Calling Custom Java from BBj

BBj® is two languages in one. It combines the business focus and ease-of-use of Business BASIC with the power of Java. In *Quick and Easy Solutions With Free Java Libraries* (Part 1 and Part 2), we showed how to find and integrate third party Java components into your BBj application. This article shows how easy it is to integrate your own custom Java modules into BBj.

Standard Java Libraries

BBj developers automatically have access to everything in the standard Java API. This effectively extends the BBj language to include everything that you can do in Java, as shown in **Figure 1**.

```
>map! = new java.util.HashMap()

>map!.put("a",1)

>map!.put("b",2)

>map!.put("c",3)

>print map!.get("b")

2

>sqllist! = sqllist(0)

>list! = java.util.Arrays.asList(sqllist!.split($0a$))

>java.util.Collections.sort(list!)

>print list!

[AddonDemoData, Barista, CD-Store, ChileCompany]

>java.awt.Desktop@.getDesktop().browse(new

java.net.URI@("http","basis.com",""))
```

Figure 1. Example of BBj utilizing Java functionality

We particularly recommend reviewing the Java Collections Framework, a sophisticated set of data structures that includes various kinds of maps, lists, and sets.

Third Party Java Libraries

A vast universe of Java libraries is freely available online. In a recent Advantage article, Quick and Easy Solutions With Free Java Libraries, we described how to integrate the Apache POI library to manipulate Microsoft Office files directly from BBj. POI is just one of the dozens of libraries available from the Apache Software Foundation. Google also develops useful Java libraries and makes them available for download. The Google Collections Library, for example, can be thought of as an extension of the standard Java Collections Framework.

Custom Java Libraries

You might decide that your BBj application can be improved by integrating a Java routine that is not available online, or that must be customized for your needs. This article cannot teach you how to program in Java (for that, see Bruce Eckel's excellent book, *Thinking in Java*). But we can describe how to easily package your custom Java modules so they can be used from BBj. We will do this in seven easy steps:

STEP 1 – Pick a package name.

STEP 2 – Make a directory structure.

STEP 3 – Write the Java program(s).

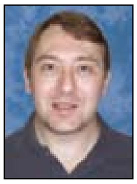
STEP 4 – Compile and test the Java program(s).

STEP 5 – Make a jar file.

STEP 6 – Add the jar file to the BBj default Classpath.

STEP 7 – Test the Java class in BBj.

>>



By Jim Douglas
Software Engineer

STEP 1 – Pick a package name

As you can see in the samples above, Java classes have a hierarchical naming structure, with pieces of the name separated by dots. For example, standard Java classes have names like **`java.net.URI`** and **`java.util.HashMap`**. Names form a tree structure – there are several packages under java.* and there are several classes under **`java.net.*`** and **`java.util.*`**. These tree structures can be as complex as you like, but at the top level, the convention is to start with your company's domain name because it is guaranteed to be unique. For this example, assume that you work for Acme Corporation and your registered domain is acme.com. The Java convention is to flip that around, so we will define our custom Java modules as **`com.acme.*`**. Our sample will be a utility module, so the complete package is **`com.acme.util`**.

STEP 2 – Make a directory structure

Once we have settled on our package name, we will need to make a directory tree that mirrors the package structure. From the command line, you might do something like this:

```
[C:\work]mkdir com com\acme com\acme\util

[C:\work]cd com\acme\util

[C:\work\com\acme\util]
```

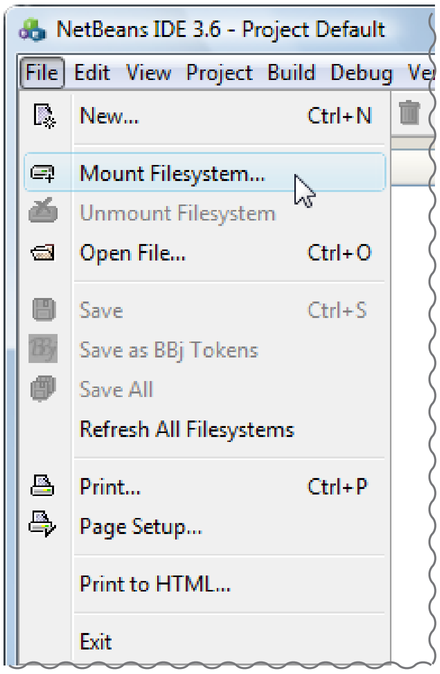


Figure 2. Mount a directory from the IDE

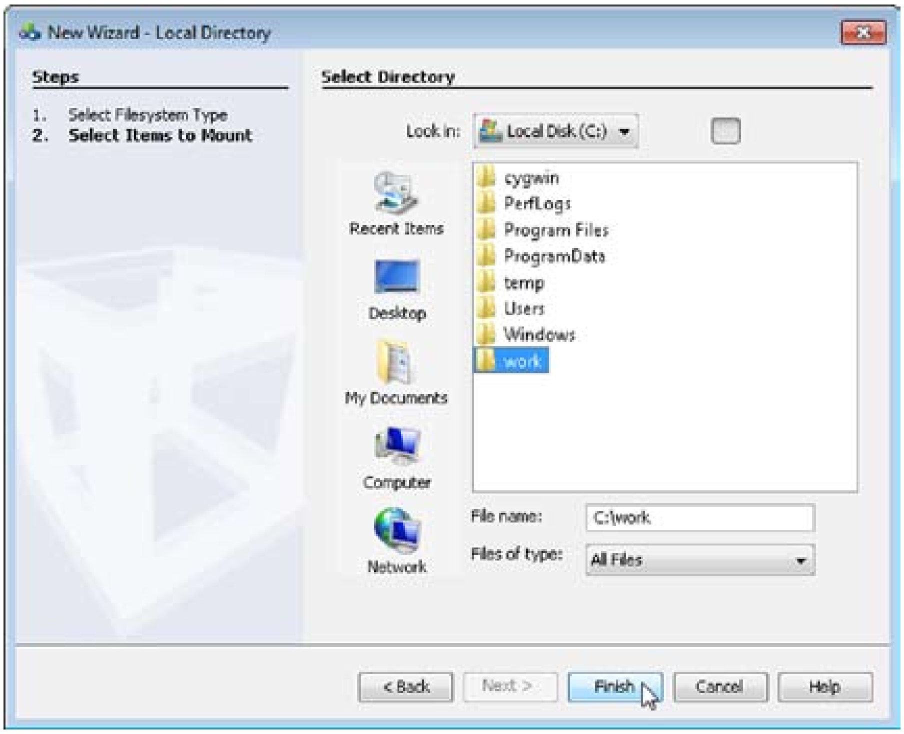
Or run the BASIS IDE and select **`File > Mount Filesystem`**… from the menu (see **Figure 2**).

Next, select a working directory and click [Finish] as shown in **Figure 3**.

Right-click on the working directory, select **`New Folder`**, and enter the first part of your package name (**`com`**) shown in **Figure 4**.

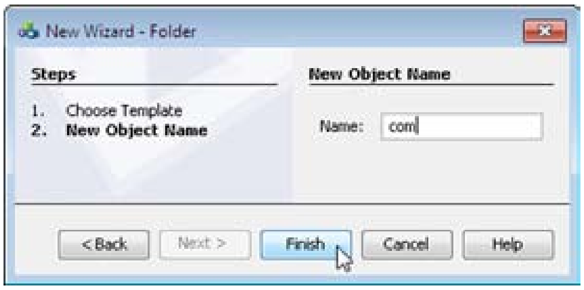
Repeat this process for the next two parts of the package name (right-click on **`com`** to create the acme folder, then right-click on acme to create the util folder).

Now that we created the package directory structure, right-click on **`util`** and select New Java Main Class. Enter the Java class name **Soundex** (**Figure 5**) and click [Finish]. **>>**



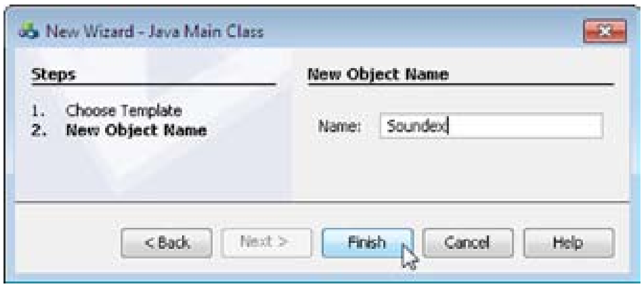
File name: C:\work
Files of type: All Files

Figure 3. Select working directory



Name: com

Figure 4. Enter the package name



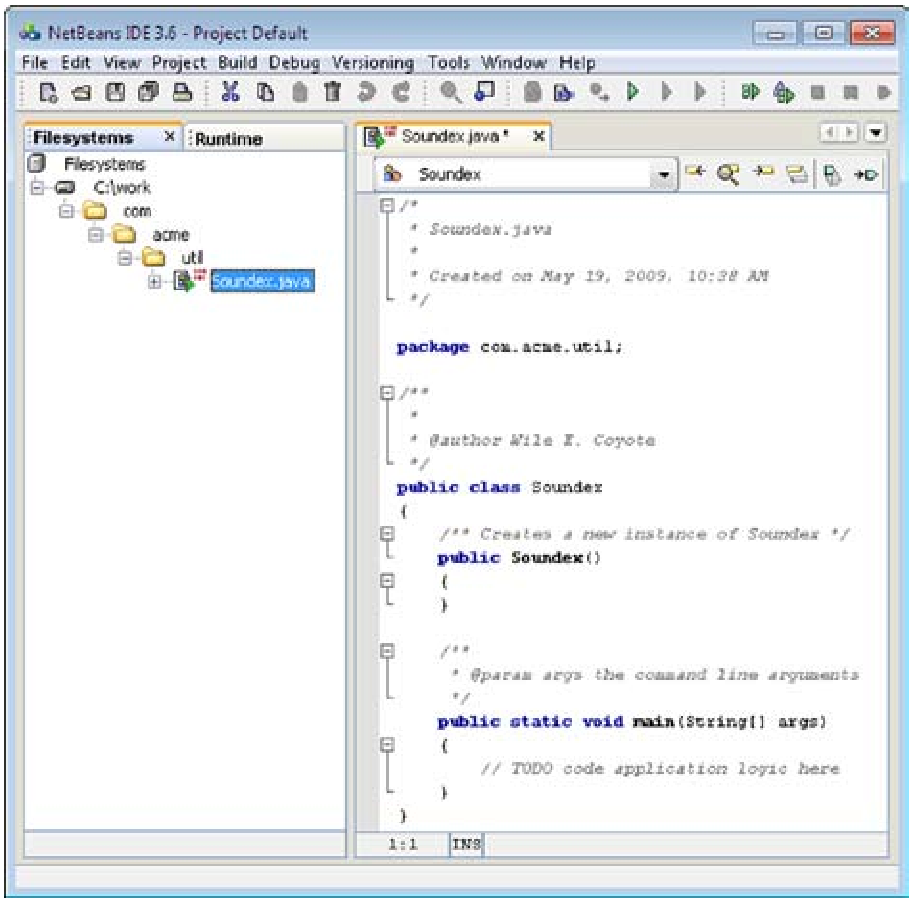
Name: Soundex

Figure 5. Name the Java class

The IDE generates a skeleton for **Soundex.java** as shown in **Figure 6.** Now we just have to fill in the details.

STEP 3 – Write the Java program(s)
Now that we have our package structure, we can implement our Java programs. Our sample program calculates the Soundex code, a commonly used way to index names for retrieval based on a "sounds like" algorithm. We will implement this in a file called **com/acme/util/Soundex.java**. The program is structured like the code sample in **Figure 7**.

The program starts with a package statement that corresponds to the subdirectory. It contains a public class with the same name as the Java program file (in this case, **Soundex**), and within that class it defines a function that we will use in our BBj program. Finally, it includes an (optional) **main** function that can be used for command-line testing.


NetBeans IDE 3.6 - Project Default
File Edit View Project Build Debug Versioning Tools Window Help
Filesystems
Runtime
Filesystems
C:\work
com
acme
util
Soundex.java
Soundex.java *
Soundex
1:1
INS

```
/*
 * Soundex.java
 *
 * Created on May 19, 2009, 10:38 AM
 */

package com.acme.util;

/**
 *
 * @author Wile E. Coyote
 */
public class Soundex
{
    /** Creates a new instance of Soundex */
    public Soundex()
    {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        // TODO code application logic here
    }
}
```

Figure 6. The generated code

```
package com.acme.util;
public class Soundex
{
    public static String soundex(String s)
    {
        // this is a stub
        return s;
    }

    public static void main(String[] args)
    {
        for (int i=0; i<args.length; ++i)
        {
            System.out.print(args[i]);
            System.out.print(": ");
            System.out.println(soundex(args[i]));
        }
    }
}
```

Figure 7. Sample Soundex structure

STEP 4 – Compile and test the Java program(s)
After filling in the details, right-click on Soundex.java and select **Compile**. When it compiles without errors, right-click on it again and select **Execute** to test it. We set it up to prompt for a name and display the calculated Soundex code. The test continues until you press [Enter] on a blank line as shown in **Figure 8**.

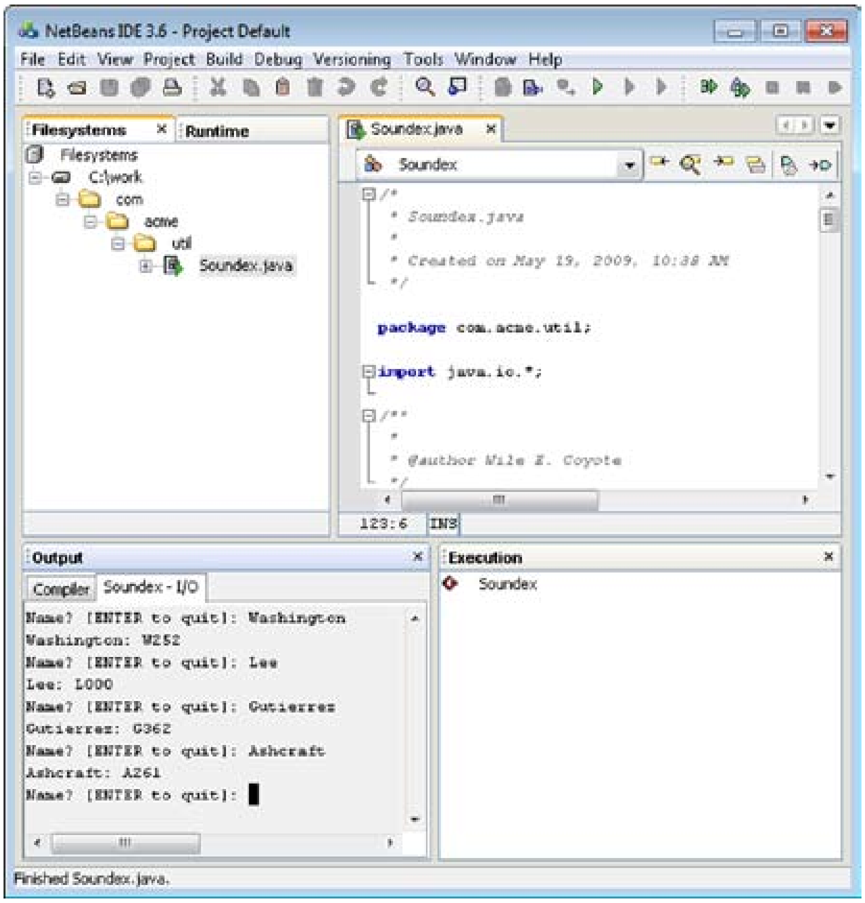
STEP 5 – Make a jar file
Java modules are deployed in Java Archive files (ZIP files with some added components). The next step is to create a jar file containing our Java class(es). Select **File > New** from the menu, then select **JAR Archives > JAR Recipe** (**Figure 9**), and click [Next>].

Enter the name **Soundex** and click [Next>] as shown in **Figure 10**.

Change the Generated JAR Location to the BASIS library directory (**C:\Program Files\basis\lib\**), as shown in **Figure 11**.

Add the com directory to the JAR Recipe as shown in **Figure 12**, then click [Next>].

Click [Finish], then right-click on the generated JAR Recipe file (Soundex.jarContent) and select [Compile]. This generates the Soundex.jar file as **C:\Program Files\basis\lib\Soundex.jar.** >>



NetBeans IDE 3.6 - Project Default
File Edit View Project Build Debug Versioning Tools Window Help
Filesystems
Runtime
Soundex.java
Output
Compiler
Soundex - I/O
Execution

Filesystems: C:\work > com > acme > util > Soundex.java

```
/*
 * Soundex.java
 *
 * Created on May 19, 2009, 10:38 AM
 */

package com.acme.util;

import java.io.*;

/**
 *
 * @author Wile E. Coyote
 */
```

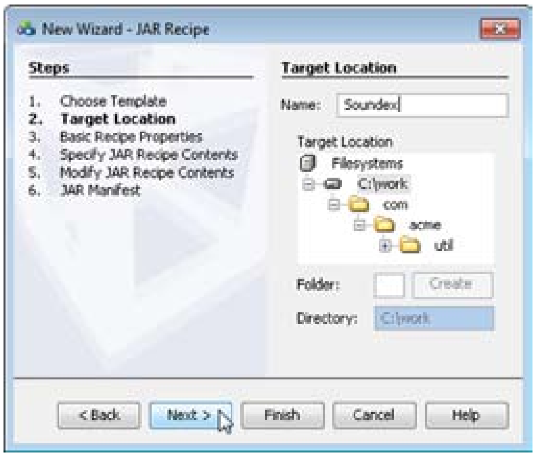
123:6 INS

```
Name? [ENTER to quit]: Washington
Washington: W252
Name? [ENTER to quit]: Lee
Lee: L000
Name? [ENTER to quit]: Gutierrez
Gutierrez: G362
Name? [ENTER to quit]: Ashcraft
Ashcraft: A261
Name? [ENTER to quit]:
```

Execution: Soundex

Finished Soundex.java.

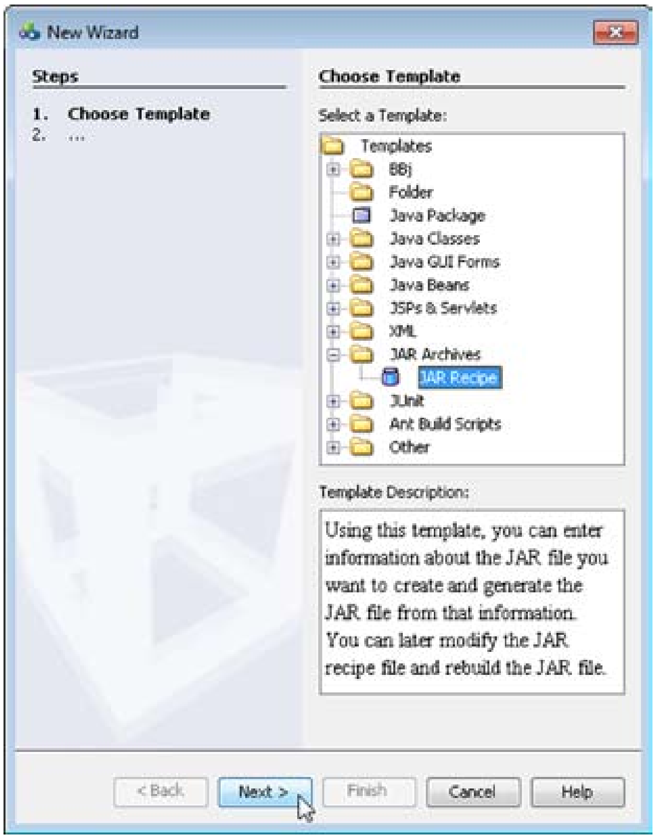
Figure 8. Press [Enter] to halt the test (back to text link)



Steps
1. Choose Template
2. **Target Location**
3. Basic Recipe Properties
4. Specify JAR Recipe Contents
5. Modify JAR Recipe Contents
6. JAR Manifest

Target Location
Name: Soundex
Target Location: Filesystems > C:\work > com > acme > util
Folder:
Directory: C:\work

Figure 10. Enter the JAR name (back to text link)

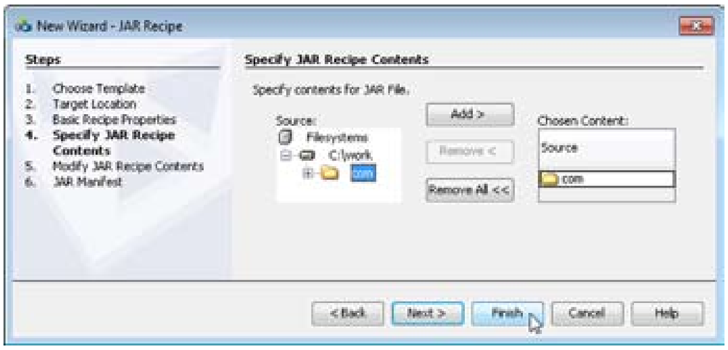


Steps
1. **Choose Template**
2. ...

Choose Template
Select a Template:
Templates
- BBj
- Folder
- Java Package
- Java Classes
- Java GUI Forms
- Java Beans
- JSPs & Servlets
- XML
- JAR Archives
  - JAR Recipe
- JUnit
- Ant Build Scripts
- Other

Template Description:
Using this template, you can enter information about the JAR file you want to create and generate the JAR file from that information. You can later modify the JAR recipe file and rebuild the JAR file.

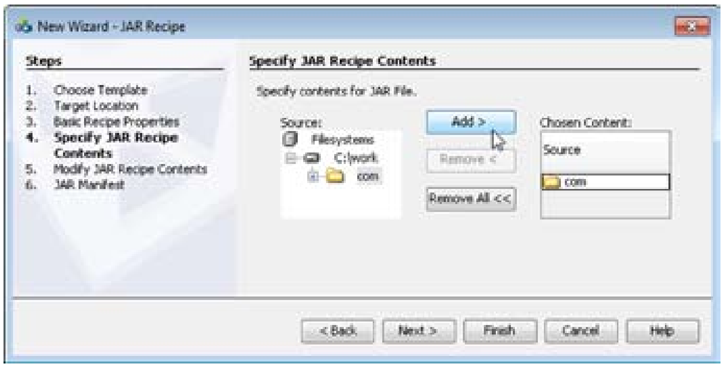
Figure 9. Select JAR Recipe (back to text link)



Steps
1. Choose Template
2. Target Location
3. Basic Recipe Properties
4. **Specify JAR Recipe Contents**
5. Modify JAR Recipe Contents
6. JAR Manifest

Specify JAR Recipe Contents
Specify contents for JAR File.
Source: Filesystems > C:\work > com
Chosen Content:
Source
com

Figure 11. Change the Generated JAR Location (back to text link)



Steps
1. Choose Template
2. Target Location
3. Basic Recipe Properties
4. **Specify JAR Recipe Contents**
5. Modify JAR Recipe Contents
6. JAR Manifest

Specify JAR Recipe Contents
Specify contents for JAR File.
Source: Filesystems > C:\work > com
Chosen Content:
Source
com

Figure 12. Add the directory to the JAR Recipe (back to text link)

>>

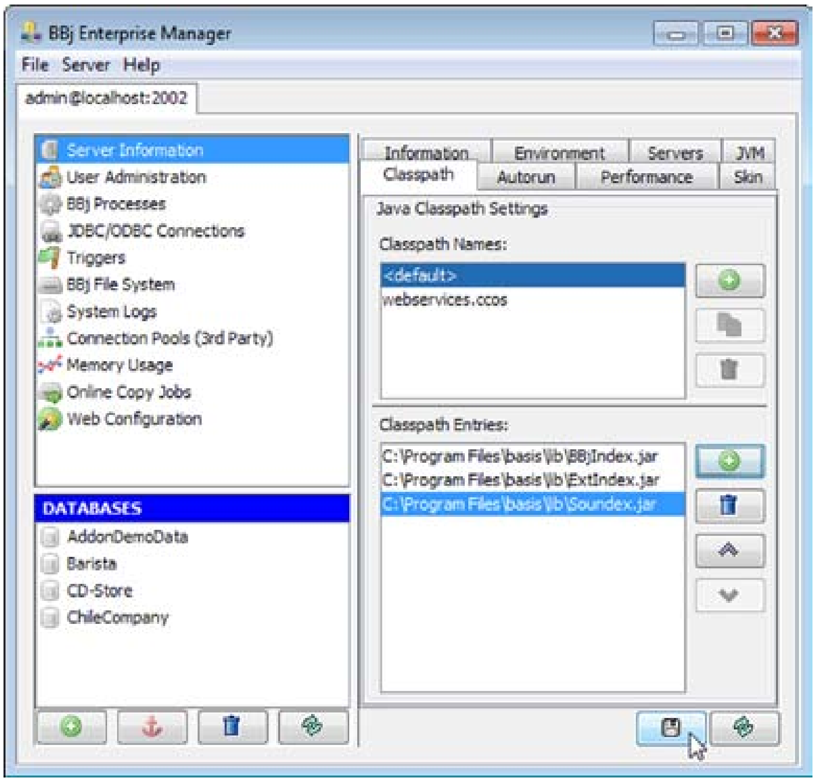
STEP 6 – Add the jar file to the BBj default classpath

It is possible to dynamically load Java classes using a *Session-specific Classpaths* as described in this issue. But for this sample, we will add the jar file to the BBj default classpath using Enterprise Manager, Start BBj Enterprise Manager (or load the EM module in the BASIS IDE from the **View > BASIS Server/Database Configuration** option). In Enterprise Manager, select the Classpath tab, then click the button next to the <default> Classpath Entries list as shown in **Figure 13**.

Highlight Soundex.jar in the list shown in **Figure 14** and click the [Select] button:

Click the button at the bottom to save the change, then close BBj Enterprise Manager.

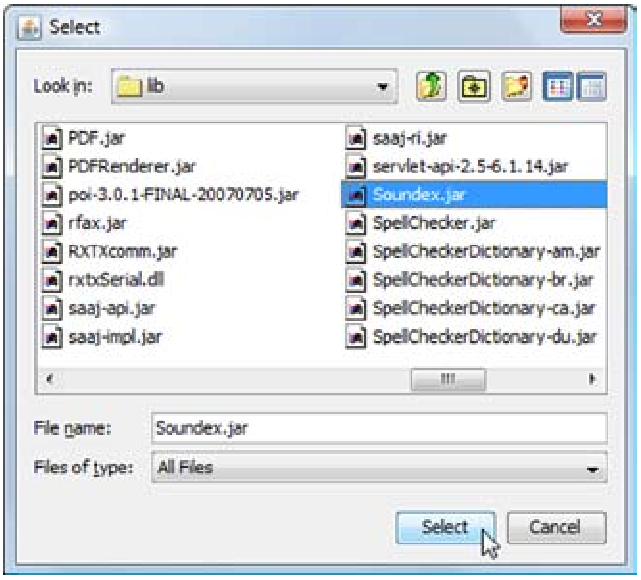
Beginning in BBj 9, BBj Services no longer needs to be restarted to recognize classpath changes.



<default>
webservices.ccos

Classpath Entries:
C:\Program Files\basis\lib\BBjIndex.jar
C:\Program Files\basis\lib\ExtIndex.jar
C:\Program Files\basis\lib\Soundex.jar

Figure 13. Adding a JAR to the default classpath


Select
Look in:
lib
PDF.jar
PDFRenderer.jar
poi-3.0.1-FINAL-20070705.jar
rfax.jar
RXTXcomm.jar
rxtxSerial.dll
saaj-api.jar
saaj-impl.jar
saaj-ri.jar
servlet-api-2.5-6.1.14.jar
Soundex.jar
SpellChecker.jar
SpellCheckerDictionary-am.jar
SpellCheckerDictionary-br.jar
SpellCheckerDictionary-ca.jar
SpellCheckerDictionary-du.jar
Select
Cancel

File name: Soundex.jar
Files of type: All Files

Figure 14. Select the Soundex.jar

>>

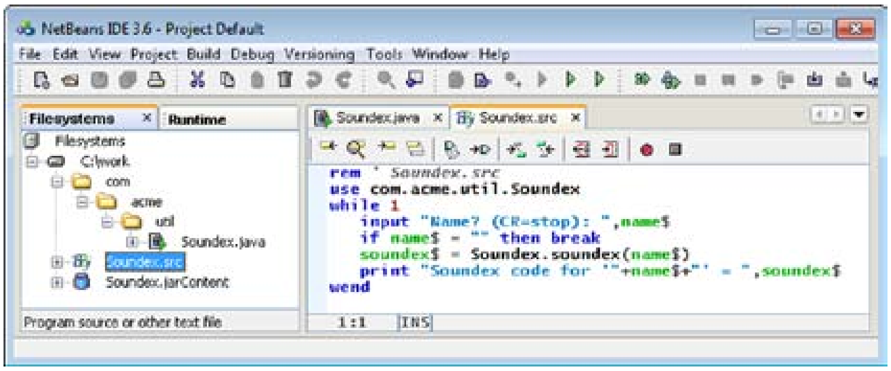
STEP 7 – Test the Java class in BBj
Write a BBj program (**`Soundex.src`**) to test this new Java class (see **Figure 15**).

When **`Soundex.src`** is running, it looks like **Figure 16**.

The above test uses names from the U.S. National Archives Soundex information.

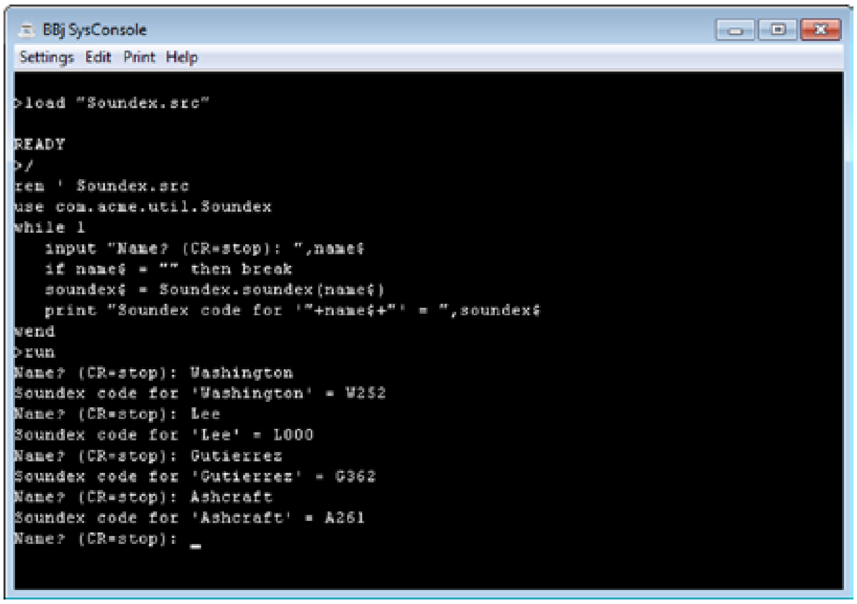
Summary

The ability to use Java code within BBj dramatically extends the power and flexibility available to you as a BBj developer. This article has shown how to quickly and easily incorporate custom Java modules into your applications. ■


NetBeans IDE 3.6 - Project Default
File Edit View Project Build Debug Versioning Tools Window Help
Filesystems
Runtime
Filesystems
C:\work
com
acme
util
Soundex.java
Soundex.src
Soundex.jarContent
Soundex.java
Soundex.src
Program source or other text file
1:1
INS

```
rem ' Soundex.src
use com.acme.util.Soundex
while 1
   input "Name? (CR=stop): ",name$
   if name$ = "" then break
   soundex$ = Soundex.soundex(name$)
   print "Soundex code for '"+name$+"' = ",soundex$
wend
```

Figure 15. A BBj program to test the Soundex Java class



```
>load "Soundex.src"

READY
>/
rem ' Soundex.src
use com.acme.util.Soundex
while 1
   input "Name? (CR=stop): ",name$
   if name$ = "" then break
   soundex$ = Soundex.soundex(name$)
   print "Soundex code for '"+name$+"' = ",soundex$
wend
>run
Name? (CR=stop): Washington
Soundex code for 'Washington' = W252
Name? (CR=stop): Lee
Soundex code for 'Lee' = L000
Name? (CR=stop): Gutierrez
Soundex code for 'Gutierrez' = G362
Name? (CR=stop): Ashcraft
Soundex code for 'Ashcraft' = A261
Name? (CR=stop): _
```

Figure 16. **`Soundex.src`** running

See Also

Calling Custom Java From BBj

Inheriting Java Types

Download samples shown in this article from
www.basis.com/advantage/mag-v13n1//customjava.zip