5

Le projet WTP (Web Tools Platform)

Le projet Web Tools est constitué de sous-projets, en particulier JST (J2EE Standard Tools), destiné à fournir à la communauté l'outillage nécessaire au support de la spécification J2EE/JEE, et WST (Web Standard Tools), qui a pour but d'enrichir la plate-forme Eclipse d'outils de développement d'applications Web en utilisant J2EE.

Le projet JST forme l'ossature de la plate-forme Web Tools en fournissant un cadre de développement respectant la spécification J2EE, ainsi qu'un ensemble d'assistants pour le support du serveur d'applications JBoss, qui sert de cible et de support au déploiement de l'application développée.

Le projet WTP inclut les éléments suivants :

- éditeurs pour les langages de pages Web (HTML, CSS, JavaScript) ;
- éditeurs pour les documents XML et associés (XML, DTD, XSD et WSDL) ;
- support de projets J2EE *via* des assistants ;
- support des services Web *via* des assistants ;
- support des bases de données *via* SQL.

Le projet WTP est constitué de sous-projets, notamment les suivants :

- JST (J2EE Standard Tools). Accessible à l'adresse *http://www.eclipse.org/Web Tools/jst/main.php,* ce sous-projet propose des plug-ins pour faciliter le développement d'applications respectant la norme J2EE 1.4 (en particulier JSP, servlets, EJB, JCA, JMS, JNDI, JDBC, Java Web Services, JAX et JSR-associées). Le support pour les spécifications JCP (Java Community Process), accessible sur *http://www.jcp.org,* utilisées dans les applications Web mais non incluses dans la spécification J2EE 1.4, est adapté au cas par cas. Par exemple, JSF fait l'objet d'une implémentation particulière (voir la page du projet JSF *http://www.eclipse.org/jsf*). Rappelons que le rôle premier du JCP est de coordonner l'évolution du langage Java et d'en maintenir la cohésion et la compatibilité, notamment par le biais de certifications.

- WST (Web Standard Tools). Propose un socle pour le développement d'applications Web sous Eclipse.
- JSF (JavaServer Faces). Propose des plug-ins pour faciliter le développement d'applications Web utilisant les JSF.
- Dali. Propose des plug-ins pour faciliter le mapping O/R avec l'API JPA.
- ATF (Ajax Toolkit Framework). Supporte plusieurs conteneurs et serveurs d'applications, en particulier Tomcat (versions 3.2 à 5.5) et JBoss 4.*x* et 5.0 Bêta. Le site officiel du projet est à l'URL *http://www.eclipse.org/Web Tools/*.

JEE5

Bien que WTP supporte la notation J2EE 1.4, nous utiliserons par raccourci la nouvelle définition de la norme Java Platform Entreprise Edition, appelée JEE5, pour le support des JDK 1.5 et 1.6 en lieu et place de J2EE, qui supporte plus spécifiquement les versions des JDK 1.4/1.3/1.2.

Le sous-projet JST (J2EE Standard Tools)

JST est né d'un effort du team Eclipse pour fournir aux utilisateurs de la plate-forme Eclipse un framework standardisé pour la création d'outils de développement adaptés aux applications fondées sur la spécification J2EE.

Le sous-projet JST fonctionne en synergie avec le sous-projet WST (Web Standard Tools), que nous présentons à la section suivante, pour constituer le noyau du projet WTP (Web Tools Platform) en offrant toute une panoplie d'outils de développement, de test, de déploiement et d'administration des applications Web fondées sur le langage Java.

JST est surtout spécialisé dans le support des technologies J2EE, même si certaines technologies hors de ce scope sont également supportées, *via* le standard JCP, à l'image de XDoclet, la populaire technologie d'annotation des sources. D'autres technologies, comme Hibernate ou Velocity, bien que très connues dans le monde du développement Java, ne sont pas supportées par ce projet.

En résumé, JST permet « l'incubation » de projets potentiellement porteurs grâce à certaines fonctionnalités qu'il embarque et qui peuvent être utilisées par le biais de plug-ins tiers supportant ces projets.

Périmètre de JST

JST fournit à Eclipse un ensemble d'outils et de technologies standards pour le développement d'applications Java conformes à la spécification J2EE 1.4.

Le tableau 5.1 récapitule les standards JCP supportés par JST.

Tableau 5.1 Standards JCP supportés par JST

JSR-3 Java Management Extensions (JMX) 1.2
JSR-5 Java API for XML Parsing (JAXP) 1.2
JSR-45 Debugging Support
JSR-54 JDBC API 3.0
JSR-67 SOAP with Attachments API for Java (SAAJ) 1.2

Tableau 5.1 Standards JCP supportés par JST *(suite)*

JSR-77 J2EE Management API 1.0
JSR-88 Deployment API 1.1
JSR-93 Java API for XML Registries (JAXR) 1.0
JSR-101 Java API for XML-based RPC (JAX-RPC) 1.1
JSR-109 Web Services
JSR-112 J2EE Connector Architecture (JCA) 1.5
JSR-115 Java Authorization Contract for Containers (JACC)
JSR-152 JavaServer Pages (JSP) 2.0
JSR-153 Enterprise JavaBeans (EJB) 2.1
JSR-181-Metadata for Web Services,
JSR-154 Servlets 2.4
JSR-175: Metadata Facility for the Java™ Programming Language
JSR-907 Java Transaction API (JTA) 1.0
JSR-914 Java Message Service (JMS) 1.1
JSR-919 JavaMail 1.3

En complément, les standards et technologies non-JCP suivants sont inclus dans le périmètre de JST :

- JAAS (Java Authentication and Authorization Service)
- JNDI (Java Naming and Directory Interface)
- XDoclet

Architecture du sous-projet JST

Comme indiqué précédemment, JST fournit un socle pour le développement d'outils J2EE dédiés. Appelé JCM (J2EE Core Model), ce socle contient un ensemble de frameworks et d'objets du modèle permettant d'abstraire les principales fonctionnalités J2EE ainsi que les composants associés et de fournir un ensemble d'API d'accès pour manipuler ces fonctionnalités.

Le modèle JCM est accessible à d'autres éditeurs ou développeurs pour étendre les outils de développement J2EE déjà disponibles dans la plate-forme Eclipse. Comme nous l'avons vu, il fournit en outre un support pour d'autres technologies prometteuse non encore directement supportées par Eclipse. Précisons que les outils fournis par le sous-projet JST sont eux-mêmes des extensions du modèle JCM.

Les extensions du modèle JCM sont les suivantes :

- Modèle de projet J2EE. Fournit un framework pour la gestion de projets J2EE et supporte une structure flexible pour la gestion des composants J2EE et le support des activités associées (compilation, déploiement, etc.).
- Modèle d'éditeur J2EE. Étend l'éditeur standard Eclipse pour fournir un support pour la création et l'édition de ressources J2EE, comme les JSP, les servlets, les EJB, etc. Il fournit en outre la base pour la création d'éditeurs spécifiques pour les textes et les images, en particulier pour le support de fonctionnalités d'édition de syntaxe colorée, d'assistant de code, de refactoring de code, etc.

- Modèle d'artefacts JEE. Représente les sources JEE ainsi que les composants JSP, EJB, descripteurs de déploiement, etc., qui peuvent être créés et gérés au sein d'un projet en même temps que d'autres artefacts, comme les ressources de type image, texte et autres fichiers qui peuvent être packagés dans un module J2EE déployable (war/ejb, jar/rar).
- Modèle de serveur JEE. Fournit les abstractions requises pour supporter le déploiement de modules pour différents types de serveurs d'applications. Il offre de manière additionnelle un mécanisme unifié pour démarrer, administrer et stopper les serveurs d'applications JEE. Ce modèle fournit le moyen de gérer les détails de configuration des serveurs, comme les paramètres JVM, les variables d'accès aux chemins (classpath), etc. Enfin, les modules J2EE tels que le packaging, le déploiement, le débogage ou l'import/export de modules J2EE sont gérés par ce modèle.

L'outillage J2EE standard de JST

Examinons à présent l'outillage JST permettant la création, le support et la gestion d'artefacts JEE. Ces outils sont créés en étendant les modèles et les API rendus disponibles par JCM. Le résultat est un ensemble de vues, de perspectives, d'éditeurs et d'assistants dédiés aux différentes activités de développement.

Chaque outil est dédié à une activité particulière du développement d'applications d'entreprise J2EE. Ainsi, les outils dédiés au support des servlets/JSP sont utiles au développement d'applications Web et à leur déploiement dans le conteneur, tandis que les outils prévus pour le support des EJB permettent de se concentrer sur le développement et le déploiement sur le conteneur d'EJB.

Notion de projets J2EE et de modules

Un projet Eclipse est utilisé pour développer des modules applicatifs J2EE Web, ou EJB. Chaque module J2EE est en fait développé dans un projet séparé et possède son propre accès aux classes définies dans le JDT (outillage Java d'Eclipse). Il ne faut toutefois pas confondre module et projet. Il est ainsi possible que différents projets partagent un même module (par exemple, un module jar contenant les bibliothèques partagées par le projet J2EE).

Un projet Eclipse « basique » fournit une structure pour organiser les ressources dans le projet (ressources de types fichiers et répertoires) :

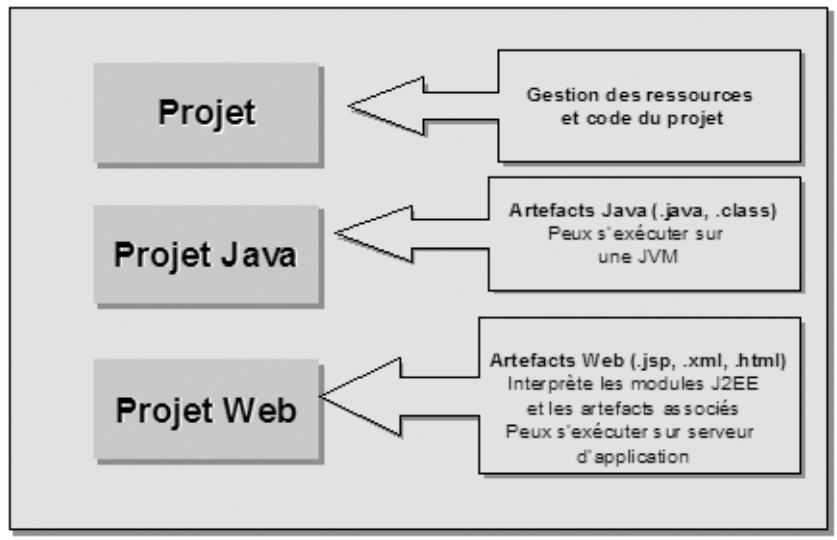
- Projet Java, qui contient les éléments et artefacts (packages, méthodes, classes et attributs) nécessaires au développement Java.
- Projet Web, qui contient en plus du code Java des ressources pour le support du développement Web (bibliothèques Struts, fichiers html, servlets, JSP, bibliothèques de balises, services Web et descripteurs de déploiement).

La figure 5.1 illustre l'organisation en composants d'un projet Eclipse.

La figure 5.2 décrit les constituants du packaging final d'une application JEE, répartis selon les modules projets précédents, l'ensemble de ces modules étant packagés dans un fichier EAR avec leurs descripteurs respectifs.

Figure 5.1

Modules projet d'un projet Eclipse



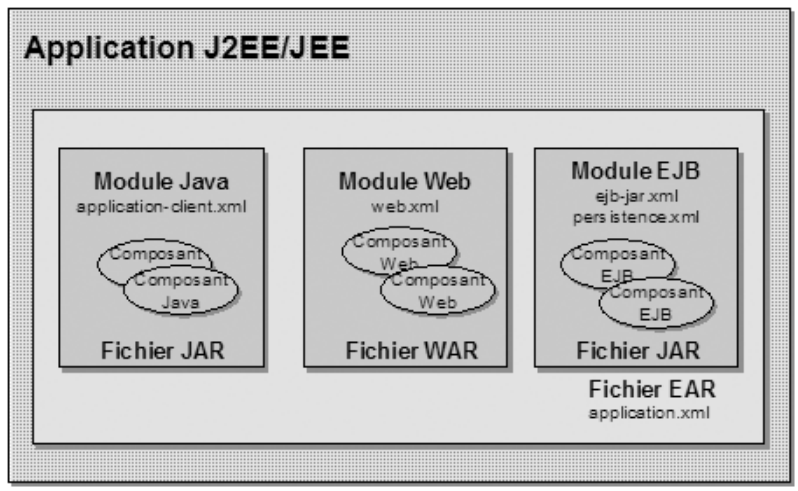


Figure 5.2

Packaging final d'une application J2EE/JEE

Outils de support à la création de projets J2EE

L'outillage de support à la création de projets J2EE permet la création d'un ensemble d'artefacts qui constituent le module de déploiement spécifique de l'application J2EE.

Cet outillage supporte la création de cinq types de projets (voir figure 5.3) :

- Projet EJB (EJB Project) : ensemble de composants beans d'entreprise supportant les artefacts qui seront compilés et déployés au sein d'un conteneur d'EJB sur le serveur d'applications cible.
- Projet de type Web dynamique : ensemble de composants servlets, JSP et Web tiers (bibliothèque de balises, pages HTML, images et documents) qui seront compilés et déployés au sein d'un conteneur Web sur le serveur d'applications cible.

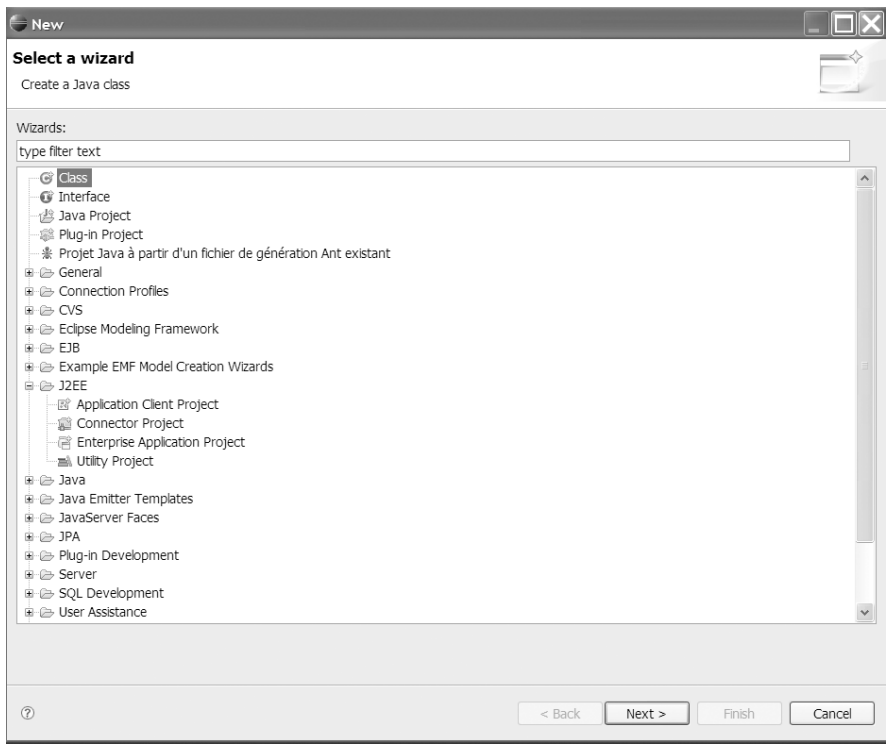
- Projet de client d'application (Application Client Project) : ensemble de composants client conçus pour la consommation de services fourni par une application d'entreprise.
- Projet connecteurs (Connector Project) : ensemble de fichiers source destinés à la création de connecteurs applicatifs pour l'intégration aux systèmes « legacy » spécifiés par la JSR-112 (J2EE Connector Architecture 1.5).
- Projet d'application d'entreprise (Enterprise Application Project) : ensemble de modules représentant une application d'entreprise complète. Ce type de projet fournit la possibilité de référencer différents composants de type EJB, Web, client d'application et connecteur qui seront déployés ensemble.

Les modules J2EE additionnels fournissent de nombreuses fonctionnalités pour la création de projet, notamment la possibilité de créer et de gérer des artefacts selon les types de projets ci-dessus et, surtout, de créer une structure de déploiement type pour le support au projet J2EE (type WAR pour un projet Web, ou EAR pour une application d'entreprise, par exemple).

La figure 5.3 illustre l'assistant de création de projet JST proposant un certain nombre d'options pour un projet d'entreprise J2EE.

Figure 5.3

Assistant de création de projet JST



Europa

À l'heure de la rédaction de cet ouvrage, la version Eclipse 3.3 « Europa » ne dispose pas d'un support de la langue française. Nous nous référons donc à la version anglaise en notre possession.

Outils de support aux serveurs J2EE

Les applications J2EE doivent être déployées dans un conteneur Web ou EJB d'un serveur d'applications J2EE/JEE compatible. Avec JST, la gestion des serveurs sur lesquels les applications J2EE seront déployées est gérées par l'outillage du serveur J2EE.

L'outillage de support aux serveur fournit un mécanisme pour la définition de l'environnement d'exécution du serveur J2EE/JEE ainsi que pour la création d'instances du serveur. Cet outillage assure que, lors de la création d'un projet, ce dernier puisse être déployé.

Comme l'illustre la figure 5.4, lors de la création d'un projet de type Enterprise Application Project, l'assistant de support au projet intégré à JST propose un certain nombre de serveurs d'applications compatibles.

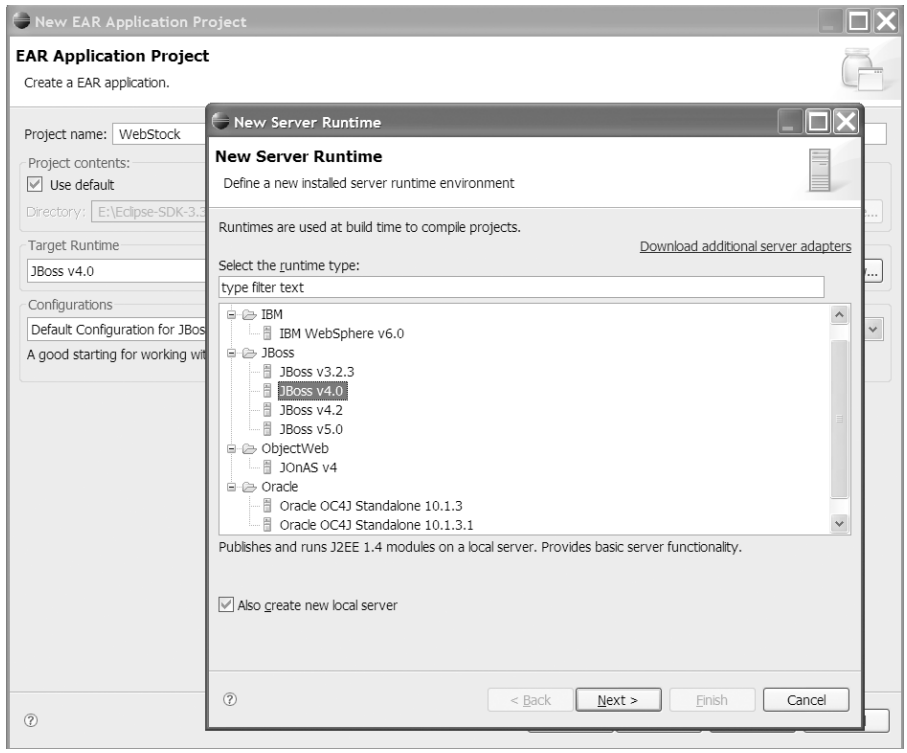


Figure 5.4

Support aux principaux serveurs J2EE/JEE compatibles

Avec JST, un projet J2EE/JEE nécessite l'existence d'une JRE (Java Runtime Environment) et d'un environnement d'exécution J2EE/JEE compatible. L'environnement d'exécution J2EE/JEE fournit un support pour les fonctionnalités d'entreprise utilisées dans le projet. JST supporte la définition de plusieurs versions d'environnements d'exécution J2EE/JEE permettant la création de projets déployables sur différentes versions de JVM et sur différentes versions de serveur d'applications cible.

Le tableau 5.2 décrit les serveurs d'applications supportés par JST.

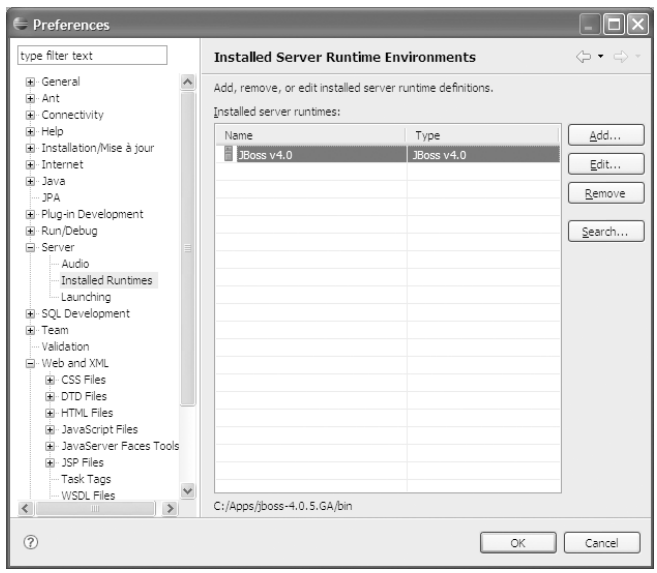
Tableau 5.2 Serveurs d'applications et spécifications J2EE supportés par JST

Version Web	Version EJB	Serveur d'applications J2EE/JEE
2.2, 2.3 et 2.4	1.1, 2.0 et 2.1	Apache Geronimo 1.0
2.2		Apache Tomcat version 3.2
2.2 et 2.3		Apache Tomcat version 4.1
2.2, 2.3 et 2.4		Apache Tomcat version 5.0
2.2, 2.3 et 2.4		Apache Tomcat version 5.5
2.2, 2.3,2.4 et 2.5		Apache Tomcat 6.0.13
2.2, 2.3 et 2.4	1.1, 2.0 et 2.1	IBM WebSphere 6.0.x
2.2, 2.3, 2.4	2.0 et 2.1	BEA WebLogic Server version 8.1
2.2, 2.3 et 2.4	1.1, 2.0 et 2.1	BEA WebLogici Server version 9.0
2.2, 2.3 et 2.4	1.1 et 2.1	JBoss 3.2.3
2.2, 2.3 et 2.4	2.0 et 2.1	JBoss 4.0
2.2, 2.3 et 2.4	2.0 et 2.1	JBoss 4.2
	3.0	JBoss 5.0 (Beta)
2.2, 2.3 et 2.4	1.1, 2.0 et 2.1	Jonas V4

La figure 5.5 illustre la définition d'une cible serveur d'applications JBoss 4.0.5 (accessible *via* le menu Preferences d'Eclipse).

Figure 5.5

Définition d'un environnement serveur JBoss 4.0.5

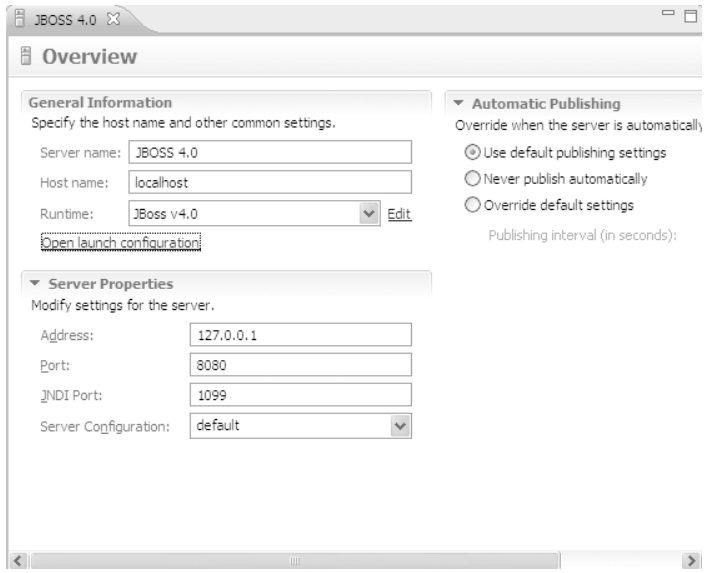


Des configurations supplémentaires sont disponibles pour chaque environnement cible défini sur le serveur permettant d'agir sur les variables d'accès aux classes, ainsi que sur les arguments d'exécution de la JVM et de démarrage de l'instance.

La figure 5.6 illustre une configuration par défaut définie pour le serveur d'applications JBoss 4.0 (vous pouvez aussi agir sur le mode de publication et de déploiement des composants serveur *via* l'option Automatic Publishing).

Figure 5.6

Configuration d'un environnement d'exécution serveur (JBoss)



Le sous-projet WST (Web Standard Tools)

Cette section donne une vue d'ensemble des principaux composants du sous-projet WST. Pour en savoir plus sur ce projet, voir *http://www.eclipse.org/Web Tools/wst/components.html.*

Ce sous-projet inclut des outils et des API offrant le support d'applications Web standards indépendantes de la technologie sous-jacente.

WST fournit des fonctionnalités de base qui peuvent être étendues par d'autres projets. Les sections qui suivent donnent des exemples de facilités offertes par ce sous-projet, partie intégrante du projet Web Tools.

Support des facets projet

Parmi ces fonctionnalités, la notion de Project Facets est la plus importante.

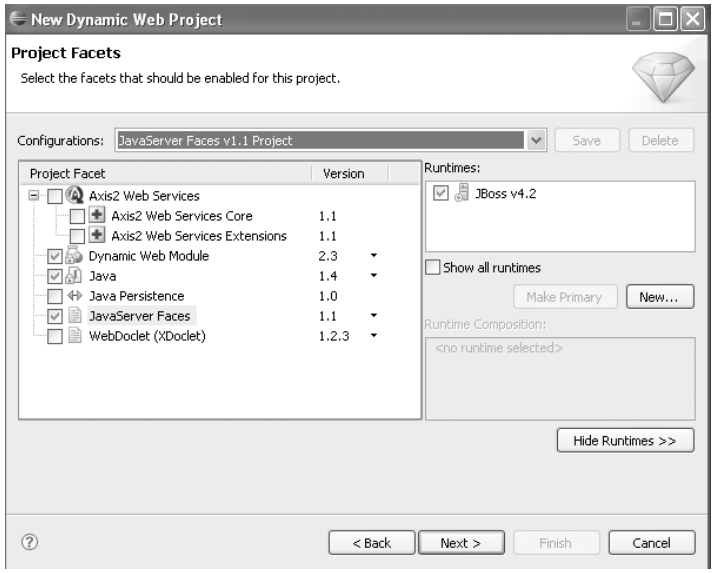
Les facets permettent d'ajouter des fonctionnalités à un projet Eclipse et de les supprimer ensuite au besoin si ceux-ci ne sont plus requis et que votre projet supporte d'autres technologies.

En résumé, une facet est une caractéristique que le projet peut supporter (par exemple JSF). Web Tools définit quelques types de projets relativement génériques (par exemple Dynamic Web Project) et offre la possibilité d'ajouter et de supprimer dynamiquement des fonctionnalités à ces projets.

Nommées Project Facets, ces fonctionnalités sont configurables lors de la création du projet et par la suite dans la page des propriétés du projet (voir figure 5.7).

Figure 5.7

Facets proposées par WST



Support à l'outillage HTML, CSS et JavaScript

Comme autre apport remarquable apporté par WST, citons le support aux ressources HTML, CSS et JavaScript.

WST propose des éditeurs de code source pour chacun de ces langages (syntaxe colorée, aide à la complétion, etc.), comme l'illustre la figure 5.8.

Figure 5.8

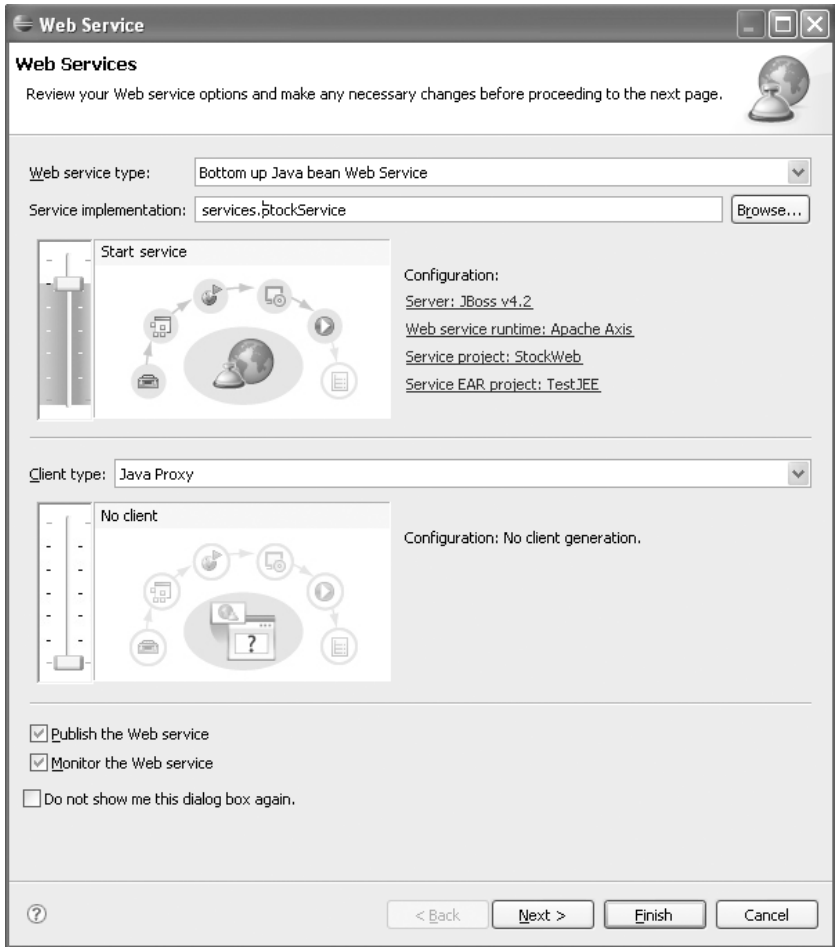
Aide au support des ressources HTML avec le projet WST

Support des Web Services

WST supporte les moteurs SOAP, en particulier Apache Axis 1.2.1. Pour le développement de Web Services, les trois fonctionnalités les plus visibles sont un éditeur de fichier WSDL (ressemblant à l'éditeur de fichier XMLSchema), des assistants pour la création de Web Services (génération à partir d'une classe Java) et un outil permettant d'invoquer un service Web depuis l'explorateur de services Web (voir figure 5.9).

Figure 5.9

Assistant WST de création et de publication de service Web



L'assistant WST de création et de publication de service Web offre les avantages suivants :

- Développement de services Web de type bottom-up/top-down (à partir d'un JavaBean existant ou non).
- Démarrage de services Web dans un projet Web.
- Génération de proxy Java.
- Test de services Web.
- Monitoring de services Web. Web Tools inclut un moniteur embarqué qui permet de surveiller les enveloppes SOAP et les flux entre le client et le service invoqué.

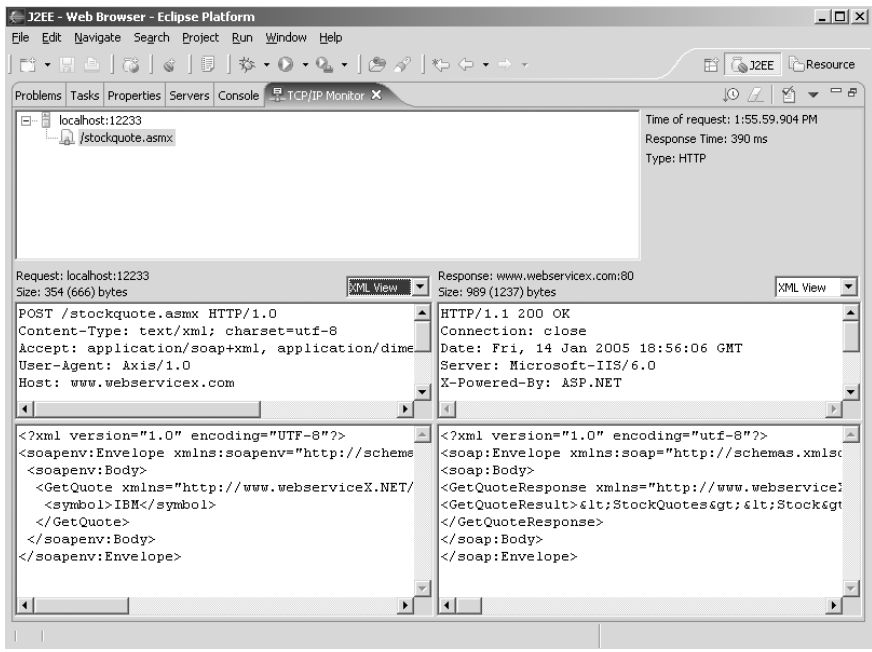
Monitoring TCP/IP

WST inclut un navigateur Web permettant d'accéder aux pages Internet à l'intérieur de l'IDE mais aussi de mesurer le trafic HTTP échangé, permettant ainsi la capture et l'analyse des messages envoyés et reçus à partir d'un port et d'un hôte spécifique.

L'outil de monitoring TCP/IP permet sauvegarde les messages dans un fichier de log, lequel peut ensuite être analysé avec la suite d'outils de test intégré (voir figure 5.10).

Figure 5.10

Outil de monitoring TCP/IP de WST



Mise en œuvre de WTP

Avant de commencer à utiliser les assistants de développement Web, vous allez commencer par installer et configurer WTP.

La version utilisée ici est celle supportée par Eclipse 3.3 (Europa), à savoir la 2.0 RC4.

La version de WTP 2.0 disponible en téléchargement sur le site du projet *(http://download.eclipse.org/Web Tools/downloads/)* suppose l'installation préalable sur la plate-forme Europa des composants récapitulés au tableau 5.3.

Tableau 5.3 Composants à installer sur Europa

Composant	Version
Eclipse Platform (JDT, PDE)	Eclipse SDK 3.3 (Europa)
Eclipse Modeling Framework (EMF, XSD InfoSet, SDO)	emf-sdo-xsd-SDK-2.3.0RC4
GEF (Graphical Editing Framework)	GEF-SDK-3.3RC4
DTP (Data Tools Platform)	dtp-sdk_1.5RC4

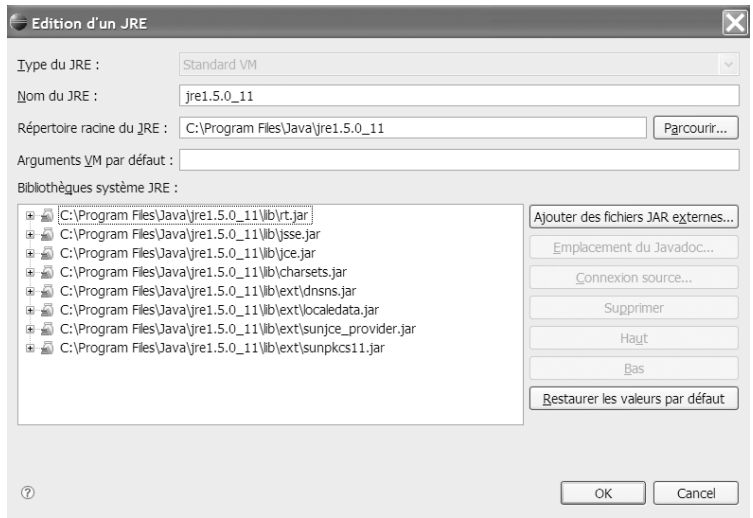
L'installation de ces composants ne présente aucune difficulté particulière et s'effectue, comme pour la distribution d'Eclipse, par une décompression des binaires dans le répertoire d'installation d'Eclipse (par exemple, sous Windows c:\MonEclipse\3.3\eclipse). Une fois ces composants téléchargés, prenez soin de lancer l'IDE Eclipse par le biais de l'option –clean (sous c:\MonEclipse\3.3\eclipse\bin).

Configuration de l'environnement d'exécution

Vous allez commencer par créer un environnement d'exécution Java. Cette étape initiale est indispensable, car la création d'un environnement d'exécution J2EE dépend de l'existence d'une JRE installée et fonctionnelle, accessible *via* Preferences, Java, JRE installés (voir figure 5.11).

Figure 5.11

Configuration de la JRE



Version de JRE

Si vous ne connaissez pas l'emplacement de votre JRE, vous pouvez repérer celle-ci en consultant la variable JAVA_HOME définie dans votre système.

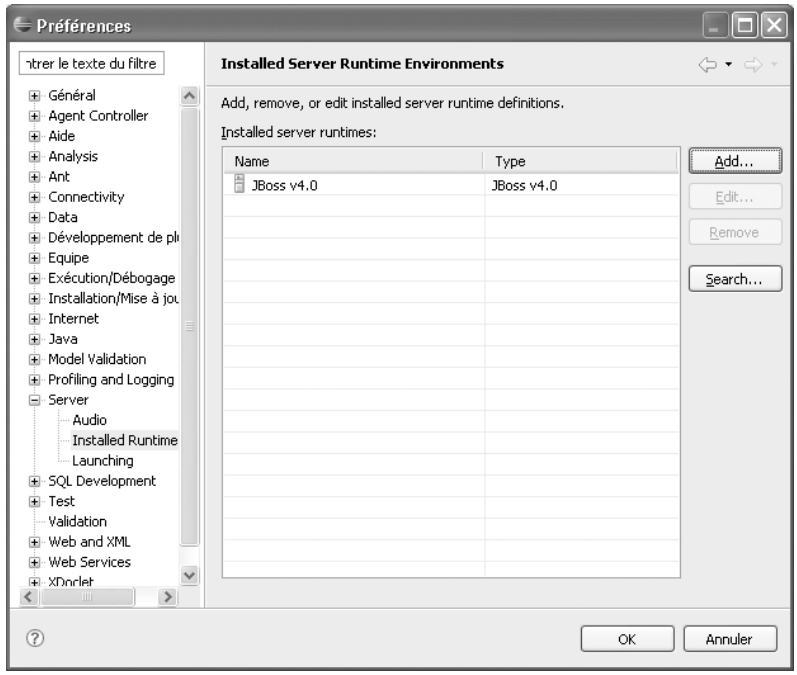
Dans cette configuration, optez pour une JRE 1.5.0 Update 11. La sélection du répertoire d'installation de la JRE installe toute les bibliothèques associées à cette dernière, sachant qu'il est aussi possible d'ajouter des bibliothèques jar additionnelles dans l'environnement d'exécution de la JVM.

L'étape suivante consiste à configurer l'environnement d'exécution du serveur J2EE offrant le support des caractéristiques spécifiques qui seront utilisées au sein de l'application J2EE déployée. À l'inverse de la configuration de la JRE vue précédemment, qui est généralement fournie par Sun Microsystems, le runtime J2EE est disponible à partir d'une variété d'éditeurs proposant leur propre implémentation de serveurs J2EE compatibles (généralement inclus dans l'environnement du SA).

La configuration du serveur est accessible par le biais de Window, Preferences, Installed Runtimes et Add. L'assistant de définition et de configuration s'affiche alors comme illustré à la figure 5.12 (ici avec JBoss).

Figure 5.12

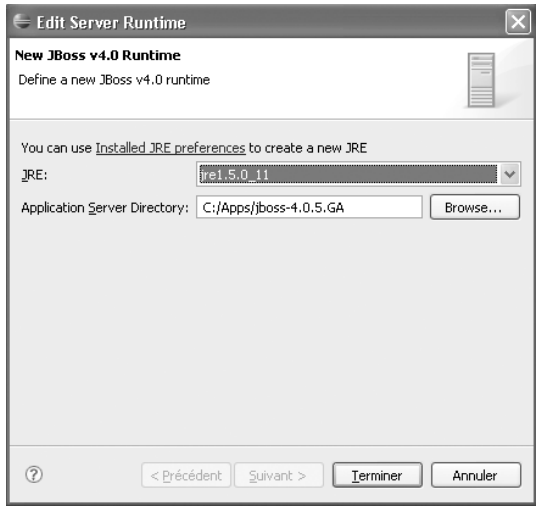
Configuration du serveur JBoss (1/2)



L'assistant de configuration du serveur permet, en fonction du serveur cible présélectionné, de définir la localisation de la JRE, ainsi que du serveur d'applications, avec intégration immédiate des bibliothèques jar disponibles pour ce serveur (voir figure 5.13).

Figure 5.13

Configuration du serveur JBoss (2/2)



Eclipse supporte la création de plusieurs environnements J2EE, chaque environnement étant dédié à un serveur d'applications particulier installé sur le système.

Il est possible de créer plusieurs configurations d'exécution fondées sur des installations particulières du serveur d'applications cible. Cela peut se révéler utile notamment pour tester l'application en utilisant différentes versions de JRE.

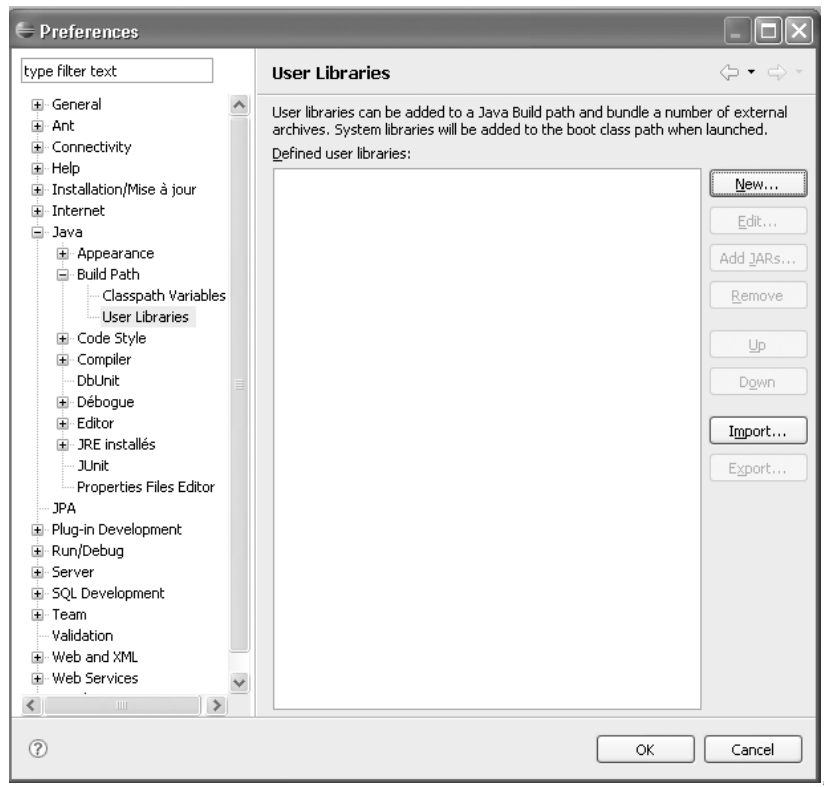
Configuration des bibliothèques additionnelles spécifiques

Dans certaines situations, il peut être nécessaire au support de certaines fonctionnalités du serveur d'applications cible d'ajouter certaines bibliothèques, appelées User Librairies, ou bibliothèques utilisateur, accessibles par le biais des options Java, Build Path et User Libraries du menu Préférences d'Eclipse (voir figure 5.14).

C'est en particulier le cas si vous utilisez la version du serveur JBoss 4.0.5 illustré dans cette section, pour supporter par exemple la technologie EJB3 du serveur JBoss (voir figure 5.14).

Figure 5.14

Ajout de bibliothèques additionnelles



Dans le contexte du développement EJB3, vous devez utiliser certaines bibliothèques de support à cette technologie pour le serveur JBoss 4.0.5.

Procédez pour cela de la façon suivante :

1. Cliquez sur New, et nommez votre bibliothèque JBOSS_EJB3.
2. Ajoutez les jar suivants (*via* le bouton Add JARS) :

   ```
   $JBOSS_HOME/server/default/lib/ejb3-persistence.jar
   $JBOSS_HOME/server/default/deploy/ejb3.deployer/jboss-ejb3.jar
   $JBOSS_HOME/server/default/deploy/ejb3.deployer/jboss-ejb3x.jar
   $JBOSS_HOME/server/default/lib/jboss-j2ee.jar
   ```

   La configuration de votre bibliothèque doit ressembler à celle illustrée à la figure 5.15 (en fonction du répertoire d'installation de votre serveur d'applications JBoss).

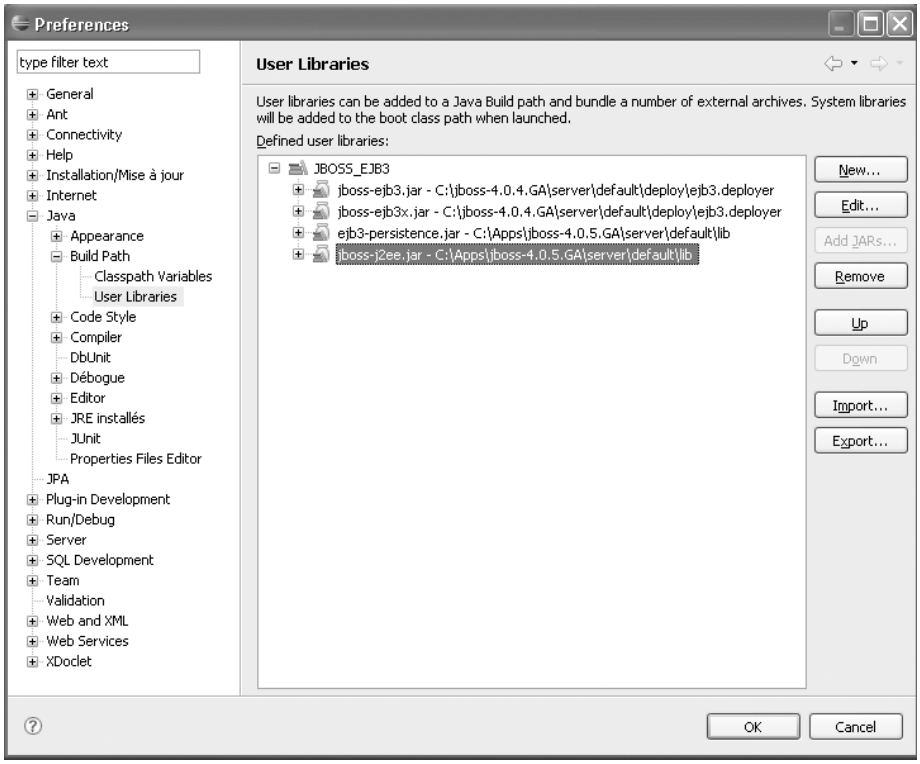


Figure 5.15

Configuration des bibliothèques JBoss de support aux technologies EJB3

3. Cliquez sur OK pour terminer l'opération. Vous intégrerez ensuite celle-ci à votre projet.

Sans anticiper sur les fonctionnalités du serveur JBoss et les configurations spécifiques aux projets EJB3, que nous présentons un peu plus loin dans cet ouvrage, cette étape de définition des bibliothèques spécifiques au projet est cruciale dans la configuration du poste de développement afin de permettre une standardisation des développements.

Passez à présent aux fonctionnalités spécifiques de support au développement Web apportées par le projet JST pour la configuration d'un projet Web.

> **Support des fichiers jar externes**
>
> L'ajout et la définition de bibliothèques externes au chemin de compilation du projet pour l'utilisation de l'API EJB3 n'est plus nécessaire avec les versions supérieures à JBoss 4.2.

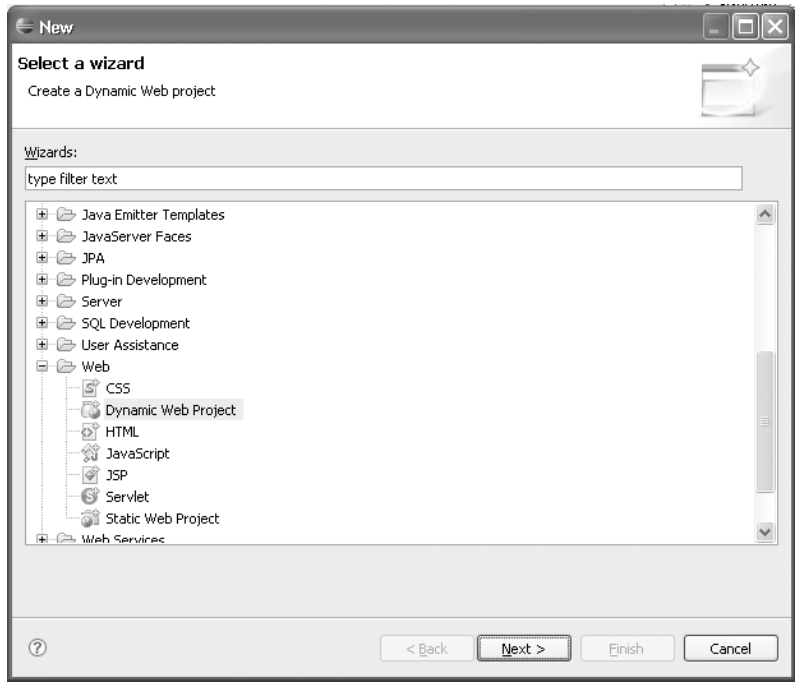
Configuration d'un projet de développement Web

Vous allez définir votre projet comme étant de type Dynamic Web, c'est-à-dire de support aux technologies JSP et servlets.

1. Sélectionnez l'assistant de création de projet Eclipse *via* Fichier, Nouveau, Autre et Dynamic Web Project (voir figure 5.16).

Figure 5.16

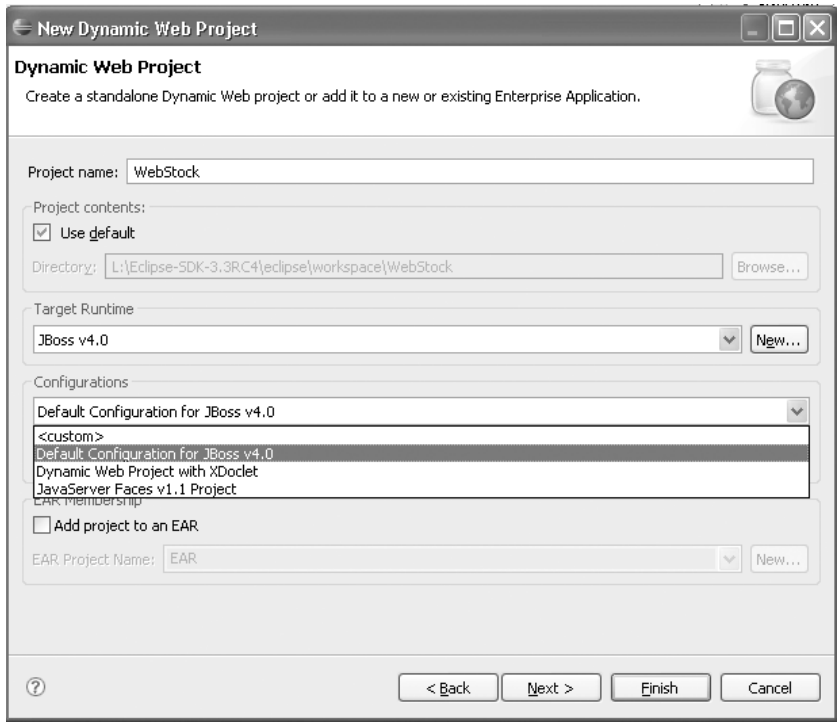
Assistant de création de projet Web dynamique



2. Cliquez sur Next, puis entrez WebStock devant le nom du projet (voir figure 5.17). Notez que l'environnement d'exécution cible du projet (champ Target Runtime) pointe sur la cible SA définie précédemment (JBoss 4.0).

Figure 5.17

Configuration du projet Web dynamique



Notez que, depuis Web Tools 2.0, trois types de configurations spécifiques sont proposés :

– Default Configuration for JBoss v4.0, par défaut pour le développement d'applications sur Jboss 4.0.

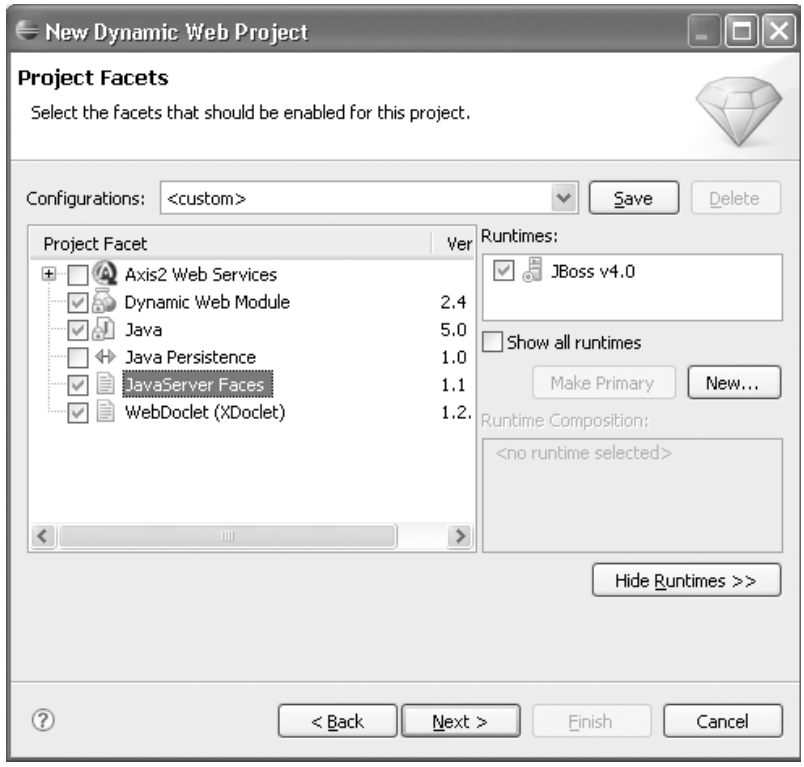
– Dynamic Web Project with XDoclet, pour le support des annotations avec le standard XDoclet.

– JavaServer Faces v1.1 Project, pour le support de l'implémentation de référence de la technologie JSF.

3. Choisissez la première option, puis cliquez sur Next (voir figure 5.18).

Figure 5.18

Configuration du projet Web dynamique avec l'option Project Facet



Project Facet

Comme évoqué plus haut, les fonctionnalités Project Facet permettent de gérer toute la variété de projets et de supports des différentes variétés de frameworks. Web Tools offre la possibilité d'ajouter et de supprimer dynamiquement ces fonctionnalités aux projets. Les Project Facets sont configurables lors de la création du projet et par la suite dans la page des propriétés du projet.

4. Sélectionnez les facets JavaServer Faces 1.1 et WebDoclet (XDoclet) 1.2, puis cliquez sur Next.

5. Dans la boîte de dialogue qui s'affiche, configurez le module Web :

– Context Root : permet de définir le chemin virtuel au sein duquel le contenu de l'application Web sera accédé sur le SA.

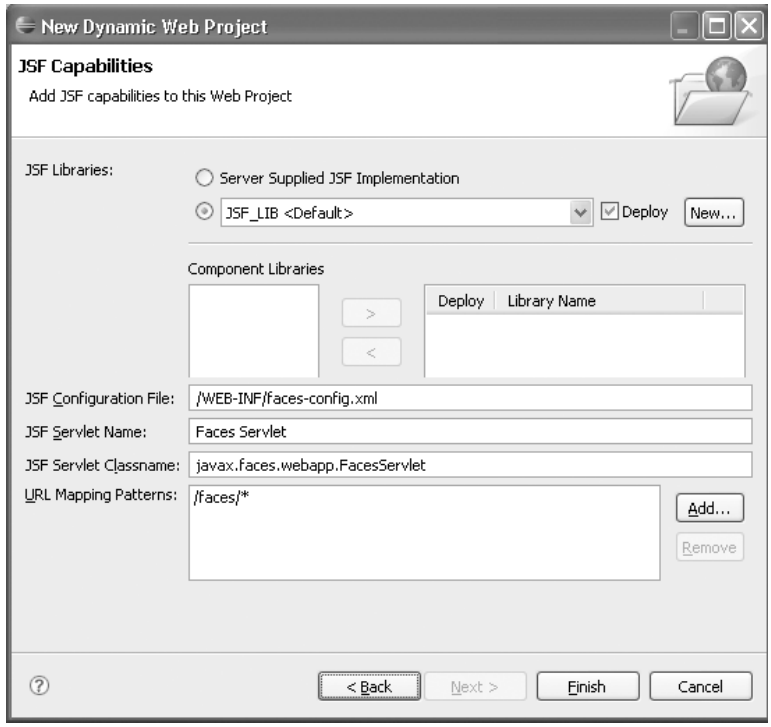
– Content Directory : répertoire contenant les artefacts Web (fichiers HTML, JSP, fichiers graphiques, etc.).

– Java Source Directory : répertoire contenant les sources des classes, des servlets et des beans. Lorsque ces ressources sont ajoutées au projet Web, celles-ci sont automatiquement compilées, et le résultat compilé est ajouté au répertoire WEB-INF/classes.

6. Cliquez sur Next. La page illustrée à la figure 5.19 s'affiche.

Figure 5.19

Options des Project Facets



Cet assistant configure les principaux fichiers de configuration JSF (faces-config.xml, répertoire de mapping des URL, etc.). Pour utiliser la fonctionnalité de référence de la technologie JavaServer Faces version 1.1, il faut télécharger l'implémentation de référence de cette technologie, à l'adresse *http://java.sun.com/javaee/javaserverfaces/download.html).*

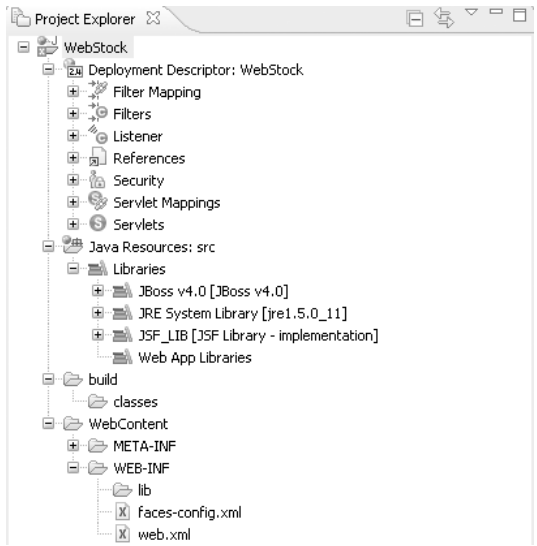
7. Cliquez sur New pour créer la bibliothèque et l'associer au jar téléchargé contenant les bibliothèques JSF 1.1.

8. Cliquez sur Finish pour terminer la configuration du projet Web dynamique, puis cliquez sur la nouvelle perspective fournie par JST, *via* Fenêtre, Ouvrir la perspective, Autre et Java JEE. Vous devez voir s'afficher l'arborescence de votre projet JEE comme illustré à la figure 5.20.

   La vue d'affichage des erreurs laisse apparaître un problème de configuration de l'outil de génération de code XDoclet, qui permet le développement orienté attribut (Attribut-Oriented Programming). Il a surtout conquis ses lettres de noblesse pour fournir, grâce aux très populaires outils de build Ant, une solution de génération multi-AS et multi-technologies J2EE (Struts, JSF, EJB, etc.).

Figure 5.20

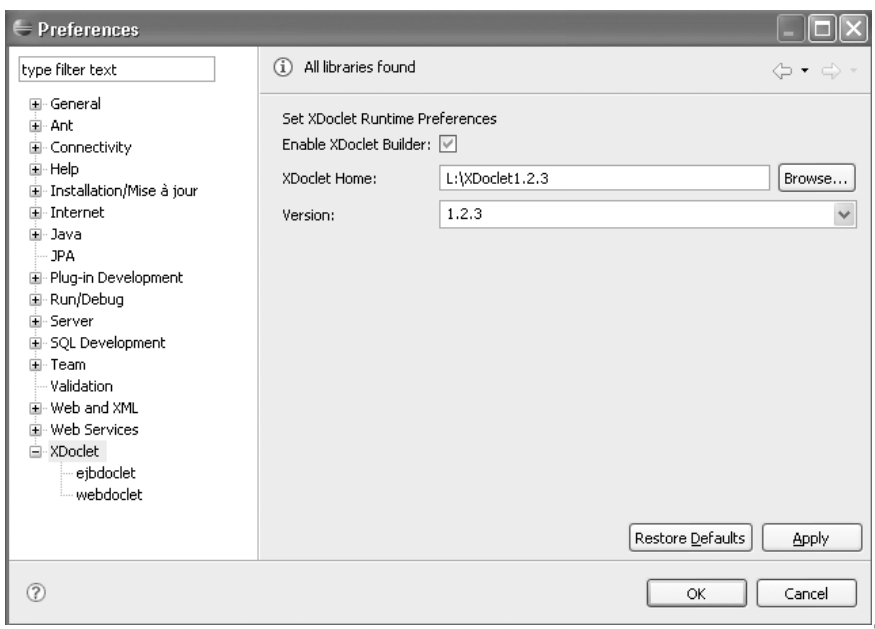
Arborescence du projet Web



9. Pour l'utiliser et l'intégrer dans Eclipse, téléchargez la dernière version de l'outil à partir du site de référence *http://xdoclet.sourceforge.net/*. À ce jour, la version la plus à jour est la 1.2.3.
10. Décompressez la distribution dans un répertoire dédiée (par exemple, L:\XDoclet1.2.3).
11. Faites pointer la version de XDoclet comme illustré à la figure 5.21.

Figure 5.21

Configuration de XDoclet



12. Cliquez sur OK, puis régénérez votre projet en sélectionnant Projet et Nettoyer.

Vous devez retrouver votre espace de travail exempt de toute erreur de compilation.

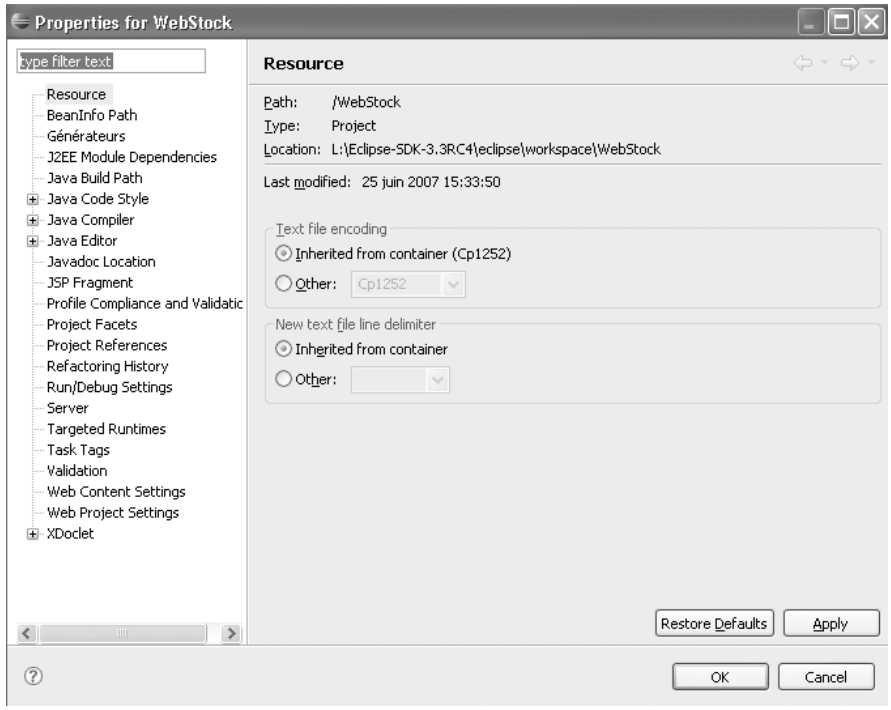
Propriétés du projet

Les projets J2EE sont créés avec des propriétés par défaut par l'assistant de création de projet fourni par JST. Ces propriétés sont généralement suffisantes pour la majorité des projets développés.

Il arrive toutefois que ces propriétés ne répondent pas au besoin des applications développées. JST fournit la possibilité de modifier les réglages du projet par le biais de l'option Properties du menu contextuel du projet en cours de développement (voir figure 5.22).

Figure 5.22

Propriétés du projet J2EE



Le tableau 5.4 récapitule les propriétés de configuration du projet J2EE.

Tableau 5.4 Propriétés de configuration du projet J2EE

Propriété	Description
Resource	Fournit un certain nombre d'informations sur le projet en cours, comme sa localisation sur le système de fichiers, la date de dernière modification, le type d'encodage, etc.
BeanInfo Path	Permet l'activation de l'introspection BeanInfo sur le projet.
Générateurs	Configure les générateurs du projet et l'ordre de leur invocation.
J2EE Module Dependencies	Module de gestion des dépendances (bibliothèque jar externe ou projet utilitaire en particulier)
Java Build Path	Indique la location de fichiers additionnels de projet ou de bibliothèques à inclure lors de la construction du projet. Gère également l'ordre de référencement de ces artefacts.

Tableau 5.4 Propriétés de configuration du projet J2EE *(suite)*

Propriété	Description
Java Code Style	Spécifie le style du code utilisé pour la génération de code pour appliquer des conventions de nommage, des règles de style et de commentaire.
Java Compiler	Spécifie les réglages pour le compilateur associé au projet Java.
Java Editor	Spécifie les actions à effectuer lors de la sauvegarde du code source.
Javadoc Location	Spécifie la localisation de la documentation au format javadoc du projet.
JSP Fragment	Définit l'encodage par défaut, la directive page et la valeur du type de contenu pour le fragment JSP (projet dynamique uniquement).
Profile Compliance and Validation	Spécifie le niveau d'interopérabilité WS-I et WS-I SSBP (Attachments Profile and Simple SOAP Binding Profile).
Project Facets	Permet de modifier les caractéristiques du projet.
Project References	Spécifie les projets à référencer.
Refactoring History	Historique de refactoring
Run/Debug Settings	Gère la configuration de lancement associée (Java Applet/Java application).
Server	Utilise le serveur spécifié lors du démarrage du projet.
Targeted Runtimes	Configuration serveur cible
Task Tags	Spécifie les balises qui seront utilisées pour indiquer les tâches.
Validation	Affiche les validateurs qui vont s'exécuter durant la validation du projet.
Web Content Settings	Propriétés par défaut du projet appliquées en particulier au type de document (xHTML 1.0 Strict/Frameset/MP 1.0/Basic 1.0)
Web Project Settings	Spécifie le contexte root du projet Web.
Xdoclet	Configure l'outil Xdoclet.

Structure et contenu du projet

Comme indiqué précédemment, un projet Eclipse est représenté dans le système de fichiers sous une forme arborescente, cette structure contenant l'ensemble des artefacts du projet.

Eclipse crée un certain nombre de fichiers au format XML au sein de cette arborescence afin de gérer l'information du projet.

Deux fichiers et un répertoire importants sont créés systématiquement lors de la création du projet J2EE. Ils peuvent être trouvés dans le système de fichiers à la racine du répertoire du projet J2EE :

- project : contient des informations concernant la nature, le générateur (builder) et des propriétés additionnelles définies dans le projet.
- classpath : contient le chemin d'accès aux différents jar et répertoires qui seront utilisés lors de la compilation du projet.
- setting : répertoire contenant les préférences du projet et sa configuration (facets utilisées, infos de localisation, etc.).

Édition

Si vous souhaitez éditer ces fichiers, ce qui n'est pas recommandé, faites en une copie.

Artefacts du projet et vues JEE

Une fois le projet créé avec Web Tools, de nouveaux artefacts peuvent être ajoutés.

Ces artefacts comprennent les composants servlets, les pages JSP et les EJB. JST fournit une perspective JEE spécialisée pour le support des projets JEE et des différents artefacts contenus au sein du projet. Par exemple, une application Web JEE contient généralement plusieurs fichiers qui nécessitent d'être modifiés, un serveur sur lequel les artefacts vont être déployés et une vue console pour afficher la sortie du serveur.

La perspective Java EE fournit un accès direct à un certain nombre de vues utiles, dont le tableau 5.4 récapitule les principales :

Tableau 5.4 Vues associées à la perspective JEE

Vue	Description
Project Explorer (Explorateur de projets)	Fournit une vue de type navigation pour parcourir les différents artefacts du projet JEE, chaque composant type étant représenté par une icône particulière (JSP, bibliothèque, etc.).
Structure	Affiche la structure des éléments contenus dans le document en cours d'édition en utilisant une vue arborescente.
Problems (Problèmes)	Liste les problèmes existants dans le projet en cours de modification (problèmes de compilation, etc.).
Tasks (Tâches)	Fournit une liste des éléments assignés sous forme de tâches ou de « pense bête » pour le développeur (ajout de commentaire, etc.).
Propriétés	Permet d'inspecter et modifier les propriétés de l'élément courant sélectionné.
Servers	Permet de contrôler les opérations de l'instance serveur configurée (arrêt/relance, etc.).

Des vues supplémentaires peuvent être ouvertes en sélectionnant les options Window, Show View et Other d'Eclipse puis en sélectionnant la vue requise. La nouvelle vue s'ouvre à côté des vues existantes de la perspective JEE et les complémente de manière personnalisée.

Il est ensuite possible de sauvegarder ces vues personnalisées par le biais Window et Save Perspective As pour les réutiliser ensuite.

La synchronisation s'opère immédiatement entre chaque vue JEE et la source du document dans l'éditeur.

En résumé

Ce chapitre vous a permis de vous familiariser avec l'outillage de support au développement JEE inclus dans WTP et les différentes vues et propriétés permettant de gérer ce type de projet.

Le chapitre suivant vous fournira l'occasion de faire connaissance avec un autre projet utile au développement Web et au support des données, le projet Data Tools.