



Developing J2EE™ Applications



VERSION 9

Borland®
JBuilder®

Borland Software Corporation
100 Enterprise Way, Scotts Valley, CA 95066-3249
www.borland.com

Refer to the file `deploy.html` located in the `redist` directory of your JBuilder product for a complete list of files that you can distribute in accordance with the JBuilder License Statement and Limited Warranty.

Borland Software Corporation may have patents and/or pending patent applications covering subject matter in this document. Please refer to the product CD or the About dialog box for the list of applicable patents. The furnishing of this document does not give you any license to these patents.

COPYRIGHT © 1997–2003 Borland Software Corporation. All rights reserved. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other marks are the property of their respective owners.

For third-party conditions and disclaimers, see the Release Notes on your JBuilder product CD.

Printed in the U.S.A.

JBE0090WW21002j2eeapps 1E0R0503

0304050607-9 8 7 6 5 4 3 2 1

PDF

Contents

Chapter 1

Introduction **1-1**

Documentation conventions	1-1
Developer support and resources	1-3
Contacting Borland Technical Support.	1-3
Online resources	1-4
World Wide Web	1-4
Borland newsgroups	1-4
Usenet newsgroups	1-4
Reporting bugs	1-5

Chapter 2

Programming for the Java 2 Platform, Enterprise Edition **2-1**

Why are J2EE applications desirable?	2-1
Benefits of the multi-tier model.	2-4
How JBuilder can help	2-5
Client tier technologies	2-5
Middle-tier technologies.	2-6
Other J2EE technologies.	2-7
Preparing to deploy J2EE applications.	2-8
Learning about J2EE.	2-8

Chapter 3

Creating applications with J2EE technologies **3-1**

Client-server applications.	3-2
Multi-tier applications.	3-3
Stand-alone clients.	3-3
Consumers of dynamic web content.	3-4
Java client calling EJBs.	3-4
Web-centric applications	3-5
Business to business	3-6

Chapter 4

Configuring the target server settings **4-1**

Supported servers	4-1
Setting up servers within JBuilder	4-2
Creating a duplicate configuration to edit.	4-5
Adding a service pack.	4-6
The created libraries	4-6
Selecting a server.	4-8

Setting up JDBC drivers	4-10
Creating the .library and .config files	4-10
Adding the JDBC driver to projects	4-11
Updating projects with the latest server settings	4-12

Chapter 5

Using JBuilder with Borland servers **5-1**

Configuring JBuilder for Borland servers	5-1
Making the ORB available to JBuilder	5-3
Partitions, partition services, and J2EE APIs.	5-4
Changing the management port.	5-6
Starting the server.	5-6
Management agent	5-6
Remote deploying	5-7
Remote debugging	5-8
Web Application workarounds	5-10
Using the Borland Enterprise Server console in JBuilder	5-10
International issues	5-10

Chapter 6

Using JBuilder with BEA WebLogic servers **6-1**

Service packs	6-1
Configuring JBuilder for WebLogic servers	6-1
Creating a duplicate configuration to edit	6-4
Created libraries	6-5
Adding a service pack	6-5
Selecting a server	6-6
Working in JBuilder.	6-8
Using existing code	6-8
Creating WebLogic entity beans in JBuilder	6-8
Using the Deployment Descriptor editor.	6-8
Compiling	6-9
Starting the server.	6-9
Remote deploying	6-10
WebLogic Application Server 7.x and WebLogic Platform Server 8.1	6-11
WebLogic Server 6.x.	6-11
WebLogic 8.x Platform Server deployment.	6-12

Remote debugging.	6-13
Remote debugging of JavaServer Pages.	6-13
Updating projects with the latest server settings	6-15

Chapter 7

Using JBuilder with IBM WebSphere servers 7-1

Configuring JBuilder for WebSphere servers	7-1
Selecting a server.	7-4
Working in JBuilder with WebSphere 4.0	7-4
Generating WebSphere deployment descriptors.	7-4
Importing WebSphere 4.0 Advanced Edition proprietary deployment descriptors into JBuilder.	7-5
Starting the server	7-6
Deploying.	7-7
Using the Deployment Descriptor editor . .	7-7
WebSphere Application Server 5.0	7-7
Remote deploying	7-7
Deploying locally	7-8
WebSphere Server 4.0	7-8
Remote deploying	7-9
Deploying locally	7-10
Enabling remote debugging	7-10
WebSphere Application Server 5.0	7-10
WebSphere Single Server 4.0	7-11
WebSphere Server Advanced Edition 4.0.	7-12
Container-managed persistence (CMP) 1.1 and 2.0 in WebSphere 5.0	7-13

Chapter 8

Using JBuilder with Sybase servers 8-1

Configuring JBuilder for Sybase servers	8-1
Selecting a server.	8-3
Starting the server	8-3
Remote deploying	8-4
Deploying web applications	8-4
Remote debugging.	8-5
New features for Sybase EAServer in JBuilder	8-6

Chapter 9

Using JBuilder with Sun iPlanet servers 9-1

Configuring JBuilder for iPlanet servers	9-1
Selecting a server	9-2
Service packs	9-2
Running or debugging on iPlanet.	9-3
Starting and stopping iPlanet from within JBuilder.	9-3
Running or debugging a servlet or JavaServer Page	9-3
Remote deploying	9-4
Remote debugging	9-4
Deployment settings	9-5
Creating clients for iPlanet.	9-5

Chapter 10

Using JBuilder with your web server 10-1

Viewing Tomcat configurations	10-1
Configuring other web servers	10-3
Selecting a server for your project.	10-4
Configuring the IDE for web run/debug . . .	10-6
Web running your servlet or JSP	10-7
Starting your web server	10-7
Web view	10-8
Web view source	10-9
Stopping the web server	10-9
Changing the web server's port number	10-10
Creating a custom server.xml file with Tomcat	10-10
Web debugging your servlet or JSP.	10-11

Index

I-1

Introduction

Developing J2EE Applications introduces you to the various technologies that make up the Java™ 2 Platform, Enterprise Edition (J2EE™), explains why they are important, and describes how you can use JBuilder to create J2EE applications that target your application server. You'll learn how to configure JBuilder to work with your server, and how to accomplish the essential programming tasks of running your applications, debugging them remotely, and deploying them to your remote server.

Documentation conventions

The Borland documentation for JBuilder uses the typefaces and symbols described in the following table to indicate special text.

Table 1.1 Typeface and symbol conventions

Typeface	Meaning
Bold	Bold is used for java tools, <i>bmj</i> (Borland Make for Java), <i>bcj</i> (Borland Compiler for Java), and compiler options. For example: javac , bmj , -classpath .
<i>Italics</i>	Italicized words are used for new terms being defined, for book titles, and occasionally for emphasis.
<i>Keycaps</i>	This typeface indicates a key on your keyboard, such as “Press <i>Esc</i> to exit a menu.”

Table 1.1 Typeface and symbol conventions (continued)

Typeface	Meaning
Monospaced type	<p>Monospaced type represents the following:</p> <ul style="list-style-type: none"> • text as it appears onscreen • anything you must type, such as “Type Hello World in the Title field of the Application wizard.” • file names • path names • directory and folder names • commands, such as <code>SET PATH</code> • Java code • Java data types, such as <code>boolean</code>, <code>int</code>, and <code>long</code>. • Java identifiers, such as names of variables, classes, package names, interfaces, components, properties, methods, and events • argument names • field names • Java keywords, such as <code>void</code> and <code>static</code>
[]	Square brackets in text or syntax listings enclose optional items. Do not type the brackets.
< >	<p>Angle brackets are used to indicate variables in directory paths, command options, and code samples.</p> <p>For example, <code><filename></code> may be used to indicate where you need to supply a file name (including file extension), and <code><username></code> typically indicates that you must provide your user name.</p> <p>When replacing variables in directory paths, command options, and code samples, replace the entire variable, including the angle brackets (<code>< ></code>). For example, you would replace <code><filename></code> with the name of a file, such as <code>employee.jds</code>, and omit the angle brackets.</p> <p>Note: Angle brackets are used in HTML, XML, JSP, and other tag-based files to demarcate document elements, such as <code></code> and <code><ejb-jar></code>. The following convention describes how variable strings are specified within code samples that are already using angle brackets for delimiters.</p>
<i>Italics, serif</i>	This formatting is used to indicate variable strings within code samples that are already using angle brackets as delimiters. For example, <code><url="jdbc:borland:jbuilder\samples\guestbook.jds"></code>
...	In code examples, an ellipsis (...) indicates code that has been omitted from the example to save space and improve clarity. On a button, an ellipsis indicates that the button links to a selection dialog box.

JBuilder is available on multiple platforms. See the following table for a description of platform conventions used in the documentation.

Table 1.2 Platform conventions

Item	Meaning
Paths	Directory paths in the documentation are indicated with a forward slash (/). For Windows platforms, use a backslash (\).
Home directory	The location of the standard home directory varies by platform and is indicated with a variable, <home>. <ul style="list-style-type: none"> For UNIX and Linux, the home directory can vary. For example, it could be /user/<username> or /home/<username> For Windows NT, the home directory is C:\Winnt\Profiles\<username> For Windows 2000 and XP, the home directory is C:\Documents and Settings\<username>
Screen shots	Screen shots reflect the Metal Look & Feel on various platforms.

Developer support and resources

Borland provides a variety of support options and information resources to help developers get the most out of their Borland products. These options include a range of Borland Technical Support programs, as well as free services on the Internet, where you can search our extensive information base and connect with other users of Borland products.

Contacting Borland Technical Support

Borland offers several support programs for customers and prospective customers. You can choose from several categories of support, ranging from free support on installation of the Borland product to fee-based consultant-level support and extensive assistance.

For more information about Borland's developer support services, see our web site at <http://www.borland.com/devsupport/>, call Borland Assist at (800) 523-7070, or contact our Sales Department at (831) 431-1064.

When contacting support, be prepared to provide complete information about your environment, the version of the product you are using, and a detailed description of the problem.

For support on third-party tools or documentation, contact the vendor of the tool.

Online resources

You can get information from any of these online sources:

World Wide Web	http://www.borland.com/ http://www.borland.com/techpubs/jbuilder/
Electronic newsletters	To subscribe to electronic newsletters, use the online form at: http://www.borland.com/products/newsletters/index.html

World Wide Web

Check www.borland.com/jbuilder regularly. This is where the Java Products Development Team posts white papers, competitive analyses, answers to frequently asked questions, sample applications, updated software, updated documentation, and information about new and existing products.

You may want to check these URLs in particular:

- <http://www.borland.com/jbuilder/> (updated software and other files)
- <http://www.borland.com/techpubs/jbuilder/> (updated documentation and other files)
- <http://community.borland.com/> (contains our web-based news magazine for developers)

Borland newsgroups

When you register JBuilder you can participate in many threaded discussion groups devoted to JBuilder. The Borland newsgroups provide a means for the global community of Borland customers to exchange tips and techniques about Borland products and related tools and technologies.

You can find user-supported newsgroups for JBuilder and other Borland products at <http://www.borland.com/newsgroups/>.

Usenet newsgroups

The following Usenet groups are devoted to Java and related programming issues:

- 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`

Note These newsgroups are maintained by users and are not official Borland sites.

Reporting bugs

If you find what you think may be a bug in the software, please report it to Borland at one of the following sites:

- **Support Programs** page at <http://www.borland.com/devsupport/namerica/>. Click the “Reporting Defects” link to bring up the Entry Form.
- **Quality Central** at <http://qc.borland.com>. Follow the instructions on the Quality Central page in the “Bugs Reports” section.

When you report a bug, please include all the steps needed to reproduce the bug, including any special environmental settings you used and other programs you were using with JBuilder. Please be specific about the expected behavior versus what actually happened.

If you have comments (compliments, suggestions, or issues) for the JBuilder documentation team, you may email jpgpubs@borland.com. This is for documentation issues only. Please note that you must address support issues to developer support.

JBuilder is made by developers for developers. We really value your input.

Programming for the Java 2 Platform, Enterprise Edition

The JBuilder WebLogic Edition provides support for WebLogic Servers only

The Java™ 2 Platform, Enterprise Edition (J2EE™) is an architecture for a Java development platform for distributed enterprise applications. It was developed by Sun Microsystems, with input from the development community, including Borland. J2EE platform products, such as the Borland Enterprise Server, offer the developer the capability of building applications with these benefits:

- Reliability and scalability, so business transactions are processed quickly and accurately.
- Excellent security to protect users' privacy and the integrity of the enterprise's data.
- Ready availability, to meet the increasing demands of the global business environment.

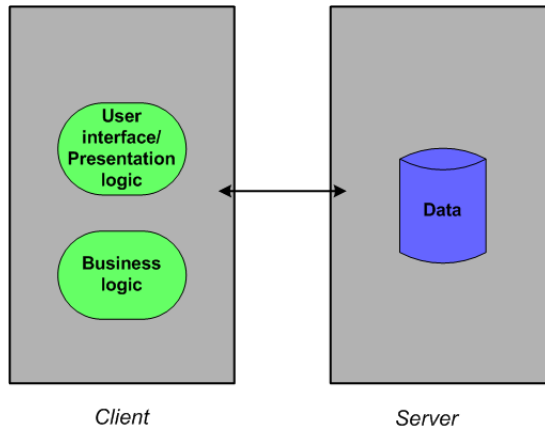
JBuilder is a Java integrated development environment that, when coupled with a supported application server from companies such as Borland, BEA, IBM, Sun, and Sybase greatly simplifies and speeds the development of J2EE applications.

Why are J2EE applications desirable?

In the early 1990s, information systems frequently used a client-server architecture. The user interface to the application usually ran on a desktop computer. This was the client tier. The enterprise data being accessed by

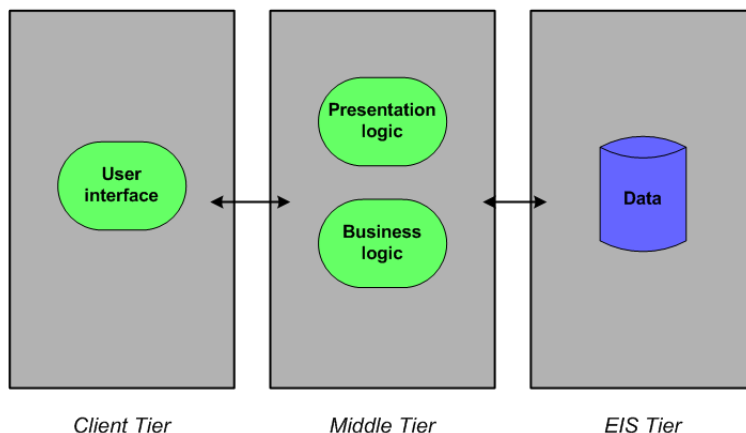
Why are J2EE applications desirable?

the client resided in a database and was “served up” by a server. This approach initially promised improved scalability and functionality.



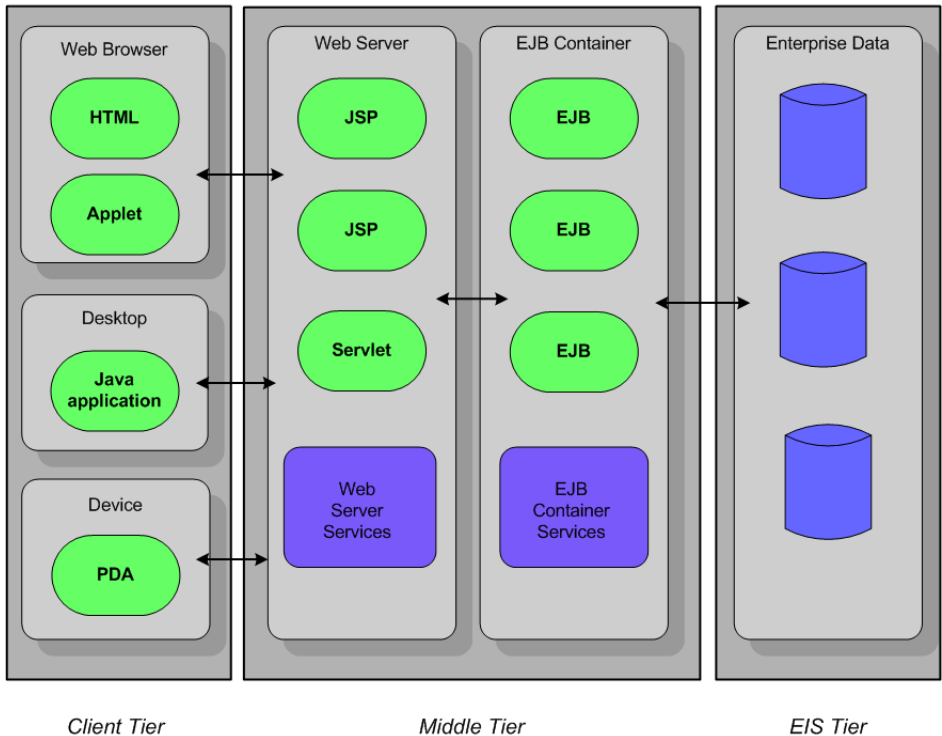
Through hard experience, however, the development community learned that building and maintaining a flexible distributed system is very difficult using the client-server model. For example, the business logic of the application was in the client application. Every time that logic needed modification, the revised application had to be installed on every client machine in the enterprise. Maintenance became a nightmare. These applications also had to manage transactions, be concerned with security, and process the data efficiently, all the while presenting an attractive, easy-to-understand interface to its users. Few developers have talents in all these areas. While a client-server architecture might be adequate for some environments, most of today's global companies demand considerably more than the client-server model can deliver.

Once the limitations of the client-server approach became apparent, the development community began seeking a better way. The result is the multi-tier model.



In the multi-tier model, the logic involved in presenting the user interface of the application to the user lives on the middle tier. The business logic is now on the middle tier also. When changes are needed, they can be updated in one place instead of on each client machine.

This expanded diagram shows you the various components you might find running on the various tiers:



The client in a J2EE application can be an HTML page or applet running in a browser, a Java application on a desktop machine, or even a Java client on some portable device, such as a personal digital assistant (PDA) or cell phone.

The middle-tier can have JavaServer Pages™ or servlets running on a web server. These elements usually make up the server-side presentation logic. An EJB container provides a runtime environment for Enterprise JavaBeans™, which contain the business logic of application. Both a web server and an Enterprise JavaBeans (EJB) container provide services to the components that run on them. Because these services are always available, programmers don't have to include them in the components they write.

The Enterprise Information System (EIS) tier is a repository for the enterprise's data. Usually it consists of the data in a relational database system.

Few J2EE applications have all of these components. They can be mixed and matched in very flexible ways to meet the needs of the enterprise. See [Chapter 3, “Creating applications with J2EE technologies”](#) for more on this topic.

Benefits of the multi-tier model

The multi-tier approach adopted by the J2EE platform has several benefits:

- It reduces the complexity of distributed development with a simplified architecture and the sharing of the work load among roles.

The business logic of the application runs in the middle tier inside an Enterprise JavaBean (EJB) container and/or on a web server. These containers and servers can handle many of the difficult tasks for developers. For example, an EJB container can handle transactions, instance pooling, and data persistence without requiring the EJB programmer to write the logic to perform these tasks. A web server can create and pool instances of servlet classes and handle multiple threads and socket connections. Instead of writing the code to do these things, a member of the development team specifies the desired behavior at deployment time.

Members of the development team play different roles. Each is a specialist in one or more areas. For example, the content of an HTML page or stylesheet would likely be created by a graphic designer or webmaster. A senior developer might be responsible for the business logic of the application encapsulated within Enterprise JavaBean components. A web developer might develop the user interface and presentation logic using JavaServer Pages (JSPs) and servlets. An application assembler takes the various components of the application and puts it all together, often creating an Enterprise Archive (EAR) file and creating the deployment descriptor that explains how the application is to be deployed. The application deployer and administrator deploy the application. By partitioning the work this way, each step of the development/deployment process is handled by someone skilled in their area while no one has to be an expert in every area.

- It is highly scalable, allowing the development of systems to meet many different needs that can change quickly.

When demands on the system increase, the logic can be updated easily in one place on the middle tier without having to load new logic on every client machine.

- New applications can integrate well with existing information systems.

JDBC, a J2EE technology, is a Java API to SQL databases, permitting the access to any type of tabular data that might exist in the enterprise. The Java Naming and Directory Interface (JNDI) allows applications that use Java technology to access enterprise naming and directory services. The J2EE Connector architecture gives Java applications connections to heterogeneous legacy systems. The Java Message Service (JMS) is the Java API for sending and receiving messages through enterprise messaging systems. CORBA services are called using JavaIDL.

- Security is enhanced.

J2EE technologies are designed with security in mind. For example, only users in assigned roles can access certain methods in enterprise beans. Who can access these methods isn't coded in the enterprise beans themselves. Instead this information is set in the enterprise beans' deployment descriptors, which are used by the deployer to establish their behavior after they are deployed. This type of security is called declarative security.

For complex security requirements, however, J2EE also allows programmable security. In these cases, the security logic is placed within the code itself.

- Developers can choose from a variety of development tools, servers, and components to develop the applications they need.

The development team can select the solutions that are best for their needs, without becoming locked into the offerings of a single vendor.

How JBuilder can help

JBuilder Enterprise Edition has many features to help your team develop J2EE applications. These are the technologies JBuilder has to help you develop the client tier:

Client tier technologies

- Applets

Applets are a special kind of Java application that are downloaded and run by a web browser on a client machine. To begin developing an applet in JBuilder, start with the Applet wizard. For information about working with applets, see "Working with applets" in *Web Application Developer's Guide*.

- Java user interface applications

JBuilder has several features that can help you develop an application that runs on a client machine. Begin a Java application using the Application wizard. Continue designing your user interface by using JBuilder's UI designer. You create your UI by adding UI components from JBuilder's component palette. For information on working with JBuilder's UI designer, see "Visual design in JBuilder" in *Designing Applications with JBuilder*.

If you want to create your own JavaBean components to use in your user interface, BeansExpress can simplify the task for you. For information about using BeansExpress, see "Creating JavaBeans with BeansExpress" in *Building Applications with JBuilder*. JBuilder's DataExpress components for enterprise beans make it easier for you to build client applications using database-aware visual components such as dbSwing or InternetBeans Express. For more information about DataExpress for EJB, see "Using the DataExpress for Enterprise JavaBeans components" in *Developing Enterprise JavaBeans*.

Both a web server and/or an EJB container can run on the middle tier. JBuilder ships with Tomcat, a servlet container that can be used as a web server, and with the Borland Enterprise Server 5.2.1, which contains the EJB container. You can build applications for these servers or you can set up JBuilder to enable you to develop applications for BEA WebLogic 6.x, 7.x, and 8.1, IBM WebSphere 4.0 and 5.0, Sun-Netscape iPlanet 6.0 and 6.5, and Sybase 4.1 and 4.2.

Middle-tier technologies

These are the middle-tier J2EE technologies that use a web server:

- Servlets

A servlet is a server-side Java application that can process requests from clients. The servlet responds to the request by generating dynamic output that is sent back to the client. You can begin developing servlets with JBuilder's Servlet wizard. To find out more about servlets and developing them, see "Working with servlets" in *Web Application Developer's Guide*.

- JavaServer Pages (JSPs)

An extension of servlet technology, JSPs offer a simplified way to develop servlets. Like servlets, they generate dynamic output that is sent back to the client's web browser. Begin developing JSPs with JBuilder's JavaServer Page wizard. To find out more about JSPs and developing them, see "JavaServer Pages (JSP)" in *Web Application Developer's Guide*.

InternetBeans Express is a component library that supplements the servlet and JSP technology available in JBuilder. This library makes it

easy to present and manipulate data from a database so you can build data-aware servlets and JSPs. For information about InternetBeans Express, see “Using InternetBeans Express” in the *Web Application Developer’s Guide*.

This is the middle-tier J2EE technology that uses an EJB container:

- Enterprise JavaBeans (EJBs)

Enterprise JavaBeans are server-side components that contain the business logic of the application. JBuilder assists you in building EJB 1.x and EJB 2.0 components. Start building enterprise beans by using the EJB wizards on the Enterprise page of the object gallery (File | New | Enterprise). For building EJB 2.0 components, JBuilder offers the EJB Designer, a Two-Way Tool™ that allows you to design your beans visually all the while keeping your code, deployment descriptors, and design synchronized. For more information about building, testing, and deploying enterprise beans, see “An introduction to EJB development” in *Developing Enterprise JavaBeans*.

As you create your enterprise beans, JBuilder is building your EJB deployment descriptors. You can use JBuilder’s Deployment Descriptor editor to modify them as you wish. For more information about the Deployment Descriptor editor, see “Using the Deployment Descriptor editor” in *Developing Enterprise JavaBeans*.

All the web application components and enterprise bean components can be combined and delivered in an Enterprise Archive (EAR) file. JBuilder has an EAR wizard to help you create your EAR files.

Other J2EE technologies

While not confined to a particular architectural tier, these technologies are enablers that make things work:

- Java Database Connectivity (JDBC)

JDBC is the standard used to access your database on the Enterprise Information Systems (EIS) tier. It defines a Java API you use to write SQL statements that are sent to your database.

JBuilder includes DataExpress, a component library for accessing data in your database. It connects your application to your database using JDBC drivers.

JBuilder also includes JDataStore, an all-Java embedded database and component library. You access JDataStore using JDBC.

Entity beans that access rows in your database, also connect to your data using JDBC.

- Java Message Service (JMS)

JMS is an enterprise messaging service that routes messages between components and processes in a distributed application.

The Borland Enterprise Server includes SonicMQ, a JMS implementation. Also JBuilder supports EJB 2.0 components, which include message-driven beans. Message-driven beans integrate JMS into enterprise beans. JBuilder also includes a JMS wizard. See “Creating JMS producers and consumers” in *Developing Enterprise JavaBeans* for information on creating classes and applications that can create and consume JMS messages.

- Java Naming and Directory Interface (JNDI)

All J2EE servers use JNDI, a Java naming service used to locate distributed objects.

- Extensible Markup Language (XML)

Although not really a J2EE technology, XML is widely used by J2EE technologies. For example, web components and enterprise beans have their deployment descriptors written in XML. These deployment descriptors describe how the components behave once they are deployed.

JBuilder has several XML features that help you accomplish common programming tasks you might encounter in your J2EE projects. For information about JBuilder’s XML features, see “Introduction” in *XML Application Developer’s Guide*.

Preparing to deploy J2EE applications

As you create and compile your web applications and enterprise beans, JBuilder can create the WAR (Web Archive) and EJB-JAR (EJB Archive) files for you automatically. You can choose to bundle the components of a J2EE application together into an EAR file. JBuilder provides an EAR wizard to help you do this.

Learning about J2EE

If you’ve read this far, you’ve been exposed very briefly to many concepts and, with all the acronyms to identify J2EE technologies, an alphabet soup. To develop a deeper understanding of J2EE benefits and concepts, begin your explorations on Sun’s [www.java.sun.com](http://java.sun.com) web site. This <http://java.sun.com/j2ee/docs.html> link takes you to Sun’s J2EE documentation home page where you can find an abundance of useful information.

If you're new to J2EE programming, look at the J2EE tutorial at <http://java.sun.com/j2ee/tutorial/index.html>. For an in-depth discussion of J2EE programming and the recommended programming practices to use in your J2EE applications, don't miss the very detailed J2EE Blueprints, found at <http://java.sun.com/j2ee/blueprints/index.html>. J2EE Blueprints is an integral part of J2EE itself. You'll find it useful when you need to understand deeper concepts and are looking for the best ways to approach J2EE development. This material is also available in book form as *Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition* written by Nicholas Kasseem and the Enterprise Team of Sun. <http://java.sun.com/j2ee/blueprints/aboutthebook.html> links you to information about the book.

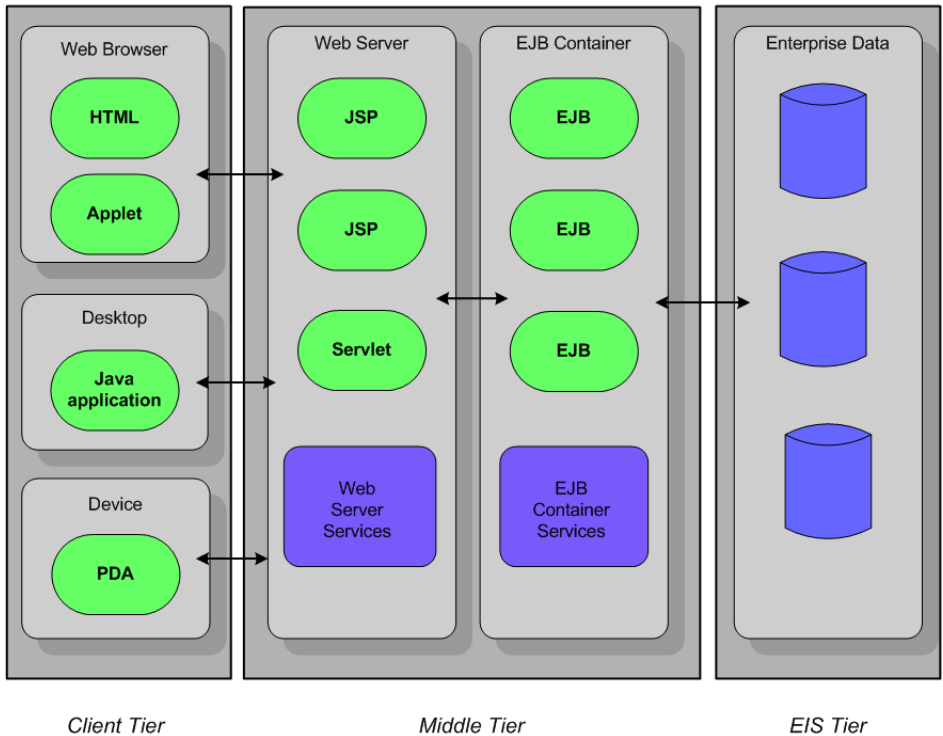
You'll find additional documentation and specifications for the various J2EE technologies on the Sun site. There are also excellent third-party books, but because J2EE is a developing product, be aware which versions of the various technologies they address.

Creating applications with J2EE technologies

The *Web Application Developer's Guide*, *Developing Enterprise JavaBeans*, and other parts of JBuilder's documentation explain how to use J2EE technologies. They describe in depth how to use JBuilder features to develop web applications, work with XML, develop Enterprise JavaBeans, access and work with your data using DataExpress, and create web services. Within each of these areas of the documentation, you should find the information to understand these technologies and work productively with them.

But today's web-based enterprise applications usually use several J2EE technologies. How do these disparate technologies fit together in applications you and your team might develop? Which technologies are right for which kind of applications? This chapter intends to help you see how the pieces fit together, while highlighting the flexibility of J2EE applications. It borrows heavily from the Introduction chapter of *Designing Enterprise Applications with the J2EE Platform* by Inderjeet Singh, Beth Stearns, Mike Johnson, and the Sun Enterprise Team. You can find the book online on the Sun web site at http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/#chapters, and it is also available for purchase.

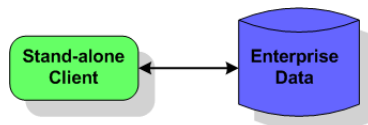
Here's the diagram from the previous chapter showing the multiple tiers used in most J2EE applications:



The diagram shows you all the different types of entities found on each tier, but it doesn't show you the common ways J2EE applications combine these technologies to create reliable, scalable, and easily distributed applications.

Client-server applications

Before looking at multi-tier J2EE models, consider the old standard, the two-tier client-server model. Using J2EE technologies does not preclude you from creating client-server applications. Client-server applications omit the middle tier shown in the multi-tier application diagram. While the client-server model scales poorly and maintaining and distributing such applications is more difficult, sometimes a Java client on a desktop that accesses the data of the company directly is all you really need.

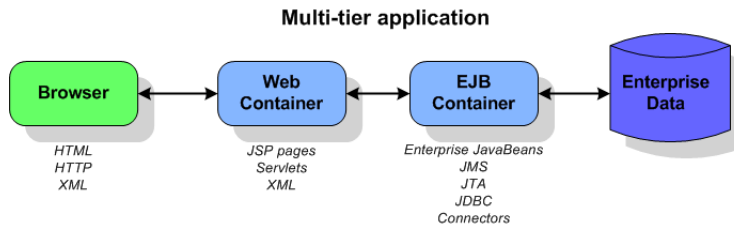


The Java client can contain both the presentation and business logic, although it's possible one application could handle the presentation tasks while another could handle business logic, all on the client tier. You would use JDBC or Connectors to access the data on the Enterprise Information System (EIS) tier.

Consider using the client-server model only when the number of desktops running clients is quite small and will remain so. Once the demand for more than a few client instances increases, you should probably consider another type of application architecture.

Multi-tier applications

Here you see the classic multi-tier, distributed architecture that uses three tiers: the browser on the client tier, the web container and EJB container on the middle tier that runs on a J2EE server, and the Enterprise Information Systems (EIS) tier that is managed by a database system. Such an application uses web components such as JavaServer Pages (JSPs), servlets, and XML to manage the application's presentation logic. The EJB container contains enterprise beans that respond to requests from the web tier components and access the data in the EIS tier.



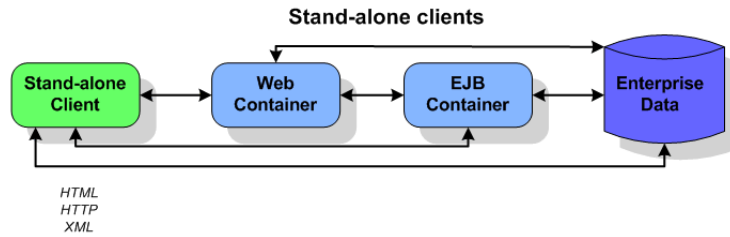
When should you use JavaServer Pages and when should you use servlets? JavaServer Pages are intended as user interface components. Servlets are usually used for processing requests and controlling application logic.

XML can be very important in multi-tier applications. XML data messages use HTTP as their transport protocol. Such data messages can encapsulate all types of data and respond to a variety of client types, including XML-enabled browsers.

Stand-alone clients

Not all multi-tier applications use a browser on the client tier. Often the client is a Java client or a client written in another language that consumes dynamic web content or an application that interacts directly with

enterprise beans running in an EJB container. This diagram shows you the possibilities of this type of application:



Note that this diagram includes the client-server application also as such an application includes a stand-alone client, too.

Consumers of dynamic web content

One type of stand-alone client is a Java application or another programming language that consumes dynamic web content. This web content is usually XML data messages:



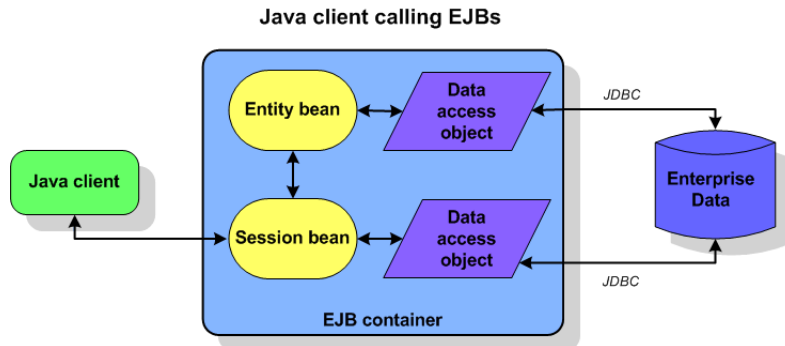
The web container performs the XML transformations and provides web connectivity to clients. The client handles the presentation logic while the web tier manages the business logic and accesses the data on the EIS tier if necessary. Business logic might be implemented as enterprise beans. The J2EE technologies likely to be used are XML, servlets, web services, and possibly enterprise beans and JDBC to access enterprise data.

Java client calling EJBs

Enterprise JavaBeans (EJBs) are server-side components that run in a middle-tier EJB container. The container provides services to the beans, such as transaction management and security. You, as the developer, don't have to implement such low-level and complex services, but instead can concentrate on developing the business logic for the beans, knowing that the services are there when they are needed. Developers often choose EJBs when building applications that will be distributed over several servers. They also like to use EJBs because of the transparent way they handle transactions.

The following diagram depicts a Java client that calls upon the business methods encapsulated with the enterprise beans that are running in an EJB container. If the beans are entity beans, they access company data on

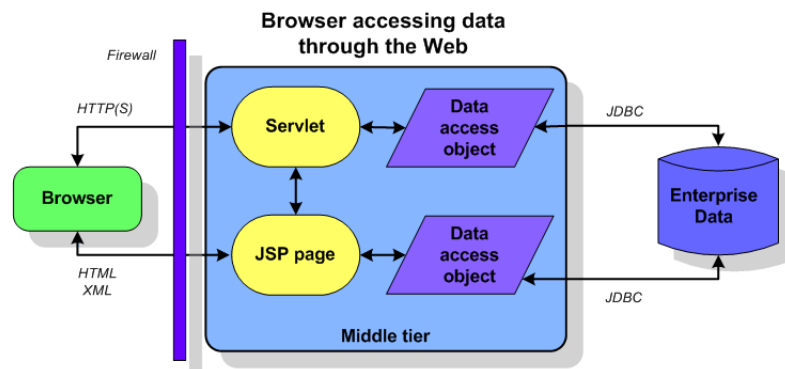
the EIS tier through JDBC and J2EE connectors. Ideally, the client should access company data through a session bean which interacts with the entity beans that model company data. A session bean that interacts with an entity bean in this way is usually using the Session Facade design pattern; see “Creating session facades for entity beans” in *Developing Enterprise JavaBeans* for more information. XML is used in the deployment descriptors for enterprise beans.



If the application you need requires secure transactions over a distributed system, consider using enterprise beans.

Web-centric applications

There are times when your application just doesn't need to use enterprise beans and if you do so, you add a layer of complexity. If your application doesn't require transactions on a distributed system, consider a web-centric application model:

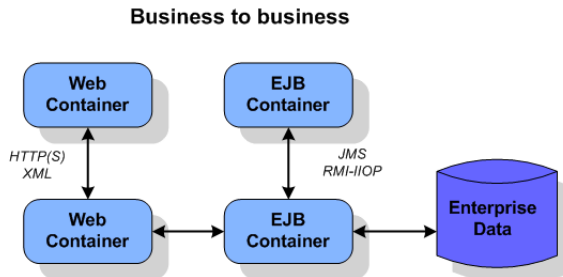


This scenario puts both the presentation and business logic on the web tier. The web container can host JavaServer Pages and servlets. The client is simply a browser and data is transferred using XML, HTTP protocol, and/or HTML pages. Servlets can access the EIS tier, when necessary,

using JDBC or connectors. Applications using this type of architecture are quite common.

Business to business

When you need a web-based commerce solution, consider the business-to-business model. This might include multiple web containers and multiple EJB containers.



Web container to web container architecture is suitable for ecommerce solutions. Communication takes through XML data message over HTTP. Such applications are very loosely coupled.

Peer-to-peer communication between EJB containers offers a more tightly coupled model that works well for intranet solutions. In these types of applications, seriously consider using message-driven enterprise beans and Java Message Service (JMS), which enable you to develop loosely coupled applications.

Configuring the target server settings

Before you begin creating Enterprise JavaBeans, you must configure the settings for the application server to which you are going to deploy your enterprise beans.

Tip Be sure to see the server-specific chapters for the various servers to find additional configuration information for each server. These are the chapters:

- [Chapter 5, “Using JBuilder with Borland servers”](#)
- [Chapter 6, “Using JBuilder with BEA WebLogic servers”](#)
- [Chapter 7, “Using JBuilder with IBM WebSphere servers”](#)
- [Chapter 8, “Using JBuilder with Sybase servers”](#)
- [Chapter 9, “Using JBuilder with Sun iPlanet servers”](#)
- [Chapter 10, “Using JBuilder with your web server”](#)

Supported servers

JBuilder supports the following servers:

- Borland Enterprise Server 5.1.1, 5.2, and 5.2.1
- BEA WebLogic Server 6.0 SP 2 (and all back versions of 6.0 service packs)
- BEA WebLogic Server 6.1 SP 4 (and all back versions of 6.1 service packs)
- BEA WebLogic Server 7.0 SP 2 (and all back versions of 7.0 service packs)
- BEA WebLogic Platform 8.1

- IBM WebSphere Application Server 4.0.1 Single Server
- IBM WebSphere Application Server 4.0.1 Advanced Edition
- IBM WebSphere Application Server 5.0
- Sun iPlanet Application Server 6.0 SP 4
- Sun iPlanet Application Server 6.5 SP 1
- Sybase Enterprise Application Server 4.1.3
- Sybase Enterprise Application Server 4.2
- Tomcat 3.3.1
- Tomcat 4.0.6
- Tomcat 4.1.12

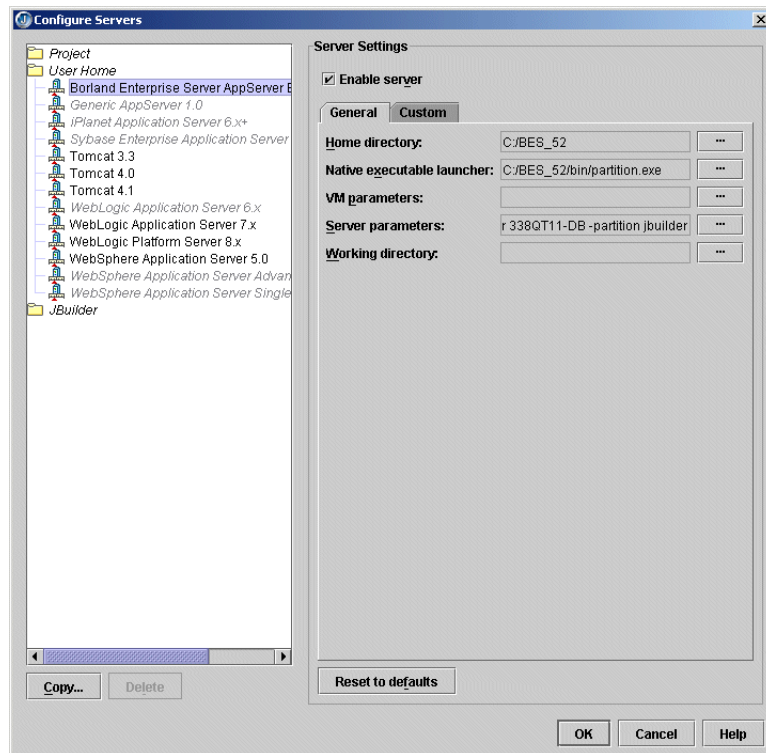
Setting up servers within JBuilder

This is a feature of
JBuilder Developer and
Enterprise

To configure the settings to target one or more application servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears:



The left side of the dialog box lists the servers that can be configured in JBuilder and for which JBuilder finds a registered OpenTool.

- 2 Select the server you want to configure by clicking it in the pane on the left.

The right side of the dialog box lists the default settings for the selected server. Each application server except Generic App Server 1.0 has both a General and a Custom page for editing settings. The General page has fields that all the application servers have in common, while the Custom page has fields that are specific to the selected application server. In some cases, modifying a Custom setting will update a setting on the General page and vice versa.

The Generic AppServer 1.0 server option is a generic option. It represents a basic application server that supports EJB 1.1 and/or EJB 2.0 development. Select it if the application server you use is not currently supported by JBuilder. You will probably want to edit the resulting deployment descriptor with tools supplied with that application server to get the exact settings you want. You could also choose this option if the you aren't targeting a specific application server.

- 3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Server page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed. Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings and usually selects the control representing the missing setting.

- 4 If you want to change any of the settings, click the ... button next to the field and make your changes.

You can add any necessary application dependencies to the server's classpath by clicking the Required Libraries tab and using the Add button to add the libraries the server needs.

- 5 Click the Custom tab to view fields unique to the server. Most application servers allow you to specify the location of the JDK the server uses; some require you to specify the JDK location for successful configuration completion. Updates you make on the Custom page can update settings on the General page.

Following are the JDKs and their installed locations for the application servers supported by JBuilder:

- Borland Enterprise Server 5.1.1: `<bes home>/jdk`
- Borland Enterprise Server 5.2/5.2.1: `<bes home>/jdk/jdk1.3.1` or `<bes home>/jdk/jdk1.4.1` (the default value)
- WebLogic Server 6.0: `<bea home>/jdk130`
- WebLogic Server 6.x: `<bea home>/jdk131`
- WebLogic Server 7.x: `<bea home>/jdk131_06`
- WebLogic Platform Server 8.1: `<bea home>/jdk141_02`. WebLogic 8.1 also supports JRockit (a JDK optimized for server applications), so the JDK directory could be `<bea home>/jrockit81_141_02`
- WebSphere 4.0 (either Advanced Edition or Single Server): `<websphere home>/java`
- WebSphere 5.0: `<websphere home>/java`
- Sybase Enterprise Application Server 4.1.x and 4.2: `<Sybase home>/_jvm`
- iPlanet 6.x: `<iPlanet home>/usr/java`

Once you've made your edits and click the OK button to accept your changes and close the dialog box, JBuilder runs a validation pass on the settings for the server. If you prefer that JBuilder not try to validate your changes, uncheck the Enable Server dialog box. Your changes will still be saved when you click OK, but you can then go back later to make further changes that you need before choosing to validate the settings. You must check the Enable Server check box, however, before you can select the server you are configuring for use in a project. Selecting OK saves changes for all servers you've enabled, disabled, or modified. If, while using the dialog box, you move between servers, JBuilder attempts to validate the server you are leaving.

6 Click OK when you are through configuring the application server.

When you click OK and the Enable Server check box is checked, JBuilder attempts to verify your settings. If errors are detected, a message box containing information about the errors appears. If the Enable Server option is unchecked, the dialog box simply closes and no setting validation is performed.

Once a server is properly configured and validated, the server you configured is listed in black type in the pane on the left. Gray type indicates the server hasn't been configured yet. Red type indicates an error condition; specifically it means the server's library containing its settings has been read, but the OpenTool representing the server can't be

found. You can delete a server shown in red by selecting it and clicking the Delete button.

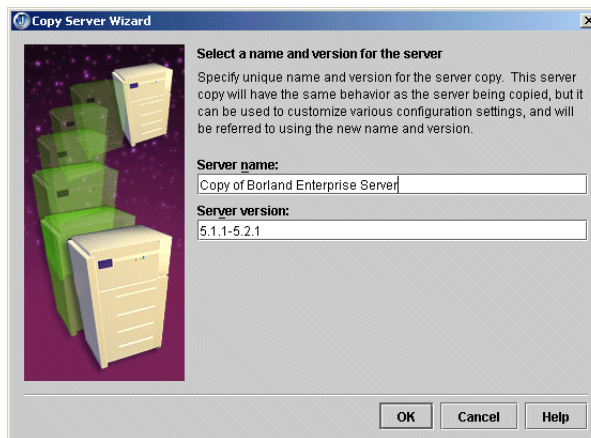
The Reset To Defaults button sets the settings back to the values they originally had when JBuilder was first installed and the server is disabled.

If a dialog box appears with a message that you must restart JBuilder, you must close and then restart JBuilder for changes to become effective; otherwise, restarting JBuilder is not necessary.

Creating a duplicate configuration to edit

Once you've created a server configuration, you might find you'd like to have a similar one, but with some slight differences. Follow these steps to create a duplicate configuration you can then edit:

- 1 Choose Tools | Configure Servers to display the Configure Servers dialog box.
- 2 Select the server configuration you would like to duplicate from the left pane.
- 3 Click the Copy button at the bottom of the left pane to display the Copy Server wizard:



- 4 Specify the name you want to use to identify this server configuration in the Name For This Server field.
- 5 Specify the server version in the Version For This Server field.
- 6 Click OK.
- 7 Select your new configuration in the left pane of the Configure Servers dialog box and make the changes to the settings you need using the General and Custom pages.

Copying a server configuration also creates copies of that server's tools. You could, therefore, create a duplicate server configuration and modify just the server's tools as the only difference between the two configurations. As another example, you could have the copied configuration refer to a different home directory if you have slightly different versions of that server installed.

Adding a service pack

If you want to add a service pack to your application server configuration, follow these steps:

- 1 Choose Tools | Configure Servers.
- 2 Select your server from the list of servers in the pane on the left side of the dialog box.
- 3 Click the General tab, then the Class tab, if it isn't already selected.
- 4 Click Add.
- 5 Navigate to the location of the service pack JAR file.
- 6 Click OK twice to close all dialog boxes.

The created libraries

When you configure the server settings, one or more libraries are created for you automatically that contain all the application server files you will need for enterprise bean development for the application server of your choice. These are the libraries created for you, listed by application server:

- **Borland Enterprise Server 5.1.1 - 5.2.1**

Borland Enterprise Server 5.1.1 - 5.2.1 Client: All JARs needed to run a client.

Borland Enterprise Server 5.1.1 - 5.2.1 Servlet: Used for web applications.

- **Sybase Enterprise Application Server 4.x**

EAServer 4.x Client: All JARs needed to run a client.

EAServer 4.x Servlet: Used for web applications.

- **WebLogic 8.1**

WebLogic 8.1 Client: All JARs needed to run a client.

WebLogic 8.1 Deploy: JARs needed to run the WebLogic 8.1 deploy tool.

WebLogic 8.1 Servlet: Used for web applications.

- **WebLogic 7.x**

WebLogic 7.x Client: All JARs needed to run a client.

WebLogic 7.x Deploy: JARs needed to run the WebLogic 7.x deploy tool.

WebLogic 7.x Servlet: Used for web applications.

- **WebLogic 6.x**

WebLogic 6.x Client: All JARs needed to run a client.

WebLogic 6.x Deploy: JARs needed to run the WebLogic 6.x deploy tool.

WebLogic 6.x Servlet: Used for web applications.

- **WebSphere 5.0**

WebSphere 5.0 Client: JARs needed to run a client.

WebSphere Application Server 5.0 Ext Dirs: JARs used during server startup.

WebSphere Application Server 5.0 EJB Deploy: JARs used to compile enterprise beans and create stubs.

WebSphere Application Server 5.0 Servlet: Used for web applications.

- **WebSphere 4.0 Single Server**

WebSphere AES 4.0 Client: JARs needed to run a client.

WebSphere AES 4.0 Ext Dirs: JARs used during server startup.

WebSphere AES 4.0 EjbDeploy: JARs used to compile enterprise beans and create stubs.

WebSphere AES 4.0 SeAppInstaller: JARs used to run the WebSphere 4.0 deploy tool (SeAppInstaller).

WebSphere AES 4.0 Servlet: Used for web applications.

- **WebSphere 4.0 Advanced Edition**

WebSphere AE 4.0 Client: JARs needed to run a client.

WebSphere AE 4.0 Ext Dirs: JARs used during server startup.

WebSphere AE 4.0 XmlConfig: JARs used to run the WebSphere 4.0 AE deploy tool.

WebSphere AE 4.0 EjbDeploy: JARs used to compile enterprise beans can create stubs.

WebSphere AE 4.0 Servlet: Used for web applications.

- **iPlanet 6.x**

iPlanet 6.x Client: JARs needed to run a client.

iPlanet 6.x Servlet: Used for web applications.

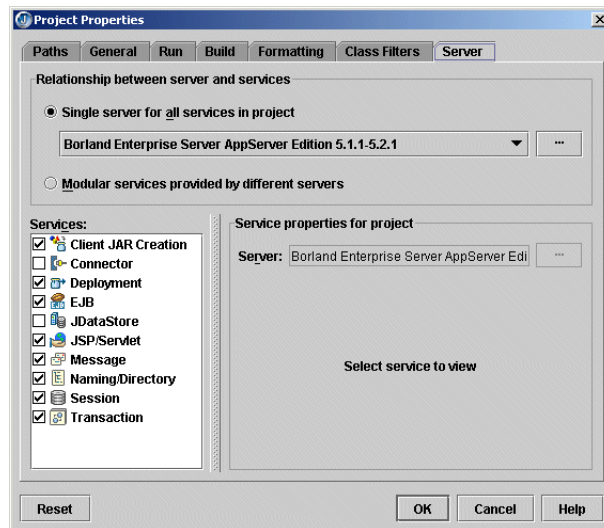
Selecting a server

JBuilder can target multiple servers. You can choose a single application server for all stages of EJB and web application development, or you can choose different servers for different aspects of development. For example, you can select one server to use to develop enterprise beans and another to develop web applications.

To select one or more servers to use for your project,

- 1 Choose Project | Project Properties.
- 2 Click the Server tab.

The Server page appears:



- 3 Decide whether you want to use a single server for all aspects of development or different servers to handle different areas of development.
 - To use a single server,
 - 1 Select the Single Server For All Services In Project option and select the server from the drop-down list.
 - 2 If you want to make changes to the configuration settings for the server, click the ... button and edit the settings you want on the General and Custom pages. Click OK.

You can also use this dialog box to select a different server.
 - 3 If you don't want a particular service started when the server starts up, uncheck the check box next to the service in the Services list. This feature currently applies only to the Borland Enterprise

Server as it is the only one which allows you to start and stop selected services.

If you uncheck the Deployment or Client JAR Creation services, the corresponding menu items will be disabled on the JBuilder Tools menu. If you uncheck the JSP/Servlet service, web server support is disabled. If you uncheck the Naming/Directory service for Tomcat, web server support is disabled.

- To use different servers for different services,
 - 1 Select the Modular Services Provided By Different Servers option.
 - 2 Check the check boxes next to all the services for which you want to specify an application server in the Services list.
 - 3 Click one of the checked services to select it in the Services list.
 - 4 In the Service Properties For Project group box, use the Server drop-down list to select the server you want to use for this service.

If server-specific properties are available for the server you selected for the particular service you are working with, they will appear below the Server drop-down list. If the Deployment service is selected, a Build Target For Deploying To Server drop-down list appears on the Service Properties For Project panel. Select the type of build you want to occur. If the EJB service is selected, you'll see read-only information appear that is intended to help you decide which server is appropriate for your needs.

For Borland Enterprise Server: The Deploy `jndi-definitions.xml` option determines whether the `jndi-definitions.xml` file is included in any of the deployment actions you select. Uncheck it if you don't want it included. `jndi-definitions.xml` is an XML deployment descriptor for EJB 2.0 resources.

For WebLogic 6.x - 8.1: If you select the JSP/Servlet service, the service properties include a Map Project Apps At Runtime option. When this option is selected (the default value) all web applications in the project deploy from the web application directory and not as a WAR when the server starts up.

For WebLogic 7.x - 8.1: If you select the EJB service, the service properties include a combo box with options to deploy (auto-deploy) data sources for all EJB modules in the project. These are the options:

- Map Project Data Sources: Maps data sources for all EJB modules in `config.xml` for the domain specified in the server configuration. DataSource entries are removed when the server shuts down.
- Map And Persist Project Data Sources: Maps data sources for all EJB modules in `config.xml` for the domain specified in the

server configuration. Data Source information is persisted in `config.xml`. DataSource entries are removed when the server shuts down.

- Do Not Map Project Data Sources: Does not map any data sources.

Note that JBuilder also adds the JDBC driver (if it can be located in the project's required libraries) to the server's classpath at start up time. All data sources created by JBuilder are transactional data sources.

- 5 If you want to make changes to the configuration settings for the server selected for the service, click the ... button and edit the settings you want on the General and Custom pages. Click OK.
- 6 Repeat these steps for each service you want access to.

- 4 Click OK.

If you discover that the wizards on the Enterprise or Web page of the object gallery are disabled, no application server is selected for your current project or the selected server is not configured. Follow the steps just described to select an application server.

Setting up JDBC drivers

To enable JBuilder's EJB wizards and EJB Designer to access a database, you must install the JDBC driver supplied by the database vendor and set up the driver in JBuilder. Install a JDBC driver following the vendor's instructions.

To begin setting up the driver in JBuilder, select Tools | Enterprise Setup to display the Enterprise Setup dialog box. Click the Database Drivers tab to display the Database Drivers page. Use this page to add a new database driver to JBuilder.

Creating the .library and .config files

There are three steps to adding a database driver to JBuilder:

- 1 Creating a library file which contains the driver's classes, typically a JAR file, and any other auxiliary files such as documentation and source.
- 2 Deriving a .config file from the library file which JBuilder adds to its classpath at start-up.
- 3 Adding the new library to your project, or to the Default project if you want it available for all new projects.

The first two steps can be accomplished from the Database Drivers page:

- 1 Open JBuilder and choose Tools | Enterprise Setup. Click the Database Drivers tab which displays .config files for all the currently known database drivers.
- 2 Click Add to add a new driver, then New to create a new library file for the driver. The library file is used to add the driver to the required libraries list for projects.

Note You can also create a new library under Tools | Configure Libraries, but since you would then have to use Enterprise Setup to derive the .config file, it is simpler to do it all here.

- 3 Type a name and select a location for the new file in the Create New Library dialog box.
- 4 Click Add, and browse to the location of the driver. You can select the directory containing the driver and all its support files, or you can select just the archive file for the driver. Either will work. JBuilder will extract the information it needs.
- 5 Click OK to close the file browser. This displays the new library at the bottom of the library list and selects it.
- 6 Click OK. JBuilder creates a new .library file in the JBuilder /lib directory with the name you specified (for example, InterClient.library). It also returns you to the Database Drivers page which displays the name of the corresponding .config file in the list which will be derived from the library file (for example, InterClient.config).
- 7 Select the new .config file in the database driver list and click OK. This places the .config file in the JBuilder /lib/ext directory.
- 8 Close and restart JBuilder so the changes to the database drivers will take effect, and the new driver will be put on the JBuilder classpath.

Important If you make changes to the .library file after the .config file has been derived, you must re-generate the .config file using Enterprise Setup, then restart JBuilder.

Now that JBuilder can see the database driver, you must add the database driver library to the Required Libraries list in Project | Properties, or Project | Default Properties.

Adding the JDBC driver to projects

Projects run from within JBuilder use only the classpath defined for that project. Therefore, to make sure the JDBC driver is available for all new projects that will need it, define the library and add it to your default list

of required libraries. This is done from within JBuilder using the following steps:

- 1 Start JBuilder and close any open projects.
- 2 Choose Project | Default Project Properties.
- 3 Select the Required Libraries tab on the Paths page, then click the Add button.
- 4 Select the new JDBC driver from the library list and click OK.
- 5 Click OK to close the Default Project Properties dialog box.

Note You can also add the JDBC driver to an existing project. Just open the project, then choose Project | Properties and use the same process as above.

Updating projects with the latest server settings

Every time you open a project, JBuilder updates it with the latest server settings for the project's selected server(s). So, for example, if you have modified server settings for one project and you have others that aren't open that use the same server configuration, the next time you open those other projects, your modified server settings will be in force automatically. If you want to use a similar but unique server configuration for a particular project, create a duplicate configuration using the Copy button, and use the Copy Servers wizard to edit it to your needs.

Be aware that this automatic updating of projects with the latest server settings can occur only with server configurations modified in JBuilder 9. If you attempt to transfer server libraries from a previous JBuilder version, your projects won't be updated with them. You can still tell JBuilder to update the project manually:

- 1 Right-click the project node of the project you want to update in the project pane and choose Properties.
- 2 Select the Servers page.
- 3 Click the ... button next to your selected single server or the separate service servers and make your changes.
- 4 Click OK to close the dialog box.

Using JBuilder with Borland servers

This chapter explains how to set up and use Borland servers with JBuilder. JBuilder supports Borland Enterprise Server 5.1.1, 5.2, and 5.2.1.

Configuring JBuilder for Borland servers

To configure JBuilder settings to target Borland servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears.

- 2 Select the Borland server you want to configure by clicking it in the pane on the left.

The right side of the dialog box lists the default settings for the selected server. The General page has fields that all servers have in common, while the Custom page has fields that are specific to the selected server. In some cases, modifying a Custom setting will update a setting on the General page.

- 3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Servers page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed.

Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings.

- 4 If you want to change any of the settings, click the ... button next to the field and make your changes. Assuming you installed Borland Enterprise Server into `[drive]:/BES`, that will be your Home Directory. If it isn't, use the ... button next to the Home Directory field to navigate to where the server is installed.
- 5 Click the Custom tab to view fields unique to the server. Change or fill in these fields:
 - **JDK Installation Directory:** The directory where the JDK version required by the server is located. JBuilder fills this field in automatically, depending on the value you specified for the Home Directory on the General page. For Borland Enterprise Server 5.1.1, the directory is usually the `jdk` subdirectory of the Home Directory. For example, if you installed the server into the `[drive:]/BES` directory, the JDK Installation Directory should be `[drive:]/BES/jdk`. For Borland Enterprise Server 5.1.1, the directory is the `jdk/jdk1.3.1` or `jdk/jdk1.4.1` (the default) subdirectory of the Home Directory.
 - **Server Name:** The name of the server you are configuring. By default, JBuilder specifies the identifier of your machine.
 - **Partition Name:** The name of the partition in which the server is run. For more information about partitions, see [“Partitions, partition services, and J2EE APIs” on page 5-4](#).
 - **Add A Management Agent Item To The Tools Menu:** Adds a Management Agent item to the JBuilder Tools menu, so you can start the Management Agent quickly from within JBuilder IDE.
 - **Add A SonicMQ Broker Agent Item To The Tools Menu:** Adds the SonicMQ Broker Agent item to the JBuilder Tools menu, so you can start the SonicMQ Broker Agent quickly from within JBuilder IDE.
 - **Add BES Console To My Project:** Adds the Borland Enterprise Server Console to as a tab in the project pane of projects that target Borland Enterprise Server. This option is valid only for Borland Enterprise Server 5.2.1.
 - **Server Realm:** The realm of the server. For more information about server realm, see your Borland Enterprise Server documentation.
 - **User Name:** The name you use to identify yourself to the server. The default value is `admin`.
 - **User Password:** The password you use to identify yourself to the server. The default fault is `admin`.

- **Advanced Settings:** Clicking this button displays an Advanced Settings dialog box. Use this dialog box if you want to change the number of the port used by the Management Agent and if you want to select the Use Security option. The management port is used in JBuilder to detect the server during startup and deployment. Change the management port only if you are deploying to a remote server with a management port that is different from the default. When you change the port number, please ensure that you entered the correct port number as configured in the server, as the server won't start up without the correct port number. The port and security settings you choose must match the settings of your server. JBuilder updates the values in this dialog box whenever the home directory changes. The values are read from the Borland Enterprise Server property files.

Making the ORB available to JBuilder

When you use Tools | Configure Server to set up the Borland Enterprise Server 5.1.1 - 5.2.1, the server will create its own VisiBroker configuration called "VisiBroker <server name and version>." The default VisiBroker configuration won't be changed. You can see your current settings on the CORBA page of the Tools | Enterprise Setup dialog box. For the VisiBroker configuration created specifically for your version of the server, you should check none of the options of the CORBA page. You might want to use this page to set the VisiBroker Smart Agent port to a unique number. All client/server communication takes through the Smart Agent port. Also, to add a command to start the Smart Agent to the Tools menu, check the Add The VisiBroker Smart Agent Item To The Tools Menu option. This option is selected by default.

Note If you have not installed Borland Enterprise Server, you will have to complete the following steps before you will be able to use VisiBroker's idl2java or java2iiop:

- 1 Choose Project | Project Properties.
- 2 Click the Build tab and then on the Build page, click the IDL tab.
- 3 Choose VisiBroker as your IDL compiler if its not already selected.
- 4 In the Additional Options field add this if your server version is 5.1.1:

```
-VBJprop borland enterprise.licenseDir=<visibroker license directory>
-VBJjavavm<jdk1.3.1 java or javaw path>
```

If your server version is 5.2 or 5.2.1, specify these options in the Additional Options field:

```
-VBJprop borland enterprise.licenseDir=<visibroker license directory>
-VBJjavavm<jdk1.4.1 java or javaw path>
```

The correct license directory is `<BESHome>\var\servers\<server directory>\adm.` The `<server directory>` is same as the server name assigned when you configured the server using Tools | Configure Servers. You can find the server name on the Custom page of this dialog box.

5 Click OK.

You must still perform one step: starting the VisiBroker Smart Agent. This handles the initial bootstrap issues such as how the client locates the naming service and so on.

To start the Smart Agent, choose Tools | VisiBroker Smart Agent.

For Borland Enterprise Server 5.1.1 - 5.2.1: If you choose Tools | Borland Enterprise Server Management Agent, the VisiBroker Smart Agent is also started as part of the process. When you are working with the Borland Enterprise Server AppServer Edition 5.1.1 - 5.2.1, you must start the Borland Enterprise Server Management Agent instead of the VisiBroker Smart Agent. If you don't start it, it is automatically started for you when you start the server within JBuilder. Note that starting the server automatically might take longer because the server searches for any running agents. The most efficient way to start the Management Agent is to start it yourself.

For more information see "Setting up JBuilder for CORBA applications" in *Developing Enterprise JavaBeans*.

Partitions, partition services, and J2EE APIs

Partitions provide containers and services for hosting your applications. Any or all of a partition's containers and services can be enabled or disabled. A server can have multiple partitions, and because each partition is a separate process, an application's functions can be distributed across multiple processes (partitions).

Additionally, a partition can be cloned (copied) on the same server or on another enterprise server located on the same local area network. Cloning a partition can save significant time in the deployment process because it not only copies all of the deployed modules, but also copies all the associated service configurations settings which can require a considerable amount of time and effort to complete.

Application components are hosted in either the web container or EJB container, or simply in the partition itself (in the case of connectors). You can deploy application EARs or smaller archives to partitions. The partitions are "smart" and will place your application components in the proper containers. Since you may create as many partitions as you wish,

you can also have as many container instances as you wish. Borland provides up to two different containers per partition instance:

- Web container (Tomcat 4.0): for hosting JSPs and servlets, and
- EJB container (Borland): for hosting EJB components

Each partition also provides Partition Services and J2EE APIs as Partition Services. These services and APIs are:

- Naming Service: in Borland Enterprise Server, the Naming Service is managed by the VisiBroker ORB
- JDataStore: Borland's all-Java database
- JDBC: for getting connections to and modeling data from databases
- Java Mail: a Java email service
- JTA: the Java transactional API
- JAXP: the Java API for XML parsing
- JNDI: the Java Naming and Directory interface
- RMI-IIOP: remote method invocation (RMI) carried out via internet inter-ORB protocol (IIOP)

The management agent manages all partitions. The default management port is 42424. By default, the server is installed with the "standard" partition.

JBuilder, by default, uses a partition named "jbuilder" as a deployment target for Borland Enterprise Server 5.1.1 - 5.2.1. If this partition doesn't exist, one will be created for you automatically. This "jbuilder" partition is a copy of the "standard" partition and shares the same port number with Tomcat and JDataStore services.

You can start multiple partitions using JBuilder run configurations. To create multiple run configurations, follow these steps:

- 1 Choose Run | Configurations.
- 2 Click New.
- 3 Click the Server tab.
- 4 Edit the Server parameters and change the partition name to the one you want to use.
- 5 Repeat these steps to create additional run configurations to run other partitions.

Please ensure that you have configured unique port numbers for Tomcat and JDataStore services in the server before starting up the partition.

Also ensure that you have the naming service enabled for only one of the partitions to avoid naming service conflicts. To disable the naming service for a partition, edit the run configuration for the partition and uncheck the naming service option.

Changing the management port

To change the management port,

- 1 Open the Borland Enterprise Server Console (for Borland Enterprise Server 5.2.1, you can access the BES Console from within JBuilder).
- 2 Double-click Installations.
- 3 Right-click the partition that uses the port.
- 4 Select Update Ports from the context menu.
- 5 Use the Configure Ports wizard that appears to change the management port number.

Starting the server

Management agent

The management agent must be started before any server related actions (server startup, deployment, and so on) are performed. Start the management agent using Tools | Borland Enterprise Server Management Agent. This will start a server process (*ias*), the management os agent, and the CORBA os agent. After the management agent has started up, launch the partition by right clicking on the EJB or Web module in the project pane and select Run Using Defaults. You can also create a Server run configuration and run the project using this configuration.

Running the partition in JBuilder accomplishes these things:

- Creates the partition, if not present in the server. The partition name used in this step is derived from the run configuration used to startup the server. If you are starting the server using the default configuration, the partition name as configured in the application server properties will be used.
- Deploys any resources defined in *jndi-definitions.xml* (if present) to the root of the partition directory, for example, `APPSERVER_HOME\var\servers\SERVER_NAME\partitions\PARTITION_NAME\`.

This file will be created if your project contains an EJB 2.0 module with data sources/messaging resources defined.

- Note** This action can be turned off by unchecking Deploy jndi-definitions.xml in the service properties for Deployment on the Server tab of the Project Properties dialog box.
- Copy selected libraries to the partition's lib directory. All archives in this directory will be added to the partition's classpath when the partition is started up.
The directory on the server is `APPSERVER_HOME\var\servers\SERVER_NAME\partitions\PARTITION_NAME\lib`. To select libraries that you want copied over to the partition, select from the list by clicking on the Libraries setting in the server run configuration.
 - Remove any archives deployed to the partition, if this option is selected. This can be set on the Archives setting in the server run configuration. Select the option Remove Archives Already Deployed To The Server, if you want to start with a clean partition.
 - Deploy archives if selected. By default all deployable archives in the project are selected. To change this, click on the Archives setting in the server run configuration and unselect any archives that you do not wish to deploy.
 - Start the partition. When partition startup is complete, you should see the message 'Partition <PARTITION_NAME> has been started'. Archives deployed at startup should be loaded and accessible.
- Note** By default, all services associated with a partition are started. This could cause the partition to take a longer time to start. To reduce startup time, uncheck the services that you don't require in the Server run configuration.

Remote deploying

To prepare to remote deploy EJBs, WARs, and EAR modules, follow these steps:

- 1 Choose Tools | Configure Servers.
- 2 Select Borland Enterprise Server 5.1.1-5.2.1.
- 3 Set the Server and Partition names to match that of the remote server.
- 4 Click Advanced Settings and make sure the Management Port matches your server configuration.
- 5 Click OK.
- 6 Choose Tools | Enterprise Setup.
- 7 Click the CORBA tab, and on the CORBA page, set the Smart Agent Port to match that of your server configuration.
- 8 Choose OK.

Now you are ready to deploy. There are two ways to deploy EJBs, WARs, and EAR modules using JBuilder:

- Deploying to the server using the deployment wizard: Tools | Enterprise Deployment. This is a Borland Enterprise Server tool and requires the management agent to be started. The wizard will detect all partitions on the server, even if they have not been started. Select the appropriate partition from the drop-down list and click on Finish to deploy the archive. If the partition has already been started, restart the partition to access the deployed archive.
- Deploying to the server using the context menu option on any deployable node. Right-click the deployable node to see the Deploy commands. Choose Deploy. You can also select multiple deployable nodes, right-click them, and use the context menu that appears to deploy more than one module.

Remote debugging

To prepare to debug your application remotely, choose one of these methods:

- Before the server starts:
 - a** Start the Borland Enterprise Console. (You can click the BES Console tab in the project pane if you are using Borland Enterprise Server 5.2.1 and have selected the BES Console option on the Custom page of the Tools | Configure Servers dialog box.)
 - b** Click Installations.
 - c** Locate the partition you wish to debug in JBuilder.
 - d** Click the patron node in the structure pane.
 - e** Search for and set the debug parameters as follows in the content pane:

```
partition.enable_jpda_debug=true
partition.jpda.transport_address=3999
partition.jpda.suspend=false
```

- Starting the server from the command line:
 - a** Start the server from the command line.
 - b** Start the Borland Enterprise Server Console. (You can click the BES Console tab in the project pane if you are using Borland Enterprise Server 5.2.1. and have selected the BES Console option on the Custom page of the Tools | Configure Servers dialog box.)
 - c** Click Enterprise Servers.

- d** Locate the partition you want to debug (under the Enterprise Servers node).
- e** Right-click and select Configure.
- f** Check the Enable JPDA Remote Debugging option
- g** If you are using Borland Enterprise Server 5.2 or 5.2.1, continue with these instructions:
 - 1** Set the transport address field to 3999.
 - 2** Uncheck the Suspend Partition Until Debugger Attaches option.
- h** If you are using Borland Enterprise Server 5.1.1, continue with these instructions:
 - 1** On the General tab, add these parameters in the Additional Command Line Arguments field:


```
-jpda: running,address=3999
```
- Starting the server and partition from the command line:
 - a** Start the server from the command line without starting the partition.
 - b** Start the partition from the command line:


```
<APPSERVER_HOME>\bin\partition.exe
-Dpartition.externally_managed_mode=true
-Dpartition.register_agent_with_server_agent=true
-Ddvbroker.agent.port=<smart agent port>
-partition <partition name> -server <server name>
-jpda:running,address=3999
```

After following one of the methods, follow these steps within JBuilder:

- 1** In the project from which you want to launch the remote debug session, click Run | Configurations. If you have not yet created a server type run configuration, click New and select type Server. If you have already created a server type run configuration, select it and choose Edit.
- 2** Click the Debug tab.
- 3** Check the Enable Remote Debugging option.
- 4** Enter the host name (the name of the machine on which the server is running.)
- 5** Click OK.
- 6** Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited. The debugger launches and attaches to the remote process. You will now be able to set breakpoints in Java code, such as EJBs, servlets, and so on.



Web Application workarounds

The default application server install includes ROOT.war, which contains the default context. If you deploy an EAR that contains a default context, you must delete ROOT.war (or rename its extension) so that it does not load and cause a conflict. If deploying WARs (with the container classloader policy), the resulting WAR is automatically copied to the partition as ROOT.war, in which case there is no conflict. Please note that it is the `<context-root>!ROOT!</context-root>` in the web-borland.xml file that designates the context as root, not the file name.

If you are working on a named context but have the root context in an already deployed archive, the internal web browser in JBuilder will attempt to load once that archive is loaded by the container, because the root context can potentially match any URL. You'll then probably get a 'Document not found' error. To prevent this remove ROOT.war if you're not using it or overwriting it.

Using the Borland Enterprise Server console in JBuilder

If you have selected Borland Enterprise Server 5.2.1 as your target application server, a BES Console tab appears in the JBuilder project pane. Click the tab and the Borland Enterprise Server Console appears within JBuilder, enabling to work with the server without leaving the JBuilder environment.

International issues

The Borland Enterprise Server 5.1.1 - 5.2.1 does support Japanese characters in the JNDI name.

Using JBuilder with BEA WebLogic servers

This chapter explains how to use to set up and use BEA WebLogic servers with JBuilder. JBuilder supports WebLogic Server 6.x, WebLogic Server 7.x, and WebLogic Platform 8.1.

Service packs

JBuilder requires WebLogic 6.0 with Service Pack 2 or WebLogic 6.1 with Service Pack 4 for proper operation. For WebLogic 7.0, Service Pack 2 is required.

Configuring JBuilder for WebLogic servers

To configure JBuilder settings to target WebLogic servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears.

- 2 Select the WebLogic server you want to configure by clicking it in the pane on the left.

The right side of the dialog box lists the default settings for the selected server. The General page has fields that all WebLogic servers have in common, while the Custom page has fields that are specific to the selected server. In some cases, modifying a Custom setting will update a setting on the General page.

3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Servers page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed. Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings.

4 If you want to change any of the settings, click the ... button next to the field and make your changes. Assuming you installed WebLogic Platform 8.1 into a BEA_HOME directory with the name bea, your home directory should be [drive]:/bea/weblogic81/server. For WebLogic Server 7.x, your home directory should be [drive]:/bea/weblogic71/server. For WebLogic 6.x, your home directory should be [drive]:/bea/wlserver6.0 or [drive]:/bea/wlserver6.1.

For WebLogic Platform 8.1 and WebLogic Server 7.x, the working directory is set when you set the domain directory using the Custom page. For now, you can leave it unchanged. For WebLogic 6.x, the home directory is the default working directory. It is not necessary to change this.

5 Click the Custom tab to view fields unique to the server. Change or fill in these fields:

- **BEA Home Directory:** The BEA Home Directory is the directory into which you installed the WebLogic Server and that contains the file `registry.xml`. JBuilder assigns a default value for the BEA Home Directory that varies depending on the version of the WebLogic Server installed.
- **JDK Installation Directory:** For WebLogic Platform 8.1, the directory is `<bea home>/jdk141_02`. For WebLogic 7.x, the directory is `<bea home>/jdk131_06`. For WebLogic 6.x, the directory is `<bea home>/jdk131`.
- **Domain Directory:** For WebLogic Platform 8.1 and WebLogic Server 7.x only. Use the ... button to select a directory as your domain directory. The domain directory is the directory where your WebLogic domain is stored. Usually this is `user_projects/mydomain`. Setting the domain directory sets the working directory value on the General page. For more information about WebLogic domains, see your WebLogic documentation.

- **Domain Name:** For WebLogic 6.x only. The WebLogic domain name. JBuilder sets the domain name by default to “mydomain.” For more information about WebLogic domains, see your WebLogic documentation.
- **User Name:** The user name you use to authenticate yourself to the server. This field appears for WebLogic Platform 8.1 and WebLogic Server 7.x.
- **Password:** The password you use to authenticate yourself to the server.
- **Server Name:** The name you specify is used as a VM parameter for starting the server. JBuilder suggests a default name of “myserver.”
- **Listen Address:** Specifies a listen address for this server. The host value must be either the DNS name or the IP address of the server. If you do not specify a Listen Address, the server uses either the machine’s DNS name or its IP address.
- **Listen Port:** Specifies the non-SSL listen port for this server. If you do not specify a Listen Port, the server uses 7001 as the default.
- **Version:** Select the combination of WebLogic version and service pack you are configuring from the drop-down list.

The Custom page of the Tools | Configure Servers dialog box for WebLogic 7.x and 8.x now includes Listen Address and Listen Port fields. If you edit fields such as User Name, Server Name, Listen Address, and so on, the corresponding entries in the Enterprise Deployment dialog box and the node deployment dialog box are updated accordingly.

- **Add An Admin Console Item To The Tools Menu:** Checking this check box adds the WebLogic Admin Console item to JBuilder’s Tools menu. You must also fill in the next field, Web Browser Path, to have the menu item added.
- **Web Browser Path:** Specify the path and file name of the web browser of your choice. For example, you might specify `c:/program files/Internet Explorer/iexplore.exe` if you want to use Microsoft’s Internet Explorer.
- **Add A Configuration Wizard Item To The Tools Menu:** For WebLogic Platform 8.1 and WebLogic Server 7.x only. Adds a WebLogic Configuration Wizard item to the JBuilder Tools menu. When you select this menu item, the BEA WebLogic Configuration wizard begins, which guides you through the steps of configuring the domain for your use.
- **Use External Compiler:** Check this check box if you want to use an external compiler to generate stub files instead of JBuilder’s

compiler, bcj. WebLogic Platform 8.1 uses the appc compiler, while earlier versions use the ejbc compiler.

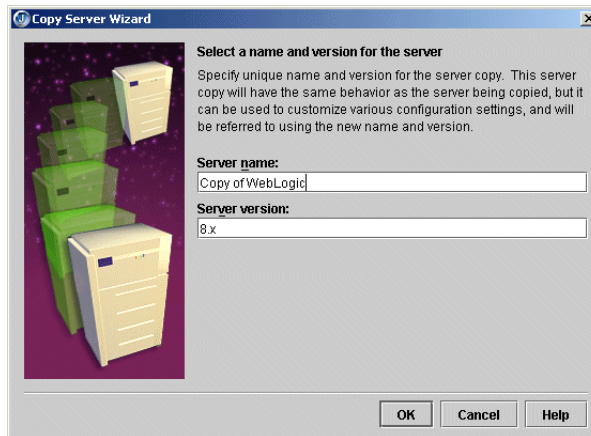
- Path: Specify the fully-qualified name of the compiler you want to use. You can use the ... button to navigate to the compiler location.

6 Choose OK to close the dialog box.

Creating a duplicate configuration to edit

Once you've created a server configuration, you might find you'd like to have a similar one, but with some slight differences. Follow these steps to create a duplicate configuration you can then edit:

- 1 Choose Tools | Configure Servers to display the Configure Servers dialog box.
- 2 Select the server configuration you would like to duplicate from the left pane.
- 3 Click the Copy button at the bottom of the left pane to display the Copy Server wizard:



- 4 Specify the name you want to use to identify this server configuration in the Name For This Server field.
- 5 Specify the server version in the Version For This Server field.
- 6 Click OK.
- 7 Select your new configuration in the left pane of the Configure Servers dialog box and make the changes to the settings you need using the General and Custom pages.

Copying a server configuration also creates copies of that server's tools. You could, therefore, create a duplicate server configuration and modify

just the server's tools as the only difference between the two configurations.

Created libraries

When you configure JBuilder to target a WebLogic server, required libraries are created for you that contain all the application server files you will need for enterprise bean development. These are the libraries created for you, listed by WebLogic server version:

- **WebLogic Platform 8.1**

WebLogic 8.1 Client: All JARs needed to run a client.

WebLogic 8.1 Deploy: JARs needed to run the WebLogic 8.1 deploy tool.

- **WebLogic Server 7.x**

WebLogic 7.x Client: All JARs needed to run a client.

WebLogic 7.x Deploy: JARs needed to run the WebLogic 7.x deploy tool.

- **WebLogic Server 6.x**

WebLogic 6.x Client: All JARs needed to run a client.

WebLogic 6.x Deploy: JARs needed to run the WebLogic 6.x deploy tool.

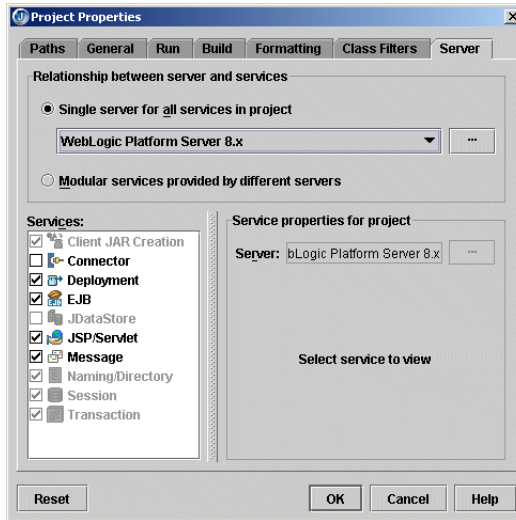
Adding a service pack

If you want to add a service pack to your application server configuration, follow these steps:

- 1 Choose Tools | Configure Servers.
- 2 Select your server from the list of servers in the pane on the left side of the dialog box.
- 3 Click the General tab, then the Class tab, if they are not already selected.
- 4 Click Add.
- 5 Navigate to the location of the service pack JAR file.
- 6 Click OK twice to close all dialog boxes.

Selecting a server

If you are working with more than one version of WebLogic Server and have configured JBuilder for all versions, you must select which version you want to use for your current project. Choose Project | Project Properties and choose Server.



You can click the Help button on this page for assistance using this page to select a server.

JBuilder can target multiple servers. You can choose a single application server for all stages of EJB and web application development, or you can choose different servers for different aspects of development. For example, you can select one server to use to develop enterprise beans and another to develop web applications.

Decide whether you want to use a single server for all aspects of development or different servers to handle different areas of development.

- To use a single server,
 - a Select the Single Server For All Services In Project option and select the server from the drop-down list.
 - b If you want to make changes to the configuration settings for the server, click the ... button and edit the settings you want on the General and Custom pages. Click OK.

You can also use this dialog box to select a different server.

- To use different servers for different services,
 - a Select the Modular Services Provided By Different Servers option.
 - b Check the check boxes next to all the services for which you want to specify an application server in the Services list.
 - c Click one of the checked services to select it in the Services list.

Note that the webapp won't be deployed automatically if it is part of an EAR.

- d In the Service Properties For Project group box, use the Server drop-down list to select the server you want to use for this service.

If server-specific properties are available for the server you selected for the particular service you are working with, they will appear below the Server drop-down list. If the Deployment service is selected, a Build Target For Deploying To Server drop-down list appears on the Service Properties For Project panel. Select the type of build you want to occur. If the EJB service is selected, you'll see read-only information appear that is intended to help you decide which server is appropriate for your needs.

If you select the JSP/Servlet service, the service properties include a Map Project Apps At Runtime option. When this option is selected (the default value) all web applications in the project deploy from the web application directory and not as a WAR when the server starts up.

For WebLogic 7.x - 8.1: If you select the EJB service, the service properties include a combo box with options to deploy (auto-deploy) data sources for all EJB modules in the project. These are the options:

- Map Project Data Sources: Maps data sources for all EJB modules in `config.xml` for the domain specified in the server configuration. Datasource entries are removed when the server shuts down.
 - Map And Persist Project Data Sources: Maps data sources for all EJB modules in `config.xml` for the domain specified in the server configuration. Data Source information is persisted in `config.xml`. Datasource entries are removed when the server shuts down.
 - Do Not Map Project Data Sources: Does not map any data sources.
- e If you want to make changes to the configuration settings for the server selected for the service, click the ... button and edit the settings you want on the General and Custom pages. Click OK.
 - f Repeat these steps for each service you want access to.

Click OK when you are finished with the dialog box to select your specified server for your current project.

If you discover that the wizards on the Enterprise or Web page of the object gallery are disabled, no application server is selected for your current project or the selected server is not configured. Follow the steps just described to select an application server.

Working in JBuilder

JBuilder supports WebLogic servers in all areas of development. This section describes some of the WebLogic-specific features available to you.

Using existing code

If you already have existing WebLogic EJB 2.0 deployment descriptors for enterprise beans you created previously, you can create a new EJB module that contains the descriptors. You have two options:

- Use the EJB Module From Descriptors wizard, which creates a new EJB module that contains the deployment descriptors.
- Use the Project For Existing Code wizard, which creates a new JBuilder project containing one or more new EJB modules that contain the deployment descriptors.

See “Creating an EJB module from existing deployment descriptors” in *Developing Enterprise JavaBeans* for complete information.

Creating WebLogic entity beans in JBuilder

JBuilder’s EJB Designer has support for WebLogic-specific features. The entity bean inspector includes options such as optimistic concurrency and field groups. There is a WebLogic version of the Table Map Editor. When creating relationships between entity beans, there are WebLogic versions of the relationship inspector and a special WebLogic RDBMS Relation Editor. For information about using these tools, see “Adding WebLogic table references” and “Specifying a WebLogic 6.x - 8.x relationship,” in *Developing Enterprise JavaBeans*.

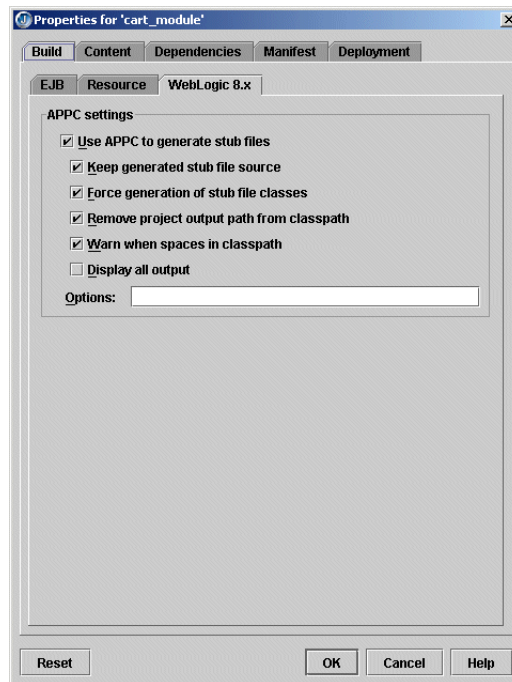
Using the Deployment Descriptor editor

The Deployment Descriptor editor presents several WebLogic-specific panels for editing WebLogic deployment descriptors. If your selected application server for your current project is WebLogic Server 6.x or later, the Deployment Descriptor editor displays an additional page when the EJB module is the current node: the WebLogic 6.x - 8.1 Properties page.

When an enterprise bean is selected in the project pane, other WebLogic-specific panels may appear if WebLogic 6.x, 7.x, or 8.1 is the target application server. The panels are a WebLogic General panel, a WebLogic Properties panel, a WebLogic Cache panel, a WebLogic Transaction Isolation panel, and an Idempotent Methods panel. Finally, a WebLogic Security Roles panel is available for WebLogic 6.x - 8.1.

Compiling

To set APPC or EJBC options for compiling your WebLogic beans, right-click the EJB module node in the project pane, choose Properties, select the Build page, and select the WebLogic page. This page contains the options for your WebLogic compiler. Here, for example, are the APPC options for WebLogic Platform 8.1:

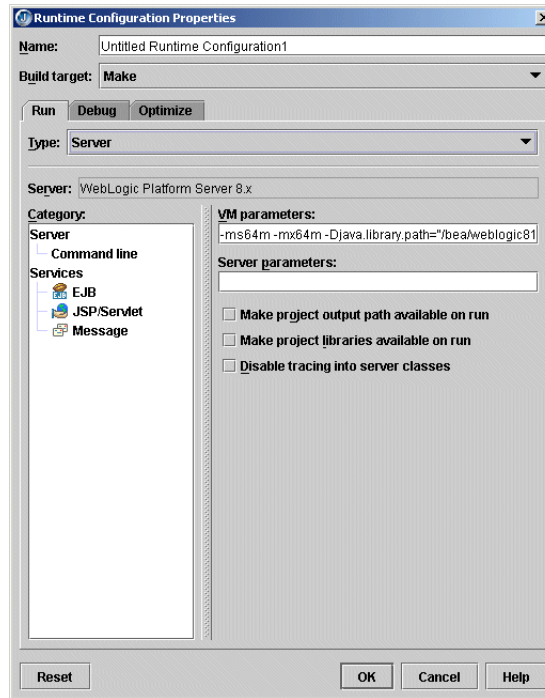


Starting the server

To prepare to start the server from within JBuilder, create a server runtime configuration. Follow these steps:

- 1 Choose Run | Configurations.
- 2 In the dialog box that appears, click the New button.

3 Select Server from the Type drop-down list.



- 4 In the Name field, enter a name, such as Server or something that will identify the run configuration.
- 5 Fill in the VM Parameters and Server Parameters needed to run the server. If you've selected a target application server as described in ["Selecting a server" on page 6-6](#), default values are already in place.
- 6 Click OK.



To start the server, click the down-arrow next to the Run Project icon and select the server run configuration you just created.

Remote deploying

To deploy a deployable module to a remote server, follow these instructions for your version of WebLogic Server:

WebLogic Application Server 7.x and WebLogic Platform Server 8.1

Choose one of these options:

- Using the Tools | Enterprise Deployment command:
 - a** Choose Tools | Enterprise Deployment to display the WebLogic Deploy Settings dialog box.
 - b** From the Action drop-down list, select Deploy.
 - c** Select the archive to deploy from the drop-down Archive To Deploy list or browser to the archive you want to deploy using the ... button.
 - d** Set the Admin URL to the remote server, such as <HOST NAME>:<port number>.
 - e** Enter the User Name and Password that matches the remote server configuration.
 - f** Choose OK.
- Deploying from the project pane:
 - a** Right-click the module you want to deploy in the project pane and choose Properties.
 - b** Click the Deployment tab.
 - c** Set the Admin URL to Set the Admin URL to the remote server, such as <HOST NAME>:<port number>.
 - d** Right-click the module you want to deploy in the project pane and choose Deploy Options | Deploy.

WebLogic Server 6.x

Choose one of these options:

- Using the Tools | Enterprise Deployment command:
 - a** Choose Tools | Enterprise Deployment to display the WebLogic Deploy Settings dialog box.
 - b** Select the Deploy action.
 - c** In the Options field, specify -host<host name> -port<port number>.
 - d** Set the password to match that required by the remote server configuration.
 - e** Choose OK.

- Deploying from the project pane:
 - a Right-click the module you want to deploy in the project pane and choose Properties.
 - b Click the Deployment tab.
 - c Set the Admin URL to Set the Admin URL to the remote server, such as <HOST NAME>:<port number>.
 - d Right-click the module you want to deploy in the project pane and choose Deploy Options | Deploy.

WebLogic 8.x Platform Server deployment

The OpenTool for WebLogic 8.x Platform Server has improved deployment features. You can now deploy submodules so you can

- Deploy a module that has been newly added to an EAR or redeploy it.
- Redeploy part of an exploded web application.

Use the deploy options on the context menu when you right-click the archive and choose Properties. You can also access the deployment options when you choose Tools | Enterprise Deployment and choose from the actions on the drop-down Action list.

You can redeploy part of an exploded web application only if you the server is remote (which means it is on a remote machine or it is running on the same machine outside JBuilder). Also only exploded WAR files can be redeployed.

To make this feature work, you must follow these steps,

- 1 Copy the web application top level directory to the <yourdomain>/applications directory of your WebLogic installation.
- 2 Start the server.
- 3 Open the config.xml file to determine the application name the server has set (usually, for instance if the web application top level directory is "webapp1", the corresponding application name will be "_appsdir_webapp1_dir").
- 4 Use this name for setting the Remote Exploded AppName field from the WebContext node deployment property page.
- 5 Right-click on any .class or .jsp files of a exploded web context node archive file. You can select multiple files.

There are two additional options on the context menu that appears when you right-click the archive and choose Deploy Options: Deactivate and Activate. Deactivate suspends deployed components, leaving staged data in place for subsequent reactivation. To reactivate a deactivated

component, choose Activate. To totally remove the application and any staged data from the server, choose Undeploy.

Remote debugging

To prepare to debug remotely, launch the WebLogic Server with debug parameters. To do this, modify `startWebLogic.sh` in your domain directory and add the follow parameters to the `JAVA_OPTIONS` variable:

```
-Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=3999,suspend=n
```

Within JBuilder, follow these steps:

- 1 In the project from which you want to launch the remote debug session, click Run | Configurations. If you have not yet created a server type run configuration, click New and select type Server. If you have already created a server type run configuration, select it and choose Edit.
- 2 Click the Debug tab.
- 3 Check the Enable Remote Debugging option.
- 4 Enter the host name (the name of the machine on which the server is running.)
- 5 Click OK.
- 6 Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited. You will now be able to set breakpoints in Java code, such as in EJBs, servlets, and so on.



Remote debugging of JavaServer Pages

To debug JSPs remotely, open the project containing the web application you want to debug and follow these steps,

- 1 Expand the web application node in the project and expand the deployment descriptors node under the web application node.
- 2 Double-click `weblogic.xml` and add the following descriptor elements:

```
<jsp-descriptor>
  <jsp-param>
    <param-name>keepgenerated</param-name>
    <param-value>true</param-value>
  </jsp-param>
  <jsp-param>
    <param-name>debug</param-name>
    <param-value>true</param-value>
  </jsp-param>
```

```
<jsp-param>
  <param-name>precompile</param-name>
  <param-value>true</param-value>
</jsp-param>
<jsp-param>
  <param-name>compileFlags</param-name>
  <param-value>-g</param-value>
</jsp-param>
</jsp-descriptor>
```

- 3** Rebuild project and deploy the web application (as a WAR).
- 4** Launch the Weblogic server with debug parameters. To do this, modify `startWebLogic.sh` in your domain directory and add the following parameters to the `JAVA_OPTIONS` variable:

```
-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=3999,suspend=n
```

Note that these steps can be done on a local machine provided Weblogic is set up and configured in JBuilder.

On the local machine, follow these steps:

- 1** Transfer the project to the local machine where you wish to debug the JSP and open the project in JBuilder.
- 2** Under the source path for the project (normally `<PROJECT_ROOT>/src`), create a directory named `jsp_servlet`. If you need to use a source path other than the default, you can obtain this setting from the General page of the Project Properties dialog box.
- 3** Copy the JSP(s) you want to debug to the directory you just specified.
- 4** Open the JSP(s) and set breakpoints where required.
- 5** Choose Run | Configurations.
- 6** Click the New button, select the Application from the drop-down list, and click the Debug tab.
- 7** Check the Enable Remote Debugging option.
- 8** Enter the host name of the remote server in the Host Name field and set the Build Target to None. Accept all other default settings.
- 9** Attach to the remote server by clicking on the debug project icon in the toolbar.
- 10** Load your JSP in a web browser. The debugger stops at breakpoints in the JSP.

Updating projects with the latest server settings

Every time you open a project, JBuilder updates it with the latest server settings for the project's selected server(s). So, for example, if you have modified server settings for one project and you have others that use the same server configuration, the next time you open those other projects, your modified server settings will be in force automatically. If you want to use a similar but unique server configuration for a particular project, create a duplicate configuration using the Copy button, and use the Copy Servers wizard to edit it to your needs.

Be aware that this automatic updating of projects with the latest server settings can occur only with server configurations modified in JBuilder 9. If you attempt to transfer server libraries from a previous JBuilder version, your projects won't be updated with them. You can still tell JBuilder to update the project manually:

- 1 Right-click the project node of the project you want to update in the project pane and choose Properties.
- 2 Select the Servers page.
- 3 Click the ... button next to your selected single server or the separate service servers and make your changes.
- 4 Click OK to close the dialog box.

Using JBuilder with IBM WebSphere servers

This chapter explains how set up and use IBM WebSphere servers with JBuilder. JBuilder supports WebSphere Application Server 4.0 Single Server, WebSphere Application Server 4.0 Advanced Edition, and WebSphere Application Server 5.0.

Configuring JBuilder for WebSphere servers

To configure JBuilder settings to target WebSphere servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears.

- 2 Select the WebSphere server you want to configure by clicking it in the pane on the left.

The right side of the dialog box lists the default settings for the selected server. The General page has fields that all servers have in common, while the Custom page has fields that are specific to the selected server. In some cases, modifying a Custom setting will update a setting on the General page.

- 3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Servers page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed. Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings.

- 4 If you want to change any of the settings, click the ... button next to the field and make your changes. If you installed WebSphere into [drive]:/Program Files/WebSphere, the default Home Directory is [drive]:/Program Files/WebSphere/AppServer. Otherwise, the Home Directory becomes whichever directory you specify.
- 5 Click the Custom tab to view fields unique to the server. Change or fill in these fields:

For WebSphere Application Server 4.0

- IBM JDK Installation Directory: Where the IBM installation of the JDK is installed. JBuilder uses this field to locate the proper version of the JDK the server requires. This field is set when you specify the Home Directory, so it should be the java directory of the Home Directory; for example, the directory would be [drive]:/Program Files/WebSphere/AppServer/java if [drive]:/Program Files/WebSphere/AppServer/java is your Home Directory.
- DB2 Installation Directory: The directory in which DB2 is installed on your system. Usually this is in [drive]:/SQLLIB.
- Add An Administrator's Console Item to The Tools Menu: Checking this check box adds the WebSphere Admin Console item to JBuilder's Tools menu so you have quick access to it from the JBuilder IDE.
- Add an Application Assembly Tool Item to The Tools Menu: Checking this check box adds the WebSphere Application Assembly item to JBuilder's Tools menu so you have quick access to this tool from the JBuilder IDE.

For WebSphere Application 5.0

- JDK Installation Directory: Where the IBM installation of the JDK is installed. This field is set when you specify the Home Directory, so it should be the java directory of the Home Directory; for example, [drive]:/Program Files/WebSphere/AppServer/java.
- Embedded Message Path: The directory where WebSphere's messaging software is located. This field is set by default and is usually in the Ibm/WebSphere MQ directory. So, if you installed WebSphere into [drive]:/Program Files/WebSphere, the Embedded

Messaging Path would be [drive]:/Program Files/Ibm/WebSphere MQ. You can, however, specify any path directly that contains the messaging software.

- **Cell Name:** The cell name is set by default. It is read in from your local server configuration. You can edit this field, but you will seldom need to do so. For more information about cells, see your WebSphere documentation.
 - **Node Name:** The node name is set by default. It is read in from your local server configuration. You can edit this field, but you will seldom need to do so. For more information about nodes, see your WebSphere documentation.
 - **Server Name:** The server name which you are configuring. It is read in from your local server configuration. JBuilder suggests a default name, which you can change to meet your needs.
 - **Add An Administrative Console Item To The Tools Menu:** Checking this check box adds the WebSphere Admin Console item to JBuilder's Tools menu so you have quick access to it from the JBuilder IDE. You must also fill in the next field, Web Browser Path, to have the menu item added.
 - **Web Browser Path:** Specify the path and file name of the web browser of your choice. For example, you might specify `c:/Program Files/Internet Explorer/iexplore.exe` if you want to use Microsoft's Internet Explorer.
 - **Add An Application Assembly Item To The Tools Menu:** Checking this check box adds the WebSphere Application Assembly item to JBuilder's Tools menu so you have quick access to this tool from the JBuilder IDE.
 - **Add An Administrative Scripting Item To The Tools Menu:** Checking this check box adds the WebSphere Administrative Scripting item to JBuilder's Tools menu so you have quick access to this tool from the JBuilder IDE.
 - This step is optional. Click the CMP Mapping button to display the CMP Mapping page. Use this page to view and possibly change the container-managed persistence (CMP) mappings JBuilder uses. By examining the tables on this page, you can see the mappings JBuilder is using. Usually you will not need to change any of these mappings, especially if you are using the more common database drivers. You can, however, edit these mappings if you are using other drivers, want to do your own type mappings, and so on. For more information about editing mappings and aliases, click the Help button.
- 6** Choose OK to close the dialog box and configure JBuilder.

When you configure JBuilder to target a WebSphere server, required libraries are created for you. To see a list of the WebSphere libraries, see [“The created libraries” on page 4-6](#).

Selecting a server

If you are working with more than one version of WebSphere Application Server and have configured JBuilder for all versions, you must select which version you want to use for your current project. Choose Project | Project Properties and choose Server. See [“Selecting a server” on page 4-8](#) for more information.

Working in JBuilder with WebSphere 4.0

Generating WebSphere deployment descriptors

When you are using the Enterprise JavaBean 1.x wizard or the EJB 1.x Entity Modeler to create entity beans that target WebSphere 4.0 Advanced Edition, the wizards do not generate the `Map.mapxmi` and `Schema.dbxmi` deployment descriptors. When you build your bean, `EjbDeploy`, which is called during the build process, will generate them for you. You can then modify the mapping and run Make again to keep the mapping in the JAR.

If you’ve used the Entity Modeler to create your entity beans and the field names in your bean don’t map directly to columns in the database, you can choose to have JBuilder create WebSphere CMP deployment descriptors that override the default `EjbDeploy` behavior:

- 1 Right-click the EJB module node in the project pane when WebSphere 4.0 Advanced Edition is your selected server.
- 2 Choose Properties on the context menu.
- 3 Click the WebSphere AE 4.0 tab.
- 4 Check the Generate CMP Descriptors check box.
- 5 Make your changes.

The list of Data Mapping Files contains the default mapping files that are used to map Java primitive types to database types. You can modify, add, and remove mapping files. The Database Type Substitution In Generated Descriptors list displays the substitutions JBuilder makes if the Java primitive types can’t be directly mapped to a type in the database. You can modify this list. For example, you might want another Java type to be substituted for a particular original type in a specified database. Or you can add additional substitutions to the list.

- 6 Click OK.

When you compile your bean, the two deployment descriptors (`Map.mapxml` and `Schema.dbxml`) are generated for entity beans with container-managed persistence for the WebSphere 4.0 Advanced Edition only. If you are using one edition of WebSphere 4.0 and change your target server to the other version, you must recompile your project to ensure JBuilder generates the correct deployment descriptors for you.

If the database schema name used is different from the username, manually modify the `Schema.dbxml` file as follows:

- 1 Make the project to generate the CMP descriptors.
- 2 Double-click the EJB module node in the project pane to open the EJB module viewer.
- 3 Click the source view for `Schema.dbxml`.
- 4 Add the following line above the last line of the source:

```
<RDBSchema:RDBSchema xmi:id="RDBSchema_1" name="<your_schema_name>"
    database="<database_id>" tables="<table_id>"/>
```

- 5 Rebuild the project to incorporate the change.

Importing WebSphere 4.0 Advanced Edition proprietary deployment descriptors into JBuilder

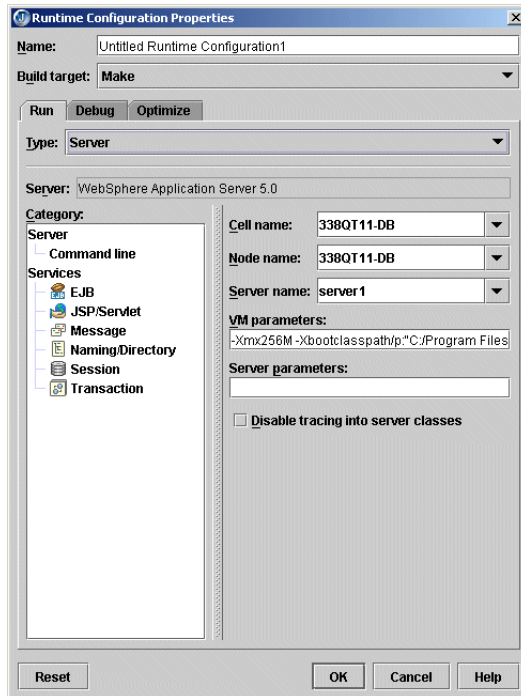
To import existing IBM proprietary descriptors into JBuilder,

- 1 Create a new project.
- 2 Choose Project | Project Properties. On the Server page set the target application server as WebSphere 4.0 Advanced Edition.
- 3 If you have not created an EJB module in JBuilder for your EJBs, create an EJB module in JBuilder.
- 4 Right click the EJB module and choose Properties.
- 5 Use the EJB page to add the proprietary descriptors to the module.
- 6 Right click the project file and choose Add Files/Packages. Import the bean source into the project.
- 7 Double-click the EJB module node in the project pane to open the EJB module viewer.
- 8 In JBuilder, the proprietary descriptors are regenerated based on the timestamp of the standard descriptor (`ejb-jar.xml`). Click the DD Source tab the `ejb-jar.xml` file to change the timestamp.
- 9 Rebuild the project.

Starting the server

To prepare to start the server from within JBuilder, create a server runtime configuration. Follow these steps:

- 1 Choose Run | Configurations.
- 2 In the dialog box that appears, click the New button.
- 3 Select Server from the Type drop-down list.



- 4 In the Name field, enter a name, such as Server or something that will identify the run configuration.
- 5 Fill in the VM Parameters and Server Parameters needed to run the server. If you've selected a target application server as described in ["Selecting a server" on page 7-4](#), default values are already in place.
- 6 Click OK.



To start the server, click the down-arrow next to the Run Project icon and select the server run configuration you just created.

By default in WebSphere 5.0, the stdout and stderr streams are redirected to log files at application server startup. To view the output in JBuilder when you start the server in JBuilder:

- 1 Start the server.
- 2 Start the Administrative Console from the JBuilder Tools menu.
- 3 Choose Servers | Application Servers | <server_name> | Process Definition | ProcessLogs.
- 4 On the Configuration page, set the Stdout File Name and the Stderr File Name to Console.

Deploying

Using the Deployment Descriptor editor

The Deployment Descriptor editor has a WebSphere 5.0 Properties panel for WebSphere 5.0 and a WebSphere 4.0 Properties panel for WebSphere 4.0. See “Server-specific Properties panel” in *Developing Enterprise JavaBeans* for information about using the panel.

On the WebSphere 4.0 and 5.0 Properties panels, the finders in the bean appear at the bottom of the list of properties. You can choose the type of query you want to use for the finder: SQL SELECT, SQL WHERE, EJB Query Language, or SQL Method. Select the finder type you want. The default value is a WHERE clause (SQL WHERE). The finder type you select is added to the WebSphere-specific deployment descriptor `ibm-ejb-jar-ext.xml`. You can then go to the Finders panel and specify the query using the query type you selected on the WebSphere 4.0 Properties panel as the value of the Where Clause on the Finders panel, even if the query type is other than a WHERE clause.

To prepare to deploy a deployable module, follow the steps outlined for your version of WebSphere:

WebSphere Application Server 5.0

Remote deploying

WebSphere Application Server 5.0 supports remote deployment using the Deployment Manager version of the application server. The Deployment Manager must be configured to manage the remote server instance. Please refer to your server documentation for more information.

Once you have configured your Deployment Manager to manage the remote server instance, you can deploy to the remote server instance in JBuilder if you follow these steps:

- 1 Right-click the archive node you want to deploy in JBuilder's project pane and choose Properties.
- 2 Click the Deployment tab, the WebSphere 5.0 tab, and finally the General tab.
- 3 Set the Cell Name to the Deployment Manager's cell name.
- 4 Set the Node Name to the remote server node name.
- 5 Set Server Name to the remote server name.
- 6 Click the Connection tab.
- 7 Set the Connection Type field to SOAP or RMI.
- 8 Set the Host field to the host name where the Deployment Manager is running.
- 9 Set the Port field to the port number of the Deployment Manager.
- 10 Set the User Name and Password fields if they are required; for example, if security is enabled.

Deploying locally

To deploy locally to a server which is not running,

- 1 Right-click the node you want to deploy in JBuilder's project pane and choose Properties.
- 2 Click the Deployment tab, the WebSphere 5.0 tab, and finally the Connection tab.
- 3 Set the Connection Type to NONE.

If the connection type is set to NONE, deployment actions performed while the server is running won't start or stop the deployed module.

WebSphere Server 4.0

If you are using WebSphere 4.0 Advanced Edition, you must create an EAR group that contains your beans. You then deploy the EAR group. For more information about creating EAR groups, see "Creating an EAR file" in *Developing Enterprise JavaBeans*.

Tools | Enterprise Deployment displays a WebSphere Deploy Settings dialog box. Configure your deployment settings with this dialog box. JBuilder passes the settings you specify on to the WebSphere deployment tool. The deployment tool for WebSphere 4.0 differs depending on the version of the server you are using. For the Single Server, WebSphere's

deployment tool is SEAppInstaller. For the Advanced Edition, the deployment tool is XmlConfig. Therefore, the appearance of the Deploy Settings dialog box will vary between these two WebSphere Server 4.0 editions.

If your target application server is WebSphere 4.0 Advanced Edition and you want an XML file to be generated as input to the WebSphere XMLConfig utility, check the Generate XML check box. If this option isn't checked, the file won't be created. If you make your own modifications to the generated XML file (named `deploy_<selectednode>.xml` and appearing under the EJB module node or EAR group), uncheck this option to be sure you don't lose your changes. If you use the Deployment Options on the context menu (right-click the EJB module and choose Deployment Options <jar name>.jar to see the deployment commands), the generated XML file is `deploy.xml`. It appears under the project node.

Remote deploying

Follow these steps to use the WebSphere Deploy Settings dialog box to deploy to a remote server:

- 1 Choose Tools | Enterprise Deployment.
- 2 Select the Deploy action.
- 3 Specify these options in the Options field:
`-nameServiceHost<host name> -nameServicePort<port number>`
- 4 Set the Primary Node and Server Name to match your server configuration.
- 5 Click OK.

You can also deploy a module to a remote server from JBuilder's project pane:

- 1 Right-click the module you wish to deploy.
- 2 Select Properties.
- 3 Click the Deployment tab.
- 4 Specify these options in the Options field:
`-nameServiceHost<host name> -nameServicePort<port number>`
- 5 Set the Primary Node and Server Name to match your server configuration.
- 6 Right-click the module in the project pane and choose Deploy Options | Deploy.

Deploying locally

These are the deploy options you'll find on the Deploy Options context menu:

- **Deploy** - Deploys a JAR to the currently running container of the project application server. If the Build Target option is Make for the current runtime configuration (Choose Run | Configurations, select the current runtime configuration, and click Edit then click Server and select Deployment in the tree of services to see the Build Target), this option will "make" the JAR's contents before deploying it to the container.
- **Redeploy** - Deploys a JAR again to the currently running container. If the Build Target option is Make for the current runtime configuration (Choose Run | Configurations, select the current runtime configuration, and click Edit then click Server and select Deployment in the tree of services to see the Build Target), this option will "make" the JAR's contents before redeploying it to the container.
- **Undeploy** - Undeploys an already deployed JAR in the running container.
- **List Deployments** - Lists all JARs deployed in the running container.
- **Stop Container** - Stops the container. This option appears for WebSphere 4.0 Advanced Edition only. When a deployed EJB changes, the container must be stopped and then restarted for the changes to register.
- **Start Container** - Starts the container. This option appears for WebSphere 4.0 Advanced Edition only.

Enabling remote debugging

The steps to enable remote debugging of a bean running on a WebSphere server vary depending on which version of WebSphere you are using.

Note You won't be able to debug JSPs in a remote debug session.

WebSphere Application Server 5.0

Follow these steps to enable remote debugging for WebSphere Application Server 5.0:

- 1 **Launch the server with the `-script` option:**

```
WEBSPHERE_HOME/bin/startserver -script
```

This command should write a script called `start_<server name>` in the `WEBSPHERE_HOME/bin` directory.

- 2 Edit the launch script and add the following remote debug parameters to the java command line:

```
-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=3999,suspend=n
```

- 3 Launch the server using the script.
- 4 In JBuilder, choose Project | Project Properties. Click the Run tab. If you have not yet created a server run configuration, choose New and create a new configuration of type Server. If you already have a server run configuration, choose Edit.
- 5 Click the Debug tab and check the Enable Remote Debugging and Attach options and click OK.



Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited. You will now be able to set breakpoints in Java code, such as in EJBs, servlets, and so on.

WebSphere Single Server 4.0

If you are using WebSphere Single Server 4.0, follow these steps to enable remote debugging:

- 1 Launch the server with the `-script` option:

```
WEBSPPHERE_HOME/bin/startserver -script
```

This command should write a script called `launch` in the `WEBSPPHERE_HOME/bin` directory.

- 2 Edit the launch script and add the following remote debug parameters to the java command line:

```
-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=5000,suspend=n
```

- 3 Launch the server using the script.
- 4 In JBuilder, choose Project | Project Properties. Click the Run tab. If you have not yet created a server run configuration, choose New and create a new configuration of type Server. If you already have a server run configuration, choose Edit.
- 5 Click the Debug tab.
- 6 Check the Enable Remote Debugging and Attach options and click OK.



Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited. You will now be able to set breakpoints in an Java code, such as EJBs, servlets, and so on.

WebSphere Server Advanced Edition 4.0

WebSphere Server Advanced Edition 4.0 launches another Java virtual machine for debugging, so you must follow instructions so you can debug your enterprise beans:

- 1 Start the WebSphere Server 4.0.
- 2 Start the WebSphere Server 4.0 console application (which you'll find at `<ws4_ae_home>\bin\adminclient.bat` if you're running on Windows.)
- 3 When the console application appears, expand the tree on the left until you see the Application servers node. Select this node.
- 4 In the panel on the right side of the console, click the JVM Settings tab. Click the Advanced Settings button. (You might need to scroll the panel to see this button.)
- 5 In the dialog box that appears, enter the following VM options:

```
-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=5000,suspend=n
```

- 6 If you start the server outside of JBuilder, choose Run | Configurations and click the New button in the dialog box that appears. In the Configuration Name field, specify a name for the new run configuration server for debugging on your application server.
- 7 Click the Debug tab and make these changes on the Debug page:
 - a Check the Enable Remote Debugging check box.
 - b Select the Attach option.
 - c In the Port Number field, enter the port number you noted during the server startup process.
 - d Click OK twice to close all dialog boxes.



- 8 Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited.

You will be attaching to the application server process. Now you can deploy your or web applications and you'll be able to stop at breakpoints you set in the bean, JSPs, or servlets.

To start the server in debug mode within JBuilder, follow these steps:

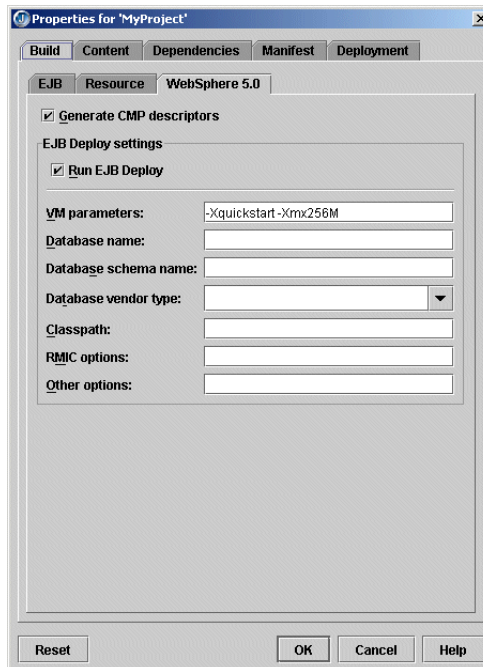
- Choose Run | Configurations.
- Select the Command node and enter the remote debug port number in the Debug Transport Address field.

Now clicking the debug toolbar button launches the server in debug mode and attaches to it, all in one session.

Container-managed persistence (CMP) 1.1 and 2.0 in WebSphere 5.0

You can set the schema name and database vendor type as properties on EJB module nodes:

- 1 Right-click the EJB module and choose Properties.
- 2 Click the Build tab, then the WebSphere 5.0 tab.
- 3 Enter your schema and database vendor information in this dialog box:

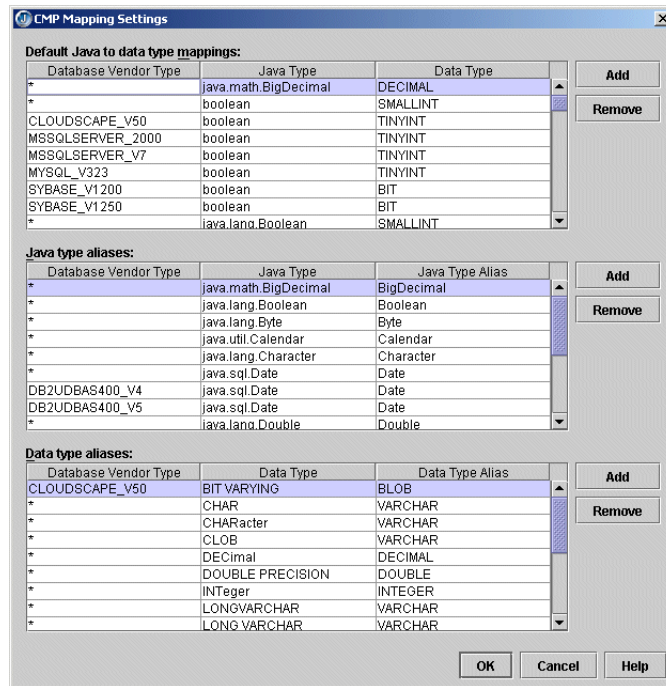


- 4 Click OK.

To change the default CMP mapping behavior such as the Java to database type mappings,

- 1 Choose Tools | Configure Servers.
- 2 Select WebSphere Application Server 5.0 in the list of application servers.
- 3 Click the Custom button.
- 4 Click the CMP Mapping button.

5 Make your changes in the CMP Mapping Settings dialog box that appears:



6 Click OK.

These are your options:

- **Default Java To Data Type Mappings:** This list is used if JBuilder is unable to resolve the database schema type for a CMP field. This could happen if the JDBC driver fails to report a type for a database column or if the corresponding column (as defined in the CMP field mapping) is missing.
- **Java Type Aliases:** This list is used to translate between actual Java types in the bean and types defined in IBM database type mapping files.
- **Data Type Aliases:** This list is used to translate between database vendor types (as reported by the JDBC driver) and types defined IBM database type mapping files.

Using JBuilder with Sybase servers

This chapter explains how to use to set up and use Sybase servers with JBuilder. JBuilder supports Sybase Enterprise Application Server 4.1.3 and Sybase Enterprise Application Server 4.2.

Configuring JBuilder for Sybase servers

To configure JBuilder settings to target Sybase servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears.

- 2 Select the Sybase server you want to configure by clicking it in the pane on the left.

The right side of the dialog box lists the default settings for the selected server. The General page has fields that all servers have in common, while the Custom page has fields that are specific to the selected server. In some cases, modifying a Custom setting will update a setting on the General page.

- 3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Servers page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed. Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings.

- 4 If you want to change any of the settings, click the ... button next to the field and make your changes. Assuming you installed Sybase into [drive]:/Program Files/Sybase, your Home Directory should be [drive]:/Program Files/Sybase/EAServer. When you specify the correct home directory, the Native Executable Launches field is filled in automatically with the location of the batch file, serverstart.bat. It is not necessary to specify VM Parameters, Server Parameters, or a Working Directory.
- 5 Click the Custom tab to view fields unique to the server. Change or fill in these fields:
 - **JDK Installation Directory:** This field is set for you automatically when you specify the Home Directory on the General page. JBuilder uses this field to locate the proper version of the JDK the server requires. The directory is usually the _jvm directory of the Home Directory; for example, it would be [drive]:/Program Files/Sybase/EAServer/_jvm if [drive]:/Program Files/Sybase/EAServer is your Home Directory.
 - **Administrator Login:** The name you use to authenticate yourself to the server.
 - **Administrator Password:** The password used to authenticate yourself to the server.
 - **Server address:** The location of the server. By default, this is your machine. You can change this.
 - **Port:** The port number used by the server. JBuilder assigns a default port number.
 - **Add Jaguar Manager Item To The Tools Menu:** Adds the EAServer Jaguar Manager to the JBuilder Tools item so that you may access it from within JBuilder.
- 6 Choose OK to close the dialog box and configure JBuilder.

When you configure JBuilder to target a Sybase server, required libraries are created for you. To see a list of the Sybase libraries, see [“The created libraries” on page 4-6](#).

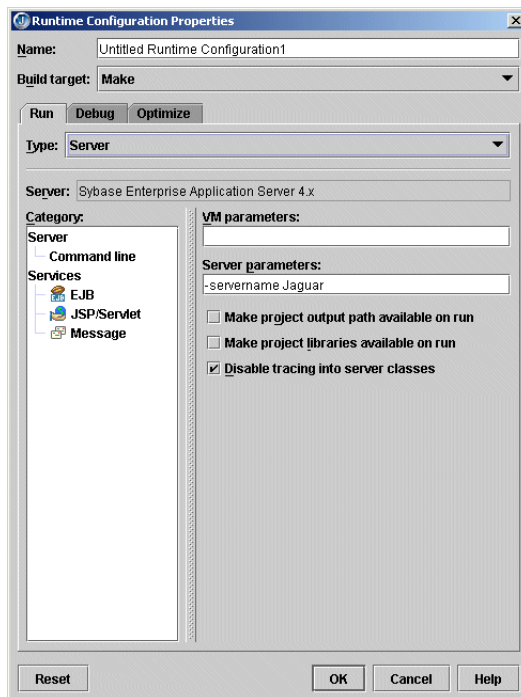
Selecting a server

If you are working with more than one version of Sybase Enterprise Application Server and have configured JBuilder for all versions, you must select which version you want to use for your current project. Choose Project | Project Properties and choose Server. See [“Selecting a server” on page 4-8](#) for more information.

Starting the server

To prepare to start the server from within JBuilder, create a server runtime configuration. Follow these steps:

- 1 Choose Run | Configurations.
- 2 In the dialog box that appears, click the New button.
- 3 Select Server from the Type drop-down list.



- 4 Fill in the VM Parameters and Server Parameters needed to run the server. If you've selected a target application server as described in [“Selecting a server” on page 8-3](#), default values are already in place.
- 5 Click OK.



To start the server, click the down-arrow next to the Run Project icon and select the server run configuration you just created.

Remote deploying

To prepare to deploy a deployable module to a remote server, choose one of these methods:

- Using the Tools | Configure Servers dialog box:
 - a** Choose Tools | Configure Servers.
 - b** Select the Sybase server from the list of servers.
 - c** Click the Custom tab.
 - d** Set the Server Address, Port, User Name, and Password to match your remote server configuration and click OK.
 - e** Right-click the module you want to deploy and choose Deploy Options | Deploy.
- Setting properties of the deployable module in the project pane:
 - a** Right-click the module you want to deploy and choose Properties.
 - b** Click the Deployment tab to select the Deployment page.
 - c** Set the Server Address, Port, User Name, and Password to match your remote server configuration.
 - d** Check the Override Global Deployment Settings option and click OK.
 - e** Right-click the module you want to deploy and choose Deploy Options | Deploy.

Deploying web applications

There are two issues you should keep in mind when deploying web applications to the Sybase server:

- Because of a limitation in the Sybase command-line tool, jagant, when you deploy a web application, you must use the Sybase Console Manager to refresh the deployed web application. If you have checked the Add A Jaguar Manager Item To The Tools Menu option on the Custom page of the Configure Servers dialog box when Sybase is your selected server, you can start the Sybase Console Manager from within JBuilder with Tools | EAServer Jaguar Manager.
- If you want to redeploy a web application that has been removed from a server but left in the Sybase repository, don't use the Deploy

command on the context menu even though the application is no longer deployed to a server instance. Instead use the Redeploy command.

Remote debugging

To prepare to debug remotely, choose one of these methods:

- 1 Start the Sybase server in debug mode at the command line:

```
<Sybase_home>\bin\serverstart -debug
```

- 2 Note the debug port number that the server starts with.

You can also use this method to start the server in debug mode:

- 1 Open the properties file `<Sybase_home>\Repository\Server\Jaguar.props`.
- 2 Search for the property `com.sybase.jaguar.server.jvm.options`. If it doesn't exist, set the property to this:

```
-Xdebug, -Xnoagent, -Djava.compiler=NONE,  
'-Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=3999'
```

If the property already exists, add on to it so it appears with the parameters show above.

- 3 Start the server at the command line:

```
<Sybase_home>\bin\serverstart
```

- 4 Note the port number the server uses in this case 3999.

After you have the server started, continue with these steps within JBuilder:

- 1 In the project from which you want to launch the remote debug session, choose Run | Configurations. If you have not yet created a server run configuration, choose New to create a new configuration and select the Type as Server. If you have already created a server run configuration, click the Edit button instead.
- 2 Click the Debug tab.
- 3 Check the Enable Remote Debugging check box and select the Attach option.
- 4 In the Port Number field, enter the port number you noted during the server startup process.
- 5 Click OK twice to close all dialog boxes.



- 6 Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited.

New features for Sybase EAServer in JBuilder

JBuilder 9 has additional support for Sybase EAServer:

- CMP descriptors are now generated in JBuilder (including relationships for EJB 2.0 components).
- JNDI resources for EJB references, both local and remote, are now generated in JBuilder.

Using JBuilder with Sun iPlanet servers

This chapter explains how to use to set up and use iPlanet servers with JBuilder. JBuilder supports iPlanet Application Server 6.0 SP 4 and 6.5 SP 1.

Configuring JBuilder for iPlanet servers

To configure JBuilder settings to target iPlanet servers,

- 1 Choose Tools | Configure Servers.

The Configure Servers dialog box appears.

- 2 Select iPlanet Application Server 6.x+ by clicking it in the pane on the left of the dialog box.

The right side of the dialog box lists the default settings for the selected server. The General page has fields that all servers have in common, while the Custom page has fields that are specific to the selected server. In some cases, modifying a Custom setting will update a setting on the General page.

- 3 Check the Enable Server check box at the top of the dialog box.

Checking this option enables the fields for the selected application server. You won't be able to edit any fields until it is checked. The Enable Server check box also determines whether this server will appear in the list of servers when you select a server for your project using the Servers page of the Project | Project Properties dialog box.

JBuilder provides default settings for the selected server. If JBuilder's default settings are not appropriate, edit any of the settings as needed. Some servers require you to enter settings in addition to the defaults. Clicking the OK button in this dialog box when not all required values are set or selecting another server while the current one is enabled displays a message dialog box that informs you about the missing settings.

- 4 If you want to change any of the settings, click the ... button next to the field and make your changes. Assuming you installed iPlanet into `[drive]:iPlanet`, by default the Home Directory becomes `[drive]:/iPlanet/ias6`. You, can, however, specify any directory that contains the iPlanet software. When you specify the correct home directory, the Main Class field is automatically filled in for you. It is not necessary to specify VM Parameters, Server Parameters, or a Working Directory.
- 5 Click the Custom tab to view fields unique to the server. Change or fill in these fields:
 - Add iPlanet Tools To The Tools Menu: Places iPlanet tools at the bottom of the JBuilder Tools menu so you have access to them from within JBuilder.
 - JDK Installation Directory: This field is set for you automatically when you specify the Home Directory on the General page. JBuilder uses this field to locate the proper version of the JDK the server requires.
- 6 Choose OK to close the dialog box and configure JBuilder.

When you configure JBuilder to target a iPlanet server, required libraries are created for you. To see a list of the iPlanet libraries, see [“The created libraries” on page 4-6](#).

Selecting a server

If you are working with more than one version of iPlanet Application Server and have configured JBuilder for all versions, you must select which version you want to use for your current project. Choose Project | Project Properties and choose Server. See [“Selecting a server” on page 4-8](#) for more information.

Service packs

When using iPlanet 6.0, use Service Pack 4. When using iPlanet 6.5, use service pack 1, then install the Maintenance Update 1, and finally, install the Buffer Overflow patch for the iPlanet web server. JBuilder has been tested using these service packs only.

Running or debugging on iPlanet

Before you begin running or debugging an application, start the iPlanet server outside of JBuilder.

Starting and stopping iPlanet from within JBuilder

When you choose to run or debug a server component when iPlanet is the target application server, the iPlanet startup process is more complicated. You will see additional message boxes appear. JBuilder must stop the iPlanet KJS processes running outside of JBuilder, deploy the selected archive, and start a single KJS process within JBuilder. At each step, you'll see a message box. As with other servers, you can view the progress of the start-up process in the message window. You will not be able to interact with JBuilder until you see the ENGINE-ready message. Once you do see the ENGINE-ready message, you should probably scroll back in the message pane to verify that no errors have occurred.

To stop the iPlanet server, click the red stop button at the bottom right of the message pane. The normal KJS processes are then restarted outside of JBuilder.

If an iPlanet error occurs, you must use iPlanet's administration tool to shutdown and restart the server. You can access it from JBuilder's Tools menu if you have checked the Add iPlanet Tools To The Tools Menu option when you set up iPlanet for JBuilder using Tools | Configure Servers.

When you stop the iPlanet server, the running JAR is undeployed. To deploy the JAR again, restart the server and deploy the JAR.

Running or debugging a servlet or JavaServer Page

If you are running or debugging a servlet or JavaServer Page (JSP) and the content you expect is not appearing in the View pane but instead errors are presented, try these steps:

- 1 Stop the web server before starting iPlanet within JBuilder.
- 2 When the ENGINE-ready message appears in the message pane, restart the web server.

Remote deploying

To prepare to deploy a deployable module to a remote server, choose one of these methods:

- Using the Tools | Enterprise Deployment dialog box:
 - a Choose Tools | Enterprise Deployment.
 - b Select PServer and set the Host Name, Port Number, User Name and Password to match your remote server configuration and click OK.
 - c Right-click the module you want to deploy and choose Deploy Options | Deploy.
- Setting properties of the deployable module in the project pane:
 - a Right-click the module you want to deploy and choose Properties.
 - b Click the Deployment tab to select the Deployment page.
 - c Select PServer and set the Host Name, Port Number, User Name and Password to match your remote server configuration and click OK.
 - d Right-click the module you want to deploy and choose Deploy Options | Deploy.

Remote debugging

Refer to your server documentation to start up the iPlanet server in debug mode outside of JBuilder. Note the port number the server is using.

After you have the server started, continue with these steps within JBuilder:

- 1 In the project from which you want to launch the remote debug session, choose Run | Configurations. If you have not yet created a server run configuration, choose New to create a new configuration and select the Type as Server. If you have already created a server run configuration, click the Edit button instead.
- 2 Click the Debug tab.
- 3 Check the Enable Remote Debugging check box and select the Attach option.
- 4 In the Port Number field, enter the port number you noted during the server startup process.
- 5 Click OK twice to close all dialog boxes.



- 6 Click the down arrow next to the Debug Project icon on the JBuilder toolbar and select the debug run configuration you just created or edited.

Deployment settings

In the iPlanet 6.x+ Deploy Settings dialog box (Tools | Enterprise Deployment), the List Deployments action works only when the Local Server option is the selected Target option even though the List Deployments action is still selectable in the Action drop-down list.

Creating clients for iPlanet

To begin creating a client with JBuilder,

- 1 On the iPlanet 6.x+ Properties page of the Deployment Descriptor editor, set the `iiop` property of your enterprise bean to true.
- 2 Display the enterprise bean source code in the editor and use the Test Client wizard on the object gallery to generate the client source file.
- 3 Within the client source code modify the “lookup” code so that it uses the naming convention iPlanet uses:

```
Object ref = initial.lookup("<use iPlanet convention>/<Bean Name>")
```

Consult your iPlanet documentation to find iPlanet’s naming convention.

Now you can modify and expand the source code of your client so that it implements the logic your client needs. When you are done, create run configurations for the server and the client as explained in “Testing your enterprise bean” in *Developing Enterprise JavaBeans*.

Chapter 10

Using JBuilder with your web server

Web Development is a feature of JBuilder Developer and JBuilder Enterprise

Support for Tomcat, Borland Enterprise Server, Sun iPlanet, Sybase Enterprise Application Server and IBM WebSphere is not provided with the JBuilder WebLogic Edition

This chapter explains how to work with your web server and JBuilder together.

Both Java servlets and JavaServer Pages (JSP) run inside web servers. Tomcat, the JavaServer Pages/Java Servlets reference implementation, is included with JBuilder. Although it might differ from your production web server, Tomcat allows you to develop and test your servlets and JSPs within the JBuilder development environment.

JBuilder Enterprise supports the Borland Enterprise Server, Sun iPlanet, IBM WebSphere and BEA WebLogic application servers. These application servers contain web servers. However, in order to use the contained web server, you must have the corresponding application server installed and configured. You cannot use the web server outside of its application server container. Once you've configured your web application and web server, you can compile, run and debug your servlet and JSP. For more information, see the chapter called "Working with web applications in JBuilder" in the *Web Application Developer's Guide*.

Note In JBuilder Enterprise, you can use the Tomcat web server as a stand-alone web server.

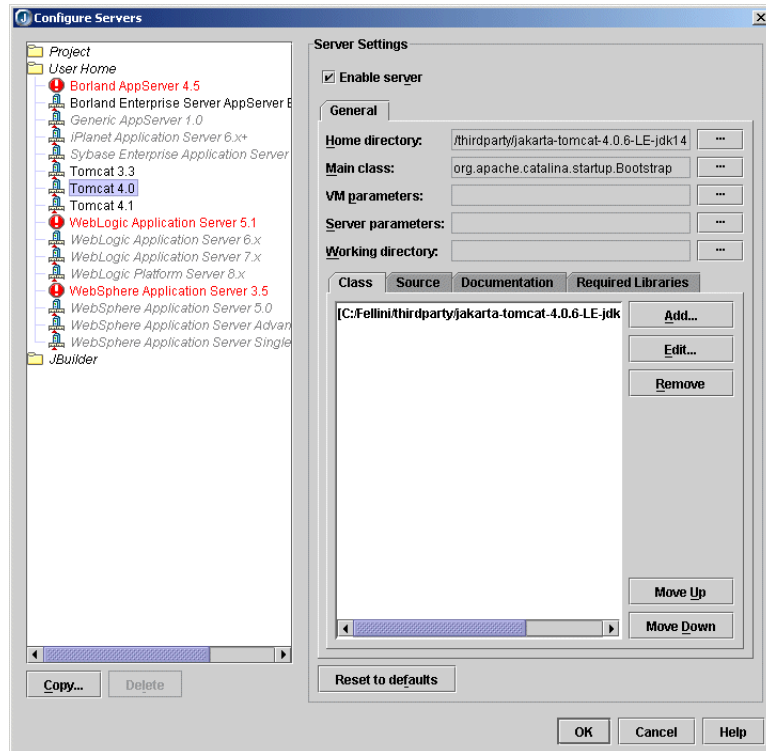
Viewing Tomcat configurations

When you install JBuilder Enterprise, three versions of Tomcat, 3.3, 4.0 and 4.1, are automatically installed in your JBuilder directory. By default, they are automatically configured. You can examine the configuration with the Configure Servers dialog box.

Tomcat 4.0 and 4.1 are the JDK 1.4 Lightweight Editions. These editions do not contain an XML parser (JDK 1.4 has one built in) and a few other publicly available components. You can download the full edition from <http://jakarta.apache.org/> then use the Configure Servers dialog box to point to them.

Note In JBuilder Enterprise, Tomcat 4.0 is the default server. To change the default for your project, see [“Selecting a server for your project” on page 10-4.](#)

- 1 Choose Tools | Configure Servers. The Configure Servers dialog box is displayed.



Note In the tree on the left side of the dialog box, entries in gray have not yet been configured. Entries in red are invalid.

- 2 Choose one of the Tomcat installations from the User Home folder. The Enable Server option at the top of the right side of the dialog box is automatically checked. You do not need to change any settings.
- 3 Click OK to close the dialog box.

If you have installed Tomcat to another directory, and want to run Tomcat from that directory instead of the default, you can change the settings to point to that directory. The following table explains the settings.

Table 10.1 Configure Server dialog box settings for Tomcat

Option	Description
Home directory	Tomcat's home directory. If you're reconfiguring one of the Tomcat versions installed with JBuilder, this will be in the <code><jbuilder>/thirdparty</code> directory.
Main class	The main class for starting the Tomcat web server.
VM parameters	The parameters to pass to the Java VM running the web server.
Server parameters	The parameters to pass to the web server.
Working directory	The working directory.
Class	The location of Tomcat class files.
Source	The location of Tomcat source files.
Documentation	The location of Tomcat documentation files.
Required Libraries	The libraries that Tomcat requires.

If you'd like more information about Tomcat or would like to run it stand-alone, refer to the documentation directory of JBuilder's Tomcat installation:

- **Tomcat 3.3:** `<jbuilder>/thirdparty/jakarta-tomcat-3.3.1/doc`
- **Tomcat 4.0:** `<jbuilder>/thirdparty/jakarta-tomcat-4.0.6-LE-jdk14/webapps/tomcat-docs`
- **Tomcat 4.1:** `<jbuilder>/thirdparty/jakarta-tomcat-4.1.12a-LE-jdk14/webapps/tomcat-docs`

Note Tomcat 4.1 does not support JSP debugging.

Configuring other web servers

JBuilder supports additional web servers other than Tomcat:

- Borland Enterprise Server
- Sun iPlanet
- Sybase Enterprise Application Server
- BEA WebLogic
- IBM WebSphere

These web servers are automatically configured when you configure their corresponding application server in the Configure Servers dialog box (Tools | Configure Servers). You cannot use these web servers outside of their application server container.

To configure the application servers that contain these web servers, see [Chapter 4, "Configuring the target server settings."](#)

Note The Borland Enterprise Server application server uses Tomcat internally as its production web server. You do not need to configure Tomcat separately.

Selecting a server for your project

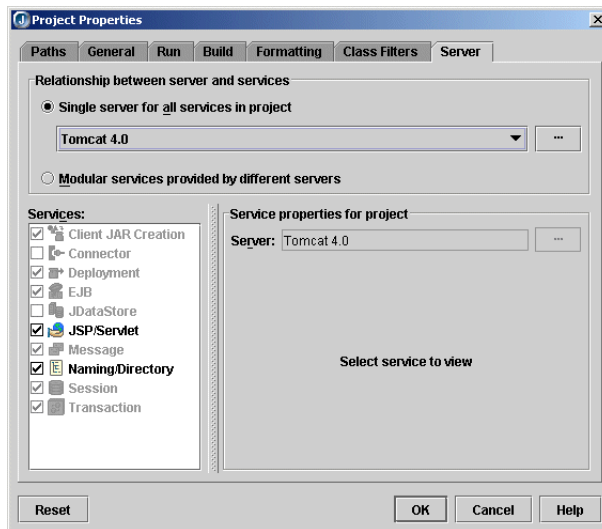
JBuilder can target multiple application servers and their contained web servers in a single project. You can choose a single application server container for all stages of EJB and web application development. You can also choose different services provided by different servers for different aspects of development. For example, you can select the EJB service provided by the Borland Enterprise Server for enterprise bean development and the JSP/Servlet service provided by Tomcat for web application development.

Tip If you don't have an application server installed, JBuilder will automatically target your project for Tomcat 4.0. However, if the web icons on the Web page of the object gallery are disabled, you may need to actively select a server for your project; follow the steps below:

Important The only service available for Tomcat 3.3 is the JSP/Servlet service. For Tomcat 4.0 and 4.1, the JSP/Servlet and Naming/Directory services are available.

To select one or more versions of an application server(s) (and their contained web servers) to use for your project,

- 1 Choose Project | Project Properties.
- 2 Click the Server tab. The Server page is displayed.



Note In order for an application server to display in the drop-down list at the top of the dialog box, it must be configured in the Configure Servers dialog box.

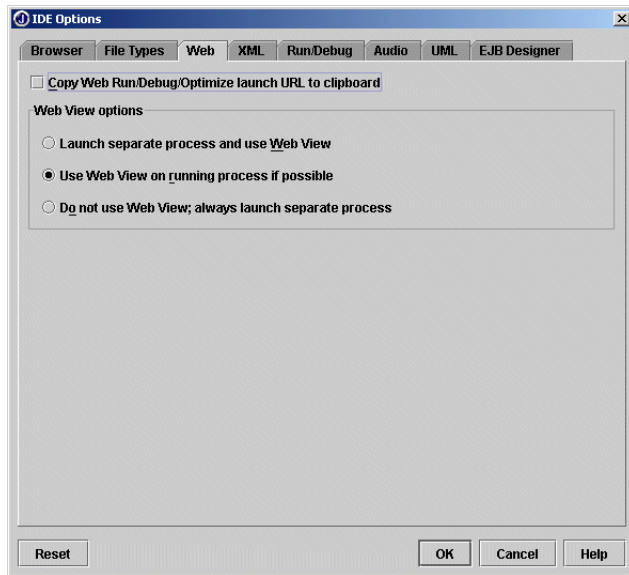
- 3 If you have one or more application servers installed and configured, decide whether you want to use a single application and web server for all aspects of development or different application servers to handle different areas of development.
 - To use a single server,
 - 1 Select the Single Server For All Services In Project option and select the server from the drop-down list. This server can either be an application server that contains a web server or the Tomcat web server that is installed with JBuilder.
 - 2 If you want to avoid having libraries added to your project that you won't use, uncheck the check box in front of the service(s) you don't need in the Services list. If you disable services, the corresponding JBuilder features will be disabled. For example, if you turn off the JSP/Servlet service, most of the Web wizards and the JSP compilation feature are disabled.
 - 3 If you want to make changes to the configuration settings for the selected server, click the ellipsis button to the right of the server name and edit the settings you want on the General and/or Custom pages. Click OK when you're finished.
 - To use different servers for different services,
 - 1 Select the Modular Services Provided By Different Servers option.
 - 2 If you want to avoid having libraries added to your project that you won't use, uncheck the check box in front of the service(s) you don't need in the Services list. If you disable services, the corresponding JBuilder features will be disabled. For example, if you turn off the JSP/Servlet service, most of the Web wizards and the JSP compilation feature are disabled.
 - 3 Update the configuration for the selected service on the right side of the dialog box. Depending on the selected server/service, this information may be able to be configured or may be read-only. For more information, see ["Selecting a server" on page 4-8](#).
 - 4 To use a selected web server, click the JSP/Servlet service. In the Server drop-down list on the right slide of the dialog box, select the web server you want to use. If you want to make changes to the configuration settings for the web server, click the ellipsis button and edit the settings you want on the General page. Click OK when you're finished.
 - 5 Click OK again to close the Servers page.

Configuring the IDE for web run/debug

Once you've set up JBuilder with a web server, you can configure options for the web server, including web view options and how the web server is launched in the IDE.

To configure how the IDE behaves when trying to Web Run or Web Debug,

- 1 Choose the Web tab on the IDE Options dialog box (Tools | IDE Options). The Web page looks like this:



- 2 Choose the Copy Web Run/Debug/Optimize Launch URL To Clipboard option to copy the URL used to launch the web application to the clipboard. This enables you to easily go to the same URL in an external browser. Set this option if you're creating a Java Web Start applet or application.
- 3 Choose Web View Options. These options work in conjunction with the Search For Unused Port option on the Runtime Configuration Properties dialog box when the specified port is in use by a non-web process. (See "Creating a runtime configuration" in the *Web Application Developer's Guide* for more information.)
 - Choose the Launch Separate Process And Use Web View option to use both the internal web browser and an external web browser. This option automatically displays your rendered servlet or JSP in the Web View page of the content pane.
 - Choose the Use Web View On Running Process If Possible option to use the internal web browser to view your web page. This option

automatically displays your rendered servlet or JSP in the Web View page of the content pane. If a web server is already running, JBuilder uses the same process on the existing port. This is the default.

- Choose the Do Not Use Web View Always Launch Separate Process option when launching your web application in an external web browser.

Web running your servlet or JSP

Before you web run your servlet or JSP in JBuilder, you should create a runtime configuration, set run properties for your project, and compile your servlet or JSP. For complete information about these topics, see “Working with web applications in JBuilder” in the *Web Application Developer’s Guide*. Once you’ve completed these tasks, you are ready to run your servlet or JSP.

To web run your servlet or JSP in JBuilder, right-click the servlet or JSP file in the project pane and choose Web Run. The Web Run command runs your configuration in the selected web server without debugging it. If your servlet runs from an HTML or SHTML file, right-click that file and choose Web Run.

If the Web Run or Web Debug command is grayed out on the context menu for a servlet or JSP, check the runtime configuration. To run a web application, you need to create a runtime configuration with the server type selected as the current runner. That server must support the JSP/Servlet service listed on the Project Properties Server page on the tree in the left side of the dialog box. To create a runtime configuration, see “Creating a runtime configuration” in the *Web Application Developer’s Guide*.

Note Applets cannot be web run or web debugged. This is because applets don’t have a URL or a web context to run in. Additionally, applets run in a client browser as opposed to a server. Typically, you run an applet in Sun’s AppletViewer or in JBuilder’s AppletTestbed. For more information, see “JBuilder’s AppletTestbed and Sun’s appletviewer” in the “Working with applets” chapter of the *Web Application Developer’s Guide*.

Starting your web server

When you choose Web Run, JBuilder starts your web server, using:

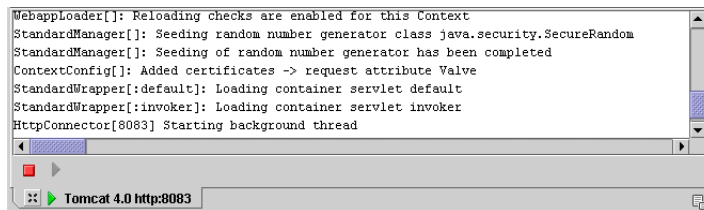
- The runtime configuration (see “Creating a runtime configuration” in the *Web Application Developer’s Guide* for more information).
- The options set on the Web page of the IDE Options dialog box (Tools | IDE Options) (see “Configuring the IDE for web run/debug” in “Configuring your web server” in the *Web Application Developer’s Guide* for more information).

Messages are logged to the web server tab displayed in the message pane at the bottom of the AppBrowser. HTTP commands and parameter values are also echoed to this pane. The name of the tab will reflect the name of the web server, for example Tomcat for the Tomcat web server.

Note When you Web Run a servlet or JSP using the iPlanet web server, you'll see two new tabs in the message pane. One is for the iPlanet preparation phase, and shows the command line used to set up the web server. The other is for output from the web server itself. Both tabs are called iPlanet.

An example of output to the message pane for the Tomcat web server is displayed below:

Figure 10.1 Tomcat messages



When the Web Run starts, two new tabs are displayed in the content pane: Web View and Web View Source. Click the tab to open the web view and the web view source.

Note Web View and Web View Source tabs will not display if you have selected the Do Not Use Web View option on the Web page of the IDE Options dialog box. For more information, see "Configuring the IDE for web run/debug" in "Configuring your web server" in the *Web Application Developer's Guide*.

Web view

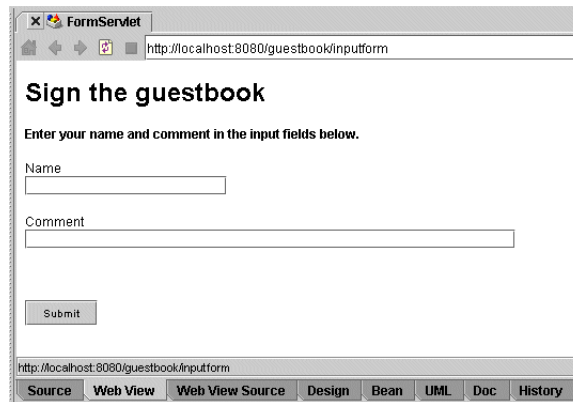
Formatted output is displayed in the web view pane of the content pane. The generated URL is displayed in the location field at the top of the web view.

Important The browser used in the web view pane is not a full-featured web browser.

The web view displays the servlet after it has been processed by the servlet engine. There may be a delay between when the servlet or JSP file is edited and when the change is shown in the web view. To see the most recent changes, select the Refresh button at the top of the web view.

Note that web view is simply a browser and is not tied to other JBuilder features. For example, you can view your formatted output in a browser external to JBuilder. Options on the Web page of the IDE Options dialog box speed up repeated use of this feature. See "Configuring the IDE for web run/debug" in "Configuring your web server" in the *Web Application Developer's Guide* for more information.

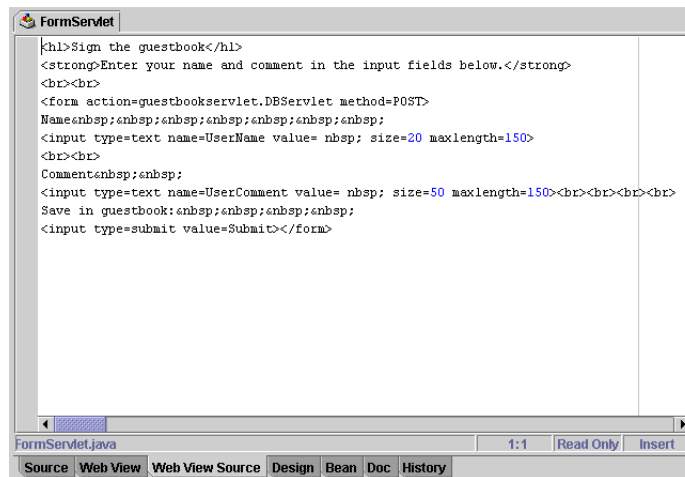
Figure 10.2 Web view output




Web view source

Raw output from the web server is displayed in the web view source pane.

Figure 10.3 Web view source



Stopping the web server

-  To stop the web server, click the Reset Program button on the web server tab. To start the web server again and re-run your web application, click the Restart Program button. You'll usually follow these steps when you make changes to source code, re-compile, and re-run. You don't need to close the web server pane each time you start the web server.

Note The first time you press the Reset Program button, it simply sends a command to the server to shutdown. In particular, if you are trying to debug your webapp's lifecycle event handlers, this would call the `Servlet.destroy()` and `ServletContextListener.contextDestroyed()` methods. If you're not debugging, the server will usually shutdown by itself (Tomcat does this in a few seconds; other server take longer). In any case, if you press the Reset Button a second time, the server process is terminated immediately.

Changing the web server's port number

Occasionally, you may run into a problem where the default port number assigned to your web server (typically 8080) is in use by another application. You can reassign the port number, or simply instruct JBuilder to search for an unused port. (In a typical installation, this is set as the default.) To do this,

- 1 Choose Run | Configurations and select the runtime configuration for your WebApp. Choose Edit.
- 2 On the Runtime Configuration Properties dialog box, choose the JSP/Servlet Service in the tree on the lower left side of the dialog box.
- 3 In the Port Number field, enter the port number the web server should listen to.
- 4 Choose the Search For Unused Port option to tell JBuilder to choose another port if the specified one is in use. (The port is only searched for the first time a web run is requested.) It is useful to select this option when you are running more than one servlet or JSP, as otherwise you might get a message that the port is busy. It is also useful to check this option in the event that a user problem brings the web server down. With this option selected, you will be protected if the web server is not shut down properly. This option works in conjunction with the Launch options on the Web page of the IDE Options dialog box when the specified port is in use by a non-web process.

Creating a custom server.xml file with Tomcat

Typically, when you run your web application using Tomcat in the JBuilder IDE, JBuilder creates a `serverXXXX.xml` file (where XXXX is the port number Tomcat is listening to) and deletes it when you shut down Tomcat. You can edit the file and force JBuilder to keep it by following these steps:

- 1 Run your web application in JBuilder as you normally would.

- 2 While Tomcat is running, go to your project's `Tomcat\conf` directory. This folder is created by JBuilder when your web application is running.
- 3 Make a copy of the `conf` directory in a `temp` directory.
- 4 Open `serverXXXX.xml` in the `conf` directory in a text editor.
- 5 Remove the second line of the file:

```
<!--This comment marks this file as generated, so it may be deleted and
regenerated at any time. To preserve manual changes to this file, delete
this comment.-->
```

With this line removed, JBuilder will not automatically delete the file when you stop the server.

- 6 Make other changes to the server configuration file as needed.
- 7 Go back to JBuilder and shut down Tomcat.
- 8 Save the edited file to the copy of the `conf` directory.
- 9 Copy the `conf` directory back to your project's Tomcat directory.
- 10 Restart the web server and your web application.

Now, when you re-start the web application and the web server, your edited `serverXXXX.xml` file will be used instead of an auto-generated one.

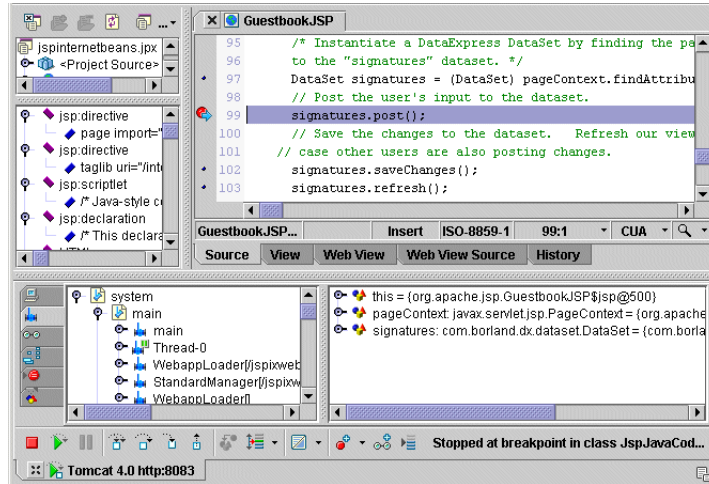
Web debugging your servlet or JSP

To web debug, launch the server in debug mode with the Web Debug command. You can then debug any servlet or JSP that the server processes. To make the debugger stop at a specific line of code, set a breakpoint in your servlet or JSP before starting the server.

Note JBuilder provides source debugging for JSPs. This allows you to trace through your JSP code directly; you don't need to trace through the servlet that the JSP is compiled to and try to match up line numbers in order to find errors. Note that Tomcat 4.1 does not support JSP source debugging.

The Web Debug command displays the debugger in the web server pane. Both web server and debugger messages are written to the Console view of the debugger.

Figure 10.4 Web debugging



Note Smart Swap does not work with JSP files.

For more information on debugging, look at the following topics:

- “Debugging JSPs” in the *Web Application Developer’s Guide*
- “Debugging Java programs” in *Building Applications with JBuilder*
- “Compiling, running and debugging tutorial” in *Building Applications with JBuilder*
- “Remote debugging” in *Building Applications with JBuilder*

Index

A

- application server
 - Borland target 4-1
 - configuring target 4-1
 - enabling 4-1
 - iPlanet target 4-1
 - libraries 4-6
 - WebLogic target 4-1
 - WebSphere target 4-1

B

- Borland
 - contacting 1-3
 - developer support 1-3
 - e-mail 1-5
 - newsgroups 1-4
 - online resources 1-4
 - reporting bugs 1-5
 - target application server 4-1
 - technical support 1-3
 - World Wide Web 1-4
- Borland Enterprise Server 5.1.1 - 5.2.1
 - deploying 5-7
 - libraries 4-6
 - starting 5-6
- Borland Enterprise Server console 5-10
- Borland Enterprise Server Management Agent 5-3
- Borland servers 5-1

C

- client tier 2-4
 - technologies 2-5
- clients
 - creating iPlanet 6.x+ 9-5
- configurations
 - copying server 4-5, 6-4
 - duplicate server 4-5, 6-4
- Configure Servers dialog box 4-1, 6-1
- configuring JBuilder
 - for Borland 5-1
 - for iPlanet 9-1
 - for Sybase 8-1
 - for WebLogic 6-1
 - for WebSphere 7-1
- configuring servers 4-1, 6-1
- configuring web server 10-1

D

- database drivers
 - adding to project 4-11
- debugging
 - iPlanet applications remotely 9-4
 - remote 5-8
 - remote for WebSphere 7-10
 - remote for WebSphere Server Advanced Edition 4.0 7-12
 - remote for WebSphere Single Server 4.0 7-11
 - WebApp 10-11
- default project
 - adding database drivers 4-11
- deploying iPlanet applications remotely 9-4
- deploying to Sybase servers 8-4
- deploying to WebSphere servers 7-7
- deployment descriptors
 - importing IBM 7-5
 - importing WebSphere 4.0 7-5
 - WebSphere 7-4
- disabled EJB wizards 4-8
- documentation conventions 1-1
 - platform conventions 1-3
- drivers
 - adding database drivers to project 4-11
 - database 4-10
 - JDBC 4-10

E

- EAR files
 - creating 2-8
- EAR wizard 2-8
- EJB wizards
 - disabled 4-8
- EJBC options 6-9
- Enable Server option 4-1
- enterprise beans
 - deploying to WebSphere servers 7-7
 - remote deploying to Sybase servers 8-4
- entity beans
 - creating WebLogic 6-8

F

- fonts
 - JBuilder documentation conventions 1-1

G

generated URL 10-8

I

iPlanet 6.0

Service Pack 4 9-2

iPlanet 6.5

Service Pack 1 9-2

iPlanet 6.x+

creating clients 9-5

deployment settings 9-5

hints and tips 9-1

startup process 9-3

iPlanet 6.x+ Deploy Settings dialog box 9-5

J

J2EE applications 2-1, 3-1

advantages 2-1

JBuilder support 2-5

learning about 2-8

J2EE technologies 3-1

JAR files 2-8

JDBC drivers 4-10

adding to project 4-11

JSP

source debugging 10-11

web debugging 10-11

JSP (JavaServer Pages)

web running 10-7

L

libraries

application server 4-6

Borland Enterprise Server 5.1.1 - 5.2.1 4-6

WebLogic Platform 8.1 4-6, 6-5

WebLogic Server 6.x 4-6, 6-5

WebLogic Server 7.x 4-6, 6-5

WebSphere 4.0 Single Server 4-6

WebSphere Server 5.0 4-6

WebSphere Server Advanced Edition 4.0 4-6

M

management agent

starting 5-3, 5-6

management port

Borland 5-6

middle tier 2-4

technologies 2-6

multi-tier architecture

benefits 2-4

N

newsgroups

Borland 1-4

public 1-4

P

partitions

Borland 5-4

port

changing Borland management 5-6

ports

VisiBroker Smart Agent 5-3

project

selecting web server for 10-4

Project Properties dialog box

Servers page 4-8

projects

adding database drivers 4-11

updating with latest server settings 4-8

updating with server settings 4-12, 6-15

R

remote debugging

Borland servers 5-8

iPlanet 9-4

JSPs on WebLogic Server 6-13

WebSphere 7-10

WebSphere 5.0 7-10

remote deploying

iPlanet 9-4

S

server

selecting iPlanet 9-2

selecting Sybase 8-3

selecting WebSphere 7-4

server configurations

copying 4-5, 6-4

creating duplicate 4-5, 6-4

creating similar 4-5, 6-4

updating 4-12, 6-15

updating projects with latest 4-8

server settings 4-1, 6-1

latest 4-12, 6-15

server.xml

creating custom file for Tomcat 10-10

servers

configuring 4-1, 6-1

enabling 4-1, 6-1

selecting 4-8

starting Borland 5-6

- supported 4-1
- service packs
 - adding 4-6, 6-5
 - required WebLogic 6-1
- servlets
 - web debugging 10-11
 - web running 10-7
- Smart Agent 5-3
- source debugging for JSPs 10-11
- startup process
 - iPlanet 6.x+ 9-3
- Sun iPlanet server
 - configuring 10-3
- supported servers 4-1
- Sybase servers 8-1
 - remote deploying to 8-4

T

- target application server 4-1
- Tomcat
 - configuring 10-1
 - creating custom server.xml file 10-10

U

- Usenet newsgroups 1-4

V

- VisiBroker
 - ORB, making available to JBuilder 5-3
 - Smart Agent 5-3

W

- WAR files 2-8
- web commands
 - configuring IDE for 10-6
- web debugging
 - JSP 10-11
 - servlets 10-11
- Web Run command 10-7, 10-8, 10-9
- web running
 - JSPs 10-7
 - servlets 10-7

- web server
 - changing port number 10-10
 - configuring 10-1
 - formatted output 10-8
 - raw output 10-9
 - selecting for project 10-4
 - starting 10-7
 - stopping 10-9
 - Sun iPlanet 10-3
 - Tomcat 10-1
 - WebLogic 10-3
 - WebSphere 10-3
- web view 10-8
- web view source 10-9
- WebApp
 - debugging 10-11
- WebLogic
 - panels in Deployment Descriptor editor 6-8
 - setting EJBC options 6-9
 - target application server 4-1
 - using existing code 6-8
- WebLogic Platform 8.1
 - libraries 4-6, 6-5
- WebLogic Server 6.x
 - libraries 4-6, 6-5
- WebLogic Server 7.x
 - libraries 4-6, 6-5
- WebLogic servers 6-1
- WebLogic web server
 - configuring 10-3
- WebSphere
 - target application server 4-1
- WebSphere 4.0 Single Server
 - libraries 4-6
- WebSphere Server 5.0
 - libraries 4-6
- WebSphere Server Advanced Edition 4.0
 - libraries 4-6
- WebSphere servers 7-1
 - deploying to 7-7
 - deployment descriptors 7-4
 - proprietary deployment descriptors 7-5
- WebSphere web server
 - configuring 10-3

