JBoss-Features und -Tools, Teil 3: Die JBoss IDE

von Marcus Redeker

Plugin for JBoss

In den ersten beiden Teilen unserer Artikelserie über den JBoss Application Server haben wir uns mit seinen Features und deren Konfiguration befasst. In diesem Teil nehmen wir weniger den eigentlichen Application Server als die JBoss IDE – ein Eclipse-Plugin – in den Blick.

Wie bereits im *Java Magazin* 1.2004 [1] kurz vorgestellt, handelt es sich bei der JBoss IDE [2] um ein Plugin für die Eclipse-Plattform [3], welches die Erstellung von J2EE-Projekten vereinfacht. Das Projekt startete Mitte 2002 als XDoclet-Plugin und wurde in die Projekte von JBoss übernommen, nachdem sich der Initiator Hans Dockter mit dem CEO der JBoss Group [4], Marc Fleury über die Notwendigkeit einer JBoss IDE geeinigt hatte.

Die aktuelle Version 1.2.2 des Plugins kann sowohl in der momentan freigegebenen Eclipse 2.1.x-Serie als auch in der sich in Vorbreitung befindlichen Version 3 von Eclipse eingesetzt werden. Die wichtigsten Features, die die JBoss IDE momentan bietet, sind:

- Konfiguration, Überwachung und Steuerung mehrerer JBoss-Instanzen
- eine sehr komfortable Unterstützung von XDoclet [5] über den Code-Assistenten von Eclipse inklusive Generierung der benötigten Ant-Skripte zum Starten von XDoclet
- Wizard-Unterstützung für das Zusammenpacken von J2EE-Anwendungen in die entsprechenden Archive
- Deployment der erzeugten J2EE-Anwendungsarchive auf die konfigurierten JBoss-Instanzen

Installation

Seit dem Release der aktuellen Version 1.2.2 wird die JBoss IDE nicht mehr in Form eines dowloadbaren Archivs angeboten, das entpackt werden muss, sondern die Installation geschieht direkt über die Install/Update-Funktion von Eclipse. Der Aufruf dieser Funktion ist in den beiden Eclipse-Versionen 2.1.x und 3.0.x etwas unterschiedlich und wird in dem unter [2] angebotenen Installationshandbuch genau beschrieben. In diesem Artikel gehen wir nur auf die kommende Version 3.0.x von Eclipse ein, die derzeit in der Version 3.0M6 verfügbar ist. Dort wird nach dem Start von Eclipse der Menüpunkt HELP | SOFTWARE UPDATES | FIND AND INSTALL… ausgewählt und dann der Punkt SEARCH FOR NEW FEATURES TO INSTALL benutzt. Anschließend gibt man über den Button ADD UPDATE SITE die folgende URL ein: jboss.sourceforge.net/jbosside/updates/. Nun muss nur noch der neu angelegte Eintrag aufgeklappt, die Version für Eclipse 3.0 ausgewählt und *FINISH* gedrückt werden (Abb. 1). Eclipse lädt dann die entsprechenden Pakete von der angegebenen Adresse und installiert sie. Nach der Installation ist ein Neustart von Eclipse erforderlich.

Wenn sich der eigene Computer hinter einem Proxy befindet, müssen gegebenenfalls unter WINDOW | PREFERENCES | INSTALL/UPDATE der Punkt ENABLE HTTP PROXY CONNECTION aktiviert, die Host-Adresse und der Port des Proxys eingetragen werden.

Managen mehrerer JBoss-Instanzen

Die JBoss IDE liefert zwei Eclipse Views, die einem beim Managen von JBoss-Instanzen unterstützen. Die Server Navigator View listet alle konfigurierten JBoss-Instanzen und man kann diese von dort aus starten, stoppen oder auch killen. Anhand eines kleinen roten bzw. grünen Punktes erkennt man, ob der jeweilige JBoss gestartet ist (Abb. 2).

Der Server Navigator wird über das Menü WINDOWS | SHOW VIEW | OTHER | JBOSS IDE | SERVER NAVIGATOR aufgeru-

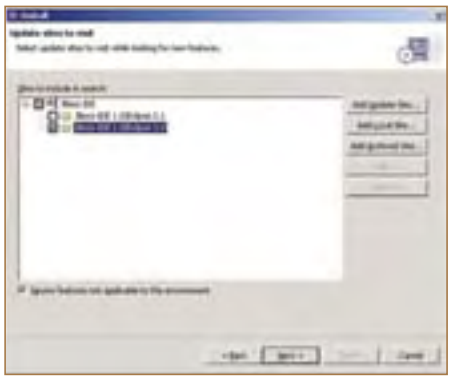


Abb. 1: Eclipse-Install/Update-Funktion

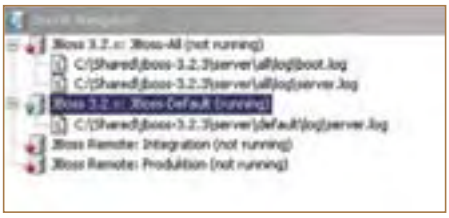
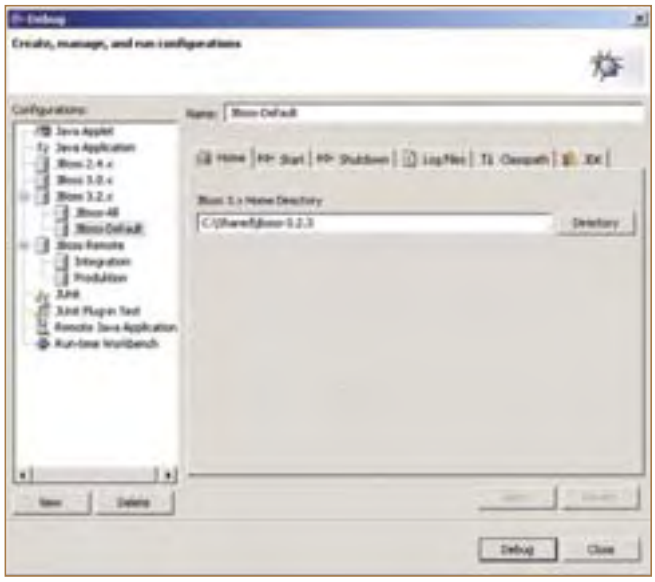


Abb. 2: JBoss IDE – Server Navigator View

Abb. 3: JBoss IDE – Server-Konfiguration



fen. Um die einzelnen Serverinstanzen zu konfigurieren, kann entweder der Menüpunkt CONFIGURATION aus dem Kontextmenü des Server Navigator benutzt werden oder in der Java-Perspektive der Menüpunkt RUN | DEBUG... Der dann erscheinende Assistent zeigt vier zusätzliche JBoss-spezifische Konfigurationen, über die man seine JBoss-Instanzen anlegt (Abb. 3). Alle dort angelegten JBoss-Instanzen erscheinen automatisch im Server Navigator und können von dort gemanagt werden. Es können beliebig viele Instanzen konfiguriert werden. Wenn eine Instanz gestartet wird, dann läuft diese automatisch immer im Debug-Modus und es können Servlets und EJBs entsprechend mit Breakpunkten versehen und gedebugt werden.

Neben der Server Navigator View gibt es noch den Log File Viewer, der die einem Server zugeordneten Logfiles anzeigt und in dem in der Konfiguration angegebenen Intervall updated. Der Log File Viewer wird über das Menü WINDOWS | SHOW VIEW | OTHER | JBOSS IDE | LOG FILE VIEWER aufgerufen.

XDoclet-Unterstützung

Ein Feature, das viele XDoclet-Benutzer sicherlich interessiert, ist der Code-Assistent der JBoss IDE zur XDoclet-Tag-Autovervollständigung. Dieser wird durch Drücken von CTRL + SPACE aktiviert, wenn sich der Cursor innerhalb des Java-Editors in einem Javadoc-Kommentar befindet. Neben den üblichen Vorschlägen für Javadoc-Tags und Eclipse Javadoc-Templates zeigt die JBoss IDE zusätzlich XDoclet-spezifische Tags und XDoclet-Templates an (Abb. 4). Wenn sich der Cursor innerhalb eines gültigen XDoclet-Tags befindet, werden automatisch die für dieses Tag gültigen Attribute zur Autovervollständigung angeboten. Der XDoclet Code-Assistent analysiert zudem den Kontext, in dem sich der Cursor gerade befindet. So werden z.B. für ein Servlet andere Tags als für EJB angezeigt und die Tags für den Kommentar einer Klasse unterscheiden sich von denen einer Methode. Über das Menü WINDOWS | PREFERENCES | JBOSS IDE | XDOCLET | CODE ASSIST | TEMPLATES lassen sich eigene Templates definieren.

XDoclet wird gestartet, indem man Ant mit einem Buildscript aufruft, welches entsprechende XDoclet Targets definiert. Durch die Vielzahl an Optionen und unterschiedlichen Doclets, die von XDoclet angeboten werden, ist das Erstellen dieses Buildscripts nicht immer ganz einfach. Daher unterstützt uns die JBoss IDE auch an dieser Stelle, indem sie einen GUI-Dialog zur Erstellung von XDoclet-Konfigurationen anbietet (Abb. 5). Aufgerufen wird dieser Dialog über das Kontextmenü eines Projektes und dort der Menüpunkt PROPERTIES | XDOCLET CONFIGURATIONS.

Konfigurationen können über das Kontextmenü der oberen Liste hinzugefügt oder gelöscht werden. Die JBoss IDE liefert bereits eine Liste vordefinierter Standardkonfigurationen, die ebenfalls über das Kontextmenü ausgewählt werden. Wurde eine Konfiguration hinzugefügt, wird über das Kontextmenü der unteren Liste festgelegt, welche Doclets benutzt werden sollen. Nach Auswahl eines Doclets, z.B. *ejbdoclet*, werden wiederum über das Kontextmenü die einzelnen Tags

Abb. 4: JBoss IDE – XDoclet Code-Assistent

für dieses Doclet konfiguriert und in der rechten Tabelle können die Attribute für ein ausgewähltes Tag angegeben werden. Wird dieser Dialog mit OK beendet oder drückt man während seiner Arbeit auf APPLY, erzeugt die JBoss IDE eine Datei mit dem Namen *xdoclet-build.xml* im ausgewählten Projekt. Das Auswählen des Eintrags RUN XDOCLET im Kontextmenü des Projektes führt dann dazu, dass Ant mit genau dieser Datei als Buildscript gestartet wird und somit alle definierten XDoclet Tasks ausgeführt werden.

J2EE-Archiv-Erzeugung

Ein weiteres Feature ist die Unterstützung bei der Erzeugung von J2EE-kompatiblen Anwendungsarchiven. Hierfür gibt es, wie bei der XDoclet-Unterstützung auch, einen Dialog (Abb. 6), der als Ergebnis ein Ant Buildscript liefert. Die erzeugte Datei bekommt den Namen *packaging-build.xml* und landet ebenfalls im ausgewählten Projekt. Über den Menüpunkt RUN PACKAGING des Projekt-Kontextmenüs wird dann wiederum Ant gestartet und als Buildfile wird diese Datei benutzt. Aufgerufen wird der Dialog zur Definition von Archiv-Konfigurationen mit dem Menüpunkt PROPERTIES | PACKAGING CONFIGURATIONS aus dem Kontextmenü des Projektes. Über das Kontextmenü der Liste in dem Dialog werden entsprechende Konfigurationen angelegt oder man kann eine Konfiguration für eines der vordefinierten Standardarchive benutzen.

Deployment-Support

Mithilfe der Deployment-Unterstützung der JBoss IDE lassen sich beliebige Ressourcen eines Projektes deployen, redeployen oder undeployen. Aufgerufen wird diese Funktion über das Kontextmenü der Ressource, die man deployen möchte. Über den Menüpunkt DEPLOYMENT | DEPLOY TO... kann dann aus der Liste der definierten Deployment-Ziele eines ausgewählt werden und die Ressource wird dorthin kopiert. Eine erfolgreich deployte Ressource wird anhand eines kleinen grünen Pfeils dargestellt. Wurde eine Ressource einmal deployt, merkt sich die JBoss IDE innerhalb einer Eclipse-Sitzung das Ziel und ein Redeployment bzw. Undeployment sind ohne ein nochmaliges Nachfragen möglich.

Alle lokalen JBoss-Instanzen, die im Server Navigator definiert sind, werden automatisch als Deployment-Ziele angezeigt. Darüber hinaus lassen sich über den Menüpunkt WINDOW | PREFERENCES | JBOSS IDE | DEPLOYER weitere Ziele definieren.

Fazit

Die Kombination von Eclipse und JBoss IDE ist eine kostengünstige Alternative für eine Java-Entwicklungsumgebung mit J2EE-Unterstützung. Durch die Benutzung von Quasi-Standards wie Ant und XDoclet bietet sich dieses Plugin allerdings nicht nur für JBoss-Benutzer an, sondern ist auch für die Entwicklung von J2EE-Anwendungen geeignet, die nicht auf JBoss installiert werden. Die Entwickler haben auch bereits eine Unterstützung von Templates für z.B. EJBs und zusätzliche Wizards angekündigt. Man kann also auf weitere interessante Features gespannt sein.

Unter [2] findet man neben dem erwähnten Installationshandbuch auch noch ein ausführliches Tutorial, welches anhand eines kleinen J2EE-Projektes auf alle Funktionen der JBoss IDE eingeht. Zu einem späteren Zeitpunkt zeigen wir die Unterstützung von JBoss in der Entwicklungsumgebung JBuilder von Borland. ■

Links & Literatur

[1] Wolfgang Korn: Eingesteckt. Eclipse-Plugins zur Entwicklung von J2EE-Anwendungen, *Java Magazin* 1.2004

[2] JBoss IDE: www.jboss.org/developers/projects/jboss/jbosside.jsp

[3] Eclipse: www.eclipse.org/

[4] JBoss Group: www.jbossgroup.com/

[5] XDoclet: xdoclet.sourceforge.net/

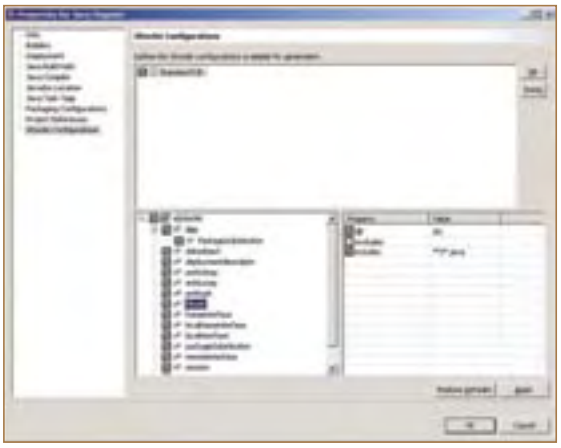


Abb. 5: JBoss IDE – XDoclet-Konfiguration

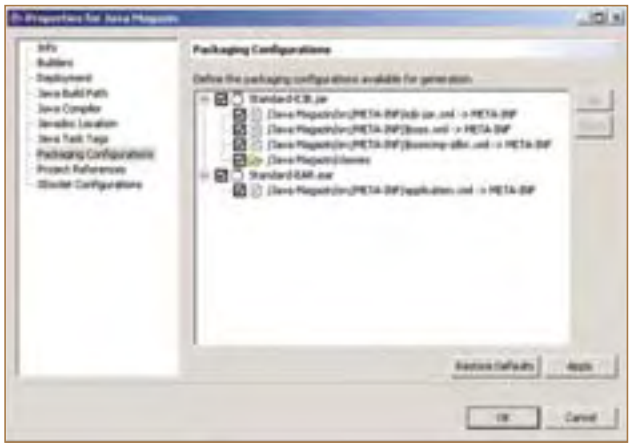


Abb. 6: JBoss IDE – Archiv-Konfiguration