Lab 4. ArcGIS Server and ArcGIS Engine (ArcObjects) programming

Part 0. Revisiting .jsp and html

Problems/solutions and lessons learned from lab 3:

1) *Null pointer exception thrown:* The name of the Java Bean is inconsistent.
2) *Apache Tomcat web development and deployment:* JBuilder has built-in web servers (Tomcat 4.0, 5.0, etc.). Outside of a testing/development scenario, Apache Tomcat often runs as a standalone web server. The PGIST development server (pgistdev.geog.washington.edu) runs Apache Tomcat 5.0 as a web server. We will be using PGISTDEV to deploy our web modules, created in this lab.

Part 1. Connecting to ArcGIS Server

In this section of the lab, we will be looking at how to connect to an ArcGIS Server object, located on PGISTDEV. Pay attention to the connection parameters, as these will become necessary for later sections of this lab.

1) Start ArcCatalog.
2) In the table of contents, expand the GIS Servers item and double click on Add ArcGIS Server. This will allow a connection to PGISTDEV.
3) A parameters window will display. Type in the server name: `pgistdev`. You will be able to access the server, because every person in this course has been added to the *agsusers* group on PGISTDEV. [Note: Since we are currently within the Geography LAN, we will not need to specify the full pathname of the PGISTDEV server.]
4) You have now created a connection to the PGISTDEV server and can now explore the server objects within. A server object named `usa` will display when expanding the server connection. The data and ArcMap project used for this server object can also be found at `c:\ArcGIS\DeveloperKit\samples\data\usa\` (however the projection has been modified on this version of the .mxd).
5) Open the original .mxd file located at `c:\ArcGIS\DeveloperKit\samples\data\usa\`, using ArcMap. Select File → Map Properties → Data Source Options. Notice the file system path is set to store the relative pathname of data referenced by the .mxd. This is important to note, as when you copy the .mxd and the source data files to the server (in order to create an ArcGIS Server object), an absolute pathname would break.

Part 2. Programming ArcGIS Server applications

In this section of the lab, we will be looking at ArcGIS Server application web module files, including functions and the deployment of a Java Server Page (.jsp) and a Java Bean.

1) Navigate within your web browser to the following web address: http://pgistdev.geog.washington.edu:8080/map_zhou/. A .jsp page displays with a simple map viewer, with four functions: zoom in, zoom out, pan, and full extent. Toggle the option buttons and click within the map pane to explore these functions.
2) Right click over a blank portion of the displayed webpage, to view the source code for this file. There are a few items to note:
    a. `<form action=""></form>` [These tags allow the client to send information to the server. All input items within these tags are collected and sent to the server

address specified within the quotation marks, following `form action=`.]

b. `<input type="" name="" />` [This is one component of a form, which could be a button, a text field, an image.]

c. `<script language="JAVASCRIPT"></script>` [This is the Javascript portion of the webpage, which handles all client-side interaction.]

3) Close the source code for this webpage, as we no longer will need this particular view. We will now explore the web module which delivers this HTML within JBuilder.
4) Start JBuilder.
5) Create a new project, and save this within your student directory on the P:\ drive.
6) Create a new web module. The name of this module will become part of the URL for the webpage, so use your *uwnetid* as part of the pathname to avoid duplication with other students.
7) Set the classpath to the required libraries:
   a. `c:\arcgis\java\opt\arcobjects.jar` [This contains the ArcObjects packages. Explore JavaDoc documentation at: http://arcgisdeveloperonline.esri.com/ArcGISDeveloper/Java/arcengine/index.html.]
   b. `c:\arcgis\java\jintegra.jar` [This contains some exception handlers. JavaDoc documentation for this package also is located at the above link.]
8) Create a new class, named "Map". Specify that the package name is "geog465.*uwnetid*".
9) Copy and paste the source code for this new class from the .java file located at `p:\geog465win05\lab4\Map.java` and replace the auto-generated code. Remember to correct the package declaration to match the package name specified in the previous step.
10) Explore the class, including attributes and methods. Select the UML tab at the bottom of the main panel, to view the overall schema for this class and imported packages. Return to the source code view to answer the following questions
    a. What parameters does the constructor take?
    b. What does the method `connect()` perform for an object of this class?
    c. What is the purpose of the method `getImageURL()`?
11) Create a new class, named "MimeDataServlet". Specify that the package name is "geog465.util". [Note: We are using .util because this is a general purpose utility class that might be used by other classes. Remember, we are creating a hierarchical package structure.]
12) Copy and paste the source code for this new class from the .java file located at `p:\geog465win05\lab4\MimeDataServlet.java` and replace the auto-generated code. Remember to correct the package declaration to match the package name specified in the previous step. A servlet runs on the server-side and is used to generate HTML which is sent to the client-side. This servlet class handles the map image data.
13) Create a new .jsp file, named "map_usa_*uwnetid*". Accept all defaults.
14) Copy and paste code from the .jsp file located at `p:\geog465win05\lab4\map_zhou.jsp` and replace the auto-generated code. Change the value of `action` within the form tag from "map_zhou.jsp" to "map_usa_*uwnetid*.jsp".
15) Navigate within the project explorer pane to the module directory "WEB-INF" and double click "web.xml" to open the deployment descriptor.
16) Copy and paste code from the .xml file located at `p:\geog465win05\lab4\web.xml` and

replace the auto-generated code. Change the content within the `<welcome-file>` tag to match your .jsp filename.

17) Right-click on the web module name within the project explorer pane, and select Properties. Select Build within the Properties window. Verify that "Build Web archive" is set to "When building module only". This option allows the Web Archive File (.war) to be generated when the module is compiled. Click OK to close the Properties window.
18) Right-click on the web module name and select Make. Now a deployable .war file should be located in your project folder. Verify this in Windows Explorer.

Part 3. Deploying a Web Archive File (.war) into Apache Tomcat 5.0

In this section of the lab, we will learn how to deploy a .war file.

1) Open your web browser and navigate to http://pgistdev.geog.washington.edu:8080/. Click the "Tomcat Manager" link. Use your *uwnetid* as the username. Your password is your student number.
2) Scroll down to the Deploy section of the management page. Under the "WAR file to deploy" subsection, we will upload a .war file.
3) Upload the .war file generated from Part 2.
4) Click Deploy.
5) After processing, the management page will reload. Under the Applications section, you should be able to see the name of your .war file listed (if all was coded correctly).

For more information about deployment descriptors:
http://e-docs.bea.com/wls/docs61/webapp/web_xml.html
http://jakarta.apache.org/tomcat/tomcat-4.0-doc/appdev/deployment.html
For more information about Java Server Pages (.jsp) v2.0:
http://java.sun.com/products/jsp/syntax/2.0/syntaxref20.html