Développement de Servlets et JSP avec Eclipse

Sommaire

- 1 Mise en place
  - 1.1 Installation de Galileo
  - 1.2 Association de Galileo avec une installation de Tomcat
  - 1.3 Pilotage des serveurs
- 2 Développement d'une application Web
  - 2.1 Création d'un projet 'Web dynamique'
  - 2.2 Développement d'un Servlet
  - 2.3 Développement d'une JSP
  - 2.4 Notion de projet dépendant
  - 2.5 Exporter l'application

Mise en place

Pour ce cours, la mise en place consiste en l'installation d'un JDK, d'Eclipse Web Tools et de Tomcat.

L'installation d'Eclipse Web Tools peut se faire de deux façons :

- Utilisation du système de Mise à jour d'Eclipse. Après avoir installé Eclipse , il est possible d'utiliser le gestionnaire des mises à jour pour télécharger Eclipse Web Tools et ses pré requis.
- Téléchargement et installation d'un fichier contenant Eclipse, et tous les pré requis pour JEE. C'est la solution que nous utiliserons.

Installation d'Eclipse

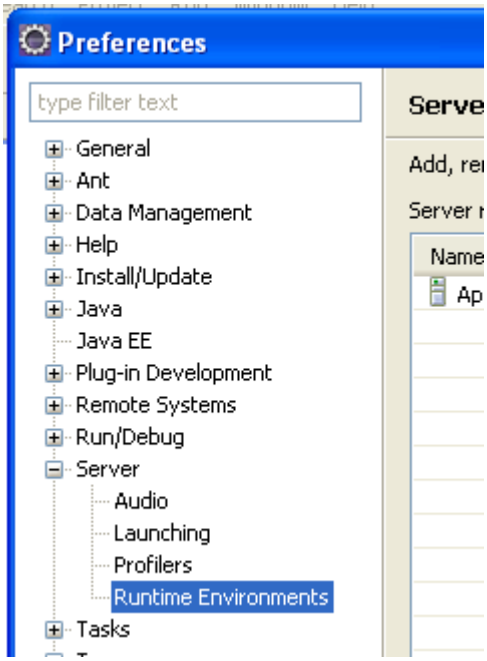
Pour pouvoir faire fonctionner Eclipse, il est nécessaire d'installer un JDK ou un JRE. L'étape suivante consiste à télécharger une version d' Eclipse et tous les fichiers nécessaires au fonctionnement . Voici le lien pour télécharger Eclipse Web Tools 3.5.1 pour Windows :

Télécharger Eclipse 3.5.1 JEE pour Windows

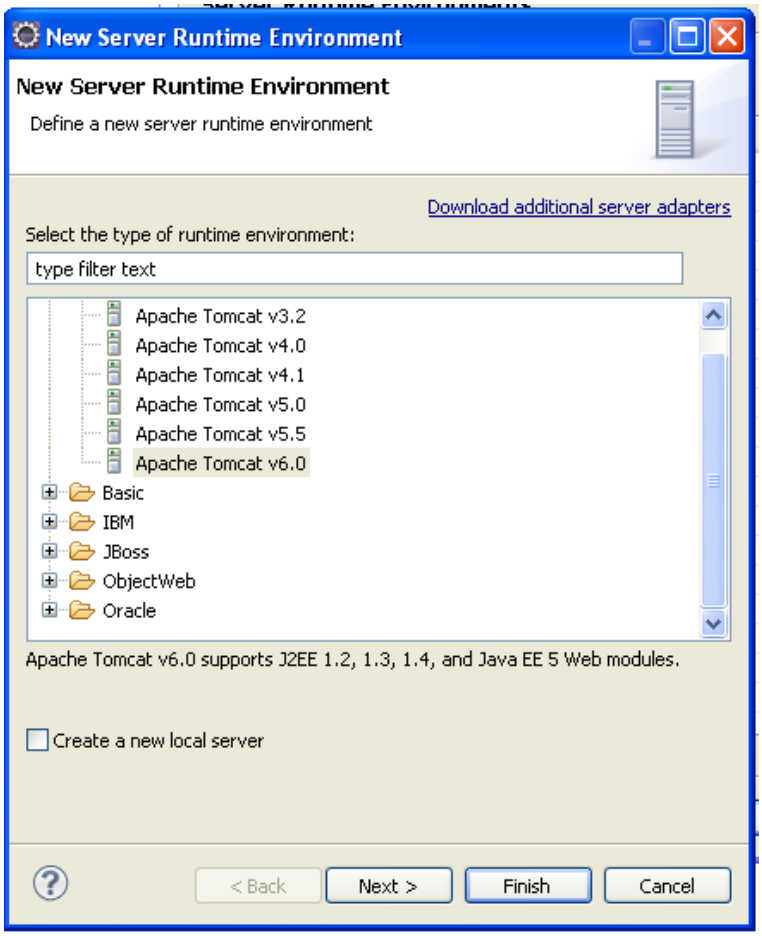
Une fois le téléchargement effectué, l'installation consiste simplement à décompresser le fichier. L'exécution se fera ensuite en lançant le fichier eclipse.exe.

Association d'Eclipse avec une installation de Tomcat

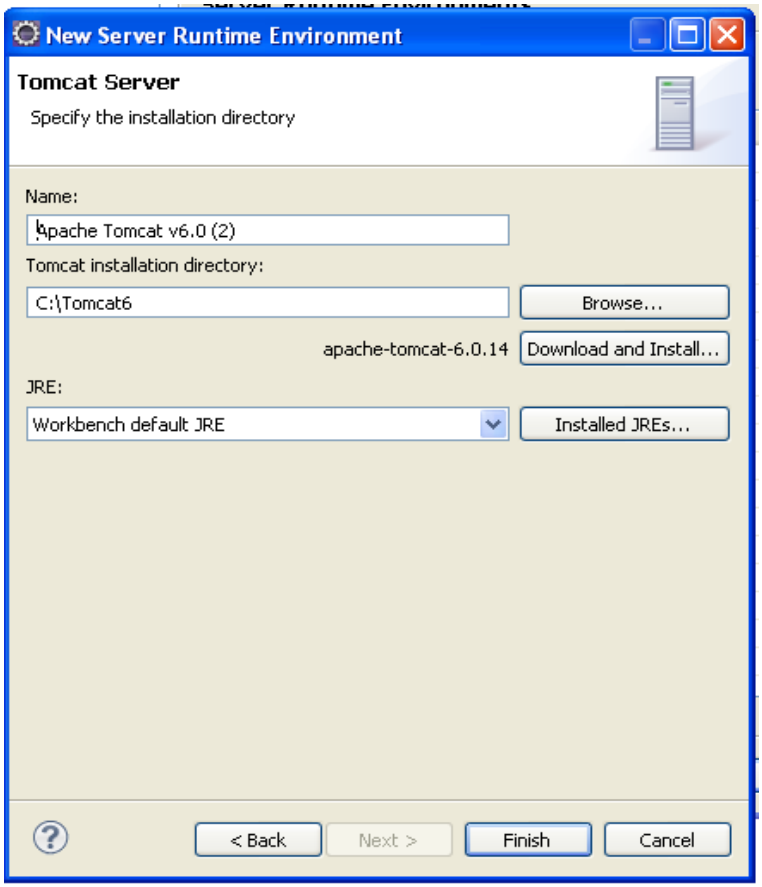
Un point fort c'est de pouvoir piloter les principaux serveurs d'applications. Une étape de base dans l'utilisation de WTP est de déclarer l'emplacement d'installation des serveurs d'applications que WTP devra gérer. Cette déclaration se fait dans les préférences d'Eclipse : **Préférences->Server->Installed Runtimes**.



Le bouton 'Add...' affiche la liste des types de serveurs supportés. Dans le cadre de notre cours nous utiliserons Tomcat 6.0.24 (pour installer Tomcat télécharger le fichier apache-tomcat-6.0.24.exe et l'installer)

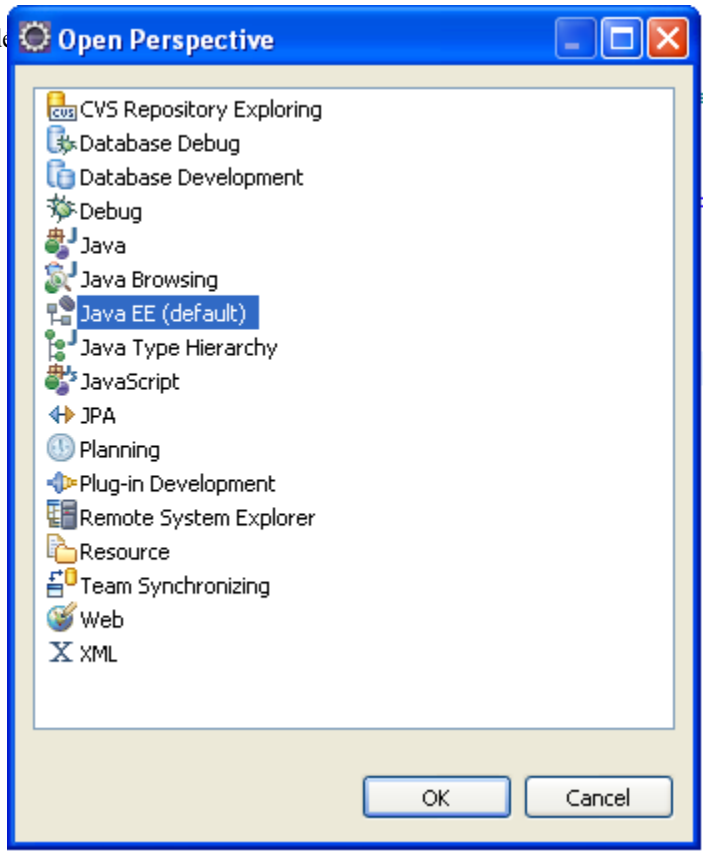


Selon le type de serveur sélectionné diverses informations sont demandées, dans le cas de Tomcat seul le répertoire d'installation doit être indiqué :

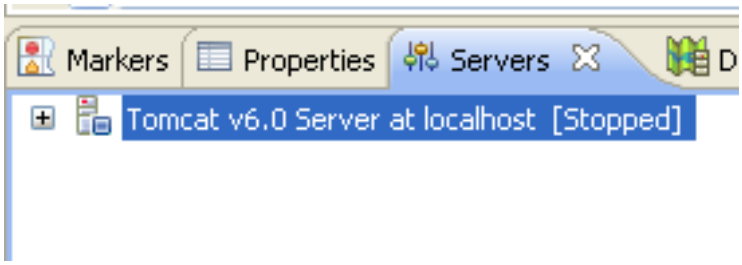


Pilotage des serveurs

Eclipse propose une vue spécifique, nommée **'Server'** pour piloter les serveurs d'applications. Cette vue est présente dans la perspective « Java EE ». Il est naturellement possible de faire afficher cette vue **'Server'** dans une autre perspective via le menu **Windows->Open Perspective->Other...**, puis **Server->Servers.**

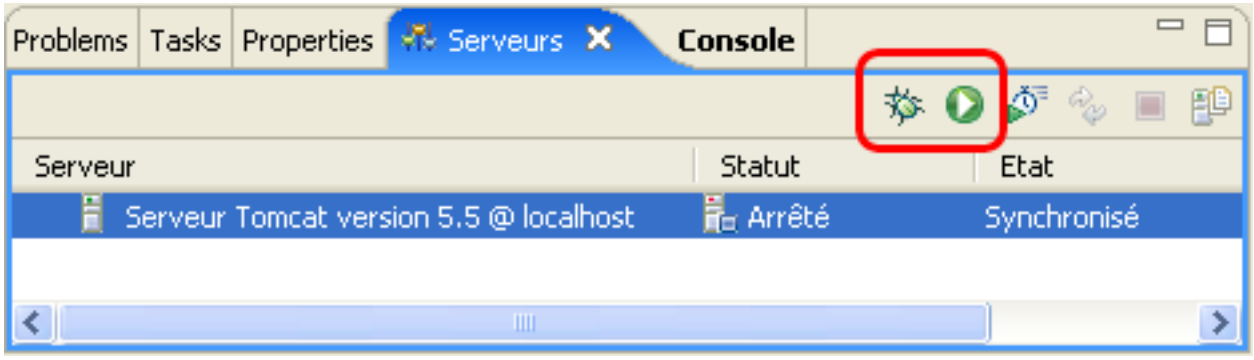


Dans cette vue il faut déclarer les serveurs qui seront pilotés, pour ce faire utiliser le menu contextuel :

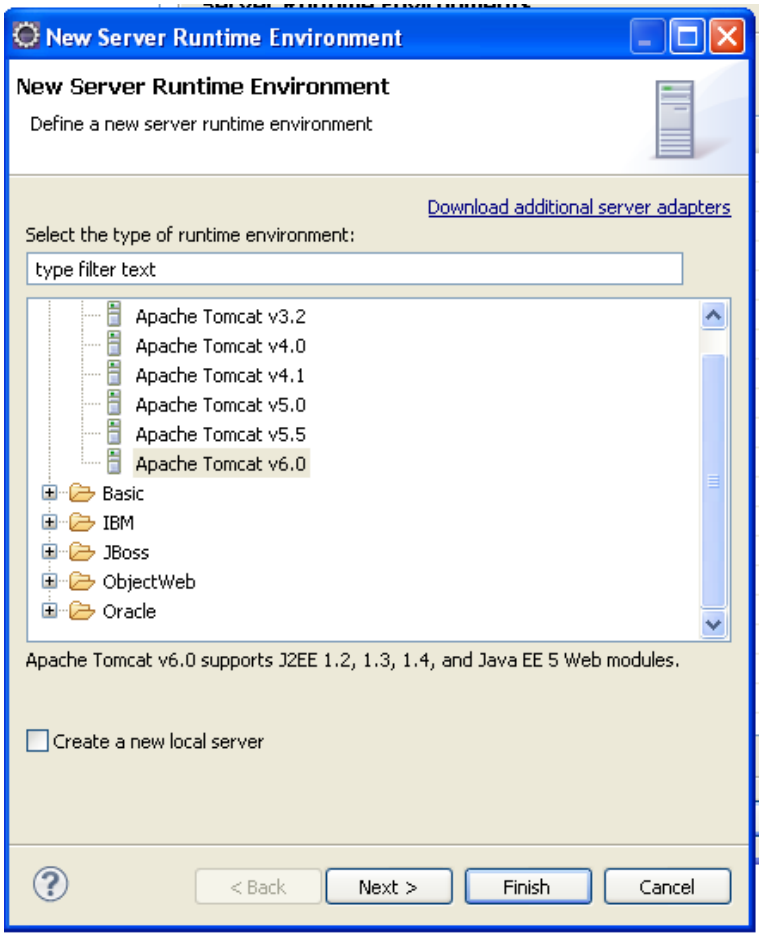


Pour l'ajout d'un serveur, l'information importante est le nom de l'installation de serveur à utiliser (tel que déclaré précédemment dans les préférences) :

La vue 'Servers' permet ensuite de piloter les serveurs déclarés, il est notamment possible de les démarrer en mode exécution ou débogage :



Pour paramétrer un serveur, il suffit de faire un clic droit dans le vue « Servers », sélectionnez le menu « **New->Server** », on obtient le panneau déjà connu ci dessous.

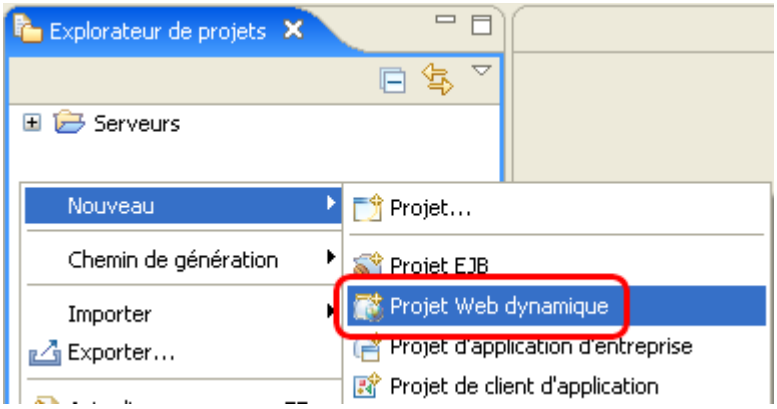


Suivre la procédure déjà décrite ci précédemment sur le paramétrage des préférences serveur.

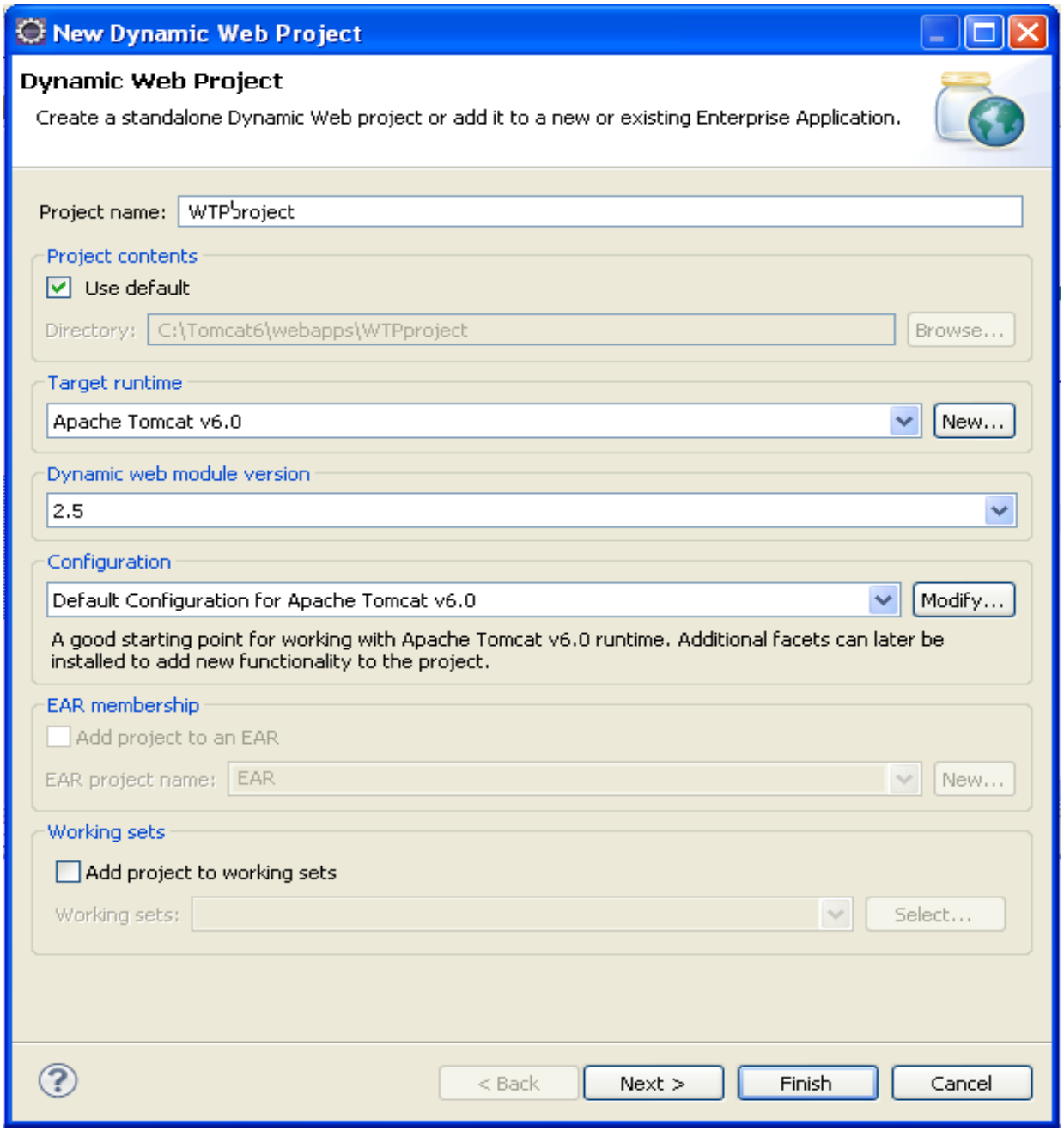
Développement d'une application Web

Création d'un projet Web

Eclipse propose un type de projet particulier pour les applications Web : '**projet Web dynamique**'. Ce type de projet est à la base un projet Java enrichi de diverses fonctionnalités : un chemin de compilation initialisé pour avoir accès aux APIs JEE, une structure conforme au format WAR et des propriétés supplémentaires.
L'ouverture de l'assistant de création de projet Web dynamique peut se faire à partir du menu contextuel de la vue 'Explorateur de projet' dans la perspective Java EE :



La première page de l'assistant demande plusieurs informations, les plus importantes sont :
1 - **Le nom du projet**.
2 - **L'environnement d'exécution cible** : il s'agit du serveur auquel sera associé ce projet, cette information permet à Eclipse d'ajouter les fichiers JAR nécessaires dans le chemin de compilation du projet.
3 - **Appartenance à un EAR** : le projet Web dynamique correspond à un WAR, lors du déploiement la spécification J2EE prévoit qu'un WAR puisse être stocké dans un fichier EAR. Eclipse propose un type de projet particulier pour les EAR : projet d'applications d'entreprise. L'association d'un WAR avec un EAR est facultative. Dans le cadre de ce cours nous ciblons Tomcat qui ne supporte pas le format EAR donc la création d'un projet d'application d'entreprise n'est pas demandé.



La seconde page de l'assistant permet d'ajouter des ensemble de fonctionnalités au projet, ces ensembles sont appelés 'project facets' en anglais. La liste des 'facettes' est extensible, elle dépend des plugins complémentaires qui sont installés.

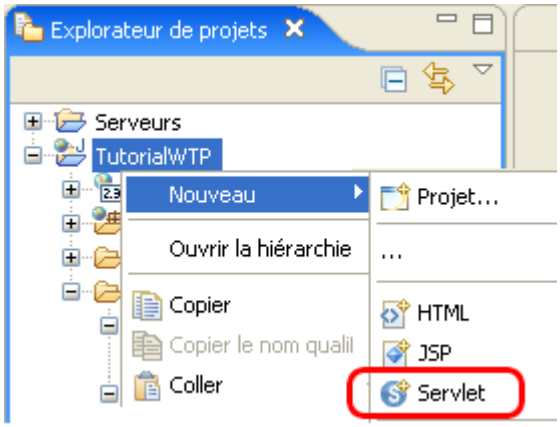
La troisième page de l'assistant permet de renseigner deux informations importantes :

1 - **Le nom de contexte** : conformément à la spécification JEE, chaque application à son propre nom de contexte qui apparaît dans les URL permettant d'accéder à l'application. Par défaut, Eclipse propose le même nom que le projet.

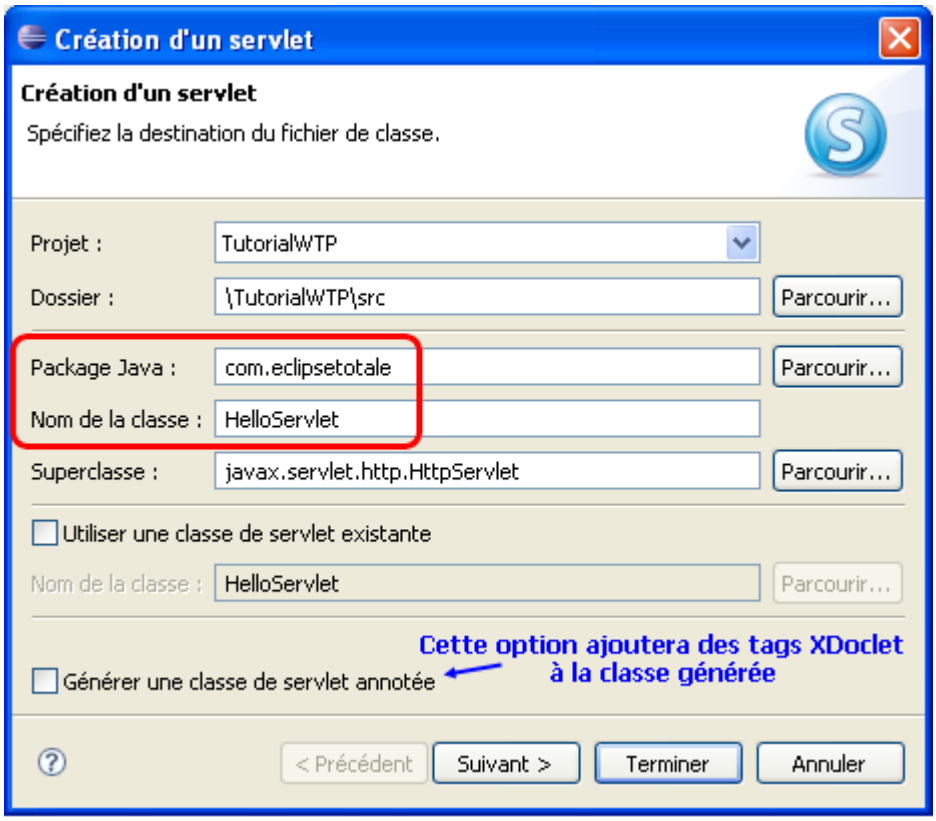
2 - **Le nom du 'répertoire de contenu'** : cette information est propre à Eclipse. Un projet Web Dynamique n'a pas directement la structure d'un WAR. La structure du WAR se trouve dans un sous-répertoire du projet, c'est le nom de ce sous-répertoire que ce champ permet de modifier (La valeur par défaut est 'WebContent'. L'intérêt de cette approche est de pouvoir stocker dans le projet des fichiers qui ne seront pas visibles par le serveur de test et qui ne seront pas exportés lors de la création du fichier WAR correspondant au projet. Par exemple, le code source n'est pas stocké dans le WAR il se trouve dans un sous-répertoire du projet nommé, par défaut, 'src'.

Développement d'un servlet

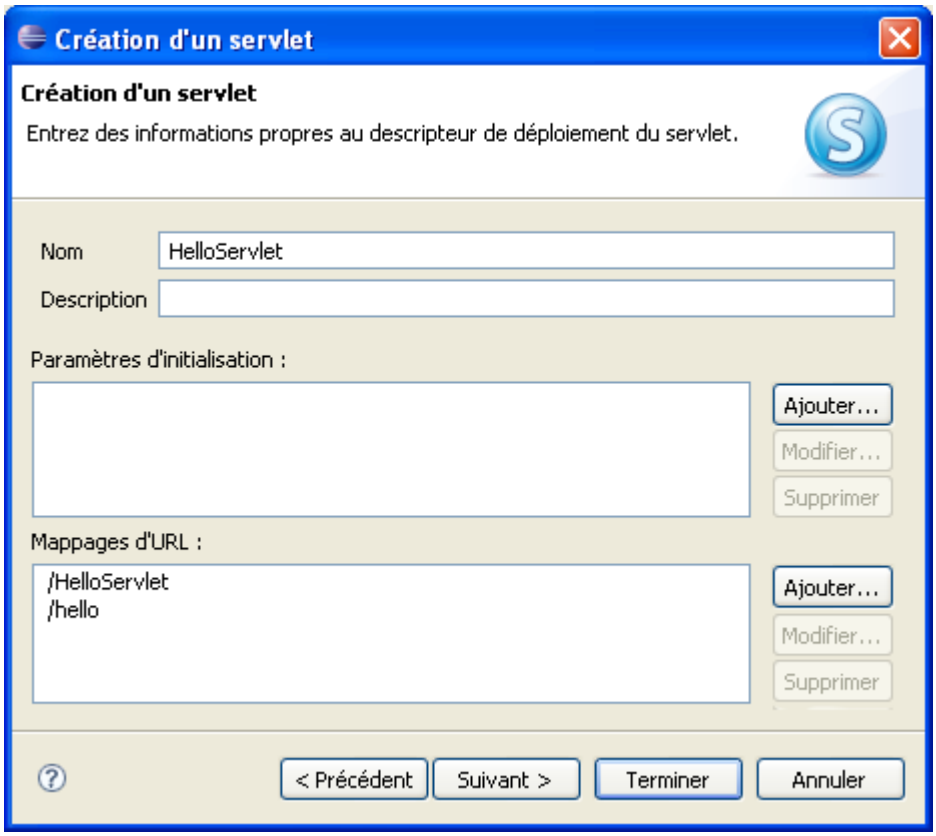
La création d'un servlet est possible en utilisant l'assistant de création de classe, mais pour simplifier les choses Eclipse propose un assistant spécifique. L'ouverture de cet assistant peut se faire à partir du menu contextuel associé au projet Web, dans ce menu choisir 'New->Servlet'



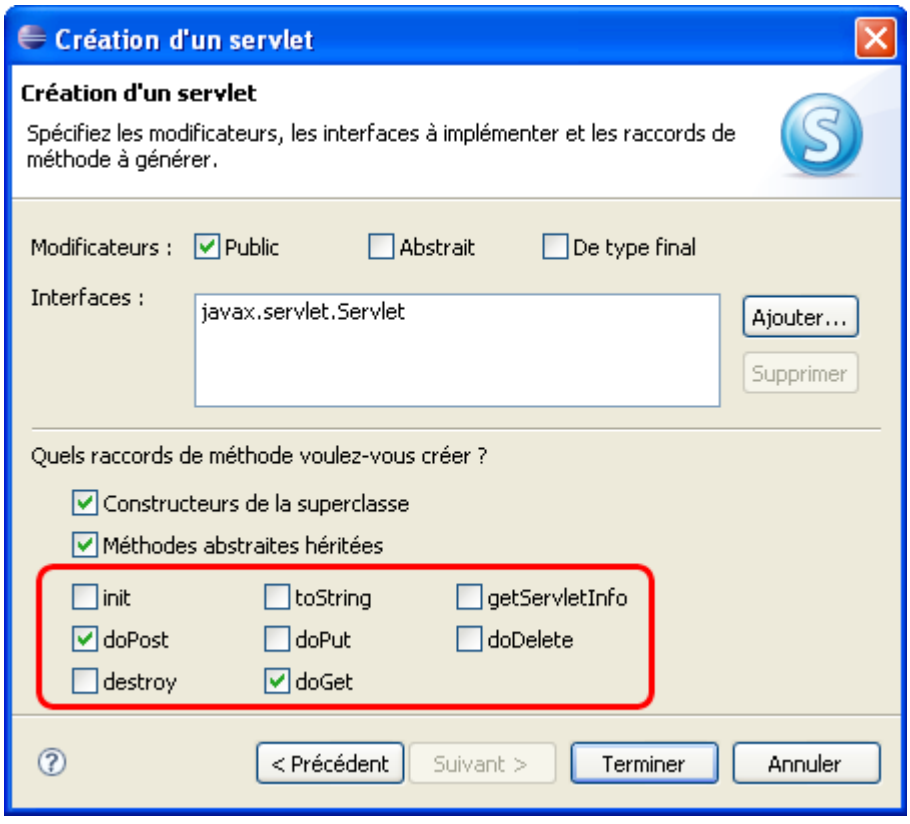
Dans la première page de l'assistant, les informations importantes sont le nom de package et le nom de classe du servlet :



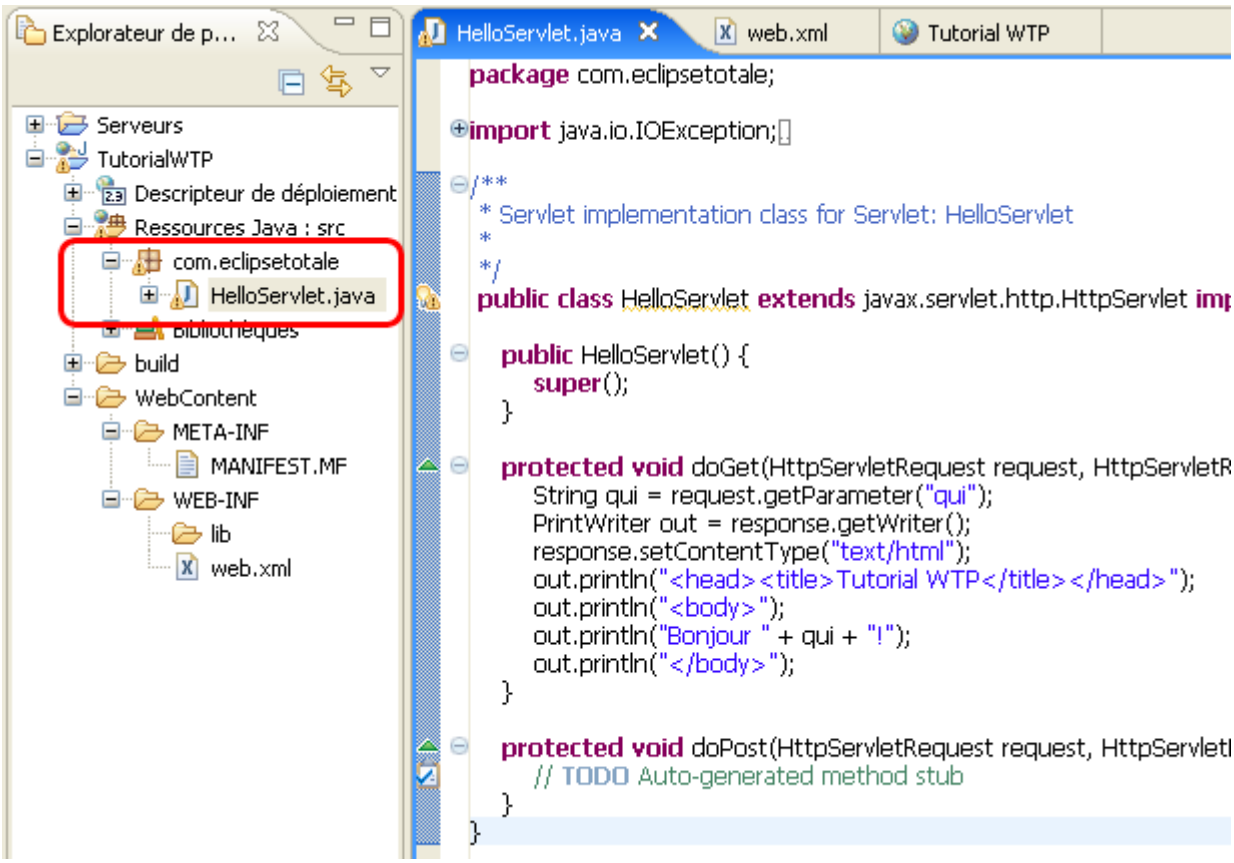
La seconde page de l'assistant permet de saisir des informations qui seront utilisées pour déclarer le servlet dans le descripteur de déploiement de l'application (fichier web.xml). La plus importante de ces informations et la liste des alias (champ 'Mappage d'URL'). Ces alias apparaîtront dans les URL permettant d'accéder au servlet (la forme de ces URLs sera **http://nomDeServeur:port/nomDeContexte/Alias**). Un servlet peut avoir plusieurs alias, par défaut Eclipse définit un alias en utilisant le nom de la classe du servlet.



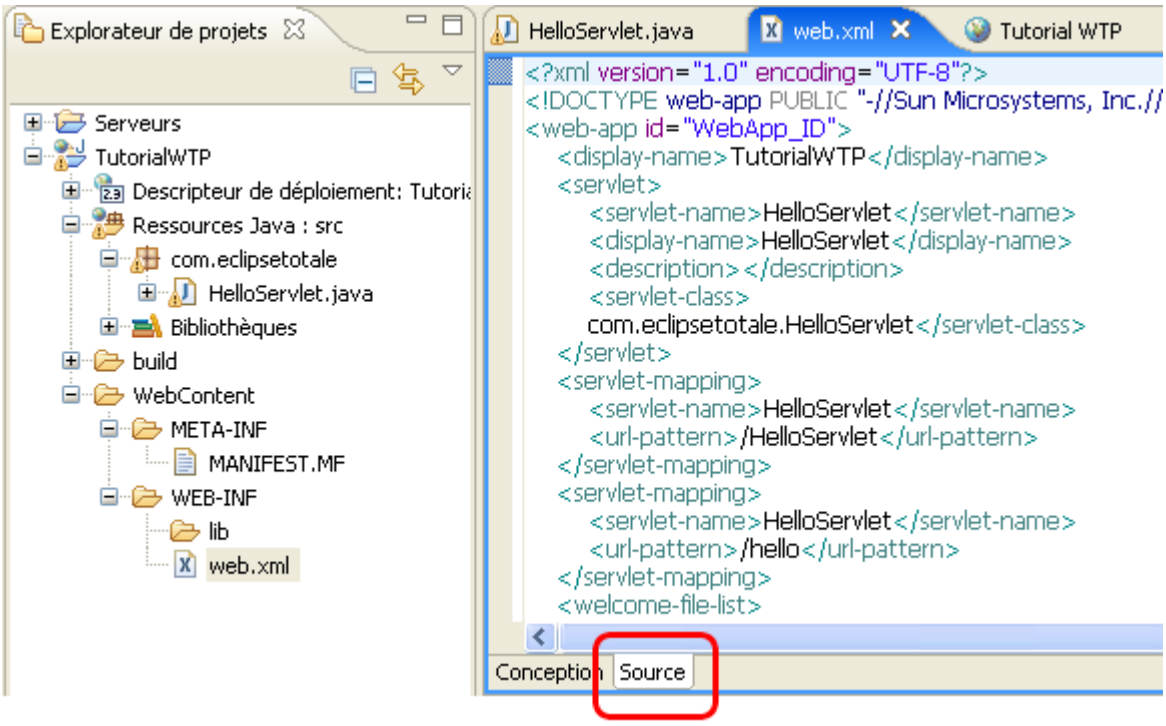
La dernière page de l'assistant permet de sélectionner les méthodes qui devront être générées lors de la création de la classe du servlet :



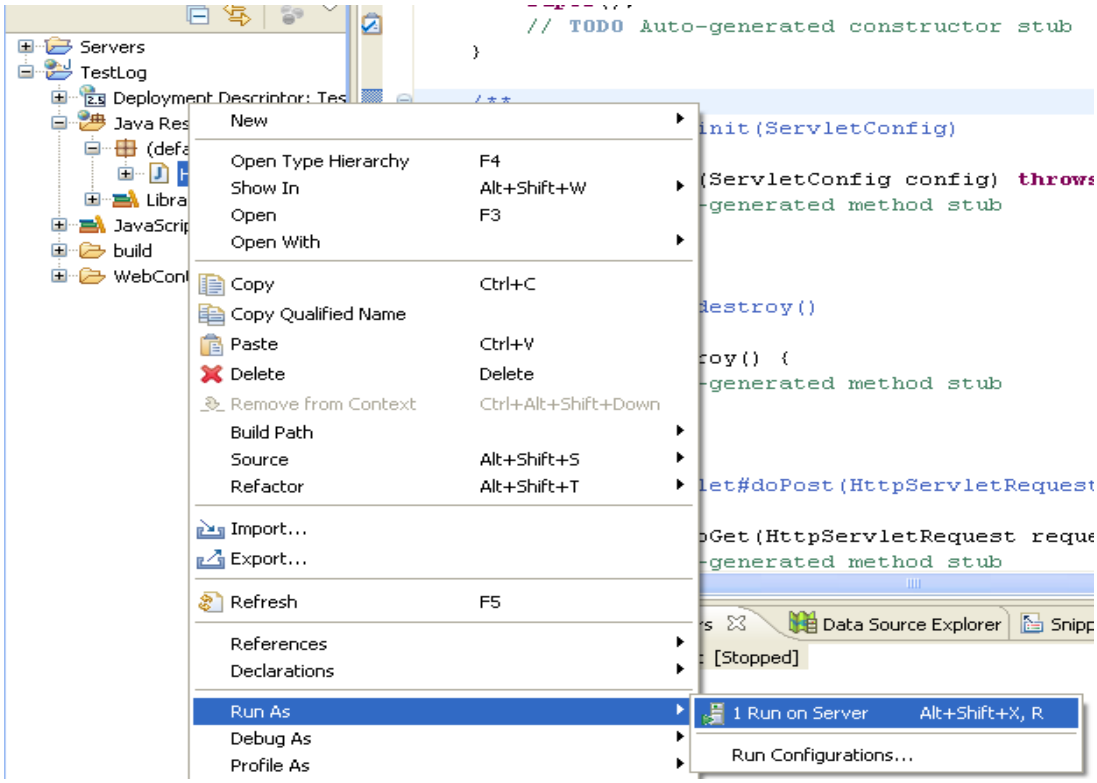
Après avoir cliqué sur 'Terminer', l'assistant crée la classe. Cette nouvelle classe apparaît dans la vue 'Explorateur de projet' et est ouverte en édition. Dans la capture suivante nous avons ajouté du code dans la méthode **doGet** :



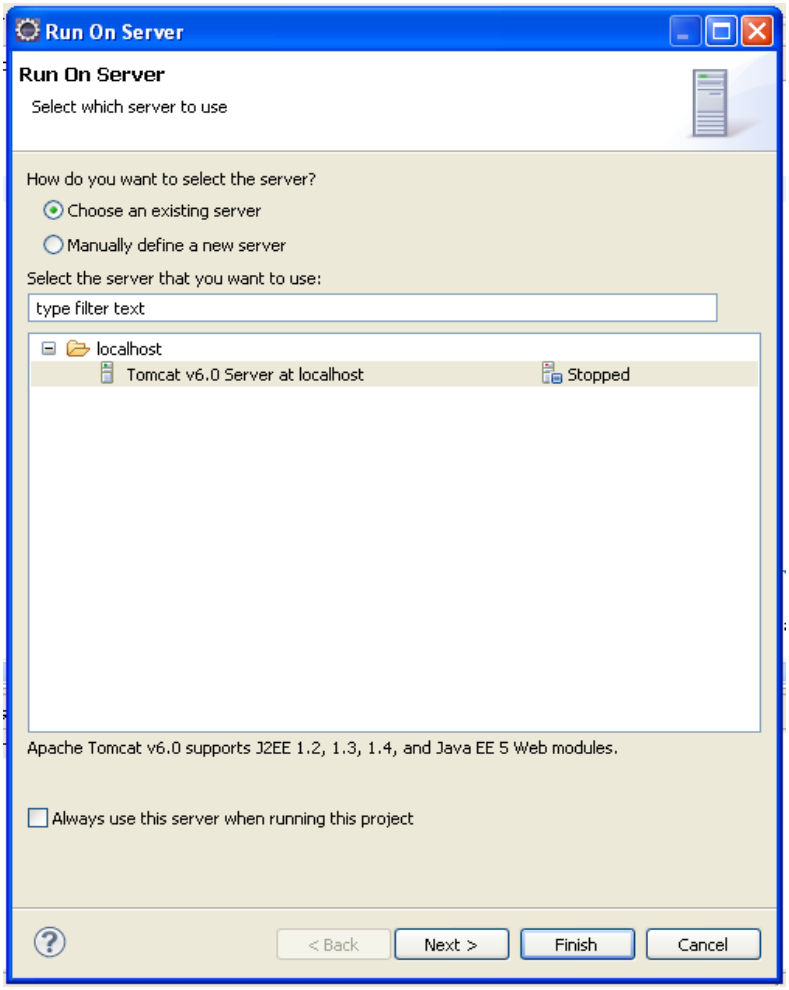
Outre la création de la classe du servlet, l'assistant de création de servlet a mis à jour le fichier web.xml. Ce fichier est éditable et son édition est simplifiée par le fait que Eclipse propose un éditeur pour les fichiers XML (A noter que cet éditeur a deux onglets, l'un propose une vue arborescente, l'autre une vue code source) :



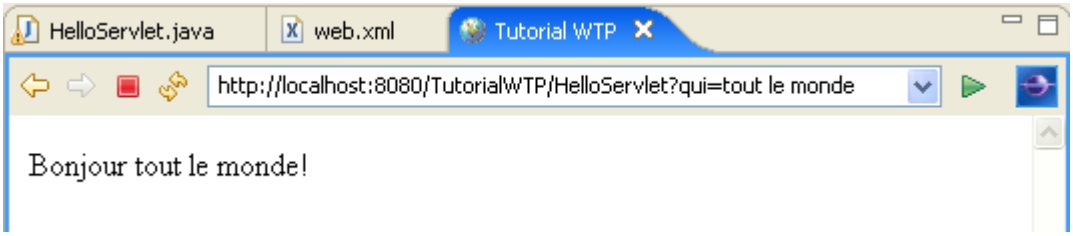
Pour tester la servlet, Eclipse propose un moyen rapide de lancer le serveur Tomcat (s'il n'est pas déjà démarré) et d'invoquer la servlet (un navigateur Web sera ouvert dans la zone d'édition et l'URL permettant d'invoquer la servlet sera appelée automatiquement). Cette action se fait à partir du menu contextuel associé à la classe de la servlet, dans ce menu choisir '**Run AS->Run on Server**'. Le serveur peut aussi être lancé en mode debug en utilisant '**Debug As->Debug on server**'.



Lors de la première utilisation de cette opération une boîte de dialogue permet d'indiquer le serveur sur lequel l'exécution doit se faire :



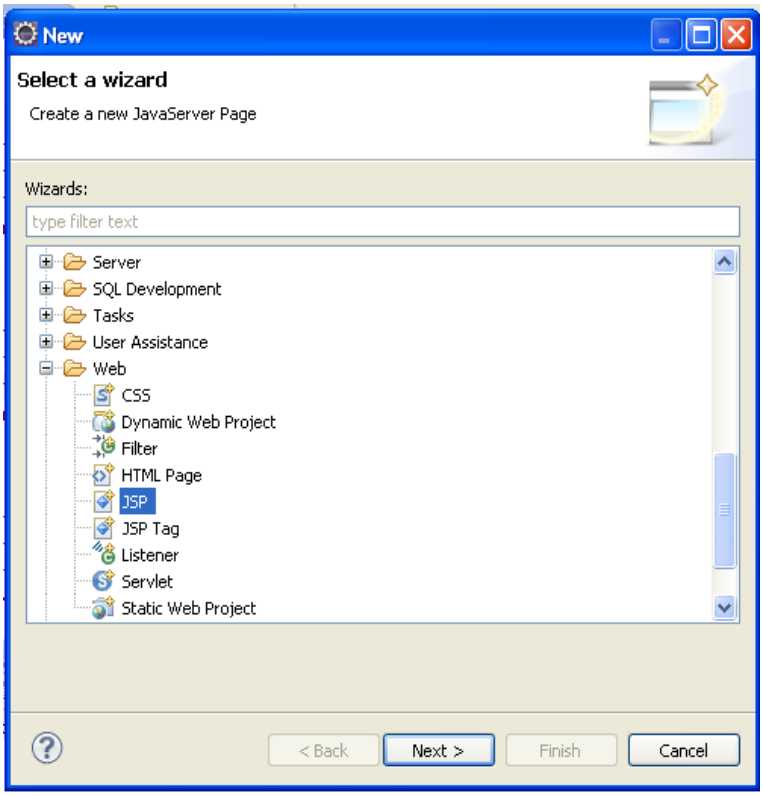
Eclipse ouvre ensuite un navigateur Web dans la zone d'édition et invoque le servlet. Pour notre exemple, nous avons modifié l'URL déclenchée automatiquement par Eclipse pour ajouter le paramètre attendu par notre servlet :



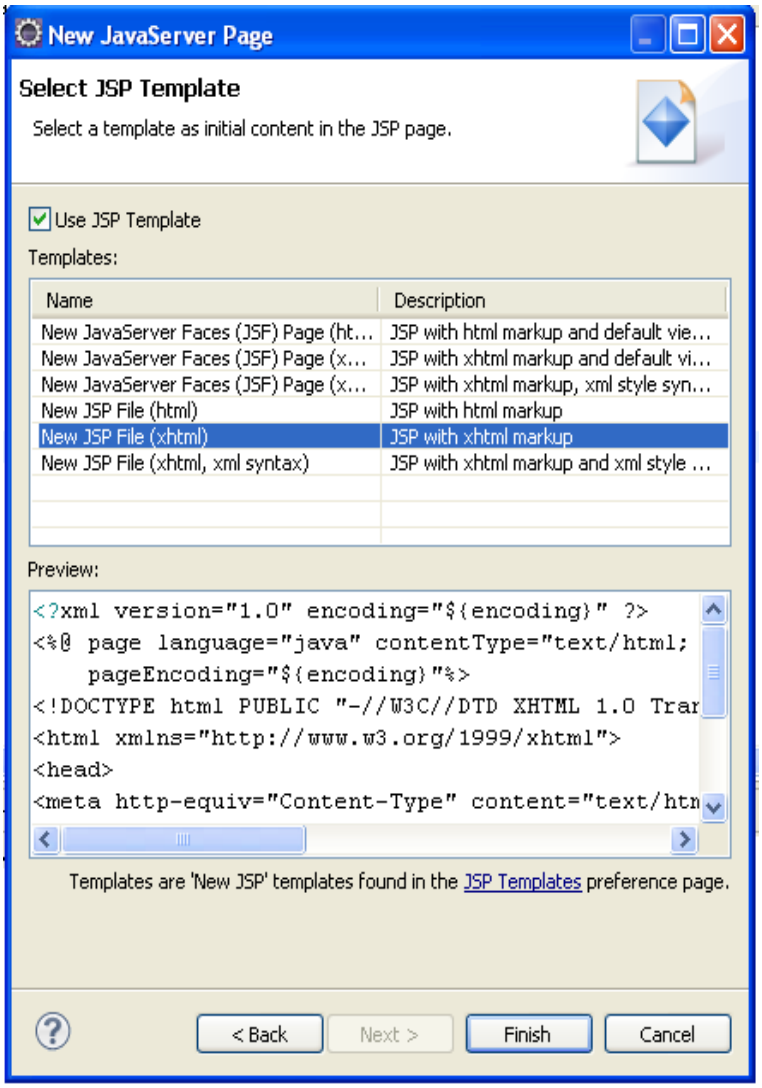
(NB: la navigateur Web utilisé pour afficher le contenu est en fait Internet Explorer sur Windows. Via les préférences il est possible de demander à Eclipse de lancer un navigateur à l'extérieur d'Eclipse. Le navigateur à utiliser peut aussi être indiqué via le menu '**Windows->Web Navigator**'.

Développement d'une JSP

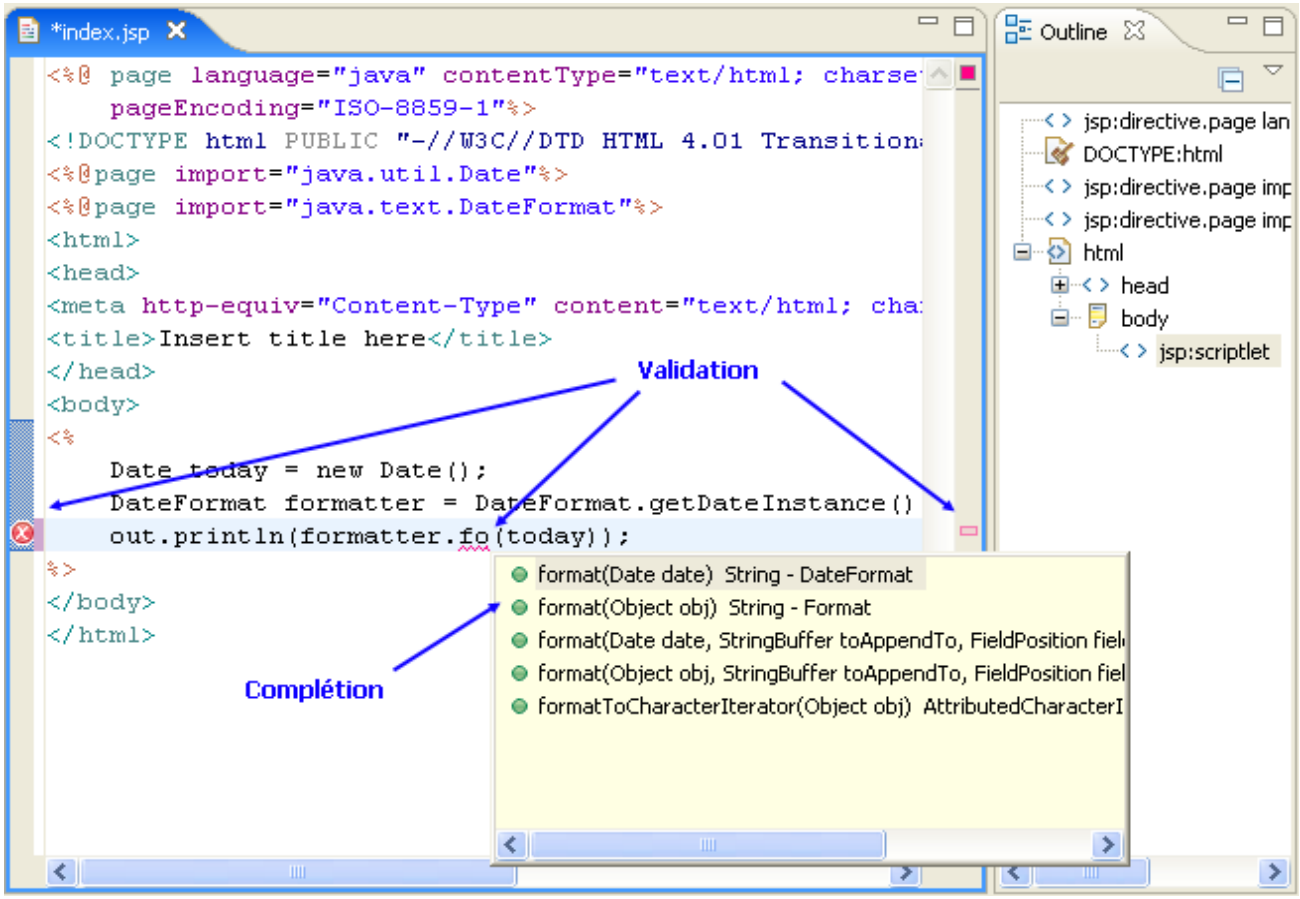
Pour le développement des JSP, Eclipse propose trois principaux outils : un assistant de création, un éditeur de code source et un débogueur.
L'ouverture de l'assistant de création de JSP peut se faire à partir du menu contextuel associé au projet Web, dans ce menu choisir '**New->JSP**'. La première page de l'assistant permet d'indiquer le nom du fichier et son emplacement (par défaut la page est placée à la racine du répertoire WebContent) :



La deuxième page de l'assistant permet de choisir un modèle de page à utiliser lors de la création du fichier. A noter que la liste des modèles est extensible :

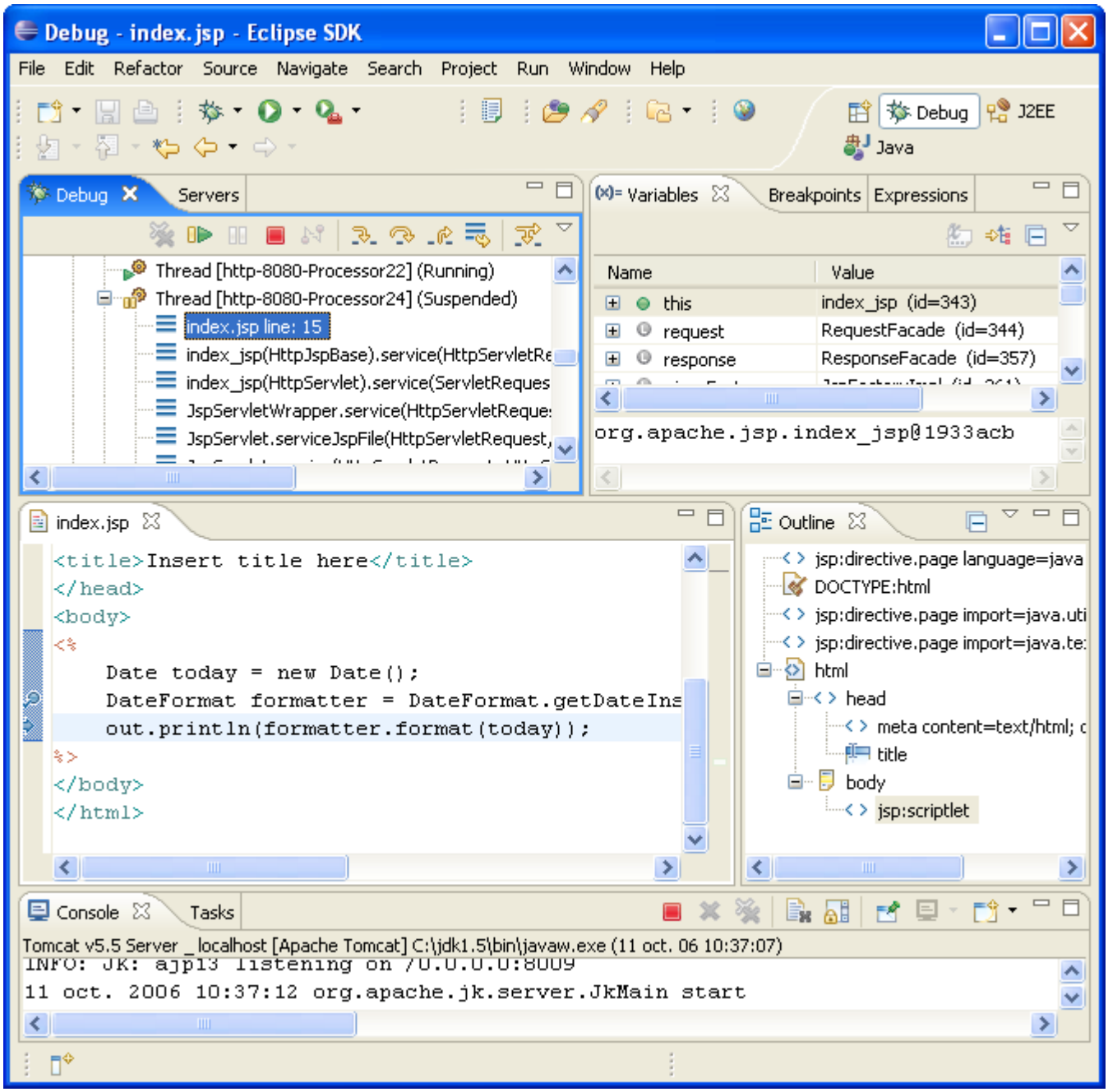


Une fois créée la page est ouverte en édition. L'éditeur de JSP reprend les principales fonctionnalités de l'éditeur de code Java que sont : la coloration syntaxique, l'assistance à la saisie ('complétion' déclenchée par Ctrl+espace), le formatage et l'affichage de la structure du fichier dans la vue 'Outline'. Les particularités des JSP sont prises en compte, soit d'une part le support des tags propres aux JSP et des taglibs, et d'autre part les facilités nécessaires pour développer le code Java intégré dans la page JSP. Le niveau d'aide à l'écriture de code Java se rapproche de celui offert par l'éditeur Java standard d'Eclipse avec notamment le support de la complétion (avec notamment l'ajout des directives d'import) et la validation pendant la saisie.



Pour exécuter la JSP, il est possible de procéder de la même façon que pour le servlet (menu contextuel, choix '**Run As**' ou '**Debug As**'. (NB : l'url générée est parfois fausse pour les JSP).

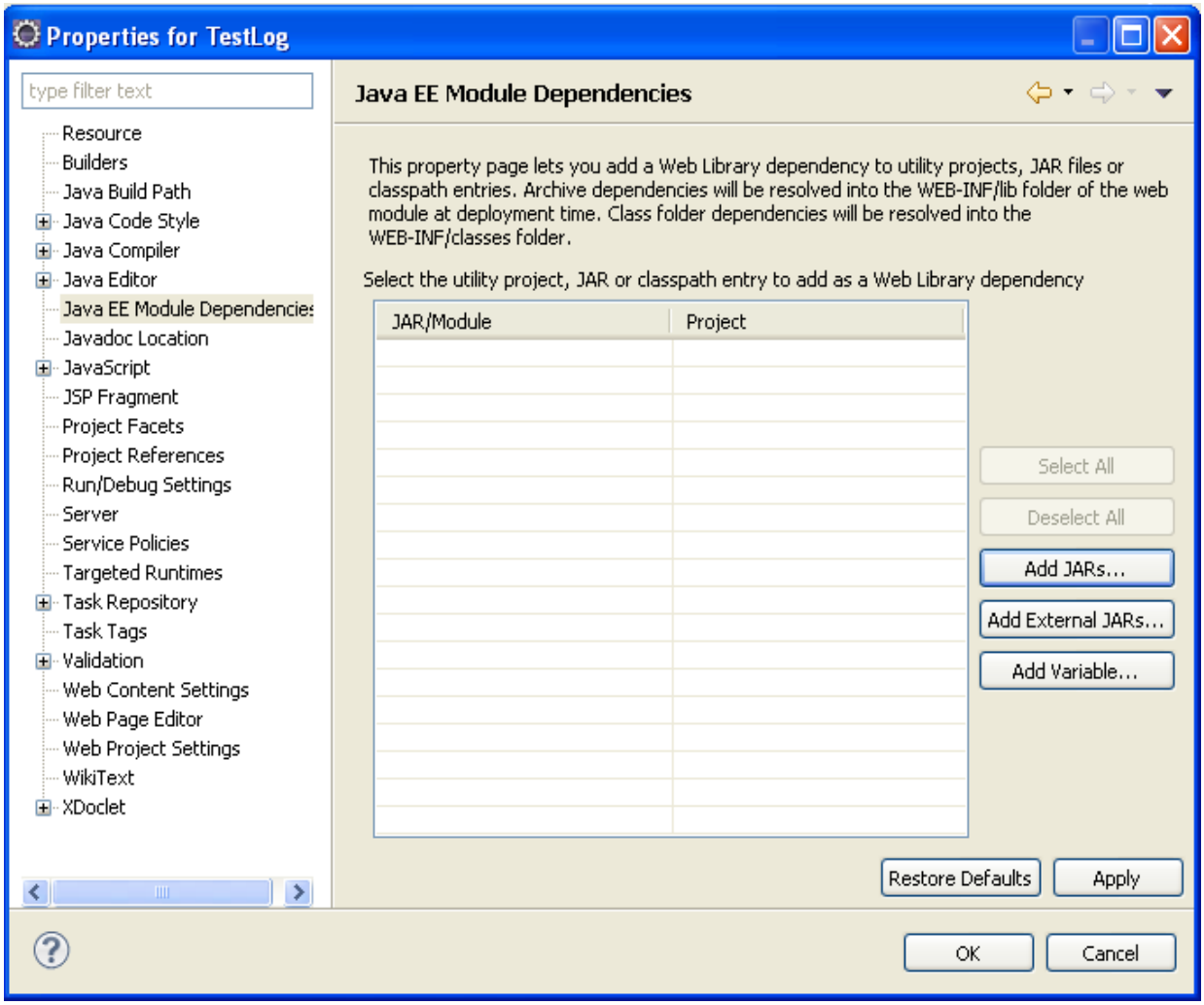
Eclipse simplifie la mise au point des JSP grâce à son débogueur de JSP. Dans une page JSP, des points d'arrêts peuvent être positionnés au niveau des lignes contenant des tags JSP. Si le serveur a bien été lancé en mode debug, le débogueur s'affichera. Le débogueur de JSP fonctionne de la même façon que le débogueur de code Java.



Notion de projet dépendant

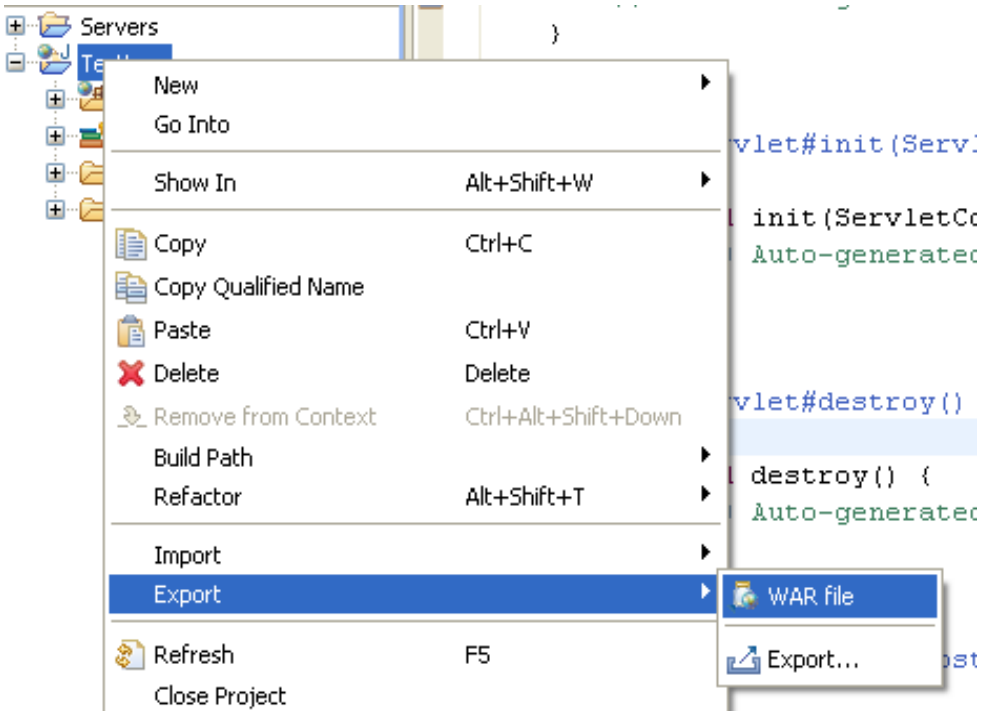
Il est courant que lors du développement le code d'une application soit réparti dans plusieurs projets. Un besoin classique est que le code Java contenu dans un projet Java soit utilisable par une application Web définie dans son propre projet. Eclipse permet de définir une telle dépendance entre un projet Web et un projet Java.

Cette définition se fait à partir des propriétés du projet Web, dans la section 'Dépendances de module Java EE'. Un projet Java coché dans la liste sera vu par l'application Web comme étant un fichier jar dans le répertoire WebContent/WEB-INF/lib.



Exporter l'application

WTP permet d'exporter simplement un 'projet Web dynamique' sous forme d'un fichier WAR à partir du menu contextuel du projet :



La seule information importante est l'emplacement du fichier WAR :