JOURNAL OF OBJECT TECHNOLOGY

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

Vol. 4, No. 7, September-October 2005

The JBoss Integration Plug-in for the IntelliJ IDEA, Part 2.

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.

Java is your friend.
PL/1 is your crazy friend.
ADA is your crazy friend who juggles porcupines.
- DL

Abstract

This paper is the second in a series of papers that describe a new plug-in for enabling the integration of the IntelliJ IDEA IDE with the JBoss application server. The JBoss plug-in was first conceived and implemented by Martin Fuhrer at Fuhrer Engineering.

Part 1 discussed how to download and install the new JBoss plug-in, allowing the JBoss application server to integrate into the IntelliJ IDEA IDE development environment. It showed how to create a project with EJBs and web modules.

This paper continues with our project created in Part 1 by describing how to add a stateless session bean. It is stateless because we are not concerned with remembering values of attributes between successive calls from the client. This paper also demonstrates that our session bean can be either remote (i.e., a bean with a remote interface) or it can be local (a bean with a remote interface).

1 CREATING A SESSION BEAN WITH REMOTE INTERFACE

This section describes the procedure for creating a session bean with a remote interface in the IntelliJ IDE. The examples in this section were performed on a Linux machine using IntelliJ IDE version 4.5 and JBoss version 4.0.1 RC1.

Right-click on the EJB module (or control-click, on a Mac) to select the session bean menu item, as shown in Figure 1.1.

Cite this column as follows: D. Lyon, M. Fuhrer and T. Rowland: “The JBoss Integration Plug-in for IntelliJ IDEA, Part2”, in *Journal of Object Technology*, vol. 4, no. 7, September-October 2005, pp. 25-34, http://www.jot.fm/issues/issue_2005_9/column3

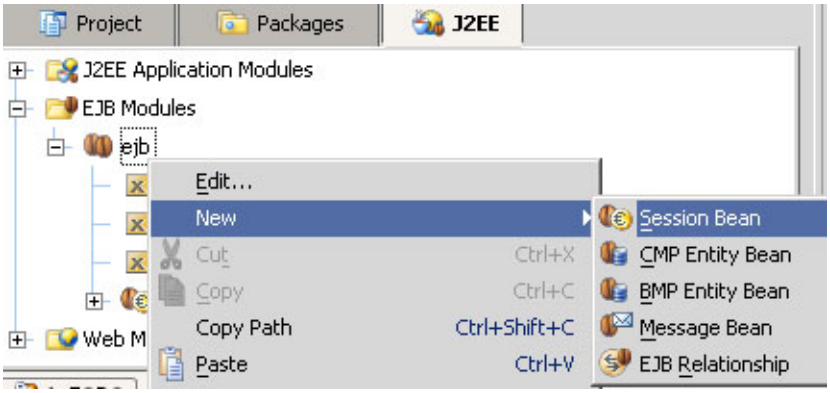


Figure 1.1 Creating a new Session Bean

In the *New Session Bean* dialog, shown in Figure 1.2, enter in the name of the session bean and the package. Select *Enable Remote Interface*.

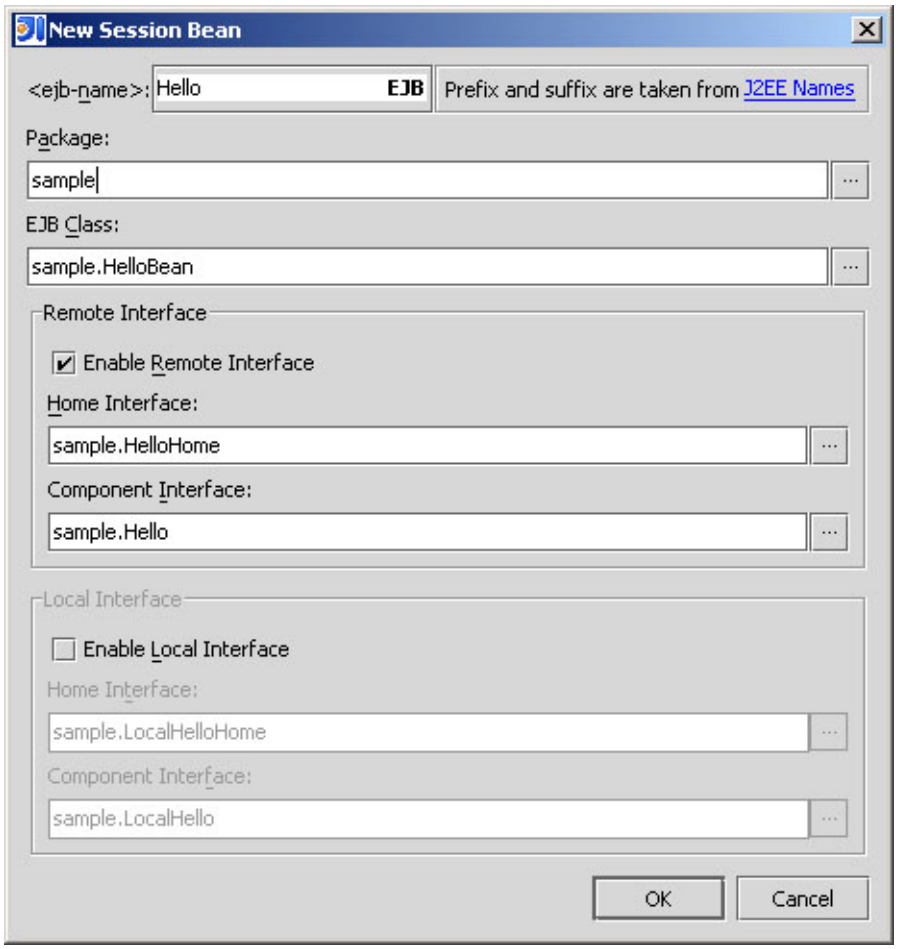


Figure 1.2 The New Session Bean dialog with remote interfaces enabled

Select *OK* and the *Session Bean* dialog *General* panel will be displayed.

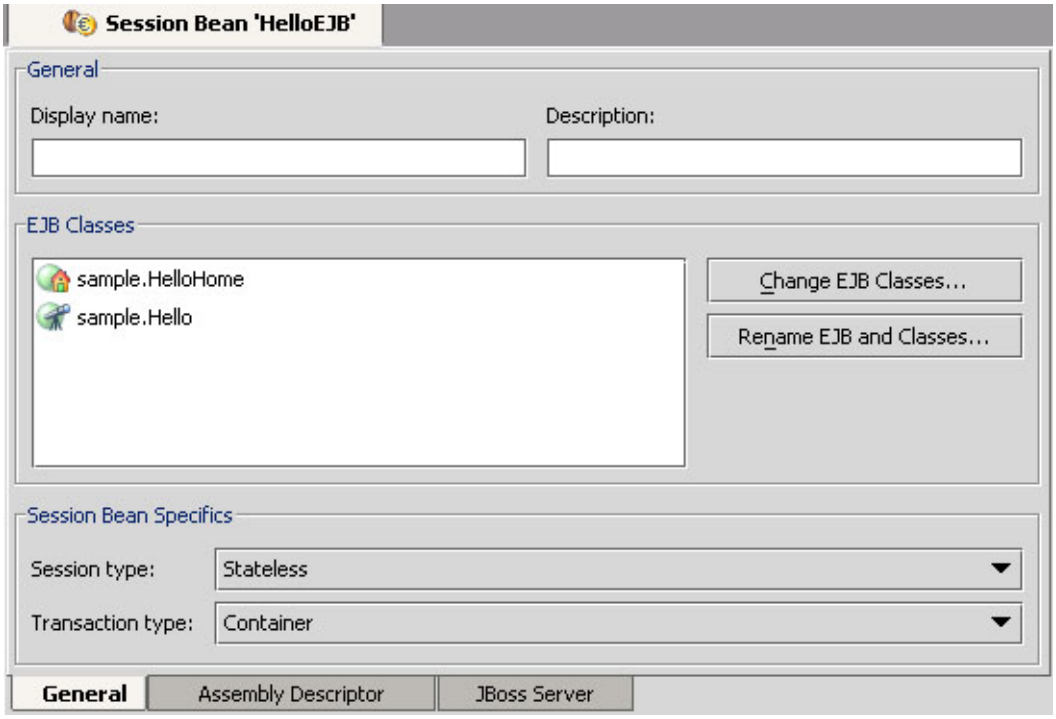


Figure 1.3 The Session Bean dialog showing remote interfaces

Now select the *JBoss Server* tab and type in the *Remote JNDI* (Java Naming and Directory Interface) *Name*, "hello".

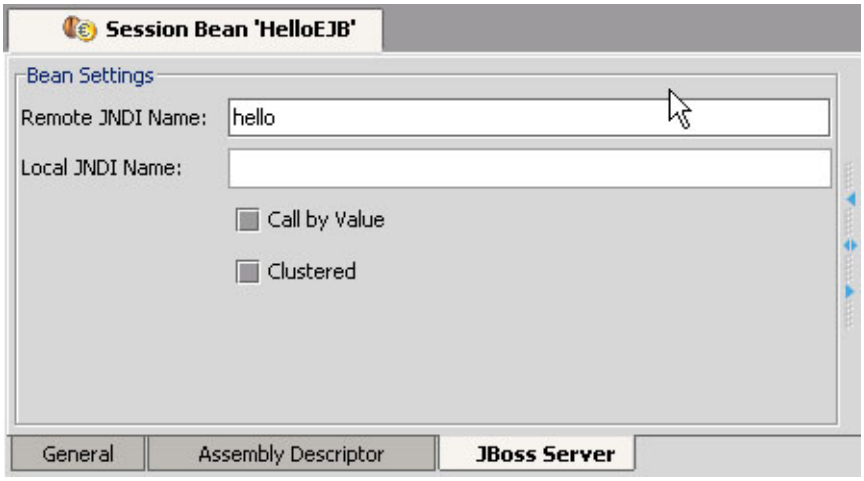


Figure 1.4 The Session Bean dialog showing the Remote JNDI Name

Close the Session Bean dialog. The project *JTree* will now show the new session bean class and interfaces that you just created, as shown in Figure 1.5.

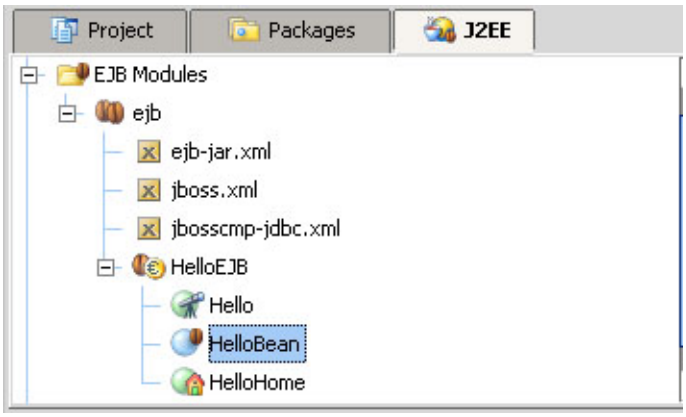


Figure 1.5 The new session bean showing class and interfaces

2 CREATING A SESSION BEAN WITH LOCAL INTERFACE

This section describes the procedure for creating a session bean with a local interface in the IntelliJ IDE. In the [EJB 2.0] specification, local interfaces were added in order to optimize calls between beans residing in the same container. Local interfaces are more efficient because they do not rely on the Java Remote Method Invocation (RMI) mechanism, but their use is a tradeoff with flexibility, and they can only be used in certain situations. It is beyond the scope of this paper to discuss the differences between local and remote bean interfaces. The steps in creating a session bean with a local interface, and the servlet which invokes methods in the session bean, are much the same as in the previous sections that use the remote interface. However, we are showing it here at length to illustrate the capabilities of the IntelliJ IDE.

The examples in this section were performed on a Windows 2000 machine using IntelliJ IDE version 4.5.4 and JBoss version 4.0.1 SP1.

Right-click on the EJB module (or control-click, on a Mac) to select the session bean menu item, as shown in Figure 2.1.

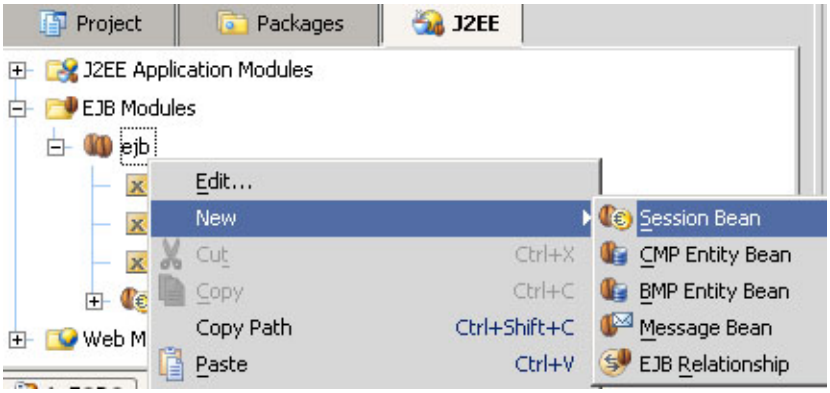


Figure 2.1 Creating a new Session Bean

In the *New Session Bean* dialog, shown in Figure 2.2, enter in the name of the session bean and the package. Select *Enable Local Interface.* IntelliJ will prefix the home and component interface names with *local*.

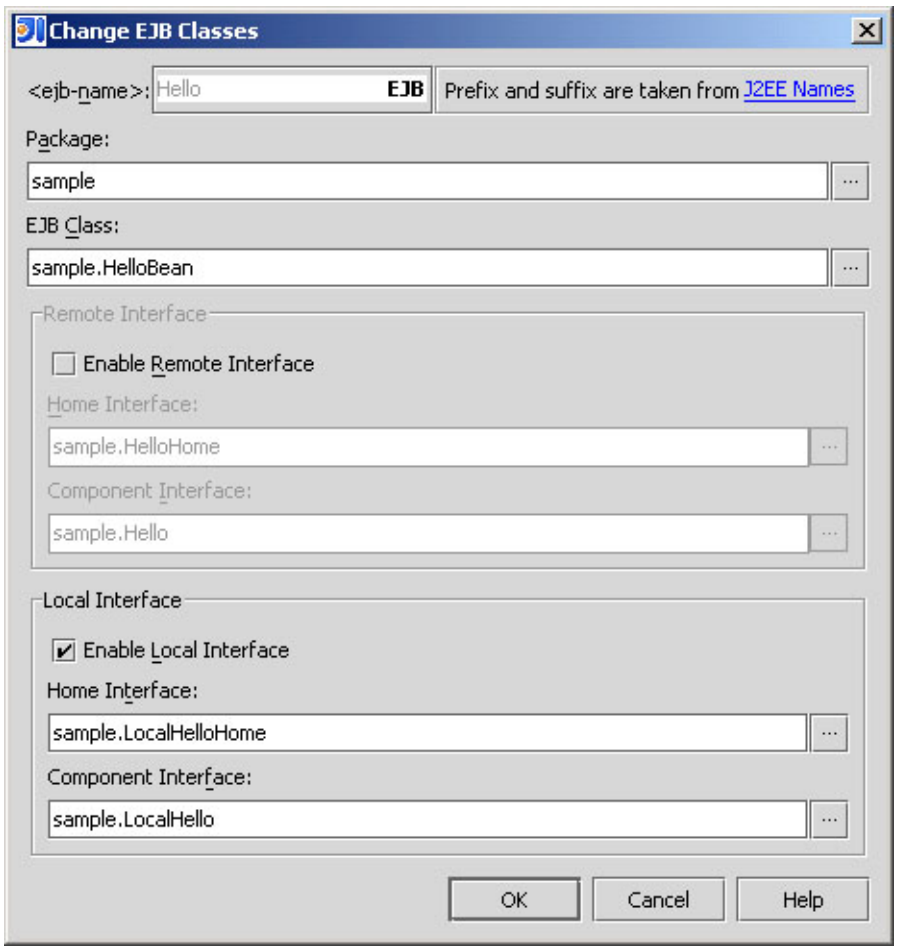


Figure 2.2 The New Session Bean dialog with Local interfaces enabled

Select *OK* and the *Session Bean* dialog *General* panel will be displayed.

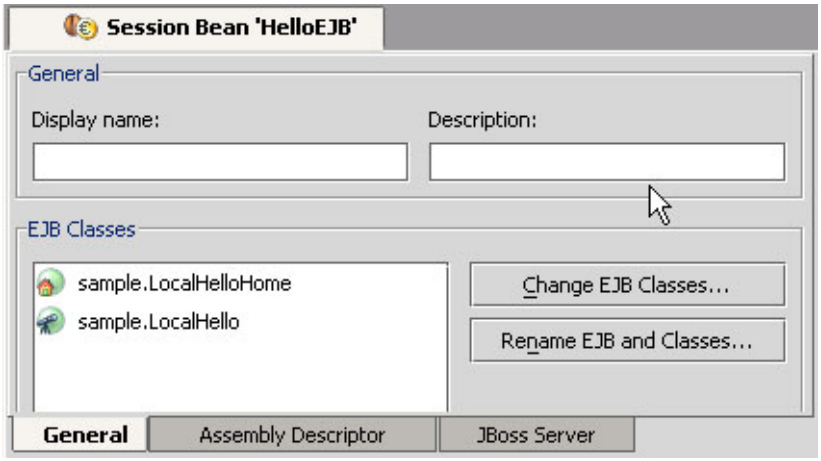


Figure 2.3 The Session Bean dialog showing Local interfaces

Now select the *JBoss Server* tab and type in the *Local JNDI* (Java Naming and Directory Interface) *Name*, "hello".

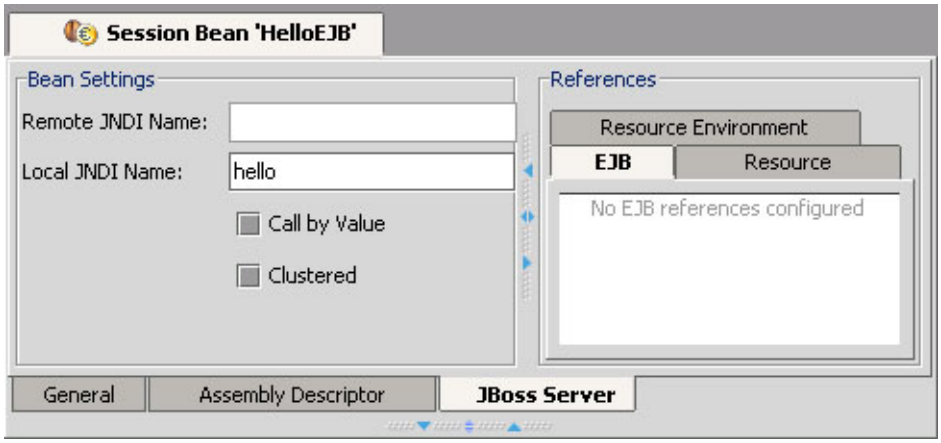


Figure 2.4 The Session Bean dialog showing the Local JNDI Name

Close the Session Bean dialog. The project *JTree* will now show the new session bean class and interfaces that you just created, as shown in Figure 2.5.

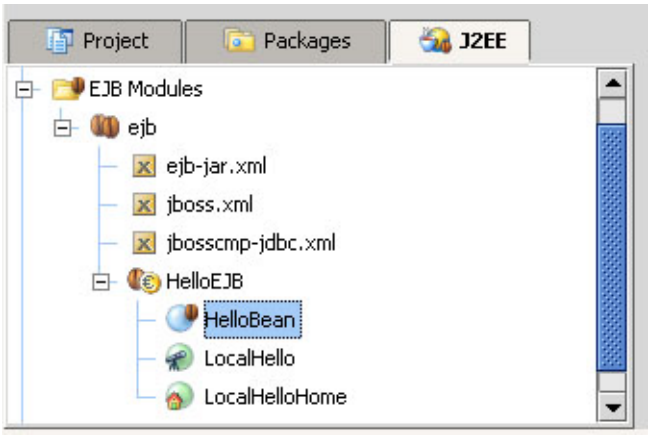


Figure 2.5 The new session bean showing class and interfaces

3 ADDING METHODS TO THE BEAN CLASS

Open the *HelloBean* class from the project *JTree* and add the two methods, *sayHello* and *getDate*, as shown in Figure 3.1. Placing the cursor on the *sayHello* and *getDate* method names will bring up light bulbs to the left of them.

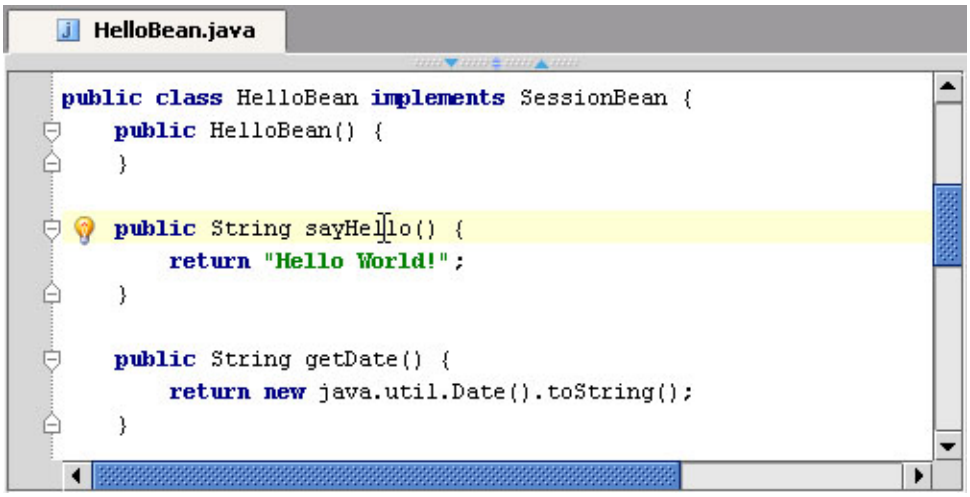


Figure 3.1 Adding new HelloBean methods

Click on the light bulbs and accept the offer from the popup (shown in figure 3.2) to add the specification for each of your new methods to the *HelloBean*'s component interface.

Figure 3.2 Popup offering to Synthesize Code

Accepting the offer to synthesize code causes the specifications to be automatically generated in the interfaces, as shown in Figures 3.3 and 3.4. Notice the difference between the *local component* interface and a *remote component* interface, instead of extending *EJBObject*, a local component interface extends *EJBLocalObject*. There is also no need for it to throw a *RemoteException*. Likewise, the *local home* interface extends *EJBLocalHome* instead of *EJBHome*, and again there is no need to throw a *RemoteException*.

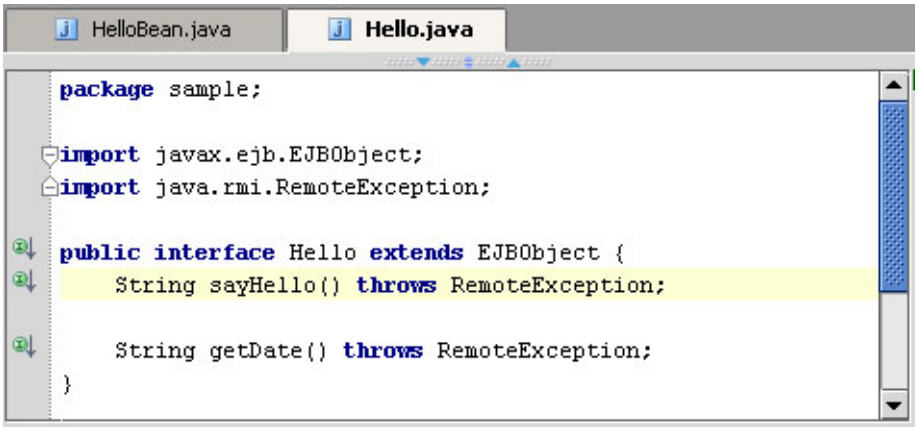


Figure 3.3 Synthesized Methods in the (remote) bean interface

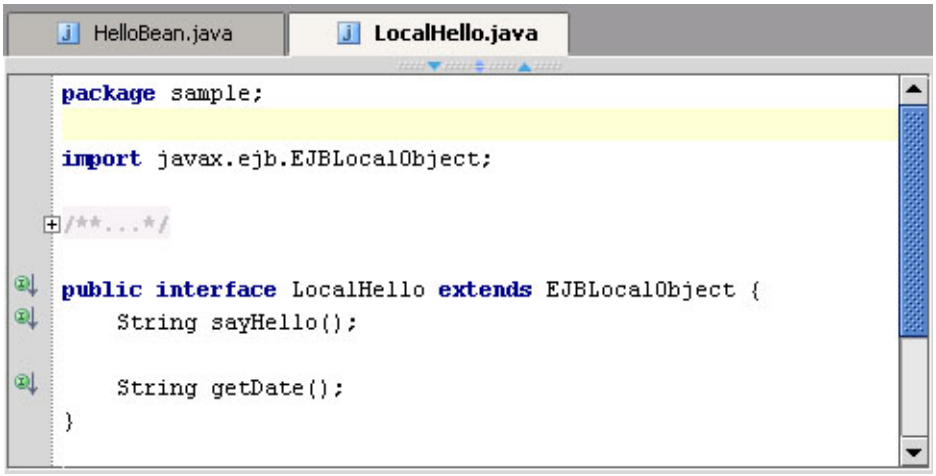
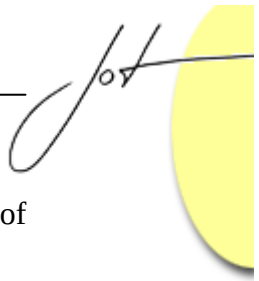


Figure 3.4 Synthesized Methods in the (local) bean interface

4 CONCLUSION

This paper discussed how the new JBoss application server integration plug-in makes it easy to create EJBs within the IntelliJ IDEA IDE environment. It showed that both remote and local bean interfaces are supported.



Now that you have created your EJB, you are ready for the next step, the creation of a servlet to interface to the EJB. This will be the topic of Part 3.

The use of plug-in technologies has become increasingly prevalent in the IDE industry. This is understandable, as there are too many frameworks for the IDE manufacturer to support at any given time. A plug-in also gives third parties the ability to support frameworks that are not high-priority for the IDE manufacturer. Further, they enable the IDE to be leaner, incorporating support for only those features that users care about. Thus, we see the plug-in to be a popular trend in the IDE world and we see no slow-down in this trend. The question of how plug-in producers can make money from their plug-ins, remains open.

LITERATURE CITED

[EJB 2.0] Enterprise JavaBeans[tm] 2.0 Specification, available from http://java.sun.com/products/ejb/2.0.html.

[Cavaness] *Special Edition Using EJB 2.0*, Chuck Cavaness and Brian Keeton, February 2002. Chapter 3, *EJB Concepts*, can be downloaded from http://java.sun.com/developer/Books/ejbtechnology.

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.

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.

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.