# EECS 4413. LAB 04: More on Servlets -- more forms, servlet context, session tracking

## A. IMPORTANT REMINDERS

- **This lab will be GRADED**.
- Lab4 is due on **Tuesday (Feb 27) at 11pm**. No late submission will be accepted.
- For this lab, you are welcome to attend the lab sessions on Feb 27. TAs or the instructor will be available to help you. The location is LAS1002. Attendance is optional. You can also ask questions after Feb26's class.
- In the lab, feel free to signal a TA for help if you are 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.
- The two war files are for Part II B (not graded)
- The UI.html and CSS files are for Part III.

.

## C. GOALS/OUTCOMES FOR LAB

- More on generating forms in Servlet.
- To learn/recap the fundamental servlet features, including context level initial values, dispatchers, attributes, session tracking.

## D. TASKS

**Part1:** More on Java Servlet program – processing and generating forms.

**Part 2:** Simple session tracking with hidden fields.

**Part 2B [not graded]** Simple session tracking with cookie and http session.

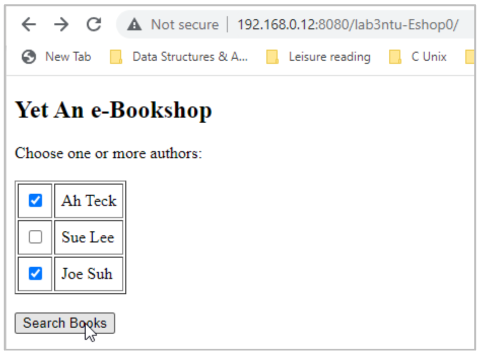
**Part 3:** Servlet context, Request dispatcher, Session tracking

## E. SUBMISSIONS

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

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

In lab3 Part IV, you created dynamic html forms from Java Servlets.
The html file renders a table of author names to search. Users can select one or more authors and then submit the form. The form data goes to the 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 but it was not used in lab3. 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.

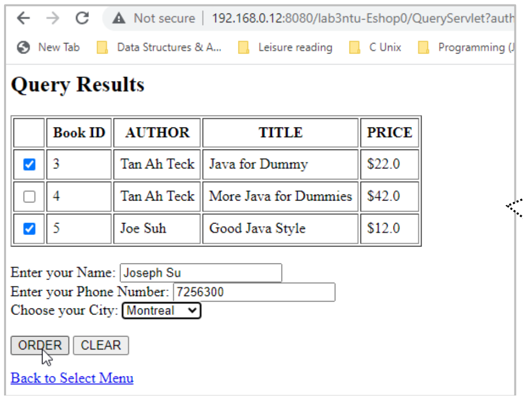


Yet An e-Bookshop

Choose one or more authors:

| | |
|---|---|
| ✓ | Ah Teck |
|  | Sue Lee |
| ✓ | Joe Suh |

⇨

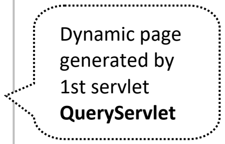


Query Results

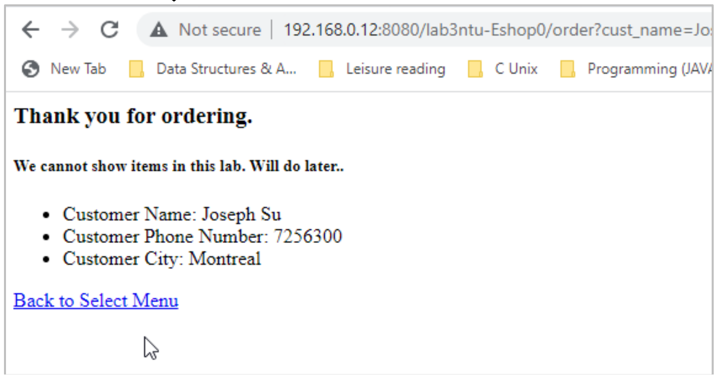
| | Book ID | AUTHOR | TITLE | PRICE |
|---|---|---|---|---|
| ✓ | 3 | Tan Ah Teck | Java for Dummy | $22.0 |
|  | 4 | Tan Ah Teck | More Java for Dummies | $42.0 |
| ✓ | 5 | Joe Suh | Good Java Style | $12.0 |

Enter your Name: Joseph Su
Enter your Phone Number: 7256300
Choose your City: Montreal

Back to Select Menu



⇩



Thank you for ordering.

We cannot show items in this lab. Will do later..

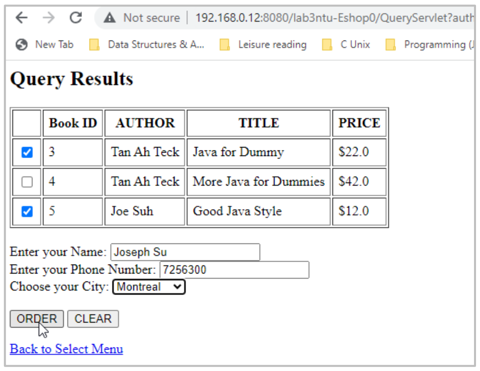
- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

In this lab, you extend your lab3 program, so that after the user to select the books, the **OrderServlet** will give a list of books that <u>the user selected</u>. Also, we want to add a quantity field in the page where the user can update the order quantity.

- Import your **lab3servletQ4.war** file that you submitted, as new project, name it ***lab4servletQ1.***

- First, to make the checkbox selection info passed to the **Orderservlet**, we need to give the checkboxes a name, and value of each checkbox is the id of the item it checked.
- Then, modify the **OrderServlet** to list all the books selected. To do this, get the checkbox parameter and retrieve the id values. Then for each id, get the book info by searching the book by the id. And then output the id, author and price information, and list the record in table format.


Not secure | 192.168.0.12:8080/lab3ntu-Eshop0/QueryServlet?auth
New Tab
Data Structures & A...
Leisure reading
C Unix
Programming

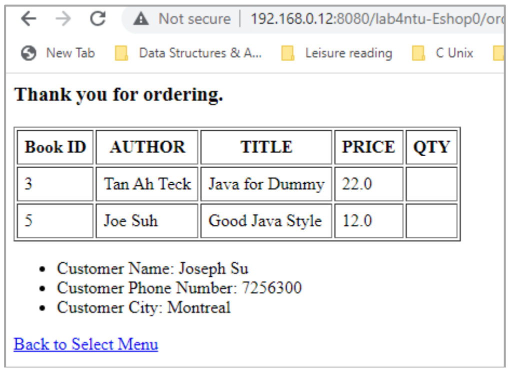
**Query Results**

| | Book ID | AUTHOR | TITLE | PRICE |
|---|---|---|---|---|
| ✓ | 3 | Tan Ah Teck | Java for Dummy | $22.0 |
| ☐ | 4 | Tan Ah Teck | More Java for Dummies | $42.0 |
| ✓ | 5 | Joe Suh | Good Java Style | $12.0 |

Enter your Name: Joseph Su
Enter your Phone Number: 7256300
Choose your City: Montreal

ORDER CLEAR

Back to Select Menu



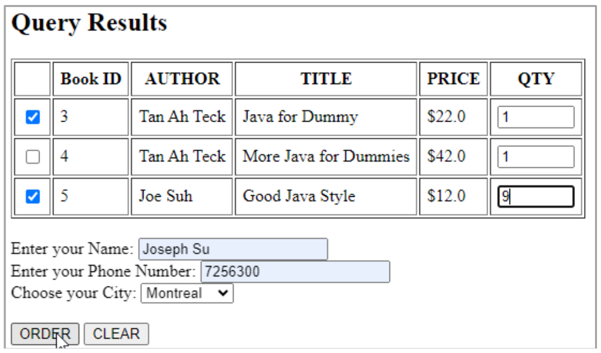
**Thank you for ordering.**

| Book ID | AUTHOR | TITLE | PRICE | QTY |
|---|---|---|---|---|
| 3 | Tan Ah Teck | Java for Dummy | 22.0 | |
| 5 | Joe Suh | Good Java Style | 12.0 | |

- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

- By now, the program looks more interactive than the lab3 version. But it still lacks quantity and price information, which we will add now.
- In the **QueryServlet** table, add a quantity textbox, which has an initial value 1 for each book. To pass the quantity info to the order servlet, we need to give each textbox a unique name. One approach is to use the ***"qty"+id*** as the name. For example, for book of id 2, the quality textbox's name should be "qty2". For book of id 5, the quality textbox's name should be "qty5". Something like *out.println( "<input ...... name='qty"+ id + " ' />" )* . You may also want to use *size* attribute of the input tag to control the width of the textbox.
  This allows the other servlet to retrieve this quantity information for a particular book by its id -- from parameter information.
- Then in the **OrderServlet**, for each record, also retrieve the quantity information by the parameter of "qty"+id. List this information the table.

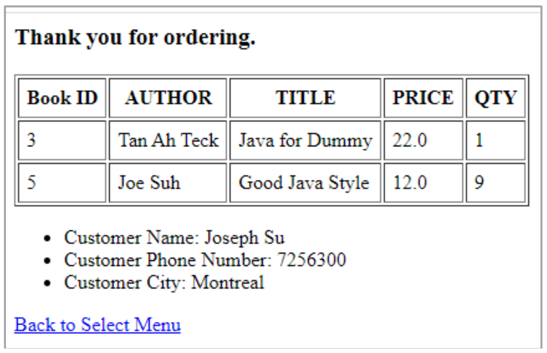


**Query Results**

| | Book ID | AUTHOR | TITLE | PRICE | QTY |
|---|---|---|---|---|---|
| ✓ | 3 | Tan Ah Teck | Java for Dummy | $22.0 | 1 |
| ☐ | 4 | Tan Ah Teck | More Java for Dummies | $42.0 | 1 |
| ✓ | 5 | Joe Suh | Good Java Style | $12.0 | 9 |

Enter your Name: Joseph Su
Enter your Phone Number: 7256300
Choose your City: Montreal

ORDER CLEAR



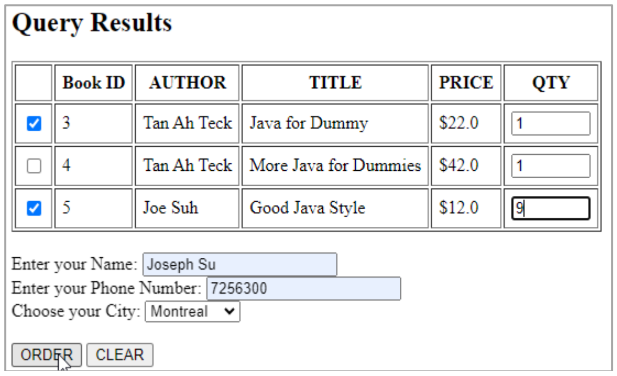
**Thank you for ordering.**

| Book ID | AUTHOR | TITLE | PRICE | QTY |
|---|---|---|---|---|
| 3 | Tan Ah Teck | Java for Dummy | 22.0 | 1 |
| 5 | Joe Suh | Good Java Style | 12.0 | 9 |

- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

- Lastly, improve on the program by calculating and showing the total price of the order. When listing the orders, add up the price and display as the last row of the table (hint: you can use *<td colspan='5'>* to make the last row merged).

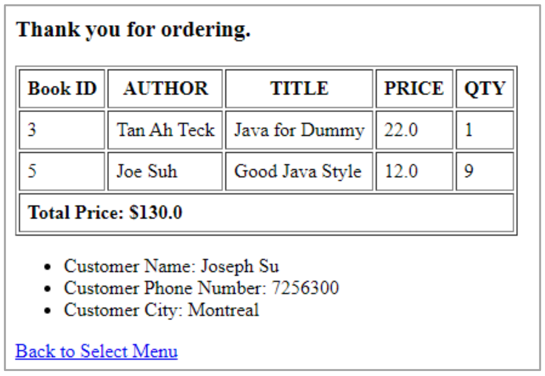


**Query Results**

| | Book ID | AUTHOR | TITLE | PRICE | QTY |
|---|---|---|---|---|---|
| ✓ | 3 | Tan Ah Teck | Java for Dummy | $22.0 | 1 |
| ☐ | 4 | Tan Ah Teck | More Java for Dummies | $42.0 | 1 |
| ✓ | 5 | Joe Suh | Good Java Style | $12.0 | 9 |

Enter your Name: Joseph Su
Enter your Phone Number: 7256300
Choose your City: Montreal

ORDER CLEAR



**Thank you for ordering.**

| Book ID | AUTHOR | TITLE | PRICE | QTY |
|---|---|---|---|---|
| 3 | Tan Ah Teck | Java for Dummy | 22.0 | 1 |
| 5 | Joe Suh | Good Java Style | 12.0 | 9 |
| **Total Price: $130.0** | | | | |

- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

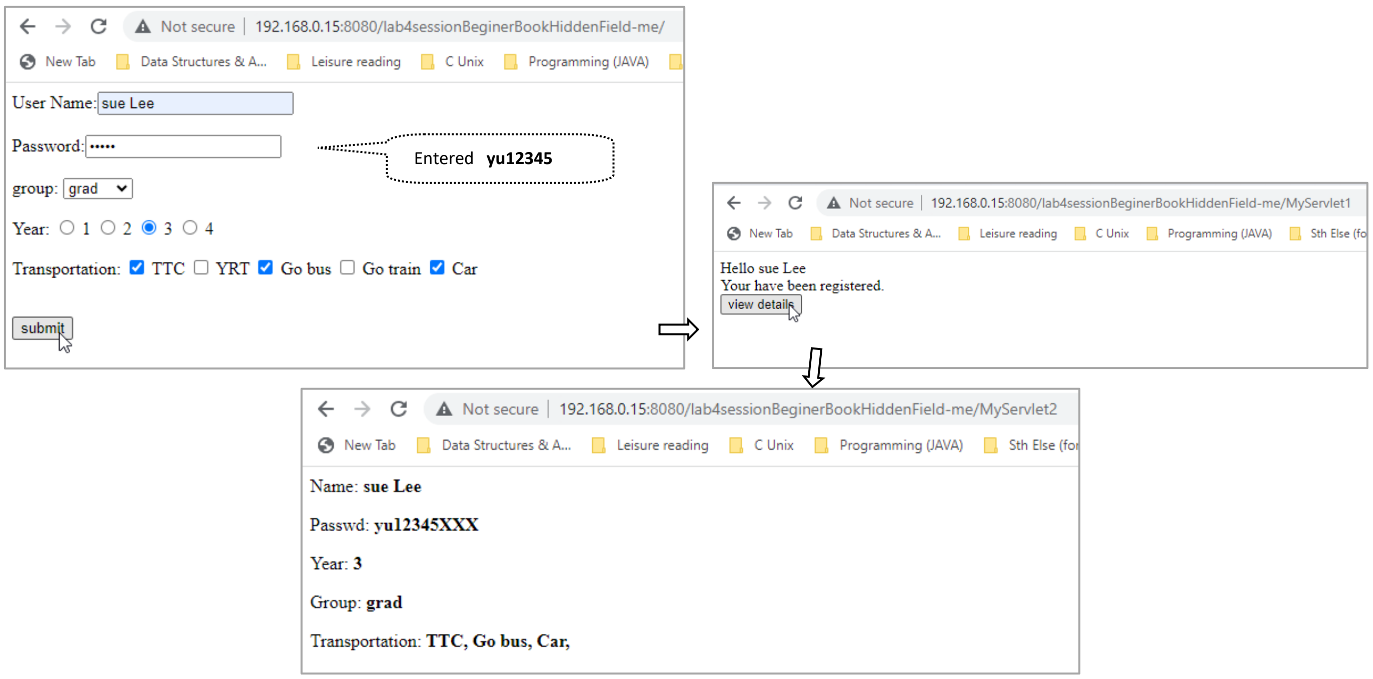
See more output from the video.

- Once the program works, export it as 'WAR' file with the default name **lab4servletQ1.war**. Check the 'Export source files' before you click Finish.

- This program is closer to a real shopping website than the lab3 version. But it does not support 'shopping cart' experience: after one order, when user goes back to select again, the previous order information is lost. In real e-commerce applications, we want to keep the selection in shopping cart and then user go back to select more items, the items are added the existing shopping cart. We need session tracking to do this. In exercise 3 of this lab, you will do a simple tracking. In later labs we continue with this example to add shopping cart functionality.

**Part II:** use hidden fields to pass user information (This is a very simple and short exercise.)

- There is a file index.html that asks user for some information. Then the form data is sent to a servlet. This servlet displays some information and then provides a button that leads to the second servlet. The second servlet will display the information that the user entered.
- How can the 2nd servlet get the user entered information? You can use cookie, Http Session, but here we experience a simple approach: hidden fields. The first servlet, in addition to generating some information, also generate a form with hidden field of the user information. The servlet adds 'XXX' to the end of the entered password, and process the checkbox input so it can be put as a hidden field.
- When the button is clicked, the hidden data in the form is sent to the 2nd servlet. The second servlet will just retrieve them as the regular form data.


Not secure
192.168.0.15:8080/lab4sessionBeginerBookHiddenField-me/
New Tab
Data Structures & A...
Leisure reading
C Unix
Programming (JAVA)
Entered yu12345
submit
192.168.0.15:8080/lab4sessionBeginerBookHiddenField-me/MyServlet1
Sth Else (fo
view details
192.168.0.15:8080/lab4sessionBeginerBookHiddenField-me/MyServlet2

User Name: sue Lee

Password: •••••

group: grad

Year: ○ 1 ○ 2 ● 3 ○ 4

Transportation: ☑ TTC ☐ YRT ☑ Go bus ☐ Go train ☑ Car

Hello sue Lee
Your have been registered.

Name: **sue Lee**

Passwd: **yu12345XXX**

Year: **3**

Group: **grad**

Transportation: **TTC, Go bus, Car,**

- Note that both the html form and the generated form should use POST method, so password and other information will not be shown in the URL bar.
- Note that as mentioned in class, hidden fields approach has some limitations. One of them is that it needs a form with buttons. If we want to provide hyperlink to the 2nd servlet, instead of a button, then it will not send to the hidden field information. In that case, cookie and http session are better choices.
- To do:
  - Create a dynamic project, name it lab4servletQ2. Create a **index.html** file, and save in the webapp folder.
  - Create the two servlets to fulfill the above tasks.

- Once the program works, export it as 'WAR' file with the default name '**lab4servletQ2.war'**. Check the 'Export source files' before you click Finish.

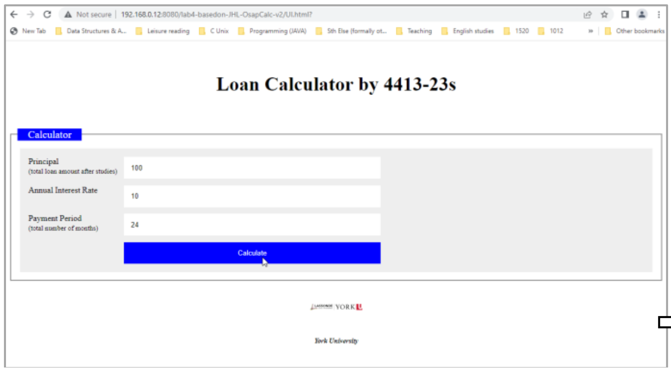
**Part II B (not graded, no submission)**

- You are provided with an implementation of the above problem using cookies. Import the war file **lab4servletQ2cookie.war**. Read and run the code. As you can see, using cookies allows us to pass information using approaches other than forms (e.g., hyperlinks). The drawback of using cookie, as you can see, is that cookies only take string values (no objects), so a cookie has to be created for each piece of information. In addition, sometimes cookies do not allow space, comma in values.
- You are provided with an implementation of the above problem using HttpSession. Import the war file **lab4servletQ2httpSession.war**. Read and run the code. Using HttpSession is easier than maintaining cookies. Moreover, HttpSession allows passing object as attribute. Refactor the provided code into an "OOP version", in which the user input fields are encapsulated into an object and passed as attribute of the HttpSession object.

**Part III:** Servlet context, Request dispatcher, Session tracking
In this exercise you implement a simple OSAP calculator, using the advanced Servlet features that is covered recently.

- You are given a UI.html and a CSS file. The UI file allows users to enter *Principal*, *Interest Rate* and *Payment Period* and submit to a servlet to calculate the monthly payment for the loan.
- When a user enters the *Principal*, *Interest Rate* and *Payment Period* and submit the form, the data is sent to the **OSAP** servlet, which extracts the three values, and calculates the monthly payment of this loan. Note that if a value is not entered, the servlet retrieves the default initial value set as a context parameter in the deployment descriptor file *web.xml*. The default values are: Principal 1000, Interest Rate 10, Payment Period 12. This way, the application developer can change the defaults without recompiling the servlet. Compute the monthly payment using the formular: $\frac{\frac{r}{12*100}*A}{1-(1+\frac{r}{12*100})^{-n}}$ where *r* is the annual interest rate, *A* is the principal, and *n* is the period measured in months. *r* and *A* can be floating point values.
- After calculating the result, the **OSAP** servlet sends the information to another servlet **ResultView** to display the result. The **OSAP** servlet uses RequestDispatcher to forward the request and response to the **ResultView** servlet. The **OSAP** servlet needs to pass 4 pieces of information to the **ResultView** servlet: principal, interest rate and period that are <u>used for calculation</u>, as well as the calculated loan payment. Note that to pass these 4 pieces of information, the servlet needs to set the attributes for each. Note that these attributes can be set in *application*, *request* or *session* scope. For these 4 values, *request* scope is sufficient, as the 2nd servlet will retrieve it from the forwarded request.
  This OSAP servlet is also responsible for checking if the connection is the first time by this client or a revisit, using http sessions. If this is the first-time visit, it creates a new http session and sets a *count* attribute to store the visit count information. If this is not the first-time visit, it should retrieve the *count* attribute information from the session and update with incremented visit count. Note that this should be a *session* scope attribute, as it is the connection between this client and the sever. Another client connected to this application should have a separate visit count information.
  Note that this OSAP servlet does not output anything to the client. It calculates the loan payment and forwards the relevant information to the 2nd servlet **ResultView**, who is responsible for displaying the result to the client.
- The **ResultView** servlet will first display some header information of the client request. Note that such information can be retrieved directly from the forwarded request object. It also displays the <u>user entered</u> *principle*, *interest* and *period* information. Note that this information may not necessarily be the same as the <u>used</u> principal, interest and period information – if the user did not enter an information, the default context parameter value was used instead. This information can also be retrieved directly from the forwarded request (how?).
  Next, the servlet retrieves sessional attributes about the visit count. If this is the first-time visit, display

“Welcome …” otherwise display “Welcome back. Calculation count: X”.
Next, the servlet displays the OSAP result, as well as the *principal*, *interest* and *period* information that were actually used in the calculation. Recall that this information has been set by the OSAP servlet as request cope attributes.
Note that in generating the dynamic HTML, you should also link to the provided CSS file so the HTML page uses the CSS styles.
This servlet also generates a “recalculate” button which will lead the user to the original UT html. When the client recalculates within a certain period of time (like within 30 minutes), it will be recognized as the re-visitor and its visiting count will be updated (in the **OSAP** servlet) and displayed (in the **ResultView** servlet).



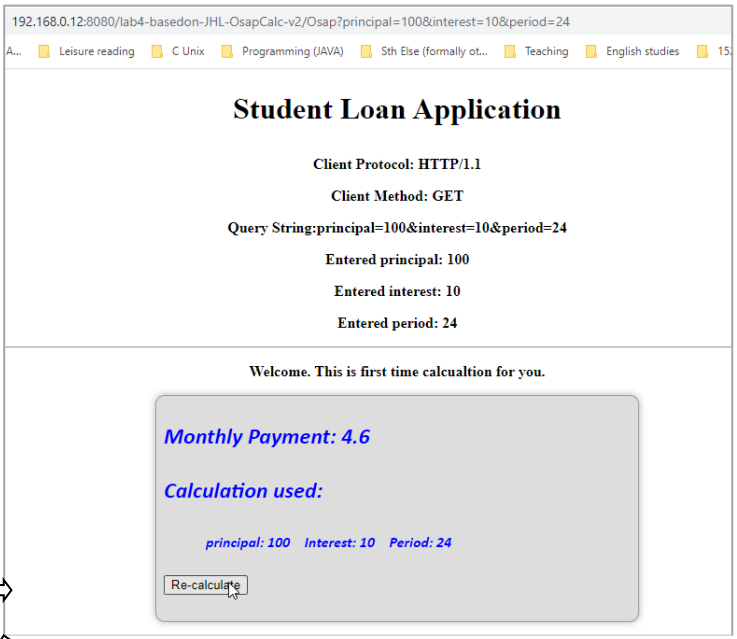
Loan Calculator by 4413-23s

Calculator

Principal (total loan amount after studies): 100
Annual Interest Rate: 10
Payment Period (total number of months): 24

Calculate

York University



Student Loan Application

Client Protocol: HTTP/1.1

Client Method: GET

Query String:principal=100&interest=10&period=24

Entered principal: 100

Entered interest: 10

Entered period: 24

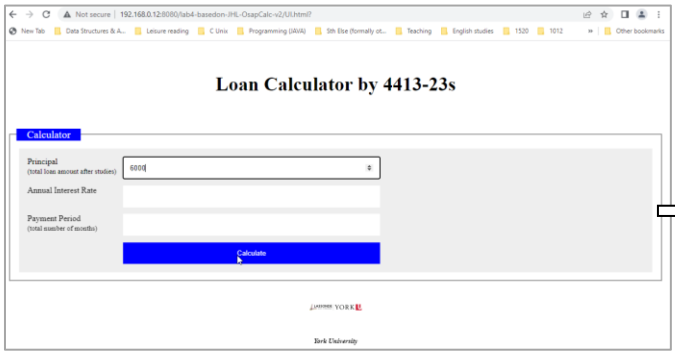
Welcome. This is first time calcualtion for you.

Monthly Payment: 4.6

Calculation used:

principal: 100 Interest: 10 Period: 24

Re-calculate



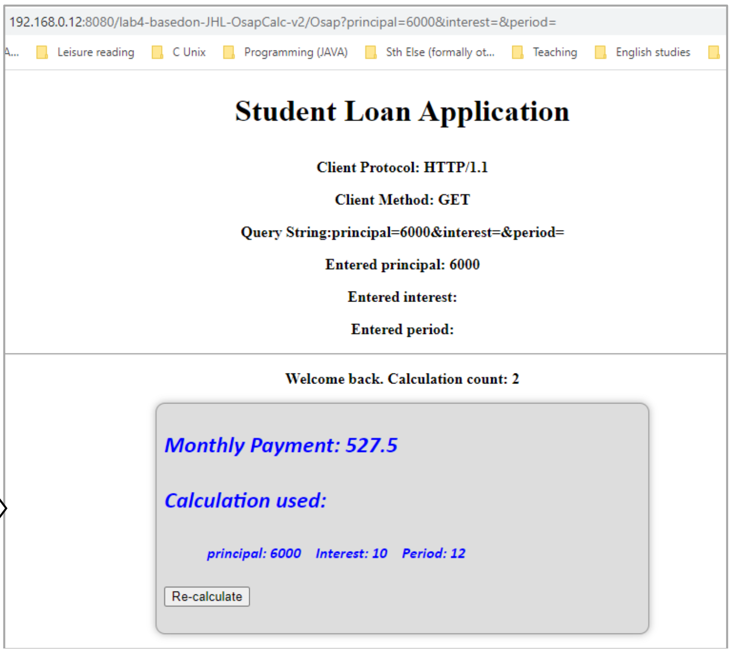
Loan Calculator by 4413-23s

Calculator

Principal (total loan amount after studies): 6000
Annual Interest Rate:
Payment Period (total number of months):

Calculate

York University



Student Loan Application

Client Protocol: HTTP/1.1

Client Method: GET

Query String:principal=6000&interest=&period=

Entered principal: 6000

Entered interest:

Entered period:

Welcome back. Calculation count: 2

Monthly Payment: 527.5

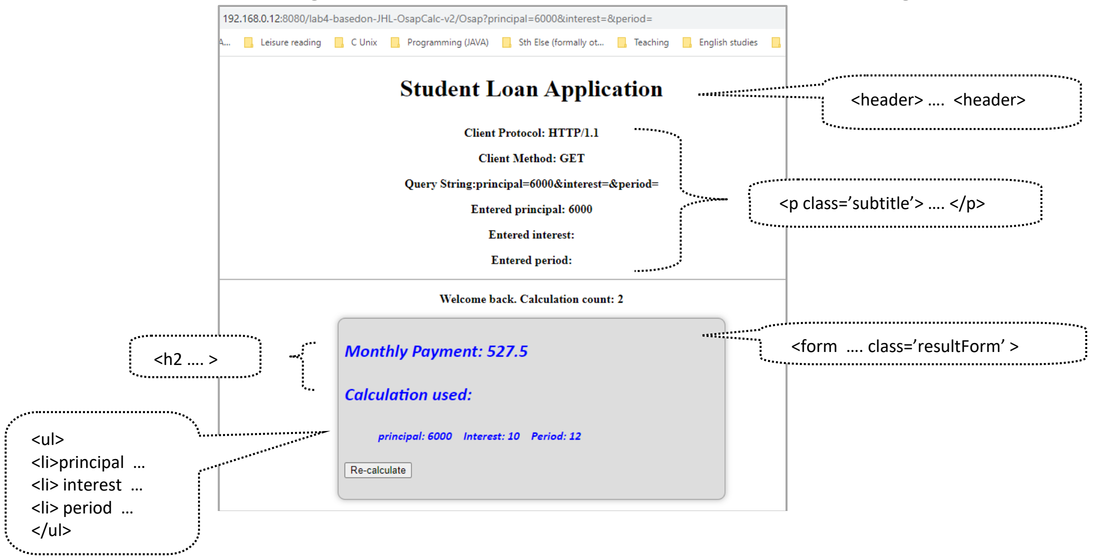
Calculation used:

principal: 6000 Interest: 10 Period: 12

Re-calculate

During the connection, if another client connects, it should be recognized as the new client. You can simulate another client by using another computer to connect. You can also use another browser in the working computer to connect, and it will be treated as another client/session. If you connect with the same browser but after a while (>30 minutes) the previous session expires, and following connection will also be treated as a new client/session.

- To do:
  - Create a dynamic project, name it **lab4servletQ3**. Put the provided html and CSS file into the project. Add the context level initial values to the deploy descriptor **web.xml**.
  - Create the two servlets to fulfill the above tasks.
  - For the generated form from **ResultView**, link to CSS and give the following styles:



192.168.0.12:8080/lab4-basedon-JHL-OsapCalc-v2/Osap?principal=6000&interest=&period=

**Student Loan Application**

Client Protocol: HTTP/1.1

Client Method: GET

Query String:principal=6000&interest=&period=

Entered principal: 6000

Entered interest:

Entered period:

Welcome back. Calculation count: 2

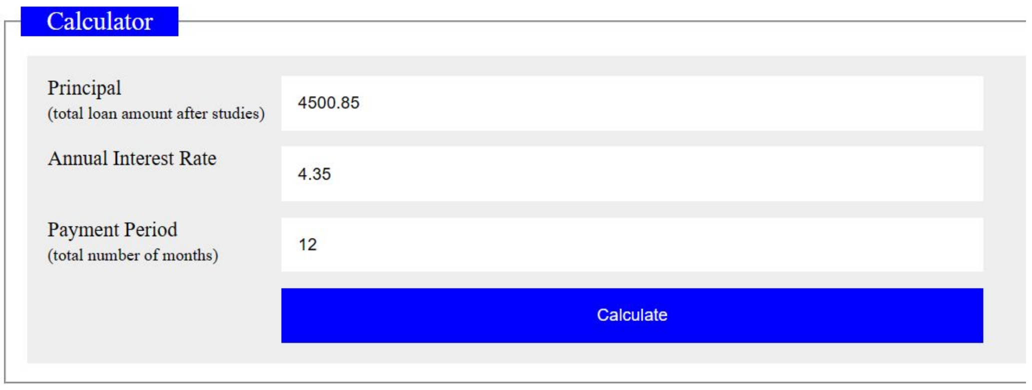
***Monthly Payment: 527.5***

***Calculation used:***

*principal: 6000   Interest: 10   Period: 12*

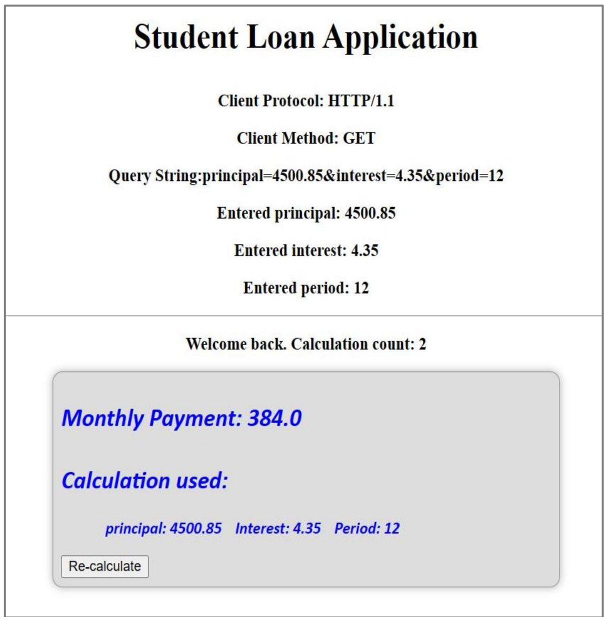
  - Once the program works, export it as 'WAR' file with the default name **lab4servletQ3.war**. Check the 'Export source files' before you click Finish.

Another example, with floating point input for Principal and Interest Rate. The payment should keep 1 or 2 decimal digits.



Calculator

| | |
|---|---|
| Principal<br>(total loan amount after studies) | 4500.85 |
| Annual Interest Rate | 4.35 |
| Payment Period<br>(total number of months) | 12 |


Student Loan Application

Client Protocol: HTTP/1.1

Client Method: GET

Query String:principal=4500.85&interest=4.35&period=12

Entered principal: 4500.85

Entered interest: 4.35

Entered period: 12

Welcome back. Calculation count: 2

Monthly Payment: 384.0

Calculation used:

principal: 4500.85 Interest: 4.35 Period: 12

**Submissions.**

You should have exported **lab4servletQ1.war lab4servletQ2.war** and **lab4servletQ3.war**. Make sure you have checked “Export source files” when exporting.

Submit the 3 war files separately on eClass.

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