HW #6 Java Servlets: Computerized Examination

Please read the entire assignment carefully before starting to work on it.

The point of this exercise is to write a server-side program using Java servlets. Users should interact with the servlet using an HTML file containing input form controls, or you may have the servlet itself generate the approproate HTML containing input widgets; either method is fine. Given more time, it would be instructive to write a shopping-cart, or search-service type of program; however, these types of programs would require you to spend a lot of effort accessing a particular database. Instead I have picked an application that requires little use of sophisticated data structures and specalized knowledge -- building an on-line examination servlet. For this assignment you will write a computerized exam about U.S. history, nutrition, chemistry lab safety, a state driver's license exam, practice GRE test questions, or just general triva. The topic is up to you, but use good taste and make it interesting.

Your program should ask the user a set of ten or so questions (all together or one-at-a-time), process the submitted data, and finally generate a response page indicating the score and providing answers to incorrect questions. The servlet should not crash, or generate null pointer exceptions because a user did not provide expected inputs.

The following features are considered important; however, an otherwise correct, baseline system that implements none of these is still worth a "B".

- Examination realism
- An aesthetically pleasing layout
- Using a diverse set of HTML input controls (instead of say, all multiple choice questions using checkboxes)
- Checking the user-supplied input, either on the server-side (by returning a page that asks the user to correct their input) or using client-side approaches (such as Java applets or JavaScript)

For fun, if you want to jazz things up, you might try one of the following:

- Having a non-static set of questions. This could be done by having the servlet generate a random set of questions from a small collection, or by asking more complicated questions that depend in part on previous answers.
- More sophisticated responses. For example, you could store a list of 'highest scores' and present this along with the user's own evaluation.

Developing the servlet

Developing your servlet code will be different in several respects from the applets and applications you have written so far. In particular you will have to download the `servlet.jar` file so that you may compile your code -- this `.jar` file must appear on your CLASS_PATH envi-

ronment variable for compilation to succeed. Secondly, all of your code must be written in a particular package (this same package name is used to specify the directory your code will reside in and will be different for each student). Finally, you will only be able to test your code by uploading it to the university system and installing it following the method described below.

> The servlet.jar file may be downloaded from the Unix system. A world-readable copy can be found in ~paulmac/481/tomcat/jakarta-tomcat-3.3.1/lib/common/servlet.jar

Intalling your servlet

If you have access to reliable, publicly accessible web server that supports servlets, you may install your servlet there. However, since most students will not have access to such a web server I have make provision for you to install your servlet(s) in a web server that I will run on the university's Unix system. To install (or test) your servlet, you must do the following:

(1) Send me email (paulmac@apl.jhu.edu) indicating the directory (on the system) that you will place your servlet class files in. Do not use a generic name like 'hw8' instead use a unique form based on your name (your login name is okay) -- this name must be the same as the package name for your Java code.

> For example. Suppose your username is "christine". You might send me email saying that you will upload your code to the directory ~christine/481/hw8/**christine**. Your code must be in the emboldened package -- 'package christine;' will be the first line of your Java files.

I will create a link so that the Tomcat web server that I am running can find your code. You will not be able to properly install your classes without first doing this (and I prefer not to get 30 such requests the night before the assignment is due), so send me your directory/package name right away

(2) Place (e.g., FTP) your class files into this directory and make sure the directory is world-readable (i.e., use chmod -r a+rx directory).

(3) Finally, test your servlet. The URL will be:

http://apl.jhu.edu:4810/servlet/directory.classname
(e.g., http://apl.jhu.edu:4810/servlet/paulmac.COBOLCertificationExam)

This URL would be used in the ACTION part of your HTML <FORM> tag.

Submitting your assignment

After installing and testing your assignment, you should turn in a printout of your code, any HTML files you may have written as a front-end, and a sample user-session (e.g., a screen shot of your browser). Clearly indicate the URL I should use to test your servlet myself.

Server problems

If you encounter problems with the server, for example, if you get no top-level page at http://apl.jhu.edu:4810/, then send me an email and I will look into the problem.

Here is a sample template for your servlet that you might find helpful:

```
package yourname;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Quizzler extends HttpServlet {

   // Thread-safe data members go here

   public void init(ServletConfig conf) throws ServletException {
      super.init(conf);
      // Probably want to add initializations for your data structures
   }

   public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws IOException{

      // Get a channel to the Web browser, so we can send output
      ServletOutputStream out = res.getOutputStream();
      res.setContentType("text/html"); // Required for HTTP

      out.println("<HTML><HEAD><TITLE>Game Show Quiz</TITLE></HEAD>");

      // More code goes here

      out.println("</HTML>");

      out.close(); // Close the output stream
   }

   // Other methods to make life (processing input, question evaluation;
   // HTML generation, etc...) easier
}
```

Suggestions:

1. Start early and simple. Try first writing a simple Hello World servlet. Once you get that working (and can compile, run, and read parameters), then try a simple set of 2-3 questions just using checkboxes. Then, flesh out the remaining details.