How often does documentation get pushed down to the bottom of the list, getting done in a rush after the new or modified code is finished, or perhaps not written completely, accurately, or ever?

Today's BBj® developer can now ***generate*** documentation quickly, easily, and accurately; and formatted in the familiar and modern look and feel of Oracle's Javadoc tool. Like Oracle's Javadoc tool, BASIS' BBjToJavadoc generates API documentation for BBj object-oriented code in HTML format, completely automatically from comments entered within the BBj source code. What a great benefit it is for object-oriented BASIS developers and Java Developers to be able to interact with both Java and BBj documentation presented almost identically!

By Brian Hipple
Quality Assurance Supervisor

BASIS now documents several of our APIs using this new BBjToJavadoc tool, which automatically updates the published documentation as we add, remove, or modify new classes, methods, and fields. For example, documentation for the new Email utility classes now joins BBJasper, Dialog Wizard, and BBjToJavadoc in the Javadoc-like format. In addition to generating more BBj documentation, the BBjToJavadoc utility also includes other very helpful and important enhancements. This article introduces these new BBj 14.0 enhancements that are now available in preview starting with 13.03.

Package Description

In Java, a "package" is the physical bundling of classes to logical units. In BBj, this traditionally works with the PREFIX and other filesystem-based concepts. When we previously documented BBj code, we set the package to "Unknown." However, we now use the notion of a package in the BBjToJavadoc utility to tag different BBj classes as a logical group to appear together in the documentation. The idea is that one of the source files in the package has package description comment before the package statement, whereas the rest of the modules just have the package statement.

The package statement, which takes the form "REM package [name]", has no programmatic function. However, it is a very powerful documentation statement as a single source directory can support more than one package. The Java paradigm does not allow this, but it fits very well into the BBj paradigm. The associated package Javadoc comment, located before the package statement, supports all Javadoc tags.

Figure 1 shows an example of the package statement and comment in BBj object-oriented source code and **Figure 2** shows the generated documentation.

```
rem /**
rem * Provides BBj access to email functionality
rem * @since 10.0
rem */
rem package Email

rem /** Email
rem * Allows for the sending of email messages, leaving the means by which the message
rem * is populated and sent up to the caller. Attachments are supported.
rem */
class public Email

rem /** EmailContact
rem * Used to keep track of email contact information including first and last names and email addresses
rem */
class public EmailContact

rem /** EmailDialog
rem * Displays a dialog which allows users to send emails and maintain email settings and contacts
rem */
class public EmailDialog
```

Figure 1. Package statement in BBj object-oriented source code with Javadoc comments

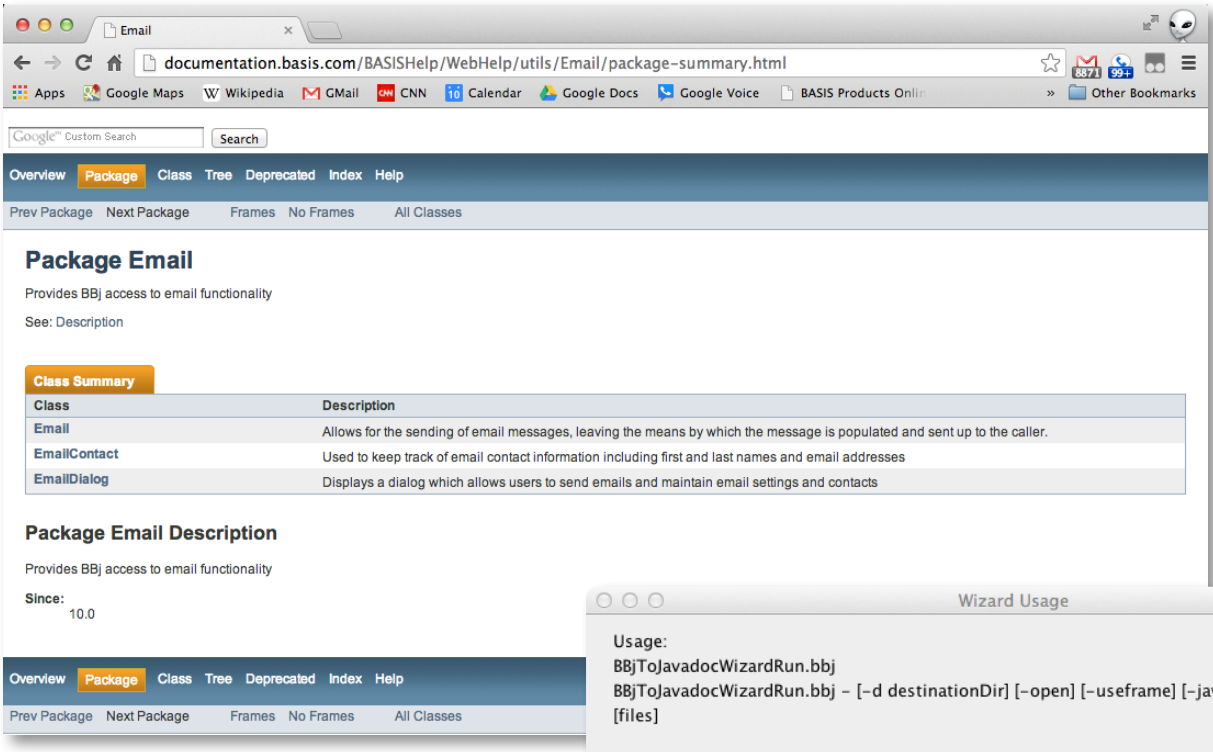


Figure 2. Generated Javadoc documentation for the Email package

Figure 3. BBjToJavadoc usage showing new command line arguments

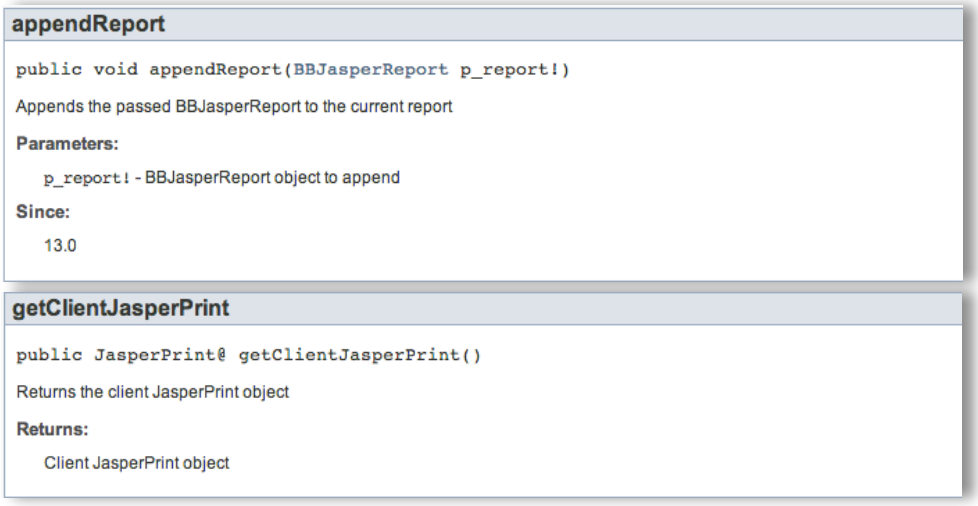
Command Line BBjToJavadoc

One of the most useful enhancements is the ability to run the BBjToJavadoc wizard from the command line. See **Figure 3** for an example of executing the utility with optional command line parameters.

In earlier revisions, this utility could only execute as a GUI wizard. BASIS now incorporates the calling of the command line BBjToJavadoc in its continuous build system to generate the BBj utility documentation. This includes the Email, BBjToJavadoc, DialogWizard, and BBJasper utilities. Anytime a developer checks in a source code change to the SVN archive for one of these utilities, the BASIS build script automatically invokes the BBjToJavadoc utility that generates the updated documentation.

Increased Performance

BASIS engineers devoted much effort to increase the processing speed dramatically in the latest version of the BBjToJavadoc utility. What once took minutes to generate documentation for a large number of source files now measures in mere seconds. In previous versions, the resultant documentation stripped such symbols as the "!" mark and the "@"symbol. BBj now fully supports such concepts as using the ! mark as a suffix for object variable names and the @ symbol in the declaration of client variables. These symbols now show up in the documentation to allow for easy distinction of BBj server and client objects as **Figure 4** shows.



appendReport

```
public void appendReport(BBJasperReport p_report!)
```

Appends the passed BBJasperReport to the current report

Parameters:

p_report! - BBJasperReport object to append

Since:

13.0

getClientJasperPrint

```
public JasperPrint@ getClientJasperPrint()
```

Returns the client JasperPrint object

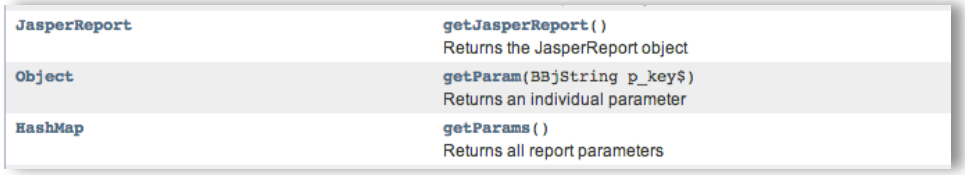
Returns:

Client JasperPrint object

Figure 4. Documentation that includes a BBj server object (top) and client object (bottom)

New Linkage

BASIS now automatically includes linkage to the Java API documentation. When a BBj program uses a Java object as a return value, a parameter to a method, or declares it as a field or variable, BBjToJavadoc creates the links to the associated Java documentation. To establish linkage to a Javadoc set that isn't part of the Java API, add the **`-javadocargs -link`** arguments to instruct the utility to link that documentation to the generated BBj source documentation. BASIS uses this capability in **Figure 5** to link to the Jasper documentation when creating the documentation of the BBJasper utility.



`JasperReport`	`getJasperReport()` Returns the JasperReport object
`Object`	`getParam(BBjString p_key$)` Returns an individual parameter
`HashMap`	`getParams()` Returns all report parameters

Figure 5. Example of the linkage of Java andJasper class documentation

Additionally, developers can add **`@see`** in the source code to embed an example in the file as shown in **Figure 6**.

```
rem /**
rem * Creates a JasperReport to view, print, or save to file in various formats.<p>
rem * To determine which iReport version is supported in BBj, check the version number that is included
rem * in the name of the <bbj install dir>/lib/jasperreports*.jar. For example, jasperreports-5.1.0.jar
rem * means that this installed version of BBj is compatible with the 5.1.0 version of iReport
rem * @since 9.0
rem * @see <a href="BBJasperReportExample.bbj">Example</a>
rem */
class public BBJasperReport
```

Figure 6. Using the **`@see`** tag to include example source code

Google Search

The Javadoc documentation published on the BASIS website now includes our familiar Google search capabilities. This search option appears by simply adding the appropriate header to the documentation, giving the user the ability to type in keywords to search all of the BASIS documentation. **Figure 7** shows the result of a search on "bbjasperreport."

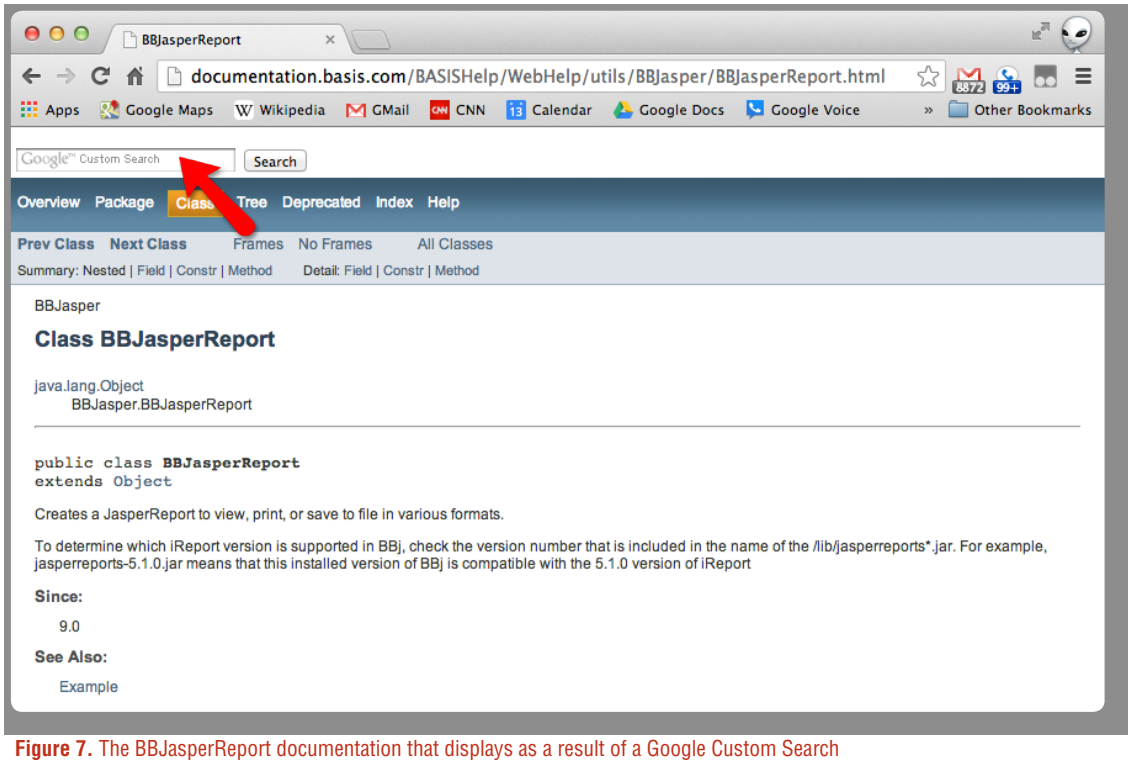


Figure 7. The BBJasperReport documentation that displays as a result of a Google Custom Search

BASIS also provides a filter to specify the BBjToJavadoc documentation from the rest of the search filters. **Figure 8** shows the result of this selection.

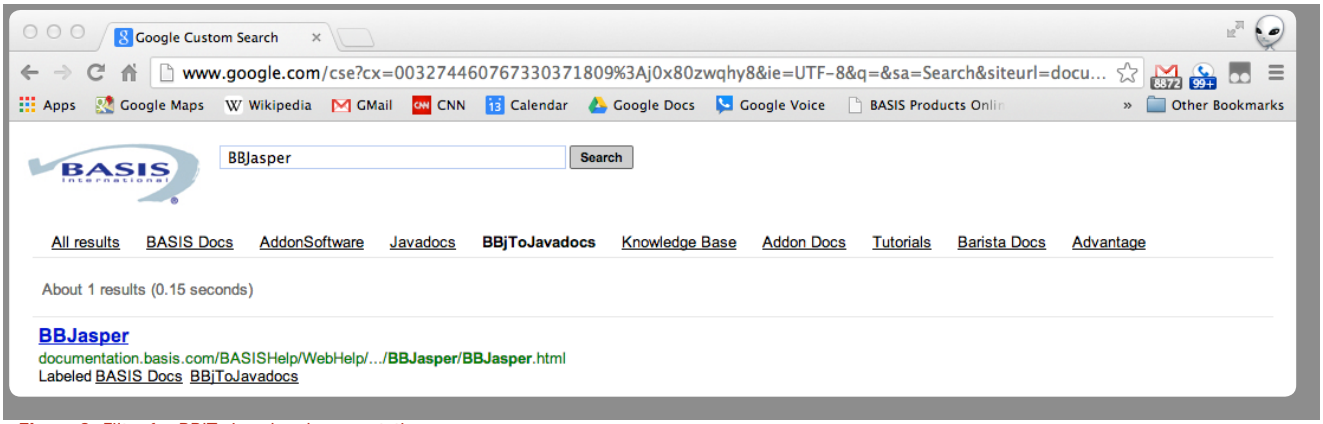


Figure 8. Filter for BBjToJavadoc documentation

Summary

The BBjToJavadoc utility enhancements are great examples of how BASIS goes the extra mile to deliver a more and more useful and efficient tool. This utility allows developers to add documentation as they write their BBj object-oriented code with the code fresh in their minds, resulting in much more accurate and complete documentation. After all, this is far better than writing it after the fact, or even worse, never doing it at all. As most will tell you, it is very hard to maintain or add functionality to poorly documented code. Now, you can *generate* your own documentation, easily and quickly, directly from the code and without the need to format or stylize it, or learn another tool. Your product is now more complete – documented accurately and easily synced with the source code. ■ links.basis.com/13code

For more information, read BBjDocsGenerator User's Guide