Perspectives and Projects and Views, Oh My!

A Survival Guide for New Eclipse Users

by Jerry Karasz

If you are a BBj© developer trying to use Eclipse and the Business BASIC Development Toolkit (BDT) plug-ins to create and maintain your software programs, you have probably noticed that Eclipse has its own way of doing things. And it tries very hard to make sure that you do everything its way - and that you like it.

For example, you cannot run Eclipse without opening a workspace - and what is a workspace, anyway? Then there is something called a perspective that decides whether you can work with BDT and BBj to edit and run code. Of course, the BDT perspective contains a number of open views. And each view helps you accomplish some aspect of software development. But why are they grouped the way they are? And how do you open or close them to get Eclipse to show you what you want? Last but not least, what do you need a BBj project for, anyway? Why can't you just edit your code and get on with it?

This article provides an overview for developers like you who want to customize what you see and how you work in the BDT Interactive Development Environment (IDE) that is available as a set of plug-ins for Eclipse. It provides tips and ideas for managing your Eclipse display so that you can be the most productive BBj developer possible.

The most advanced combination of Eclipse/BDT available at the time this article was written was BDT 18.10 (released in conjunction with BBj 18.10), which requires Eclipse Oxygen (version 4.7) to run. This document assumes that you have already installed Eclipse Oxygen and the BDT plug-ins at this time - if not, see BASIS' Eclipse Plug-ins page (http://www.basis.com/eclipseplug-ins) for detailed instructions on doing so.

In our look at Eclipse's model, let's start at the "top" - with workspaces.

What is a Workspace, Anyway?

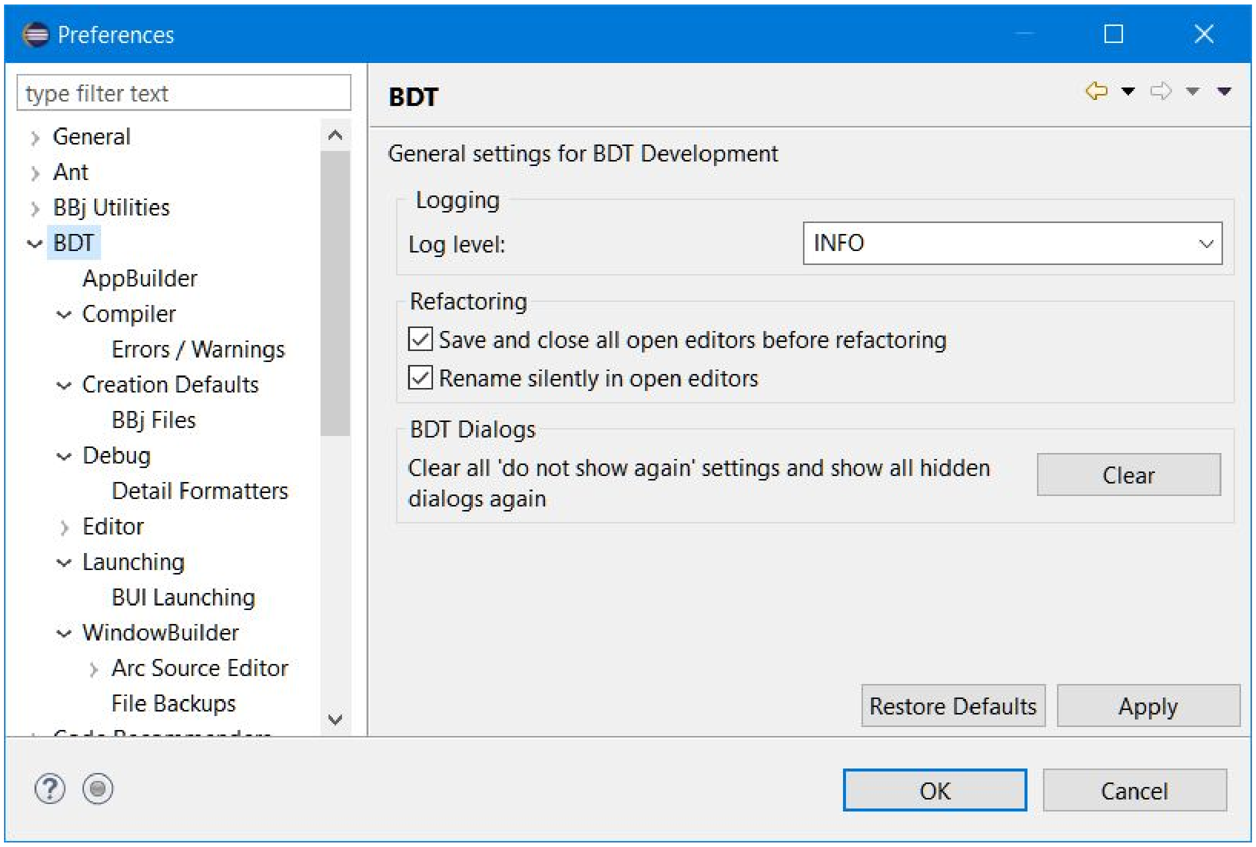
Eclipse always has a home - the place where you installed it. But it also needs a working location where it can keep everything that it needs to run. Some examples of what it needs to keep are:

1. Housekeeping details such as user preferences and log files.
2. The collection of source code that you are currently working on, to help you manage and build that code. This includes the definition of what is included when you ask to do a "build all", how to synchronize your files with a configuration management archive (SVN, Git, etc.), and so on.
3. Isolated groups of source code that have some relationship with each other but are not necessarily directly tied together to create a single program.
4. Settings to help you shift gears more productively when changing to a different collection of products, especially if you are part of a team working on the same source. Each workspace can be named to help you be more organized. It holds the configuration for the installed plug-ins, perspectives, views, and even for your configuration management systems (allowing these to vary by workspace). Even if you are not using configuration management, the workspace boundaries still allow you to organize and track the folders which need to be backed up or replicated to avoid losing work.

Eclipse allows you to maintain multiple workspaces, each of which can have different user preferences, projects, BBj programs, and so on. For example, if you work on more than one customer's software at a time, it is probably to your advantage to keep those different customers' products and intellectual property separate from each other by using a different workspace for each customer (isolation).

Configuring a Workspace

Eclipse maintains a set of preferences that are unique to you (the user) for each workspace. When you are using a workspace, open Eclipse's Preferences window to configure your settings for Eclipse itself as well as for each installed plug-in that offers preferences. For example, see **Figure 1** below for the BDT Preferences display:



BDT

General settings for BDT Development

Logging

Log level: INFO

Refactoring

☑ Save and close all open editors before refactoring

☑ Rename silently in open editors

BDT Dialogs

Clear all 'do not show again' settings and show all hidden dialogs again

Figure 1. Eclipse's BDT Preferences

In general, preferences changed here apply to all of the BBj project or source files in this workspace. One exception to this rule of thumb is the Creation Defaults, which apply only at the time a resource is being created. For any resource, though, once it exists you can modify its properties (such as the BBj project properties) and override the preference values (which essentially represent 'default values' for many properties).

NOTE: Remember, changing any preference value here will only apply in the open workspace - other workspaces will not be affected.

Eclipse lets you create as many workspaces as you like. Each must have a unique name and location on disk, but otherwise, they are unrelated. Eclipse's File menu offers an option to **Switch Workspace**, essentially closing whatever workspace is open and then opening the selected one. This menu option also allows you to create new workspaces, by selecting the **Other…** menu item under **Switch Workspace**. Simply specify a location that does not yet exist, and Eclipse will create a new workspace there with a name that matches the newly-created folder's name.

Tips: Be intentional about where you create your workspaces.

- Do not create workspaces in “temp” areas that can be deleted by the operating system, or in areas where you may forget that they exist. Removing or emptying a workspace’s “root folder” via the operating system will empty the workspace, and result in a complete loss of your files.
- Organize your workspaces as you create them rather than scattering them around.
- Do not create workspaces on mapped or network drives, as this will degrade your performance when editing, using code completion, building, executing, and debugging.

What is a Perspective and Why Do I Need One?

Eclipse organizes its user interface based on the perspective you select. Consider opening a perspective to be a way of telling Eclipse, “I want to work on this kind of source code now”. For example, to work on Java source code (and get the full JDT support), you need to select the Java perspective; to work on BBj source code (and get the full BDT support), you need to select the BDT perspective; to synchronize your source code with your version control system source, such as SVN or Git, select the Team Synchronizing perspective; and so on for the Debug and other perspectives.

Eclipse remembers how you have it configured in each perspective, and will attempt to change its layout (which panes or views are open, where they are docked, etc.) accordingly. Use this to your advantage - for example, in the BDT perspective, once you have laid out the views that you like to use to develop your BBj code, the next time you open that perspective that layout will again be shown. This can be useful, for example, if you have Java projects and source files in the same workspace as your BBj project and BBj source files; switch to the Java perspective when working on your Java projects/files, and switch back to the BDT perspective to work on your BBj programs.

Tips: Here are some tips on getting the most out of your perspectives:

- Eclipse’s **Window** menu and toolbar let you switch perspectives easily.
- Eclipse’s user preferences let you set configuration options for the set of perspectives (such as which is the ‘default’).
- You can customize each perspective by right-clicking its toolbar button in Eclipse’s Perspectives toolbar.
- To use the features of the BDT IDE to organize, edit, execute, and manage your BBj source code, you must use the BDT perspective. Do not use other perspectives (such as the Java perspective) when you develop your BBj source code, as then you will not be able to use the full BDT functionality.

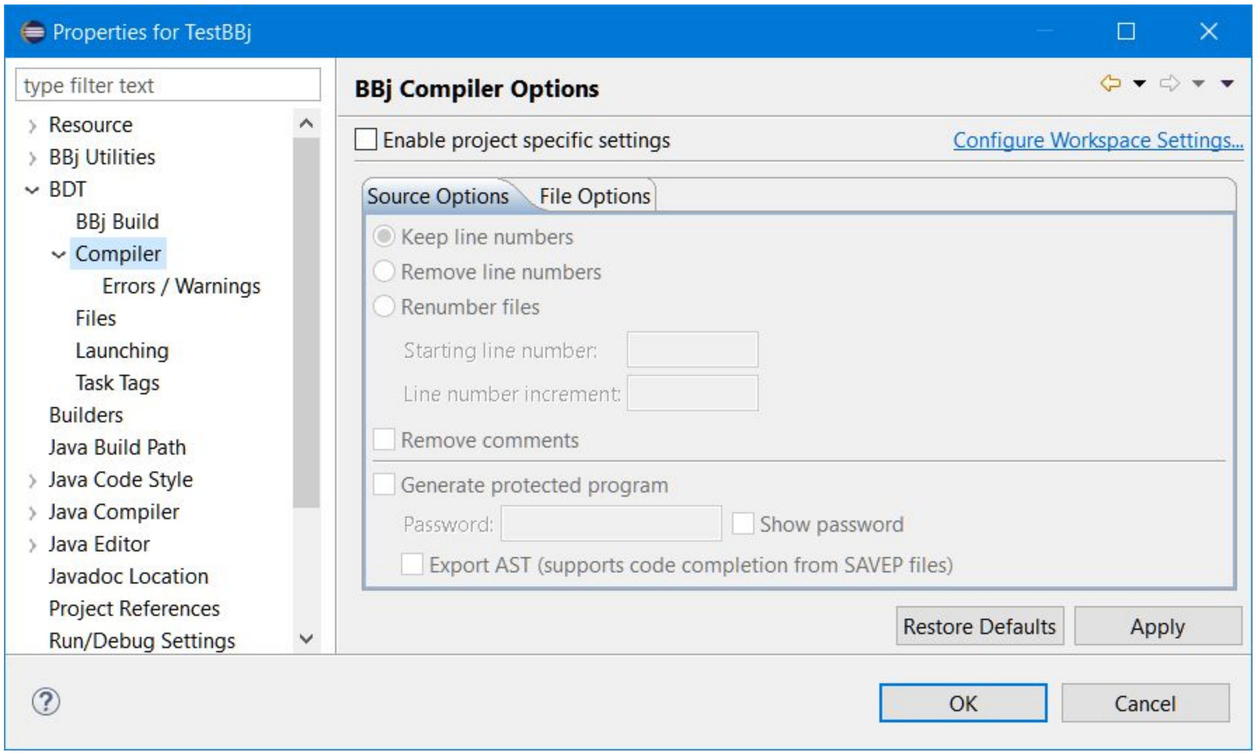
What is a BBj Project?

Each workspace consists of zero or more projects, where each project holds a collection of resources (source code files, images, data files, and so on). Eclipse requires that all source code be part of a project to provide editing assistance (such as code completion, problem markers, and so on). The project is also the unit for building/compiling sets of source files, so it functions as a natural grouping construct for related source files.

With BDT and BBj, there is a special kind of project to hold your BBj source files and other resources: a BBj project. Do not use another project type (such as Java projects) to hold BBj source code, as that code will not be able to use BDT functionality. This document assumes that you know how to create a BBj project - if not, see BASIS' *Creating Your First BBj Project* page (http://documentation.basis.com/BASISHelp/WebHelp/eclipse-bdt/getting_started/creating_your_first_bbj_project.htm) for detailed instructions on doing so.

Configuring Project Properties

One advantage of using a BBj project is the set of properties that are available to configure how BDT interacts with the resources and files in that project. Eclipse maintains a set of properties that are unique to each project. When you are using a view such as the BDT Explorer (more on this in the next section), you can open the Project Properties display by right-clicking the project and selecting **Properties** (see **Figure 2** below for the BBj Project Properties display):


Properties for TestBBj

type filter text

- Resource
- BBj Utilities
- BDT
  - BBj Build
  - Compiler
    - Errors / Warnings
  - Files
  - Launching
  - Task Tags
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project References
- Run/Debug Settings

BBj Compiler Options

☐ Enable project specific settings   Configure Workspace Settings...

Source Options | File Options

● Keep line numbers
○ Remove line numbers
○ Renumber files
Starting line number:
Line number increment:
☐ Remove comments
☐ Generate protected program
Password: ☐ Show password
☐ Export AST (supports code completion from SAVEP files)

Figure 2. BBj Project Properties

Project Properties changed here apply to only the selected project (notice the title text in **Figure 2** which shows that we are viewing the properties for the project 'TestBBj'). Some BBj project properties are only available through this properties display, while others are offered here as ‘overrides’ to the workspace preferences. The BBj Compiler Options screen in **Figure 2** is an ‘override’ type; by default, all BBj projects follow the workspace preferences for the BBj compiler properties on this screen (thus the properties are grayed out and read-only). To override those workspace preferences, check the **Enable project specific settings** checkbox and edit the values shown, clicking [OK] to save your changes. Once you have done so, changes to the corresponding workspace preferences will have no effect on this BBj project because you have overridden the workspace settings.

NOTE: Remember, changing any project property value here will only apply to the BBj project selected - other projects will not be affected.

The most important BBj project property to understand is the BBj Runtime Environment (BRE), which can be changed from the BDT Explorer by right-clicking a project’s BRE as shown in **Figure 3** and selecting **Select BRE**.

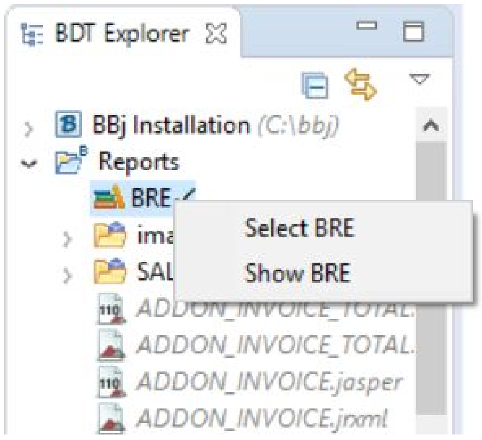


Figure 3. Changing the BRE

A BRE defines the environment in which BDT runs BBj programs. It provides the **`config.bbx`** file and PREFIX BDT will use for code completion and assistance, it sets the BBj language offerings such as verbs and functions, and it lets you specify runtime settings to accompany your BBj programs when you execute them. By default, all new BBj projects have the “Default BRE” selected. To change this, simply select the desired BRE in the **Select Environment** dropdown. To manage the BREs available to all of the BBj projects in this workspace, expand the BDT Explorer’s **BBj Installation** node and right-click the **BBj Runtime Environments** node (as shown in **Figure 4**); this includes creating, editing, and deleting BREs.

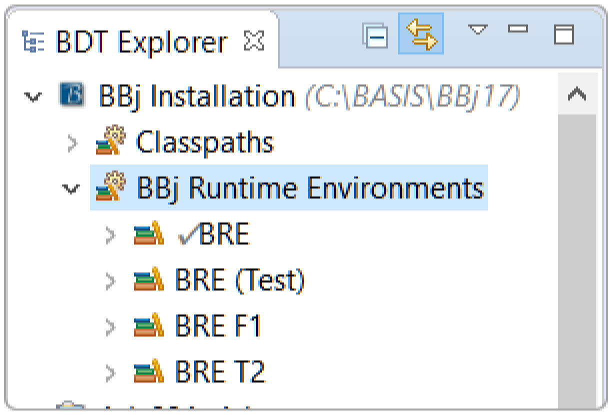


Figure 4. BBj Installation - Managing the BREs

Tips: Here are some tips on getting the most out of your BBj projects:

- If you work on code for one customer for several different purposes or programs, it can be to your advantage to isolate those different sets of source files from each other. Do so by using a different BBj project for each set, even if the two projects are in the same workspace. This is especially true if there is some commonality across the projects such as shared code, or they run in the same environment.
- Use each project's BRE property to control the editing and runtime BBj environment BDT uses.

Be intentional about where you create your BBj projects:

- Do not create BBj projects in "temp" areas that can be deleted by the operating system, or in areas where you may forget that they exist. Removing or emptying a BBj project's "root folder" via the operating system will empty the project, and result in a complete loss of your files.
- Organize your BBj projects as you create them rather than scattering them around.
- Do not create BBj projects on mapped or network drives, as this will degrade your performance when editing, using code completion, building, executing, and debugging.

What Views Do I Need?

Eclipse uses views to display information that spans multiple user actions or resources (such as the list of errors and warnings for the workspace, even if no editor is open). It expects that, if you need a view to do your work, then you will need it whenever you are in a particular perspective. Because of that model, opening any perspective automatically opens the set of views that you had open the last time you used that perspective. And it opens each of those views in the same layout position (docked to the top, and so on) where you had them last time. You might even think of a perspective as nothing more than the collection of views that you see when it is open; this is not true, of course, but the simplification may make it easier to follow our discussion of views below.

Each view window helps you to perform some set of work tasks. Each plug-in installed in Eclipse can add views to the list available to you. Depending on which of the available BDT plug-ins are installed, Eclipse supports some BBj-specific views:

- BDT Plug-in: BDT Explorer - to manage the BBj Installation, BBj Projects, and the resources/files in those projects
- BDT AppBuilder Plug-in: BDT ARC Inspector and BDT Explored Objects - to interact with the event code objects (in .gbf files) for some .arc file

BDT also displays important BBj-specific information in some of the general Eclipse views:

- Problems - to see a list of the compile problems (information, warning, or error) from the most recent workspace build; you can filter this view to show only a specific subset as desired
- Error Log - to see entries Eclipse is adding to its error log, such as exception stack traces
- Outline - to see an outline of whatever content exists in the editor with the focus; selecting a different editor changes the Outline view's display accordingly

Still other Eclipse views are helpful for any general development (including BBj):

- Progress - to see Eclipse's progress on whatever task(s) it is currently performing such as building or saving
- Search - to examine and navigate through the result set from the most recent search action
- Tasks - to examine and navigate through the set of "task tag" markers you have placed in the source code files in this workspace

There is a long list of views that are available from Eclipse and from the installed plug-ins - too many for us to cover here in any detail. Consider exploring the various views for yourself and sharing those you find useful with the BDT community. To see a list of views and open one, use Eclipse's **Show View** window (accessed from the main menu via **Window > Show View > Other…**) as shown in **Figure 5**):

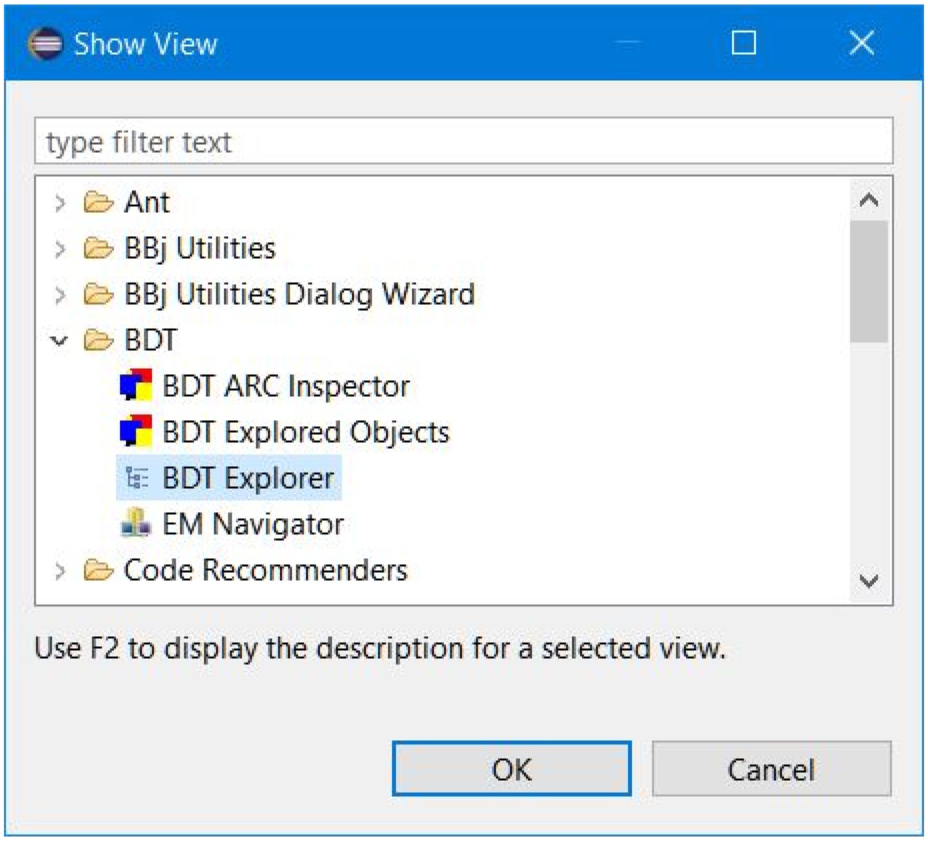


Figure 5. Eclipse's Show View Window

Tips: Here are some tips on getting the most out of your views:

- Editors are not considered views in Eclipse; editors open to display the content of one resource or file and to support editing and saving it. Views, on the other hand, display information that spans multiple user actions or resources (such as the list of errors and warnings for the workspace, even if no editor is open).
- Use the BDT Explorer view only in the BDT perspective. The BDT Explorer relies upon the BDT Perspective for much of its internal administration and for its access to the BBj installation.
- Eclipse will automatically open the BDT Explorer view for you the first time the BDT Perspective is opened.
- To re-open the BDT Explorer once you have closed it, use the **Show View** window mentioned above.

Where Do I Go From Here?

As you have seen, the Eclipse IDE defines a structure for developing software that is designed to support extensive customization by you, the user, and that provides you with help laying out and maintaining your BBj projects and their resources. We have presented only a few of the basic Eclipse constructs here - mainly those that you need to understand in order to use the

BDT IDE effectively. Following the Eclipse model as you set up your workspaces will let you be more productive in your BBj development, and will help you avoid some of the confusion that comes with adopting a new IDE. There are too many details to cover them all in this article, but for more details and to learn more about how BDT works within this Eclipse model, here are some additional resources you may find helpful:

- For helpful discussions related to the BDT IDE, use the ide-user-group Google group. BASIS engineers also participate in these discussions.
- The IDE User Group wiki site is also available as a developer community resource for sharing documents, links, source code, examples, images, or other artifacts that are useful with either IDE. Occasionally BASIS also adds links or other information to the wiki site that are helpful. To learn about accessing and using ide-user-group resources, see BASIS' Discussion Forums page.
- For independent discussions of Eclipse's structures, you may wish to examine these articles:
    - http://www.tutorialspoint.com/eclipse/eclipse_perspectives.htm
    - https://eclipse.org/articles/using-perspectives/PerspectiveArticle.html
    - Perspectives in Eclipse Oxygen: https://help.eclipse.org/oxygen/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Fconcepts%2Fconcepts-4.htm&cp=0_2_2
    - Views in Eclipse Oxygen: https://help.eclipse.org/oxygen/topic/org.eclipse.platform.doc.user/concepts/concepts-5.htm?cp=0_2_4

As you have seen, understanding the Eclipse model is important to using BDT and to developing your BBj programs. And once you are working with BDT, maintaining your BBj programs will be easy and efficient. Give Eclipse and the BDT IDE a try and see for yourself.