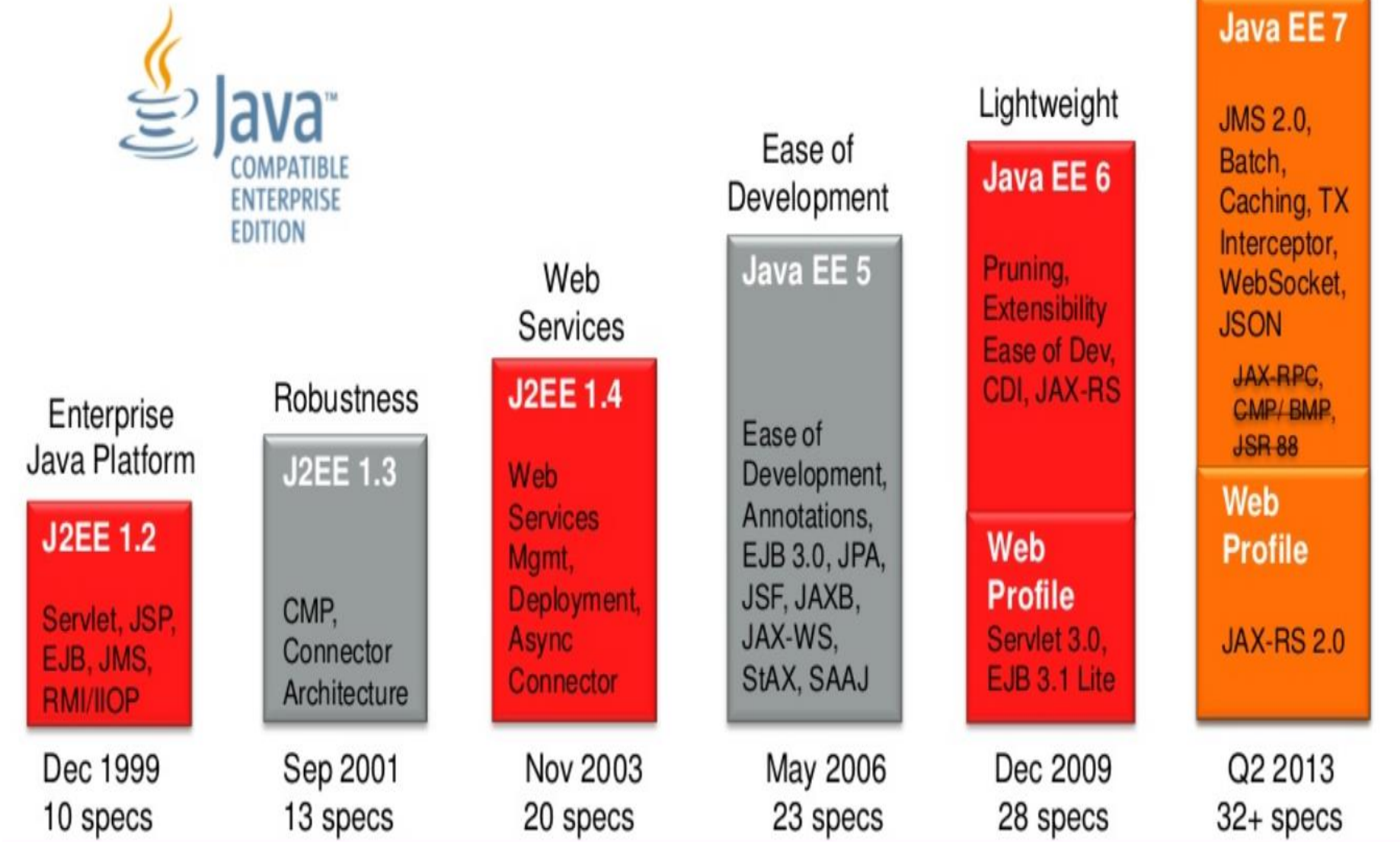
Introduction

- Internet est vu comme média transactionnel , mais pas forcément adapté à ce type d'usage.
- Les serveurs d'applications sont nés pour s'interfacer entre le client Internet (client léger) et le système d'information de l'entreprise.
    - Un serveur d'application a deux rôles principaux
        - contenir la logique métier (business) de l'application internent
        - mettre en forme les données envoyées vers le client

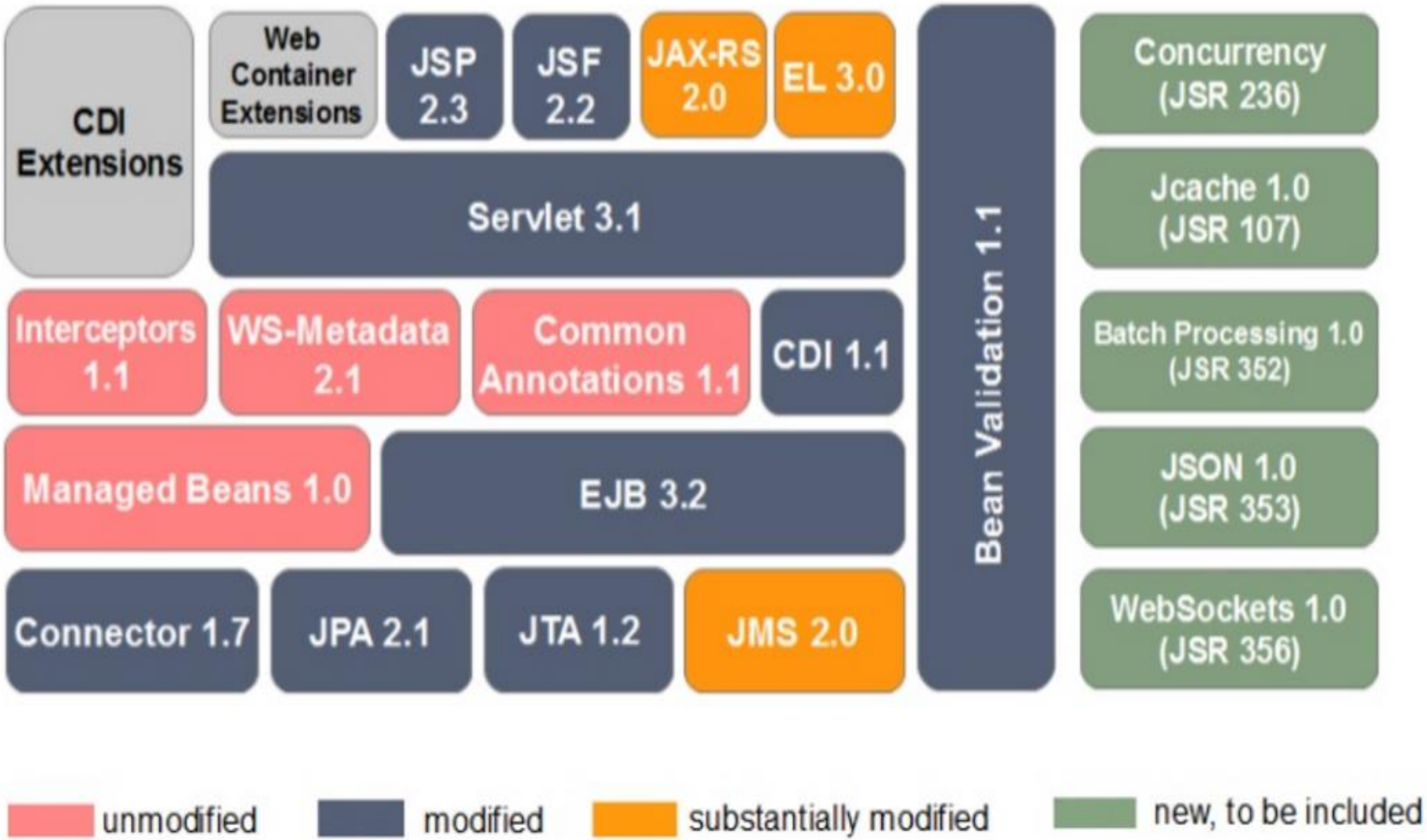
Java Entreprise Edition : J2EE

- La plate-forme J2EE (basée sur java) offre une solution complète de développement d'applications Internet.
- J2EE reprend le principe d'application 3-tiers.
- J2EE utilise les API java:
    - EJB
    - Servlet/jsp
    - JTS
    - JTA
    - JMS
    - JavaMail et d’autres …

Java EE Past, Present, & Future



Java EE 7 – Candidate JSRs

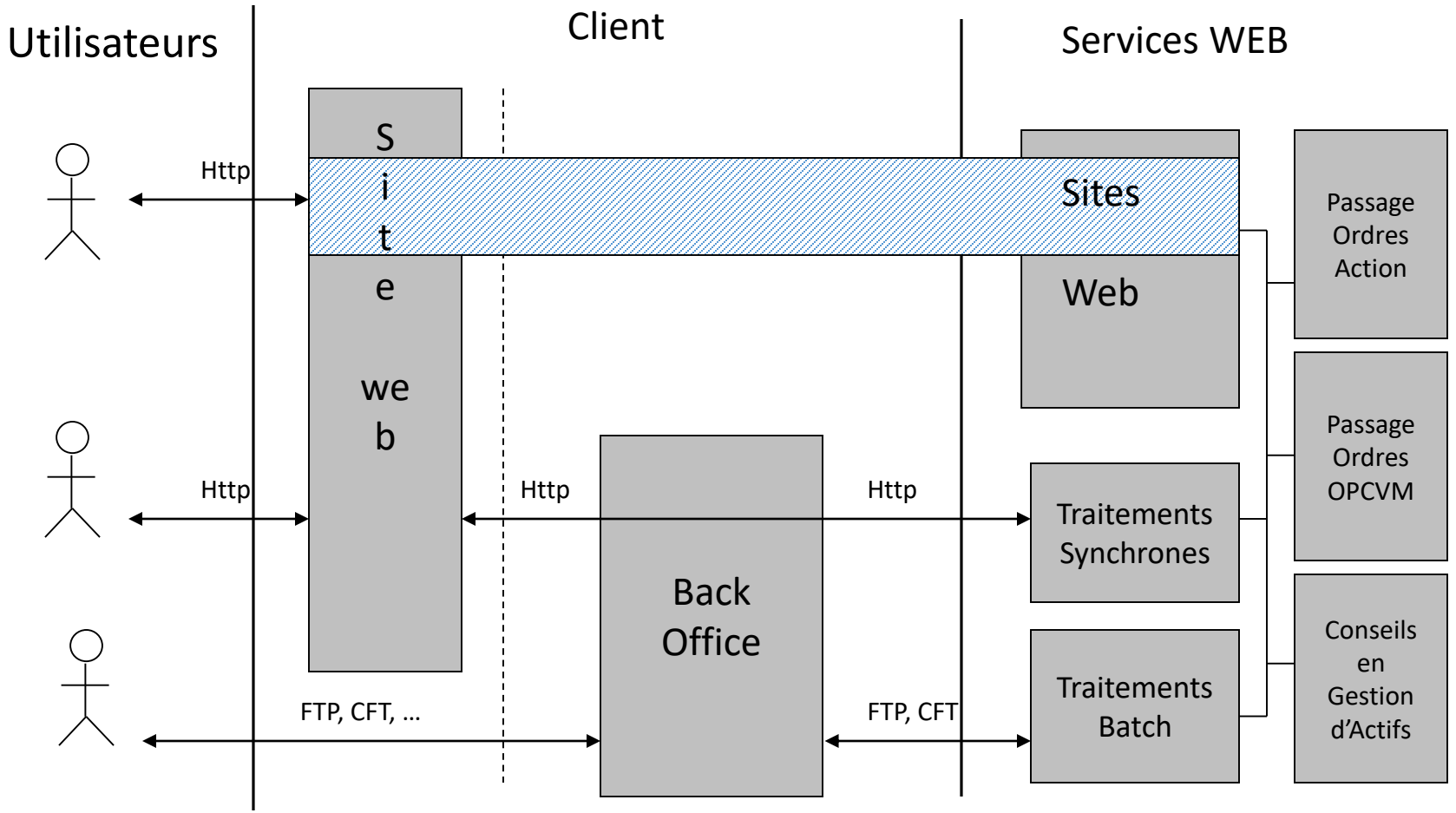


Répartition Client-Serveur

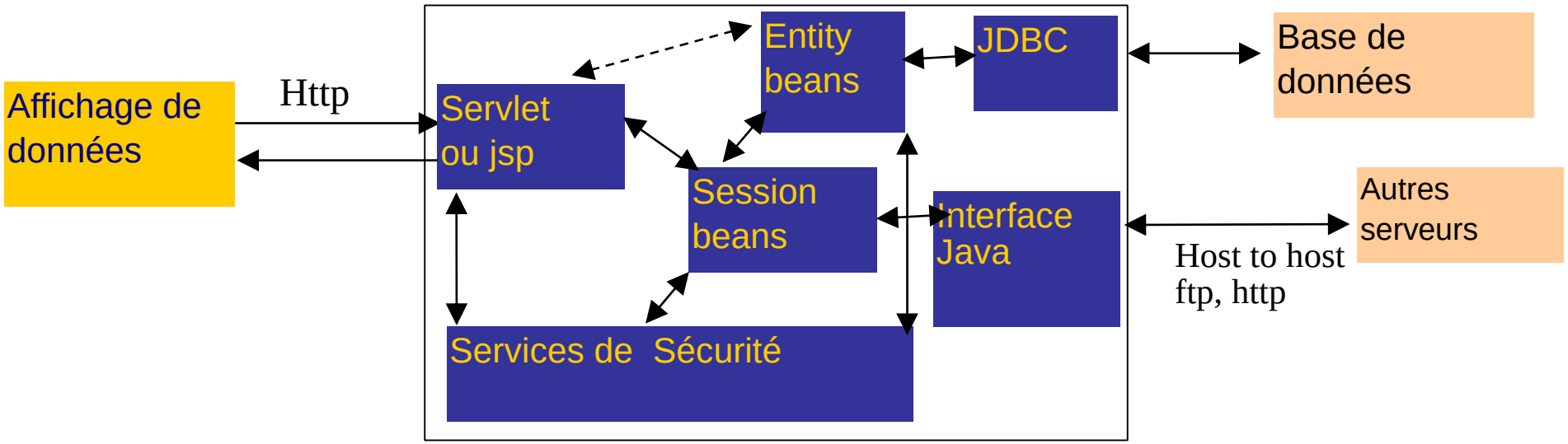
- Traitement coté client
  - extériorisation de l ’information
  - formats visualisables par le client
  - interactivité non lié au réseau
- Traitement coté serveur
  - serveur applicatif
  - conserve les traitements en interne (business logic)
  - adapte le document retourné au capacité du visualisateur
  - charge le serveur

Exemple pratique:

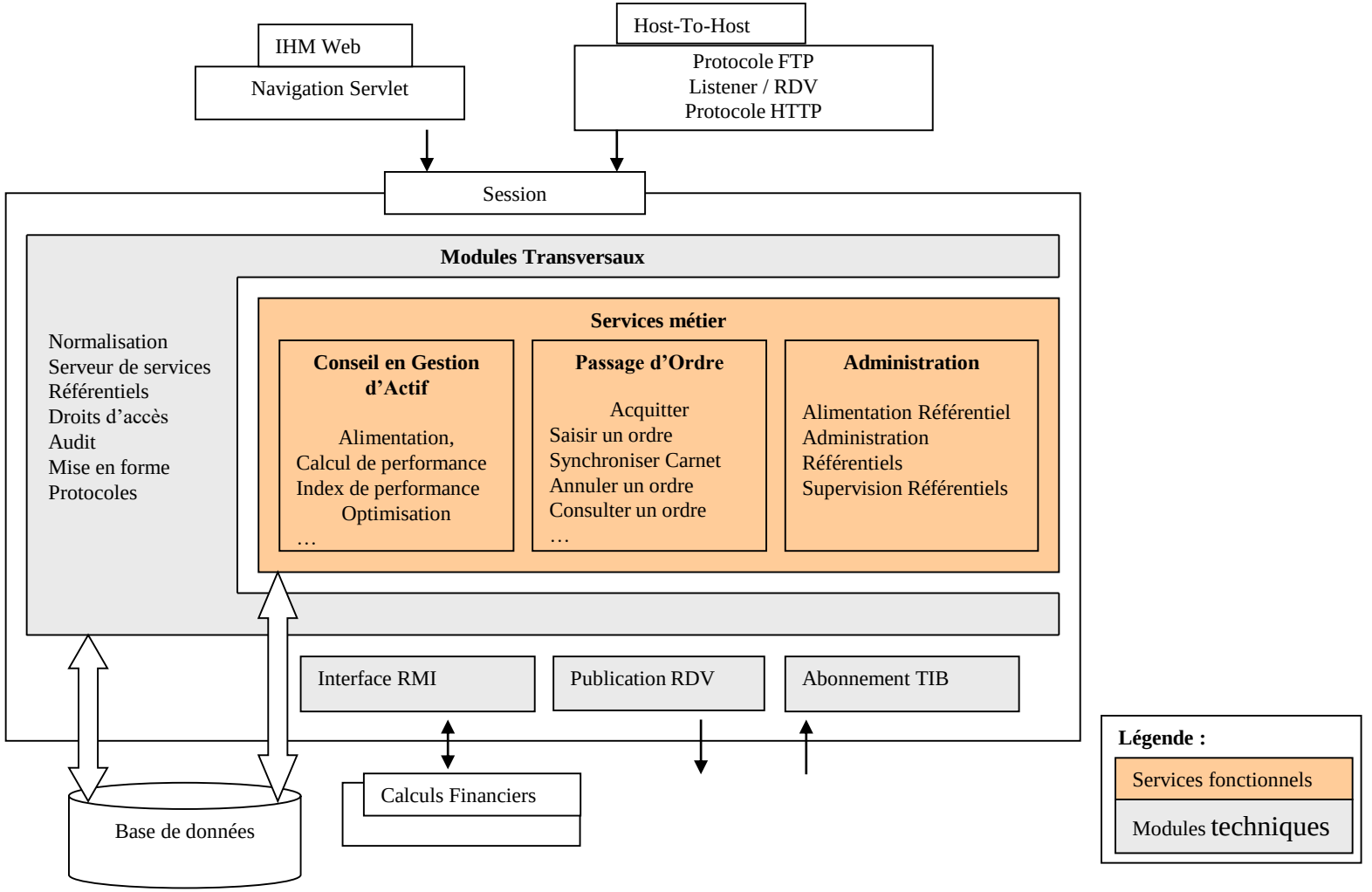
Présentation générale



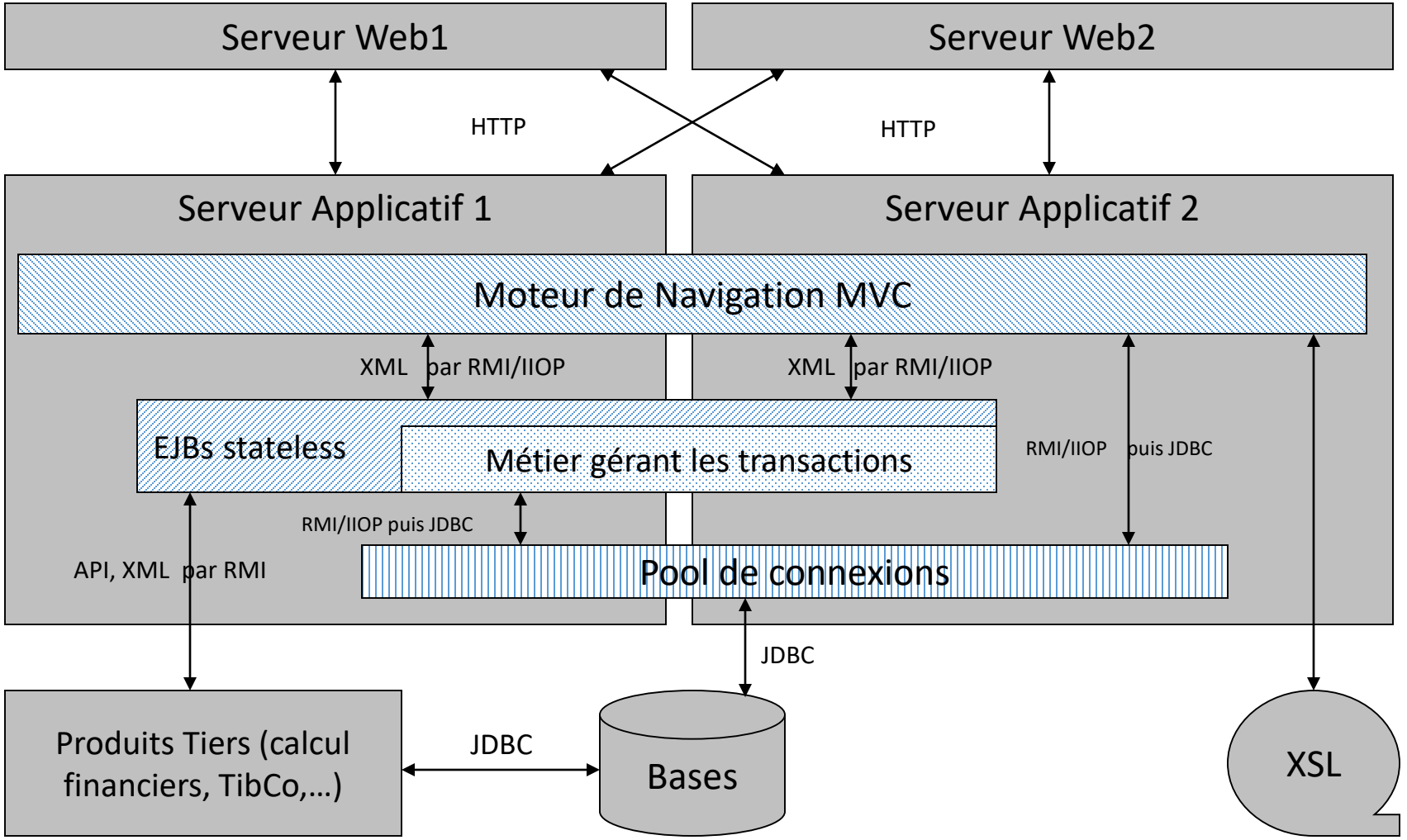
Architecture J2EE - suite



Architecture Logicielle



Architecture : plus en détail



Web application

- Partie « présentation » d'une application J2EE
  - 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 J2EE

- Web Component
  - **Une application Web (*.html, *.jsp, servlets, …) packagée dans un .jar (.war) et est paramêtrée dans le fichier WEB-INF/web.xml**
  - **L'application est installée dans le répertoire webapps du serveur web J2EE**
- Structure d'une Web Application Archive (.war)
  - ***.html, *.png, *.jsp, …, applets.jar, midlets.jar**
  - **WEB-INF/web.xml**
    - **Fichier de déploiement,Paramêtrage 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/**

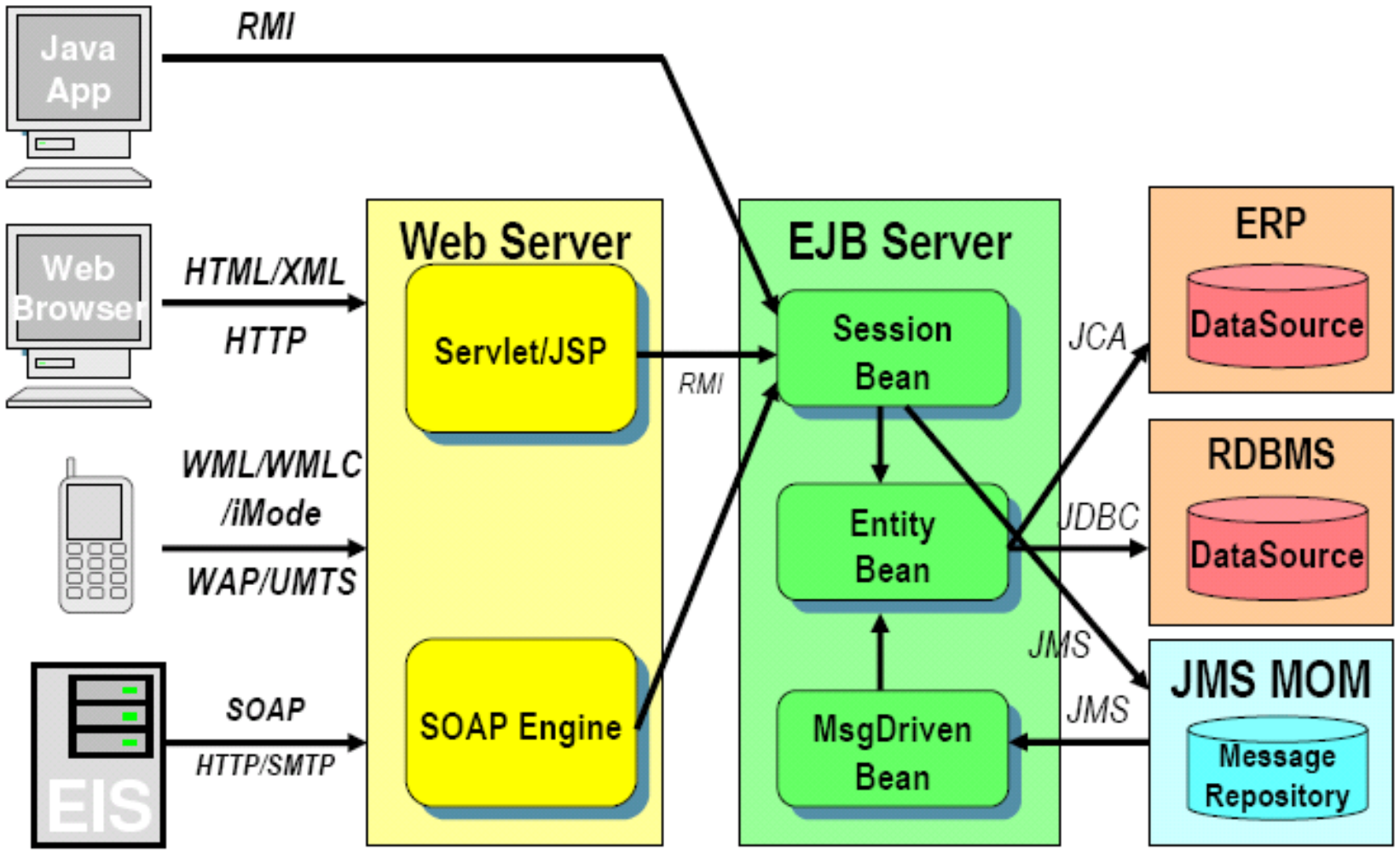
Le fichier WEB-INF/web.xml

- 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
    - Realms..

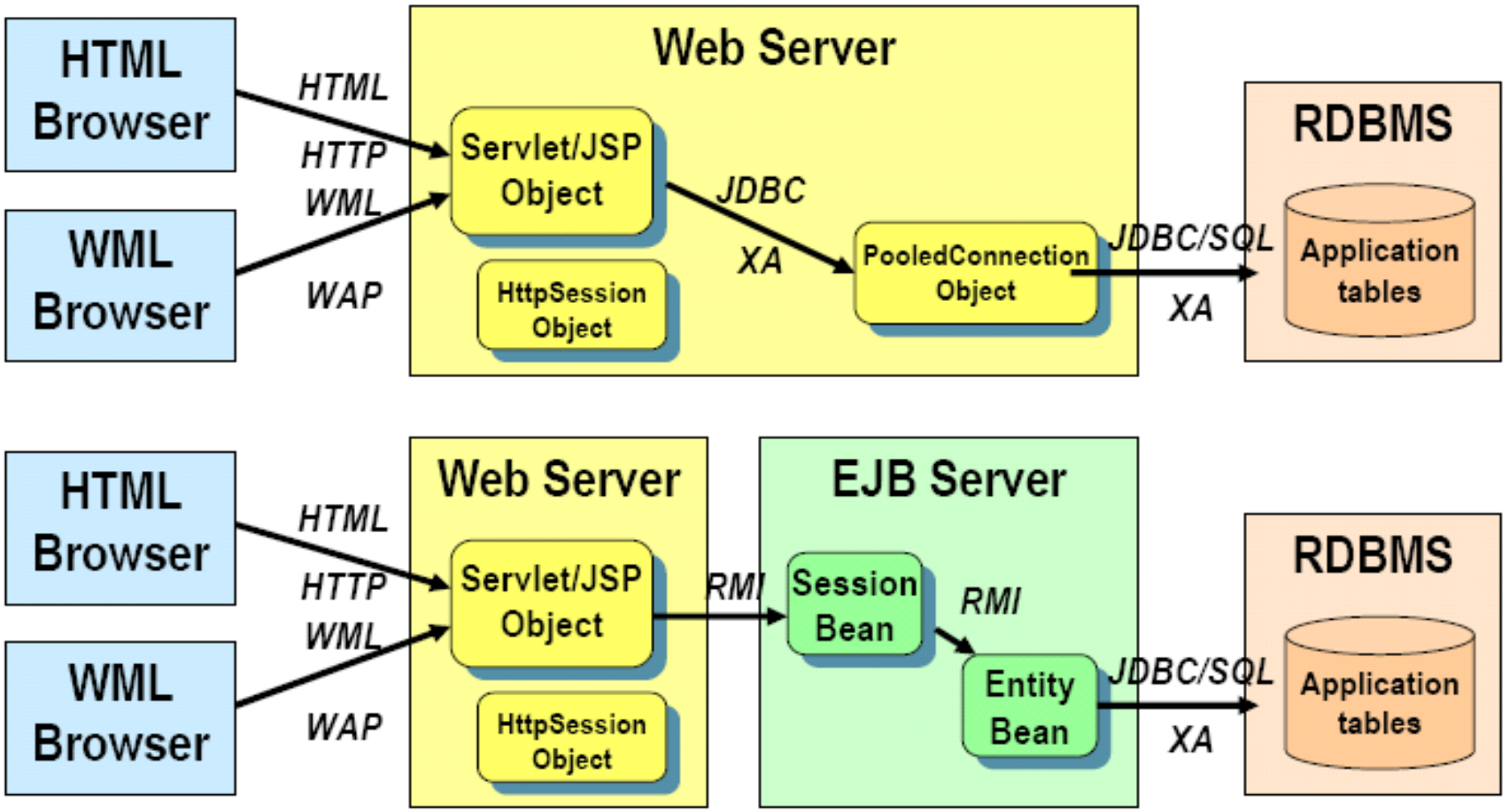
Packaging d’une application J2EE complète

- Une application complète est packagée dans un fichier ear constitué de plusieurs war et jar, ainsi que d'un descripteur application.xml

Architecture d'un service J2EE (1)



Architecture d'un service J2EE (2)

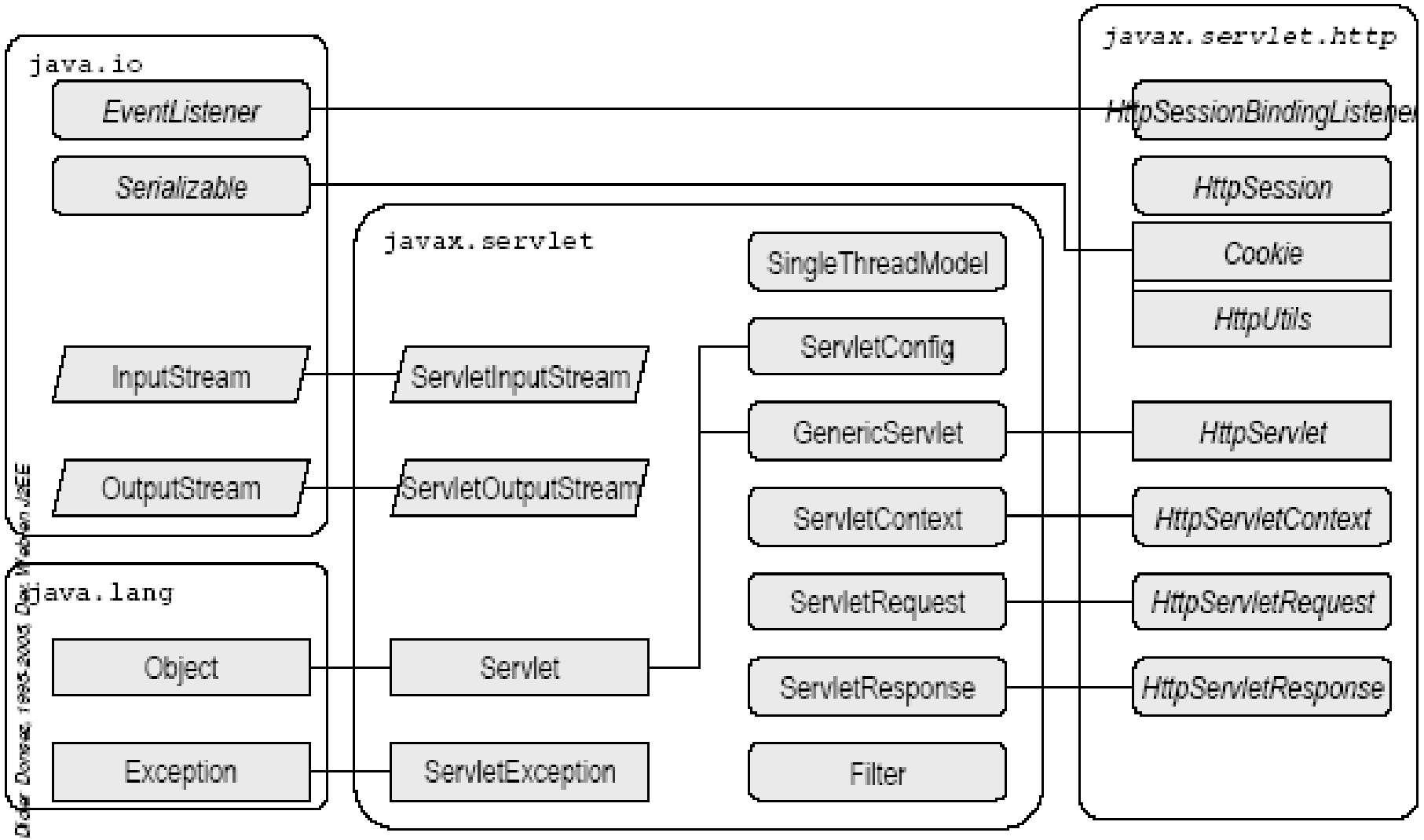


Les Servlets

- Servlets
  - Scripts serveur écrit en Java
    - Servlets de Base : FileServlet, CGIServlet, …
    - HttpServlet
  - Exécution dans un espace isolé (Web Application)
  - Spécification : Sun (sous partie de J2EE)
  - Implémentation de référence : Apache Group (Jakarta Tomcat)
  - Différence avec les CGI et les LD (NSAPI, ISAPI)
    - performance sur les passages des paramêtres (vs CGI)
    - sûreté de fonctionnement (NSAPI, ISAPI)

L'API Servlet

- javax.servlet et javax.servlet.http



Exemple de Servlet

```
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 {
    res.setContentType("text/html"); // Set the Content-Type header
    PrintWriter out = res.getWriter(); // Get the output
    String pname = req.getParameter("name"); // Get a parameter
    if(pname==null) pname="World !";
    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); }
}
```

Cycle de Vie d'une servlet

- Chargement
  - Le mode de chargement est spécifié dans le descripteur de déploiement
- Initialisation
  - méthode init()
- Traitement de 1 à N requêtes
  - méthode service()
- Destruction
  - méthode destroy()

L’initialisation de la servlet

Définition des paramêtres d ’initialisation dans web.xml

```
<servlet>
  <servlet-name>MyServlet</servlet-name>
  <servlet-class>myapp.servlets.MyServlet</servlet-class>
  <init-param><param-name>language</param-name><param-value>en</param-value>
  </init-param>
  <init-param><param-name>currency</param-name><param-value>EUR</param-value>
  </init-param>
</servlet>
```

Code pour récupérer ces paramètres

```
public class MyServlet extends GenericServlet {
  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    out.println("Servlet init parameters:");
    Enumeration e = getInitParameterNames();
    while (e.hasMoreElements()) {
      String key = (String)e.nextElement(); String value = getInitParameter(key);
      out.println("   " + key + " = " + value);
  }.. }
```

La Requête

- L ’interface ServletRequest
  - `String getCharacterEncoding()`
  - `int getContentLength()`
  - `String getProtocol()`
  - `String getContentType()`
  - `String getRealPath(String path)`
  - `Enumeration getAttributeNames() Ex: "javax.servlet.request.X509Certificate"`
  - `Object getAttribute(java.lang.String name)`
  - `void setAttribute(String key, Object o)`
  - `String getParameter(String name)`
  - `Enumeration getParameterNames()`
  - `String[] getParameterValues(String name)`
  - `Map getParameterMap()`
  - `ServletInputStream getInputStream()`

La Requête

- L ’interface HttpServletRequest
  - `String getAuthType()`
  - `java.lang.String getHeader(java.lang.String name)`
  - `java.util.Enumeration getHeaderNames()`
  - `int getIntHeader(java.lang.String name)`
  - `Cookie[] getCookies()`
  - `HttpSession getSession(boolean create)`
  - `boolean isRequestedSessionIdFromCookie()`
  - `boolean isRequestedSessionIdFromURL()`
  - `boolean isRequestedSessionIdValid()`

La Requête

- La récupération de l'URI

http://localhost:8080/examples/servlet/SnoopServlet/tutu?toto=tata

```
String getMethod() GET
String getPathInfo() /tutu
String getQueryString() toto=tata
String getRequestURI()
/examples/servlet/SnoopServlet/tutu
String getServletPath() /servlet/SnoopServlet
String getContextPath() /examples
```

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 recupération 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 recupération 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 Réponse

- L ’interface ServletResponse
  - Récupération du flot de sortie (c.a.d-vers le client)

```
ServletOutputStream getOutputStream();
PrintWriter getWriter()
```

  - Longueur et type du corps

```
void setContentLength(int len)
void setContentType(String type)
```

  - i18n

```
void setLocale(Locale loc)
void setCharacterEncoding(String charset)
```

  - Gestion du buffer du flot de sortie

```
void reset()
boolean isCommitted() // status and headers are written
void setBufferSize(int size)
void resetBuffer()
void flushBuffer()
```

La Réponse

- L ’interface HttpServletResponse
  - Ajout de champs d’entête et de cookies

```
void addCookie(Cookie cookie)
void setHeader(String name, String value)
void setIntHeader(String name, int value)
```

  - Retour du status

```
static final int SC_OK = 200
static final int SC_BAD_REQUEST = 400
static final int SC_NOT_FOUND = 404 …
void setStatus(int sc)
void setStatus(int sc, String sm)
void sendError(int sc, String msg)
void sendError(int sc)
void sendRedirect(String location)
```

La Réponse

- L ’interface HttpServletResponse
  - Réécriture d’URL (session tracking)
    - String encodeURL (String url)
    - String encodeRedirectURL (String url)
    - String encodeUrl(String url)

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

- L ’ajout d ’entête

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

La Réponse

- Le forwarding, à utiliser pour les redirections internes, par exemple d'une servlet vers une JSP

```
protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
    dispatcher.forward(request, response);
}
```

- La redirection, à utiliser pour les redirections externes

```
protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    String urlWithSessionID = response.encodeRedirectURL("index.jsp");
    response.sendRedirect( urlWithSessionID );
}
```

- L'inclusion, à utiliser pour inclure du contenu généré par une autre ressource du

```
protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    response.getWriter().print("bonjour de " + this.getClass().getName());
    RequestDispatcher dispatcher = request.getRequestDispatcher("footer.jsp");
    dispatcher.include(request, response);
}
```

La Session

javax.servlet.http.HttpSession

- Le suivi de session
  - Le serveur maintient une session de 2 manières :
    - **Cookie (Name: SESSIONID Value:To1010mC86010218357411 67At)**
      - **(si les cookies peuvent être déactivés sur le navigateur)**
    - **Réécriture des URLs**
      - **(dans le cas contraire)**
- 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: To1010mC860102183574l167At
  - 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();

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

```
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");
 }
 ...
```

Accès à une Source de Données JDBC

WEB-INF/web.xml

```
<resource-ref>
<description>Ma Base de Données</description>
<res-ref-name>jdbc/EmployeeDB</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

Code

```
Context initContext = new InitialContext();
Context envContext = (Context)initContext.lookup("java:/comp/env");
DataSource ds = (DataSource)envContext.lookup("jdbc/EmployeeDB");
Connection conn = ds.getConnection();
...
comm.close();
```

Accès à une Source de Données JDBC

Configuration de la fabrique de ressources JDBC dans TomCat

- $CATALINA_HOME/conf/server.xml

```
<Context ...>
  ...
  <Resource name="jdbc/EmployeeDB" auth="Container" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/EmployeeDB">
    <parameter><name>username</name><value>dbadmin</value></parameter>
    <parameter><name>password</name><value>toto</value></parameter>
    <parameter><name>driverClassName</name><value>org.hsql.jdbcDriver</value>
    </parameter>
    <parameter><name>url</name><value>jdbc:HypersonicSQL:database</value>
    </parameter>
    ...
  </ResourceParams>
  ...
</Context>
```

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
  - J2EESDK et Jakarta JASPER/Tomcat

Insertion des scripts

- Directives

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

- Éléments de script

```
Scriplets <% 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 (1)

```
<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>
```

Scriptlet (source Java)

Expression (EL)

Exemple de traitement d ’un formulaire (2)

```
<%@ 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>
```

JSP : Exemple avec une session

- 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 addItem(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) addItem(item);
       if (submit.equals("add")) addItem(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>)

Design Pattern

- Technique d'architecture logiciels --> model, patron de conception.
- Description d'une solution technique à un problème récurrent.
- Avantages
    - Un vocabulaire commun.
    - capitalisation de l'expérience, et réduction de la complexité.
    - Construction logicielle de meilleure qualité.
- Inconvénients
    - Effort de Synthèse
    - Nombreux patterns (imbriquation)

Modèle Vue Contrôleur