Have it Your Way With New BDT Preferences and Properties

Like most good software applications, the Eclipse platform supports preferences or optional settings that allow you to configure how your Eclipse workspace operates in general, as well as how your plug-ins perform. These preferences are persisted in your workspace between restarts.

The latest Business BASIC Development Tools (BDT) Eclipse CodeEditor plug-in provides a number of advanced workspace preferences and project properties settings. These options give you much finer control over how you create and manage BBj® projects, putting you in control of your development experience. In this article, we will explore BDT's new preferences in a hands-on tutorial, after which BDT's look and feel will meet your own personal needs. Download the BDT plug-in that comes with BBj 14.20 or higher to try it out. Then, you can have your burger and BDT "your way"!

To follow along with the rest of this article, be sure you installed both Eclipse and the Eclipse BDT plug-in per the online instructions in *Preparing Eclipse for BASIS-Provided Plug-ins* (links.basis.com/preparingeclipse).

By Kevin Hagel
Software Developer

Next, access the Eclipse preferences as instructed below. However you decide to access the preferences in your Eclipse environment, we will refer to it as Preferences > in this article.

- On Linux and Windows, go to Windows > Preferences.. menu.
- On Mac, go to Eclipse > Preferences... menu or press the Command key and the comma as shown in **Figure 1**.

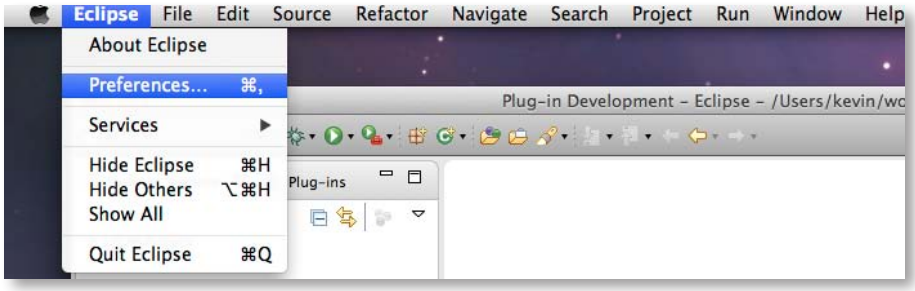

Figure 1. Accessing Preferences on the Mac

Preference settings appear listed in categories, in the left navigation pane of the Preferences window. Like other tree structures, clicking on the triangle to the left of a category expands it, and clicking on a subcategory displays its available items. Let's take a look at a few 15.0 preferences that are available in a preview release.

BDT

Click 'BDT' in the left navigation tree to display the general setting in the 'BDT Preferences' dialog as shown in **Figure 2**.

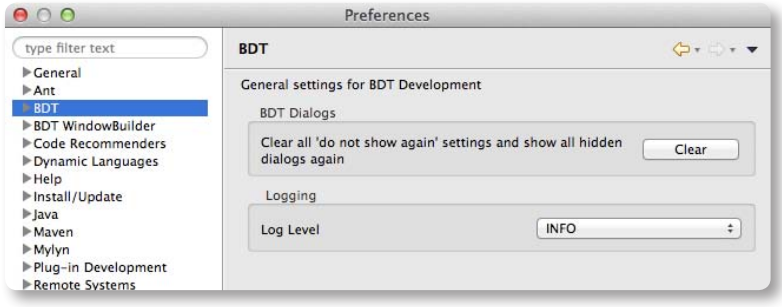

Figure 2. BDT General Settings dialog

BDT Dialogs

During your development, you may occasionally see a dialog popup warning you that something has (or hasn't) happened. Many of these dialogs include an optional checkbox 'Don't show this dialog again', which you should mark if these dialogs become annoying or you no longer need to see them. If, however, at some point in the future you decide you want those dialogs displayed next time they trigger, navigate to this window and click the [Clear] button in the 'BDT Dialogs' preference.

Logging

BDT writes error and informational messages to a **`bdt.log`** file (located in **`<your workspace directory>/.metadata/.plugins/com.basis.bdt.eclipse.core/bdt.log`**). If you contact BASIS Technical Support with a problem, they may ask you to set a particular Log Level value here so that specific information will appear in the log file, and then ask you to send the **`bdt.log`** file to help the engineers investigate your problem. The default 'Log Level' is **`INFO`** and you should leave it at this setting unless a BASIS tech support rep or engineer asks you to change it.

Creation Defaults

Expand BDT and click 'Creation Defaults' to display the options in **Figure 3**.

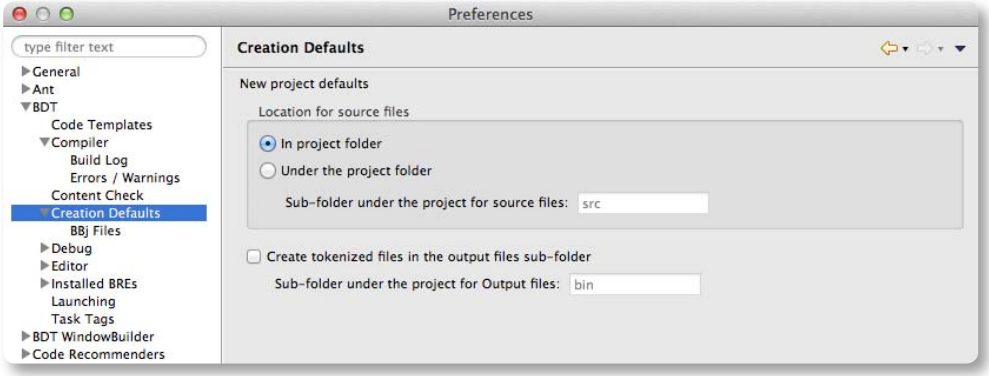

Figure 3. New project default options

Source and Output File Locations

The 'New project defaults' preferences allow you to set how BDT creates and configures new projects; these settings will not affect any existing projects.

- Location for source files
  - In project folder - By default, new projects maintain your source files in the project root folder as opposed to a subfolder.
  - Under the project folder - Choose to keep your source files in a subfolder of the project root folder by clicking 'Under the project folder' and entering a name for the sub-folder.
- Create tokenized files... - By default, BDT does not tokenize the files in your project. To have new projects' programs compiled into tokenized files, check 'Create tokenized files in the output files sub-folder'. You must also enter a name, such as **`tok`**, for the subfolder under the project root folder that will hold the tokenized files. Once you have done this, new BBj projects will create tokenized versions of their source files in a **`tok`** folder under your project root each time you build them. To skip creating tokenized output in new BBj projects, unmark the 'Create tokenized files...' checkbox.

Creation Defaults/BBj Files

Click BDT > Creation Defaults > BBj Files in the left navigation tree to display the New BBj file defaults.

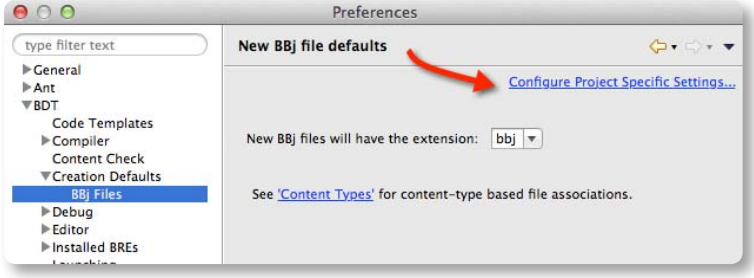

Figure 4. Accessing project-specific settings for the BBj Files

Configure Project Specific Settings

Whenever the [Configure Project Specific Settings...] appears in a 'Preferences' dialog (as it does in the BBj Files display in **Figure 4**), it means that you can override the displayed set of preferences on a per-project basis.

Simply click the link and select the desired project from the list (see **Figure 5**).

Click [OK] to proceed to a filtered Properties window and set any preferences that you wish to be unique to that project (**see Figure 6**).

Since project specific preferences are optional, Eclipse requires you to mark the 'Enable project specific settings' checkbox in order to make changes. If you have set project specific settings for this project before, the box should already be marked; if not, mark it now. Once there are any preferences that are specific to this project, BDT creates a **`.settings`** folder in that project's folder, and places all of the settings in a **`.prefs`** file there. Normally, the **`.settings`** folder will not appear in the BDT Navigator view because it has no name preceding the extension ('settings' is seen as an extension).

You may notice that only the preferences that displayed when you clicked the link (in this case, the BBj Files shown in **Figure 4**) are available for you to set project specific preferences. In fact, that is why the title of the window includes the text '(Filtered)'.

To view or set all of the project specific settings at one time, close all of the 'Preferences' windows and return to the 'BDT Navigator' or 'Navigator' view. Right-click the project you want to view and select 'Properties' in the menu that appears. This properties display is unfiltered (notice there is no text '(Filtered)' in the window title) and allows you to view or edit all of the project-specific preferences, even those that are not part of BDT (as shown in **Figure 7**).
If you checked out the Project Properties display, navigate back to Preferences > BDT > Creation Defaults > BBj Files.

New BBj Files Extension
The file extension dropdown (see **Figure 8**) allows you to set the default file extension for new BBj files that you create. For information on creating new files, refer to the "Create a program source file in the project" section in *Creating Your First BBj Project* at links.basis.com/creatingbbjproj.

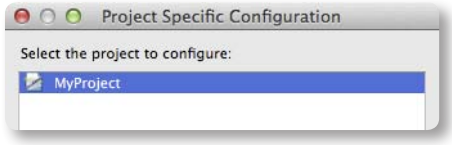

Figure 5. Selecting the project to configure

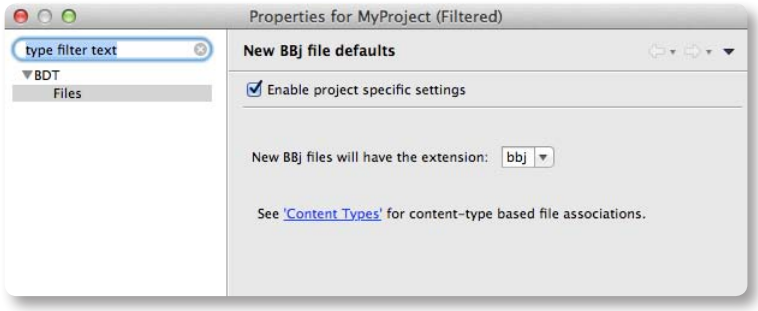

Figure 6. Filtered project-specific BBj File options

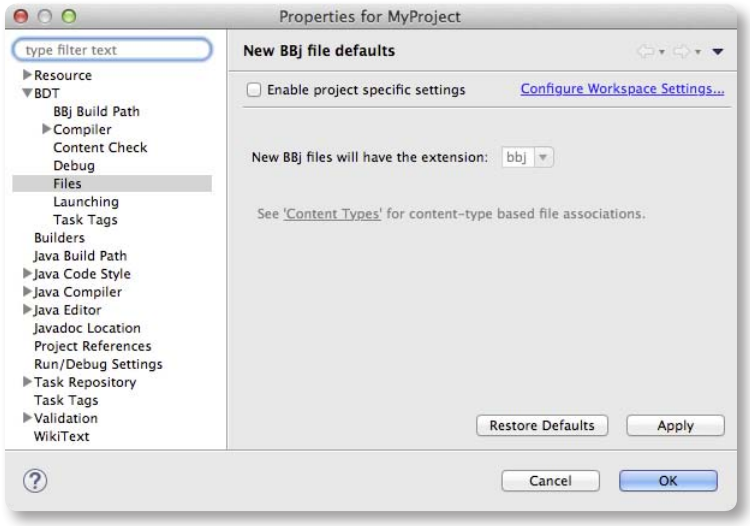

Figure 7. Project properties offering all project-specific preferences

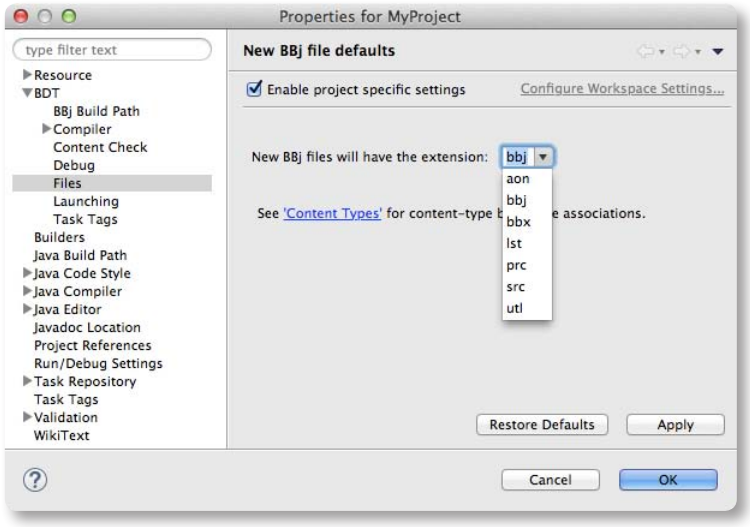

Figure 8. The list of recognized BBj source file extensions

To choose an extension other than the default .bbj, click on your preferred selection in the dropdown list or simply type in another extension. If you type in a new extension, you must also register the extension as a BBj content type (see below).

Content Types

In order for BDT to recognize files with your new extension as BBj source files, you must register the extension as a 'BDT Content Type'. Click the Content Types link to jump to the Preferences > General > Appearance > Content Types display. The dialog displays a 'Content types' navigation tree; Select Text > BBj Content Type in that tree as shown in **Figure 9**.

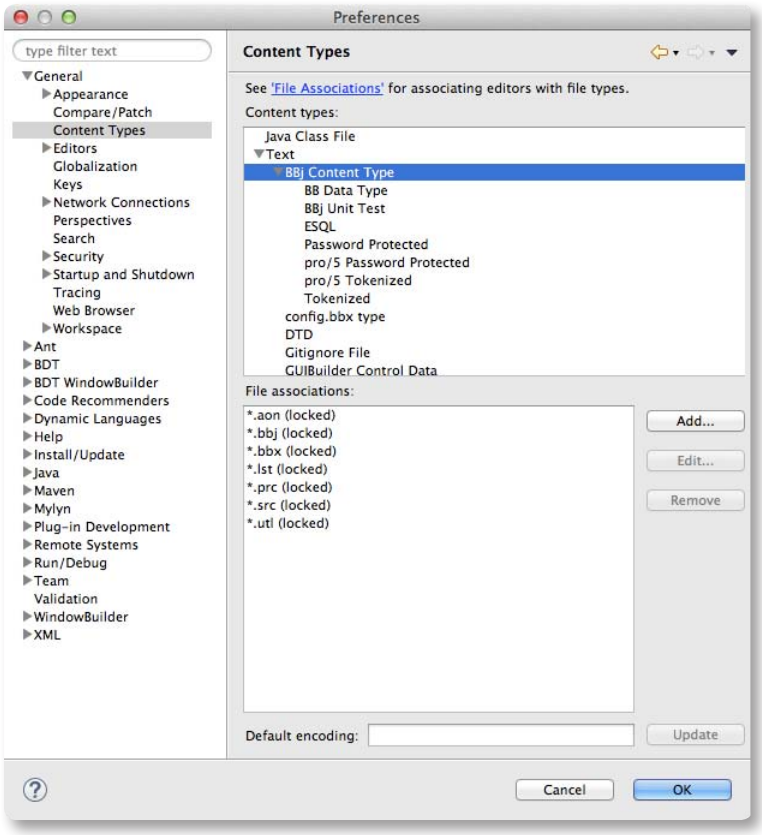

Figure 9. The files or extensions associated with the BBj Content Type

Click [Add...] to open a new dialog (**Figure 10**).

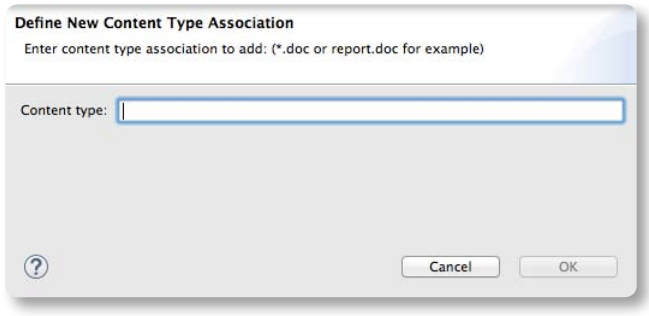

Figure 10. Associating a new extension with a BBj content type

Type in a file extension such as

```
*.ext
```

Click [OK] and verify that your new extension now appears in the list of 'File Associations' for the 'BBj Content Type'. Your new extension is now registered as a BBj source file extension, and BDT's CodeEditor will now support editing, debugging, and running files with that extension as BBj programs.

To continue on, navigate to Preferences > BDT > Compiler.

BBj Compiler Options

Click on [Output Folders] at the bottom of the display to expand it as shown in **Figure 11**.

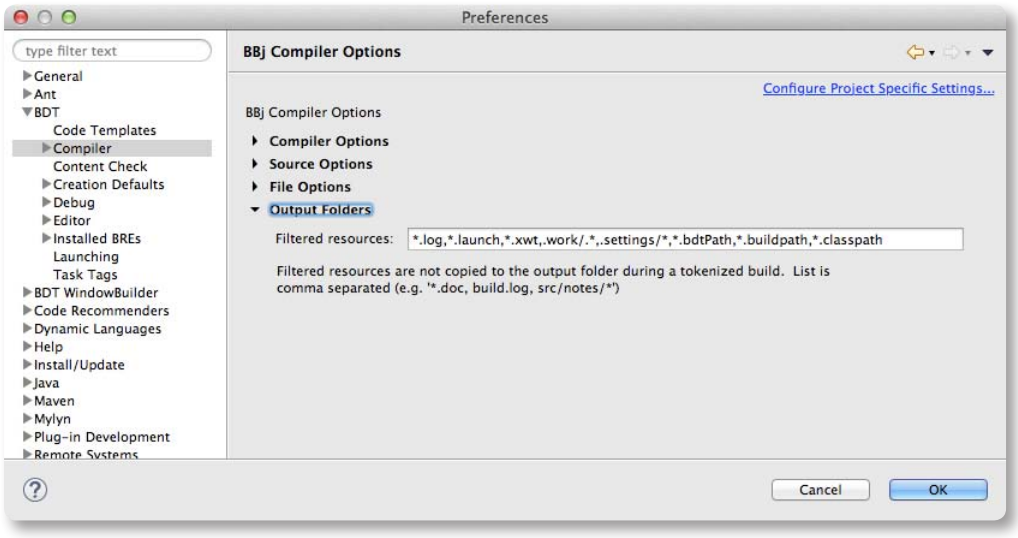

Figure 11. Filtering the files that will be copied to your Output Folder

Filtered Resources
BDT will not copy files that match these naming patterns into your output folder during a tokenized project build. By default this list includes files such as *.bdtPath, *.buildpath, and *.classpath - files which BDT and Java create for project configuration but which have no use in your output environment. You can add as many naming patterns as you like, separated by commas, using the asterisk (*) as a wildcard symbol.

To continue, navigate to Preferences > BDT > Compiler > Errors / Warnings.

Errors/Warnings

The next set of preferences we will investigate relates to warnings that may be generated when you compile a BBj program (see **Figure 12**).

Potential Programming Problems
By default, BDT ignores a number of low priority programming issues when it detects them in a BBj program. If you would like instead to receive a warning notification in the Problems view when BDT encounters one of these issues, select 'Warning' in the dropdown list for that issue.

As an example, if you attempt to import a BBj library class that exists in a SAVEP form into the library, BDT will refuse. If you change the 'Import password protected' severity level from 'Ignore' to 'Warning', the next time you compile you will see something like **Figure 13** in your Problems view in Eclipse.

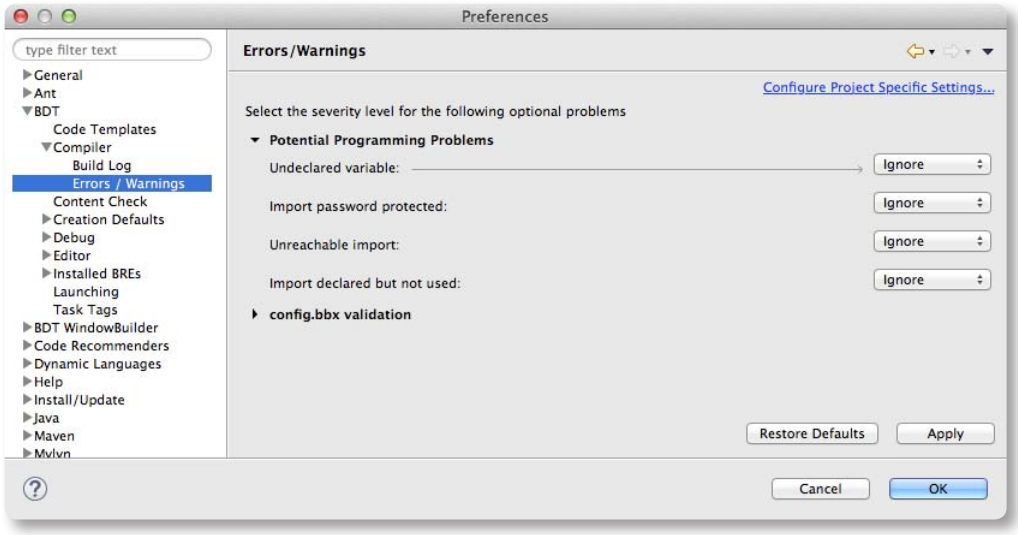

Figure 12. Notifications for potential programming problems

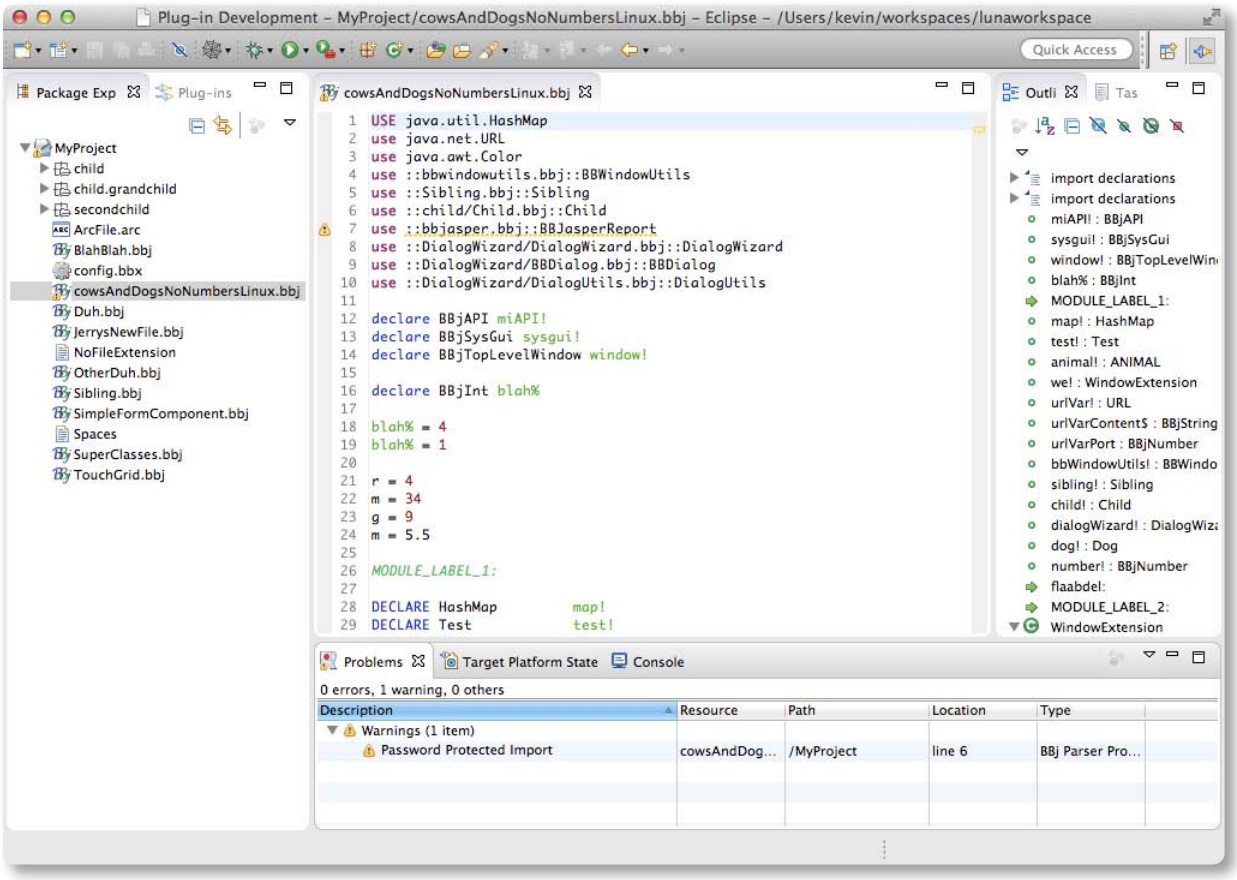

Figure 13. Password Protected import warning in the 'Problems View'

You will see similar warnings if you change the other severity levels to Warning. BASIS considers it a 'Best Practice' to always declare your variables before using them and set the 'Undeclared variable' severity level to 'Warning'.

To continue, navigate to Preferences > BDT > Content Check.

Content Check Preferences

What is Content Check? It's a time saving practice that lets BDT *handle your files correctly by displaying them with an appropriate content icon, parsing them for code completion information, and so on*. But when are content checks done and how?

Whenever events such as a project build begins, a project first opens, or when indexing occurs, Eclipse will ask BDT, "Is this a source file?" Some workspace events cause Eclipse to send every file in the project to the BDT source file validator. If BBj demands that every file have a fixed extension that correctly reflects its content, we would always be able to tell what is in a file just by looking at its file extension. But, since we can't always tell what is in a BBj file by looking at the file extension, BDT uses Apache Tika to determine each file's content type. For Tika to determine a file's content type, it must open the file and look for 'magic bytes' and other header information which might be present there. Although individual files can be opened and closed fairly quickly, when your project has hundreds or thousands of files it may take a significant amount of time to content check all of your files.

BDT is only concerned with BBj source files, tokenized files, SAVEP files, BBj data files, and so on. We call these 'BBj-interesting' files. Among other things, BBj-interesting files always copy into the output folder during tokenized builds.

Examine the 'Exclusion Naming Patterns' list in the 'Content Check Preferences' display shown in **Figure 14**.

One way to stop BDT from opening files to validate their content is to add their name to an exclusion list of naming patterns: for example, *.txt, *.xml, *.htm*. Files in the exclusion list of naming patterns are always treated as non-BBj-interesting, so BDT will not even send them to Tika for content type detection. The naming patterns supported in the exclusion list follow file search wildcard rules you may be familiar with when searching for files on your operating system.

Exclusion isn't the only interesting thing here. There is also a contrasting 'Inclusion Naming Patterns' list. So why have both? Let's say that *.txt is in the exclusion list, but you have files in your project with the .txt file extension that actually are BBj source files. If they follow a specific naming pattern, say for example `MyFile_*.txt`, you can add that to the inclusion patterns list and still see those files as BBj-interesting while excluding all other *.txt files.

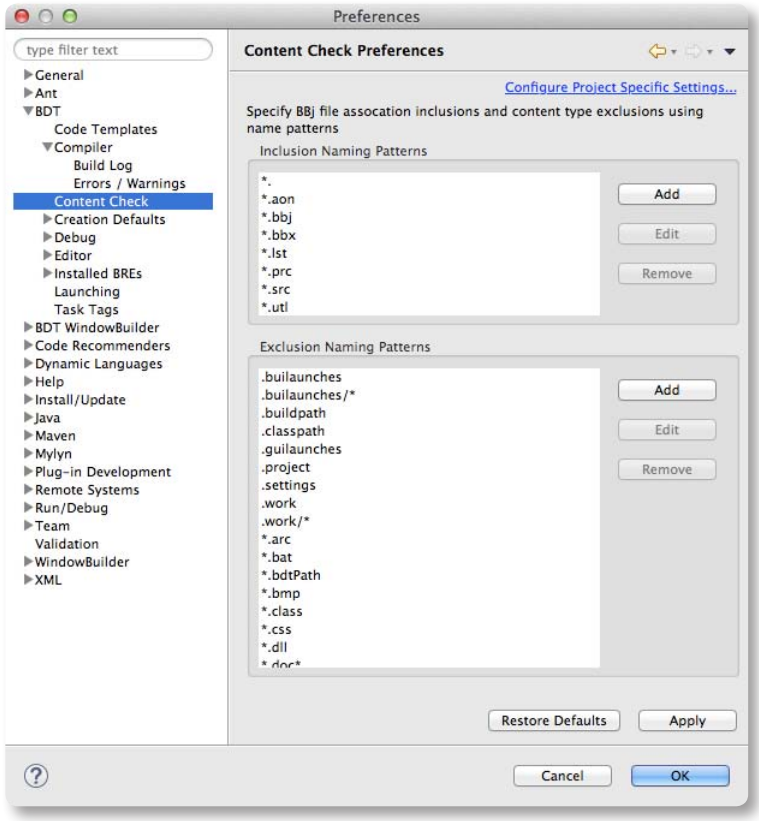

Figure 14. Exclusion Naming Patterns in Content Check Preferences

Figure 15 is a screenshot of a **`MyFile_1.txt`** file in an editor window before adding this naming pattern to the inclusion list:

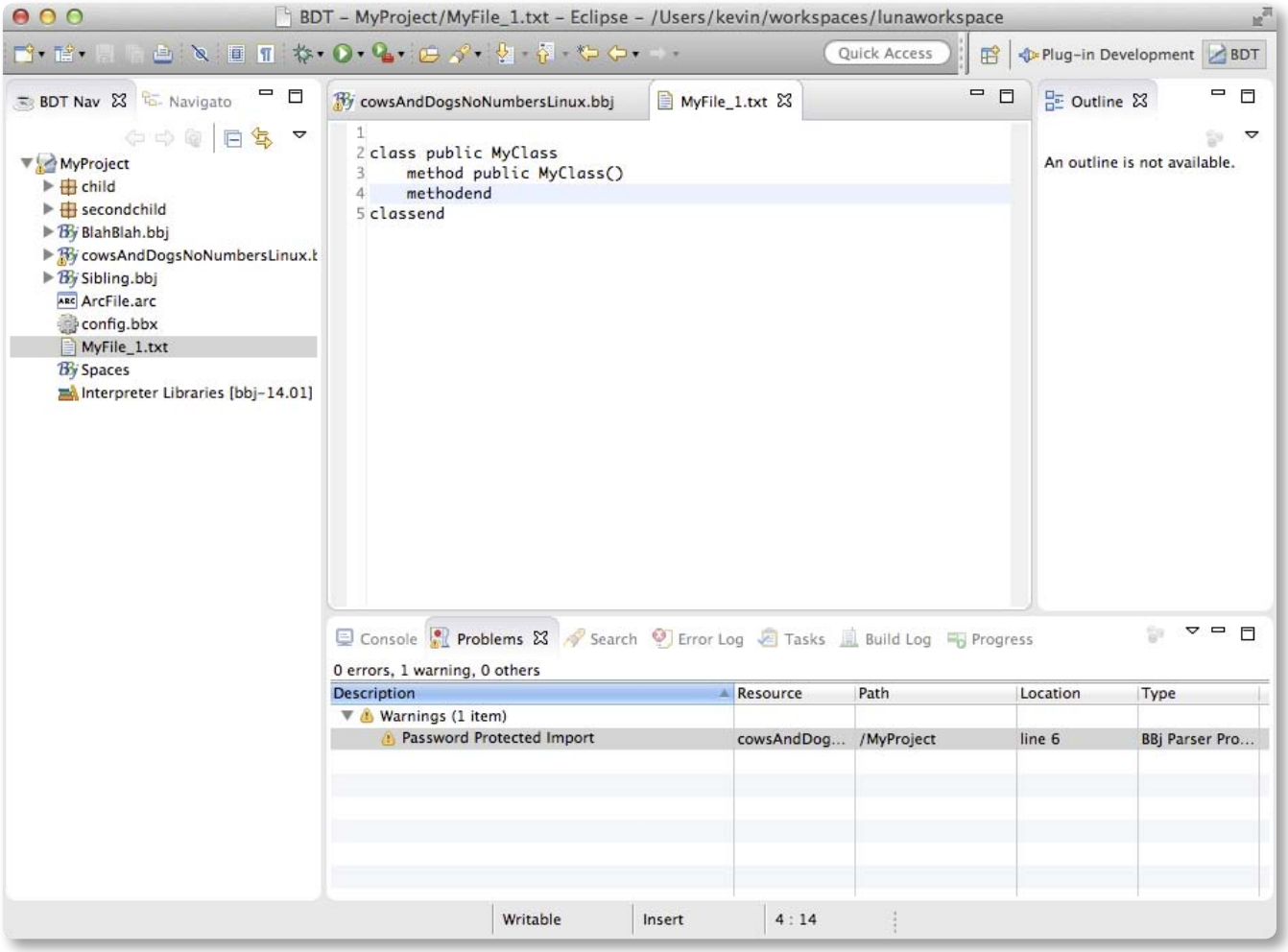

Figure 15. Workspace before adding an entry to the inclusion list

Notice in the BDT Navigator's tree control that **`MyFile_1.txt`** has a text file icon. Also, notice that Eclipse opened the text editor to edit the file (the icon on the 'Editor' tab is a text file icon, and there is no syntax coloring in the text). Back in the Preferences > BDT > Content Check display, click on the [Add] button next to the 'Inclusion Naming Patterns' area shown in **Figure 14**. Enter the value **`MyFile_*.txt`**, and click [OK]. Examine the file **`MyFile_1.txt`** in the 'BDT Navigator' and notice that it now shows a BBj content type icon. **`MyFile_1.txt`** now appears as BBj-interesting, as a BBj source file.

Close the text editor window in **MyFile_1.txt**, and double-click on **MyFile_1.txt**in the BDT Navigator. This opens a new editor window. Notice that now the BBj editor opens the file in a BBj code editor, not the text editor (the icon on the tab is a BBj file icon, and there is now syntax coloring in the text) as shown in **Figure 16**.

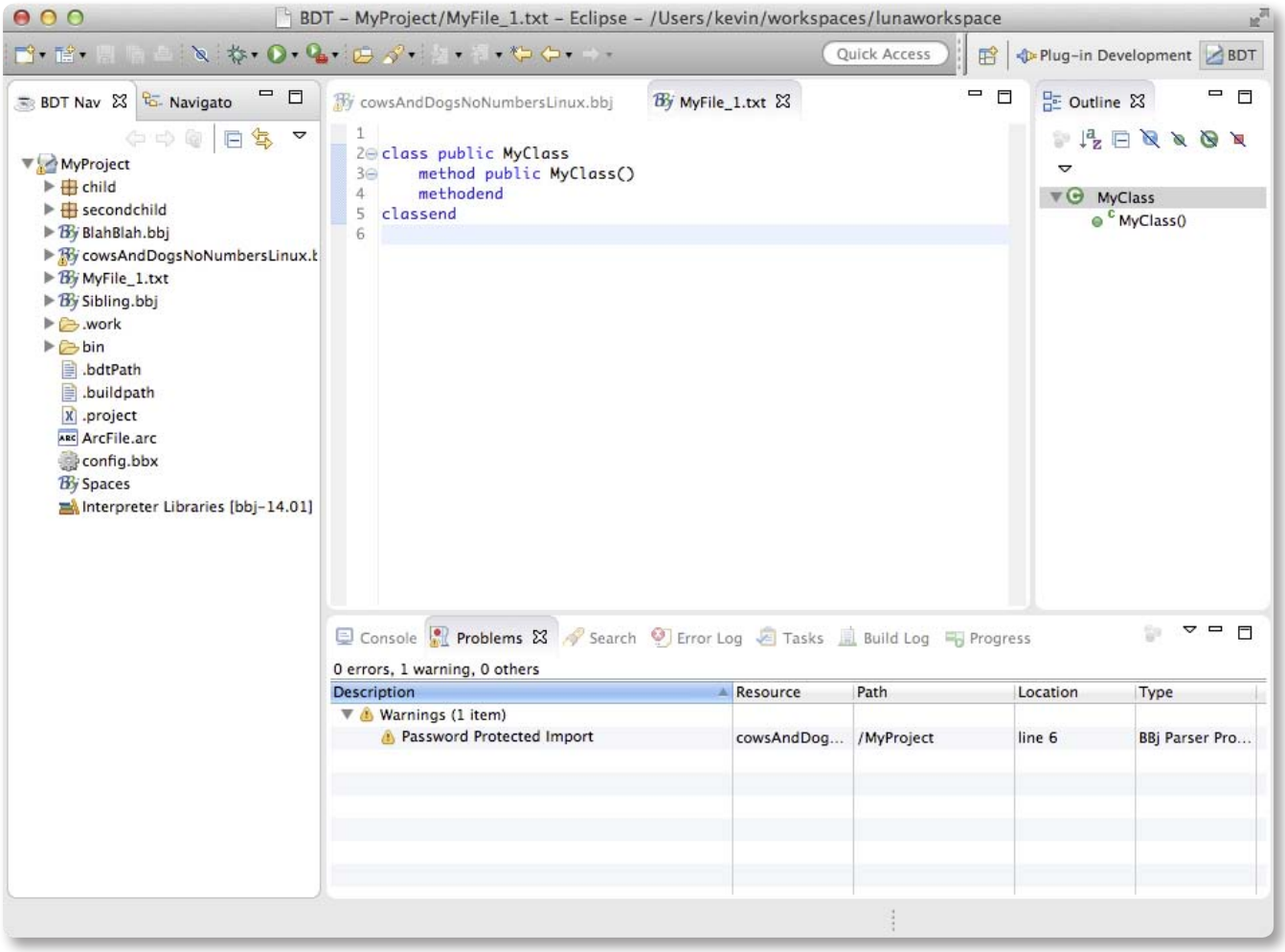

Figure 16. MyFile_1.txt being edited with a BBj Content Check inclusion preference set

Summary

The BDT Eclipse CodeEditor plug-in is a very powerful and configurable tool. We have covered a number of BDT preferences in this article, but there are still quite a few that we just couldn't fit in. Please check out the online documentation for more information. If you are already familiar with Eclipse's Java Development Tools (JDT), you will find that much of BDT is already familiar to you. Either way, BDT preferences can make you much more productive, and you can make it work, look, and feel 'your way'!

- For instructions on installing Eclipse and the BDT plug-in, refer to *Preparing Eclipse for BASIS-Provided Plug-ins* in the online Help
- Read *BDT Tips for Less Pain and More Gain* in this issue
- Review *Creating Your First BBj Project*
- Visit Apache Tika