**CS 631-002-SPRING 2017 – Online Hotel Reservation System Project**

**Deliverable 3**

**GROUP3**

**LIN TANG**

**YUHAO SHI**

**XIAOXI YANG**

**PROFESSOR:**

**Dimitrios Theodoratos**

**Ananya Dass**

INTRODUCTION

[Goal 3](#_Toc481620977)

[Revisions 3](#_Toc481620978)

[Problems Encountered 3](#_Toc481620979)

[Application Implementation 3](#_Toc481620980)

[Programming Concept 4](#_Toc481620981)

[Database Creation 5](#_Toc481620982)

[Table Population 9](#_Toc481620983)

[Run Program 11](#_Toc481620984)

[Users Guide 37](#_Toc481620985)

[Customer Guide 37](#_Toc481620986)

[Administrator Guide 39](#_Toc481620987)

[SQL Commands 40](#_Toc481620988)

[SQL Commands of Table Creation 40](#_Toc481620989)

[SQL Commands of Table Population (Partial) 48](#_Toc481620990)

[Source Code 52](#_Toc481620991)

[Java (Partial) 53](#_Toc481620992)

[JSP and JS (Partial) 58](#_Toc481620993)

# Goal

The goal of this part is translating our Relational schema to a real world database framework, populate the database, and create a website for Hulton.

# Revisions

1. Adding a new attribute named c\_id in table of credit\_card, it is a foreign key to table customer c\_id. Based on this action, every customer can store a lots of credit cards in our database, they can choose any card to pay their reservation.
2. Creating a new table named manager, to store the managers’ account information. The recommendation attribute is used to give some good news to the customer, eg. discount information.

# Problems Encountered

This is my first time to use eclipse to develop a website, too many things need to be dealt with. This is not a perfect website, but we already do our best.

1. Tomcat port default is 8080, but on our computer it always shows a error message that says this port is already in use, so we change to port of tomcat to 8081.
2. To develop a fancy interface for our website is too difficult.
3. It is hard to let a room reservation include multiple breakfast/service types and can have multiple orders for each breakfast type, and the total number of orders for different breakfast types for a room reservation should not exceed the number of persons that can stay in that room. By referring a lot of information on the internet, we solved this problem with using JQuery.

# Application Implementation

**Source code folder:**

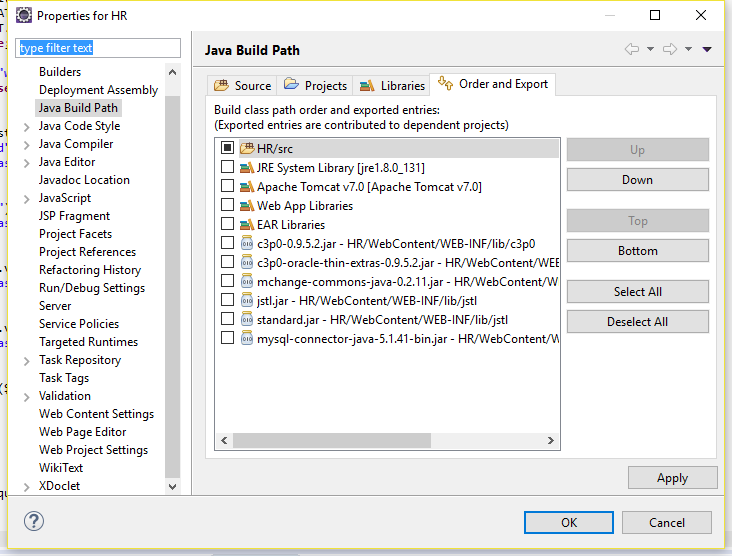
HR (using Java programming language, JSP and JQuery, JS)

**Tools that we use:**

1. JDK 1.8.0
2. Eclipse IDE for Java EE Developers

Version: Neon.1a Release (4.6.1)

1. apache-tomcat-7.0.73
2. MySQL server 5.7
3. Navicat for MYSQL 10.0.11



## Programming Concept

Our entire website is create as an mvc model, which is Model–view–controller. MVC is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

We divided the source code into 6 layers, which are dao, domain, action, service and util.

1. Dao layer: it is responsible for the communication between the entire program and the database.
2. Domain layer: it is contain some classes which is one by one corresponding to the database table, and some other classes. It packages some data, so that the parameters can easy to be passed. Some key database commands are as follows:

Compute the highest rated room type for each hotel:

Compute the 5 best customers:

Compute the highest rated breakfast type across all hotels:

Compute the highest rated service type across all hotels:

1. Sevice layer: it is the glue that connects the dao layer and the action layer. It can get an available room that can be reserved by a customer, calculate the final total amount and pay the reservation bill.

We create a class named FindARoom. Firstly, finding all the rooms in a hotel, then based on the room to find its reservation time in room reservation table, if the time is not conflict with the customer’s reservation time, return the room number of this room. If conflict, then continue to find another avaliable room. If cannot find a available room, return 0.

We create a class named EvaluatePrice. It calculate the sum price of all the orders. Because some room has a discount period, we need to compare the check in and check out date of the customer with the discount date. Getting which days have the discount right. Then sum the room price, breakfast price and service price.

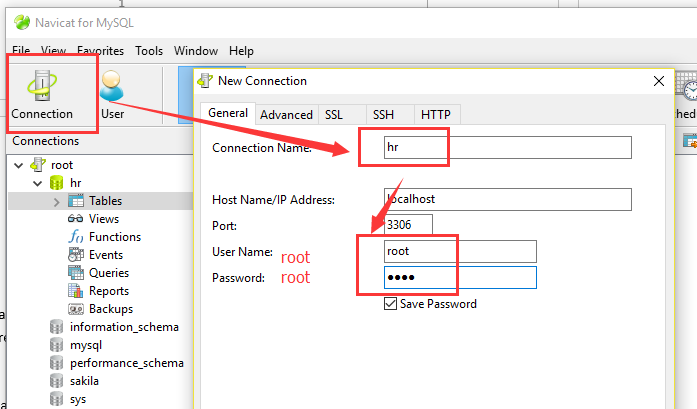
We create a class named Pay. This class inputed a room reservation bill, reservation amount, name of the customer and credit card number. Before input the room reservation information, we need store the invoice number of this bill into database reservation table (the invoice number is created in util layer). Then traversal the room reservation bill, store all the detail information into database.

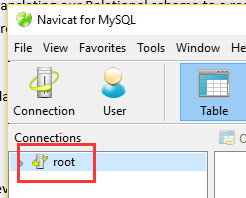
1. Action layer: this layer is equal to the control layer of MVC, it is responsible for calling the function of service layer and processing the data, then return the result to jsp.
2. Jsp layer: it is the same as the view layer of MVC. We store all the jsp files which can not see by the user in the WEB-INF folder. The page can use by the users we put them in webcontent folder. In jsp communicate with server in two ways. One is in the form, another is using href of a label.
3. Util layer: it contains MyUtil.java and FillData.java. By running the FillData.java, can populate the tables in database. MyUtil.java is used to generate a invoice number of reservation.

## Database Creation

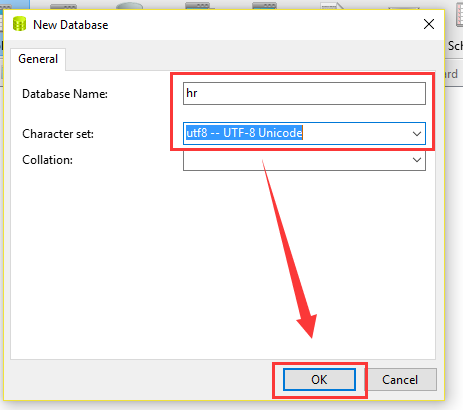
Our database is based on Mysql, and we use Navicat for MySQL to create tables and manage data in the database. To see all SQL commands of table creation, please see the <SQL Commands of Table Creation> part.

1. Creating a connection between mysql and Navicat, click Connection -> type the connection information -> click OK. Then you can see connection icon appear on the left named root.

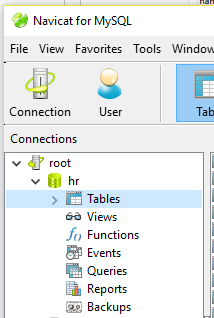




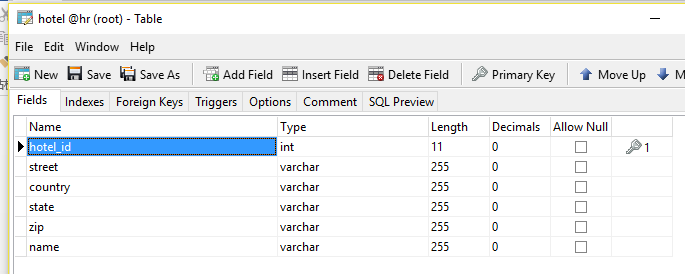
1. Click the connection icon -> right click world -> click New Database… -> type information as follow -> click OK.



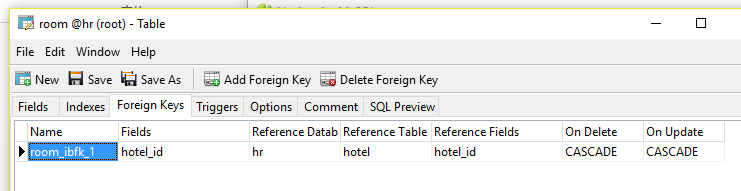
1. Click hr database, right click Tables -> New Table.



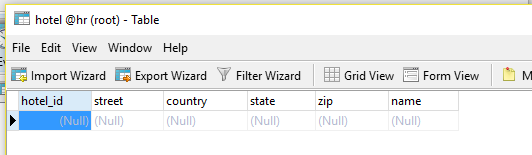
1. Based our relation schema, insert the attribute names, primary keys, foreign keys and so on

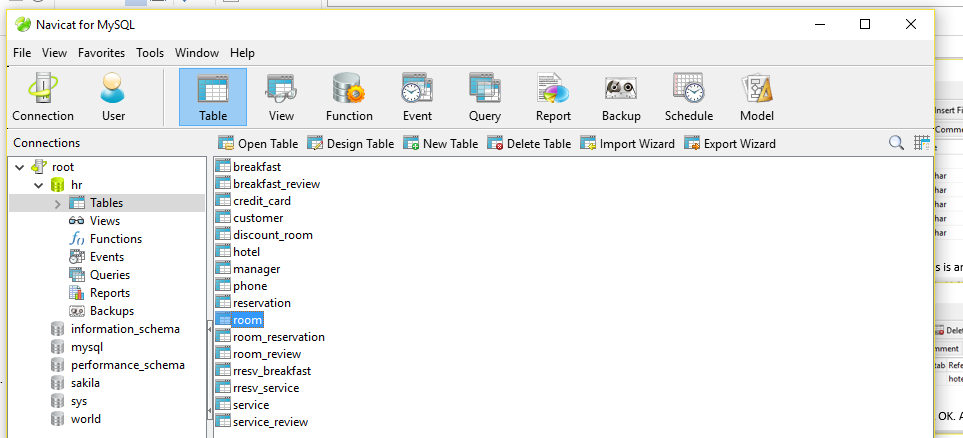


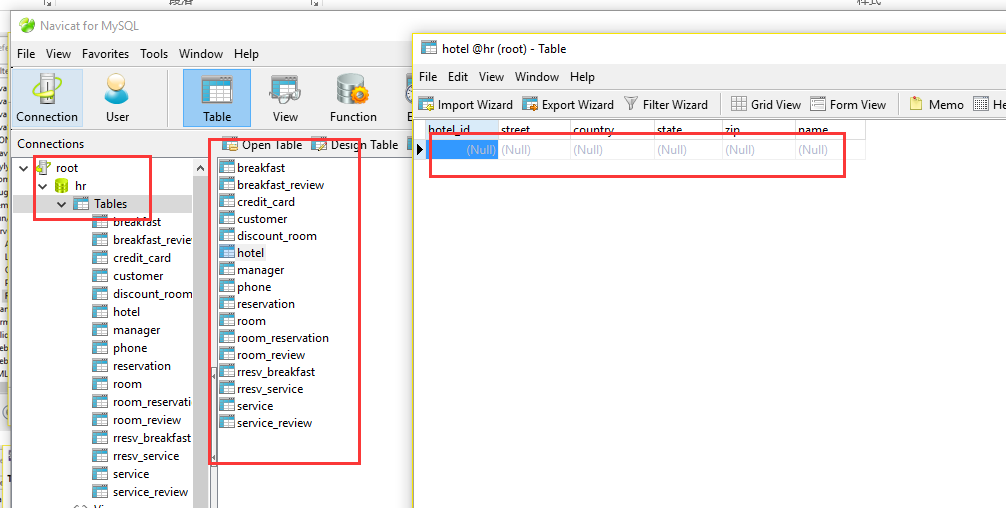
1. Click Foreign Keys to create foreign key, this is an example of room table



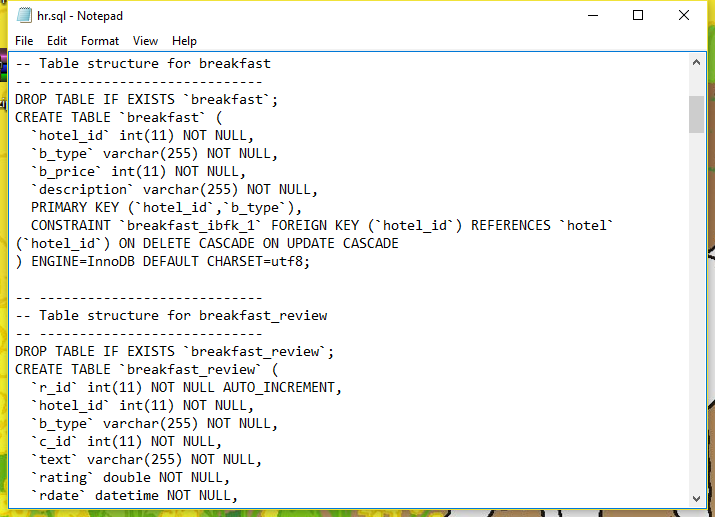
1. Click Save, and type the table’s name, click OK. A new table is created.







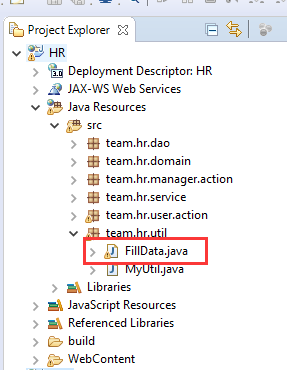
1. To generate a sql file which is include all create table SQL commands, right click hr -> Dump SQL File… -> Save. Here is partial of the table creation SQL commands.

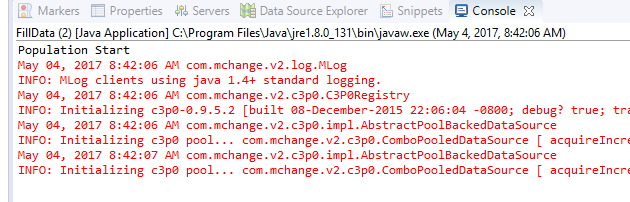


## Table Population

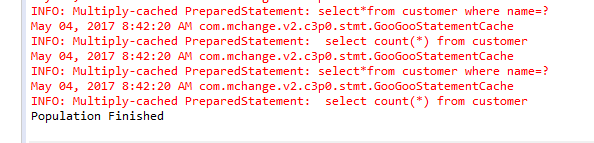
We use java program to populate all of our tables. <SQL Commands of Table Population (partial)> part of our report is giving partial SQL commands of table population. There are 200 hotels in the hotel chain.

1. Just right click the FillData.java on Eclipse, the click run as -> java application. You will see a Population start message in the console panel:

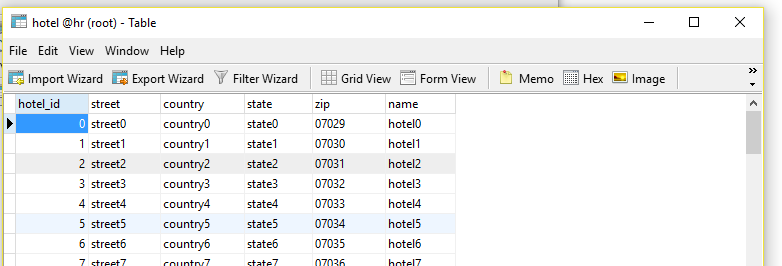


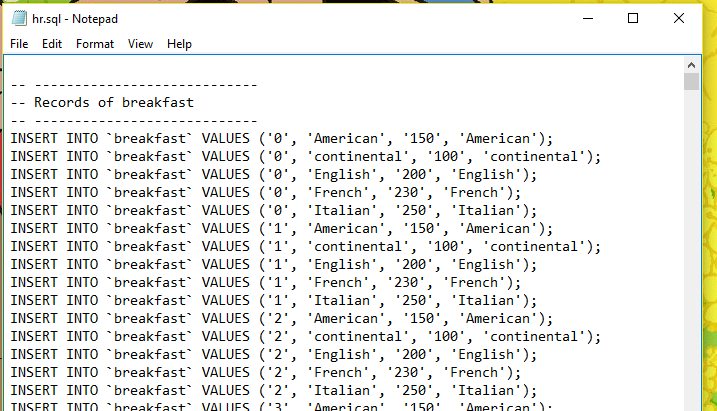


1. When a Population Finished message shows up, it means tables population is done.



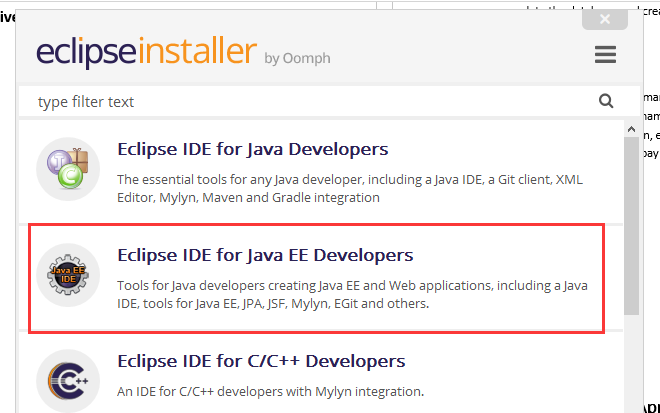
1. Then you will see all tables are full of data.



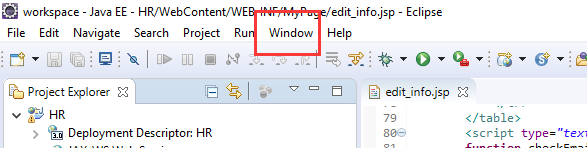
1. To generate a sql file which is include all SQL commands that populate the tables, after run the FillData.java in Eclipse, then right click hr -> Dump SQL File… -> Save in Navicat. 

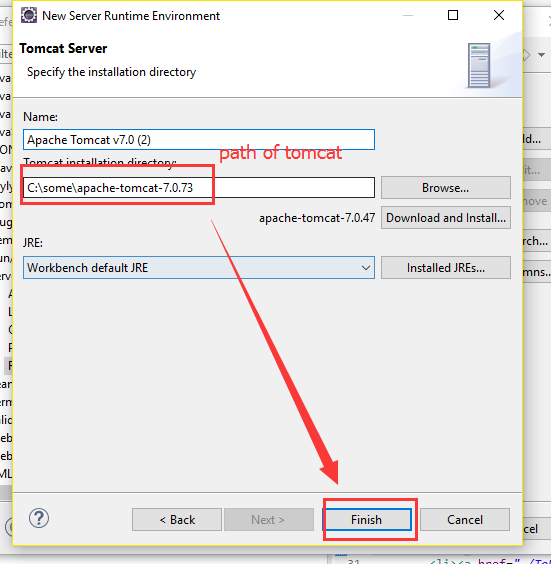
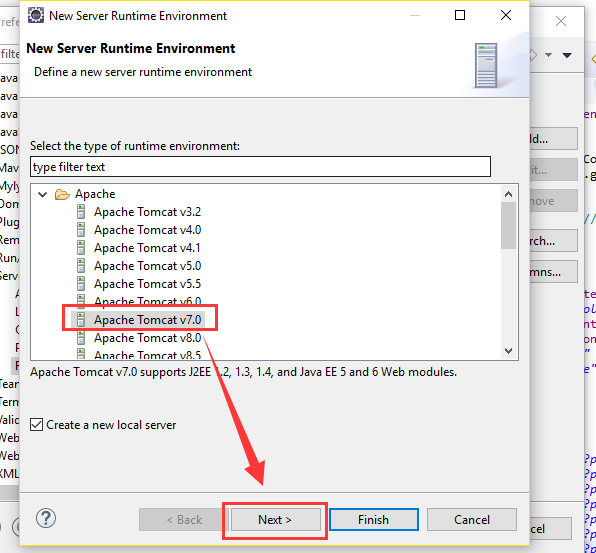
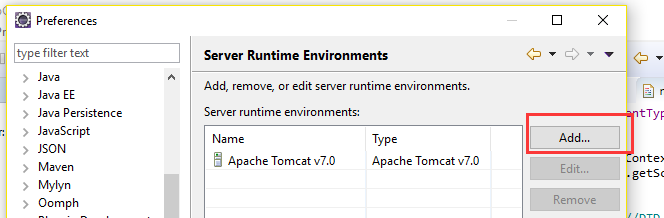
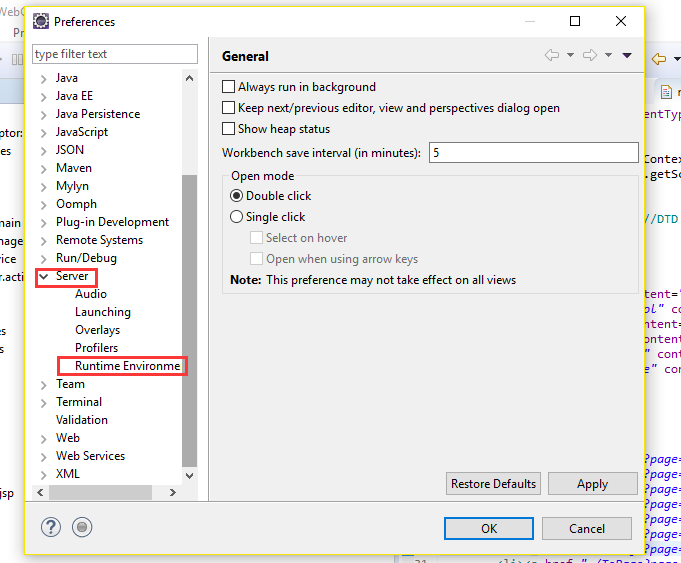
## Run Program

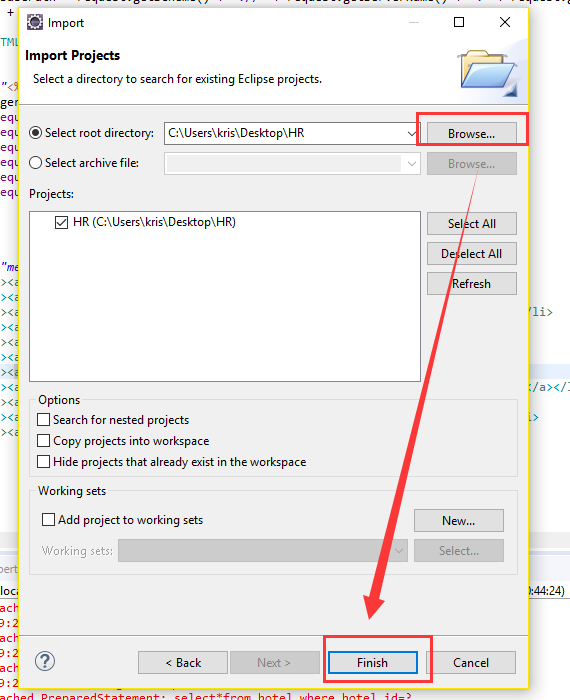
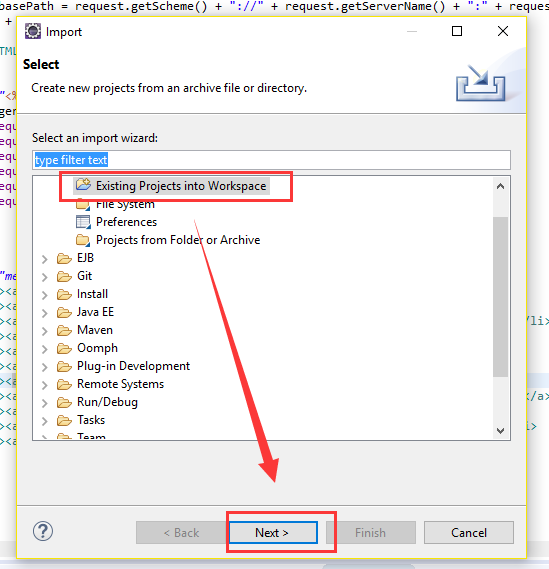
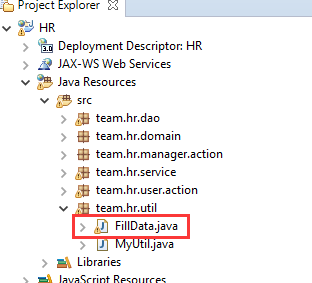
1. First we should setup the environment of JDK, consider this is a very easy thing to be solve, we will not show the steps here. After done of the setup enviroment of JDK, install Eclipse IDE for Java EE Developers:

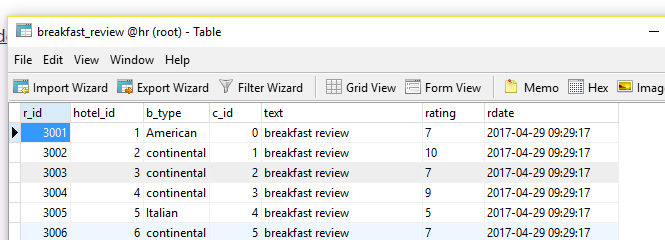


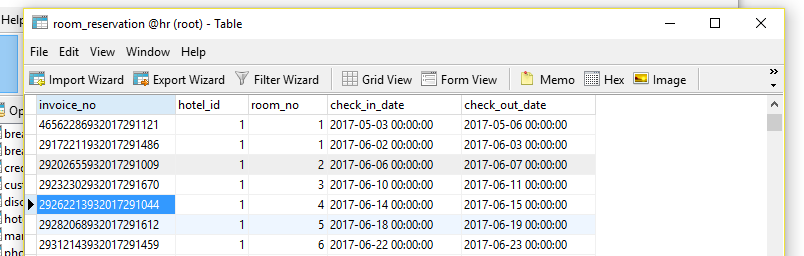
1. Add tomcat to Eclipse, our version of tomat is apache-tomcat-7.0.73:



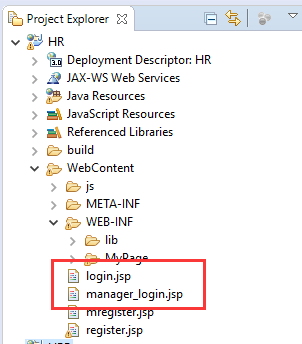


1. To run our source code, just simply import the HR file by click File -> Import, run the fillData.java by click Run As Java Application to populate the tables in database.  
2. After populate all the tables, there are 200 hotels in the Hulton hotel chain, the following is part of the breakfast review table and part of the room reservation table:

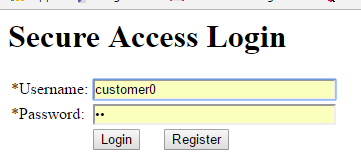




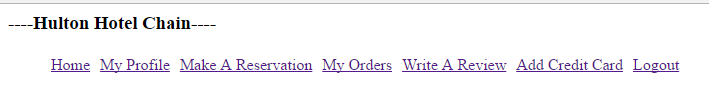
1. For customers, run login.jps by click Run on Server. For managers, run manager\_login.jsp by click Run on Server.

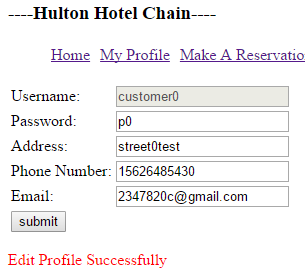


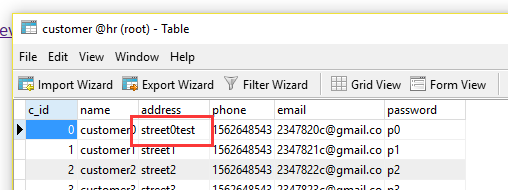
1. After run login.jps, a website will pop up, Type customer0 as username and p0 as password to do a test, then click login in:



1. Then you will see the web page for customers. On the top-left you can see a discount information for one of a hotel:



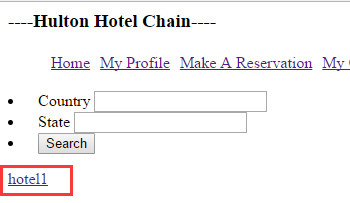
1. Click My Profile to change user’s information, after click submit button, you will see the new information is already in the database. 



1. To make a reservation, just simply click the Make A Reservation button, choose the country and state to find a hotel, use coutry1 and state1 to do a test.

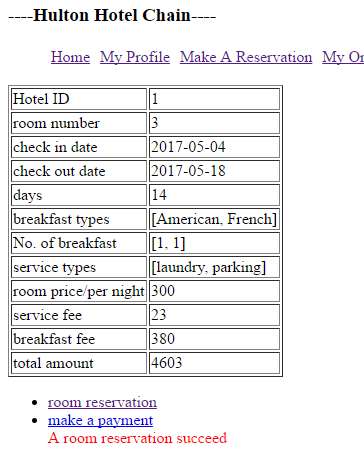




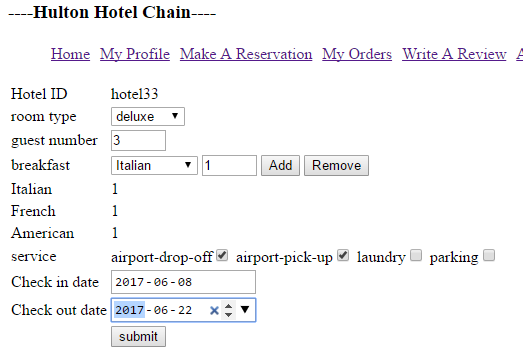


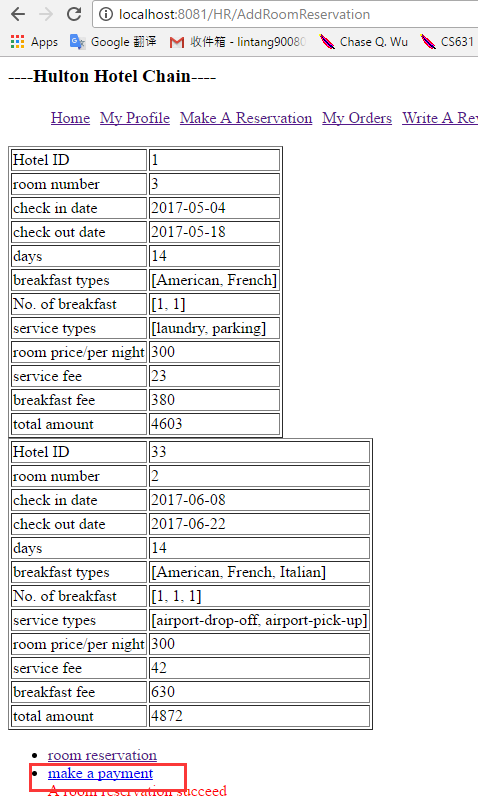
1. Reserve a room:



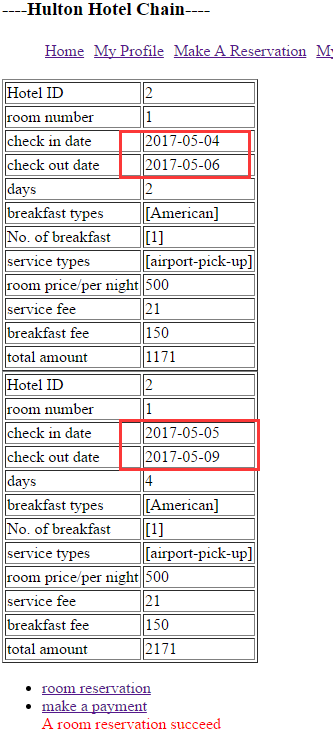


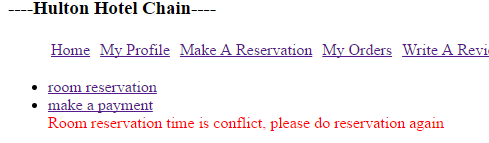
1. Reserve another room, also click room reservation:





1. To pay the reservation, customer just need to click the make a payment button. Or if she wants to add another card to pay the bill, she just need to click the Add a new card button. 
2. If there are same room’s reservation date is overlape, it will give a error message, if the date is no conflict, then it will let the customer to pay the bill.

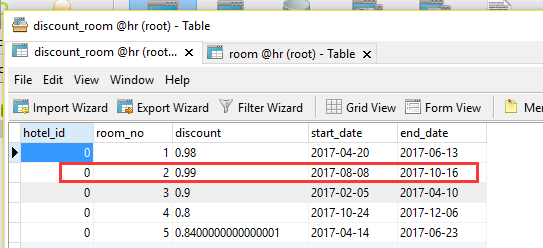


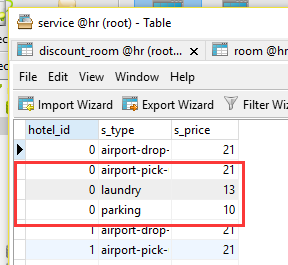


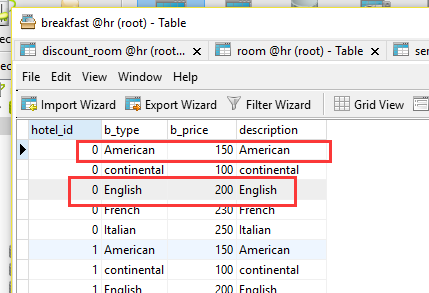
1. By clicking My Orders, customer can see the orders he made:



1. If the room check in and check out date is during the discount period, discounted prices for the room will be taken into account. Breakfast fee is the total fee of all the breakfast orders, service fee also too. For example: room 2 of hotel0’s discount is 0.99 during 2017-08-08 to 2017-10-16, the amount is: (4 \* 600) + (39 \* 600 \* 0.99) + (150 + 200) + (21 + 13 + 10) = 25960

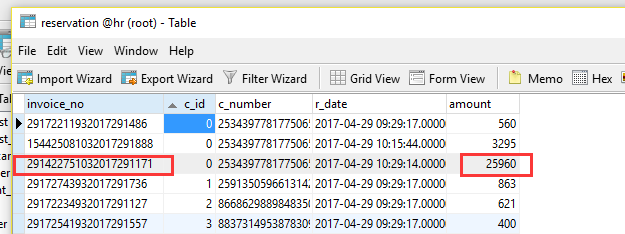


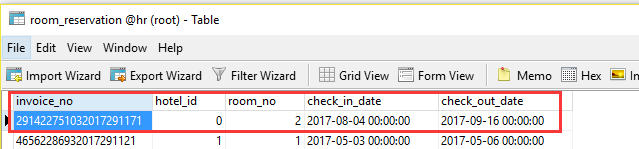


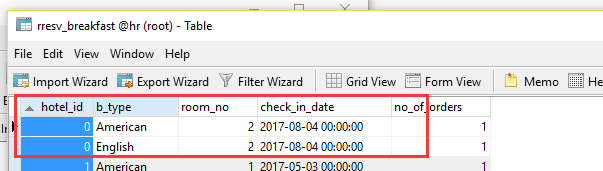


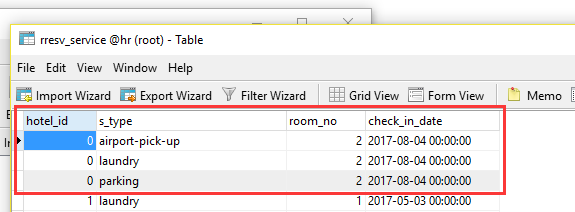


1. And you can find the reservation information in the database:

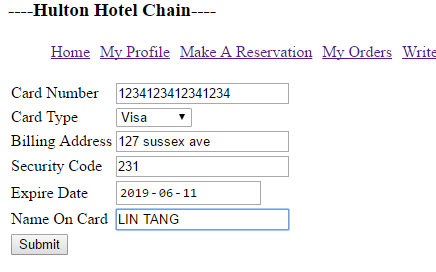


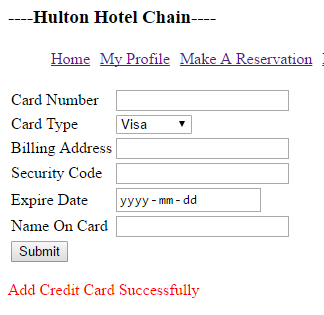




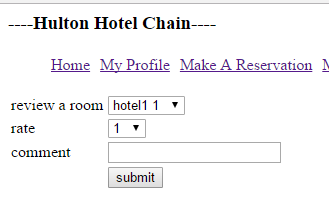
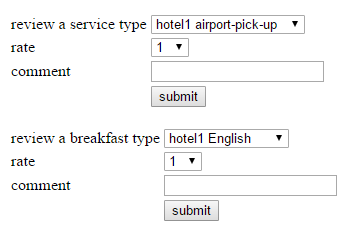


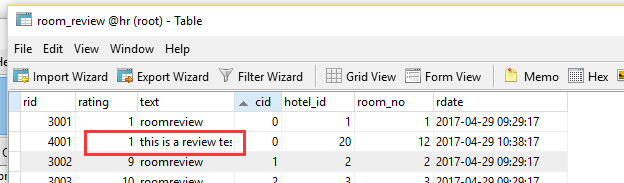
1. By click Add Credit Card, customer can add any card for the future payment.



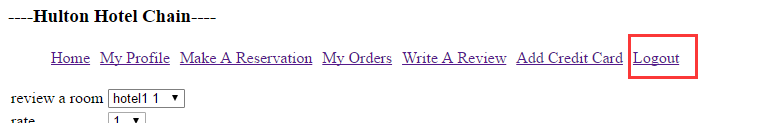


1. To write a review, just click Write A Review, after finish the review, click the submit button to submit your review

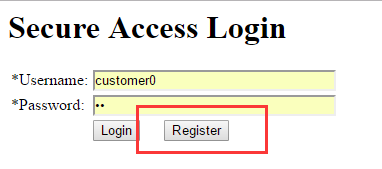
 

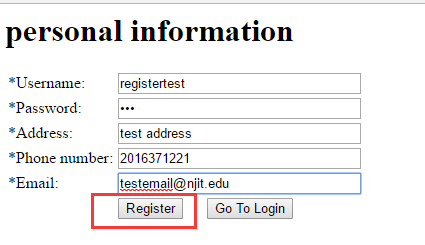


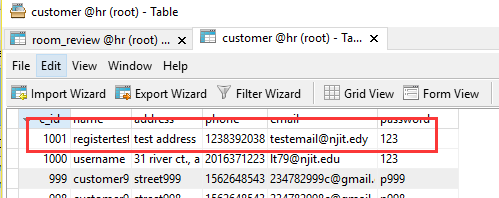
1. Click Logout to log out the website:



1. Click register to create a new account, click signup to submit your account creation



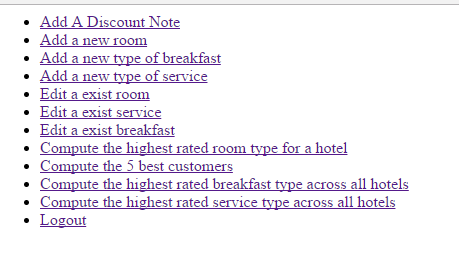




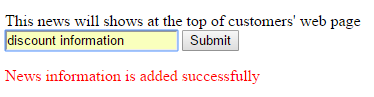
1. For the managers page, run the manager\_login.jsp. We create a default account: username: admin, password: admin for a manager account when doing the table population.

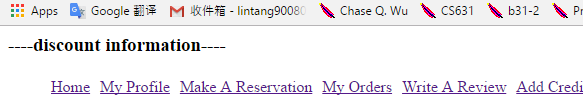


1. User interface of manager:

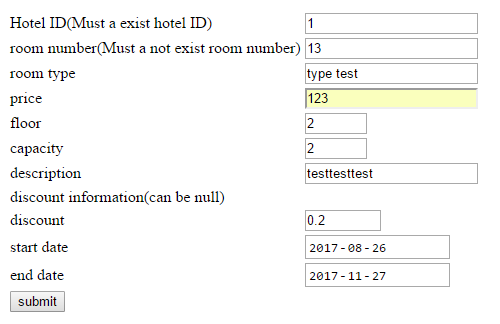


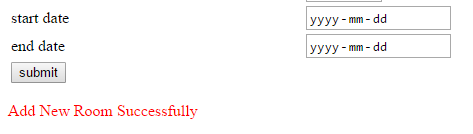
1. Add A Discount Note:

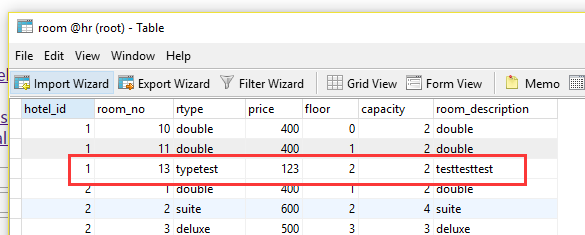


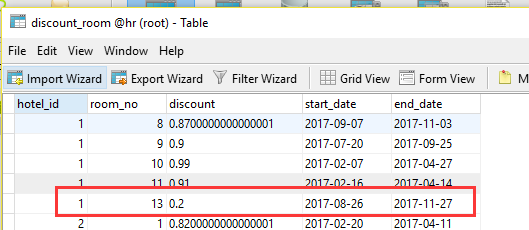


1. Add a new room:

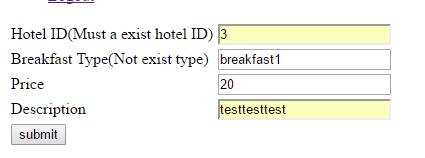




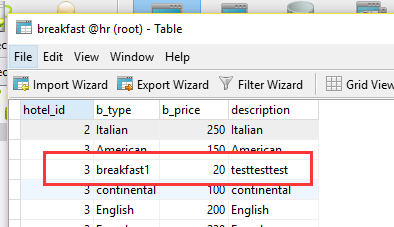




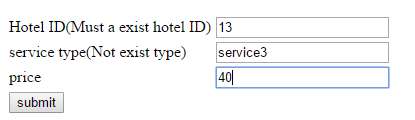
1. Add a new type of breakfast:

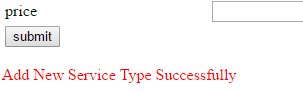


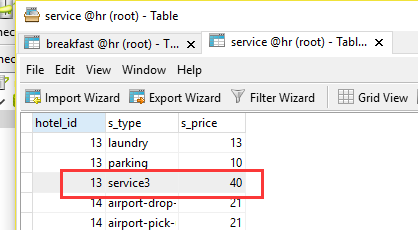




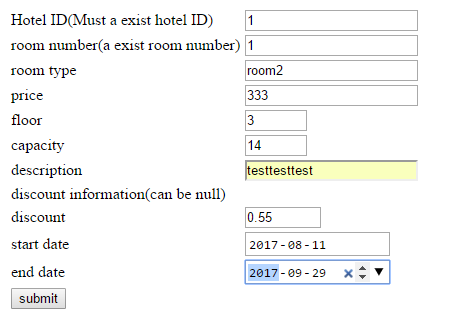
1. Add a new type of service:

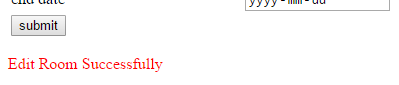


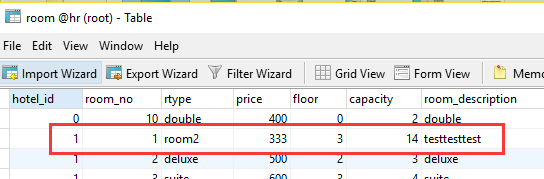


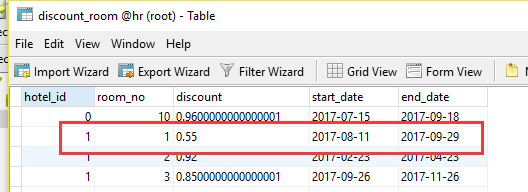


1. Edit a exist room:

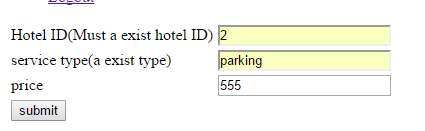


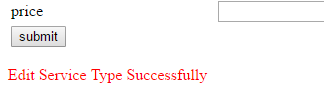


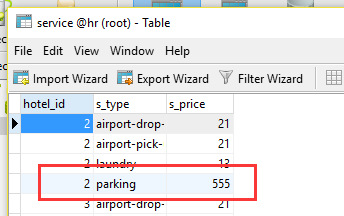




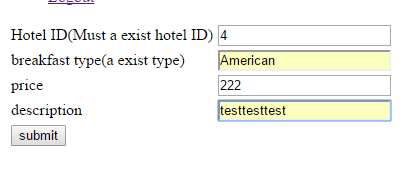
1. Edit a exist service:

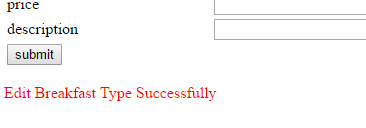


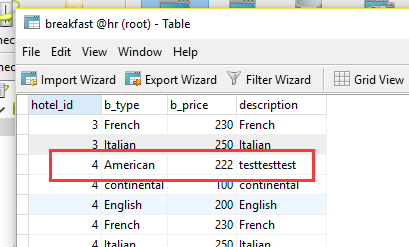




1. Edit a exist breakfast:

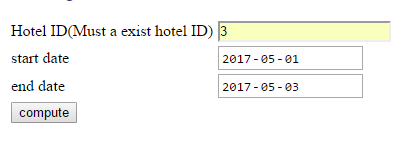


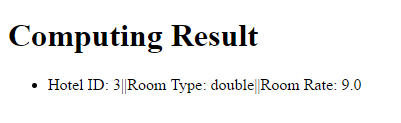




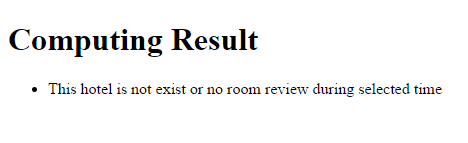
1. Compute the highest rated room type for a hotel:



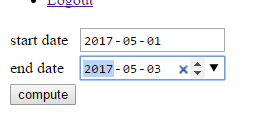


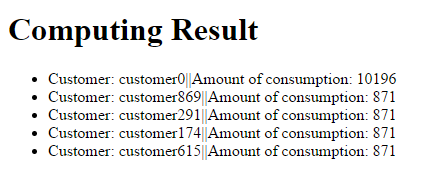


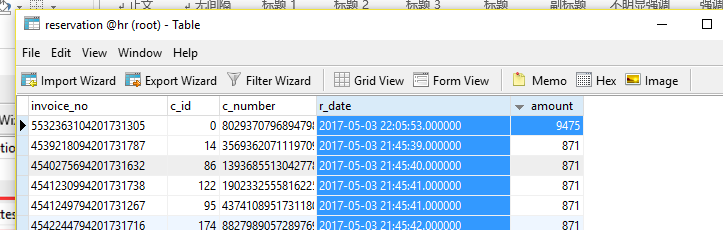


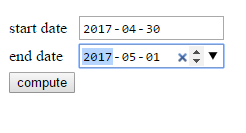
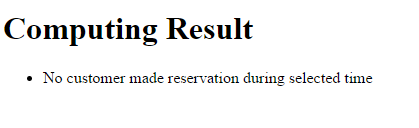


1. Compute the 5 best customers

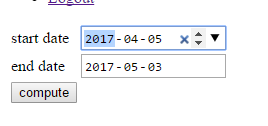




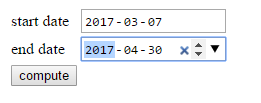


1. Compute the highest rated breakfast type

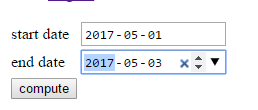
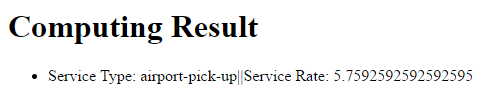
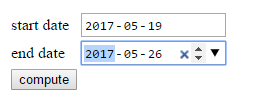




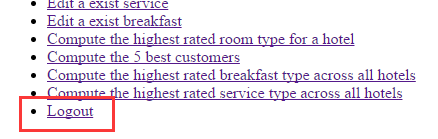




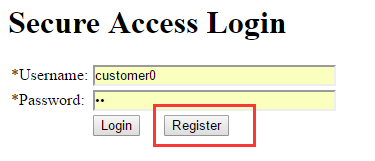
1. Compute the highest rated service type:

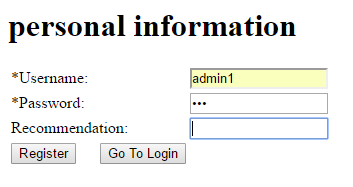
1. Click Logout to logout:

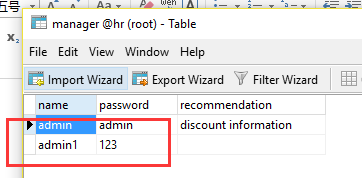


1. Click Register to add a new manager account:



1. Type the username and password, the recommendation can be null, the click Register:

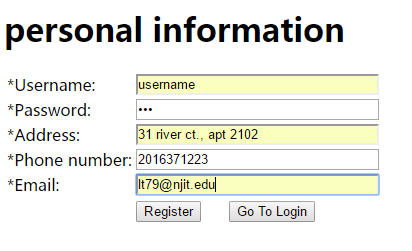
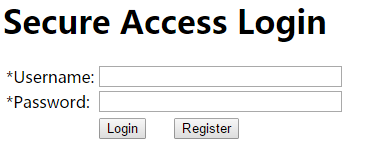




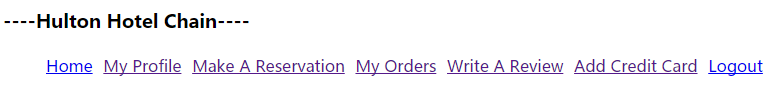
# Users Guide

## Customer Guide

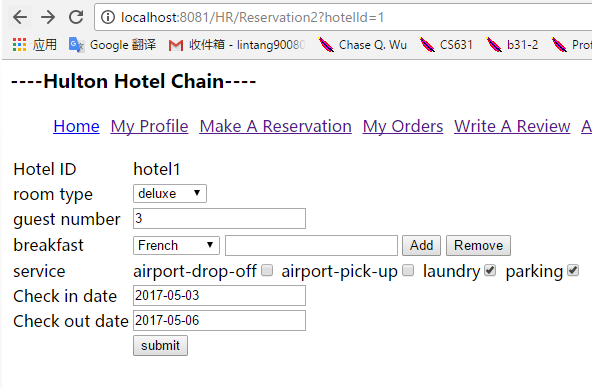
1. Sign up: click the signup button, type your information, click Register. Or click Go To Login to cancel the registration

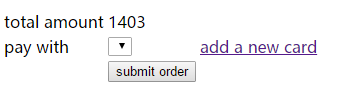


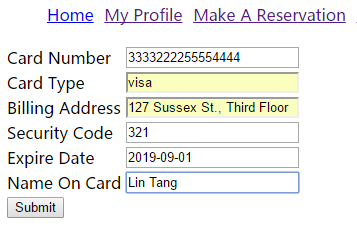
1. Log in: type your username and password, click login button

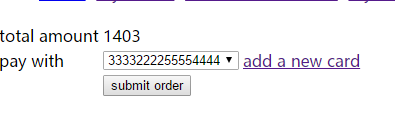


1. Edit your information: click My profile
2. Make a room reservation: click Make A Reservation -> Room reservation -> type country and state -> click Search -> click the hotel -> fill up the reservation information -> click submit –> click make a payment -> add a new card -> click submit -> Make A Reservation -> make a payment -> submit order

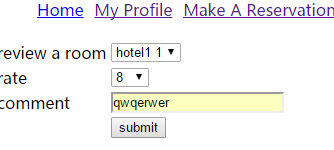








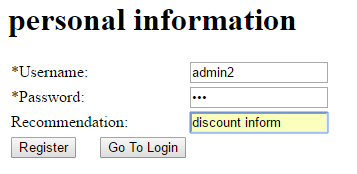
1. See orders: click My Orders
2. Write a review: click Write A Review -> click submit



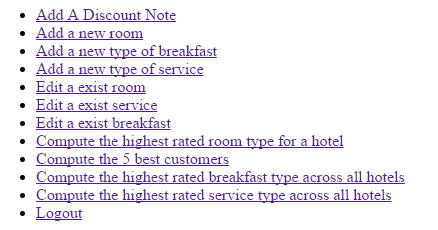
1. Add new credit card: click Add Credit Card and type your card information
2. Log out: click Logout

## Administrator Guide

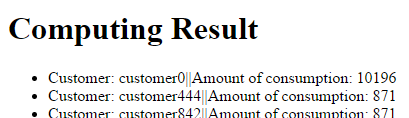
1. Sign up: click Register, type your information, recommendation can be null, then click Register to submit, or click Go To Login to cancel your registration.

1. All the options you can choose to do:



1. Using the Add A Discount Note option to add a welcome message or a discount information, this message will shows at the top of customer’s web page.
2. Using the Add a new room option to add a new room to any of hotels in the hulton hotel chain, and you also can add the room discount information in this option
3. Using the Add a new type of breakfast option to add a new type of breakfast to any of hotels in the hulton hotel chain
4. Using the Add a new type of service option to add a new type of service to any of hotels in the hulton hotel chain
5. Using the Edit a exit room option to edit a room of any of the hotels in the hulton hotel chain and you also can edit the room discount information in this option
6. Using the Edit a exist service option to edit a exist service of any of hotels in the hulton hotel chain.
7. Using the Edit a exist breakfast option to edit a exist breakfast to any of hotels in the hulton hotel chain.
8. The four compute options can let you know the highest rated room type for a hotel, the 5 best customers. Highest rated breakfast type across all hotels and the highest rated service type across all hotels in a certain period of time.



1. You can click the Logout option to log out.

# SQL Commands

## SQL Commands of Table Creation

/\*

Navicat MySQL Data Transfer

Source Server : root

Source Server Version : 50718

Source Host : localhost:3306

Source Database : hr

Target Server Type : MYSQL

Target Server Version : 50718

File Encoding : 65001

Date: 2017-04-29 09:24:59

\*/

SET FOREIGN\_KEY\_CHECKS=0;

-- ----------------------------

-- Table structure for `breakfast`

-- ----------------------------

DROP TABLE IF EXISTS `breakfast`;

CREATE TABLE `breakfast` (

`hotel\_id` int(11) NOT NULL,

`b\_type` varchar(255) NOT NULL,

`b\_price` int(11) NOT NULL,

`description` varchar(255) NOT NULL,

PRIMARY KEY (`hotel\_id`,`b\_type`),

CONSTRAINT `breakfast\_ibfk\_1` FOREIGN KEY (`hotel\_id`) REFERENCES `hotel` (`hotel\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of breakfast

-- ----------------------------

-- ----------------------------

-- Table structure for `breakfast\_review`

-- ----------------------------

DROP TABLE IF EXISTS `breakfast\_review`;

CREATE TABLE `breakfast\_review` (

`r\_id` int(11) NOT NULL AUTO\_INCREMENT,

`hotel\_id` int(11) NOT NULL,

`b\_type` varchar(255) NOT NULL,

`c\_id` int(11) NOT NULL,

`text` varchar(255) NOT NULL,

`rating` double NOT NULL,

`rdate` datetime NOT NULL,

PRIMARY KEY (`r\_id`),

KEY `breakfase\_review\_ibfk\_2` (`hotel\_id`,`b\_type`),

KEY `c\_id` (`c\_id`),

CONSTRAINT `breakfast\_review\_ibfk\_2` FOREIGN KEY (`hotel\_id`, `b\_type`) REFERENCES `breakfast` (`hotel\_id`, `b\_type`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `breakfast\_review\_ibfk\_3` FOREIGN KEY (`c\_id`) REFERENCES `customer` (`c\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=3001 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of breakfast\_review

-- ----------------------------

-- ----------------------------

-- Table structure for `credit\_card`

-- ----------------------------

DROP TABLE IF EXISTS `credit\_card`;

CREATE TABLE `credit\_card` (

`c\_number` varchar(255) NOT NULL,

`c\_type` varchar(255) NOT NULL,

`baddress` varchar(255) NOT NULL,

`code` int(11) NOT NULL,

`exp\_date` datetime NOT NULL,

`name` varchar(255) NOT NULL,

`c\_id` int(11) NOT NULL,

PRIMARY KEY (`c\_number`),

KEY `c\_id` (`c\_id`),

CONSTRAINT `credit\_card\_ibfk\_1` FOREIGN KEY (`c\_id`) REFERENCES `customer` (`c\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of credit\_card

-- ----------------------------

-- ----------------------------

-- Table structure for `customer`

-- ----------------------------

DROP TABLE IF EXISTS `customer`;

CREATE TABLE `customer` (

`c\_id` int(11) NOT NULL,

`name` varchar(255) NOT NULL,

`address` varchar(255) NOT NULL,

`phone` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

PRIMARY KEY (`c\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of customer

-- ----------------------------

-- ----------------------------

-- Table structure for `discount\_room`

-- ----------------------------

DROP TABLE IF EXISTS `discount\_room`;

CREATE TABLE `discount\_room` (

`hotel\_id` int(11) NOT NULL,

`room\_no` int(11) NOT NULL,

`discount` double NOT NULL,

`start\_date` varchar(255) NOT NULL,

`end\_date` varchar(255) NOT NULL,

PRIMARY KEY (`hotel\_id`,`room\_no`),

CONSTRAINT `discount\_room\_ibfk\_1` FOREIGN KEY (`hotel\_id`, `room\_no`) REFERENCES `room` (`hotel\_id`, `room\_no`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of discount\_room

-- ----------------------------

-- ----------------------------

-- Table structure for `hotel`

-- ----------------------------

DROP TABLE IF EXISTS `hotel`;

CREATE TABLE `hotel` (

`hotel\_id` int(11) NOT NULL,

`street` varchar(255) NOT NULL,

`country` varchar(255) NOT NULL,

`state` varchar(255) NOT NULL,

`zip` varchar(255) NOT NULL,

`name` varchar(255) NOT NULL,

PRIMARY KEY (`hotel\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of hotel

-- ----------------------------

-- ----------------------------

-- Table structure for `manager`

-- ----------------------------

DROP TABLE IF EXISTS `manager`;

CREATE TABLE `manager` (

`name` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`recommendation` varchar(1000) DEFAULT NULL,

PRIMARY KEY (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of manager

-- ----------------------------

-- ----------------------------

-- Table structure for `phone`

-- ----------------------------

DROP TABLE IF EXISTS `phone`;

CREATE TABLE `phone` (

`hotel\_id` int(11) NOT NULL,

`phone` varchar(255) NOT NULL,

PRIMARY KEY (`hotel\_id`,`phone`),

KEY `phone\_ibfk\_1` (`hotel\_id`),

CONSTRAINT `phone\_ibfk\_1` FOREIGN KEY (`hotel\_id`) REFERENCES `hotel` (`hotel\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of phone

-- ----------------------------

-- ----------------------------

-- Table structure for `reservation`

-- ----------------------------

DROP TABLE IF EXISTS `reservation`;

CREATE TABLE `reservation` (

`invoice\_no` varchar(255) NOT NULL,

`c\_id` int(11) NOT NULL,

`c\_number` varchar(255) NOT NULL,

`r\_date` datetime(6) NOT NULL,

`amount` int(11) NOT NULL,

PRIMARY KEY (`invoice\_no`),

KEY `c\_id` (`c\_id`),

KEY `c\_number` (`c\_number`),

CONSTRAINT `reservation\_ibfk\_2` FOREIGN KEY (`c\_id`) REFERENCES `customer` (`c\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `reservation\_ibfk\_3` FOREIGN KEY (`c\_number`) REFERENCES `credit\_card` (`c\_number`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of reservation

-- ----------------------------

-- ----------------------------

-- Table structure for `room`

-- ----------------------------

DROP TABLE IF EXISTS `room`;

CREATE TABLE `room` (

`hotel\_id` int(11) NOT NULL,

`room\_no` int(11) NOT NULL,

`rtype` varchar(255) NOT NULL,

`price` int(11) NOT NULL,

`floor` int(11) NOT NULL,

`capacity` int(11) NOT NULL,

`room\_description` varchar(255) NOT NULL,

PRIMARY KEY (`hotel\_id`,`room\_no`),

CONSTRAINT `room\_ibfk\_1` FOREIGN KEY (`hotel\_id`) REFERENCES `hotel` (`hotel\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of room

-- ----------------------------

-- ----------------------------

-- Table structure for `room\_reservation`

-- ----------------------------

DROP TABLE IF EXISTS `room\_reservation`;

CREATE TABLE `room\_reservation` (

`invoice\_no` varchar(255) NOT NULL,

`hotel\_id` int(11) NOT NULL,

`room\_no` int(11) NOT NULL,

`check\_in\_date` datetime NOT NULL,

`check\_out\_date` datetime NOT NULL,

PRIMARY KEY (`hotel\_id`,`room\_no`,`check\_in\_date`),

KEY `invoice\_no` (`invoice\_no`),

CONSTRAINT `room\_reservation\_ibfk\_1` FOREIGN KEY (`hotel\_id`, `room\_no`) REFERENCES `room` (`hotel\_id`, `room\_no`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `room\_reservation\_ibfk\_2` FOREIGN KEY (`invoice\_no`) REFERENCES `reservation` (`invoice\_no`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of room\_reservation

-- ----------------------------

-- ----------------------------

-- Table structure for `room\_review`

-- ----------------------------

DROP TABLE IF EXISTS `room\_review`;

CREATE TABLE `room\_review` (

`rid` int(11) NOT NULL AUTO\_INCREMENT,

`rating` int(11) NOT NULL,

`text` varchar(255) NOT NULL,

`cid` int(11) NOT NULL,

`hotel\_id` int(11) NOT NULL,

`room\_no` int(11) NOT NULL,

`rdate` datetime DEFAULT NULL,

PRIMARY KEY (`rid`),

KEY `room\_review\_ibfk\_1` (`hotel\_id`,`room\_no`),

KEY `cid` (`cid`),

CONSTRAINT `room\_review\_ibfk\_1` FOREIGN KEY (`hotel\_id`, `room\_no`) REFERENCES `room` (`hotel\_id`, `room\_no`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `room\_review\_ibfk\_2` FOREIGN KEY (`cid`) REFERENCES `customer` (`c\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=3001 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of room\_review

-- ----------------------------

-- ----------------------------

-- Table structure for `rresv\_breakfast`

-- ----------------------------

DROP TABLE IF EXISTS `rresv\_breakfast`;

CREATE TABLE `rresv\_breakfast` (

`hotel\_id` int(11) NOT NULL,

`b\_type` varchar(255) NOT NULL,

`room\_no` int(11) NOT NULL,

`check\_in\_date` datetime NOT NULL,

`no\_of\_orders` int(255) NOT NULL,

PRIMARY KEY (`b\_type`,`hotel\_id`,`room\_no`,`check\_in\_date`),

KEY `rresv\_breakfast\_ibfk\_1` (`hotel\_id`,`b\_type`),

KEY `hotel\_id` (`hotel\_id`,`room\_no`,`check\_in\_date`),

CONSTRAINT `rresv\_breakfast\_ibfk\_1` FOREIGN KEY (`hotel\_id`, `b\_type`) REFERENCES `breakfast` (`hotel\_id`, `b\_type`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `rresv\_breakfast\_ibfk\_2` FOREIGN KEY (`hotel\_id`, `room\_no`, `check\_in\_date`) REFERENCES `room\_reservation` (`hotel\_id`, `room\_no`, `check\_in\_date`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of rresv\_breakfast

-- ----------------------------

-- ----------------------------

-- Table structure for `rresv\_service`

-- ----------------------------

DROP TABLE IF EXISTS `rresv\_service`;

CREATE TABLE `rresv\_service` (

`hotel\_id` int(11) NOT NULL,

`s\_type` varchar(255) NOT NULL,

`room\_no` int(11) NOT NULL,

`check\_in\_date` datetime NOT NULL,

PRIMARY KEY (`s\_type`,`hotel\_id`,`room\_no`,`check\_in\_date`),

KEY `rresv\_service\_ibfk\_1` (`hotel\_id`,`s\_type`),

KEY `hotel\_id` (`hotel\_id`,`room\_no`,`check\_in\_date`),

CONSTRAINT `rresv\_service\_ibfk\_1` FOREIGN KEY (`hotel\_id`, `s\_type`) REFERENCES `service` (`hotel\_id`, `s\_type`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `rresv\_service\_ibfk\_2` FOREIGN KEY (`hotel\_id`, `room\_no`, `check\_in\_date`) REFERENCES `room\_reservation` (`hotel\_id`, `room\_no`, `check\_in\_date`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of rresv\_service

-- ----------------------------

-- ----------------------------

-- Table structure for `service`

-- ----------------------------

DROP TABLE IF EXISTS `service`;

CREATE TABLE `service` (

`hotel\_id` int(11) NOT NULL,

`s\_type` varchar(255) NOT NULL,

`s\_price` int(11) NOT NULL,

PRIMARY KEY (`hotel\_id`,`s\_type`),

CONSTRAINT `service\_ibfk\_1` FOREIGN KEY (`hotel\_id`) REFERENCES `hotel` (`hotel\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of service

-- ----------------------------

-- ----------------------------

-- Table structure for `service\_review`

-- ----------------------------

DROP TABLE IF EXISTS `service\_review`;

CREATE TABLE `service\_review` (

`r\_id` int(11) NOT NULL AUTO\_INCREMENT,

`hotel\_id` int(11) NOT NULL,

`s\_type` varchar(255) NOT NULL,

`c\_id` int(11) NOT NULL,

`rating` double NOT NULL,

`text` varchar(255) NOT NULL,

`rdate` datetime NOT NULL,

PRIMARY KEY (`r\_id`),

KEY `service\_review\_ibfk\_2` (`hotel\_id`,`s\_type`),

KEY `c\_id` (`c\_id`),

CONSTRAINT `service\_review\_ibfk\_2` FOREIGN KEY (`hotel\_id`, `s\_type`) REFERENCES `service` (`hotel\_id`, `s\_type`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `service\_review\_ibfk\_3` FOREIGN KEY (`c\_id`) REFERENCES `customer` (`c\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=3001 DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of service\_review

-- ----------------------------

## SQL Commands of Table Population (Partial)

-- ----------------------------

-- Records of breakfast

-- ----------------------------

INSERT INTO `breakfast` VALUES ('0', 'American', '150', 'American');

INSERT INTO `breakfast` VALUES ('0', 'continental', '100', 'continental');

INSERT INTO `breakfast` VALUES ('0', 'English', '200', 'English');

INSERT INTO `breakfast` VALUES ('0', 'French', '230', 'French');

INSERT INTO `breakfast` VALUES ('0', 'Italian', '250', 'Italian');

INSERT INTO `breakfast` VALUES ('1', 'American', '150', 'American');

-- ----------------------------

-- Records of breakfast\_review

-- ----------------------------

INSERT INTO `breakfast\_review` VALUES ('3001', '1', 'American', '0', 'breakfast review', '7', '2017-04-29 09:29:17');

INSERT INTO `breakfast\_review` VALUES ('3002', '2', 'continental', '1', 'breakfast review', '10', '2017-04-29 09:29:17');

INSERT INTO `breakfast\_review` VALUES ('3003', '3', 'continental', '2', 'breakfast review', '7', '2017-04-29 09:29:17');

INSERT INTO `breakfast\_review` VALUES ('3004', '4', 'continental', '3', 'breakfast review', '9', '2017-04-29 09:29:17');

-- ----------------------------

-- Records of credit\_card

-- ----------------------------

INSERT INTO `credit\_card` VALUES ('1017258780368178', 'Amex', 'address410', '535', '2030-04-03 00:00:00', 'c410', '410');

INSERT INTO `credit\_card` VALUES ('1043210066621620', 'Discover', 'address450', '539', '2030-04-03 00:00:00', 'c450', '450');

INSERT INTO `credit\_card` VALUES ('1052634640785415', 'Discover', 'address389', '692', '2030-04-03 00:00:00', 'c389', '389');

INSERT INTO `credit\_card` VALUES ('1066565136632600', 'Master', 'address129', '456', '2030-04-03 00:00:00', 'c129', '129');

-- ----------------------------

-- Records of customer

-- ----------------------------

INSERT INTO `customer` VALUES ('0', 'customer0', 'street0', '15626485430', '2347820c@gmail.com', 'p0');

INSERT INTO `customer` VALUES ('1', 'customer1', 'street1', '15626485431', '2347821c@gmail.com', 'p1');

INSERT INTO `customer` VALUES ('2', 'customer2', 'street2', '15626485432', '2347822c@gmail.com', 'p2');

INSERT INTO `customer` VALUES ('3', 'customer3', 'street3', '15626485433', '2347823c@gmail.com', 'p3');

-- ----------------------------

-- Records of discount\_room

-- ----------------------------

INSERT INTO `discount\_room` VALUES ('0', '1', '0.98', '2017-04-20', '2017-06-13');

INSERT INTO `discount\_room` VALUES ('0', '2', '0.99', '2017-08-08', '2017-10-16');

INSERT INTO `discount\_room` VALUES ('0', '3', '0.9', '2017-02-05', '2017-04-10');

INSERT INTO `discount\_room` VALUES ('0', '4', '0.8', '2017-10-24', '2017-12-06');

INSERT INTO `discount\_room` VALUES ('0', '5', '0.8400000000000001', '2017-04-14', '2017-06-23');

-- ----------------------------

-- Records of hotel

-- ----------------------------

INSERT INTO `hotel` VALUES ('0', 'street0', 'country0', 'state0', '07029', 'hotel0');

INSERT INTO `hotel` VALUES ('1', 'street1', 'country1', 'state1', '07030', 'hotel1');

INSERT INTO `hotel` VALUES ('2', 'street2', 'country2', 'state2', '07031', 'hotel2');

INSERT INTO `hotel` VALUES ('3', 'street3', 'country3', 'state3', '07032', 'hotel3');

-- ----------------------------

-- Records of manager

-- ----------------------------

INSERT INTO `manager` VALUES ('admin', 'admin', 'Hulton Hotel Chain');

-- ----------------------------

-- Records of phone

-- ----------------------------

INSERT INTO `phone` VALUES ('0', '13911124555');

INSERT INTO `phone` VALUES ('0', '13917793333');

INSERT INTO `phone` VALUES ('0', '13927698906');

INSERT INTO `phone` VALUES ('1', '13956801052');

INSERT INTO `phone` VALUES ('1', '13957156215');

INSERT INTO `phone` VALUES ('1', '13959076272');

-- ----------------------------

-- Records of reservation

-- ----------------------------

INSERT INTO `reservation` VALUES ('29172056932017291190', '12', '1400655115772624', '2017-04-29 09:29:17.000000', '510');

INSERT INTO `reservation` VALUES ('29172124932017291466', '9', '7389753215178781', '2017-04-29 09:29:17.000000', '560');

INSERT INTO `reservation` VALUES ('29172211932017291486', '0', '2534397781775065', '2017-04-29 09:29:17.000000', '560');

INSERT INTO `reservation` VALUES ('29172234932017291127', '2', '8668629889848350', '2017-04-29 09:29:17.000000', '621');

-- ----------------------------

-- Records of room

-- ----------------------------

INSERT INTO `room` VALUES ('0', '1', 'standard', '300', '1', '1', 'standard');

INSERT INTO `room` VALUES ('0', '2', 'suite', '600', '2', '4', 'suite');

INSERT INTO `room` VALUES ('0', '3', 'deluxe', '500', '3', '3', 'deluxe');

INSERT INTO `room` VALUES ('0', '4', 'standard', '300', '4', '1', 'standard');

INSERT INTO `room` VALUES ('0', '5', 'standard', '300', '0', '1', 'standard');

INSERT INTO `room` VALUES ('1', '1', 'deluxe', '500', '1', '3', 'deluxe');

INSERT INTO `room` VALUES ('1', '2', 'deluxe', '500', '2', '3', 'deluxe');

INSERT INTO `room` VALUES ('1', '3', 'suite', '600', '3', '4', 'suite');

-- ----------------------------

-- Records of room\_reservation

-- ----------------------------

INSERT INTO `room\_reservation` VALUES ('29172211932017291486', '1', '1', '2017-06-02 00:00:00', '2017-06-03 00:00:00');

INSERT INTO `room\_reservation` VALUES ('29202655932017291009', '1', '2', '2017-06-06 00:00:00', '2017-06-07 00:00:00');

INSERT INTO `room\_reservation` VALUES ('29232302932017291670', '1', '3', '2017-06-10 00:00:00', '2017-06-11 00:00:00');

INSERT INTO `room\_reservation` VALUES ('29262213932017291044', '1', '4', '2017-06-14 00:00:00', '2017-06-15 00:00:00');

INSERT INTO `room\_reservation` VALUES ('29282068932017291612', '1', '5', '2017-06-18 00:00:00', '2017-06-19 00:00:00');

-- ----------------------------

-- Records of room\_review

-- ----------------------------

INSERT INTO `room\_review` VALUES ('3001', '1', 'roomreview', '0', '1', '1', '2017-04-29 09:29:17');

INSERT INTO `room\_review` VALUES ('3002', '9', 'roomreview', '1', '2', '2', '2017-04-29 09:29:17');

INSERT INTO `room\_review` VALUES ('3003', '10', 'roomreview', '2', '3', '3', '2017-04-29 09:29:17');

INSERT INTO `room\_review` VALUES ('3004', '10', 'roomreview', '3', '4', '4', '2017-04-29 09:29:17');

INSERT INTO `room\_review` VALUES ('3005', '7', 'roomreview', '4', '5', '5', '2017-04-29 09:29:17');

-- ----------------------------

-- Records of rresv\_breakfast

-- ----------------------------

INSERT INTO `rresv\_breakfast` VALUES ('2', 'American', '6', '2017-06-18 00:00:00', '1');

INSERT INTO `rresv\_breakfast` VALUES ('3', 'American', '6', '2017-06-14 00:00:00', '1');

INSERT INTO `rresv\_breakfast` VALUES ('4', 'American', '4', '2017-06-02 00:00:00', '1');

INSERT INTO `rresv\_breakfast` VALUES ('5', 'American', '1', '2017-06-22 00:00:00', '1');

INSERT INTO `rresv\_breakfast` VALUES ('5', 'American', '9', '2017-06-18 00:00:00', '1');

-- ----------------------------

-- Records of rresv\_service

-- ----------------------------

INSERT INTO `rresv\_service` VALUES ('1', 'parking', '1', '2017-06-02 00:00:00');

INSERT INTO `rresv\_service` VALUES ('1', 'airport-drop-off', '2', '2017-06-06 00:00:00');

INSERT INTO `rresv\_service` VALUES ('1', 'airport-drop-off', '3', '2017-06-10 00:00:00');

INSERT INTO `rresv\_service` VALUES ('1', 'airport-drop-off', '4', '2017-06-14 00:00:00');

INSERT INTO `rresv\_service` VALUES ('1', 'airport-pick-up', '5', '2017-06-18 00:00:00');

INSERT INTO `rresv\_service` VALUES ('1', 'airport-pick-up', '6', '2017-06-22 00:00:00');

-- ----------------------------

-- Records of service

-- ----------------------------

INSERT INTO `service` VALUES ('0', 'airport-drop-off', '21');

INSERT INTO `service` VALUES ('0', 'airport-pick-up', '21');

INSERT INTO `service` VALUES ('0', 'laundry', '13');

INSERT INTO `service` VALUES ('0', 'parking', '10');

INSERT INTO `service` VALUES ('1', 'airport-drop-off', '21');

INSERT INTO `service` VALUES ('1', 'airport-pick-up', '21');

INSERT INTO `service` VALUES ('1', 'laundry', '13');

INSERT INTO `service` VALUES ('1', 'parking', '10');

-- ----------------------------

-- Records of service\_review

-- ----------------------------

INSERT INTO `service\_review` VALUES ('3001', '1', 'airport-pick-up', '0', '4', 'service review', '2017-04-29 09:29:17');

INSERT INTO `service\_review` VALUES ('3002', '2', 'parking', '1', '5', 'service review', '2017-04-29 09:29:17');

INSERT INTO `service\_review` VALUES ('3003', '3', 'airport-drop-off', '2', '3', 'service review', '2017-04-29 09:29:17');

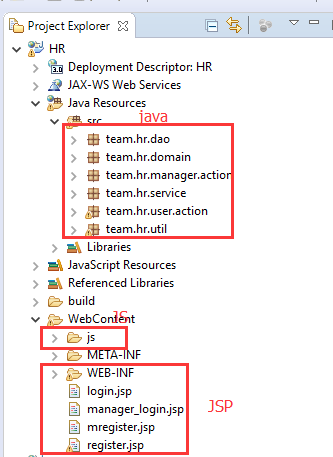
INSERT INTO `service\_review` VALUES ('3004', '4', 'laundry', '3', '5', 'service review', '2017-04-29 09:29:17');

INSERT INTO `service\_review` VALUES ('3005', '5', 'airport-pick-up', '4', '2', 'service review', '2017-04-29 09:29:17');

INSERT INTO `service\_review` VALUES ('3006', '6', 'parking', '5', '9', 'service review', '2017-04-29 09:29:17');

# Source Code

There are three part of our source code: java part, jsp and js part.



## Java (Partial)

Part of MyDao.java

**private** **void** execSQLNoRs(String sql, Object... args) {

PreparedStatement ps = **null**;

ResultSet rs = **null**;

**try** {

ps = getCon().prepareStatement(sql);

**for** (**int** i = 0; i < args.length; i++) {

**if**(args.getClass().getName().equals("java.sql.Date")){

ps.setTimestamp(i+1, (Timestamp) args[i]);

}

**else**{

ps.setObject(i + 1, args[i]);

}

}

ps.execute();

rs = ps.getResultSet();

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

**try** {

**if**(rs!=**null**)

rs.close();

**if**(ps!=**null**)

ps.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

}

}

}

**public** **int** queryCount(String tableName){

**int** result = 0;

String sql = " select count(\*) from "+tableName;

ResultSet rs = execSQL(sql);

**try** {

**if**(rs.next()){

result = rs.getInt(1);

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

**try** {

**if**(rs!=**null**)rs.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**return** result;

}

Hotel.java

**package** team.hr.domain;

**import** java.util.List;

**public** **class** Hotel {

**private** Integer hotelId;

**private** String stree;

**private** String country;

**private** String state;

**private** String zip;

**private** String name;

**private** List<String> rtype;

**private** List<String> btype;

**private** List<String> stype;

**public** List<String> getRtype() {

**return** rtype;

}

**public** **void** setRtype(List<String> rtype) {

**this**.rtype = rtype;

}

**public** List<String> getBtype() {

**return** btype;

}

**public** **void** setBtype(List<String> btype) {

**this**.btype = btype;

}

**public** List<String> getStype() {

**return** stype;

}

**public** **void** setStype(List<String> stype) {

**this**.stype = stype;

}

**public** Hotel(){

}

**public** Hotel(Integer hotelId, String stree, String country, String state, String zip, String name) {

**super**();

**this**.hotelId = hotelId;

**this**.stree = stree;

**this**.country = country;

**this**.state = state;

**this**.zip = zip;

**this**.name = name;

}

**public** Integer getHotelId() {

**return** hotelId;

}

**public** **void** setHotelId(Integer hotelId) {

**this**.hotelId = hotelId;

}

**public** String getStree() {

**return** stree;

}

**public** **void** setStree(String stree) {

**this**.stree = stree;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** String getZip() {

**return** zip;

}

**public** **void** setZip(String zip) {

**this**.zip = zip;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

Part of MyService.java

**public** Integer findARoom(RoomReservation roomReservation) {

Integer hotelId = roomReservation.getHotelId();

ResultSet rs1 = *myDao*.queryRoomByHotelId(hotelId);

Integer result = 0;

**try** {

**while** (rs1.next()) {

Integer capacity = rs1.getInt(6);

**if**(capacity<roomReservation.getCapacity()){

**continue**;

}

**if**(!roomReservation.getRtype().equals(rs1.getString(3))){

**continue**;

}

Integer roomNo = rs1.getInt(2);

ResultSet rs2 = *myDao*.queryRoomReservation(roomNo,hotelId);

**boolean** flag = **true**;

**while** (rs2.next()) {

Date inDate = rs2.getDate(4);

Date outDate = rs2.getDate(5);

**if**(!(roomReservation.getInDate().getTime()>=outDate.getTime()||roomReservation.getOutDate().getTime()<=inDate.getTime())){

flag = **false**;

**break**;

}

}

**if** (flag) {

result = roomNo;

**break**;

}

}

} **catch** (SQLException e) {

e.printStackTrace();

}**finally**{

**try** {

**if**(rs1!=**null**)rs1.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**return** result;

}

**public** **void** pay(List<RoomReservation> list, Integer amount, String cName, String cNumber) {

Customer customer = *myDao*.queryCustomer(cName);

Reservation reservation = **new** Reservation();

String invoiceNo = MyUtil.*getInvoiceNo*();

reservation.setInvoiceNo(invoiceNo);

reservation.setcId(customer.getcId());

reservation.setAmount(amount);

reservation.setcNumber(cNumber);

*myDao*.saveReservation(reservation);

**for** (**int** i = 0; i < list.size(); i++) {

RoomReservation r = list.get(i);

*myDao*.saveRoomReservation(r, invoiceNo);

**for**(**int** j=0;j<r.getBtype().size();j++){

String b = r.getBtype().get(j);

*myDao*.saveRresvBreakfast(r.getHotelId(), b, r.getRoomNo(), r.getInDate(), r.getBamount().get(j));

}

**for**(**int** j=0;j<r.getStype().size();j++){

String s = r.getStype().get(j);

*myDao*.saveRresvService(r.getHotelId(), s, r.getRoomNo(), r.getInDate());

}

}

}

Reservation2.java

**protected** **void** service(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=UTF-8");

Integer hotelId = Integer.*parseInt*(request.getParameter("hotelId"));

MyService myService = **new** MyService();

Hotel hotel = myService.getHotelDetail(hotelId);

request.getSession().setAttribute("hotel", hotel);

request.setAttribute("recommendation", myService.getRecommendation());

request.getRequestDispatcher("/WEB-INF/MyPage/reservation2.jsp").forward(request, response);

}

}

## JSP and JS (Partial)

Part of reservation2.jsp

<script type=*"text/javascript"*>

**var** btypeArray = **new** Array();

**function** checkReservation() {

**var** capacity = $("#capacity").val();

**var** sum =0;

**var** bamount = "";

**for**(**var** i=0;i<btypeArray.length;i++){

**var** number = btypeArray[i].split(":")[1] ;

bamount +=number+" ";

sum =sum+parseInt(number);

}

$("#bamount").val(bamount);

**if** (sum==0) {

alert("Breakfast cannot be null");

**return**;

}

**if**(sum>capacity){

alert("No. of breakfast cannot more than guest number");

**return**;

}

**var** inDate = document.getElementById("inDate");

**if**(inDate.value!=""){

**if**(!checkDate(inDate.value)){

**return**;

}

}

**else**{

alert("Check in date cannot be null");

**return**;

}

**var** outDate = document.getElementById("outDate");

**if**(outDate.value!=""){

**if**(!checkDate(outDate.value)){

**return**;

}

}

**else**{

alert("Check out date cannot be null");

**return**;

}

$("#btype").val(btypeArray.join());

reservation\_form.submit();

}

**function** checkDate(DateStr) {

**return** **true**;

}

**function** add(){

**var** btype = $('#typeSelected option:selected').val();

**var** value = $('#bbamount').val();

**if**(value == undefined ||value==""){

value=1;

}

btypeArray.push(btype+":"+value);

$("#addB").after("<tr id='addTr'><td>"+btype+"</td><td>"+value+"</td></tr>");

}

**function** del(){

**if**(btypeArray.length>0){

btypeArray.splice(-1,1);

}

$("#addTr").remove();

}

</script>

Part of manager\_login.jsp

<body>

<h1>Secure Access Login</h1>

<form name=*"login\_form"* action=*"./ManagerLogin"* method=*"post"*>

<table>

<tr>

<td><span class=*"require"*>\*</span>Username: </td>

<td><input type=*"text"* id=*"name"* name=*"name"* size=*"30"* value=*"*${requestScope.name}*"*

maxLength=*"30"* /></td>

<td><font color=*"red"*>${requestScope.userError}</font></td>

</tr>

<tr>

<td><span class=*"require"*>\*</span>Password: </td>

<td><input type=*"password"* id=*"password"* name=*"password"*

size=*"30"* maxlength=*"30"* /></td>

<td><font color=*"red"*>${requestScope.passwordError}</font></td>

</tr>

<tr>

<td></td>

<td><input type=*"button"* value=*"Login"* onclick="checkLogin()" />&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input type=*"submit"* value=*"Register"* name=*"mregister"* /></td>

</tr>

</table>

</form>

<script type=*"text/javascript"*>

**function** checkLogin() {

**var** name = document.getElementById("name");

**if** (name.value == "") {

alert("Please enter a valid username");

name.focus();

**return**;

}

**var** password = document.getElementById("password");

**if**(password.value == ""){

alert("Please enter your password");

password.focus();

**return**;

}

login\_form.submit();

}

</script>

</body>