This project has five objectives:

First, you are introduced to GlassFish. GlassFish is an open source application server that implements the latest JEE specification. This tool is used throughout the course. The NetBeans integrated development environment is used to build source code and interact with GlassFish.

Second, you build your first set of distributed systems. These are three small web applications using Servlets and Java Server Pages.

Third, you are introduced to simple mobile device awareness and adapting content to be suitable for either desktop or mobile devices.

Fourth, you are introduced to the MVC pattern if you have not used it before.

And finally, as in all projects this semester, you should reflect on the functional and non-functional characteristics (e.g. security, scalability, failure handling, interoperability) of your solutions. There will be questions on the midterm and final exam concerning these characteristics. You should be able to demonstrate a nuanced comprehension of course content and be able to explain the technical aspects in relation to potential real-world applications.

For each project task, software documentation is required. The software that you write (HTML files, Java files and so on) must contain comments that describe what each significant piece of code is intended to accomplish. Points will be deducted if code is not well documented. Read the documentation-related links provided on the course schedule (for class #1) to understand what is expected.

Be sure to consult the rubric linked from the course schedule for details on grading.

For each task below, you must submit screenshots that demonstrate your programs running. These screenshots will aid the grader in evaluating your project.
Write an index.jsp page that asks the user to enter a string of text data, and to make a choice of two hash functions using radio buttons. The hash function choices should be MD5 and SHA-1, with MD5 being the default. When the submit button is pressed a servlet is executed. The servlet must be named ComputeHashes.java. The servlet will compute the appropriate cryptographic hash value from the text transmitted by the browser. You will need to employ the Java crypto API to compute the MD5 or SHA-1 hash of the text. The original text will be echoed back to the browser along with the name of the hash, and the hash value. The hash values sent back to the browser should be displayed in two forms: as hexadecimal text and as base 64 notation. We will discuss the use of such hash values later in the course.

To compute the MD5 and SHA-1 hashes, use these standard java packages:

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

To print the Base64 encoding, use the following method:

```java
javax.xml.bind.DatatypeConverter.printBase64Binary
```

To print the hexadecimal encoding, use the following method:

```java
javax.xml.bind.DatatypeConverter.printHexBinary
```

Be sure to provide a user friendly and attractive user interface.

If you are unfamiliar with HTML forms, a simple explanation can be found at: http://www.w3schools.com/html/html_forms.asp.

So that you may test your program, here are example hashes.

Hashes of the string "Hello":

- SHA-1 (Hex): F7FF9E8B7BB2E09B70935A5D785E0CC5D9D0ABF0
- SHA-1 (Base 64): 9/+ei3uy4Jtwk1pdeF4MxdnQq/A=
- MD5: (Hex): 8B1A9953C4611296A827ABF8C47804D7
- MD5: (Base 64): ixqZU8RhEpaoJ6v4xHgE1w==
Task 2

Use the NetBeans Project Name: Project1Task2
You must use an MVC framework for this project.

Implement a web application that implements a simple desktop and mobile “clicker” for class. Your app should allow users to submit answers to questions posed in class, and should provide a separate URL end point for getting the results of the submitted responses.

The welcome page for your app should be similar to the picture on the right. You can make it more stylish if you like, but it is not required.

When the user makes a choice and hits “submit”, their answer should be stored in your MVC model. The response should be similar to the picture on the left.

Notice that it is required to provide feedback to the user regarding the choice that they made (i.e. “D” in this example).

The user should also have the ability to submit another answer as shown in the screenshot.

You can test the application by repeatedly submitting answers and allowing your model to tally the results.

Your web app should also have a URL path “/getResults” (shown on the right) for listing the results of user voting.

The results from the survey are as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
</tbody>
</table>

These results have now been cleared.
Requirements for the /getResults path:

- List each answer that has been given, and the number of times a user has submitted that answer.
- You do not have to list options that have been chosen zero times.
- The results should be displayed sorted in alphabetical order.
- /getResults should also clear the stored results so that a new question can be posed.
- If there are no results available, then report this as shown on the right.

The web app should work with a mobile browser. For this project you can use a simple check like was used in InterestingPicture and then use an appropriate mobile doctype.

An easy way to check your web app for mobile is to use the Google Chrome DevTools. If you don’t already have it, download and install the Google Chrome browser.

- Browse to your web application in Chrome
- Toggle device mode to mobile and choose an Android or iPhone device (https://developers.google.com/web/tools/chrome-devtools/iterate/device-mode/?hl=en)
- Reload the page.
- In addition to testing, you use this to produce a screen shot showing your web app working for mobile.

If your page looks like the one on the right, even after reloading, then the doctype is not being set correctly.
This is what the web app should look like for mobile if the doctype is set correctly.

Overall web app requirements:

- You must use MVC to separate concerns.
- Use appropriate HTTP methods (i.e. not just GET)
- Implement only one HttpServlet

Hints:

- You can have multiple URL patterns in a WebServlet annotation. For example, you can indicate that a servlet can be called by two paths such as:
  urlPatterns = {"/submit", "/getResults"}

- In order to determine within the servlet which path was actually requested, you can use request.getServletPath();
Task 3
Use NetBeans Project Name: Project1Task3
You must use an MVC framework for this project.

For task 3, build an application that displays the flags of countries around the world, and gives a short description about the flag. (This application will be demonstrated in class on Monday.)

The CIA World Factbook is a good source of this information: https://www.cia.gov/library/publications/the-world-factbook/

It provides an image of each country's flag, and provides a short description of it. The flag is shown near the top of each country's information page, and the flag description can be found by clicking on the flag image, or by looking under the "Government" section.

For example, regarding the country of Gabon, the World Factbook shows the image on the right.

And in the Government section it describes the flag:

Your goal is to screen scrape the flag picture and flag description to create an application that can display just the flag and description, such as is shown on the left.

The user should also be given the opportunity to choose another country flag to display.
In more detail:

1. The user is presented with a screen with instructions: "Choose a country to display their flag"
   - You should include all the countries listed in the CIA Factbook pulldown menu.
   - Be sure to eliminate from that list all entries that are not actually countries, for example World, Arctic Ocean, Antarctica and the US Wildlife Refuges that share the country code "um".

2. Upon Submit, your web application should use screen scraping (like the example shown in class) to find the flag URL and flag description.

3. The response should be as shown above. Your response page should allow the user to choose another country to display their flag.

Exceptions:

4. If a flag description cannot be found, or if a request to cia.gov fails, then an appropriate graceful error message should be provided, as well as the ability to "Choose another country...". The actual availability of the flag need not be confirmed (In other words, the flag URL will be in an html img tag, and you do not have to check if that image was successfully retrieved by the browser.)

Your application need only work with a desktop browser (not mobile).

Notes and hints:

- Refer to http://www.w3schools.com for good help on the basic HTML you need for this task.
- Look at the HTML source for the Factbook page to find the <select> and <options> that generates the country drop-down list. You can copy and modify this list for your purposes (this will not be considered cheating).
- Notice that the general form of the flag image URLs is: https://www.cia.gov/library/publications/the-world-factbook/graphics/flags/large/ZZ-lgflag.gif where ZZ is the selected country's 2 letter country code (given in the <option> list).

Produce screen shots of your application displaying pictures of 5 different countries.

You are allowed to and encouraged to build your solution based on the InterestingPicture code you have been given in class. You MUST refactor it, however, to have project, variable, and class names that make sense for your application. For example, you will lose points if your class is still named InterestingPictureServlet.
Questions:
If you have questions, you can post them to the Blackboard Discussion Board, under the Project 1 Discussion Forum.

Summary:
Be sure to review the Rubric linked on the course schedule for the first day.

There should be three projects in NetBeans.

The NetBeans projects will be named as follows:
Project1Task1
Project1Task2
Project1Task3

You should also have three screenshots folders:
Project1Task1 Screenshots
Project1Task2 Screenshots
Project1Task3 Screenshots

Be sure to review the Rubric linked on the course schedule for the first day. For each NetBeans project, File->Export Project->To Zip... each. You must export in this way and NOT just zip the NetBeans project folders.

Now you should have for .zip files and three screenshot folders:
(Note: these diagrams show 4 tasks, but we only have 3 this semester.)

Mac OS
Create a new empty folder named with your andrew id (very important). Put all files mentioned above in to the new folder you created.

Zip that folder, and submit it to Blackboard. The submission should be a single zip file. Now you should have only one .zip file named with your andrew id:
Submission File Structure:
YourAndrewID.zip
--- Project1Task1.zip
--- Project1Task2.zip
--- Project1Task3.zip
--- Project1Task1-screenshot
--- Project1Task2-screenshot
--- Project1Task3-screenshot