Chapter 19.

July 2008

Welcome to the July 2008 edition of IBM Information Server Developer's Notebook. This month we answer the question;

> Part II of: How do I make use of some of the extensibility of IBM Information Server; more specifically, How do I link in C/C++, Java, and other type program code into my DataStage component Jobs?
>
> *Excellent question! Last month we answered the overview, C/C++, and BASIC language portions of this question. This month we give focus to Java. Java is especially powerful (over say C/C++ and BASIC), because of all of the free code and capabilities you get with the built in Java object hierarchy.*

In short, we are going discuss all of the relevant terms and technologies above, and provide examples that detail how to deliver this functionality using IBM Information Server.

Software versions

All of these solutions were *developed and tested* on IBM Information Server (IIS) version 8.01, FixPak 1A, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server components.

IBM Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM Information Server product contains the following major components;

> WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

19.1 Terms and core concepts

As stated above, this edition of IBM Information Server Developer's Notebook (IISDN) is a continuation from the prior month. Last month we introduced extensibility within IBM Information Server (IIS); more specifically, linking in new functionality to the DataStage component of IIS with C/C++ code, BASIC code, and more. We also reviewed 3 or more of the prior editions of IISDN which covered XSLT, Web services invocation, operating system commands and utilities, and more.

This month we give focus to running custom Java programs and Java code fragments within the DataStage component to IIS, and Java is huge. In part, Java is huge because it comes with so much free stuff. Consider the following;

- A given IBM Information Server customer wanted to have DataStage send and receive messages to a non-standard (non-supported) messaging server. That non-standard messaging server had a seldom used Java/JMS (Java Messaging Service) interface.

  By finding and modifying 40-60 lines of Java code we identified through Google.com, and linking that Java code into DataStage, we were able to accomplish our goal.

- Another IBM Information Server (IIS) customer had previously created a home grown version of something similar to the DataStage component of IIS, and their legacy system just was not scaling (it was not performing adequately once it hit certain moderately sized data volumes); as good as their efforts were, they had not created a supporting parallel framework as capable as IIS. But, this customer had created a good volume of intellectual property, custom and now strategic algorithms to mark/score and evaluate their data.

  Luckily, everything this customer had created was written in Java. We extracted the strategic bits of their legacy application, and made them available to IIS and DataStage through much of the technology discussed in this document.

IIS, DataStage, and Java

The version of IBM Information Server (IIS) discussed in this document (8.01FP1A), ships with an embedded version 1.4.2 of a Java Runtime Environment (JRE). A JRE supplies a Java Virtual Machine (JVM) and other elements of Java necessary to run Java programs and Java code fragments. So from the run time aspect of this equation, you are set; every program resource is in place. What remains to be done before you can run Java code inside IIS includes;

- Make a few Java specific settings within the IIS `dsenv` file.

Because this is the dsenv file, which includes environment variables, you will have to recycle the DSEgine. (Environment variables famously require a server reset; any server reset.)

- Write and compile your Java program or Java code fragment.
- Create a DataStage Job, which has one or more Stages using the Java assets written and compiled above.

Figure19-1 displays the relationships between the various Java related acronyms listed above.

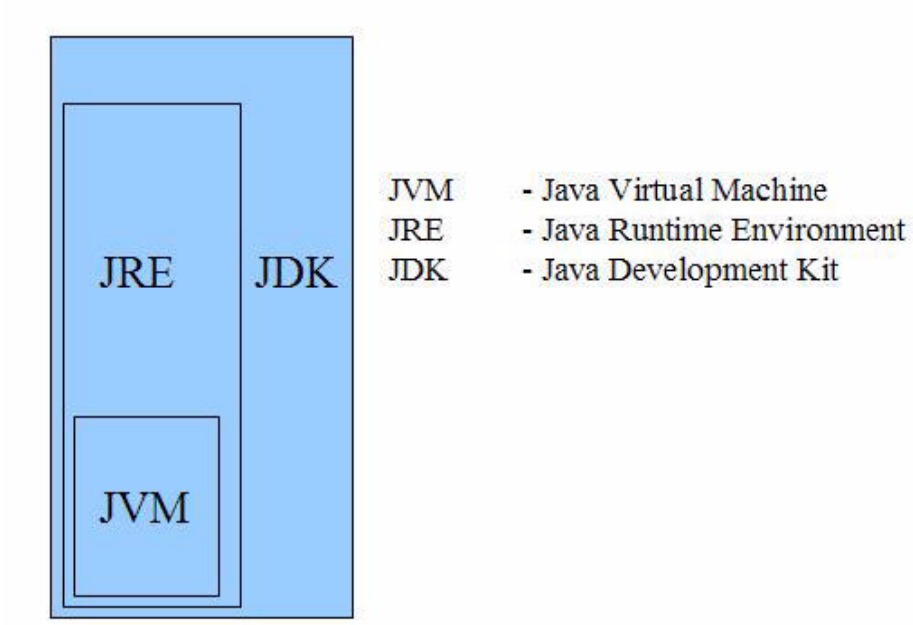

Figure 19-1 Relationship of Java acronyms.

Regarding Figure19-1, the following is offered;

- The Java Virtual Machine (JVM) is the actual agent that executes your Java program or Java code fragment. The binary program name of the JVM is java (Linux/Unix), or java.exe (Windows). Because the JVM is ported to, and made standard across the various operating systems, a Java program is supposed to offer, write (and compile) once, run anywhere. And, largely this is true.

- The Java Runtime Environment (JRE) includes the JVM listed above, and the Java object hierarchy; in effect, all of the free buttons, widgets, and hundreds of other useful program objects which you can re-use.

  As stated above, IBM Information Server (IIS) includes an embedded JRE. IIS 8.01 FP1A includes a version 1.4.2 JRE.

  If you have a compiled Java program or Java code fragment, you are pre-installed to be able to run it inside IIS. (Pre-installed, but not entirely pre-configured. See far below.)

- A Java Development Kit (JDK) allows one to compile Java source code into something that the JVM can run. Simple and free JDKs are available from Sun Microsystems (Sun) and others. The default and free JDK from Sun allows for a command line compile only. If you wish a graphical developer's workbench, you need to go elsewhere.

  Java source code files have a ".java" file name suffix.

  Compiled Java code has a ".class" file name suffix.

- Not listed above-

  You can deploy your compiled Java code as Class files, Jar files, and more. JAR files are basically Zip files. There are more rigidly defined and more advanced versions of JARs, called WARs and EARs; basically these are JARs with a rigidly defined table of contents and related.

  With IBM Information Server, you will most commonly use Class files and JAR files. WARs and EARs are expected to run elsewhere (inside a Java/J2EE compliant application server).

  CLASSPATH is an environment variable that points to a single or set of directories (full pathname) that contain your target Class files, JARs, and so on.

IIS, DataStage, and the (IIS/DS) Java Pack

While you can use any Java Development Kit (JDK) to write and compile your Java program and Java code fragments, you need a Java accessible API (programming interface) in order to have that asset speak with IBM Information Server (IIS). In its physical form, that API is the IIS DataStage component Java Pack. Simply, the Java Pack is a single Java JAR file, with an associated definition of various objects and methods that allow your custom Java source code to interact with IIS DataStage component Jobs.

Note: Where the IBM Information Server installation directory is,

/opt/IBM/InformationServer

The single public facing Java Pack JAR file you need to include for Java compilation is located in,

/opt/IBM/InformationServer/Server/DSEngine/java/lib/tr4j.jar

And dsenv is located in,

/opt/IBM/InformationServer/Server/DSEngine/dsenv

A single PDF file on just the Java Pack accompanies the full IIS product documentation. The current file name is, i46dejva.pdf. Really this document gives you everything you need, complete with numerous Java program code examples.

In general, the following is offered;

- There are two IBM Information Server (IIS) DataStage component, Parallel Canvas Stage types you use when accessing Java programs and Java code fragments;
  - Java Transformer Stage

    Resembles the Parallel Transformer we commonly use in DataStage jobs; generally these Stages sit in the middle of an input data stream.

    This is the example we create in this document.
  - Java Client Stage

    Generally act as the end points of a data stream; either generating the initial input, or receiving the final output. A Java Pack Client Stage can also satisfy the Reference Link to other standard Stages like Lookup.
- Both Stages above offer a Java object for the incoming and outgoing data records, and columns, and also the Job Properties, and more.
- Both Stages have Java methods for an on every row event, as well as start up and shut down. Its very complete and straight forward.
- Both Stages are located in the Palette view, Real Time drawer, of the DataStage Designer program.

In the example that follows, we create a simple IIS DataStage component Job that uses a Java Transformer; our classic and simple example, Add Two Numbers.

19.2 Creating the example(s) outlined above

In this section of this document, we are going to create a single IBM Information Server (IIS) DataStage component Job which invokes custom end user created Java program code, through use of the Java Pack, Java Transformer Stage.

Perform the following;

1. Create a sample input data source.

   We used our favorite program editor (Vi), and created a 4 column ASCII text file. Columns 1 and 2 of this file contain Integer values, so that we may add these values later with our Java program code.

2. Find and modify the IBM Information Server (IIS) configuration file, dsenv. (This part of the process also known as, set you run time environment variables.)

   Information to help you locate this file is listed above. In effect, we need to add 4 new lines to the bottom of the dsenv file. Example as shown;

   ```
   DATASTAGE_JRE=/opt/IBM/InformationServer/ASBNode/apps
   export DATASTAGE_JRE
   DATASTAGE_JVM=jre/bin/j9vm
   export DATASTAGE_JVM
   ```

   The above assumes that IBM Information Server is installed in,

   ```
   /opt/IBM/InformationServer
   ```

   Because these are environment variables, and environment variables are only examined when a process starts, you need to recycle DSEngine to have these changes take effect. If you don't know how to recycle DSEngine, and if its okay, you can always just recycle the whole operating system.

3. Set your compilation area environment variables.

   The environment variables we set above affect the run time environment. If we are going to successfully compile our Java program code, we need to set additional environment variables there. Set your CLASSPATH, example as shown;

   ```
   CLASSPATH=$CLASSPATH:/opt/IBM/InformationServer/Server/DSEngine/java/lib/tr4j.jar
   export CLASSPATH
   ```

   Again, where IBM Information Server is installed as stated.

Note: Above, we set our CLASSPATH to a full pathname to the Java Pack JAR file, tr4j.jar. This is for the **compilation environment**.

Later, in our **run time environment**, when we need to set our CLASSPATH to find our custom end user created Java program code, we will set CLASSPATH to the parent directory containing the Class file.

Later, we set it to the parent directory, not the complete filename.

4. Create your custom end user created Java program or Java code fragment.

   Example19-1 lists the Java source code used in this example. A code review follows.

Example 19-1 Sample Java code, compiled and used in Java Pack Java Transformer.

```
import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;


public class MyJavaTransform extends Stage
{

  public void initialize() {
  }


  public void terminate() {
  }


  public int process() {

    Row inputRow = readRow();
      //
    if (inputRow == null) {
```



```
      return OUTPUT_STATUS_END_OF_DATA;
    }

    String col1 = inputRow.getValueAsString(0);
    String col2 = inputRow.getValueAsString(1);
    String col3 = inputRow.getValueAsString(2);
    String col4 = inputRow.getValueAsString(3);

    int    col5 = Integer.parseInt(col1) +
                  Integer.parseInt(col2) ;

    Row outputRow = createOutputRow();
      //
    outputRow.setValueAsString(0, Integer.toString(col5));
    outputRow.setValueAsString(1, col3                  );
    outputRow.setValueAsString(2, col4                  );
      //
    writeRow(outputRow);


    return OUTPUT_STATUS_READY;
  }


}
```

A code review related to Example19-1, is offered below;

– import {two lines}

In this case, the Java import statements specify that we are using two objects from the IIS DataStage component Java Pack API; objects named Row and Stage.

The Row object gives us methods and properties to examine and set input and output data rows.

The Stage object gives us the ability to interact with the DSEngine as a whole, detecting on every input row events, as well as start up and shut down of the Stage proper.

- public class MyJavaTransform extends Stage

  MyJavaTransform is the name of the routine we are creating.

- public void initialize(), and terminate()

  These are the methods in which we place Java source code to execute upon start up and shut down of this Stage. Presence of these methods is optional because they are empty; we list them only to document their existence.

- public int process

  This is the on every input row method to this stage. As marked by pairs of curly braces, this Java source code block continues to the end of the displayed file.

- readRow(), calls to get the next input row of data.

- return OUTPUT_STATUS_END_OF_DATA, is a return code we need to send to normally terminate operation of this Java Pack Stage.

  Other codes for errors and warnings are also available and standard.

- inputRow.getValueAsString(), calls to get the current input row column values. Notice that columns are retrieved by ordinal number; in this case, starting with zero. E.g., this first input column is column zero, the second column is 1, and so on.

  (Blame Java; Java counts starting at zero, not one.)

- int col5 = .., is our Java source code to add column values 1 and 2.

- outputRow.setValueAsString(), starts an area where we set our output data row column values.

- writeRow() sends the assembled output data row to IIS, and the Java Pack.

- return OUTPUT_STATUS_READY, is required to state we are ready for the next input data row.

5. Compile the Java source code from above.

   With a properly configured compilation environment (CLASSPATH, and more), and a source code file named, MyJavaTransform.java, this is done with a command equal to,

   ```
   javac  MyJavaTransform.java
   ```

   A successful compile will produce a file named, MyJavaTransform.class.

6. Create a new IBM Information Server (IIS) DataStage component Job using the DataStage Designer program.

   Figure19-2 displays the Job we need to create.

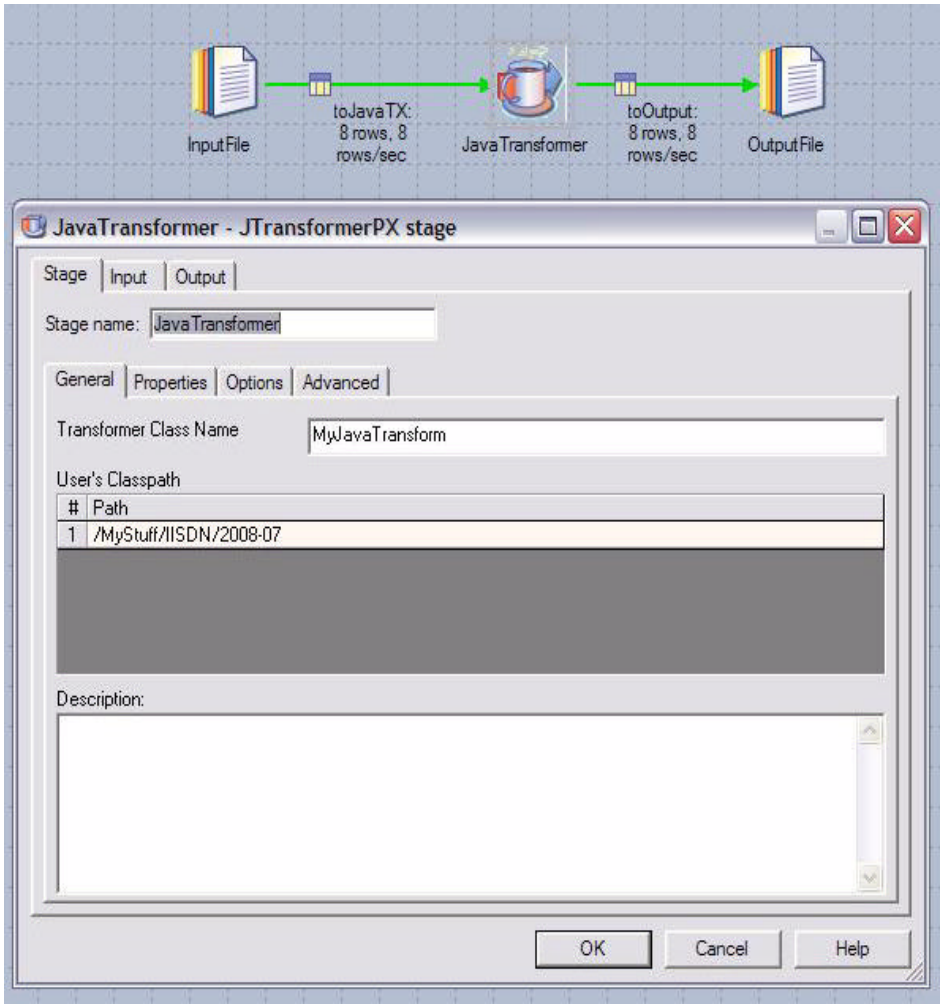

Figure 19-2 Sample IIS DataStage Job, with Java Transformer.

As displayed in Figure19-2, complete the following;

a. From the Menu Bar, select, File -> New -> Jobs -> Parallel Job.

   Then from the Menu Bar, select, File -> Save as.

   Give your new Parallel Job a name.

b. Drag and Drop three Stages from the Palette view to the Parallel Canvas.

   From the File Drawer, Drag and Drop two Sequential File Stages. From the Real Time Drawer, Drag and Drop one Java Transformer Stage.

Position and rename all Stages and Links as displayed in Figure19-2.

c. Configure the Properties to the 2 Sequential File Stages to include the filename, delimiters, and related. For the input Sequential File Stage, specify input column names and data types.

   Per our example, the input file has 4 columns, 2 of which are of type Integer (so that we may add their values). The output file will receive only 3 columns. We do not need to specify output columns in the output Sequential file Stage, as these columns will be inherited from the Java Transformer.

d. We need to configure 2 areas of the Java Transformer Stage, the first of which is displayed in Figure19-2;

   - Per Figure19-2, set the Transformer Class Name.
   - Also per Figure19-2, set the CLASSPATH value to equal the directory where this custom end user created Java class file is located.

e. Still inside the Java Transformer, move to the Input TAB -> Columns TAB. Example as shown in Figure19-3.

   - In Figure 19-3, we specify 4 input columns as displayed.

     We called our input columns col1, col2, etcetera. It is most important, given our smallish Java code example, that columns 1 and 2 contain Integer values.

   - Not displayed- move also to the Output TAB -> Columns TAB, and specify 3 output columns, the first of type Integer (holding our summation result).

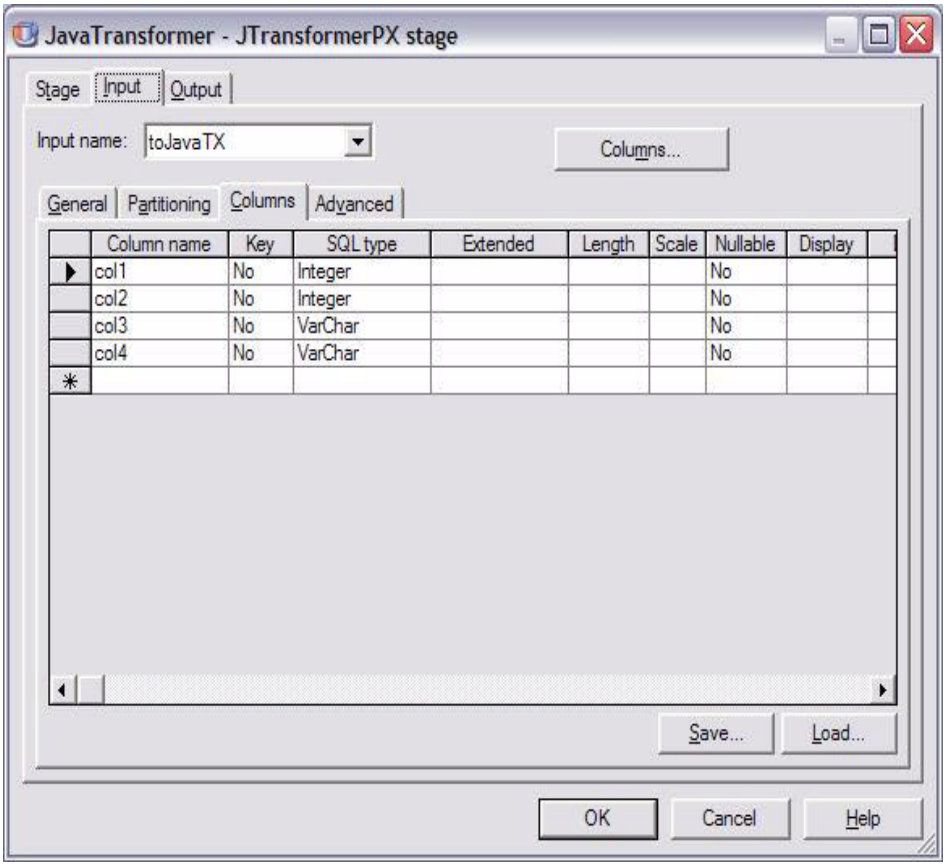

Figure 19-3 Java Transformer, Input TAB -> Columns TAB.

7. When you have configured all 3 Stages in the example Job, Compile and Test.

   A successful result reads the input data file, and adds input columns 1 and 2.

19.3 Summation

In this document, we reviewed or created:

We examined some of the extensibility of the IBM Information Server (IIS) DataStage component; specifically, we created and used a Java Pack Java Transformer. This being a continuation of the prior edition of this document, where we created similar examples using C/C++ and BASIC.

Through the IIS DataStage component Java Pack Java Transformer, we can execute custom end user created Java programs and Java code fragments, a very handy thing to be able to do.

Persons who help this month.

Aaron Titus, and Daniel Schallenkamp.

Additional resources:

None.

Legal statements:

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.

