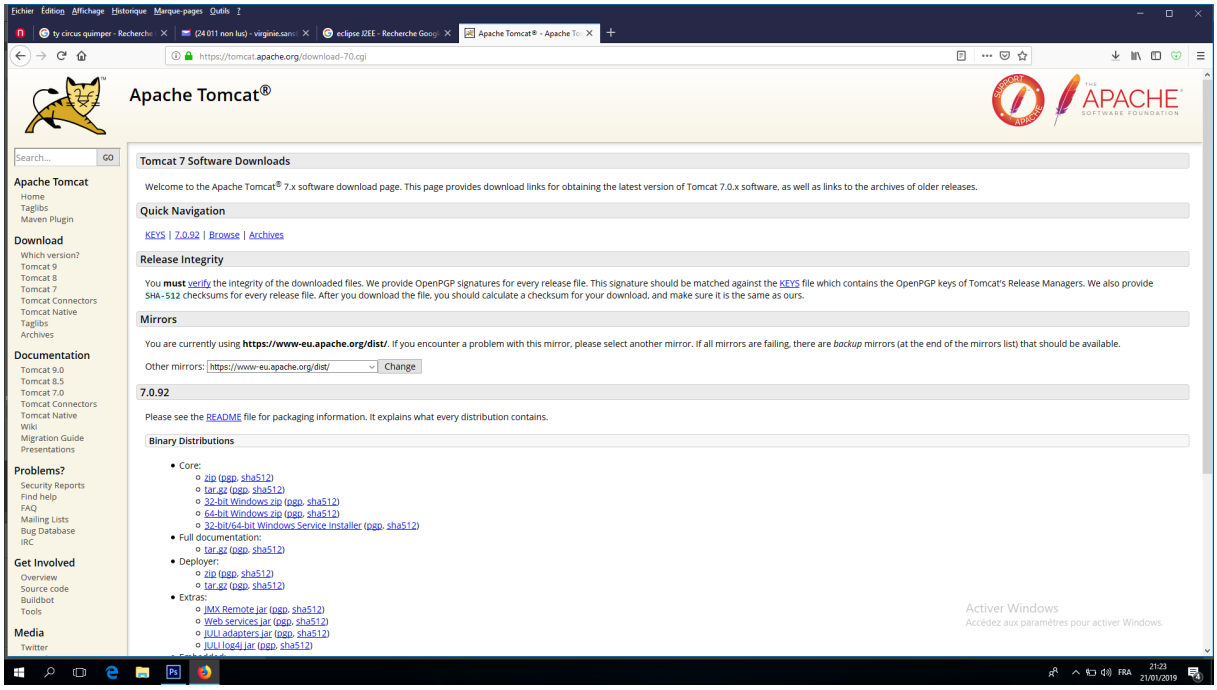
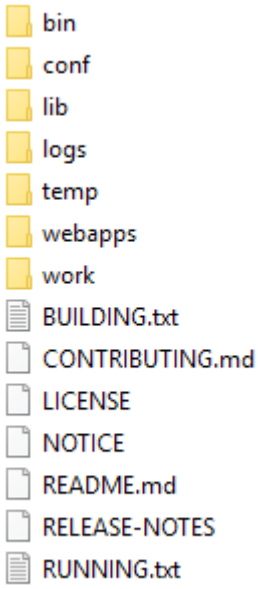
## Etape 1 - Installation de TOMCAT

Allez sur le site de Tomcat et récupérer la version de Tomcat 8

Prenez la version Core correspond à votre OS




Extraire l'archive dans votre système de fichier C:/Programmes par exemple et regardez la structure des répertoires du serveur Tomcat, elle doit ressembler à ceci :




Dans le répertoire bin, vous devez avoir un script startup et un script shutdown qui permettent de démarrer et d'arrêter le serveur. Nous n'allons pas les utiliser tels quels car nous allons passer par eclipse.

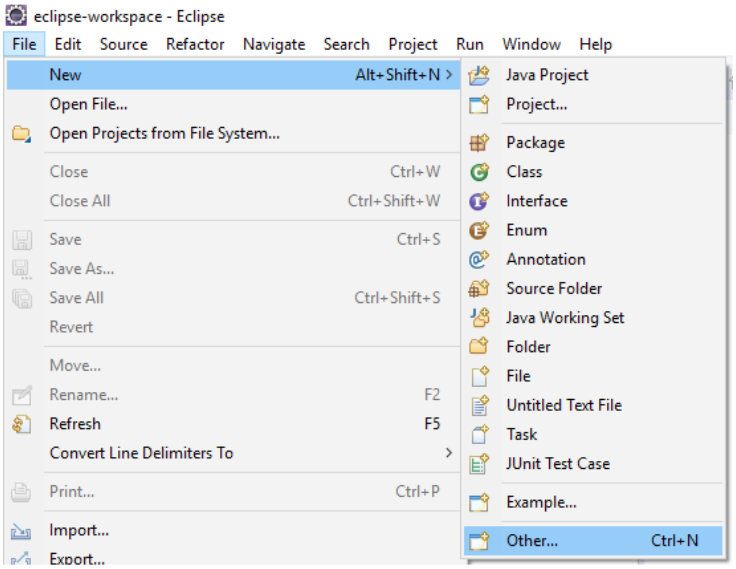
Le répertoire lib doit contenir servlet-api.jar qui est la librairie Java nous permettant de coder des servlets.

Enfin le répertoire webapps contient toutes les web applications qui seront déployées sur notre serveur en production.

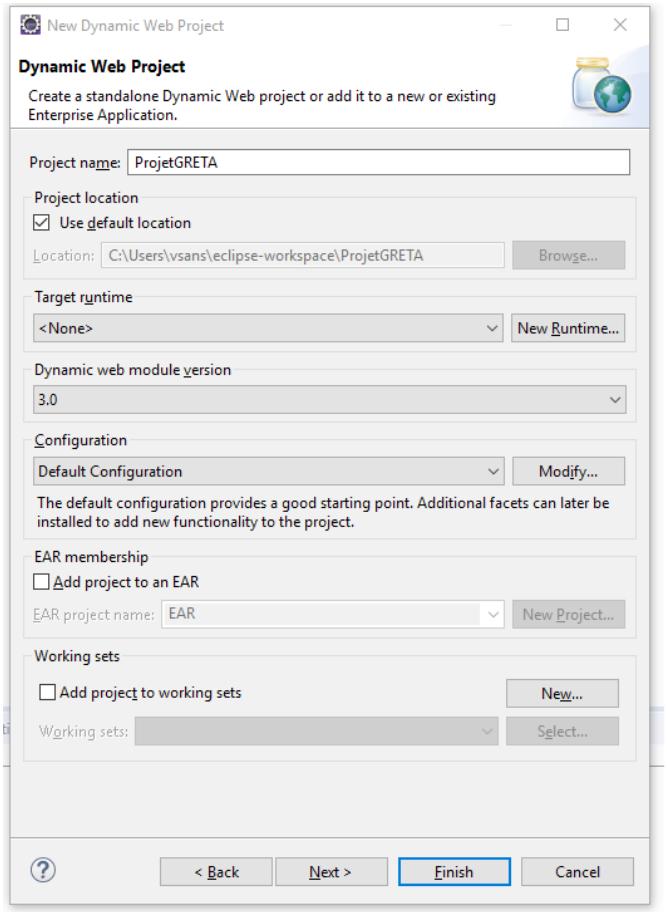
### Etape 2 - Installation d'Eclipse J2EE

### Etape 3 – Votre premier projet

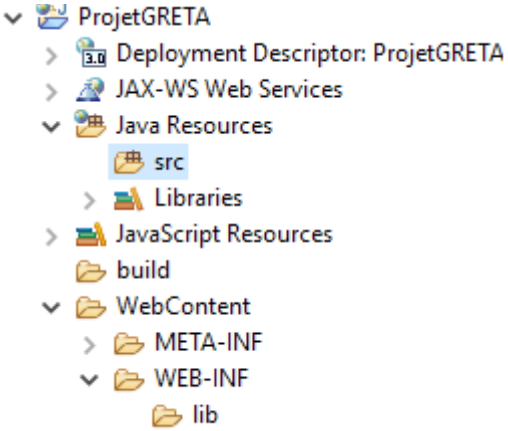
Allez dans File > Other




Puis choisissez Dynamic Web Project et donnez un nom à votre projet : projetGRETA




Eclipse crée alors un projet dans la perspective avec l'arborescence suivante :

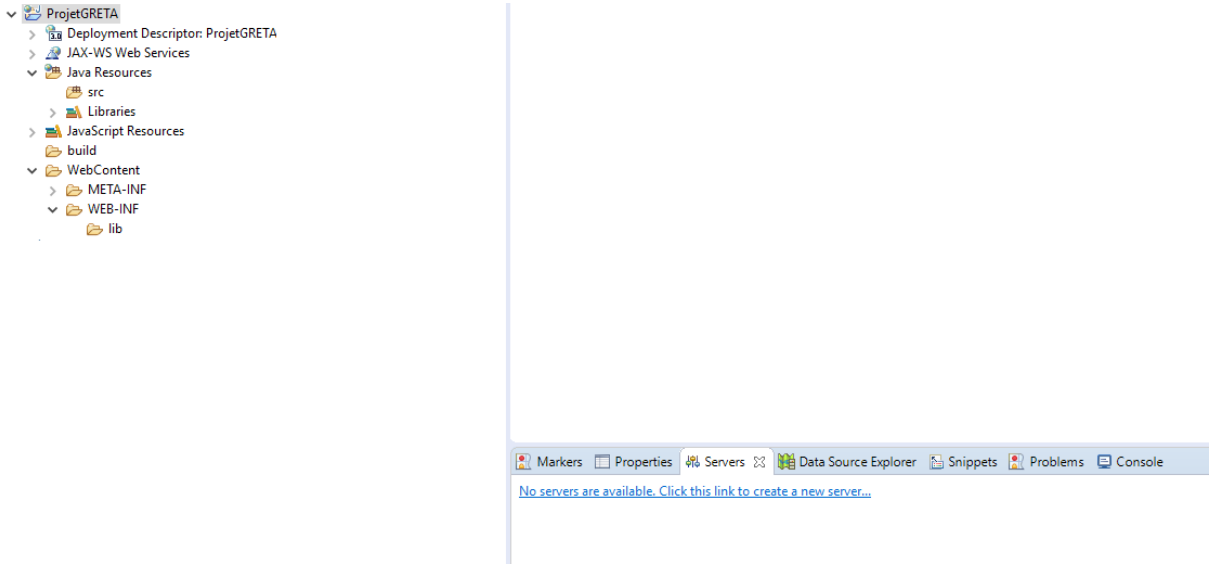



Vous pouvez constater que l'arborescence de ce projet possède un dossier src dans lequel l'ensemble de vos fichiers java seront placés (servlets ou modèles)

Ainsi qu'un répertoire WebContent qui fait office de racine de votre web-app, on y trouve d'ailleurs le répertoire WEB-INF et WEB-INF/lib. Nous créerons le répertoire classes dedans tout à l'heure.

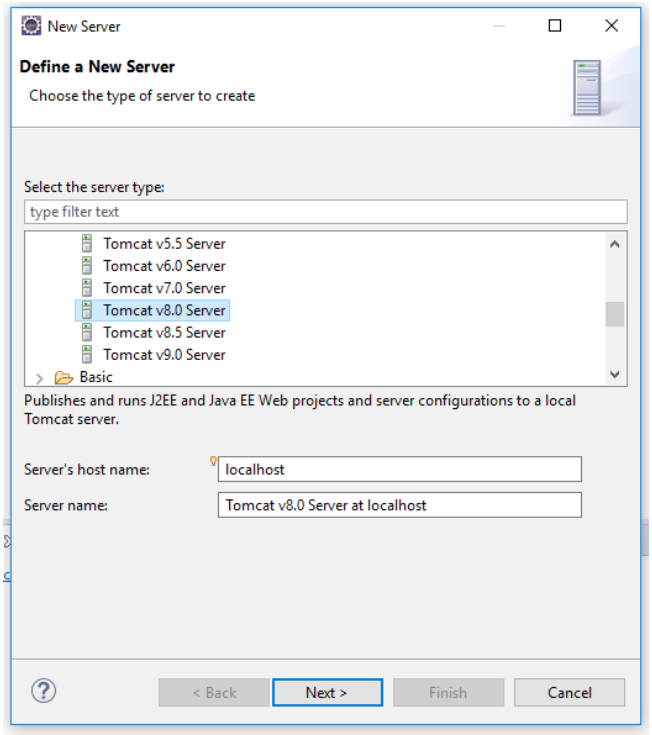
### Etape 3 – Configuration du serveur Tomcat dans Eclipse

Dans la perspective de votre projet, en bas, vous devriez avoir un onglet nommé serveur.

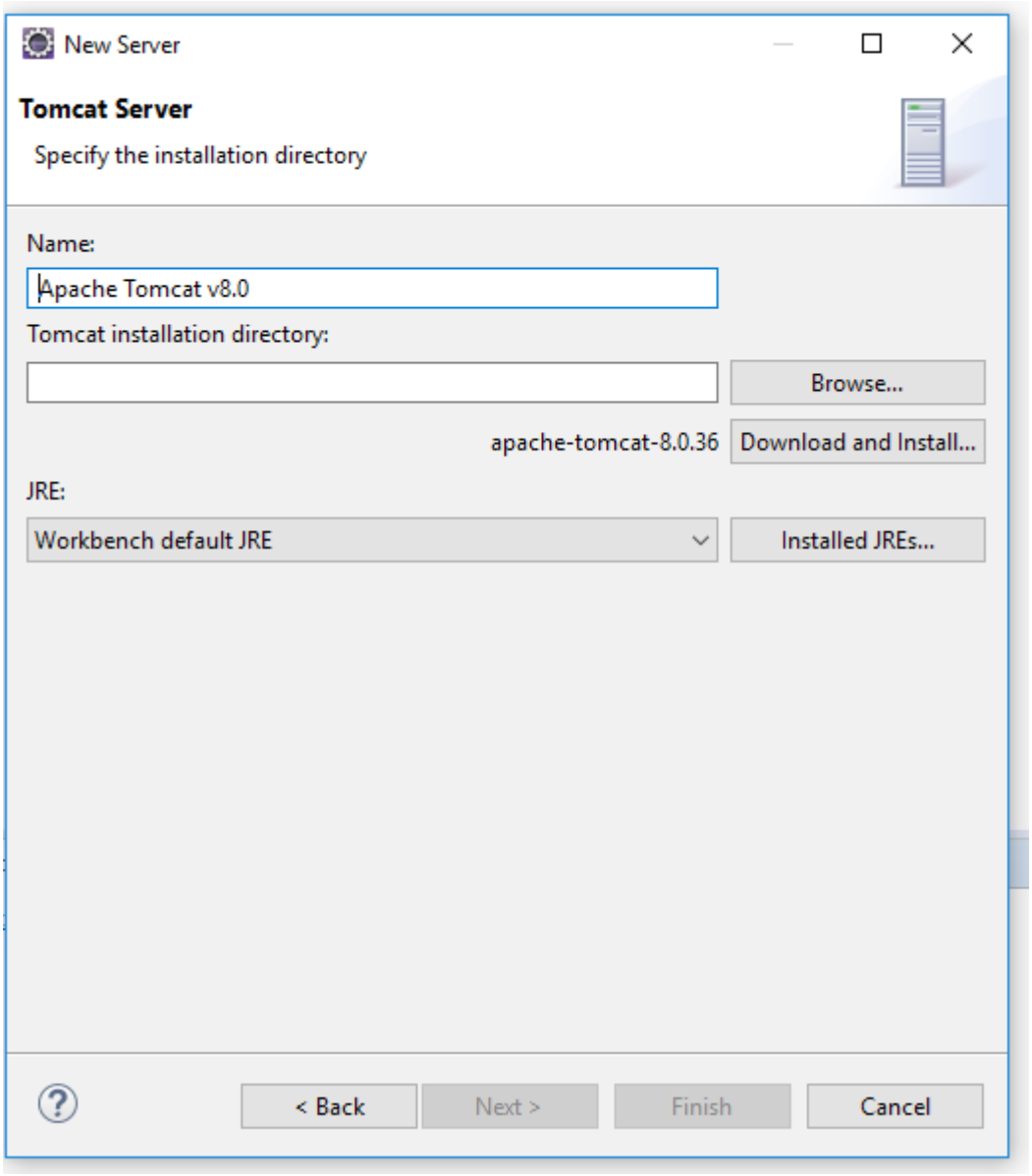



Cliquez sur le lien pour créer un nouveau serveur.

Choisissez d'abord le type de serveur : Apache Tomcat version X.X




Indiquez lui le répertoire d'installation de l'étape 1 (ou téléchargez le serveur tomcat directement depuis cette étape).




Si tout se passe bien, vous pouvez démarrer directement votre serveur avec un clic droit sur le nom du serveur depuis l'onglet serveur, et le stopper également.

Verifiez déjà cette étape avant d'aller plus loin.

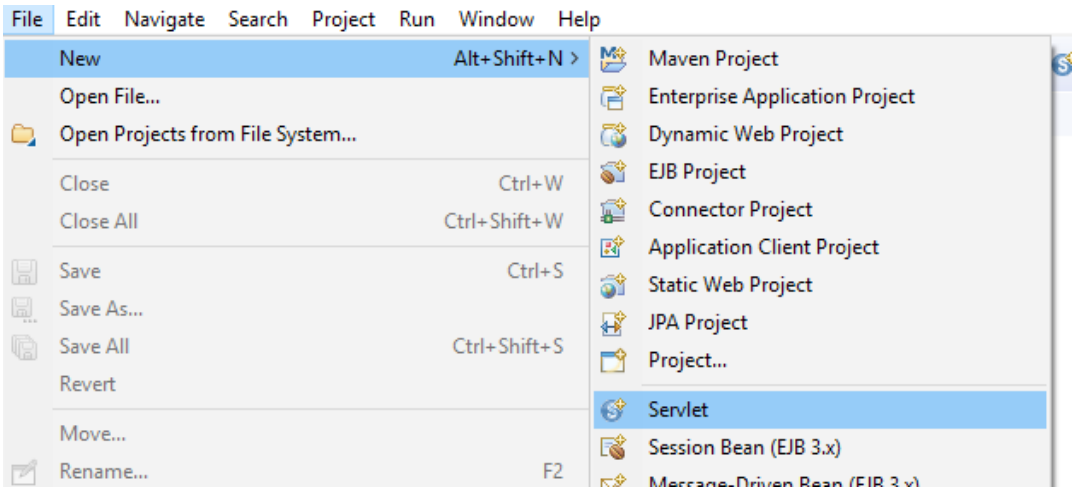
Si il y a un problème de port, double cliquez sur le nom du serveur, il vous ouvre le fichier server.xml en mode graphique, vous pouvez changer les ports (Exemple 8080 en 8081, 8005 en 8006 etc…)

En effet, plusieurs services peuvent utiliser les mêmes ports. Par exemple, easyphp avec son serveur Apache PHP utilise aussi d'habitude le 8080.

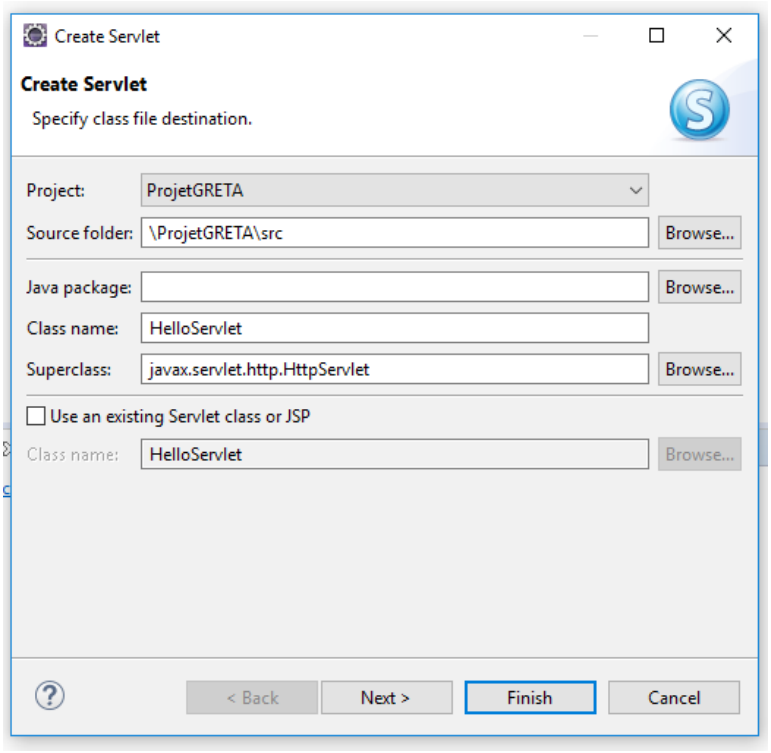
Pensez donc à ne pas avoir 2 serveurs identiques ou différents qui tournent en même temps sur le même port.

**Etape 4 – Votre première servlet**

Dans File, Choissisez New puis Servlet




Donnez un nom à votre servlet et vérifiez qu'il est dans le bon projet.



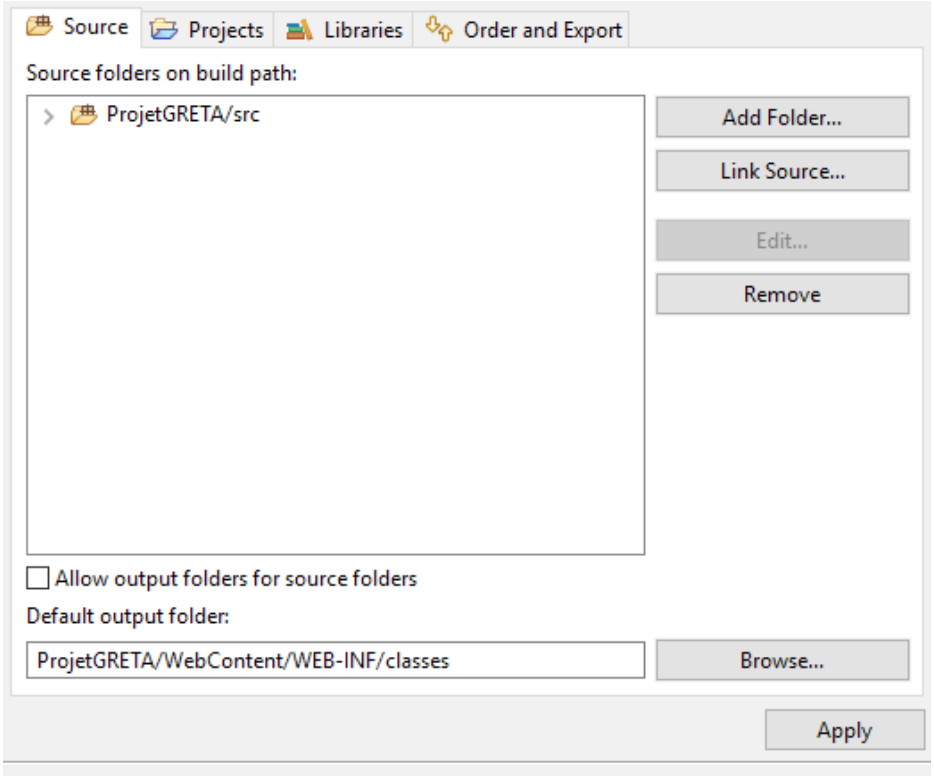

Eclipse créé automatiquement une classe Java qui hérite de HttpServlet avec le squelette de la classe avec les méthodes doGet et doPost.


Vous pourrez constater que les mots clefs concernant les servlets n'est pas reconnu. Il va falloir indiquer à eclipse quelles sont les classes nécessaires en les référencant dans BuildPath> Configure BuildPath > Librairies > Add External jar ( et si vous avez une version récente dans module path) et indiquez l'emplacement du servlet-api.jar qui se trouve dans le répertoire lib de votre répertoire Tomcat.

Le rouge devrait disparaitre…si votre version de JRE est compatible avec votre jar de servlet.

La configuration d'un serveur Tomcat peut parfois être longue et fastidieuse car elle nécessite d'avoir un Tomcat compatible avec votre JRE, compatible également avec votre eclipse et votre jar de servlet…

Pour faire véritablement propre, nous allons faire en sorte que le .class qui sera généré se place au bon endroit dans votre web-app, pour cela, cliquez droit sur le nom de votre projet > Build Path > Configure build path Puis dans l'onglet source, vérifiez que le Default Output Folder soit bien le répertoire classes du WEB-INF



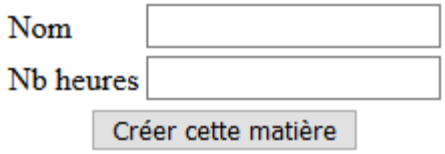

Voilà tout est prêt pour enfin lancer une servlet !!! (*)

Pour lancer cette servlet, faites un clic droit sur le code et faites Run As > Run on Server, vous verrez un navigateur interne s'ouvrir avec le résultat de la servlet en mode GET.

(*Rassurez vous la configuration du serveur et du répertoire classes, et de l'api ne se fait qu'une fois par projet)

### Etape 5 – Passage de valeur dans un formulaire

Créez d'abord une servlet Form.java avec un formulaire en GET qui ressemble à cela :




Vérifiez l'allure de votre page en la runnant sur le serveur. Observez bien l'URL….

Créez ensuite une servlet de nom Resultat.java dans le répertoire src qui affiche les valeurs des Nom et Nb heures de la page Form

Pour cela, modifier l'action du formulaire de Form et créez le code adéquat dans Resultat.java

Par quelle URL, la servlet Resultat est elle appelable ? Comment faire pour modifier l'URL d'appel de cette servlet ?

Il existe 2 techniques qui ne peuvent pas coexister pour une même servlet :

- La rapide avec les annotations

- La plus fastidieuse mais aussi plus complète avec le fichier web.xml

**Etape 6 – Redirection**

Reprenez le code de Form.java et faites la pointer vers une servlet de nom Controlleur1.java

Ecrire une servlet Controlleur1.java qui vérifie sur le nom est bien "Math", si c'est le cas on est redirigée vers une servlet Final.java qui affiche "Matière Math", sinon on est redirigé vers la servlet du début Form.

Faites une première version de cet exercice en utilisant SendRedirect, puis avec Forward… quelle différence faites vous entre les deux ? Regardez bien les URL en haut du navigateur.

En fait, on a maintenant un début de pattern MVC, nous avons le V et le C…

Les vues correspondent à Form et à Final et le controlleur à Controlleur1.

**Etape 7 – Pattern MVC et sessions.**

Pour  implémenter un vrai pattern MVC, il faut avoir des modèles. Ce sont des classes que vous avez déjà implémentées.

Refactorisons un peu notre code avec les étapes suivantes :

Créez 3 packages View, Model, Controller

- Dans View, déplacez les fichiers Form.java et Final.java
- Dans Controlleur, déplacez le fichier Controllleur1.java
- Dans Model, importez les classes faites précédemment en cours

Créez ensuite un Controlleur2.java de façon à ce qu'il crée une Matière m avec les valeurs récupérées du formulaire (Attention au cast et aux valeurs nulles dans le formulaire …gare au NullPointerException) puis stocker le nom de cette matiere dans une variable de session si elle n'est pas nulle. Si le nom est null, on redirige vers la page de formulaire.

La vue View3.java affichera la valeur de cette variable de session.

Runnez sur le serveur et vérifiez le résultat.