EECS 4413. LAB 03: RMI, CGI, and Java Servlet basics

A. IMPORTANT REMINDERS

- **This lab will be GRADED**.
- Lab3 is due on **Tuesday (Feb 6) at 11pm**. No late submission will be accepted.
- For this lab, you are welcome to attend the lab sessions on Jan 30 and Feb 6. TAs or instructor will be available to help you. The location is LAS1002. Attendance is optional. You can also ask questions after class)
- Feel free to signal a TA for help if you stuck on any of the steps below. Yet, note that TAs would need to help other students too.
- You can submit your lab work any time before the specified deadline.

B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

- Download this lab description and the associated files and read it completely. Unzip the compressed file. If uncompressing is successful, you should get a folder **4413Lab03**, which contains 4 subfolders, for the 4 questions of this lab. Each subfolder also contains a short video that demonstrates the question requirements and expectations.

  Download the Java program from eClass (week 4 or 5), review the code and play with it. Those are the program demonstrated in class.

C. GOALS/OUTCOMES FOR LAB

- To learn/recap the RMI in Java. Server side technologies in CGI and Java Servlets.

D. TASKS

Part1: Java RMI program.

Part2: CGI/LAMP with PHP.

Part3: Java Servlet program – processing forms.

Part4: Java Servlet program – processing and generating forms.

E. SUBMISSIONS

- eClass submission. More information can be found at the end of this document.

Part I: Simple Java RMI

Write a simple RMI server and client program.

On the server side, there is a (remote) interface **Processor** that defines the method *upper(s), reverse(s), sort(s)* and *isPalindrome(s)* methods. There is also a **ProcessorImpl.java** file that implements this interface. There is also a server file **ServerRMI.java** that implements the RMI server and interacts with the client.

On the client side, there is the same interface **Processor** (identical to the server-side interface), and the client program **ClientRMI**. The client program connects with the RMI server, getting the remote object. Then it

prompts user for a string until “stop” is entered by the user. For each string, (remotely) invoke methods on the remote object to process the string.
Complete the provided code. Leave the Server-side code and client side code in the two different folders. Or, two different machines if you have multiple machines that can talk to each other by host name or IP address.

Note that with RMI, you don’t create socket yourself. RMI server will create a socket under the hood.

Develop and run your Server program in command line, not in IDE – it is not trivial to run the server in IDE as it involves running RMIregistry. Client side program can be run in command line or IDE.

Test your program on different terminals in lab, or if possible, on two different computers at home if they can recognize each other with host name or private IP address.

Sample output on the client side (first start RMIregistry and server program, using the default port or a port number of your choice.)

java ClientRMI (or use IDE for client side)

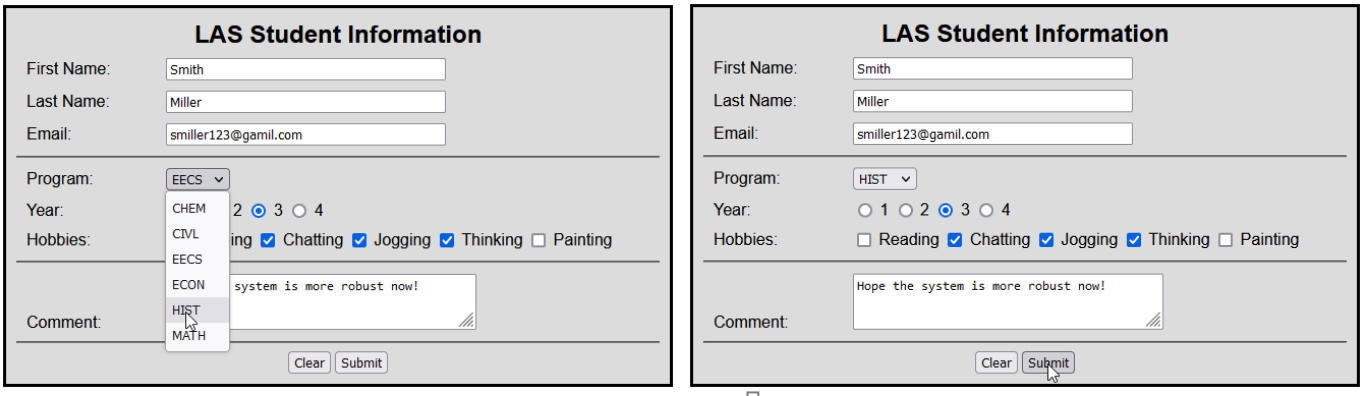
```
Enter your string:
How are you
upper:  HOW ARE YOU
sort:     Haeooruwy
rever:  uoy era woH
isPanl: false


Enter your string:
MamdaM
upper:  MAMDAM
sort:   MMaadm
rever:  MadmaM
isPanl: false

Enter your string:
madam
upper:  MADAM
sort:   aadmm
rever:  madam
isPanl: true

Enter your string:
8469469526acdbehgf
upper:  8469469526ACDBEHGF
sort:   2445666899abcdefgh
rever:  fghebdca6259649648
isPanl: false

Enter your string:
stop
```

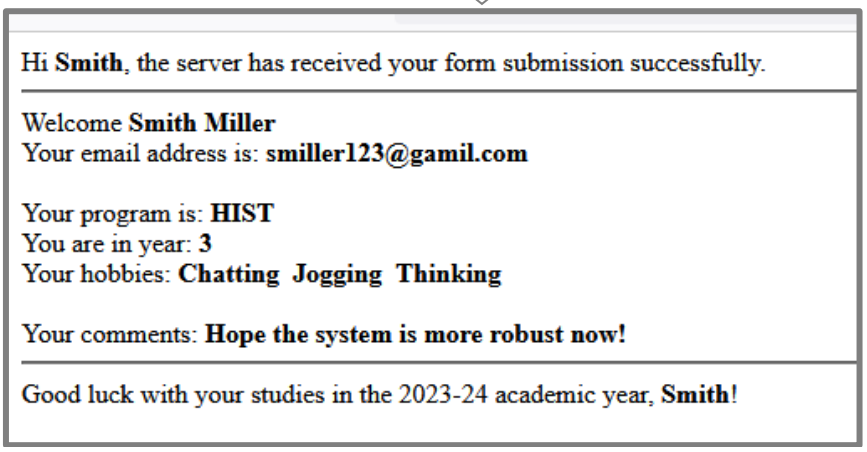
Part II: Simple LAMP and CGI scripting

In this exercise you get exposure to LAMP model and CGI script using PHP.

You are given file **FormsInputLab3.html**, which is a simplified version of the student form as you did in lab1. (So this file also serves as the partial solution to lab1). You are also give the CSS file for the HTML file.

Your job is to create a PHP script that processes the form, and then outputs the result as shown below.





- Name your PHP script *myScript.php*.

- Run script from EECS server.
  - first, put the PHP script to your prism account under the **www** directory. Set the permission of the script to have *r* permission for 'others'. Your **www** folder should have *r* and *x* permission for 'others'.
  - Keep the html at your local computer. Set the form action to **="***http://www.cse.yorku.ca/~yourUserID/myScript.php***"**.
  - Double click the html file to load the local html file (at home machine).
  - If it works, then, put the html file to the **www** folder too, set the permissions and then load the html file by issuing **www.cse.yorku.ca/~yourUserID/FormsInputLab3.html** in your browser.

- Install and run a LAMP server on your local computer
  - **Set up the environment**. There are several pre-configured LAMP models that bundle the components (Apache Web server, MySQL, PHP) together, so you don't need to install separately and configure them (which could be time-consuming). If you are using Windows machine, you can download software WAMP server. If you are using MAC, you can download MAMP server. Alternatively, both Window users and MAC users can download XMAPP server.
  - Put the script in the installation folder of the server you downloaded, e.g., **www** folder in WMAP server, or **htdocs** folder in XMAPP server. Check your server's document if you use other servers. You can test by issuing ***localhost/myScript.php*** in the browser address bar.
  - First, keep the html file outside of the LAMP environment (folder). Set the form action to *="http://localhost/myScript.php"*. Make sure the server is running. Double click the html file to load the html file in your browser. Enter some data and click the submit button.

- Next, put the html file inside the server environment, in the same folder as the PHP script file. Since the HTML file and the PHP script are in the same folder now, you can change the form action to *="myScript.php"* or *"/myScript.php"* or *"./myScript.php"*.
  Load the html by issuing **localhost/FormsInputLab3.html** in your browser, and you should get the same result.

- [extra exercise] You can also try to create a small database from phpadmin from the server, e.g., *courses* in the MySQL database, and then use PHP script to connect to the server, and take some user input to retrieve for the database.

- [extra exercise] you can try to make your local server accessible from the public. This may not be easy from home, as you may need to configure your router (e.g., port forwarding). Instead, bring your laptop to campus, connect to Air York Plus. Then get the public IP of your laptop, and use any other computer (e.g., lab computer or a friend's computer outside York) to connect by issuing
  yourPublicIPaddress/FormsInputLab3.html e.g., 130.68.52.77/ForsInputLab3.html
  in their browsers, and they should see the form and send data. (If you are using WAMP, you may need to change configuration files to grant public access (see https://www.youtube.com/watch?v=gA1NNWIL9jA).

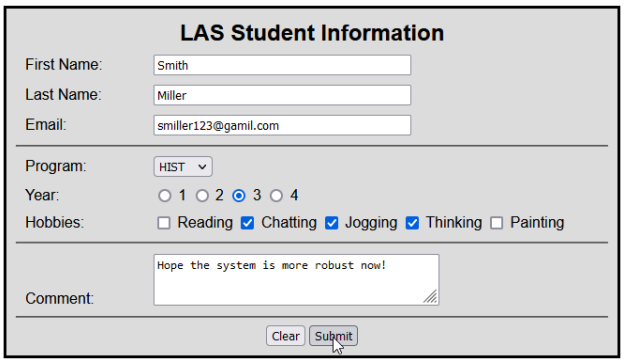
Part III Simple Java Servlet (processing HTML forms)

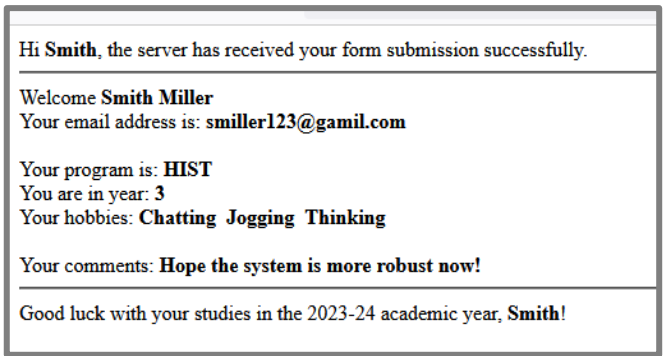
In this exercise, you write the first Java Servlet to process the form.

- **Set up environment**. For this course, we use Apache Tomcat + Eclipse IDE to develop Servlet/JSP programs. Assume you already have "Eclipse IDE for Enterprise Java and Web Developers" installed (probably from the other Java course. Also note this is not the "IDE for Java Developers"). Download Apache Tomcat 9 (the zip file), unzip it, then set up Tomcat as a server in your Eclipse IDE (see the video for a demo).

- In Eclipse, create the Dynamic Web Project, call it ***lab3servlet***, and check 'Generate web.xml deployment descriptor' (this file is not necessarily needed for this project but we just generate it in case).
- Then create a new HTML file in the project, with the same content as the **FormsInputLab3.html** used in part II. Also create the CSS file that has the same content as the one used for part II. Create the files under the default '**webapp'** directory. (Alternatively, you can copy paste the two provided files to the **webapp** folder shown in the Project Explorer pane in Eclipse.)
- Then, create a new Servlet class that processes the form.
- Note that you need to set the form action to the url of your servlet.

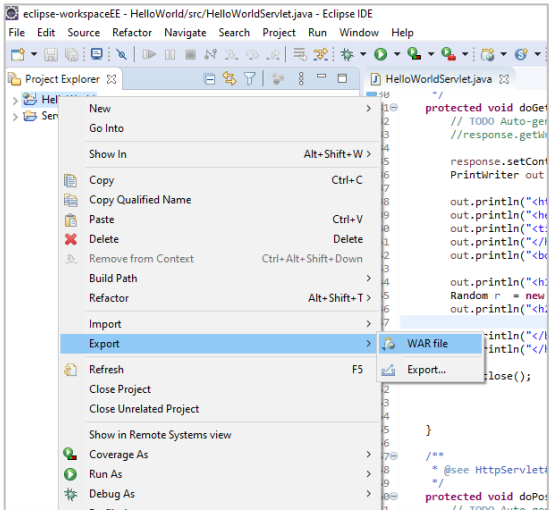
- Load the form by issuing *localhost:8080/lab3servlet/FormsInputLab3.html* in your browser. If implemented correctly, you will get the same result as before, which is repeated here again.

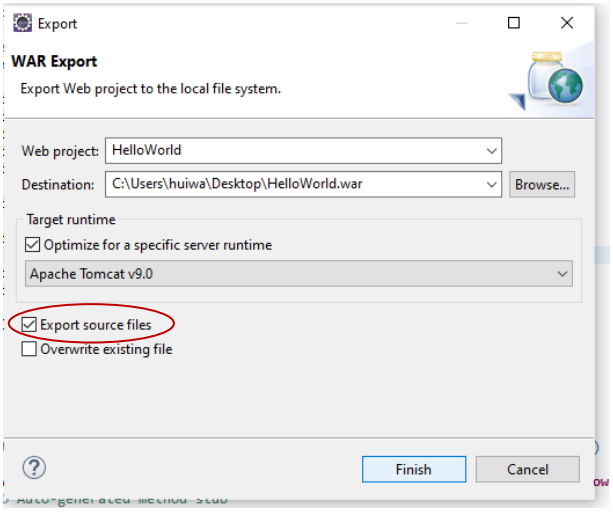




- Note that your program should also work if the form uses the POST method.

- When the program works, export it as 'WAR' file with the default name **lab3servlet.war**. Check the 'Export source files' before you click Finish button.





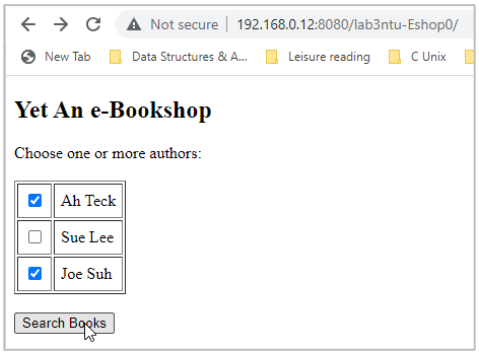
Part IV Simple Java Servlet (generating HTML forms as well as processing HTML forms)

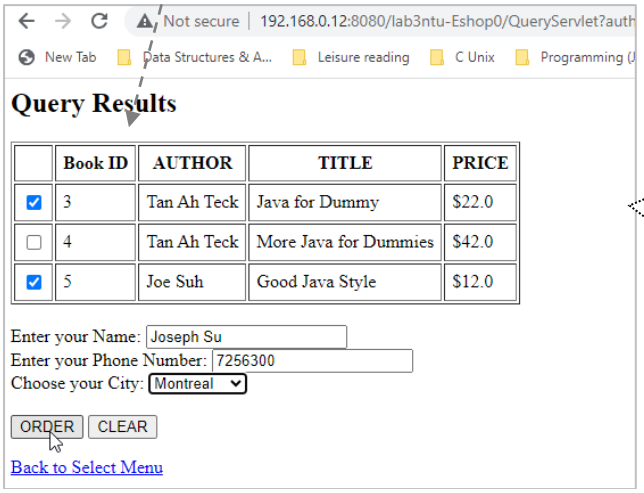
In this exercise, you create dynamic html forms from Java Servlet.

You are provided with a html file and some Java classes, including **Book** class that models books, and a **Table** class that maintains list of books, which serve as database of books – in real application, as we see later, this should be a real database

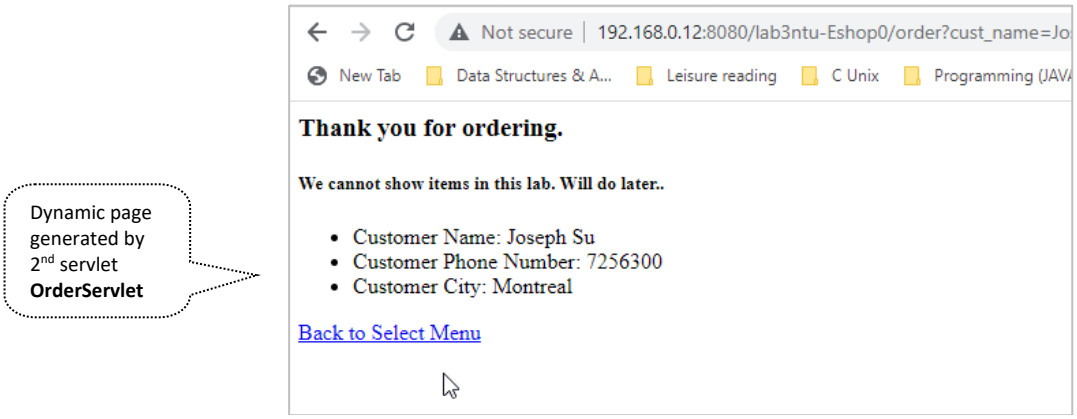
.

The html file provides a table of author names to search. Users can select one or more authors and then submit the form. The form data goes to your first servlet, call it **QueryServlet**. This Servlet processes the form data, searching the books and then generates a dynamic html page showing the result of search. This page contains a form, inside it lists the books by the authors, in a HTML table format. There is also a checkbox for each book record, which allows users to select the book for ordering (not used for this lab but will be used in future labs). The form also contains two textboxes and a dropdown list for users to enter name, phone and city information. When user clicks Order button, it sends the form data to the 2nd servlet, called **OrderServlet**, which reads the form data and displays some information (similar to Part III).





Dynamic page generated by 1st servlet **QueryServlet**



- Create a Dynamic Web Project, call it ***lab3servletQ4***, check 'Generate web.xml deployment descriptor'.
- Then create an HTML file called **index.html** and copy the content of the provided HTML file in it. (Alternatively, you can copy the whole html file into the project on eclipse in Project Explorer). In either case, put the HTML file under 'webapp' directory.
- Put the provided Java files **Book.java** and **Table.java** into the project, under the 'default package'. Complete the class **Table**.
- Create a new servlet **QueryServlet** to process the html form data and generate a HTML form and table that contains a list of books. Generating html forms in Servlet could be a bit tricky, so you are provided with some partial code for this servlet to help you. You can also view the course code of **index.html** to see how a HTML table is generated. W3schools is another good resource. Use annotation or deployment descriptor to define the URL mapping of the servlet.
- Then, create the 2nd servlet **OrderServlet** to process the form generated by the first servlet.
- Make sure the form of the html file has the action to the correct URL of your first servlet **QueryServlet**. Likewise, make sure the form generated by the **QueryServlet** has the action to the correct URL of the 2nd servlet **OrderServlet**.
- Also make sure that both the generated dynamic pages provide a hyperlink that link back to the index.html file.
- Run the program by issuing ***localhost:8080/lab3servletQ4/index.html*** or simply ***localhost:8080/lab3servletQ4*** in your browser. (Note that since the html file is named index.html, which is in the welcome list of the auto-generated deployment descriptor **web.xml**, just ***localhost:8080/lab3servletQ4*** will also load the html file.)

- Note that your program should also work if the forms use the POST method.

- When the program works, export it as 'WAR' file with the default name '**lab3servletQ4.war'**. Check the 'Export source files' before you click Finish.

- Finally, we may sense that the order form is very simple and even did not list the selected books (the checkbox in the second page is not used yet). In future labs, we may add more functionalities. Also, you may also sense that generating HTML pages is not that straightforward, that's one reason that JSP or other technologies comes into play. We will explore JSP later.

Submissions.

You should already have a **4413Lab03** folder that contains the 4 folders for the Part I - IV.

For part I, put your java programs in that Part I folder. You are expected to develop the Java program using plain editor and command line, put the java files inside the corresponding client and server java folders.

If you develop using Eclipse, export project as Archive File (zip file), and put the zip file in the corresponding java folders.

For part II, put your **myScript.php** and the **FormsInputlab3.html** form in that Part II folder.

For part III and IV, you should have exported **lab3servlet.war** and **lab3servletQ4.war**. Make sure you have checked "Export source files" when exporting. Put them on the top level of the lab folder, or their corresponding folders.

Finally, compress the *4413Lab03* folder (.zip or .tar or .gz), and then submit the (single) compressed file on eClass.

Please don't include the videos in your submission package.

Late submissions or submissions by email will NOT be accepted. Plan ahead and submit early.