Borland®

What's New in Borland® JBuilder™ 7

Improve developer productivity, enhance performance

By Technical Publications
Borland Software Corp.

May 2002

Contents

Introduction

Borland® JBuilder™ 7 contains major improvements in developer productivity, as well as a cleaner, more intuitive user interface and dramatic performance enhancements.

Specific areas of change are as follows:

- Productivity enhancements
- Build system
- Runtime configurations
- Tool configurations
- Debugger
- UML™ code visualization
- JBuilder Test Runner
- Refactoring
- Javadoc
- Deployment
- Web applications
- Enterprise JavaBeans™ development
- Optimizeit™ Suite
- TeamSource™ DSP (Development Services Platform)

JBuilder™

white paper

The JBuilder 7 environment supports development, testing, and debugging of applications that take advantage of JDK® 1.4 features like the new `assert` keyword. In order to provide the best possible customer experience, JBuilder 7 is hosted on JDK 1.3.1.

Productivity enhancements

JBuilder 7 provides a number of productivity enhancements. One of them is apparent when you first start JBuilder: the Configure File Associations dialog box. This allows you to associate file types with JBuilder so that they open in JBuilder by default.

In the Configure File Associations dialog box, you can choose file types to associate with JBuilder. Choose from:

- Java™ class file (.class)
- Java source file (.java)
- JBuilder project (.jpr, .jpx)
- Change these settings by selecting **Tools|Configure File Associations**.

Other productivity enhancements include helpful editor enhancements, additions, and various improvements to search and find functionality.

Editor enhancements

Error handling

ErrorInsight™ performs a second pass for deeper errors. It flags missing imports, captures syntax errors such as missing semicolons, and catches agreement problems such as type mismatches and method signature mismatches.

Syntax errors are underlined in the editor. JBuilder uses red zigzags by default. You can change the color scheme in **Tools|Editor Options** by choosing the **Color** page.

Place your cursor over the syntax error to display a Tooltip that indicates the nature of the error. The Tooltip contains one of two buttons, depending on the nature of the error:

- **Magnifying glass button**: click this to open ClassInsight.™
- **Question mark button**: click this to open the Compiler error message help file.

Switch dialog box

The Switch dialog box allows you to switch quickly from one open file to another. Press **`Ctrl+b`** to invoke the Switch dialog box. All open files load in the list. Type a file name to narrow the list. You can select directly from the list. The selected file becomes the active one in the editor.

Brace matching

Select a brace to see that brace and its partner highlighted. Set Brace Match options in Tools|Editor Options, Editor page, Editor Options field. Change the color scheme in the Color page of Tools|Editor Options.

Indenting

Smart Indent behaves more intelligently in three important ways:

- The cursor can be anywhere in the line for the entire line to be indented.
- It nests new lines in the appropriate number of columns after an opening curly brace.
- With the Smart Paste option checked, lines that are pasted in are indented appropriately in the new location.

Change indent options on the Editor page of Tools|Editor Options. The behavior used by Smart Indent in older editions of JBuilder is still available from this page: deselect **Smart Indent** and select **Use Classic Behavior For Smart Indent.**

Single-click tab closure

Click the **X** icon on a tab or in the project toolbar to close the tab's associated file, message, or project. A whole X means that the file is unchanged or that the message does not require attention. A broken X means that the file has changed or that the message contains new information.

Pane control

View|Hide All and View|Show All replace View|Toggle Curtain. The menu item name toggles appropriately between Hide All and Show All, depending on the state of the panes:

- If all panes are visible, the menu item is View|Hide All. Selecting it hides the project, structure, and message panes. The message pane reappears when needed and can be individually closed by clicking its X icon.
- If the content pane alone is visible, the menu item is View|Show All. Selecting it reveals the project and structure panes. The message pane reappears when needed.

Shortcuts and searching

Project|Add Files/Packages

A new Classes page has been added so you can quickly find classes to add to your project. Enter a name in the Search For field and a list of matching classes is loaded dynamically.

Favorites folder

The file selection dialog box, which is used for File|Open, File|Add, and in other areas, now has a Favorites folder in the upper right corner. Anything added to the Favorites folder displays in a scrollable pane below the other existing shortcuts on the left side of the dialog box.

To add a custom favorite to the shortcuts, choose the **down arrow** next to the Favorites folder on the toolbar and choose **Add To Favorites**.

Find Classes dialog box

Find Classes supersedes Browse Classes. To invoke Find Classes, select **Search|Find Classes** or type either **`Ctrl+Minus sign`** or **`Ctrl+Hyphen`** and start typing the name of the file you want to open. Classes with names matching what you type are loaded in the Matching Classes list. Choose the class or classes you want to open and click **OK**. The source of those classes opens in the editor.

Toolbar search

Type the text you want to search for in the Find icon's combobox and press **Enter**. JBuilder scrolls to and highlights the line of code containing the text you searched for.

Press **`F3`** to search again forward. Press **`Shift+F3`** to search again backward. Previous search strings are available from the popup menu.

Replace In Path

Select **Search|Replace In Path** to invoke the Replace In Path dialog box. It offers the same set of options and parameters as the Find In Path dialog box, with the Replace option checked.

Build system

Build features vary by JBuilder edition.

The build system of JBuilder is based on Apache™ Ant, an open-source, Java-based build tool, and is also extensible as an OpenTool. The build system has several advantages and allows you to:

- Build projects with Ant.
- Extend the build system with an OpenTool.
- Specify dependencies between build targets.
- Create external build tasks, such as shell/console commands, to execute during the build process with the External Build Task wizard.
- Convert SQLj files to Java source files.
- Filter packages and exclude them from the build process.
- Add build targets to the Project menu and the toolbar.
- Use Clean to remove build output.

Wizards related to the build system, such as the External Build Task wizard, are on the Build page of the object gallery (File|New).

For more information on these features, see "Building and compiling Java programs" in *Building Applications with JBuilder* in the JBuilder documentation.

For more information on extending the build system, see "JBuilder build system concepts" in the JBuilder OpenTools documentation.

Running external Apache™ Ant files

This is a feature of JBuilder Enterprise.

If you have an existing project that already uses Ant, you can run Ant within JBuilder. JBuilder automatically recognizes `build.xml` files as Ant files. When you add a `build.xml` to your project (Project|Add Files/Packages), the node displays with an Ant icon instead of the usual XML icon. The targets in the `build.xml` file are displayed as child nodes.

Note: You can use a name other than `build.xml` , but you must modify the properties of the XML file for JBuilder to recognize it as an Ant build file. Right-click the **XML file** in the project pane and choose **Properties**. Choose the **Ant tab** and select the **Ant Build File** option.

If you right-click a **`build.xml`** file in the project pane and choose **Make**, then Ant is run against the file using the file's default target. If you select one or more of its target nodes, Ant is run against the file using the selected targets. Output from the Ant run is routed to the Ant tab of the JBuilder message pane. You can navigate to files with errors in them by clicking on the error messages in the message pane.

External Build Task wizard

This is a feature of JBuilder Enterprise.
Use the External Build Task wizard to create external build tasks, such as shell/console commands, to execute during the build process. For example, you might have a `.BAT` or `.EXE` on Windows® or a `.sh` or executable on Linux® or UNIX that you want to execute every time you do a build. The External Build Task wizard is located on the Wizards menu and on the Build page of the object gallery.

SQLj

This is a feature of JBuilder Enterprise.

JBuilder now recognizes `.sqlj` files in the build process. Use Tools|Enterprise Setup|SQLj to configure your IBM® DB2® or Oracle® SQLj by pointing to the SQLj executable. In the Project Properties, you can specify which SQLj translator should be active for the project.

Once your project has an active SQLj translator, SQLj is run against any `.sqlj` files in your project as part of the build process, and the generated `.java` files appear as children of the SQLj node. The generated `.java` files are then compiled as part of the overall build process.

Filtering packages

This is a feature of JBuilder SE and Enterprise.

JBuilder provides a new feature that allows you to exclude packages from the build process. However, if the Dependency Checker determines that there is a dependency on classes in the excluded packages, those classes are compiled.

The automatic source packages feature **must** be enabled on the General page of the Project Properties dialog box (Project|Project Properties) to use the package filtering feature. A <Project Source> node also displays at the top of the project pane when the automatic source packages feature is enabled. This node contains all the source packages and source files in the project, except packages and files that have been added manually. You can use this node to quickly filter source packages. Right-click the <Project Source> node, choose Apply Filter, then choose one of the submenu commands.

To exclude packages, right-click a **package** or packages, choose **Apply Filter**, and choose a command from the submenu. Filtered packages appear in a Package Filters folder in the project pane. To remove filters and include the packages in the build system, right-click the **Package Filters** folder and choose **Remove All Filters**. You can also expand the Package Filters folder and selectively

include packages. The Apply Filter and Remove All Filters commands are also available on the Project menu.

Manually added packages and files can't be excluded with the Apply Filter command. You must remove them from the project to exclude them from the build process. Also, any buildable nodes that are children of the project node, such as Java source files added by wizards, are compiled and can't be excluded unless you remove them from the project. Essentially, any files or packages that display above the Package Filters folder are not filtered and are included in the build process.

Changes to build menus

Configuring the Project menu

This is a feature of JBuilder Enterprise.

For convenience, JBuilder allows you to configure the first group of the Project menu. You can change the order of targets and add additional targets, such as Clean, external build tasks, and Ant targets. Configuring the Project menu also configures the toolbar. Choose **Project|Project Properties|Build|Menu Items** to configure the Project menu.

To configure the Project menu for future projects, make the changes in the Default Project Properties.

Project pane context menu

The Clean command has been added to the project pane's context menu. Clean removes all build output, such as the `classes` directory, JARs, and so on. Right-click the project file and choose Clean to remove the entire project's build output. To clean individual files, rather than the entire project, select them in the project pane, right-click, and choose Clean. As with the Make and Rebuild commands, Clean only appears on the context menu when appropriate nodes are selected.

Toolbar changes

By default, Make is on the main toolbar and Rebuild is now on the drop-down list next to Make. If you add any targets to the Project menu, the first target in the list displays on the toolbar and additional targets display on the drop-down list.

Build page of Project Properties

The Build page of the Project Properties has new tabs: These are features of JBuilder Enterprise.

- SQLj: specify the SQLj translator.
- Ant: add custom Ant libraries containing your own build tasks and use a different version of Ant by adding a library with the Ant JARs.
- Menu Items: configure the Project menu.

Building from the command-line

This is a feature of JBuilder SE and Enterprise.

The JBuilder command-line **-build** option has been modified to accept build targets as arguments. You can now specify one or more targets, which are executed in the order listed. If a target isn't specified, Make is the default target.

Among the **-build** arguments, projects are distinguished from targets by the the `.jpx` or `.jpr` extension. Any argument that ends with .`jpx` or `.jpr` is assumed to be a project; any argument that doesn't have these extensions is assumed to be a target name.

The command is in this form:

```
jbuilder -build <project1.jpx> [ [<target1> <target2> ...]
    [<project2.jpx> [<target3> ...] ] ... ]
```

For example:

```
jbuilder -build myproject.jpx rebuild
```

```
jbuilder -build myproject.jpx clean make myotherproject.jpx
```

For more information on this option, see "JBuilder command-line arguments" in *Building Applications with JBuilder* in the JBuilder documentation.

Runtime configurations

The interface for tailoring running and debugging configurations and for selecting among configurations has changed. Available features vary by JBuilder edition.

The Run page of the Project Properties dialog box displays the available configurations. There are two ways to access this page: select **Project|Project Properties** and choose the **Run** page, or select **Run|Configurations**.

- Click **New** in the Run page to access the Runtime Properties dialog box. Create a configuration for running, debugging, or optimizing a project or runnable file.
  - In JBuilder Personal: Only one configuration at a time is supported, but that configuration is completely editable.
  - In JBuilder SE and Enterprise: Multiple configurations are supported. All are editable.
- The Debug page of the Project Properties box is now part of the Runtime Properties dialog box.

You can tailor configurations in the following ways:

- Specify which configuration is the default. (In JBuilder Personal, the existing configuration is always the default.)
- Associate a build target with run and debug configurations.
- Determine which configurations appear as choices on the Run and Debug context menus. This is a feature of JBuilder SE and Enterprise. These configurations show in submenus in the following locations:
  - On the project pane context menu.
  - On the Run menu, either as menu items or submenus, depending on context.
- Delete all configurations, including the default.

All runtime configurations for a file or project are listed in the Run menu.

Some wizards that generate runnable files prompt you to configure runtime/debugging properties for that file, if no configuration for that file type exists.

Each runnable file's runtime configuration can be used to run other files of the same type. For instance, a configuration made for one applet can be used to run other applets.

Runtime keybindings

`F9` runs the active project using the default configuration. `Shift+F9` debugs the active project using the default configuration. If no default configuration is specified, the Project Properties|Run page appears. Select or create a configuration and click OK. Press the keystrokes again to run or debug the project using that configuration. This does not make the new configuration the default.

Tool configurations

A program that you add to the Tools menu of JBuilder can run as an external tool or as a service within JBuilder. When you run a program as an external tool, you can shut down JBuilder and the tool continues to run. When you run a program as a service, its output appears in the message pane. You can stop the service by clicking the red Stop button on the message pane's tab, and you can start it again by clicking its green Run button. If you attempt to exit JBuilder without stopping the service first, a message box appears asking if you want to terminate the service. If you decide to terminate the service, it stops and you exit JBuilder; otherwise the service keeps running and JBuilder does not exit.

Debugger

The following debugger enhancements have been added to JBuilder 7:

Debugger options

You now set debugger options, such as SmartStep settings and remote debugging options, through a runtime configuration. A runtime configuration is a set of pre-configured parameters. Using preset parameters saves you time when running and debugging, because you only have to set the parameters once.

You use the Debug page of the Runtime Properties dialog box to set debug options. To display this page, choose

Run|Configurations, then **New** or choose Project|properties, then choose the **Run** tab and click the **New** button. On the **Runtime Properties** dialog box, click the **Debug** tab. Debug options have not changed.

Classes With Tracing Disabled

The Classes With Tracing Disabled view and dialog box are now available in all JBuilder editions.

In JBuilder Personal, three basic classes (`java.lang.Object`, `java.lang.String` and `java.lang.ClassLoader`) are available. You cannot add, modify, or delete classes; however, you can choose to step or not step into those classes.

SmartStep configuration is now available in all JBuilder editions.

Evaluate/Modify dialog box

You use the Evaluate/Modify dialog box (Run|Evaluate/Modify) to evaluate expressions, change the values of data items, and evaluate method calls. CodeInsight™ and syntax highlighting features now display when you enter an expression into the Expression field.

Remote debugging

This is a feature of JBuilder Enterprise.

The default address when attaching to a remote computer has been changed to 3999.

For more information on debugging, see the following chapters in *Building Applications with JBuilder* in the JBuilder documentation.

- "Debugging Java programs"
- "Remote debugging"

UML™ Code Visualization

This is a feature of JBuilder Enterprise.

The UML browser now displays Java source files dynamically even if they haven't been compiled, but only if they are on the source path. A message displays in the UML browser indicating that the UML diagram may not be accurate. However, if a source file isn't on the source path, the .class file must be generated first. A message prompts you to compile the project to generate the class files for the UML diagram. If the class files are out of date, for example the source file has been changed but hasn't been recompiled, a message displays in the UML browser indicating that the UML diagram may not be accurate. For an up-to-date and accurate UML diagram, it's always best to compile before you choose the UML tab.

The UML browser now recognizes reverse dependencies from classes to JSPs. For example, a bean generated by the JSP™ wizard can now link to the JSP that uses it. It doesn't have to be a JSP bean; it could be any class that the JSP uses.

Printing support

File|Print now supports larger paper sizes for printing large UML diagrams: A1 (594 mm x 841 mm) and A0 (841 mm x 1189 mm).

JBuilder™ Test Runner

This is a feature of JBuilder Enterprise.

Improvements have been made to the GUI of the JBuilder Test Runner:

- A new progress bar indicates the percentage of tests completed.
- Text output and icons have been improved.
- Now you can right click any node in the Test Hierarchy page or the Test Failures page of the JBuilder Test Runner and choose Run Selected or Debug Selected. This is useful when you want to investigate a test failure.

For more information on JBuilder Test Runner, see "Unit Testing" in *Building Applications with JBuilder* in the JBuilder documentation.

Refactoring

This is a feature of JBuilder SE and Enterprise. The following new refactoring features have been added to JBuilder 7:

New refactoring features

Feature	Description
Extract method	Turns a selected code fragment into a method.
Change method parameter	Adds, renames, deletes, and re-orders a method's parameters.
Introduce variable	Replaces the result of a complex expression, or part of the expression, with a temporary variable name.
Surround with try/catch	Adds a `try/catch` statement around the selected block of code.

Refactoring commands have now been added to the Edit menu.

Warnings are now displayed for EJB™ refactorings. You will need to update all relevant source files to support the refactoring.

The Refactoring tab now displays a open X to visually indicate that a refactoring has not yet been completed. When the refactoring is finished, the X is closed.

For more information on refactoring, see "Refactoring code symbols" in *Building Applications with JBuilder* in the JBuilder documentation.

Javadoc

This is a feature of JBuilder SE and Enterprise.

You can now use the the project JDK when running Javadoc instead of the JDK that hosts JBuilder. For example, if your project uses JDK 1.4, you can run that version of Javadoc instead of JDK 1.3.1, the version that hosts JBuilder.

On the specify doclet command-line options page of the wizard, the @use option has been moved to the Generate "Use" page option. The option generates documentation for @use tags in your source code. It generates one Use page for each package and a separate one for each class and interface. The package use file is called `package-use.html`; the class use file is `class-use/classname.html`. This page describes what packages, classes, methods, constructors, and fields use any part of the given class, interface, or package. The option is ignored for the JDK 1.1 doclet type.

To delete all HTML files in the configured doc directory, choose the **Clean** build target. Choose **Rebuild** to clean the directory first, then rebuild the HTML files.
For more information on Javadoc, see "Creating Javadoc from API source files" in *Building Applications with JBuilder*.

Deployment

These are features of JBuilder SE and Enterprise.

Archive Builder™

The Archive Builder™ has been moved to the Build page of the object gallery.

The Archive Builder has a new archive type, Native Executable, that can be used to bundle an application JAR file with native executable wrappers for Windows, Linux, Solaris,™ and Mac® OS X. Several other archive types, such as Application and J2EE™ Application Client, also provide this feature. See "Native Executable Builder" below.

Selecting a method for determining the application's main class

Due to the changes in runtime configurations, this step has changed. The first option, Determine Main Class From Runtime Configurations, has this behavior:

- Uses the main class specified in the runtime configuration set as the default on the Run page of the Project Properties dialog box.
- Uses the first configuration in the list if a default runtime isn't specified on the Run page.

- Doesn't specify a main class if an application runtime doesn't exist or the one selected doesn't have a main class.

Caution: If a main class isn't specified, the following **won't** execute:

- Launching native executables
- Using `java -jar <jarname>` from the command-line
- Double-clicking a JAR

Deleting contents of an archive node

The Remove Generated Files command on the project pane's context menu has been replaced with the Clean command. Right-click the **archive node** and choose **Clean** to remove the node contents but not the node itself.

Native Executable Builder

The Native Executable Builder, available on the Wizards menu and the Build page of the object gallery, is a shortcut to the new Native Executable archive type. This wizard bundles an application JAR file with native executable wrappers for Windows, Linux, Solaris, and Mac OS X.

Important: Because the JDK is not bundled with the JAR file, it must be installed on the user's computer in order to run the executable.

Running executables

Note that the JDK is **not** bundled with the JAR file, so the JDK must be installed on the user's computer in order for the executable to run. The platform-specific executable file looks for the installed JDK in the following location:

- Windows: Registry.
- Linux/Solaris: JAVA_HOME environment variable and the user's path.
- Mac OS X: pre-defined location for the JDK.

If you create the executable on the Windows platform and move it to other platforms, you may need to change the permissions to make it executable.

Choosing the Mac OS X option creates an application that is launchable only from a command line. In order to create an application that is launchable from the Finder, Mac users need to create an application bundle. Please refer to the Apple Mac OS X Developer documentation with regards to bundles and application packaging.

Web applications

Web development is a feature of JBuilder Enterprise. Applet development is a feature of all editions of JBuilder.

The following new Web applications features have been added to JBuilder 7:

Server configuration

If you are not using the Borland® Enterprise Server, Tomcat 4.0 is the default server. To enable the Apache Tomcat configuration,

1. Open the **Configure Servers** dialog box (Tools|Configure Servers).
2. In the tree on the left, choose **Tomcat 4.0**.
3. Click **Enable Server** at the top of the right side of the dialog box.
4. Click **OK**.

Project configuration

When you run your servlet or JSP, you now need to configure a single server or modular services for the project.

To configure a single server for your project:

1. Open the **Server page** of the Project Properties dialog box.
2. Select the **Single Server For All Services In Project** option.
3. Select the **server** from the drop-down list.
    - 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.

- If you want to make changes to the configuration settings for the selected server, click the ellipsis button and edit the settings you want on the General and Custom pages. Click OK when you're finished.

To use different servers for different services:

1. Open the **Server page** of the Project Properties dialog box.
2. Select the **Modular Services Provided By Different Servers** option.
   - 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.
   - 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.
   - To use a selected Web server, click the JSP/Servlet service. In the Server drop-down list on the right side 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.
3. Click **OK** again to close the Servers page.

Runtime configuration

You now Web run and Web debug through a Web run configuration. You can create the configuration when you create an applet, servlet or JSP with a wizard. You can also create a configuration with the Run Configurations dialog box.

For servlets and JSPs, the configuration consists of two parts—the server configuration and the servlet or JSP configuration. The UI for this configuration is on the Runtime Properties Server page. For applets, the configuration UI is on the Runtime Properties Applet page.

Web applications wizards

The Applet, JSP, and Servlet wizards now contain a Define runtime configuration page where you create a runtime configuration.

For more information on Web applications, see "Developing Web applications" in the *Web Applications Developer's Guide* in the JBuilder documentation.

Enterprise JavaBeans™ development

EJB™ designer improvements

The EJB designer includes several improvements:

- You can group sets of Enterprise Javabeans with views , which help you organize and develop complex EJB projects. While your EJB module can contain many enterprise beans, only the ones on the current view are visible at any one time in the EJB Designer.
- You can quickly arrange the bean representations on a view with a single menu command, Views|Arrange EJBs. The bean representations are automatically arranged in a logical pattern.
- The EJB Designer context menus have been redesigned.
- The EJB Designer has a toolbar that contains icons for the menu commands.
- You can add new ejbCreate methods to a bean using the EJB Designer, and you can use a bean's inspector to add and modify parameters of an ejbCreate method.
- You can import an enterprise bean into an EJB module using the EJB Designer. You'll find this useful if you want to import a bean from one EJB module to another or if you've obtained a bean that has no accompanying deployment descriptors.

- The EJB Designer can get you started developing entity beans with bean-managed persistence (BMP).
- You can use the EJB Designer to design your entity beans, then use the Create Schema From Selection menu command or icon on the toolbar to create a schema from the beans you designed.

Wizards

The EJB Module From Descriptors wizard can import vendor-specific information in Borland and BEA® WebLogic® deployment descriptor files when creating an EJB module. It can also detect and update your project source directories using class names in the descriptors.

The new Project From Existing Code wizard can identify EJB deployment descriptor files and import vendor-specific information in Borland and BEA WebLogic descriptors to create EJB modules for your new project. The wizard is on the Project page of the object gallery.

Saving a copy of the deployment descriptors

You can choose to save a copy of the deployment descriptors in their current state each time you save an EJB or EAR module by setting the new Copy Descriptors option. To find this option, right-click the **EJB** or **EAR module node** in the project pane, choose **Properties**, click the **Build** tab, and select the **EJB** or **EAR page**.

Server support

JBuilder now supports Borland® Enterprise Server, AppServer™ Edition 5.0.2.

JBuilder has improved support for WebLogic 6.x+ servers including WebLogic 7.0. It also has improved support for IBM® WebSphere® 4.0 and iPlanet™ 6.x+ servers.

Configuring and selecting a server

You now configure the servers you are going to use for EJB development using the Tools|Configure Servers dialog box.

How you select a server for EJB development has changed. Use Project|Project Properties and click the Server tab to use the Server page to make your selection. You can select a single server for all of your development needs, or you can specify different servers for different services. For example, you might choose one server to provide EJB services and another to provide JSP/Servlet services for your Web development.

More flexible runtime configurations for the server

To create a runtime configuration to run the server within JBuilder, choose **Run|Configurations** and click the **Run** tab, click the **New** button, then click the **Server** tab, which replaces the EJB tab in previous JBuilder releases. Using the Server page, you can decide exactly which services you need when you run the server and eliminate unneeded ones. You can also customize the runtime configuration command-line, the archives to be deployed, libraries, launch URI for JSPs and servlets, and so on.

Optimizeit™ Suite

JBuilder 7 provides tight integration with Borland Optimizeit Suite, a complete performance solution for Java.

- Optimizeit™ Profiler: Helps you find and fix Java memory leaks, CPU bottlenecks, and excessive temporary object use.
- Optimizeit™ Thread Debugger: Checks the progress of all threads running in an application to assure speed and reliability of code.
- Optimizeit™ Code Coverage: Lets you visually monitor classes and methods in your code and assure applications are ready to deploy.

Optimizeit Suite features are available on the following JBuilder menus and dialog boxes:

- Run|Optimize Program
- Web Optimize

- Runtime Properties | Optimize tab | Profiler page
- Runtime Properties | Optimize tab | Code Coverage page
- Runtime Properties | Optimize tab | Thread Debugger page

From within Optimizeit Suite, you can return to JBuilder 7 by clicking the JBuilder button on the source code window; this returns you to JBuilder with your tested code in the content pane.

Go to http://www.borland.com/optimizeit/ for more information on purchasing and installing Optimizeit.

TeamSource™ DSP (Development Services Platform)

The TeamSource DSP plug-in is a feature of JBuilder SE and Enterprise.

Borland TeamSource DSP (Development Services Platform) is a collaborative product development platform for distributed teams—whether they are down the street or around the world. TeamSource is one of the first secure software collaboration solutions based on an Internet architecture that cuts across firewalls. It combines essential source code management features (storage, versioning, and file sharing) with integrated business messaging (secure, archived, channel-based communications that are integrated into the development process) to unite teams across functions, locations, and companies. For a limited time, JBuilder SE and Enterprise will include a free 90-day trial of TeamSource! For information on the trial, see teamsource.html in the offer/teamsource directory on your JBuilder CD.

Conclusion

Increase productivity with the leading cross-platform Java development environment. The JBuilder development environment significiantly enchances developer productivity with timesaving tools that speed time-to-market. Leverage existing projects and reduce development costs with a two-way visual EJB 2.0 designer, UML code visualization, refactoring, unit testing and more. The enhanced JBuilder provides a flexible, open solution for developing and deloying Java applications.

For more information on JBuilder 7, please visit our Web site at http://www.borland.com/jbuilder.

Borland®

100 Enterprise Way
Scotts Valley, CA 95066-3249
www.borland.com | 831-431-1000