



Guide du développeur d'applications web



VERSION 5

Borland®
JBuilder™

Consultez le fichier DEPLOY.TXT situé dans le répertoire `redist` de JBuilder pour avoir la liste complète des fichiers que vous pouvez distribuer en accord avec votre contrat de licence JBuilder.

Les applications mentionnées dans ce manuel sont brevetées par Borland Software Corporation ou en attente de brevet. Ce document ne donne aucun droit sur ces brevets.

COPYRIGHT © 1997, 2001 Borland Software Corporation. Tous droits réservés. Tous les produits Borland sont des marques commerciales ou des marques déposées de Borland Software Corporation aux Etats-Unis et dans les autres pays. Les autres noms de produits sont des marques commerciales ou des marques déposées de leurs fabricants respectifs.

Pour connaître les conditions et limites des fournisseurs tiers, lisez les Remarques version sur votre CD JBuilder.

JBE0050WW21004 1E0R0401
0102030405-9 8 7 6 5 4 3 2 1
PDF

Table des matières

Chapitre 1

Développement d'applications web 1-1

Comment contacter le support développeur de Borland	1-3
Ressources en ligne.	1-4
World Wide Web	1-4
Groupes de discussion Borland.	1-4
Groupes de discussion Usenet	1-5
Rapports de bogues	1-5
Conventions de la documentation	1-6
Conventions pour le Macintosh	1-7

Chapitre 2

Présentation générale du processus de développement des applications web 2-1

Applets	2-2
Servlets	2-3
Pages JavaServer (JSP).	2-4
InternetBeans Express	2-5
Choix des technologies à utiliser dans votre application web	2-5
Présentation générale du processus de développement des applications web	2-6
Applications web et applications distribuées.	2-7

Chapitre 3

Utilisation des WebApps et des fichiers WAR 3-1

La WebApp	3-1
Fichiers archive web (WAR)	3-2
Outils pour travailler avec les WebApps et les fichiers WAR	3-2
L'expert application web	3-3
La WebApp et ses propriétés	3-4
Répertoire racine	3-5
Descripteurs de déploiement	3-5
Propriétés de la WebApp	3-6
L'onglet WebApp	3-6
L'onglet Classes	3-7
L'onglet Dépendances	3-9
L'onglet Manifest	3-10
Le fichier WAR	3-11
Applets dans un fichier WAR.	3-12

Chapitre 4

Utilisation des applets 4-1

Fonctionnement des applets	4-2
La balise <applet>	4-2
Exemple de balise <applet>	4-2
Attributs de la balise <applet>.	4-3
Erreurs couramment commises dans la balise <applet>	4-4
Questions liées au navigateur	4-5
Support Java	4-5
Fourniture du navigateur préféré à l'utilisateur final	4-6
Support de plusieurs navigateurs	4-6
Différences dans l'implémentation Java	4-6
Solutions aux questions liées aux navigateurs.	4-7
Conseils supplémentaires pour faire fonctionner les applets	4-9
La sécurité et le gestionnaire de sécurité	4-11
Le sas de sécurité	4-11
Restrictions des applets.	4-12
Solutions aux problèmes de sécurité	4-12
Utilisation de bibliothèques tierces	4-14
Déploiement des applets.	4-14
Test des applets	4-15
Etapes de test élémentaires.	4-15
Test des navigateurs.	4-16
JBuilder et les applets.	4-17
Création d'applets avec l'expert applet.	4-17
Exécution d'applets	4-20
Applet TestBed de JBuilder et appletviewer de Sun	4-21
Exécution des applets JDK 1.1.x dans JBuilder	4-22
Exécution des applets JDK 1.2 dans JBuilder	4-22
Débogage des applets.	4-22
Débogage des applets dans le module de mise à jour Java	4-23
Déploiement des applets dans JBuilder.	4-25

Chapitre 5

Utilisation des servlets 5-1

Servlets et JSP	5-3
Servlets et serveurs web	5-3
L'API des servlets	5-4
La paquet servlet.HTTP	5-5

Le cycle de vie des servlets	5-6
Construction et initialisation du servlet	5-6
Gestion des demandes client	5-7
Servlets et multithread	5-7
Destruction d'un servlet	5-7
Servlets orientés HTML	5-8
Servlets spécifiques à HTTP	5-8
Comment sont utilisés les servlets ?	5-9
Déploiement des servlets	5-9
Chapitre 6	
Création de servlets dans JBuilder	6-1
Options de l'expert servlet	6-1
Expert servlet – Page Choisir un nom et un type de servlet	6-1
Expert servlet – Page Modifier les détails du servlet standard	6-3
Option Générer le type de contenu	6-4
Options Implémenter les méthodes	6-5
Options Détails du fichier SHTML	6-6
Expert servlet – Page des options de nom et d'affectation	6-6
Expert servlet – Page Paramètres	6-8
Expert servlet – Page Saisie des détails du servlet de l'auditeur	6-9
Invocation des servlets	6-10
Invocation d'un servlet depuis la fenêtre d'un navigateur	6-10
Invocation d'un servlet depuis une page HTML	6-11
Internationalisation des servlets	6-12
Ecriture d'un servlet orienté données	6-12
Chapitre 7	
Tutoriel :	
Création d'un servlet simple	7-1
Etape 1 : Création du projet	7-2
Etape 2 : Création de la WebApp	7-3
Etape 3 : Création du servlet avec l'expert servlet	7-4
Etape 4 : Ajout de code au servlet	7-7
Etape 5 : Compilation et exécution du servlet .	7-7
Chapitre 8	
Tutoriel : Crédation d'un servlet mettant à jour un guestbook	8-1
Etape 1 : Crédation du projet	8-2
Etape 2 : Crédation de la WebApp	8-3
Etape 3 : Crédation des servlets	8-4
Etape 4 : Crédation du module de données	8-8
Etape 5 : Ajout des composants de base de données au module de données	8-9
Etape 6 : Crédation de la connexion de données au DBServlet	8-13
Etape 7 : Ajout d'un formulaire de saisie à FormServlet	8-13
Etape 8 : Ajout du code connectant DBServlet au module de données	8-14
Etape 9 : Ajout du code affichant la table SIGNATURES du Guestbook	8-15
Que fait la méthode doGet() ?	8-16
Etape 10 : Ajout de la logique métier au module de données	8-18
Etape 11 : Compilation et exécution de votre projet	8-19
Chapitre 9	
Développement	
des pages JavaServer	9-1
L'API JSP	9-3
JSP dans JBuilder	9-4
L'expert JSP	9-4
Développement d'une JSP	9-4
Compilation d'une JSP	9-5
Exécution d'une JSP	9-5
Débogage d'une JSP	9-5
Déploiement d'une JSP	9-5
Ressources JSP supplémentaires	9-5
Chapitre 10	
Tutoriel : Crédation d'une JSP en utilisant l'expert JSP	10-1
Etape 1 : Crédation d'un nouveau projet	10-2
Etape 2 : Crédation d'une nouvelle WebApp . .	10-2
Etape 3 : Utilisation de l'expert JSP	10-3
Etape 4 : Ajout de fonctionnalités au JavaBean	10-4

Etape 5 : Modification du code JSP	10-5	
Etape 6 : Exécution de la JSP	10-6	
Utilisation de la Vue Web	10-8	
Débogage de la JSP	10-8	
Déploiement de la JSP	10-8	
Chapitre 11		
Utilisation d'InternetBeans Express	11-1	
Présentation des classes InternetBeans Express	11-2	
Utilisation d'InternetBeans Express		
avec les servlets.	11-3	
Affichage de pages web dynamiques avec des servlets utilisant InternetBeans Express . .	11-3	
Enregistrement (POST) des données avec des servlets utilisant InternetBeans Express . .	11-5	
Analyse des pages	11-5	
Génération de tableaux	11-6	
Utilisation d'InternetBeans Express		
avec les JSP	11-6	
Tableau des balises InternetBeans	11-9	
Format de internetbeans.tld.	11-10	
Chapitre 12		
Tutoriel : Création d'un servlet avec InternetBeans Express	12-1	
Etape 1 : Création d'un nouveau projet	12-2	
Etape 2 : Création d'une nouvelle WebApp. . .	12-2	
Etape 3 : Utilisation de l'expert servlet	12-3	
Etape 4 : Création du module de données . . .	12-5	
Etape 5 : Conception du modèle de la page HTML.	12-6	
Etape 6 : Connexion du servlet au module de données	12-8	
Etape 7 : Conception du servlet	12-9	
Etape 8 : Modification du servlet.	12-11	
Etape 9 : Exécution du servlet	12-11	
Déploiement du servlet	12-12	
Chapitre 13		
Tutoriel : Création d'une JSP avec InternetBeans Express	13-1	
Etape 1 : Création d'un nouveau projet	13-2	
Etape 2 : Création d'une nouvelle WebApp. . .	13-2	
Etape 3 : Utilisation de l'expert JSP	13-3	
Etape 4 : Conception de la partie HTML de la JSP	13-4	
Etape 5 : Ajout de la balise InternetBeans database	13-5	
Etape 6 : Ajout de la balise InternetBeans query	13-6	
Etape 7 : Ajout de la balise InternetBeans table	13-6	
Etape 8 : Ajout des balises InternetBeans control	13-7	
Etape 9 : Ajout de la balise InternetBeans submit	13-8	
Etape 10 : Ajout de la méthode submitPerformed()	13-8	
Etape 11 : Ajout de code pour insérer une ligne	13-9	
Etape 12 : Ajout de la bibliothèque JDataStore Server à votre projet.	13-9	
Etape 13 : Exécution de la JSP	13-10	
Déploiement de la JSP.	13-11	
Chapitre 14		
Configuration de votre serveur web	14-1	
Configuration de Tomcat.	14-2	
Configuration de JBuilder pour des serveurs web autres que Tomcat.	14-5	
Configuration de JBuilder pour des serveurs web autres que Tomcat (utilisateurs de JBuilder Entreprise)	14-5	
Configuration de JBuilder pour des serveurs web autres que Tomcat (utilisateurs de JBuilder Professionnel)	14-7	
Configuration du serveur web sélectionné. . .	14-7	
Définition des options de vue web	14-8	
Définition des options d'exécution web . .	14-9	
Création du plugin de votre serveur web . . .	14-10	
Recensement d'un OpenTool.	14-11	
Configuration du serveur web.	14-11	
Démarrage et arrêt du serveur web	14-11	
Considérations sur JSP	14-12	
Editeur GUI de descripteur de déploiement	14-12	
Chapitre 15		
Utilisation des applications web dans JBuilder	15-1	
Compilation de votre servlet ou de votre JSP .	15-2	
Comment les URL exécutent les servlets . . .	15-3	
Exécution de votre servlet ou de votre JSP. .	15-5	
Démarrage de votre serveur web	15-6	
Vue Web.	15-7	

Source Vue Web	15-8	Page Références de ressource.	16-15
Arrêt du serveur web	15-8	Page Références EJB	16-15
Activation des commandes web	15-8	Page Connexion	16-16
Définition des paramètres d'exécution de votre servlet ou de votre JSP	15-9	Page Sécurité	16-17
Définition des propriétés d'exécution d'un servlet	15-12	Modification des descripteurs de déploiement spécifiques au fournisseur	16-19
Débogage de votre servlet ou de votre JSP . .	15-13	Informations supplémentaires sur les descripteurs de déploiement . . .	16-20
Chapitre 16			
Déploiement			
de votre application web		16-1	17-1
Présentation	16-1	Observations sur les applications	
Constructeur d'archives	16-1	Java Web Start	17-2
Descripteurs de déploiement	16-2	Installation de Java Web Start	17-3
Applets	16-2	Java Web Start et JBuilder	17-3
Servlets	16-2	Le fichier JAR de l'application	17-4
JSP	16-3	Le fichier JNLP et la page d'accueil de l'application	17-4
Test de votre application web	16-4	Tutoriel : Exécution de l'application exemple	
Descripteurs de déploiement	16-4	CheckBoxControl avec Java Web Start	17-6
L'éditeur DD WebApp	16-4	Etape 1 : Ouverture et configuration du projet	17-6
Menu contextuel de l'éditeur		Etape 2 : Création de la WebApp de l'application	17-7
DD WebApp	16-5	Etape 3 : Création du fichier JAR de l'application	17-8
Page Descripteur de déploiement		Etape 4 : Création de la page d'accueil et du fichier JNLP de l'application	17-9
WebApp	16-6	Etape 5 : Lancement de l'application	17-12
Page Paramètres de contexte	16-7		
Page Filtres	16-7		
Page Auditeurs	16-9		
Page Servlets	16-10		
Page Bibliothèques de balises	16-12		
Page Types MÎME.	16-13		
Page Pages erreur	16-13		
Page Environnement	16-14		
Index		I-1	

Développement d'applications web

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Le développement des applets est une fonctionnalité de toutes les éditions de JBuilder.

Le *Guide du développeur d'applications web* présente certaines des technologies disponibles pour le développement d'applications multiniveaux destinées au web. Une application web est une collection de documents HTML/XML, de composants web (servlets et pages JavaServer) et d'autres ressources, dans une structure de répertoires ou sous forme d'une archive appelée fichier WAR (Web ARchive). Une application web réside sur un serveur central et fournit des services à une variété de clients.

Ce manuel décrit en détail comment ces technologies apparaissent dans JBuilder et comment vous travaillerez avec elles dans l'EDI et l'éditeur. Il explique également comment ces technologies s'ajustent dans une application web. Pour avoir plus d'informations, choisissez l'un des chapitres suivants :

- Chapitre 2, "Présentation générale du processus de développement des applications web"

Introduit les technologies traitées dans ce manuel, notamment les applets, les servlets, les JavaServer Pages (JSP) et InternetBeans Express.

- Chapitre 3, "Utilisation des WebApps et des fichiers WAR"

Explique comment créer une application web et l'archiver dans un fichier WAR depuis JBuilder. Ce chapitre expose également les concepts généraux et la structure WebApp.

- Chapitre 4, "Utilisation des applets"
Explique comment créer des applets dans JBuilder et les déployer sur un serveur web. Aborde les principaux problèmes posés par le développement et le déploiement des applets et présente les solutions.
- Chapitre 5, "Utilisation des servlets"
Présente les servlets et l'API servlet.
- Chapitre 6, "Création de servlets dans JBuilder"
Explique les options de l'expert servlet, comment exécuter les servlets, comment les internationaliser et comment créer des servlets orientés données.
- Chapitre 7, "Tutoriel : Crédation d'un servlet simple"
Vous guide pas à pas dans l'écriture d'un servlet simple acceptant la saisie de l'utilisateur et comptant le nombre de visiteurs d'un site.
- Chapitre 8, "Tutoriel : Crédation d'un servlet mettant à jour un guestbook"
Vous guide pas à pas dans l'écriture d'un servlet se connectant à une base de données JDataStore, acceptant la saisie de l'utilisateur et enregistrant celle-ci en retour dans la base.
- Chapitre 9, "Développement des pages JavaServer"
Présente les JSP et l'API JSP. Explique comment utiliser l'expert JSP pour créer une JSP.
- Chapitre 10, "Tutoriel : Crédation d'une JSP en utilisant l'expert JSP"
Vous guide pas à pas dans l'écriture d'une JSP acceptant et affichant la saisie de l'utilisateur et comptant le nombre de fois où une page web a été consultée.
- Chapitre 11, "Utilisation d'InternetBeans Express"
Explique la bibliothèque InternetBeans et comment utiliser les composants avec des servlets et des JSP.
- Chapitre 12, "Tutoriel : Crédation d'un servlet avec InternetBeans Express"
Vous guide pas à pas dans l'écriture d'un servlet utilisant les composants InternetBeans pour interroger une table de base de données, afficher son contenu, accepter la saisie de l'utilisateur et l'enregistrer en retour dans la base.
- Chapitre 13, "Tutoriel : Crédation d'une JSP avec InternetBeans Express"
Vous guide pas à pas dans l'écriture d'une JSP utilisant les composants InternetBeans pour interroger une table de base de données, afficher son contenu, accepter la saisie de l'utilisateur et l'enregistrer en retour dans la base.

- Chapitre 14, "Configuration de votre serveur web"
Explique comment configurer votre serveur web pour l'exécuter dans JBuilder.
- Chapitre 15, "Utilisation des applications web dans JBuilder"
Explique comment compiler, exécuter, déboguer des servlets et des JSP.
- Chapitre 16, "Déploiement de votre application web"
Explique comment gérer les descripteurs de déploiement de votre application web, utiliser l'éditeur de descripteur de déploiement de JBuilder et déployer votre application web.
- Chapitre 17, "Lancement de votre application web avec Java Web Start"
Explique comment utiliser Web Start pour lancer des applications non web à partir d'un navigateur web.

Ce document contient de nombreux liens vers des sites Web externes. Ces adresses et ces liens étaient valables au moment de l'impression de ce manuel. Borland ne contrôle pas ces sites et n'est pas responsable de leur contenu ni de leur pérennité.

Si vous avez des questions spécifiques au développement des applications web dans JBuilder, envoyez-les au groupe de discussion Servlet-JSP, borland.public.jbuilder.servlet-jsp, à l'adresse <http://www.borland.fr/Newsgroups/>.

Comment contacter le support développeur de Borland

Borland offre aux développeurs diverses options de support. Elles comprennent des services gratuits sur Internet, où vous pouvez consulter notre importante base d'informations et entrer en contact avec d'autres utilisateurs de produits Borland. En outre, vous pouvez choisir parmi plusieurs catégories de support, allant de l'installation des produits Borland au support tarifé de niveau consultant, en passant par une assistance complète.

Pour obtenir des informations sur les services Borland de support aux développeurs, veuillez consulter notre site Web, à l'adresse <http://www.borland.fr/Support/>.

Quand vous contacterez le support, préparez des informations complètes sur votre environnement, la version du produit que vous utilisez et une description détaillée du problème.

Pour avoir de l'aide sur les outils tiers, ou leur documentation, contactez votre fournisseur.

Ressources en ligne

Vous pouvez obtenir des informations depuis les sources ci-après :

World Wide Web <http://www.borland.fr/>

FTP [ftp.borland.com](ftp://ftp.borland.com)
Documents techniques accessibles par anonymous ftp.

Listserv Pour vous abonner aux bulletins électroniques, utilisez
le formulaire en ligne :
<http://www.borland.com/contact/listserv.html>

ou, pour l'international,
<http://www.borland.com/contact/intlist.html>

TECHFAX 1-800-822-4269 (Amérique du Nord)
Documents techniques accessibles par télécopie.

World Wide Web

Consultez régulièrement www.borland.fr/jbuilder. L'équipe produit de JBuilder y place notes techniques, analyses des produits concurrents, réponses aux questions fréquemment posées, exemples d'applications, mises à jour du logiciel et informations sur les produits existants ou nouveaux.

Vous pouvez vous connecter en particulier aux URL suivantes :

- <http://www.borland.fr/Produits/jbuilder/> (mises à jour du logiciel et autres fichiers)
- <http://community.borland.com/> (contient notre magazine d'informations web pour les développeurs)

Groupes de discussion Borland

Vous pouvez vous inscrire à JBuilder et participer à de nombreux groupes de discussion dédiés à JBuilder.

Vous trouverez des groupes de discussion, animés par les utilisateurs, pour JBuilder et d'autres produits Borland, à l'adresse
<http://www.borland.fr/Newsgroups/>

Groupes de discussion Usenet

Les groupes Usenet suivants sont dédiées à Java et concernent la programmation :

- news:comp.lang.java.advocacy
- news:comp.lang.java.announce
- news:comp.lang.java.beans
- news:comp.lang.java.databases
- news:comp.lang.java.gui
- news:comp.lang.java.help
- news:comp.lang.java.machine
- news:comp.lang.java.programmer
- news:comp.lang.java.security
- news:comp.lang.java.softwaretools

Remarque Ces groupes de discussion sont maintenus par les utilisateurs et ne sont pas des sites Borland officiels.

Rapports de bogues

Si vous pensez avoir trouvé un bogue dans le logiciel, merci de le signaler dans la page du support développeur de JBuilder, à l'adresse <http://www.borland.fr/Support/jbuilder/>. Sur ce site, vous pouvez également demander une fonctionnalité ou consulter la liste des bogues déjà signalés.

Quand vous signalez un bogue, indiquez toutes les étapes nécessaires à la reproduction de ce bogue, ainsi que tout paramètre spécial de votre environnement et les autres programmes utilisés avec JBuilder. Précisez bien le comportement attendu et ce qui s'est réellement passé.

Si vous avez des commentaires (compliments, suggestions ou questions) concernant la documentation de JBuilder, vous pouvez envoyer un e-mail à jgpubs@borland.com. Uniquement pour la documentation. Les questions de support doivent être adressées au support développeur.

JBuilder est fait par des développeurs pour des développeurs. Nous apprécions vraiment vos remarques, car elles nous aident à améliorer notre produit.

Conventions de la documentation

La documentation Borland sur JBuilder utilise les polices et les symboles décrits dans le tableau ci-dessous pour signaler du texte particulier.

Tableau 1.1 Polices et symboles

Police	Signification
Police à espacement fixe	<p>La police à espacement fixe représente :</p> <ul style="list-style-type: none"> • du texte tel qu'il apparaît à l'écran • du texte que vous devez taper, comme "Entrez Hello World dans le champ Titre de l'expert Application." • des noms de fichiers • des noms de chemins • des noms de répertoires ou de dossiers • des commandes, comme <code>SET PATH, CLASSPATH</code> • du code Java • des types de données Java, comme <code>boolean, int et long</code>. • des identificateurs Java, comme des noms de variables, classes, interfaces, composants, propriétés, méthodes et événements • des noms de paquets • des noms d'argument • des noms de champs • des mots clés Java, comme <code>void et static</code>
Gras	Le gras est utilisé pour désigner les outils <code>java</code> , <code>bmj</code> (Borland Make for Java), <code>bij</code> (Borland Compiler for Java) et les options du compilateur. Par exemple : javac, bmj, -classpath .
<i>Italiques</i>	L'italique est utilisé pour les termes nouveaux, les titres des manuels et, parfois, pour la mise en valeur.
Touches	Cette police indique une touche de votre clavier. Par exemple, "Appuyez sur <i>Echap</i> pour quitter un menu."
[]	Les crochets droits dans le texte ou dans la syntaxe entourent les éléments facultatifs. Ne tapez pas ces crochets.
< >	Les crochets angulaires dans le texte ou dans la syntaxe indiquent une chaîne variable ; entrez la chaîne appropriée à votre code. Ne tapez pas ces crochets. Les crochets angulaires sont également utilisés pour les balises HTML.
...	Dans les exemples de code, les points de suspension indiquent du code manquant. Sur un bouton, les points de suspension indiquent que ce bouton ouvre un dialogue de sélection.

JBuilder est disponible sur plusieurs plates-formes. Reportez-vous au tableau ci-dessous pour une description des conventions et répertoires associés aux diverses plates-formes dans la documentation.

Tableau 1.2 Conventions et répertoires associés aux plates-formes

Elément	Signification
Chemins	<p>Les chemins d'accès mentionnés dans la documentation sont indiqués par une barre oblique (/).</p> <p>Pour la plate-forme Windows, utilisez une barre oblique inverse (\).</p>
Répertoire de base	<p>L'emplacement du répertoire de base dépend de la plate-forme.</p> <ul style="list-style-type: none"> • Pour UNIX et Linux, le répertoire de base peut varier. Par exemple, ce peut être /user/[username] ou /home/[username] • Pour Windows 95/98, le répertoire de base est C:\Windows • Pour Windows NT, le répertoire de base est C:\Winnt\Profiles\[username] • Pour Windows 2000, le répertoire de base est C:\Documents and Settings\[username]
Répertoire .jbuilder5	Le répertoire .jbuilder5, où sont stockés les paramètres de fonctionnement de JBuilder, est placé dans le répertoire de base.
Répertoire jbproject	Le répertoire jbproject, qui contient les fichiers projet, classe et source, est placé dans le répertoire de base. JBuilder enregistre les fichiers dans ce chemin par défaut.

Conventions pour le Macintosh

JBuilder a été conçu pour supporter le Macintosh OS X si étroitement qu'il prend l'apparence d'une application native. La plate-forme Macintosh a des conventions d'apparence et de style qui diffèrent de celles de JBuilder ; quand cela se produit, JBuilder prend l'apparence du Mac. Cela veut dire qu'il y a quelques différences entre la façon dont JBuilder se présente sur le Mac et ce qui est décrit dans la documentation. Par exemple, la documentation utilise le mot "répertoire" alors que sur le Mac, on dit "dossier". Pour plus d'informations sur les chemins, la terminologie et les conventions de l'interface utilisateur de Macintosh OS X, consultez la documentation livrée avec votre OS X.

Présentation générale du processus de développement des applications web

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Le développement des applets est une fonctionnalité de toutes les éditions de JBuilder.

Cette section présente les technologies mises en œuvre par les applications web, expose les points qui les différencient et explique comment choisir entre elles. Nous allons commencer par un résumé des technologies présentées dans ce manuel :

Tableau 2.1 Technologies des applications web

Technologie	Description
Applets	Un type d'application Java spécialisé pouvant être téléchargé par un navigateur client et exécuté sur la machine client.
Servlets	Une application Java côté serveur pouvant traiter les requêtes issues des clients.
Pages JavaServer (JSP)	Une extension de la technologie des servlets. Les pages JavaServer offrent essentiellement une façon simplifiée de développer des servlets. Elles semblent différentes pendant le développement mais, une fois exécutées, elles sont compilées en servlets par le serveur web.
InternetBeans Express	Une bibliothèque de composants qui fournit un moyen simple de présenter et de manipuler les données d'une base de données. Cette technologie est utilisée avec la technologie des servlets ou des JSP pour simplifier le développement des servlets ou des JSP orientés données.

Le résumé vous donne une idée de la nature de chacune de ces technologies, mais comment allez-vous choisir entre elles ? Quels sont les avantages et les inconvénients de chacune ? Nous allons répondre à ces questions, ainsi qu'à d'autres, dans la suite de ce document.

Applets

Il y a eu beaucoup d'agitation autour des applets lorsqu'est apparu le langage Java. La technologie Web était alors moins développée et les applets pouvaient passer comme la solution à une partie des problèmes qui se posaient aux développeurs à ce moment là. En fait, les applets sont devenues si populaires qu'aujourd'hui presque toutes les formations Java commencent par vous faire développer une applet. Il s'en suit que beaucoup de développeurs Java commettent l'erreur de trop compter sur les applets. Les applets ont certes leur place mais ne peuvent constituer la solution magique à vos problèmes de développement pour le web.

Parmi les inconvénients des applets, citons :

- Le déploiement et le test peuvent s'avérer difficiles.
- Vous êtes dépendant de la machine client sur laquelle se trouve le navigateur Java.
- Des navigateurs différents supportent des versions du JDK différentes, et généralement pas la dernière.
- L'applet peut être lente à démarrer la première fois, car elle a besoin d'être téléchargée par le client.

Certains de ces problèmes ont de réelles solutions que nous présentons en détail au Chapitre 4, "Utilisation des applets". Lorsque vous envisagez d'utiliser une applet, cherchez si une autre technologie Java ne pourrait pas vous rendre le même service en mieux.

Il y a cependant quelques avantages à utiliser les applets. Parmi lesquels :

- Les applets peuvent présenter des interfaces utilisateur plus complexes que les servlets ou les JSP.
- Comme les applets sont téléchargées et exécutées côté client, le serveur web n'a pas besoin de prendre en charge Java. Cela peut être important si vous écrivez une application web pour un site où vous n'avez pas le contrôle du serveur web (par exemple, un site hébergé par un ISP extérieur).
- Les applets peuvent valider en local les données saisies par l'utilisateur, ce qui évite la validation côté serveur. Vous pourriez également accomplir cette tâche avec du JavaScript et un servlet ou une JSP.
- Après le téléchargement initial de l'applet, le nombre de requêtes du navigateur au serveur peut être réduit, un bon nombre de traitements pouvant s'accomplir sur la machine client.

Pour plus d'informations sur les applets et la résolution des problèmes qui peuvent se poser, voir Chapitre 4, "Utilisation des applets".

Servlets

Les servlets sont des programmes Java qui s'intègrent à un serveur web pour fournir le traitement côté serveur des requêtes issues d'un navigateur client. Ils ont besoin d'un serveur web supportant la technologie JavaServer, par exemple le serveur Tomcat livré avec JBuilder. (Tomcat peut aussi être intégré à des serveurs web ne supportant pas la technologie JavaServer pour leur fournir la prise en charge de cette technologie. IIS est un exemple d'un tel serveur.) Les servlets Java peuvent remplacer des programmes CGI (Common Gateway Interface), ou être utilisés dans les situations où auparavant vous auriez utilisé CGI. Il y a cependant quelques avantages à utiliser les servlets plutôt que CGI :

- Réduction de la charge mémoire
- Indépendance par rapport aux plates-formes
- Indépendance par rapport aux protocoles

Vous utilisez un programme client écrit dans n'importe quel langage pour envoyer vos demandes au servlet. Le client peut être une simple page HTML. Vous pourriez utiliser une applet comme client, ou un programme écrit dans un autre langage que Java. Côté serveur, le servlet traite la requête et génère une sortie dynamique qui est retournée au client. Généralement, les servlets n'ont pas d'interface utilisateur, mais vous pouvez en fournir une côté client. Les servlets présentent quelques avantages par rapport aux applets :

- Vous n'avez pas besoin de savoir sur quel JDK le navigateur client va s'exécuter. Java n'a même pas besoin d'être activé sur le navigateur client. Tout le code Java est exécuté côté serveur. Cela apporte le maximum de contrôle à l'administrateur du serveur.
- Après le démarrage du servlet, les requêtes issues des navigateurs client invoquent tout simplement la méthode `service()` du servlet qui s'exécute. Cela signifie que ces clients ne ressentent pas d'effet sur leurs performances lorsqu'ils attendent que le servlet se charge. Comparez cela au chargement d'une applet.

Le déploiement d'un servlet sur le serveur web peut s'avérer difficile, mais certainement pas impossible. JBuilder propose des outils qui facilitent le déploiement. Il sont présentés au Chapitre 16, "Déploiement de votre application web".

Pour plus d'informations sur l'exécution les servlets Java, voir Chapitre 5, "Utilisation des servlets", et Chapitre 6, "Création de servlets dans JBuilder".

Pages JavaServer (JSP)

Les pages JavaServer sont une extension de la technologie des servlets. C'est essentiellement un moyen simplifié d'écrire des servlets, avec une attention plus forte accordée à la présentation de votre application.

La principale différence entre servlets et JSP est la suivante : avec les servlets, la logique de l'application est dans un fichier Java et totalement séparée du HTML dans la couche présentation. Avec les JSP, Java et HTML sont combinés dans un même fichier dont l'extension est .jsp.

Lorsque le serveur web traite le fichier JSP, un servlet est réellement généré mais, en tant que développeur, vous ne verrez pas ce servlet généré. En fait, lorsque vous compilez et exécutez votre JSP dans l'EDI de JBuilder, vous risquez de voir apparaître des exceptions ou des erreurs qui se produisent en fait dans le servlet généré. Il peut s'avérer difficile de distinguer quelle partie de votre JSP pose problème lorsque le message d'erreur fait référence à une ligne appartenant au code généré.

Les JSP présentent et des avantages et des inconvénients par rapport aux servlets. Parmi les avantages des JSP, citons :

- Moins de code à écrire.
- Facilité d'intégration des Java Beans existants.
- Déploiement plus facile. Prise en charge automatique d'un nombre supérieur de problèmes de déploiement, car les JSP établissent la correspondance avec un serveur web comme le font les fichiers HTML.
- Vous n'avez pas à imbriquer le code HTML que vous prévoyez de faire générer par le servlet dans votre code Java. A la place, des blocs individuels de code Java sont intégrés dans le HTML. Si vous planifiez bien votre travail, ces blocs de code Java peuvent être clairement séparés du code HTML dans le fichier, ce qui rend la JSP plus lisible.

Parmi les inconvénients des JSP, citons :

- La non visibilité du code du servlet généré peut engendrer une certaine confusion, comme indiqué précédemment.
- Le code HTML et le code Java n'étant pas dans des fichiers séparés, un développeur Java et un concepteur web travaillant ensemble sur une application doivent faire très attention à ne pas écraser mutuellement leurs modifications.
- Le code HTML et Java combiné en un seul fichier peut être difficile à lire, surtout si vos pratiques de codage sont peu soigneuses ou peu élégantes.

Les JSP ressemblent beaucoup aux ASP (Active Server Pages) de la plate-forme Microsoft. La principale différence entre JSP et ASP est la suivante : les objets manipulés par les JSP sont des JavaBeans et donc

indépendants des plates-formes. Les objets manipulés par les ASP sont des objets COM, ce qui lie complètement les ASP à la plate-forme Microsoft.

Pour plus d'informations sur la technologie des JSP, voir Chapitre 9, "Développement des pages JavaServer".

InternetBeans Express

La technologie InternetBeans Express s'intègre aux servlets et aux JSP pour améliorer vos applications et simplifier les tâches de développement des servlets et des JSP. InternetBeans Express est un ensemble de composants associé à une bibliothèque de balises JSP, permettant de générer la couche présentation d'une application web, et d'y répondre. InternetBeans Express prend des pages modèles statiques, leur insère un contenu dynamique issu du modèle de données réel, et les présente au client ; ensuite, il écrit dans le modèle de données les modifications postées par le client. Cela facilite la création des servlets et des JSP orientés données.

InternetBeans Express supporte les ensembles de données et les modules de données DataExpress. InternetBeans Express peut aussi être utilisé avec des modèles de données et des EJB génériques.

Pour plus d'informations sur InternetBeans Express, voir Chapitre 11, "Utilisation d'InternetBeans Express".

Choix des technologies à utiliser dans votre application web

Vous avez maintenant un aperçu des diverses technologies web, il vous reste à choisir celle que vous allez utiliser dans votre application web. Les conseils suivants vous aideront à prendre cette décision :

- Votre interface utilisateur est-elle particulièrement complexe ? Si votre interface utilisateur doit contenir d'autres composants que ceux destinés à la saisie de données (comme des champs texte, des boutons radio, des boîtes à options, des boîtes liste, des boutons submit, etc.) et aux images, vous souhaiterez sans doute utiliser une applet.
- Si vous voulez effectuer beaucoup de traitements côté serveur, vous devrez utiliser un servlet ou une JSP.
- Si vous voulez éviter à vos utilisateurs de charger beaucoup de code et si vous souhaitez accélérer le démarrage de votre application, choisissez un servlet ou une JSP.

- Si vous voulez contrôler le JDK utilisé par votre application (sans téléchargement), ou si vous savez que vos utilisateurs ont des navigateurs non Java, utilisez un servlet ou une JSP.
- Si vous recherchez un moyen de remplacer CGI, avec une charge mémoire moindre, utilisez un servlet ou une JSP.
- Si vous avez besoin de quelque chose se rapprochant d'une ASP, mais privilégiez l'indépendance par rapport aux plates-formes, vous choisirez une JSP.
- Si votre interface utilisateur est complexe et si vous voulez bénéficier de certaines caractéristiques des servlets ou des JSP, envisagez de combiner une applet et un servlet. Vous pouvez avoir une configuration faisant dialoguer une applet côté navigateur (client) et un servlet côté serveur.
- Si vous voulez utiliser un servlet ou une JSP et si vous voulez que ce dispositif contienne des données, vous devrez employer InternetBeans Express.
- Les servlets et les JSP se ressemblent suffisamment pour que votre choix se fonde sur vos préférences personnelles.
- En fait, beaucoup d'applications web utiliseront une combinaison de deux de ces technologies, ou même plus.

Présentation générale du processus de développement des applications web

Quelles que soient les technologies web choisies, les étapes fondamentales du développement de votre application web et de son fonctionnement sur un serveur web seront les mêmes. Voici ces étapes :

Etape	Description
Conception de votre application	Décider comment structurer votre application et quelles technologies utiliser. Décider ce qu'accomplira l'application et à quoi elle ressemblera. A ce stade, vous pouvez envisager de créer une WebApp.
Configuration de votre serveur web dans l'EDI de JBuilder	Vous pouvez, de manière facultative, configurer votre serveur web pour qu'il fonctionne dans l'EDI de JBuilder, et que vous puissiez compiler, exécuter et déboguer votre application avec le serveur web que vous utiliserez pour le déploiement. Si vous sautez cette étape, JBuilder utilisera automatiquement Tomcat, le serveur web livré avec JBuilder, lors de la compilation, de l'exécution et du débogage.

Etape	Description
Développement de votre application	Ecrire le code de l'application. Que votre application soit constituée d'applets, de servlets ou de pages JavaServer, les nombreux outils de JBuilder simplifient les tâches de développement.
Compilation de votre application	Compiler l'application dans l'EDI de JBuilder
Exécution de votre application	Exécuter l'application dans l'EDI de JBuilder Cela vous donnera un aperçu de l'application, sans obligation de la déployer au préalable. A ce stade, vous pouvez effectuer en local une partie des tests.
Déploiement de votre application	Déployer votre application sur le serveur web. La façon d'aborder cette étape dépend beaucoup du serveur web que vous utilisez.
Test de votre application	Tester votre application en train de s'exécuter sur le serveur web. Cela vous aide à détecter les problèmes concernant le déploiement ou l'application elle-même. Vous pouvez également effectuer les tests à partir d'un navigateur client, sur une autre machine que le serveur web. Vous pouvez essayer différents navigateurs, l'application pouvant se révéler légèrement différente sur chacun d'entre eux.

Applications web et applications distribuées

Vous pouvez envisager d'utiliser pour votre programme une application distribuée plutôt qu'une application web. Les deux gèrent la programmation client/serveur. Cependant, les deux technologies sont différentes.

En général, les applications distribuées gèrent et lisent les données des systèmes existants. Le système standard peut exister sur de nombreux ordinateurs travaillant sous des systèmes d'exploitation différents. Une application distribuée utilise un serveur d'applications, comme le Borland Application Server, pour la gestion de l'application. Les applications distribuées n'ont pas besoin d'être fondées sur Java ; en fait, une application distribuée peut contenir beaucoup de programmes différents, indépendamment du langage dans lequel ils sont écrits ou de l'emplacement où ils résident.

Les applications distribuées sont habituellement cantonnées à un réseau à l'intérieur d'une société. Des parties de votre application distribuée peuvent être rendues accessibles à des clients sur Internet, mais ensuite vous devrez combiner application distribuée et application web.

Les technologies utilisées dans une application distribuée sont notamment l'architecture CORBA (Common Object Request Broker Architecture) et RMI (Remote Method Invocation) :

- Le principal avantage de CORBA est que les clients et les serveurs peuvent être écrits dans n'importe quel langage de programmation. Cette possibilité existe parce que les objets sont définis avec le langage IDL (Interface Definition Language) et que la communication entre objets, clients et serveurs est gérée via les ORB (Object Request Brokers).
- RMI ou Remote Method Invocation permet de créer des applications Java-à-Java distribuées, dans lesquelles les méthodes des objets Java distants peuvent être appelées depuis des machines virtuelles Java et sur différents hôtes.

Les applications web peuvent être rendues accessibles à quiconque dispose d'un accès à Internet, ou placées derrière un "pare-feu" et uniquement utilisés à l'intérieur de l'intranet de votre société.

Les applications web nécessitent l'utilisation d'un navigateur côté client et d'un serveur web côté serveur. Par exemple des applets sont téléchargées sur des plates-formes client multiples où elles sont exécutées par la machine virtuelle Java (Java Virtual Machine, JVM) fournie par le navigateur qui s'exécute sur l'ordinateur client. Les servlets et les JSP s'exécutent à l'intérieur d'un serveur web Java supportant les spécifications JSP/Servlet.

Une application web peut faire partie d'une application distribuée plus vaste qui, à son tour, peut faire partie d'une application entreprise, ou J2EE. Pour un exemple d'application J2EE et la documentation correspondante, voir *Java 2 Platform, Enterprise Edition Blueprints* à l'adresse <http://java.sun.com/j2ee/blueprints/>. Consultez en particulier les sections appelées "The Client Tier" et "The Web Tier".

Utilisation des WebApps et des fichiers WAR

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

JBuilder propose des fonctionnalités très importantes pour la gestion de la structure de votre application web. Il y a deux concepts majeurs que vous devez comprendre pour tirer parti de ces fonctionnalités : les WebApps et les fichiers d'archive web (WAR).

La WebApp

Une WebApp décrit la structure d'une application web. C'est essentiellement une arborescence de répertoires renfermant le contenu web utilisé dans votre application. Elle correspond à un `ServletContext`. Un fichier de descripteur de déploiement, appelé `web.xml`, est toujours associé à chaque WebApp. Ce descripteur de déploiement contient les informations que vous devez fournir au serveur web lorsque vous déployez votre application.

L'utilisation d'une WebApp est utile si votre projet contient des servlets ou des JSP. Vous n'utiliserez probablement pas de WebApp si votre application web ne contient qu'une applet, mais vous souhaiterez en utiliser une dans une application web contenant une applet et un servlet ou une JSP. Ainsi, vous pourrez stocker la WebApp entière dans un fichier WAR unique. Vous pourriez avoir plusieurs WebApps sur un site web. JBuilder prend en compte cette notion en vous permettant d'avoir plusieurs WebApps dans un même projet.

Il est important de réfléchir à la structure de vos applications web pendant les phases de planning. Combien de WebApps y a-t-il ? Quels sont leurs noms ? Stockerez-vous ces WebApps dans des fichiers WAR ?

En prévoyant la structure depuis le début, vous faciliterez le déploiement ultérieur. JBuilder ne se sert pas de la notion d'une WebApp par défaut qui serait "enracinée" dans le répertoire <répertoireprojet>/racinedéfaut. Si vous ne spécifiez pas de WebApps pour vos applications web, elles utilisent la WebApp par défaut.

Pour plus d'informations sur la façon dont JBuilder fonctionne avec les WebApps, voir "L'expert application web", page 3-3, et "La WebApp et ses propriétés", page 3-4.

Fichiers archive web (WAR)

Un fichier WAR est le fichier servant d'archive à une application web. Il ressemble à un fichier JAR. En enregistrant toute votre application et les ressources qu'elle nécessite dans un fichier WAR, vous facilitez son déploiement. Vous copiez simplement le fichier WAR sur votre serveur web, au lieu de vérifier qu'un grand nombre de petits fichiers ont été copiés aux bons emplacements. JBuilder peut générer automatiquement un fichier WAR lorsque vous construisez votre projet.

Pour plus d'informations sur la façon dont JBuilder fonctionne avec les fichiers WAR, voir "Le fichier WAR", page 3-11.

Outils pour travailler avec les WebApps et les fichiers WAR

Voici la liste des outils fournis par JBuilder pour travailler avec les WebApps et les fichiers WAR :

Tableau 3.1 Outils de JBuilder destinés aux WebApps et aux fichiers WAR

outil	Description
Expert application web	C'est un expert servant à créer une WebApp. Il vous permet de spécifier le nom de la WebApp, le répertoire racine pour les documents de l'application web, et si vous voulez ou non générer un fichier WAR.
Nœud WebApp	Un nœud dans le volet projet de l'EDI de JBuilder, représentant la WebApp. Ce nœud possède une boîte de dialogue de propriétés permettant de configurer la WebApp. Contenus dans le nœud WebApp, vous trouverez des nœuds pour les descripteurs de déploiement, le répertoire racine et un fichier WAR facultatif.

Tableau 3.1 Outils de JBuilder destinés aux WebApps et aux fichiers WAR (suite)

outil	Description
Nœud du fichier WAR	Un nœud dans le volet projet de l'EDI de JBuilder, représentant la WebApp. Il possède une boîte de dialogue de propriétés et un visualiseur pour le contenu du fichier WAR.
Editeur DD WebApp	Une interface utilisateur et un éditeur pour le fichier descripteur de déploiement, <code>web.xml</code> , nécessaire à chaque WebApp. Dans JBuilder, vous pouvez également modifier des descripteurs de déploiement spécifiques au serveur, comme celui de WebLogic, <code>weblogic.xml</code> . Les descripteurs de déploiement et l'éditeur DD WebApp sont décrits en détail à la section "Descripteurs de déploiement", page 16-4.

L'expert application web

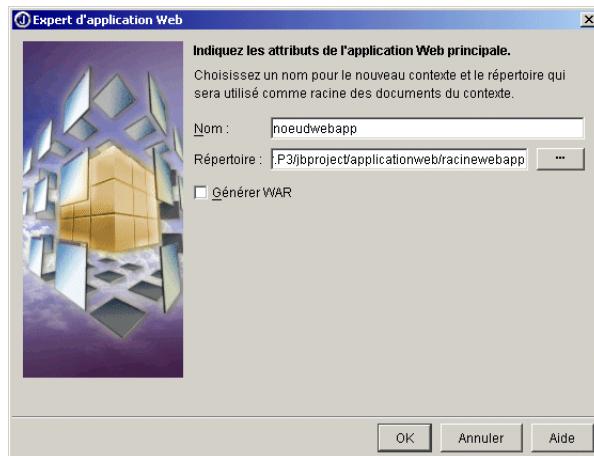
L'expert application web crée une nouvelle WebApp. Pour afficher l'expert application web, ouvrez la galerie d'objets (Fichier | Nouveau), cliquez sur l'onglet Web, sélectionnez Application web et cliquez sur OK.

L'expert est très simple. Il est constitué d'une page, avec deux champs texte et une case à cocher. Dans le premier champ texte, dont le libellé est Nom, entrez un nom pour votre WebApp. Dans le seconde champ texte, dont le libellé est Répertoire, entrez l'emplacement qui sera la racine du document de votre WebApp. Y saisir un nom de répertoire crée un sous-répertoire dans le répertoire du projet. Vous pouvez également cliquer sur le bouton Points de suspension pour naviguer jusqu'à un répertoire existant où vous pouvez créer un nouveau répertoire ou le choisir. Il n'est pas conseillé de choisir la racine du projet ni le répertoire src.

En cochant la case Générer WAR, vous indiquez que vous voulez générer un fichier WAR lors de la construction du projet. Si la case à cocher est sélectionnée, le fichier WAR aura le même nom que la WebApp et sera placé dans le répertoire contenant le répertoire racine du document. Si vous ne cochez pas Générer WAR, ne vous inquiétez pas. Vous pourrez toujours changer d'avis. Cette case à cocher correspond à un valeur des Propriétés de la WebApp.

Vous pouvez utiliser également l'expert application web pour importer une application web existante que vous pouvez avoir construite auparavant. Donnez un nom à la WebApp, et pointez sur l'emplacement du répertoire contenant votre application web existante. Si l'application web est valide, elle peut être exécutée dans l'EDI de JBuilder immédiatement.

Figure 3.1 Expert application web



La WebApp et ses propriétés

Un serveur web Java localise une application web par son `ServletContext`, qui est associé à la WebApp. Une WebApp est représentée dans l'EDI de JBuilder sous la forme d'un nœud. Il se trouve dans l'arborescence du volet projet et porte le nom de la WebApp.

Figure 3.2 Volet projet montrant un nœud WebApp



Le nœud WebApp a deux ou trois nœuds enfant, le Répertoire racine de l'application, un nœud Descripteurs de déploiement représentant le répertoire `WEB-INF` de la WebApp, et un nœud Fichier WAR facultatif.

Vous devez placer les fichiers du contenu web (comme les fichiers HTML, SHTML et JSP) dans le répertoire racine de la WebApp ou dans un de ses sous-réertoires. Les fichiers de contenu web sont des fichiers directement accessibles par le navigateur client. Les ressources Java utilisées par la WebApp (comme les servlets ou les JavaBeans) doivent avoir leur fichiers source dans le répertoire source du projet. Ces fichiers ne sont pas directement accessibles par le navigateur client, mais ils sont invoqués par autre chose, par exemple un fichier JSP. Les experts servlet, JSP et Lanceur de démarrage Web de JBuilder facilitent la création des applications web respectant ces règles. Soyez bien sûr de spécifier une WebApp nommée existante lorsque vous utilisez ces experts.

Répertoire racine

Le répertoire racine définit l'emplacement de base de l'application web. Chaque partie de l'application web sera placée relativement à ce répertoire racine. Tous les fichiers de contenu web, comme les fichiers HTML, SHTML, JSP ou d'images, seront placés dans ce répertoire. Les fichiers de contenu web sont des fichiers directement accessibles par le navigateur client.

Les fichiers du répertoire racine de la WebApp (et tous les sous-répertoires de ce répertoire) sont automatiquement affichés dans le nœud Répertoire racine du volet projet. Seuls les fichiers des types que vous spécifiez dans la page WebApp de la boîte de dialogue Propriétés de la WebApp sont affichés. Les types de fichiers par défaut sont ceux que vous utilisez habituellement dans une application web. Cela vous permet de travailler avec des fichiers HTML ou des fichiers d'images sans renoncer à vos outils favoris. Enregistrez ces fichiers dans le répertoire racine de la WebApp ou dans un de ses sous-répertoires. Cliquer ensuite simplement sur le bouton Rafraîchir du volet projet permet de voir l'ensemble de fichiers en cours dans JBuilder.



Descripteurs de déploiement

Chaque WebApp doit avoir un répertoire **WEB-INF**. Ce répertoire contient les informations nécessaires au serveur web pour le déploiement de l'application. Ces informations sont mises en forme dans les fichiers descripteurs de déploiement. Ces fichiers utilisent l'extension **.xml**. Ils sont affichés dans le nœud Descripteurs de déploiement du volet projet. Le répertoire **WEB-INF** est en fait un sous-répertoire du répertoire racine de votre WebApp. Il ne figure pas dans le volet projet sous le nœud Répertoire racine car c'est un répertoire protégé qui ne peut être servi par le serveur web.

Le nœud Descripteurs de déploiement de la WebApp contient toujours un fichier descripteur de déploiement appelé **web.xml**. Ce fichier est nécessaire à tous les serveurs web Java et il est créé par JBuilder lorsque vous créez la WebApp. Votre serveur web peut exiger des descripteurs de déploiement supplémentaires qui lui sont spécifiques. Ils peuvent être modifiés dans JBuilder et seront créés s'ils figurent dans la liste des descripteurs requis par le plugin du serveur web en cours de configuration. Recherchez dans la documentation de votre serveur web les descripteurs de déploiement qui lui sont nécessaires.

JBuilder fournit un éditeur de descripteur de déploiement pour modifier le fichier **web.xml**. Dans JBuilder, vous pouvez également modifier des descripteurs de déploiement spécifiques au serveur. Dans le fichier **web.xml**, certaines affectations, ou mappages, doivent être faites pour que la WebApp fonctionne comme prévu dans l'EDI de JBuilder. Ces

mappages seront écrits pour vous si vous utilisez les experts de JBuilder pour créer les différentes parties de votre application web. D'autres mappages peuvent être nécessaires au moment du déploiement de votre application. Pour plus d'informations sur les descripteurs de déploiement et l'éditeur DD WebApp, voir la section "Descripteurs de déploiement", page 16-4.

Propriétés de la WebApp

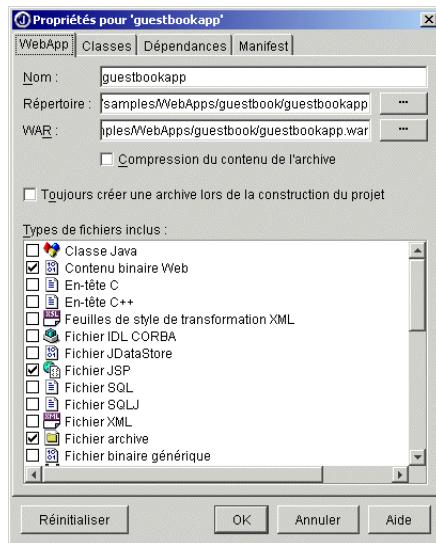
Le nœud WebApp du volet projet possède diverses propriétés que vous pouvez modifier. Pour modifier les propriétés du nœud WebApp, cliquez avec le bouton droit sur le nœud et sélectionnez Propriétés dans le menu. La boîte de dialogue Propriétés de la WebApp a quatre onglets :

- WebApp
- Classes
- Dépendances
- Manifest

L'onglet WebApp

L'onglet WebApp de la boîte de dialogue Propriétés de la WebApp indique le nom de la WebApp, l'emplacement du répertoire de la WebApp, l'emplacement du répertoire du fichier WAR. Il y a deux cases à cocher indiquant de générer (ou de mettre à jour) ou non le fichier WAR lorsqu'est construit le projet, et de compresser ou non l'archive. La case à cocher de création de l'archive correspond à la case à cocher "Générer WAR" de l'expert application web. Il peut être commode de désactiver la génération du fichier WAR pendant le développement et de l'activer au moment de la dernière construction du projet, juste avant le déploiement.

Cet onglet contient également la liste des types de fichiers à inclure, que ce soit pour la navigation dans les fichiers ou la génération du fichier WAR. Seuls les types de fichiers qui sont sélectionnés seront inclus, conformément aux extensions qu'ils utilisent. Les extensions associées à chaque type de fichiers peuvent être visualisées et modifiées dans la boîte de dialogue Options de l'EDI, accessible depuis le menu Outils.

Figure 3.3 Onglet WebApp de Propriétés de la WebApp

L'onglet Classes

La page Classes contrôle les classes Java et les ressources qui sont copiées dans le sous-répertoire WEB-INF/classes relatif à la racine du projet sur le disque (utilisé pendant le développement), et dans le sous-répertoire WEB-INF/classes du fichier WAR généré.

La page Classes contient trois boutons radio vous permettant de sélectionner le mode de gestion des classes et des ressources. Les options sont les suivantes :

Inclure les classes requises et les ressources connues

Cette option copie toutes les classes que vous avez spécifiquement ajoutées à votre WebApp avec le bouton Ajouter des classes. Elle ajoute également toutes les classes qui sont utilisées par une ou plusieurs des classes ajoutées. Rappelez-vous que les classes sont copiées dans WEB-INF/classes ou dans ses sous-répertoires. Les classes que vous sélectionnez doivent donc être des classes accédées côté serveur et non des classes ayant besoin d'être servies par le serveur web. Par exemple, les classes des servlets doivent être sélectionnées mais pas les classes des applets.

Cette option ajoute également les ressources connues. Il s'agit des ressources que vous ajoutez spécifiquement à l'archive avec le bouton Ajouter des fichiers. Si vous sélectionnez cette option et n'ajoutez ni classe ni fichier à la liste qui se trouve dessous, aucune classe ni aucune ressource ne sera copiée.

Inclure les classes et toutes les ressources requises

Cette option copie toutes les classes que vous avez spécifiquement ajoutées à votre WebApp avec le bouton Ajouter des classes. Elle ajoute également toutes les classes qui sont utilisées par une ou plusieurs des classes ajoutées. Rappelez-vous que les classes sont copiées dans WEB-INF/classes ou dans ses sous-répertoires. Les classes que vous sélectionnez doivent donc être des classes accédées côté serveur et non des classes ayant besoin d'être servies par le serveur web. Par exemple, les classes des servlets doivent être sélectionnées mais pas les classes des applets.

Cette option ajoute également toutes les ressources du chemin Source du projet, telles qu'images, clips vidéo, fichiers audio, etc.

Toujours inclure toutes les classes et les ressources

Cette option rassemble toutes les classes dans le chemin de sortie de votre projet. Le chemin de sortie est défini dans la page Chemins de la boîte de dialogue Propriétés du projet. Habituellement, c'est le répertoire classes de votre projet.

Elle regroupe également toutes les ressources du chemin Source du projet, également défini sur la page Chemins de la boîte de dialogue Propriétés. Il s'agit habituellement du répertoire src de votre projet. Les ressources sont des fichiers autres que les fichiers de classes, comme les images, les clips vidéo, les fichiers audio, etc.

Cette option est activée par défaut. Cette option est la plus sûre, car elle regroupe :

- Toutes les classes utilisées dans votre projet.
- Toutes les classes utilisées par toutes les classe ajoutées
- Les ressources utilisées par les classes de votre projet
- Les ressources que vous avez ajoutées à votre projet

Attention Quand vous sélectionnez cette option, tous les fichiers classe de votre chemin de sortie sont inclus dans le fichier WAR. Cela peut provoquer l'inclusion de fichiers classe et de ressources qui ne sont pas nécessaires. Sachez que générer un WAR avec cette option risque de prendre beaucoup de temps et de créer un fichier très volumineux.

Ajouter des classes

Le bouton Ajouter des classes affiche la boîte de dialogue Sélectionnez les classes ou paquets à inclure dans la webapp, dans laquelle vous pouvez sélectionner une ou plusieurs classes et un ou plusieurs paquets à ajouter à votre WebApp. La classe n'a pas besoin d'être dans le chemin de sortie de votre projet. Que vous choisissiez l'option Inclure classes requises et ressources connues ou l'option Inclure classes et toutes ressources requises, JBuilder analyse les fichiers des classes que vous ajoutez pour rechercher les éventuelles dépendances supplémentaires, et place les classes trouvées dans le répertoire WEB-INF/classes.

Ajouter des fichiers

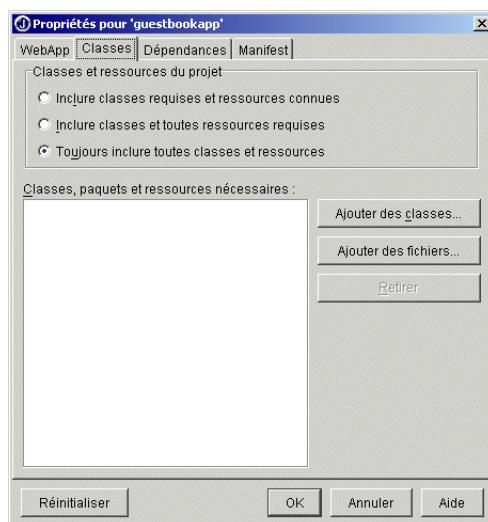
Le bouton Ajouter des fichiers affiche la boîte de dialogue Sélectionnez les ressources à inclure dans la webapp, dans laquelle vous pouvez sélectionner le ou les fichiers à ajouter à votre WebApp. Ce fichier doit se trouver dans le chemin source de votre projet. Utilisez cette option pour ajouter les divers fichiers nécessaires aux classes de la WebApp, comme les fichiers .properties et les pilotes de bases de données.

- Remarque** La boîte de dialogue Sélectionnez les ressources à inclure dans la webapp n'est pas capable d'examiner l'intérieur des fichiers d'archive. Si un fichier ou un paquet dont vous avez besoin se trouve à l'intérieur d'un fichier d'archive, commencez par l'extraire dans votre dossier source, puis ajoutez-le à l'aide du bouton Ajouter des fichiers.

Retirer

Supprime la classe ou le fichier sélectionné de la liste.

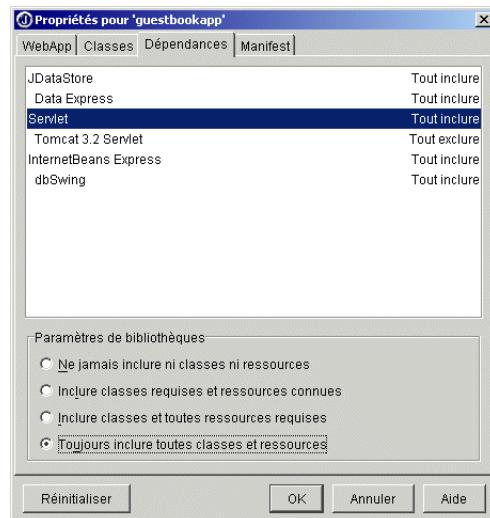
Figure 3.4 Onglet Classes de Propriétés de la WebApp



L'onglet Dépendances

L'onglet Dépendances vous permet de spécifier ce qu'il convient de faire pour les dépendances de bibliothèques existant dans votre WebApp. Vous souhaiterez généralement inclure les bibliothèques nécessaires. Au contraire des archives classiques, les fichiers JAR des bibliothèques sont stockés tels quels dans le répertoire WEB-INF/lib (lorsque l'option Paramètres de bibliothèques de l'onglet est défini par Toujours inclure toutes classes et ressources – ce que nous recommandons). L'onglet Dépendances de la boîte de dialogue Propriétés de la WebApp ressemble à l'onglet Dépendances de tout autre type d'archive. Pour plus d'informations, voir dans l'aide en ligne la rubrique "Expert Constructeur d'archives : Déterminez que faire des dépendances de bibliothèques".

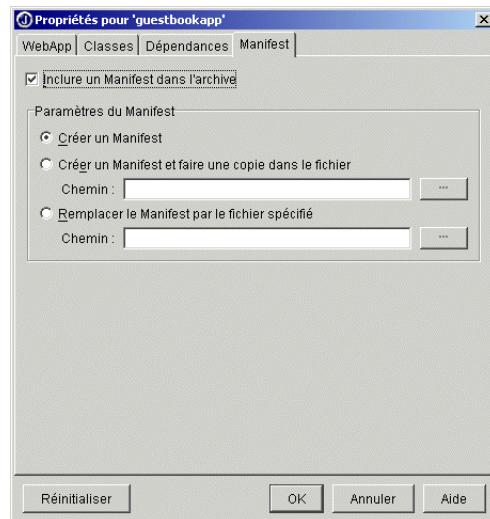
Figure 3.5 Onglet Dépendances de Propriétés de la WebApp



L'onglet Manifest

L'onglet Manifest de la boîte de dialogue Propriétés de la WebApp est identique à l'onglet Manifest de tout autre type d'archive. Pour plus d'informations, voir dans l'aide en ligne la rubrique "Expert Constructeur d'archives, Définissez les options du Manifest".

Figure 3.6 Onglet Manifest de Propriétés de la WebApp

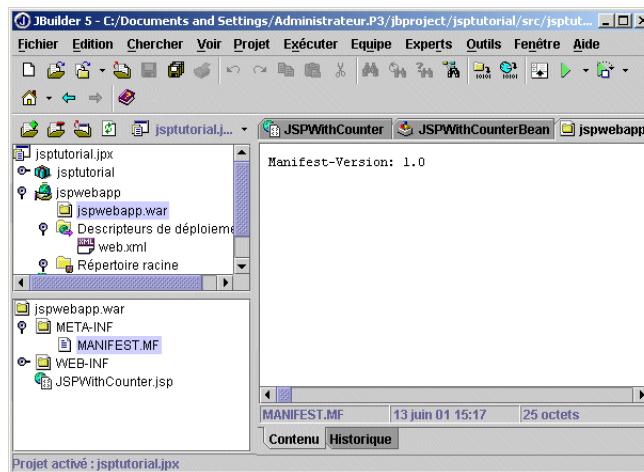


Le fichier WAR

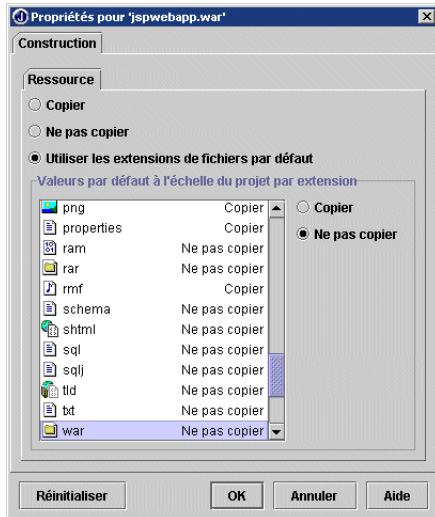
Un fichier WAR est le fichier servant d'archive à une application web. Placer tous les fichiers et toutes les ressources nécessaires à la WebApp dans un fichier WAR simplifie le déploiement. Si tout est correctement configuré et si toutes les ressources nécessaires se trouvent dans le fichier WAR, il suffit pour déployer de copier le fichier WAR au bon endroit sur le serveur web.

Si un fichier WAR est présent, un nœud représentant le fichier WAR apparaîtra sous le nœud WebApp dans l'arborescence du volet projet. Ouvrir le nœud du fichier WAR affiche le contenu de ce fichier dans le volet structure ; cela affiche également les visualiseurs de fichiers WAR dans l'AppBrowser. Les visualiseurs de fichiers WAR sont constitués d'un onglet Classes et d'un onglet Historique.

Figure 3.7 Nœud d'un fichier WAR ouvert dans l'EDI de JBuilder



Le nœud du fichier WAR a une boîte de dialogue Propriétés, accessible en cliquant avec le bouton droit sur le nœud et en sélectionnant Propriétés. Dans la boîte de dialogue Propriétés, vous pouvez spécifier si le fichier WAR est considéré comme étant une ressource Java. C'est le même onglet Ressource qui se trouve dans les propriétés de construction pour tous les types de fichiers non Java. Pour plus d'informations sur les propriétés des ressources, voir dans l'aide en ligne la section "Ressource" de la rubrique "Construction, page (boîte de dialogue Propriétés du projet)". Les propriétés les plus importantes d'un fichier WAR se trouvent dans la page WebApp de la boîte de dialogue Propriétés de la WebApp.

Figure 3.8 Dialogue des propriétés des fichiers WAR

Pour que JBuilder crée un fichier WAR pour votre application web, votre projet doit comporter un nœud WebApp. Vous pouvez créer cette WebApp avec l'expert application web, accessible dans la galerie d'objets, ou utiliser la WebApp par défaut.

Vous pouvez indiquer à JBuilder de créer ou de mettre à jour un fichier WAR à chaque construction du projet. Pour cela, cliquez avec le bouton droit sur le nœud WebApp dans le volet projet et sélectionnez Propriétés dans le menu. Cliquez sur l'onglet WebApp dans la boîte de dialogue Propriétés de la WebApp. Assurez-vous que l'option Toujours créer une archive lors de la construction du projet est sélectionnée. Vérifiez également que le nom et l'emplacement pour votre fichier WAR sont corrects. Il existe une option pour compresser le fichier WAR.

Maintenant qu'un fichier WAR est associé à votre WebApp, il faut que les diverses pièces de votre application web soient associées à cette WebApp pour être ajoutées au fichier WAR. Les experts servlet, JSP et Lanceur de démarrage Web ont une liste déroulante dans laquelle vous pouvez sélectionner votre WebApp lorsque vous créez ces pièces.

Applets dans un fichier WAR

Vous pouvez ajouter une applet au fichier WAR, mais cela nécessite quelque travail supplémentaire. Pour que l'applet fonctionne dans la WebApp, les classes doivent être accessibles. Cela signifie que le fichier HTML et les classes de l'applet doivent être localisés dans le répertoire racine de la WebApp ou dans un de ses sous-répertoires. Ils ne doivent pas se trouver dans le répertoire WEB-INF ni dans ses sous-répertoires.

L'expert applet ne permet pas que les fichiers des applets soient placés dans une WebApp, aussi vous devrez déplacer le fichier HTML, et les fichiers des classes compilées de l'applet, dans le répertoire racine de la WebApp (ou un de ses sous-répertoires). Selon l'endroit où vous placez les fichiers classe, vous devrez modifier l'attribut `codebase` de la balise `<applet>` pour qu'il permette de les trouver.

Si vous incluez dans une WebApp des fichiers classe d'une applet non archivés, vérifiez que vous avez sélectionné les classes Java parmi les types de fichiers inclus dans la page WebApp de Propriétés de la WebApp. Sinon, les fichiers classe ne seraient pas inclus dans le WAR.

Vous pouvez également placer le fichier JAR d'une applet dans le fichier WAR. Les mêmes règles s'appliquent. Le fichier JAR doit être accessible. Cela signifie qu'il doit être dans le répertoire racine de la WebApp ou dans un des ses sous-répertoires, mais pas sous le répertoire `WEB-INF`.

4

Utilisation des applets

Le développement des applets est une fonctionnalité de toutes les éditions de JBuilder.

Si vous avez déjà écrit et déployé des applets dans un site Web, vous avez sûrement remarqué que, par rapport aux applications, leur fonctionnement repose sur davantage de paramètres. Le déploiement et le test d'applets hors de l'EDI marquent le début des difficultés, et si vous ne tenez pas compte de certains points fondamentaux sur les applets, celles-ci ne peuvent pas fonctionner correctement. Pour réussir, vous devez savoir comment les applets fonctionnent et comment tous les éléments s'assemblent, notamment lorsque l'heure est venue de les déployer et de les télécharger vers un site Web externe.

Voir aussi

- “Tutoriel : Construction d’une applet” dans la *Prise en main*
- Le tutoriel Java “Writing Applets”, à l’adresse <http://www.java.sun.com/docs/books/tutorial/applet/index.html>
- Le site web de Sun, à l’adresse <http://www.javasoft.com/applets/index.html>
- Les FAQ Java, à l’adresse <http://www.afu.com/javafaq.html>
- de Charlie Calvert, “Java Madness, Part II: Applets”, à l’adresse <http://homepages.borland.com/ccalvert/JavaCourse/index.htm>
- Le site web Java Curmudgeon de Rich Wilkman, à l’adresse <http://formlessvoid.com/jc/applets/>
- de John Moore, “Applet design and deployment”, à l’adresse http://www.microps.com/mps/p_appletdesign.html

Fonctionnement des applets

Les applets sont des applications Java stockées sur un serveur Internet/intranet. Elles sont téléchargées vers plusieurs plates-formes clients sur lesquelles elles sont exécutées dans une machine virtuelle Java (Java Virtual Machine, JVM) fournie par le navigateur exécuté sur la machine client. La livraison et l'exécution se font sous la surveillance d'un gestionnaire de sécurité, qui peut empêcher les applets d'exécuter certaines tâches, comme le formatage du disque dur ou l'ouverture de connexions vers des machines "douteuses".

Lorsque le navigateur rencontre une page Web contenant une applet, il lance la JVM et lui fournit les informations contenues dans la balise `<applet>`. Le chargeur de classes à l'intérieur de la JVM essaie de savoir quelles sont les classes requises pour l'applet. Au cours du processus de chargement des classes, les fichiers classe sont analysés par un vérificateur garantissant la validité des fichiers classe et la non malveillance du code. Une fois vérifiée, l'applet est exécutée. Il s'agit, bien évidemment, d'une vision simplifiée du processus.

Le mécanisme de livraison est le point central des applets. Toutefois, certains points propres au développement des applets doivent être considérés pour garantir le succès de leur implémentation.

La balise `<applet>`

Tout ce qu'une applet doit savoir au moment de son exécution est déterminé par le contenu de la balise `<applet>` dans le fichier HTML. Les attributs de la balise lui indiquent la classe à exécuter, les archives éventuelles à consulter pour trouver des classes, et l'emplacement de ces archives et/ou ces classes.

A la différence des applications Java, qui utilisent une variable d'environnement appelée `CLASSPATH` pour rechercher les classes, les applets utilisent uniquement l'attribut `codebase` dans la balise `<applet>` pour spécifier où rechercher les classes qu'elles requièrent.

La bonne exécution d'une applet repose sur la capacité de celle-ci à trouver les classes qu'elle requiert.

Exemple de balise `<applet>`

La balise `<applet>` exemple suivante utilise les attributs les plus courants.

```
<applet  
    codebase = ". "  
    archive = "test.jar"  
    code = "test.Applet1.class"
```

```

name = "TestApplet"
width = 400
height = 300
vspace = 0
hspace = 0
>
<param name = "param1" value = "xyz">
</applet>

```

Attributs de la balise <applet>

Le tableau suivant décrit les attributs et paramètres les plus courants utilisés dans la balise <applet>. Notez que certains éléments sont requis.

Elément	Description
codebase	<p>Permet de spécifier un emplacement pour vos classes ou fichiers archive autre que celui du fichier HTML contenant la balise <applet>. Ce chemin dépend de l'emplacement du fichier HTML et se limite souvent aux sous-répertoires de l'emplacement du fichier HTML. <code>codebase</code> est semblable à <code>CLASSPATH</code> en ce sens qu'il indique au navigateur où rechercher les classes.</p> <p>Par exemple, si les classes d'applets sont stockées dans un sous-répertoire <code>classes</code>, l'attribut <code>codebase</code> est <code>codebase= "classes"</code>.</p> <p>La valeur <code>..</code> spécifie le même répertoire que le fichier HTML exécutant l'applet.</p> <p>Important : Cet attribut est requis si la classe ou les fichiers archive se trouvent dans un répertoire autre que celui du fichier HTML de l'applet.</p>
archive	<p>Permet d'identifier une ou plusieurs archives (ZIP, JAR ou CAB) contenant les classes requises par l'applet. Les fichiers d'archive doivent être placés dans le répertoire spécifié par <code>codebase</code>. Vous pouvez vous servir de plusieurs fichiers archive et les répertorier en les séparant par des virgules : <code>archive="archive1.jar, archive2.jar"</code>.</p> <p>Important : Cet attribut est requis si votre applet est déployée dans des fichiers archive.</p> <p>Remarque : Certains anciens navigateurs ne prennent en charge que les archives ZIP.</p>
code (requis)	Nom complet de la classe applet contenant la méthode <code>init()</code> . Par exemple, si la classe <code>Applet1.class</code> appartient à un paquet appelé <code>test</code> , <code>code</code> prend la valeur <code>code="test.Applet1.class"</code> .
name (facultatif)	Chaîne décrivant votre applet. Cette chaîne apparaît dans la barre d'état du navigateur.
width/height (requis)	Permet de définir la taille en pixels allouée à l'applet dans le navigateur. Ces informations sont aussi transmises au gestionnaire de disposition d'applet.

Elément	Description
hspace/vspace (facultatif)	Permet de représenter le remplissage horizontal ou vertical en pixels autour de l'applet. Cela est utile si du texte dans la page Web enveloppe l'applet ou si la page contient plusieurs applets.
param (facultatif)	Permet de spécifier des paramètres pouvant être lus par la balise <applet>. Ils sont assimilables aux paramètres de la liste d'arguments utilisée par main() dans les applications. Les paramètres sont constitués de deux parties, name et value, qui sont deux chaînes entre guillemets. Le nom (name) est utilisé à l'intérieur de l'applet lorsque vous voulez demander une valeur (value). Vous devez appliquer toute conversion du type chaîne en un autre type de données dans le code de votre applet. Assurez-vous de respecter la casse entre le paramètre HTML et le code d'applet qui la requiert.

Erreurs couramment commises dans la balise <applet>

La plupart des problèmes rencontrés avec la balise <applet> sont dus aux erreurs suivantes :

- **Balise </applet> manquante**

Si vous obtenez un message d'erreur indiquant que la page HTML ne contient aucune balise </applet>, vérifiez si une balise fermante existe.

- **Oubli du fait que Java distingue les majuscules des minuscules**

La notion de casse est très importante. Habituellement, les noms de classe utilisent les majuscules et les minuscules, tandis que les répertoires et les archives sont tout en minuscules. Les noms de classe, d'archive et de répertoire dans la balise <applet> doivent correspondre exactement à la casse de ceux utilisés sur le serveur Web. Si la casse n'est pas exactement respectée, des messages d'erreur de type "Impossible de trouver la classe xyz" apparaissent.

Si votre classe se trouve dans le paquet foo.bar.baz, tous les répertoires créés pour gérer cette classe doivent être créés et référencés à l'aide de minuscules.

Le déplacement de fichiers entre Windows 95/98 et Windows NT et Internet accentue la probabilité d'erreurs liées à la distinction entre majuscules et minuscules.

Une fois que vous avez envoyé vos fichiers sur le Web, vérifiez que les fichiers, répertoires et archives présentent les mêmes cassettes (lettres majuscules et minuscules) que votre machine locale. De même, si vous avez archivé tous vos fichiers classe, assurez-vous que l'outil d'archivage a conservé la casse.

- **Utilisation du “nom abrégé” de la classe dans l’attribut code**

Vous devez utiliser le nom complet car il s’agit du nom véritable de la classe. Si le nom de classe est `Applet1.class` et figure dans le paquet `test`, vous devez y faire correctement référence dans l’attribut `code` sous la forme `test.Applet1.class`.

- **Valeur codebase incorrecte**

Le `codebase` est résolu par rapport au répertoire qui contient le fichier HTML, en formant une entrée de classpath effective. La valeur `codebase` n’est pas une URL ni un autre type de référence. Dans l’exemple fourni d’applet simplifiée, `codebase = ". "` a été utilisé. Cela permet de spécifier que les fichiers requis par l’applet se trouvent dans le même répertoire que le fichier HTML ou, si aucune archive n’est spécifiée, dans les sous-répertoires de paquet appropriés rattachés à celui contenant le fichier HTML.

- **Attribut archive manquant**

Si votre applet est déployée dans un ou plusieurs fichiers JAR ou ZIP, vous devez ajouter l’attribut `archive` au fichier HTML.

- **Guillemets manquants autour des valeurs utilisées pour code, codebase, archive, name, etc.** Pour améliorer la compatibilité avec XHTML, nous vous recommandons d’utiliser les guillemets également autour des nombres.

Questions liées au navigateur

L’une des principales tâches permettant de faire fonctionner les applets au moment de l’exécution consiste à résoudre les différentes questions liées aux navigateurs qui hébergent les applets. Les navigateurs varient selon le niveau de support JDK, et chaque navigateur aborde l’implémentation Java à sa manière. Les principales questions liées aux navigateurs sont les suivantes.

Support Java

Un problème courant lié aux applets est la non correspondance des versions de JDK entre votre applet et le navigateur qui l’exécute. De nombreux utilisateurs n’ont pas mis à niveau ou à jour leur navigateur. Ces navigateurs prennent en charge le code JDK version 1.1.5 et antérieures, mais pas les nouveaux JDK de Sun. Swing, introduit dans JDK 1.1.7, n’est pas encore pris en charge par de nombreux navigateurs. L’erreur la plus courante est `NoClassDefFoundError` pour telle ou telle classe dans les paquets `java.*` lorsque le chargeur de classes détermine qu’il a

besoin d'une classe ou méthode Java non prise en charge par le JDK du navigateur.

JDK 1.2 et 1.3 ne sont pas totalement pris en charge dans tous les navigateurs. Bien que les versions récentes des navigateurs supportent effectivement Java, vous devez connaître la version spécifique de Java gérée dans les navigateurs qui exécuteront votre applet. Ceci est important afin que vous puissiez faire tout le nécessaire pour créer une correspondance entre la version de JDK utilisée dans votre applet et celles gérées par les navigateurs. Les navigateurs qui gèrent JDK 1.1 sont Netscape version 4.06 et ultérieures et Internet Explorer version 4.01 et ultérieures. La prise en charge des versions JDK peut varier d'une plate-forme à l'autre.

Pour rechercher la version JDK prise en charge par le navigateur, consultez la Console Java dans le navigateur ou le site Web du navigateur.

Pour ouvrir la Console Java :

- Sélectionnez Communicator | Outils | Console Java dans Netscape.
- Sélectionnez Affichage | Console Java dans Internet Explorer.

Le module de mise à jour Java™ de Sun résout facilement le problème de la non correspondance entre les versions JDK en téléchargeant systématiquement la même version de JDK sur toutes les machines des utilisateurs finaux.

Fourniture du navigateur préféré à l'utilisateur final

Vous devez d'abord amener vos clients à installer une version minimale d'un navigateur. Si leur navigateur favori est déjà installé, vous devez les convaincre que tout inconvénient causé par l'obtention de mises à jour est contrebalancé par la valeur de votre applet. Chaque mise à jour d'un navigateur suppose le téléchargement de celle-ci et/ou l'installation d'un correctif.

Support de plusieurs navigateurs

Si vous avez choisi de gérer plusieurs navigateurs avec votre applet, vous devrez le faire aussi sur leur lieu d'utilisation. Cela signifie que si les utilisateurs ont des problèmes liés au navigateur lors de l'utilisation de votre applet, ils vous appelleront pour obtenir de l'aide.

Différences dans l'implémentation Java

Il existe des instructions et des spécifications sur les navigateurs mais leur implémentation ainsi que celle de toutes les extensions est du ressort de concepteur du navigateur.

Les navigateurs diffèrent par leur implémentation du gestionnaire de sécurité et des niveaux de sécurité. Ce qui est autorisé dans un navigateur au niveau de sécurité "moyen" ne l'est pas dans un autre. Un ajustement de sécurité est réalisé côté client, et chaque navigateur le fait à sa manière. Par exemple, le mécanisme d'assouplissement de la sécurité pour une applet est la signature. Signer une applet et obtenir du navigateur qu'il accepte cette signature apporte davantage de souplesse à l'applet. Le mécanisme permettant cela est intégré dans Java avec l'utilitaire **Javakey**. Malheureusement, les navigateurs ne sont pas obligés d'utiliser ce mécanisme. De fait, certains utilisent leurs propres mécanismes propriétaires pour la signature, qui ne fonctionnent qu'avec ces navigateurs.

Une autre différence réside dans le fait que les concepteurs de navigateurs ajustent unilatéralement la fonctionnalité Java centrale, y compris en supprimant une partie. Ceci peut aboutir à des comportements inattendus à l'exécution. Par exemple, des appels de dessin ne se produisent pas, laissant l'applet dans un état instable ou même invisible. Il est possible d'écrire du code pour des situations spécifiques, mais ce n'est pas toujours possible.

De plus, il existe souvent des différences dans le support Java suivant les plates-formes dans un même navigateur. Par exemple, une applet multithread peut fonctionner sur une plate-forme, mais en raison de problèmes dans le modèle de threading d'une autre plate-forme, les threads risquent d'être exécutés de façon séquentielle sur celle-ci. Souvent, les navigateurs ne sont pas mis à jour simultanément d'une plate-forme à l'autre et une mise à jour de navigateur pour une plate-forme peu répandue peut être diffusée ultérieurement.

Ces différences accroissent le temps que vous devez consacrer au test et au débogage de tel ou tel navigateur.

Solutions aux questions liées aux navigateurs

- **Utilisez le module de mise à jour Java**

La plupart des questions liées aux navigateurs mentionnées ci-dessus peuvent être résolues par l'utilisation du module de mise à jour Java, disponible à l'adresse <http://java.sun.com/products/plugin/>. Ce module offre des archives JDK à jour, dont vous tirez pleinement profit si votre applet utilise Swing (JFC).

Vous devez modifier vos fichiers HTML pour que l'applet utilise le module de mise à jour. Sun fournit un utilitaire, le convertisseur HTML, permettant de réaliser la conversion rapidement et facilement.

Seule l'utilisation du module de mise à jour Java permet d'avoir la même version de JDK dans les différents navigateurs. De plus vous

obtenez ainsi une version de JDK plus à jour que celle du navigateur. Certaines extensions ajoutées par les concepteurs de navigateurs peuvent fonctionner différemment avec le module de mise à jour.

Du point de vue de l'utilisateur final, la première fois qu'il rencontre une page activée par un module de mise à jour Java, il est invité à installer le module de mise à jour. Une fois que le module est installé, la machine client dispose d'une VM de Sun et la version la plus récente de JDK (y compris Swing) est installée localement. Cela ne sera utilisé que dans les pages reconnaissant le "module de mise à jour", mais cela permet d'économiser du temps système dans la livraison au navigateur de la dernière technologie JDK.

Important

Il existe plusieurs versions du module de mise à jour : JDK 1.1.x, 1.2 et 1.3. Vous devez utiliser la version qui correspond au JDK utilisé par votre applet. En outre, la version du convertisseur HTML doit correspondre à la version du module de mise à jour. Notez également que vous ne pouvez installer qu'une seule version du module de mise à jour sur la machine cliente.

- **Utilisez la même version de JDK supportée par les navigateurs**

Vous pouvez éviter de nombreux problèmes en écrivant des applets à l'aide de la version de JDK prise en charge par les navigateurs. JBuilder Professionnel et JBuilder Entreprise vous permettent de changer de version de JDK utilisée pour le développement. Bien que JBuilder Personnel ne prenne pas en charge le changement de version de JDK, vous pouvez modifier la version de JDK actuelle. Voir Outils | Configurer les JDK et "Définition des propriétés d'un projet" dans *Construction d'applications avec JBuilder*.

- **Ecrivez uniquement du code JDK 1.1.x et version antérieure**

L'écriture des applets uniquement à l'aide de JDK 1.1.x et version antérieure limite à la fonctionnalité (et aux bogues) de ces versions. De plus, la plupart des outils de développement génèrent du code 1.1.x ou ultérieur. JBuilder, par exemple, dispose de nombreux outils qui génèrent du code 1.1.x ou ultérieur inutilisable pour vos applets. Toutefois, JBuilder offre une fonctionnalité qui vous permet de modifier votre version du JDK ou de permuter plusieurs versions. Voir "Définition des propriétés d'un projet" dans *Construction d'applications avec JBuilder*.

JDK 1.02 offre la meilleure prise en charge dans les navigateurs mais présente aussi pour les événements des mécanismes différents de ceux de JDK 1.1.x. La plupart des navigateurs prennent désormais en charge JDK 1.1.5 mais pas les composants Swing, introduits dans JDK 1.1.7. Si vous écrivez vos applets dans des versions de JDK plus récentes, utilisez uniquement des composants AWT et n'ayez recours à aucune fonctionnalité nouvelle comme les composants Swing.

L'inconvénient majeur lié à l'utilisation de versions de JDK 1.1.x et antérieures est que toutes les améliorations se trouvent dans les versions ultérieures, ce qui vous obligera à migrer et à porter l'ancien code. Un autre inconvénient est que certaines versions de navigateurs compatibles avec JDK 1.1.x présentent davantage de problèmes avec le code JDK 1.02. Faites également attention à ne pas déboguer avec JDK 1.1 et des versions antérieures, car elles ne comportent pas d'implémentation JPDA (Java Platform Debugger Architecture).

- **Installez sur les postes clients les classes Java non intégrées**

Chaque navigateur dispose d'un répertoire que Java utilise pour les recherches basées sur `CLASSPATH`. Si vous stockez les archives volumineuses utilisées par votre applet localement, le client n'a pas besoin de les télécharger. Cela n'est généralement possible que dans un environnement intranet, ou dans des situations Internet dans lesquelles vous définissez le contrôle d'un client (comme dans un service basé sur un abonnement). Toutefois, cela rend plus difficile les futures mises à jour de code.

- **Sélectionnez un navigateur**

Assurez-vous que les utilisateurs n'utilisent qu'un seul navigateur afin de réduire au minimum les besoins particuliers en maintenance et en code. Cela n'est généralement possible que dans des situations intranet contrôlées par IS (International Standard).

- **Utilisation de Java Web Start**

Java Web Start est une technologie de déploiement des applications conçue par Sun Microsystems. Elle permet de lancer des applications et des applets Java à partir d'un lien présent sur une page web affichée dans votre navigateur web.

Pour plus d'informations sur Web Start, voir Chapitre 17, "Lancement de votre application web avec Java Web Start", et la page Java Web Start, à l'adresse <http://java.sun.com/products/javawebstart/>.

Conseils supplémentaires pour faire fonctionner les applets

Vous trouverez ci-dessous d'autres conseils permettant d'éviter des problèmes pendant le développement et le déploiement des applets :

- **Placez les fichiers à l'endroit correct**

Un problème courant lié au déploiement d'une applet est le placement des fichiers à un mauvais endroit (par exemple, "Impossible de trouver : xyz.class"). Tout dépend du fichier HTML qui lance l'applet. En l'absence d'un `CLASSPATH` facilitant la recherche des classes de l'applet, l'applet se base sur les attributs `codebase` et `code` figurant dans le fichier HTML pour rechercher les classes nécessaires à son exécution.

Par conséquent, avant le démarrage du déploiement, il est important que les fichiers figurent à l'endroit correct.

Le processus de déploiement vous semblera plus facile si l'environnement de travail de votre projet reflète la réalité d'une applet déployée, rapprochant un petit peu votre développement du déploiement instantané. Par exemple, lorsque la valeur de `codebase` est `..`, le navigateur recherche les fichiers classe et archive de l'applet dans le même répertoire que le fichier HTML. De même, si la classe est dans un paquet, le navigateur construit un chemin de répertoires, sous celui du fichier HTML, en respectant la structure du paquet. Si vous utilisez l'expert Applet de JBuilder, cette structure est créée automatiquement par l'expert. Une fois que l'applet est constituée et exécutée dans cette situation, vous pouvez tout archiver dans un seul fichier JAR ou ZIP. Ensuite, testez votre applet hors de JBuilder dans les navigateurs.

Astuce

Vous pouvez utiliser une copie de la page HTML réelle présente sur votre site dans le projet local, au lieu d'une simple page de test. Ainsi, cela rend un peu plus réaliste le test externe. Lorsque vous êtes satisfait, téléchargez tout le répertoire vers votre site Web.

- **Utilisez des paquets**

Vous devez toujours utiliser des paquets pour organiser des classes similaires lorsque vous codez en Java. Cela permet de les trouver plus facilement, et simplifie sensiblement le déploiement. Les paquets vous permettent également d'éviter les conflits de noms de classes puisque le nom de paquet précède le nom de classe en code Java.

- **Utilisez le navigateur le plus récent**

Déterminez les navigateurs sur lesquels votre applet sera exécutée. Si vous utilisez des outils courants comme JBuilder, vous serez amené à générer du code JDK version 1.1.x ou ultérieure et devrez utiliser les navigateurs les plus récents pour exécuter vos applets. Assurez-vous de fournir le module de mise à jour ou le correctif le plus récent afin de faire correspondre la version JDK du navigateur à celle de l'applet. Voir le module de mise à jour Java, à l'adresse <http://java.sun.com/products/plugin/>.

- **Respectez la casse**

N'oubliez pas que Java fait la distinction entre les majuscules et les minuscules. Vérifiez que la casse de tous les noms de classe, paquet, archive et répertoire de la balise `<applet>` correspond exactement aux noms sur le serveur.

- **Utilisez les fichiers archive**

Les archives simplifient le déploiement car vous ne devez télécharger que le fichier HTML et votre ou vos fichiers archive. Elles accélèrent par ailleurs le processus de téléchargement vers le client ; un seul fichier

compressé est téléchargé au lieu de chaque fichier classe non compressé. Vérifiez systématiquement la structure des répertoires et la casse des noms de fichiers dans le fichier archive. Voir "Déploiement des applets", page 4-14, et "Déploiement des applets dans JBuilder", page 4-25, pour plus d'informations sur la création des fichiers archive.

Important

Les anciennes versions de certains navigateurs ne gèrent pas plusieurs archives et ne traitent que des fichiers ZIP non compressés.

- **Déployez tous les fichiers**

Sauf si vous écrivez du code utilisant uniquement des classes Java intégrées, vous devez déployer tous les fichiers classe requis par votre applet sur le Web. Bien sûr, vous devez déployer chaque élément au bon endroit. Voir "Déploiement des applets", page 4-14, pour plus d'informations.

- **Testez à nouveau l'applet après l'avoir déployée sur le serveur**

Il ne suffit pas de tester l'applet localement. Vous devez aussi la tester dans plusieurs navigateurs après l'avoir déployée sur le serveur. C'est essentiel pour que toutes les classes requises par l'applet soient déployées correctement sur le serveur, pour que la balise `<applet>` concorde avec le déploiement sur le serveur, et pour résoudre les éventuels problèmes spécifiques à un navigateur particulier.

Voir aussi

"Test des applets", page 4-15

La sécurité et le gestionnaire de sécurité

L'un des points essentiels liés à l'exécution de programmes sur Internet est la sécurité. Les utilisateurs sont à juste titre prudents dans leur souhait de télécharger et d'exécuter des programmes sur leurs machines avec la garantie d'un minimum de sécurité pour éviter certaines déconvenues comme la suppression de fichiers par des programmes ou le téléchargement d'informations personnelles à partir de leurs ordinateurs.

La sécurité est un atout pour le développeur car sans elle de nombreux utilisateurs finals ne permettraient pas l'exécution des applets. Toutefois, elle présente certains inconvénients pour le développeur, notamment si celui-ci n'est pas au fait des restrictions définies au début du projet. De nombreuses activités considérées comme acquises dans une application seront refusées dans une applet.

Le sas de sécurité

Les applets traitent les problèmes de sécurité en exécutant tout code non sécurisé dans un environnement fiable appelé *sas de sécurité*. Lorsqu'une applet se trouve dans le sas de sécurité, elle est sous le contrôle d'un

gestionnaire de sécurité. Le gestionnaire de sécurité protège la machine client contre les programmes nocifs pouvant détruire des fichiers, lire des informations sécurisées ou mener toute action constituant une violation du modèle de sécurité utilisé. En revanche, les restrictions définies par le gestionnaire de sécurité, limitent le champ d'action d'une applet.

La compréhension de ces limites avant l'écriture des applets peut induire une économie de temps et d'argent. Les applets, comme tout outil, réalisent certaines tâches parfaitement. Mais l'utilisation d'une applet dans un contexte inadéquat ne peut que générer des problèmes.

Restrictions des applets

Par défaut, toutes les applets sont "non sécurisées" et sont écartées de certaines activités, dont les suivantes :

- **Lecture et écriture à partir du disque local**

Vous ne pouvez pas lire des fichiers, vérifier leur existence, en écrire, en renommer, etc.

- **Connexion à une machine autre que celle dont est issue l'applet**

Cela rend difficile la fourniture de données à partir d'une base de données se trouvant sur une machine autre que le serveur Web.

Il existe d'autres activités considérées comme évidentes par les développeurs d'applications, comme l'exécution de programmes systèmes locaux pour gagner du temps, qui ne sont pas autorisées dans les applets. N'importe quel ouvrage correct consacré à Java dresse la liste des restrictions appliquées aux applets.

S'il vous arrive d'essayer de court-circuiter le gestionnaire de sécurité dans la planification ou le développement, il est préférable d'utiliser une application à la place. Sun a fourni un effort considérable pour définir son modèle de sécurité et pour identifier les bits d'informations clés ou les types de données, dont dépendent des ensembles entiers de fonctionnalités. En conséquence, il est très difficile de passer outre le modèle de sécurité.

Solutions aux problèmes de sécurité

- **Signez l'applet**

Cela vous permet d'assouplir une partie des restrictions appliquées par le gestionnaire de sécurité. Il présente certains inconvénients, dont le fait qu'il n'existe pas de mécanisme de signature standard fonctionnant dans tous les navigateurs. Vérifiez si le site Web mentionne votre navigateur afin d'obtenir davantage d'informations sur la signature de

fichiers archive d'applet. Pour plus d'informations sur la signature des applets, voir "Code signing for Java applets", à l'adresse http://www.suitable.com/Doc_CodeSigning.shtml.

- **Amenez vos utilisateurs à modifier le modèle de sécurité de leur côté**

Les navigateurs disposent de paramètres permettant d'ajuster le modèle de sécurité. Ces paramètres vont de choix très simples parmi "high," "medium" et "low" à des schémas assez souples.

Malheureusement, ces paramètres sont globaux à toutes les applets. Tandis que l'utilisateur peut penser que votre applet ne présente aucun danger lorsque la sécurité est assouplie, il est souvent difficile de le convaincre de la même chose pour toutes les applets qu'il peut rencontrer.

- **Utilisez des technologies différentes pour court-circuiter le gestionnaire de sécurité**

Par exemple, si vous avez uniquement besoin de réécrire sur le serveur, écrivez des applications Java côté serveur, comme les servlets, avec lesquelles votre applet peut communiquer. Ce code côté serveur ne présente pas les limites de l'applet et peut être assez puissant. Les applications côté serveur Java nécessitent un accès au serveur physique ou à un fournisseur de services Internet (ISP) très souple et ouvert. Il s'agit de solutions qui conviennent mieux à un intranet.

- **Déterminez les tâches ne convenant pas aux applets**

Le rassemblement de données à partir de formulaires sur le Web et leur collecte conviennent mieux aux servlets ou aux pages JSP. Les opérations complexes, comme l'utilisation de bases de données, nécessitent d'autres logiciels sur le serveur, comme JDBC. Il s'agit d'une solution plus complexe mais, sans elle, les actions de base de données "normales", comme la validation de nouvelles données, ne sont pas possibles dans les applets. Si vous écrivez un servlet ou une JSP qui utilise JDBC pour se connecter à une base de données, utilisez InternetBeans Express pour "orienter données" le servlet ou la page JSP. Voir les autres chapitres de ce manuel pour plus d'informations sur ces sujets.

- **Envisagez l'écriture d'une application, d'un servlet ou d'une page JSP, plutôt que d'une applet**

Les avantages des applications Java, un accès complet au langage Java et l'absence de problèmes de sécurité suffisent largement à contrebalancer les avantages des applets dans certaines situations. Les servlets et les pages JSP ont également des avantages sur les applets. Ils ne sont pas basés sur le JDK du navigateur client, puisque Java est exécuté côté serveur. De plus, les servlets et les pages JSP sont souvent plus rapides car ils n'ont pas besoin d'un téléchargement sur le navigateur client, contrairement aux applets.

Utilisation de bibliothèques tierces

Les bibliothèques tierces sont un énorme avantage pour le développeur. Ces bibliothèques contiennent souvent de nombreux composants qui permettent d'économiser du temps et de l'argent. Toutefois, il est important de mettre en balance les avantages des bibliothèques de composants et les inconvénients pour l'utilisateur final.

Avec les applets, toutes les classes référencées doivent être envoyées via Internet jusqu'à la machine client. Si la bibliothèque tierce offre une architecture importante, elle peut sensiblement augmenter la taille du fichier de l'applet. Vous devez aussi tenir compte de la licence de redistribution de la bibliothèque. Pour certaines bibliothèques, vous êtes obligé de les redistribuer dans leur totalité (sans vous limiter aux éléments que vous utilisez).

Rappel Plus l'applet est volumineuse, plus son téléchargement et son exécution demandent de temps.

Déploiement des applets

Le déploiement est le processus consistant rassembler les classes de l'applet et leurs composants dépendants dans la hiérarchie correcte en vue de leur déploiement sur un serveur. Le Constructeur d'archives présent dans JBuilder Professionnel et Entreprise permet d'effectuer un déploiement dans l'EDI de JBuilder après le développement de l'applet. Sun fournit l'outil **jar** dans le JDK.

Plusieurs choses facilitent le déploiement :

- Développer votre applet dans un environnement local qui est l'image miroir de la structure des répertoires de votre site Web.
- Utiliser les fichiers archive.
- Utiliser le Constructeur d'archives dans JBuilder Professionnel et Entreprise ou l'outil JDK **jar** pour créer les fichiers archive.

Important Si vous écrivez des applets JDK 1.1.1 avec des composants Swing, ce qui n'est pas recommandé à cause des problèmes de navigateur, vous devez aussi télécharger et livrer les classes JFC (Swing), `swingall.jar`, car elles ne sont pas livrées en tant que parties du JDK.

Voir aussi

- “Déploiement des applets dans JBuilder”, page 4-25
- “Déploiement des programmes Java” dans *Construction d'applications avec JBuilder*
- L'outil **jar**, à l'adresse <http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#basic>

Test des applets

L'objet principal d'un EDI est de vous permettre de développer et de tester votre code localement. Lorsque vous exécutez une application ou une applet dans JBuilder, si le programme requiert une classe particulière, JBuilder dispose d'un rayon d'action important pour trouver cette classe. Cela vous permet réellement de vous concentrer sur le développement de votre application ou applet et d'en assurer le fonctionnement.

Toutefois, cela ne permet pas de connaître précisément le comportement des applets lors de leur exécution. Pour tester correctement une applet, vous devez la déployer sur le serveur Web dans son environnement d'exécution réel. Ceci est la seule façon de s'assurer que toutes les classes requises par l'applet sont disponibles sur le serveur qui l'exécute.

Lorsque vous exécutez votre applet dans un navigateur, elle doit contenir suffisamment d'informations dans la balise `<applet>` du fichier HTML pour que le chargeur d'applet puisse trouver toutes les classes, toutes les images et tous les fichiers divers requis par votre applet. Cet environnement représente l'ensemble du "monde réel" de votre applet, et sa seule source d'informations lui permettant de trouver les fichiers.

Pour les premiers tests de votre applet déployée, envisagez l'utilisation du "navigateur" de test JDK **appletviewer**. Cet outil à ligne de commande fourni avec JDK affiche utilement les messages d'erreur et les suivis de pile. Comme le montrent les étapes suivantes, supprimez le chemin de classe (`CLASSPATH`) dans la session de test.

Etapes de test élémentaires

Vous trouverez ci-dessous les étapes élémentaires à suivre pour tester vos applets :

1 Déployez les fichiers d'applet nécessaires sur le serveur.

Vérifiez les points suivants :

- Les fichiers sont transférés sur le serveur web en mode binaire lorsque vous utilisez un client FTP.
- Les fichiers se trouvent au bon emplacement **par rapport au fichier HTML** qui lance l'applet et la valeur `codebase` est correcte.
- La casse des noms sur le serveur correspond aux noms d'origine.
- La balise `<applet>` contient l'attribut `archive` avec le ou les noms de fichiers archive si l'applet est déployée dans un fichier JAR ou ZIP.
- La structure des répertoires correspond à la structure du paquet.

Le déploiement est traité en détails dans "Déploiement des programmes Java" dans *Construction d'applications avec JBuilder*.

- 2 Ouvrez une fenêtre ligne de commande.
- 3 Retirez tous les chemins de classe (CLASSPATH) éventuellement définis pour cette session. **appletviewer** sait où se trouvent les fichiers Java intégrés.
- 4 Passez au répertoire contenant le fichier HTML et le fichier JAR (ou les fichiers de classe).
- 5 Exécutez **appletviewer**.

```
<jbuilder>/jdk/bin/appletviewer TestApplet.html
```

où <jbuilder> est le nom du répertoire de JBuilder Par exemple, jbuilder5/.

Remarque

Si JBuilder se trouve sur un autre lecteur, incluez le nom de ce lecteur. Pour Windows, utilisez une barre oblique inverse (\).

Si c'est la première fois que vous exéutez **appletviewer**, un écran d'accord de licence apparaîtra. Cliquez sur Oui.

Si tout se passe bien, votre applet est exécutée dans **appletviewer**. Dans le cas contraire, consultez la console de **appletviewer** pour lire les messages d'erreur.

- Si un message d'erreur de type "impossible de trouver la classe" apparaît, vérifiez votre déploiement.
- Si une exception d'exécution apparaît, comme NullPointerException, déboguez votre code pour trouver l'erreur.

Test des navigateurs

Testez toujours votre applet dans les navigateurs après l'avoir déployée sur le serveur. Cela est essentiel pour que toutes les classes requises par l'applet soient déployées correctement sur le serveur et que la balise `<applet>` et le déploiement sur le serveur concordent.

Voici quelques conseils pour la réalisation du test dans le navigateur :

- Assurez-vous que le navigateur supporte Java. Reportez-vous à l'aide en ligne du navigateur.
- N'utilisez pas le bouton de rechargement ou de réactualisation de votre navigateur après avoir recompilé et déployé votre applet révisée. En effet, le navigateur peut poursuivre le chargement de l'ancienne version de l'applet à partir de la mémoire cache.
- Utilisez la Console Java pour lire les messages d'erreur du navigateur. Pour ouvrir la Console Java :
 - Sélectionnez Communicator | Outils | Console Java dans Netscape.
 - Sélectionnez Affichage | Console Java dans Internet Explorer.

Voir aussi “Solving common applet problems” dans le tutoriel Java, “Writing Applets” (<http://www.java.sun.com/docs/books/tutorial/applet/problems/index.html>)

JBuilder et les applets

JBuilder offre divers outils pour développer votre applet :

- Un expert applet pour créer rapidement une applet.
- Un concepteur pour concevoir visuellement l’interface utilisateur d’une applet.
- Applet TestBed de JBuilder pour exécuter et déboguer les applets.
- L’**appletviewer** de Sun pour exécuter et déboguer les applets.

Pour des informations sur la création d’applets dans JBuilder, voir :

- “Tutoriel : Construction d’une applet” dans *Prise en main*
- “Construction d’une interface utilisateur” dans *Construction d’applications avec JBuilder*

Création d’applets avec l’expert applet

JBuilder fournit l’expert applet pour générer le code principal d’une applet. Pour créer une applet en utilisant l’expert applet, effectuez les étapes suivantes.

Attention

Si vous créez une applet pour le web, consultez “Questions liées au navigateur”, page 4-5, pour avoir des informations sur la compatibilité des navigateurs et des JDK avant de concevoir l’applet.

- 1 Choisissez Fichier | Nouveau projet pour créer un nouveau projet pour votre applet. L’expert projet apparaît, en proposant par défaut le nom du projet, le nom du répertoire ainsi que les chemins racine, source, de sauvegarde, de documentation et de destination. Changez le nom du fichier projet et le nom du répertoire du projet, si vous voulez donner au projet un nom particulier. Laissez inchangé le reste des chemins.

Remarque

Les chemins des fichiers projet sont prédéfinis dans les propriétés du projet par défaut. Tant que vous n’êtes pas un utilisateur Java plus expérimenté, il vaut mieux ne pas les modifier. Pour plus d’informations sur les propriétés de projet par défaut, voir “Création et gestion des projets” dans *Construction d’applications avec JBuilder*.

Le nom du paquet du projet, dérivé du nom du fichier projet, s’affiche dans l’expert applet. Par exemple, si vous créez un projet appelé / <home>/<nomutilisateur>/jbproject/appletproject1/appletproject1.jpr, l’expert applet proposera le nom de paquet appletproject1.

Pour plus d’informations sur les paquets, voir “Paquets” dans *Construction d’applications avec JBuilder*.

- 2** Cliquez sur Suivant pour passer à l'étape 2 de l'expert projet. Notez les chemins générés à l'étape 1 :

Chemin du projet	Où est enregistré le fichier projet.
Chemin source	Où sont enregistrés les fichiers source.
Chemin de sauvegarde	Où sont enregistrés les fichiers de sauvegarde.
Chemin de la documentation	Où sont enregistrés les fichiers de documentation.
Chemin de sortie	Où sont enregistrés les fichiers classe et les HTML des applets.

Ces chemins et le chemin du JDK peuvent être modifiés si vous sélectionnez le bouton Points de suspension. Les bibliothèques nécessaires peuvent aussi être ajoutées.

Voir aussi

“Définition du JDK” et “Construction des chemins d'accès par JBuilder” dans *Construction d'applications avec JBuilder*

- 3** Cliquez sur Suivant pour passer à l'étape 3 dans laquelle vous pouvez entrer les informations concernant le projet dans le fichier HTML des notes sur le projet.

- 4** Cliquez sur Terminer pour quitter l'expert projet.

L'expert projet crée deux fichiers :

- Un fichier projet (.jpr ou .jpx).
- Un fichier HTML dont le nom correspond au projet et qui contient les informations par défaut sur le projet. Modifiez-le pour enregistrer toutes autres informations sur le projet plus pertinentes à afficher.

- 5** Ensuite, choisissez Fichier | Nouveau et choisissez l'onglet Web de la galerie d'objets. (Dans JBuilder Personnel, l'icône Applet se trouve sur la page Nouveau.)

- 6** Double-cliquez sur l'icône Applet dans la galerie d'objets pour ouvrir l'expert applet.

- 7** Tapez éventuellement un nouveau nom pour la classe de l'applet.

- 8** Sélectionnez la classe de base que l'applet va étendre : `java.applet.Applet` (classe AWT) ou `java.swing.JApplet` (classe JFC Swing).

Important

Dans la plupart des cas, il vaut mieux choisir `java.applet.Applet` comme classe de base, car la plupart des navigateurs web ne supportent pas encore Swing.

9 Cochez les options restantes que vous souhaitez :

Générer les commentaires d'en-tête	Utilise les informations du fichier projet comme commentaires d'en-tête au début du fichier classe de l'applet.
Peut s'exécuter indépendamment	Crée une fonction <code>main()</code> pour tester l'applet sans qu'elle soit appelée par une page HTML.
Créer les méthodes standard	Crée les stubs des méthodes standard de l'applet : <code>start()</code> , <code>stop()</code> , <code>destroy()</code> , <code>getAppletInfo()</code> et <code>getParameterInfo()</code> .

10 Cliquez sur Suivant pour passer à l'étape 2. Au cours de cette étape, vous pouvez ajouter des paramètres. A partir de ces informations, l'expert génère les balises `<param>` à l'intérieur de la balise `<applet>` du fichier HTML de l'applet et le code de gestion des paramètres dans le fichier java de l'applet.

Remplissez une ligne pour chaque paramètre voulu.

- Pour ajouter un nouveau paramètre, cliquez sur le bouton Ajouter un paramètre.
- Pour sélectionner une cellule, cliquez dessus ou utilisez les flèches de navigation du clavier pour vous positionner dessus.
- Pour entrer une valeur dans une cellule, tapez une valeur ou sélectionnez-la si il existe une liste déroulante.
- Pour retirer un paramètre, cliquez sur une cellule de la ligne du paramètre, puis cliquez sur le bouton Retirer un paramètre.

Voici les définitions des champs paramètres :

Nom	Le nom du paramètre. Il sera utilisé pour fournir l'attribut <code>name</code> de la balise <code><param></code> dans le fichier HTML et pour fournir le paramètre <code>name</code> de l'appel à <code>getParameter()</code> correspondant dans le source Java.
Type	Le type de la variable qui sera insérée dans le code source Java de votre applet pour contenir la valeur du paramètre issu de la page HTML.
Desc	Une courte description du paramètre. Elle est utilisée pour décrire le paramètre lorsque des outils externes demandent à l'applet les paramètres qu'elle supporte. Un exemple de ces outils est Applet Info de l'appletviewer.
Variable	Le nom de la variable qui sera insérée dans le code source Java de votre applet pour contenir la valeur du paramètre issu de la page HTML.

Défaut La valeur par défaut du paramètre. C'est la valeur que le code source Java de cette applet utilise si un futur fichier HTML utilisant cette applet n'a pas de balise `<param>` pour ce paramètre. Pour qu'un fichier HTML fournit ce paramètre, l'attribut `name` de la balise `<param>` doit correspondre exactement à ce que vous avez entré dans la colonne Nom. Cette correspondance fait la distinction entre les majuscules et les minuscules.

11 Cliquez sur Suivant pour aller à l'étape 3. Si vous ne voulez pas générer automatiquement la page HTML, vous pouvez désactiver l'option Crée une page HTML. Sinon, laissez-la cochée et entrez les informations demandées, comme le titre de la page, le nom de l'applet, la taille de l'applet dans la page et l'emplacement du codebase. Par défaut, l'expert Applet enregistre le fichier HTML dans le répertoire de sortie avec les classes compilées.

12 Choisissez Terminer pour générer les fichiers .java et HTML de l'applet.

L'expert Applet crée deux fichiers si vous avez sélectionné l'option Crée une page HTML à l'étape 3 :

- Un fichier HTML qui contient une balise `<applet>` référençant votre classe applet. C'est le fichier que vous devez sélectionner pour exécuter ou déboguer votre applet.
- Une classe Java qui étend `Applet` ou `JApplet` et dont vous pouvez effectuer la conception dans le concepteur d'interface utilisateur.

Remarque

Dans JBuilder Professionnel et JBuilder Entreprise, un nœud de paquet source automatique apparaît également dans le volet projet si l'option Activer automatiquement les paquets source est sélectionnée dans la page Général de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet).

Voir aussi

- "Création d'applets avec l'expert applet", page 4-17
- "Tutoriel : Construction d'une applet" dans *Prise en main*

Exécution d'applets

Pour exécuter une applet dans un projet :

- 1 Enregistrez le fichier.
- 2 Compilez le projet.

3 Faites l'une des choses suivantes :

- Cliquez avec le bouton droit de la souris sur le fichier HTML dans le volet projet. Ce fichier doit contenir une balise <applet>. Sélectionnez Exécuter. L'applet s'exécute dans l'**appletviewer** de Sun.
- Choisissez Projet | Exécuter le projet ou le bouton Exécuter dans la barre d'outils pour exécuter l'applet à partir de la classe principale dans le AppletTestbed de JBuilder.

La classe principale est définie dans la page Exécution de Propriétés du projet (Projet | Propriétés du projet).

Remarque

Vous pouvez aussi sélectionner le fichier .java de l'applet, s'il contient une méthode main, et choisir Exécuter. Vous pouvez créer une applet avec une méthode main() en sélectionnant l'option Peut s'exécuter indépendamment dans l'étape 1 de l'Expert applet.

L'applet est compilée et elle est exécutée s'il n'y a pas d'erreurs. La progression de la construction est affichée dans une boîte de dialogue et le volet des messages affiche toutes les erreurs de compilation. Si la compilation du programme réussit, le chemin des classes est affiché dans le volet des messages et l'applet est exécutée.

AppletTestBed de JBuilder et appletviewer de Sun

Il y a deux façons d'exécuter une applet dans JBuilder :

- L'AppletTestbed de JBuilder
- L'**appletviewer** de Sun.

Le comportement par défaut est le suivant si vous avez créé votre applet avec l'expert Applet :

- Sélectionnez Exécuter | Exécuter le projet ou le bouton Exécuter pour exécuter l'applet à partir de la classe principale dans le visualiseur d'applet de JBuilder, AppletTestbed.
- Cliquez avec le bouton droit de la souris sur le fichier applet HTML et sélectionnez Exécuter pour exécuter l'applet dans l'**appletviewer** de Sun.

Vous pouvez changer le comportement par défaut de Exécuter | Exécuter le projet et du bouton Exécuter, dans l'onglet Applet de la page Exécution de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet). Il y a deux possibilités pour exécuter une applet dans JBuilder :

- Classe principale - l'AppletTestbed de JBuilder
- Fichier HTML - l'**appletviewer** de Sun

Classe principale

Quand vous sélectionnez une classe principale pour exécuter l'applet, elle s'exécute dans le visualiseur d'applet de JBuilder, AppletTestbed. Quand



vous créez votre applet à l'aide de l'expert applet, la classe principale est automatiquement définie. La classe sélectionnée doit contenir une méthode `init()`.

Fichier HTML

Quand vous sélectionnez un fichier HTML pour exécuter l'applet, celle-ci s'exécute dans l'**appletviewer** de Sun. Le fichier HTML doit contenir la balise `<applet>` et l'attribut `code` doit contenir le nom de classe complet. Si le fichier classe ne se trouve pas dans le même répertoire que le fichier HTML, l'attribut `codebase` doit spécifier son emplacement par rapport au fichier HTML.

Exécution des applets JDK 1.1.x dans JBuilder

Si vous exécutez votre applet JDK 1.1.x depuis sa classe principale dans JBuilder, elle s'exécute en utilisant l'Applet TestBed de JBuilder, lequel nécessite JFC (Swing) pour démarrer. Comme JDK 1.1.1 n'est **pas** inclus dans les classes JFC, il vous faut télécharger la version spécifique JDK 1.1.x de JFC (Swing), `swingall.jar`, depuis le site JavaSoft (<http://www.javasoft.com/products/>). Ensuite, créez une bibliothèque pour les classes JFC et ajoutez-la au projet (Outils | Configurer les bibliothèques).

Exécution des applets JDK 1.2 dans JBuilder

Pour exécuter une applet sous Solaris ou Linux depuis JBuilder, vous devez ajouter à votre projet la bibliothèque SDK Open Tools. Si vous ne le faites pas, vous risquez une exception `NoClassDefFoundError:AppletTestbed`. Ce problème affecte certains exemples d'applet, comme l'exemple Primes Swing.

Débogage des applets

Vous pouvez déboguer des applets dans l'EDI de JBuilder exactement comme vous le feriez pour tout autre programme Java. Comme il y a deux façons d'exécuter des applets dans JBuilder, vous pouvez déboguer les applets de deux manières : Applet TestBed de JBuilder et **appletviewer** de Sun.

Pour déboguer l'applet avec l'Applet TestBed de JBuilder, vous devez exécuter la classe principale de l'applet qui contient la méthode `init()` :

- 1 Définissez la classe principale de l'applet, qui contient la méthode `init()`, comme la classe exécutable, dans l'onglet Applet de la page Exécution (Projet | Propriétés du projet).
- 2 Compilez le projet.
- 3 Définissez un point d'arrêt dans le fichier applet.



- 4** Choisissez Exécuter | Déboguer le projet ou cliquez sur le bouton Déboguer de la barre d'outils. Le débogueur se charge dans le volet message et l'applet s'exécute dans l'AppletTestbed de JBuilder.

Remarque

Si vous développez votre applet en utilisant un ancien JDK, vous devrez passer à un JDK plus récent (JDK 1.2.2 ou 1.3) pour le débogage. Les premiers JDK ne supportent pas l'API de débogage JPDA (Java Platform Debugger Architecture) qu'utilise JBuilder. Voir "Définition du JDK" dans *Construction d'applications avec JBuilder*.

Pour des instructions détaillées sur le débogage dans JBuilder, voir "Débogage des programmes Java" dans *Construction d'applications avec JBuilder*.

Pour déboguer l'applet avec l'**appletviewer** de Sun, vous devez exécuter le fichier HTML à l'aide d'une de ces méthodes :

- A partir du volet projet :
 - 1** Compilez le projet.
 - 2** Définissez un point d'arrêt dans le fichier applet.
 - 3** Cliquez avec le bouton droit sur le fichier HTML dans le volet projet et choisissez Déboguer. Le débogueur se charge dans le volet message et l'applet s'exécute dans l'**appletviewer**.
- A partir du menu Exécuter de JBuilder :
 - 1** Définissez le fichier HTML de l'applet comme fichier exécutable, dans l'onglet Applet de la page Exécution (Projet | Propriétés du projet).
 - 2** Compilez le projet.
 - 3** Définissez un point d'arrêt dans le fichier applet.
 - 4** Choisissez Exécuter | Déboguer le projet ou cliquez sur le bouton Déboguer de la barre d'outils. Le débogueur se charge dans le volet message et l'applet s'exécute dans l'**appletviewer** de Sun.

**Débogage des applets dans le module de mise à jour Java**

C'est une fonctionnalité de JBuilder Entreprise.

Vous pouvez aussi déboguer vos applets depuis Internet Explorer 5 ou Netscape Navigator 4.72 en utilisant le module de mise à jour Java et le débogueur de JBuilder.

Pour déboguer en utilisant le JDK 1.3,

- 1** Téléchargez et installez le module de mise à jour (plugin) pour JDK 1.3 à partir du site Sun : <http://java.sun.com/products/plugin/>.
- 2** Téléchargez et installez le convertisseur HTML à partir du site Sun, à l'adresse <http://java.sun.com/products/plugin/1.3/features.html>.

(La version du convertisseur HTML doit correspondre à la version du module de mise à jour).

- 3** Suivez les instructions pour définir les paramètres de débogage du module de mise à jour Java pour les applets : <http://java.sun.com/products/plugin/1.3/docs/debugging.html>.

Remarque

Ajoutez **-classic** comme premier paramètre dans le panneau de configuration Java Plug-in afin d'utiliser la VM classique et déboguer plus rapidement.

- 4** Lancez JBuilder et effectuez les opérations suivantes :
 - 1** Créez une applet simple avec un fichier HTML. Voir "Tutoriel : Construction d'une applet" dans *Prise en main*.
 - 2** Compilez le projet.
 - 3** Convertissez le fichier HTML en utilisant le convertisseur HTML. Les instructions d'utilisation du convertisseur HTML se trouvent à l'adresse : <http://java.sun.com/products/plugin/1.3/docs/htmlconv.html>.
 - 4** Définissez un point d'arrêt dans le fichier applet.
 - 5** Définissez ce qui suit dans la page Débogage de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet).
 - 1** Activez le débogage à distance.
 - 2** Sélectionnez l'option Attacher.
 - 3** Définissez les mêmes paramètres de débogage que ceux que vous feriez dans le panneau de configuration du module de mise à jour Java. Choisissez l'une de ces options :
 - Choisissez le type de transport `dt_shmem` et `javadefbug` dans le champ Adresse.
 - Choisissez le type de transport `dt_socket` et entrez dans le champ Adresse l'adresse que vous avez indiquée dans le panneau de configuration (la valeur par défaut fournie dans JBuilder est 5000).

Pour de plus amples informations sur le débogage à distance, voir "Débogage des applications distribuées" dans le *Guide du développeur d'applications distribuées*.
 - 5** Démarrez Netscape ou Internet Explorer et ouvrez le fichier HTML. Assurez-vous que le navigateur supporte Java. Reportez-vous à l'aide en ligne du navigateur.
 - 6** Passez dans JBuilder et commencez le débogage de votre projet. Le débogueur doit démarrer sans problème. Revenez dans le navigateur et actualisez la page. Dans JBuilder, le débogueur doit s'arrêter au point d'arrêt de l'applet.

Déploiement des applets dans JBuilder

JBuilder, version Professionnel ou Entreprise, offre un Constructeur d'archives qui peut automatiquement créer les fichiers archive ZIP et JAR. Vous pouvez aussi créer des fichiers JAR manuellement en utilisant l'outil **jar** de Sun qui est fourni avec le JDK. De nombreux outils ZIP permettent la création de fichiers ZIP, mais vous devez vous assurer d'utiliser une version qui accepte les noms de fichier longs.

Si votre application web est volumineuse avec des servlets, des JSP, des applets et d'autres contenus web, utilisez un fichier WAR, qui est un fichier archive d'application web. Votre application web, fichier WAR ou fichier JAR, peut aussi être empaquetée dans un fichier EAR.

Voir aussi

- “Utilisation du constructeur d'archives” dans *Construction d'applications avec JBuilder*
- “Déploiement des programmes Java” dans *Construction d'applications avec JBuilder*
- L'outil **jar**, à l'adresse <http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html#basic>
- Chapitre 3, “Utilisation des WebApps et des fichiers WAR”

Utilisation des servlets

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Les servlets Java offrent un protocole et une méthode indépendante des plates-formes pour construire des applications web, sans les limites de performance des programmes CGI. Les servlets s'exécutent dans les serveurs web et, au contraire des applets, ne possèdent pas d'interface utilisateur graphique. Ils interagissent avec le moteur de servlets qui s'exécute sur le serveur web par le biais de demandes et de réponses.

Un programme client, qui peut être écrit dans n'importe quel langage de programmation, accède au serveur web et fait sa demande. La demande est alors traitée par le moteur de servlets du serveur web, lequel la transmet au servlet. Le servlet envoie ensuite une réponse au client via le serveur web.

Aujourd'hui, les servlets sont très souvent utilisés pour construire des applications web interactives. Une grande variété de serveurs web tiers possédant des extensions moteur de servlets sont actuellement disponibles : Tomcat (l'implémentation de référence des API Servlet/JSP), iPlanet Web Server (anciennement Netscape Enterprise Server), etc. Les serveurs web avec moteurs de servlets, également appelés conteneurs servlet, peuvent aussi être intégrés aux serveurs d'applications web, comme Borland AppServer (inclus dans JBuilder Entreprise), BEA WebLogic Application Server (supporté par JBuilder Entreprise), IBM WebSphere (supporté par JBuilder Entreprise), Netscape Application Server, etc.

Remarque

Le serveur web Tomcat est inclus avec JBuilder Professionnel et JBuilder Entreprise comme serveur web par défaut.

Le principal avantage de la technologie servlet est la vitesse. Au contraire des programmes CGI, les servlets sont chargés en mémoire une seule fois et exécutés depuis la mémoire après leur chargement initial. Les servlets sont générés dynamiquement comme un thread et ils sont, par nature,

multithreads. De plus, étant basés sur le langage Java, ils sont indépendants des plates-formes.

La technologie JSP (JavaServer Pages) est une extension de la technologie des servlets créée spécifiquement pour faciliter l'écriture des pages HTML et XML. Elle permet de mélanger aisément des modèles fixes ou statiques avec du contenu dynamique. Même si vous vous sentez à l'aise pour écrire des servlets, il y a plusieurs raisons valables d'étudier la technologie JSP comme complément de votre travail actuel. Pour plus d'informations sur l'écriture des JSP, voir Chapitre 9, "Développement des pages JavaServer".

Pour plus d'informations sur les servlets, visitez les sites web suivants. Ces adresses et ces liens étaient valables au moment de l'impression de ce manuel. Borland ne contrôle pas ces sites et n'est pas responsable de leur contenu ni de leur pérennité.

- Java Servlet Technology, à l'adresse <http://java.sun.com/products/servlet/index.html>
- Java Servlet Technology: White Paper, à l'adresse <http://java.sun.com/products/servlet/whitepaper.html>
- Page Java Servlet Technical Resources, à l'adresse <http://java.sun.com/products/servlet/technical.html>
- Page Java Servlet Third-Party Resources, à l'adresse <http://java.sun.com/products/servlet/resources.html>
- Java Developer Connection: page Servlets, à l'adresse <http://developer.java.sun.com/developer/technicalArticles/Servlets/index.html>
- The Servlet Trail of The Java tutorial, à l'adresse <http://java.sun.com/docs/books/tutorial/servlets/index.html>

Vous pouvez également consulter les tutoriels suivants sur la création des servlets dans JBuilder :

- Chapitre 7, "Tutoriel : Crédation d'un servlet simple"
- Chapitre 8, "Tutoriel : Crédation d'un servlet mettant à jour un guestbook"
- Chapitre 12, "Tutoriel : Crédation d'un servlet avec InternetBeans Express"

Servlets et JSP

Les technologies JSP et servlets ont toutes deux des avantages. Comment choisir celle à utiliser dans une situation donnée ?

- Les **Servlets** représentent un outil de programmation et sont parfaitement adaptés aux fonctions de bas niveau des applications qui nécessitent peu de logique de présentation.
- Les **JSP** représentent un moyen déclaratif, axé sur la présentation, de lier de façon dynamique contenu et logique. Les JSP peuvent être utilisées pour manipuler la représentation HTML générée par une page. Elles sont codées en pages de type HTML, avec une structure et un contenu familiers aux fournisseurs de contenu web. Cependant, les pages JSP sont moins puissantes que les pages HTML normales. Les JSP peuvent gérer la logique d'une application via l'utilisation de composants JavaBeans, de composants EJB (Enterprise JavaBeans) et de balises personnalisées. Les JSP elles-mêmes peuvent être utilisées comme composants de présentation modulaires et réutilisables, qui peuvent être liés ensemble en utilisant un mécanisme de modélisation.

Les JSP étant compilées dans des servlets, vous pourriez théoriquement écrire des servlets pour prendre en charge vos applications basées sur le web. Cependant, la technologie JSP cherche à simplifier le processus de création des pages en séparant la présentation web du contenu web. Dans beaucoup d'applications, la réponse envoyée au client est une combinaison de données modèle et de données générées dynamiquement. Dans une telle situation, il est beaucoup plus facile de travailler avec des pages JSP que de tout faire avec des servlets.

Servlets et serveurs web

En 1999, Sun Microsystems livrait les dernières versions des API des servlets et des JSP à l'Apache Software Foundation. Apache, aidé de Sun et de nombreuses autres compagnies, a développé et livré l'implémentation de référence officielle des JSP/Servlets, appelée Tomcat. C'est la seule implémentation de référence disponible. Tomcat est gratuite pour toutes les entreprises et tous les développeurs. Pour plus d'informations sur Apache Software Foundation et Tomcat, reportez-vous à l'adresse <http://jakarta.apache.org>. Pour plus d'informations sur JBuilder et les serveurs web, voir Chapitre 14, "Configuration de votre serveur web".

Les éditions JBuilder Professionnel et JBuilder Entreprise fournissent tout Tomcat pour que vous l'utilisiez comme serveur web et réussissiez le développement et les tests de vos servlets et de vos JSP dans l'environnement de développement de JBuilder. Pour en savoir plus sur

Tomcat, voir la documentation Tomcat dans le dossier `doc` de l'installation Tomcat de JBuilder.

De nombreux autres serveurs web supportent les API des servlets et des JSP de JavaSoft. La liste de ces produits se trouve dans la rubrique "Servers and Engines" de la page Servlet Technology Industry Momentum de JavaSoft, à l'adresse <http://java.sun.com/products/servlet/industry.html>.

L'API des servlets

L'API des servlets est contenue dans le paquet `javax.servlet`. Tous les servlets doivent implémenter, directement ou indirectement, l'interface `javax.servlet.Servlet`. Cette interface permet aux servlets de s'exécuter dans un moteur de servlets (une extension du serveur web). Elle définit également le cycle de vie du servlet. L'API des servlets est intégrée à de nombreux serveurs web, y compris au serveur web par défaut livré avec JBuilder, Tomcat. Le tableau suivant donne la liste des classes et des interfaces les plus utilisées du paquet `servlet` et décrit brièvement chacune d'entre elles.

Tableau 5.1 Présentation de l'API des servlets

Nom	Classe ou Interface	Description
<code>GenericServlet</code>	Classe	Définit un servlet générique, indépendant des protocoles.
<code>RequestDispatcher</code>	Interface	Définit un objet qui reçoit les demandes en provenance du client et les envoie à une ressource quelconque (comme un servlet, un fichier HTML ou un fichier JSP) sur le serveur.
<code>Servlet</code>	Interface	Définit les méthodes que tous les servlets doivent implémenter.
<code>ServletConfig</code>	Interface	Un objet configuration servlet utilisé par un conteneur servlet pour transmettre des informations à un servlet pendant l'initialisation.
<code>ServletContext</code>	Interface	Définit un ensemble de méthodes utilisées par un servlet pour communiquer avec son conteneur, par exemple, pour obtenir le type MIME d'un fichier, répartir les requêtes ou écrire dans un fichier historique.
<code>ServletException</code>	Classe	Définit l'exception générale que peut déclencher un servlet lorsqu'il rencontre des difficultés.

Tableau 5.1 Présentation de l'API des servlets (suite)

Nom	Classe ou Interface	Description
ServletInputStream	Classe	Fournit un flux d'entrée pour lire des données binaires issues de la demande d'un client, y compris une méthode <code>readLine</code> permettant de lire les données ligne par ligne.
ServletOutputStream	Classe	Fournit un flux de sortie pour envoyer des données binaires au client.
ServletRequest	Interface	Définit un objet pour fournir au servlet des informations sur les demandes client.
ServletResponse	Interface	Définit un objet pour assister un servlet lorsqu'il envoie une réponse au client.
SingleThreadModel	Interface	Garantit que les servlets ne gèrent qu'une demande à la fois.
UnavailableException	Classe	Définit l'exception qu'un servlet déclenche pour indiquer qu'il est indisponible de façon temporaire ou permanente.

La paquet `servlet.HTTP`

Utilisez le paquet `javax.servlet.http` pour créer des servlets qui supportent le protocole HTTP et la génération HTML. Le protocole HTTP utilise un ensemble de méthodes de requêtes et de réponses de type texte (méthodes HTTP), notamment :

- GET
- POST
- PUT
- DELETE
- HEAD
- TRACE
- CONNECT
- OPTIONS

La classe `HttpServlet` implémente ces méthodes HTTP. Pour commencer, étendez simplement `HttpServlet` et redéfinissez soit la méthode `doGet()`, soit la méthode `doPost()`. De plus, pour un meilleur contrôle, vous pouvez redéfinir les méthodes `doPut()` et `doDelete()`. Si vous créez un servlet avec l'expert servlet de JBuilder, vous pouvez sélectionner les méthodes que vous souhaitez redéfinir, JBuilder créera le code squelette pour vous.

Le tableau suivant donne la liste des classes et des interfaces les plus utilisées du paquet `javax.servlet.http` et décrit brièvement chacune d'entre elles.

Nom	Classe ou Interface	Description
Cookie	Classe	Crée un cookie, petite quantité d'informations envoyée par un servlet à un navigateur web, enregistrée par le, navigateur et, ultérieurement, réexpédiée au serveur.
HttpServlet	Classe	Fournit une classe abstraite à sous-classer pour créer un servlet HTTP approprié à un site web.
HttpServletRequest	Interface	Etend l'interface <code>ServletRequest</code> pour fournir les informations des demandes pour les servlets HTTP.
HttpServletResponse	Interface	Etend l'interface <code>ServletResponse</code> pour fournir la fonctionnalité spécifique HTTP dans l'envoi d'une réponse.
HttpSession	Interface	Fournit le moyen d'identifier un utilisateur, d'une demande de page à l'autre ou lors de la visite d'un site web, et de stocker des informations sur cet utilisateur.
HttpSessionBindingEvent	Classe	Envoie un objet qui implémente <code>HttpSessionBindingListener</code> lorsque l'objet est attaché ou détaché de la session.
HttpSessionBindingListener	Interface	Avertit un objet lorsque l'objet est attaché ou détaché d'une session.

Le cycle de vie des servlets

L'interface `javax.servlet.Servlet` contient les méthodes consacrées au cycle de vie des servlets. Implémentez ces méthodes pour :

- Construire le servlet et l'initialiser avec la méthode `init()`.
- Gérer les appels à la méthode `service()` effectués par les clients.
- Met le servlet hors service, le détruit avec la méthode `destroy()`, élimine les données périmées (garbage collection).

Construction et initialisation du servlet

Lorsque le moteur de servlets démarre ou lorsqu'un servlet a besoin de répondre à une demande, la méthode `init()` est appelée par le moteur de servlets pour indiquer au servlet qu'il va être mis en service. Le moteur de

servlets appellera `init()` une seule fois après l'instanciation du servlet. La méthode `init()` doit être finie pour que le servlet puisse recevoir des demandes.

Le paramètre `ServletConfig` de la méthode `init()` est un objet contenant les paramètres de configuration et d'initialisation du servlet. (Après l'initialisation du servlet, vous pouvez utiliser `getServletConfig()` pour récupérer ces informations.)

Lorsque le servlet a été chargé en mémoire, il peut résider dans un système de fichiers local, un système de fichiers distant ou sur un réseau.

Gestion des demandes client

Le moteur de servlets appelle la méthode `service()` pour permettre au servlet de répondre à une demande. Les objets demande et réponse sont transmis comme paramètres à la méthode `service()` lorsqu'un client fait une demande.

Le servlet peut aussi implémenter l'interface `ServletRequest` et/ou l'interface `ServletResponse` pour pouvoir accéder aux paramètres de la demande et aux données de la réponse. Les paramètres de la demande incluent des données ou des méthodes de protocole. Les données de la réponse incluent des en-têtes de réponse et des codes d'état.

Servlets et multithread

Habituellement, les servlets sont multithreads, ce qui permet à un seul servlet de traiter plusieurs demandes en même temps. En tant que développeur, vous devez protéger toutes les ressources partagées – comme les fichiers, les connexions au réseau, les variables de classe et d'instance du servlet – par rapport aux threads. Pour plus d'informations sur les threads et la sécurité par rapport aux threads, voir “Techniques de thread” dans *Introduction à Java*.

Notez que vous pouvez créer un servlet avec un seul thread, en utilisant l'interface `SingleThreadModel`. Cette interface permet au servlet de répondre à une seule demande à la fois. Habituellement, cela n'est pas très pratique pour les servlets. Si un servlet est limité à un thread, ce thread doit terminer successivement chaque tâche avant de passer à la suivante.

Destruction d'un servlet

Un moteur de servlets ne conserve pas un servlet chargé pendant une durée particulière ni pour la durée de vie du serveur. Les moteurs de servlets peuvent éliminer les servlets à tout moment. De ce fait, vous

devez programmer votre servlet sans qu'il stocke d'informations d'état. Pour libérer les ressources, utilisez la méthode `destroy()`.

Servlets orientés HTML

Les servlets peuvent facilement générer du texte formaté HTML, ce qui vous permet de les utiliser pour générer ou modifier dynamiquement des pages HTML. Avec la technologie servlet, vous n'avez pas besoin d'utiliser les langages de script. Par exemple, vous pouvez utiliser des servlets pour personnaliser ce que perçoit le visiteur d'un site web en modifiant en permanence la même page HTML.

Si votre serveur web supporte complètement les servlets, vous pouvez utiliser la balise `<servlet>` pour pré-traiter des pages web. Cette balise doit figurer dans un fichier utilisant l'extension `.shtml`. (Cette fonctionnalité est également appelée *côté serveur compris*.) La balise `<servlet>` indique au serveur web qu'un servlet pré-configuré doit être chargé et initialisé avec le jeu de paramètres de configuration spécifié. La sortie du servlet est incluse dans une réponse formatée HTML. Comme avec la balise `<applet>`, vous pouvez utiliser les attributs `class` et `codebase` pour spécifier les emplacements des servlets.

Voici un exemple de balise `<servlet>` :

```
<servlet>
  codebase=""
  code="dbServlet.Servlet1.class"
  param1=in
  param2=out
</servlet>
```

Notez que deux types de contenus de servlet (HTML et XHTML) créés par l'expert servlet de JBuilder utilisent la balise `<servlet>` dans un fichier `.shtml`.

Servlets spécifiques à HTTP

Les servlets utilisés avec le protocole HTTP (en étendant `HTTPServlet`), peuvent supporter n'importe quelle méthode HTTP. Ils peuvent rediriger les demandes vers d'autres emplacements et envoyer des messages d'erreur spécifiques à HTTP. De plus, ils peuvent accéder aux paramètres qui ont été transmis par les formes HTML standard, y compris la méthode HTTP à traiter et le URI identifiant la destination de la demande :

```
String méthode = request.getMethod();
String uri     = request.getRequestURI();
String nom     = request.getParameter("nom");
String téléphone = request.getParameter("téléphone");
```

```

String adresse = request.getParameter("adresse");
String ville   = request.getParameter("ville");

```

Pour les servlets spécifiques à HTTP, les données des demandes et des réponses sont fournies en tant que données formatées MIME. Le servlet spécifie le type des données et écrit les données encodées dans ce format. Cela permet à un servlet de recevoir les données saisies dans un certain type et de renvoyer ces données dans la forme appropriée à la demande.

Comment sont utilisés les servlets ?

Grâce aux robustes fonctionnalités de leur API, les servlets peuvent servir à de multiples usages :

- En tant que partie d'un système de saisie et de traitement des commandes, travaillant avec des bases de données clients, produits et stock. Par exemple, un servlet peut traiter des données, comme les données concernant une carte de crédit, POSTées sur HTTPS en utilisant une balise HTML <form>.
- En conjonction avec une applet, en tant que partie de l'intranet d'une entreprise, pour conserver les informations concernant les employés ou l'historique des salaires.
- En tant que partie d'un système collaboratif, comme un système de messagerie, où plusieurs servlets gèrent de multiples demandes de manière concurrentielle.
- En tant que partie d'un processus d'équilibrage des charges, où des servlets font suivre des demandes appartenant à une série.
- En tant que servlet orienté HTML, pour insérer des données formatées dans une page web à partir d'une requête dans une base de données ou d'une recherche web, ou pour créer des bannières publicitaires ciblées.

Déploiement des servlets

Habituellement, les servlets ne sont pas déployés de manière autonome sur un serveur web de production. Généralement, ils sont déployés en tant que module J2EE, l'unité de base de la composition d'une application J2EE. Un module J2EE sera constitué d'un ou de plusieurs composants J2EE de même type (web, EJB, client, etc.) qui partagent un même descripteur de déploiement. Une application J2EE sera constituée d'un ou de plusieurs modules J2EE.

Le descripteur de déploiement d'un module J2EE est un fichier XML (Extensible Markup Language). Le fichier descripteur de déploiement contient toutes les données déclaratives nécessaires au déploiement des

composants du module. Il contient également des instructions d'assemblage décrivant comment sont organisés les composants dans l'application. Il doit résider dans le même paquet que le servlet et que les autres composants web que vous empaquetez. Pour plus d'informations, voir le chapitre 8, "Application Assembly and Deployment", de Java 2 Platform Enterprise Edition, v1.3 Proposed Final Draft. Vous pouvez télécharger ce document depuis <http://java.sun.com/j2ee/download.html#platformspec>.

Vous pouvez utiliser JBuilder pour créer le descripteur de déploiement et le paquet de votre module J2EE ou de votre application. Pour plus d'informations, voir Chapitre 16, "Déploiement de votre application web".

Création de servlets dans JBuilder

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Dans JBuilder Professionnel et JBuilder Entreprise, vous pouvez utiliser l'expert servlet pour créer un fichier Java étendant `javax.servlet.http.HttpServlet`. L'expert servlet de JBuilder crée des servlets qui génèrent les types de contenu suivants :

- HTML – HyperText Markup Language.
- XHTML – une reformulation de HTML 4.0 en tant qu'application de XML.
- WML – Wireless Markup Language.
- XML – eXtensible Markup Language.

Avec l'expert servlet, vous pouvez aussi créer des servlets standard, des servlets filtre ou des servlets auditeur (si votre serveur web supporte les spécifications JavaServer Pages v1.2/Java Servlets v2.3).

Options de l'expert servlet

L'expert servlet facilite la création des servlets. Pour démarrer l'expert, choisissez Fichier | Nouveau. Cliquez sur l'onglet Web de la galerie d'objets, sélectionnez Servlet et cliquez sur OK.

Expert servlet – Page Choisir un nom et un type de servlet

Dans la première étape de l'expert servlet, vous pouvez choisir le paquet auquel appartiendra le servlet. Vous pouvez aussi entrer le nom du servlet dans le champ Classe. Si vous voulez répliquer les commentaires d'en-tête

ajoutés aux autres classes du projet (voir la rubrique Expert projet dans l'aide en ligne), choisissez l'option Générer les commentaires d'en-tête.

L'option Modèle de thread simple implémente l'interface SingleThreadModel. Choisir cette option peut rendre votre servlet moins efficace, car le serveur web va mettre les demandes en file d'attente et démarrer une autre instance du servlet pour servir la demande.

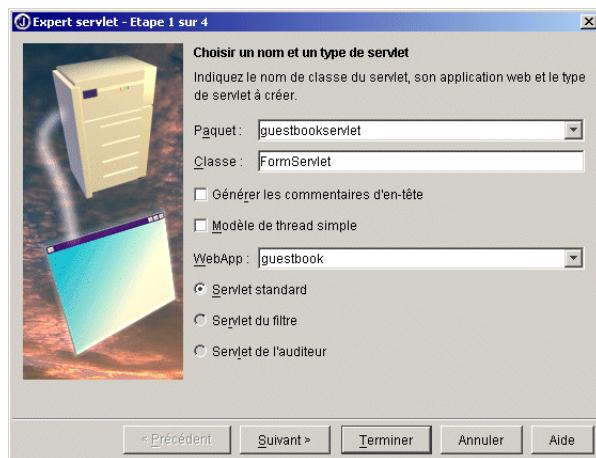
Sélectionnez le nom de la WebApp où vous voulez exécuter votre servlet, dans la liste déroulante WebApp. Tous les fichiers de contenu web, comme le fichier SHTML du servlet, seront placés dans le répertoire WebApp.

Les options situées dans la partie inférieure du dialogue vous permettent de choisir le type de servlet que vous voulez créer.

Remarque Ces options sont uniquement disponibles si votre serveur web supporte les spécifications JavaServer Pages v1.2/Java Servlets v2.3. (Tomcat 4.0 est l'implémentation de référence de ces spécifications.) Sinon, un servlet standard est créé par défaut.

Tableau 6.1 Options des types de servlet

Type de servlet	Description
Servlet standard	Un servlet qui n'est ni un filtre ni un auditeur. L'option Servlet standard est toujours disponible que vous ayez sélectionné la WebApp <défaut> ou une WebApp nommée. Lorsque cette option est sélectionnée pour la WebApp par défaut, le nom du servlet (voir "Expert servlet – Page des options de nom et d'affectation", page 6-6) est, par défaut, le nom de la classe du servlet, en minuscules. Si la WebApp est une WebApp nommée, l'URL du servlet (également sur la page des options de nom et d'affectation) se conforme par défaut au motif d'URL suivant : /nomservlet (tout en minuscules).
Servlet du filtre	Un servlet qui agit comme filtre pour un autre servlet ou pour la WebApp. Outre le nom, vous devez choisir un motif d'URL pour l'autre servlet (qui doit avoir un nom). Cette option est disponible uniquement si une WebApp nommée est sélectionnée. Si cette option est sélectionnée, la page Expert servlet – Page des options de nom et d'affectation, où vous nommez le servlet et spécifiez l'affectation du filtre, est la seule autre page disponible dans l'expert servlet.
Servlet auditeur	Un servlet qui est ajouté à la liste des auditeurs de la WebApp. Cette option est disponible uniquement si une WebApp nommée est sélectionnée. Si cette option est sélectionnée, la page Expert servlet – Saisie des détails du servlet de l'auditeur est la seule autre page disponible dans l'expert servlet.

Figure 6.1 Expert servlet – Page Choisir un nom et un type de servlet

Expert servlet – Page Modifier les détails du servlet standard

Si vous avez sélectionné Servlet standard comme type de servlet dans la page Expert servlet -- Choisir un nom et un type de servlet de l'expert servlet, la page Modifier les détails du servlet standard est l'étape 2 de l'expert. Cette page vous permet de sélectionner le type de contenu du servlet, les méthodes à implémenter et le fichier SHTML à générer.

Figure 6.2 Expert servlet – Page Modifier les détails du servlet standard

Option Générer le type de contenu

Vous pouvez utiliser la liste déroulante Générer le type de contenu pour choisir le type du contenu de votre servlet. Ces options comprennent :

- HTML – HyperText Markup Language. Un langage balisé pour les documents hypertexte sur Internet. HTML permet d'intégrer des images, des sons, de la vidéo, des champs de formulaire, des références à d'autres objets avec des URL et du formatage de texte.
- XHTML – Une reformulation de HTML 4.0 en tant qu'application de XML. Les balises et les attributs sont presque identiques à HTML, mais XHTML est une version du HTML plus rigoureuse. Par exemple, les balises XHTML sont toutes en minuscules et à chaque balise d'ouverture doit correspondre une balise de fermeture. XHTML permet aux concepteurs web d'aller vers un langage web plus modulaire et évolutif, basé sur XML, tout en conservant la compatibilité avec les navigateurs HTML 4 actuels.

Dans le fichier classe généré, .java, une ligne de code indiquant que DOC TYPE est XHTML est ajoutée et ressemble à ceci :

```
private static final String DOC_TYPE = "<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\n" +
" \\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\\"" ;
```

Du texte similaire est ajouté en haut du fichier SHTML, s'il a été généré.

- WML – Wireless Markup Language. Les fichiers WML sont des fichiers texte normaux avec du contenu XML. En raison de la largeur de bande limitée et de la capacité mémoire des postes mobiles, ce contenu doit être compilé dans un format binaire compact pour pouvoir être présenté sur une machine WAP (Wireless Application Protocol).

Pour les servlets WML, seul un fichier classe .java est généré. Il n'y a pas d'option pour un fichier SHTML avec un servlet de type WML. Le fichier classe généré, .java, contient une ligne de code indiquant que DOC TYPE est WML et une autre que CONTENT TYPE est WAP. Il ressemble à ceci :

```
private static final String CONTENT_TYPE = "text/vnd.wap.wml";
private static final String DOC_TYPE = "<!DOCTYPE wml
PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"\n" +
" \\"http://www.wapforum.org/DTD/wml12.dtd\\"" ;
```

Le code pour exécuter la demande GET du servlet en tant que servlet WML est également ajouté et ressemble à ceci :

```
out.println("<?xml version=\\"1.0\\"?>");
out.println(DOC_TYPE);
out.println("<wml>");
out.println("<card>");
out.println("<p>Le servlet a reçu un GET. Ceci est
la réponse.</p>");
out.println("</card></wml>");
```

- XML – eXtensible Markup Language. Un langage balisé qui vous permet de définir les balises requises pour identifier les données et le texte dans les documents XML. Pour plus d'informations sur le support de XML dans JBuilder, voir "Utilisation de XML dans JBuilder" dans le *Guide du développeur d'applications XML*.

Un fichier .java est généré. L'étape servant à créer un fichier SHTML est éliminée. Le fichier .java inclut du code pour identifier le servlet en tant que servlet XML. Il ressemble à ceci :

```
private static final String CONTENT_TYPE = "text/xml";
```

La méthode `doGet()` du servlet est modifiée à partir de la version HTML avec le code suivant :

```
out.println("<?xml version=\"1.0\"?>");
if (DOC_TYPE != null) {
    out.println(DOC_TYPE);
}
```

Options Implémenter les méthodes

Cette partie de l'expert servlet offre des options pour surcharger les méthodes standard du servlet. `HttpServlet` fournit une classe abstraite que vous pouvez sous-classer pour créer un servlet HTTP, qui reçoit des demandes depuis un serveur web et envoie des réponses à un client web. Quand vous sous-classez `HttpServlet`, vous devez surcharger au moins une méthode. Pour plus d'informations sur les méthodes, reportez-vous à la documentation sur les servlets, à l'adresse <http://java.sun.com/products/servlet/index.html>.

Les méthodes suivantes peuvent être générées automatiquement dans votre servlet :

- `doGet()` – Permet à un client de lire des informations sur le serveur web, en transmettant une chaîne de requête ajoutée à une URL pour indiquer au serveur l'information à renvoyer. Un GET se produit également lorsque vous tapez une adresse, cliquez sur un lien ou utilisez un signet. Surchargez cette méthode si le servlet supporte les demandes GET de HTTP.
- `doPost()` – Permet à un client d'envoyer au serveur web des données de longueur illimitée. Un POST se produit également lorsque vous cliquez sur le bouton Soumettre d'un navigateur. Surchargez cette méthode si le servlet supporte les demandes POST de HTTP.
- `doPut()` – Permet à un client de placer un fichier sur le serveur, est semblable à l'envoi d'un fichier au serveur par FTP. Surchargez cette méthode si le servlet supporte les demandes PUT de HTTP.
- `doDelete()` – Permet à un client de supprimer du serveur un document ou une page web. Surchargez cette méthode si le servlet supporte les demandes DELETE de HTTP.

Options Détails du fichier SHTML

Les options de détail du fichier SHTML sont affichées si vous avez sélectionné HTML ou XHTML dans la liste déroulante Générer le type de contenu. Si vous voulez appeler le servlet depuis une page HTML, comme décrit dans “Invocation d'un servlet depuis une page HTML”, page 6-11, sélectionnez l'option Créer un fichier SHTML. Un fichier SHTML, portant le même nom que le servlet, est ajouté au projet. Si vous avez sélectionné une WebApp dans la page Choisir un nom et un type de servlet de l'expert servlet, le fichier SHTML sera placé dans ce répertoire.

Si vous voulez exécuter le servlet directement, comme décrit dans “Invocation d'un servlet depuis la fenêtre d'un navigateur”, page 6-10, ne sélectionnez pas l'option Créer un fichier SHTML.

Choisissez la couleur de fond du fichier SHTML dans la liste déroulante Couleur du fond. Pour connecter le servlet à partir du fichier SHTML, vous pouvez utiliser la balise <servlet> ou la balise des liens <a href>.

- Si vous choisissez l'option Créer la balise <SERVLET>, le fichier SHTML exécute le servlet au moyen d'une balise <servlet>. La balise contient un `codebase` et une propriété `code`, un peu comme une applet. Par défaut, le `codebase` est défini par le dossier en cours et le `code` par le nom de classe du servlet.
- Si vous choisissez d'exécuter le servlet à partir d'un lien, une balise <a href> est placée dans votre fichier SHTML. La balise établit le lien vers le nom de la classe du servlet.

Expert servlet – Page des options de nom et d'affectation

Cette page vous permet de définir rapidement les affectations WebApp de base. (La WebApp sélectionnée est utilisée pour exécuter ou déboguer le servlet lorsque vous sélectionnez Exécution Web ou Débogage Web.)

L'expert Servlet ajoute automatiquement les affectations du servlet aux sections Servlet ou Filters du fichier descripteur de déploiement web.xml. Pour plus d'informations, voir “Servlets”, page 16-2, ou “Page Filtres”, page 16-7.

- Si vous avez sélectionné Servlet standard comme type de servlet dans la page Choisir un nom et un type de servlet de l'expert servlet, c'est l'étape 3.
- Si vous avez sélectionné Servlet filtre, c'est l'étape 2 et la dernière étape de l'expert servlet.

Pour plus d'informations sur l'affectation entre noms de servlets et motifs d'URL, voir “Comment les URL exécutent les servlets”, page 15-3.

Le champ Nom s'applique aux servlets standard et aux servlets filtre. Ce champ est affiché pour les servlets standard s'exécutant dans la WebApp <défaut> ou une WebApp nommée, et pour les servlets filtre.

Dans ce champ, entrez le nom du servlet. Pour un servlet standard, c'est le nom utilisé pour exécuter le servlet. Par convention, il est en minuscules.

Le Nom représente une façon rapide d'exécuter un servlet. Par exemple, au lieu de pointer votre navigateur sur le nom de classe complet du servlet s'exécutant sur le serveur web, il suffit de taper : `inputform`. Cela correspond au servlet nommé `formservlet`, qui à son tour correspond à la classe servlet `http://localhost:8080/servlet/FormServlet` s'exécutant en local sur le serveur web Tomcat dans l'EDI de JBuilder.

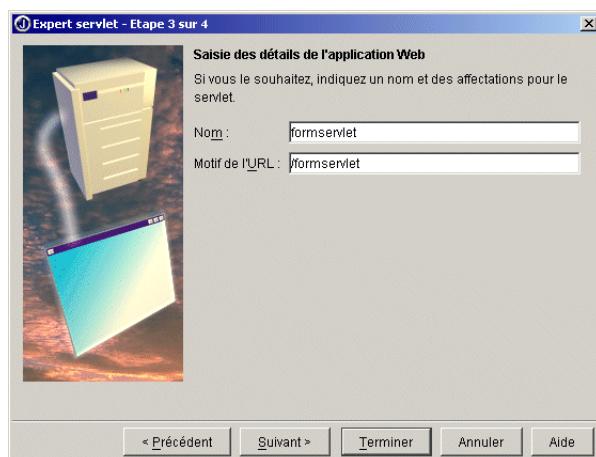
Pour les servlets standard et les servlets filtre, le champ Motif d'URL est l'endroit où vous choisissez le motif d'URL qui sera utilisé pour exécuter le servlet. Le motif d'URL est toujours en minuscules et précédé d'une barre oblique. Cela est utile lorsque vous invoquez le servlet directement, à partir d'un navigateur. (Voir "Invocation d'un servlet depuis la fenêtre d'un navigateur", page 6-10, pour plus d'informations.)

Un motif d'URL par défaut est proposé. Pour un servlet standard, le motif par défaut est constitué du nom de classe du servlet en minuscules précédé d'une barre oblique : `/formservlet`. Il doit correspondre au nom complet de la classe du servlet. Si le servlet est un servlet filtre, le champ Motif d'URL a par défaut la valeur `/*`, ce qui signifie que le filtre est appliqué à toutes les demandes.

Remarque

Ce champ n'est pas disponible si vous n'avez pas sélectionné de WebApp dans la page Choisir un nom et un type de servlet de l'expert servlet.

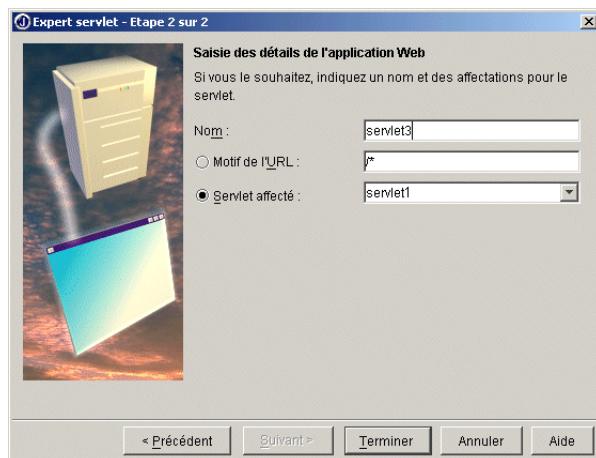
Figure 6.3 Expert servlet – Page des options de nom et d'affectation d'un servlet standard



Pour les servlets filtre, vous pouvez choisir comment est mappé le servlet – ce peut être au moyen du motif d'URL ou de l'option Servlet affecté.

La liste déroulante Servlet affecté vous permet de choisir un autre servlet dans votre projet qui sera filtré par ce servlet. (Par défaut, ce champ contient le nom du premier servlet de votre projet, en minuscules, précédé par une barre oblique.) Par exemple, si votre projet contient déjà `Servlet1.java` et `Servlet2.java`, qui sont tous les deux des servlets standard, le champ Servlet affecté prendra par défaut la valeur `/servlet1`. La liste déroulante montre tous les autres servlets du projet précédemment affectés. S'il en existe pas, l'option est désactivée.

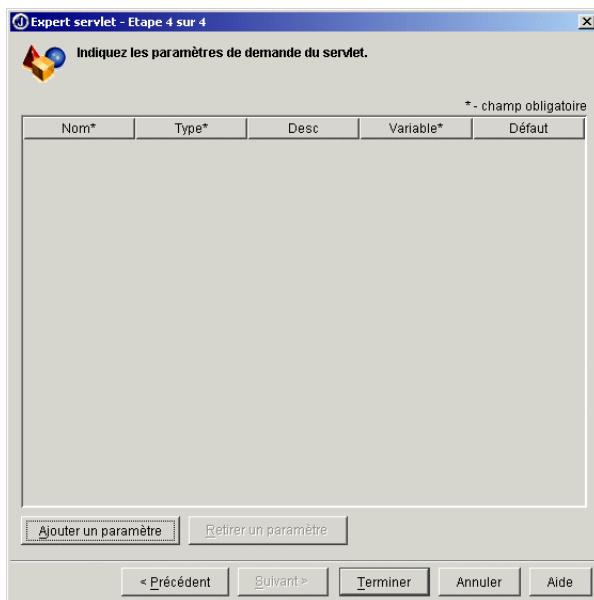
Figure 6.4 Expert servlet – Page des options de nom et d'affectation d'un servlet Filtre



Expert servlet – Page Paramètres

La page Paramètres de l'expert servlet est l'emplacement où vous saisissez les paramètres du servlet. Les paramètres sont des valeurs transmises au servlet. Les valeurs peuvent être de simples valeurs saisies. Mais, elles peuvent aussi bien affecter la logique de l'exécution du servlet. Par exemple, la valeur entrée par l'utilisateur peut déterminer la table de base de données qui sera affichée ou mise à jour. Ou encore, la valeur saisie par l'utilisateur peut déterminer la couleur du fond du servlet.

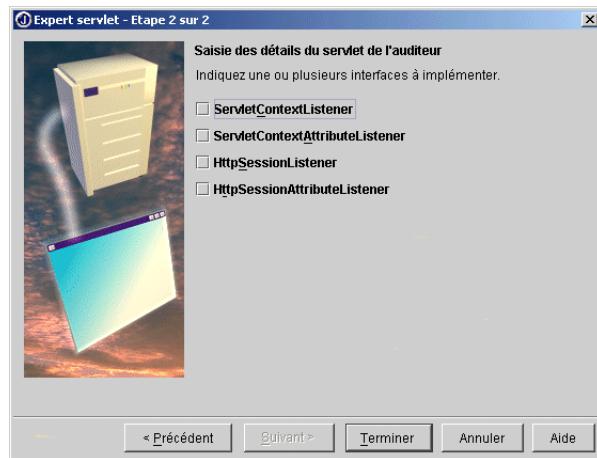
Le servlet dans le Chapitre 7, “Tutoriel : Création d'un servlet simple”, utilise un paramètre de type `String` pour permettre la saisie du nom de l'utilisateur. Le nom du paramètre dans le fichier SHTML est `UserName` ; la variable correspondante, utilisée dans le fichier `servlet.java`, est `userName`.

Figure 6.5 Expert servlet – Page Paramètres

Expert servlet – Page Saisie des détails du servlet de l'auditeur

Cette page n'est disponible que si vous avez sélectionné Servlet de l'auditeur comme type de servlet dans la page Choisir un nom et un type de servlet de l'expert servlet. C'est l'étape 2, et la dernière étape de l'expert servlet pour les servlets auditeur. Utilisez cette page pour implémenter une ou plusieurs interfaces de servlet auditeur. Les méthodes correspondantes sont ajoutées au servlet.

L'expert Servlet ajoute automatiquement les auditeurs sélectionnés à la section `Listeners` du fichier descripteur de déploiement `web.xml`. Pour plus d'informations, voir "Page Auditeurs", page 16-9.

Figure 6.6 Expert servlet – Page Saisie des détails du servlet de l'auditeur

Invocation des servlets

Les rubriques suivantes donnent quelques moyens d'invoquer les servlets :

- Invocation d'un servlet depuis la fenêtre d'un navigateur
- Invocation d'un servlet depuis une page HTML

Invocation d'un servlet depuis la fenêtre d'un navigateur

Un servlet peut être appelé directement si on tape leur URL dans la fenêtre d'adresses d'un navigateur. Le format général de l'URL d'un servlet, où *nom-classe-servlet* correspond au nom de classe du servlet, est :

`http://nom-machine:numéro-port/servlet/nom-classe-servlet`

Par exemple, toute URL utilisant le format des exemples suivants vous permettrait d'exécuter un servlet :

- `http://localhost:8080/servlet/Servlet1` (exécution en local sur votre ordinateur)
- `http://www.borland.com/servlet/Servlet1` (exécution depuis cette URL)
- `http://127.0.0.1/servlet/Servlet1` (exécution depuis cette adresse IP)

Remarque Si vous omettez le numéro de port, le protocole HTTP utilise par défaut le port 80. La première URL de l'exemple précédent devrait fonctionner dans l'EDI si vous avez défini le port de la configuration d'exécution par 80 et si vous n'exécutez pas déjà un serveur web sur ce port. Les autres exemples devraient fonctionner en situation réelle, après le déploiement sur un serveur web de l'application.

Les URL des servlets peuvent contenir des requêtes pour les demandes HTTP GET. Par exemple, le servlet du Chapitre 7, “Tutoriel : Création d’un servlet simple”, peut être exécuté en entrant l’URL suivante pour le servlet :

```
http://localhost:8080/servlet/simple servlet.Servlet1?userName=Mary
```

`simple servlet.Servlet1` est le nom complet de la classe. Le `?` indique qu’une chaîne de requête est ajoutée à l’URL. `userName=Mary` est le nom et la valeur du paramètre.

Si le servlet a utilisé le nom `firstservlet`, vous pouvez entrer :

```
firstservlet?userName=Mary
```

Pour plus d’informations sur l’exécution des servlets, voir “Comment les URL exécutent les servlets”, page 15-3.

Invocation d’un servlet depuis une page HTML

Pour invoquer un servlet depuis une page HTML, utilisez simplement l’URL du servlet dans la balise HTML appropriée. Les balises qui acceptent des URL sont celles qui débutent les ancrages ou les formulaires, et les balises meta. Les URL de servlets peuvent être utilisées dans les balises HTML chaque fois qu’une URL normale peut l’être, comme la destination d’un ancrage, comme l’action d’un formulaire, et comme l’emplacement utilisé lorsqu’une balise meta commande le rafraîchissement d’une page. Cette section suppose une bonne connaissance de HTML. Vous pouvez vous familiariser avec HTML dans de nombreux manuels, ou en consultant *HTML 3.2 Reference Specification* sur le web, à l’adresse <http://www.w3c.org/TR/REC-html32.html>.

Par exemple, dans le Chapitre 7, “Tutoriel : Création d’un servlet simple”, la page SHTML contient un ancrage avec un servlet comme destination :

```
<a href="/servlet/simple servlet.Servlet1">Cliquez ici pour appeler le servlet :  
Servlet1</a><br>
```

Les pages HTML utilisent également les balises suivantes pour invoquer les servlets :

- Une balise `<form>`

```
<form action="http://localhost:8080/servlet/simple servlet.Servlet1  
method="post">
```

- Une balise meta qui utilise une URL de servlet comme partie de la valeur de l’attribut `http-equiv`.

```
<meta http-equiv="refresh" content="4;url=http://localhost:8080/servlet/  
simple servlet.Servlet1;">
```

Notez que vous pouvez substituer le motif d’URL du servlet au nom de classe complet. Par exemple, si vous avez attribué au servlet le nom de

inputform (voir "Expert servlet – Page des options de nom et d'affectation", page 6-6), vous pouvez utiliser la balise <form> suivante pour exécuter le servlet :

```
<form action="inputform" method="post">
```

Internationalisation des servlets

Les servlets offrent un procédé d'internationalisation intéressant. Comme le servlet produit du source HTML chez le client, si ce code HTML contient des caractères n'appartenant pas au jeu de caractères supporté par le serveur sur lequel le servlet s'exécute, ces caractères risquent d'être illisibles dans le navigateur du client. Par exemple, si l'encodage du client est défini par ISO-8859-1, et si le code HTML écrit par le servlet contient des caractères sur deux octets, ces caractères n'apparaîtront pas correctement dans le navigateur du client, même si le navigateur est correctement défini pour les voir.

En spécifiant un encodage dans le servlet, le servlet peut être déployé sur n'importe quel serveur sans besoin de connaître l'encodage de celui-ci. Les servlets peuvent également répondre à l'entrée des utilisateurs et écrire du code HTML dans la langue souhaitée.

L'exemple suivant montre comment spécifier une définition d'encodage dans un servlet. Dans le source Java généré par l'expert servlet, la méthode `doPost()` contient la ligne suivante :

```
PrintWriter out = response.getWriter();
```

Cette ligne peut être remplacée par :

```
OutputStreamWriter writer = new OutputStreamWriter(  
    response.getOutputStream(), "encoding");  
PrintWriter out = new PrintWriter(writer);
```

Le second argument du constructeur `OutputStreamWriter` est une chaîne représentant l'encodage souhaité. La chaîne peut être ressourcée, codée hard, ou définie par une variable. Un appel à `System.getProperty("file.encoding")` renverra une `String` représentant la définition d'encodage du système en cours d'utilisation.

Si le constructeur `OutputStreamWriter` est appelé avec seulement le premier argument, l'encodage en cours sera utilisé.

Ecriture d'un servlet orienté données

Les technologies de développement pour le web dans JBuilder comprennent l'API InternetBeans Express qui simplifie la création de servlets orientés données. InternetBeans Express est un ensemble de composants qui lisent des formulaires HTML et génèrent du HTML à

partir de modèles de données DataExpress, ce qui facilite la création de servlets et de JSP orientés données. Les composants génèrent du HTML et sont utilisés avec les servlets et les JSP pour créer du contenu dynamique. Ils apportent des connexions et des optimisations spécifiques lorsqu'ils sont utilisés en conjonction avec les composants DataExpress, mais ils peuvent tout aussi bien être utilisés avec les modèles de données Swing génériques.

Pour plus d'informations sur InternetBeans Express, voir Chapitre 11, "Utilisation d'InternetBeans Express". Vous pouvez également vous reporter au Chapitre 12, "Tutoriel : Création d'un servlet avec InternetBeans Express".

Tutoriel : Cr éation d'un servlet simple

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Ce tutoriel montre comment créer un servlet élémentaire qui accepte la saisie de l'utilisateur et compte le nombre de visiteurs d'un site web. Vous pouvez développer et tester des servlets dans JBuilder en utilisant le moteur de servlets Tomcat, livré avec JBuilder. Ce tutoriel montre comment développer et exécuter un servlet dans JBuilder.

Pour aborder le développement d'un servlet Java, vous allez concevoir une application relativement simple (du type Hello World) qui illustre l'organisation générale d'un servlet. Ce servlet affiche un message de bienvenue, le nom de l'utilisateur et le nombre de connexions depuis le démarrage du servlet.

Tous les servlets sont conçus en étendant la classe de base `Servlet` et en définissant des méthodes Java pour gérer les connexions entrantes. Ce servlet exemple étend la classe `HttpServlet` qui gère le protocole HTTP du Web et l'essentiel des fonctions nécessaires à une application Web.

Pour construire `SimpleServlet`, nous allons utiliser l'expert servlet afin d'étendre la classe de base `HttpServlet`. Puis, nous définirons une méthode produisant en sortie quelques lignes de HTML contenant le nom de l'utilisateur.

Pour plus d'informations sur les servlets, voir les chapitres suivants :

- Chapitre 5, "Utilisation des servlets"
- Chapitre 6, "Cr éation de servlets dans JBuilder"

Ce tutoriel suppose que vous vous êtes familiarisé avec Java et avec l'EDI de JBuilder. Pour plus d'informations sur Java, voir *Introduction à Java*.

Pour plus d'informations sur l'EDI de JBuilder, voir "L'environnement de JBuilder" dans *Construction d'applications avec JBuilder*.

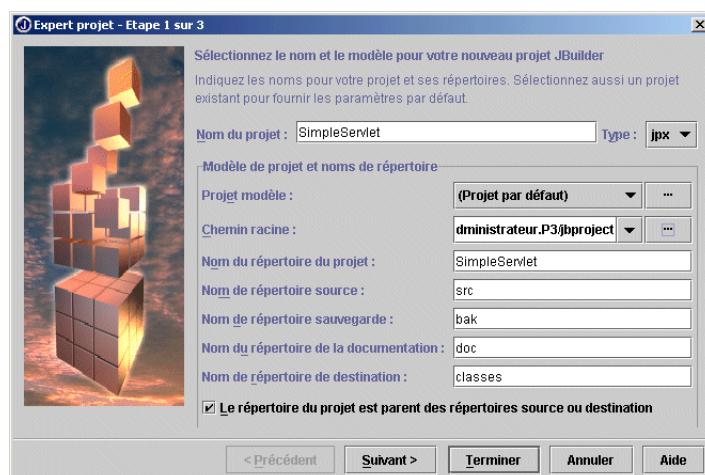
Si vous avez des suggestions à faire pour améliorer ce tutoriel, envoyez un e-mail à jgpubs@borland.com.

Etape 1 : Création du projet

Pour développer le servlet exemple Hello World dans JBuilder, vous devez d'abord créer un nouveau projet. Pour ce faire,

- 1 Sélectionnez Fichier | Nouveau projet pour afficher l'expert projet.
- 2 Entrez SimpleServlet dans le champ Nom du projet.
- 3 Vérifiez que l'option Le répertoire du projet est parent des répertoires source ou destination est sélectionnée.

L'étape 1 de l'expert projet doit ressembler à ceci :



- 4 Cliquez sur le bouton Terminer pour fermer l'expert et créer le projet. Vous n'avez pas à modifier les valeurs par défaut des étapes 2 et 3 de l'expert.

Le fichier projet `SimpleServlet.jpx` et le fichier HTML du projet sont affichés dans le volet projet.

Dans l'étape suivante, vous allez créer une WebApp pour votre servlet. Bien que nous n'allions pas déployer ce projet, dans la réalité, vous devrez toujours créer une WebApp.

Etape 2 : Création de la WebApp

Lorsque vous développez des applications web, une des premières étapes consiste à créer une WebApp, la collection des fichiers de contenu web de votre application web. Pour créer une WebApp,

- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Cliquez sur l'onglet Web et choisissez Application Web. Cliquez sur OK.

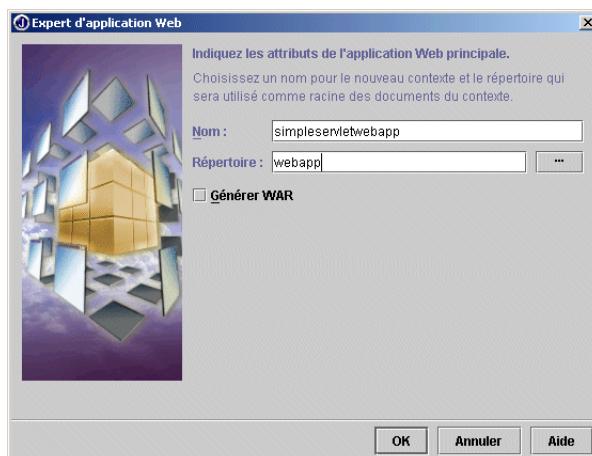
L'expert Application web est affiché.

- 2 Entrez simpleservletwebapp dans le champ Nom.

- 3 Entrez webapp dans le champ Répertoire.

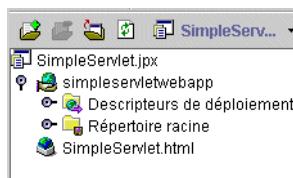
- 4 Vérifiez que l'option Générer WAR n'est pas sélectionnée.

L'expert Application web doit ressembler à ceci :



- 5 Cliquez sur OK pour fermer l'expert et créer la WebApp.

La WebApp simpleservletwebapp se présente sous la forme d'un nœud dans le volet projet. Développez ce nœud pour voir les nœuds Descripteurs de déploiement et Répertoire racine.



Pour plus d'informations sur les WebApps, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

Dans l'étape suivante, vous allez créer le servlet.

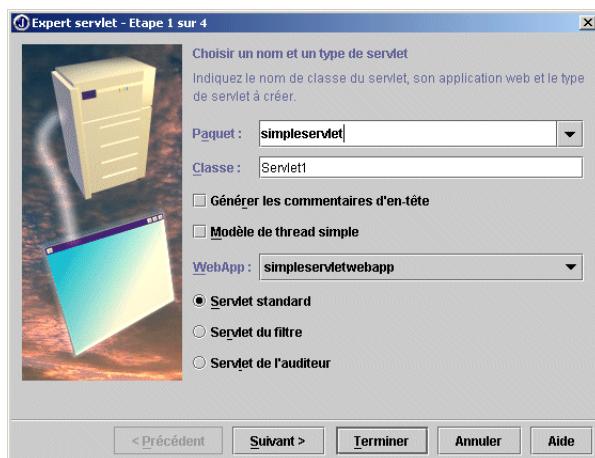
Etape 3 : Création du servlet avec l'expert servlet

Dans cette étape, vous allez créer le servlet en utilisant l'expert servlet. Vous utiliserez l'expert pour :

- Entrer le nom de la classe du servlet.
- Choisir le type du servlet et son type de contenu.
- Choisir les méthodes HTTP à redéfinir.
- Créer un fichier SHTML pour exécuter le servlet.
- Créer les paramètres du servlet.

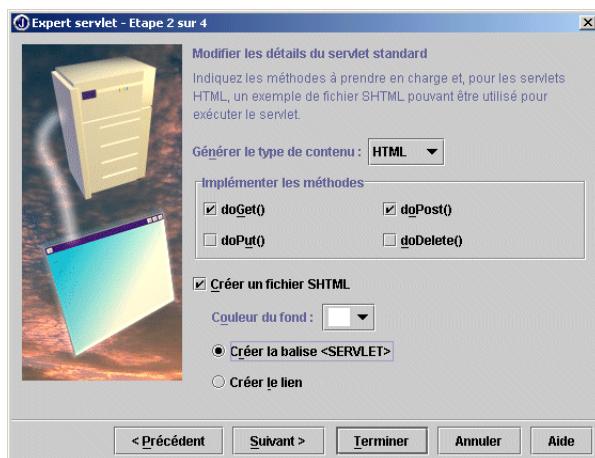
Pour créer le servlet,

- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets.
- 2 Cliquez sur l'onglet Web et choisissez Servlet. Cliquez sur OK. L'étape 1 de l'expert servlet s'affiche.
- 3 Acceptez les noms de paquet et de classe par défaut. Choisissez simpleservletwebapp dans la liste déroulante WebApp. Acceptez toutes les autres valeurs par défaut. L'étape 1 doit ressembler à ceci :

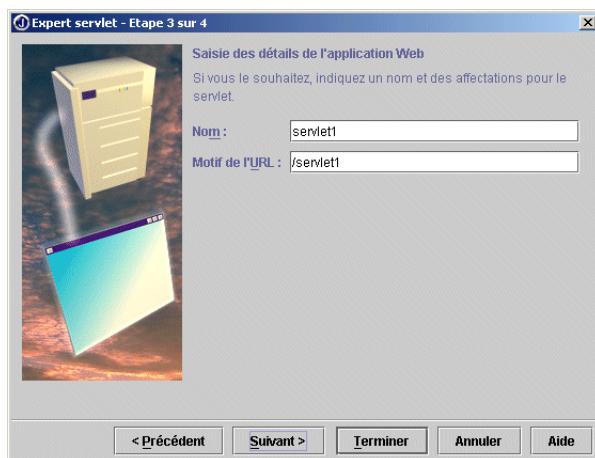


- 4 Cliquez sur Suivant pour aller à l'étape 2.
- 5 A l'étape 2 de l'expert, sélectionnez la méthode `doPost()`. Assurez-vous que la méthode `doGet()` est sélectionnée. Sélectionnez l'option Créez un

fichier SHTML et l'option Crée la balise <SERVLET>. L'étape 2 doit ressembler à ceci :



- 6 Cliquez sur Suivant pour aller à l'étape 3.
- 7 Acceptez le nom et le motif de l'URL proposés par défaut à l'étape 3 de l'expert. L'étape 3 doit ressembler à ceci :



- 8 Cliquez sur Suivant pour aller à l'étape 4.
- 9 Cliquez sur le bouton Ajouter un paramètre pour créer un nouveau paramètre. Ce paramètre contient le nom entré dans le champ de saisie de texte du servlet.

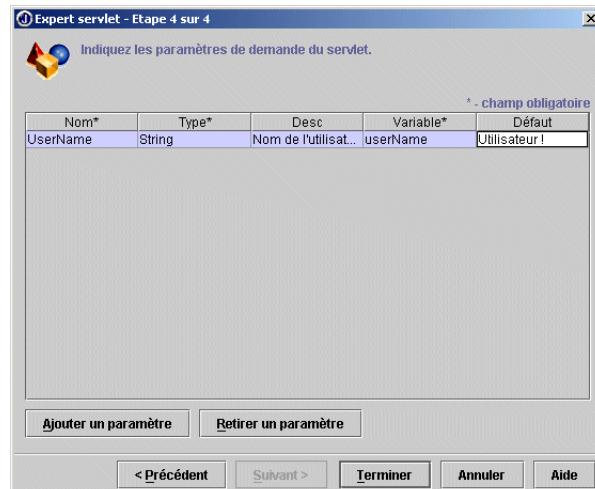
Etape 3 : Création du servlet avec l'expert servlet

Le tableau suivant décrit les champs et les valeurs requises. Vous devez entrer une valeur dans la colonne Valeur du tableau suivant.

Tableau 7.1 Options des paramètres de l'expert servlet

Nom du paramètre	Valeur	Description
Nom (obligatoire)	UserName	Le paramètre utilisé dans la balise <form> du fichier SHTML. Il contient la chaîne entrée par l'utilisateur dans le champ de saisie de texte de la fiche.
Type (obligatoire)	String	Le type de la variable pour le langage Java. (Il s'agit de la valeur par défaut et elle est déjà sélectionnée.)
Desc	Nom de l'utilisateur	Le commentaire qui est ajouté au code source de votre servlet.
Variable (obligatoire)	userName	Le nom de la variable utilisée dans <code>Servlet1.java</code> pour contenir le nom de l'utilisateur qui lui a été transmis par le fichier SHTML.
Défaut	Utilisateur !	La valeur par défaut de la variable <code>userName</code> .

Quand elle est terminée, l'étape 4 de l'expert doit ressembler à ceci :



10 Appuyez sur Terminer pour créer le servlet.

Les fichiers `Servlet1.java` et `Servlet1.shtml` sont ajoutés à votre projet. Notez que `Servlet1.shtml` a été ajouté au nœud Répertoire racine de la WebApp `simpleservletwebapp`. La bibliothèque `Servlet` est ajoutée à la liste des Bibliothèques nécessaires dans la page Chemins de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet).

Les commandes Exécution Web et Débogage Web sont activées pour le servlet et son fichier SHTML.

- 11** Choisissez Fichier | Tout enregistrer pour enregistrer votre travail.
A l'étape suivante, vous ajouterez du code à `Servlet1.java`.

Etape 4 : Ajout de code au servlet

A cette étape, vous ajouterez du code à `Servlet1.java`. Le code crée un compteur du nombre de fois où la page a été consultée et affiche ce compte.

- 1** Ouvrez `Servlet1.java` dans l'éditeur et utilisez la commande Chercher | Chercher pour trouver le commentaire `/**Initialiser les variables globales**/` au début du fichier. Immédiatement avant cette ligne, entrez la ligne de code suivante :

```
int connections = 0;
```

Cette ligne de code crée la variable `connections` et l'initialise à zéro.

- 2** Recherchez la ligne de code contenant la chaîne :

Le servlet a reçu un POST. Ceci est la réponse

Immédiatement après cette ligne, entrez le code suivant :

```
out.println("<p>Merci pour la visite, ");
out.println(request.getParameter("UserName"));
out.println("<p>");
out.println("Hello World - mon premier programme de servlet Java !");
out.println("<p>Vous êtes le visiteur numéro ");
out.println(Integer.toString(++connections));
```

Ces lignes de code obtiennent le paramètre `UserName` et l'affichent dans une instruction `out.println`. Le code incrémente ensuite le nombre de visiteurs et l'affiche.

- 3** Choisissez Fichier | Tout enregistrer pour enregistrer votre travail.

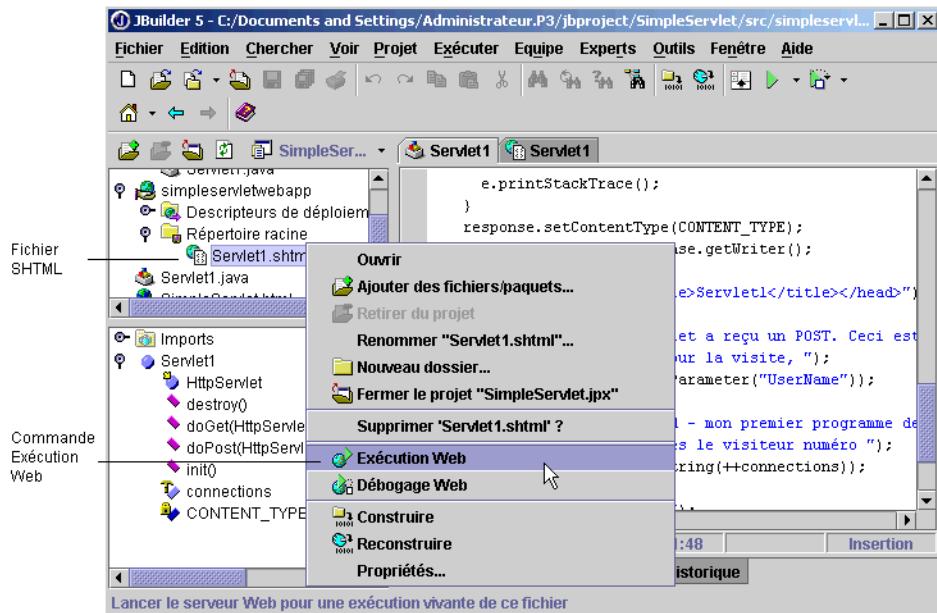
Dans l'étape suivante, vous allez compiler et exécuter le servlet.

Etape 5 : Compilation et exécution du servlet

Pour compiler et exécuter le servlet,

- 1** Choisissez Projet | Construire le projet “SimpleServlet.jpx”.
- 2** Cliquez avec le bouton droit de la souris sur `Servlet1.shtml` dans le volet projet.

3 Choisissez Exécution Web.

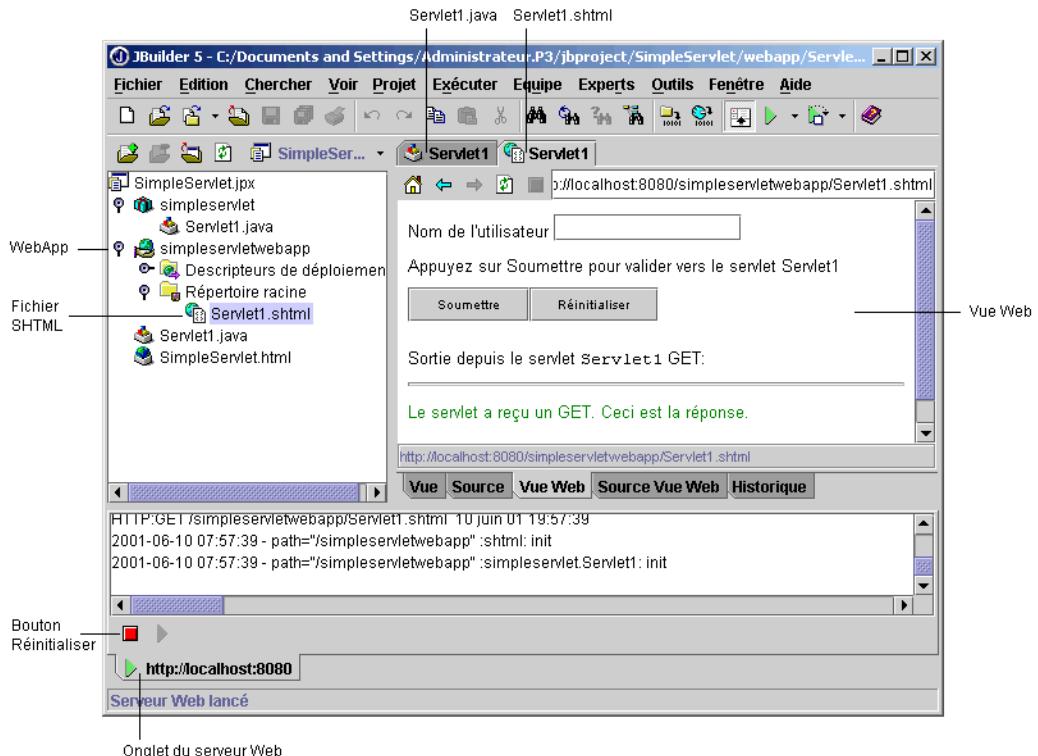


Remarque Vous pouvez aussi exécuter les servlets en cliquant avec le bouton droit sur le fichier java dans le volet projet et en sélectionnant Exécution Web. Dans cet exemple, nous exécutons le servlet depuis le fichier SHTML car c'est là que sont programmés les champs de saisie des paramètres et le bouton Soumettre, selon les sélections que vous avez effectuées dans l'expert Servlet.

L'exécution du fichier SHTML démarre Tomcat, le serveur web inclus avec JBuilder. Les sorties issues de Tomcat s'affichent dans le volet message. Les commandes HTTP et les valeurs des paramètres seront aussi envoyées dans ce panneau de sortie. Deux nouveaux onglets apparaissent dans le volet contenu pour le servlet : Vue Web et Source Vue Web.

Le servlet en train de s'exécuter s'affiche dans la Vue Web. Il ressemble à ceci :

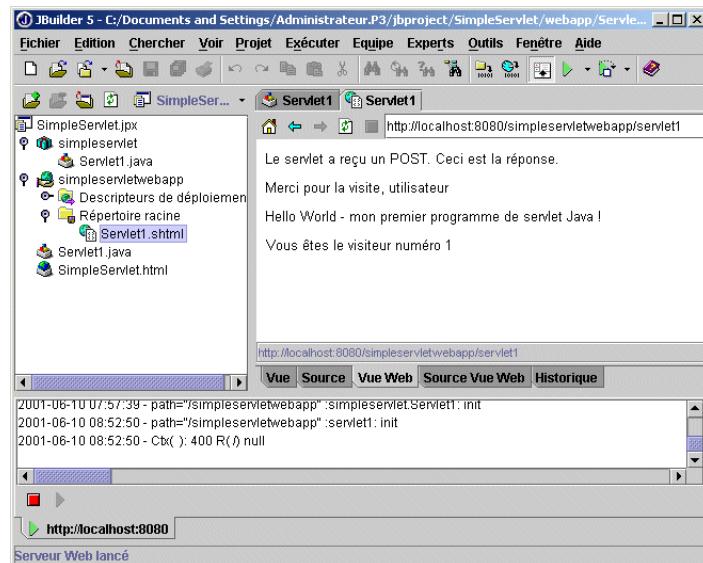
Figure 7.1 Servlet s'exécutant dans la Vue Web



Pour exécuter le servlet, entrez un nom dans la boîte texte, puis cliquez sur le bouton Soumettre. La méthode `doPost()` est appelée, et la réponse s'affiche dans la Vue Web, comme illustré ci-après.

Etape 5 : Compilation et exécution du servlet

Figure 7.2 Servlet s'exécutant après la soumission du nom



Pour exécuter à nouveau le servlet et voir augmenter le nombre de connexions, cliquez sur la flèche vers l'arrière en haut de la vue web. Entrez un autre utilisateur et cliquez sur le bouton Soumettre. Lorsque la réponse de la méthode `doPost()` sera affichée, vous verrez que le nombre de connexions a augmenté.

Pour arrêter le serveur web, cliquez sur le bouton Réinitialiser le programme directement au-dessus de l'onglet du serveur web. Si vous apportez des changements à votre code source, vous devez arrêter le serveur web avant de le recompiler et de l'exécuter à nouveau.

Félicitations ! Vous avez terminé votre premier programme servlet. Pour des servlets plus puissants qui se connectent à une base de données, voir Chapitre 8, "Tutoriel : Création d'un servlet mettant à jour un guestbook".

8

Tutoriel : Création d'un servlet mettant à jour un guestbook

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Ce tutoriel montre comment créer un servlet qui accepte la saisie de l'utilisateur et enregistre des données dans une base de données JDataStore.

Lorsque vous aurez fini ce tutoriel, votre projet contiendra les classes suivantes :

- `FormServlet.java` – La classe exécutable du programme. Sa méthode `doGet()` affiche un formulaire de saisie utilisant une balise HTML `<form>`. Les servlets postent les valeurs de l'utilisateur (via des paramètres) dans `DBServlet`.
- `DBServlet.java` – Ce servlet passe des valeurs de paramètres (dans sa méthode `doPost()`) à `ModuleDeDonnees1`. Le code de la méthode `doGet()` affiche le JDataStore Guestbook en tant que table HTML.
- `ModuleDedonnees1.java` – Le module de données contenant la logique métier du programme. Le code du module de données connecte le JDataStore Guestbook, le met à jour, et enregistre les modifications.

Vous exécuterez la classe exécutable, `FormServlet`, depuis la WebApp `guestbook`, en utilisant le motif d'URL `/inputform` au lieu du nom de la classe. Vous appellerez `DBServlet` depuis la balise `<form>` de `FormServlet` en utilisant le nom `table` du servlet au lieu du nom de la classe.

Ce tutoriel suppose que vous vous êtes familiarisé avec les servlets ainsi qu'avec les technologies JDataStore et DataExpress de JBuilder. Pour plus d'informations sur les servlets, voir les chapitres suivants :

- Chapitre 5, "Utilisation des servlets"
- Chapitre 6, "Création de servlets dans JBuilder"

Etape 1 : Création du projet

Vous pouvez vous entraîner sur un servlet moins complexe, "Hello World" – voir Chapitre 7, "Tutoriel : Création d'un servlet simple", pour plus d'informations. Pour plus d'informations sur les fonctionnalités de bases de données de JBuilder, voir le *Guide du développeur d'applications de bases de données*. Pour plus d'informations sur JDataStore, voir le *Guide du développeur JDataStore*.

Remarque

Ce tutoriel suppose que vous avez entré vos informations de licence dans le Gestionnaire de licence JDataStore. Pour plus d'informations, voir "Première utilisation de JDataStore" dans le *Guide du développeur JDataStore*.

Les classes terminées de ce tutoriel se trouvent dans le répertoire samples/WebApps/GuestbookServlet de votre installation de JBuilder.

Si vous avez des suggestions à faire pour améliorer ce tutoriel, envoyez un e-mail à jpgpubs@borland.com.

Etape 1 : Création du projet

Pour développer le servlet exemple Guestbook dans JBuilder, vous devez d'abord créer un nouveau projet. Pour ce faire,

- 1 Sélectionnez Fichier | Nouveau projet pour afficher l'expert projet.
- 2 Entrez GuestbookServlet dans le champ Nom du projet.
- 3 Vérifiez que l'option Le répertoire du projet est parent des répertoires source ou destination est sélectionnée.

L'étape 1 de l'expert projet doit ressembler à ceci :



- 4** Cliquez sur le bouton Terminer pour fermer l'expert et créer le projet. Vous n'avez pas à modifier les valeurs par défaut des étapes 2 et 3 de l'expert.

Le fichier projet `GuestbookServlet.jpx` et le fichier HTML du projet sont affichés dans le volet projet.

Etape 2 : Création de la WebApp

Lorsque vous développez des applications web, une des premières étapes consiste à créer une WebApp, la collection des fichiers de contenu web de votre application web. Pour créer une WebApp,

- 1** Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Cliquez sur l'onglet Web et choisissez Application Web. Cliquez sur OK.

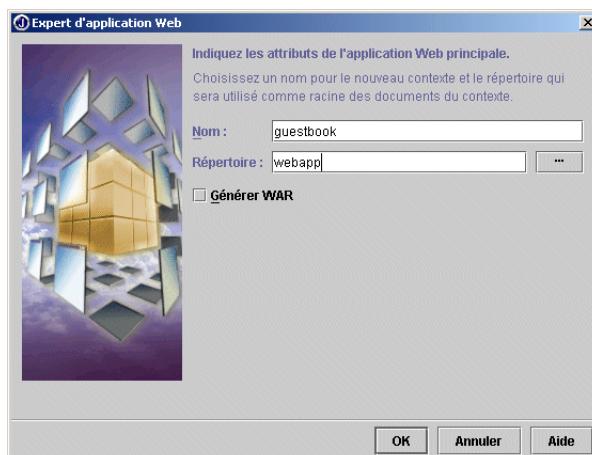
L'expert Application web est affiché.

- 2** Entrez `guestbook` dans le champ Nom.

- 3** Entrez `webapp` dans le champ Répertoire.

- 4** Vérifiez que l'option Générer WAR n'est pas sélectionnée.

L'expert Application web doit ressembler à ceci :



- 5** Cliquez sur OK pour fermer l'expert.

Etape 3 : Création des servlets

La WebApp guestbook se présente sous la forme d'un nœud dans le volet projet. Développez ce nœud pour voir les nœuds Descripteurs de déploiement et Répertoire racine.



Pour plus d'informations sur les WebApps, voir Chapitre 3, “Utilisation des WebApps et des fichiers WAR”.

Etape 3 : Création des servlets

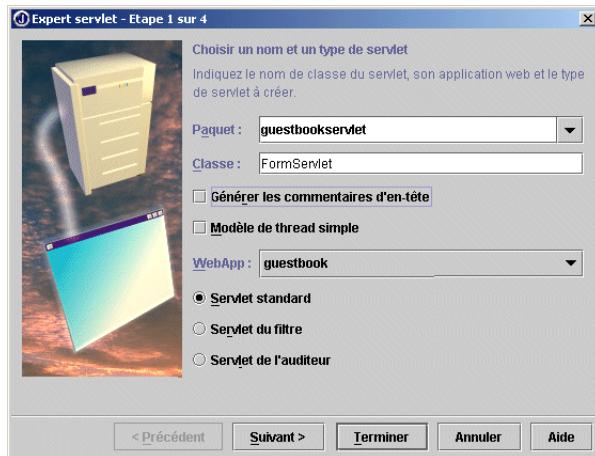
Au cours de cette étape, vous allez créer deux servlets dans le projet :

- `FormServlet.java` - La classe exécutable du programme. Sa méthode `doGet()` affiche un formulaire de saisie utilisant une balise HTML `<form>`. Les servlets postent les valeurs de l'utilisateur (via des paramètres) dans `DBServlet`.
- `DBServlet.java` - Ce servlet passe des valeurs de paramètres (dans sa méthode `doPost()`) à `ModuleDeDonnees1`. Le code de la méthode `doGet()` affiche le JDataStore Guestbook en tant que table HTML.

Pour créer `FormServlet.java`,

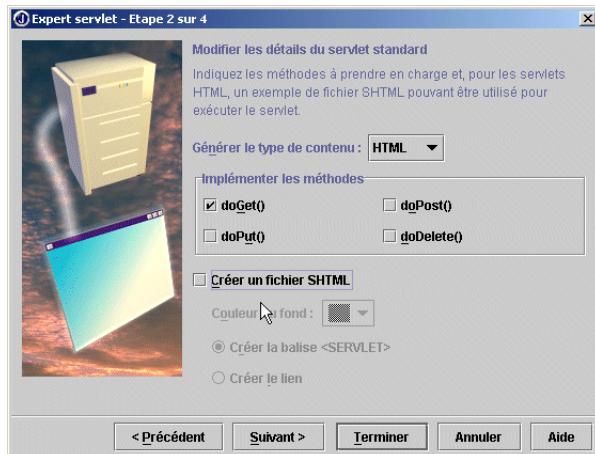
- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Cliquez sur l'onglet Web et choisissez Servlet. Cliquez sur OK.
L'expert servlet est affiché.
- 2 Laissez le nom du paquet défini par `guestbookservlet`. Changez le champ Classe en `FormServlet`.
- 3 Vérifiez que les options Générer les commentaires d'en-tête et Modèle de thread simple ne sont pas sélectionnées. Choisissez `guestbook` dans la liste déroulante WebApp. (Vous venez de créer cette WebApp dans l'étape précédente.) Conservez sélectionnée l'option Servlet standard.

L'étape 1 de l'expert doit ressembler à ceci :



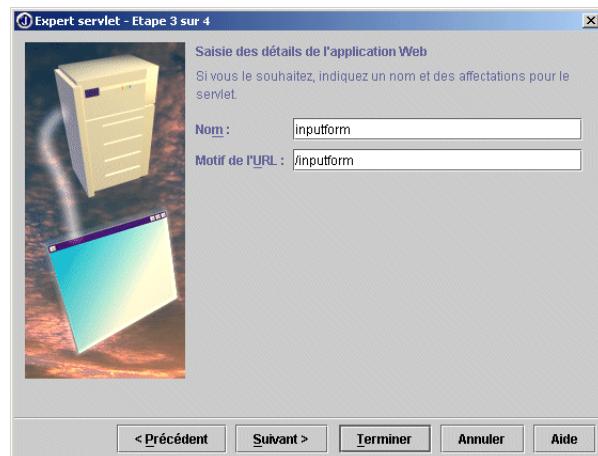
- 4 Cliquez sur Suivant pour passer à l'étape 2 de l'expert.
- 5 Vérifiez que l'option Générer le type de contenu est définie par HTML et que la méthode doGet() est sélectionnée. Ne sélectionnez pas l'option Créer un fichier SHTML.

Quand elle est terminée, l'étape 2 de l'expert doit ressembler à ceci :



- 6 Cliquez sur Suivant pour passer à l'étape 3 de l'expert.
- 7 Changez la valeur par défaut du champ Nom en inputform. Changez la valeur du champ Motif de l'URL en /inputform. Vérifiez que les entrées sont tout en minuscules. L'entrée Motif de l'URL sera utilisée à la place du nom de classe pour exécuter le servlet.

L'étape 3 de l'expert doit ressembler à ceci :

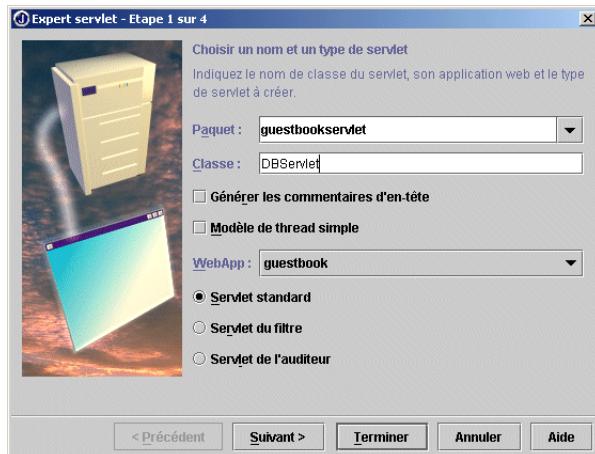


- 8 Appuyez sur Terminer pour créer le servlet. Vous n'avez pas à ajouter de paramètre à l'étape 4 de l'expert.

Pour créer `DBServlet.java`,

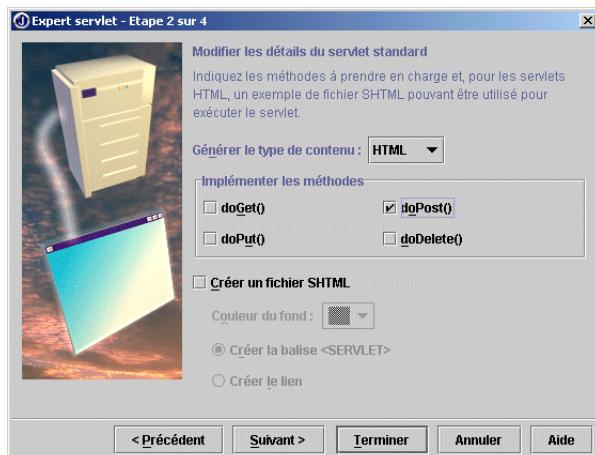
- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Cliquez sur l'onglet Web et choisissez Servlet. Cliquez sur OK.
L'expert servlet est affiché.
- 2 Laissez le nom du paquet défini par `guestbookservlet`. Changez le champ Classe en `DBServlet`.
- 3 Vérifiez que les options Générer les commentaires d'en-tête et Modèle de thread simple ne sont pas sélectionnées. La WebApp `guestbook` doit être sélectionnée dans la liste déroulante WebApp. Conservez sélectionnée l'option Servlet standard.

L'étape 1 de l'expert doit ressembler à ceci :



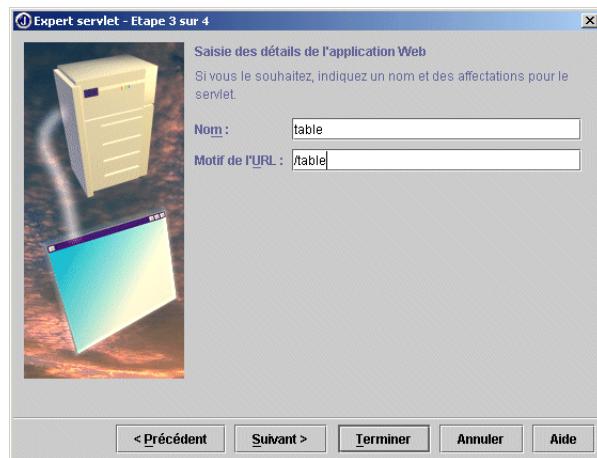
- 4 Cliquez sur Suivant pour passer à l'étape 2 de l'expert.
- 5 Vérifiez que l'option Générer le type de contenu est définie par HTML. Désélectionnez la méthode `doGet()`. Sélectionnez à la place la méthode `doPost()`. Ne sélectionnez pas l'option Crée un fichier SHTML.

Quand elle est terminée, l'étape 2 de l'expert doit ressembler à ceci :



- 6 Cliquez sur Suivant pour passer à l'étape 3 de l'expert.
- 7 Changez la valeur par défaut du champ Nom en `table`. Changez la valeur du champ Motif de l'URL en `/table`. Vérifiez que les entrées sont tout en minuscules. Vous utiliserez le nom `table`, au lieu du nom de la classe `DBServlet`, dans la balise HTML `<form>` de `FormServlet`.

L'étape 3 de l'expert projet doit ressembler à ceci :



- 8 Appuyez sur Terminer pour créer le servlet. Vous n'avez pas à ajouter de paramètre à l'étape 4 de l'expert.

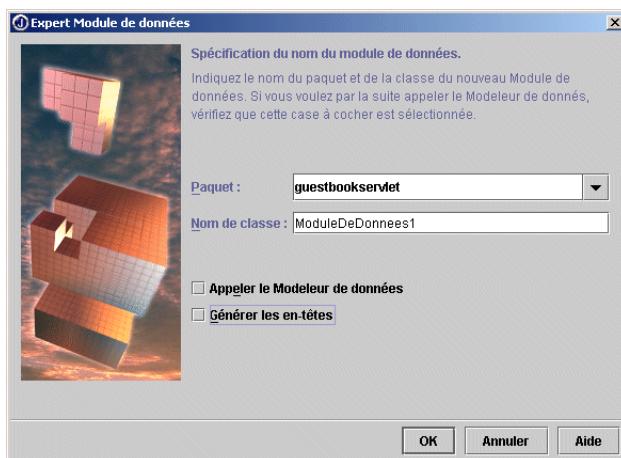
Dans l'étape suivante, vous allez créer le module de données qui contiendra la logique métier.

Etape 4 : Cr éation du module de donn ees

Pour cr éer le module de donn ees qui connecte le JDataStore Guestbook et effectue les tâches de mise à jour,

- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Dans la page Nouveau, choisissez Module de donn ees et cliquez sur OK.
L'expert module de donn ees est affiché.
- 2 Laissez le nom du paquet d efini par guestbookservlet . Conservez le Nom de classe par d efault, ModuleDeDonnees1.
- 3 V rifiez que les options Appeler le modeleur de donn ees et Générer les en-têtes ne sont pas s lectionnées.

Quand il est terminé, l'expert module de données doit ressembler à ceci :



- 4 Cliquez sur OK pour créer le module de données.
- 5 Choisissez Fichier | Tout enregistrer ou cliquez sur  dans la barre d'outils pour enregistrer votre travail.

Dans l'étape suivante, vous ajouterez les composants de base de données au module de données.

Etape 5 : Ajout des composants de base de données au module de données

Dans cette étape, vous utiliserez le concepteur d'interface utilisateur de JBuilder pour ajouter des composants de base de données au module de données. Pour plus d'informations sur le concepteur, voir "Outils de conception visuelle de JBuilder" dans *Construction d'applications avec JBuilder*. Pour plus d'informations sur les composants de base de données, voir "Connexion à une base de données" in the *Guide du développeur d'applications de bases de données*.

Pour ouvrir le concepteur, double-cliquez sur `ModuleDeDonnees1.java` dans le volet projet. Cliquez ensuite sur l'onglet Conception en bas du volet contenu.

Pour ajouter des composants d'accès aux données au module de données

- 1 Choisissez l'onglet DataExpress dans la palette de composants.
- 2 Cliquez sur l'icône Database. Cliquez ensuite dans l'arborescence des composants pour ajouter le composant base de données au module de données.





- 3** Cliquez sur l'icône QueryDataSet. Cliquez dans l'arborescence des composants.

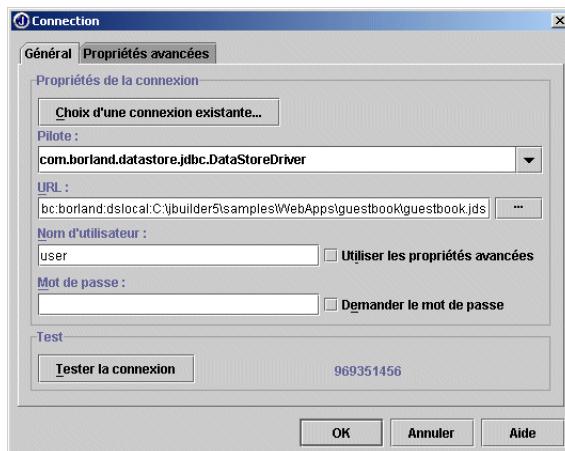
Lorsque vous aurez terminé, l'arborescence des composants ressemblera à ceci :



Pour connecter le JDataStore Guestbook,

- 1** Choisissez `database1` dans l'arborescence des composants.
- 2** Cliquez sur la propriété `connection`, dans l'inspecteur, puis cliquez sur le bouton Points de suspension situé à droite du nom de la propriété. Cela ouvre la boîte de dialogue `Connection`.
- 3** Dans la page Général de la boîte de dialogue `Connection`, vérifiez que le Pilote est défini par `com.borland.datastore.jdbc.DataStoreDriver`.
- 4** Cliquez sur le bouton Points de suspension situé à côté du champ `URL` pour afficher la boîte de dialogue `Création d'une URL pour DataStore`. Cette boîte de dialogue permet de choisir le JDataStore auquel vous connecter.
- 5** Cliquez sur l'option `Base de données DataStore locale`.
- 6** Choisissez le bouton Points de suspension pour naviguer jusqu'au JDataStore Guestbook (`guestbook.jds`) dans le répertoire `samples/WebApps/guestbook` de votre installation de JBuilder. Cliquez sur `Ouvrir` pour sélectionner le JDataStore.
- 7** Cliquez sur `OK` pour fermer la boîte de dialogue `Création d'une URL pour DataStore`.
- 8** Sur la page Général de la boîte de dialogue `Connection`, entrez `user` dans le champ `Nom d'utilisateur`.

La boîte de dialogue Connection ressemblera à ce qui suit :



- 9 Cliquez sur le bouton Tester la connexion pour tester la connexion au JDataStore Guestbook. Si la connexion réussit, le mot Succès s'affichera à droite du bouton. Si la connexion échoue, un message d'erreur tentera d'expliquer pourquoi.
- 10 Cliquez sur OK pour fermer la boîte de dialogue Connection.

JBuilder ajoute la méthode `database1.setConnection()` à la méthode `jbInit()` du module de données.

Pour accéder aux données du Guestbook, vous devez définir une requête. Pour plus d'informations sur les requêtes, voir "Interrogation d'une base de données" dans le *Guide du développeur d'applications de bases de données*. Pour créer la requête dans JBuilder,

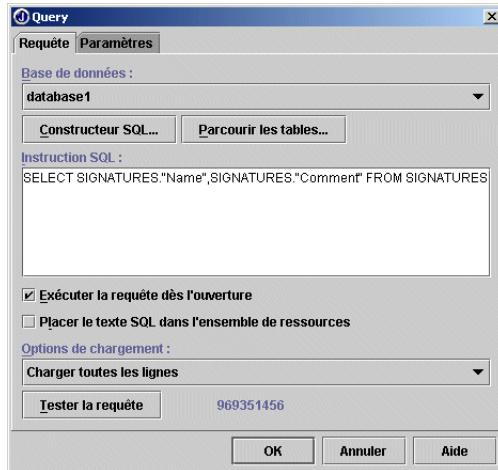
- 1 Cliquez sur le composant `queryDataSet1` dans l'arborescence des composants.
 - 2 Cliquez sur la propriété `query` dans l'inspecteur et cliquez sur le bouton Points de suspension.
- La boîte de dialogue Query s'affiche.
- 3 Dans la page Requête, choisissez `databse1` dans la liste déroulante Base de données.
 - 4 Dans la boîte Instruction SQL, entrez :

```
SELECT SIGNATURES."Name",SIGNATURES."Comment" FROM SIGNATURES
```

La requête charge toutes les valeurs des champs Name et Comment de la table SIGNATURES du JDataStore Guestbook. Lorsque vous saisissez la requête, employez les majuscules et minuscules telles que ci-dessus.

- 5 Assurez-vous que l'option Exécuter la requête dès l'ouverture est sélectionnée.
- 6 Dans la liste déroulante Options de chargement, vérifiez que l'option Charger toutes les lignes est sélectionnée.

La page Requête de la boîte de dialogue Query doit ressembler à ceci :



- 7 Cliquez sur le bouton Tester la requête pour vérifier qu'elle peut être exécutée. Si la requête réussit, le mot Succès s'affichera à droite du bouton. Si la requête échoue, un message d'erreur tentera d'expliquer pourquoi.
- 8 Cliquez sur OK pour fermer la boîte de dialogue Query.
- 9 Cliquez sur l'onglet Source pour revenir dans l'éditeur.

Remarque

Vous risquez de voir apparaître le message suivant dans la page Concepteur du volet message :

Echec de création d'une instance dynamique pour la variable 'myDM'
guestbookservlet.ModuleDeDonnees1

Vous pouvez ignorer ce message pour l'instant. Vous corrigerez le problème ultérieurement. Cliquez avec le bouton droit sur l'onglet Conception en bas de l'AppBrowser et choisissez Retirer l'onglet "Concepteur".

- 10 Choisissez Fichier | Tout enregistrer ou cliquez sur dans la barre d'outils pour enregistrer votre travail.
- JBuilder ajoute la méthode `queryDataSet1.setQuery()` à la méthode `jbInit()`. Dans l'étape suivante, vous allez configurer la connexion des données au DBServlet.

Etape 6 : Création de la connexion de données au DBServlet

Dans cette étape, vous allez créer une connexion des données au DBServlet. La connexion permet au servlet de transmettre des données au module de données.

- 1 Double-cliquez sur `DBServlet.java` dans le volet projet pour ouvrir le fichier dans l'éditeur.
- 2 Trouvez dans la définition des classes la ligne de code qui se lit :

```
private static final String CONTENT_TYPE="text/html"
```

- 3 Positionnez le curseur après cette ligne et ajoutez la ligne de code suivante :

```
ModuleDeDonnees1 dm = guestbookservlet.ModuleDeDonnees1.getDataModule();
```

- 
- 4 Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Maintenant que le servlet est connecté au module de données, nous devons faire faire quelque chose aux servlets et au module de données. A partir de ce point du tutoriel, nous entrerons le code directement dans l'éditeur. La première étape consistera à créer un formulaire de saisie dans `FormServlet`.

Etape 7 : Ajout d'un formulaire de saisie à FormServlet

Dans cette étape, vous ajouterez du code à la méthode `doGet()` de `FormServlet`. Ce code créera un formulaire, en utilisant la balise HTML `<form>`. Ce formulaire va lire deux valeurs – `UserName` et `UserComment` – entrées par l'utilisateur. Ces données seront "postées" vers `DBServlet`, le servlet qui communique avec le module de données.

Une balise `<form>` est une balise standard HTML qui crée un formulaire de saisie pour collecter les données provenant de l'utilisateur ou pour afficher des informations à son intention. La balise contient un attribut `action` et un attribut `method`. Ces attributs indiquent au servlet ce qu'il doit faire lorsqu'est actionné le bouton Soumettre du formulaire. Dans notre tutoriel, l'attribut `action` appelle `DBServlet`. L'attribut `method` poste les paramètres `UserName` et `UserComment` dans `DBServlet`.

Pour ajouter du code à `FormServlet`,

- 1 Double-cliquez sur `FormServlet` dans le volet projet pour ouvrir le fichier dans l'éditeur.
- 2 Recherchez la méthode `doGet`. Elle se trouve au début du fichier.

Astuce

Vous pouvez rechercher la méthode en plaçant le curseur dans le volet structure et en tapant `doGet`.

3 Supprimez les lignes de code suivantes de la méthode doGet() :

```
out.println("<font color=\"green\">");  
out.println("<p>Le servlet a reçu un GET. Ceci est la réponse.</p>");  
out.println("</font>");
```

4 Ajoutez les lignes de code suivantes à la méthode doGet(), entre les balises <body> ouvrante et fermante :

```
out.println("<h1>Signez le guestbook</h1>");  
out.println("<strong>Entrez votre nom et vos commentaires dans les champs de  
saisie ci-dessous.</strong>");  
out.println("<br><br>");  
out.println("<form action=table method=POST>");  
out.println("Nom<br>");  
out.println("<input type=text name=UserName value=\"\" size=20  
maxlength=150>");  
out.println("<br><br>");  
out.println("Commentaire<br>");  
out.print("<input type=text name=UserComment value=\"\" size=50  
maxlength=150>");  
out.println("<br><br><br><br>");  
out.print("<input type=submit value=Soumettre>");  
out.println("</form>");
```

Astuce

Vous pouvez copier et coller ce code directement dans l'éditeur, ou le copier depuis l'exemple qui se trouve dans le dossier samples/WebApps/GuestbookServlet de votre installation JBuilder.



5 Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Dans l'étape suivante, vous ajouterez le code connectant DBServlet au module de données.

Etape 8 : Ajout du code connectant DBServlet au module de données

Dans cette étape, nous ajouterons du code à la méthode doPost() de DBServlet pour :

- Lire les paramètres UserName et UserComment depuis FormServlet.
- Appeler la méthode du module de données qui met à jour le JDataStore Guestbook, en transmettant les valeurs des paramètres UserName et UserComment.
- Appeler la méthode du module de données qui enregistre les modifications dans le JDataStore.

Pour ce faire,

- 1 Double-cliquez sur DBServlet dans le volet projet pour ouvrir le fichier dans l'éditeur.
- 2 Recherchez la méthode doPost().
- 3 Supprimez les lignes de code suivantes de la méthode doPost() :

```
out.println("<p>Le servlet a reçu un GET. Ceci est la réponse.</p>");
```
- 4 Ajoutez les lignes de code suivantes à la méthode doPost() :

```
String userName = request.getParameter("UserName");
String userComment = request.getParameter("UserComment");
dm.insertNewRow(userName, userComment);
dm.saveNewRow();
doGet(request, response);
```

Astuce

Vous pouvez copier et coller ce code directement dans l'éditeur, ou le copier depuis l'exemple qui se trouve dans le dossier samples/WebApps/GuestbookServlet de votre installation JBuilder.

Les deux premières lignes de code obtiennent les valeurs dans les paramètres UserName et UserComment qui sont transmis par FormServlet. Les lignes suivantes appellent deux méthodes dans le module de données :

- insertNewRow() – insère les nouvelles valeurs Name et Comment dans la dernière ligne de la table.
- saveNewRow() – enregistre les modifications dans le JDataStore Guestbook.

La dernière ligne appelle la méthode doGet() du servlet qui affiche la table Guestbook en HTML.



- 5 Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Dans l'étape suivante, vous ajouterez à DBServlet le code qui affiche la table Guestbook, y compris la nouvelle ligne ajoutée, en HTML.

Etape 9 : Ajout du code affichant la table SIGNATURES du Guestbook

Dans cette étape, nous ajouterons une méthode doGet() à DBServlet pour afficher la table SIGNATURES du Guestbook en HTML. A la fois les lignes existantes et la nouvelle ligne sont affichées.

Insérez les lignes de code suivantes après la méthode doPost() du servlet :

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
```

Etape 9 : Ajout du code affichant la table SIGNATURES du Guestbook

```
out.println("<html>");
out.println("<body>");
out.println("<h2>" + dm.queryDataSet1.getTableName() + "</h2>");
Column[] columns = dm.queryDataSet1.getColumns();
out.println ("<table border = 1><tr>");
for (int i=1; i < columns.length; i++) {
    out.print("<th>" + columns[i].getCaption() + "</th>");
}
out.println("</tr>");
dm.queryDataSet1.first();
while (dm.queryDataSet1.inBounds()) {
    out.print("<tr>");
    for (int i = 1; i < columns.length; i++) {
        out.print ("<td>" + dm.queryDataSet1.format(i) + "</td>");
    }
    out.println("</tr>");
    dm.queryDataSet1.next();
}
out.println("</table>");
out.println("</body>");
out.println("</html>");
}
```

Astuce Vous pouvez copier et coller ce code directement dans l'éditeur, ou le copier depuis l'exemple qui se trouve dans le dossier samples/WebApps/GuestbookServlet de votre installation JBuilder.

Pour garantir que ce servlet se compilera, vérifiez les instructions `import` au début du fichier. Ajouter à la liste des imports les paquets suivants :

```
import com.borland.dx.dataset.*;
import com.borland.dx.sql.dataset.*;
import com.borland.datastore.*;
```



Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Que fait la méthode doGet() ?

La méthode `doGet()` que vous venez d'ajouter affiche la table SIGNATURES du Guestbook en HTML. Il parcourt les lignes de la table du JDataStore et les affiche dans le navigateur web.

Les lignes de code suivantes contiennent la déclaration standard de la méthode.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
```

Les deux lignes de code suivantes définissent la sortie comme étant du HTML et démarre la page HTML.

```
out.println("<html>");  
out.println("<body>");
```

La ligne de code suivante imprime le nom de la table du JDataStore, SIGNATURES, en haut de la page HTML. Le code utilise la méthode queryDataSet1.getTableName() pour obtenir le nom de la table.

```
out.println("<h2>" + dm.queryDataSet1.getTableName() + "</h2>");
```

La ligne suivante appelle la méthode queryDataSet1.getColumns() pour obtenir les noms des colonnes et les retourner sous la forme d'un tableau.

```
Column[] columns = dm.queryDataSet1.getColumns();
```

La ligne suivante crée la table avec la balise <table> et crée la première ligne de la table.

```
out.println ("<table border = 1><tr>");
```

Ensuite, le code utilise une boucle for pour parcourir les noms des colonnes du tableau, extraire les libellés des colonnes et afficher chaque libellé dans une ligne de la table. Dans ce tutoriel, nous n'afficherons que les deuxième et troisième colonnes du JDataStore. Nous n'afficherons pas la première colonne, le numéro de ligne à usage interne.

```
for (int i = 1; i < columns.length; i++) {  
    out.print ("<th>" + columns[i].getCaption() + "</th>");  
}
```

La ligne qui suit le bloc for ferme la ligne de la table.

```
out.println("</tr>");
```

La ligne suivante positionne le curseur sur la première ligne du JDataStore.

```
dm.queryDataSet1.first();
```

La boucle while parcourt le JDataStore et affiche les données des deuxième et troisième colonnes de la table. La première fois, elle affiche les données figurant dans la première ligne. Ensuite, la méthode next() positionne le curseur sur la ligne d'après dans le JDataStore. La boucle while continue à afficher des données tant que le curseur reste entre les bornes, c'est-à-dire tant que la méthode inBounds() signale que la navigation se trouve entre le premier et le dernier enregistrement visibles sous le curseur. Lorsque cette condition n'est pas remplie, la table et la page HTML sont fermées.

```
while (dm.queryDataSet1.inBounds()) {  
    out.print("<tr>");  
    for (int i = 1; i < columns.length; i++) {  
        out.print ("<td>" + dm.queryDataSet1.format(i) + "</td>");  
    }  
    out.println("</tr>");  
    dm.queryDataSet1.next();  
}  
out.println("</table>");  
out.println("</body>");  
out.println("</html>");
```

Etape 10 : Ajout de la logique métier au module de données

Vous avez presque fini ! Jusqu'à ce stade le programme ne fait rien car il n'y a toujours pas de code pour écrire les données nouvellement ajoutées au JDataStore Guestbook et pour les enregistrer. Ce code sera ajouté à ModuleDeDonnees1. Ce code va ouvrir l'ensemble de données, insérer la nouvelle ligne (en utilisant les chaînes `userName` et `userComment` transmises par `DBServlet`) et enregistrer la nouvelle ligne dans le JDataStore.

Suivez les étapes ci-après pour ajouter la logique métier au module de données :

- 1** Double-cliquez sur `ModuleDeDonnees1.java` dans le volet projet pour ouvrir le fichier dans l'éditeur.
- 2** Avant de pouvoir insérer ou enregistrer des données, vous devez ouvrir l'ensemble de données. Dans votre module de données, nous allons l'ouvrir juste après le code qui se connecte à la base de données et définit la requête. Pour cela, trouvez la méthode `jbInit()` en utilisant la commande Chercher | Chercher. Ajoutez le code suivant avant l'accolade fermante de la méthode :

```
queryDataSet1.open();
```

- 3** Ajoutez le code de la méthode qui insère une nouvelle ligne. Pour ajouter une ligne, vous devez créer un objet `DataRow` et transmettre les données, depuis les paramètres `userName` et `userComment`, dans ce `DataRow`. Vous ajouterez cette méthode après la méthode `jbInit()`. Simplement, déplacez le curseur d'une ligne vers le bas, après l'accolade fermante de la méthode, et appuyez plusieurs fois sur Entrée. Ajoutez la méthode suivante :

```
public void insert newRow(String userName, String userComment) {
    try {
        DataRow dataRow1 = new DataRow(queryDataSet1, new String[] { "Name",
    "Comment" });
        dataRow1.setString("Name", userName);
        dataRow1.setString("Comment", userComment);
        queryDataSet1.addRow(dataRow1);
    }
    catch (DataSetException ex) {
        ex.printStackTrace();
    }
}
```

La première ligne de la méthode crée un nouvel objet `DataRow` qui contient les nouvelles valeurs de `Name` et de `Comment`. La deuxième et la troisième lignes transmettent les valeurs des paramètres `UserName` et `UserComment` dans les champs `Name` et `Comment`. La dernière ligne ajoute l'objet `DataRow` à l'ensemble de données.

- 4 Après la méthode `insertNewRow()`, ajoutez la méthode suivante qui enregistre la nouvelle ligne dans l'ensemble de données :

```
public void saveNewRow() {  
    try {  
        database1.saveChanges(queryDataSet1);  
    }  
    catch (DataSetException ex) {  
        ex.printStackTrace();  
    }  
}
```



- 5 Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Vous avez ajouté à votre programme toute le code nécessaire. Dans l'étape suivante, vous allez le compiler et l'exécuter.

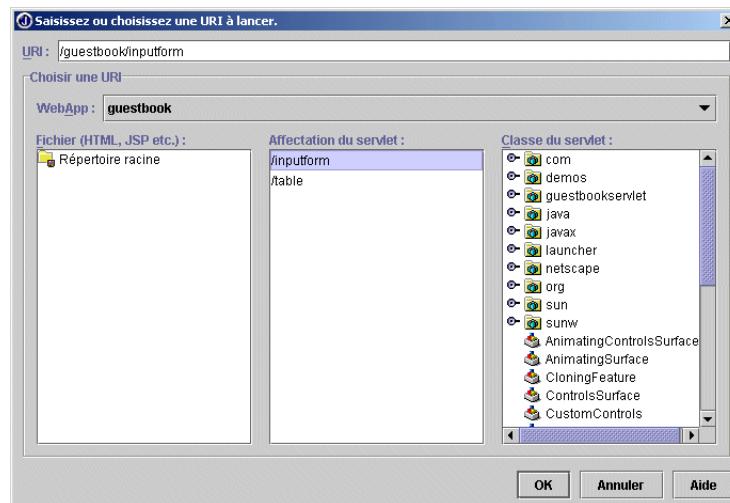
Etape 11 : Compilation et exécution de votre projet

Pour définir les propriétés d'exécution du projet,

- 1 Choisissez Projet | Propriétés du projet. Cliquez sur l'onglet Exécution, puis sur l'onglet JSP/Servlet dans la page Exécution.
- 2 Cliquez sur le bouton Points de suspension situé à côté du bouton URI de lancement pour afficher la boîte de dialogue Saisissez ou choisissez une URI à lancer. Vous choisirez le nom du servlet à démarrer. C'est le nom que vous avez saisi dans le champ Nom à l'étape 3 de l'expert servlet pour `FormServlet`.
- 3 Dans l'arborescence des répertoires Affectation du servlet, au centre de la boîte de dialogue, choisissez `/inputform`. Le champ URI en haut de la boîte de dialogue contiendra : `/guestbook/inputform`. C'est le nom de la WebApp que vous avez créée dans l'expert application web, suivi du nom du servlet.

Etape 11 : Compilation et exécution de votre projet

La boîte de dialogue Saisissez ou choisissez une URI à lancer doit ressembler à ce qui suit :



- 4 Cliquez sur OK pour fermer la boîte de dialogue Saisissez ou choisissez une URI à lancer, puis à nouveau pour fermer la boîte de dialogue Propriétés du projet.



- 5 Cliquez sur l'icône Tout enregistrer, dans la barre d'outils, pour enregistrer votre travail.

Pour compiler et exécuter votre projet,

- 1 Choisissez Projet | Construire le projet "GuestbookServlet.jpx".
 - 2 Choisissez Exécuter | Exécuter le projet.
- Le serveur web Tomcat s'affiche dans le volet message.
- 3 Le formulaire de saisie de FormServlet est affiché dans la Vue Web.

Signez le guestbook

Entrez votre nom et vos commentaires dans les champs de saisie ci-dessous.

Nom

Commentaire

Soumettre

http://localhost:8080/guestbook/inputform

Source | Vue Web | Source Vue Web | Conception | Bean | Doc | Historique

- 4 Entrez MonNom dans le champ Name et MonCommentaire dans le champ Comment.
- 5 Cliquez sur le bouton Soumettre.

La table SIGNATURES du Guestbook est affichée en HTML. MonNom et MonCommentaire sont affichés dans la dernière ligne de la table.

Name	Comment
Aries	I am
MonNom	MonCommentaire

Notez que l'URI est /guestbook/inputform/ et qu'il correspond à ce que vous avez sélectionné dans la boîte de dialogue URI de lancement. Si vous avez exécuté le servlet en cliquant avec le bouton droit sur FormServlet.java et en choisissant Exécution Web, vous verrez une autre URL : guestbook/servlet/guestbookservlet.FormServlet. Pour plus d'informations sur les URL, les URI et les servlets, voir "Comment les URL exécutent les servlets", page 15-3.

- 6 Vous pouvez cliquer sur la flèche vers l'arrière à gauche du champ de l'URL pour revenir au formulaire de saisie et entrer un autre nom et un autre commentaire.
- 7 Pour arrêter le serveur web, cliquez sur le bouton Réinitialiser le programme directement au-dessus de l'onglet du serveur web. Vous devez arrêter le serveur web avant de compiler et d'exécuter à nouveau le servlet après avoir effectué vos modifications.

Remarque Vous pouvez ouvrir le JDataStore Guestbook dans l'explorateur JDataStore pour vérifier que les nouvelles données ont été enregistrées dans la table.

Félicitations ! Vous avez terminé le tutoriel. Vous savez désormais créer un formulaire de saisie HTML pour l'utiliser dans un servlet, passer un paramètre d'un servlet à un autre servlet, connecter un servlet à un module de données, passer des paramètres d'un servlet à un module de données et utiliser un module de données pour mettre à jour un JDataStore.

Développement des pages JavaServer

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

La technologie JSP (JavaServer Pages) permet aux développeurs et concepteurs web de développer rapidement et de maintenir facilement des pages web, dynamiques et riches en information, qui tirent profit des systèmes d'entreprise existants. Faisant partie de la famille Java, la technologie JSP permet de développer rapidement des applications web indépendantes des plates-formes.

Théoriquement, la technologie JSP sépare l'interface utilisateur de la génération de son contenu, ce qui permet aux concepteurs de modifier la disposition générale des pages sans altérer le contenu dynamique sous-jacent. En pratique, il faut un peu d'organisation et le respect de quelques règles de programmation pour que le HTML soit bien séparé du code Java dans la JSP, puisque les deux se trouvent dans le même fichier. Les concepteurs Web gérant la partie HTML doivent avoir une compréhension minimale des balises qui représentent du code Java incorporé, pour éviter les problèmes lors de la conception de l'interface utilisateur.

La technologie JSP utilise des balises XML et des scriptlets écrits en langage de programmation Java pour encapsuler la logique qui génère le contenu de la page. De plus, la logique de l'application peut résider dans des ressources basées sur le serveur (comme l'architecture de composants JavaBeans), auxquelles la page accède par le biais des balises et des scriptlets. Toutes les balises de formatage (HTML ou XML) sont transmises directement à la page réponse. En séparant la logique de la page de sa conception et de son affichage, et en supportant une conception basée sur les composants réutilisables, la technologie JSP rend plus rapide et plus facile que jamais la construction des applications web.

La technologie JSP est une extension de l'API Servlet de Java. La technologie JSP apporte essentiellement une façon simplifiée de développer des servlets. Les servlets sont des modules côté serveur indépendants des plates-formes et entièrement Java qui s'adaptent de façon transparente à une architecture de serveur web et servent à augmenter les possibilités du serveur web avec des temps système, une maintenance et un support minimaux. Au contraire des autres langages de script, les servlets ne demandent aucune attention ni aucune modification par rapport aux plates-formes. Ensemble, la technologie JSP et les servlets sont une alternative intéressante aux autres types de programmation ou d'écriture de scripts dynamiques pour le web, en offrant l'indépendance par rapport aux plates-formes, l'amélioration des performances, la séparation entre la logique et l'affichage, la simplicité d'administration, l'extensibilité dans l'entreprise et, le plus important, la facilité d'emploi.

Les JSP ressemblent beaucoup aux ASP (Active Server Pages) de la plate-forme Microsoft. La principale différence entre JSP et ASP est la suivante : les objets manipulés par les JSP sont des JavaBeans et donc indépendants des plates-formes. Les objets manipulés par les ASP sont des objets COM, ce qui lie complètement les ASP à la plate-forme Microsoft.

Tout ce qu'il faut à une JSP, c'est une page basée sur la technologie JSP. C'est une page qui contient des balises spécifiques à JSP, des déclarations et, éventuellement, des scriptlets, combinés aux autres contenus statiques (HTML ou XML). Une page basée sur la technologie JSP porte l'extension .jsp ; cette extension signale au serveur web que son moteur JSP devra traiter les éléments qu'elle contient. Une JSP peut aussi utiliser de manière optionnelle un ou plusieurs JavaBeans dans des fichiers .java séparés.

Lorsqu'une JSP est compilée par le moteur de JSP sur le serveur web, elle est compilée dans un servlet. En tant que développeur, vous ne verrez généralement pas le code du servlet généré. Cela signifie que lorsque vous compilez des JSP dans l'EDI de JBuilder, vous risquez de voir des messages d'erreur faisant directement référence au code du servlet généré, et uniquement indirectement au code de la JSP. Si vous obtenez des messages d'erreur lorsque vous compilez vos JSP, n'oubliez pas qu'ils peuvent se référer aux lignes de code du servlet généré. Il vous sera plus facile de comprendre ce qui se passe dans votre JSP si vous avez une idée de la façon dont elles sont traduites en servlets. Pour cela, vous devez comprendre l'API JSP.

Le Chapitre 10, "Tutorial : Création d'une JSP en utilisant l'expert JSP", montre comment créer une JSP en utilisant l'expert JSP comme point de départ.

Pour vous connecter à des pages web contenant davantage d'informations sur la technologie JavaServer Pages, reportez-vous à la rubrique "Ressources JSP supplémentaires", page 9-5.

L'API JSP

Une JSP inclut habituellement un nombre de balises spécialisées qui contiennent du code Java, ou des fragments de code Java. Voici quelques balises JSP, parmi les plus importantes :

Syntaxe de la balise	Description
<code><% fragment de code %></code>	Balise scriptlet. Contient un fragment de code, constitué d'une ou de plusieurs lignes de code qui seraient normalement apparues dans le corps d'une méthode d'une application Java. Nul besoin de déclarer les méthodes, car ces fragments de code deviennent parties intégrantes de la méthode service() du servlet lorsque la JSP est compilée.
<code><%! déclaration %></code>	Déclaration d'une méthode ou d'une variable. Lorsque vous déclarez une méthode dans cette balise, la totalité de la méthode doit être contenue dans la balise. Est compilée en une déclaration de méthode ou de variable dans le servlet.
<code><%-- commentaire --%></code>	Commentaire. C'est un commentaire de style JSP qui n'est pas transmis au navigateur client. (Vous pouvez aussi utiliser les commentaires HTML, mais ils seront transmis au navigateur client.)
<code><%= expression %></code>	Expression. Peut contenir n'importe quelle expression Java correcte. Le résultat est affiché sur la page à cet emplacement.
<code><%@ page [attributs] %></code>	Directive page. Spécifie les attributs de la page JSP. Les directives de cette sorte et la directive taglib doivent constituer les premières lignes de la JSP. Un des attributs les plus fréquemment spécifiés dans la directive page est l'instruction import. Exemple : <code><%@ page import="com.borland.internetbeans.*" %></code>
<code><%@ taglib uri="chemin d'accès à la bibliothèque de balises" prefix="préfixe des balises" %></code>	Directive taglib. Rend une bibliothèque de balises utilisable par la JSP, en spécifiant l'emplacement de la bibliothèque et le préfixe utilisé par les balises qui en font partie. Les directives de cette sorte et la directive page doivent constituer les premières lignes de la JSP.

La spécification JSP comprend aussi des balises standard pour l'utilisation et la manipulation des beans. La balise `useBean` crée l'instance de la classe d'un JavaBean donné. Si une instance existe déjà, elle est utilisée. Sinon, l'instance est créée. Les balises `setProperty` et `getProperty` vous permettent de manipuler les propriétés d'un bean donné. Ces balises et les autres sont décrites avec plus de détails dans la spécification et dans le guide utilisateur JSP, que vous trouverez à l'adresse <http://java.sun.com/products/jsp/techinfo.html>.

Il est important de comprendre que la majorité du code Java contenu dans les balises JSP devient partie intégrante de la méthode `service()` du servlet lorsque la JSP est compilée en servlet. Cela ne s'applique pas au code contenu dans les balises de déclaration qui est transformé en déclarations autonomes complètes de méthodes ou de variables. La méthode `service()` est appelée chaque fois que le client fait un `GET` ou un `POST`.

JSP dans JBuilder

JBuilder fournit un système de développement complet des JSP, y compris un expert JSP pour créer de nouvelles pages JSP, l'audit de code pour les balises spécifiques aux JSP, le débogage à l'intérieur d'un fichier JSP, les tests et l'exécution des JSP dans le moteur de servlets Tomcat depuis l'environnement de développement de JBuilder.

L'expert JSP

JBuilder propose un expert JSP. C'est un point de départ logique pour le développement d'une JSP. Vous verrez l'expert en cliquant sur l'onglet Web de la galerie d'objets.

L'expert JSP génère le squelette d'une JSP. Il crée les fichiers de base dont vous avez besoin dans votre JSP et dont vous remplirez les détails par la suite.

Dans l'expert JSP, vous pouvez spécifier à quelle WebApp votre JSP appartient, s'il faut définir votre JSP pour qu'elle utilise la bibliothèque de balises InternetBeans et si votre JSP utilise les JavaBeans. Pour plus d'informations sur les fonctionnalités de l'expert JSP, voir la rubrique correspondante dans l'aide en ligne.

Le Chapitre 10, "Tutoriel : Création d'une JSP en utilisant l'expert JSP", montre comment créer une JSP en utilisant l'expert JSP comme point de départ.

Développement d'une JSP

L'expert JSP est un point de départ facultatif pour le développement d'une JSP. L'éditeur de JBuilder fournit la mise en évidence syntaxique pour les JSP. JBuilder fournit également l'audit de code et l'audit d'erreur appliqués au code Java incorporé à une page JSP.

Dans JBuilder, le volet structure de l'EDI montre la structure des balises de la JSP, ainsi que toutes les erreurs HTML ou Java figurant dans votre code. Ces erreurs sont très utiles, par exemple, elles vous permettront souvent de compléter une balise laissée incomplète.

Compilation d'une JSP

Voir "Compilation de votre servlet ou de votre JSP", page 15-2, pour des informations sur la compilation des JSP.

Exécution d'une JSP

Voir "Exécution de votre servlet ou de votre JSP", page 15-5, pour des informations sur l'exécution des JSP.

Débogage d'une JSP

Voir "Débogage de votre servlet ou de votre JSP", page 15-13, pour des informations sur le débogage des JSP.

Déploiement d'une JSP

Voir "Déploiement de votre application web", page 16-1, pour des informations sur le déploiement des JSP.

Ressources JSP supplémentaires

Pour trouver de l'aide pendant le développement des JSP dans JBuilder, rejoignez le groupe de discussion Borland borland.public.jbuilder.servlets-jsp. Tous les groupes de discussion JBuilder sont accessibles depuis le site web de Borland à l'adresse <http://www.borland.fr/Newsgroups/#jbuilder>.

Pour plus d'informations sur le développement des pages JavaServer, consultez les sites web suivants. Ces adresses et ces liens étaient valables au moment de l'impression de ce manuel. Borland ne contrôle pas ces sites et n'est pas responsable de leur contenu ni de leur pérennité.

- Téléchargez une carte de syntaxe JSP, au format HTML, à l'adresse <http://www.java.sun.com/products/jsp/tags.html>.
- Téléchargez une carte de syntaxe JSP, au format PDF lisible dans Adobe Acrobat Reader, à l'adresse <http://www.java.sun.com/products/jsp/technical.html>. Cette page contient également d'autres ressources techniques consacrées à la technologie JSP.
- Lisez les FAQ sur les pages JavaServer sur le site java.sun.com. Allez à l'adresse <http://www.java.sun.com/products/jsp/faq.html>.
- La spécification JSP peut être consultée sur le site web de JavaSoft, à l'adresse <http://www.java.sun.com/products/jsp/download.html>.

Ressources JSP supplémentaires

- GNUJSP est une implémentation gratuite des pages JavaServer de Sun. Pour en savoir plus sur le compilateur GNUJSP, visitez <http://klomp.org/gnujsp>.
- “Web Development with JavaServer Pages” (<http://www.burridge.net/jsp/jspinfo.html>) est un site Web riche de nombreux liens vers les sujets JSP.
- Un serveur de listes JSP est maintenu par JSP-INTEREST@JAVA.SUN.COM. Pour vous inscrire à la liste d'e-mail, envoyez le message suivant à listserv@java.sun.com :

subscribe jsp-interest Votre nom complet

ou consultez le site web de JavaSoft (<http://www.javasoft.com/>) pour avoir plus d'informations.

10

Tutoriel : Cr éation d'une JSP en utilisant l'expert JSP

Le développement
pour le web
est une fonctionnalité
de JBuilder Professionnel
et de JBuilder Entreprise.

Ce tutoriel vous guide pas à pas dans le développement d'une page JSP en utilisant l'expert JSP de JBuilder. Cette page accepte la saisie d'un texte, affiche ce texte en sortie quand on clique sur le bouton Soumettre et utilise un JavaBean pour compter le nombre de visiteurs de la page web.

L'expert JSP est un excellent point de départ pour développer des JSP. Il ne générera pas une application complète, mais il prendra en charge tous les détails fastidieux nécessaires à la construction de votre application et à son exécution. Vous invoquerez l'expert en sélectionnant Nouveau dans le menu Fichier, en cliquant sur l'onglet Web et en sélectionnant Page JavaServer. Pour plus d'informations sur les options de l'expert JSP, voir la rubrique correspondante dans l'aide en ligne.

Dans le cadre du développement, nous utilisons Tomcat. Tomcat est l'implémentation de référence des spécifications Java Servlet et JavaServer Pages. Cette implémentation peut être utilisée dans le serveur web Apache ainsi que dans d'autres serveurs web et outils de développement. Pour plus d'informations sur Tomcat, consultez <http://jakarta.apache.org>.

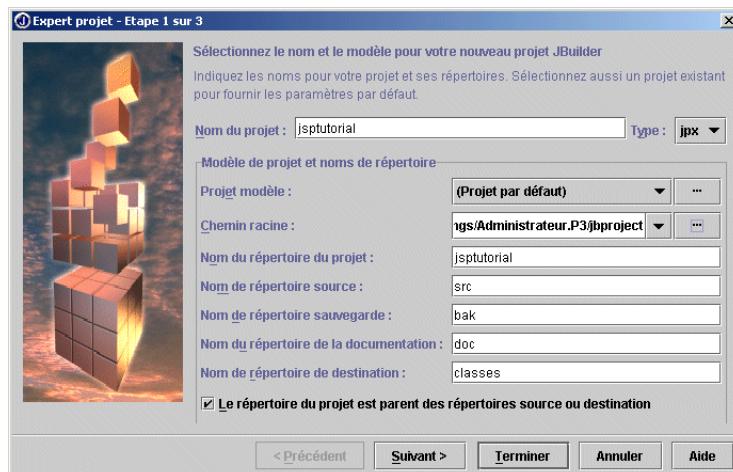
Ce tutoriel suppose que vous vous êtes familiarisé avec Java et avec l'EDI de JBuilder. Pour plus d'informations sur Java, voir *Introduction à Java*. Pour plus d'informations sur l'EDI de JBuilder, voir "L'environnement de JBuilder" dans *Construction d'applications avec JBuilder*.

Si vous avez des suggestions à faire pour améliorer ce tutoriel, envoyez un e-mail à jgppubs@borland.com.

Etape 1 : Création d'un nouveau projet

- 1 Sélectionnez Fichier | Nouveau projet pour afficher l'expert projet.
- 2 Dans le champ Nom du projet, entrez le nom du projet, par exemple jsptutorial.
- 3 Cliquez sur Terminer.
- 4 Un nouveau projet est créé, contenant un fichier HTML qui décrit le projet.

Figure 10.1 Expert projet

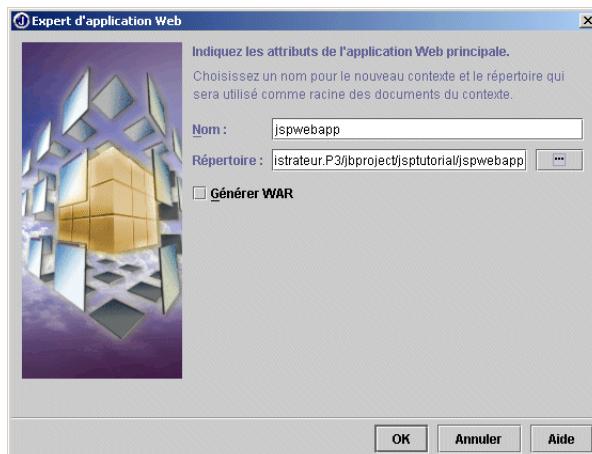


Etape 2 : Création d'une nouvelle WebApp

Cette étape est facultative mais conseillée. Pour plus d'informations sur les WebApp et les fichiers WAR, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

- 1 Sélectionnez Fichier | Nouveau.
- 2 Cliquez sur l'onglet Web. Sélectionnez Application Web.
- 3 Cliquez sur OK. L'expert WebApp apparaît.
- 4 Saisissez un nom pour la WebApp, par exemple jspwebapp.
- 5 Choisissez le bouton Points de suspension situé à droite du champ Répertoire.
- 6 Saisissez le nom du répertoire racine de la WebApp, par exemple jspwebapp.
- 7 Cliquez sur OK.

- 8 Cliquez sur Oui pour créer le répertoire.
- 9 Conservez non cochée l'option Générer WAR, car vous ne voudrez certainement pas déployer réellement cette application.
- 10 L'expert doit se présenter ainsi :



- 11 Cliquez sur OK pour fermer l'expert.

Un nœud WebApp, `jspwebapp`, est affiché dans le volet projet. Développez ce nœud pour voir les nœuds Répertoire racine et Descripteurs de déploiement.

Figure 10.2 Nœud WebApp dans le volet projet

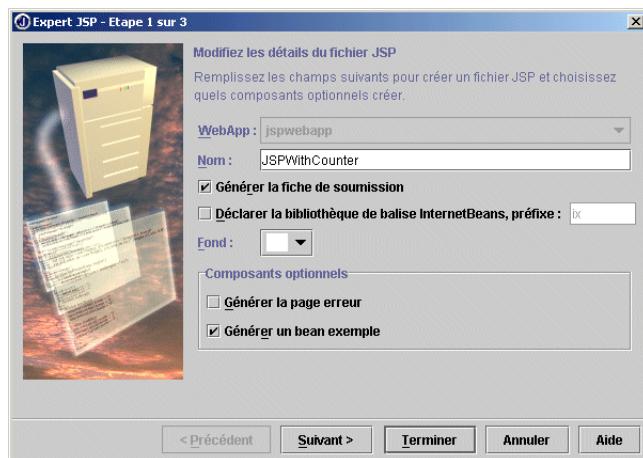


Etape 3 : Utilisation de l'expert JSP

- 1 Sélectionnez Fichier | Nouveau.
- 2 Cliquez sur l'onglet Web. Sélectionnez Page JavaServer.
- 3 Cliquez sur OK. L'expert JSP apparaît.
- 4 Sélectionnez une WebApp pour la JSP dans la liste déroulante WebApp. Vous souhaiterez sélectionner `jspwebapp` si vous avez créé cette WebApp à l'étape 2.
- 5 Entrez un nom pour la JSP : `JSPWithCounter`

6 Cliquez sur Terminer pour accepter toutes les valeurs par défaut.

Figure 10.3 Expert JSP



Un fichier `JSPWithCounter.jsp` est ajouté au répertoire racine de votre WebApp. Développez le nœud Répertoire racine dans le volet projet pour voir ce fichier. Un bean exemple `JSPWithCounterBean.java` est également ajouté à votre projet. Ce bean est appelé par la JSP.

Etape 4 : Ajout de fonctionnalités au JavaBean

A ce stade, ont été créés un fichier JSP et un JavaBean pouvant être utilisé par le fichier JSP.

L'étape suivante de ce tutoriel consiste à créer une méthode pour comptabiliser le nombre de consultations d'une page web.

Double-cliquez sur `JSPWithCounterBean.java` dans le volet projet. Modifiez le code source comme suit, en ajoutant le code qui apparaît en **gras** au code existant :

```
package jsptutorial;

public class JSPWithCounterBean {
    /**initialise la variable ici*/
    private int myCount=0;

    private String sample = "Valeur de départ";
    /**Accès à la propriété sample*/
    public String getSample() {
        return sample;
    }
}
```

```

    /**Accès à la propriété sample*/
    public void setSample(String newValue) {
        if (newValue!=null) {
            sample = newValue;
        }
    }
    /**Nouvelle méthode comptant le nombre de visiteurs*/
    public int count() {
        return ++myCount;
    }
}

```

Etape 5 : Modification du code JSP

Double-cliquez sur `JSPWithCounter.jsp` dans le volet projet pour ouvrir le fichier dans l'éditeur. N'oubliez pas qu'il se trouve dans le nœud Répertoire racine de la WebApp. Sélectionnez l'onglet Source. Vous pouvez utiliser l'audit de code et la mise en évidence du source JSP pour faciliter la programmation. Modifiez le fichier généré comme suit, en ajoutant le code qui apparaît en **gras**. L'audit de code de JBuilder sera activé pour vous aider dans la syntaxe.

```

<html>
<head>
<jsp:useBean id="JSPWithCounterBeanId" scope="session"
    class="jsptutorial.JSPWithCounterBean" />
<jsp:setProperty name="JSPWithCounterBeanId" property="*" />
<title>
JspWithCounter
</title>
</head>
<body>
<h1>
JSP JBuilder généré
</h1>
<form method="post">
<br>Entrez une nouvelle valeur : <input name="sample"><br>
<br><br>
<input type="submit" name="Submit" value="Soumettre">
<input type="reset" value="Réinitialiser">
<br>
La valeur de la propriété Bean est : <jsp:getProperty
name="JSPWithCounterBeanId"
    property="sample" />
<p>Cette page a été visitée :<%= JSPWithCounterBeanId.count() %> fois.</p>
</form>
</body>
</html>

```

La ligne de code que vous venez d'ajouter utilise une balise JSP d'expression pour appeler la méthode `count()` de la classe `JSPWithCounterBean` et insérer la valeur renvoyée dans le HTML généré. Pour plus d'informations sur les balises JSP, voir "L'API JSP", page 9-3.

Remarquez les balises `<jsp:useBean/>`, `<jsp:setProperty/>` et `<jsp:getProperty/>` dans le code précédent. Elles ont été ajoutées par l'expert JSP. La balise `useBean` crée une instance de la classe `JSPWithCounterBean`. Si une instance existe déjà, elle est utilisée. Sinon, l'instance est créée. Les balises `setProperty` et `getProperty` vous permettent de manipuler les propriétés du bean.

Le reste du code généré par l'expert JSP est simplement du HTML standard.

Choisissez Fichier | Tout enregistrer pour enregistrer votre travail.

Etape 6 : Exécution de la JSP

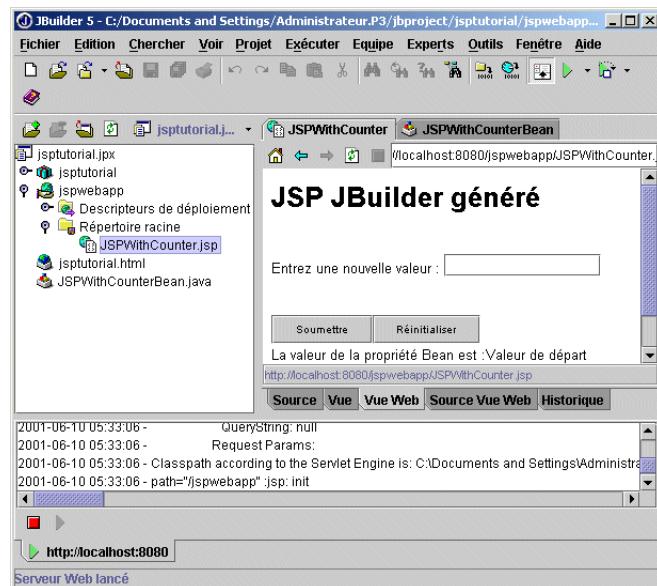
Afin d'exécuter la JSP, les propriétés d'exécution du projet doivent être définies correctement. Dans ce tutoriel, les propriétés d'exécution ont déjà été définies par l'expert JSP, vous pouvez donc poursuivre et exécuter votre JSP. Cliquez avec le bouton droit sur le fichier JSP et sélectionnez Exécution Web dans le menu.

Le projet est compilé et il s'exécute. Les erreurs de compilation s'affichent dans le volet message. S'il y a des erreurs, consultez la rubrique "Débogage de votre servlet ou de votre JSP", page 15-13.

S'il n'y a pas d'erreur, le serveur web est démarré et deux nouveaux onglets, la Vue Web et la Source Vue Web, apparaissent dans le volet contenu. Le serveur web par défaut de JBuilder est Tomcat, un moteur de servlets qui supporte les servlets et les fichiers JSP. La Vue Web est un navigateur web qui affiche la sortie de votre JSP en cours d'exécution. L'onglet Source Vue Web affiche le code HTML réel qui a été généré de

manière dynamique par la JSP. Si tout va bien, la JSP en exécution ressemble à ceci :

Figure 10.4 JSP dans la Vue Web



La Vue Web du volet contenu affiche la page JSP. Pour un test en local, l'URL pointe sur localhost:8080, là où s'exécute Tomcat. Pour tester la JSP, entrez une valeur dans le champ texte, puis cliquez sur le bouton Soumettre. La valeur saisie est affichée au dessous des boutons et le compteur est incrémenté.

Les données de sortie et d'historique de Tomcat apparaîtront dans un nouvel onglet du volet message. Les sorties des servlets ou des beans, ainsi que les commandes HTTP et les valeurs des paramètres, sont renvoyées dans le volet message. Vous pouvez définir les propriétés d'exécution du serveur Web en sélectionnant Projet | Propriétés du projet, puis JSP/Servlet dans la page Exécution. Le numéro de port désigne le port sur lequel s'exécutera le serveur Web. La valeur par défaut est 8080. Si le port 8080 est utilisé, JBuilder recherchera automatiquement un port libre.

Pour plus d'informations sur la définition des paramètres d'exécution de votre servlet ou de votre JSP, voir "Définition des paramètres d'exécution de votre servlet ou de votre JSP", page 15-9.

Utilisation de la Vue Web

La Vue Web du volet contenu affiche le fichier JSP après son traitement par le moteur de JSP. En ce cas, le moteur de JSP est Tomcat. La Vue Web se comportera autrement que l'onglet Vue. Il peut y avoir un délai entre le moment où le fichier JSP est édité et celui où les modifications apparaissent dans la Vue Web. Pour voir les changements les plus récents apportés à un fichier JSP, sélectionnez le bouton Rafraîchissement, dans la barre d'outils de la Vue Web, comme vous le feriez dans n'importe quel navigateur web.



Si vous êtes en train de déboguer votre JSP, vous pouvez appuyer sur la touche F9 pour ramener l'affichage à la Vue Web.

Débogage de la JSP

Les JSP sont compilées en servlets. Dans JBuilder, vous pouvez déboguer les snippets de code Java dans le fichier JSP original, contrairement au servlet Java correspondant généré. Pour plus d'informations sur le débogage de votre JSP, voir "Débogage de votre servlet ou de votre JSP", page 15-13

Déploiement de la JSP

Reportez-vous à la documentation de votre serveur web de production pour savoir comment y déployer des JSP. Pour plus d'informations sur le déploiement des JSP, voir Chapitre 16, "Déploiement de votre application web".

D'après les FAQ JSP (à l'adresse <http://www.java.sun.com/products/jsp/faq.html>), il existe une grande variété d'implémentations de la technologie JSP pour les différents serveurs web. Vous trouverez les annonces de support officielles les plus récentes à l'adresse www.java.sun.com.

11

Utilisation d'InternetBeans Express

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

La technologie InternetBeans Express s'intègre aux servlets et aux JSP pour améliorer vos applications et simplifier les tâches de développement des servlets et des JSP. InternetBeans Express est un ensemble de composants associé à une bibliothèque de balises JSP, permettant de générer la couche présentation d'une application web, et d'y répondre. InternetBeans Express prend des pages modèles statiques, leur insère un contenu dynamique issu du modèle de données réel, et les présente au client ; ensuite, il écrit dans le modèle de données les modifications postées par le client. Cela facilite la création des servlets et des JSP orientés données. Par exemple, vous pouvez utiliser les composants InternetBeans Express pour créer un servlet proposant à un nouvel utilisateur un formulaire d'inscription à un site web ou une JSP affichant les résultats d'une recherche dans une table.

InternetBeans Express supporte les ensembles de données et les modules de données DataExpress. InternetBeans Express peut aussi être utilisé avec des modèles de données et des EJB génériques. Les classes et les interfaces sont de trois types :

- Les InternetBeans sont des composants qui agissent directement sur la génération dynamique des markup et la gestion de la sémantique des demandes/réponses HTTP.
- Les gestionnaires de balises JSP, qui invoquent en interne les InternetBeans, et leur infrastructure de support. Il sont utilisés par les extensions de balises JSP dans la bibliothèque de balises InternetBeans.
- Le support des modèles de données.

Il existe aussi une bibliothèque de balises JSP contenant les extensions de balises JSP utilisées pour la prise en charge des InternetBeans dans une JSP.

Présentation des classes InternetBeans Express

Type de composant	Description
	IxPageProducer Lit et analyse des fichiers HTML statiques, afin de pouvoir ultérieurement les régénérer en insérant du contenu dynamique fourni par d'autres composants.
	IxControl La plupart des servlets utiliseront un IxPageProducer, ce qui permet au servlet de générer entièrement la réponse à partir d'une page web conçue au préalable, en insérant dans cette page des données dynamiques, comme des morceaux de texte ou les valeurs des contrôles d'un formulaire.
	IxTable Un contrôle générique qui détermine à l'exécution quel est le type de contrôle HTML qu'il remplace et qui émule ce contrôle. Ce composant peut être utilisé à la place de n'importe quel autre InternetBean spécifique à un contrôle particulier. Dans un servlet, ce composant doit être utilisé avec un IxPageProducer. Vous n'avez pas besoin d'un IxPageProducer pour utiliser ce composant dans une JSP.
	IxImage Génère des tableaux HTML à partir de modèles d'ensembles de données ou de tables.
	IxLink Représente une image qui est simplement affichée ou utilisée comme lien. Si vous voulez utiliser une image pour soumettre une fiche, utilisez un IxImageButton.
	IxSpan Représente un lien. Si la réécriture de l'URL est nécessaire, IxLink le gère pour vous en appelant de manière interne la méthode HttpServletResponse.encodeURL().
	IxSpan Remplace le contenu en lecture seule par un attribut ID, généralement un , mais cela peut être un autre type d'élément. Pour les contrôles d'une fiche, utilisez plutôt IxControl.
	IxCheckBox Représente une case à cocher. XHTML : <input type="checkbox" />
	IxComboBox Représente une boîte à options. XHTML : <select size="1" />
	IxHidden Représente une valeur cachée. XHTML : <input type="hidden" />

Type de composant	Description
 IxImageButton	Représente une image qui soumet la fiche lorsque vous cliquez dessus. A ne pas confondre avec une image qui est simplement affichée ou utilisée comme lien ; pour cela, utilisez IxImage. XHTML : <input type="image" /> Si l'image correspondant à ce composant est l'image qui a soumis la fiche, l'événement submitPerformed() de IxImageButton sera déclenché.
 IxListBox	Représente une boîte liste. XHTML : <select size="3" />
 IxPassword	Représente un champ mot de passe. XHTML : <input type="password" />
 IxPushButton	Représente un bouton côté client. XHTML : <input type="button" />
 IxRadioButton	Représente une bouton radio. XHTML : <input type="radio" />
 IxSubmitButton	Représente un bouton soumission sur une fiche. XHTML : <input type="submit" /> Si le bouton correspondant à ce composant est le bouton qui a soumis la fiche, l'événement submitPerformed() de IxSubmitButton sera déclenché.
 IxTextArea	Représente une zone de texte. XHTML : <textarea>
 IxTextField	Représente un champ de saisie. XHTML : <input type="text" />

Utilisation d'InternetBeans Express avec les servlets

Les composants InternetBeans Express simplifient l'affichage des données réelles dans la page web comme l'enregistrement des données issues de cette page web dans une base de données ou tout autre modèle de données.

Affichage de pages web dynamiques avec des servlets utilisant InternetBeans Express

La plupart des servlets utiliseront un composant IxPageProducer. Cela permet au servlet de générer entièrement la réponse à partir d'une page web conçue au préalable, en insérant dans cette page des données

dynamiques, comme des morceaux de texte ou les valeurs des contrôles d'un formulaire. Cela offre plusieurs avantages :

- Vous savez à quoi doit ressembler la réponse. La page peut contenir des données fictives, qui seront remplacées.
- Vous pouvez modifier ces aspects en changeant la page, sans toucher au code.

Par exemple, lorsque vous utilisez InternetBeans Express avec un servlet, vous pouvez ouvrir celui-ci dans le concepteur. Un `Database` et un `QueryDataSet`, issus de l'onglet DataExpress de la palette, peuvent fournir les données au servlet. Vous pouvez ajouter un `IxPageProducer` à partir de l'onglet InternetBeans Express de la palette. Définissez la propriété `htmlFile` du `IxPageProducer` par le nom du fichier de la page web conçue au préalable. Lorsque le servlet est exécuté, le `HtmlParser` interne sera désormais invoqué par le `IxPageProducer` pour localiser tous les contrôles HTML remplaçables.

La façon la plus simple de remplacer les contrôles HTML par des contrôles contenant des données générées de manière dynamique est d'utiliser des `IxControl`. Vous devez ajouter un `IxControl` pour chaque contrôle HTML de la page modèle qui contiendra des données. Définissez la propriété `pageProducer` de chaque `IxControl` par `IxPageProducer`. Définissez la propriété `controlName` du `IxControl` pour qu'elle corresponde à l'attribut `name` du contrôle HTML approprié. La définition des propriétés `dataSet` et `columnName` du `IxControl` termine la liaison aux données.

La méthode `IxPageProducer.servletGet()` est la méthode que, normalement, vous appellerez pour générer la page pour l'affichage. Cette méthode doit être appelée à l'intérieur de la méthode `doGet()` du servlet. Souvent, le corps de la méthode `doGet()` d'un servlet sera être aussi simple que :

```
ixPageProducer1.servletGet(this, request, response);
```

Ce seul appel définit le type de contenu de la réponse et renvoie la page dynamique dans un scripteur de flux de sortie en fonction de la réponse. Notez que, par défaut, le type de contenu de la réponse est HTML.

En interne, la méthode `IxPageProducer.render()` génère la version dynamique de la page : elle remplace les contrôles à qui est affecté un `IxControl` en interrogeant le composant à afficher, ce qui génère un HTML équivalent dont les données proviennent de la ligne en cours dans l'ensemble de données. Vous pourriez appeler vous-même la méthode `render()`, mais il est plus simple d'appeler la méthode `servletGet()`.

Parmi les situations où vous n'utiliserez pas de `IxPageProducer` dans un servlet, citons :

- Lorsque vous n'avez pas besoin de la gestion des sessions de bases de données ni des fonctionnalités de "posting" du `IxPageProducer` et que

vous avez uniquement besoin du moteur de modèles de pages, vous pouvez utiliser à la place le PageProducer.

- Lorsque vous utilisez des composants individuels spécifiques pour afficher le HTML. Par exemple, vous pouvez créer une IxComboBox contenant une liste de pays, et l'utiliser dans un servlet avec du code HTML manuel.

Lorsque vous utilisez des InternetBeans dans un servlet, vous devez habituellement employer un IxPageProducer. Lorsque vous utilisez des IxControl, vous devez utiliser un IxPageProducer.

Enregistrement (POST) des données avec des servlets utilisant InternetBeans Express

Effectuer un HTTP POST est simple avec la méthode IxPageProducer.servletPost() :

```
ixPageProducer1.servletPost(this, request, response);
```

Cette méthode doit être appelée dans la méthode doPost() du servlet, avec tout autre code devant être exécuté au cours de l'opération post.

Au moment de la conception, vous devez ajouter un IxSubmitButton pour chaque bouton de soumission de la fiche. Ajoutez un auditeur submitPerformed() pour chacun des IxSubmitButton. L'auditeur doit appeler le code qui doit être exécuté lorsque l'utilisateur appuie sur le bouton. Par exemple, un bouton Suivant doit faire un dataSet.next() et un bouton Précédent un dataSet.prior().

Au moment de l'exécution, lorsque la méthode servletPost() est appelée, elle écrit les nouvelles valeurs du POST dans les composants InternetBeans correspondants et fait passer ces valeurs du côté client au côté serveur. Ensuite l'événement submitPerformed() approprié est déclenché pour le bouton qui a soumis la fiche. Pour poster réellement et enregistrer les modifications dans l'ensemble de données sous-jacent, vous devez appeler les méthodes post() et saveChanges() de l'ensemble de données depuis la méthode submitPerformed(). La méthode doPost() du servlet appelle ensuite doGet() ou directement IxPageProducer.servletGet() pour afficher la nouvelle page.

Analyse des pages

Contrairement à XML, qui est strict, l'analyseur HTML est laxiste. En particulier, les noms des éléments (balises) et des attributs HTML ne distinguent pas les majuscules des minuscules. En revanche, XHTML distingue les majuscules des minuscules ; les noms standard sont par définition en minuscules.

Pour aller plus vite, les noms d'éléments et d'attributs HTML seront convertis en minuscules au standard XHTML pour le stockage. Lors de la recherche d'un attribut particulier, utilisez donc des minuscules.

Quand les composants InternetBeans Express sont mis en correspondance avec les contrôles HTML de la page, leurs attributs sont fusionnés, les propriétés définies dans le composant InternetBeans Express étant prioritaires. Lorsque vous définissez des propriétés dans le concepteur, vous devez vous demander si vous voulez vraiment remplacer tel attribut HTML particulier en définissant la propriété correspondante du composant. Par exemple, si le concepteur de page web crée un `textarea` d'une certaine taille, ce serait gênant que cette taille soit modifiée au moment de la génération dynamique du contrôle.

Génération de tableaux

C'est une tâche assez courante et complexe d'afficher les données d'un tableau selon un format particulier. Par exemple, certaines cellules peuvent être regroupées et les couleurs des lignes alternées.

Il suffit au concepteur de page web de fournir suffisamment de lignes fictives pour montrer l'aspect du tableau (par exemple deux lignes pour représenter l'alternance des couleurs). Quand le tableau réel sera généré par le composant `IxTable`, cet aspect sera automatiquement répété.

Vous pouvez définir des afficheurs de cellules par classe, ou affecter à chaque colonne son propre `IxTableCellRenderer` pour permettre la personnalisation du contenu ; par exemple, les valeurs négatives peuvent apparaître en rouge (de préférence en définissant un style CSS approprié et non en codant en dur la couleur rouge).

Vous trouverez un tutoriel utilisant des InternetBeans dans un servlet, voir Chapitre 12, "Tutoriel : Crédation d'un servlet avec InternetBeans Express".

Utilisation d'InternetBeans Express avec les JSP

Le moyen d'utiliser InternetBeans Express avec des JSP est la bibliothèque de balises InternetBeans Express, définie dans le fichier `internetbeans.tld`. La bibliothèque de balises contient un ensemble de balises InternetBeans pouvant être utilisées dans votre fichier JSP chaque fois que vous voulez utiliser un composant InternetBeans. Ces balises nécessitent peu de code, mais une fois la JSP intégrée à un servlet, elles se retrouvent sous forme de composants InternetBeans complets insérés dans le code.

Pour utiliser InternetBeans Express dans une JSP, il faut qu'une ligne de code particulière soit présente au début de la JSP. C'est la directive `taglib` ; elle indique que les balises de la bibliothèque de balises InternetBeans

Express seront utilisées dans la JSP et elle spécifie le préfixe utilisé pour ces balises. La balise `taglib` permettant d'utiliser la bibliothèque de balises InternetBeans doit ressembler à ceci :

```
<%@ taglib uri="/internetbeans.tld" prefix="ix" %>
```

Si vous voulez instancier des classes dans vos scriptlets, et si vous ne voulez pas saisir les noms complets des classes, vous pouvez importer des fichiers ou des paquets dans votre JSP en utilisant la directive `page`. Cette directive `page` peut spécifier que le paquet `com.borland.internetbeans` doit être importé dans la JSP. La directive `page` doit ressembler à ceci :

```
<%@ page import="com.borland.internetbeans.* , com.borland.dx.dataset.* ,  
com.borland.dx.sql.dataset.*" %>
```

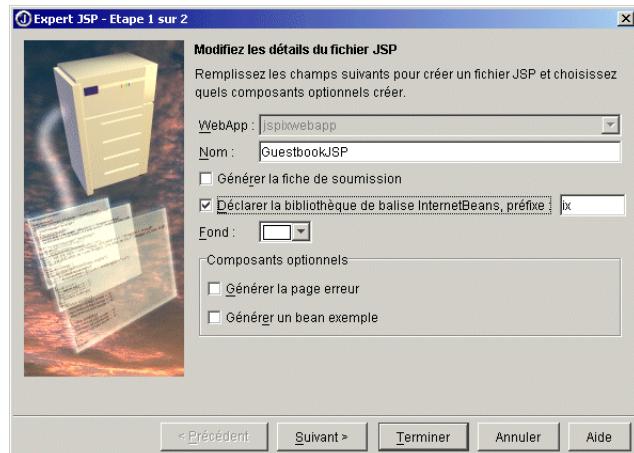
Souvenez-vous que des directives comme la directive `taglib` ou la directive `page` doivent toujours être les premières lignes de votre JSP.

L'expert JSP de JBuilder insère pour vous une directive `taglib` et une directive `page` si vous sélectionnez l'option "Déclarer la bibliothèque de balises InternetBeans, préfixe :". Saisissez à la suite de l'option le préfixe que vous voulez utiliser. Ce préfixe sera ensuite utilisé avec chaque balise InternetBeans de votre JSP pour l'identifier en tant que balise InternetBeans. Si cette option est sélectionnée, l'expert JSP exécute également les autres étapes nécessaires à l'ajout de la bibliothèque InternetBeans à votre projet. Voici ces étapes :

- Il copie la totalité du fichier `internetbeans.jar` dans le répertoire `/WEB-INF/lib` de la WebApp
- Il ajoute un mappage entre le fichier `internetbeans.tld`, `internetbeans.jar` et le fichier `web.xml`. Dans tout JAR de bibliothèque de balises, il existe un emplacement standard (`/META-INF/taglib.tld`) qui contient le descripteur de la bibliothèque. Cette copie est utilisée pour la compilation et l'exécution de la JSP.

L'expert JSP ajoute également un fichier `internetbeans.tld` à la racine du projet. En réalité, il pointe sur ce fichier à l'intérieur du JAR qui a été copié. Etant dans le JAR, il n'est accessible qu'en lecture.

L'option Déclarer la bibliothèque de balises InternetBeans est affichée comme suit dans l'expert JSP :



Voici un exemple de balise InternetBeans lors de son utilisation dans une JSP :

```
<ix:database id="database1"
  driver="com.borland.datastore.jdbc.DataStoreDriver"
  url="jdbc:borland:dslocal...\\guestbook\\guestbook.jds"
  username="user">
</ix:database>
```

Cet exemple utilise la balise `database`. Le préfixe `ix` pourrait être n'importe quel texte. Cela ne dépend que du préfixe spécifié dans l'expert JSP. Lorsque vous utiliserez réellement la balise `database` dans une JSP, vous voudrez la plupart du temps imbriquer d'autres balises à l'intérieur cette balise, par exemple une balise `query`. Cela n'est pas obligatoire, mais cela facilite la lecture du code de la JSP.

Pour trouver un tutoriel sur ce sujet, voir Chapitre 13, "Tutoriel : Création d'une JSP avec InternetBeans Express".

Tableau des balises InternetBeans

Les balises contenues dans la bibliothèque de balises InternetBeans Express sont décrites dans le tableau ci-dessous. Les attributs mentionnés en gras sont obligatoires.

Nom de balise	Description	Attributs
database	Définit une <code>Database DataExpress</code>	<ul style="list-style-type: none"> • id - texte utilisé pour identifier cette base de données • driver - propriété <code>driver</code> de <code>Database</code> • url - propriété <code>url</code> de <code>Database</code> • username • password
query	Définit un <code>QueryDataSet DataExpress</code>	<ul style="list-style-type: none"> • id - texte utilisé pour identifier cette requête • database - identifie la base de données à laquelle appartient cette requête. N'est pas obligatoire car implicite si la balise query est imbriquée dans la balise database. Si la balise query n'est pas imbriquée dans la balise database, cet attribut doit être spécifié. • statement - l'instruction SQL exécutée par cette requête.
control	Définit un <code>IxControl InternetBeans Express</code>	<ul style="list-style-type: none"> • id - texte utilisé pour identifier ce contrôle • tupleModel - le <code>tupleModel</code> pour ce contrôle • dataSet - identifie l'ensemble de données (la requête) auquel ce contrôle est connecté. Soit le <code>dataSet</code> soit le <code>tupleModel</code> est requis, mais vous ne pouvez pas définir les deux attributs. • columnName - identifie le <code>columnName</code> auquel ce contrôle est connecté.
image	Définit une <code>IxImage InternetBeans Express</code>	<ul style="list-style-type: none"> • id - texte utilisé pour identifier cette image • tupleModel - le <code>tupleModel</code> pour ce contrôle • dataSet - identifie l'ensemble de données (la requête) auquel cette image est connectée. Soit le <code>dataSet</code> soit le <code>tupleModel</code> est requis, mais vous ne pouvez pas définir les deux attributs. • columnName - identifie le <code>columnName</code> auquel cette image est connectée.

Nom de balise	Description	Attributs
submit	Définit un <code>IxSubmitButton</code> InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code> - texte utilisé pour identifier ce bouton de soumission • <code>methodName</code> - nom de la méthode qui sera exécutée lorsque l'utilisateur appuiera sur ce bouton.
table	Définit une <code>IxTable</code> InternetBeans Express	<ul style="list-style-type: none"> • <code>id</code> - texte utilisé pour identifier cette table • <code>dataSet</code> - identifie l'ensemble de données (la requête) auquel cette table est connectée. • <code>tableModel</code> - le modèle de données pour cette table. Soit le <code>dataSet</code> soit le <code>tableModel</code> est requis, mais vous ne pouvez pas définir les deux attributs.

Il n'y a que six balises dans la bibliothèque de balises InternetBeans Express, alors qu'il existe dix-sept composants InternetBeans. Cela peut sembler une grave carence, pourtant il n'en est rien. La balise `control` correspond à un `IxControl`, qui délègue à tous les autres InternetBeans spécifiques à un contrôle particulier. Les seuls InternetBeans non couverts par la bibliothèque de balises sont `IxSpan` et `IxLink`. Aucun n'est utile dans une JSP, car vous pouvez simplement vous servir de votre propre scriptlet JSP pour accomplir la même tâche.

Bien évidemment, il est également possible d'utiliser des InternetBeans directement, comme tout autre bean ou classe Java. L'utilisation de la bibliothèque de balises est simplement beaucoup plus commode et apporte quelques fonctionnalités supplémentaires (comme la maintenance d'un état session pour la saisie de données).

Format de internetbeans.tld

Sachez qu'il est toujours possible de trouver dans le fichier `internetbeans.tld` des conseils sur l'utilisation des diverses balises. Pour cela, ouvrez le fichier dans l'éditeur de JBuilder. Le fichier ne peut pas (et ne doit pas) être modifié.

Les informations en tout début du fichier `internetbeans.tld` sont sans intérêt pour vous. Celles qui vous seront utiles de comprendre commence avec la première balise `<tag>` du fichier. Chaque balise `<tag>` représente la définition d'une balise InternetBeans.

Au début de chaque définition de balise, se trouve une balise `<name>` qui indique le nom de la balise. La première est la balise `database`. Imbriqués dans chaque définition de balise, vous verrez des balises `<tagclass>`, `<info>` et `<attribute>`. Pour un exemple de définition de balise InternetBeans tag,

voir le fragment du fichier internetbeans.tld suivant ; il définit la balise submit :

```
<tag>
<name>submit</name>
<tagclass>com.borland.internetbeans.taglib.SubmitTag</tagclass>
<bodycontent>JSP</bodycontent>
<info>Submit button or submit image control</info>
<attribute>
<name>id</name>
<required>false</required>
<rteprvalue>false</rteprvalue>
</attribute>
<attribute>
<name>methodName</name>
<required>true</required>
<rteprvalue>false</rteprvalue>
</attribute>
</tag>
```

La balise `<tagclass>` indique le nom de la classe du paquet `com.borland.internetbeans.taglib` responsable de l'interprétation de la balise InternetBeans lors de son utilisation dans une JSP. La balise `<info>` fournit une description de la balise InternetBeans.

La balise `<attribute>` décrit un attribut de la balise InternetBeans. Il y a une balise `<attribute>` pour chaque attribut. Vous pouvez les considérer comme ses propriétés. Imbriquées dans la balise `<attribute>` vous verrez ces propriétés. Chaque propriété a un nom, une valeur booléenne indiquant s'il s'agit ou non d'une propriété obligatoire, ainsi qu'une valeur booléenne indiquant si sa valeur peut ou non être définie par une expression Java. Le nom se trouve dans la balise `<name>`, le booléen indiquant s'il s'agit ou non d'une propriété obligatoire dans la balise `<required>`, le booléen indiquant si sa valeur peut ou non être définie par une expression Java dans la balise `<rteprvalue>`. Les propriétés ne pouvant être définies par une expression exigent une valeur littérale.

12

Tutoriel : Création d'un servlet avec InternetBeans Express

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Ce tutoriel montre comment construire un servlet en utilisant InternetBeans. Lorsque vous aurez terminé ce tutoriel, vous aurez un servlet utilisant un module de données pour interroger une table dans un JDataStore, affichant les commentaires du guestbook dans un contrôle IxTable et permettant aux visiteurs du site de saisir leurs propres commentaires et de les voir affichés dans le guestbook. La version complète de l'application créée dans ce tutoriel se trouve dans le répertoire <JBuilder>/samples/WebApps/guestbook.

Ce tutoriel suppose que vous vous êtes familiarisé avec Java et les servlets Java, avec l'EDI de JBuilder et avec les JDataStore. Pour plus d'informations sur Java, voir *Introduction à Java*. Pour plus d'informations sur les servlets Java, voir "Utilisation des servlets" dans le *Guide du développeur d'applications web*. Pour plus d'informations sur l'EDI de JBuilder, voir "L'environnement de JBuilder" dans *Construction d'applications avec JBuilder*. Pour plus d'informations sur JDataStore, voir le *Guide du développeur JDataStore*.

Remarque

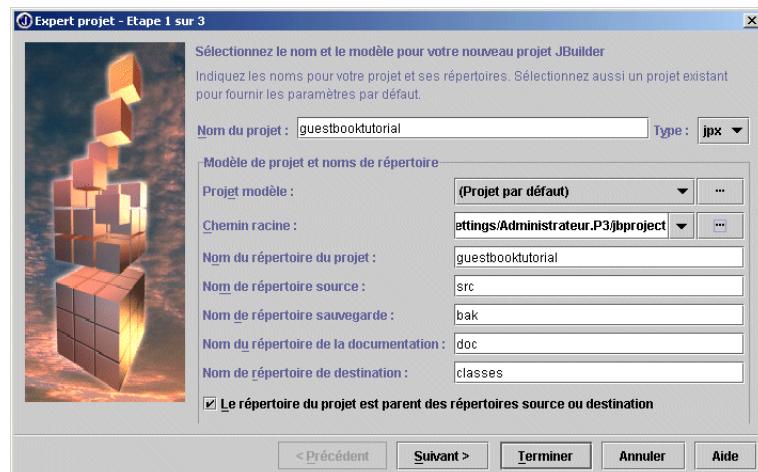
Ce tutoriel suppose que vous avez entré vos informations de licence dans le Gestionnaire de licence JDataStore. Pour plus d'informations, voir "Première utilisation de JDataStore" dans le *Guide du développeur JDataStore*.

Si vous avez des suggestions à faire pour améliorer ce tutoriel, envoyez un e-mail à jpgpubs@borland.com.

Etape 1 : Création d'un nouveau projet

- 1 Sélectionnez Fichier | Nouveau projet pour afficher l'expert projet.
- 2 Saisissez un nom pour le projet, par exemple guestbooktutorial.
- 3 Cliquez sur Terminer.
- 4 Un nouveau projet est créé, contenant un fichier HTML qui décrit le projet.

Figure 12.1 Expert projet

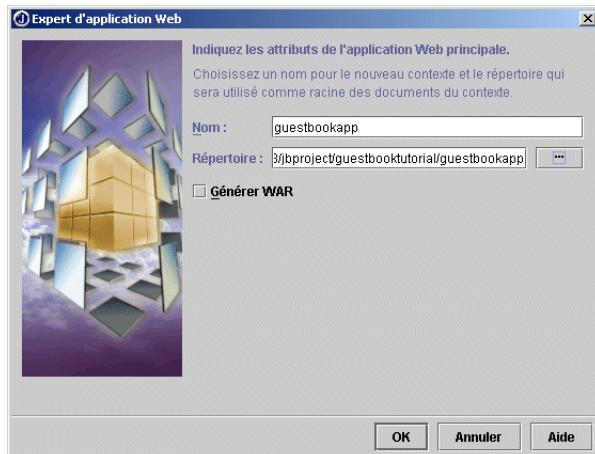


Etape 2 : Création d'une nouvelle WebApp

Cette étape est facultative mais conseillée. Pour plus d'informations sur les WebApp et les fichiers WAR, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

- 1 Sélectionnez Fichier | Nouveau.
- 2 Cliquez sur l'onglet Web de la galerie d'objets. Sélectionnez Application Web.
- 3 Cliquez sur OK. L'expert WebApp apparaît.
- 4 Saisissez un nom pour la WebApp, par exemple guestbookapp.
- 5 Choisissez le bouton Points de suspension situé à droite du champ Répertoire.
- 6 Saisissez le nom du répertoire racine de la WebApp, par exemple guestbookapp.
- 7 Cliquez sur OK.

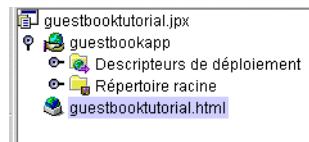
- 8 Cliquez sur Oui pour créer le répertoire.
- 9 Conservez non cochée l'option Générer WAR, car vous ne voudrez certainement pas déployer réellement cette application.
- 10 L'expert doit se présenter ainsi :



- 11 Cliquez sur OK.

Un nœud WebApp, `guestbookapp`, est affiché dans le volet projet. Développez ce nœud pour voir les nœuds Répertoire racine et Descripteur de déploiement.

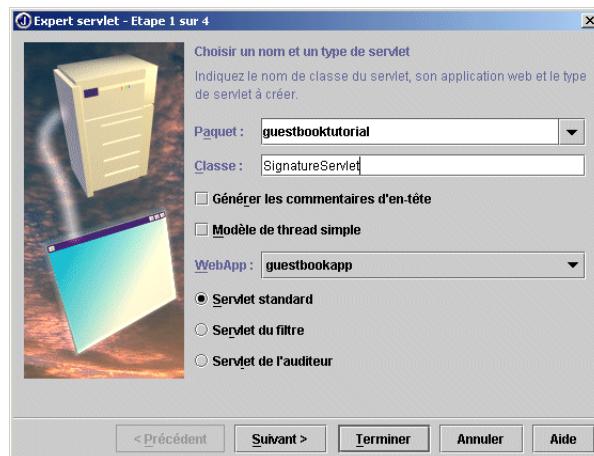
Figure 12.2 Nœud WebApp dans le volet projet



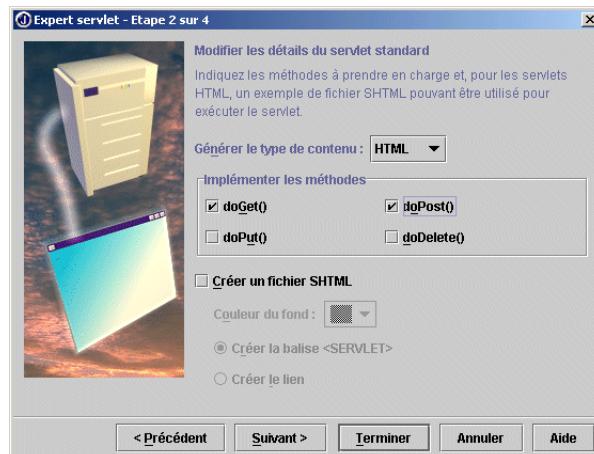
Etape 3 : Utilisation de l'expert servlet

- 1 Sélectionnez Fichier | Nouveau.
- 2 Cliquez sur l'onglet Web de la galerie d'objets. Sélectionnez Servlet.
- 3 Cliquez sur OK. L'expert servlet apparaît.
- 4 Entrez un nom pour la classe. `SignatureServlet`

- 5 Sélectionnez guestbookapp pour la WebApp. L'expert doit se présenter ainsi :



- 6 Cliquez sur Suivant pour passer à l'étape 2 de l'expert.
- 7 Vérifiez que le Générer le type de contenu est défini par HTML.
- 8 Assurez-vous que les méthodes doGet() et doPost() sont sélectionnées.
- 9 Vérifiez que l'option Créer un fichier SHTML n'est pas cochée. L'expert doit se présenter ainsi :



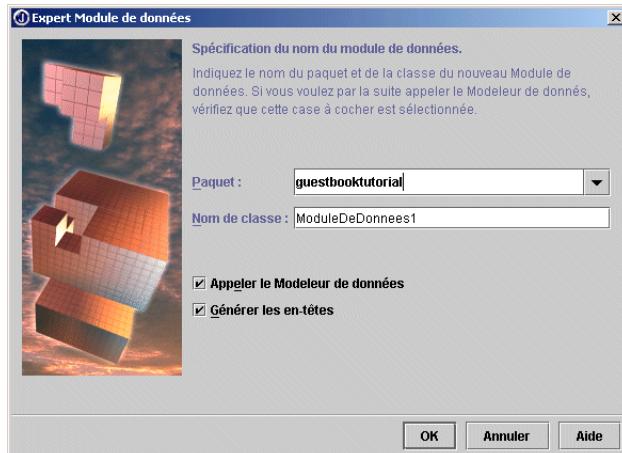
- 10 Cliquez sur Terminer. Un fichier SignatureServlet.java est ajouté à votre projet.



- 11 Cliquez sur le bouton Tout enregistrer dans la barre d'outils.

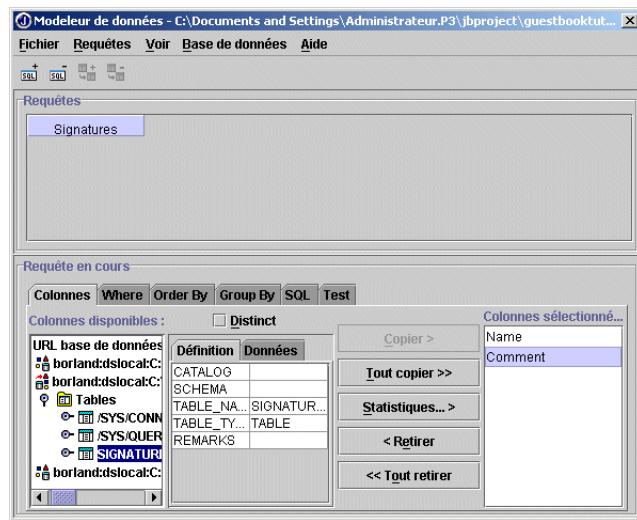
Etape 4 : Cr éation du module de donn ées

- 1 Sélectionnez Fichier | Nouveau.
- 2 Sélectionnez l'icône Module de données dans la page Nouveau de la galerie d'objets.
- 3 Cliquez sur OK. L'expert Module de données apparaît.



- 4 Vérifiez que l'option Appeler le modeleur de données est sélectionnée.
- 5 Cliquez sur OK. L'expert Modéleur de données apparaît.
- 6 Allez dans le menu Base de données et sélectionnez Ajouter une URL de connexion.
- 7 Comme pilote, sélectionnez com.borland.datastore.jdbc.DataStoreDriver dans la liste déroulante.
- 8 Comme URL, tapez le chemin d'accès au fichier guestbook.jds qui se trouve dans le dossier <JBuilder>/samples/WebApps/guestbook.
- 9 Cliquez sur OK. Une nouvelle URL base de données est ajoutée.
- 10 Connectez la nouvelle URL base de données en cliquant dessus et en entrant user dans le dialogue d'ouverture. Le mot de passe n'est pas nécessaire.
- 11 Ouvrez la liste des tables en cliquant sur le nœud Tables dans l'arborescence des Colonnes disponibles.
- 12 Sélectionnez la table SIGNATURES.

13 Cliquez sur le bouton Tout copier. L'expert doit se présenter ainsi :



14 Choisissez Enregistrer dans le menu Fichier.

15 Choisissez Quitter dans le menu Fichier. Le fichier `ModuleDeDonnees1.java` est mis à jour avec les informations de connexion appropriées.

16 Cliquez sur le bouton Tout enregistrer dans la barre d'outils.



Etape 5 : Conception du modèle de la page HTML



- 1 Cliquez sur le bouton Ajouter des fichiers/paquets de la barre d'outils projet.
- 2 Cliquez sur le bouton Projet sur l'onglet Explorateur de la boîte de dialogue Ajout de fichiers ou paquets au projet.
- 3 Sélectionnez le répertoire de la WebApp (ex. : guestbookapp).
- 4 Tapez le nom de fichier : `gb1.html`
- 5 Cliquez sur OK.
- 6 Cliquez sur OK pour créer le fichier.
- 7 Double-cliquez sur le fichier dans le volet projet pour l'ouvrir. Vous verrez un message d'avertissement signalant que le fichier ne peut être ouvert. C'est que le fichier n'a pas encore été enregistré. Vous pouvez ignorer cet avertissement en toute sécurité.
- 8 Sélectionnez l'onglet Source pour ouvrir le code source du HTML. A ce stade, il sera vierge.

- 9** Entrez le code HTML suivant dans le fichier. Vous pouvez aussi le copier/coller à partir de ce tutoriel.

```
<html>
<head>

<title>Signatures Guestbook</title>

</head>

<body>

<h1>Signez le guestbook</h1>

<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
<tr>
<th>Nom</th><th>Commentaire</th>
</tr>
<tr>
<td>Léo</td><td>Tout baigne !</td>
</tr>
</table>

<form method="POST">
<p>Entrez votre nom :</p>

<input type="text" id="Name" name="Name" size="50">

<p>Entrez votre commentaire :</p>

<input type="text" id="Comment" name="Comment" size="100">
<p>
<input type="submit" name="submit" value="Soumettre"></p>
</form>

</body>
</html>
```

Notez que la balise `<table>` contient des données fictives. Ces données seront remplacées par des données réelles provenant du JDataStore lorsque le servlet s'exécutera. Ces données fictives donnent une idée de l'aspect que prendront les données réelles lorsqu'elles seront affichées. Pour plus d'informations sur la façon dont InternetBeans Express affiche les tables, voir "Génération de tableaux", page 11-6.

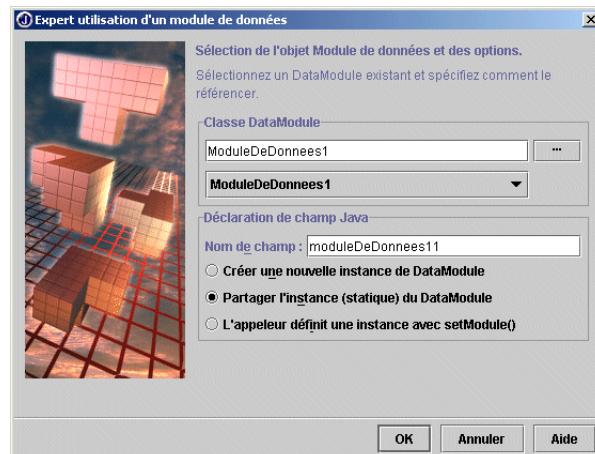


Cliquez sur le bouton Tout enregistrer dans la barre d'outils. Cliquez sur l'onglet Vue. Dans la vue, le code HTML doit ressembler à ceci :

The screenshot shows a web page titled "Signez le guestbook". It features a table with two rows: "Nom" and "Commentaire". The first row contains "Léo" and "Tout baigne !". Below the table are two input fields labeled "Entrez votre nom:" and "Entrez votre commentaire:", followed by a "Soumettre" button. At the bottom, there is a status bar showing the URL "http://localhost:8080/jr.P3/jbproject/guestbooktutorial/guestbookapp/gb1.html" and tabs for "Vue", "Source", and "Historique".

Etape 6 : Connexion du servlet au module de données

- 1 Sélectionnez Projet | Construire le projet “guestbooktutorial.jpx”. Cela construit le projet et crée le fichier `ModuleDeDonnees1.class`.
- 2 Ouvrez le fichier `SignatureServlet.java` dans l'éditeur.
- 3 Sélectionnez Experts | Utiliser un module de données.
- 4 Cliquez sur le bouton Points de suspension en regard du champ Classe DataModule.
- 5 Sélectionnez `ModuleDeDonnees1` dans le paquet `guestbooktutorial`.
- 6 Assurez-vous que l'option Partager l'instance (statique) du DataModule est sélectionnée. L'expert doit se présenter ainsi :



- 7 Cliquez sur OK.
- 8 Une ligne de code est ajoutée à la méthode `jbInit()` pour associer le module de données et le servlet.

Etape 7 : Conception du servlet

Dans cette étape, vous allez utiliser le concepteur pour ajouter des composants InternetBeans au servlet. Ces composants ne seront pas visibles dans le concepteur, car l'interface utilisateur graphique du servlet se trouve en réalité dans le fichier HTML. Mais, les propriétés des composants seront visibles dans l'inspecteur. Lorsque le servlet sera exécuté, les composants InternetBeans que vous ajoutez au cours de cette étape remplaceront les données fictives du fichier HTML par des données provenant du JDataStore.

- 1 Vérifiez que le fichier `SignatureServlet.java` est ouvert dans l'éditeur.
- 2 Cliquez sur l'onglet Conception pour ouvrir le concepteur d'interface utilisateur.
- 3 Sélectionnez l'onglet InternetBeans de la palette de composants.
- 4 Sélectionnez l'icône `IxPageProducer` et placez un `IxPageProducer` dans le servlet en cliquant dans le concepteur.

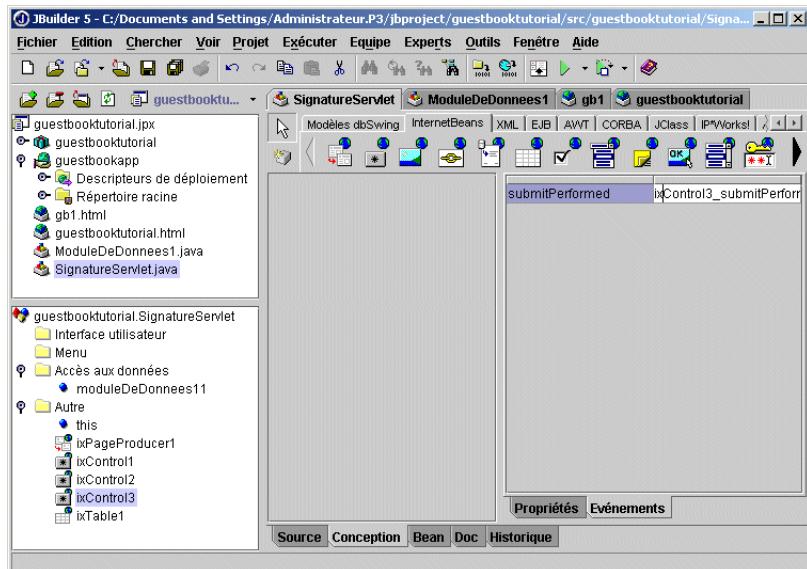
- 5 Définissez la propriété `ixPageProducer.dataModule` par `ModuleDeDonnees11`.
- 6 Définissez la propriété `ixPageProducer.htmlFile` pour qu'elle pointe sur le fichier `gb1.html` (en incluant le chemin d'accès).
- 7 Sélectionnez l'icône `IxControl` dans la palette. Lâchez le contrôle `IxControl` dans le servlet en cliquant dans le concepteur.

- 8 Placez deux `IxControl` supplémentaires dans le servlet en répétant les étapes précédentes.
- 9 Sélectionnez `ixControl1`.
- 10 Définissez la propriété `ixControl1.dataSet` par l'ensemble de données Signatures qui apparaît dans la liste déroulante.
- 11 Définissez la propriété `ixControl1.columnName` par "Name".
- 12 Définissez la propriété `ixControl1.pageProducer` par `ixPageProducer1`.
- 13 Définissez la propriété `ixControl1.controlName` par "Name".
- 14 Sélectionnez `ixControl2`.
- 15 Définissez la propriété `ixControl2.dataSet` par l'ensemble de données Signatures qui apparaît dans la liste déroulante.
- 16 Définissez la propriété `ixControl2.columnName` par "Comment".

Etape 7 : Conception du servlet

- 17 Définissez la propriété ixControl2.pageProducer par ixPageProducer1.
- 18 Définissez la propriété ixControl2.controlName par "Comment".
- 19 Sélectionnez l'icône **IxTable** dans la palette. Lâchez le contrôle IxTable dans le servlet en cliquant dans le concepteur.

- 20 Définissez la propriété ixTable1.pageProducer par ixPageProducer1.
- 21 Définissez la propriété ixTable1.dataSet par l'ensemble de données Signatures qui apparaît dans la liste déroulante.
- 22 Définissez la propriété ixTable1.elementId par "guestbooktable".
- 23 Sélectionnez ixControl3.
- 24 Définissez la propriété ixControl3.pageProducer par ixPageProducer1.
- 25 Définissez la propriété ixControl3.controlName par "submit".



- 26 Sélectionnez l'onglet Evénements de l'inspecteur.
- 27 Cliquez une fois sur submitPerformed dans l'inspecteur pour définir l'événement submitPerformed() par ixControl3_submitPerformed. Cela ajoute un auditeur d'événement à ixControl3. Appuyez sur Entrée pour générer la méthode ixControl3_submitPerformed(). Cela ouvre l'éditeur à la page Source, le curseur étant positionné dans la nouvelle méthode.

- 28 Cliquez sur le bouton Tout enregistrer dans la barre d'outils.

Etape 8 : Modification du servlet.

- 1 Vérifiez que le fichier SignatureServlet.java est ouvert dans l'éditeur.
- 2 Supprimez ce qui a été généré par l'expert comme corps de la méthode doGet() du servlet.
- 3 Supprimez ce qui a été généré par l'expert comme corps de la méthode doPost() du servlet.
- 4 Tapez la ligne de code suivante dans le corps de la méthode doGet() :

```
ixPageProducer1.servletGet(this, request, response);
```

La méthode doGet() est désormais complète. Souvent, la seule chose à faire ici sera d'appeler la méthode servletGet() du IxPageProducer.

- 5 Tapez les lignes de code suivantes dans le corps de la méthode doPost() :

```
ModuleDeDonnees1 dm = (ModuleDeDonnees1)
ixPageProducer1.getSessionDataModule(request.getSession());
dm.getSignatures().insertRow(false);
ixPageProducer1.servletPost(this, request, response);
doGet(request, response);
```

Lorsque la fiche est postée, ce code obtient une instance par session du module de données, insère une ligne vide, appelle IxPageProducer.servletPost() pour la remplir avec les valeurs saisies par l'utilisateur, et appelle à nouveau doGet() pour afficher les données qui ont été postées.

- 6 Ensuite, vous devez remplir le corps de la méthode ixControl3_submitPerformed(). Cette méthode est appelée par la méthode servletPost(). Tapez le code suivant dans le corps de la méthode ixControl3_submitPerformed() :

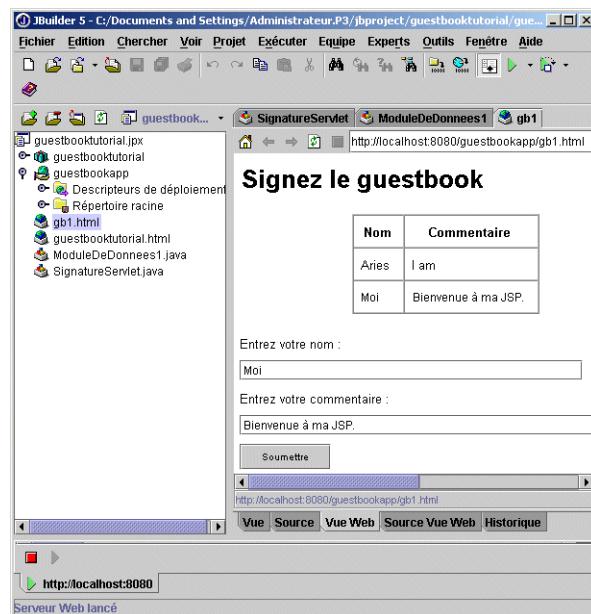
```
ModuleDeDonnees1 dm = (ModuleDeDonnees1)
ixPageProducer1.getSessionDataModule(e.getSession());
dm.getSignatures().post();
dm.getSignatures().saveChanges();
```

Ce code obtient une instance par session du module de données, poste et sauvegarde la saisie de l'utilisateur dans le JDataStore. Notez que cette instance par session est différente de l'instance partagée stockée dans la variable ModuleDeDonnees1.

Etape 9 : Exécution du servlet

- 1 Cliquez avec le bouton droit de la souris sur le fichier SignatureServlet.java dans le volet projet.
- 2 Sélectionnez Exécution Web dans le menu contextuel.

- 3 Le servlet s'exécute dans l'EDI de JBuilder.
- 4 Testez le servlet. Supprimez les valeurs existant dans les champs Name et Comment. Entrez votre nom et un commentaire ; cliquez sur Soumettre.
- 5 Votre nom et votre commentaire sont affichés dans la table ; ils sont enregistrés dans le JDataStore.



- 6 Pour arrêter le servlet, cliquez sur le bouton Réinitialiser le programme dans l'onglet Serveur web du volet message.

Déploiement du servlet

Pour plus d'informations sur le déploiement de votre servlet, voir Chapitre 16, "Déploiement de votre application web".

13

Tutoriel : Cr éation d'une JSP avec InternetBeans Express

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Ce tutoriel montre comment construire une JSP en utilisant InternetBeans. Lorsque vous aurez terminé ce tutoriel, vous aurez une JSP interrogeant une table dans un JDataStore, affichant les commentaires du guestbook dans un contrôle `IxTable` et permettant aux visiteurs du site de saisir leurs propres commentaires et de les voir affichés dans le guestbook. La version complète de l'application créée dans ce tutoriel se trouvent dans le répertoire `<JBuilder>/samples/WebApps/jspinternetbeans`.

Ce tutoriel suppose que vous vous êtes familiarisé avec Java et les pages JavaServer (JSP), avec l'EDI de JBuilder et avec les JDataStore. Pour plus d'informations sur Java, voir *Introduction à Java*. Pour plus d'informations sur les pages JavaServer, voir Chapitre 9, "Développement des pages JavaServer". Pour plus d'informations sur l'EDI de JBuilder, voir "L'environnement de JBuilder" dans *Construction d'applications avec JBuilder*. Pour plus d'informations sur JDataStore, voir le *Guide du développeur JDataStore*.

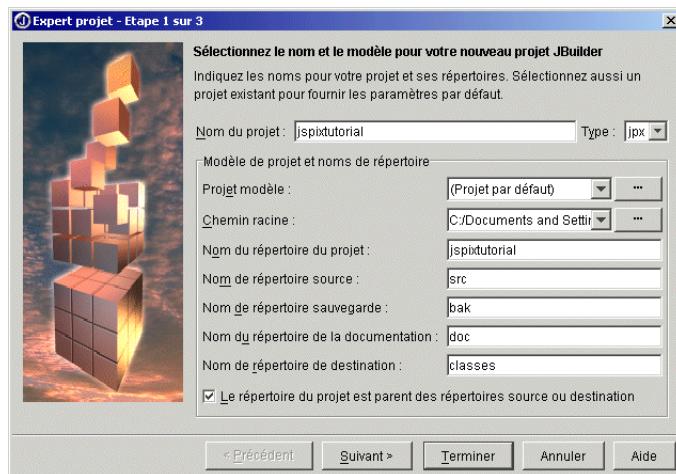
Remarque Ce tutoriel suppose que vous avez entré vos informations de licence dans le Gestionnaire de licence JDataStore. Pour plus d'informations, voir "Première utilisation de JDataStore" dans le *Guide du développeur JDataStore*.

Si vous avez des suggestions à faire pour améliorer ce tutoriel, envoyez un e-mail à jpgpubs@borland.com.

Etape 1 : Création d'un nouveau projet

- 1 Sélectionnez Fichier | Nouveau projet pour afficher l'expert projet.
- 2 Saisissez un nom pour le projet, par exemple jspixtutorial.
- 3 Cliquez sur Terminer.
- 4 Un nouveau projet est créé, contenant un fichier HTML qui décrit le projet.

Figure 13.1 Expert projet

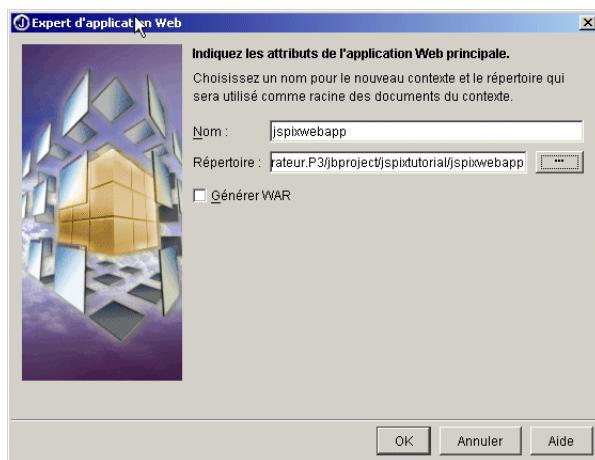


Etape 2 : Création d'une nouvelle WebApp

Cette étape est facultative mais conseillée. Pour plus d'informations sur les WebApp et les fichiers WAR, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

- 1 Choisissez Nouveau dans le menu Fichier.
- 2 Cliquez sur l'onglet Web de la galerie d'objets. Sélectionnez Application Web.
- 3 Cliquez sur OK. L'expert WebApp apparaît.
- 4 Saisissez un nom pour la WebApp, par exemple jspixwebapp.
- 5 Choisissez le bouton Points de suspension situé à droite du champ Répertoire.
- 6 Saisissez le nom du répertoire racine de la WebApp, par exemple jspixwebapp.
- 7 Cliquez sur OK.

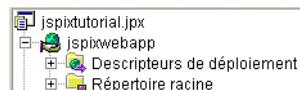
- 8 Cliquez sur Oui pour créer le répertoire.
- 9 Conservez non cochée l'option Générer WAR, car vous ne voudrez certainement pas déployer réellement cette application. L'expert doit se présenter ainsi :



- 10 Cliquez sur OK.

Un nœud WebApp, `jspxwebapp`, est affiché dans le volet projet. Développez ce nœud pour voir les nœuds Répertoire racine et Descripteur de déploiement.

Figure 13.2 Nœud WebApp dans le volet projet



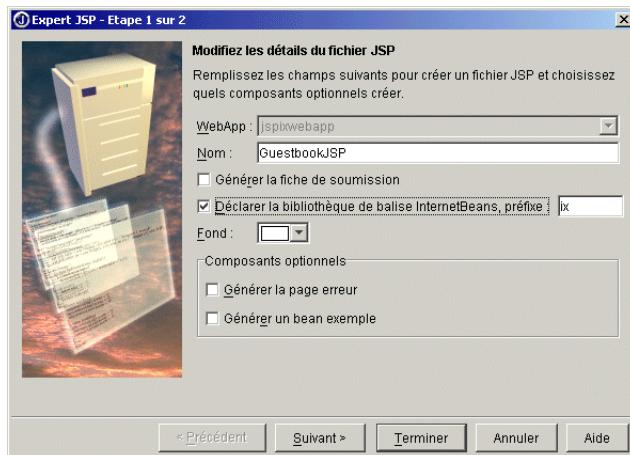
Etape 3 : Utilisation de l'expert JSP

Dans cette étape, vous allez créer le squelette de la JSP en utilisant l'expert JSP.

- 1 Sélectionnez Fichier | Nouveau.
- 2 Cliquez sur l'onglet Web. Sélectionnez Page JavaServer.
- 3 Cliquez sur OK. L'expert JSP apparaît.
- 4 Sélectionnez la WebApp dans la liste déroulante. Sélectionnez `jspxwebapp` si vous avez créé cette WebApp à l'étape 2.
- 5 Entrez un nom pour la JSP : `GuestbookJSP`
- 6 Désélectionnez Générer la fiche de soumission.

- 7 Cochez Déclarer la bibliothèque de balises InternetBeans, en conservant la valeur par défaut du préfixe : ix.
- 8 Désélectionnez Générer un bean exemple. L'expert JSP doit ressembler à ceci :

Figure 13.3 Expert JSP



- 9 Cliquez sur Terminer. Un fichier GuestbookJSP.jsp est ajouté au nœud Répertoire racine de votre WebApp dans le volet projet. Développez le nœud Répertoire racine pour voir le fichier. La JSP contient la directive page et la directive taglib nécessaires à l'utilisation de la bibliothèque de balises InternetBeans. L'expert JSP prend également en charge les étapes permettant d'ajouter la bibliothèque InternetBeans à votre projet, comme décrit dans "Utilisation d'InternetBeans Express avec les JSP", page 11-6.

Etape 4 : Conception de la partie HTML de la JSP

- 1 Ouvrez le fichier GuestbookJSP.jsp dans l'éditeur.
- 2 Modifiez le contenu de la balise <title> en JSP/InternetBeans Tutorial.
- 3 Modifiez le contenu de la balise <h1> en Signez le guestbook.
- 4 Tapez la ligne de code HTML suivante dans le corps du fichier, sous la balise </h1> :

```
<table id="guestbooktable" align="CENTER" cellspacing="0" border="1">
<tr>
<th>Nom</th><th>Commentaire</th>
</tr>
<tr>
<td>Léo</td><td>Tout baigne !</td>
```

```

</tr>
</table>

<form method="POST">
<p>Entrez votre nom :</p>

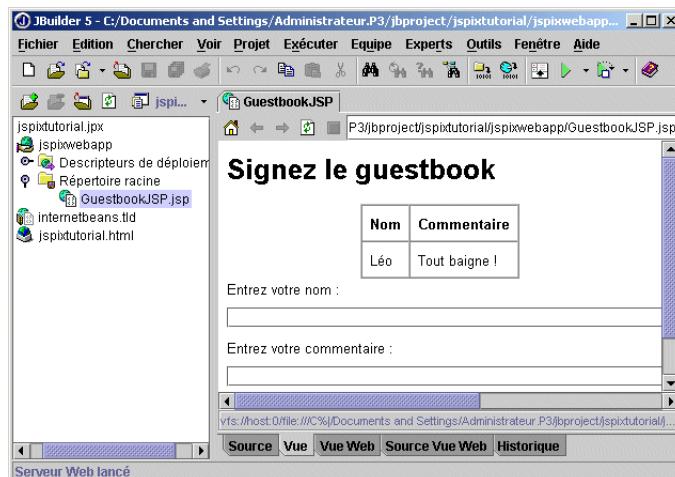
<input type="text" name="Name" size="50">

<p>Entrez votre commentaire :</p>

<input type="text" name="Comment" size="100">
<p>
<input type="submit" name="submit" value="Soumettre"></p>
</form>

```

Lorsque vous aurez terminé, le HTML sur l'onglet Vue doit ressembler à ceci :



Etape 5 : Ajout de la balise InternetBeans database

- Ajoutez la balise database d'ouverture montrée ci-après en **gras**. Modifiez la valeur de l'attribut url de la balise database pour qu'il pointe sur le JDataStore guestbook.jds, dans <JBuilder>\samples\WebApps\guestbook.

```

<h1>
Signez le guestbook
</h1>

<ix:database id="database1" driver="com.borland.datastore.jdbc.DataStoreDriver"
url="jdbc:borland:dslocal:C:\\JBuilder\\samples\\WebApps\\guestbook\\
guestbook.jds"
username="user">

```

Etape 6 : Ajout de la balise InternetBeans query

```
<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
```

- 2 Modifiez le chemin d'accès à guestbook.jds dans l'attribut url.
guestbook.jds se trouve dans le dossier <jbuilder>/samples/WebApps/guestbook.
- 3 Ajoutez la balise database de fermeture montrée ci-après en **gras**.

```
</form>

</ix:database>

</body>
```

Etape 6 : Ajout de la balise InternetBeans query

- 1 Ajoutez la balise query d'ouverture montrée ci-après en **gras**.

```
<ix:database id="database1"
driver="com.borland.datastore.jdbc.DataStoreDriver"
url="jdbc:borland:dslocal:C:\\JBuilder\\samples\\WebApps\\guestbook\\
guestbook.jds"
username="user">

<ix:query id="signatures" statement="select * from signatures">

<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
```

Remarquez que vous êtes en train d'imbriquer la balise query dans la balise database. Il s'en suit que l'attribut database de la balise query n'a pas besoin d'être spécifié, il est implicite. Cela rend le code plus élégant.

- 2 Ajoutez la balise query de fermeture montrée ci-après en **gras**.

```
</ix:query>

</ix:database>
```

Etape 7 : Ajout de la balise InternetBeans table

Ajoutez les balises table d'ouverture et de fermeture montrées ci-après en **gras**.

```
<ix:query id="signatures" statement="select * from signatures">

<ix:table dataSet="signatures">

<table id="guestbooktable" align="CENTER" cellspacing="0" border="1"
cellpadding="7">
<tr>
```

```
<th>Nom</th><th>Commentaire</th>
</tr>
<tr>
<td>Léo</td><td>Tout baigne !</td>
</tr>
</table>

</ix:table>
```

```
<form method="POST">
```

Remarquez que vous êtes en train d'envelopper la balise HTML `table` dans la balise InternetBeans `table`. Cela permet au contrôle InternetBeans `IxTable` de comprendre implicitement quelle table il remplace. La balise InternetBeans `table` est imbriquée dans la balise InternetBeans `query`. Cela n'est pas obligatoire, car l'attribut `dataSet` de la table établit clairement la relation. Imbriquer ainsi la table InternetBeans dans la balise `query` rend simplement le code plus élégant.

Etape 8 : Ajout des balises InternetBeans control

Vous allez maintenant ajouter deux balises control pour les deux champs de saisie de texte. Ajoutez les balises montrées ci-après en **gras**.

```
<form method="POST">
<p>Entrez votre nom :</p>

<ix:control dataSet="signatures" columnName="Name">

<input type="text" name="Name" size="50">

</ix:control>

<p>Entrez votre commentaire :</p>

<ix:control dataSet="signatures" columnName="Comment">

<input type="text" name="Comment" size="100">

</ix:control>

<p>
```

Remarquez que vous êtes en train d'envelopper les balises HTML `input` de type texte dans une balise InternetBeans `control`. Cela permet aux contrôles InternetBeans `IxControls` de comprendre implicitement quels champs de saisie ils remplacent.

Etape 9 : Ajout de la balise InternetBeans submit

Ajoutez les balises submit d'ouverture et de fermeture montrées ci-après en **gras**.

```
<input type="text" name="Comment" size="100">
</ix:control>

<p>
<b><ix:submit methodName="submitPerformed"></b>

<input type="submit" name="submit" value="Soumettre"></p>

</ix:submit>

</form>
```

Remarquez que vous êtes en train d'envelopper la balise HTML `input` de type soumission dans une balise InternetBeans `submit`. Cela permet au contrôle InternetBeans `IxSubmitButton` de comprendre implicitement quel bouton de soumission il remplace. Ce n'est pas fini pour le bouton de soumission. Vous devez encore ajouter la méthode qui sera exécutée lorsque l'utilisateur appuiera sur le bouton. Nous nous occuperons de cela dans la prochaine étape.

Etape 10 : Ajout de la méthode submitPerformed()

Ajoutez le code montré ci-après en **gras**.

```
<ix:submit methodName="submitPerformed">

<%!
    public void submitPerformed(PageContext pageContext){
        DataSet signatures = (DataSet) pageContext.findAttribute( "signatures" );
        signatures.post();
        signatures.saveChanges();

    }
%>

<input type="submit" name="submit" value="Soumettre"></p>

</ix:submit>
```

La méthode `submitPerformed()` est contenue dans une balise de déclaration JSP. La déclaration de la méthode n'a pas besoin d'être imbriquée dans la balise InternetBeans `submit` mais c'est une façon élégante d'écrire le code. Le paramètre transmis à la méthode est le `PageContext`. C'est un objet contenant des informations sur la page et qui existe pour toutes les JSP. La méthode retrouve le `DataSet DataExpress` en trouvant l'attribut de page

correspondant à l'ensemble de données "signatures". Il poste ensuite la saisie de l'utilisateur pour l'ensemble de données et enregistre ces modifications dans l'ensemble de données.

Etape 11 : Ajout de code pour insérer une ligne

Il y a encore quelques fragments de code à ajouter avant que la JSP fonctionne correctement. Lorsque la fiche est postée, nous devons ajouter une ligne vide à l'ensemble de données pour recueillir la saisie de l'utilisateur. Ajoutez le code montré ci-après en **gras**.

```
</ix:table>

<%
    signatures.insertRow(false);
%>

<form method="POST">
```

Ce dernier fragment de code Java risque de paraître confus car il ne semble pas être inclus dans une déclaration de méthode. Il l'est en réalité. Lorsque la JSP est compilée, il est intégré à la méthode `service()` dans le servlet généré (que vous ne pouvez pas voir, mais qui est là). Toute ligne de code à l'intérieur d'une balise scriptlet d'une JSP devient partie intégrante de la méthode `service()`.

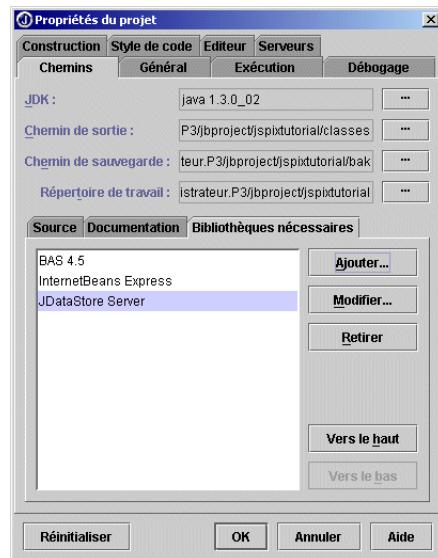
Ce fragment de code insère une ligne dans l'ensemble de données, juste avant l'affichage de la fiche. La fiche montre des valeurs vides. Puis, lorsque la fiche est postée, les données sont écrites dans la ligne vierge avant l'appel de la méthode `submitPerformed()`.

Etape 12 : Ajout de la bibliothèque JDataStore Server à votre projet

Ce projet nécessite la bibliothèque JDataStore Server. Pour ajouter cette bibliothèque aux Propriétés du projet:

- 1** Sélectionnez Propriétés du projet dans le menu Projet.
- 2** Cliquez sur l'onglet Chemins.
- 3** Cliquez sur l'onglet Bibliothèques nécessaires.
- 4** Cliquez sur Ajouter.
- 5** Sélectionnez JDataStore Server.
- 6** Cliquez sur OK.
- 7** Cliquez à nouveau sur OK pour fermer la boîte de dialogue Propriétés du projet.

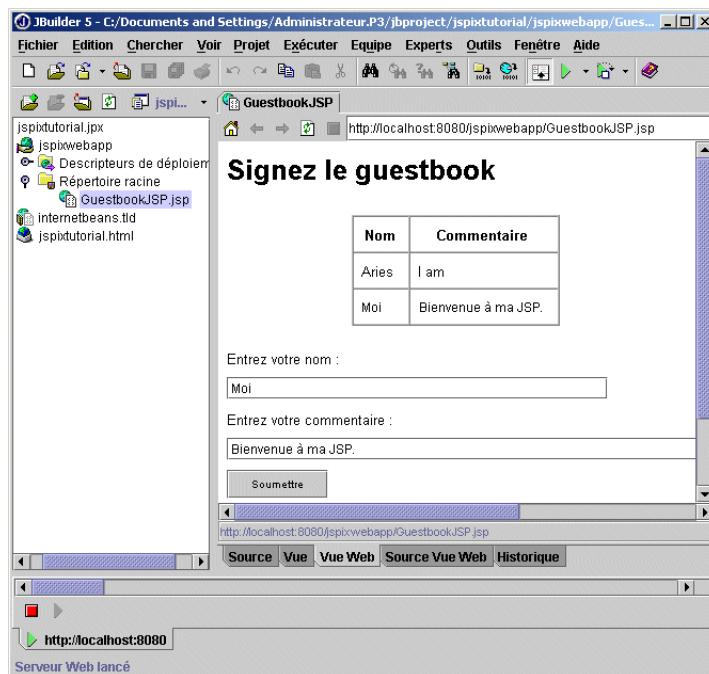
Figure 13.4 Onglet Bibliothèques nécessaires des Propriétés du projet



Etape 13 : Exécution de la JSP

Il est temps d'exécuter et de tester la JSP.

- 1 Développez le nœud Répertoire racine de la WebApp dans le volet projet.
- 2 Cliquez avec le bouton droit de la souris sur le fichier `GuestbookJSP.jsp` dans le volet projet.
- 3 Sélectionnez Exécution Web dans le menu contextuel.
Tomcat est démarré et la JSP s'exécute dans l'EDI de JBuilder.
- 4 Entrez votre nom et votre commentaire.
- 5 Cliquez sur le bouton Soumettre.
Votre nom et votre commentaire sont ajoutés à la table (et enregistrés dans le JDataStore).

Figure 13.5 JSP s'exécutant dans la Vue Web

Déploiement de la JSP

Les JSP sont plus faciles à déployer que les servlets. C'est que les serveurs web les trouvent comme ils trouvent les fichiers HTML. Vous n'avez aucune installation spéciale à effectuer, c'est le serveur web qui sait que faire de votre JSP. Pour plus d'informations sur le déploiement de votre JSP, voir Chapitre 16, "Déploiement de votre application web".

14

Configuration de votre serveur web

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Les servlets Java et les pages JSP (JavaServer Pages) s'exécutent tous deux dans des serveurs web. Tomcat, l'implémentation de référence des pages JSP et des servlets Java, est inclus dans le coffret JBuilder. Bien qu'il puisse différer de votre serveur web de production, Tomcat vous permet de développer et de tester vos servlets et vos JSP dans l'environnement de développement de JBuilder.

JBuilder fournit des plugins pour les serveurs web suivants :

- Tomcat 3.1
- Tomcat 3.2
- Tomcat 4.0
- Borland Application Server 4.5 (Entreprise seulement)
- WebLogic 5.1 (Entreprise seulement)
- WebLogic 6.0 (Entreprise seulement)

De tous ces serveurs web, seul Tomcat 3.2 est livré avec JBuilder. Une fois que vous avez installé un serveur web tiers, le plugin vous permet de configurer ce serveur web pour l'utiliser avec JBuilder.

Une fois votre application web et votre serveur web configurés, vous pouvez compiler, exécuter et déboguer votre servlet et votre JSP. Pour plus d'informations, voir Chapitre 15, "Utilisation des applications web dans JBuilder".

Configuration de Tomcat

Quand vous installez JBuilder Professionnel ou JBuilder Entreprise, Tomcat est installé automatiquement dans le répertoire de JBuilder. Les chemins et les bibliothèques sont configurés pour vous automatiquement.

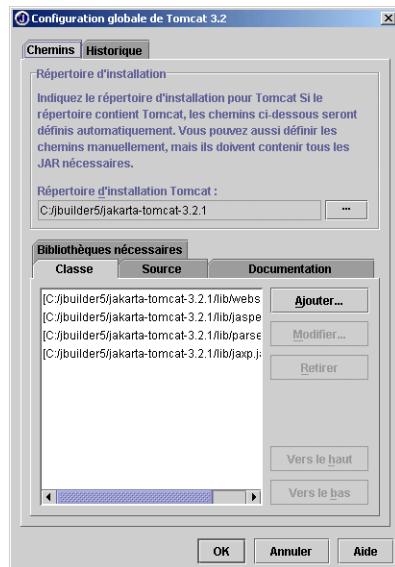
Si vous voulez utiliser Tomcat tel qu'il est fourni, vous n'avez pas besoin de modifier les options de configuration. Mais, si vous voulez examiner, et éventuellement modifier ces options, suivez ces étapes :

Remarque Ces étapes concernent la configuration de Tomcat 3.1, 3.2 ou 4.0.

- 1 Choisissez Projet | Propriétés du projet.** Dans la boîte de dialogue Propriétés du projet, choisissez l'onglet Serveurs (JBuilder Entreprise) ou l'onglet Serveur web (JBuilder Professionnel).

Remarque Les utilisateurs de l'édition Entreprise peuvent changer le serveur d'applications sélectionné en cliquant sur le bouton Points de suspension, à droite du champ AppServer. Cela peut affecter vos paramètres de serveur web. Pour plus d'informations, voir "Configuration de JBuilder pour des serveurs web autres que Tomcat" page 14-5.

- 2** Choisissez le bouton Configurer situé à droite du champ Serveur.
La boîte de dialogue Configuration globale de Tomcat s'affiche.
 - 3** Pour définir les options d'installation, choisissez la page Chemins. Pour Tomcat 3.2, la page Chemins se présente ainsi :



Remarque

Si vous utilisez la version de Tomcat installée avec JBuilder, ne changez pas ces options. Si vous voulez changer ces options, suivez ces étapes :

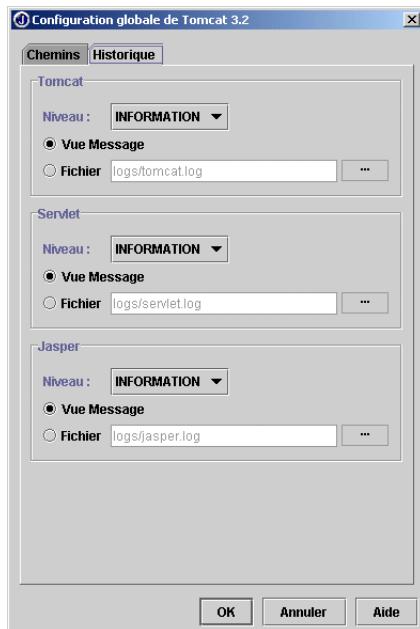
- 1** Entrez le nom du répertoire où Tomcat est installé, dans le champ Répertoire d'installation. Pour parcourir les répertoires à la recherche de cet emplacement, choisissez le bouton Points de suspension.
- 2** Dans la partie inférieure de la boîte de dialogue, choisissez l'emplacement des fichiers classe, des fichiers source, des fichiers de la documentation et des bibliothèques nécessaires de Tomcat :

Tableau 14.1 Options de configuration de Tomcat

Onglet	Description
Classe	L'emplacement des fichiers classe de Tomcat. Vous devez inclure les fichiers jar suivants, sinon les commandes Exécution Web/Débogage Web échoueraient : webserver.jar, jasper.jar, parser.jar et jaxp.jar. Dans une installation standard, ces fichiers se trouvent dans le répertoire /lib du dossier Tomcat de JBuilder.
Source	L'emplacement des fichiers source de Tomcat. Dans une installation standard, ces fichiers se trouvent dans le répertoire /src du dossier Tomcat de JBuilder.
Documentation	L'emplacement des fichiers de documentation de Tomcat. Dans une installation standard, cet onglet reste vide.
Bibliothèques nécessaires	Les bibliothèques que requiert Tomcat. Dans une installation standard, il s'agit de la bibliothèque Servlet.

Utilisez les boutons Ajouter, Modifier et Retirer pour mettre à jour les entrées de la boîte de dialogue.

- 4 Pour définir les options du fichier historique, choisissez la page Historique de la boîte de dialogue Configuration globale de Tomcat. Pour Tomcat 3.2, la page Historique se présente ainsi :



Cette page contient trois zones : Pour définir les options du fichier historique pour le moteur du serveur principal, utilisez la zone Tomcat de la boîte de dialogue. Pour définir les options du conteneur de servlets, utilisez la zone Servlet. Pour définir les options du fichier historique pour le conteneur JSP, utilisez la zone Jasper.

Tableau 14.2 Options du fichier historique Tomcat

Champ	Description
Niveau	Le niveau du message d'informations qu'affichera le serveur web. Choisissez d'afficher les erreurs fatales, les messages d'erreur, les messages d'avertissement, les messages du débogueur ou les messages d'informations.
Vue Message	Affiche les messages dans le volet message
Fichier	Le nom du fichier historique. Les fichiers sont écrits dans le répertoire <code>log</code> . Le chemin est relatif par rapport au répertoire du projet.

Remarque Si vous voulez davantage d'informations sur Tomcat ou si vous voulez l'exécuter de façon autonome, reportez-vous au répertoire `/doc` de l'installation de Tomcat dans JBuilder.

Configuration de JBuilder pour des serveurs web autres que Tomcat

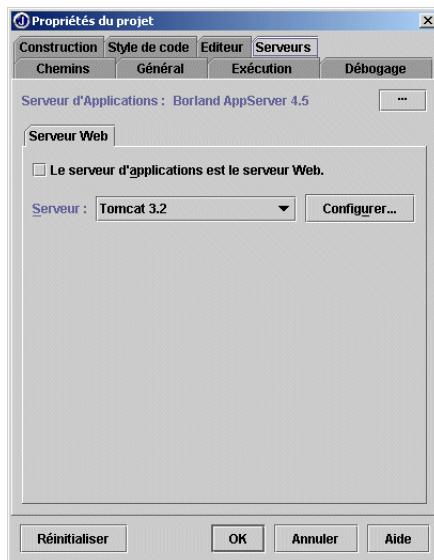
La boîte de dialogue Propriétés du projet vous permet également de choisir d'utiliser avec JBuilder un autre serveur web que Tomcat. Les étapes pour les utilisateurs de JBuilder Professionnel et de JBuilder Entreprise diffèrent légèrement. Suivez les instructions de la section appropriée.

- Si vous voulez que la sélection de serveur web s'applique uniquement au projet en cours, choisissez Projet | Propriétés du projet, afin d'afficher la boîte de dialogue Propriétés du projet.
- Si vous voulez que la sélection de serveur web s'applique à tous les projets, choisissez Projet | Propriétés du projet par défaut, afin d'afficher la boîte de dialogue Propriétés du projet par défaut.

Configuration de JBuilder pour des serveurs web autres que Tomcat (utilisateurs de JBuilder Entreprise)

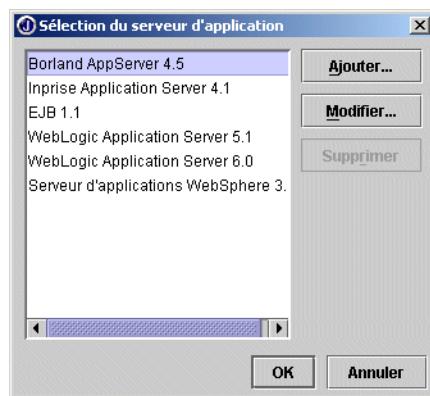
Quand la boîte de dialogue Propriétés du projet est ouverte, suivez ces étapes :

- 1 Choisissez l'onglet Serveurs. La page se présente ainsi :



- 2 Si le serveur d'applications est aussi un serveur web, choisissez le serveur d'applications à utiliser en cliquant sur le bouton Points de

suspension, à droite du champ Serveur d'applications. La boîte de dialogue Sélection du serveur d'application s'affiche :



Sélectionnez le serveur d'applications que vous voulez et cliquez sur OK pour fermer la boîte de dialogue. Pour plus d'informations sur la configuration des serveurs d'applications, voir "Configuration du serveur d'applications cible" dans la Partie II, "Guide du développeur Enterprise JavaBeans", du *Guide du développeur d'applications entreprise*.

- 3 Si le serveur d'applications sélectionné est également un serveur web, sélectionnez l'option Le serveur d'applications est le serveur Web. Aucune autre option de configuration n'est requise.

Remarque

Cette option est activée uniquement si le plugin du serveur d'applications a recensé un plugin de serveur web.

- 4 Pour sélectionner un serveur web, choisissez-le dans la liste déroulante Serveur. Tomcat est le serveur web par défaut. Pour les serveurs web autres que Tomcat, choisissez le bouton Configurer afin d'afficher l'interface utilisateur de configuration spécifique au serveur.

JBuilder fournit des plugins pour les serveurs web suivants :

- Tomcat 3.1
- Tomcat 3.2
- Tomcat 4.0
- Borland Application Server 4.5 (utilise Tomcat 3.2 comme serveur web)
- WebLogic 5.1 (est dans son propre serveur web)
- WebLogic 6.0 (est dans son propre serveur web)

Si vous achetez un serveur web tiers qui ne figure pas dans la liste, contactez votre fournisseur pour obtenir le plugin. Vous pouvez aussi écrire un plugin en utilisant l'API OpenTools. Pour plus d'informations, voir "Création du plugin de votre serveur web", page 14-10.

Si un choix de la liste est en rouge, il est disponible mais pas dans le chemin des classes. Cliquez sur le bouton Configurer pour configurer le serveur web.

Configuration de JBuilder pour des serveurs web autres que Tomcat (utilisateurs de JBuilder Professionnel)

Quand la boîte de dialogue Propriétés du projet est ouverte, suivez ces étapes :

- 1 Choisissez l'onglet Serveur web. La page se présente ainsi :



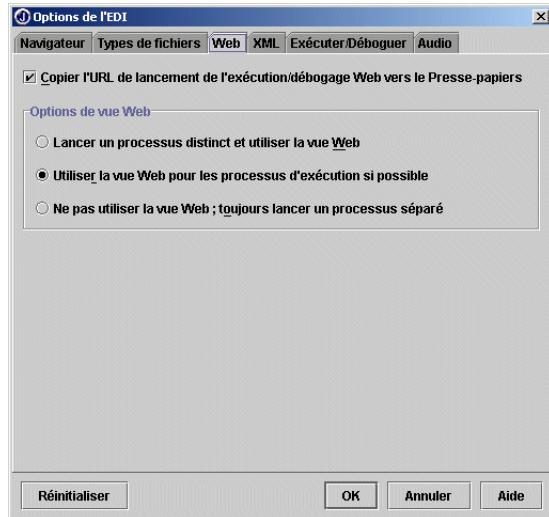
- 2 Dans la liste déroulante Serveur, choisissez le serveur web à utiliser. Par défaut, il s'agit de Tomcat.
- 3 Choisissez le bouton Configurer pour afficher la boîte de dialogue Configuration globale de Tomcat. (Pour plus d'informations, voir "Configuration de Tomcat", page 14-2.) Pour les serveurs web autres que Tomcat, le bouton Configurer affiche l'interface utilisateur de configuration spécifique au serveur.

Configuration du serveur web sélectionné

Une fois que vous avez configuré JBuilder avec un serveur web, vous pouvez configurer des options pour ce serveur web, notamment les options de visualisation, le nom de l'ordinateur hôte du serveur web, son numéro de port et la façon de le lancer.

Définition des options de vue web

Pour configurer l'affichage de la vue web et choisir comment le serveur web est lancé, choisissez l'onglet Web dans la boîte de dialogue Options de l'EDI (Outils | Options de l'EDI). La page Web se présente ainsi :



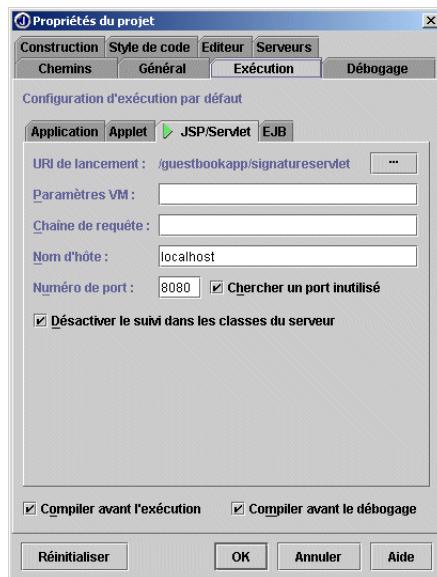
- 1** Choisissez l'option Copier l'URL de lancement de l'exécution / débogage Web vers le Presse-papiers afin de copier dans le presse-papiers l'URL utilisée pour lancer l'application web. Cela vous permettra d'aller facilement à la même URL dans un navigateur externe. Définissez cette option si vous créez une application ou une applet Java Web Start.
- 2** Choisissez Options de vue Web dans la deuxième partie de la page. Ces options fonctionnent conjointement avec l'option Chercher un port inutilisé de la page Exécution JSP/Servlet (boîte de dialogue Propriétés du projet) quand le port spécifié est utilisé par un processus non web. (Voir "Définition des options d'exécution web", page 14-9, pour plus d'informations.)
 - Choisissez l'option Lancer un processus distinct et utiliser la vue Web, pour utiliser à la fois le navigateur web interne et un navigateur web externe. Cette option affiche automatiquement votre servlet ou JSP dans la page Vue Web du volet contenu et dans le navigateur web externe.
 - Choisissez l'option Utiliser la vue Web pour les processus d'exécution si possible, pour voir votre page web dans le navigateur web interne. Cette option affiche automatiquement votre servlet ou JSP dans la page Vue Web du volet contenu. Si un serveur web est

déjà exécuté, JBuilder utilisera le même processus sur le port existant. Il s'agit de l'option par défaut.

- Choisissez l'option Ne pas utiliser la vue Web ; toujours lancer un processus séparé, quand vous exécutez votre application web dans un navigateur web externe.

Définition des options d'exécution web

Pour définir les options d'exécution web, choisissez l'onglet JSP/Servlet dans la page Exécution de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet). La page Exécution JSP/Servlet se présente ainsi :



- 1 Entrez le nom que doit porter le serveur web, dans le champ Nom d'hôte. Ne choisissez pas un nom déjà utilisé dans votre sous-réseau. `localhost` est le nom par défaut.
- 2 Entrez le numéro du port que doit écouter le serveur web, dans le champ Numéro de port. Utilisez le numéro de port par défaut, `8080`. Changez cette valeur uniquement si la valeur par défaut est déjà utilisée.
- 3 Choisissez l'option Chercher un port inutilisé, pour dire à JBuilder de choisir un autre port si celui qui est spécifié est déjà utilisé. (Le port est recherché uniquement la première fois qu'une exécution web est demandée.) Il est utile de sélectionner cette option quand vous exécutez plusieurs servlets ou JSP, sinon, vous risquez d'obtenir un message signalant que le port est occupé. Il est également utile de sélectionner

cette option au cas où un problème utilisateur ferait tomber le serveur web. Si cette option est sélectionnée, vous êtes protégé lorsque le serveur ne s'arrête pas correctement. Cette option fonctionne conjointement avec les options de lancement dans la page Options de l'EDI, quand le port spécifié est utilisé par un processus non web. (Voir "Définition des options de vue web", page 14-8, pour plus d'informations.)

- 4 Choisissez l'option Désactiver le suivi dans les classes du serveur, pour empêcher le suivi dans les classes côté serveur. Pour plus d'informations, voir "Contrôle des classes à exécuter pas à pas" au chapitre "Débogage des programmes Java", dans *Construction d'applications avec JBuilder*.

Remarque D'autres options de cette page - URI de lancement, Paramètres VM et Chaîne de requête - s'appliquent à la JSP ou au servlet exécutable. Pour plus d'informations, voir "Définition des paramètres d'exécution de votre servlet ou de votre JSP", page 15-9.

Création du plugin de votre serveur web

Par défaut, JBuilder fournit Tomcat pour exécuter des JSP et des servlets. D'autres serveurs web peuvent être configurés pour fonctionner avec JBuilder. Ce support peut être apporté par le fournisseur du serveur web, un tiers, ou bien vous pouvez écrire le vôtre par le biais de l'API OpenTools de JBuilder.

Un plugin de serveur web OpenTool doit effectuer les tâches suivantes :

- Recensement d'un OpenTool
- Configuration du serveur web
- Démarrage et arrêt du serveur web

Cette section fournit une introduction de haut niveau à l'API OpenTools du plugin de serveur web. Pour plus de détails, voir la documentation `OpenTool servlet` et `jsp`.

Pour commencer avec les OpenTools de JBuilder, consultez les documents suivants :

- "Notions fondamentales sur les OpenTools de JBuilder" dans *Développement OpenTools*
- "Présentation des OpenTools de JBuilder" dans *Développement OpenTools*

Pour avoir un exemple de plugin de serveur web, ouvrez le projet Tomcat dans le répertoire `samples/OpenToolsAPI/web/tomcat` de votre installation de JBuilder.

Recensement d'un OpenTool

La classe de configuration de votre plugin serveur web doit identifier une implémentation `ServerSetup` en tant qu'`OpenTool`, en utilisant la méthode `initOpenTool()`. Dans la méthode `initOpenTool()`, elle doit recenser une instance d'elle-même avec le `ServerManager`. Par exemple, dans `Tomcat32Setup.java` (le fichier source de l'exemple qui configure Tomcat 3.2 pour JBuilder), le code de recensement ressemble à ceci :

```
public static void initOpenTool( byte majorVersion, byte minorVersion ) {
    ServerManager.registerServer( SERVER_NAME, new Tomcat32Setup() );
}
```

Configuration du serveur web

La classe qui implémente le plugin du serveur web doit implémenter l'interface `ServerSetup`. L'interface fournit et vérifie la configuration du serveur web et agit en tant que factory des objets `ServerStarter` correspondants. L'implémentation doit vérifier que :

- Chaque fois que les commandes Exécution Web et Débogage Web sont appelées, le serveur est configuré.
- L'API Servlet, le serveur web, le conteneur JSP et le support XML sont dans le chemin d'accès

La classe `ServerSetup` fournit diverses méthodes de configuration et de vérification. Dans l'exemple, l'essentiel de la classe `Tomcat32Setup.java` fournit le code qui configure le serveur web.

Démarrage et arrêt du serveur web

Votre plugin de serveur web doit fournir le code qui démarre et arrête le serveur. L'interface `ServerStarter` maintient et renvoie les informations spécifiques à l'exécution pour une exécution particulière d'un serveur web/moteur de servlet. Chaque objet `ServerStarter` est associé à un objet `ServerSetup` et peut lui déléguer des éléments qui ne changent pas. Le code qui démarre le serveur web doit :

- Préparer le serveur pour l'exécution, en réalisant une configuration spécifique au serveur pour une exécution particulière.
- Renvoyer la classe principale, d'éventuels arguments, d'éventuels paramètres de la VM et le répertoire de travail de la VM.
- Arrêter le serveur et effectuer tout nettoyage nécessaire, comme la suppression des fichiers de configuration.

Dans l'exemple, la classe `Tomcat32Starter.java` fournit le code qui démarre et arrête le serveur web.

Considérations sur JSP

Il y a de nombreux points à prendre en compte lorsque vous créez un plugin. On peut citer notamment la compilation des JSP en servlets, le débogage des JSP et la déclaration d'une bibliothèque de balises JSP. Pour plus d'informations, voir [JSP package](#) dans la référence de l'API OpenTools.

Editeur GUI de descripteur de déploiement

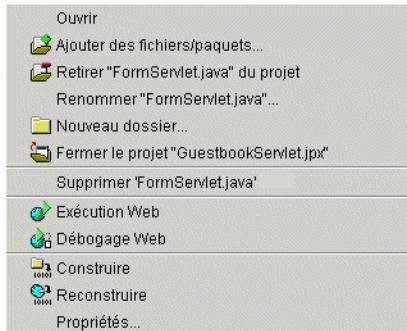
Il est intéressant, quoique non obligatoire, qu'un plugin fournit un éditeur GUI pour tout descripteur de déploiement spécifique au serveur. Pour plus d'informations, voir "Modification des descripteurs de déploiement spécifiques au fournisseur", page 16-19.

15

Utilisation des applications web dans JBuilder

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

JBuilder proposent deux commandes dans le menu contextuel du volet projet, Exécution Web et Débogage Web, pour faciliter l'exécution et le débogage des servlets et des JSP. Exécution Web exécute votre application web en utilisant le serveur web sélectionné. Débogage Web débogue votre JSP ou votre servlet tout en l'exécutant dans la fenêtre du serveur web, ce qui vous permet de parcourir pas à pas et d'examiner facilement votre code.



La sélection de Exécution Web ou de Débogage Web va exécuter ou déboguer le servlet ou la JSP dans son contexte WebApp. Si le fichier JSP, HTML ou SHTML n'est pas dans la WebApp, il ne peut pas être exécuté ni débogué pour le web, et les commandes Exécution Web et Débogage Web ne s'affichent pas dans le menu contextuel. Pour plus d'informations sur les WebApps, voir "La WebApp et ses propriétés", page 3-4.

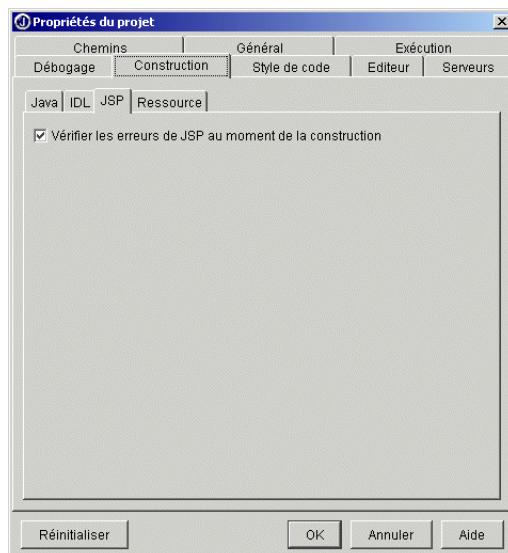
Quand vous créez un servlet ou une JSP en utilisant l'expert servlet ou l'expert JSP de JBuilder, les commandes Exécution Web et Débogage Web sont activées automatiquement.

Compilation de votre servlet ou de votre JSP

Comme tout programme Java, avant de l'exécuter, il faut le compiler. Vous pouvez compiler le projet entier à l'aide des commandes Projet | Construire ou Projet | Reconstruire. Vous pouvez aussi compiler le fichier servlet ou JSP seul, en cliquant avec le bouton droit sur le fichier dans le volet projet et en choisissant Construire ou Reconstruire. Les erreurs de compilation s'affichent dans la page Compilateur du volet message.

Pour plus d'informations sur la compilation, voir le chapitre "Compilation des programmes Java" dans le manuel en ligne *Construction d'applications avec JBuilder*.

Les JSP sont une extension de l'API Servlet et sont compilées en servlets avant d'être utilisées. Cela exige du processus de compilation la conversion des noms des fichiers JSP et des numéros des lignes en leurs équivalents Java. Depuis JBuilder 5, les JSP peuvent être compilées au moment de la construction. Pour que cette fonctionnalité s'applique à toutes les JSP de votre projet, choisissez l'option Vérifier les erreurs de JSP au moment de la construction, dans l'onglet JSP de la page Construction de la boîte de dialogue Propriétés du projet (Projet | Propriétés du projet).



Vous pouvez définir cette propriété pour chaque JSP de votre projet et donc exclure certains fichiers de la compilation. Par exemple, les JSP

prévues pour être incluses dans d'autres JSP ne se compileront probablement pas seules et vous devrez exclure ces fichiers.

Comment les URL exécutent les servlets

Une URL (Uniform Resource Locator) est utilisée pour exécuter un servlet. Un URI (Uniform Resource Identifier) est un concept plus général qui inclut à la fois les URL et les *request-URI-path*. Le request-URI-path suit le nom du serveur et le numéro de port facultatif. Il commence par une barre oblique.

Par exemple, dans l'URL :

`http://localhost:8080/nomfichier.html`

`/nomfichier.html` est le request-URI-path.

Dans un serveur web non conteneur de servlet, comme IIS ou Apache sans Tomcat, la gestion de base du request-URI-path est simple. Le contenu web est "enraciné" dans un répertoire particulier, de sorte que le serveur web peut résoudre ce chemin, en utilisant la barre oblique de tête pour indiquer le répertoire racine-web. Il peut alors renvoyer le fichier correspondant, s'il y est.

Les conteneurs de servlet, comme Tomcat ou WebLogic, sont plus complexes, mais moins souples. Ces conteneurs autorisent les contextes et les mappages, et vous pouvez avoir un nombre quelconque de contextes nommés pour votre application. Chaque contexte est mappé avec son propre répertoire.

Le premier travail du conteneur du servlet concernant l'évaluation du request-URI-path est de voir si la première partie du chemin correspond à un nom de contexte. Dans JBuilder, ce sont les noms des WebApps. Au cours d'une Exécution Web, ces noms sont associés aux répertoires racine des WebApps. (Pour plus d'informations sur les WebApps, voir "La WebApp", page 3-1.)

S'il y a correspondance, la première partie du request-URI-path devient le chemin contextuel. La partie restante du chemin, commençant par une barre oblique, devient le *URL-path-to-map*. S'il n'y a pas de correspondance, le chemin contextuel est une chaîne vide et le request-URI-path entier est considéré comme étant le URL-path-to-map.

Par exemple, pour un projet ayant une seule WebApp nommée `myprojectwebapp`, le request-URI-path `/myprojectwebapp/sub/somename.jsp` sera évalué comme suit :

- Le chemin contextuel sera `/myprojectwebapp`.
- Le URL-path-to-map sera `/sub/somename.jsp`.

Mais, pour le request-URI-path `/test/subtest/somename.jsp`, l'évaluation ne trouvera pas de chemin contextuel, puisque la seule WebApp existante est `myprojectwebapp`. Dans ce cas, le chemin contextuel sera vide et le URL-path-to-map sera l'URI entier : `/test/subtest/somename.jsp`

Remarquez que la configuration contextuelle est faite d'une manière spécifique au serveur. Cependant, la correspondance de l'URL-path-to-map est faite via les entrées de mappage des servlets existant dans le descripteur standard de déploiement de la WebApp de chaque contexte, le fichier `web.xml`. Chaque servlet-mapping est constitué de deux parties : une partie url-pattern et une partie servlet-name.

Il y a trois types spéciaux de motifs d'URL :

Tableau 15.1 Motifs d'URL

Type de motif	Description
Mappage de chemin	Commence par / et se termine par /*
Mappage d'extension	Commence par *.
Mappage par défaut	Contient uniquement /

Remarque Les trois arborescences en bas de la boîte de dialogue URI de lancement correspondent grossièrement aux trois différentes sortes de mappages. Pour plus de détails sur la façon dont fonctionnent ces mappages dans la boîte de dialogue URI de lancement, voir "Définition des paramètres d'exécution de votre servlet ou de votre JSP", page 15-9.

Toutes les autres chaînes url-pattern sont utilisées uniquement pour des correspondances exactes. Lors de la recherche de URL-path-to-map, une correspondance exacte est essayée en premier. Par exemple, si la WebApp `somewebapp` contient un url-pattern `/test/jspname.jsp`, le servlet correspondant sera utilisé.

S'il n'y a pas de correspondance exacte, une correspondance de chemin sera essayée, en commençant par le chemin le plus long. Dans le contexte par défaut, le url-pattern `/test/jspname.jsp/*` serait la première correspondance.

S'il n'y a pas de correspondance de chemin, alors une correspondance d'extension est essayée. Le url-pattern `*.jsp` correspondrait à la fois aux deux request-URI-paths suivants : `/testwebapp/subtest/jspname.jsp` et `/myprojectwebapp/anyfolder/myjsp.jsp`.

Enfin, s'il n'y a aucune correspondance d'extension, le servlet correspondant à / (le servlet par défaut) est utilisé.

La plupart des serveurs web ont déjà des mappages par défaut, encore une fois effectués de façon spécifique aux serveurs. Par exemple, `*.jsp` serait mappé à un servlet en :

- Prenant le chemin qui correspondait (par exemple, `/sub/somename.jsp` ou `/test/subtest/somename.jsp`)

- Trouvant le fichier correspondant relatif au répertoire racine-web du contexte
- Le convertissant en un servlet si ce n'est déjà fait
- Exécutant ce servlet

Un autre mappage par défaut classique est `/servlet/*`, qui est un *servlet invocateur*. Un servlet invocateur :

- Prend tout ce qui suit `/servlet/` comme un nom de classe
- Essaie d'exécuter cette classe en tant que servlet

Le servlet par défaut (celui qui est mappé à l'url-pattern `/`) :

- Prend le URL-path-to-map (qui n'est mappé à rien)
- Trouve le fichier correspondant relatif au répertoire racine-web du contexte
- L'envoie au navigateur.

Quand vous créez un servlet standard dans l'expert servlet, il fait le minimum nécessaire pour créer un mappage. Par exemple, avec le nom `myproject.MyServlet`, l'expert servlet :

- Crée un servlet avec le servlet-name `myservlet` associé à la servlet-class `myproject.MyServlet`
- Crée le url-pattern `/myservlet` et le mappe au servlet-name `myservlet`

Si cela était fait pour la WebApp le test, alors `/test/myservlet` devrait exécuter la classe servlet `myproject.MyServlet` dans le contexte WebApp `test`. Pour plus d'informations sur la façon dont l'expert servlet crée un mappage voir "Expert servlet – Page des options de nom et d'affectation", page 6-6.

Quand vous employez Exécution Web pour exécuter le fichier `.java` (ou `.class`) d'un servlet, le servlet invocateur est utilisé, sous la WebApp spécifiée ; par exemple, `/test/servlet/myproject.MyServlet`.

Exécution de votre servlet ou de votre JSP

Pour exécuter votre servlet ou votre JSP dans JBuilder, cliquez avec le bouton droit sur le fichier servlet ou JSP dans le volet projet et choisissez Exécution Web. La commande Exécution Web exécute votre programme sur le serveur web sélectionné, sans le déboguer.

Remarque Si votre servlet s'exécute depuis un fichier HTML ou SHTML, cliquez avec le bouton droit sur ce fichier et choisissez Exécution Web.

Remarquez que vous pouvez aussi créer une configuration d'exécution pour exécuter votre servlet ou votre JSP. C'est plus souple et plus

permanent que l'utilisation de la commande Exécution Web à chaque exécution.

Pour créer une configuration d'exécution,

- 1 Choisissez Exécuter | Configurations. Dans la boîte de dialogue Configurations d'exécution, choisissez Nouveau pour afficher la boîte de dialogue Propriétés d'exécution.
- 2 Dans le champ Nom de configuration, entrez un nom pour votre configuration.
- 3 Choisissez l'onglet JSP/Servlet. Entrez les paramètres d'exécution de la configuration. Pour plus d'informations sur ces options, voir "Définition des paramètres d'exécution de votre servlet ou de votre JSP", page 15-9.
- 4 Cliquez sur OK deux fois pour fermer les boîtes de dialogue Propriétés d'exécution et Configurations d'exécution.
- 5 Pour exécuter la configuration, cliquez sur la flèche vers le bas à droite du bouton Exécuter, dans la barre d'outils principale. Choisissez dans la liste déroulante la configuration que vous voulez exécuter.

Pour plus d'informations sur les configurations d'exécution, voir "Définition de configurations d'exécution" dans le chapitre "Exécution des programmes Java" de *Construction d'applications avec JBuilder*.

Démarrage de votre serveur web

Quand vous choisissez Exécution Web, JBuilder démarre votre serveur web, en utilisant :

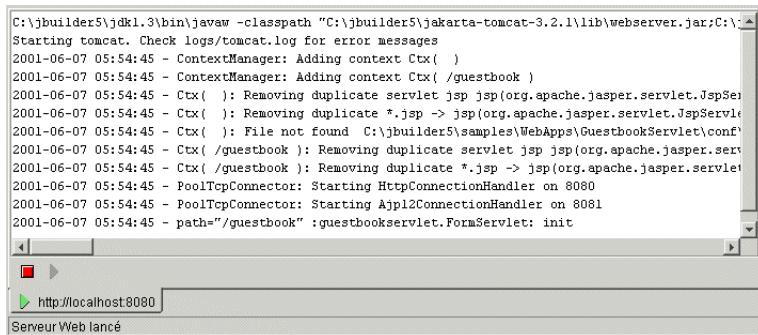
- Les paramètres définis dans les pages Exécution et Serveurs de la boîte de dialogue Propriétés du projet.
- Les propriétés définies dans la boîte de dialogue Propriétés du fichier Java.
- Les options définies dans la page Web de la boîte de dialogue Options de l'EDI (Outils | Options de l'EDI).

Les messages s'inscrivent dans l'onglet Serveur web, affiché dans le volet message en bas de l'AppBrowser. Les commandes HTTP et les valeurs des paramètres sont envoyées dans ce volet.

Par exemple, pour le serveur web Tomcat, les informations suivantes sont affichées :

- Chemin de la classe
- Répertoire de base
- Chemin de l'historique de contexte
- Message de démarrage de Tomcat

- Message de démarrage du servlet ou de la JSP
- Le numéro de port sur lequel s'exécute le serveur web.
- Le chemin du servlet ou de la JSP

Figure 15.1 Messages de démarrage de Tomcat


C:\jbuilder5\jdk1.3\bin>java -classpath "C:\jbuilder5\jakarta-tomcat-3.2.1\lib\webserver.jar;C:\jakarta-tomcat-3.2.1\lib*" &
Starting tomcat. Check logs/tomcat.log for error messages
2001-06-07 05:54:45 - ContextManager: Adding context Ctx()
2001-06-07 05:54:45 - ContextManager: Adding context Ctx(/guestbook)
2001-06-07 05:54:45 - Ctx(): Removing duplicate servlet jsp jsp(org.apache.jasper.servlet.JspServlet)
2001-06-07 05:54:45 - Ctx(): Removing duplicate *.jsp -> jsp(org.apache.jasper.servlet.JspServlet)
2001-06-07 05:54:45 - Ctx(): File not found C:\jbuilder5\samples\WebApps\GuestbookServlet\conf\index.jsp
2001-06-07 05:54:45 - Ctx(/guestbook): Removing duplicate servlet jsp jsp(org.apache.jasper.servlet.JspServlet)
2001-06-07 05:54:45 - Ctx(/guestbook): Removing duplicate *.jsp -> jsp(org.apache.jasper.servlet.JspServlet)
2001-06-07 05:54:45 - PoolTcpConnector: Starting HttpConnectionHandler on 8080
2001-06-07 05:54:45 - PoolTcpConnector: Starting Ajp12ConnectionHandler on 8081
2001-06-07 05:54:45 - path="/guestbook" :guestbookservlet.FormServlet: init

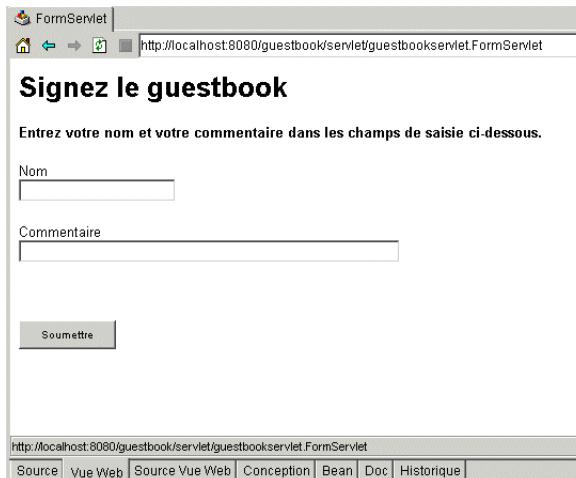
http://localhost:8080

Serveur Web lancé

Quand l'Exécution Web commence, deux nouveaux onglets apparaissent dans le volet contenu : Vue Web et Source Vue Web. Cliquez sur l'onglet pour ouvrir la vue web et le source de la vue web.

Vue Web

La sortie formatée s'affiche dans le volet Vue Web du volet contenu. L'URL générée s'affiche dans le champ d'emplacement, en haut de la Vue Web.

Figure 15.2 Sortie dans la Vue Web

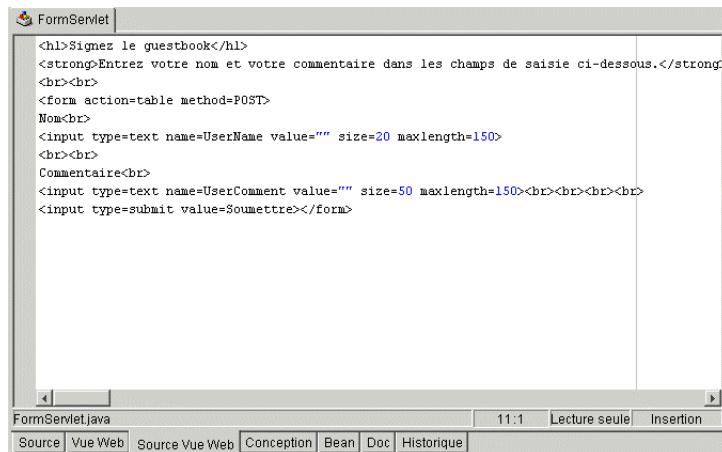
La Vue Web montre le servlet après son traitement par le moteur de servlet. Il peut y avoir un délai entre le moment où le fichier servlet ou JSP est édité et celui où les modifications apparaissent dans le volet Vue Web.

Pour voir les dernières modifications, sélectionnez le bouton Rafraîchir, en haut de la Vue Web.

Source Vue Web

La sortie brute du serveur web s'affiche dans le volet Source Vue Web.

Figure 15.3 Source Vue Web



```
<hl>Signez le guestbook</hl>
<strong>Entrez votre nom et votre commentaire dans les champs de saisie ci-dessous.</strong>
<br><br>
<form action=table method=POST>
Nom<br>
<input type=text name=UserName value="" size=20 maxlength=150>
<br><br>
Commentaire<br>
<input type=text name=UserComment value="" size=50 maxlength=150><br><br><br><br>
<input type=submit value=Soumettre></form>
```

FormServlet.java 11:1 Lecture seule Insertion

Source | Vue Web | Source Vue Web | **Conception** | Bean | Doc | Historique

Arrêt du serveur web

Pour arrêter le serveur web, cliquez sur le bouton Réinitialiser le programme  de l'onglet du serveur web. Pour redémarrer le serveur web et relancer votre application web, cliquez sur le bouton Exécuter le programme . En général, vous suivez ces étapes lorsque vous effectuez des modifications dans le code source, le recompilez et l'exécutez à nouveau. Vous n'avez pas besoin de fermer le volet du serveur web chaque fois que vous démarrez ce dernier.

Activation des commandes web

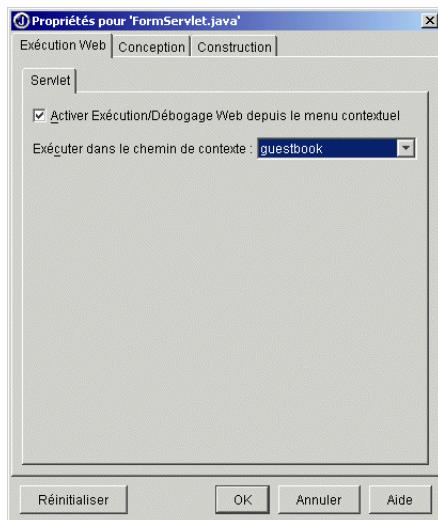
Si vous avez créé un servlet de toutes pièces, sans utiliser l'expert servlet, vous devez activer les commandes Exécution Web et Débogage Web. Si vous utilisez une JSP ou un servlet créé avec l'expert correspondant, vous n'avez pas besoin de suivre ces étapes. JBuilder a déjà activé ces commandes.

Pour activer les commandes web pour un servlet,

- 1 Cliquez avec le bouton droit de la souris sur le fichier servlet .java dans le volet projet.

- 2** Choisissez Propriétés pour afficher la boîte de dialogue Propriétés du fichier Java.
- 3** Choisissez l'onglet Servlet dans la page Exécution Web.
- 4** Choisissez l'option Activer Exécution/Débogage depuis le menu contextuel.

Cette option affiche les commandes Exécution Web et Débogage Web, lorsque vous cliquez avec le bouton droit sur le fichier servlet dans le volet projet. (Si vous créez votre fichier à l'aide de l'expert Servlet ou JSP, cette option est automatiquement activée.)



- 5** Cliquez sur OK pour fermer la boîte de dialogue.

Remarque

Comme les fichiers JSP ont l'extension .jsp, l'EDI peut déterminer que les commandes web doivent être activées. Mais, comme indiqué précédemment, la JSP doit faire partie d'une WebApp pour que ces commandes soient disponibles.

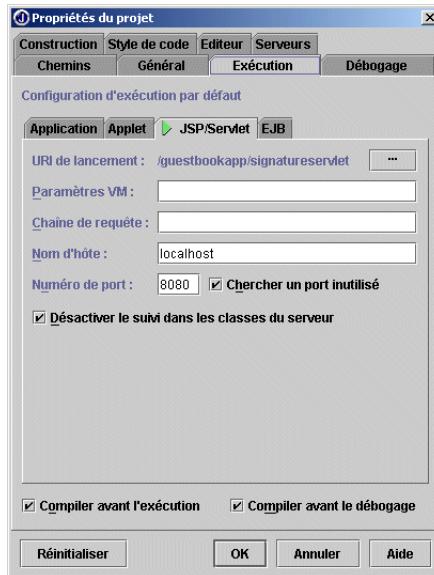
Définition des paramètres d'exécution de votre servlet ou de votre JSP

Les paramètres d'exécution indiquent à l'EDI quel fichier ou quelle classe exécuter après le démarrage du serveur web. Vous pouvez aussi entrer des paramètres à transmettre à la VM ainsi qu'une chaîne de requête à transmettre au client. Pour définir des paramètres d'exécution,

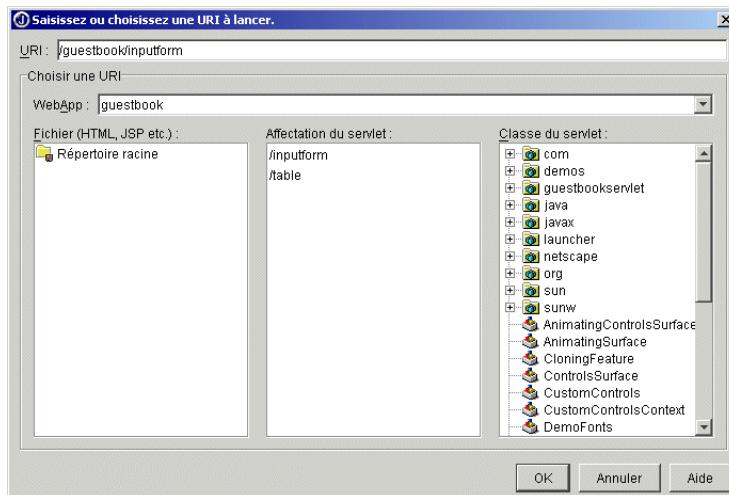
- 1** Choisissez Projet | Propriétés du projet pour afficher la boîte de dialogue Propriétés du projet.

Exécution de votre servlet ou de votre JSP

- 2 Sélectionnez l'onglet Exécution. Choisissez l'onglet JSP/Servlet. Notez que l'icône ➤ est affichée sur l'onglet, montrant le processus actif. La page Exécution se présente ainsi :



- 3 Pour choisir le servlet ou la JSP à lancer, cliquez sur le bouton Points de suspension, à droite du champ URI de lancement. La boîte de dialogue Saisissez ou choisissez une URI à lancer s'affiche.



Vous pouvez utiliser la boîte de dialogue Saisissez ou choisissez une URI à lancer pour lancer une partie spécifique d'une application web. Vous pouvez taper directement l'URI (Universal Resource Identifier) dans le champ texte URL, en haut de la boîte de dialogue, ou choisir la

WebApp et l'URI dans les arborescences, en bas de la boîte de dialogue :

- Choisissez la WebApp de l'URI que vous voulez exécuter, dans la liste déroulante WebApp. Cette liste montre toutes les WebApps définies dans votre projet.
- Les trois arborescences en bas de la boîte de dialogue Saisissez ou choisissez une URI à lancer correspondant grossièrement aux trois différentes sortes de mappages d'un servlet. L'arborescence Fichier, ou contenu web, à gauche, crée les URI qui correspondront probablement à des mappages d'extension (comme *.jsp) ou ne correspondront à rien et seront servis par le servlet par défaut. L'arborescence Affectation du servlet, au milieu, contient les motifs d'URL pour des correspondances exactes. L'arborescence Classe du servlet, à droite, crée les URI qui pourraient correspondre à l'invocateur du servlet.

Pour plus d'informations sur les mappages d'URL, voir "Comment les URL exécutent les servlets", page 15-3.

Tableau 15.2 Arborescences de la boîte de dialogue URI

Nom de l'arborescence	Description	Exemple d'URI résultant
Fichier (HTML, JSP, etc.)	Tous les fichiers de type HTML de la WebApp sélectionnée.	/selectedwebapp/hello.html ou /selectedwebapp/login.jsp
Affectation du servlet	Tous les motifs d'URL qui ne contiennent pas de caractères génériques. Permet à un utilisateur d'invoquer un servlet ou une JSP par son nom. Cette valeur a été entrée dans les champs Nom/Motif de l'URL, dans la page Expert servlet - Options de nom et d'affectation.	/selectedwebapp/form ou /selectedwebapp/table
Classe du servlet	Toutes les classes du projet ouvert ; mais, les classes affichées sont supposées être des servlets et sont exécutées en utilisant l'invocateur de servlet du serveur web.	/selectedwebapp/servlet/com.test.MyServlet

L'URI est ajouté au hostname:port au cours d'une exécution, par exemple :

```
http://localhost:8080/selectedwebapp/hello.html
http://localhost:8080/selectedwebapp/login.jsp
http://localhost:8080/selectedwebapp/form
http://localhost:8080/selectedwebapp/table
http://localhost:8080/selectedwebapp/servlet/com.test.MyServlet
```

- 4 Cliquez sur OK pour fermer la boîte de dialogue Saisissez ou choisissez une URI à lancer.
- 5 Dans le champ Paramètres VM, entrez d'éventuels paramètres à transmettre au compilateur de la machine virtuelle Java (VM). Pour plus d'informations sur le compilateur de la VM Java et sur les options que vous pouvez lui passer, voir "Basic Tools: java - The launcher for Java technology applications" à l'adresse <http://java.sun.com/products/jdk/1.2/docs/tooldocs/tools.html>.
- 6 Dans le champ Chaîne de requête, entrez d'éventuels paramètres utilisateur. Les paramètres de l'utilisateur sont constitués d'une série de couples nom/valeur séparés par un et commercial, par exemple, `a=1&b=2`. Vous pouvez aussi entrer une chaîne de requête si le client utilise la méthode `doGet()` pour lire des informations issues du servlet ou de la JSP. Cette chaîne est ajoutée à la fin de toute URL générée pour une exécution web, et contient en général des paramètres pour le servlet ou la JSP. Par exemple, votre application web peut contenir à la fois un servlet (`SalesHistory.java`, pour le nom de serveur `saleshistory`) et une JSP (`showproduct.jsp`) qui utilisent un numéro d'identificateur de produit. Quand vous lancez l'application web, la commande Exécution Web va générer les URL suivantes :

`http://localhost:8080/showproduct.jsp`
`http://localhost:8080/saleshistory`

Si vous spécifiez "product=1234" dans le champ Chaîne de requête, la requête sera ajoutée à la fin de l'URL :

`http://localhost:8080/showproduct.jsp?product=1234`
`http://localhost:8080/saleshistory?product=1234`

Le point d'interrogation (?) sépare le nom de la JSP ou du servlet à exécuter de la chaîne de requête. Cela permet à la JSP ou au servlet de demander le paramètre `product` et d'obtenir en retour `1234`. Notez que les valeurs de paramètres sont toujours des chaînes. Les paramètres multiples sont séparés par un et commercial (&), par exemple, `product=1234&customer=6789`.

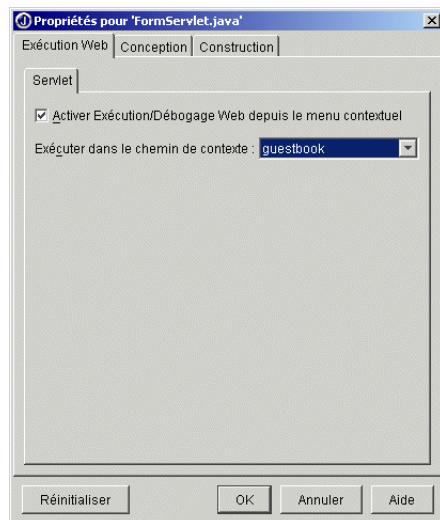
Remarque Les champs restants de cette page - Nom d'hôte, Numéro de port, Chercher d'un port inutilisé et Désactiver le suivi dans les classes du serveur - sont spécifiques à votre serveur web. Ces options sont expliquées dans "Définition des options d'exécution web", page 14-9.

Définition des propriétés d'exécution d'un servlet

Pour définir les propriétés d'exécution d'un servlet,

- 1 Cliquez avec le bouton droit sur le servlet qui va être exécuté et choisissez Propriétés. La boîte de dialogue Propriétés du fichier Java s'affiche.

- 2** Choisissez l'onglet Servlet dans la page Exécution Web. La page Servlet s'affiche.



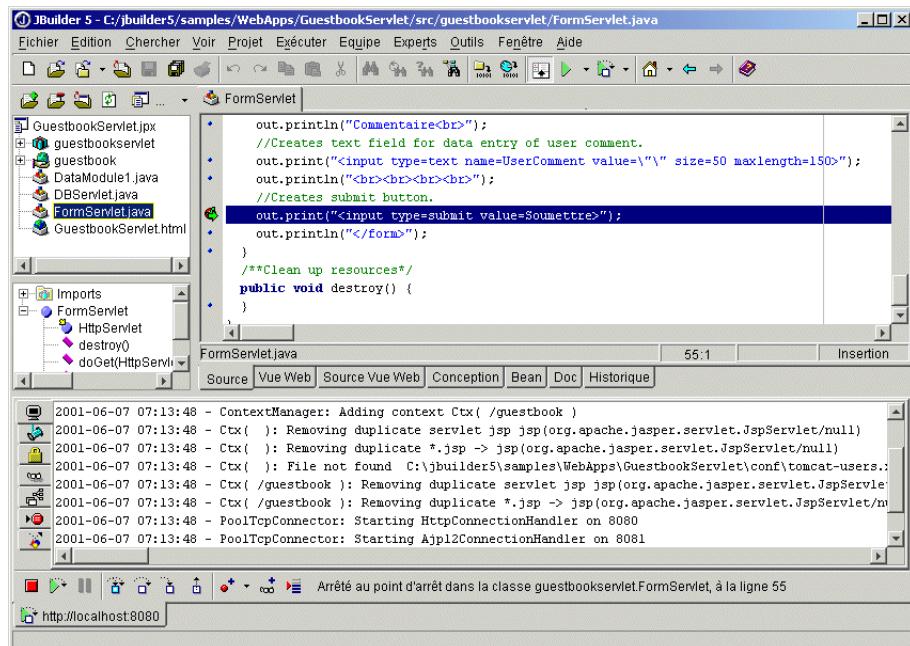
- 3** Assurez-vous que l'option Activer Exécution/Débogage depuis le menu contextuel est activée. Cela vous permet d'exécuter votre servlet ou votre JSP dans un serveur web dans JBuilder. Notez que vous pouvez toujours l'exécuter depuis la boîte de dialogue URI de lancement.
- 4** Choisissez la WebApp sous laquelle l'exécuter dans la liste déroulante Exécuter dans le chemin de contexte. Ce chemin est utilisé quand vous sélectionnez la commande Exécution Web. Il est défini dans l'expert Application Web.
- 5** Cliquez sur OK pour fermer la boîte de dialogue.

Débogage de votre servlet ou de votre JSP

Pour déboguer votre servlet ou votre JSP, cliquez avec le bouton droit sur le fichier à déboguer et choisissez Débogage Web. Si vous avez créé une configuration d'exécution (voir "Exécution de votre servlet ou de votre JSP"), vous pouvez utiliser les commandes de débogage (Déboguer, Pas à pas ou Pas à pas approfondi) du menu Exécuter. Pour que le débogueur s'arrête sur une ligne de code spécifique, définissez un point d'arrêt dans votre servlet ou votre JSP. L'éditeur s'affiche automatiquement lorsque le point d'arrêt est atteint.

Remarque JBuilder fournit le débogage des sources pour les JSP. Cela vous permet de suivre directement votre code JSP - vous n'avez pas besoin d'effectuer le suivi dans le servlet dans lequel est compilée la JSP et d'essayer de faire la correspondance des numéros de lignes pour trouver les erreurs.

La commande Débogage Web affiche le débogueur dans le volet du serveur web. Les messages du serveur web et du débogueur s'affichent tous deux dans la vue Console d'entrée/sortie et d'erreurs du débogueur.



Pour plus d'informations sur le débogage, consultez les rubriques suivantes :

- “Débogage des programmes Java” dans *Construction d'applications avec JBuilder*
- “Tutoriel : Compilation, exécution et débogage”
- “Débogage des applications distribuées” dans la Partie III, “Guide du développeur d'applications distribuées”, du *Guide du développeur d'applications entreprise*

16

Déploiement de votre application web

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Le déploiement de votre application web est un processus impliquant les phases suivantes : déplacement de l'application sur le serveur web, insertion au bon emplacement sur le serveur web, accomplissement de toutes les étapes nécessaires à la reconnaissance correcte de votre application par le serveur web. Différents serveurs web impliquent différentes étapes pour le déploiement correct d'une application web. Vous devez consulter la documentation de votre serveur pour les questions spécifiques au serveur.

Présentation

Les sections suivantes traitent certains des problèmes à envisager lorsque vous déployer vos programmes sur un serveur web.

Constructeur d'archives

Rassembler tous les fichiers constituant votre application web dans une archive unique peut grandement simplifier son déploiement. Un fichier archive, comme un fichier WAR ou JAR, doit organiser les fichiers dont vous avez besoin pour votre application web dans la bonne hiérarchie. Vous pouvez alors copier le fichier archive à l'emplacement approprié de votre serveur web. Cela élimine le besoin de copier chaque fichier individuellement et garantit qu'ils se retrouvent tous dans des emplacements adaptés.

Un fichier WAR est le fichier servant d'archive à une application web. Il contient toute votre application web dans une structure appropriée et rend cette structure plus facile à dupliquer sur le serveur web. Les fichiers WAR sont traités en détails dans Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

Si votre application web contient une ou plusieurs applets, vous pouvez envisager de les placer dans un fichier JAR. Pour plus d'informations sur l'utilisation des fichiers JAR afin qu'ils contiennent des applets, voir Chapitre 4, "Utilisation des applets".

Votre application web, fichier WAR ou fichier JAR, peut aussi être empaquetée dans un fichier EAR. Voir dans l'aide en ligne l'expert EAR.

Descripteurs de déploiement

Les descripteurs de déploiement sont des fichiers XML qui contiennent les informations sur la WebApp nécessaires au serveur web. Vous utiliserez certainement un ou plusieurs descripteurs de déploiement si votre application web contient des servlets ou des JSP. Recherchez dans la documentation de votre serveur web les informations concernant le ou les descripteurs de déploiement qui lui sont nécessaires.

Applets

Voir "Déploiement des applets", page 4-14, pour plus d'informations sur le déploiement des applets.

Servlets

Le déploiement des servlets peut être difficile, car s'il n'est pas réalisé correctement, le serveur web ne parviendra pas à reconnaître votre servlet. Afin de simplifier le déploiement, vous devez envisager d'archiver votre servlet dans un fichier WAR. Cela vous permet de rassembler tous les fichiers et les ressources nécessaires au servlet dans une hiérarchie organisée correctement avant le déploiement. Ensuite vous n'aurez qu'à déployer le fichier WAR sur le serveur web.

Quelque soit le serveur web, le succès du déploiement nécessite la présence de certaines informations dans le descripteur de déploiement, `web.xml`. Votre serveur web peut aussi avoir des exigences supplémentaires. Au minimum, avant de déployer un servlet standard, vous aurez besoin de spécifier ce qui suit dans un élément `servlet` du fichier `web.xml` :

- `servlet-name` - un nom pour le servlet
- `servlet-class` - le nom complètement qualifié de la classe pour le servlet

Chaque servlet standard doit aussi spécifier un `servlet-mapping` dans le `web.xml`. Vous devrez spécifier ce qui suit dans un élément `servlet-mapping` :

- `servlet-name` - le nom utilisé dans la balise `servlet`
- `url-pattern` - le motif d'URL auquel est affecté le servlet

Un servlet filtre ou un servlet auditeur nécessitent des balises différentes. Voir les sections “Page Filtres” et “Page Auditeurs” dans l’éditeur DD WebApp pour plus d’informations sur les balises nécessaires à ces types de servlets.

Si vous utilisez l’expert servlet de JBuilder pour construire votre servlet, l’expert va insérer pour vous dans le `web.xml` les informations requises par le servlet. Cela est vrai pour un servlet standard, un servlet filtre, ou un servlet auditeur.

Les servlets doivent être compilés avant d’être déployés sur le serveur web.

JSP

Les JSP sont plus faciles à déployer que les servlets. Vous pouvez souhaiter les archiver dans un fichier WAR pour faciliter encore le déploiement.

Les JSP sont mappées par un serveur web de la même façon que le sont les fichiers HTML ; le serveur reconnaît l’extension de fichier. La compilation est également gérée par le serveur web. N’oubliez pas, les JSP sont compilées dans les servlets par le serveur web avant leur exécution.

Certaines JSP utilisent des bibliothèques de balises JSP. Ces bibliothèques doivent aussi être déployées sur le serveur web ; le serveur web doit savoir comment les trouver. Pour cette raison, si vous avez une JSP qui utilise une bibliothèque de balises, votre descripteur de déploiement, `web.xml`, aura besoin d’un élément `taglib` qui indique quelle bibliothèque de balises utiliser et où elle se trouve. Les informations suivantes se trouvent dans l’élément `taglib` :

- `taglib-uri` - l’URI utilisé dans la JSP pour identifier la bibliothèque de balises
- `taglib-location` - l’emplacement réel de la bibliothèque de balises

Si vous utilisez l’expert JSP de JBuilder pour créer une JSP qui utilise la bibliothèque de balises InternetBeans, l’information sur la bibliothèque de balises InternetBeans est ajoutée pour vous au fichier `web.xml`.

Test de votre application web

Après avoir déployé votre application web sur le serveur web, vous devez la tester pour savoir si elle a été correctement déployée. Vous essayerez d'accéder à toutes les pages, servlets, JSP et applets de votre application pour savoir s'ils fonctionnent comme prévu. Cela doit se faire depuis un navigateur présent sur une autre machine, afin d'être certain que l'application web est accessible sur le Web, et non seulement en local. Vous pouvez également envisager le test sur plusieurs types de navigateurs, l'aspect de l'application dans les différents navigateurs pouvant varier, en particulier lorsque vous utilisez des applets.

Descripteurs de déploiement

Les descripteurs de déploiement sont des fichiers XML qui contiennent les informations sur la WebApp nécessaires au serveur web. Tous les serveurs compatibles Java sont en attente du descripteur de déploiement standard appelé `web.xml`. Certains serveurs peuvent aussi avoir des descripteurs de déploiement spécifiques au fournisseur qu'ils utilisent en plus du fichier `web.xml`. Par exemple, WebLogic utilise `weblogic.xml`. Recherchez dans la documentation de votre serveur web les descripteurs de déploiement qui lui sont nécessaires. Tomcat, le serveur web fourni avec JBuilder ne nécessite que `web.xml`.

JBuilder fournit un éditeur de descripteur de déploiement permettant de modifier le fichier `web.xml`. Il s'appelle l'éditeur DD WebApp. Il fournit une interface utilisateur graphique (Graphical User Interface, GUI) permettant de modifier les informations les plus fréquemment utilisées du fichier `web.xml`.

Lorsque le fichier `web.xml` est ouvert dans l'EDI de JBuilder, son contenu est affiché dans le volet structure tandis que l'AppBrowser affiche l'éditeur DD WebApp, l'éditeur Source et la vue Historique. Vous pouvez modifier le fichier `web.xml` soit dans l'éditeur DD WebApp soit dans l'éditeur Source. Les modifications apportées dans l'éditeur DD WebApp seront reflétées par le source et les changements de code effectués dans le source dans l'éditeur DD WebApp. Mais, n'oubliez pas que si vous saisissez des commentaires dans le fichier `web.xml`, ils disparaîtront si vous ouvrez par la suite le fichier dans l'éditeur DD WebApp.

L'éditeur DD WebApp

L'éditeur DD WebApp est actif lorsque le fichier `web.xml` est ouvert dans le volet contenu. En même temps, le volet structure montre le développement du contenu de ce fichier. Cliquer sur les divers nœuds dans le volet structure affiche les pages correspondantes dans l'éditeur.

Il y a 12 nœuds principaux, certains d'entre eux ont des nœuds enfant. Voici la liste des nœuds principaux :

- Descripteur de déploiement WebApp
- Paramètres de contexte
- Filtres (Spécification Servlet 2.3)
- Auditeurs (Spécification Servlet 2.3)
- Servlets
- Bibliothèques de balises
- Types MIME
- Pages erreur
- Environnement
- Références de ressource
- Références EJB
- Connexion
- Sécurité

Chacun de ces nœuds contient des balises que vous pouvez modifier dans l'éditeur DD WebApp. L'éditeur DD WebApp prend en compte toutes les balises du descripteur de déploiement `web.xml` conformes à la spécification Servlet 2.3. Vous pouvez aussi modifier le source du fichier `web.xml`. Les balises contenues dans chacun des nœuds de l'éditeur DD WebApp sont décrites dans les prochaines sections.

Menu contextuel de l'éditeur DD WebApp

Cliquez avec le bouton droit sur les nœuds principaux de l'éditeur DD WebApp fait apparaître un menu contextuel qui vous permet d'ajouter un nouveau nœud filtre, un nouveau nœud servlet ou un nouveau nœud contrainte de sécurité. Notez que l'ajout d'un nœud filtre n'est possible que si votre serveur web supporte la spécification Servlet 2.3, puisque les servlets filtre ont été introduits par cette version de la spécification des servlets.

Cliquez avec le bouton droit sur un nœud contrainte de sécurité fait apparaître un menu contextuel qui vous permet d'ajouter un nouveau nœud collection de ressources web à cette contrainte de sécurité ou à un autre nœud contrainte de sécurité. Il doit exister au moins un nœud contrainte de sécurité avant qu'un nœud collection de ressources web puisse être ajouté.

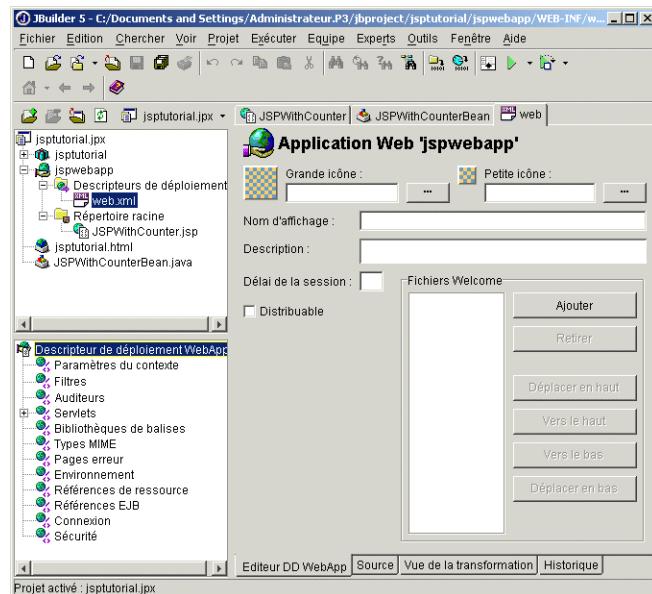
Le menu contextuel d'un nœud servlet, filtre ou collection de ressources web existant contient également des options permettant de le renommer ou de le supprimer. Le menu contextuel d'un nœud contrainte de sécurité contient une option de suppression du nœud en cours (il ne contient pas d'option permettant de le renommer car les nœuds contrainte de sécurité n'ont pas de nom). Renommer ou supprimer un nœud répercute la modification dans toutes les parties du fichier `web.xml` associées à ce nœud.

Page Descripteur de déploiement WebApp

La page principale de l'éditeur DD WebApp contient les informations de base permettant l'identification de votre WebApp. Voici la liste des informations que vous pouvez modifier sur cette page :

Elément	Description
Grande icône	Pointe sur l'emplacement d'une grande icône (32 x 32 pixels) pour la WebApp, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Petite icône	Pointe sur l'emplacement d'une petite icône (16 x 16 pixels) pour la WebApp, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Nom d'affichage	Nom à afficher pour la WebApp.
Description	Description de la WebApp.
Délai de la session	Un nombre entier de minutes pouvant s'écouler avant l'épuisement du délai d'une session.
Distribuable	Indique si l'application web peut être déployée dans un conteneur de servlet distribué (multi-VM).
Fichiers Welcome	Le ou les fichiers devant être affichés lorsque l'URL pointe sur un répertoire, par exemple : index.html

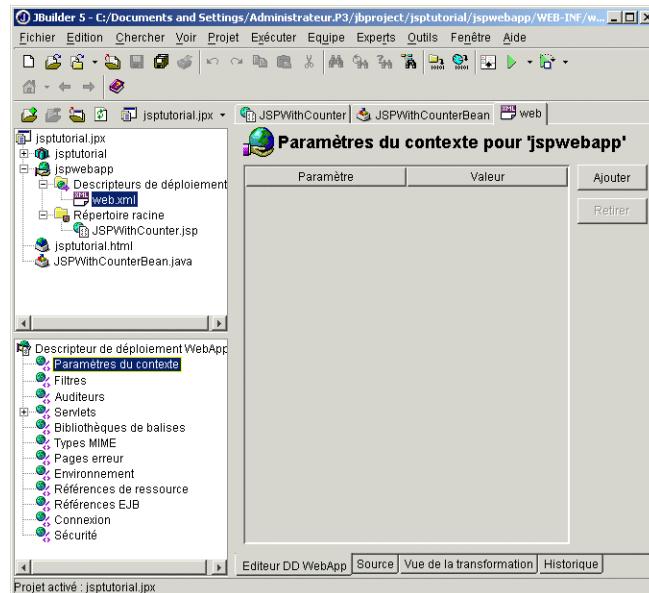
Figure 16.1 Page Descripteur de déploiement WebApp de l'éditeur DD WebApp



Page Paramètres de contexte

La page Paramètres de contexte contient une grille des paramètres d'initialisation du `ServletContext` de la WebApp et des valeurs de ces paramètres.

Figure 16.2 Page Paramètres de contexte de l'éditeur DD WebApp



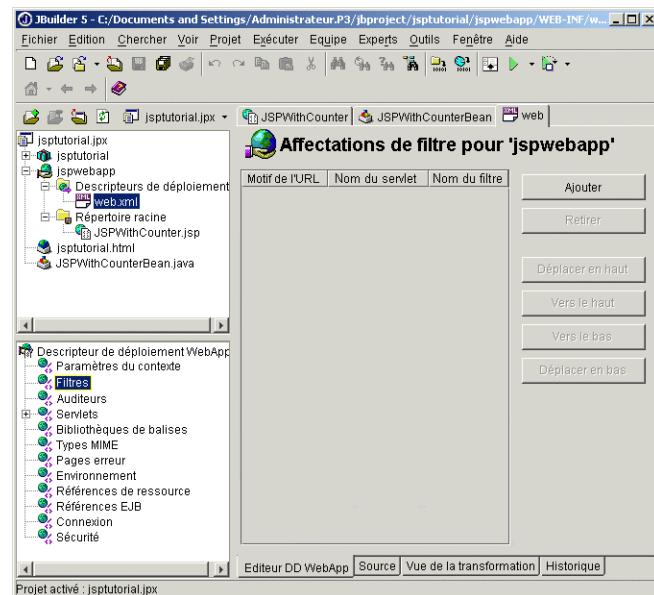
Page Filtres

La page Filtres ne sera visible que si votre serveur web supporte la spécification Servlet 2.3. Cette page contient une grille faisant correspondre les filtres (par le `filter-name`) soit à un motif d'URL, soit à un nom de servlet (mais pas les deux). L'ordre des filtres est important car c'est l'ordre dans lequel ils seront appliqués. Cette page vous permet de modifier l'ordre dans lequel sont appliqués les filtres. Ce qui suit décrit les informations présentées dans la page Filtres :

Elément	Description
Motif de l'URL	Le <code>url-pattern</code> pour l'emplacement du filtre. Soit cet élément soit le <code>servlet-name</code> est nécessaire lorsque vous déployez un servlet filtre.
Nom du servlet	Le <code>servlet-name</code> qui est utilisé pour affecter le filtre. Soit cet élément soit le <code>url-pattern</code> est nécessaire lorsque vous déployez un servlet filtre.
Nom du filtre	Le <code>filter-name</code> qui est utilisé pour affecter le filtre. Cet élément est nécessaire lorsque vous déployez un servlet filtre.

Si vous utilisez l'expert servlet de JBuilder pour créer un servlet, l'expert va ajouter pour vous l'affectation nécessaire au filtre.

Figure 16.3 Page Filtres de l'éditeur DD WebApp



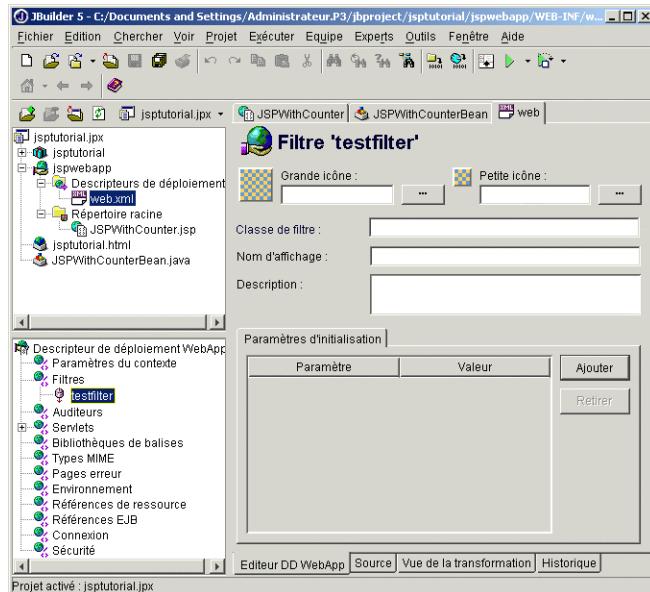
Chaque filtre individuel est présenté dans le volet structure comme un nœud enfant séparé du nœud Filtres. Le `filter-name` est affiché dans l'arborescence. Vous pouvez renommer ou supprimer un filtre en cliquant avec le bouton droit sur le nœud et en choisissant Renommer ou Supprimer dans le menu contextuel. Si vous renommez ou supprimez un filtre, cette modification sera répercutee dans toutes les parties du descripteur de déploiement qui lui sont associées.

Lorsque le nœud d'un filtre particulier est ouvert, l'éditeur DD WebApp affiche une page pour ce filtre. Cette page contient les informations d'identification suivantes pour le filtre :

Elément	Description
Grande icône	Pointe sur l'emplacement d'une grande icône (32 x 32 pixels) pour le filtre, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Petite icône	Pointe sur l'emplacement d'une petite icône (16 x 16 pixels) pour le filtre, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Classe de filtre	Nom de classe complet pour le filtre. Cet élément est nécessaire lorsque vous déployez un servlet filtre.
Nom d'affichage	Nom à afficher pour le filtre.

Elément	Description
Description	Description du filtre.
Paramètres d'initialisation	Paramètres d'initialisation pour le filtre.

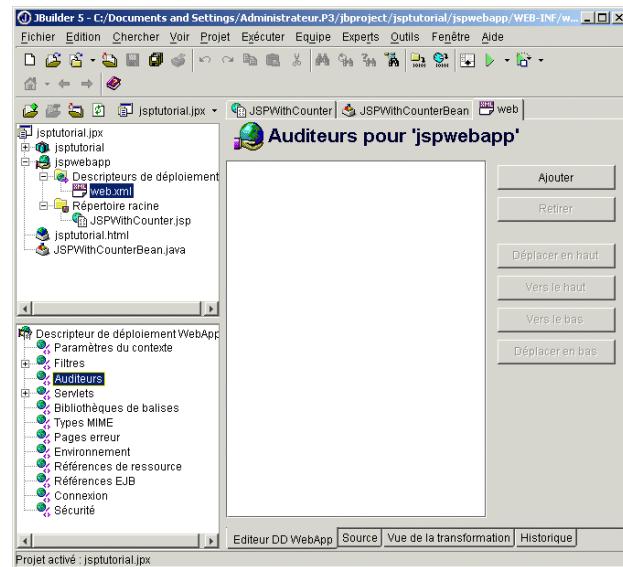
Figure 16.4 Nœud d'un filtre individuel dans l'éditeur DD WebApp



Page Auditeurs

La page Auditeurs ne sera visible que si votre serveur web supporte la spécification Servlet 2.3. La page Auditeurs possède une boîte listant les classes bean auditeur de l'application web. Cette information est nécessaire lorsque vous déployez un servlet filtre. Si vous utilisez l'expert servlet de JBuilder pour créer un servlet auditeur, la classe du servlet sera ajoutée pour vous à cette liste.

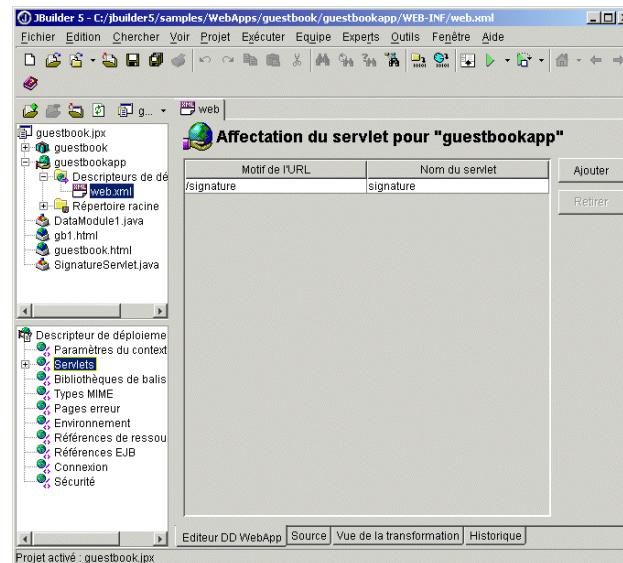
Figure 16.5 Page Auditeurs de l'éditeur DD WebApp



Page Servlets

La page Servlets contient une grille faisant correspondre les motifs d'URL à un nom de servlet. Notez qu'un servlet peut être affecté à plus d'un motif d'URL. Au moins une affectation est recommandée pour chaque servlet. Si vous utilisez l'expert servlet de JBuilder pour créer un servlet standard, l'expert remplira pour vous l'affectation nécessaire au servlet.

Figure 16.6 Page Servlets de l'éditeur DD WebApp



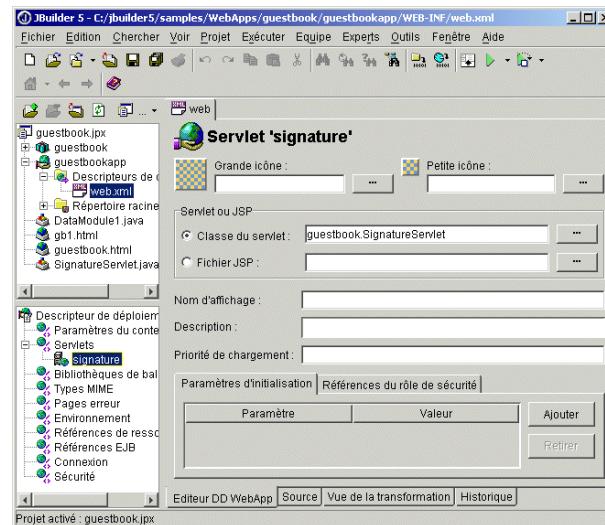
Si des servlets sont définis, vous trouverez des nœuds enfant représentant chacun un servlet particulier sous le nœud de la page Servlets dans le volet structure. Le `servlet-name` est affiché dans l'arborescence. Vous pouvez renommer ou supprimer un servlet en cliquant avec le bouton droit sur le nœud du servlet et en choisissant Renommer ou Supprimer dans le menu contextuel. Si vous renommez ou supprimez un servlet, cette modification sera répercutée dans toutes les parties du descripteur de déploiement qui lui sont associées. Par exemple, supprimer un servlet supprime également toutes ses affectations d'URL et de filtres.

N'oubliez pas que vous pouvez également affecter des motifs d'URL à des JSP. Cela signifie que le nœud d'un servlet individuel dans le volet structure peut représenter une JSP ou un servlet.

Lorsque le nœud d'un servlet particulier est ouvert, l'éditeur DD WebApp affiche une page pour ce servlet. Cette page contient les informations d'identification suivantes pour le servlet :

Elément	Description
Grande icône	Pointe sur l'emplacement d'une grande icône (32 x 32 pixels) pour le filtre, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Petite icône	Pointe sur l'emplacement d'une petite icône (16 x 16 pixels) pour le filtre, qui doit se trouver dans la hiérarchie de répertoire de la WebApp.
Classe du servlet	Sélectionnez ce bouton radio pour un servlet. Entrez dans le champ texte le nom de classe complet pour le servlet.
Fichier JSP	Sélectionnez ce bouton radio pour une JSP. Entrez dans le champ texte le chemin d'accès au fichier JSP.
Nom d'affichage	Nom à afficher pour le servlet.
Description	Description du servlet.
Priorité de chargement	La priorité <code>load-on-startup</code> pour le servlet, sous forme d'entier positif. Un servlet associé à un entier moindre sera chargé avant un servlet associé à un entier plus élevé.
Paramètres d'initialisation	Paramètres d'initialisation pour le servlet.

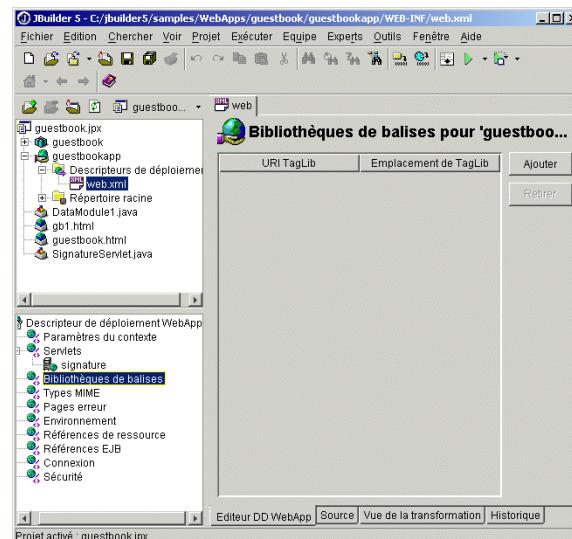
Figure 16.7 Nœud d'un servlet individuel dans l'éditeur DD WebApp



Page Bibliothèques de balises

La page Bibliothèques de balises contient une grille faisant correspondre les URI utilisés dans une JSP avec les emplacements réels des fichiers .tld (Tag Library Definition). Cette information est nécessaire pour le déploiement d'une JSP utilisant une bibliothèque de balises. Si vous utilisez l'expert JSP de JBuilder pour créer une JSP qui utilise la bibliothèque de balises InternetBeans, l'information sur la bibliothèque de balises InternetBeans est ajoutée pour vous par l'expert.

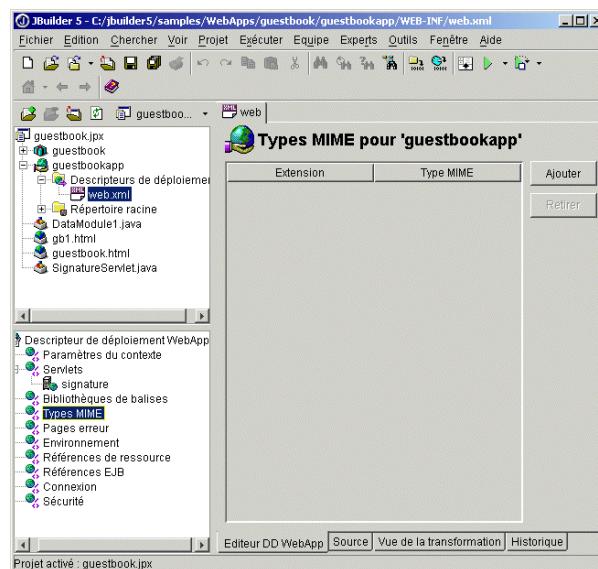
Figure 16.8 Page Bibliothèques de balises de l'éditeur DD WebApp



Page Types MIME

La page Types MIME contient une grille faisant correspondre les extensions aux noms de types.

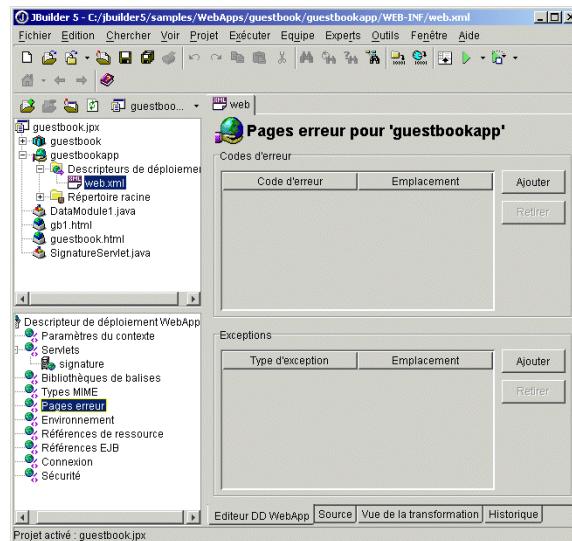
Figure 16.9 Page Types MIME de l'éditeur DD WebApp



Page Pages erreur

La page Pages erreur contient deux grilles, une pour les numéros des codes et une pour les noms des classes d'exception, qui sont associés aux emplacements des pages à afficher dans le cas d'erreur ou d'exception.

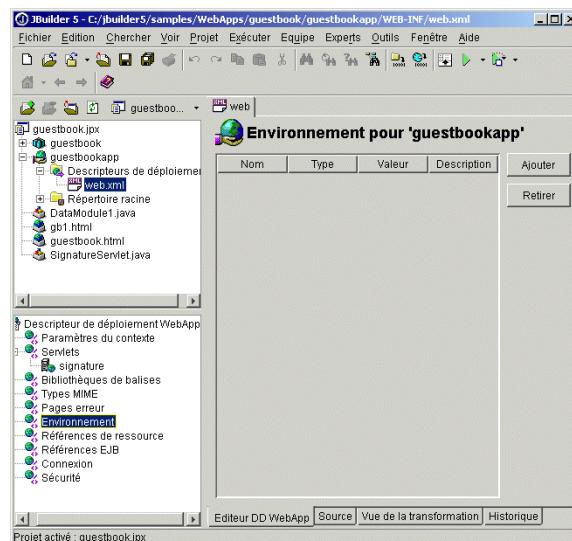
Figure 16.10 Page Pages erreur de l'éditeur DD WebApp



Page Environnement

La page Environnement contient une grille présentant les noms des entrées d'environnement, leurs valeurs, leurs types et un champ texte de description pour l'entrée en cours de sélection.

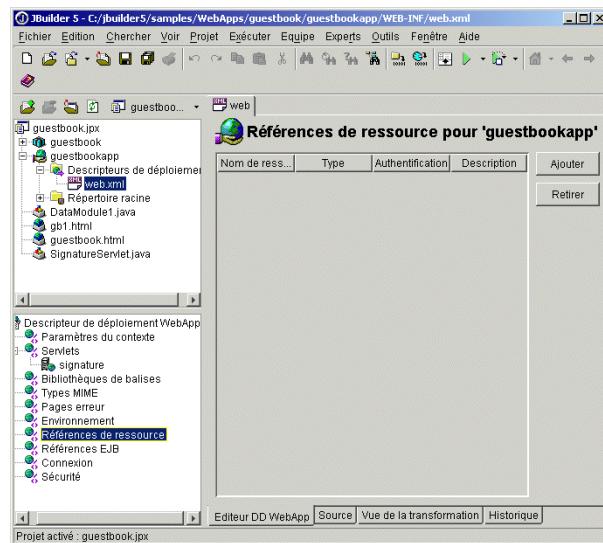
Figure 16.11 Page Environnement de l'éditeur DD WebApp



Page Références de ressource

La page Références de ressource contient une grille présentant les noms des ressources, leurs types et l'information indiquant s'ils utilisent une authentification Container ou Servlet, plus un champ texte de description pour l'entrée en cours de sélection.

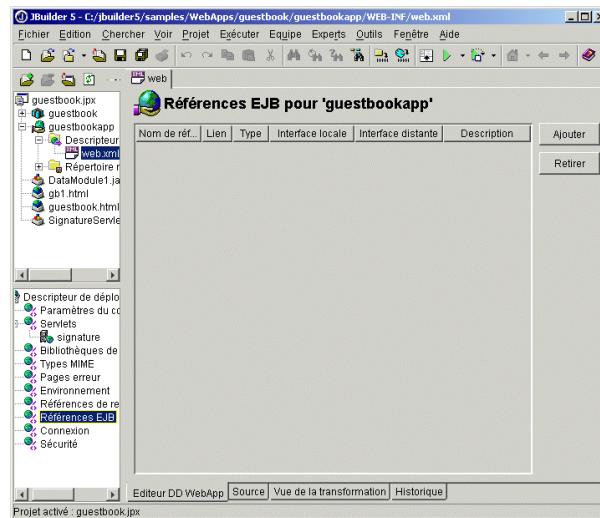
Figure 16.12 Page Références de ressource de l'éditeur DD WebApp



Page Références EJB

La page Références EJB contient une grille présentant les noms EJB, leurs types, les interfaces locales et distantes, un ejb-link facultatif, plus un champ texte de description pour l'entrée en cours de sélection. Cette page ressemble à la page EJB References de l'EJB DD Editor.

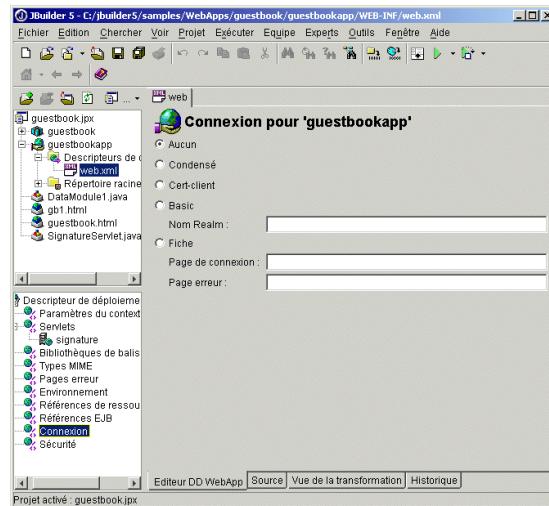
Figure 16.13 Page Références EJB de l'éditeur DD WebApp



Page Connexion

La page Connexion affiche un ensemble de boutons radio permettant de choisir la méthode d'authentification pour la WebApp. La valeur par défaut est aucun. Les autres options sont Condensé, Cert-client, Basic et Fiche. Pour Basic, vous pouvez spécifier un nom Realm. Pour Fiche, vous pouvez spécifier une page de connexion et une page erreur.

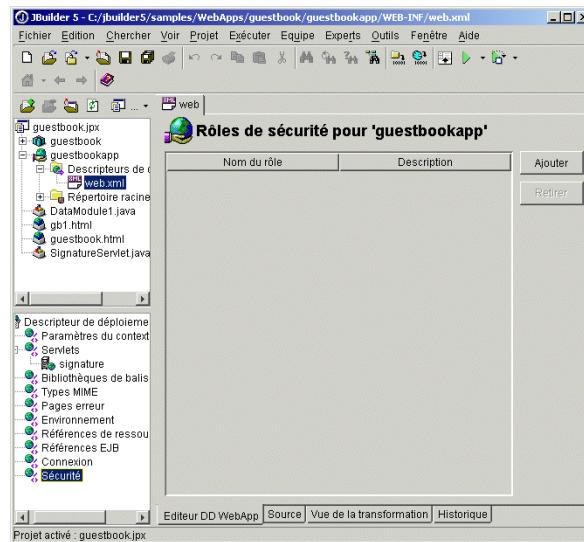
Figure 16.14 Page Connexion de l'éditeur DD WebApp



Page Sécurité

La page Sécurité contient une grille de noms de rôles et leurs descriptions.

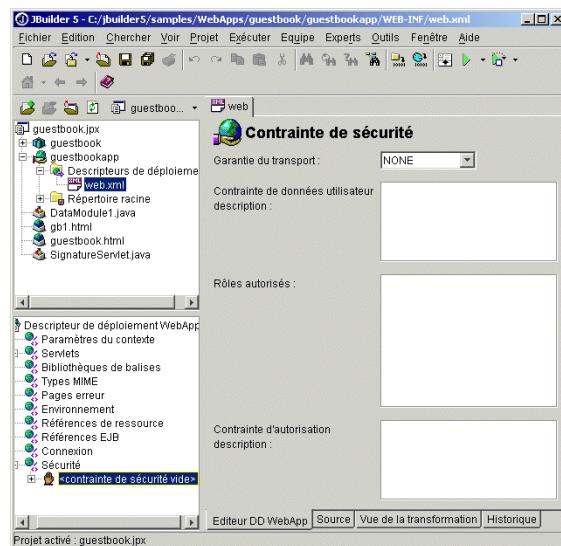
Figure 16.15 Page Sécurité de l'éditeur DD WebApp



Chaque contrainte de sécurité est présentée comme un nœud enfant séparé du nœud Sécurité. Vous pouvez renommer ou supprimer un contrainte de sécurité en cliquant avec le bouton droit sur le nœud et en choisissant Renommer ou Supprimer dans le menu contextuel. Si vous renommez ou supprimez une contrainte, cette modification sera répercutée dans toutes les parties du descripteur de déploiement qui lui sont associées.

Lorsqu'un nœud contrainte de sécurité est ouvert, l'éditeur DD WebApp affiche les informations sur cette contrainte – la garantie du transport, les noms de rôles autorisés, et la description de ces deux éléments.

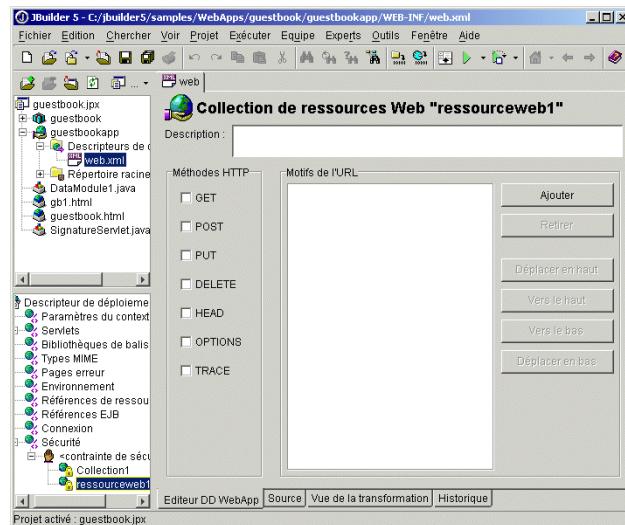
Figure 16.16 Contrainte de sécurité de l'éditeur DD WebApp



Chaque web-resource-name de la collection de ressources web est présenté comme un nœud enfant séparé d'une ou de plusieurs contraintes de sécurité applicables. Vous ne pouvez pas ajouter une collection de ressources web si vous n'avez pas au moins une contrainte de sécurité. Chaque contrainte de sécurité doit avoir au moins une collection de ressources web. Vous pouvez renommer ou supprimer une collection de ressources web en cliquant avec le bouton droit sur le nœud et en choisissant Renommer ou Supprimer dans le menu contextuel. Si vous renommez ou supprimez une collection de ressources web, cette modification sera répercutée dans toutes les parties du descripteur de déploiement qui lui sont associées. La suppression de la dernière collection de ressources web d'une contrainte supprime également cette contrainte.

Lorsqu'un nœud collection de ressources web est ouvert, l'éditeur DD WebApp contient les informations d'identification de la collection de ressources web :

Elément	Description
Description	Description de la collection de ressources web.
Motifs d'URL	Une boîte liste des motifs d'URL pour la collection de ressources web.
Méthodes HTTP	Un ensemble de cases à cocher pour les méthodes HTTP (comme GET et POST) qui s'appliquent aux motifs d'URL. Sélectionner aucune case revient à sélectionner toutes les cases.

Figure 16.17 Nœud collection de ressources web dans l'éditeur DD WebApp

Modification des descripteurs de déploiement spécifiques au fournisseur

Vous pouvez également modifier les descripteurs de déploiement spécifiques au fournisseur dans l'EDI de JBuilder. Un fichier descripteur de déploiement spécifique au fournisseur est reconnu par JBuilder et affiché dans votre projet seulement si le plugin du serveur correspondant est correctement configuré. Pour plus d'informations sur la configuration du plugin de votre serveur web, voir "Configuration de JBuilder pour des serveurs web autres que Tomcat", page 14-5.

Les fournisseurs qui proposent des plugins pour des serveurs web autres que Tomcat sont incités à fournir des éditeurs GUI pour leurs descripteurs de déploiement. Si votre fournisseur n'a pas fourni d'éditeur GUI pour ses descripteurs de déploiement, vous pouvez toujours modifier le code source des descripteurs de déploiement dans l'éditeur de source XML de JBuilder. Pour cela, ouvrez le descripteur de déploiement spécifique au fournisseur et cliquez sur l'onglet Source du volet contenu.

Lorsqu'un descripteur de déploiement spécifique au fournisseur est ouvert dans l'EDI de JBuilder, son contenu est affiché dans le volet structure, et l'AppBrowser affiche l'éditeur Source et la vue Historique, ainsi que l'éditeur GUI spécifique au fournisseur dans les cas où il est fourni par le plugin de votre serveur web.

Pour plus d'informations sur la création d'un plugin personnalisé pour un serveur web, voir "Création du plugin de votre serveur web", page 14-10.

Informations supplémentaires sur les descripteurs de déploiement

Pour plus d'informations sur les descripteurs de déploiement, et sur le descripteur `web.xml` en particulier, voir la spécification Java Servlet, téléchargeable à l'adresse <http://java.sun.com/aboutJava/communityprocess/first/jsr053/index.html>.

Lancement de votre application web avec Java Web Start

Le développement pour le web est une fonctionnalité de JBuilder Professionnel et de JBuilder Entreprise.

Java Web Start est une nouvelle technologie de déploiement des applications conçue par Sun Microsystems. Elle permet de lancer n'importe quelle applet ou application Java à partir d'un lien présent sur une page web affichée dans votre navigateur web. Si l'application n'est pas sur votre ordinateur, Java Web Start télécharge tous les fichiers nécessaires et les met en mémoire cache locale de sorte que l'application puisse être relancée à partir d'une icône de votre bureau ou du lien de la page web.

Java Web Start est l'implémentation de référence de la technologie JNLP (Java Network Launching Protocol). Cette technologie définit un format de fichier standard décrivant comment lancer une application JNLP. L'expert JBuilder Lanceur de démarrage Web génère à votre place un fichier JNLP, ainsi qu'une page d'accueil HTML pour votre application.

Pour ce qui concerne Web Start, voir la page Java Web Start, à l'adresse <http://java.sun.com/products/javawebstart/>. Vous pouvez aussi consulter :

- Le Java Web Start *Developer's Guide*, à l'adresse <http://java.sun.com/products/javawebstart/docs/developersguide.html>
- Les "Frequently Asked Questions", à l'adresse <http://java.sun.com/products/javawebstart/faq.html>

Pour plus d'informations sur JBuilder et Web, voir "Java Web Start et JBuilder", page 17-3.

Observations sur les applications Java Web Start

Généralement, le développement d'une application pour son déploiement avec Web Start sera identique au développement d'une application autonome. Le point d'entrée de l'application est la méthode `main()` et l'application doit être livrée sous forme d'un fichier JAR ou d'un ensemble de fichiers JAR. Cependant, toutes les ressources de l'application doivent être appelées en utilisant le mécanisme `getResource`. Pour plus d'informations, voir "Application Development Considerations" dans le *Java Web Start Developer's Guide*, à l'adresse <http://java.sun.com/products/javawebstart/docs/developersguide.html#dev>.

L'un des points essentiels concernant l'exécution des applications sur Internet est la sécurité. Les utilisateurs ne seront pas disposés à télécharger et à exécuter des programmes sur leurs ordinateurs s'ils n'ont pas la garantie d'un minimum de sécurité qui évite la suppression de fichiers ou le téléchargement d'informations personnelles à partir de leur machine.

Java Web Start répond à ce souci en exécutant tout le code non sûr dans un environnement limité, connu sous le nom de *sas de sécurité*.

L'application s'exécutant dans le sas de sécurité, Java Web Start peut garantir qu'elle ne compromettra pas la sécurité des fichiers locaux ni des fichiers sur le réseau.

De plus, Java Web Start prend en charge la signature électronique du code. Cette technologie permet à Java Web Start de vérifier que le contenu d'un fichier JAR n'a pas été modifiée depuis la signature. Si la vérification échoue, Java Web Start n'exécutera pas l'application. Pour plus d'informations sur Java Web Start et la sécurité, voir "Security And Code Signing" dans la section "Application Development Considerations" du *Java Web Start Developer's Guide*, à l'adresse <http://java.sun.com/products/javawebstart/docs/developersguide.html#dev>.

L'API JNLP fournit des services supplémentaires de gestion des fichiers destinés à l'exécution dans un environnement limité. Ces classes remplacent les opérations standard de téléchargement, d'ouverture, d'écriture de d'enregistrement des fichiers. Par exemple, vous devrez utiliser les méthodes de `javax.jnlp.FileOpenService` pour importer des fichiers depuis le disque local, même pour les applications s'exécutant dans le sas de sécurité.

Tableau 17.1 Présentation de l'API JNLP

Nom	Méthodes pour
<code>BasicService</code>	Interroger et interagir avec l'environnement.
<code>ClipboardService</code>	Accéder au presse-papiers du niveau système.
<code>DownloadService</code>	Permettre à une application de contrôler comment sont mises en cache ses ressources.

Tableau 17.1 Présentation de l'API JNLP (suite)

Nom	Méthodes pour
FileOpenService	Importer des fichiers à partir du disque local.
FileSaveService	Exporter des fichiers dans le disque local.
PrintService	Accéder à l'imprimante.
PersistenceService	Stocker des données en local.
FileContents	Encapsuler le nom et le contenu d'un fichier.

Pour plus d'informations, voir "JNLP API Examples" dans le Java Web Start *Developer's Guide*, à l'adresse <http://java.sun.com/products/javawebstart/docs/developersguide.html#api>.

Installation de Java Web Start

Java Web Start n'est pas intégré à JBuilder. Pour les instructions de téléchargement et d'installation, voir la page Java Web Start, à l'adresse <http://java.sun.com/products/javawebstart/> et cliquer sur l'icône "Download Now". L'installation dépendra du système d'exploitation de votre ordinateur. Lorsque vous avez installé Java Web Start, vous n'avez pas à configurer Java Web Start, JBuilder, ni votre navigateur web : tous les trois s'exécuteront de façon transparente.

Java Web Start et JBuilder

JBuilder fournit un certain nombre de fonctionnalités permettant de transformer votre application autonome en application Web Start. Pour cela, vous devrez respecter les étapes générales suivantes :

- 1 Créez une WebApp avec l'expert Application web.

Pour plus d'informations sur les WebApps, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

- 2 Créez le fichier JAR de l'application avec le constructeur d'archives, en utilisant les options suivantes dans les étapes 1 et 2 de l'expert. Les options des autres étapes peuvent garder leurs valeurs par défaut.

Tableau 17.2 Options du constructeur d'archives

Constructeur d'archives	Option
Etape 1	Type d'archive - Applet de démarrage Web ou Application de démarrage Web
Etape 2	Nom - nom du fichier JAR Fichier - Emplacement dans la structure de répertoires de la WebApp. C'est, par défaut, dans la racine de la WebApp.

Pour plus d'informations sur la création des fichiers JAR, voir “Utilisation du constructeur d'archives” dans le manuel en ligne *Construction d'applications avec JBuilder*.

- 3** Construisez le projet pour créer le fichier JAR.
- 4** Créez le fichier JNLP et la page d'accueil de l'application, en utilisant les options suivantes dans l'expert Lanceur de démarrage Web :

Tableau 17.3 Options du Lanceur de démarrage Web

Options du Lanceur de démarrage Web	Option
Etape 1	Nom - Nom de l'application. Fichier JAR - Nom et chemin d'accès du fichier JAR. Classe principale - Classe contenant la méthode main(). Créer une page d'accueil - A conserver sélectionnée.
Etape 2	Titre - Titre de l'application. Fournisseur - Nom de l'entreprise. Description - Description de l'application. Autoriser l'usage hors ligne - A cocher pour démarrer à partir du bureau.

- 5** Cliquez avec le bouton droit sur le fichier HTML de l'application (créé par l'expert Lanceur de démarrage Web) et choisissez Exécution Web.
 - 6** Copiez l'URL de l'application à partir du champ URL situé en haut de la Vue Web. Vous pouvez configurer cette option automatiquement sur la page Web de la boîte de dialogue Options de l'EDI (Outils | Options de l'EDI).
 - 7** Collez l'URL dans votre navigateur externe.
 - 8** Cliquez sur le lien vers votre application dans la page web.
- Chacune de ces étapes est largement développée dans le “Tutoriel : Exécution de l'application exemple CheckBoxControl avec Java Web Start”, page 17-6.

Le fichier JAR de l'application

Le constructeur d'archives de JBuilder vous permet de choisir entre deux types d'archive JAR, Applet de démarrage Web et Application de démarrage Web. Le constructeur d'archives place le fichier JAR résultant dans le répertoire de l'application web sélectionnée (WebApp), afin qu'il soit servi par le serveur web.

Le fichier JNLP et la page d'accueil de l'application

L'expert JBuilder Lanceur de démarrage Web crée la page d'accueil HTML et le fichier JNLP de votre application. L'expert vous permet aussi

de spécifier le titre de l'application, le nom de la société qui l'a créée, et une description. Cette information est affichée lorsque Java Web Start lance votre application. De plus, vous pouvez utiliser l'expert pour permettre à l'application d'être démarrée hors ligne, à partir d'une icône sur le bureau.

Remarque L'expert Lanceur de démarrage Web nécessite que vous ayez préalablement créé et construit le fichier JAR de votre application.

Attention L'expert Lanceur de démarrage Web donne le même nom aux fichiers JNLP et HTML. Si le nom saisi dans le champ Nom de l'étape 1 de l'expert correspond au nom d'un fichier HTML ou JNLP existant dans votre projet, il vous sera demandé si vous voulez écraser ce fichier.

Le fichier JNLP est un document XML. Les éléments du fichier décrivent les caractéristiques de l'application, comme son nom, son fournisseur, sa page d'accueil; ainsi que des particularités JNLP. Pour plus d'informations, voir "JNLP File Syntax" dans le Java Web Start *Developer's Guide*, à l'adresse <http://java.sun.com/products/javawebstart/docs/developersguide.html>.

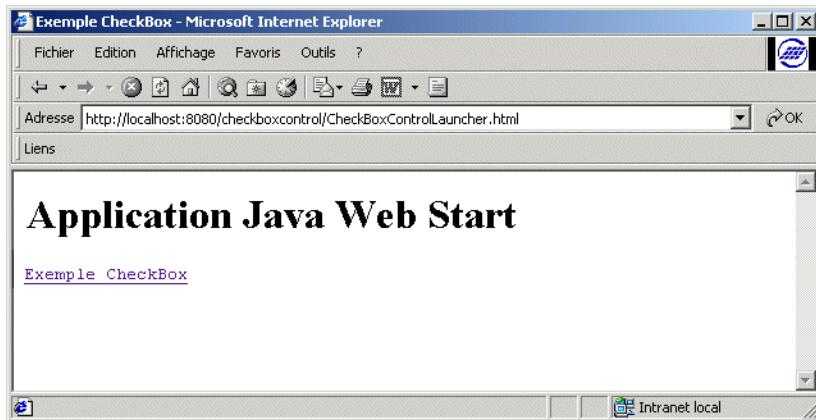
Remarque Avant de déployer votre application Web Start, vous devez modifier l'attribut `codebase` du fichier JNLP. Cet attribut est généré automatiquement à la valeur `localhost:8080`. Vous devrez le modifier pour qu'il corresponde à votre serveur web.

La page d'accueil de votre application est un fichier HTML ; il contient du code JavaScript, du code VBScript, ainsi que des balises HTML. Le script détermine si vous exécutez le fichier HTML depuis JBuilder ou depuis un navigateur web externe. Si vous êtes dans JBuilder, vous verrez un message dans la Vue Web expliquant qu'il vous faut Java Web Start pour exécuter cette application. La Vue Web doit ressembler à ceci :



Si vous êtes dans un navigateur externe, vous verrez un lien vers votre application (si vous avez déjà installé Web Start). Cliquez sur le lien pour lancer Java Web Start et exécuter votre application.

Le navigateur externe doit ressembler à ceci :



Important Les applications Java Web Start ne peuvent pas être lancées depuis JBuilder. Pour lancer votre application, vous devez coller l'URL de l'application dans votre lanceur externe.

Tutoriel : Exécution de l'application exemple CheckBoxControl avec Java Web Start

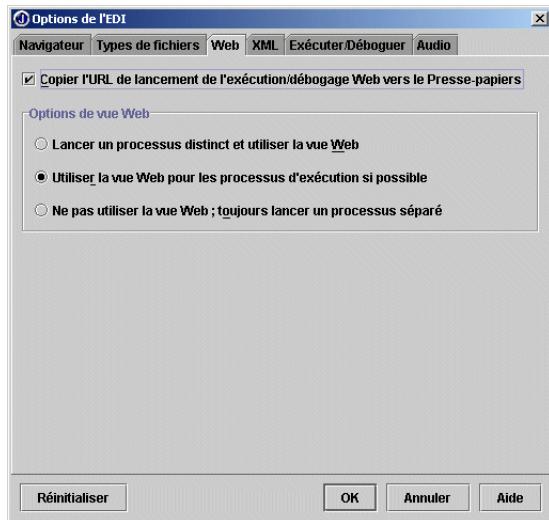
Cette section parcourt les étapes du lancement avec Web Start d'une application exemple fondée sur les composants Swing. L'exemple, CheckBoxControl, se trouve dans le répertoire samples/Swing de votre installation JBuilder.

Etape 1 : Ouverture et configuration du projet

D'abord, vous ouvrirez le projet dans JBuilder et définirez les options de l'EDI concernant la vue web. Pour ce faire,

- 1 Choisissez Fichier | Ouvrir un projet pour afficher la boîte de dialogue Ouverture de projet.
- 2 Dans la boîte de dialogue Ouverture de projet, cliquez sur le bouton Exemples. Naviguez jusqu'à `Swing/CheckBoxControl/`. Cliquez sur `CheckBoxControl.jpx` et sur OK pour ouvrir ce projet.
- 3 Choisissez Outils | Options de l'EDI pour ouvrir la boîte de dialogue Options de l'EDI. Sur la page Web, assurez-vous que l'option Copier l'URL de lancement de l'exécution/débogage Web vers le Presse-papiers est sélectionnée. Cette option copie dans le presse-papiers l'URL générée par une exécution web, pour que vous le copiez directement dans le navigateur externe.

La page Web se présente ainsi :



4 Choisissez Fichier | Tout enregistrer pour enregistrer votre travail.

Pour savoir ce que fait cette application, choisissez Projet | Construire le projet "CheckBoxControl.jpx" pour le compiler et Exécuter | Exécuter le projet pour l'exécuter. L'application démontre l'utilisation du contrôle Swing CheckBox.

Remarquez que cette application ne contient aucune opération de gestion de fichiers. S'il en contenait, nous aurions dû soit signer électroniquement le fichier JAR, soit réécrire les opérations de gestion de fichiers pour utiliser les classes de la bibliothèque JNLP. Pour plus d'informations sur Web Start et les problèmes de sécurité, voir "Observations sur les applications Java Web Start", page 17-2.

Dans l'étape suivante, vous allez utiliser l'expert Application web pour créer la WebApp de l'application.

Etape 2 : Création de la WebApp de l'application

Pour créer une WebApp, utilisez l'expert Application web. Pour plus d'informations sur les WebApps, voir Chapitre 3, "Utilisation des WebApps et des fichiers WAR".

Pour créer la WebApp de l'application,

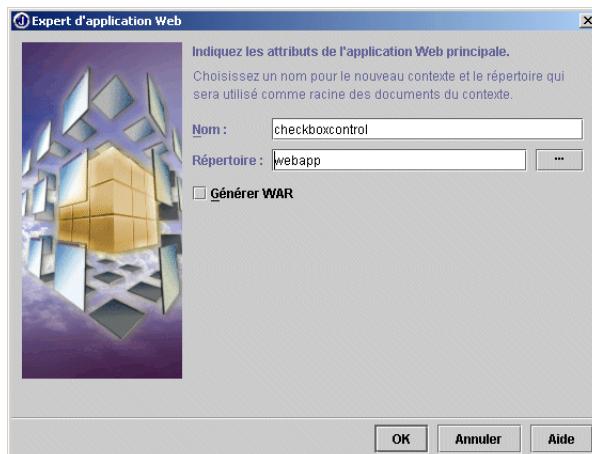
1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets. Dans la page Web, choisissez Application web et cliquez sur OK.

L'expert Application web est affiché.

2 Entrez checkboxcontrol dans le champ Nom.

- 3 Entrez webapp dans le champ Répertoire.
- 4 Dans l'expert application web, vérifiez que l'option Générer WAR n'est pas sélectionnée.

L'expert Application web doit ressembler à ceci :



- 5 Cliquez sur OK pour fermer l'expert.

La WebApp `checkboxcontrol` se présente sous la forme d'un noeud dans le volet projet. Développez ce noeud pour voir les noeuds Descripteurs de déploiement et Répertoire racine.

Dans l'étape suivante, vous allez utiliser le constructeur d'archives pour créer le fichier JAR de l'application.

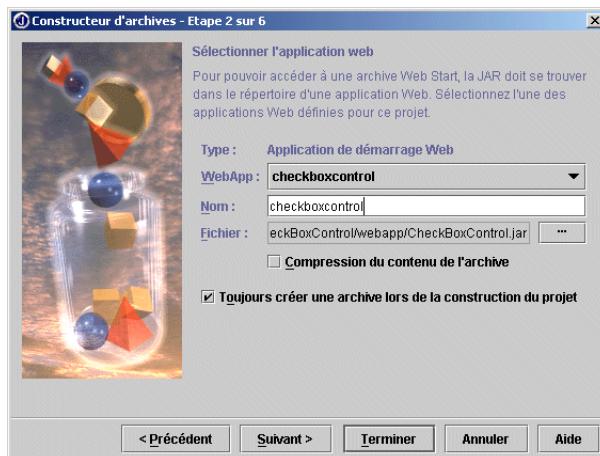
Etape 3 : Création du fichier JAR de l'application

Pour démarrer une application en tant qu'application Web Start, vous devez créer un fichier JAR. Utilisez le constructeur d'archives de JBuilder pour créer des fichiers JAR :

- 1 Si vous n'avez pas encore compilé le projet, faites-le maintenant. Choisissez Projet | Construire le projet "CheckBoxControl.jpx".
- 2 Choisissez Experts | Constructeur d'archives.
- 3 A l'étape 1 du constructeur d'archives, modifiez le type d'archive en Application de démarrage Web. Cliquez sur Suivant pour aller à l'étape 2.
- 4 A l'étape 2, `checkboxcontrol` doit être sélectionnée dans la liste déroulante WebApp.
- 5 Changez le nom en `CheckBoxControl`.

Le champ Fichier est déjà rempli. Le nom du fichier JAR est basé sur le nom du projet. Le fichier JAR est placé dans le dossier samples/Swing/CheckBoxControl/webapp de votre projet.

L'étape 2 du constructeur d'archives doit ressembler à ceci :



- 6 Cliquez sur Terminer pour créer l'archive et fermer l'expert. Vous n'avez pas à changer d'options dans les autres étapes de l'expert.
- 7 Choisissez Fichier | Tout enregistrer.
- 8 Choisissez Projet | Construire le projet "CheckBoxControl.jpx" pour créer le fichier JAR.

L'expert crée un nœud d'archive et l'affiche dans le volet projet. L'archive sera construite chaque fois que vous construisez le projet.

Dans l'étape suivante, vous allez utiliser l'expert Lanceur de démarrage Web pour créer la page d'accueil et le fichier JNLP de l'application.

Etape 4 : Création de la page d'accueil et du fichier JNLP de l'application

Dans cette étape, vous allez utiliser l'expert Lanceur de démarrage Web pour créer la page d'accueil et le fichier JNLP de l'application. La page d'accueil est le fichier HTML que vous chargez dans votre navigateur web externe. Elle contient un lien vers votre application – lorsque vous cliquez sur ce lien, le fichier JNLP commande à Java Web Start de lancer votre application.

Pour créer ces fichiers,

- 1 Choisissez Fichier | Nouveau pour afficher la galerie d'objets.

- 2** Dans la page Web, choisissez Lanceur de démarrage Web et cliquez sur OK.

L'expert Lanceur de démarrage Web est affiché.

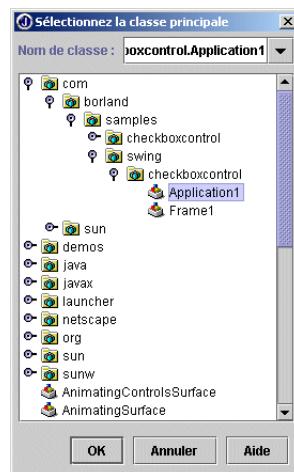
- 3** Entrez CheckBoxControlLauncher dans le champ Nom.

Cette option détermine le nom du fichier HTML et du fichier JNLP.

- 4** Choisissez checkboxcontrol dans la liste déroulante WebApp.

- 5** Choisissez le bouton Points de suspension situé à droite du champ Fichier JAR. Cela ouvre la boîte de dialogue Choisir un JAR pour WebStart où vous choisissez le nom du fichier JAR que vous avez créé avec le constructeur d'archive. C'est CheckBoxControl.jar. Il se trouve dans le répertoire CheckBoxControl/webapp. Sélectionnez le fichier JAR dans la boîte de dialogue Choisir un JAR pour WebStart, puis cliquez sur OK pour la fermer.

- 6** Si le champ Classe principale n'est pas déjà rempli, cliquez sur le bouton Points de suspension situé à droite du champ. Cela affiche la boîte de dialogue Sélectionnez la classe principale. Développez le dossier com, en haut de la boîte de dialogue, pour choisir com.borland.samples.swing.checkboxcontrol.Application1. La boîte de dialogue Sélectionnez la classe principale ressemble à ceci :



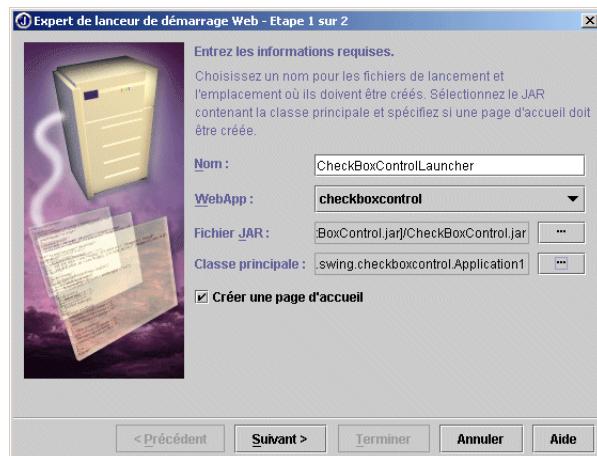
- 7** Cliquez sur OK pour fermer la boîte de dialogue.

- 8** Dans l'expert Lanceur de démarrage Web, vérifiez que l'option Créer une page d'accueil est cochée. Cette option crée le fichier HTML qui lance l'application.

Attention

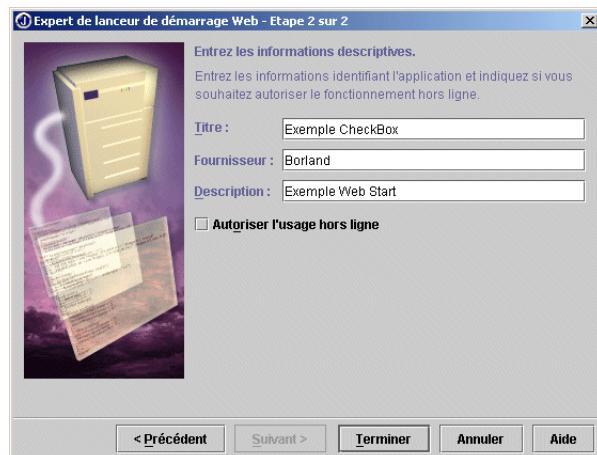
Si le nom saisi dans le champ Nom correspond au nom d'un fichier HTML ou JNLP existant dans votre projet, il vous sera demandé si vous voulez écraser ce fichier.

L'étape 1 de l'expert Lanceur de démarrage Web doit ressembler à ceci :



9 Cliquez sur Suivant pour passer à l'étape 2 de l'expert.

10 Tapez Exemple CheckBox dans le champ Titre. Entrez Borland dans le champ Fournisseur et Exemple Web Start dans le champ Description. Vérifiez que l'option Autoriser l'usage hors ligne n'est pas sélectionnée.
L'étape 2 de l'expert Lanceur de démarrage Web doit ressembler à ceci :



11 Cliquez sur Terminer.

12 Choisissez Fichier | Tout enregistrer pour enregistrer votre travail.

L'expert crée un fichier HTML et un fichier JNLP appelés CheckBoxControlLauncher et les place dans le dossier webapp de votre projet, c'est-à-dire dans la WebApp de l'application. Pour voir ces fichiers dans le volet projet, développez le noeud Répertoire racine du noeud WebApp checkboxcontrol.

Remarque Le Répertoire racine fait référence au répertoire racine de votre WebApp, c'est-à-dire au répertoire webapp.

Le volet projet doit ressembler à ceci :



Remarque Vous pouvez ouvrir ces fichiers dans l'éditeur, mais non les modifier.

Dans l'étape suivante, vous allez démarrer le serveur web et lancer votre application avec Web Start.

Etape 5 : Lancement de l'application

Cette étape vous indique comment lancer votre application avec Web Start. Pour ce faire,

- 1 Cliquez avec le bouton droit sur CheckBoxControlLauncher.html dans le volet projet et choisissez Exécution Web.

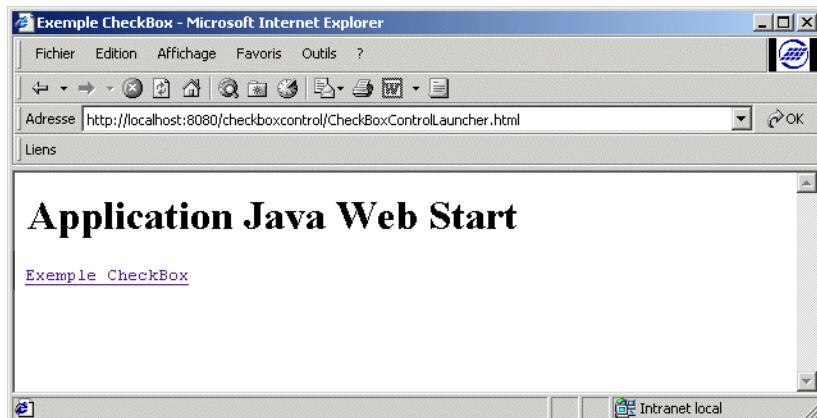
JBuilder compile les fichiers et démarre le serveur web Tomcat. Comme le fichier JNLP spécifie que cette application doit être exécuté avec Web Start, JBuilder affiche un message d'avertissement dans la vue web. La vue web se présente ainsi :



- 2 Dans votre navigateur externe, positionnez le curseur dans le champ Emplacement et appuyez sur **Ctrl+V**. Cela copie l'URL de débogage

Web depuis le presse-papiers. JBuilder a copié cette URL dans le presse-papiers, selon la sélection que vous avez effectuée dans la page Web de la boîte de dialogue Options de l'EDI. (Vous avez sélectionné cette option dans une étape précédente de ce tutoriel.) Appuyez sur Entrée pour aller à cette URL.

- 3 Votre navigateur web affiche la page d'accueil de l'application, CheckBoxControlLauncher.html. La page web contient un lien vers votre application.



- 4 Cliquez sur le lien dans la page web. Java Web Start charge et lance l'application. L'écran d'accueil affiche les informations que vous avez saisies dans l'expert Lanceur de démarrage Web.
- 5 Pour quitter l'application exemple, choisissez Fichier | Quitter. Pour arrêter le serveur web, cliquez sur le bouton Réinitialiser le programme sur l'onglet Serveur web.

Félicitations ! L'application s'exécute désormais à partir du lien du navigateur web externe. Dans ce tutoriel, vous avez appris comment définir un projet pour une application Web Start, comment utiliser l'expert Lanceur de démarrage Web pour créer la page d'accueil et le fichier JNLP de l'application, et comment démarrer l'application avec Web Start.

Index

A

activation des commandes web 15-8
affectation des servlets 6-6, 15-3, 15-9
API des servlets 5-4
API JSP
 balise comment 9-2
 balise declaration 9-2
 balise expression 9-2
 balise getProperty 9-2
 balise scriptlet 9-2
 balise setProperty 9-2
 balise useBean 9-2
 directive page 9-2
 directive taglib 9-2
applet, balise 4-2
 attributs 4-3
 erreurs 4-4
applet, déploiement 4-9, 4-14
 dans des archives 4-10
applet, expert 4-17
applets 4-1
 archivage 4-10
 bibliothèques tierces 4-14
 conseils 4-9
 dans un fichier WAR 3-12
 débogage 4-22
 débogage dans un navigateur 4-23
 exécution 4-20
 exécution des applets JDK 1.1.x dans JBuilder 4-22
 exécution JDK 1.2 4-22
 implémentation Java du navigateur 4-6
 machine virtuelle Java 4-2
 module de mise à jour Java 4-7
 présentation 2-1, 2-2, 4-2
 questions liées au navigateurs 4-5, 4-7
 test 4-15
 utilisation des paquets 4-10
AppletTestbed 4-21
 débogage des applets 4-22
appletviewer 4-21
 débogage des applets 4-22
Application web, expert 3-3
applications distribuées
 et applications web 2-7
applications web 1-1, 2-1, 2-7, 3-1, 15-1
 Voir aussi WebApp
 dans JBuilder
 et applications distribuées
 présentation

présentation du développement
utilisation
archive (attribut), balise applet 4-3
archive web 3-2
 Voir aussi WAR, fichier
arrêt du serveur web 14-11, 15-8
auditeur, servlet 6-1, 16-9
 interfaces 6-9
Auditeurs, page
 dans l'éditeur DD WebApp 16-9
authentification
 pour une WebApp 16-16

B

bibliothèques de balises
 InternetBeans Express 11-6
Bibliothèques de balises, page
 dans l'éditeur DD WebApp 16-12
bibliothèques tierces
 applets 4-14
Borland
 contacter 1-3
Borland Application Server 14-5
Borland Online 1-4

C

CGI (Common Gateway Interface)
 comparé aux servlets 2-3
Classes, onglet
 de Propriétés de la WebApp 3-7
code (attribut), balise applet 4-3
codebase (attribut), balise applet 4-3
collection de ressources web
 ajout à web.xml 16-5
commandes web
 activation 15-8
Common Gateway Interface (CGI)
 comparé aux servlets 2-3
compilation
 applets 4-20
 JSP 15-2
 servlets 15-2
configuration d'un serveur web 14-1
Connexion, page
 dans l'éditeur DD WebApp 16-16
contacter Borland 1-3
 groupes de discussion 1-4
 World Wide Web 1-4
contrainte de sécurité
 ajout à web.xml 16-5

control, balise InternetBeans Express 11-8
conventions de la documentation 1-6
conventions des plates-formes 1-7

D

database, balise InternetBeans Express 11-8
Database, classe
utilisation dans un servlet 11-3
utiliser dans une JSP 11-8
DataExpress
utilisation dans un servlet 11-1, 11-3
utilisation dans une JSP 11-1, 11-6
débogage
applets 4-22
applets dans un navigateur 4-23
JSP 15-13
servlets 15-13
débogage du source des JSP 15-13
Débogage Web, commande 15-1
activation 15-8
demandes client à destination du servlet 5-7
démarrage d'un serveur web 15-6
Dépendances, onglet
de Propriétés de la WebApp 3-9
déploiement
applets 4-9, 4-14
applications 17-1
fichier d'archive 16-1
fichier WAR 16-1
fichiers archive des applets 4-10
JSP 16-3
par type de fichiers 3-11
servlets 5-9, 16-2
WebApp 16-1
descripteurs de déploiement 3-1
application web 16-4
Editeur DD WebApp 16-4
édition 3-2
fichier web.xml 3-5, 16-4
informations supplémentaires 16-20
nœud d'une WebApp 3-5
pour une WebApp 3-4
spécifiques au fournisseur pour une WebApp
16-19
développement web
processus de base 2-6
différence MAJ/min
dans les applets et la balise applet 4-10

E

Editeur DD WebApp 16-4
éditeur DD WebApp
ajout d'un filtre 16-5

ajout d'un servlet 16-5
ajout d'une collection de ressources web 16-5
ajout d'une contrainte de sécurité 16-5
page Auditeurs 16-9
page Bibliothèques de balises 16-12
page Connexion 16-16
Page Descripteur de déploiement WebApp 16-6
page Environnement 16-14
page Filtres 16-7
page Pages erreur 16-13
page Références de ressource 16-15
page Références EJB 16-15
page Sécurité 16-17
page Servlets 16-10
page Types MIME 16-13
emplacements des fichiers
dans une WebApp 3-4
Environnement, page
dans l'éditeur DD WebApp 16-14
exécution
applets 4-20
JSP 15-5
servlets 15-3, 15-5
Exécution Web, commande 15-1, 15-6, 15-7, 15-8
activation 15-8
expert servlet 12-1
options 6-1
experts
applet 4-17
Application web 3-3
JSP (JavaServer Page) 9-4, 10-1, 11-6
Servlet 12-1

F

fichier d'archive
déploiement sur un serveur web 16-1
fichiers JAR
attribut archive d'applet 4-3
signature 4-12
fichiers ZIP
attribut archive d'applet 4-3
filtre
ajout à web.xml 16-5
filtre, servlet 6-1, 16-7
Filtres, page
dans l'éditeur DD WebApp 16-7

G

génération de tableaux
avec InternetBeans Express 11-6
groupes de discussion 1-5
Borland 1-4

H

height (attribut), balise applet 4-3
hspace (attribut), balise applet 4-3
HTML, servlets 6-4
HTML, servlets orientés 5-8
HTTP, servlets spécifiques 5-8

I

image, balise InternetBeans Express 11-8
importation
 application web 3-3
 fichiers dans une WebApp 3-4
installation de Web Start 17-3
InternetBeans Express 11-1
 affiche des données d'un servlet 11-3
 analyse du HTML 11-5
 bibliothèque de balises 11-6
 et JSP 11-6
 et servlets 11-3
 format du fichier de la bibliothèque de balises 11-10
 génération de tableaux 11-6
 présentation 2-1, 2-5
 tableau des balises JSP 11-8
 tableau des classes 11-2
 validation des données d'un servlet 11-5
InternetBeans Express, bibliothèque de balises
 balise control 11-8
 balise database 11-8
 balise image 11-8
 balise query 11-8
 balise submit 11-8
 balise table 11-8
internetbeans.tld, fichier 11-6
 format 11-10
 tableau des balises JSP 11-8
invocation des servlets 6-10
IxCheckBox, classe 11-2
IxComboBox, classe 11-2
IxControl, classe 11-2
 utilisation dans un servlet 11-3
 utiliser dans une JSP 11-8
IxHidden, classe 11-2
IxImage, classe 11-2
 utiliser dans une JSP 11-8
IxImageButton, classe 11-2
IxLink, classe 11-2
IxListBox, classe 11-2
IxPageProducer, classe 11-2
 servletGet(), méthode 11-3
 servletPost(), méthode 11-5
 utilisation dans un servlet 11-3
IxPassword, classe 11-2

IxPushButton, classe 11-2
IxRadioButton, classe 11-2
IxSpan, classe 11-2
IxSubmitButton, classe 11-2
 utilisation dans un servlet 11-5
 utiliser dans une JSP 11-8
IxTable, classe 11-2
 génération de tableaux 11-6
 utiliser dans une JSP 11-8
IxTextArea, classe 11-2
IxTextField, classe 11-2

J

JAR, fichiers
 inclusion dans un fichier WAR 3-12
Java Network Launching Protocol 17-1
 Voir aussi JNLP
Java Web Start 17-1
 Voir aussi Web Start
Java, module de mise à jour 4-7
JavaServer Pages 9-1
 Voir aussi JSP
JBuilder
 utiliser les WebApps 15-1
JNLP
 fichier JNLP 17-4
JSP (JavaServer Pages) 9-1
 compilation 15-2
 connexion à un JDataStore 13-1
 création dans l'expert 9-4, 10-1
 débogage 15-13
 débogage du source 15-13
 définition des paramètres d'exécution 15-9
 déploiement 16-3
 et InternetBeans Express 11-6
 et servlets 5-3
 exécution 15-5
 expert 10-1
 fonctionnalités de JBuilder 9-4
 inclusion dans un fichier WAR 3-11
 liens 9-5
 mappage des bibliothèques de balises 16-12
 orientées données 13-1
 orientés données 11-1
 présentation 2-1, 2-4
 syntaxe 9-2
 utilisation avec les JavaBeans 10-1
JSP, API
 directive page 11-6
 directive taglib 11-6
JSP, bibliothèques de balises
 InternetBeans Express 11-6
JSP, expert 9-4
 et InternetBeans Express 11-6

L

lancement, URI 15-9

M

Manifest, onglet

 de Propriétés de la WebApp 3-10
module de mise à jour Java 4-7
motif d'URL 6-6, 15-3
multithread et servlet 5-7

N

name (attribut)
 applet, balise 4-3

name, attribut
 internetbeans.tld 11-10

navigateurs
 différence d'implémentation Java 4-6
 exécution des applets 4-5
 module de mise à jour Java 4-7
 questions de support du JDK 4-5
 solutions pour exécuter des applets 4-7
nom des servlets 6-6, 15-3

O

options d'exécution web 14-9
options de vue web 14-8
orientés données
 JSP 11-1
 servlets 6-12, 11-1

P

page d'accueil de l'application
 Web Start 17-4
Page Descripteur de déploiement WebApp 16-6
Page JavaServer, expert 10-1
Pages erreur, page
 dans l'éditeur DD WebApp 16-13
paquet servlet.HTTP 5-5
paquets
 dans les applets 4-10
param (attribut), balise applet 4-3
paramètres
 applet, balise 4-3
paramètres d'exécution
 JSP 15-9
 servlets 15-9
plugin de serveur web 14-11
 configuration du serveur web 14-11
 considérations sur JSP 14-12
 création 14-10
 démarrage du serveur web 14-11

éditeur GUI 14-12

recensement comme OpenTool 14-11
polices

 conventions de la documentation 1-6
projets

 configuration pour Web Start 17-3

propriétés

 d'un fichier WAR 3-11

Propriétés de la WebApp

 compression du WAR 3-6
 emplacement du WAR 3-6
 génération du WAR 3-6
 nom du WAR 3-6
 onglet Classes 3-7
 onglet Dépendances 3-9
 onglet Manifest 3-10
 onglet WebApp 3-6
 répertoire 3-6
 types de fichiers inclus 3-6

Q

query, balise InternetBeans Express 11-8

QueryDataSet, classe

 utilisation dans un servlet 11-3
 utiliser dans une JSP 11-8

R

Références de ressource, page
 dans l'éditeur DD WebApp 16-15

Références EJB, page
 dans l'éditeur DD WebApp 16-15

répertoire racine
 d'une WebApp 3-4
required, attribut
 internetbeans.tld 11-10
ressources en ligne 1-4
rtexprvalue, attribut:internetbeans.tld 11-10

S

sas de sécurité

 sécurité des applets 4-11
 sécurité des applications Web Start 17-2

sécurité
 applets 4-12
 gestionnaire de sécurité 4-11
 pour une application Web Start 17-2
 pour une WebApp 16-17
 restrictions des applets 4-12
 sas de sécurité 4-11
 signature des applets 4-12
sécurité des applets
 gestionnaire de sécurité 4-2, 4-11
 restrictions 4-12

sas de sécurité 4-11
signature 4-12
solutions 4-12
Sécurité, page
dans l'éditeur DD WebApp 16-17
serveurs web 5-3
arrêt 15-8
configuration 14-1, 14-7
création de plugin pour 14-10
démarrage 15-6
options d'exécution web 14-9
options de vue web 14-8
sortie brute 15-8
sortie formatée 15-7
Tomcat 5-3, 14-2
WebLogic 14-5
servlet standard 6-1
fichier SHTML 6-6
ServletContext 3-4
servletGet(), méthode 11-3
servletPost(), méthode 11-5
servlets 5-1
ajout à web.xml 16-5
auditeurs 6-1, 6-9, 16-9
comparés à CGI (Common Gateway Interface)
 2-3, 5-1
cycle de vie 5-6
démarrage 5-6
déploiement 5-9, 16-2
destruction 5-7
et InternetBeans Express 6-12, 11-3
et JSP 5-3
et serveurs web 5-3
et URI 15-3
et URL 6-6, 15-3
et WebApp 6-1, 6-6, 15-3
exemples d'utilisation 5-9
filtre 6-1, 16-7
inclusion dans un fichier WAR 3-11
initialisation 5-6
internationalisation 6-12
invocation 6-10, 6-11
mappage 6-6, 15-3, 15-9
motif d'URL 6-6, 15-3
multithreads 5-7
nom 15-3
orientés données 11-1
orientés HTML 5-8
présentation 2-1, 2-3
serveurs web 5-1
ServletContext 3-4
spécifiques à HTTP 5-8
standard 6-1, 16-10
traitement des demandes client 5-7
transmission des réponses 5-7
servlets dans JBuilder 6-1
à thread unique 6-1
compilation 15-2
création avec l'expert 6-1
création des paramètres 6-8
débogage 15-13
définition des paramètres d'exécution 15-9
définition des propriétés d'exécution 15-12
exécution 15-3, 15-5
fichier SHTML 6-6
nom 6-6
orientés données 6-12
redéfinition des méthodes standard des servlets
 6-5
type de contenu 6-4
Servlets, page
dans l'éditeur DD WebApp 16-10
SHTML, fichier 6-6
signature des applets 4-12
source vue web 15-8
submit, balise InternetBeans Express 11-8
support aux développeurs 1-3
support Java, navigateurs 4-5
support technique 1-3

T

table, balise InternetBeans Express 11-8
tableaux
 génération avec InternetBeans Express 11-6
tagclass, attribut
 internetbeans.tld 11-10
technologies internet
 tableau 2-1
technologies web
 tableau 2-1
test
 applets 4-15
thread unique et servlet 6-1
Tomcat 5-3
 configuration 14-2
 utilisation avec JSP 10-1
tutoriels
 JSP (JavaServer Page) 10-1
 JSP utilisant InternetBeans 13-1
 servlet guestbook 8-1
 servlet simple 7-1
 servlet utilisant des InternetBeans 12-1
tutoriels InternetBeans Express
 JSP 13-1
 servlet 12-1
tutoriels JSP (JavaServer Pages)
 utilisation des InternetBeans 13-1
tutoriels servlet
 guestbook 8-1

hello world 7-1
utilisation des InternetBeans 12-1
types de fichiers
 inclus dans un fichier WAR 3-11
Types MIME, page
 dans l'éditeur DD WebApp 16-13

U

URI de lancement 15-9
URI et servlets 15-3
URL et servlets 6-6, 15-3
URL générée 15-7
Usenet, groupes de discussion 1-5

V

vspace (attribut), balise applet 4-3
vue web 15-7

W

WAR (archive web)
 compression 3-6
 création 3-6
 définition de l'emplacement 3-6
 définition du nom 3-6
WAR, fichier (archive web) 3-2
 ajout d'applets 3-12
 ajout de fichiers JAR 3-12
 définition 3-11
 déploiement 16-1
 génération 3-3
 outils 3-2
 propriétés 3-11
 relation avec la WebApp 3-11
 types de fichiers inclus 3-11
 visualisation du contenu 3-11
Web Start 17-1
 applet 17-4
 application 17-4
 configuration de votre projet 17-3
 et JBuilder 17-3
 expert 17-4
 fichier JAR 17-4
 fichier JNLP 17-4
 installation 17-3
 page d'accueil de l'application 17-4
 sécurité des applications 17-2
 tutoriel 17-6

web.xml
 ajout d'un filtre 16-5
 ajout d'un servlet 16-5
 ajout d'une collection de ressources web 16-5
 ajout d'une contrainte de sécurité 16-5
web.xml, fichier 3-1, 16-4
 balises auditeur 16-9
 balises de mappage des filtres 16-7
 balises servlet 16-10
 balises taglib 16-12
 bibliothèques de balises 16-12
 contraintes de sécurité 16-17
 création 3-5
 édition 3-2, 16-4
 entrées d'environnement 16-14
 informations supplémentaires 16-20
 mappage des pages erreur 16-13
 mappage des types MIME 16-13
 méthode d'authentification 16-16
 références EJB 16-15
 ressources 16-15

WebApp
 dans JBuilder 15-1
 déploiement 16-1
 descripteurs de déploiement 3-4, 16-4
 descripteurs de déploiement, spécifiques au fournisseur 16-19
 éditeur de descripteur de déploiement 16-4
 emplacements des fichiers 3-4
 expert 3-3
 importation dans JBuilder 3-3
 importation de fichiers 3-4
 outils 3-2
 répertoire racine 3-4
 répertoire WEB-INF 3-4
 structure 3-1
 test 16-4
 utilisation 15-1
WebApp, nœud 3-4
WEB-INF, répertoire 3-4
WebLogic 14-5
 descripteur de déploiement 16-4
 fichier weblogic.xml 16-4
width (attribut), balise applet 4-3
WML, servlets 6-4

X

XHTML, servlets 6-4
XML, servlets 6-4