Année 2019 - 2020

Cours 01:

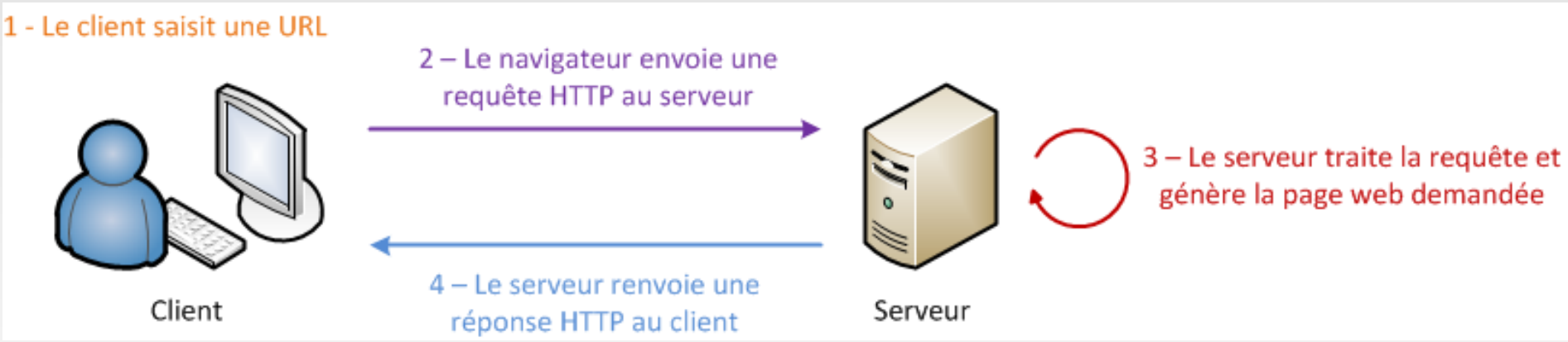
Java Front Office

Hayri ACAR
hayri.acar@reseau-cd.net

Introduction Java EE

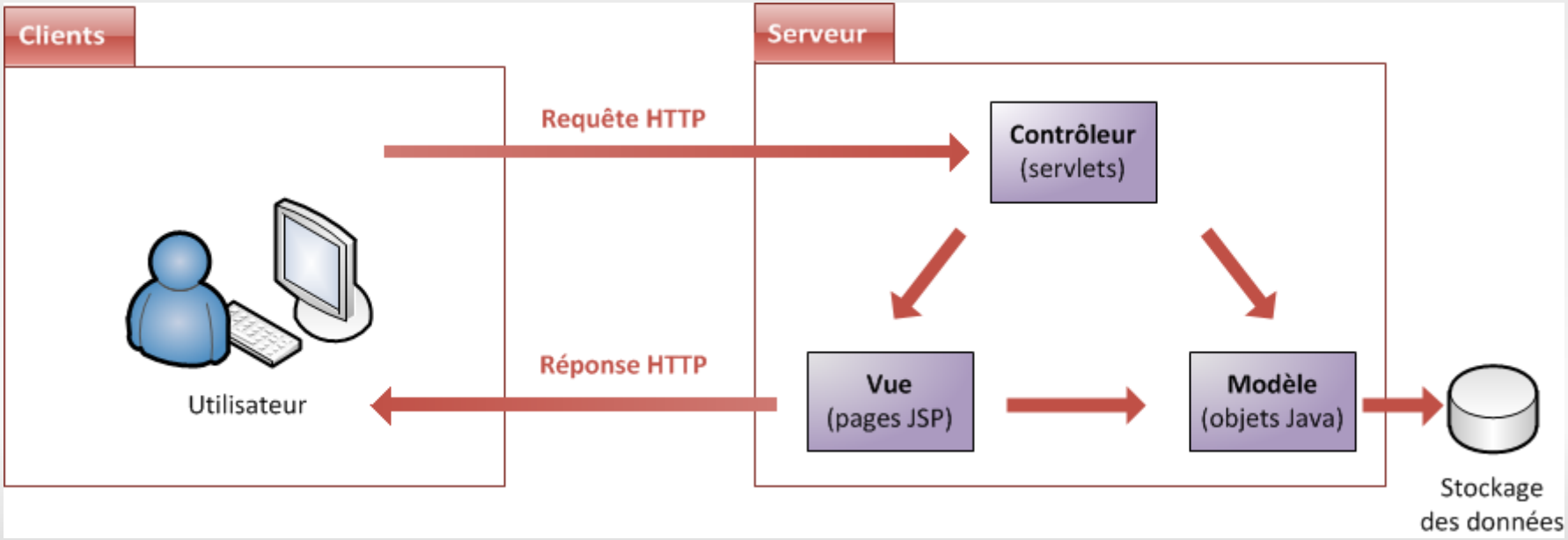
- Prérequis :
  - Notions Java
  - Notions HTML/CSS
  - Notions SQL

- Java EE (Java Enterprise Edition) différent de Java et de Javascript.
- Internet <> web
  - Sites statiques : HTML, CSS
  - Sites dynamiques : HTML, CSS, Java EE ou PHP ou .NET ou Django ou …


1 - Le client saisit une URL
2 – Le navigateur envoie une requête HTTP au serveur
3 – Le serveur traite la requête et génère la page web demandée
4 – Le serveur renvoie une réponse HTTP au client
Client
Serveur

Modèle MVC

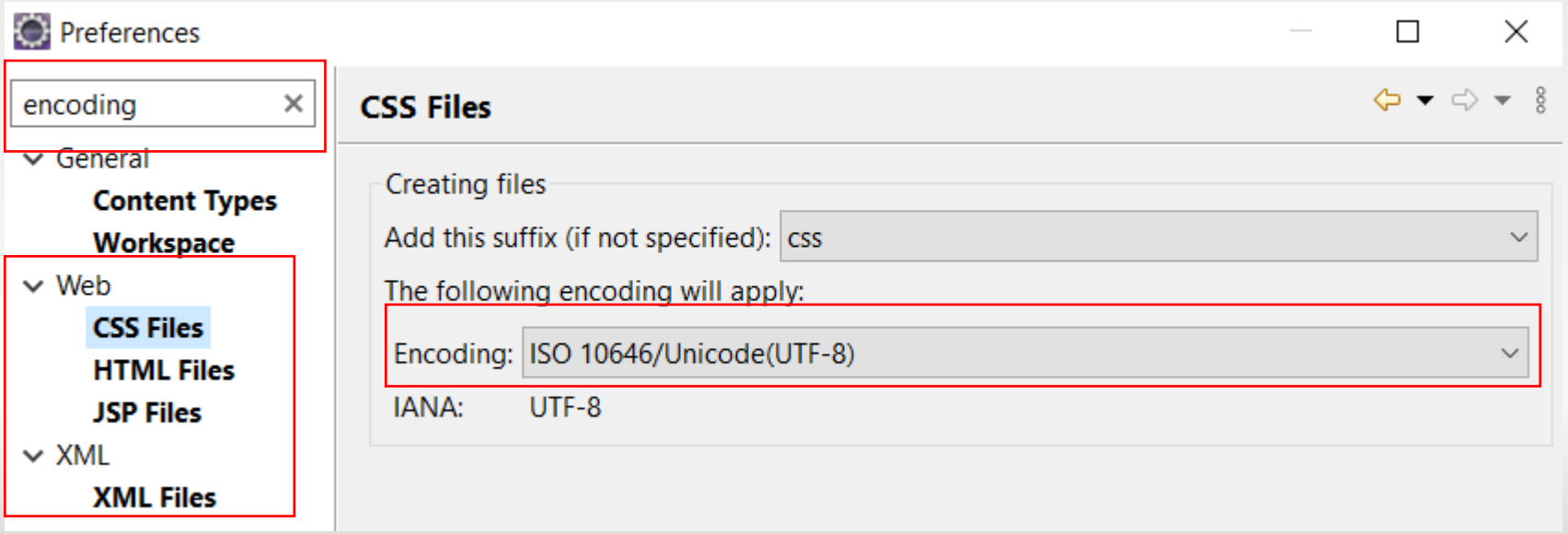
- **Modèle** (M) : traitement, stockage et mise à jour des données de l'application. Objets Java : attributs et méthodes.
- **Vue** (V) : interaction avec l'utilisateur et présentation des données (mise en forme, affichage). Pages JSP.
- **Contrôleur** (C) : contrôle des actions de l'utilisateur et des données. Servlet : intercepter requêtes client, personnaliser une réponse.



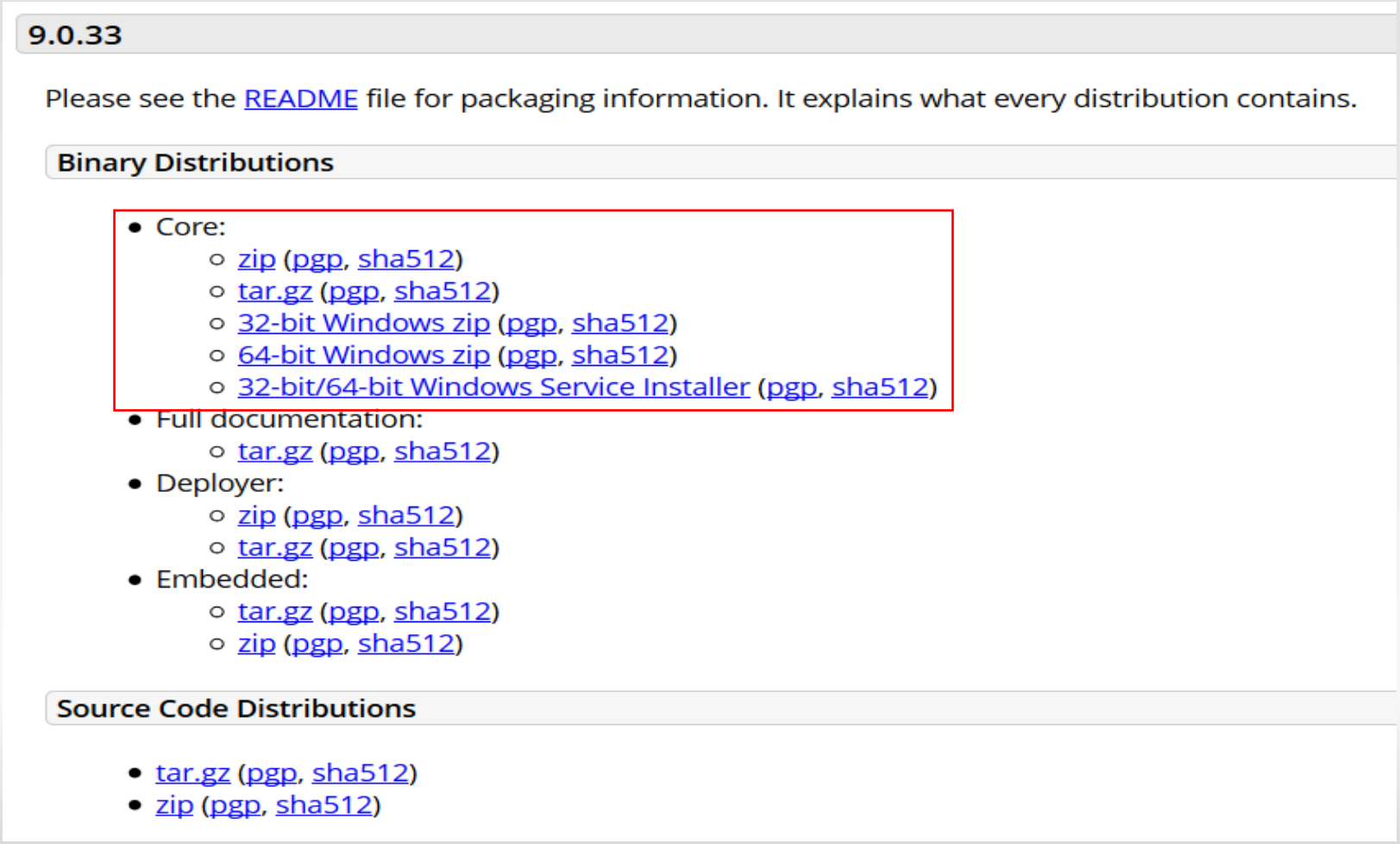
Outils

- Eclipse Java EE : https://www.eclipse.org/downloads/packages/

Windows->Preferences



- Tomcat : https://tomcat.apache.org/download-90.cgi



9.0.33

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

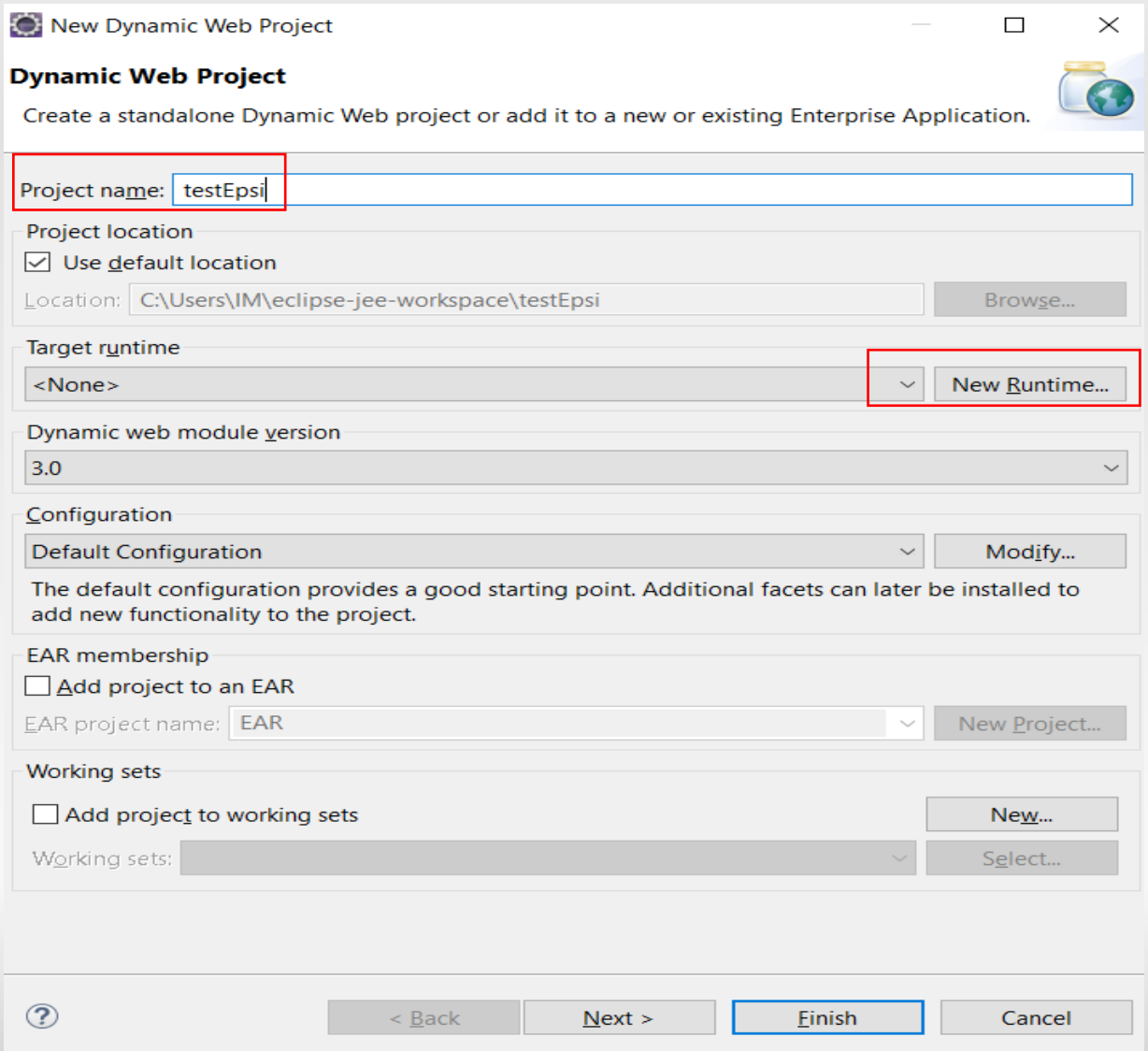
- Core:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
  - 32-bit Windows zip (pgp, sha512)
  - 64-bit Windows zip (pgp, sha512)
  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

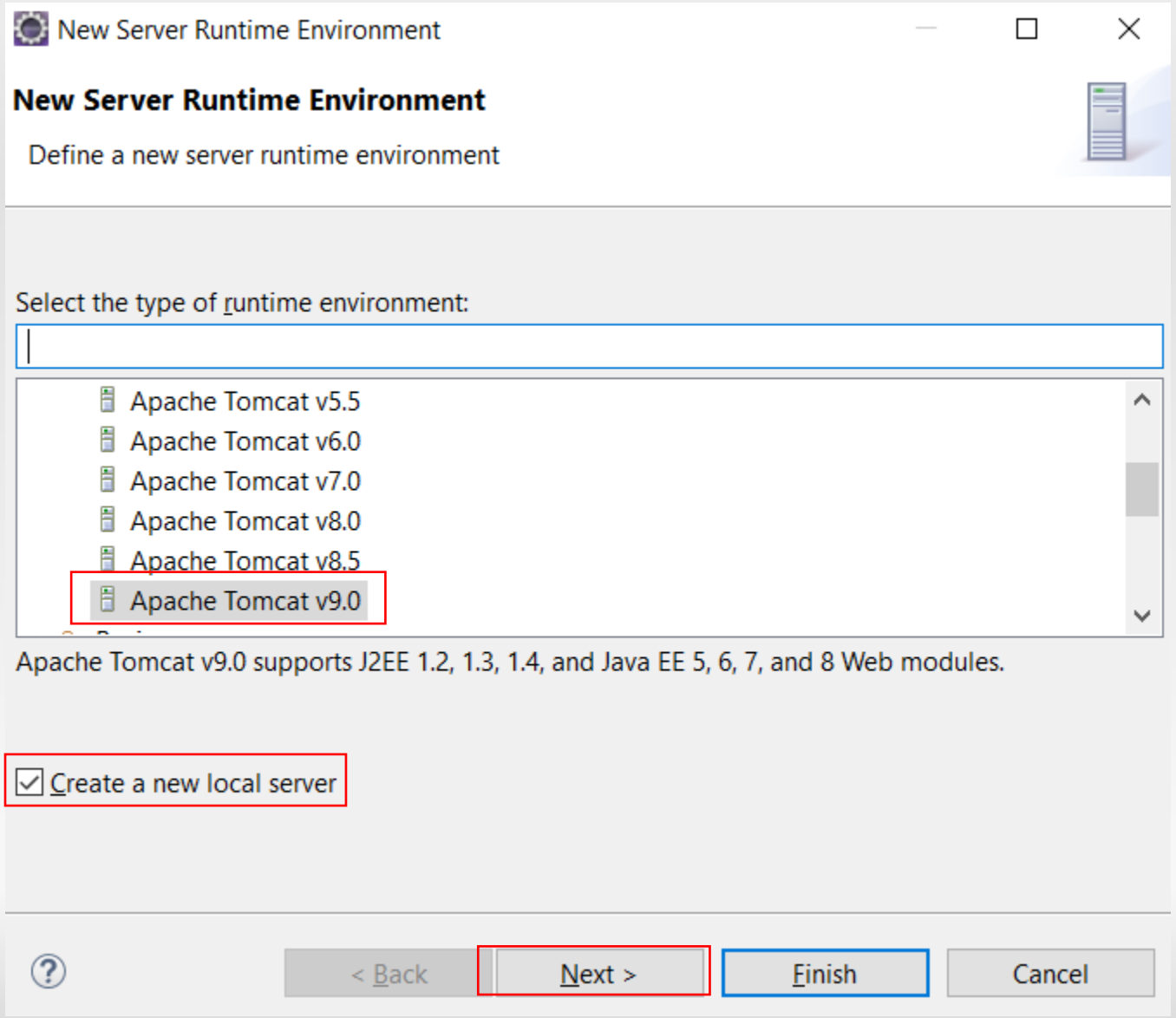
Source Code Distributions

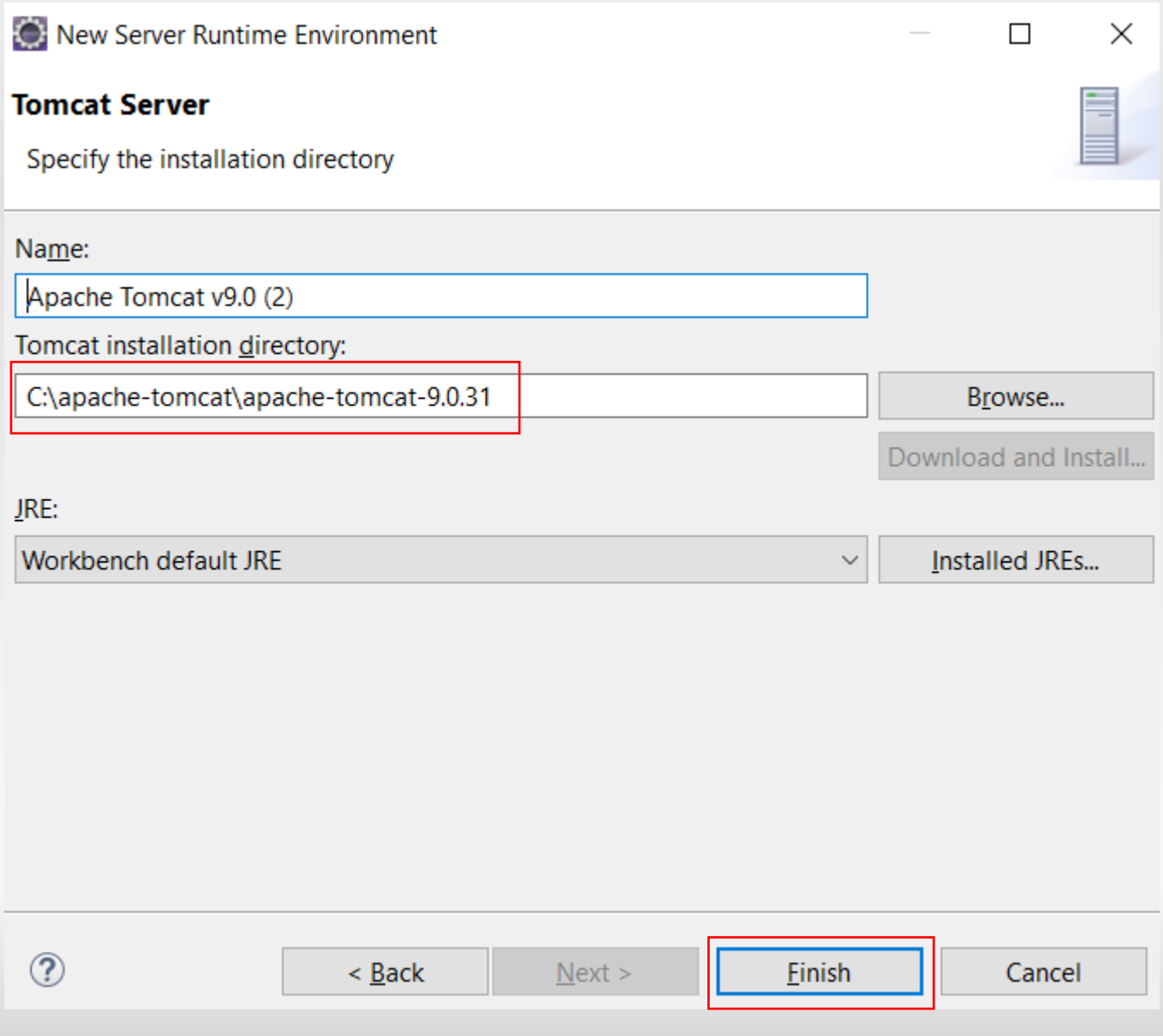
- tar.gz (pgp, sha512)
- zip (pgp, sha512)

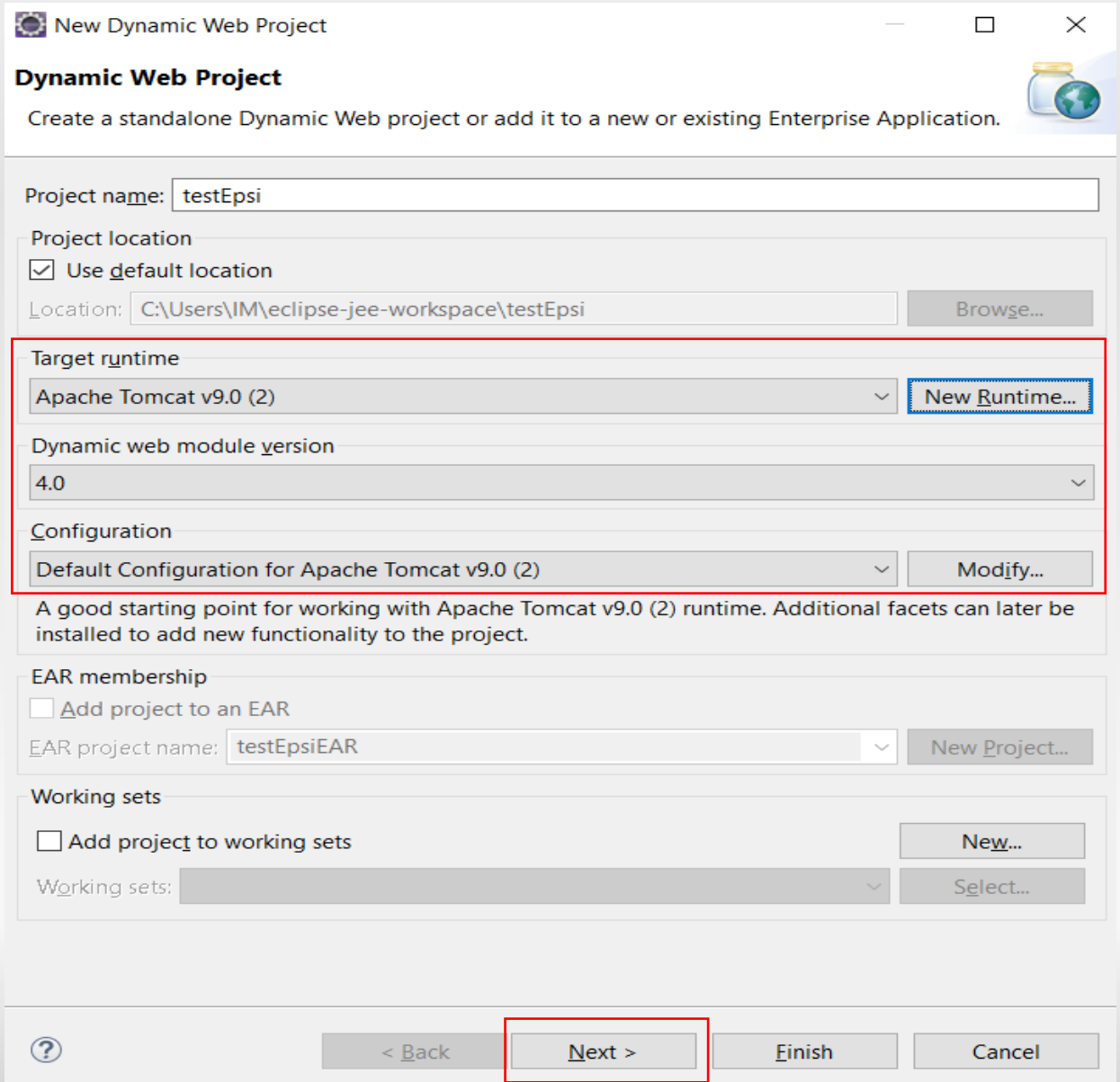
Projet Eclipse

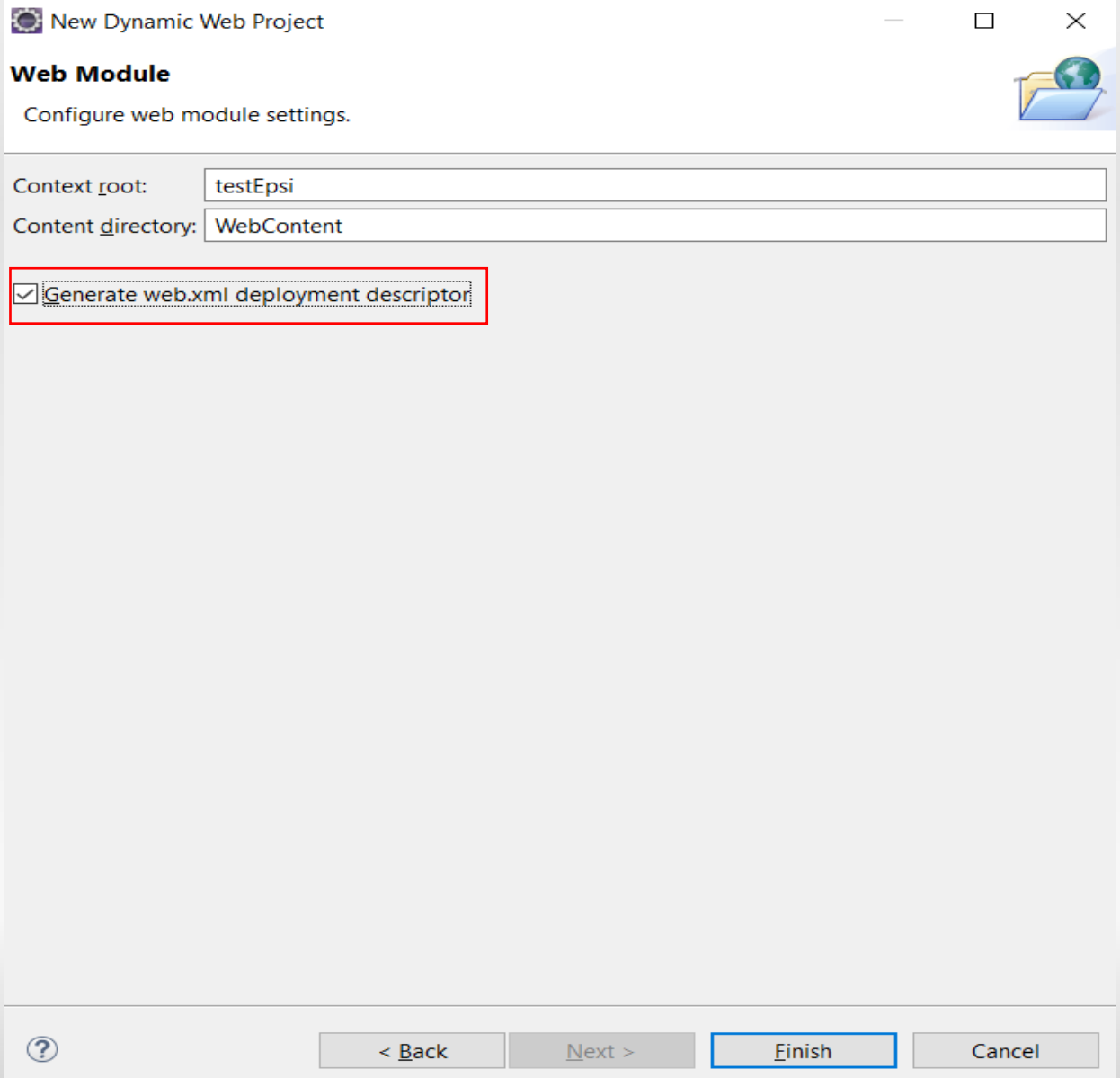
- File->New->Dynamic Web Project



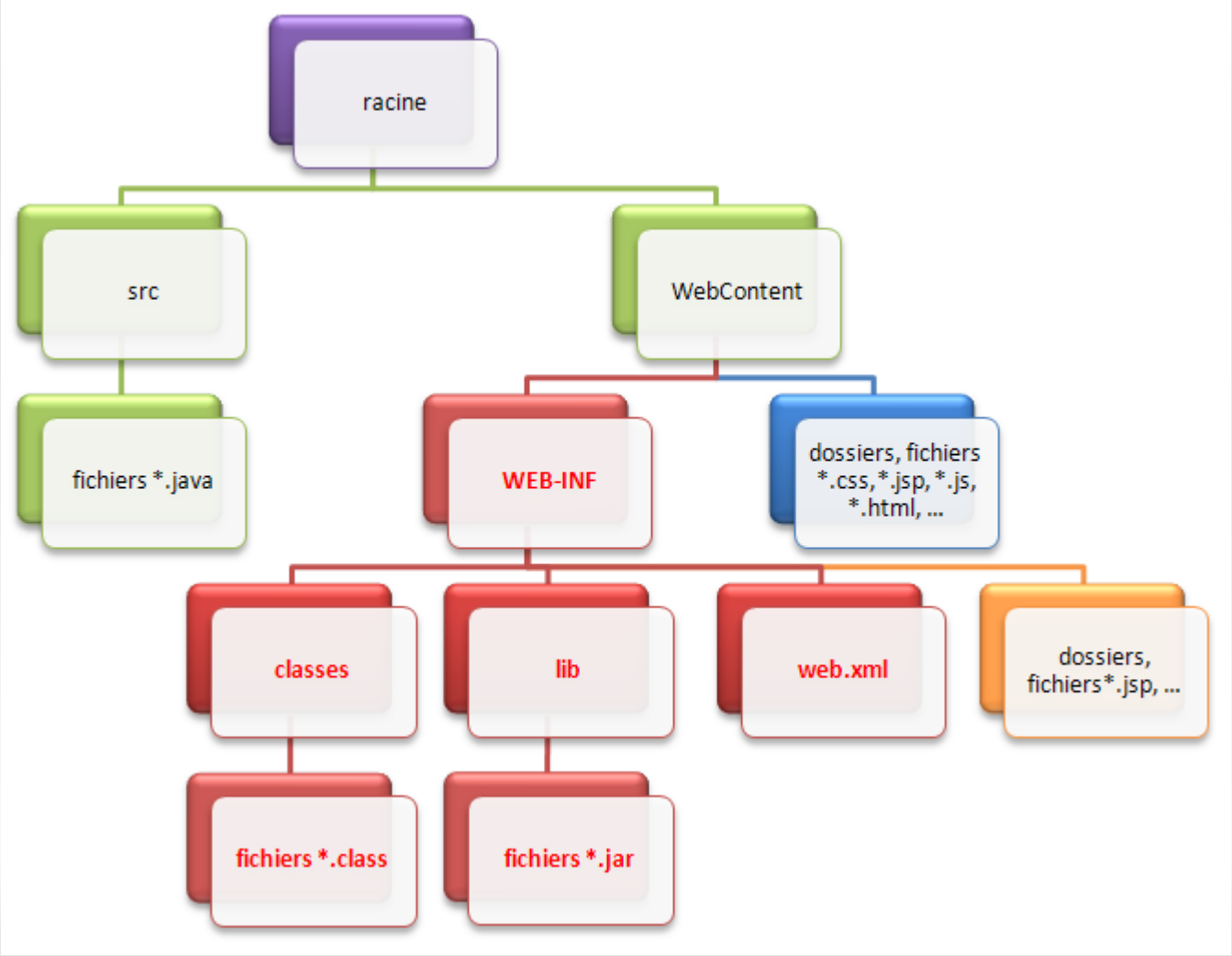

New Server Runtime Environment
New Server Runtime Environment
Define a new server runtime environment
Select the type of runtime environment:
Apache Tomcat v5.5
Apache Tomcat v6.0
Apache Tomcat v7.0
Apache Tomcat v8.0
Apache Tomcat v8.5
Apache Tomcat v9.0
Apache Tomcat v9.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, and 8 Web modules.
Create a new local server
< Back
Next >
Finish
Cancel


New Server Runtime Environment
Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v9.0 (2)
Tomcat installation directory:
C:\apache-tomcat\apache-tomcat-9.0.31
Browse...
Download and Install...
JRE:
Workbench default JRE
Installed JREs...
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: testEpsi
Project location
Use default location
Location: C:\Users\IM\eclipse-jee-workspace\testEpsi
Browse...
Target runtime
Apache Tomcat v9.0 (2)
New Runtime...
Dynamic web module version
4.0
Configuration
Default Configuration for Apache Tomcat v9.0 (2)
Modify...
A good starting point for working with Apache Tomcat v9.0 (2) runtime. Additional facets can later be installed to add new functionality to the project.
EAR membership
Add project to an EAR
EAR project name: testEpsiEAR
New Project...
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Web Module
Configure web module settings.
Context root:
testEpsi
Content directory:
WebContent
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel

Structure sous Eclipse



Créer une page web

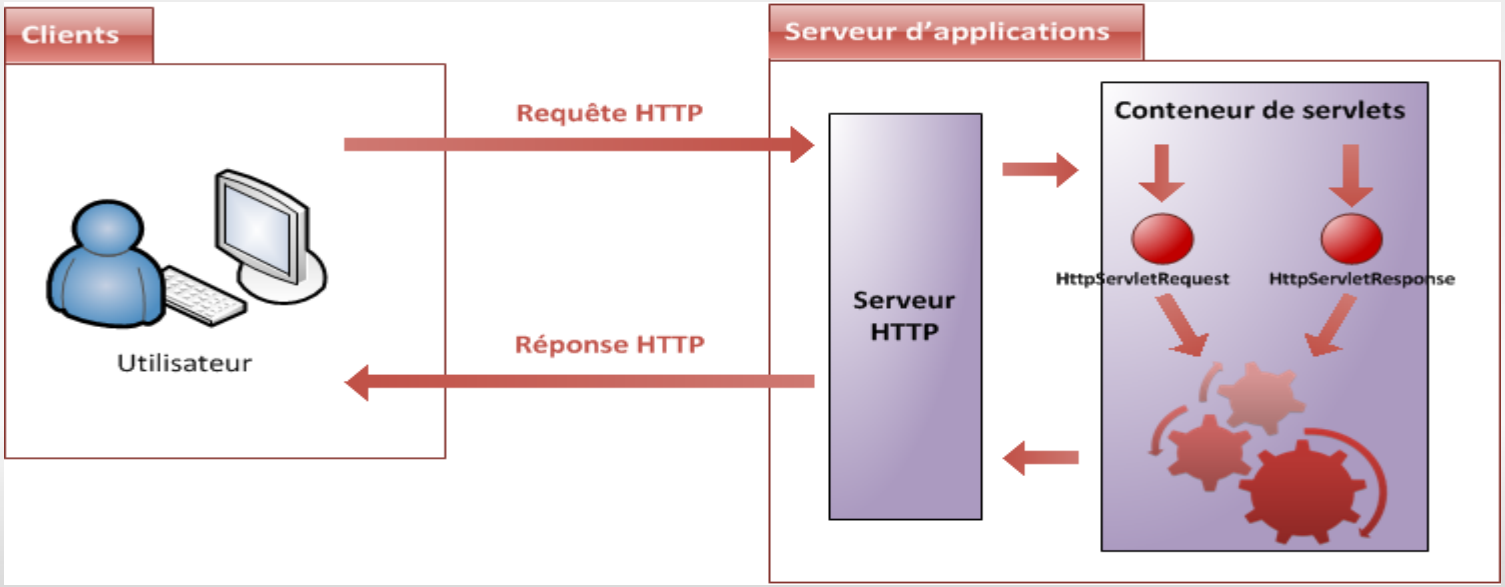
- Clic droit sur WebContent : New->HTML File : index.html
- Ecrire le code suivant dans index.html :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Test Epsi</title>
6      </head>
7      <body>
8          <p>Hello World !</p>
9      </body>
10 </html>
```

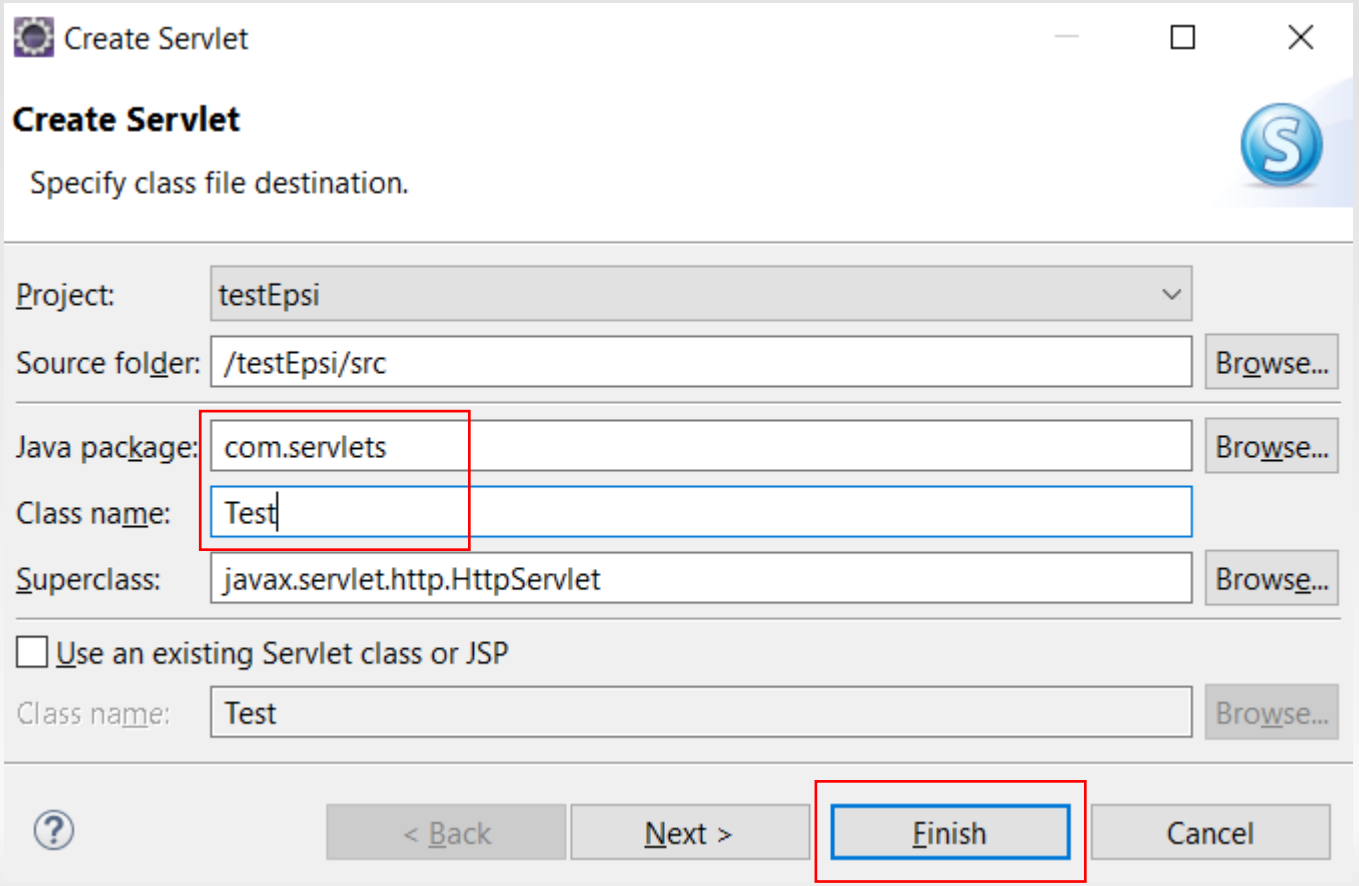
- Exécuter : 

Servlet

- Servlet : Simple classe Java, qui a la particularité de permettre le traitement de requêtes et la personnalisation de réponses.
- Les servlets vont être les points d'entrée de notre application web, c'est par elles que tout va passer.
- **HttpServletRequest** : cet objet contient la requête HTTP, et donne accès à toutes ses informations, telles que les en-têtes (headers) et le corps de la requête.
- **HttpServletReponse** : cet objet initialise la réponse HTTP qui sera renvoyée au client, et permet de la personnaliser.



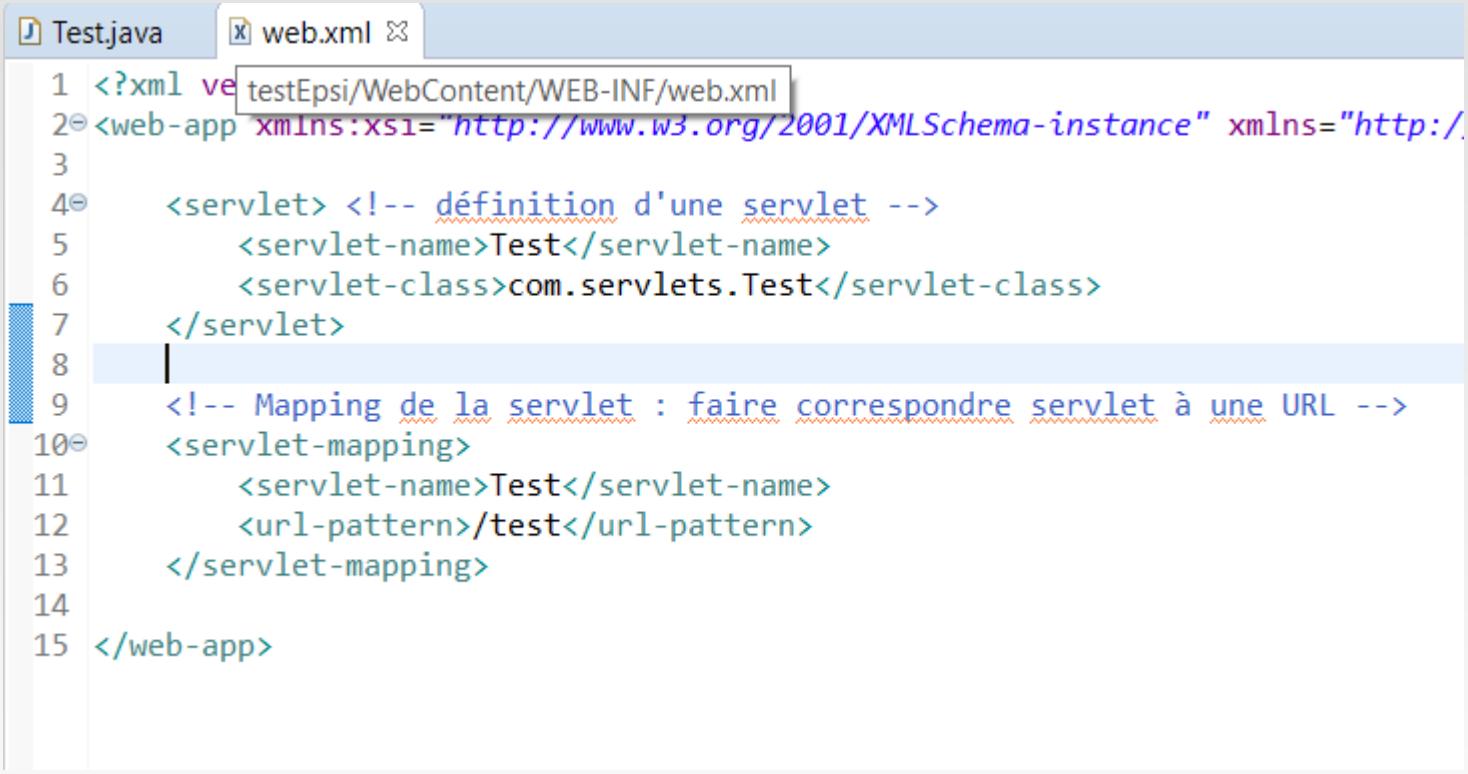
- Clic droit sur Java Resources->src : New->Servlet.



Test.java

```java
package com.servlets;

import java.io.IOException;

/**
 * Servlet implementation class Test
 */
@WebServlet("/Test")
public class Test extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Test() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

- Répertoire /WEB-INF, fichier web.xml : lier notre servlet à une URL.



- Exécuter : 

Envoyer des données

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    //response.getWriter().append("Served at: ").append(request.getContextPath());

    response.setContentType("text/html");
    response.setCharacterEncoding( "UTF-8" );
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<meta charset=\"utf-8\" />");
    out.println("<title>Test</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<p>Hello World !</p>");
    out.println("</body>");
    out.println("</html>");
}
```

Vue : JSP

- Page JSP : balises HTML, balises JSP (appelent du code Java), exécutée côté serveur, XML, Servlet, JavaBeans.
- Servlet difficille
- Séparation code métier (M) et code contrôle (C).
- Un langage dédié, la simplicité d'accès aux objets Java, des mécanismes permettant l'extension du langage.
- Clic droit sur WEB-INF, New->JSP file

Test.java | web.xml | test.jsp | Test Epsi

```jsp
<%@ page pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Test Epsi</title>
    </head>
    <body>
        <p>Hello World ! depuis un fichier JSP</p>
    </body>
</html>
```

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    //response.getWriter().append("Served at: ").append(request.getContextPath());

    this.getServletContext().getRequestDispatcher("/WEB-INF/bonjour.jsp").forward(request, response);
}
```

Les attributs

- Transmettre des variables de la servlet à la JSP :

request.setAttribute(nomAttribut, objet);

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String message = "Ceci est une variable !";
    request.setAttribute( "message", message );

    this.getServletContext().getRequestDispatcher("/WEB-INF/bonjour.jsp").forward(request, response);
}
```

Récupérer dans la JSP : request.getAttribute(nomAttribut);

```
<%@ page pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Test Epsi</title>
    </head>
    <body>
        <p>Hello World ! depuis un fichier JSP</p>

        <p>
            <%
            String attribut = (String) request.getAttribute("test");
            out.println( attribut );
            %>
        </p>

    </body>
</html>
```

<% %> : permet d'inclure du code Java

Récupération des paramètres par le serveur

- URL avec un paramètre : /page.jsp?nom=Jean
- URL avec deux paramètres : /page.jsp?nom=Jean prenom=Philippe

```
protected void doGet(HttpServletRequest request, Http
    String nom = request.getParameter( "nom" );
    String message = "Ceci est une variable !";
    request.setAttribute( "message", message );

    this.getServletContext().getRequestDispatcher("/W
}
```

```
<p>
    <%
    String attribut = (String) request.getAttribute("test");
    out.println( attribut );

    String parametre = request.getParameter( "nom" );
    out.println( parametre );
    %>
</p>
```

JavaBean

- **JavaBean** : composant réutilisable, simple objet Java.
- **Objectifs** :
- **Les propriétés** : un bean est conçu pour être paramétrable. On appelle "propriétés" les champs non publics présents dans un bean.
- **La sérialisation** : permet de sauvegarder l'état d'un bean, et donne ainsi la possibilité de le restaurer par la suite. Ce mécanisme permet une persistance des données.
- **La réutilisation** : n'a pas de lien direct avec la couche de présentation, et peut également être distant de la couche d'accès aux données
- **L'introspection** : permet de connaître le contenu d'un composant (attributs, méthodes et événements) de manière dynamique, sans disposer de son code source.

Structure JavaBean

- doit être une classe publique ;
- doit avoir au moins un constructeur par défaut, public et sans paramètres. Java l'ajoutera de lui-même si aucun constructeur n'est explicité ;
- peut implémenter l'interface Serializable, il devient ainsi persistant et son état peut être sauvegardé ;
- ne doit pas avoir de champs publics ;
- peut définir des propriétés (des champs non publics), qui doivent être accessibles via des méthodes publiques getter et setter, suivant des règles de nommage.

- Dans src :

```java
package com.beans;

public class Auteur {
    private String nom, prenom;
    private boolean actif;


    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
    public boolean isActif() {
        return actif;
    }
    public void setActif(boolean actif) {
        this.actif = actif;
    }
}
```

```java
protected void doGet(HttpServletRequest request, HttpServletResponse re
    String nom = request.getParameter( "nom" );
    String message = "Ceci est une variable !";


    /* Création du bean */
    Auteur premierBean = new Auteur();
    /* Initialisation de ses propriétés */
    premierBean.setNom( "Jean" );
    premierBean.setPrenom( "Patrice" );

    /* Stockage du message et du bean dans l'objet request */
    request.setAttribute( "message", message );
    request.setAttribute( "bean", premierBean );

    this.getServletContext().getRequestDispatcher("/WEB-INF/test.jsp").
}
```

```jsp
<p>
    <%
    String attribut = (String) request.getAttribute("message");
    out.println( attribut );

    String parametre = request.getParameter( "nom" );
    out.println( parametre );
    %>
</p>

<p>
    Récupération du bean :
    <%
    com.beans.Auteur notreBean = (com.beans.Auteur) request.getAttribute("bean");
    out.println( notreBean.getPrenom() );
    out.println( notreBean.getNom() );
    %>
</p>
```

La technologie JSP

- **Commentaires** :

<%-- Ceci est un commentaire JSP. --%>

<!-- Ceci est un simple commentaire HTML. -->

- **Balises de déclaration** :

```
<%! String test = null;
public boolean jeSuisUnZero() {
  return true;
}
%>
```

- **Balises d'expression**:

<% out.println("Bip bip !"); %> équivaut à : <%= "Bip bip !" %>

Les balises JSP

- **Balises de scriptlet** :

```
<form action="/tirage" method="post">
  <%
    for(int i = 1; i < 3; i++){
      out.println("Numéro " + i + ": <select name=\"number"+i+"\">");
      for(int j = 1; j <= 10; j++){
        out.println("<option value=\""+j+"\">"+ j + "</option>");
      }
      out.println("</select><br />");
    }
    %>
    <br />
    <input type="submit" value="Valider" />
</form>
```

Les directives JSP

- importer un package ;
- inclure d'autres pages JSP ;
- inclure des bibliothèques de balises;
- définir des propriétés et informations relatives à une page JSP.
- Elles sont toujours comprises entre les balises <%@ %>
- **Directive taglib** : inclut une bibliothèque personnalisée

```
<%@ taglib uri="maTagLib.tld" prefix="tagExemple" %>
```

- **Directive page** :

```
<%@ page import="java.util.List, java.util.Date"  %>
```

- **Directive include** :

```
<%@ include file="uneAutreJSP.jsp" %>
```

Portée des objets

- Durée de vie.
- **page** (JSP seulement) : les objets dans cette portée sont uniquement accessibles dans la page JSP en question ;
- **requête** : les objets dans cette portée sont uniquement accessibles durant l'existence de la requête en cours ;
- **session** : les objets dans cette portée sont accessibles durant l'existence de la session en cours ;
- **application** : les objets dans cette portée sont accessibles durant toute l'existence de l'application.

Les actions standards

useBean :

```
<%-- L'action suivante récupère un bean de type Auteur et nommé "auteur" dans
la portée requête s'il existe, ou en crée un sinon. --%>
<jsp:useBean id="auteur" class="com.beans.Auteur" scope="request" />

<%-- Elle a le même effet que le code Java suivant : --%>
<%
com.beans.Auteur aut = (com.beans.Auteur) request.getAttribute("bean");
if ( aut == null ){
    aut = new com.beans.Auteur();
    request.setAttribute( "bean", aut );
}
%>
```

getProperty :

<%-- L'action suivante affiche le contenu de la propriété 'prenom' du bean 'auteur' : --%>
<jsp:getProperty name="auteur" property="prenom" />

<%-- Elle a le même effet que le code Java suivant : --%>
<%= auteur.getPrenom() %>

forward :

<%-- Le forwarding vers une page de l'application fonctionne par URL relative : --%>
<jsp:forward page="/page.jsp" />

<%-- Son équivalent en code Java est : --%>
<% request.getRequestDispatcher("/page.jsp").forward(request, response); %>

<%-- Et il est impossible de rediriger vers un site externe comme ci-dessous : --%>
<jsp:forward page="http://www.epsi.fr" />

Expression Language (EL)

- Utilisation optimale des JSP. Plus besoin de Java.
- Syntaxe : ${ expression }
- opérateurs arithmétiques : +, -, *, /, % ;
- opérateurs logiques : &&, ||, ! ;
- opérateurs relationnels : equals() et compareTo()

```
    pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Test des expressions EL</title>
    </head>
    <body>
    <p>
        <!-- Logiques sur des booléens -->
        ${ true && true } <br /> <!-- Affiche true -->
        ${ true && false } <br /> <!-- Affiche false -->
        ${ !true || false } <br /> <!-- Affiche false -->

        <!-- Calculs arithmétiques -->
        ${ 10 / 4 } <br /> <!-- Affiche 2.5 -->
        ${ 10 mod 4 } <br /> <!-- Affiche le reste de la division entière, soit 2 -->
        ${ 10 % 4 } <br /> <!-- Affiche le reste de la division entière, soit 2 -->
        ${ 6 * 7 } <br /> <!-- Affiche 42 -->
        ${ 63 - 8 } <br /> <!-- Affiche 55 -->
        ${ 12 / -8 } <br /> <!-- Affiche -1.5 -->
        ${ 7 / 0 } <br /> <!-- Affiche Infinity -->

        <!-- Compare les caractères 'a' et 'b'. Le caractère 'a' étant bien situé avant le caractère 'b' dans l'alphabet ASCII,
        cette EL affiche true. -->
        ${ 'a' < 'b' } <br />

        <!-- Compare les chaînes 'hip' et 'hit'. Puisque 'p' < 't', cette EL affiche false. -->
        ${ 'hip' gt 'hit' } <br />

        <!-- Compare les caractères 'a' et 'b', puis les chaînes 'hip' et 'hit'. Puisque le premier test renvoie true et le second
        false, le résultat est false. -->
        ${ 'a' < 'b' && 'hip' gt 'hit' } <br />

        <!-- Compare le résultat d'un calcul à une valeur fixe. Ici, 6 x 7 vaut 42 et non pas 48, le résultat est false. -->
        ${ 6 * 7 == 48 } <br />
    </p>
    </body>
</html>
```

```
<!-- Vérifications si vide ou null -->
${ empty 'test' } <!-- La chaîne testée n'est pas vide, le résultat est false -->
${ empty '' } <!-- La chaîne testée est vide, le résultat est true -->
${ !empty '' } <!-- La chaîne testée est vide, le résultat est false -->

<!-- Conditions ternaires -->
${ true ? 'vrai' : 'faux' } <!-- Le booléen testé vaut true, vrai est affiché -->
${ 'a' > 'b' ? 'oui'  : 'non' } <!-- Le résultat de la comparaison vaut false, non est affiché -->
${ empty 'test' ? 'vide' : 'non  vide'  } <!-- La chaîne testée n'est pas vide, non vide est affiché -->

<!-- La ligne suivante : -->
<p>12 est inférieur à 8 : ${ 12 lt 8 }.</p>

<!-- Sera rendue ainsi après interprétation de l'expression, 12 n'étant pas inférieur à 8 : -->
<p>12 est inférieur à 8 : false.</p>
```

- ${ bean.propriete }
- ${ bean.getPrenom() }

```
<p>
<!-- Initialisation d'un bean de type Auteur avec une action standard, pour l'exemple : -->
<jsp:useBean id="auteur" class="com.beans.Auteur" />
<!-- Initialisation de sa propriété 'prénom' : -->
<jsp:setProperty name="auteur" property="prenom" value="Patrice"/>
<!-- Et affichage de sa valeur : -->
${ auteur.prenom }
</p>
```

```
<!-- Comparaison d'égalité entre la propriété prenom et la chaîne "Jean-Paul" -->
${ auteur.prenom == "Jean-Paul" }

<!-- Vérification si la propriété prenom est vide ou nulle -->
${ empty auteur.prenom }

${ !empty auteur.prenom ? auteur.prenom : "Veuillez préciser un prénom" }
```

Listes

```
<p>
<%
/* Création d'une liste de légumes et insertion de quatre éléments */
java.util.List<String> legumes = new java.util.ArrayList<String>();
legumes.add( "poireau" );
legumes.add( "haricot" );
legumes.add( "carotte");
legumes.add( "pomme de terre" );
request.setAttribute( "legumes" , legumes );
%>

<!-- Les quatre syntaxes suivantes retournent le deuxième élément de la liste de légumes  -->
${ legumes.get(1) }<br />
${ legumes[1] }<br />
${ legumes['1'] }<br />
${ legumes["1"] }<br />
</p>

<%
/* Création d'un tableau */
String[] animaux = {"chien", "chat", "souris", "cheval"};
request.setAttribute("animaux" , animaux);
%>

<!-- Les trois syntaxes suivantes retournent le troisième élément du tableau  -->
${ animaux[2] }<br />
${ animaux['2'] }<br />
${ animaux["2"] }<br />
```