EECS 4413. LAB 05: More on session tracking in Servlets, JSP, MVC model, more on Deployment descriptor file

A. IMPORTANT REMINDERS

- **This lab will be graded.**
- Lab5 is due on **May 6 (Monday) at 11pm**. No late submission will be accepted.
- For this lab, you are welcome to attend the lab sessions on ~~this and~~ next Tuesday. TAs or instructor will be available to help you. The location is LAS1002 or online TBD. Attendance is optional. You can also ask questions on Monday (May 6 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.

C. GOALS/OUTCOMES FOR LAB

- More on session tracking on Servlet.
- JSP.
- MVC model.

D. TASKS

Part1A: JSP for views.

Part1B: MVC model

Part 2A: More on session tracking

Part 2B: JSP for views

E. SUBMISSIONS

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

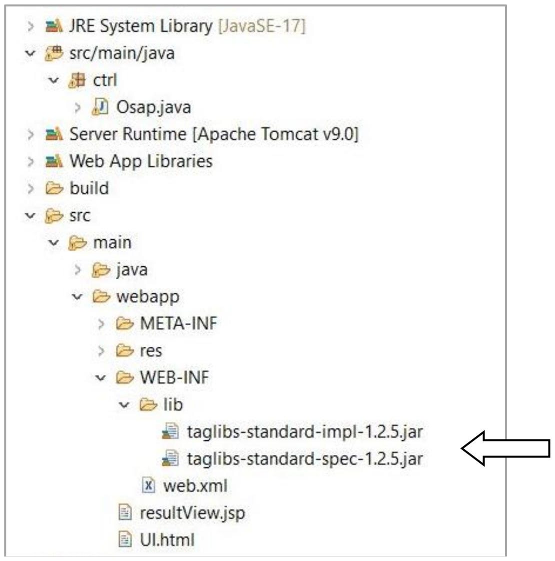
Part I A: Use JSP for view, separating logic and presentation (this is relatively simple and short exercise)

In the OSAP calculation exercise in the previous lab, the OSAP servlet performs the parameter processing and loan calculation, and then uses dispatcher to forward the another servlet ResultView for displaying the result. In this exercise you use JSP to replace the 2nd servlet ResultView.

- Import your **lab4servletQ3.war** file that you submitted for lab4, as a new project, name it **lab5servletQ1A.**
- In **OSAP** servlet, instead of dispatching to servlet ResultView, now dispatch to "**ResultView.jsp**"
- Create a new JSP file named "**ResultView.jsp**" and save it in the default location (under Webapp).
- In the JSP file, do all the display that servlet ResultView did.
  - Note that for the header information, you can use scriptlet or expression, with the implicit object *request*, e.g., **<%= request.getProtocol() %>** or **<% out.print(request.getProtocol()) %>**
  - For the different scoped parameters and attributes, you should use EL language, e.g., **${param.principal}** or **${param['principal']}, ${payment}** or **${requestScope.payment}, ${count}** or **${sessionScope.count}**
  - you need a 'if else' block to check if this is first time visit or a re-visit, and then display accordingly. You can use scriptlet <% if … else… > to embed java code directly, but we want to avoid that as much as possible. Instead, we use the core JSTL library tag <c:choose>, along with EL, with code like

```
<c:choose>
  <c:when test= "${sessionScope.count > 1}"> <p class='subtitle'>Welcome back, Calculation count: ${count}
    </p></c:when>
  <c:otherwise> <p class='subtitle'>Welcome. This is first time calculation for you. </p></c:otherwise>
 </c:choose>
```

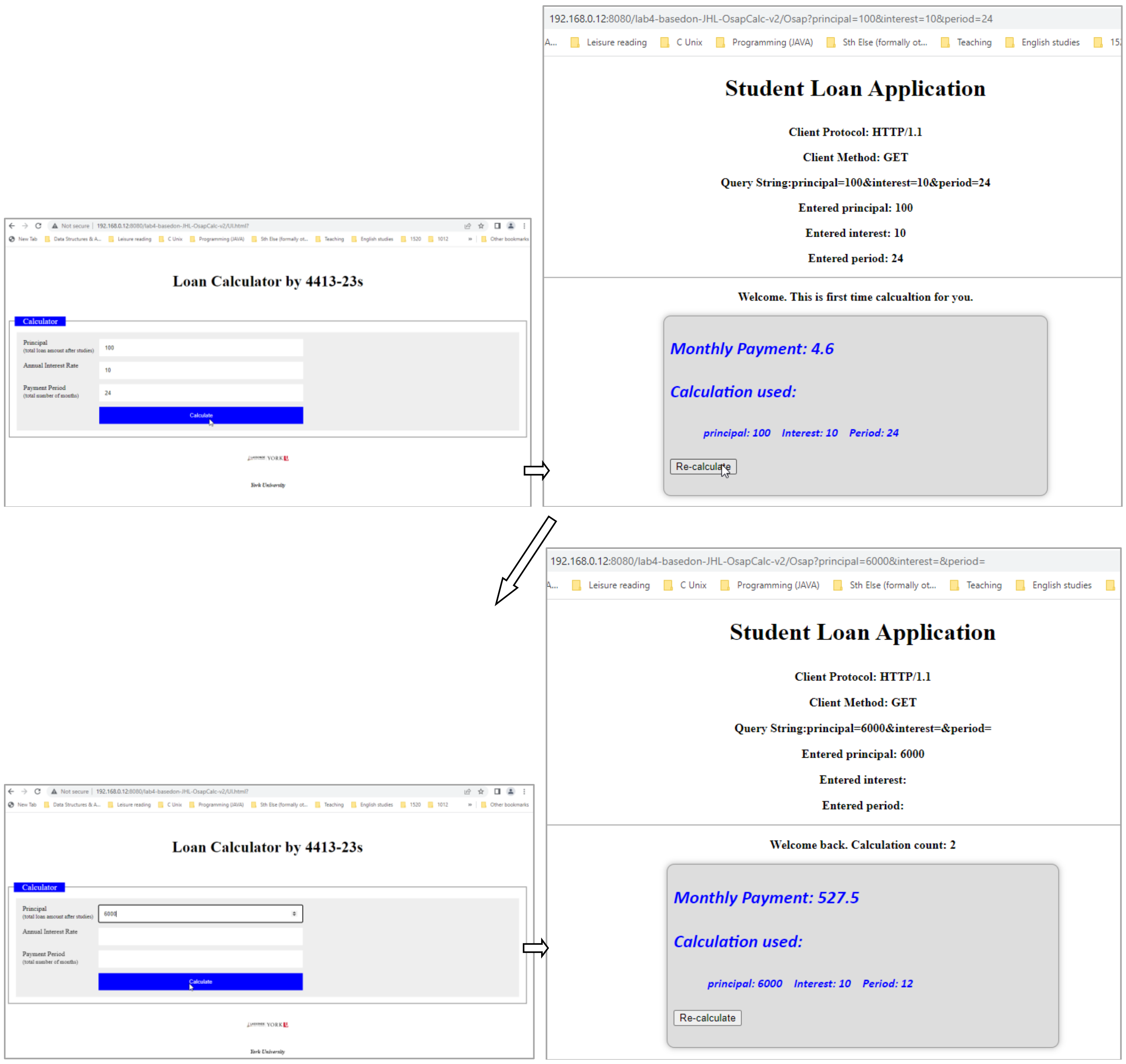
    - To use JSTL core library, you need to do two more things 1) put the import line **<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>** before the <c:choose> block (e.g., on top of the file or body part), 2) put library jar files into the lib folder of your project, as shown in the figure below. You are provided with one version of the jar files, which contains the two jar files. (These two jar files are used in Tomcat's example code, so they can also be found at *yourTomcatInstallDir/webapps/examples/WEB-INF/lib*



```
> JRE System Library [JavaSE-17]
v src/main/java
  v ctrl
    > Osap.java
> Server Runtime [Apache Tomcat v9.0]
> Web App Libraries
> build
v src
  v main
    > java
    v webapp
      > META-INF
      > res
      v WEB-INF
        v lib
            taglibs-standard-impl-1.2.5.jar
            taglibs-standard-spec-1.2.5.jar
          web.xml
        resultView.jsp
        UI.html
```

- Now, you can delete the ResultView.java servlet as it is replaced with the JSP file.
- Finally, add the **UI.html** to the deployment descriptor file web.xml's welcome file list, so that it can also be loaded without explicit path name. That is, in addition to using ***localhost8080:lab5servletQ1A/UI.html***, just entering ***localhost8080:lab5servletQ1A*** loads the html file too.

Now you should have the same output as lab4, which is repeated here.


192.168.0.12:8080/lab4-basedon-JHL-OsapCalc-v2/UI.html?

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

192.168.0.12:8080/lab4-basedon-JHL-OsapCalc-v2/Osap?principal=100&interest=10&period=24

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

192.168.0.12:8080/lab4-basedon-JHL-OsapCalc-v2/UI.html?

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

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

Re-calculate

Same as before, during the connection if another client connects, it should be recognized as the new client. You can simulate another client by using another computer or another browser in the same computer

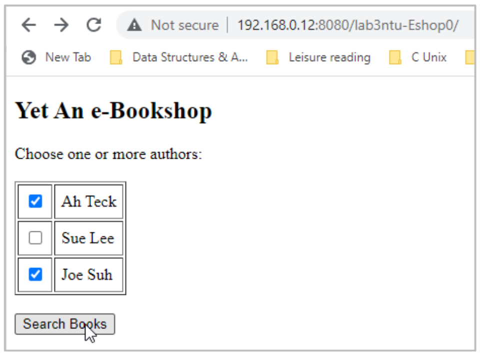
- Once the program works, export it as 'WAR' file with the default name '**lab5servletQ1A.war**'. Check the 'Export source files' before you click Finish.

Part I B: use JSP for view, MVC model, bean access. (This is relatively simple and short exercise.)
In this exercise you refactor the program in part I A, by implementing the MVC model, and use bean class to model the OSAP info.

- Import your **lab5servletQ1A.war** file that you exported in part A1, as a new project, name it **lab5servletQ1B.**
- Create a package **controler**, and move the OSAP servlet file there.
- Create a package **model**, and create a new class **Loan** in the package. The class has attributes (instance variables) *interest*, *period*, *principal* and *payment*, and follow the Java Bean property API for getters and setters of the attributes. It also has a method **void calculatePayment (),** which calculates loan payment using its attributes *interest*, *period* and *principal* and set result to its attribute *payment*.
- Modify the **OSAP** servlet program, such that, after processing the principal, interest and period parameters, it creates a Loan object with the 3 values, and then call *calculatePayment()* on this Loan object to calculate the payment. Then, before dispatching to the JSP, set the object as a request scope attribute with name "load". By encapsulating the values into the bean object, we now pass the single object to the JSP as a scoped attribute, instead of setting several attribute values as before.
- Then in JSP, access the bean loan and its instance variables *principal*, *interest*, *period* and *payment*.
- You should get the same result as before.
- Once the program works, export it as 'WAR' file with the default name '**lab5servletQ1B.war**'. Check the 'Export source files' before you click Finish.

Part II A More on the bookshop, session tracking

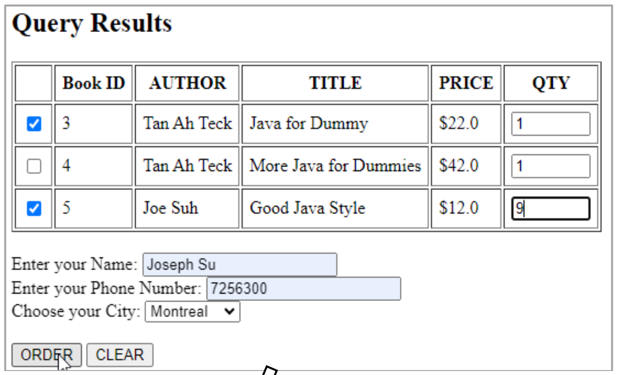
Recall in lab4, we have implemented the bookshop with the following functionalities.



Yet An e-Bookshop

Choose one or more authors:

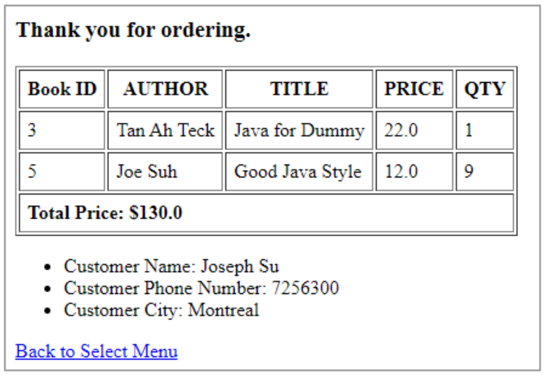
✓	Ah Teck
☐	Sue Lee
✓	Joe Suh



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



Thank you for ordering.

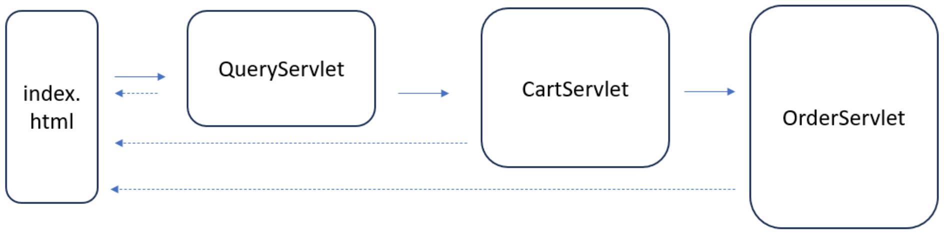
Book ID	AUTHOR	TITLE	PRICE	QTY
3	Tan Ah Teck	Java for Dummy	22.0	1
5	Joe Suh	Good Java Style	12.0	9
Total Price: $130.0				

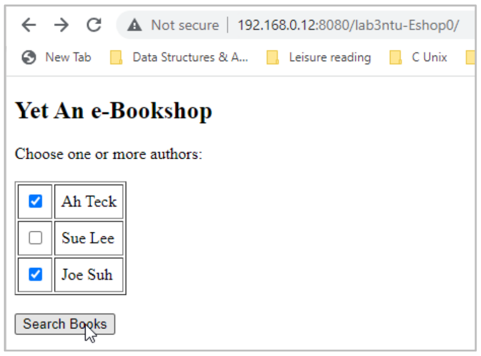
- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

- This version lacks session tracking and does not support the 'shopping cart' experience: after one order, when user goes back to select books again, the previous order information is lost. In real e-commerce application, there should be a shopping cart page before ordering. We want to keep the selection in the shopping cart and then user go back to select more, the items is added the shopping cart. We need session tracking to do this. In this labs we continue with this example to add shopping cart functionality. (No JSPs for this exercise. In exercise B, you will create JSPs to do some displays).

We add one step between the query and order page: view shopping cart page. From the query result page, user can add items to the current shopping cart and view the cart. From shopping cart page, they can checkout, which will show the final result.
From the shopping cart page, user can also go back to query and select again and newly selected items will be added to the shopping cart. If the item has the same id, the qty of the item will be updated.
The shopping cart page also allows the user to update the qty, and remove the item, before checking out.

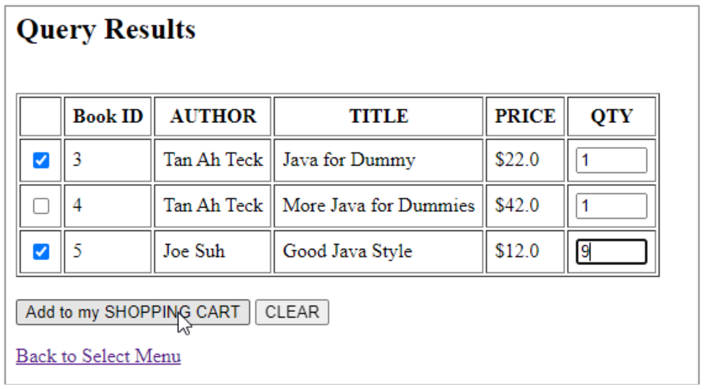




Yet An e-Bookshop

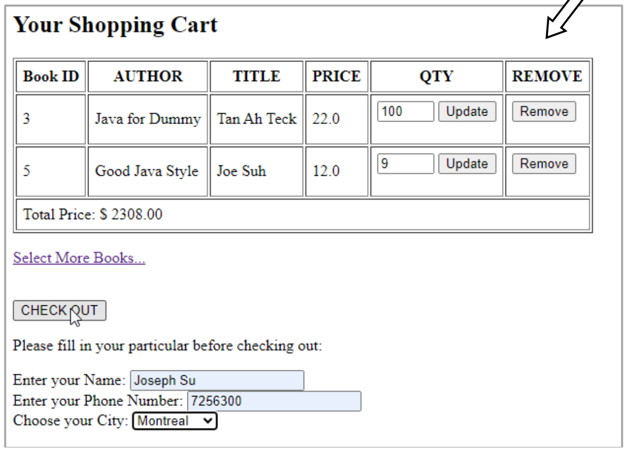
Choose one or more authors:

☑	Ah Teck
☐	Sue Lee
☑	Joe Suh



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

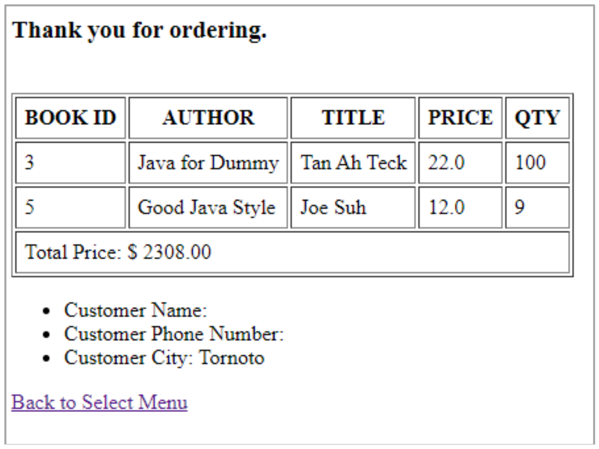


Your Shopping Cart

Book ID	AUTHOR	TITLE	PRICE	QTY	REMOVE
3	Java for Dummy	Tan Ah Teck	22.0	100 Update	Remove
5	Good Java Style	Joe Suh	12.0	9 Update	Remove
Total Price: $ 2308.00					

Please fill in your particular before checking out:

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



Thank you for ordering.

BOOK ID	AUTHOR	TITLE	PRICE	QTY
3	Java for Dummy	Tan Ah Teck	22.0	100
5	Good Java Style	Joe Suh	12.0	9
Total Price: $ 2308.00				

- Customer Name:
- Customer Phone Number:
- Customer City: Tornoto

See the demo video for more page flows.

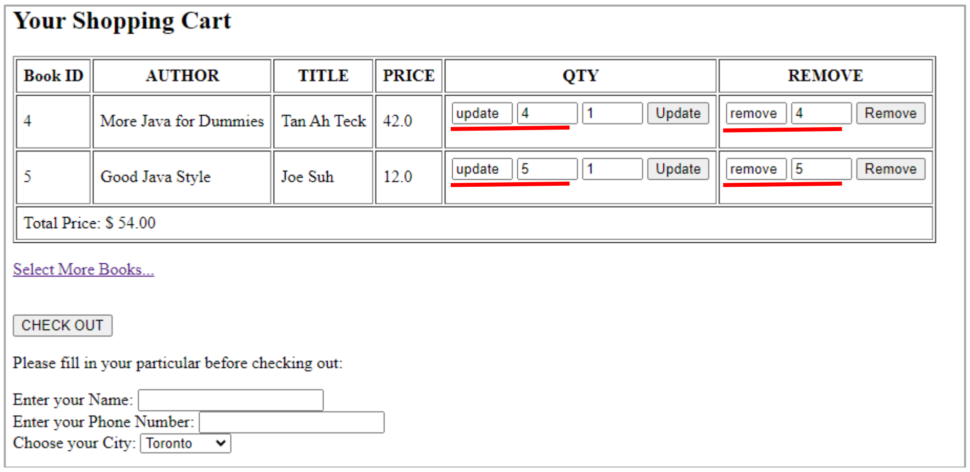
- Import your **lab4servletQ1.war** file that you exported for lab4, as a new project, name it **lab5servletQ2A**
- In the query servlet page, make a few minor changes:
  - Change the form action to the url of **CartServlet**.
  - Change the button text to "Add to my SHOPPING CART"
  - Remove the block to output of textbox for user's name, phone and city information, as it now makes more

sense to ask user to enter the information in the shopping cart page.

   - Somewhere in the form, add a hidden field **out.println("<input type='hidden' name='todo' value='add' />").** When the form is sent to the cart servlet, this adds a parameter 'todo' with value 'add', telling the servlet to do 'adding' --- explained next.

- We need a **Cart** class to model the shopping cart. A Cart contains an array of Books, and provides methods to add, update, and remove a book from the Cart. Partial code is provided. Put this class, as well as Book class, into the model package.
   - Add an attribute **orderedQty** to the **Book** class, and implement getter and setter for this attribute.
   - Complete the Cart class.
- We need a **CartServlet** class. Create this class. This is the more complicated class. It will do session tracking, allowing items to be added. It allows items in the current shopping cart to have the quality updated or removed. Implementing these details is a bit complicated, so most of the code has been provided. Copy the provided code to the class. Your main work here is to read the code, trying to understand the various techniques to implement the functionalities, and complete some parts of the code – no much coding for you. Here are the main features of this class:
   - It first tries to get the session and the Cart attribute from the session. If this is the first time, it will create a new session and a new Cart object and add the new Cart to the new session. In the following code, the Cart will be updated, by adding, updating, or removing items on it.
   - It then handled 3 cases: "add" "update" and "remove". These are the possible values of a form parameter 'todo'. The "add" value comes from the QueryServlet (that is why we added a hidden field in the form in QueryServlet). "update" value comes from the Update button for each qty of this page. 'remove' value comes from the Remove button for each item of this page (explained next).
     For 'add', the code retrieves all current selections from the checkbox in the queryServlet page. It retrieves the books based on the ids, and then adds the books into the Cart. For 'update', as explained below, it gets an id parameter value and a qty parameter value, and then do update on the Cart using the id and the qty information. For 'remove', it gets an id parameter value, and do removal on the Cart based on the id value. This is explained next.
   - Next, it displays the table of items in the Cart. Id, title and price are retrieved from the Cart in a straightforward way. The trickier part is the QTY and REMOVE column, which allow users to update and remove each item in the cart.
      - There is an update button for each item, that allows user to update quantity in the shopping cart. When a new query is entered and the button is clicked, it will stay on the same page, with the updated qty and total price. Behind the scenes, the Cart is also updated.
        To implement the qty update functionality, each cell is a 'small' form, with the qty textbox and an update button. This 'small' form needs two hidden fields, one is 'todo' with value 'update'. Another is 'id' with value of the book id. When the button is clicked, we want to stay on the same page, with the updated qty and total price. Recall that a form with no or empty action will stay on the same page. This form has no action specified, so when the update button is clicked, the doGet() of the same servlet will be invoked, with the parameter data from the 'small' form. Form data sent are the parameter 'todo' of value 'update', parameter 'id' of value id, and the updated qty of name 'qty'id. Since the 'todo' has value 'update', the code of update will be executed, using the id and the updated qty.
      - For each item, there is also a 'Remove' button that allows user to remove the item from the shopping cart. When the button is clicked, it will stay on the same page, with the item removed. Behind the scenes, the item is removed from the Cart.
        To implement the remove functionality, each cell is a 'small' form, with a remove button. This small form also needs two hidden fields, one is 'todo' with value 'remove'. Another is 'id' with value of the book id. This form has not action specified, so when the remove button is clicked, the data is sent to the doGet() of the same servlet, staying on the same page. Form data sent are the parameter 'todo' of value 'remove', parameter 'id' of value id. Since the 'todo' has value 'remove', the code of remove will be executed, removing the item of id.

To help you understand the implementation, in the provided code the hidden fields are left as visible textbox deliberately. When you run the code, you will see these values. Once you understand the code, change the fields to hidden fields.



Your Shopping Cart

Book ID	AUTHOR	TITLE	PRICE	QTY	REMOVE
4	More Java for Dummies	Tan Ah Teck	42.0	update 4 1 Update	remove 4 Remove
5	Good Java Style	Joe Suh	12.0	update 5 1 Update	remove 5 Remove
Total Price: $ 54.00					

Select More Books...

CHECK OUT

Please fill in your particular before checking out:

Enter your Name:
Enter your Phone Number:
Choose your City: Toronto

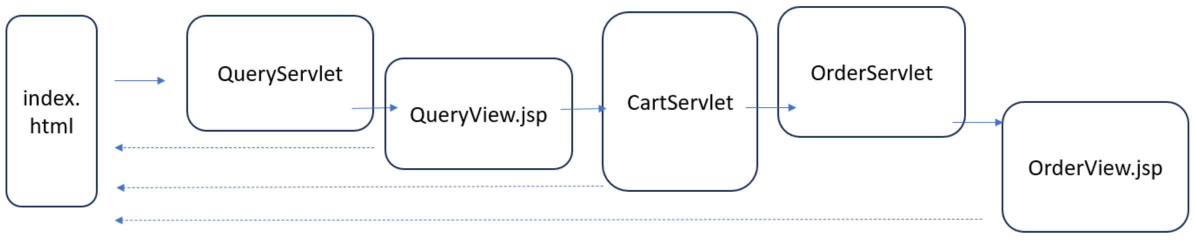
Next, it outputs the user info input, as we did in query page previously.

- The **orderQuery**, modify the doGet() code such that
  - it first gets the session and the shopping cart, and then displays the items in the shopping cart.
  - After that, it also displays the three user info, as did previously.
  - At the very end of the method, add a **cart.clear();** so that after ordering, the shopping cart is emptied.

- Once the program works, export it as 'WAR' file with the default name '**lab5servletQ2A.war**'. Check the 'Export source files' before you click Finish.

Part II B View with JSP.

In the previous implementation, the 3 servlets process data and also generate views. In this exercise we separate the views from **QueryServlet** and **OrderServlet**.
That is, the flow is



(Separating the view from **CartServlet** is bit more complicated so it is not required for this lab.)

- Import the '**lab5servletQ2A.war**' file that you exported in part II A, as new project, name it **lab5servletQ2B**
- In the **QueryServlet,** after getting the list of books, we want to pass the list of book info to a JSP file to do the display. To do this,
  - set the list of books as request attribute, and then
  - use dispatcher to forward to '**bookResultView.jsp**'.
  - remove all the display code.

- Create a JSP file **bookResultView.jsp.** In this file, it retrieves the list of books, and then displays the books.
    - Use EL to retrieve the list of books from the request scope attribute set in the **QueryServlet**.
    - Use JSTL tag <c:forEach> to iterate over the list
      with code like

```
<c:forEach items="${requestScope.bookList}" var="e">
  <tr>
   <td> <input type='checkbox' name='id' value="${e.id}"  /> </td>
   <td> ${e.id} </td>
   ….
   <td><input type='text'  size='5' value='1' name="qty${e.id}" /></td>

   </tr>
</c:forEach>
```

    - Same as in part A, to use JSTL core library, you need to do two more things 1) put the import line **<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>** before the <c:forEach> block (e.g., on top of the file or body part), 2) put two library jar files into the lib folder of your project.

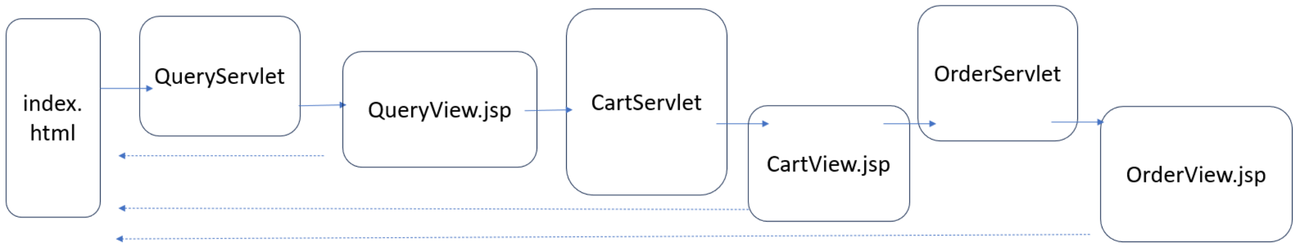
- In the **OrderServlet,** just need to forward to a JSP file, and then clear the cart.
    - keep the code that gets the session and cart
    - use dispatcher to forward to '**orderView.jsp'**.
    - remove all the display code.
    - keep the last line of code that clear the cart at the end.
- Create a JSP file **orderView.jsp.** In this file, it retrieves the list of book from the session, and then displays the ordered books.
    - Use EL to retrieve the session scope attribute of the shopping cart. **${sessionScope.cart.items}** or simply **${cart.items}**
    - Use JSTL tag <c:forEach> to iterate over the list
      with code like

```
<c:forEach items="${sessionScope.cart.items}" var="item">
  <tr>
   <td> ${item.id} </td>
   ….
  </tr>
</c:forEach>
```

    - The little tricky part is to output the totalPrice in JSP. In Servlet, we calculate in the loop. You can calculate in the loop in JSP (similar to in Servlet) but an easier way is that, in the CartQuery servlet, after getting the total price, set it as a session scope attribute, and then in the JSP just retrieve that attribute using code like ${totalPrice} or ${sessionScope.totalPrice}
        - In CartQuery, setting totalPrice as a request scope attribute is not working, so you should set a session or even context/application scope attribute. Think about why.
        - Alternatively, you can add a totalPrice attribute in the Cart class, and implement getter and setter for the attribute. Then in CertQuery, set the totalPrice to the Cart. Then in JSP get it from the Cart bean via ${cart.totalPrice}.
        - You can also explore other ways to display the price in JSP.

- Once the program works, export it as 'WAR' file with the default name '**lab5servletQ2B.war'**. Check the 'Export source files' before you click Finish.

- In this exercise you are not required to separate the view from CartQuery. It still does the output in Java code. It will be a bit more complicated but it would be a good practice to try to separate the view also.



Submissions.

You should have exported **lab5servletQ1A.war lab5servletQ1B.war lab5servletQ2A.war** and **lab5servletQ2B.war**. Make sure you have checked “Export source files” when exporting.

Submit the 4 war files separately on eClass.

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