What's New in Borland® JBuilder® 8

The leading Java™ development solution

A Borland White Paper

By Borland Staff

November, 2002

Borland®

Contents

What's new in JBuilder® 8

Borland® JBuilder® 8 contains major improvements in developer productivity, project management, and the build system. The Borland Web Services Kit for Java™ has been incorporated into this release and a new set of features help development, using the Apache™ Struts Open Source framework. There are also improvements to many other areas of the product, as listed in this document.

Specific areas of change are as follows:

- JDK 1.4.1
- Project management
- Debugger enhancements
- Unit Testing enhancements
- UML™ code visualization
- Web development
- Web Services
- XML
- Run configurations
- Build system
- Deployment
- Productivity enhancements
- Team development enhancements
- Supported J2EE™ servers
- Javadoc enhancements
- Refactoring enhancements
- Accessibility features
- New file I/O API in Borland JDataStore™

JDK™ 1.4.1

JBuilder 8 is now hosted on JDK™ 1.4.1, in order to take advantage of performance and productivity improvements. These include faster performance in the client, mousewheel and drag-and-drop support, improved focus management, Section 508 compliance for accessibility, and, on Linux,® more predictable UI behavior.

Project management

Project management has been extended to include new features.

Drag and drop into the project pane

You can now drag and drop files from the desktop into the project pane so that they become part of your project. The file must be a type that JBuilder recognizes. The file is opened in the editor, but not added to the project. If you want the file added to your project, add it using the Add Files/Packages button at the top of the project pane.

Creating a new empty file

The quickest way to create a new empty file in a JBuilder project is to use the Create New File dialog box accessed with the File|New File command.

Project groups

This is a feature of JBuilder Enterprise.

JBuilder now has project groups. Project groups are containers for projects and can be useful when working with related projects. For example, you might have two projects that have dependencies on each other, such as a client and a server. Another logical grouping would be projects that use the same source files but have different settings, such as different target application servers or different JDKs. In addition, project groups provide other advantages, such as ease of navigation between projects and building projects as a group.

For information about project groups, see "Working with project groups" in "Building Applications with JBuilder" in the product documentation.

Directory views

This is a feature of JBuilder Enterprise.

You can now add a directory view to your project which points to any directory of your choosing. A directory view is much like a live view of your directory or directory tree that displays all file types. Once you've created a directory view and when changes occur to the actual contents of the directory or its subdirectories, the directory view will be updated in the project pane. For information about directory views, see "Adding a new directory view" in "Building Applications with JBuilder" in the product documentation.

Importing VisualCafé™ projects

JBuilder has a new Import VisualCafé Project wizard that creates a new JBuilder project and imports all the VisualCafé™ project files into the JBuilder project. To use this wizard, choose File|New and click the Import VisualCafé Project icon.

Adding a project as a required library

If you have a project that is dependent on another, you can add the project upon which your project is dependent to the list of required libraries for your project. Use the Required Libraries tab on the Paths page of the Project Properties dialog box.

Debugger enhancements

The debugger includes the following new features, as well as various productivity updates:

- Smart Swap
- Set Execution Point
- Smart Source

Smart Swap

This is a feature of JBuilder Enterprise.

If you change source code while debugging, Smart Swap modifies, compiles, and updates the modified files. You continue debugging in the same debugging session. With Smart Swap, you can test your code, make changes, and continue in the same session from the current execution point. For more information on Smart Swap, see "Modifying code while debugging" in the "Debugging Java programs" chapter of "Building Applications with JBuilder" in the product documentation.

The Smart Swap feature of JBuilder is based on the Hot Swap technology in JDK 1.4.

Set Execution Point

This is a feature of JBuilder Enterprise.

Set Execution Point allows you to set the execution point for the current stepping thread. This will change the execution point from its current location. You may want to set the execution point after you use Smart Swap. For more information, see "Setting the execution point" in the "Debugging Java programs" chapter of "Building Applications with JBuilder" in the product documentation.

Smart Source

This is a feature of JBuilder Enterprise.

Smart Source allows you to change the view of your code, so that you can view either the Java source code or the non-Java source code while debugging. For more information, see "Debugging non-Java source" in the "Debugging Java programs" chapter of "Building Applications with JBuilder" in the product documentation.

The Smart Source feature of JBuilder is based on the JSR 45 technology in JDK 1.4.

Debugger productivity updates

JBuilder 8 includes the following debugger productivity updates:

- The Cut, Copy, and Paste commands are available for all variable types.
- The Change Watch Expression and Change Watch Description commands have been combined into the Change Watch command.
- The ExpressionInsight™ window now displays until you press a key to close it.
- The Files Modified dialog box now allows you to choose how to proceed while debugging.

Unit testing enhancements

These are features of JBuilder Enterprise.

JBuilder unit testing support has been expanded to support server-side testing with Cactus.

JUnit support has also been enhanced. Here's a list of new unit testing features.

- Cactus support—Cactus extends JUnit to provide unit testing of server-side Java code. See "Working With Cactus" in "Building Applications with JBuilder" in the product documentation.
- Cactus Setup wizard—The Cactus Setup wizard (Wizards|Cactus Setup) configures Cactus support for your project. See "Cactus Setup wizard" in "Building Applications with JBuilder" in product documentation.
- EJB™ Test Client wizard—The EJB Test Client wizard can now generate two new types of test clients: JUnit test client and Cactus JUnit test client. The EJB Test Client Wizard is available from the Enterprise page of the object gallery. See "Running and testing an enterprise bean" in "Enterprise JavaBeans™ Developer's Guide" in the product documentation.
- JUnit Test Collector—New options in the Runtime Configuration Properties dialog box for a test-type runtime configuration allow you to easily identify your test cases. See "JUnit Test Collector" in "Building Applications with JBuilder."
- Unit Testing Stack Trace Filter—You can now specify a stack trace filter for unit tests, allowing you to concentrate on the stack trace information that's useful to you. See "Defining a stack trace filter" in "Building Applications with JBuilder."
- JUnit 3.8—The version of JUnit that ships with JBuilder has been upgraded to 3.8.

UML™ code visualization

This is a feature of JBuilder Enterprise.

The UML™ page has been removed from Project Properties and the two options on that page have moved to other property pages:

- Filtering of UML diagrams has moved to the new Class Filters page of Project Properties. Choose UML Diagram from the Name drop-down list and add or remove classes and packages to include or exclude them from the diagrams.
- The UML option for including generated source references, Diagram References From Generated Source, has moved to the General page of Project Properties. This option, when selected, includes such references as IIOP® files and EJB stubs in the UML diagrams.

Packages and classes that aren't shown in the UML diagram now display in a lighter color in the structure pane. They aren't displayed if they're filtered out or if they're redundant and removed for clarity.

Web development

These are features of JBuilder Enterprise.

A new set of JBuilder features helps Web development with the Struts Open Source framework. Struts is based on the Model 2, or Model-View-Controller, approach to software design. In this framework, the model contains the data, the view is the presentation of the data, and the controller controls the interaction between the model and the view. The view is typically a JSP™ page. The controller is a Struts servlet called ActionServlet. The model can be any data access technology, from JDBC® to an EJB. This framework, which is a collection of classes, servlets, JSP tags, a tag library, and utility classes, provides a clean division of the HTML from the Java code, and the visual presentation from the business logic.

The feature set of JBuilder provides wizards and tools that allow you quickly create a Struts-enabled Web application. Tools and wizards include:

- Frameworks as libraries—you can now use the Configure Libraries dialog box (Tools|Configure Libraries) to create a framework, or a collection of files, as a library. The Struts, Cocoon, JSTL, and InternetBeans™ Express frameworks are available with JBuilder.
- Struts-enabled WebApp and Web Application wizard.
- Struts-enabled JSP and JSP wizard.
- Struts-enabled web.xml file.
- JSP to Struts Conversion wizard—converts an existing HTML or JSP page to use Struts-specific tags.
- Struts Configuration Editor—visual editor for editing the struts-config.xml file.
- Action wizard—creates the Action class for your Struts WebApp and registers it in the struts-config.xml file.
- ActionForm wizard—creates an ActionForm using fields from the specified JSP and registers it in the struts-config.xml file.
- JSP From ActionForm wizard—Takes an ActionForm and creates a JSP page, populating the JSP with fields from the ActionForm class.

For more information on JBuilder and Struts, see the following chapters in the "Web Application Developer's Guide" in the product documentation.

- "Developing JavaServer Pages™"
- "Using the Struts framework in JBuilder"
- "Editing the struts-config.xml file"

Other enhancements include:

- WebApp Wizard, WAR generation options—The Always Create Archive When Building option in the Project option on the WebApp Wizard has been renamed to Build WAR and expanded to four choices
  1. When Building Project Or WebApp——creates the WAR when building the project or the Web application.
  2. When Building WebApp Only——creates the WAR only when building the Web application.
  3. When Building Project Only——builds the WAR only when building the project.
  4. Never—Does not build a WAR file.

- Improved JSP tag library and framework support allows you to select tag libraries and frameworks to use in your JSP or WebApp.

- JSTL (JavaServer Pages Standard Tag Library) ships with JBuilder and is automatically available as a tag library or framework in the JSP Wizard and Web Application Wizard.
- The Classes page of the Properties dialog box for a WebApp node has been changed to allow you to exclude dependencies.
- The icons on the Web tab of the object gallery are only available once you have selected a server for your project.
- Tomcat 4.1 is available with JBuilder 8. Note that Tomcat 4.1 does not support JSP debugging.

Web Services

This is a feature of JBuilder Enterprise. The Web Services features vary according to the Web Services toolkit selected.

The Borland Web Services Kit for Java is now incorporated into JBuilder 8 with further feature enhancements. For more information, see "Developing Web Services" in the online Help.

Although it's useful to learn about the technologies behind Web Services, JBuilder provides wizards and tools for quickly developing and consuming Web Services. JBuilder works with a variety of Web Services toolkits. Once a toolkit is selected, the JBuilder wizards use the toolkit to enable projects for Web Services, to import services, and to export Java classes as Web Services.

Exporting Enterprise JavaBeans (EJBs) as Web Services in JBuilder requires no additional work other than configuring the project for Web Services. Simply develop your EJB application as usual, configure the project for Web Services with the Web Services Configuration Wizard, and run the project with the Web Services Server run configuration created by the wizard. By default, JBuilder automatically exposes all the stateless session beans with business methods in the remote interface. You can also override this default behavior and select only the EJB modules, beans, and methods that you want to expose as Web Services.

Web Services features in JBuilder, which vary according to the toolkit selected, include:

- Configuring your project for working with Web Services.

- Creating a WSDL document that describes the Web Service you've developed.
- Creating client stubs to invoke a Web Service.
- Importing an EAR or a WSDL describing a Web Service and creating classes to invoke the service.
- Creating server-side code to host the service locally.
- Explicitly exporting Java classes as Web Services.
- Automatically exporting EJB stateless session beans with business methods in the remote interface.
- Monitoring SOAP messages between the client and the service while invoking the service.
- Searching for and publishing Web Services to a UDDI registry with the Web Services Explorer.

The JBuilder Web Services wizards are located on the Web Services page of the object gallery (File|New):

- Web Services Configuration wizard for configuring your project for Web Services.
- Export As A Web Service wizard for exporting Java classes as Web Services.
- Import A Web Service wizard for importing Web Services from a WSDL or an EAR.

JBuilder also provides these tools for working with Web Services:

- Web Services Explorer for browsing and publishing Web Services. You can use the Explorer to browse to WSDL documents and import them, browse to WSIL documents, publish businesses and services, and monitor UDDI SOAP messages.
- TCP Monitor for monitoring SOAP requests and responses between the client and the server.

XML

SAX Handler wizard

This is a feature of JBuilder Enterprise.

The SAX Handler wizard (File|New|XML) has been updated to use JAXP (Java API for XML Processing), which has been added to JDK 1.4. The wizard also allows you to specify a JAXP parser or Xerces for parsing. In addition, the wizard has a new runtime configuration page.

Cocoon

This is a feature of JBuilder Enterprise.

- Cocoon has been upgraded to version 2.0.3 and new files are added to the Cocoon node.
- The Web Application wizard (File|New|Web) includes the option to select Cocoon as the framework for the Web application.
- The Cocoon Web Application wizard now opens the Web Application wizard with Cocoon selected as the framework by default.
- Cocoon Run has changed to Web Run Using Defaults.

For more information on the Cocoon features, see the "XML Developer's Guide" in product documentation.

XML options in the IDE

This is a feature of JBuilder SE and JBuilder Enterprise.

There is a new option on the XML page of the IDE Options dialog box (Tools|IDE Options) called Ignore DTD. By default, JBuilder ignores DTDs in the editor. When this option is checked, JBuilder doesn't resolve the DTD and doesn't report any errors in the structure pane. This makes it possible to work offline and also results in faster response time if you're working online. If this option is checked, the editor must resolve the DTD each time and also reports any errors in the structure pane.

Run configurations

The Runtime Configuration Properties dialog box has been re-designed. The Run tab contains a drop-down list from which you can choose the type of application, rather than tabs. When you choose Edit or Copy, this list is not available for modification, although the type of application is displayed in the field. The Build Target field has moved to the top of the dialog box.

JBuilder now includes an OpenTool runtime configuration type which lets you run, debug, and optimize your OpenTool project in JBuilder just like other projects. You no longer have

to exit JBuilder, create a JAR file and copy it to the `<JBuilder home>\lib\ext` or `<JBuilder home>\patch` directory, then restart JBuilder.

When you run an OpenTool project using the OpenTool runner, a temporary config file is generated in the `<outpath>\...\config-temp` directory, and a new instance of JBuilder is started using this temporary configuration file. This configuration file will be removed when the tracker is closed. To use the OpenTool runner in your project, choose Run|Configurations... and create a new runtime configuration, selecting Open Tool as the type to create.

Build system

Apache™ Ant support

Apache Ant has been updated to version 1.5.1 and there are several new Ant features:

These are features of JBuilder Enterprise.

- Ant build.xml files display in the project pane with the relative path. If you want to turn this option off, right-click the build.xml file in the project pane, choose Properties, choose the Ant tab, and uncheck Show Relative Path.
- Use the Ant wizard (Wizards|Ant) to add build files to your project. If you use the wizard, build files of any name are automatically recognized as Ant build files. You can also add build.xml files recursively in the wizard.
- If you want Ant to use the project JDK, set the Use Project JDK When Running Ant option on the Ant page of the Build page of Project Properties (Project|Project Properties).

Build page of Project Properties

Several changes have been made to the Build page of Project Properties. For more information on the options, click the Help button.

Java page

The Java page includes these changes:

- You can specify a compiler in JBuilder Enterprise.
- Borland Make (bmj) has been added.
- javac—uses the JDK javac hosted by JBuilder.
- Project javac—uses the project's JDK javac.
- The Enable Assert Keyword option has been moved to this page from the General page of Project Properties.

General page

The new General page includes these options, which vary by JBuilder edition:

- Autosave All Files Before Compiling.
- Refresh Project Before Building.
- Generate Source To Output Path.
- Autocancel Build On Error.
- Always Build Before Refactoring.
- Check JSPs For Errors At Build-Time.
- SQLJ™ Translator.

Some of these options have been moved from other pages.

Ant page

This is a feature of JBuilder Enterprise.

A new Ant option, Use Project JDK When Running Ant, uses the project JDK when running Ant.

Menu Items page

This is a feature of JBuilder Enterprise.

You can now add custom targets to the Project menu.

Web Services page

This is a feature of JBuilder Enterprise.

There's a new Web Services page with the option, Regenerate Deployment. This option determines if JBuilder overwrites the Web Services deployment files.

Building project groups

This is a feature of JBuilder Enterprise.

Subprojects within a project group are built in the order they appear in the project pane. You can change the build order of subprojects within a project group on the Build page of Project Group Properties (Project|Project Group Properties).

In addition, you can customize menu items for building project groups on the Menu Items page of the Build page of Project Group Properties (Project|Project Group Properties).

For more information on project groups, see "Project groups" in the online help.

Resource management

Automatic source discovery and archiving are features of JBuilder SE and JBuilder Enterprise.

If you have file types that aren't recognized by JBuilder, you can associate them with the generic resource file type. Then JBuilder will recognize them and include them in the automatic source discovery process. You'll also be able to bundle them into archives. For example, JBuilder doesn't recognize Flash files, but you might want to access them in your project and deploy them to an archive with the rest of your project. To bundle an unrecognized file type in an archive is a two-step process. First, you need to add it as a recognized file type. Then you need to set it to copy to the output path when the archive is built.

1. To add unrecognized file types to the list of recognized file types for JBuilder, complete these steps:

    A. Choose Tools|IDE Options and choose the File Types tab.
    B. Choose Generic Resource File in the Recognized File Types list.

C. Click the Add button, enter the new file extension, and click OK.

D. Click OK to close the IDE Options dialog box.

Important: By default, any file type added as a Generic Resource File is not included in archives. You need to set the new file type to copy to the output path on the Resource tab of the Build page in Project Properties. Continue on to the next step to do this.

2. To bundle the new file type with your archive, complete these steps:

A. Choose Project|Project Properties and click the Build tab. To bundle the new file type for all future projects, choose Project|Default Project Properties.

B. Choose the Resource tab and select the new file type in the Recognized File Types list. Notice that by default, it's set to Do Not Copy.

C. Choose the Copy radio button to set this file type to copy to the output path.

D. Click OK to close the Project Properties dialog box.

Build system OpenTools API

The build system OTAPI now allows you to listen for the creation and deletion of some build output. See the BuildListener.buildOutputEvent() method, as well as the BuildOutputEvent class and its subclasses. This is also discussed in the "Build Systems Concept" document in Developing OpenTools in the product documentation.

Deployment

Changes have been made to archive deployment, as well as changes to creating executables and executable run configurations.

Excluding dependencies from archives

This feature varies by JBuilder edition.

When creating archives using the Archive Builder and the Native Executable Builder and when creating WAR archives, you can now exclude dependencies from the archive. The Contents page of the archive node, the Classes page of a WebApp node, and the equivalent

step of the Archive Builder™ and the Native Executable Builder, and Specify The Parts Of This Project To Archive have been reworked to add this new feature.

Choose from these combinations for classes and resources:
Classes:

- Specified Only
- Specified And Dependent
- All

Resources:

- Specified Only
- All

For complete control over the classes and resources included in the archive, choose Specified Only and use the Add button to add the classes and resources you want included. If you choose Specified And Dependent, classes that you add with the Add Classes button are added to the archive, as well as any classes on the output path that the added classes depend on.

For example, if you want to include all the project classes and resources in the archive, you would select All for both classes and resources. If you don't want to include any dependencies in your archive and only specified resources, you would choose Specified Only for both classes and resources and then add the classes and resources you want with the Add Classes and Add Files buttons. To add classes and files, classes must be on the project output path and files must be on the project source path.

For EJB modules, classes that are referenced in the bean's deployment descriptor are automatically added to the JAR. Therefore, the Content page of an EJB module's Project Properties dialog appears much like the corresponding page for any other archive, except the word Specified is replaced with the word Required in each instance. You can use this page to exclude dependencies on classes you list as required. Classes referenced in EJB deployment descriptors are not affected.

Archive Builder™

This is a feature of JBuilder SE and JBuilder Enterprise.

The Archive Builder has a new archive type, Executable JAR. Select this type when you want to make an existing JAR into an executable. You can also set a runtime configuration for the executable.

The Contents page of the archive node and the equivalent step of the Archive Builder and the Native Executable Builder (called Specify The Parts Of This Project To Archive) has been reworked to allow you to exclude dependencies. See "Excluding dependencies from archives" in the product documentation

If you choose <Auto Select> as the configuration on the page called Selecting A Method For Determining The Application's Main Class, the project runtime configuration marked as default is used to run the application, including any runtime parameters. If there isn't a default or the default isn't an Application runtime configuration, the first Application runtime configuration in the project runtime configuration list is used. See Run|Configurations for a list of runtime configurations for the project.

Creating native executables

This is a feature of JBuilder SE and JBuilder Enterprise.

You can now create only executable files with the Archive Builder if you select the Native Executable or Executable JAR (for existing JARs) archive type. Steps for creating executables and adding run configurations have been removed from the Application archive type. As before, you can also use the Native Executable Builder to create executables.

If you choose <Auto Select> as the configuration on the page Selecting A Method For Determining The Application's Main Class, the project runtime configuration marked as default is used is used to run the exeuctable, including any runtime parameters. If there isn't a default or the default isn't an Application runtime configuration, the first Application runtime

configuration in the project runtime configuration list is used. See Run|Configurations for a list of runtime configurations for the project.

Configuration files for native executables

This is a feature of JBuilder SE and JBuilder Enterprise.

When you're creating executables with the Native Executable Builder or the Archive Builder, you can now create or overwrite the configuration file that specifies the runtime configuration settings for the executable. These configuration settings are available on the last page of the Native Executable Builder and the Archive Builder. Choose from the following options:

- Create executable configuration.
- Create executable configuration and save a copy in the specified file.
- Override the executable configuration with the specified file.

Note: If you choose to save a copy or override the configuration, the configuration file is added to the project.

You can modify these configuration settings after creating the executable. Right-click the native executable or archive node in the project pane and choose Properties. Then choose the Runtime page, make your changes, and rebuild the archive node.

For information on creating configuration files, see "Creating configuration files for native executables" in "Building Applications with JBuilder" in the online Help.

Productivity enhancements

JBuilder has new productivity features.

Free-floating message pane

You can now undock the message pane so it becomes a free-floating window that you can position on your screen as you like.

Status bar messages

You can determine how long you want messages on the status bar to remain on the status bar before they clear using the Tools|IDE Options|Browser|Status Message Timeout option.

Search using regular expressions

The Find/Replace and Find In Path/Replace In Path dialog boxes available from the Search menu now allow you to search using regular expressions.

Editor

Most of the productivity enhancements focus on the editor.

Drag and drop

You can drag and drop selected text in the editor. Highlight the desired text then drag it with the mouse to the new location. If you are moving an entire line of text, and the Smart Paste option is selected in the Editor Options, the line will be automatically indented for you when you drop it on the new line.

Line numbering

The editor now displays line numbers in the left margin. This feature can be turned on and off with the Line Numbering option on the Editor page of the Editor Options dialog box (Tools|Editor Options). To quickly turn off the line numbering, right-click in the line number margin and uncheck Show Line Numbers.

Text selection

There are shortcuts in the editor to selecting blocks of text or lines. Using the line numbering display in the gutter margin, you can select an entire line, a block of lines, or all the lines.

Code formatting

Some of these formatting options are only available in JBuilder Enterprise.

You now have the ability to specify your formatting preferences and automatically format your source code. To access these settings, choose Project|Project Properties and click the Formatting tab. The tabs on this page allow you to customize your code formatting by specifying the following types of preferences:

- Indenting.
- Tab size.
- End of line characters.
- Multi-line parameters.
- Continuation indent.
- Braces.
- Spaces.
- Blank lines.
- Event handling.
- Import statements and their order.

On each page is a Preview window that displays what the selected preference will look like.

To apply the formatting to your code, open a file in the editor and choose Edit|Format, or press the Tab key.

Note: SmartDiff™ is a new feature to filter out code formatting changes when comparing files. When SmartDiff is enabled, format-related changes, such as the location and number of blank lines, the amount of indentation, or the use of spaces, are ignored when you use JBuilder to compare two versions of a file. The SmartDiff button for enabling or disabling SmartDiff is located on the Diff page in the history view, and on the Diff View of the Compare Files dialog box. SmartDiff is a feature of JBuilder SE and JBuilder Enterprise.

You can also export your code formatting preferences, or import previously saved preferences using the Import and Export buttons at the bottom of the Formatting tab in the Project Properties dialog box.

Zoom tool

There is a magnifying glass icon at the bottom of the editor that lets you instantly adjust the size of your text. Click the icon to zoom in and enlarge the text, or click the down-arrow

beside the icon and choose Zoom In. To zoom out and decrease the size of the text, click the down-arrow beside the icon and choose Zoom Out. To return to the default text size, click the down-arrow and choose Normal.

Quick keymap changer

You can quickly change the keymap used by editor with the keymap selection tool found on the editor status bar. Clicking the keymap name text (such as CUA® or Macintosh®) on the status bar returns the keymap to the last selected keymap, thereby allowing two users to conveniently share the keyboard. Click the arrow to display a list of keymaps so you can select from among all available keymaps.

Team Development enhancements

Numerous changes have been made to improve the integrations of supported version control systems.

Merge conflict resolution

This is a feature of JBuilder SE and JBuilder Enterprise.

A Merge Conflicts page has been added to the history pane. The Merge Conflicts page lets you view and resolve conflicts between a file in your workspace and the corresponding file in the version control repository. The Merge Conflicts page displays side-by-side views of the workspace source and the repository source for the file, with the conflicting changes highlighted. Radio buttons to the left of the conflicting changes are used to choose which changes to keep, and a preview pane at the bottom of the Merge Conflicts page shows how the file will look when you apply the selected changes. The Merge Conflicts page supports merge conflict resolution for Java files, as well as text files, such as XML, HTML, and EBJ group files.

Note: Merge conflict resolution within JBuilder is only supported in CVS (CVS integration is a feature of JBuilder SE and JBuilder Enterprise) and Microsoft® Visual SourceSafe® integrations (VisualSource Safe integration is a feature of JBuilder Enterprise).

For more information about the Merge Conflict page, see "CVS in JBuilder" and "Visual SourceSafe in JBuilder" in "Team Development Using JBuilder" in the product documentation.

CVS enhancements

CVS integration is a feature of JBuilder SE and JBuilder Enterprise.

There are two key improvements to the CVS integration:

- You can use a port number other than the default port number when using the pserver connection protocol.
- You can scan the repository for a list of modules or branches in the Pull Project From CVS wizard.

Note: Because of this added ability to scan for modules and branches in the CVS repository, CVS Helper support has been removed from JBuilder.

For information on using the CVS enhancements, see "CVS in JBuilder" in "Team Development Using JBuilder" in the product documentation.

ClearCase® enhancements

Rational® ClearCase® integration is a feature of JBuilder Enterprise.

Many changes have been made to improve the integration of ClearCase for version control and configuration management:

Status Browser

The new Status Browser (Team|Status Browser) is primarily a viewing tool: it browses the active project, and it displays the version control status of each changed file, the source for each available version, and differences between any two versions of a changed file.

Commit Browser

The new Commit Browser (Team|Commit Browser) provides the viewing capabilities of the Status Browser, and provides access to common version control operations for files that have changed. With the Commit Browser, you can set the version control command you want to apply to each changed file, enter comments for individual files and for the whole group, then execute all of the commands with one click.

Snapshot and dynamic view support

ClearCase integration in JBuilder supports the use of both dynamic and snapshot views. Snapshot and dynamic views are created with the ClearCase Create View wizard, which can be accessed from with JBuilder (Team|ClearCase Tools|Create View). The Project For ClearCase wizard opens or creates a JBuilder project for an existing ClearCase view. The wizard mounts a VOB and starts a dynamic view or snapshot view to access file and directory elements in the VOB. This wizard does a checkout of the files in the project if they are not already checked out.

When working with a snapshot view, you can perform a ClearCase Hijack (Team|Hijack File) to take files out of ClearCase control. This feature allows you to work on files when the ClearCase server is unavailable. Hijacked files can be converted back into checked out files using the ClearCase Update View wizard (Team|ClearCase Tools|Update View).

Console output

The ClearCase Properties dialog box provides a Show Console Messages checkbox to enable console output. When Show Console Messages is checked, all ClearCase commands and their generated output are logged in the message pane, in a ClearCase Console page.

Support for reserved and unreserved checkouts

ClearCase integration in JBuilder supports reserved and unreserved checkouts. The ClearCase Checkout dialog box lets you specify whether to checkout a reserved version of the selected files, an unreserved version, or to checkout a reserved version unless some other view has a reserved checkout. If you check out a reserved version of a file, that means you have

exclusive rights to check in the file. If you checkout an unreserved version of a file, you may need to merge changes at checkin time if another user has the same file checked out.

Improved access to ClearCase® tools and tool commands

The ClearCase Tools command (Team|ClearCase Tools) opens a menu of commands to launch ClearCase tools, such as the ClearCase® Explorer (Team|ClearCase Tools|Explorer). The following commands or tools can be accessed from the ClearCase Tools menu:

- Explorer
- User Options
- Merge Manager
- Find Checkouts
- Apply Label
- Create VOB
- Create View
- Update View
- Checkin "<filename>"
- Checkout "<filename>"
- Undo Checkout "<filename>"
- Add File "<filename>"
- Properties For "<filename>"
- History For "<filename>"
- Compare With Prior Version "<filename>"
- Version Tree For "<filename>"
- Project Explorer
- Join UCM Project
- Deliver From Stream
- Rebase Stream

Unified Change Management (UCM) support

ClearCase integration in JBuilder supports the use of the UCM process with ClearCase, or you can use the base ClearCase. UCM is an optional, out-of-the-box process for organizing software development teams and their work products. Using the base ClearCase, you have the flexibility to implement a configuration management solution of your choice.

For more information about ClearCase integration in JBuilder, see "Rational ClearCase in JBuilder" in "Team Development Using JBuilder" in the product documentation.

Supported J2EE™ servers

J2EE server support is a feature of JBuilder Enterprise.

Sybase® EAServer support

JBuilder now has support for the Sybase® Enterprise Application Server 4.1.

BEA WebLogic Server™ support

JBuilder has improved support for BEA WebLogic Server™ 7.x, including optimistic concurrency for enterprise beans with container-managed persistence.

Now WebLogic Servers 6.x and 7.x each have their own pages for configuring the server in the JBuilder IDE.

Borland® Enterprise Server support

There are incremental improvements in the support for the Borland® Enterprise Server 5.02 and 5.1.x servers.

Oracle9i™ Application Server support

Plug-ins to support the Oracle9i™ Application Server are provided on your JBuilder 8 product CD in the Oracle9i AS directory. Please see the README file in that directory for instructions on installing and configuring these plug-ins. Please note that these plug-ins are supported by Oracle.® Contact Oracle Technical Support if you have any problems or issues regarding these plug-ins.

Javadoc enhancements

This is a feature of JBuilder SE and JBuilder Enterprise.

JBuilder 8 includes a package file editor that allows you to create, edit, or remove the package.html file for individual packages in your project. For more information, see "Package-level files" in "Creating Javadoc from API source files" in "Building Applications with JBuilder" in the product documentation.

Refactoring enhancements

This is a feature of JBuilder SE and JBuilder Enterprise.

The following refactoring commands are now available from the structure pane:

- Find References
- Optimize Imports
- Rename Package
- Rename Class
- Rename Method
- Rename Field
- Change Method Parameters

Accessibility features

New accessibility features make every part of JBuilder accessible via the keyboard. Assistive devices and software use keyboard-accessible commands.

Standard accessibility strategies are respected in JBuilder. Use the Shift and Ctrl keys for multiple selection. Use Enter to select items from within the IDE. Use Enter or the Space bar to choose items from a menu. Use Tab or Ctrl+Tab to navigate from place to place within the IDE.

IDE accessibility

Switch between views, such as Source, Design, or History, from the Window menu.

Select an IDE splitter with the Select Browser Splitter command, so you can resize the content, project, or structure panes without using a mouse.

The View menu now contains a Context Menu command. This command displays the context menu, if any, associated with the current location of the cursor focus. For example, if the cursor is in the editor and you choose View|Context Menu, the editor context menu will be displayed. The keyboard shortcut is Shift+F10.

Float and dock the message pane using the mapped keyboard shortcuts. Consult the keymapping for your editor emulation for the correct keystrokes.

You can change the font size used by the IDE with Tools|IDE Options|Browser|Font Adjustment option.

Designer accessibility

Add components to your design using the Edit menu. Choose Edit|Add Component to open the Add Component dialog box. Cut and paste to move the component to where you want it in the component tree.

Use Tab or Ctrl+Tab to navigate in the IDE when using the Designer. Focus moves in the following order:

- Project pane
- Component tree
- Component palette
- Design surface
- Inspector

Generally, you can use arrow keys inside an area to navigate within the area.

New FileI/O API

There is a new FileIO property that can be used to improve the performance of write operations to Borland JDataStore log files. This property can be set in two ways:

1. Through the extended JDBC property "fileio," which can be set in the JBuilder UI designer as part of a database's connection property.
2. Through the "JDATASTORE_FILEIO" Java system property, which can be specified at the Java command line using a "-D" command line option when running JDataStore stand-alone.

Settings made through the extended JDBC property takes precedence over the JDATASTORE_FILEIO system property setting.

This property can be set to the following values:

- default

  Uses the default FileI/O. This causes JDataStore to use the Java FileI/O APIs that it has in prior releases. This is the most portable option across all Java platforms and provides the most complete log file.

- new

  This causes JDataStore to use the Java "new I/O" APIs surfaced in JDK 1.4. Note that this is not used by default on the Microsoft® Win32® platform due to known bugs in the Sun® JVM for these platforms prior to and including JDK 1.4.1. However, we know of no such problems on non-Win32 platforms. This gives better performance than "default" but is similar to the JDataStore soft commit option.

- native

  This uses native APIs to provide the same benefit as "new." Currently, it is only implemented for the Win32 platform. Other platforms should just use the "new" setting if they are running under JDK 1.4 or above.

For more information, see the Javadoc for com/borland/datastore/cons/FileIO.java and com/borland/datastore/jdbc/cons/ExtendedProperties.java.

Conclusion

JBuilder 8 offers a wealth of capabilities that provide unparalleled cross-platform freedom and flexibility:

- Deliver Web applications faster with a new Web development framework based on proven standards.
- Make large teams more efficient with enterprise-level productivity enhancements including project groups, code formatting, and tighter-than-ever integration with leading source control management systems.
- Get applications to market quicker using agile development methodologies supported by new unit testing for J2EE, Web applications and EJB.
- Increase productivity and code reliability with an enhanced editor, HotSwap debugging, and improved performance.

Find out more about JBuilder at http://www.borland.com/jbuilder.