

JEE

Java Enterprise Edition

J. Saraydaryan

CPE - Lyon



I Introduction

II JEE Web Container

III JEE EJB Container



JEE Web Container

J. Saraydaryan

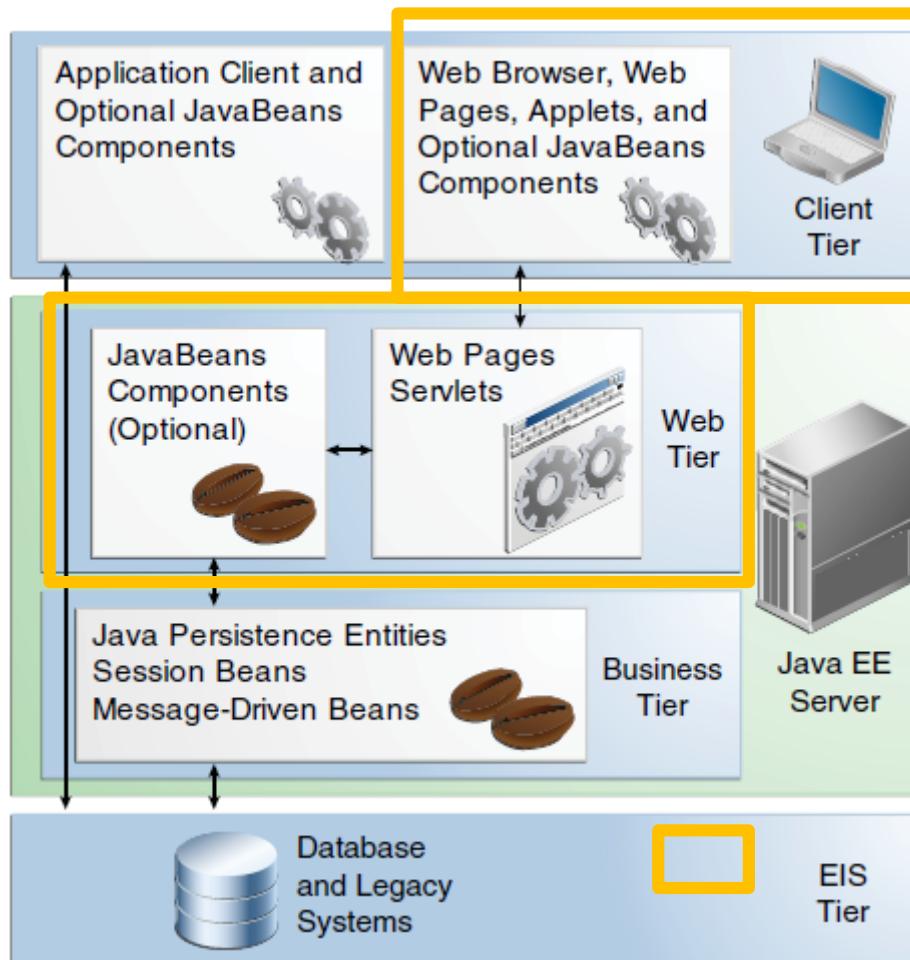
CPE - Lyon



Introduction

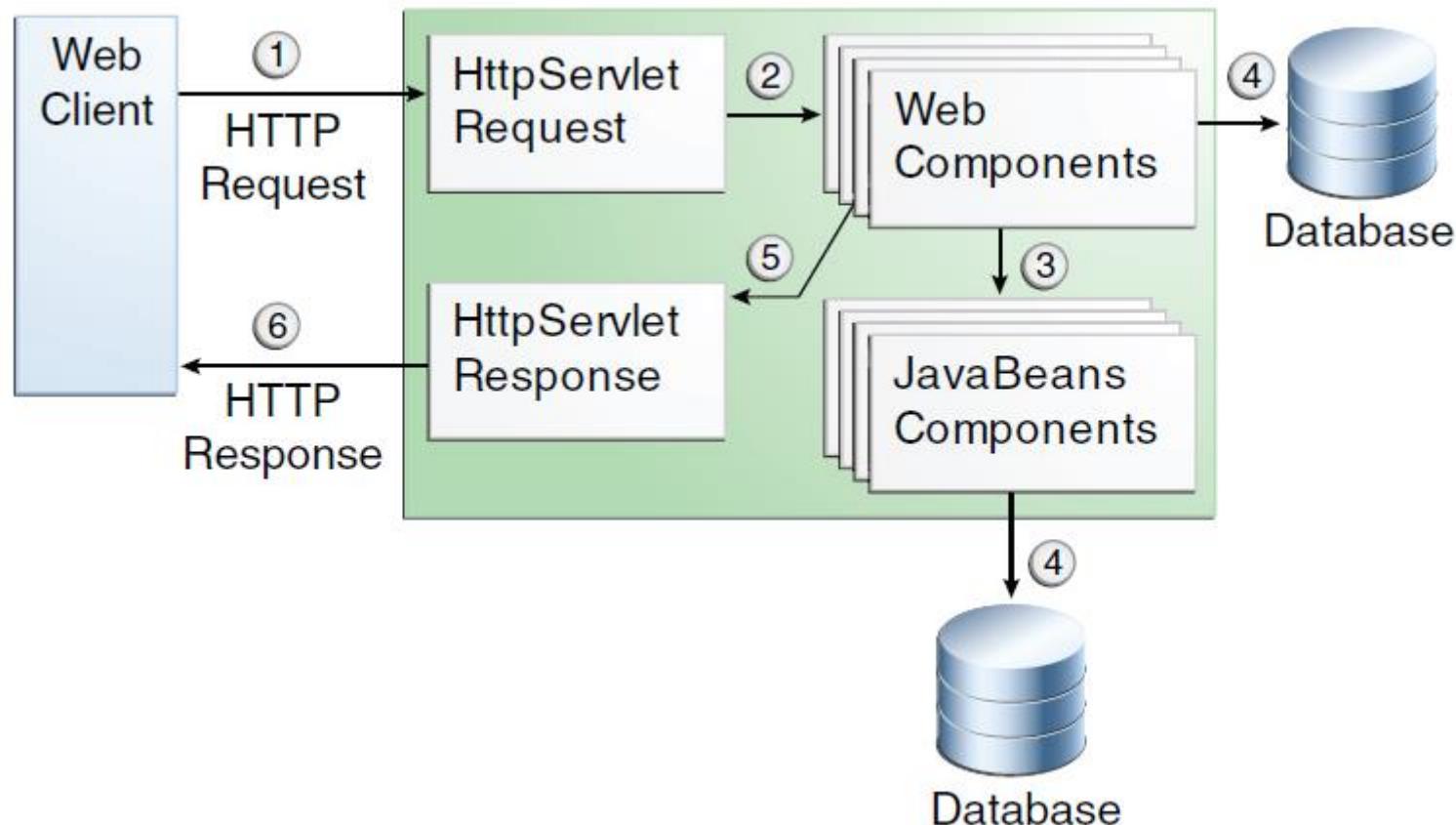
Java Entreprise Edition

- Rappel



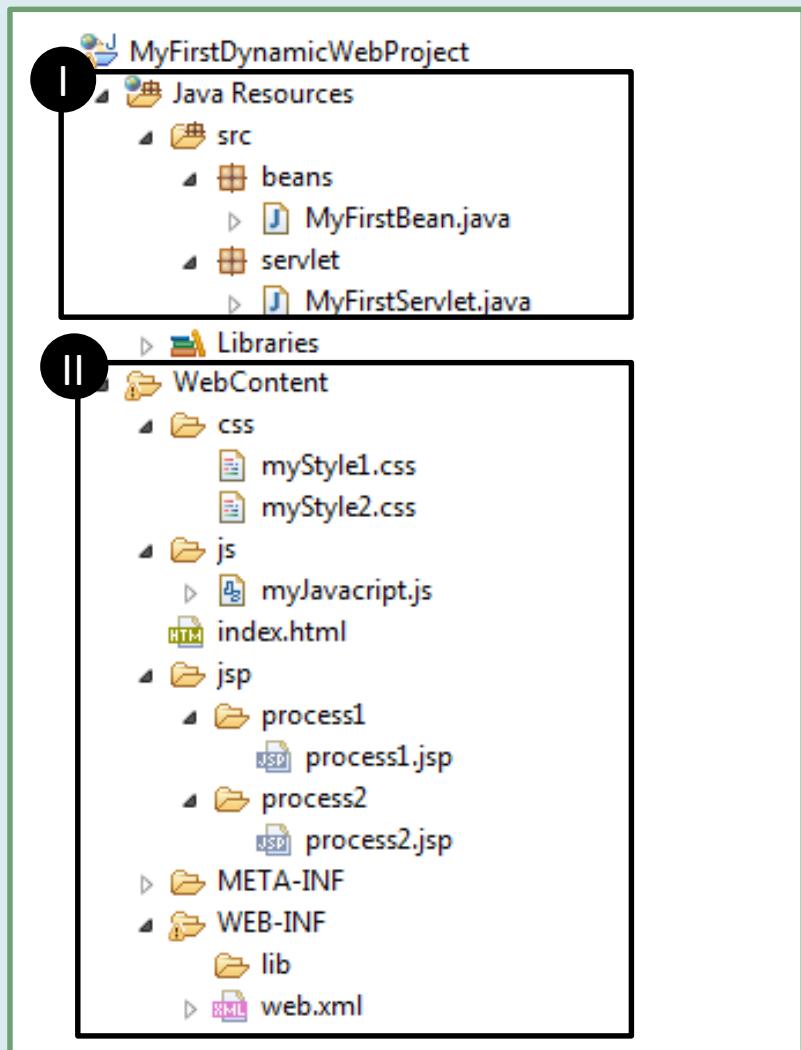
Java Entreprise Edition

- Rappel



Java Entreprise Edition

• Anatomie d'un projet Web Dynamique JEE



I Objet java pur:

- ❑ Servlet (Contrôleur de l'application)
- ❑ Java Bean (gestion du Modèle)

II Conteneur des informations Web

❑ Information Web Classique

- Css
- Js
- Html,xhtml

❑ Information Web Compilée

- JSP (jsf vu plus tard)

❑ Métadonnées et données complémentaires

- WEB-INF/lib : jar complémentaires
- WEB-INF/lib: web.xml fichier de configuration de notre application WEB

Java Entreprise Edition

- A vous de Jouer !

- Créez votre premier projet web dynamique sous éclipse
- Créez une page index.html lancée automatique au démarrage



Java Servlet

Java Entreprise Edition

❑ Objectif

- ❑ Développer des applications Web
- ❑ Réagir aux requêtes HTTP

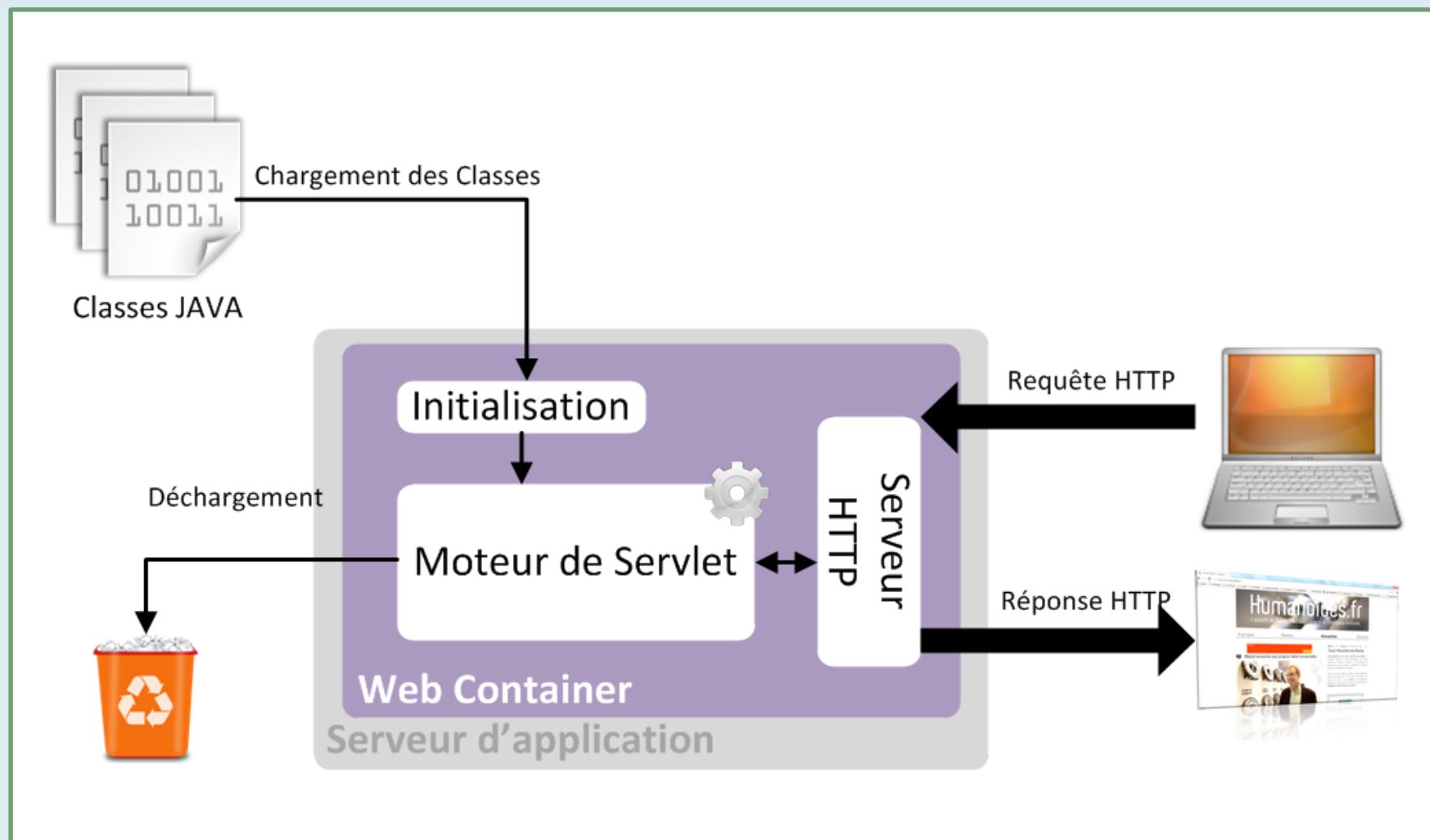
❑ Propriétés

- ❑ Utilisation du langage Java
- ❑ Retourne des flux de type texte, HTML, XML,..
- ❑ Peut effectuer des traitements complexes
 - Utilisation, Manipulation XML
 - Utilisation, Manipulation Base de données
 - Lancement de traitements multithread, batch



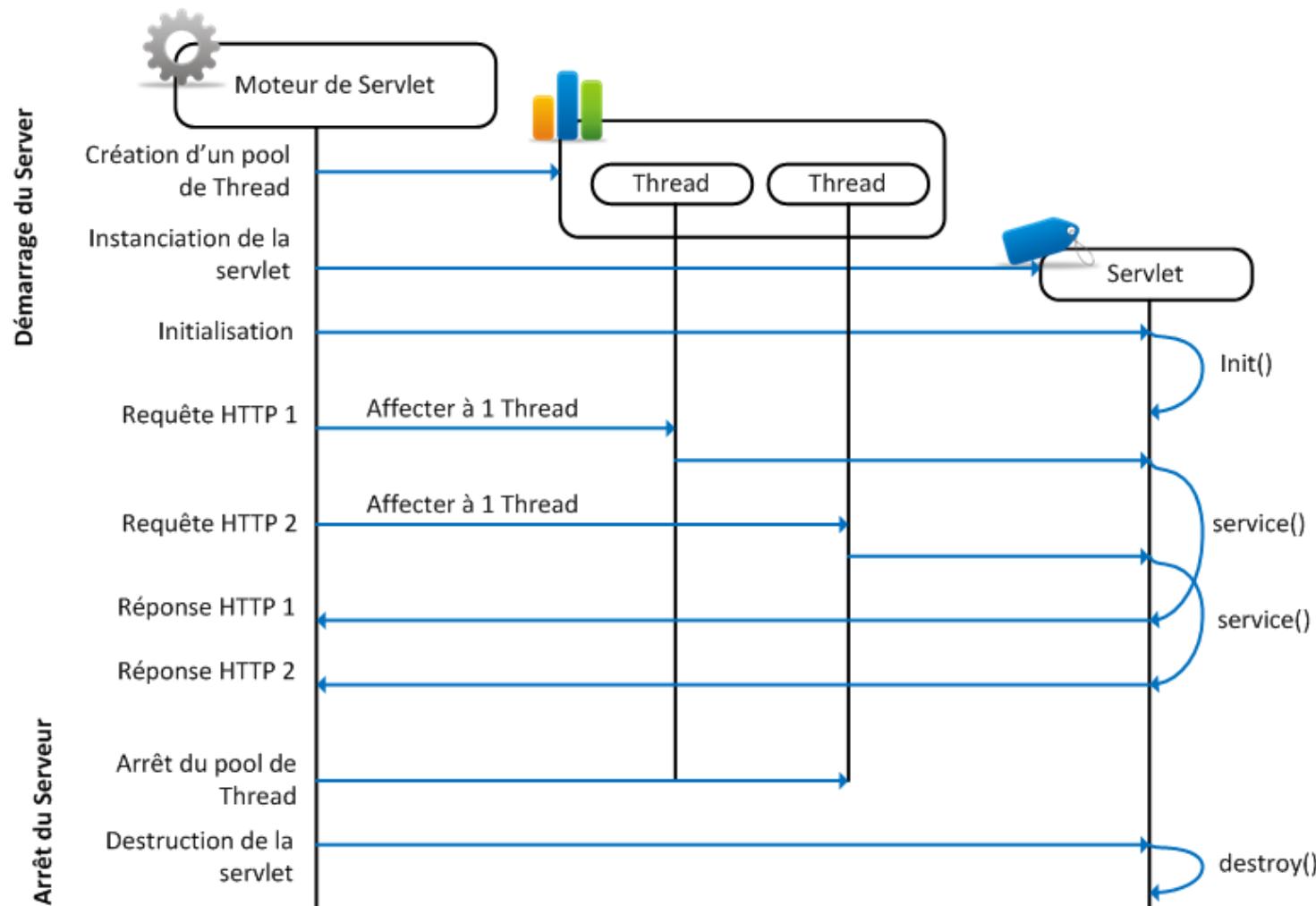
Java Entreprise Edition

• Cycle de vie 1/2



Java Entreprise Edition

• Cycle de vie 2/2



Java Entreprise Edition

• Mise en Œuvre

- ❑ Package javax.servlet, javax.servlet.http
- ❑ Héritage de javax.servlet.http.HttpServlet Pour créer une servlet

❑ API

- ***init(ServletConfig)*** → initialisation nécessaire de la servlet
- ***doDelete(HttpServletRequest, HttpServletResponse)***
- ***doPost(HttpServletRequest, HttpServletResponse)***
- ***doGet(HttpServletRequest, HttpServletResponse)***
- ***doOptions(HttpServletRequest, HttpServletResponse)***
- ***destroy()*** → appelée lorsque le serveur décide de supprimer la servlet
 - Fermeture des connexions base de données
 - Arrêts des Threads démarrés
 - log des information d'activité ...



Java Entreprise Edition

• Mise en Œuvre

```
@WebServlet("/MyFirstServlet")
public class MyFirstServlet extends HttpServlet {
    public MyFirstServlet() {
        super();
    }
    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        //use configuration Parameters
        String dbName=(String)config.getInitParameter("DbName");
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // Process on Http Get
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // Process on Http Post
    }

    @Override
    public void destroy() {
        super.destroy();
    }
}
```

Java Entreprise Edition

- **Mise en Œuvre**

- **HTTP Request**

- Utiliser pour récupérer les informations provenant du web browser

- **API**

- String getParameter(String s)
 - Map<String, String[]> getParameterMap()
 - Enum<String> getParameterNames()

```
http://monServeur/myURL?param1=val1&param2=val2
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)  
{  
    // Process on Http Get  
    String valueParam1=request.getParameter("param1");  
    Map<String, String[]> paramMap = request.getParameterMap();  
}
```

Java Entreprise Edition

- Mise en Œuvre

- HTTP Response

- Utiliser pour gérer la réponse au web browser

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
{
    // Process on Http Post

    response.setContentType("text/html");      Définition du type de retour

    response.getWriter();                     Flux d'écriture en sortie caractère

    response.getOutputStream();                Flux d'écriture en sortie binaire

    response.sendRedirect("/redirectUrl");     Redirection vers une autre URL

}
```

Java Entreprise Edition

- Exemple 1/2

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
@WebServlet("/MyFirstServlet")
public class MyFirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private String initParam1;
```

```
    public MyFirstServlet() {super(
```

```
        @Override
```

```
        public void init(ServletConfig config) throws ServletException {
            super.init(config);
            is.initParam1=(String)config.getInitParameter("initParam1");
```

Initialisation

Héritage
HTTPServlet

Package API

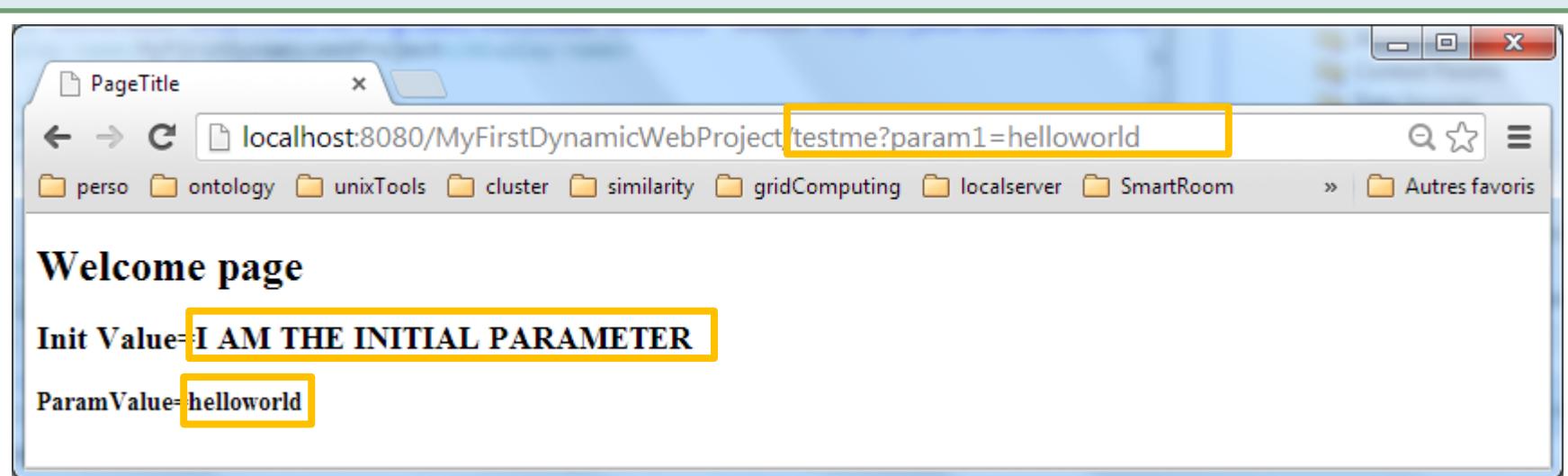
Java Entreprise Edition

- Exemple 2/2

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // Process on Http Get
    String valueParam1=request.getParameter("param1");
    if("redirect".equals(valueParam1)){
        response.sendRedirect("/redirectUrl");
    }else{
        PrintWriter wr = response.getWriter();
        wr.println("<!DOCTYPE html>");
        wr.println("<html>");
        wr.println("<head>");
        wr.println("<title>PageTitle</title>");
        wr.println("</head>");
        wr.println("<body>");
        wr.println("<h1> Welcome page </h1>");
        wr.println("<h2> Init Value="+this.initParam1+"</h2>");
        wr.println("<h3> ParamValue="+valueParam1+"</h3>");
        wr.println("</body></html>");
    }
}
```

Java Entreprise Edition

- Exemple: résultat



```
<servlet>
<servlet-name>MyFirstServlet</servlet-name>
<servlet-class>servlet.MyFirstServlet</servlet-class>
<init-param>
<param-name>initParam1</param-name>
<param-value>I AM THE INITIAL PARAMETER</param-value>
</init-param>
</servlet>

<servlet-mapping>
  <servlet-name>MyFirstServlet</servlet-name>
  <url-pattern>/testme</url-pattern>
</servlet-mapping>
```

WEB.XML

Java Entreprise Edition

- A vous de Jouer !

- Créer une servlet permettant
 - Afficher l'ensemble des paramètres (clés, valeurs) d'une requête
 - Afficher des informations contextuelles
 - Remote Address IP
 - Local Port
 - Protocol
 - Current URI



Java Entreprise Edition

- A vous de Jouer !

The screenshot shows a web browser window titled "pageTitle". The address bar displays "localhost:8080/MyFirstDynamicWebProject/pocme?param1=valueA¶m2=valueB¶m3=valueC". The page content is divided into two sections: "Info Param" and "Info Web browser".

Info Param

- param1 valueA
- param2 valueB
- param3 valueC

Info Web browser

- Remote Add: 0:0:0:0:0:0:1
- Protocole: HTTP/1.1
- ContextPath: /MyFirstDynamicWebProject
- LocalPort: 8080
- URI/MyFirstDynamicWebProject/pocme

Java Entreprise Edition

- A vous de Jouer !

- Créer un page html auth.html contenant un formulaire de login et s'affichant par default à l'ouverture du site
- Créer une servlet permettant de vérifier le login pwd depuis une liste et d'afficher :
 - « Welcome login +pwd » en cas de success
 - « Mauvaise Authentification » en cas d'échec



Java Entreprise Edition

- A vous de Jouer !

The image displays three separate browser windows, each showing a different outcome of a Java EE authentication attempt:

- Test Authentication**: This window shows a login form with fields for "Login:" and "Password:", and a "Submit" button. The URL in the address bar is `localhost:8080/MyFirstDynamicWebProject/`.
- FAIL**: This window shows the error message Mauvaise authentification (Bad authentication). The URL in the address bar is `localhost:8080/MyFirstDynamicWebProject/`.
- OK**: This window shows a welcome message: **Welcome**, **AgentJ**, and **black**. The URL in the address bar is `localhost:8080/MyFirstDynamicWebProject/`.

JSP: Java Server Page

Java Entreprise Edition

• Blian Servlet

❑ Avantages

- Création de contenu dynamiquement
- Utilisation des paramètres pour des traitements complexes
- Orchestrateur de site (redirection conditionnelle)



❑ Inconvénient

- Langage inapproprié à la création de page WEB
- Gestion des données de stockage et d'affichage complexe

❑ Besoin : nouvel outil adapté à la création de page Web

→ *Java Server Pages*

Java Entreprise Edition

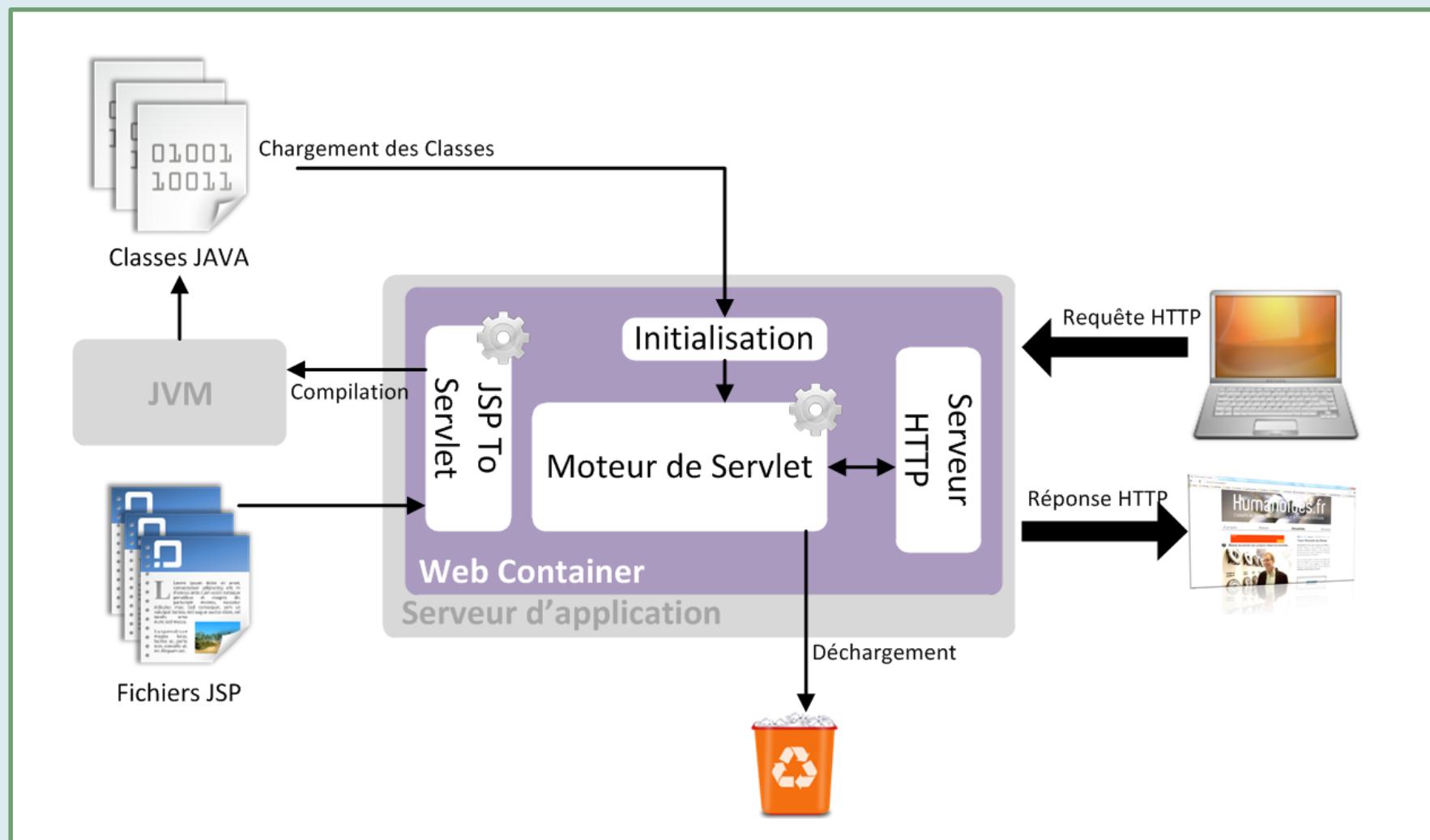
• Java Server Page

- ❑ Les JSP sont basées sur la technologie des Servlets
 - Le cycle de vie d'une JSP est le même que celui d'une Servlet
- ❑ La différence : les fichiers JSP sont compilés sous forme de Servlets
 - Au premier appel
 - Recompilés à chaque modification du code source
- ❑ Le code Java est intégré au code HTML avec des balises d'échappement <% et %>
- ❑ Une page JSP peut être écrite en XML



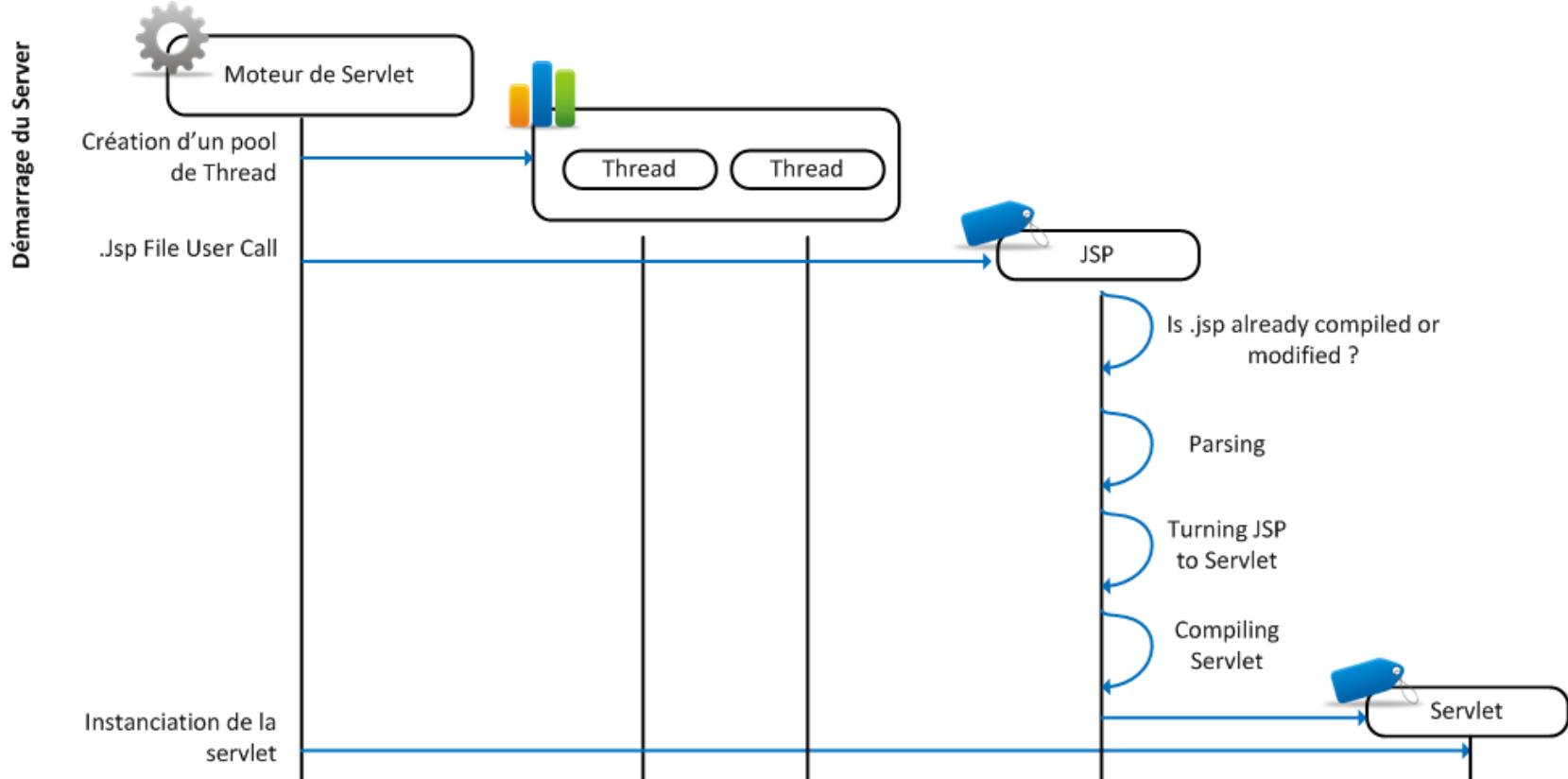
Java Entreprise Edition

• Cycle de vie 1/3



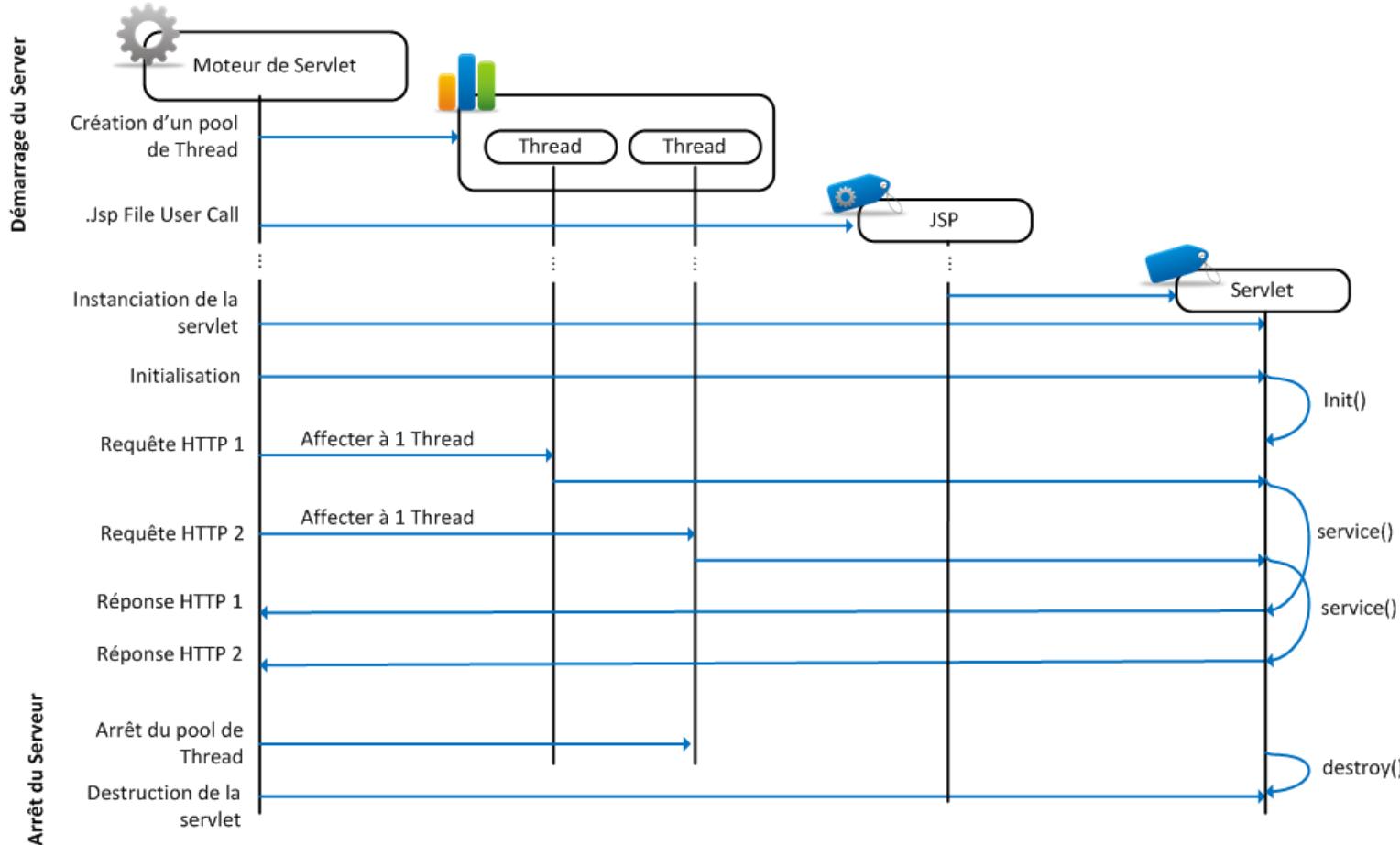
Java Entreprise Edition

• Cycle de vie 2/3



Java Entreprise Edition

• Cycle de vie 3/3



Java Entreprise Edition

• Java Server Page

- Pour simplifier l'écriture, on dispose de variables

prédéfinies :

- request : la requête HttpServletRequest
- response : la réponse HttpServletResponse
- session : instance de HttpSession
- out : le PrintWriter utilisé pour écrire sur la sortie
- application : Cette variable correspond à l'objet ServletContext obtenu grâce à getServletConfig().getServletContext(). Permet aux JSPs d'enregistrer des données persistantes
- config : instance de ServletConfig associée à la page courante
- pageContext : permet de partager des informations entre plusieurs pages
- page : synonyme de this, peu utilisé



Java Entreprise Edition

• Mise en Oeuvre

- Expression (insertion direct de valeur dans la sortie)

<%= expression %>

- Scriptlets: portion de java

<% code %>

- Déclarations

<%! code%>



Java Entreprise Edition

• Mise en Œuvre : Déclaration

- ❑ **Une déclaration ou directive JSP** affecte la structure globale de la Servlet qui est générée pour la JSP
- ❑ **Import** : spécifie les classes java à importer dans la classe courante



```
<%@ page import="java.util.Vector,java.io.*" %>
```

- ❑ **Content Type:** permet de spécifier le type MIME du document retourné au client

```
<%@ page contentType="text/html" %>
```

Équivalent à :

```
<% response.setContentType("text/html");
```

Java Entreprise Edition

• Mise en Œuvre : Déclaration

- ❑ **Session:** spécifie si la page participe à une session

```
<%@ page session="true" %>
```



- ❑ **ErrorPage:** définit une page à appeler afin de traiter les exceptions dans la page courante

```
<%@ page errorPage="error/myErrorPage.jsp" %>
```

- ❑ **isErrorHandler:** permet de spécifier si la page courante peut servir de page d'erreur pour une autre JSP

```
<%@ page isErrorPage="true" %>
```

Java Entreprise Edition

- Exemple

```
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.Vector, java.util.ListIterator"%>
<html>
<head>
    <title>JSP PAGE</title>
</head>
<body>

<h1>Sample Page</h1>
<%
Vector contacts = new Vector();
contacts.add("Ulysse");
contacts.add("Nono");
contacts.add("Hermes");
contacts.add("Calypso");
contacts.add("Helene");
%>
```

Déclaration
Classe Java

Code Java

Declaration
taDonnées

Java Entreprise Edition

- Exemple

```
<p>People List</p>

<ul>
    <%
        ListIterator lEnum = contacts.listIterator();
        while (lEnum.hasNext()) {
            String aName = (String) lEnum.next();
            %>
    <li><%=aName%></li>
    <%
    }
    %>
</ul>

</body>
</html>
```

Début
Code Java

Insertion
directe
d'une valeur

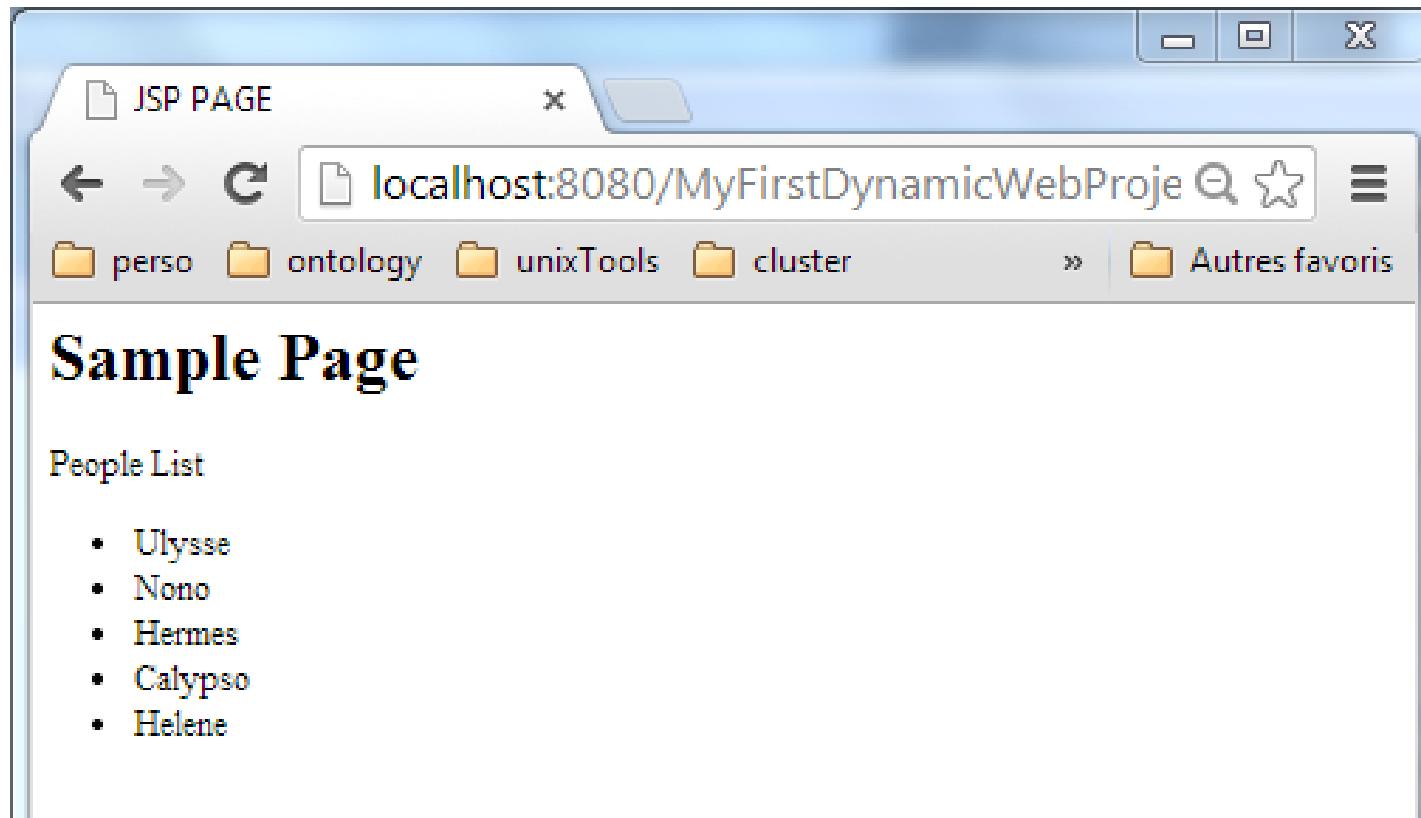


Pas de ; à
la fin

Fin
Code Java

Java Entreprise Edition

- Exemple: Résultat



Java Entreprise Edition

- A vous de Jouer !

- ❑ Créer une Servlet comme URL par default

- ❑ Création d'une liste d'utilisateurs à l'initialisation

- ❑ Une classe utilisateur

- nom, prenom, age, pwd, preference

- ❑ à l'appel de la servlet (GET) redirection vers affichage.jsp en ajoutant les propriétés d'un utilisateur tiré au hasard.

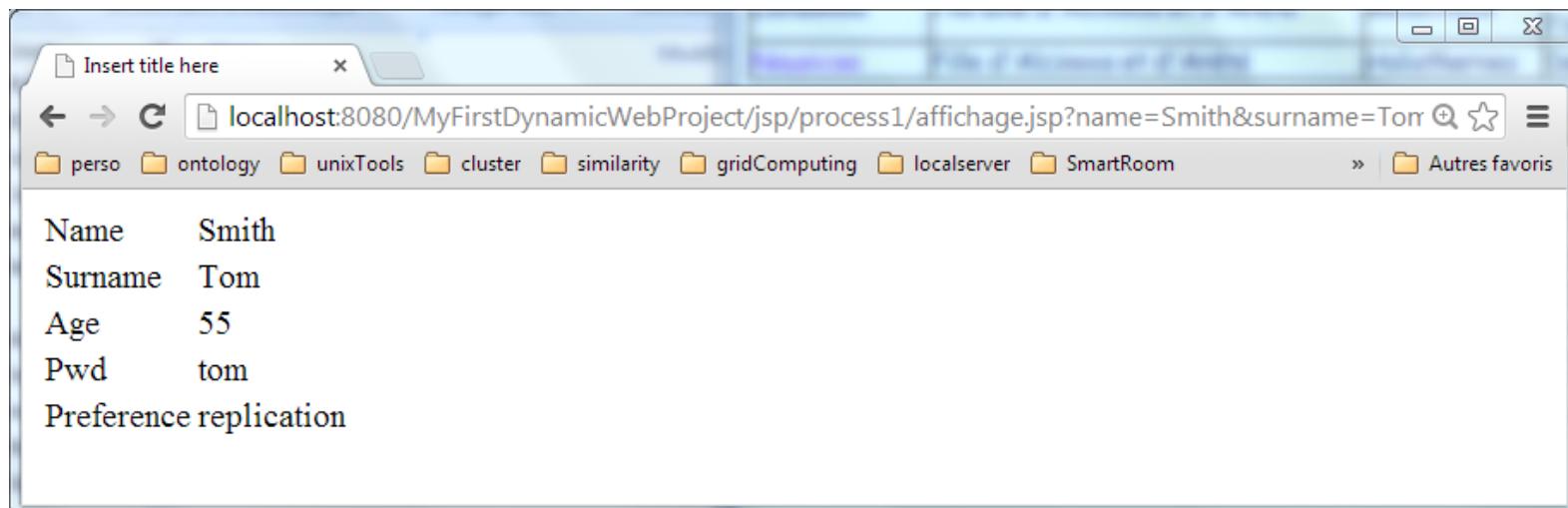
- ❑ Créer une JSP affichage.jsp

- ❑ Affichage de la liste des utilisateurs et de leur propriétés sous forme de tableau



Java Entreprise Edition

- A vous de Jouer !



Communication et partage d'information

Java Entreprise Edition

• Partage d'information

❑ Les cookies

→ Espace de données disponible sur les navigateurs web

❑ Les données de sessions/application

→ Espace de mémoire du Web Container

❑ Les bases de données

→ Requêtes sur base de données

❑ Les JavaBeans

→ Outils de partage d'information web/jsp – Java



Java Entreprise Edition

• Les cookies

- ❑ Utilisation du package javax.servlet.http.Cookie
- ❑ Stockage sous forme clé/valeur
- ❑ Propriétés optionnelles importantes
 - ❑ Max Age : durée de rétention de l'information
 - ❑ Domain : domaine auquel l'information peut –être associée
 - ❑ Path: chemin où le client do renvoyer l'information

❑ Récupération des cookies sur HttpServletRequest

```
Cookie[] cookieList = request.getCookies();
cookieList[0].getPath();
cookieList[0].getValue();
```

❑ Ajout de cookie à l'aide de HttpServletResponse

```
Cookie cook=new Cookie("Name", "value1");
cook.setMaxAge(50);
response.addCookie(cook);
```



Java Entreprise Edition

- A vous de Jouer !

- ❑ Réalisation d'une page d'authentification JSP auth.jsp
- ❑ Créer une Servlet vérifiant login Mot de passe dans une liste prédéfinie
- ❑ Auth.jsp devra afficher dans le champ login l'ancienne valeur de login stockée dans les cookies
- ❑ La servlet de vérification devra stocker le login dans les cookies du navigateur web



Java Entreprise Edition

• Les données de Session-Application

- ❑ Web container possède 2 espaces de mémoires utilisables
 - ❑ Application: disponible entre toutes les Servlets et JSP
 - ❑ Session: disponible uniquement durant la session d'un utilisateur

- ❑ Définition Session utilisateur



Série d'interactions utilisateur/application étant suivies par le serveur. Les sessions sont utilisées pour maintenir l'état d'un utilisateur (incluant la persistance des objets associés), pour authentifier l'identité de l'utilisateur

Java Entreprise Edition

• Session

- Récupération de la session utilisateur via HttpServletRequest
- Portée: uniquement les servlet/Jsp participant à la session de l'utilisateur courant
 - API
 - `getAttribute(String key)`
 - `setAttribute(String key, Object value)`

True: creation session si elle n'existe pas

```
HttpSession session = request.getSession(true);  
session.setAttribute("USER_VALUE",  
    new UserBean("Doe", "John", 42, "test", "GTAV"));  
UserBean u=(UserBean) session.getAttribute("USER_VALUE");
```

Java Entreprise Edition

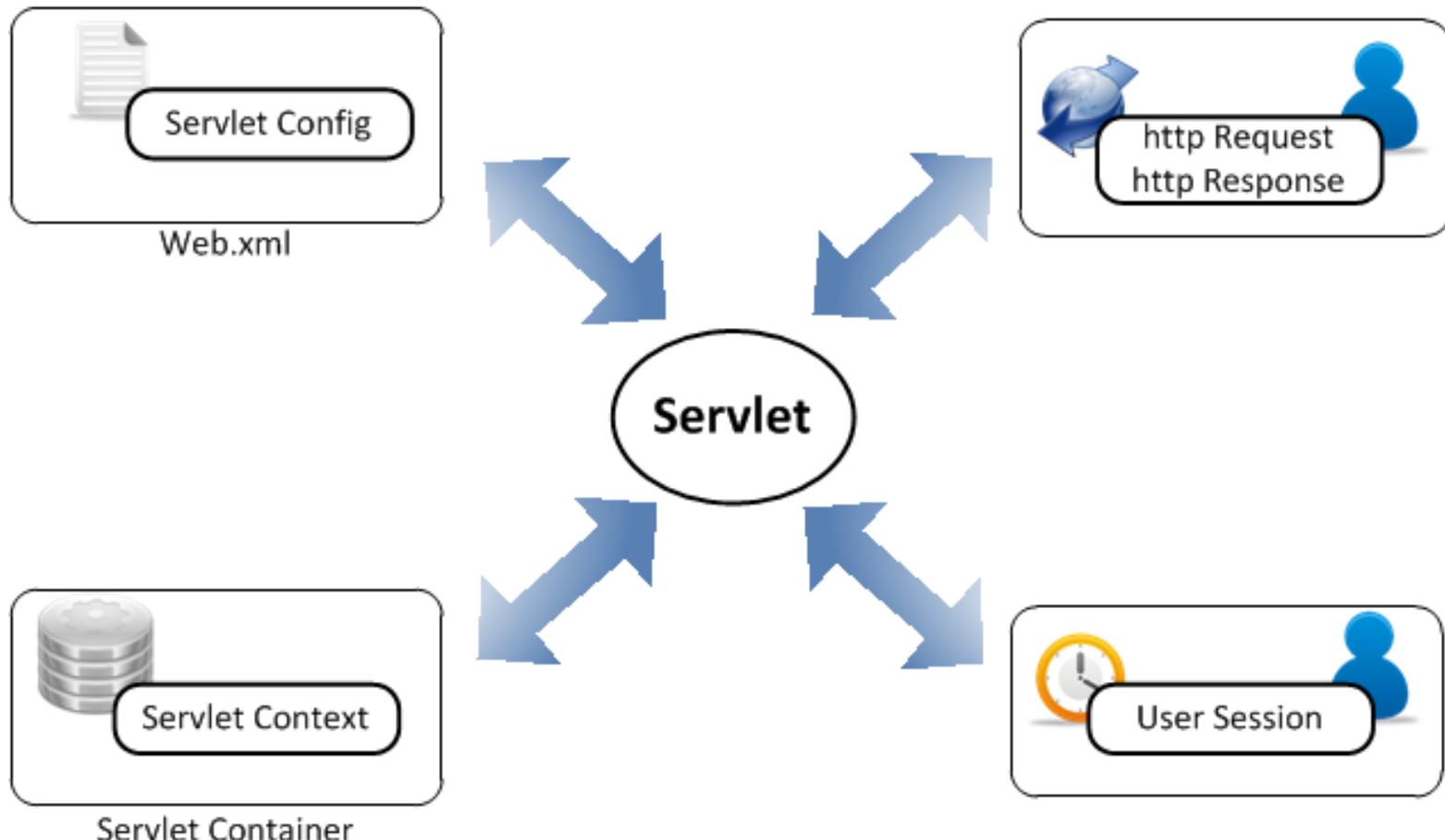
• Application: ServletContext

- Récupération de la session utilisateur via HttpServletRequest
- Portée: L'ensemble des servlet/jsp (voir configuration serveur)
 - API
 - `getAttribute(String key)`
 - `setAttribute(String key, Object value)`

```
ServletContext context = request.getServletContext();
context.setAttribute("USER_VALUE",
                     new UserBean("Doe", "John", 42, "test", "GTAV"));
UserBean user=(UserBean)
context.getAttribute("USER_VALUE");
```

Java Entreprise Edition

- Blian: comment échanger de l'information



Java Entreprise Edition

- A vous de Jouer !

- Créer une Servlet comme URL par default
 - Dans cette servlet
 - créer un compteur si il n'existe pas dans la mémoire Session
 - créer un compteur si il n'existe pas dans la mémoire Servlet Context
 - A chaque appel de la servlet, cette dernière redirigera vers une page affichant
 - la valeur du compteur session
 - la valeur du compteur context
 - Ping une fois sur deux, Pong sinon



Les JavaBeans

Java Entreprise Edition

• Les JavaBeans

❑ Définition: Composant accessible réutilisable

❑ Caractéristiques:

- La persistance : sauvegarde/ restauration de l'état du bean grâce à la sérialisation
- La communication: Interaction via des évènements
- L'introspection : Découverte dynamique des éléments du bean (sans code source).
- Composant Paramétrable: les données du paramétrage sont conservées dans des propriétés.

❑ Contraintes

- Constructeur sans paramètre
- Doit être sérialisable (implements Serializable)
- Propriétés normalisées: attribut=nom, méthode d'accès getNom(),
setNom(String s)
- Peuvent émettre des évènements en gérant une liste de listener.



Java Entreprise Edition

• Utilisation

- ❑ JSP interaction direct avec les objets java
- ❑ JSP échange d'information grâce à la portée des JavaBean
 - ❑ Syntaxe

```
<jsp:useBean id="nom_du.Bean" class="package.Class"/>

<jsp:getProperty name="nom_du.Bean" property="nom_propriété">

<jsp:setProperty name="nom_du.Bean" property="nom_propriété" value="valeur"/>
```

❑ Exemple

```
<jsp:useBean id="MyUser" class= "bean.UserBean" />

<jsp:getProperty name= "MyUser" property= "age" />

<jsp:setProperty name= "MyUser" property= "preference" value= "Turtle" />
```

Java Entreprise Edition

• Utilisation

- ❑ Portée d'un JavaBean
- ❑ Scope

```
<jsp:useBean id="nom_du_bean" class="package.Class" scope="valeur_du_scope"/>  
  
<jsp:useBean id="MyUser" class= "bean.UserBean"  scope="session" />
```

❑ Valeurs possibles du scope

- ❑ Page
- ❑ Request
- ❑ Session
- ❑ application

```
import java.io.Serializable;

public class UserBean implements Serializable {
    private String name;
    private String surname;
    private int age;
    private String pwd;
    private String preference;
        public UserBean() {}
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getSurname() {
            return surname;
        }
        public void setSurname(String surname) {
            this.surname = surname;
        }
        public int getAge() {
            return age;
        }
        public void setAge(int age) {
            this.age = age;
        }
}
```

...

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Register</title>

    <jsp:useBean id="myUser" scope="session" class="beans.UserBean">
        <!-- INITIALISATION -->
        <jsp:setProperty name="myUser" property="name" value="none" />
        <jsp:setProperty name="myUser" property="surname" value="none" />
        <jsp:setProperty name="myUser" property="age" value="0" />
        <jsp:setProperty name="myUser" property="preference" value="EVERY THING IS GOOD" />
    </jsp:useBean>

    <jsp:setProperty name="myUser" property="name" />
    <jsp:setProperty name="myUser" property="surname" />
    <jsp:setProperty name="myUser" property="age" />
</head>

<body>
    <form action="register.jsp" method="post">
        Put your name <input type="text" name="name" />
        Put your surname <input type="text" name="surname" />
        <input type="submit" value="GO!" />
    </form>
    Current name <jsp:getProperty name="myUser" property="name"/>
    Current surname <jsp:getProperty name="myUser" property="surname"/>
    Current age <jsp:getProperty name="myUser" property="age"/>
    Current preference <jsp:getProperty name="myUser" property="preference"/>
</body></html>
```

Java Entreprise Edition

- Sample

localhost:8080/JspJavaBeanUsage/register.jsp

Applications perso ontology unixTools cluster simil

Form

Put your name

Put your surname

Current Content

Current name none
Current surname none
Current age 0
Current preference EVERY THING IS GOOD

localhost:8080/JspJavaBeanUsage/register.jsp

Applications perso ontology unixTools cluster simil

Form

Put your name

Put your surname

Current Content

Current name Doe
Current surname John
Current age 0
Current preference EVERY THING IS GOOD

Before Submit

After Submit

Java Entreprise Edition

- A vous de Jouer !

- Créer un fichier JSP permettant de remplir les propriétés d'un utilisateur (name, surname, age, preference)
- Créer dans ce fichier JSP un Bean permettant de sauvegarder ces informations
- Créer un second JSP affichant les propriétés de l'utilisateur



Java Entreprise Edition

• Bonnes pratiques: Servlet/JSP/JavaBean Qui fait quoi?

❑ Servlet

- Adapté pour les traitements complexes, orchestration des pages
- Non adapté à la création de rendu de page



❑ JSP

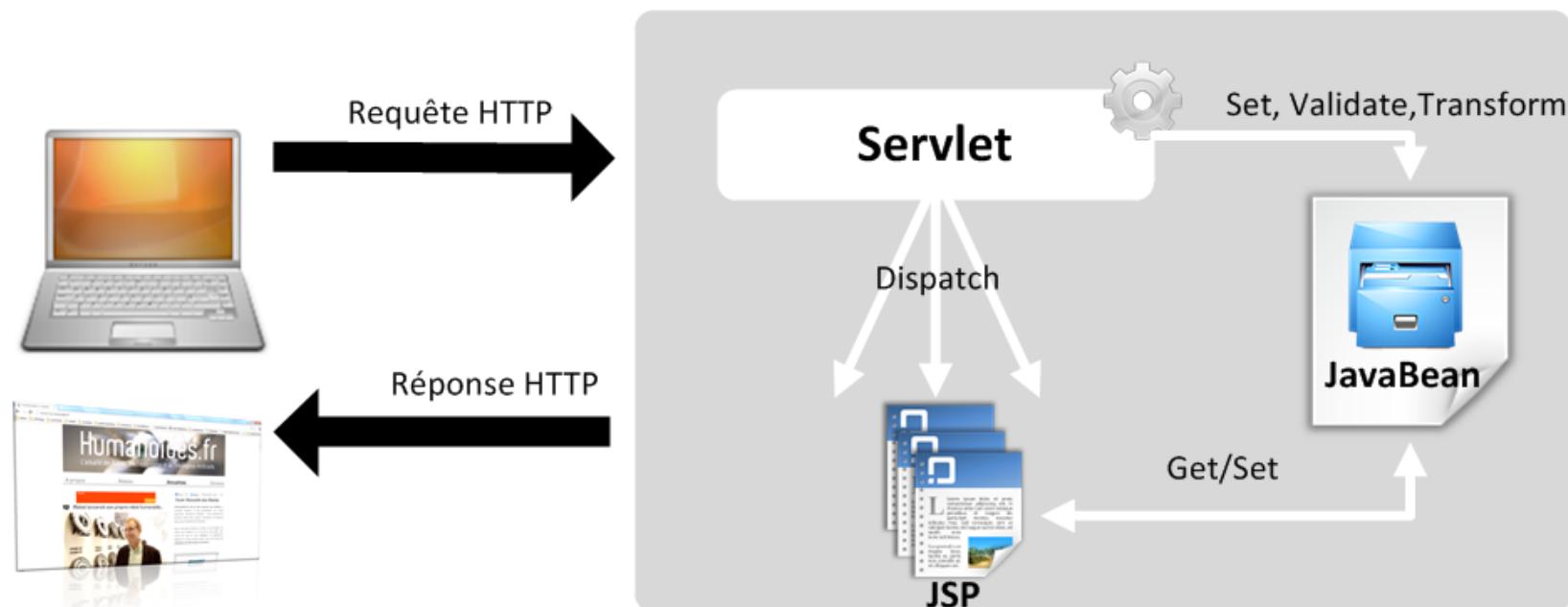
- Adapté à la création de rendu de page
- Inadapté aux traitements complexes

❑ JavaBean

- Container de données
- Lien object java / objet manipuler en JSP

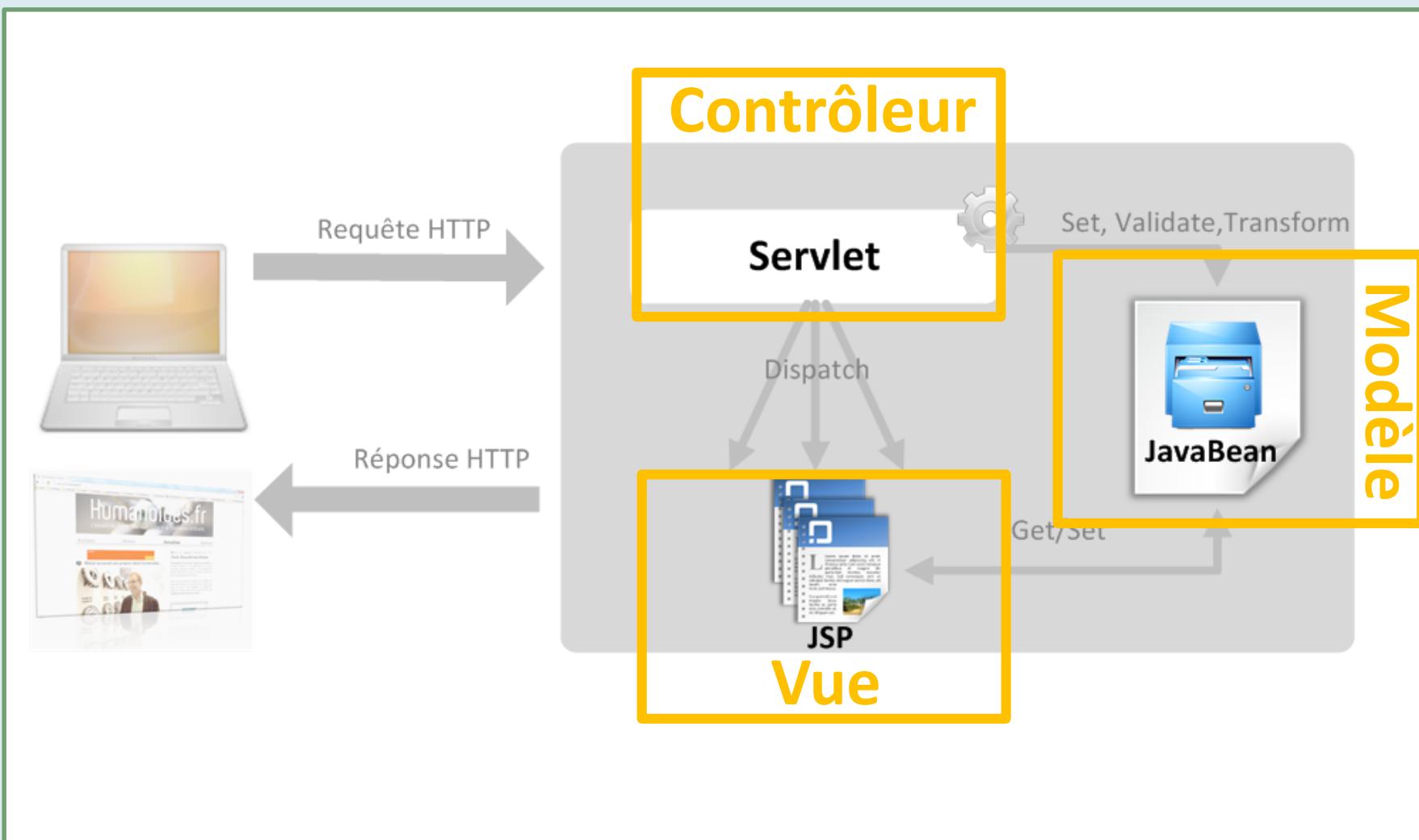
Java Entreprise Edition

- Bonnes pratiques: MVC, Modèle Vue Contrôleur



Java Entreprise Edition

- Bonnes pratiques: MVC, Modèle Vue Contrôleur



Java Entreprise Edition

• Outils complémentaires: Servlet

❑ Dispatcher

- Déléguer des requêtes HTTP vers une autre Servlet ou une autre JSP

❑ dispach.forward

- Les éléments de requêtes et de réponses sont FORWARDÉ au composant désigné. La servlet à l'origine du dispatch ne pourra plus modifier la réponse.

❑ dispach.include

- Les élément de requêtes et de réponses sont INCLUS au composant désigné. La réponse de la servlet d'origine ET la réponse du composant désigné seront fusionnées.



Java Entreprise Edition

• Outils complémentaires

```
//Récupération du ServletContext, création d'un dispatcher à la
// destination '/follow.jsp'
RequestDispatcher dispatch =
getServletContext().getRequestDispatcher(
"/follow.jsp");

// Pour redirection
dispatch.forward(request, response);

// Pour inclusion
dispatch.include(request, response);
```

Java Entreprise Edition

• Outils complémentaires: JSP

- Comment limiter l'impact des modifications effectuées sur les pages de visualisation

→ Centraliser l'utilisation des composants: **include**

- **@include:** copie colle le contenu du fichier à l'endroit de l'appel

```
<%@include file='header.html'%>  
<%@include file='introduction.html'%>
```

→ Centraliser l'utilisation des composants: TagLib

- **jsp:include:** inclusion du résultat et non pas du code, n'utilise pas les mêmes informations que la page courante

```
<jsp:include page="header.html">  
<jsp:include page="introduction.html">
```



Java Entreprise Edition

- A vous de Jouer !

- Une seule JSP doit construire le formulaire d'enregistrement utilisateur
- Votre JSP doit utiliser le Bean UserInfo pour valider votre saisie et afficher les messages d'erreurs



Java Entreprise Edition

- A vous de Jouer !

The screenshot displays two browser windows side-by-side, illustrating a user input validation scenario.

Left Browser Window: The title bar reads "exo9 simple page JSP invalidant des entrées utilisateur © Ph. Isorce 2007 - Mozilla...". The page content includes a message: "Veuillez remplir tous les champs avec des valeurs valides". Below this, there are five input fields with their respective placeholder text and validation instructions:

- Nom :
- Date de naissance : (Utilisez le format aaaa-mm-jj)
- Adresse e-mail : (Utilisez le format nom@societe.com)
- Sexe : (M : masculin ou F : féminin)
- Chiffre porte bonheur : (Un nombre entre 1 et 100)

At the bottom left is a blue "Envoyer" button, and at the bottom right is a "Terminé" link.

Right Browser Window: The title bar reads "exo9 simple page JSP invalidant des entrées utilisateur © Ph. Isorce 2007 - Mozilla...". The address bar shows "http://localhost:8080/mesjssps/mapage4.jsp". The page content displays an error message: "Les valeurs suivantes ont été oubliées ou invalides :". A red bulleted list follows:

- Date de naissance oubliée
- Chiffre porte bonheur oublié
- Sexe oublié

Below this, a form titled "SVP Entrez de nouvelles valeurs valides" is shown. It contains five input fields with pre-filled values and their respective validation messages:

- Nom :
- Date de naissance :
- Adresse e-mail :
- Sexe :
- Chiffre porte bonheur :

To the right of the input fields, explanatory text is provided: "Utilisez le format aaaa-mm-jj", "Utilisez le format nom@societe.com", "(M : masculin ou F : féminin)", and "(Un nombre entre 1 et 100)". A blue "Envoyer" button is located below the fields, and a "Terminé" link is at the bottom right.

JEE et Web Service REST

Java Entreprise Edition

• JAX-RS

- ❑ Permet de créer des web services RESTfull
 - ❑ Chaque ressource est identifiée par une URI
 - ❑ Chaque ressource est manipulé par une interface uniforme (http PUT, GET, POST, DELETE)
 - ❑ Chaque Interaction est sans état (stateless)



- ❑ Utilisation d'annotation afin d'interagir avec la servlet

```
@GET  
@Produces(MediaType.APPLICATION_JSON) @Path("/{eventId}/")  
public String getData(@PathParam("eventId") Long eventId)  
{  
    return "{\"uuid\":\""+eventId+"\"}";  
}
```

Java Entreprise Edition

• JAX-RS annotations

Annotations	Description
@Path	URI Relative indiquant quelle Classe, Méthode Java sera déclenchée
@GET / @POST / @PUT @DELETE / @ HEAD	Méthode http sur laquelle la méthode java sera déclenchée
@PathParam	Elément de l'URI pouvant être exploité par une méthode java
@QueryParam	Type de paramètre pouvant être extrait de la requête
@Consumes	Définit le MIME media type pouvant être consomé par la méthode java
@Produces	Définit le MIME media type étant produit par la méthode java
@Provider	Permet d'étendre les interactions sur le service JAX-RS: e.g redirection des exceptions java vers des réponses JAX-RS
@ApplicationPath	Définit l'URI de base de toutes les ressources spécifiées par @PATH

Java Entreprise Edition

- JAX-RS

- Plusieurs implémentations

- Apache CXF

- <http://cxf.apache.org/>

- Jersey

- <https://jersey.java.net/>

- RESTEasy

- <http://resteasy.jboss.org/>

- Restlet

- <https://restlet.com/>



Restlet

Java Entreprise Edition

- Exemple de mise en œuvre

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>jee-rs</display-name>
  <servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>com.rest.services</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Web.xml

Java Entreprise Edition

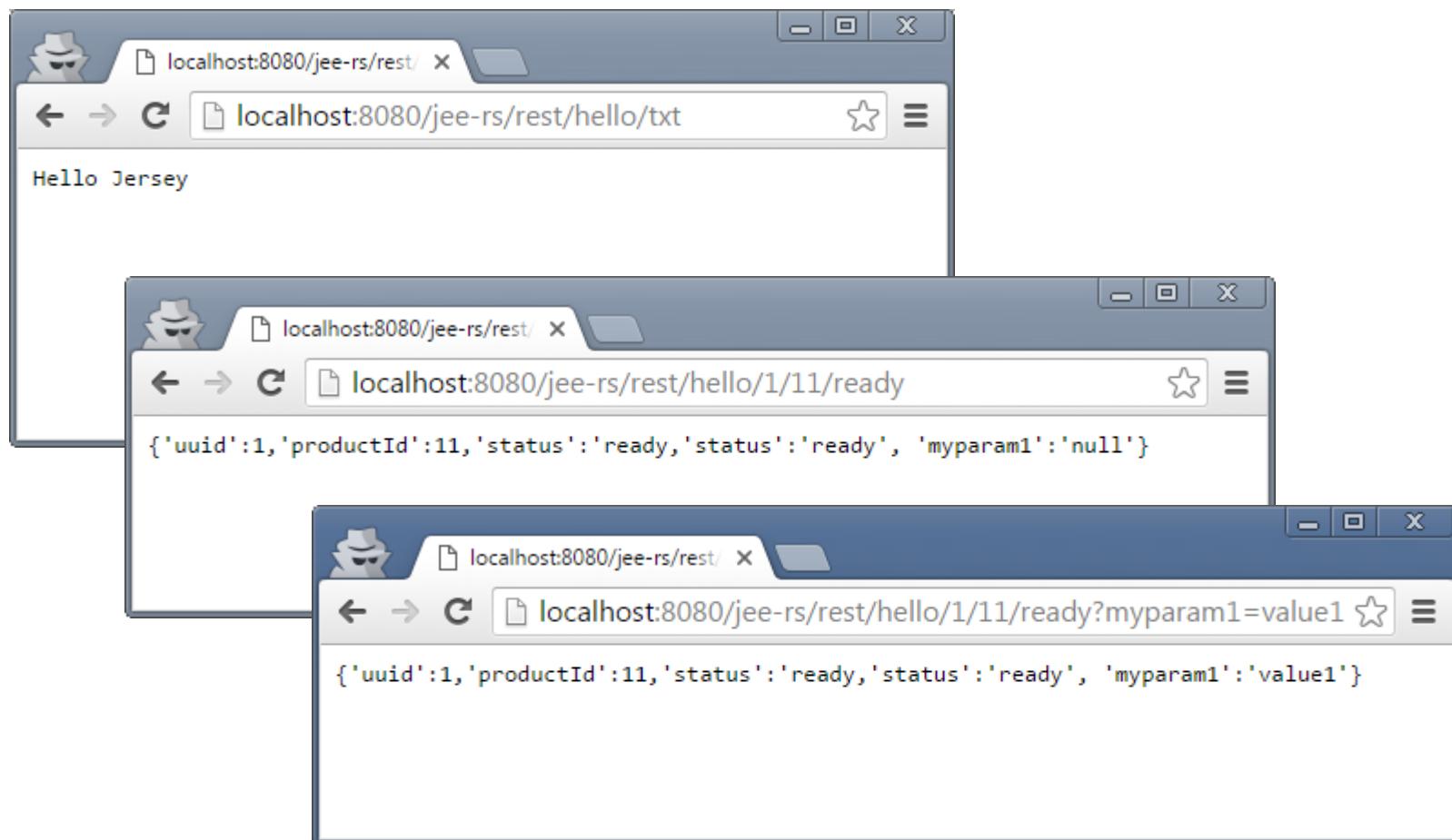
- Exemple de mise en ouvre

```
@Path("/hello")
public class SampleService {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/txt")
    public String sayPlainTextHello() {
        return "Hello Jersey";
    }
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("{eventId}/{productId}/{status}")
    public String getData(@PathParam("eventId") Long eventId,
                         @PathParam("productId") Long productId,
                         @PathParam("status") String status,
                         @QueryParam("myparam1") String value1) {
        return "{uuid:"+eventId+",'productId':" +productId+",'status':'"+status+"', 'myparam1':'" +value1+"'}";
    }
}
```

SampleService.java

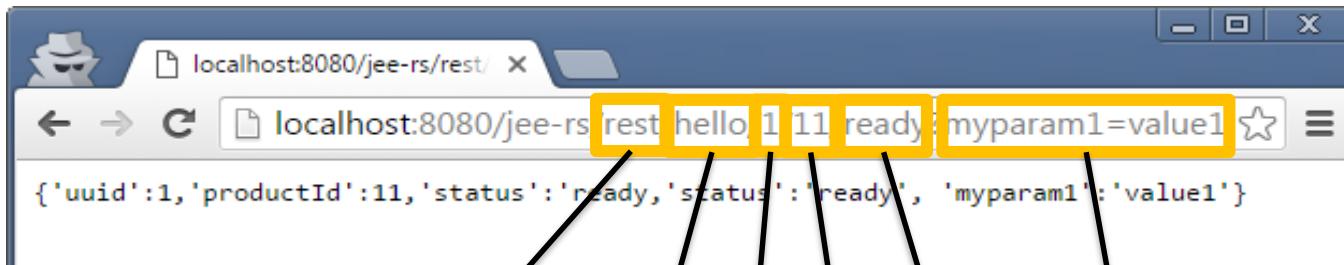
Java Entreprise Edition

• Résultat



Java Entreprise Edition

• Résultat



Web.xml

```
<servlet-mapping>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

`@Path("/hello")`

`@GET`

`@Produces(MediaType.TEXT_PLAIN)`

`@Path("{eventId}/{productId}/{status}")`

```
public String getData(@PathParam("eventId") Long eventId,
                      @PathParam("productId") Long productId,
                      @PathParam("status") String status,
                      @QueryParam("myparam1") String value1) {
```

`return`

```
  "{\"uuid\":\"+eventId+\",\"productId\":\"+productId+\",\"status\":\"+status+\"}";
```

`}`

Copyright © Jacques Saraydaryan

Java Entreprise Edition

- A vous de Jouer !

- Dao.java

- addUser
 - getUser
 - GetAllUsers

- RestServlet.java

- GET /users/list
 - Récupération de la liste de tous les utilisateurs
 - GET /users/jdoe
 - Récupération d'un utilisateur précis e.g jdoe
 - PUT /users/jdoe
 - Ajout d'un utilisateur e.g jdoe



Persistance des données: Le service JDBC

Java Entreprise Edition

• Java Database Connector (JDBC)

- ❑ Permet d'invoquer des requêtes SQL depuis un programme

Java

- ❑ Gestion de plusieurs Drivers (mysql,postgre).
- ❑ Un Driver permet d'assurer la connexion à une SGBD spécifique
- ❑ Utilisation
 - Ouverture de connexion
 - Construction de requêtes
 - SQL,
 - Appel de procédure stockées (*CallableStatement*)
 - Exécution de requêtes
 - Gestion de transactions
 - Manipulation des résultats de la requête



Java Entreprise Edition

• Java Database Connector (JDBC)

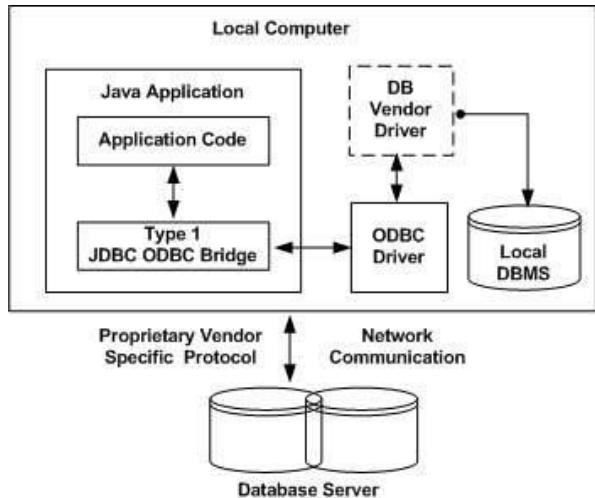
- Il existe 4 types de drivers JDBC :
- Type 1 - Pont JDBC-ODBC** : API Java pour accéder à une ou plusieurs sources de données ODBC
- Type 2 - API Native, partiellement Java** : convertit les appels JDBC en appel base de données natifs Oracle, Sybase etc. Inconvénient : du code machine doit être chargé sur la machine.
- Type 3 - Protocole réseau Java** : Convertit les appels JDBC dans un protocole réseau indépendant, puis sont ensuite convertis en appels dans le protocole du SGBD.
- Type 4 - Protocole natif Java** : convertit les appels JDBC dans le protocole réseau utilisé par le SGBD. Ceci permet des appels directs entre le client et son SGBD.



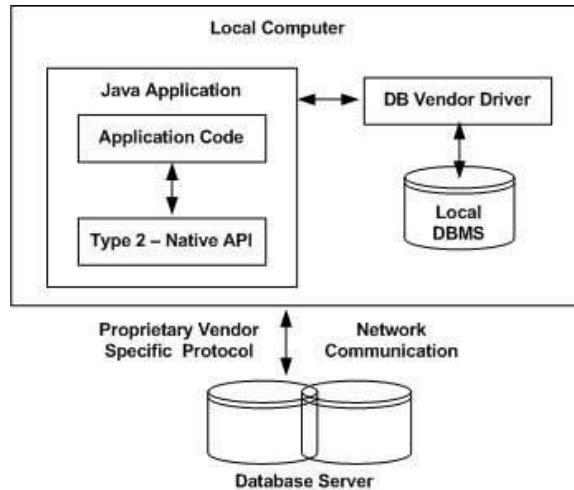
Java Entreprise Edition

- Java Database Connector (JDBC)

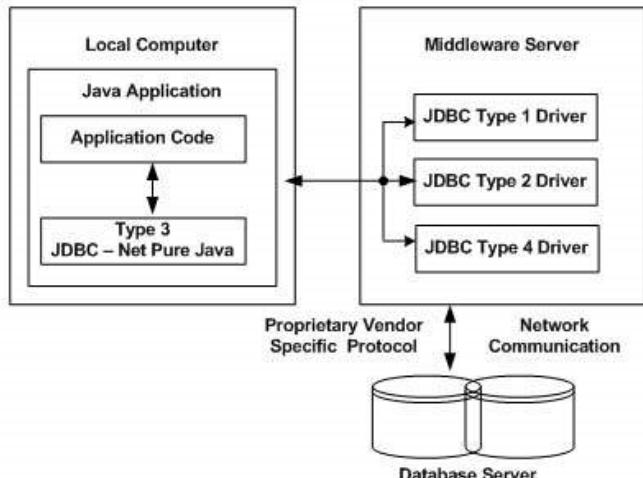
Type 1



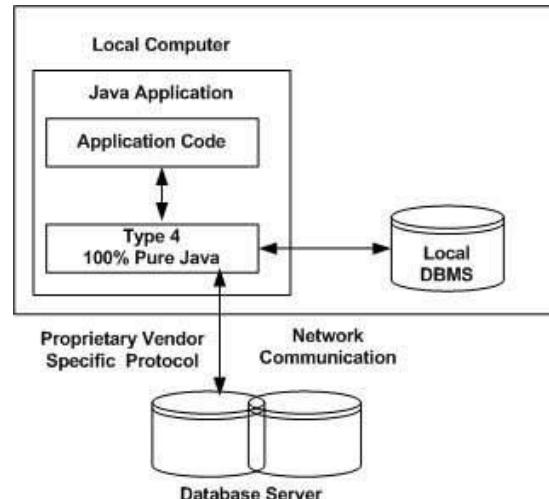
Type 2



Type 3



Type 4



Java Entreprise Edition

• Interface Connection

- ❑ Encapsule toutes les caractéristiques et tous les comportements d'une connexion de base
- ❑ Assure
 - ❑ Gestion des transactions
 - ❑ Commit(): valide les modifications
 - ❑ Rollback(): annule les modifications
 - ❑ Gestion des requêtes
 - ❑ createStatement()
 - ❑ prepareStatement()
 - ❑ prepareCall()



Java Entreprise Edition

• Interface Statement

- ❑ Interface pour l'exécution de requête SQL statique
- ❑ L'instance du type est créée par la méthode **createStatement()** de **Connection**.
- ❑ 2 Modes d'exécution
 - ❑ **executeQuery(String)** : requêtes de sélection
 - ❑ **executeUpdate(String)** : requêtes d'insertion, suppression, update



Java Entreprise Edition

• Exemple simple

```
// Chargement du Driver, puis établissement de la connexion
Class.forName("com.mysql.jdbc.Driver");
java.sql.Connection conn =
    java.sql.DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/test" , "root", "7777");

// Création de la requête
java.sql.Statement query = conn.createStatement();

// Executer puis parcourir les résultats
java.sql.ResultSet rs = query.executeQuery( "SELECT * FROM Chaine");
while( rs.next() )
{
    System.out.println("Chaine : " + rs.getString( "nom" ) );
}
rs.close();
query.close();
conn.close();
```

Java Entreprise Edition

• Interface PreparedStatement

- ❑ Construction de requêtes SQL compilées et stockées dans l'instance → Optimisation d'exécution lors d'appels successifs
- ❑ L'instance du type est créée par la méthode **prepareStatement()** de **Connection**.
- ❑ Modes d'exécution identique à Statement
- ❑ Permet la création de requêtes paramétrées
 - ❑ Paramètres représentés par un ? Dans le code
 - ❑ Définition des valeurs des paramètres à l'aide de **setString(int)**, **setInt(int)...**



Java Entreprise Edition

- **Usage : Recherche d'information**

```
// Chargement du Driver, puis établissement de la connexion
Class.forName("com.mysql.jdbc.Driver");
java.sql.Connection cnx = java.sql.DriverManager.getConnection(
"jdbc:mysql://localhost:3306/testdb", "root", "pwd");

// Cr ation de la requ te
java.sql.Statement query = cnx.createStatement();
// Executer puis parcourir les r sultats
java.sql.ResultSet rs = query.executeQuery("SELECT * FROM Chaine");

// Cr ation de la requ te compil e
PreparedStatement querySt = cnx.prepareStatement("select * from chaine where code=?");
// D efinition de la valeur du premier param tre
querySt.setString(1, "TF1");
// Ex cution
ResultSet rst = querySt.executeQuery();

while (rs.next()) {
    System.out.println("Chaine : " + rs.getString("nom"));
}
rs.close();
query.close();
cnx.close();
```

Java Entreprise Edition

- **Usage : Modification de la base**

```
// Chargement du Driver, puis établissement de la connexion
Class.forName("com.mysql.jdbc.Driver");
java.sql.Connection cnx = java.sql.DriverManager.getConnection(
"jdbc:mysql://localhost:3306/testdb", "root", "pwd");

// Création de la requête
java.sql.Statement query = cnx.createStatement();

//Création du contenu de la requête à exécuter
String sql = "CREATE TABLE REGISTRATION " +
    "(id INTEGER not NULL, " +
    " first VARCHAR(255), " +
    " last VARCHAR(255), " +
    " age INTEGER, " +
    " PRIMARY KEY ( id ))";

//Exécution de la requête de modification
query.executeUpdate(sql);

cnx.commit();
//cnx.rollback();
query.close();
cnx.close();
```



La méthode retourne un entier représentant le nombre de lignes mises-à-jour, ou 0 si la requête ne fait rien.

Java Entreprise Edition

- A vous de Jouer !

- ❑ Un formulaire inscription utilisateur
- ❑ Un javabean associé à ce formulaire
- ❑ Un Object Data Acces Object (DAO) gérant les requetés en base de données et stocké dans le servlet context
- ❑ Lorsque l'utilisateur est valide ce dernier sera enregistré dans la base de données

