Documentation

Andrew id: yiq

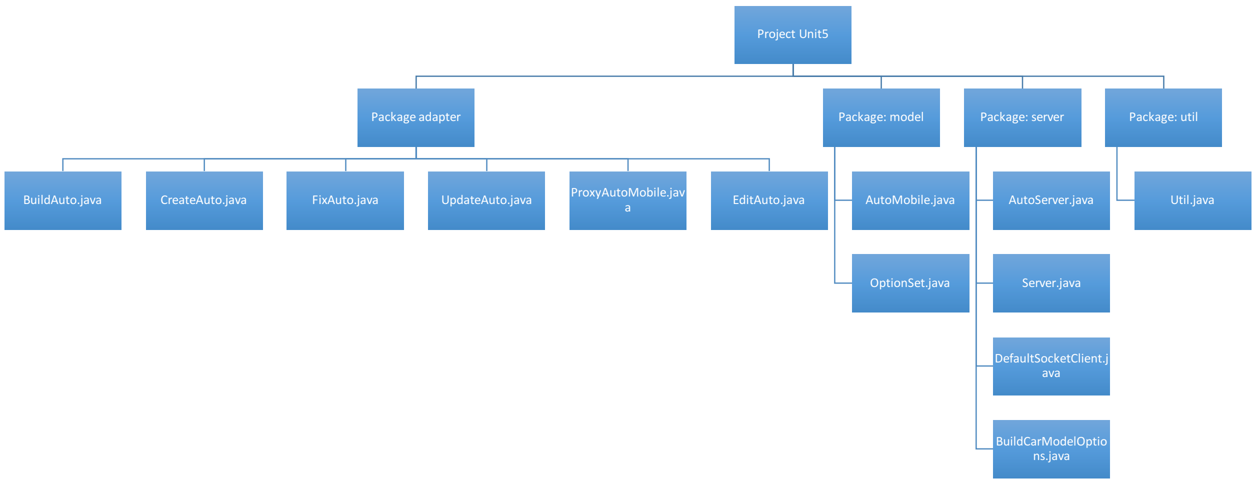
Attention:

This documentation is for the project 1 unit 5 of the course 18641 in CMU. Design details and test details are included.

1: Server Design Details

In this part, no extra package is added in the server part but I add two cases in the DefaultSocketClient.java to handle website client operations, which are select model and select model option.

The following diagram is the package and class relationship in client part. For your convenience, I only draw out the associated packages in the part. Other packages such as exception won't be included.



In summary, Server.java is just a wrapper and all specific implementations are done in DefaultSocketClient.java. AutoServer.java is an interface and it supports three methods that can make server can handle different input file types as required in handout. BuildModelOption.java implements AutoServer.java interface and call three methods that are specifically defined in ProxyAutoMobile as required in handout.

Just like what is demonstrated above, ProxyAutoMobile.java should implement AutoServer.java interface and give specific definition of these three methods and be called in BuildAutoOption.java.

Also in package util, I add a method called readAutoFromProperty() in Util.java to upload car model information with input argument is properties file type. (So far, these are the same as previous unit.)

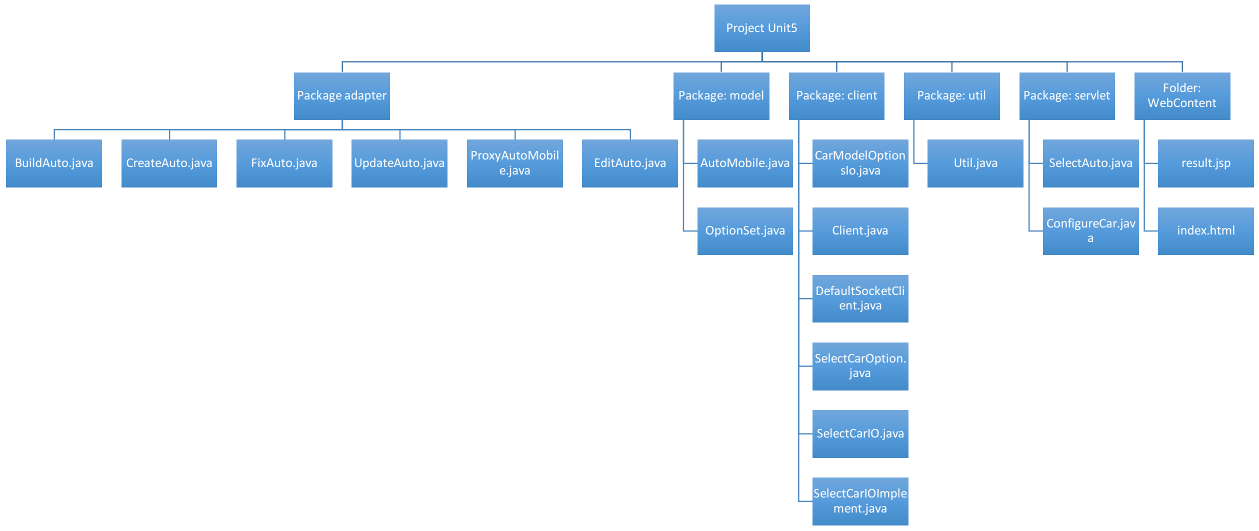
Specifically, two newly added cases in DefaultSocketClient.java file, select car model and select model options, are to handle web requests.

2: Servlet Design Details

In this part, other than a normal java project, I built a dynamic web project called project1_unit5_servlet. I added the client project in the previous unit into this web project but I also created a new package called servlet, to which I added two java files, SelectAuto.java and ConfigureCar.java. SelectAuto.java is to handle web select model request and ConfigureCar.java is to handle configure car request. Besides these, I added two files, result.jsp and index.html to the WebContent folder. result.jsp file is to display the result of configured car and index.html is the welcome page and will turn to SelectAuto.java.

But how can SelectAuto.java and ConfigureCar.java interact with server part since I need get available models and get customer wanted car model from user input in server part. To handle this, as recommended in project handout, I set an interface called SelectCarIO.java and it contains two methods getModelList() and getAutomobile(modelName). GetModelList method returns the available models in the model library and getAutomobile(modelName) returns an instance of automobile according to the ModelName. SelectCarIOImplement.java implements this interface and give specific implementation of these two methods. In this way, servlet can interact with server part.

The following diagram is the package and class relationship in servlet part. For your convenience, I only draw out the associated packages in the part. Other packages such as exception won't be included.



In summary, I built this servlet project based on the unit4 client project. I keep all the client part but added two java files, SelectAuto.java and ConfigureCar.java and two web files, result.jsp and index.html. Remember I added two cases in DefaultSocketClient.java of server part, and here SelectAuto.java relates to the third case and ConfigureCar.java relates to the fourth case in the server part. Also in order to interact with server part, I set up an interface and its implementation to handle this.

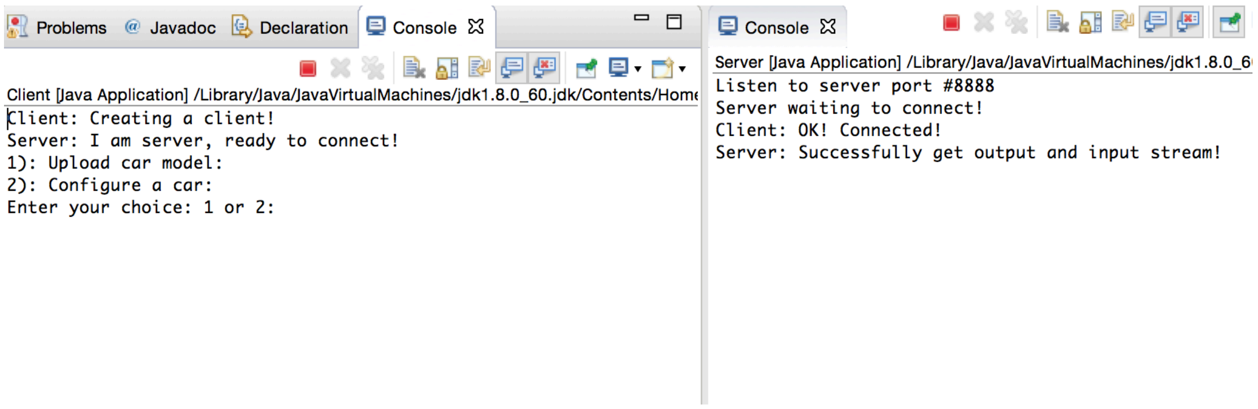
All server part and servlet part design details are strictly following project handout.

3: Test details:

In this part, I test client and server communication by uploading four car models information and customize two cars for user.

Please read following instruction carefully and for each step I will give my test snapshot:

1: First build connection between server and client.



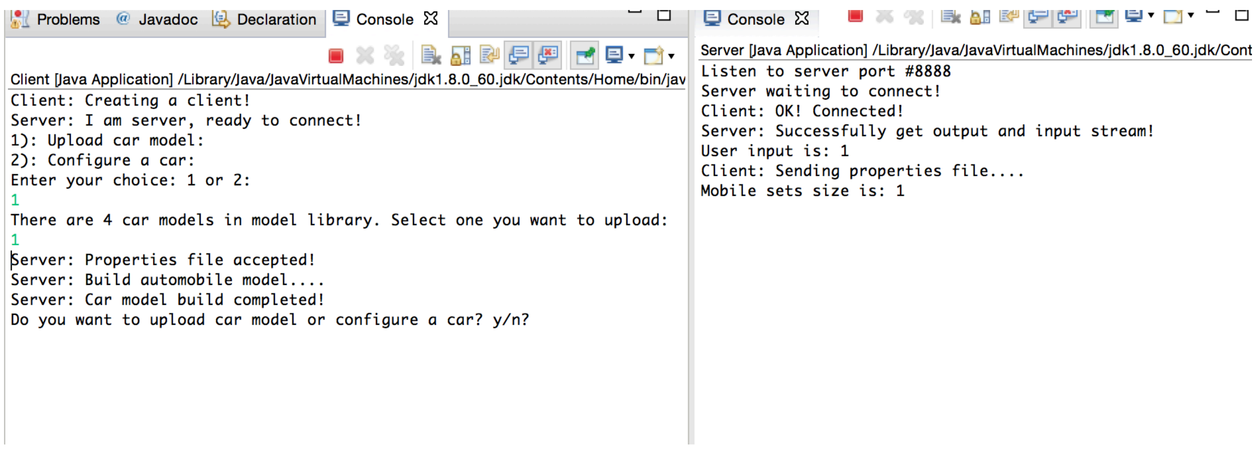
```
Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
```

```
Server [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_6
Listen to server port #8888
Server waiting to connect!
Client: OK! Connected!
Server: Successfully get output and input stream!
```

In this step, I firstly run Server.java and Client.java (the order can not be reversed.) Then from the two consoles information (left one is the client console and right one is the server console), I know that server and client connect successfully.

Then upon the client console information, next step is to ask user to input a choice, here since no car models have been uploaded so I first upload car models. So, go to step 2.

2: Upload car models



```
Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/jav
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
1
There are 4 car models in model library. Select one you want to upload:
1
Server: Properties file accepted!
Server: Build automobile model....
Server: Car model build completed!
Do you want to upload car model or configure a car? y/n?
```

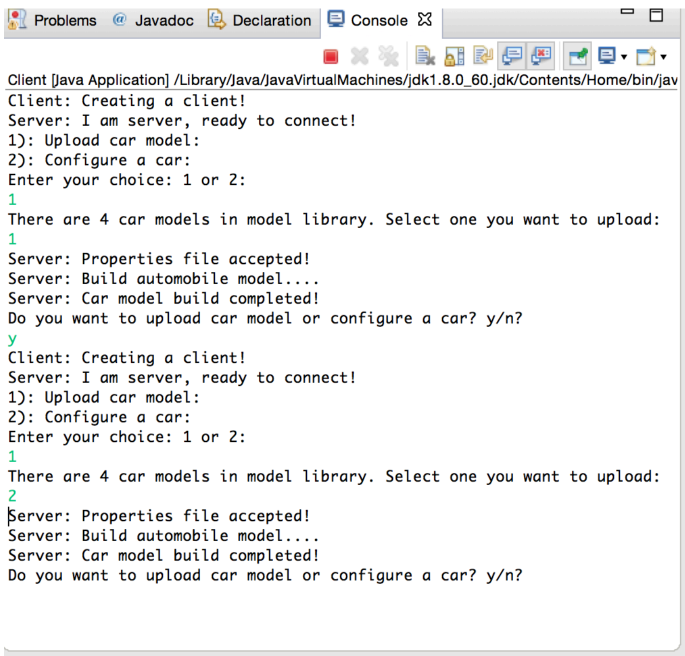
```
Server [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Cont
Listen to server port #8888
Server waiting to connect!
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 1
```

In this step, I input 1 to upload a car model, from the information, I know that there are 4 car models in the model library. So I first choose the first one to upload.

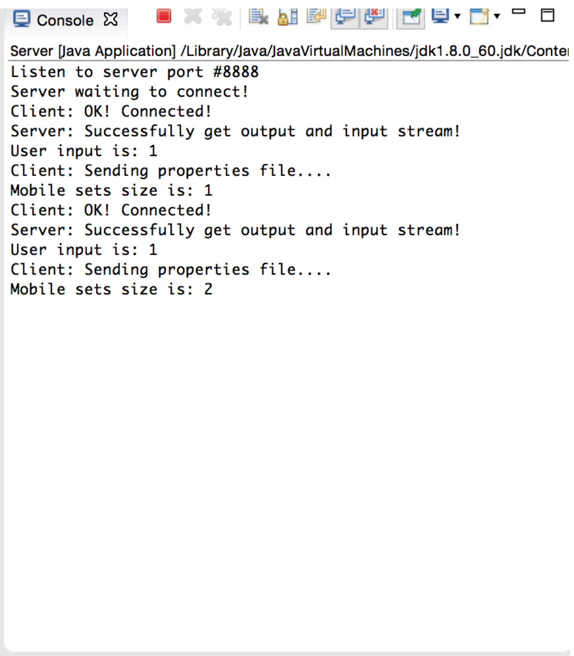
Then we can see from the client console, server send information to tell the client car model has been successfully uploaded.

From the server console, we know that the mobile sets size is increasing indicates that the car model has been uploaded successfully. Then I continue input y and digit to upload all these 4 car models into the mobile sets.

In this part, another part should be explained.

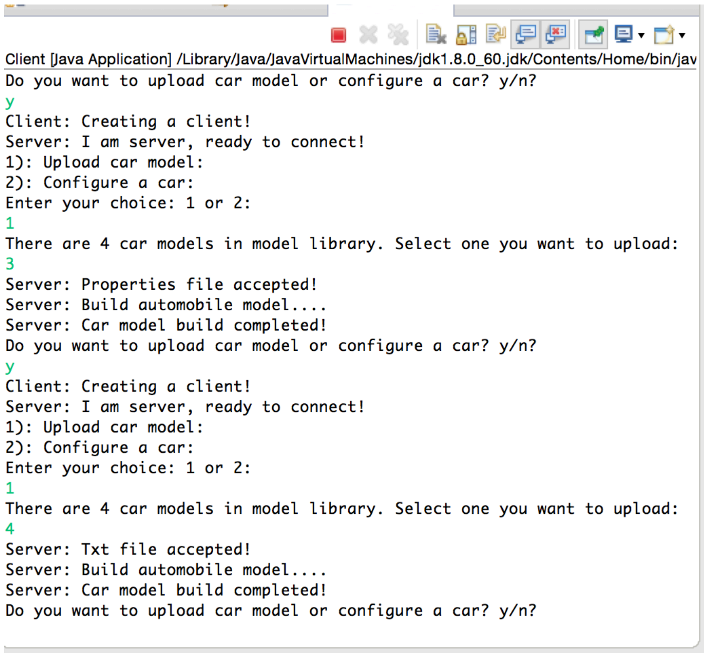


```
Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/jav
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
1
There are 4 car models in model library. Select one you want to upload:
1
Server: Properties file accepted!
Server: Build automobile model....
Server: Car model build completed!
Do you want to upload car model or configure a car? y/n?
y
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
1
There are 4 car models in model library. Select one you want to upload:
2
Server: Properties file accepted!
Server: Build automobile model....
Server: Car model build completed!
Do you want to upload car model or configure a car? y/n?
```

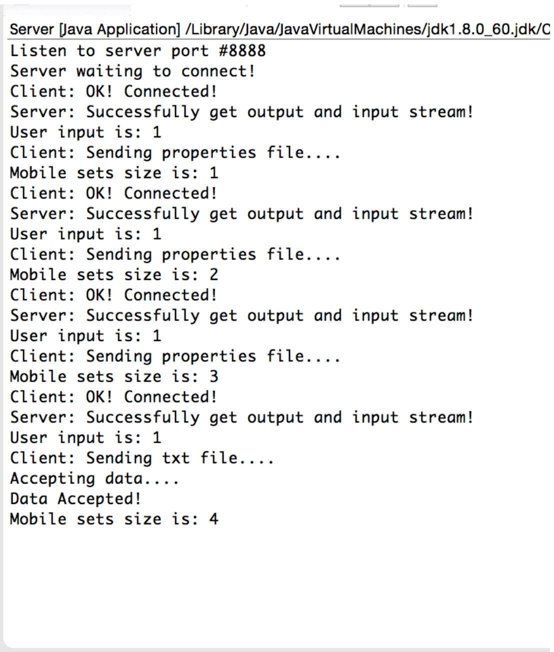


```
Server [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Conte
Listen to server port #8888
Server waiting to connect!
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 1
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 2
```

Now I upload two car models, then continue to upload next car models. Finally, we can get:



```
Client [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/jav
Do you want to upload car model or configure a car? y/n?
y
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
1
There are 4 car models in model library. Select one you want to upload:
3
Server: Properties file accepted!
Server: Build automobile model....
Server: Car model build completed!
Do you want to upload car model or configure a car? y/n?
y
Client: Creating a client!
Server: I am server, ready to connect!
1): Upload car model:
2): Configure a car:
Enter your choice: 1 or 2:
1
There are 4 car models in model library. Select one you want to upload:
4
Server: Txt file accepted!
Server: Build automobile model....
Server: Car model build completed!
Do you want to upload car model or configure a car? y/n?
```



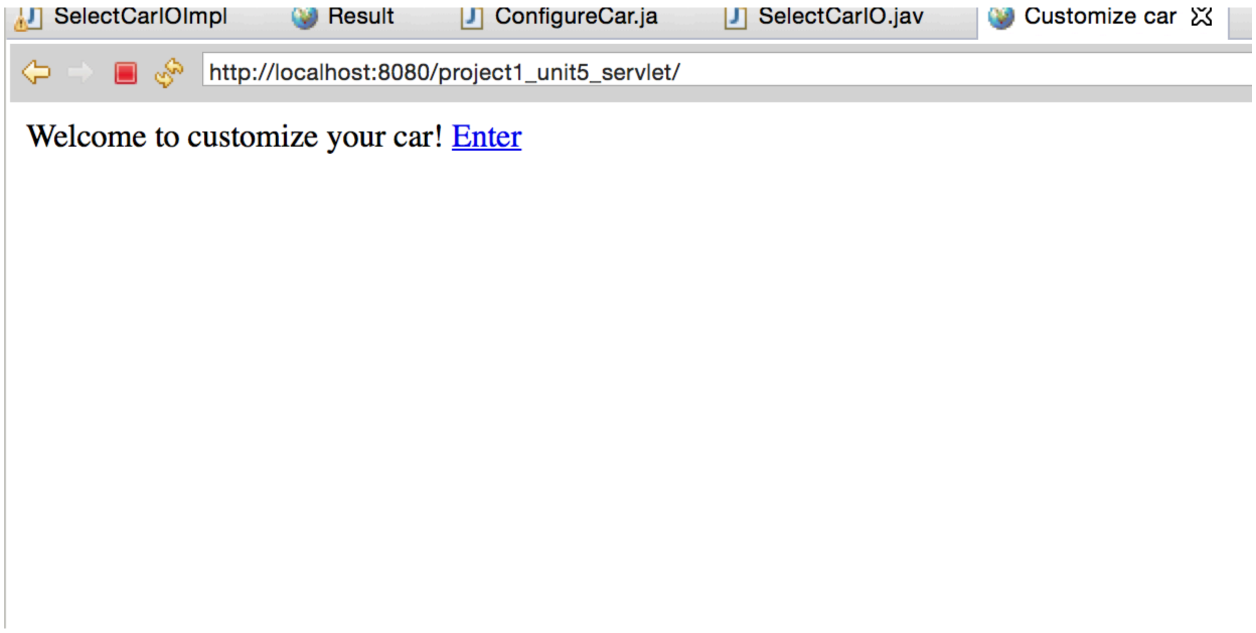
```
Server [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/C
Listen to server port #8888
Server waiting to connect!
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 1
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 2
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending properties file....
Mobile sets size is: 3
Client: OK! Connected!
Server: Successfully get output and input stream!
User input is: 1
Client: Sending txt file....
Accepting data....
Data Accepted!
Mobile sets size is: 4
```

As you can see form the 4 input, this is a txt file not a properties file, so some different information is printed. After upload all car models, now I try to customize user's car from servlet. See below:

3: Customize car from website

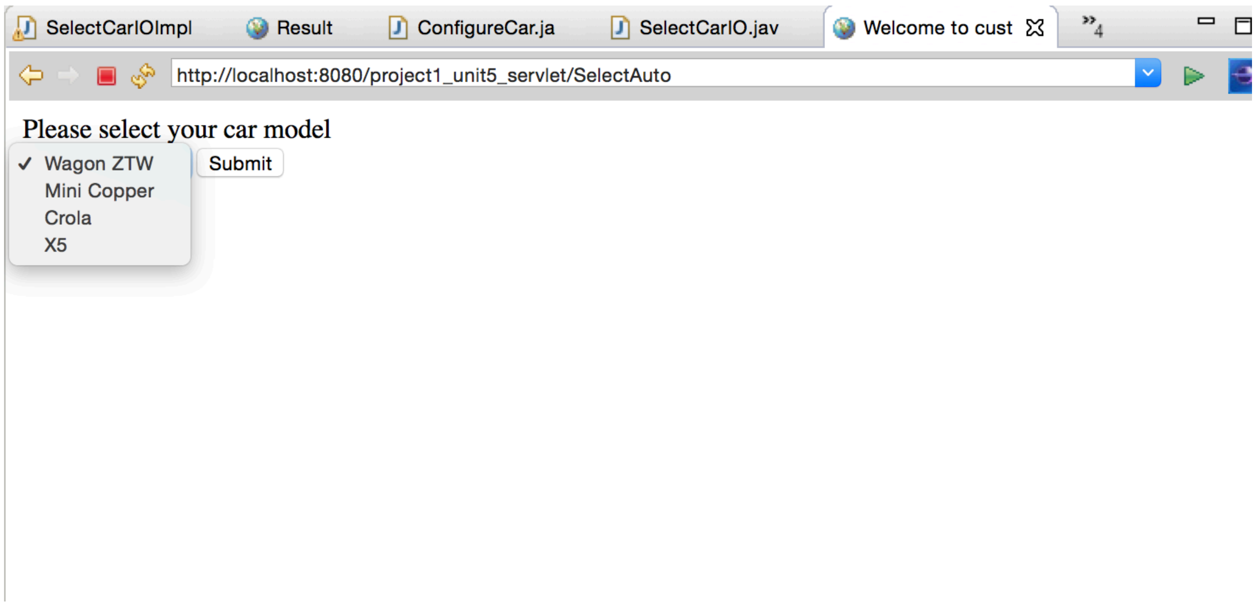
So far, the upload operations are the same as in the unit4. Now test configure car from website. Remember if you want to test this part, Tomcat server should be installed in your computer and add the runtime to your workspace.

3.1: Find the index.html file in WebContent and run this file on the Server. Find the Tomcat server and finish this operation. Then you will see this welcome homepage:



Welcome to customize your car! Enter

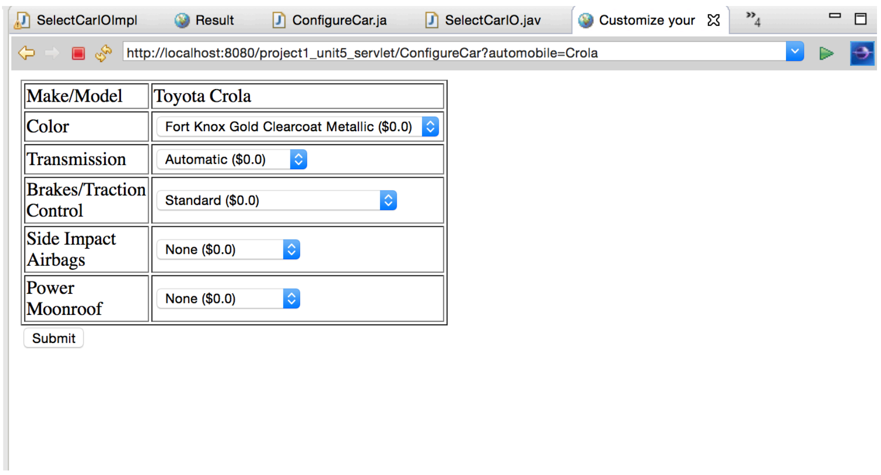
Press the Enter, the page will turn to select car model page:



Please select your car model

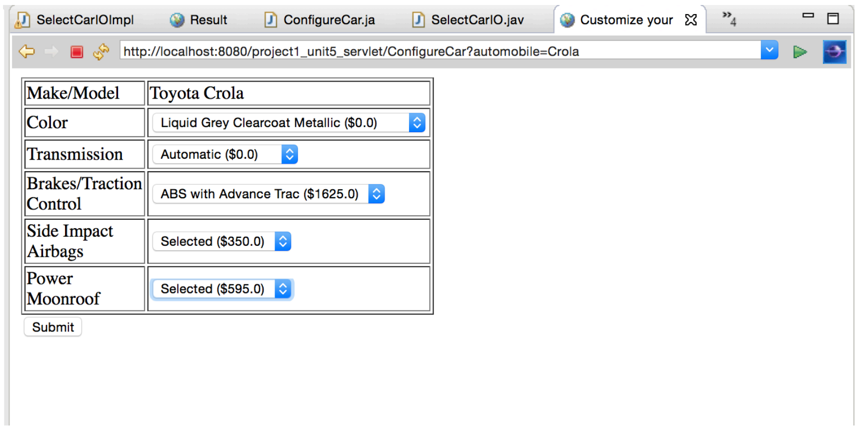
✓ Wagon ZTW
Mini Copper
Crola
X5

As you can see, there are in total four car models which I have uploaded in the previous test. You can randomly select a car model as you like, here I select Crola and commit submit. Then you will see:


SelectCarIOImpl
Result
ConfigureCar.ja
SelectCarIO.jav
Customize your
http://localhost:8080/project1_unit5_servlet/ConfigureCar?automobile=Crola
Submit

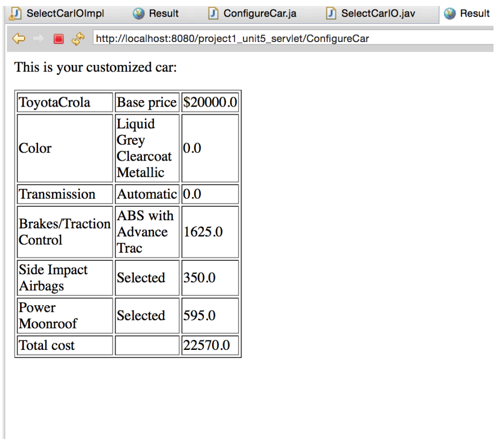
Make/Model	Toyota Crola
Color	Fort Knox Gold Clearcoat Metallic ($0.0)
Transmission	Automatic ($0.0)
Brakes/Traction Control	Standard ($0.0)
Side Impact Airbags	None ($0.0)
Power Moonroof	None ($0.0)

From the figure, you can see the Make/Model information and some optionsets Color, Transmission and so on. Within each optionset, there are several options you can select. Here I randomly select:


SelectCarIOImpl
Result
ConfigureCar.ja
SelectCarIO.jav
Customize your
http://localhost:8080/project1_unit5_servlet/ConfigureCar?automobile=Crola
Submit

Make/Model	Toyota Crola
Color	Liquid Grey Clearcoat Metallic ($0.0)
Transmission	Automatic ($0.0)
Brakes/Traction Control	ABS with Advance Trac ($1625.0)
Side Impact Airbags	Selected ($350.0)
Power Moonroof	Selected ($595.0)

You will see your options and now I commit submit the you will see you customized car information:



This is your customized car:

ToyotaCrola	Base price	$20000.0
Color	Liquid Grey Clearcoat Metallic	0.0
Transmission	Automatic	0.0
Brakes/Traction Control	ABS with Advance Trac	1625.0
Side Impact Airbags	Selected	350.0
Power Moonroof	Selected	595.0
Total cost		22570.0

Now the test in completed successfully.