Automobile Configuration System

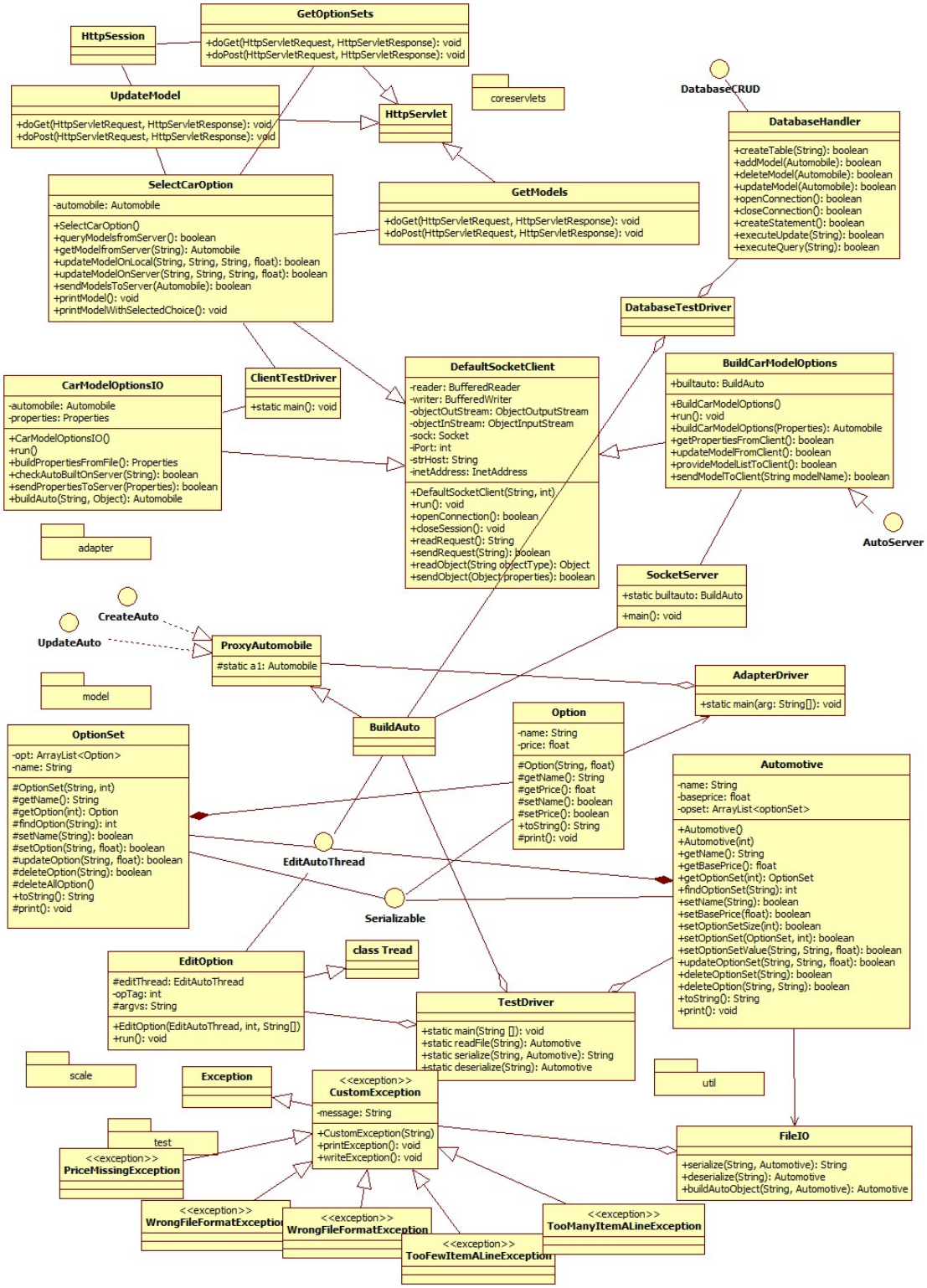
Project 1 Unit 5-6

This document contains description of a simple web server for 18641 project1 Unit5-6.

Xuping Lei (xlei@andrew.cmu.edu)
February 2014

Project 1 Unit 5/6 Automobile Configuration System

1. Class diagram and description



Name	Type	Package	Description
Newly added (Client and Web APP) Branch Project: JSPHDClient_Tomcat			
GetModels	servlet	coreservlets	Get models from the server and display. Path: http://localhost:8080/client/GetModels
GetOptionSets	servlet	coreservlets	Display the model and selective menus for users.
UpdateModel	servlet	coreservlets	Called by GetOptionSets to update the model instance both on client side and server side.
displayModel.jsp	JSP	Web Root Directory	Redirected to this file when UpdateModel finish updation. The JSP file will display updated results.
index.jsp	JSP	Web Root Directory	Used to test the validation of below path. http://localhost:8080/client/
Newly added (Server and Database APP) Branch Project: JSPHDProject2Server			
DatabaseCRUD	interface	database	Interface of main database operations to add, update, delete models.
DatabaseHandler	class	database	The class implements database operations.
DatabaseTestDriver	class	database	Test driver to check the validation of class DatabaseHandler.
Updated (Feb 21):			

CarModelOptionsIO	class	client	In Client project. Create models on server via properties files in clients.
SelectCarOption	class	client	In Client project. Query and update models on server via properties files in clients.
ClientTestDriver	class	client	In Client project. Test the performance of client.
AutoServer	interface	server	In Server project.
BuildCarModelOptions	class	server	In Server project. Work as a thread on server responsing to client request.
SocketServer	class	server	In Server project. Server main program. Work consistently.
Updated (Feb 21):			
ProxyAutomobile	abstract Class	adapter	Added editOptionPrice() and editOptionSetName(). It also contains all the implementation of any method declared in the Interfaces CreateAuto and UpdataAuto.
Automobile (Original name: Automotive)	class	model	Represent Automotive model; Include model name, base price and OptionSet[](properties).
FileIO	class	util	Changed methods writeObject and readObject to serialize and deserialize. Move SourceReader.buildAutoObject() to FileIO.buildAutoObject().
TestDriver	class	test	Used to test entire project, meeting requirements of part b. Includes main(), the entry of the whole project.

Other classes			
EditOption	class	scale	Edit Options for a given model set in its own thread. Internally access the model set.
EditAutoThread	interface	adapter	Enable class EditOption to interact with LinkedHashMap<String, Automobile> modelSets in ProxyAutomobile.
WrongFileFormat Exception	class	test	Added exception. Indicate possible wrong format of a input file.
TooFewItemALine Exception	class	test	Added exception. Indicate too few parameters in a line of a input file.
TooManyItemALi neException	class	test	Added exception. Indicate too many parameters in a line of a input file.
BuildAuto	class	adapter	Inherited from abstract class ProxyAutomobile.
CreateAuto	interface	adapter	Interface with methods: createAut() and printAuto().
UpdataAuto	interface	adapter	Interface with methods: updateAuto() and updateOptionSetNam ().
AdapterDriver	class	adapter	Used to test the project, maily the adapter package, meeting requirements of part a.
PriceMissingExce ption	class	test	Extends Custom exception handlers. Used when missing price for Automobile in text file.

WrongFilenameException	class	test	Extends Custom exception handlers. Used when missing filename or wrong filename.
OptionSet	class	model	Represent properties in Automotive model; Include OptionSet name and Option[]; Contains inner class Option.
~~SourceReader~~ (Already merged into FileIO)	~~class~~	~~util~~	~~Contains method buildAutoObject; Read and build an Automotive object from file.~~
CustomException	class	test	Custom exception handlers.

2. Design of Database Tables

There are 5 tables in total.

TABLE automobile:

auto_id	name	model	make	baseprice
int	VARCHAR(255)	VARCHAR(255)	VARCHAR(255)	int

TABLE optionset:

set_id	name
int	VARCHAR(255)

TABLE optionunit:

opt_id	name	price
int	VARCHAR(255)	int

TABLE auto_optset:

bind_id	auto_id	set_id
int	int	int

TABLE auto_optset:

bind_id	set_id	opt_id
int	int	int

Files about creating tables could be found on "proj-5-6-submission/proj6/JSPHDProject2Server/sqlStatements/ createTable.txt". Below is its contents:

```
NEW
CREATE TABLE automobile
(auto_id int NOT NULL AUTO_INCREMENT,
name VARCHAR(255),
model VARCHAR(255),
make VARCHAR(255),
baseprice int,
PRIMARY KEY (auto_id));
END

NEW
CREATE TABLE optionset
(set_id int NOT NULL AUTO_INCREMENT,
name VARCHAR(255),
PRIMARY KEY (set_id));
END

NEW
CREATE TABLE optionunit
(opt_id int NOT NULL AUTO_INCREMENT,
name VARCHAR(255),
price int,
PRIMARY KEY (opt_id));
END

NEW
CREATE TABLE auto_optset
(bind_id int NOT NULL AUTO_INCREMENT,
auto_id int,
set_id int,
PRIMARY KEY (bind_id),
FOREIGN KEY (auto_id) REFERENCES automobile (auto_id),
FOREIGN KEY (set_id) REFERENCES optionset (set_id));
END

NEW
CREATE TABLE set_opt
(bind_id int NOT NULL AUTO_INCREMENT,
set_id int,
opt_id int,
PRIMARY KEY (bind_id),
FOREIGN KEY (set_id) REFERENCES optionset (set_id),
FOREIGN KEY (opt_id) REFERENCES optionunit (opt_id));
END
```

3. What I learn

Project 1-5:

- Concepts of JSP programming
- Concepts of Servlets programming
- How to get HTTP request
- How to write HTTP response
- Use sessions to communicate among differenct servlets
- Configure Tomcat

Project 1-6:

- Concepts of Database: Primary key, Foreign key, etc
- Design table based on rules of normalization
- Concepts of JDBC
- Write basic SQL query
- DB Programming using JDBC

4. Results of unit 5 project

The porject files locate on "proj-5-6-submission/ proj5":

1. server_v8.jar:

   the server program used to test the project. It can be started using command "java -jar server_v8.jar 8050". After started, it will read models from file "testdata3.txt".

```
proj5
fishtekiMacBook-Pro-2:proj-5-6-submission fish$ cd proj5
fishtekiMacBook-Pro-2:proj5 fish$ ls
server_v8.jar   testdata3.txt
fishtekiMacBook-Pro-2:proj5 fish$ java -jar server_v8.jar 8050
Build Automobile from file testdata3.txt
initiate preset models:
-- 1 --
Automotive infomation
Model: ZTW
Make: Ford
Name: Ford Focus Wagon ZTW
Base Price: 18445.0
```

2. testdata3.txt:

   Store initial model data for the server.

3. apache-tomcat-7.0.52:

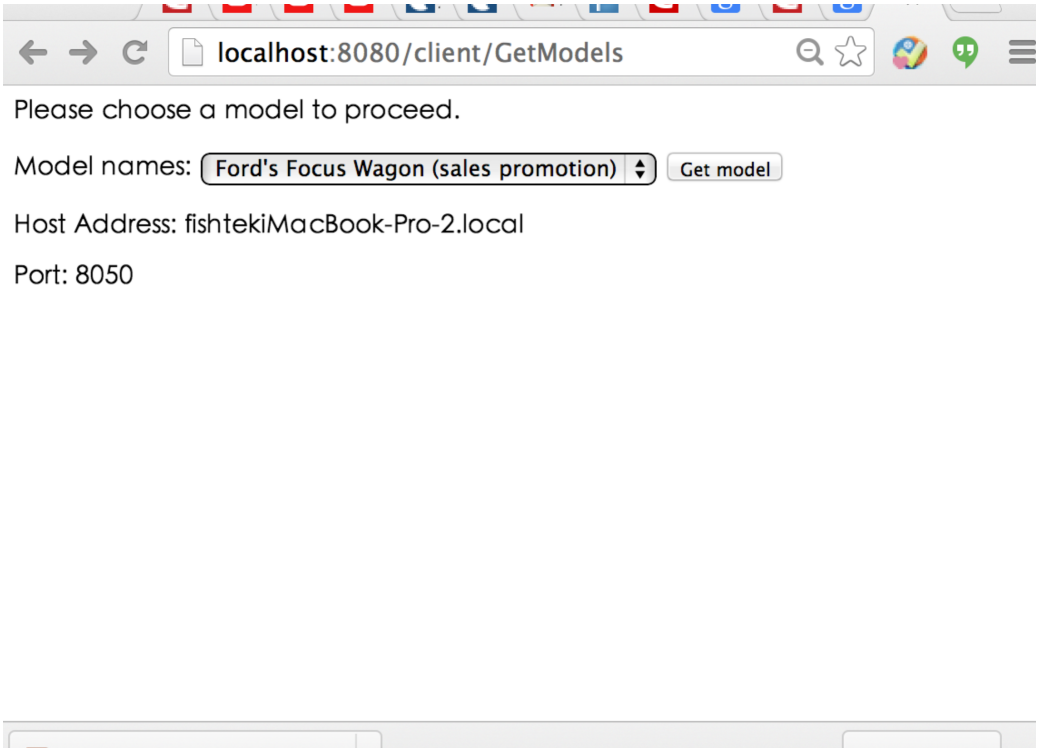
   The configured server that used to implement tests.

4. **JSPHDClient_Tomcat**:

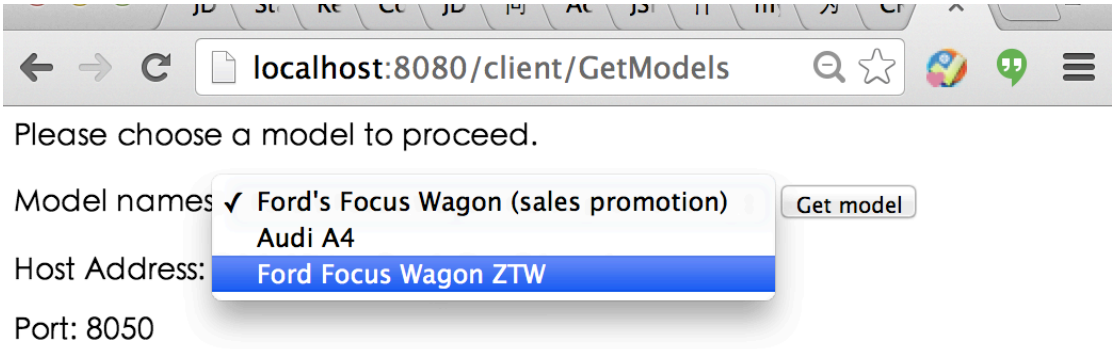
The project containing main servlets and JSP for unit 5 assignment. It could be added to servers on other hosts to realize required funcitons.

Test case 1: get models

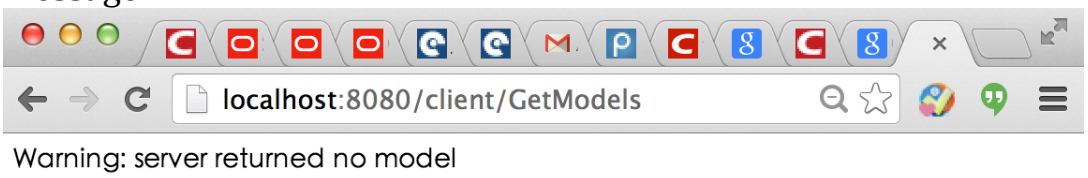
Typing path http://localhost:8080/client/GetModels, there will be a drop-down menu to choose models.



Visitors can choose one item and click the button “Get models”.

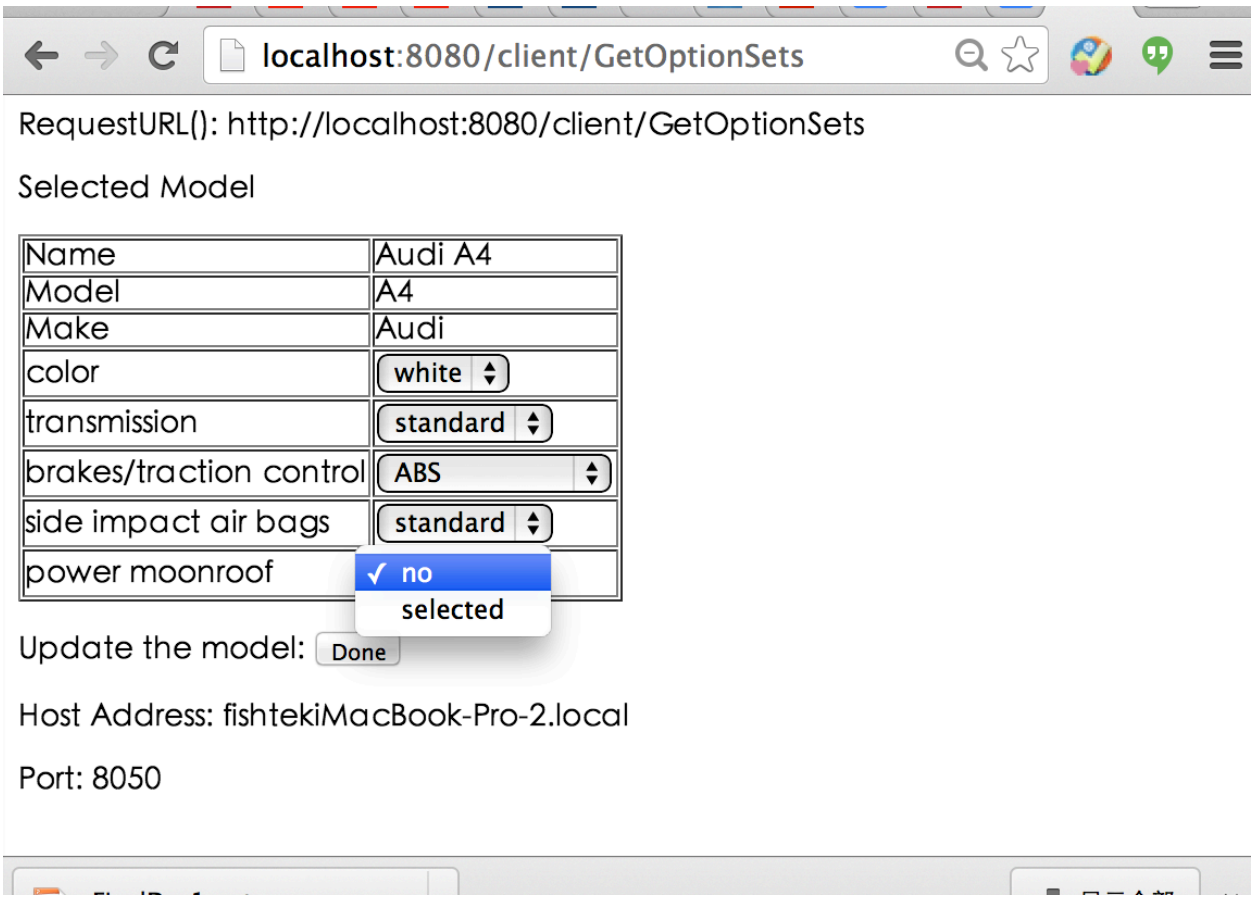


If the server does not provide models, the page will display the warning message.



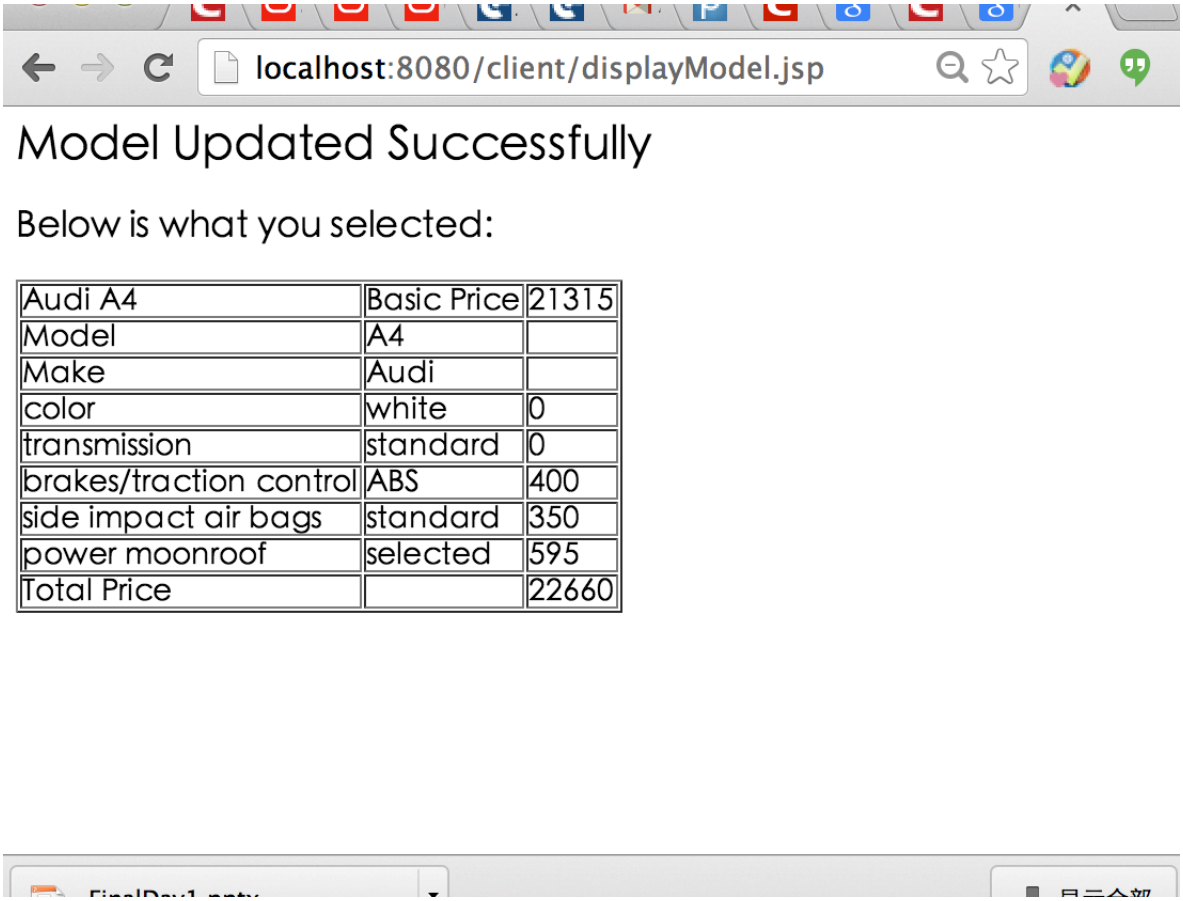
Test case 2: display models

If get models successfully, the browser will jump to http://localhost:8080/client/GetOptionSets. Details about models will be display.



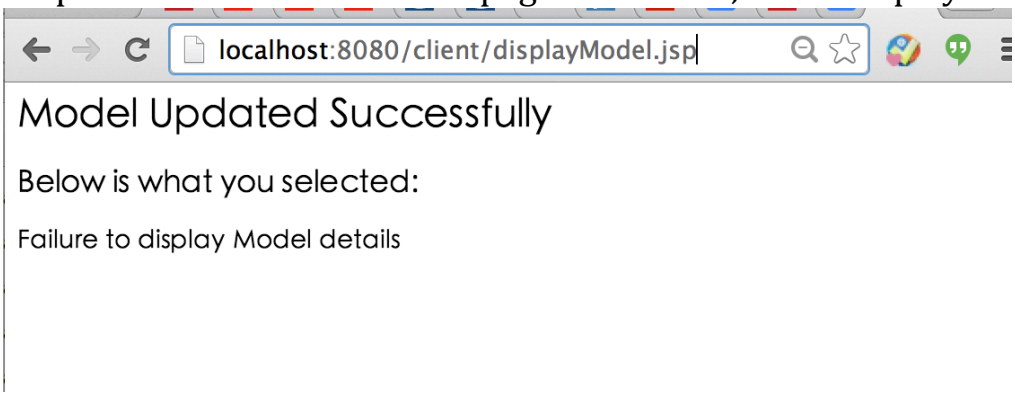
Test case 3: update models

If visitors change some choices and click button “done”, the browser will jump to path http://localhost:8080/client/displayModel.jsp. The total price will be calculated.



Audi A4	Basic Price	21315
Model	A4	
Make	Audi	
color	white	0
transmission	standard	0
brakes/traction control	ABS	400
side impact air bags	standard	350
power moonroof	selected	595
Total Price		22660

If updation is finished but the page if time-out, it will display.



5. Results of unit 6 project

The porject files locate on "proj-5-6-submission/ proj6". New added classes are in the package database. Test is executed using DatabaseTestDriver.java.

Below are test results:

```
Register JDBC driver ...

Test 1: Open connection
Build Automobile from file testdata2.txt
Automotive infomation
Model: ZTW
Make: Ford
Name: Ford Wagon ZTW 2
Base Price: 18445.0

OptionSet: color
Option name: red          price: 0.0

OptionSet: transmission
Option name: standard   price: -815.0

OptionSet: brakes/traction control
Option name: ABS        price: 400.0
Option name: Advance Trac       price: 1225.0

OptionSet: side impact air bags
Option name: selected    price: 350.0

OptionSet: power moonroof
Option name: selected    price: 595.0


Test 1: Open connection
Open connection successfully: com.mysql.jdbc.JDBC4Connection@4e4d6444

---- Test 2: create tables ----

Executing SQL: CREATE TABLE automobile (auto_id int NOT NULL AUTO_INCREMENT, name
VARCHAR(255), model VARCHAR(255), make VARCHAR(255), baseprice int, PRIMARY KEY
(auto_id));
Executing SQL: CREATE TABLE optionset (set_id int NOT NULL AUTO_INCREMENT, name
VARCHAR(255),  PRIMARY KEY (set_id));
Executing SQL: CREATE TABLE optionunit (opt_id int NOT NULL AUTO_INCREMENT, name
VARCHAR(255), price int, PRIMARY KEY (opt_id));
Executing SQL: CREATE TABLE auto_optset (bind_id int NOT NULL AUTO_INCREMENT, auto_id
int, set_id int, PRIMARY KEY (bind_id), FOREIGN KEY (auto_id) REFERENCES automobile
(auto_id), FOREIGN KEY (set_id) REFERENCES optionset (set_id));
Executing SQL: CREATE TABLE set_opt (bind_id int NOT NULL AUTO_INCREMENT, set_id int,
opt_id int, PRIMARY KEY (bind_id), FOREIGN KEY (set_id) REFERENCES optionset (set_id),
FOREIGN KEY (opt_id) REFERENCES optionunit (opt_id));
create tables successfully
```

---- Test 3: add models ----

Executing SQL: INSERT INTO automobile (name, model, make, baseprice) VALUES ('Ford Wagon ZTW 2', 'ZTW', 'Ford', 18445);
modelAtoIncKey: 1
Executing SQL: INSERT INTO optionset (name) VALUES (' color') ;
Executing SQL: INSERT INTO auto_optset (auto_id, set_id) VALUES (1, 1) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('red', 0) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (1, 1) ;
Executing SQL: INSERT INTO optionset (name) VALUES (' transmission') ;
Executing SQL: INSERT INTO auto_optset (auto_id, set_id) VALUES (1, 2) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('standard', -815) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (2, 2) ;
Executing SQL: INSERT INTO optionset (name) VALUES (' brakes/traction control') ;
Executing SQL: INSERT INTO auto_optset (auto_id, set_id) VALUES (1, 3) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('ABS', 400) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (3, 3) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('Advance Trac', 1225) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (3, 4) ;
Executing SQL: INSERT INTO optionset (name) VALUES (' side impact air bags') ;
Executing SQL: INSERT INTO auto_optset (auto_id, set_id) VALUES (1, 4) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('selected', 350) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (4, 5) ;
Executing SQL: INSERT INTO optionset (name) VALUES (' power moonroof') ;
Executing SQL: INSERT INTO auto_optset (auto_id, set_id) VALUES (1, 5) ;
Executing SQL: INSERT INTO optionunit (name, price) VALUES ('selected', 595) ;
Executing SQL: INSERT INTO set_opt (set_id, opt_id) VALUES (5, 6) ;
add a model successfully

---- Test 4: delete models ----

Execute Query SQL: SELECT auto_id FROM automobile WHERE name = 'Ford Wagon ZTW 2' ;
modelAtoIncKey: 1
Execute Query SQL: SELECT set_id FROM auto_optset WHERE auto_id = 1 ;
Execute Query SQL: SELECT opt_id FROM set_opt WHERE set_id = 1 ;
Executing SQL: DELETE FROM set_opt WHERE set_id = 1 AND opt_id = 1;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 1;
Executing SQL: DELETE FROM auto_optset WHERE set_id = 1;
Executing SQL: DELETE FROM optionset WHERE set_id = 1;
Execute Query SQL: SELECT opt_id FROM set_opt WHERE set_id = 2 ;
Executing SQL: DELETE FROM set_opt WHERE set_id = 2 AND opt_id = 2;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 2;
Executing SQL: DELETE FROM auto_optset WHERE set_id = 2;
Executing SQL: DELETE FROM optionset WHERE set_id = 2;
Execute Query SQL: SELECT opt_id FROM set_opt WHERE set_id = 3 ;
Executing SQL: DELETE FROM set_opt WHERE set_id = 3 AND opt_id = 3;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 3;
Executing SQL: DELETE FROM set_opt WHERE set_id = 3 AND opt_id = 4;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 4;
Executing SQL: DELETE FROM auto_optset WHERE set_id = 3;
Executing SQL: DELETE FROM optionset WHERE set_id = 3;

```
Execute Query SQL: SELECT opt_id FROM set_opt WHERE set_id = 4 ;
Executing SQL: DELETE FROM set_opt WHERE set_id = 4 AND opt_id = 5;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 5;
Executing SQL: DELETE FROM auto_optset WHERE set_id = 4;
Executing SQL: DELETE FROM optionset WHERE set_id = 4;
Execute Query SQL: SELECT opt_id FROM set_opt WHERE set_id = 5 ;
Executing SQL: DELETE FROM set_opt WHERE set_id = 5 AND opt_id = 6;
Executing SQL: DELETE FROM optionunit WHERE opt_id = 6;
Executing SQL: DELETE FROM auto_optset WHERE set_id = 5;
Executing SQL: DELETE FROM optionset WHERE set_id = 5;
Executing SQL: DELETE FROM automobile WHERE auto_id = 1;
delete a model successfully
```