The Toolset of the Future

In our ongoing effort to make your BBj project development life easier and more productive, BASIS announces the new BASIS Development Tools (BDT), which now integrates with one of the most popular and powerful IDEs available, Eclipse. This article uncovers the strong evidence on which BASIS chose to move to Eclipse and provides some simple getting-started tutorials.

Goodbye NetBeans IDE (RIP)

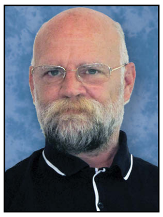
Before explaining why BASIS moved to Eclipse and taking a glimpse at the BDT, let's have a moment of silence for our BASIS NetBeans IDE.

Why are we saying goodbye to the old BASIS IDE? Here are just a few reasons:

- Newer versions of NetBeans were no longer compatible with our original architecture for BASIS plug-ins.
- Oracle/Sun refactored the NetBeans and Java infra-structure that would require a rewrite of all existing BASIS plug-ins.
- BASIS has extensive experience with Eclipse, the IDE of choice for our BASIS Java developers.

Why Eclipse?

- **It has strong roots.** In November 1998, IBM Software Group began creating a development tools platform that eventually became known as Eclipse. Industry leaders Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, and Webgain formed the initial eclipse.org Board of Stewards in November 2001, which eventually became the Eclipse Foundation.
- **It is widely used.** Eclipse has become the standard for Java-based IDEs, product development tools, modeling tools, GUI builder tools, database tools. Take a look at this 'brief' list of available Eclipse-based software at links.basis.com/esw. Eclipse is everywhere - *ubiquitous*. Eclipse released Juno on June 27, 2012 and was downloaded 1,200,000 times during its first 20 days. Of an estimated 10.3 million Java developers, 68% use Eclipse as their primary development environment.



By Kevin Hagel
Software Developer

- **It is consistent across all platforms.** Eclipse uses JFace and the Standard Widget Toolkit (JFace/SWT), an alternative to Oracle's AWT/Swing. JFace and SWT can be used independently of the rest of the Eclipse Platform, rivaling native applications written in AWT/Swing. It gives applications a polished and consistent look and feel across all platforms.
- **It is a platform.** In Eclipse's own words published on their site wiki.eclipse.org/Platform, *"The Platform defines the set of frameworks and common services that collectively make up infrastructure required to support the use of Eclipse as a component model, as a Rich Client Platform (RCP) and as a comprehensive tool integration platform. These services and frameworks include a standard workbench user interface model and portable native widget toolkit, a project model for managing resources, automatic resource delta management for incremental compilers and builders, language-independent debug infrastructure, and infrastructure for distributed multi-user versioned resource management."*
- **It is free and easy.** The platform is easily extended simply by downloading plug-ins, most of which are open source and free.
- **It is current.** The Eclipse Marketplace offers more than 6 million new and updated solutions installed directly from Eclipse. Just about every programming language there is has an Eclipse IDE available here. If you have a software problem looking for a solution, here's the place to look.
- **It is robust.** It is the Eclipse Workbench; the term Workbench refers to the entire desktop development environment, containing toolbars, editors, menus, views, etc. IBM Developerworks provides an excellent series of articles titled Mastering Eclipse v3.4. The content is dated, but the terms and names of the parts of the workbench are relevant today.
- **It is prolific.** The Eclipse Marketplace currently lists 85 different Team Development tools.
- **It is compatible.**
  - Integrates with SVN, CVS, Git, Mercurial, and Perforce for source code management.
  - Integrates with Bugzilla for bug tracking.
  - Integrates with Mylyn for task and application lifecycle management.
- **It is multi-purpose.** Eclipse can manage your databases, your team repositories, your web servers, your tasks lists, your report generators…there is a never-ending list of ways it facilitates project development of all kinds.

First Glimpse at the BDT

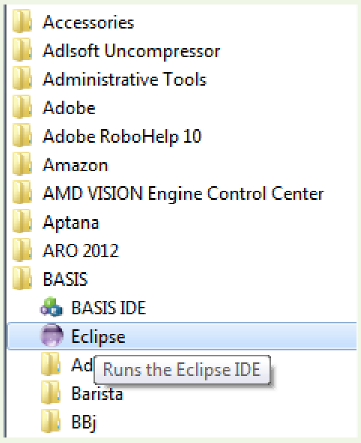
Before taking the first glimpse at BDT, let's install and run Eclipse. Even if you already have Eclipse installed, follow the **Installing the BDT** tutorial to ensure you have the correct version of Java and Eclipse before you install your BDT plug-in.

Installing the BDT

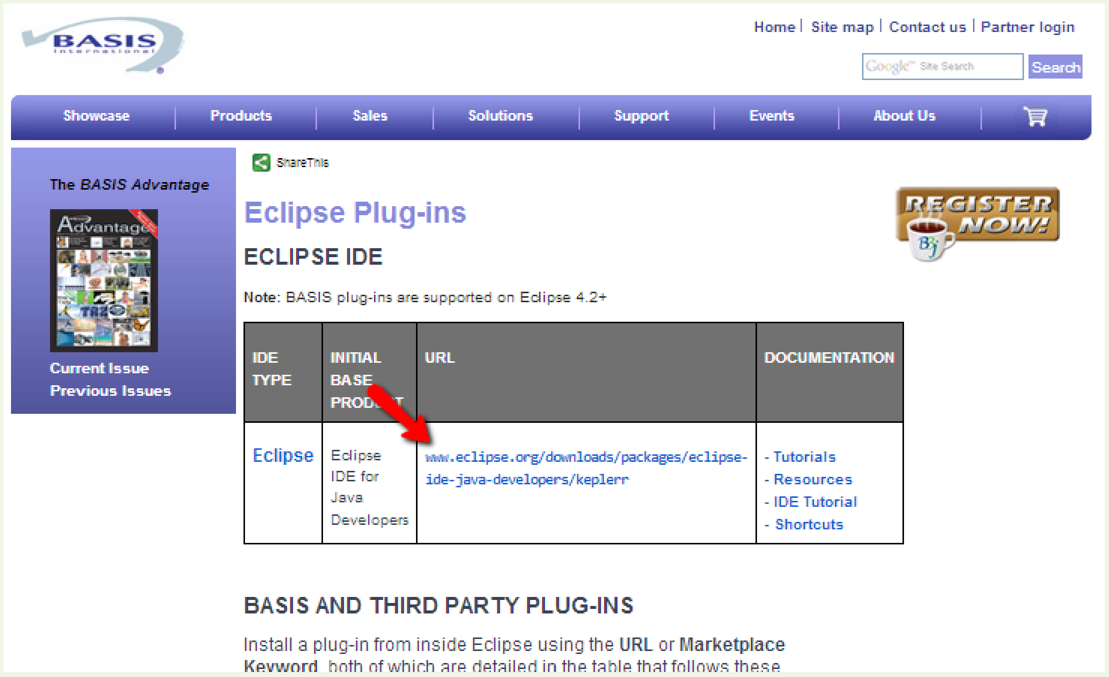
- Install Eclipse
- Determine Your Workspace
- Launch Eclipse
- Install the BDT Plug-in

Install Eclipse

1. Locate the Eclipse shortcut installed with your BBj 13.x installation, shown in the BASIS menu below.



2. Click the Eclipse icon, which opens your browser to the BASIS web page
3. Select the URL link noted below.



Eclipse Plug-ins

ECLIPSE IDE

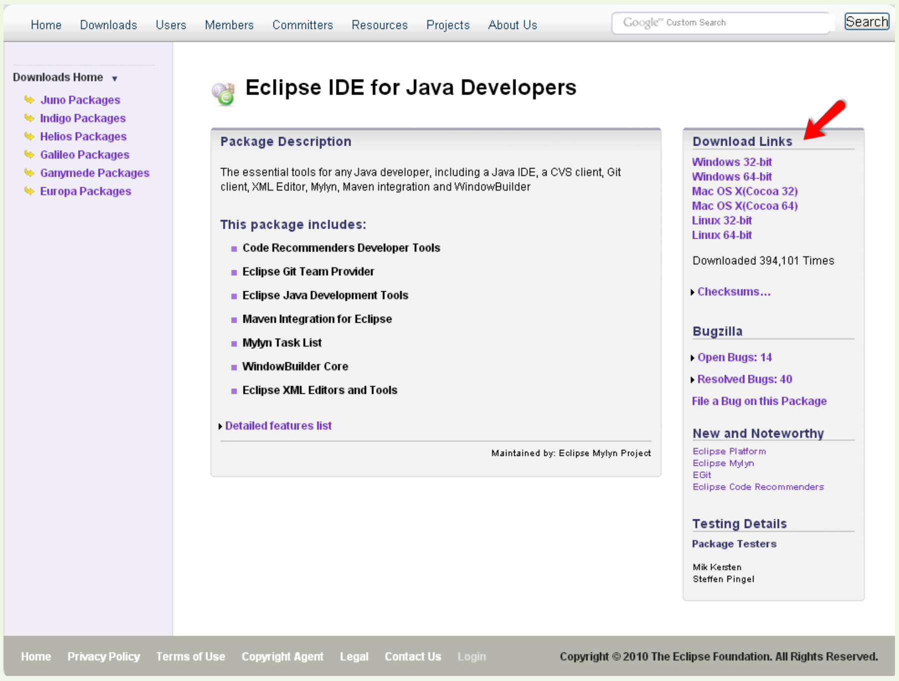
Note: BASIS plug-ins are supported on Eclipse 4.2+

IDE TYPE	INITIAL BASE PRODUCT	URL	DOCUMENTATION
Eclipse	Eclipse IDE for Java Developers	www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplerr	- Tutorials - Resources - IDE Tutorial - Shortcuts

BASIS AND THIRD PARTY PLUG-INS

Install a plug-in from inside Eclipse using the URL or **Marketplace Keyword**, both of which are detailed in the table that follows these

4. On the Eclipse IDE for Java Developers page, locate your operating system in the "Download Links" section to the right. As of this writing. the current Eclipse version is "Kepler," but BDT will run on Eclipse Juno or any version 4.x or higher.


Home Downloads Users Members Committers Resources Projects About Us Google™ Custom Search Search

Downloads Home
- Juno Packages
- Indigo Packages
- Helios Packages
- Galileo Packages
- Ganymede Packages
- Europa Packages

Eclipse IDE for Java Developers

Package Description

The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration and WindowBuilder

This package includes:
- Code Recommenders Developer Tools
- Eclipse Git Team Provider
- Eclipse Java Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- WindowBuilder Core
- Eclipse XML Editors and Tools

Detailed features list

Maintained by: Eclipse Mylyn Project

Download Links
- Windows 32-bit
- Windows 64-bit
- Mac OS X(Cocoa 32)
- Mac OS X(Cocoa 64)
- Linux 32-bit
- Linux 64-bit

Downloaded 394,101 Times

Checksums...

Bugzilla
- Open Bugs: 14
- Resolved Bugs: 40
- File a Bug on this Package

New and Noteworthy
- Eclipse Platform
- Eclipse Mylyn
- EGit
- Eclipse Code Recommenders

Testing Details

Package Testers

Mik Kersten
Steffen Pingel

Home Privacy Policy Terms of Use Copyright Agent Legal Contact Us Login Copyright © 2010 The Eclipse Foundation. All Rights Reserved.

NOTE: On Windows systems, we strongly advise defining an installation location with a name as short as possible and without spaces like `C:\eclipse` rather than `C:\Program Files\eclipse`. Windows can run into difficulties with long directory paths and filenames since some of the plug-ins Eclipse uses have very long file names. LINUX and Mac system do not encounter this problem.

Determine Your Workspace

Back to Installing BDT

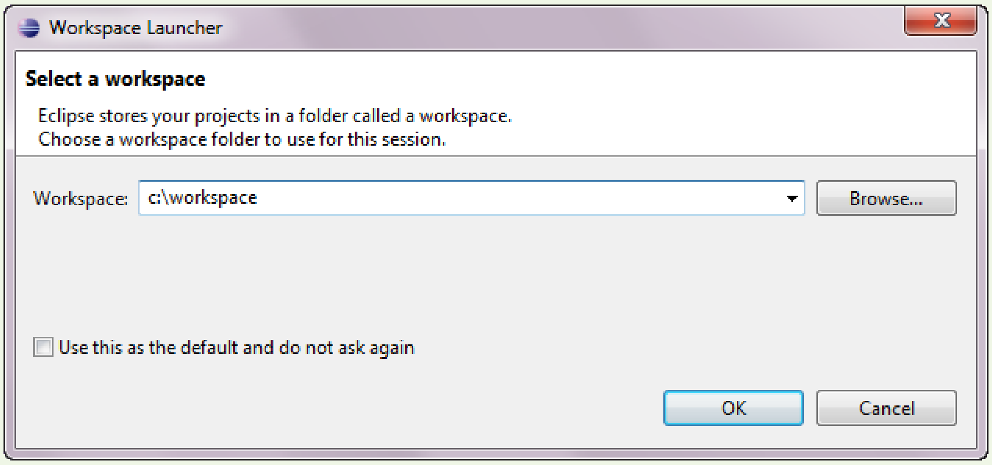
A workspace is a logical set of projects. You can have as many workspaces as you like, each with their own location and their own set of preferences and customizations.

By default project directories will be created inside the workspace directory. It is possible to create a project in an external location but still be referenced in the workspace.

Launch Eclipse

Back to Installing BDT

1. Now that you've installed Eclipse, run the executable.
2. The Workspace Launcher now appears, asking for the location of your workspace. In this case, we'll use **`C:\workspace`** shown below.


Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: c:\workspace Browse...

☐ Use this as the default and do not ask again

OK Cancel

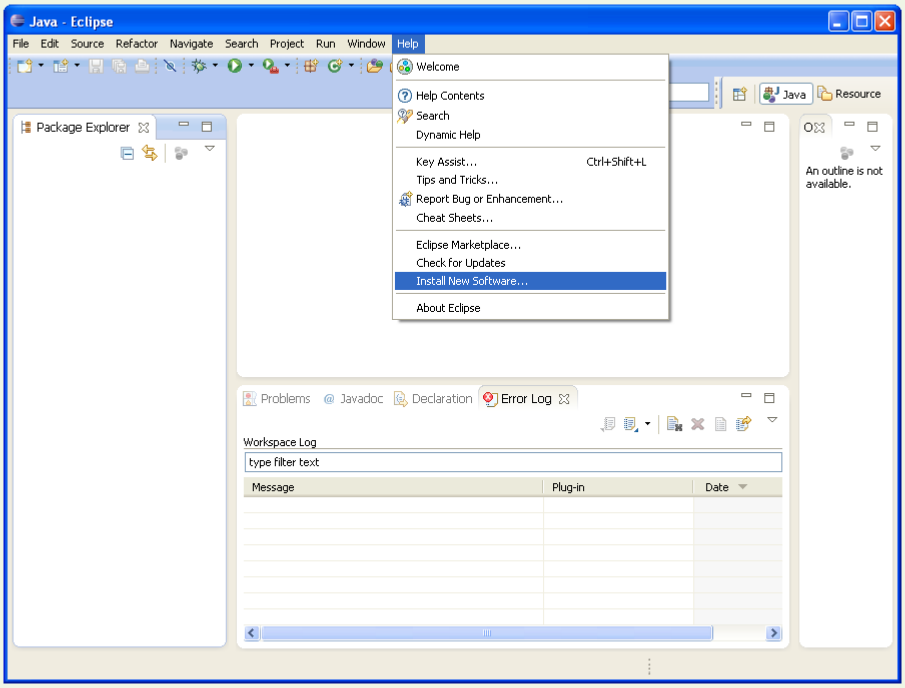
3. Optional - select the checkbox option "Use this as the default and do not ask again," if you like, and click [OK].

Install the BDT Plug-in

Back to Installing BDT

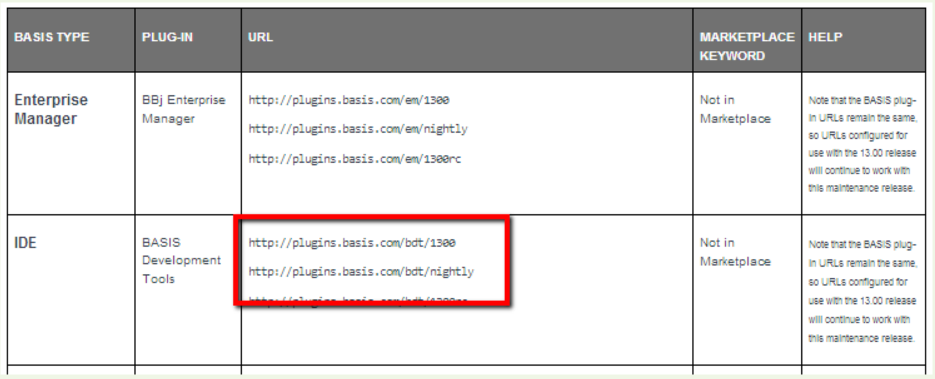
Since BDT is an Eclipse plug-in, we will install it following the same process as any other plug-in – through an “update” site located under the Help menu.

1. Select the Help > Install New Software.



You should still have the BASIS Eclipse Plug-in page open in your browser from Install Eclipse step #3. If not, locate the shortcut in your Start menu as instruction in Install Eclipse or go directly to links.basis.com/eclipse.

2. Scroll down until you locate the two BDT URLs:

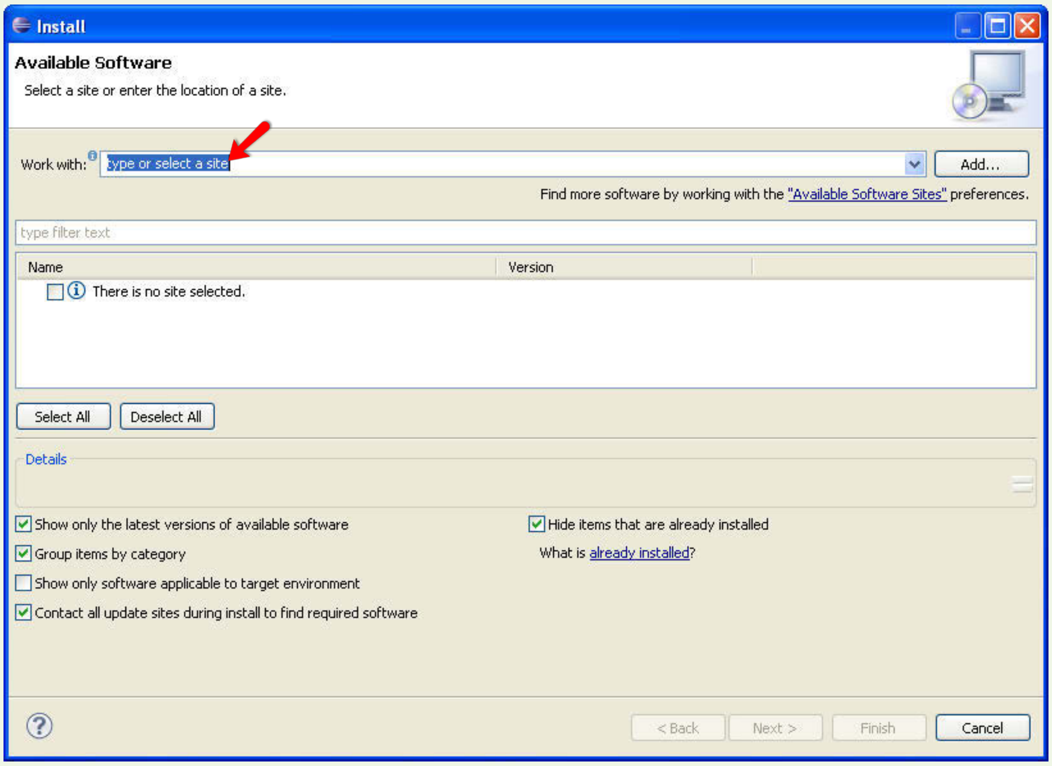


BASIS TYPE	PLUG-IN	URL	MARKETPLACE KEYWORD	HELP
Enterprise Manager	BBj Enterprise Manager	http://plugins.basis.com/em/1300 http://plugins.basis.com/em/nightly http://plugins.basis.com/em/1300rc	Not in Marketplace	Note that the BASIS plug-in URLs remain the same, so URLs configured for use with the 13.00 release will continue to work with this maintenance release.
IDE	BASIS Development Tools	http://plugins.basis.com/bdt/1300 http://plugins.basis.com/bdt/nightly http://plugins.basis.com/bdt/1300rc	Not in Marketplace	Note that the BASIS plug-in URLs remain the same, so URLs configured for use with the 13.00 release will continue to work with this maintenance release.

The first URL provides the latest stable release; the second (for the more adventurous), gives you the latest bleeding edge release.

3. Enter the following in the dialog window below where it says “Type or select a site”:

```
BDT - http://plugins.basis.com/bdt/1300
```



Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

type filter text

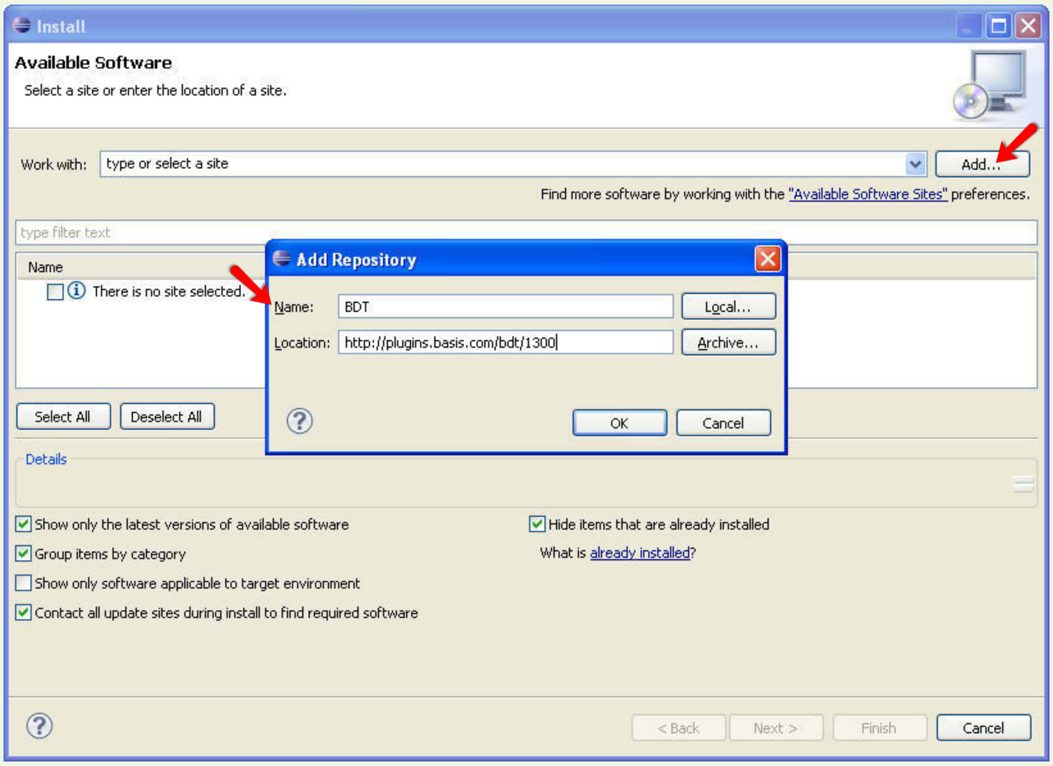
Name	Version
☐ There is no site selected.	

Details

- [✓] Show only the latest versions of available software
- [✓] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software
- [✓] Hide items that are already installed

What is already installed?

or click [Add…] and enter the same information in this way:



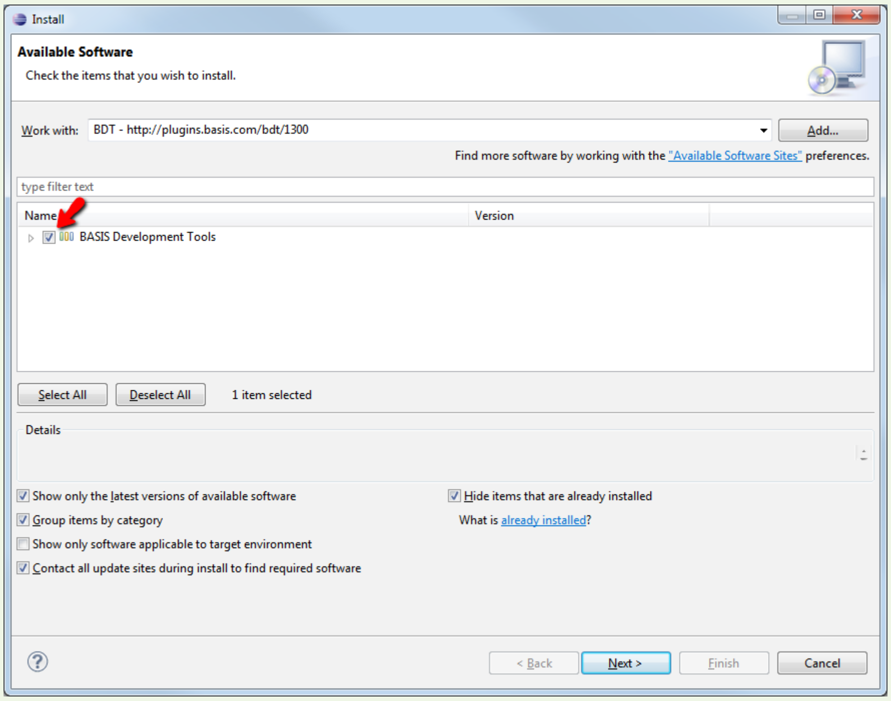
Add Repository

Name: BDT

Location: http://plugins.basis.com/bdt/1300

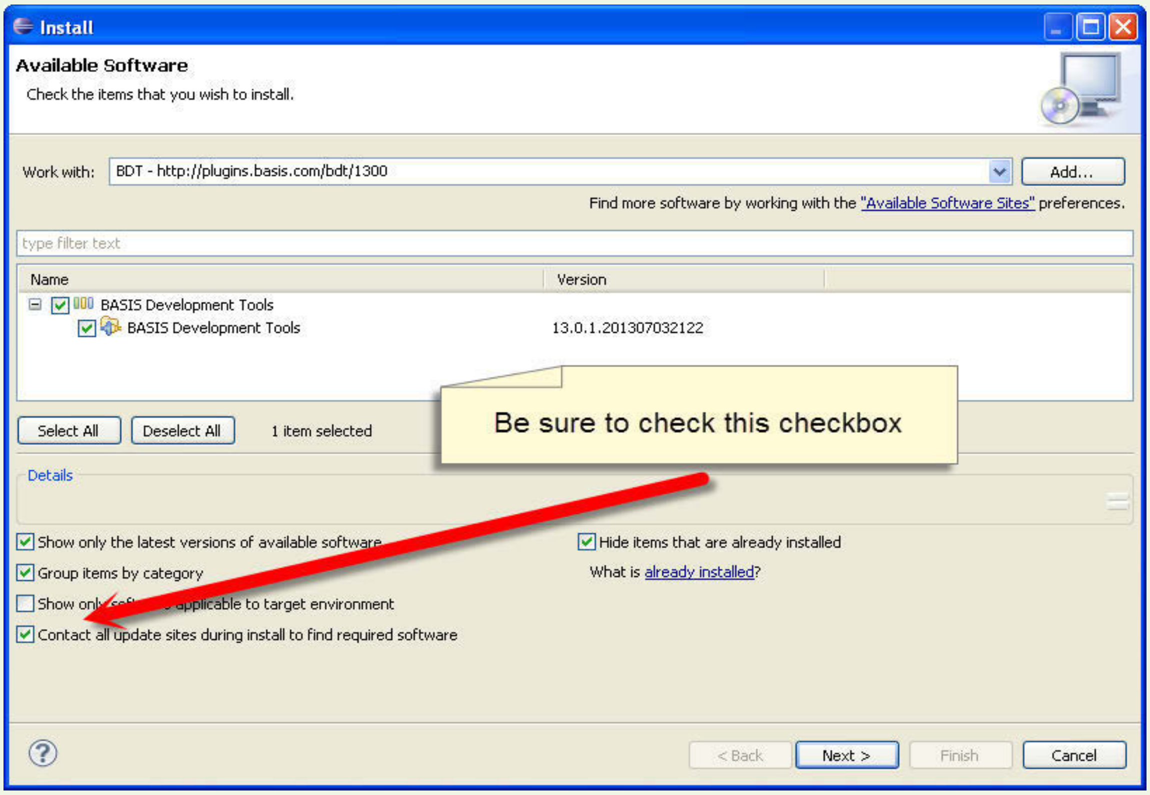
4. Click [OK]

5. In the dialog box shown below, mark the checkbox next to BASIS Development Tools:


Install
Available Software
Check the items that you wish to install.
Work with: BDT - http://plugins.basis.com/bdt/1300
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
BASIS Development Tools
Select All
Deselect All
1 item selected
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel

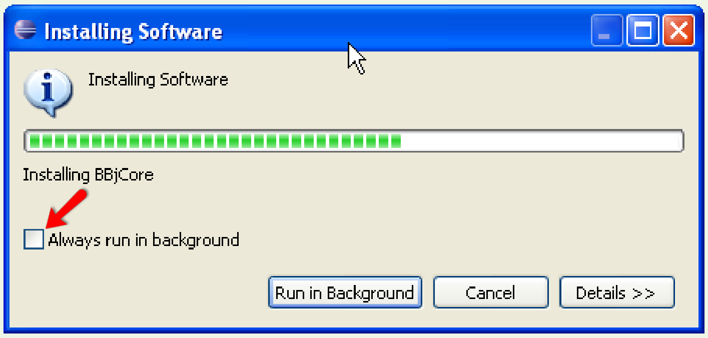
6. Mark the checkbox "Contact all update sites during install to find required software."

NOTE: BDT has certain resource requirements that do not come preinstalled with Eclipse, yet are obtainable through Eclipse. If you do not select this checkbox, BDT will not install correctly. In fact, it is advisable to always mark this checkbox for every plug-in you install.

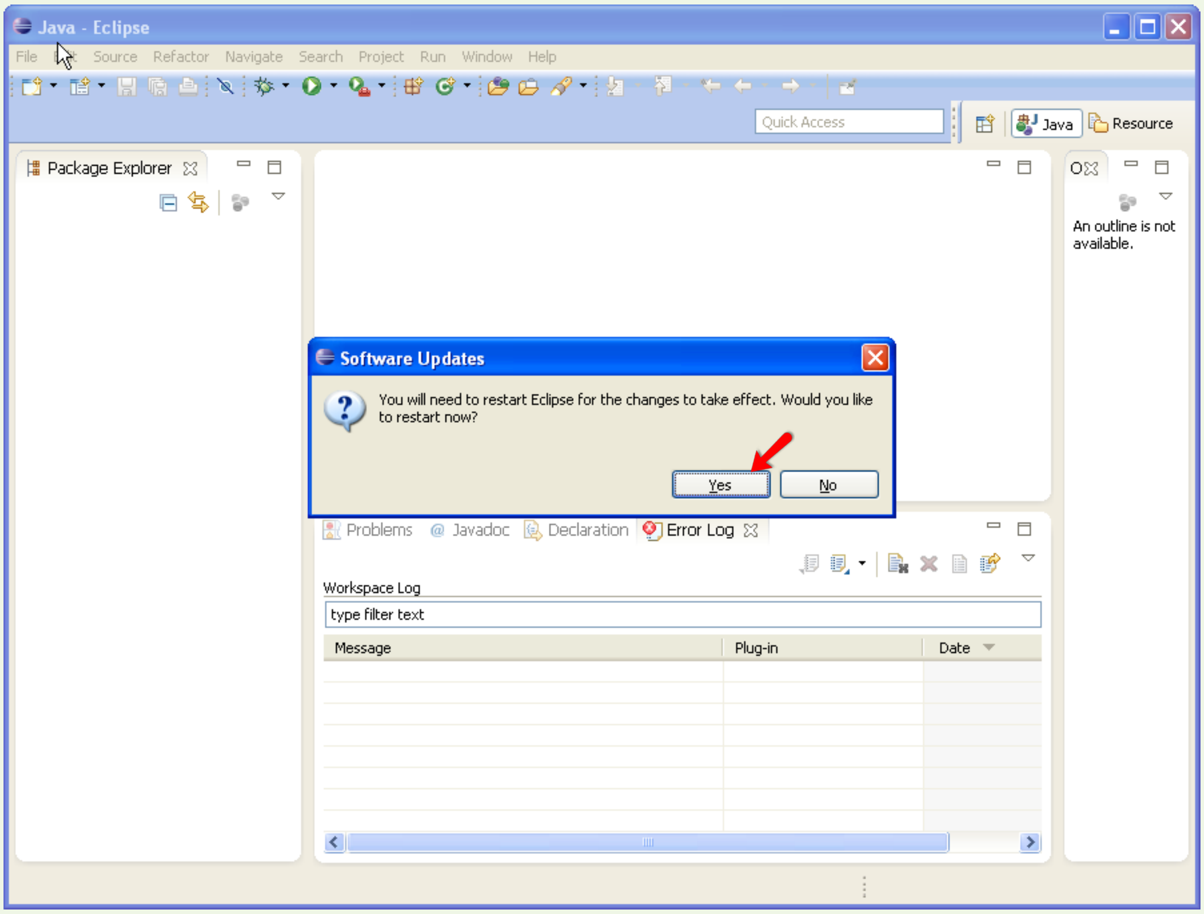

Install
Available Software
Check the items that you wish to install.
Work with: BDT - http://plugins.basis.com/bdt/1300
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
BASIS Development Tools
BASIS Development Tools
13.0.1.201307032122
Be sure to check this checkbox
Select All
Deselect All
1 item selected
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel

7. Click [Next >] twice.
8. Select "I accept the terms of the license agreement" and press [Finish].

9. Optional - mark “Always run in background,” if desired, in the progress window.



10. Click [Yes…] once installation is complete to restart Eclipse for the changes to take effect.


Software Updates

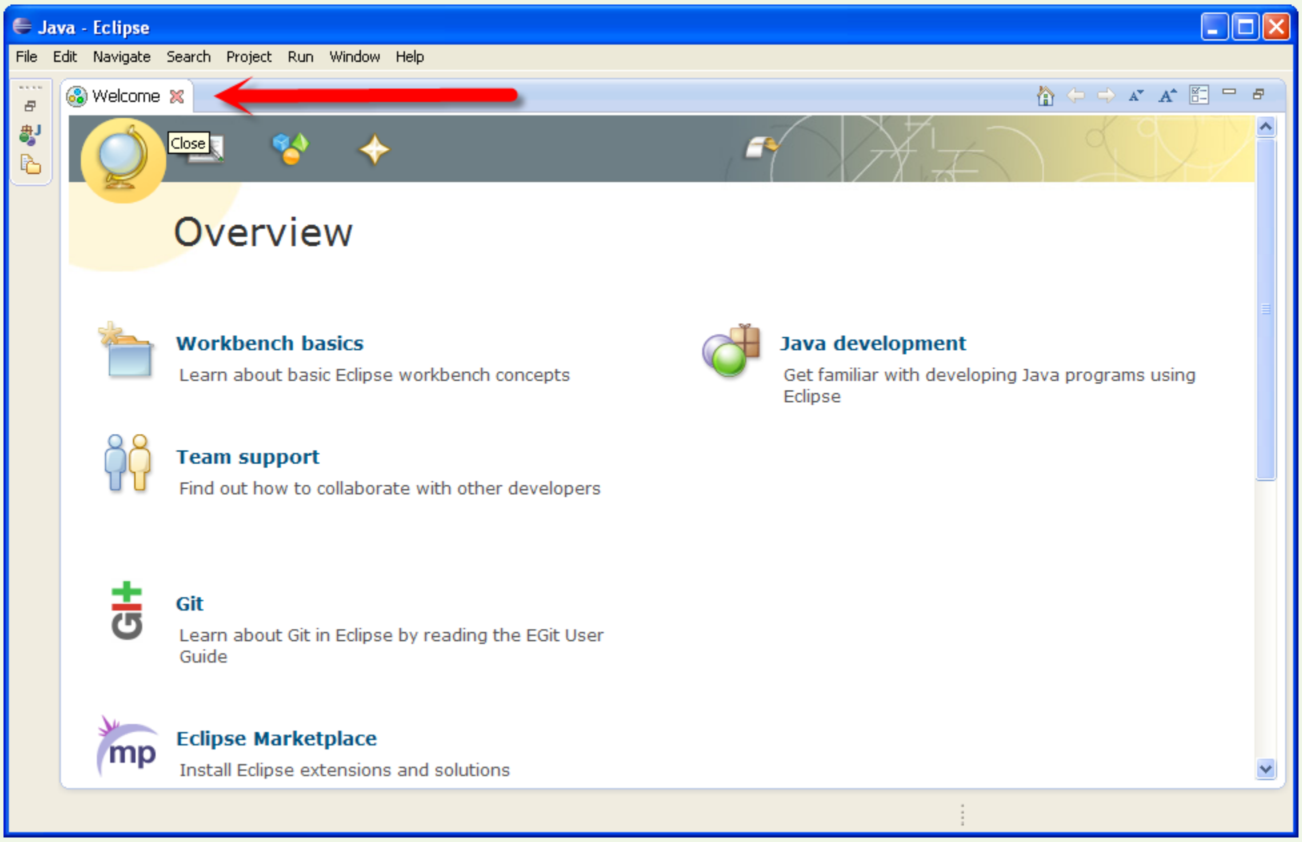
You will need to restart Eclipse for the changes to take effect. Would you like to restart now?

Eclipse will restart and once again, ask you to set your workspace.

NOTE: Eclipse will not ask for your workspace if you marked “Use this as default and don’t ask again...” in Launch Eclipse step #3.

11. Select your workspace and press [OK].

12. Optional - when Eclipse restarts, it will open with the Welcome tab that you can close by clicking the "x" shown below.


Java - Eclipse
File Edit Navigate Search Project Run Window Help
Welcome
Close
Overview
Workbench basics
Learn about basic Eclipse workbench concepts
Java development
Get familiar with developing Java programs using Eclipse
Team support
Find out how to collaborate with other developers
Git
Learn about Git in Eclipse by reading the EGit User Guide
Eclipse Marketplace
Install Eclipse extensions and solutions

Congratulations, you have installed BASIS Development Tools. The next step isn't very obvious. You will need to switch to the BASIS "perspective." Follow along in the **Guide to Perspectives**.

Guide to Perspectives

- Perspectives
- Views
- Editors
- The BASIS Perspective

Perspectives

An Eclipse "perspective" is a visual container for a set of views, menus, editors, and toolbars, available only after installing the plug-in that provides the perspective. A perspective defines the initial set and layout of views in the workbench window. Within that window, each perspective shares the same set of editors, and each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. Perspectives control what appears in certain menus and toolbars, defining visible action sets that you can change to customize a perspective. You can save a perspective that you built in this manner so that you can open it again later.

Some common perspectives you may use are BASIS, BBj Enterprise Manager, Debug, SVN Repository, Java, Report Design, Resource…many more. Typically a plug-in provides its own perspective.

You can customize a perspective by selecting **`Window > Customize Perspective`** and then choosing which toolbars appear, what appears in them, the contents of the views, etc. Then just save this customized perspective with your own chosen name.

Views

Views support editors and provide alternative presentations as well as ways to navigate the information in your workbench. For example, the BASIS Navigator and Project Explorer and other navigation views display projects and other resources that you are working with.

Views also have their own menus. Some views also have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view. A view might appear by itself, or stacked with other views in a tabbed notebook.

To change the layout of a perspective, open and close the views and by docking them in different positions in the workbench window. In addition, you can detach views from the workbench outside the bounds of the IDE to take advantage of multiple monitors.

Editors

Back to Guide to Perspectives

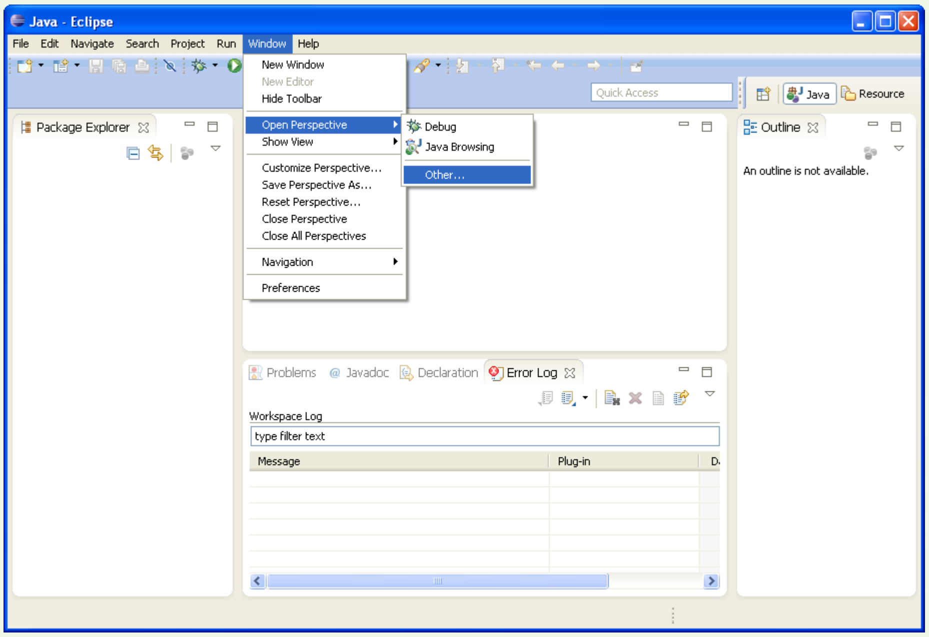
Most perspectives in the workbench are comprised of an editor area and one or more views. Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor. Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes. By default, editors are stacked in the editor area, but you can choose to tile them in order to view source files simultaneously.

The BASIS Perspective

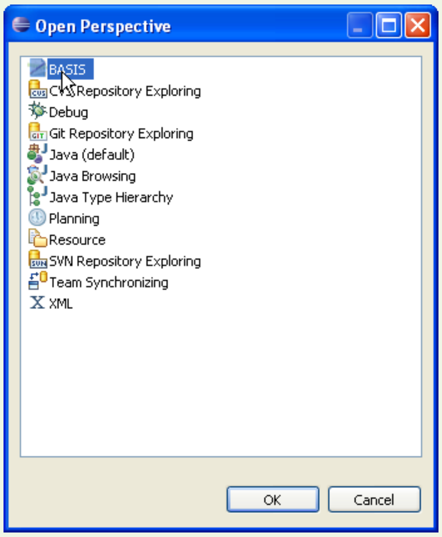
Back to Guide to Perspectives

To open the BASIS perspective,

1. Choose Window > Open Perspective > Other...



2. Double-click on BASIS or select BASIS in the Open Perspective window, and press [OK].



Now you have the BDT's BASIS Perspective open and selected. Let's go over some of the component parts.

BDT Components

- Projects, Folders, and Files
- BASIS Navigator
- Editor
- Builder
- Executor
- Debugger

Projects, Folders, and Files

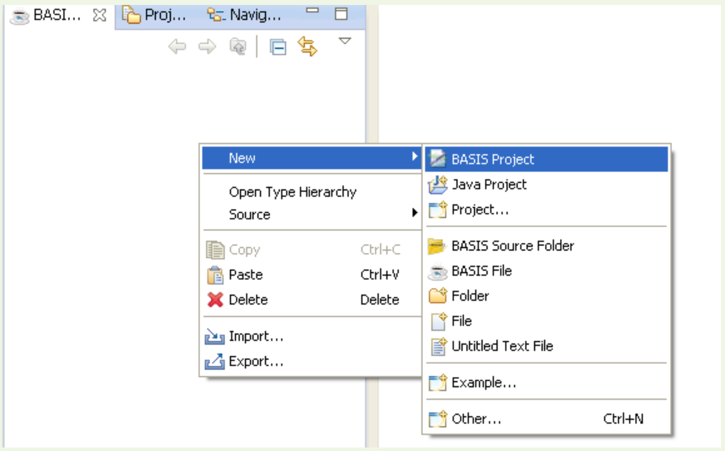
There are three different types of resources in the workbench: projects, folders, and files. Projects are the largest structural unit used by the workbench. Projects contain folders and files, and developer can open, close, or build them. Folders can contain other folders and files. The workbench and BDT itself provide a number of mechanisms for working with projects, folders, and files.

BDT projects contain source folders, output folders, source modules (files, as in “Foo.bbj”), tokenized files, and related configuration files for building a BBj program. BDT projects can also contain “foreign resources,” which typically are image files, html, Jasper configurations, etc. Source modules pass through the compiler to produce tokenized files. All foreign resources are simply copied into the output folder(s).

Let’s illustrate these concepts by creating a simple BASIS project.

Create a project

1. Select from the menu at top, File > New > BASIS Project, or right-click in the BASIS Navigator and select New > BASIS Project.



2. Create new project as follows:

a. Project name: Demo

b. Accept all the defaults for the new project configuration for:

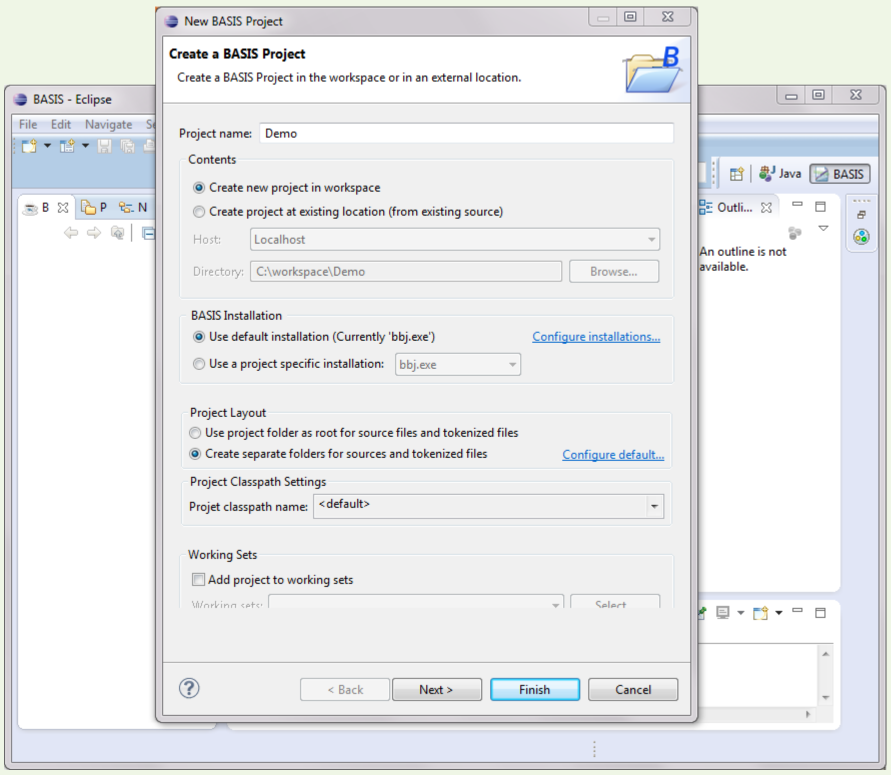
i. Contents - Create the new project in workspace. If you originally chose a workspace name **`C:\workspace`**, this creates a new directory **`C:\workspace\Demo`**.

ii. BASIS Installation - Use default installation. This sets up your project to use the default BASIS installation resources for your project.

iii. Project Layout - Create separate folders for sources and tokenized files. By default this will create two subdirectories in your project **`C:\workspace\Demo\src`** to contain the BBj source files and foreign resources, and **`C:\workspace\Demo\bin`** to contain the tokenized files and copied foreign resources

iv. Project Classpath Setting - Leave as default for this tutorial.

v. Working Sets - Leave as default for this tutorial.



New BASIS Project

Create a BASIS Project

Create a BASIS Project in the workspace or in an external location.

Project name: Demo

Contents
- (●) Create new project in workspace
- ( ) Create project at existing location (from existing source)

Host: Localhost

Directory: C:\workspace\Demo Browse...

BASIS Installation
- (●) Use default installation (Currently 'bbj.exe') Configure installations...
- ( ) Use a project specific installation: bbj.exe

Project Layout
- ( ) Use project folder as root for source files and tokenized files
- (●) Create separate folders for sources and tokenized files Configure default...

Project Classpath Settings

Projet classpath name: <default>

Working Sets
- [ ] Add project to working sets

3. Click [Finish]. The New BASIS Project wizard closes and reveals your project in the BASIS Navigator.

4. In the BASIS Navigator,

a. Select the Demo project to expand the project and reveal the current project contents.
b. Right-click on the src folder.
c. Select New > BASIS File.
d. Type foo as the file name in the dialog that appears.

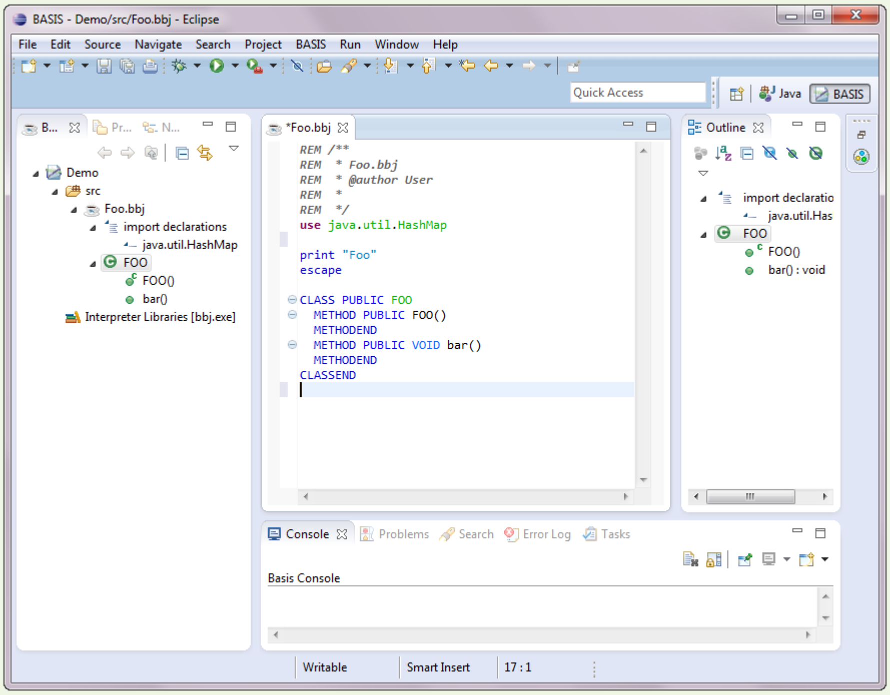
The wizard now creates Foo.bbj and opens it in the editor, using the preconfigured source file template. Next, we are going to add a simple class with some simple methods to help us see what the BASIS Navigator can do.

Add a Class

1. Enter the following source so that your file Foo.bbj contains the following ...

```
REM /**
REM * Foo.bbj
REM * @author User
REM *
REM */

use java.util.HashMap
print "Foo"
escape

CLASS PUBLIC FOO
 METHOD PUBLIC FOO()
 METHODEND
 METHOD PUBLIC VOID bar()
 METHODEND
CLASSEND
```

… and you should see the BASIS Navigator and new Demo project in the left pane and other information in the following:



```
REM /**
REM  * Foo.bbj
REM  * @author User
REM  *
REM  */
use java.util.HashMap

print "Foo"
escape

CLASS PUBLIC FOO
  METHOD PUBLIC FOO()
  METHODEND
  METHOD PUBLIC VOID bar()
  METHODEND
CLASSEND
```

Back to BDT Components

BASIS Navigator

The far left pane contains three tabs: the BASIS Navigator, the Project Explorer, and the Resource Navigator. The BASIS Navigator presents a project-level view of your project in a tree structure, representing the BDT project model. This model starts at the top with the source folder, the source module (Foo.bbj), then down to the internals of the source module – the import declarations (your use statement), class declarations, field declarations, method declarations, and so on.

To navigate to the various parts of your source in the editor, double-click on the element in the navigator. If you press the Link with Editor icon (the two arrows at the top of the navigator, then you can navigate in the editor by simply clicking on the element. This isn't such a magical feature with a tiny source file like this, but imagine a very large file of 10-20 classes, many methods per class, and 1000+ lines of code…this is where navigator really comes in handy!

You can also create new source folders here. For example, you want to keep your .html or your images or other file types in a separate directory. Simply select Demo in the Navigator, File > New > BASIS Source Folder and name it: html. Then create a second source folder and name it: images. To place image files in this new folder, you can import them or drag them into the images folder from your File Explorer (or whatever your OS calls it).

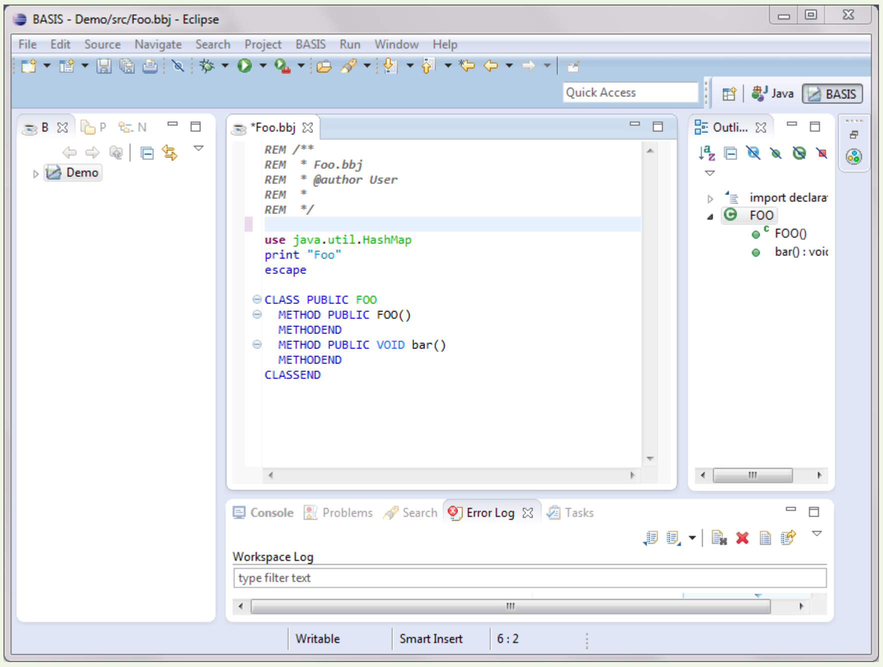
It is possible to have all your source in the project root directory, but it is considered best practice to always put your sources in folders underneath the project. Keep in mind this has consequences in how you Build, Execute, and Debug your project.

Back to BDT Components

Editor

The BASIS Editor is just one of many different editors available in the Eclipse workspace. Different editors are associated with different types of files. For example, the Java editor is associated with files matching *.java. You may have default editors for .xml files, or special-purpose xml editors if you've downloaded plug-ins for them. All BBj source files open in the BASIS Editor, by default. Simply open an editor by double-clicking on the file in the Navigator.

You can open any number of editors at the same time, but only one can be active at any one time. The main menu bar and toolbar for the workbench window contain operations that are applicable to the active editor. Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.



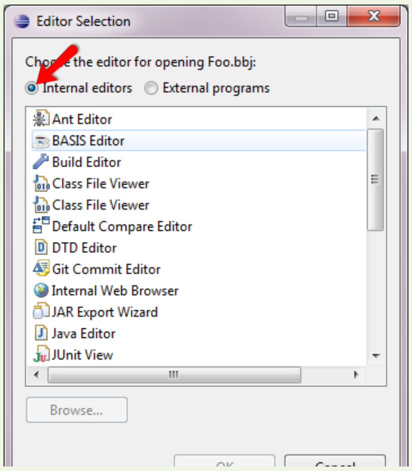
```
REM /**
REM  * Foo.bbj
REM  * @author User
REM  *
REM  */

use java.util.HashMap
print "Foo"
escape

CLASS PUBLIC FOO
  METHOD PUBLIC FOO()
  METHODEND
  METHOD PUBLIC VOID bar()
  METHODEND
CLASSEND
```

The gray border at the left margin of the editor area may contain icons that flag errors, warnings, or problems the system detects. Icons also appear if you have created bookmarks, added breakpoints for debugging, or recorded notes in the Tasks view. You can view details for any icons in the left margin of the editor by hovering the mouse over them.

Sometimes you may need to use an external editor to modify a file in the workbench. This can occur if you don't have an editor registered for a certain file type, or if you wish to use a particular external editor to modify a file in the workbench even if you do have an internal editor registered for that file type.

Use an Internal Editor

1. Select the file in the Navigator.
2. Right click on the file and choose Open With > Other…
3. Check the "Internal editors" radio button (or "External programs" to select an external editor).



Choose the editor for opening Foo.bbj:
[✓ Internal editors] External programs

4. Select the Internal editor of your choice.
5. Click [OK].

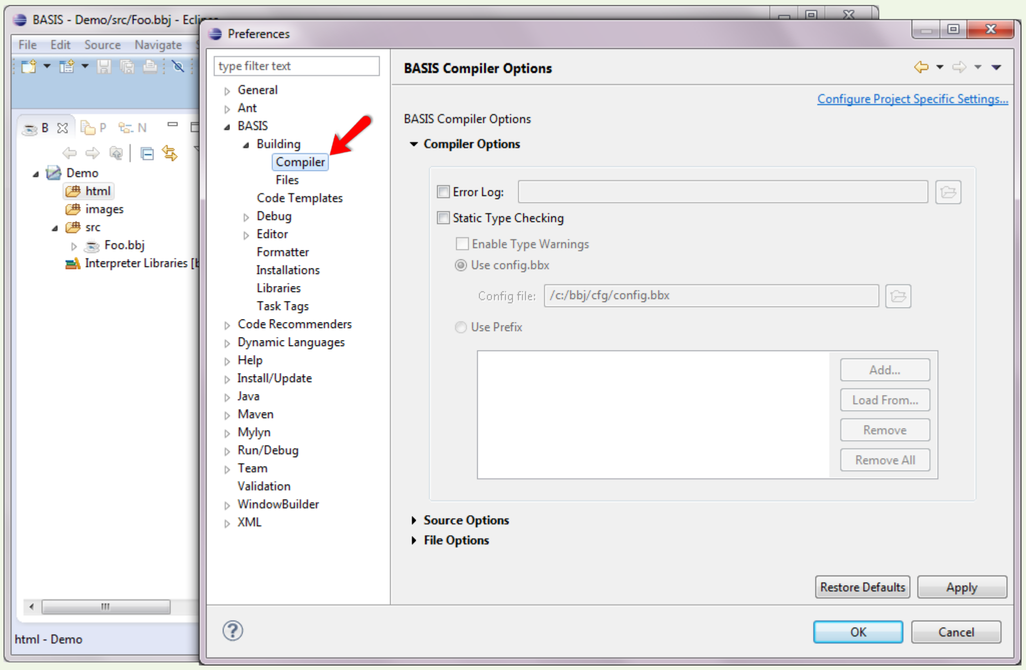
Back to BDT Components

Builder

The BDT Builder is what you use to build your BASIS projects. There are many options to support the kind of project build you'd like. Do you want to tokenize files? What options do you want to pass to the bbjcpl compiler? Where do you want your tokenized files to go? What kind of type checking do you want performed during the build?

We need to use the Eclipse Preferences dialogs to see how you can configure your Build options and Compile options for the project.

Configure Build and Compile
1. In Windows and Linux, select Window > Preferences; on the Mac, select the Eclipse > Preferences.
2. Under Preferences, find and expand the BASIS node (aka Preferences > BASIS).
3. Click on Building > Compiler.



Error Log: ☐
Static Type Checking ☐
Enable Type Warnings ☐
[✓ Use config.bbx]
Config file: /c:/bbj/cfg/config.bbx
Use Prefix

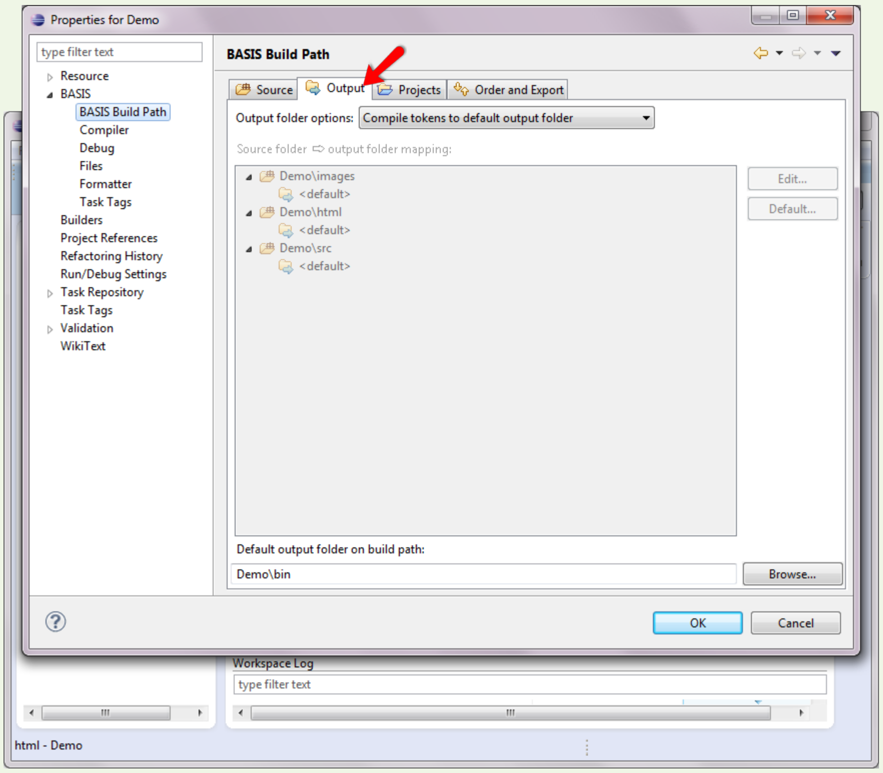
Set Compiler Options

Refer to the topic bbjcpl - BBj Compiler in the online help. In this dialog, we can set the default command line options to send to bbjcpl to build each file.

These are workspace-wide settings. If you prefer to override them and set project-specific settings for individual projects in your workspace, click the Configure Project Specific Settings... link in the top right corner of this dialog window.

There's more to it, however; you need to choose how to build the entire project.

Build the Project
1. Select Demo in the Navigator.
2. From the project menu at top, select Project > Properties > BASIS > Compiler. Here you see the same dialog as in previously appeared, giving you the option now to modify the default compiler options or set a project specific compiler options settings.
3. Select BASIS Build Path and then select the Output tab shown below.


Properties for Demo

type filter text
- Resource
- BASIS
  - BASIS Build Path
  - Compiler
  - Debug
  - Files
  - Formatter
  - Task Tags
- Builders
- Project References
- Refactoring History
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

BASIS Build Path

Source | Output | Projects | Order and Export

Output folder options: Compile tokens to default output folder

Source folder ⇨ output folder mapping:

- Demo\images
  - <default>
- Demo\html
  - <default>
- Demo\src
  - <default>

Edit... Default...

Default output folder on build path:

Demo\bin

Browse...

OK Cancel

Workspace Log

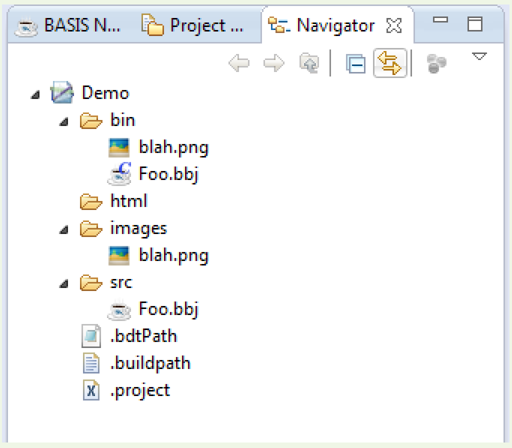
type filter text

html - Demo

We created our Demo project with the default project options, which include the option to Compile tokens to the default output folder. This default output folder “bin” does not appear in the BASIS Navigator, but this is where all the tokenized files are written, with respect to any subfolders they might appear in. Since we created the images and html subdirectory, you can see they are all set to also compile to default output folder. Non-source files will not be compiled of course; they will simply be copied into the bin folder.

It is possible to have multiple output folders as well, however, we will keep the defaults for this purpose.

Hint - if you would like to see the ‘hidden’ directories and files, go to the navigators section at left, select the Navigator tab. See the screenshot below of the project fully expanded to show all the parts. This navigator, commonly called the Resource Navigator, is very much like a directory listing view of your project.


BASIS N... | Project ... | Navigator

- Demo
  - bin
    - blah.png
    - Foo.bbj
  - html
  - images
    - blah.png
  - src
    - Foo.bbj
  - .bdtPath
  - .buildpath
  - .project

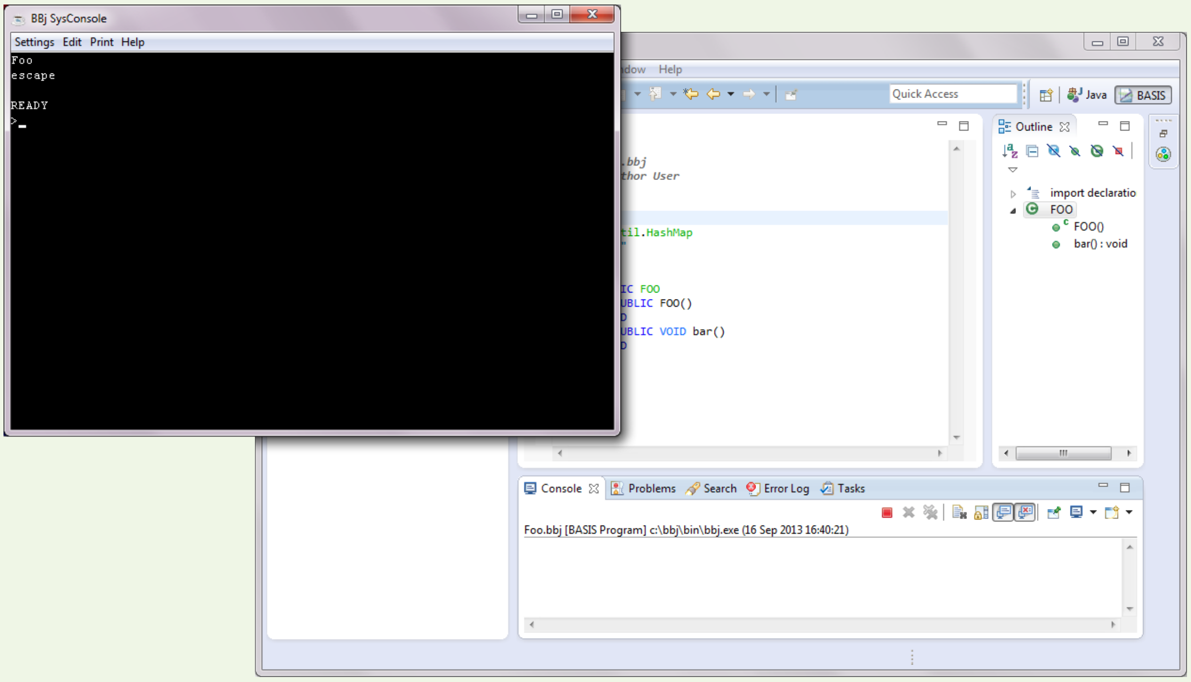
In this image, you can see the bin folder. You can see that your file Foo.bbj has been compiled into Foo.bbj - notice the little blue ‘C’ above and to the right of the file in the bin folder. This indicates a Compiled file. Notice also that our image file blah.png copied into the bin folder from its location in the images source folder.

Back to BDT Components

Executor

Execute a file in Eclipse through a Launch Configuration. To create a default launch configuration, follow the steps below.

1. Select Demo in the BASIS Navigator (not in the Resource Navigator) using the Run menu; select Run > Run As > BASIS Project to display what is shown below.



```
Foo
escape

READY
>_
```

```
Foo.bbj [BASIS Program] c:\bbj\bin\bbj.exe (16 Sep 2013 16:40:21)
```

2. Type **bye** at the Ready prompt in the BBj SysConsole to exit the program.

Of course, this was a very simple BBj program and project. Since we only have one source module in this project, BDT knows to create a Launch Configuration for that file only. If you have more than one source file, BDT will ask which file you want to run.

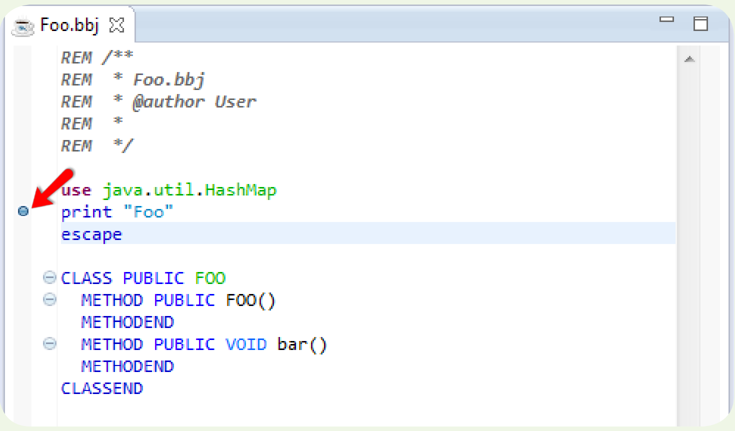
Back to BDT Components

Debugger

Debugging a file is launched the same way as running a file, through Launch Configurations. In fact, you can use the same launch configuration you created to Run > Run As > BASIS Program.

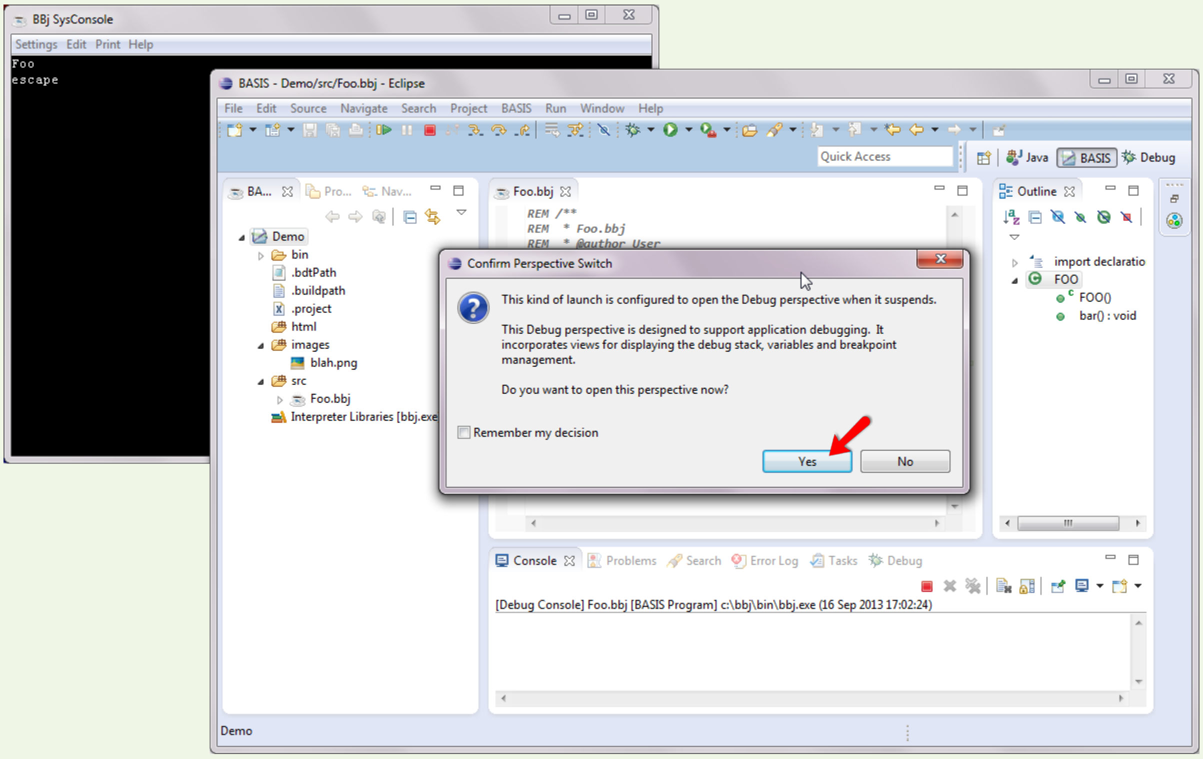
First, let's set a breakpoint on the line **`print "Foo"`** in Foo.bbj.

1. Move your mouse cursor into the gray column just to the left of that line and double-click. The small blue ball in that column indicates a breakpoint is set on that line.



```
REM /**
REM  * Foo.bbj
REM  * @author User
REM  *
REM  */

use java.util.HashMap
print "Foo"
escape

CLASS PUBLIC FOO
  METHOD PUBLIC FOO()
  METHODEND
  METHOD PUBLIC VOID bar()
  METHODEND
CLASSEND
```

2. Now select the project, Run > Debug As > BASIS Program.
3. Click [Yes] to confirm you want to switch to the Debug perspective.



```
Foo
escape
```

```
REM /**
REM  * Foo.bbj
REM  * @author User
```

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

Yes No

```
[Debug Console] Foo.bbj [BASIS Program] c:\bbj\bin\bbj.exe (16 Sep 2013 17:02:24)
```

You will then switch to the Debug perspective and see the line highlighted where the breakpoint hit.

4. Press [F8] to continue running.

Summary

The BASIS Development Tools, tightly integrated as a plug-in to the ever-popular Eclipse IDE, is leap years ahead of its NetBeans-based predecessor. Regardless of which platform you develop on, Eclipse looks good and offers a host of existing plug-ins and configurability options to optimize and streamline your development cycle. If you have not tried it yet, now is the time. Find out how truly powerful and integrated this development environment really is! 🟩

For additional assistance, refer to these resources

- Online Juno Help System
- Online Kepler Help System
- Eclipse IDE Tutorial
- Vogella Eclipse Tutorials
- IBM developerWorks
- Eclipse Tutorial on sourceforge
- Eclipse Resources