

## Initialisation de l'environnement de développement

### Paramètres par défaut

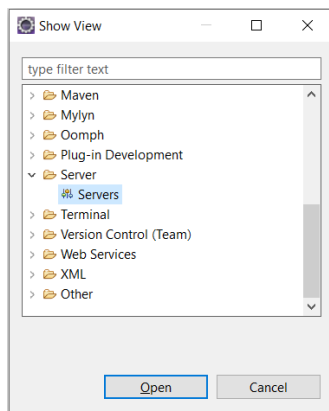
Dans Eclipse, ouvrir les préférences (Window > Preferences) et effectuer les réglages suivants :

- General > Workspace : Encodage = UTF-8
- Java > Installed JREs : utilisation une JDK 12.
  - Dans la liste des JRE, si le nom ne commence par jdk\_12, cliquer sur « Add » pour ajouter une JDK.
  - Sélectionner ensuite le dossier d'installation (ex : C:\Program Files\Java\jdk-12....)
  - Valider
  - Cocher la JDK pour l'utiliser par défaut
- Java > Compiler : Compiler compliance level : 12
- Java > Debug : cocher uniquement la case pour s'arrêter sur les points d'arrêts (breakpoints)
- Editors > Text Editors > Spelling : décocher la verification orthographique

### Installation du serveur d'application

#### Solution 1 - depuis Eclipse

Ouvrir la vue « Servers » dans Eclipse : Window > Show View > Server. Si la vue « Servers » n'est pas disponible, ouvrir la liste complète des vues en cliquant sur « Other ». Si elle n'est toujours pas disponible, c'est que la version d'Eclipse n'est pas bonne.

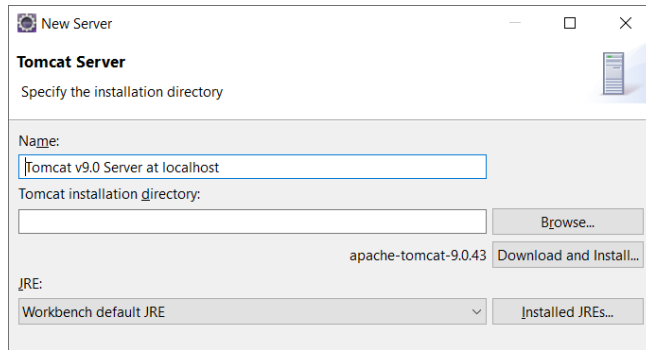


Dans la vue Server, créer un nouveau serveur : clic-droit > New > Server.

Sélectionner « Tomcat v9.0 ». Laisser le host name par défaut et cliquer sur suivant.

Cliquer sur « download and install » et sélectionner le dossier où installer Tomcat (ex : C:\Java\Tomcat).

Cliquer ensuite sur finish.



## Solution 2 - installation manuelle

Télécharger Tomcat version 9 sur le site d'Apache : <https://tomcat.apache.org/download-90.cgi>

Prendre la dernière version zip et l'extraire dans un dossier (ex : C:\Java\Tomcat\apache-tomcat-9.X.XX).

Reprendre ensuite la solution 1, mais sans utiliser le bouton « Download and install » à la dernière étape, et en spécifiant le chemin vers le tomcat téléchargé manuellement.

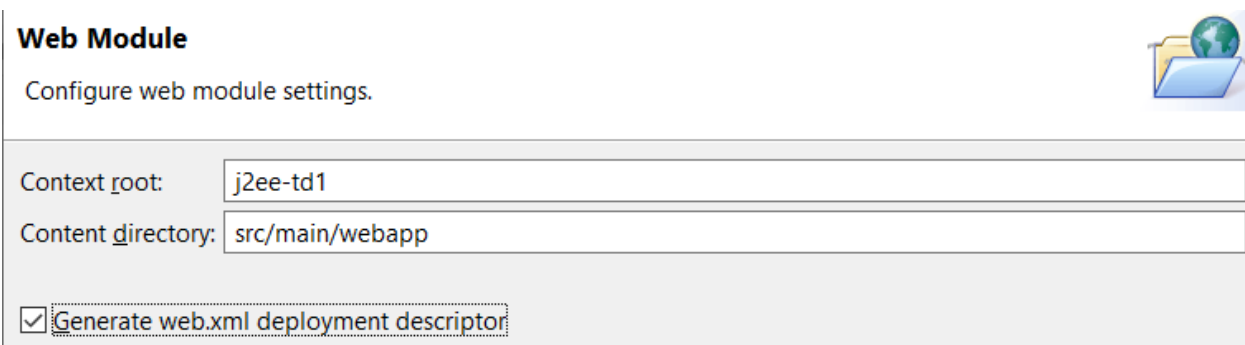
## Premier projet Web

### Création d'un projet Web

Dans Eclipse, aller dans la vue « Project Explorer », faire clic-droit : New : Project

Sélectionner le type de projet : « Dynamic Web Project »

Entrer un nom du projet : *j2ee-td3*, utiliser le runtime défini précédemment, et cliquer sur Next. A la dernière étape, cocher la case pour générer le fichier « web.xml ».



### Création d'un fichier index statique

Dans le dossier *webapp* (src/main/webapp), créer le fichier *index.html* suivant :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
```

```

        <title>J2EE - TD3</title>
    </head>
    <body>
        <p>Page d'accueil du TD 3 de la formation</p>
    </body>
</html>

```

## Première exécution

Ajouter le projet au serveur d'application : dans la vue « Servers », cliquer sur le serveur créé précédemment, puis clic-droit > Add and Remove. Sélectionner le projet, cliquer sur « Add », puis « Finish ».

Lancer le serveur : clic-droit > Start.

Ouvrir un navigateur à l'adresse suivante : <http://localhost:8080/j2ee-td3/>

La page statique définie précédemment doit s'afficher.

## Servlet

### Création d'une Servlet

Créer une classe IndexServlet dans le package *com.j2ee.td3* :

```

@WebServlet("/indexservlet")
public class IndexServlet extends HttpServlet {

}

```

Dans cette classe, ajouter une méthode « doGet »

```

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>TD3 - Accueil Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<p>TD3 - Page d'accueil - mode Servlet</p>");
        out.println("</body>");
        out.println("</html>");
    }

```

Relancer le serveur et ouvrir le navigateur à l'adresse : <http://localhost:8080/j2ee-td3/indexservlet>

Vérifier que la servlet est bien exécutée.

## Modification

Modifier le texte affiché par la Servlet sur la page /index (dans la méthode *doGet*) : la console indique le rechargement.

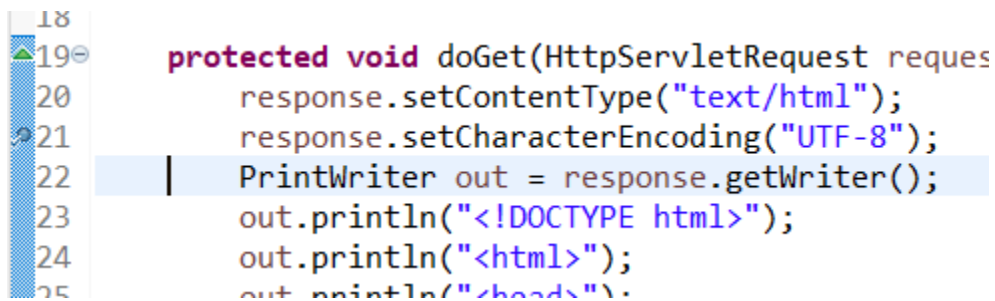
INFO: Reloading Context with name [/j2ee-td3] is completed

Vérifier que la page affichée prend bien en compte les modifications en rafraichissant le navigateur.


## Débogage

Relancer le serveur en mode debug et ajouter un point d'arrêt au début de la méthode « *doGet* ».

Pour ajouter (ou enlever) un point d'arrêt dans Eclipse, il suffit de faire un double-clic à gauche du numéro de la ligne où s'arrêter.



```
18  
19  
20     response.setContentType("text/html");  
21     response.setCharacterEncoding("UTF-8");  
22     | PrintWriter out = response.getWriter();  
23     out.println("<!DOCTYPE html>");  
24     out.println("<html>");  
25     out.println("<head>");
```

Rafraichir la page dans le navigateur : Eclipse doit proposer l'ouverture de la perspective « Debug ». Celle-ci permet de regarder le contenu des variables (vue Eclipse : « Variables »), voir la pile d'appel (vue « Debug »), faire du pas à pas (touches F5 et F6, ou avec les boutons du bandeau supérieur ) ou de continuer l'exécution (F8, ou bouton « play »).



Terminer le débogage avec la touche F8.

## Création d'une Servlet qui affiche une JSP

Créer une nouvelle Servlet Java :

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        getServletContext().getRequestDispatcher("/WEB-INF/helloworld.jsp").forward(request, response);
    }
}
```

Créer un fichier JSP dans le dossier « WEB-INF » (src/main/webapp/WEB-INF) : helloworld.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>TD3 - Hello World</title>
</head>
<body>
    <p>Hello World</p>
</body>
</html>
```

Redémarrer le serveur et ouvrir le navigateur à l'adresse : <http://localhost:8080/j2ee-td3/helloworld>

Essayer d'ouvrir le navigateur à l'adresse <http://localhost:8080/j2ee-td3/WEB-INF/helloworld.jsp> : on obtient une erreur 404. En effet, le dossier « WEB-INF » est protégé : les fichiers JSP ne sont pas accessibles directement.

## Exercices

### Création d'une Servlet « Hello World » avec paramètre

Créer une Servlet « HelloServlet ». Cette servlet doit renvoyer (en HTML) le texte « Hello [user] » où [user] est passé en paramètre à la servlet : `request.getParameter("user")`. Cet affichage doit se faire avec une JSP (on évite d'écrire le code HTML dans la classe Java)

Dans la méthode *doGet* de la Servlet, ajouter avant le chargement de la JSP :

```
// Récupération du paramètre "user"
String user = request.getParameter("user");

// Transmission à la JSP
request.setAttribute("userToGreet", user);
```

Dans la JSP, ajouter ensuite un scriptlet pour récupérer l'attribut :

```
<%
String userToGreet = (String) request.getAttribute("userToGreet");
%>
```

Changer ensuite le message affiché pour utiliser cette variable :

```
<p>Hello <%= userToGreet %></p>
```

Ouvrir ensuite le navigateur en ajoutant le paramètre « user » dans l'URL ( ?user=VALEUR). Exemple :

<http://localhost:8080/j2ee-td3/hello?user=there...%20General%20Kenobi>

Vérifier que le paramètre est bien récupéré et affiché.

## Utilisation de la session

On va créer une mini-application permettant à un utilisateur d'enregistrer des notes.

On va avoir besoin de deux servlets :

- La première servlet va être appelée pour ajouter une note : <http://localhost:8080/j2ee-td3/note/add>.  
Elle va récupérer le paramètre passé dans l'URL, l'ajouter ou initialiser une liste de notes dans la session, et rediriger vers la page *notes*
- La deuxième servlet va être appelée pour afficher les notes : <http://localhost:8080/j2ee-td3/notes>  
Elle va lire la liste des notes contenue dans la session. Elle contient également un formulaire avec un champ texte pour ajouter une note (le nom de ce champ doit correspondre au paramètre déclaré dans la première servlet)

## Notes

- On utilise les méthodes :
  - `HttpSession session = request.getSession(true)` pour récupérer la session de l'utilisateur ou l'initialiser si elle ne l'était pas
  - `session.getAttribute("notes")` pour lire l'attribut « notes », et la méthode `session.setAttribute("notes", notes)` pour mettre à jour celui-ci.

- `resp.sendRedirect(req.getContextPath() + "/redirected");` pour rediriger vers la page `/redirected`

## Lecture et écriture de fichiers

Créer une application permettant d'ajouter et de supprimer des fichiers dans un dossier sur le serveur (Ex : D:\Temp\j2ee-td3)

On a besoin de deux servlets :

- Une première permet d'ajouter un fichier dans le dossier : `POST /files/add`
- Une seconde permet de récupérer un fichier du dossier `GET /files/get?fileName=xyz`

On a également besoin d'un formulaire permettant d'ajouter un fichier (on peut également utiliser un client http comme postman ou curl pour envoyer des fichiers en POST)

Points importants :

- Si le nom de fichier n'est pas trouvé, on s'attend à avoir une erreur 404
- Si le fichier est bien trouvé, on s'attend à ce que la servlet renvoie bien le bon type MIME pour la réponse http. Pour ça, on peut utiliser :  
`String mimeType = Files.probeContentType(file.toPath());`
- Si on envoie en corps de requête, on utilise directement l'`InputStream` :  
`request.getInputStream()` . On peut ensuite enregistrer avec :

```
Files.copy(stream, file.toPath(), StandardCopyOption.REPLACE_EXISTING);
IOUtils.closeQuietly(stream);
```

- Si on envoie en *multipart*, on utilise on utilise la « part » correspondante au nom du fichier :  
`request.getPart("file");` . Cet objet peut ensuite être utilisé pour récupérer le nom du fichier et son contenu (`InputStream`)  
Il faut également ajouter l'annotation `@MultipartConfig` sur la classe Servlet