JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 5, July - August 2005

The JBoss Integration Plug-in for the IntelliJ IDEA, Part 1.

Douglas Lyon, Fairfield University, Fairfield CT, U.S.A.
Martin Fuhrer, President of Furher Engineering AG, Biel , Switzerland
Thomas Rowland, Pitney Bowes, Shelton CT, U.S.A.

The man who claims
to be the boss in his own home
will lie about other things as well.
– Amish saying

Abstract

This paper describes the use of a new plug-in that eases the integration of a popular IDE, called IntelliJ with a popular open-source application server, called JBoss.
The IntelliJ IDEA is an Integrated Development Environment (IDE) used for Java development. It is known for its strong refactoring capabilities. It is a closed-source, proprietary product, which is used in both educational and industrial settings. IntelliJ functionality is extended by a set of open APIs that third-party developers can use to integrate their solutions by the development of plug-ins.
This paper describes the installation and use of the JBoss plug-in with the IntelliJ IDE. The JBoss integration plug-in was first conceived and implemented by Martin Fuhrer at Fuhrer Engineering.

1. THE DISTRIBUTED COMPUTING MODEL

It is typical for enterprise systems to be run on multiple computers. For example, a web server, an application server and a development machine are often run on three different computers.
The web server's primary goal is to quickly serve static data (HTML, images, audio, etc.) to a browser, over the Internet, using HTTP (Hypertext Transfer Protocol). This is typically done in a stop-and-wait protocol, and as such, throughput during the waiting period drops to zero while a request is processed. Thus, there is a distinct advantage to minimizing processing time for HTTP requests in order to provide a positive experience for the web client.

The application server's primary goal is to run programs that create dynamic data (servlets, JSP's, EJB's, etc.). This is typically done using a framework based on a large, and sometimes computationally cumbersome, API. Part of the API may be sourced from a vendor (like Sun), but often the API contains custom business logic that was authored

Cite this column as follows: D. Lyon, M. Fuhrer and T. Rowland: "The JBoss Integration Plug-in for IntelliJ IDEA, Part1", in *Journal of Object Technology*, vol. 4, no. 5, July-August 2005, pp. 7-17
http://www.jot.fm/issues/issue_2005_7/column1

by in-house programmers. As a result, the speed of execution of methods in the application server, as well as their reliability, may not be at the same standard as a more widely available API. Typically, systems run daemon tasks in order to handle requests for services. They are often started by a single task that runs multiple threads, thus enabling fast context switching (due, in part, to the shared memory architecture used by multi-threaded systems). Even so, a few slow threads can make a task, and even a whole computer system, seem sluggish.

The development machine is generally under direct control of a programmer. During the process of development, machines can crash, or be deliberately rebooted. Bugs can be introduced into code that cause slow-downs and the development environment itself can cause a major computational load.

The environment of having one-programmer with one application server may prove to be increasingly rare in an industrial setting. Frequently, there will be multiple programmers and several application servers, some of which are development machines, and others that are live (i.e., production machines). This is illustrated in Figure 1.

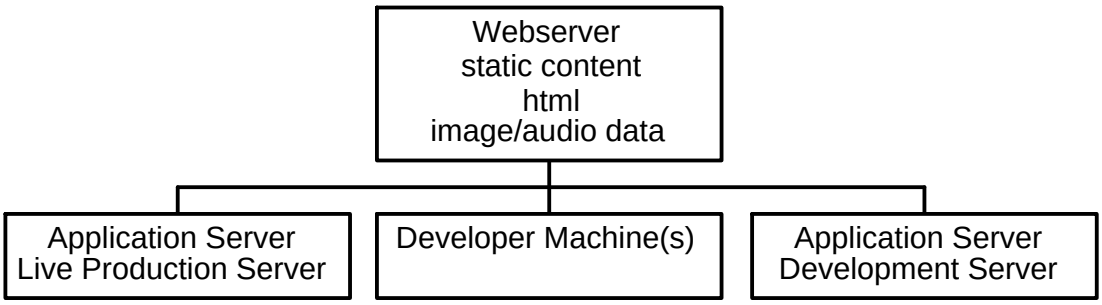

Figure 1. A Simplified Distributed Enterprise Architecture

Figure 1 depicts an oversimplification of an enterprise architecture. There are frequently large repositories of company proprietary data in relational database management systems (RDBMS) as well as firewalls, perimeter networks, bastion (i.e., DMZ) machines, etc.

One of the basic questions is “why JBoss?” There are many application severs on the market, and JBoss is just one of many, as documented in [Lyon 2004]. Some of them are even free and open-source. However, JBoss has doubled in popularity between 2002 and 2003, taking over 26% of the application server market [BUSINESS WIRE 2004].

The remainder of this article will address how to install the JBoss plug-in, and create an IntelliJ IDEA project using JBoss as the application server.

2. THE JBOSS PLUG-IN

This section describes how to download and install the JBoss plug-in. From the IntelliJ IDE, select the *File:Settings* menu item, then select the ***Plugins*** icon, as shown in Figure 2.1.

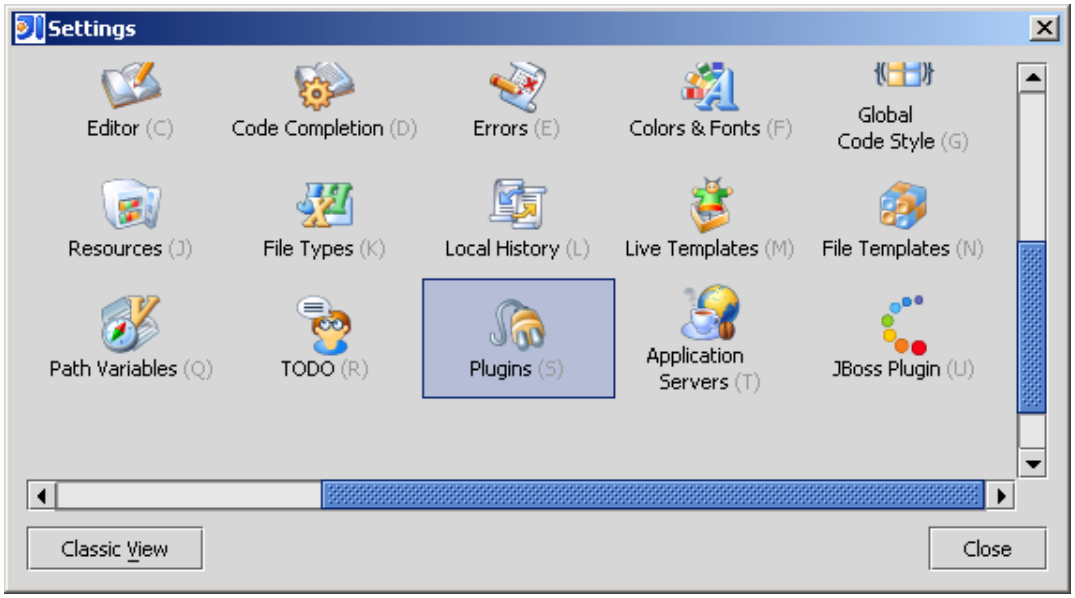

Figure 2.1 Accessing the plug-ins

The *Plugins* dialog displays two tabs. Select the *Available* tab and accept the prompt to download latest repository information. Sort the list of plug-ins by category, proceed to the J2EE category and select *JBoss* as shown in Figure 2.2. Right-click and Select *Download and Install Plugin.*

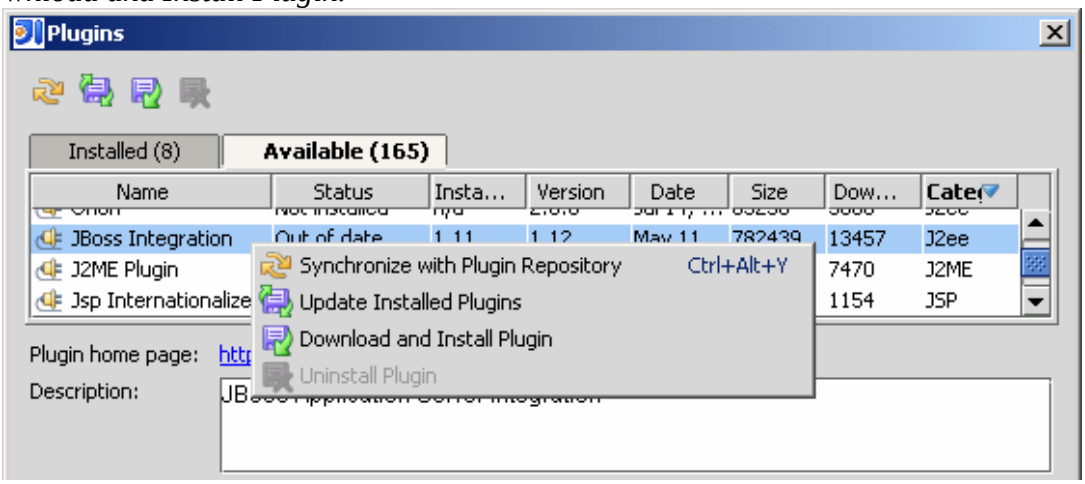

Figure 2.2 Selecting the JBoss plug-in for download and install

Accept the prompt to install the JBoss plug-in. A dialog will appear that says “you need to shut down the IDE to activate the plug-ins”. Select *Yes* and the IntelliJ IDE will shut down. Restart the IDE and you are ready for the next section.

3. CONFIGURE INTELLIJ TO RECOGNIZE JBOSS

Select the *File:Settings* menu item to bring up the *Settings* dialog. Select the *Application Servers* icon. Select *Add* and then *JBoss Server*, as shown in Figure 3.1

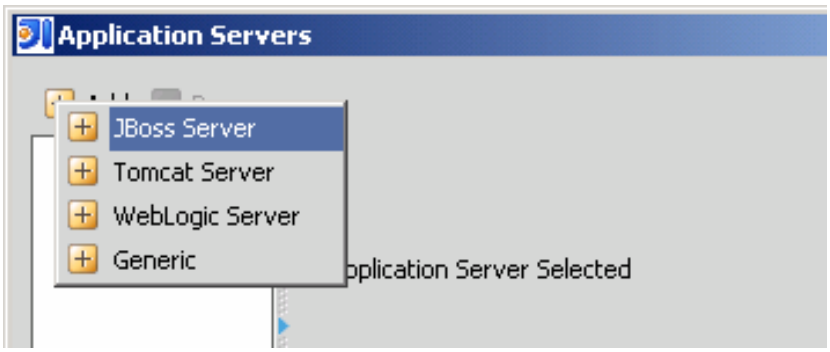

Figure 3.1 Adding a new application server to the IntelliJ IDE

Once the JBoss Server is selected, its home location must be identified to the IDE. JBoss can be run remotely on another machine; however, there are libraries that are required in order for compilation to work. These libraries must be on the local (i.e., developer’s) machine. Select the JBoss home, as shown in Figure 3.2, and select *OK*.

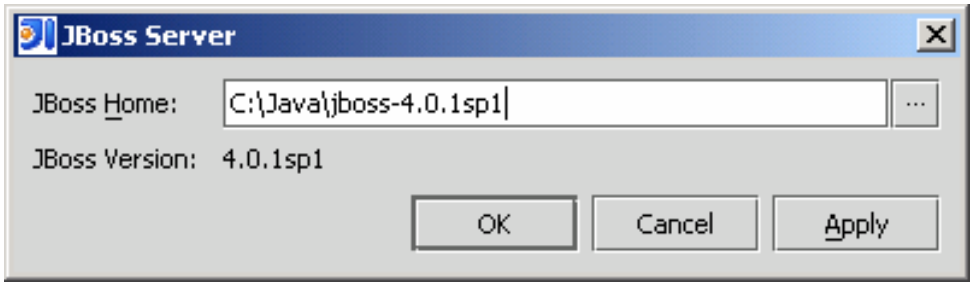

Figure 3.2 Select the JBoss Home

The *Application Servers* dialog shows the location of the libraries (needed for a successful linkage). Select *OK* to finish.

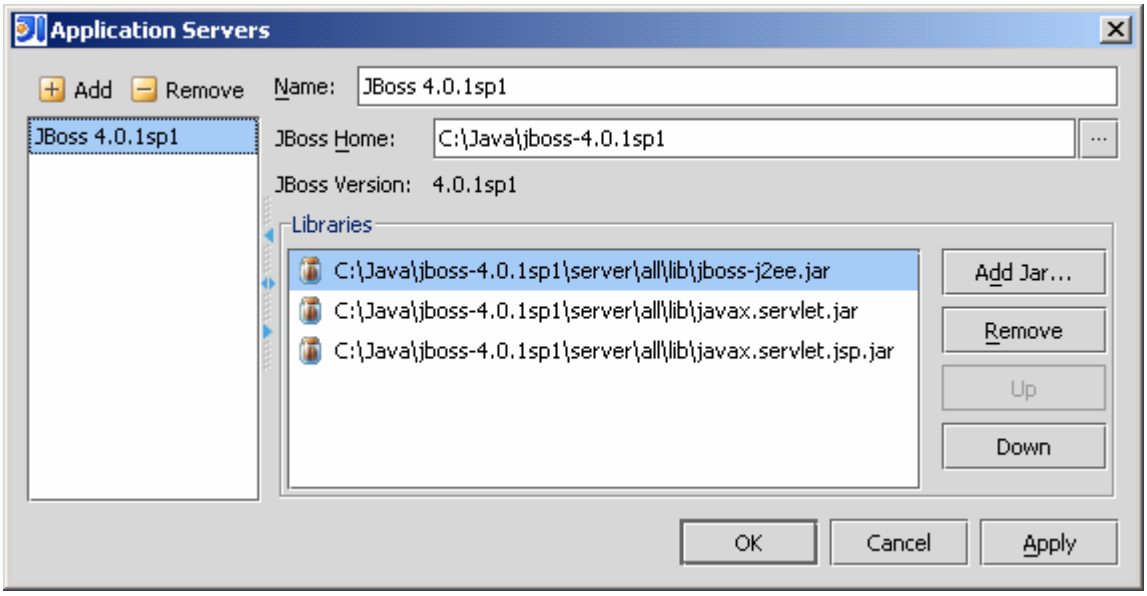

Figure 3.3 The Application Servers dialog

4. CREATING A PROJECT WITH EJB AND WEB MODULES

This section describes the procedure for creating a multi-module project that contains both an EJB and a web module. From the IntelliJ IDE, select the *File:New Project* menu item in order to bring up the *New Project* dialog. Set the project file name and location, As shown in Figure 4.1. Accept any prompts to create the project directory.

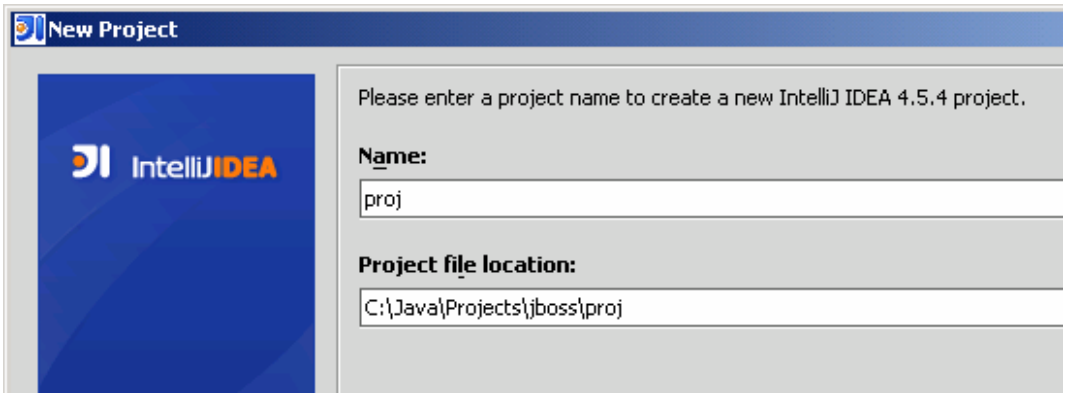

Figure 4.1 The Project Dialog Box

Select *Next*, select/configure the project JDK (not shown here) and select *Next* again. Select the *Create/configure multi-module project* radio button, as shown in Figure 4.2.

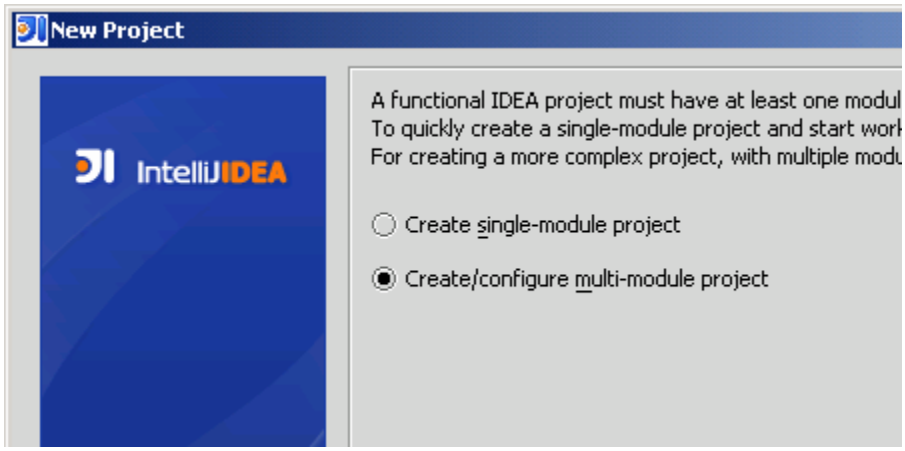

Figure 4.2 Creating a Multi-module project

Selecting *Finish* will display the *Add Module* dialog. You are now ready to add an *EJB Module*. Select *Create new module - Ejb Module* as shown in Figure 4.3.

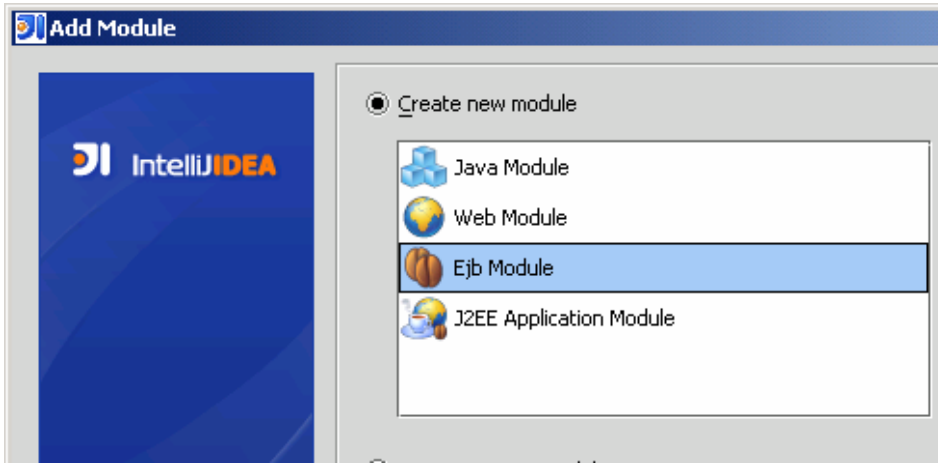

Figure 4.3 Creating a new EJB Module

Select *Next* and enter in the module name, as shown in Figure 4.4.

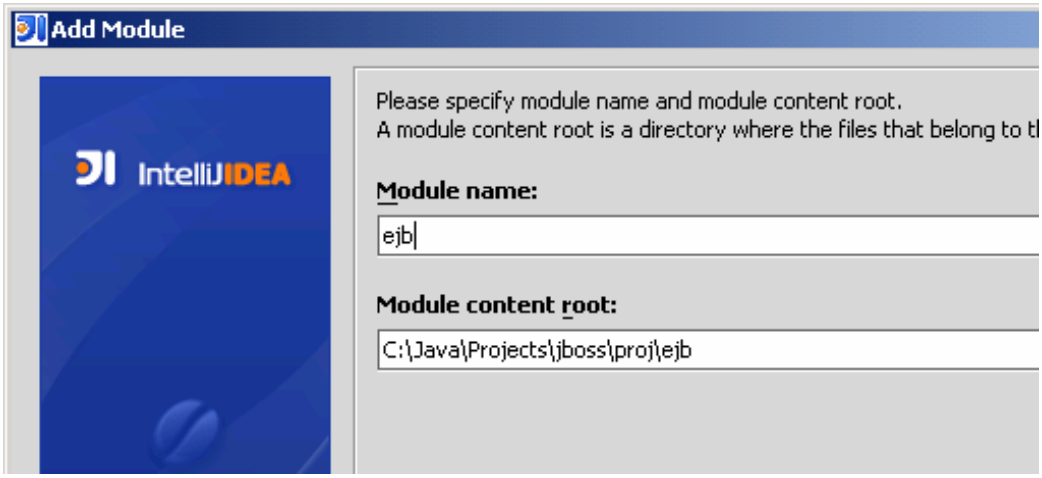

Figure 4.4 Setting the EJB Module name and location

Select *Next* through the next set of screens (the defaults should be acceptable) and then select *Finish.* This will reveal the *Paths* dialog, as shown in Figure 4.5.

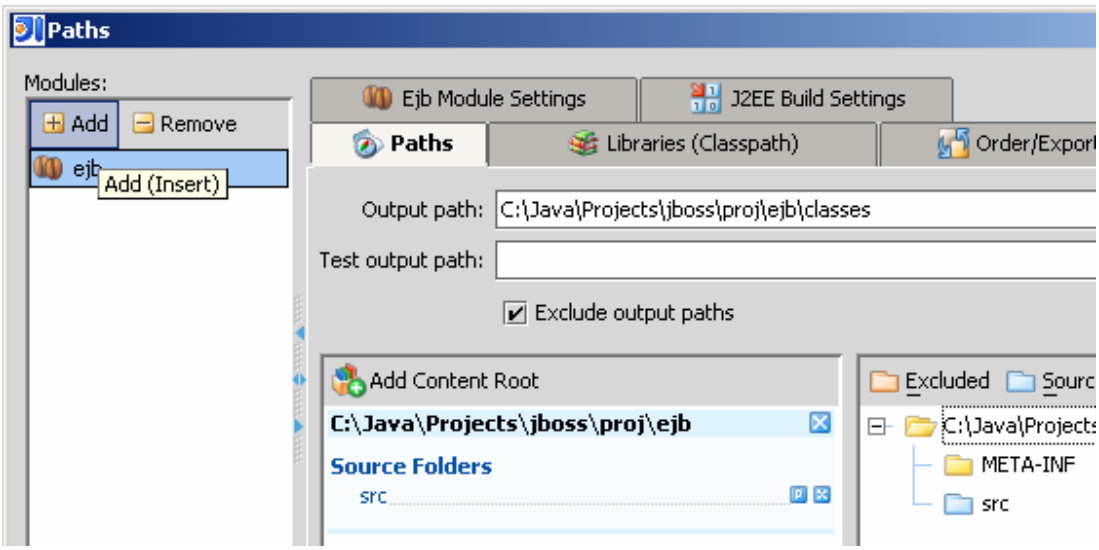

Figure 4.5 The Paths Dialog with EJB module

Select *Add* in the *Paths* dialog and select *Web Modules* as shown in Figure 4.6.

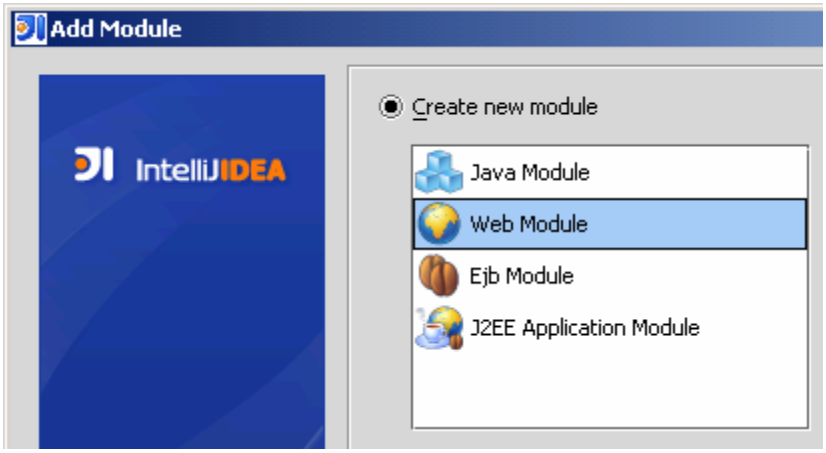

Figure 4.6 Creating a New Web Module

Select *Next* and enter in the module name, as shown in Figure 4.7.

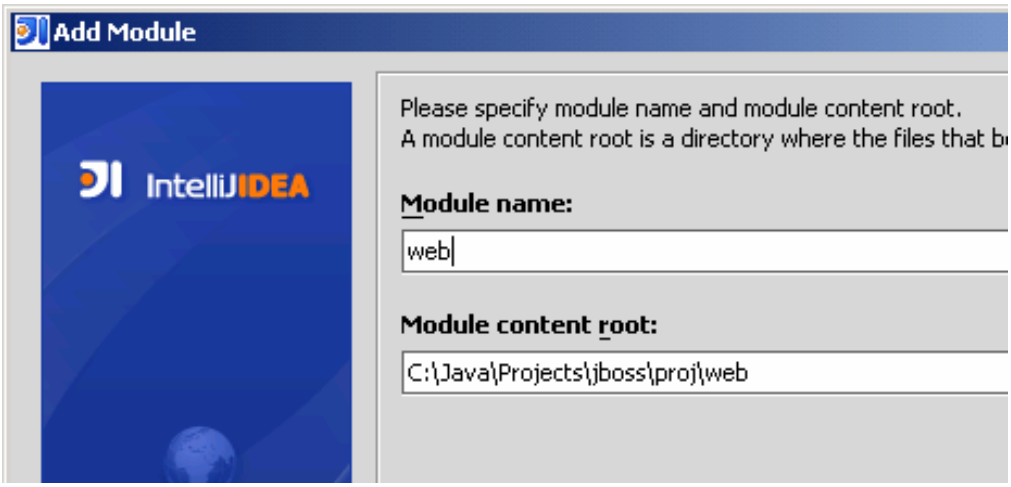

Figure 4.7 Setting the Web Module name and location

Select *Next* through the next set of screens (the defaults, as they appear, are typically reasonable), and then select *Finish*. This will return you to the *Paths* dialog, as shown in Figure 4.8.

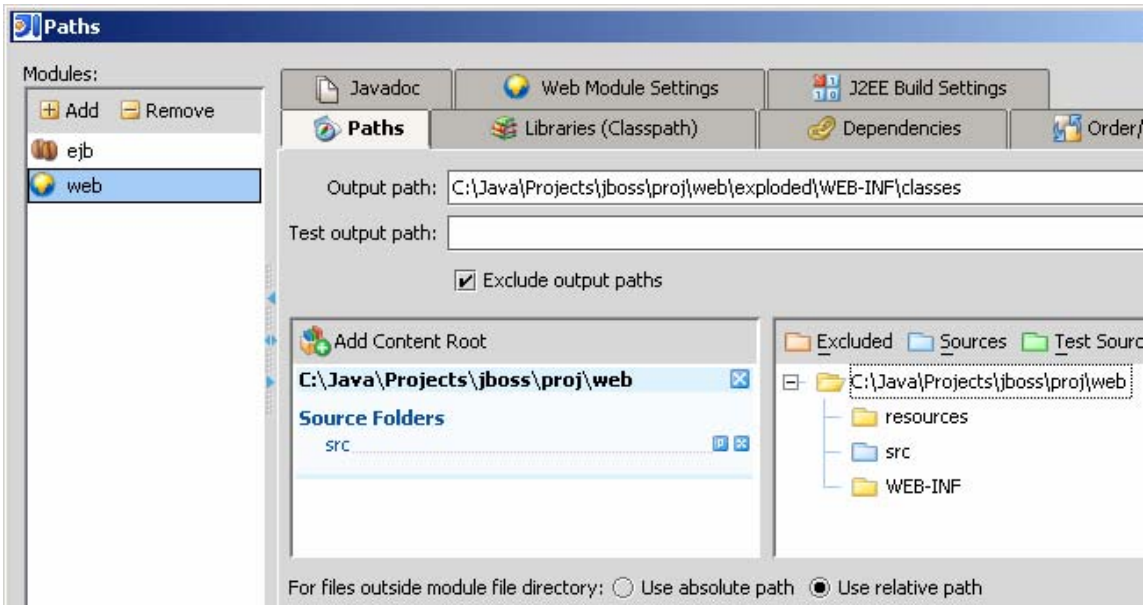

Figure 4.8 The Paths dialog with EJB and Web modules

With the Web module selected, select the *Dependencies* tab and check the checkbox labeled *ejb*, as shown in Figure 4.9

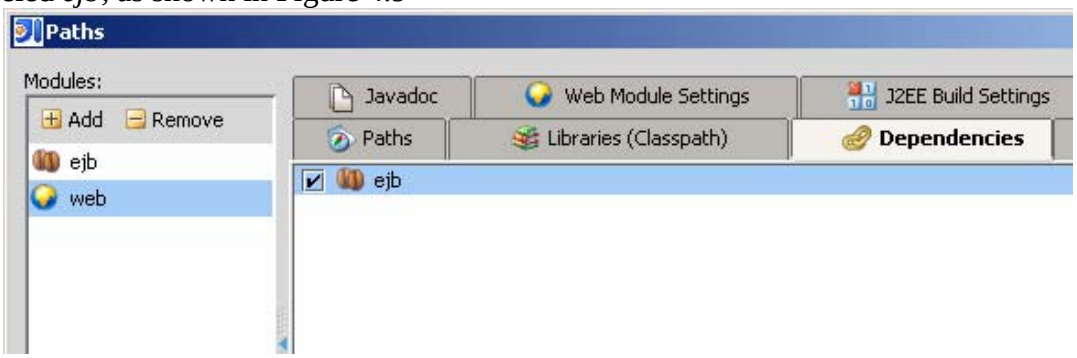

Figure 4.9 Setting the EJB Dependency

Selecting *OK* will save your settings, close the *Paths* dialog, to return you to the project window. The *J2EE* tab will enable a project windows display that now shows the new files, as shown in Figure 4.10.

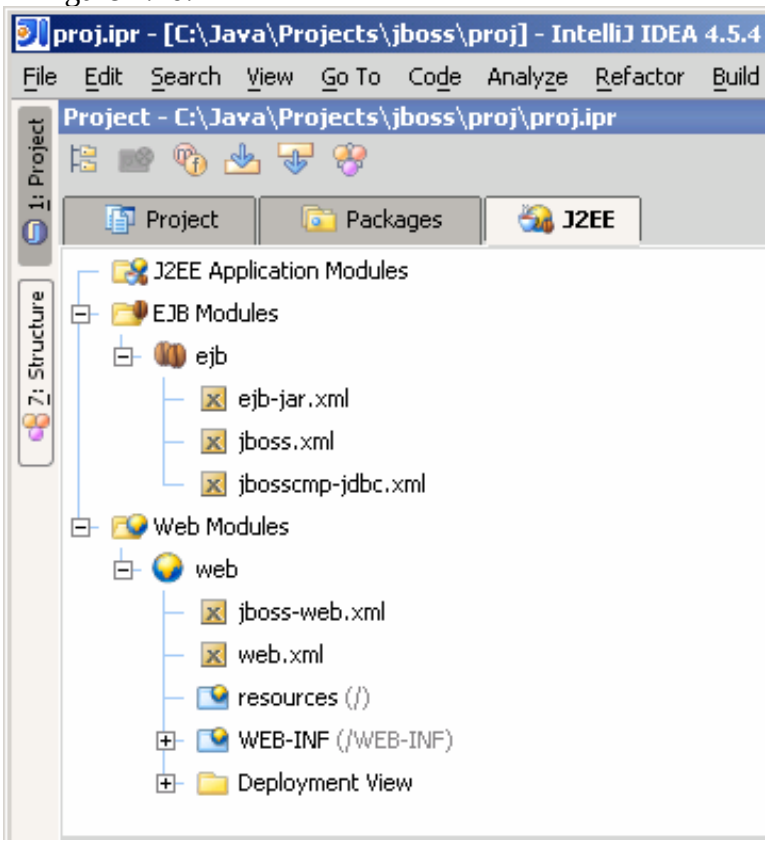

Figure 4.10 The new modules displayed in the Project Window

5. CONCLUSION

The JBoss plug-in is freely available and its download is integrated into the IntelliJ IDEA IDE.

In this paper we discussed how to download and install the JBoss plug-in, allowing the JBoss application server to integrate into the IntelliJ IDEA development environment. We created a project with EJB and web modules, and marked the EJB module as a web module dependency.

In Part 2 we will describe how to add a session bean, implementing a local or remote interface to our bean.

REFERENCES

[BUSINESS WIRE 2004] (BW)(GA-JBOSS-GROUP) Independent Survey Shows Surge in Use of Open Source JBoss Application Server, Business Editors/High-Tech Writers, ATLANTA-- (BUSINESS WIRE)--Jan. 20, 2004-- http://www.businesswire.com/cgi-bin/f_headline.cgi?bw.012004/240205555

[Lyon 2004] *Java for Programmers*, by Douglas A. Lyon, Prentice Hall, 2004. Available from http://www.docjava.com.

About the authors

After receiving his Ph.D. from Rensselaer Polytechnic Institute, **Dr. Lyon** worked at AT&T Bell Laboratories. He has also worked for the Jet Propulsion Laboratory at the California Institute of Technology. He is currently the Chairman of the Computer Engineering Department at Fairfield University, a senior member of the IEEE and President of DocJava, Inc., a consulting firm in Connecticut. E-mail Dr. Lyon at Lyon@DocJava.com. His website is http://www.DocJava.com.

Thomas Rowland has a B.S. in Electrical Engineering and an M.S. in Software Engineering. He has been consulting as a Software Engineer for the past four years, working for Pfizer Pharmaceutical, Travelers Life & Annuity, and currently at Pitney Bowes. He has also worked for Hyperion Solutions for over 5 years. Mr. Rowland has also had some teaching stints along the way. He is listed in the National Register's 2005-2006 edition of the Who's Who in Executives and Professionals. He resides in Connecticut and can be reached at rowlandtf@netscape.net.

Martin Fuhrer has a degree as engineer in computer science from the School of Engineering and Information Technology in Biel/Switzerland. He is founder and president of Fuhrer Engineering Inc., a software development company located in Biel/Switzerland. He's mainly working in the field of web-based financial services and the online processing of realtime stock exchange data. He can be reached at info@fuhrer.com or through http://www.fuhrer.com.