Contents

Introduction .................................................................................................................. 1

1.1 Prerequisites ........................................................................................................ 2
   Required .................................................................................................................... 2
   Recommended ............................................................................................................ 2

1.2 Installing MapleNet Publisher ........................................................................... 2

Developing Java-based Applets for MapleNet ............................................. 5
   In this chapter ........................................................................................................... 5

2.1 Developing Java-based Applets ........................................................................ 5

2.2 Writing Java-based Applets .............................................................................. 6
   Passing Requests to MapleNet .............................................................................. 6
   Sequence of Events ................................................................................................. 8
   Maple Notes ............................................................................................................ 8

2.3 Compiling Java Code .......................................................................................... 9

2.4 Creating a JAR File that Contains a Manifest ................................................... 9
   Creating a Manifest ................................................................................................. 9
   Creating a JAR File ................................................................................................. 10

2.5 Creating an HTML Page ..................................................................................... 10
   Specifying Parameters ........................................................................................... 12
   Loading the Web Page ............................................................................................ 12

2.6 Publishing Files to the MapleNet Server ......................................................... 13
   Using MapleNet Publishing Scripts ...................................................................... 13
   Example Procedure: How to Create and Publish a JAR File ............................ 14

2.7 Accessing Published Content .............................................................................. 15

Developing Maplets for MapleNet ................................................................. 17
   In this chapter ......................................................................................................... 17

3.1 Writing a MapleNet-Compliant Maplet ........................................................... 17
   Restrictions .............................................................................................................. 17

3.2 Creating an HTML Page ..................................................................................... 18
   Specifying Parameters ........................................................................................... 19
   Loading the Web Page ............................................................................................ 19

3.3 Publishing Files to the MapleNet Server ......................................................... 20
   Using MapleNet Publishing Scripts ...................................................................... 20
Example Procedure: How to Create and Publish a JAR File ...... 20
3.4 Accessing Published Content .................................................... 22
1 Introduction

The MapleNet™ Publisher’s Guide provides instructions and examples for the following.

• Installing MapleNet Publisher software to your computer.  
  See Installing MapleNet Publisher on page 2.
• Developing Java™-based applets that communicate with the MapleNet Server, including:
  • Writing a Java-based Applet
  • Compiling Java Code
  • Creating a JAR File that Contains a Manifest
  • Creating an HTML Page
  • Publishing Files to the MapleNet Server
  • Accessing Published Content
  See Developing Java-based Applets for MapleNet on page 5.
• Developing Maplets™ for MapleNet, including:
  • Writing a MapleNet-Compliant Maplet
  • Creating an HTML Page
  • Publishing to the MapleNet Server
  • Accessing Published Content
  See Developing Maplets for MapleNet on page 17.
1.1 Prerequisites

Required

• Java 2 SDK, v1.4 or later
• Java 2 Runtime Environment (J2RE), v1.4 or later
• Maple 8.01

Note: If you do not have SDK and J2RE installed, go the Sun Microsystems, Inc. http://java.sun.com/j2se/ Web page to download and install the version appropriate for your system.

Recommended

• Java development environment for developing Java applets
• Web page development environment for creating HTML pages

1.2 Installing MapleNet Publisher

To install the publisher component of the MapleNet software:

1. Double-click the publisherinstaller.jar file or issue the following command using the command line.
   java -jar publisherinstaller.jar

2. Once finished, the directory structure should resemble the following.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Location of batch files required for bundling and publishing content</td>
</tr>
<tr>
<td>doc</td>
<td>Location of Javadoc files describing the MapleNet classes</td>
</tr>
<tr>
<td>jar</td>
<td>Location of Java code files</td>
</tr>
<tr>
<td>lib</td>
<td>Location of new Maple library for maplets</td>
</tr>
<tr>
<td>maple</td>
<td>Location of required Maple files</td>
</tr>
<tr>
<td>sample</td>
<td>Location of sample Java and HTML files</td>
</tr>
<tr>
<td>publisherguide.pdf</td>
<td>Publisher's Guide</td>
</tr>
</tbody>
</table>
**Note:** To access these files from *any* directory, the `bin` directory must be added to the path environment variable.
2 Developing Java-based Applets for MapleNet

In this chapter
• Developing Java-based applets that communicate with the MapleNet Server
• Compiling applet code
• Creating a JAR file that contains a manifest
• Creating an HTML page that references applets
• Publishing to the MapleNet Server
• Accessing published applets

2.1 Developing Java-based Applets

To create content for publication on the MapleNet Server, complete the following tasks.

1. Write an applet that communicates with the MapleNet Server.
   See Writing Java-based Applets on page 6.

2. Compile the Java code.

3. Create a JAR file that contains a manifest.
   See Creating a JAR File that Contains a Manifest on page 9.

4. Create an HTML page that references the applet.
2.2 Writing Java-based Applets

The following Sample Integration Applet (Figure 1) requires a user to enter a UserID, Password, and expression. By clicking Calculate, a request is sent to the MapleNet Server to integrate the equation. The results are displayed in the Result textbox. This applet is used to illustrate how requests are passed to the MapleNet Server. The input and output are text-based Standard Maple syntax.

![Sample Integration Applet](Image)

**Figure 1 Sample Integration Applet**

**Passing Requests to MapleNet** The Sample Integration Applet Code (Listing 1) provides the section of code that performs the connection with MapleNet. In this example, the computer name is derived from the Web page that contains the applet during the initialization phase of the applet, which is stored in the host string variable. The complete code for this example is in the source file integration.java, which is available on the CD.

**Listing 1 Sample Integration Applet Code**

```java
//Routine to take the input expression and send //it to the MapleNet Server //using the MapleStatement class
void do_calc(ActionEvent e) {

//Erase current answer
TF_Result.setText(" ");
```

//Retrieve the expression
String expression = TF_Expression.getText();
//If there is an expression, process it
if ( expression.length () > 0 ) {
    //Need user and password
    String user = TF_Username.getText () ;
    String pswd = TF_Password.getText () ;
    //Must encapsulate the ‘execute()’ method in a try/catch
    try {
        //Create the host connection. Will use the default MapleNet port
        HostInfo hostinfo = new HostInfo(host);
        //Create the user login object
        UserInfo userinfo = new UserInfo( user, pswd );
        //Create the MapleStatement that will connect to the server
        MapleStatement stmnt = new MapleStatement(hostinfo, userinfo );
        //Encapsulate the expression into maple integration form
        String question = "int( " + expression + ", x)";
        //Get MapleNet server to process the question
        String answer = stmnt.execute( question );
        //Verify that an answer was retrieved
        if (answer==null || answer.length() <= 0 ) {
            //No answer - check the XML returned from the server
            answer = "Error: " + stmnt.getXMLString();
        }
        //Display the result
        TF_Result.setText( answer );
    } catch (Exception ex ) {
        //Catch all exceptions here
        //MapleStatement.execute will throw
        //MapleStatementException - usually syntax error in question
        //MapleException - server connection or login errors
        System.out.println("do_calc exception =" + ex.toString() );
    }
    //Put keyboard focus back to expression box
    TF_Expression.setSelectionStart (0);
    TF_Expression.setSelectionEnd( equation.length() );
    TF_Expression.requestFocus();
} else {
    // No expression specified
    Toolkit.getDefaultToolkit() .beep();
}
}
**Sequence of Events** When the user clicks the **Calculate** button on the *Sample Integration Applet* (Figure 1), the `do_calc()` routine is invoked with the following sequence of events.

1. The **Result** textbox is set to blank.

2. The expression is extracted from the **Enter** text box.

3. The username and password are obtained from the appropriate text boxes.

4. A **try/catch** block is used to catch any exceptions that are raised when communicating with the MapleNet Server.

5. The **HostInfo** and **UserInfo** objects are created. These objects encapsulate connection information.

6. A **MapleStatement** object is created using the **HostInfo** and **UserInfo** objects. Note that this step creates only the object. The actual connection to the server is not made until the **MapleStatement** execute method is used.

7. The actual request for the server is built using the Maple syntax for integrating and inserting the expression from event 2.

8. The question is passed to the server using the **MapleStatement** execute method. The result is stored in the **String variable answer**. Note that any error from the server is raised as an exception. If an exception occurs, it is handled in the **catch()** portion of code.

9. The answer is displayed in the **Result** text box.

**Maple Notes**

- To pass **multiple questions** to the server, you can build a **string array** containing the questions and pass them to the execute method. This method returns a string array of resulting answers corresponding to each question.

- Expression sequences cannot be rendered in MathML. Therefore, the output for any Maple command you are using that returns an expression sequence (for example, `solve`) must be converted to a list if you want to render it in MathML. If you do not convert it, only the first element of the expression sequence will be displayed.
2.3 Compiling Java Code

Compile the applet code with a Java compiler. This produces an integration.class file. This class file contains the executable code for the applet. To use the executable code, create an HTML page that references this class as an applet.

*Note:* The maplenetclient.jar must be included in the classpath argument of the compiler.

2.4 Creating a JAR File that Contains a Manifest

For the applet to access the MapleNet Server correctly, it must use the classes contained in the maplenetclient.jar file. There are several ways to provide this file (see *Loading the Web Page* on page 12). The preferred method uses the Java Extension Mechanism to distribute the maplenetclient.jar file that is stored on the Web server. If you reference the maplenetclient.jar file in the manifest of the JAR file that contains the compiled code, maplenetclient.jar will be installed automatically when required.

To implement this, you need to first create a manifest, then create a JAR file that includes both this manifest and the compiled class files for your applet, and then reference this JAR file in the .html file.

**Creating a Manifest**

To create the manifest, you will need to know the URL path to the maplenetclient.jar in your site's Web root directory. For example, http://mysite.com/maplenet/jar/maplenetclient.jar. If you do not know this information, ask your system administrator.

Create a text file, say mymanifest.txt, that contains the following lines.

```
Extension-List: maplenetclient
maplenetclient-Extension-Name: maplenetclient
maplenetclient-Specification-Vendor: Waterloo Maple Inc.
maplenetclient-Specification-Version: 1.3
maplenetclient-Implementation-Vendor: Waterloo Maple Inc.
maplenetclient-Implementation-Vendor-Id: com.maplesoft
maplenetclient-Implementation-Version: 1.3.0
maplenetclient-Implementation-URL: http://mysite.com/maplenet/jar/maplenetclient.jar
```

where the URL specified in the last line is the path to the maplenetclient.jar in your site's Web root directory.
Once this text file is created, and the class files for the applet have been compiled, you can create the JAR file that will be referenced in the .html file.

**Creating a JAR File**

Use the following command to create a JAR file, say `myapplet.jar`, that contains the manifest and the required class files. This example assumes the manifest file and the class files are in the same directory, and that the user is in that directory.

```
jar -cmf mymanifest.txt myapplet.jar *.class
```

where `mymanifest.txt` is the manifest file created in *Creating a Manifest* on page 9, `myapplet.jar` is the name of the JAR file being created, and `*.class` is used to include all class files in the present directory.

Once this JAR file is created, specify it in the .html file, as shown in the next section.

### 2.5 Creating an HTML Page

Before you can publish to the MapleNet Server, create an HTML page that references the compiled Java code (that is, the `.class` file) as an applet.

The following *Integration.html* sample code (Listing 2 on page 11) illustrates how to build a page for Microsoft® Internet Explorer® and Netscape®.

**Note:** Java applets that require a Java version greater than 1.1 must use the Java Plugin. Use the `<object>` tag in Internet Explorer and the `<embed>` tag in Netscape. For a complete discussion of setting up Web pages for simultaneous Internet Explorer and Netscape support, refer to [http://java.sun.com/products/plugin/1.3/docs/intranet.html](http://java.sun.com/products/plugin/1.3/docs/intranet.html)
Listing 2 Integration.html

```html
<html>
<head>
<title>
Integration Example
</title>
</head>
<body>
<center>
<H2>Integration Applet will appear below in a Java-enabled browser</H2>

<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499093" width="400" height="300"
align="middle">
    <param name="type" value="application/x-java-applet;version=1.3" />
    <param name="CODEBASE" value="./" />
    <param name="CODE" value="integration.class" />
    <param name="NAME" value="IntegrationApplet" />
    <param name="ARCHIVE" value="myapplet.jar" />
    <param name="expression" value="3*x^2 + 2*x +1" />
    <comment>
        <embed type="application/x-java-applet;version=1.3"
            width="400"
            height="300"
            align="middle"
            code="integration.class"
            codebase="./"
            archive="myapplet.jar"
            expression="3*x^2 + 2*x +1"/>
    </comment>
</object>
</center>
</body>
</html>
```
Specifying Parameters

In the .html file, specify the parameters listed in Table 1.

Table 1  HTML File Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Specifies the size of the applet on the Web page.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the size of the applet on the Web page.</td>
</tr>
<tr>
<td>codebase</td>
<td>Specifies the directory from which to load the class file.</td>
</tr>
<tr>
<td>code</td>
<td>Specifies the class of the applet. In the example, integration.class is specified.</td>
</tr>
<tr>
<td>archive</td>
<td>Specifies additional JAR files that are required, for example, myapplet.jar.</td>
</tr>
</tbody>
</table>

For this example, integration.java, you can add optional input parameters in the Java source code, as listed in Table 2.

Table 2  Java Source Code File Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Specifies the name of the MapleNet Server. The default is the name of the server for the Web page. If not specified, the default is used.</td>
</tr>
<tr>
<td>expression</td>
<td>Specifies the initial value for the Enter Expression text box. The applet does not perform the integration until the Calculate button is clicked. If not specified, the field is blank.</td>
</tr>
</tbody>
</table>

Loading the Web Page

When the Web page loads, it runs the applet specified by the code parameter. This code accesses the MapleNet Server by using the classes contained in the maplenetclient.jar file. This file must be accessible to the Web browser.

The maplenetclient.jar file can be referenced in any of the following ways.

- Store the maplenetclient.jar file on Web server and distribute it using the Java Extension Mechanism. That is, reference it in the manifest of the JAR file that contains the compiled code and maplenetclient.jar will automatically install when required.
• Copy the maplenetclient.jar file to the extension directory of the Java 2 Runtime Environment (J2RE) that is used by the Web browser. For example, the lib\ext directory of the J2RE.

• Store the maplenetclient.jar file on the Web server and access it using the archive parameter.

Once the applet is compiled and the Web page is created, publish the associated files (.html, .java, and .jar) to the MapleNet Server.

### 2.6 Publishing Files to the MapleNet Server

The Web page, source code, and compiled code files can be published to the Web server either directly by using the mechanism already in place at your site or by using the MapleNet publishing scripts.

#### Using MapleNet Publishing Scripts

The MapleNet publishing scripts bundle and send files to the server to be installed in a location accessible by the client.

**To publish files to the server:**

1. Place the files required for publishing in the same subdirectory.

2. Run the CreateJar (createjar, UNIX) script to wrap the contents of the subdirectory into a JAR file. (A JAR file is a Java version of a zip file. As such, it can be viewed with a zip program such as WinZip 7.0.)

3. Run the PublishJar (publishjar, UNIX) script to publish the contents of the JAR file on the server.

For an example procedure, see *Example Procedure: How to Create and Publish a JAR File* on page 14.
Example Procedure: How to Create and Publish a JAR File

The following example files are used to illustrate the procedure for moving files to the server.

<table>
<thead>
<tr>
<th>Directory\File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Projects\Tutorial</td>
<td>Main directory</td>
</tr>
<tr>
<td>\Projects\Tutorial\integration.html</td>
<td>Web page</td>
</tr>
<tr>
<td>\Projects\Tutorial\integration.java</td>
<td>Source code for the applet to be contained in the page</td>
</tr>
<tr>
<td>\Projects\Tutorial\integration.class</td>
<td>Compiled Java code.</td>
</tr>
</tbody>
</table>

1. Create a JAR file using the example files.

   **Windows**
   Run the CreateJar.bat file from the Projects directory.
   cd Projects
   CreateJar tutor1.jar Tutorial
   
   where tutor1.jar is the JAR file to be created and Tutorial is the subdirectory to be searched for source files.

   The CreateJar command searches the directory (Tutorial) and all its subdirectories for source files and then creates the tutor1.jar file in the Projects directory.

   **UNIX**
   Run the createjar script from the Projects directory.
   cd Projects
   createjar tutor1.jar Tutorial
   
   where tutor1.jar is the JAR file to be created and Tutorial is the subdirectory to be searched for source files.

   The createjar command searches the directory (Tutorial) and all its subdirectories for source files and then creates the tutor1.jar file in the Projects directory.

2. Verify the contents of the tutor1.jar file by using the jar tool from the SDK. Enter the command:
   
   jar -tf tutor1.jar

3. Publish the JAR file on the server. The command parameters are described in Table 3.
Windows
Run the PublishJar.bat file.
PublishJar user password jarfile destination

UNIX
Run the publishjar command.
publishjar user password jarfile destination
For example,
publishjar admin spiff tutor1.jar //Servername/Projects/tutor1.jar

The server recreates all the files and directories contained in the JAR file. For example, the files are extracted to \Projects\Tutorial on the server.

Table 3 Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>User name already defined on the server to which you are publishing</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user</td>
</tr>
<tr>
<td>jarfile</td>
<td>Name of the JAR file</td>
</tr>
</tbody>
</table>
| destination | Name of the remote server and directory where the JAR file is sent and unwrapped. The directory will be placed in the Web Server's root directory.  
   **Note:** The publisher must have write permission to this directory. |

**Note:** You can publish to the server only when the MapleNet Server and the Web Server are running. For more information, refer to the MapleNet Installation Guide.

2.7 Accessing Published Content

Demonstrating with the example files of the previous section, a user can access the published Web pages by using the following URL.

http://Servername/Projects/Tutorial/integration.html

The path is based on the destination parameter specified in the PublishJar (publishjar, UNIX) command. (See Table 3.)
Developing Java-based Applets for MapleNet
3 Developing Maplets for MapleNet

In this chapter

• Writing a MapleNet-Compliant Maplet
• Creating an HTML page that references a maplet
• Publishing files to the MapleNet Server
• Accessing published content

3.1 Writing a MapleNet-Compliant Maplet

When writing a maplet for MapleNet, follow these guidelines.

• Ensure that the last line in the file is the `Maplets:-Display` command.
• Save the file as a `.maplet` file.
• It is recommended that you use the long form of function names, for example, `plots[animate](arguments)`. For Maplets (or other modules), accessing the short names may be implemented with a `use` statement.
• It is recommended that you use the `waitforresult='false'` setting in your `Evaluate` statements. See Listing 3 on page 18.
• Expression sequences cannot be rendered in MathML. Therefore, the output for any Maple command you are using that returns an expression sequence (for example, `solve`) must be converted to a list if you want to render it in MathML. If you do not convert it, only the first element of the expression sequence will be displayed.

Restrictions

When using maplets in MapleNet, consider the following restrictions.

• You cannot open files.
• You cannot open a maplet from another maplet.

Note: The `maplets.jar` file was updated for use with MapleNet and was added to your Maple 8.01 directory during installation of the publisher components.
3.2 Creating an HTML Page

Before you can publish to the MapleNet Server, create an HTML page that references the saved .maplet file.

The following sample code (Listing 3) illustrates how to build a page for Internet Explorer and Netscape.

Note: Java applets that require a Java version greater than 1.1 must use the Java Plugin. Use the \texttt{<object> tag in Internet Explorer and the \texttt{<embed> tag in Netscape. For a complete discussion of setting up Web pages for simultaneous Internet Explorer and Netscape support, refer to http://java.sun.com/products/plugin/1.3/docs/intranet.html

Listing 3 Maplets HTML Code

```html
<head>
<title>Maplets Test Page</title>
</head>
<body>
This Web page contains an applet that will call the maplet specified below. The maplet will appear in a separate window. <br>
<center>
<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499093" width="400" height="300" align="middle"  
> </param name="CODEBASE" value = "." />
</param name="CODE" value = "com.maplesoft.client.maplets.MapletLoader.class" />
</param name="ARCHIVE" value = "../jar/mapletloader.jar" />
</param name="NAME" value = "plot_02_x" />

<param name="mapleFile" value="mytest.maplet" />
<param name="user" value="client" />
<param name="password" value="demopass" />
<comment>
<embed type="application/x-java-applet;version=1.3" width="400" height="300" align="middle" code="com.maplesoft.client.maplets.MapletLoader.class" codebase="." />
</object>
```
archive="../jar/mapletloader.jar"
mapleFile="mytest.maplet"
user="client"
password="demopass">
  <noembed> </comment> Error Java 1.3 not supported </noembed>
  </div>
</object>
</center>
</body>
</html>

### Specifying Parameters

In the .html file, specify the parameters listed in Table 4.

**Table 4 HTML File Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Specifies the size of the maplet on the Web page.</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the size of the maplet on the Web page.</td>
</tr>
<tr>
<td>codebase</td>
<td>Specifies the directory from which to load the class file.</td>
</tr>
<tr>
<td>code</td>
<td>Specifies the required class file. For maplets, MapletLoader class is always called.</td>
</tr>
<tr>
<td>archive</td>
<td>Specifies JAR files that are required, for example, mapletloader.jar.</td>
</tr>
<tr>
<td>mapleFile</td>
<td>Specifies the maplet file.</td>
</tr>
</tbody>
</table>

**Loading the Web Page**

When the Web page loads, it runs the MapletLoader applet, which in turn starts the maplet specified by the mapleFile parameter. This code accesses the MapleNet Server by using the classes contained in the maplenetclient.jar file. This file must be accessible to the Web browser.

The maplenetclient.jar file can be referenced in any of the following ways.

- Store the maplenetclient.jar file on Web server and distribute it using the Java Extension Mechanism. That is, reference it in the manifest of the
• Developing Maplets for MapleNet

JAR file that contains the compiled code and maplenetclient.jar will automatically download when required.

• Copy the maplenetclient.jar file to the extension directory of the Java 2 Runtime Environment (J2RE) that is used by the Web browser. For example, the lib\ext directory of the J2RE.

• Store the maplenetclient.jar file on the Web server and access it using the archive parameter.

Once the maplet and the Web page are created, publish them to the MapleNet Server.

3.3 Publishing Files to the MapleNet Server

The Web page and maplet file can be published to the Web server either directly by using the mechanism already in place at your site or by using the MapleNet publishing scripts.

Using MapleNet Publishing Scripts

The MapleNet publishing scripts bundle and send files to the server to be installed in a location accessible by the client.

To publish files to the server:

1. Place the files required for publishing in the same subdirectory.

2. Run the CreateJar (createjar, UNIX) script to wrap the contents of the subdirectory into a JAR file. (A JAR file is a Java version of a zip file. As such, it can be viewed with a zip program such as WinZip 7.0.)

3. Run the PublishJar (publishjar, UNIX) script to publish the contents of the JAR file on the server.

For an example procedure, see Example Procedure: How to Create and Publish a JAR File.

Example Procedure: How to Create and Publish a JAR File

The following example files are used to illustrate the procedure for moving files to the server.

<table>
<thead>
<tr>
<th>Directory\File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Projects\Tutorial</td>
<td>Main directory</td>
</tr>
<tr>
<td>\Projects\Tutorial\integration.html</td>
<td>Web page</td>
</tr>
</tbody>
</table>
1. Create a JAR file using the example files.

   **Windows**
   
   Run the `CreateJar.bat` file from the `Projects` directory.
   
   ```
   cd Projects
   CreateJar tutor1.jar Tutorial
   ```
   
   where `tutor1.jar` is the JAR file to be created and `Tutorial` is the subdirectory to be searched for source files.
   
   The `CreateJar` command searches the directory (`Tutorial`) and all its subdirectories for source files and then creates the `tutor1.jar` file in the `Projects` directory.

   **UNIX**
   
   Run the `createjar` script from the `Projects` directory.
   
   ```
   cd Projects
   createjar tutor1.jar Tutorial
   ```
   
   where `tutor1.jar` is the jar file to be created and `Tutorial` is the subdirectory to be searched for source files.
   
   The `createjar` command searches the directory (`Tutorial`) and all its subdirectories for source files and then creates the `tutor1.jar` file in the `Projects` directory.

2. Verify the contents of the `tutor1.jar` file by using the `jar` tool from the SDK. Enter the command:

   ```
   jar -tf tutor1.jar
   ```

<table>
<thead>
<tr>
<th>Directory\File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\Projects\Tutorial\integration.maplet</code></td>
<td>Source code for the maplet to be contained in the page</td>
</tr>
</tbody>
</table>
3. Publish the JAR file on the server. The command parameters are described in Table 5.

**Windows**

Run the `PublishJar.bat` file.

```
PublishJar user password jarfile destination
```

**UNIX**

Run the `publishjar` command.

```
publishjar user password jarfile destination
```

For example,

```
publishjar admin spiff tutor1.jar //Servername/Projects/tutor1.jar
```

The server recreates all the files and directories contained in the JAR file. For example, the files are extracted to `\Projects\Tutorial` on the server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>User name already defined on the server to which you are publishing</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user</td>
</tr>
<tr>
<td>jarfile</td>
<td>Name of the JAR file</td>
</tr>
<tr>
<td>destination</td>
<td>Name of the remote server and directory where the JAR file is sent and unwrapped. The directory will be placed in the Web Server’s root directory.</td>
</tr>
</tbody>
</table>

**Note:** The publisher must have write permission to this directory.

**Table 5 Command Parameters**

The path is based on the destination parameter specified in the `PublishJar` (`publishjar`, UNIX) command. (See Table 5.)

### 3.4 Accessing Published Content

Demonstrating with the example files of the previous section, a user can access the published Web pages using the following URL.

```
http://Servername/Projects/Tutorial/integration.html
```

The path is based on the destination parameter specified in the `PublishJar` (`publishjar`, UNIX) command. (See Table 5.)