Ticketing Service

by

Muhammad Faraz Rafi

Table of Contents

List of Figures

System Design

The Ticket Service application is an enterprise Java application written using Spring MVC, Hibernate and Velocity frameworks. For the sake of time and resources HSQL In-Memory database is used at the back-end that allows flexibility and agility to get an application up and running quickly. Since there is a web application interface, apache tomcat 6.0.44 has been used to deploy/test/execute this application. However, the interface methods are implemented in a manner that independent jUnit test classes can test the application functionality. Velocity has been used for ease of variables access in the models.

Building & Execution

The application is maven enabled and can be easily built using maven tool either from the command line or from within Eclipse IDE. If tests are enabled, all existing jUnit tests will be executed upon building and will display the results as shown in Figure below:

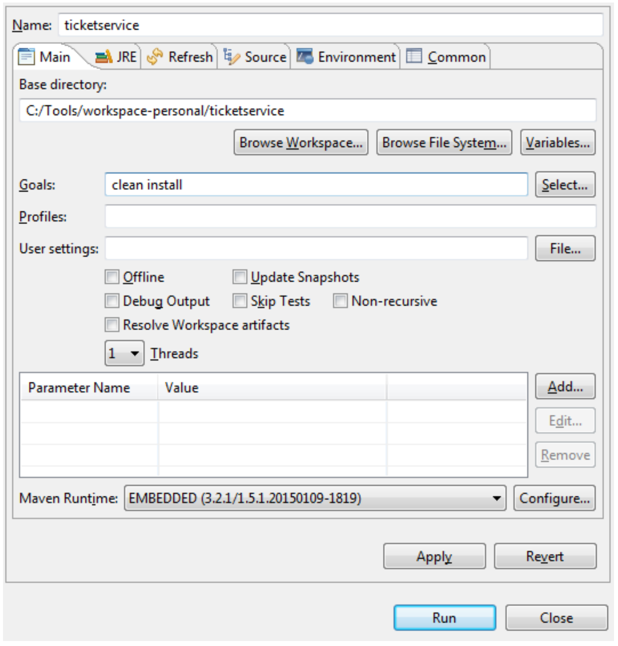
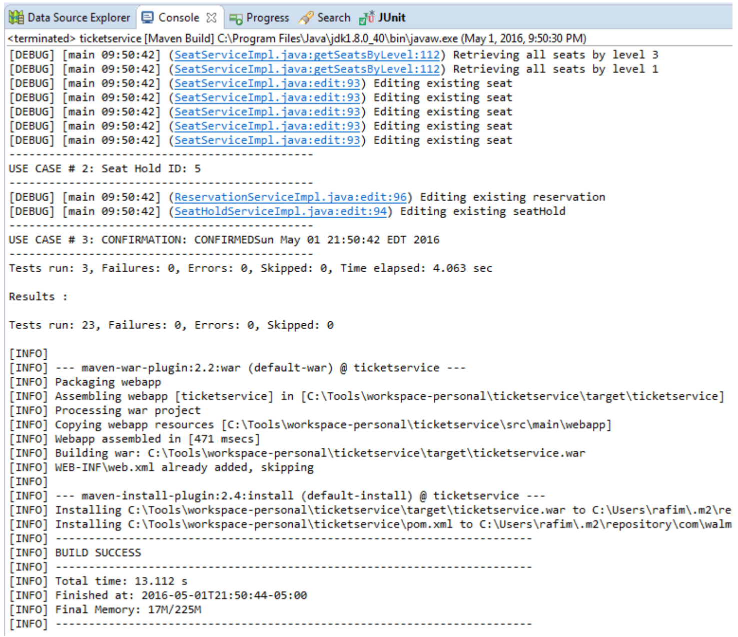

Name: ticketservice
Base directory: C:/Tools/workspace-personal/ticketservice
Goals: clean install
Profiles:
User settings:
Threads: 1
Maven Runtime: EMBEDDED (3.2.1/1.5.1.20150109-1819)

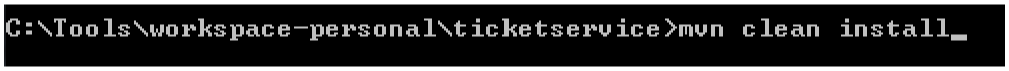
Figure 1: Maven Build



```
<terminated> ticketservice [Maven Build] C:\Program Files\Java\jdk1.8.0_40\bin\javaw.exe (May 1, 2016, 9:50:30 PM)
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:getSeatsByLevel:112) Retrieving all seats by level 3
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:getSeatsByLevel:112) Retrieving all seats by level 1
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 09:50:42] (SeatServiceImpl.java:edit:93) Editing existing seat
----------------------------------------------
USE CASE # 2: Seat Hold ID: 5
----------------------------------------------
[DEBUG] [main 09:50:42] (ReservationServiceImpl.java:edit:96) Editing existing reservation
[DEBUG] [main 09:50:42] (SeatHoldServiceImpl.java:edit:94) Editing existing seatHold
----------------------------------------------
USE CASE # 3: CONFIRMATION: CONFIRMEDSun May 01 21:50:42 EDT 2016
----------------------------------------------
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.063 sec

Results :

Tests run: 23, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ ticketservice ---
[INFO] Packaging webapp
[INFO] Assembling webapp [ticketservice] in [C:\Tools\workspace-personal\ticketservice\target\ticketservice]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Tools\workspace-personal\ticketservice\src\main\webapp]
[INFO] Webapp assembled in [471 msecs]
[INFO] Building war: C:\Tools\workspace-personal\ticketservice\target\ticketservice.war
[INFO] WEB-INF\web.xml already added, skipping
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ ticketservice ---
[INFO] Installing C:\Tools\workspace-personal\ticketservice\target\ticketservice.war to C:\Users\rafim\.m2\re
[INFO] Installing C:\Tools\workspace-personal\ticketservice\pom.xml to C:\Users\rafim\.m2\repository\com\walm
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 13.112 s
[INFO] Finished at: 2016-05-01T21:50:44-05:00
[INFO] Final Memory: 17M/225M
[INFO] ------------------------------------------------------------------------
```

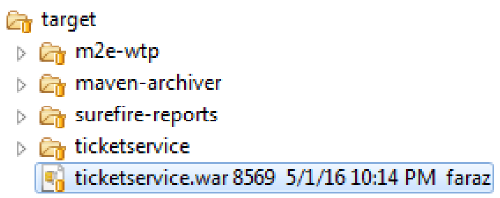
Figure 2: All Tests must pass

Use the following command to build this application through maven build tool:



```
C:\Tools\workspace-personal\ticketservice>mvn clean install_
```

Once the application is built successfully, it generates a “ticketservice.war” file under the target folder. This war file can be deployed into the webapps folder of tomcat usually located under “\apache-tomcat-6.0.44\webapps”. Now start the server and the application should be up and running.

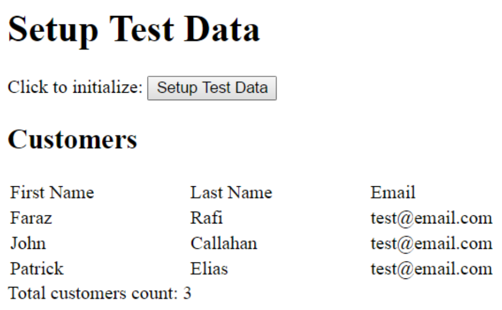


```
target
  ▷ m2e-wtp
  ▷ maven-archiver
  ▷ surefire-reports
  ▷ ticketservice
    ticketservice.war 8569  5/1/16 10:14 PM  faraz
```

Web Interface

This application has been implemented as a web application for easy use/testing. The following pages have been designed to implement the three user stories:

a. http://localhost:8080/ticketservice/walmart/initialize: This webpage lets you initialize the test data for the application. This test data includes a few customer user accounts, the complete records for look up "Seat" table so they can be held, reserved. Once you click on the "Setup Test Data", it invokes *initializeCustomers()* method – that initializes the look up values for "Seat" and "Customer" tables.



Setup Test Data

Click to initialize: Setup Test Data

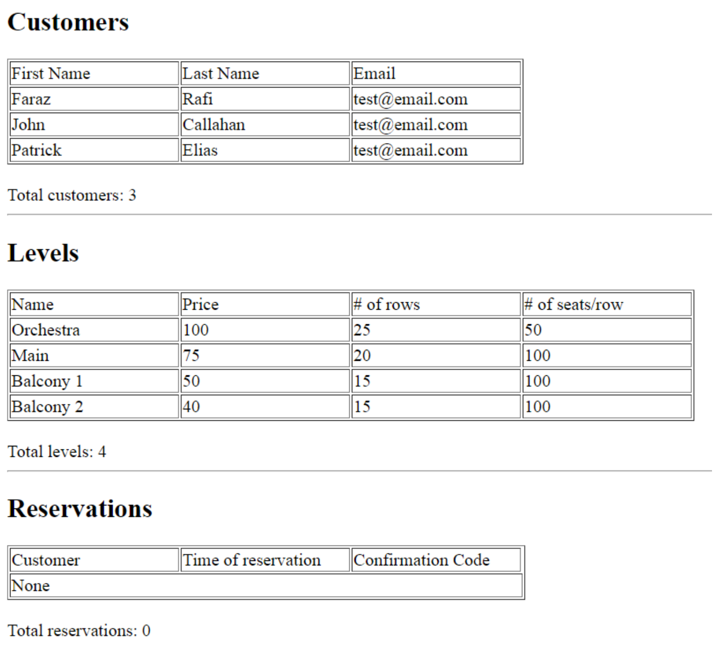
Customers

First Name	Last Name	Email
Faraz	Rafi	test@email.com
John	Callahan	test@email.com
Patrick	Elias	test@email.com

Total customers count: 3

Figure 3: Setup Test Data

b. http://localhost:8080/ticketservice/walmart/testdata: This page lets you view the test data that's been generated as a result of the step a. This view serves as a data view of all current tables and current reservations in progress. You can monitor which seats got reserved/held/available and if there's a new reservation made easily by refreshing this page.



Customers

First Name	Last Name	Email
Faraz	Rafi	test@email.com
John	Callahan	test@email.com
Patrick	Elias	test@email.com

Total customers: 3

Levels

Name	Price	# of rows	# of seats/row
Orchestra	100	25	50
Main	75	20	100
Balcony 1	50	15	100
Balcony 2	40	15	100

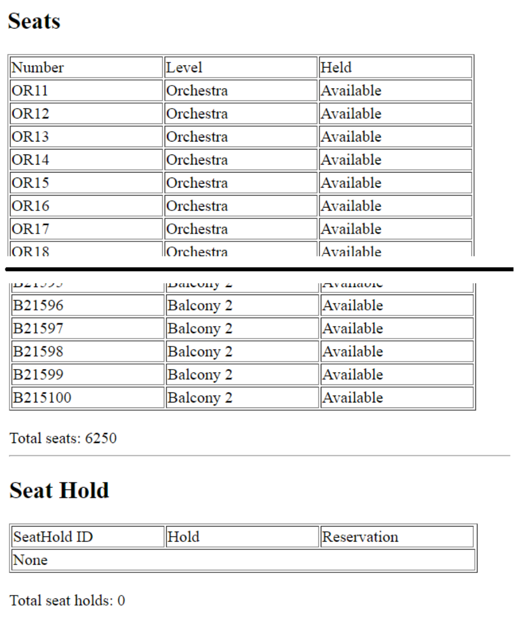
Total levels: 4

Reservations

Customer	Time of reservation	Confirmation Code
None		

Total reservations: 0

Figure 4: Test Data view 1



Seats

Number	Level	Held
OR11	Orchestra	Available
OR12	Orchestra	Available
OR13	Orchestra	Available
OR14	Orchestra	Available
OR15	Orchestra	Available
OR16	Orchestra	Available
OR17	Orchestra	Available
OR18	Orchestra	Available
B21595	Balcony 2	Available
B21596	Balcony 2	Available
B21597	Balcony 2	Available
B21598	Balcony 2	Available
B21599	Balcony 2	Available
B215100	Balcony 2	Available

Total seats: 6250

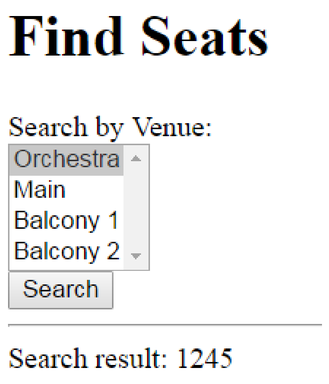
Seat Hold

SeatHold ID	Hold	Reservation
None		

Total seat holds: 0

Figure 5: Test Data view 2

c. http://localhost:8080/ticketservice/walmart/numseatsavailable: This page lets you test the **numseatsavailable** method. You can view the total number of seats that are available only. This view updates the search results if seats get held/reserved. The following figure shows you how 5 Orchestra seats have been held, as a result the search finds only 1245 Orchestra seats out of total 1250.

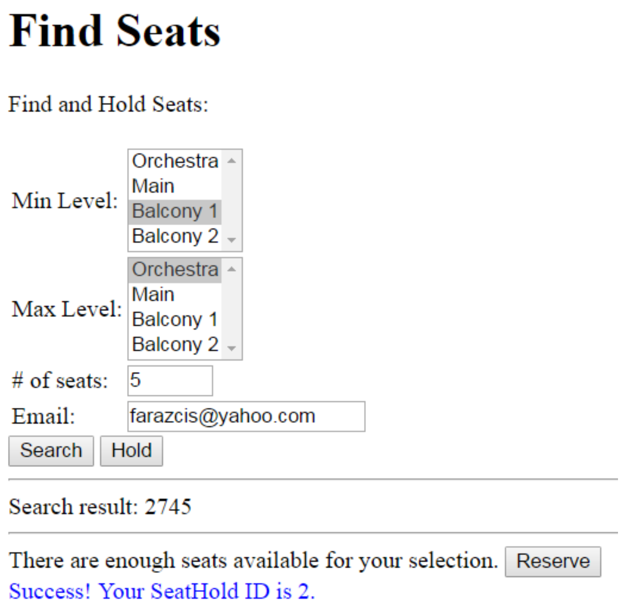

Find Seats

Search by Venue: Orchestra

Search result: 1245

Figure 6: Find Seats

d. http://localhost:8080/ticketservice/walmart/findAndHoldSeats: Once you've verified there are seats available in a particular level. You can find and hold those seats from this page. You can select Min/Max levels. The findAndHoldSeats method makes sure it holds the seats in your "Max level" preference first. If it doesn't find any, then it'll pick from the "Min level" preference. When you click "Hold" button, it generates a Seat Hold Id for you as shown below and generates a SeatHold Object returning the Id.


Find Seats

Find and Hold Seats:

Min Level: Balcony 1

Max Level: Orchestra

# of seats: 5

Email: farazcis@yahoo.com

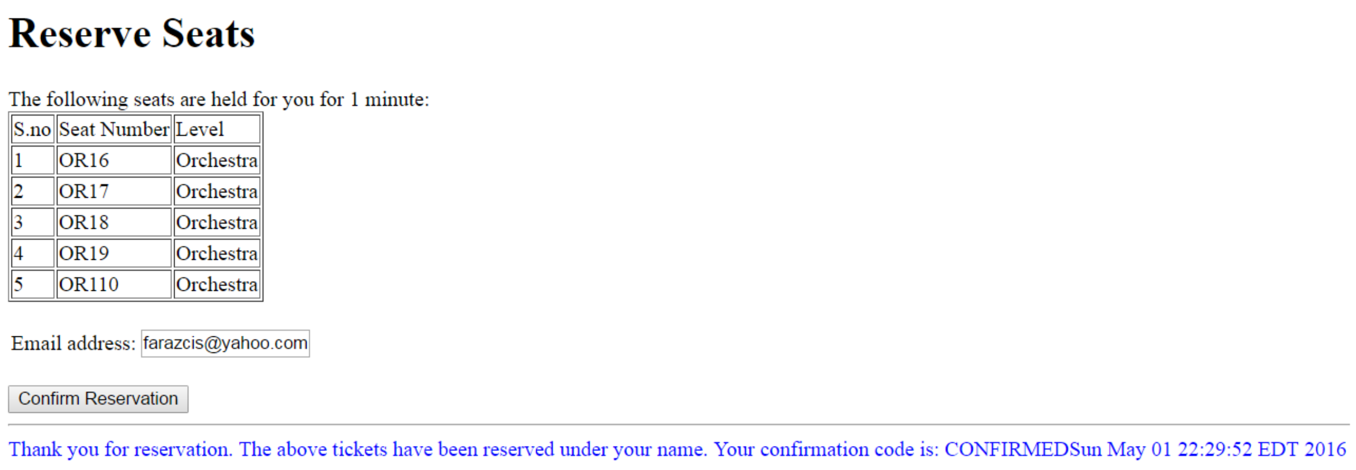
Search result: 2745

There are enough seats available for your selection.

Success! Your SeatHold ID is 2.

Figure 7: Hold Seats

e. http://localhost:8080/ticketservice/walmart/reserve: Once your desired seats are Held. You can reserve them by clicking on the "Reserve" button. The next page displays your assigned seats before you confirm your reservation by entering your email address. Once you confirm your reservation, a success message is displayed with you confirmation code.



Reserve Seats

The following seats are held for you for 1 minute:

S.no	Seat Number	Level
1	OR16	Orchestra
2	OR17	Orchestra
3	OR18	Orchestra
4	OR19	Orchestra
5	OR110	Orchestra

Email address: farazcis@yahoo.com

Confirm Reservation

Thank you for reservation. The above tickets have been reserved under your name. Your confirmation code is: CONFIRMEDSun May 01 22:29:52 EDT 2016

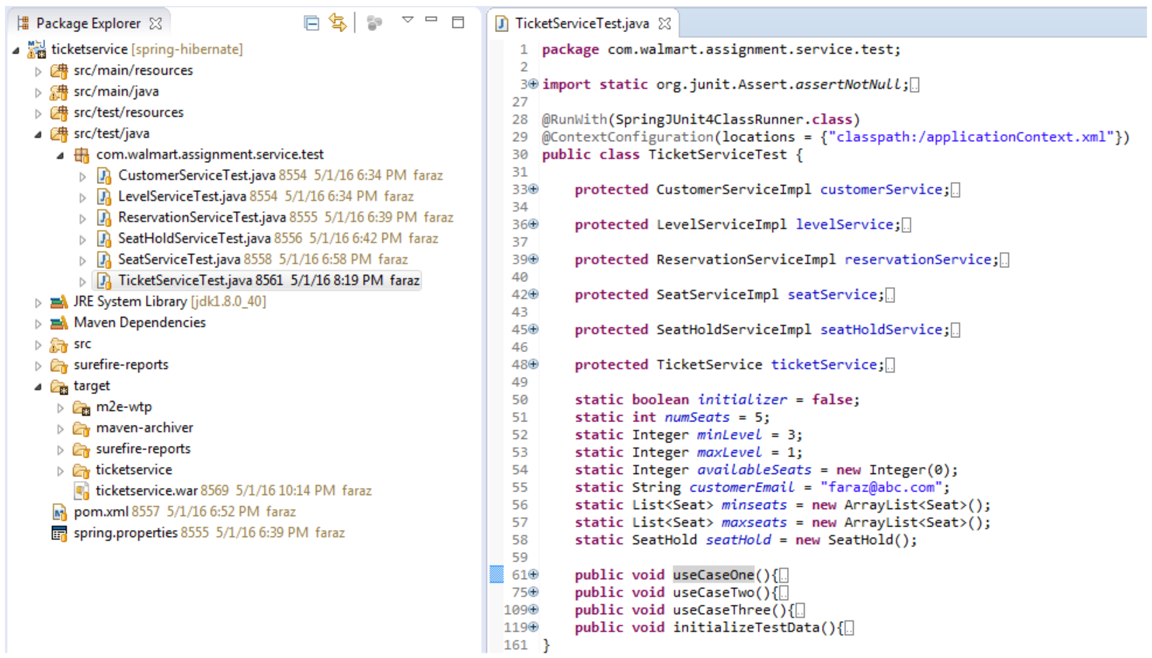
Figure 8: Reserve Seats

jUnit Test Interface

The web based functionality can also be tested/verified via jUnit test cases. A sample input data is provided within the jUnit test cases. Every time you build the application using maven with tests enabled, these sets of jUnit test cases will be executed that:

a. Initialize test data in HSQLDB
b. Execute test case 1
c. Execute test case 2
d. Execute test case 3

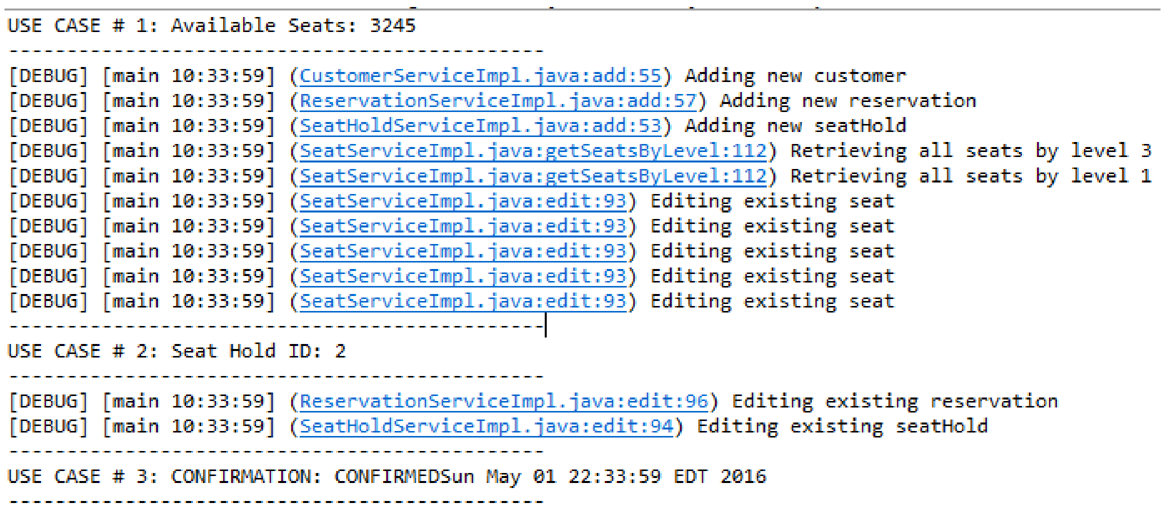
The above steps are explained in the figures below:



```
Package Explorer
ticketservice [spring-hibernate]
  src/main/resources
  src/main/java
  src/test/resources
  src/test/java
    com.walmart.assignment.service.test
      CustomerServiceTest.java 8554  5/1/16 6:34 PM  faraz
      LevelServiceTest.java 8554  5/1/16 6:34 PM  faraz
      ReservationServiceTest.java 8555  5/1/16 6:39 PM  faraz
      SeatHoldServiceTest.java 8556  5/1/16 6:42 PM  faraz
      SeatServiceTest.java 8558  5/1/16 6:58 PM  faraz
      TicketServiceTest.java 8561  5/1/16 8:19 PM  faraz
  JRE System Library [jdk1.8.0_40]
  Maven Dependencies
  src
  surefire-reports
  target
    m2e-wtp
    maven-archiver
    surefire-reports
    ticketservice
    ticketservice.war 8569  5/1/16 10:14 PM  faraz
  pom.xml 8557  5/1/16 6:52 PM  faraz
  spring.properties 8555  5/1/16 6:39 PM  faraz
```

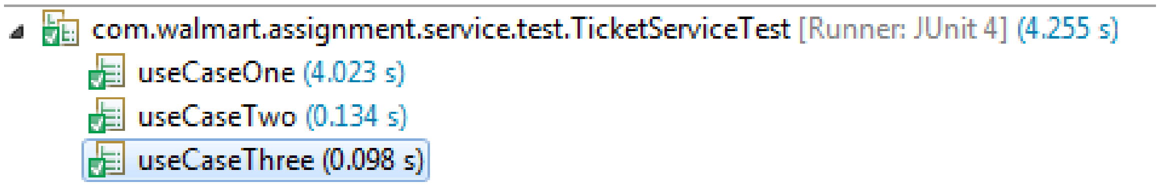
```java
TicketServiceTest.java

package com.walmart.assignment.service.test;

import static org.junit.Assert.assertNull;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath:/applicationContext.xml"})
public class TicketServiceTest {

    protected CustomerServiceImpl customerService;

    protected LevelServiceImpl levelService;

    protected ReservationServiceImpl reservationService;

    protected SeatServiceImpl seatService;

    protected SeatHoldServiceImpl seatHoldService;

    protected TicketService ticketService;

    static boolean initializer = false;
    static int numSeats = 5;
    static Integer minLevel = 3;
    static Integer maxLevel = 1;
    static Integer availableSeats = new Integer(0);
    static String customerEmail = "faraz@abc.com";
    static List<Seat> minseats = new ArrayList<Seat>();
    static List<Seat> maxseats = new ArrayList<Seat>();
    static SeatHold seatHold = new SeatHold();

    public void useCaseOne(){
    public void useCaseTwo(){
    public void useCaseThree(){
    public void initializeTestData(){
}
```

Figure 9: jUnit Test Package



```
USE CASE # 1: Available Seats: 3245
---------------------------------------------
[DEBUG] [main 10:33:59] (CustomerServiceImpl.java:add:55) Adding new customer
[DEBUG] [main 10:33:59] (ReservationServiceImpl.java:add:57) Adding new reservation
[DEBUG] [main 10:33:59] (SeatHoldServiceImpl.java:add:53) Adding new seatHold
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:getSeatsByLevel:112) Retrieving all seats by level 3
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:getSeatsByLevel:112) Retrieving all seats by level 1
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:edit:93) Editing existing seat
[DEBUG] [main 10:33:59] (SeatServiceImpl.java:edit:93) Editing existing seat
---------------------------------------------
USE CASE # 2: Seat Hold ID: 2
---------------------------------------------
[DEBUG] [main 10:33:59] (ReservationServiceImpl.java:edit:96) Editing existing reservation
[DEBUG] [main 10:33:59] (SeatHoldServiceImpl.java:edit:94) Editing existing seatHold
---------------------------------------------
USE CASE # 3: CONFIRMATION: CONFIRMEDSun May 01 22:33:59 EDT 2016
---------------------------------------------
```

Figure 10: jUnit Test Results



```
com.walmart.assignment.service.test.TicketServiceTest [Runner: JUnit 4] (4.255 s)
    useCaseOne (4.023 s)
    useCaseTwo (0.134 s)
    useCaseThree (0.098 s)
```

Figure 11: jUnit Tests Successful

Use Cases

Figure 12 explains the three major use cases as required by the Homework assignment:

a. Finding available seats by the seating level
b. Finding and holding best available seats
c. Reserving a specific group of held seats and returning reservation confirmation

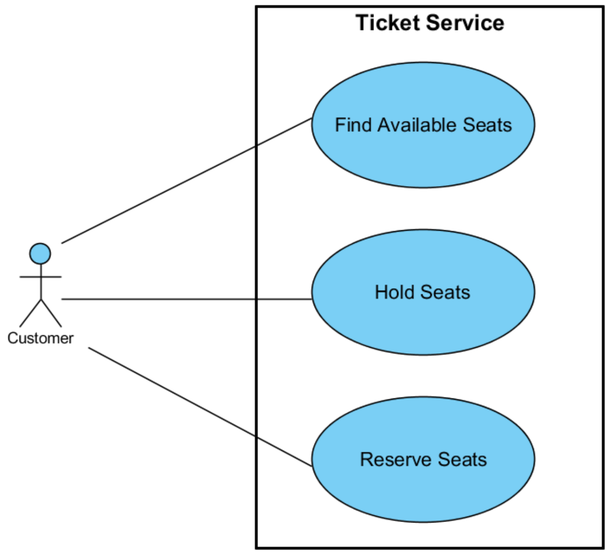


Figure 12: Use Case Diagram

Entity Model

Figure 13 shows the entity diagram identifying the major entities needed for Ticket Service application to work effectively. These entities/tables are explained as follows:

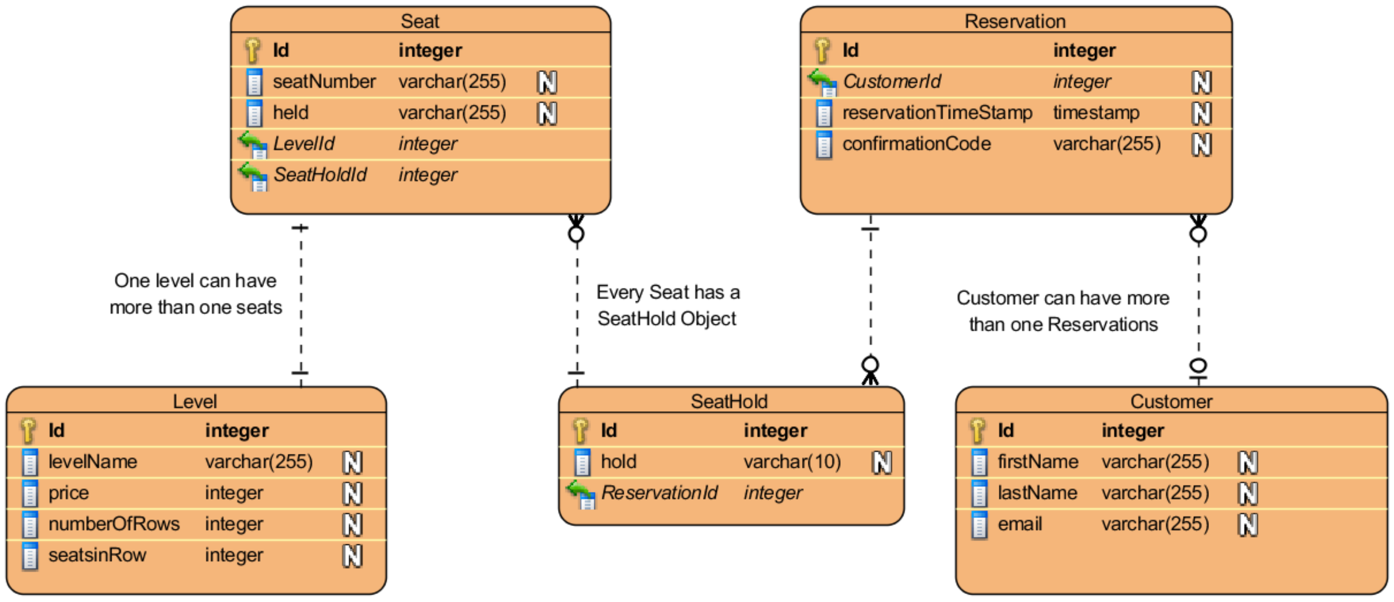


Figure 13: Entity Diagram

a. **Level:** A level describes the various levels of the auditorium. This includes Orchestra, Main, Balcony 1 and Balcony 2. This table also holds all the fields provided in the assignment including price, rows and seats/row.

Name	DataType	Constraints	Nullable
Id	integer(10)	PK	No
levelName	varchar(255)		Yes
price	integer(10)		Yes
numberOfRows	integer(10)		Yes
seatsinRow	integer(10)		Yes

b. **Seat:** This is a look up table that stores seat number, held (“available”, “held”, “reserved”) and is associated with one level. This table has a foreign key association with Level and SeatHold tables.

Name	DataType	Constraints	Nullable
Id	integer	PK	No
seatNumber	varchar(255)		Yes
held	varchar(255)		Yes
LevelId	integer(10)	FK (Level.Id)	No
SeatHoldId	integer(10)	FK (SeatHold.Id)	No

c. **Customer:** This table stores a customer information such as email, first name (optional), last name (optional). This record is needed to initiate a reservation process.

Name	DataType	Constraints	Nullable
Id	integer(10)	PK	No
firstName	varchar(255)		Yes
lastName	varchar(255)		Yes
email	varchar(255)		Yes

d. **Reservation:** A reservation record is created as soon a seat hold is attempted. This table is later updated to store the generated confirmation code for the particular customer. A Many to One relationship exists with Customer table as one customer can hold one/many reservations.

Name	DataType	Constraints	Nullable
Id	integer(10)	PK	No
CustomerId	integer(10)	FK (Customer.Id)	Yes
reservationTimeStamp	timestamp		Yes
confirmationCode	varchar(255)		Yes

e. **SeatHold:** This table is the final outcome that stores the "Hold" status for a given reservation. Once the user holds it, the "Hold" column is set to "Held" but once, he confirms reservation, its set to "Reserved" against a particular reservation. This table has an association with the reservation table.

Name	DataType	Constraints	Nullable
Id	integer(10)	PK	No
hold	varchar(10)		Yes
ReservationId	integer(10)	FK (Reservation.Id)	No

Class Diagram

Figure 3 shows the class diagram for the given ORM objects. Javax.Persistence.Entity objects were generated from the above data model.

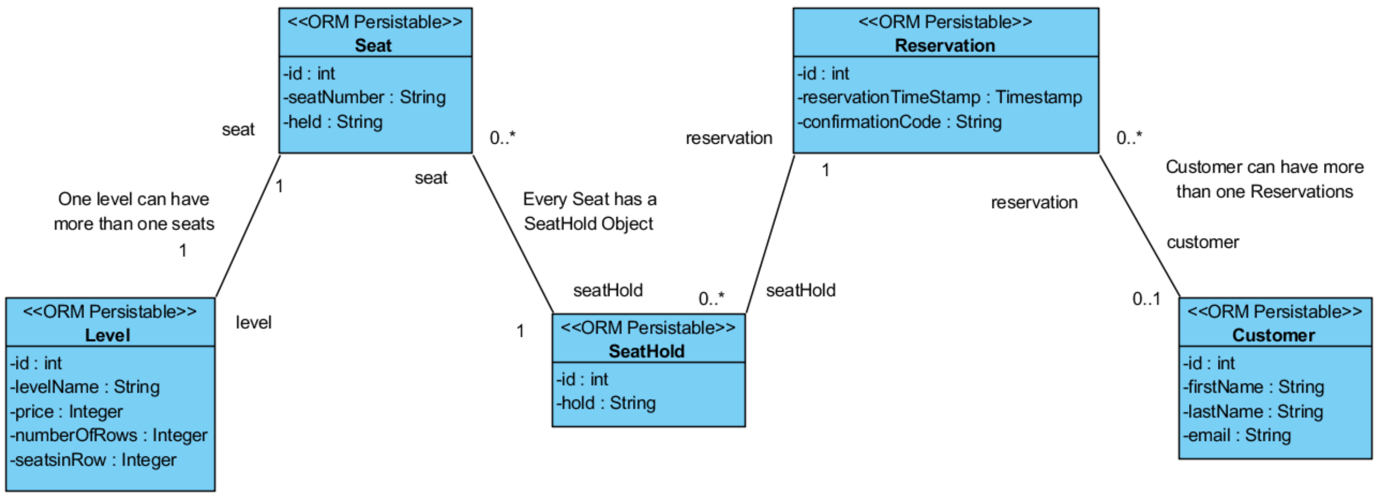


Figure 14: Class Diagram

Sequence Diagram

The following section explains the implementation of each user story with the help of its sequence diagram.

Use Case 1:

Figure 15 depicts the implementation of method **numseatsavailable**. This method takes the “level” as user input and returns the total number of seats available in that/those level(s).

Use Case 2:

Figure 16 depicts the implementation of method **findAndHoldSeats**. This method takes the number of seats to hold, minimum desired level, maximum desired level and the customer email to hold the seats with. This method queries creates a few records including a new customer record, a new reservation record and a new seat hold record. The seat hold record is returned to the callee.

Use Case 3:

Figure 17 depicts the implementation of method **reserveSeats.** This method takes the SeatHoldId object that was returned in Use Case 2. This Id holds the information of the reservation belonging to a customer. If the user confirms the reservation, the SeatHold Object is updated to be reserved and the reservation is finalized generating a confirmation code for the user. This confirmation code is the return parameter to the callee.

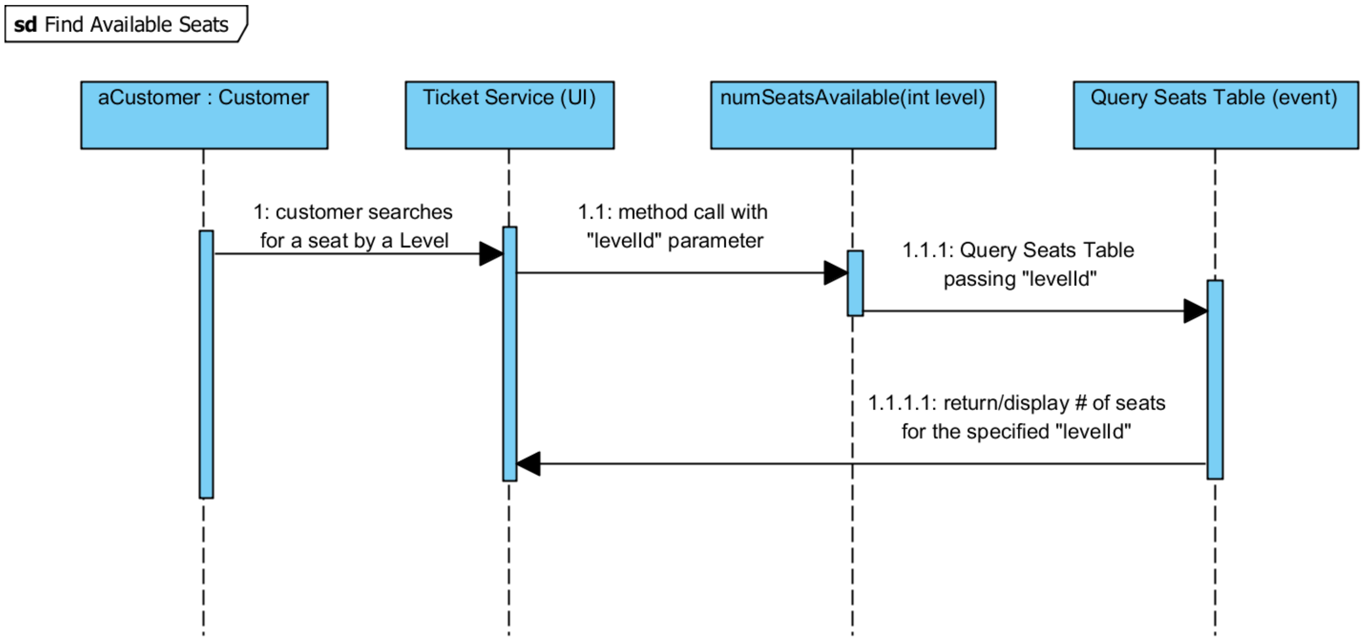

sd Find Available Seats
aCustomer : Customer
Ticket Service (UI)
numSeatsAvailable(int level)
Query Seats Table (event)
1: customer searches for a seat by a Level
1.1: method call with "levelId" parameter
1.1.1: Query Seats Table passing "levelId"
1.1.1.1: return/display # of seats for the specified "levelId"

Figure 15: Sequence Diagram for Find Available Seats

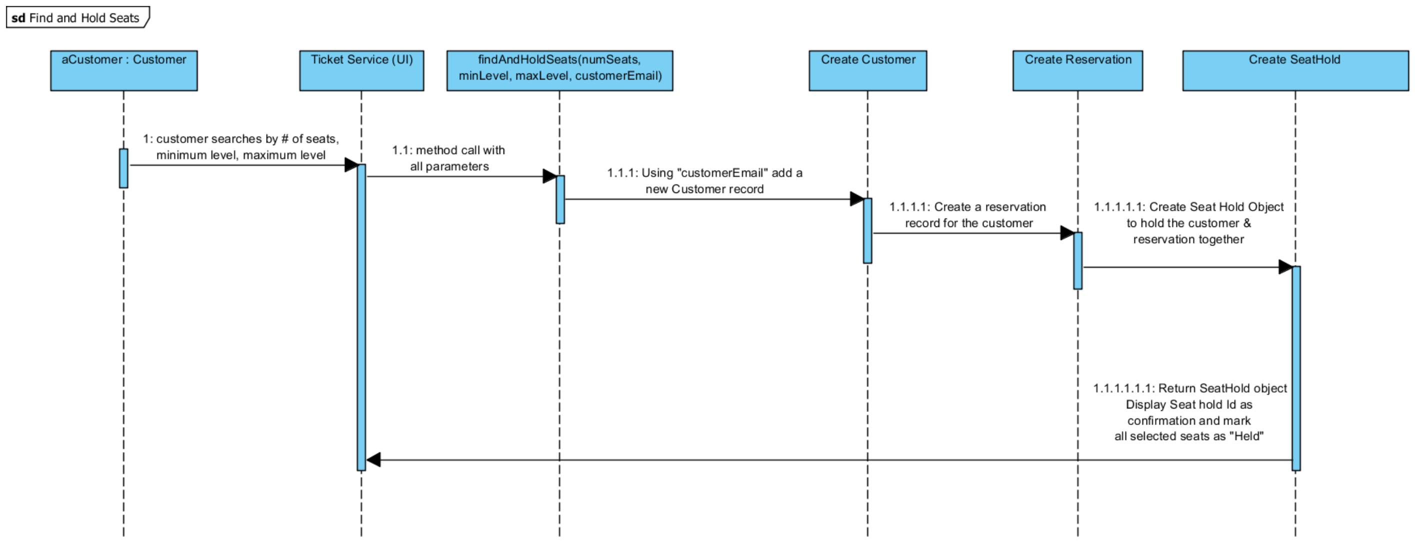

sd Find and Hold Seats
aCustomer : Customer
Ticket Service (UI)
findAndHoldSeats(numSeats, minLevel, maxLevel, customerEmail)
Create Customer
Create Reservation
Create SeatHold
1: customer searches by # of seats, minimum level, maximum level
1.1: method call with all parameters
1.1.1: Using "customerEmail" add a new Customer record
1.1.1.1: Create a reservation record for the customer
1.1.1.1.1: Create Seat Hold Object to hold the customer & reservation together
1.1.1.1.1.1: Return SeatHold object Display Seat hold Id as confirmation and mark all selected seats as "Held"

Figure 16: Sequence Diagram for Find/Hold Seats

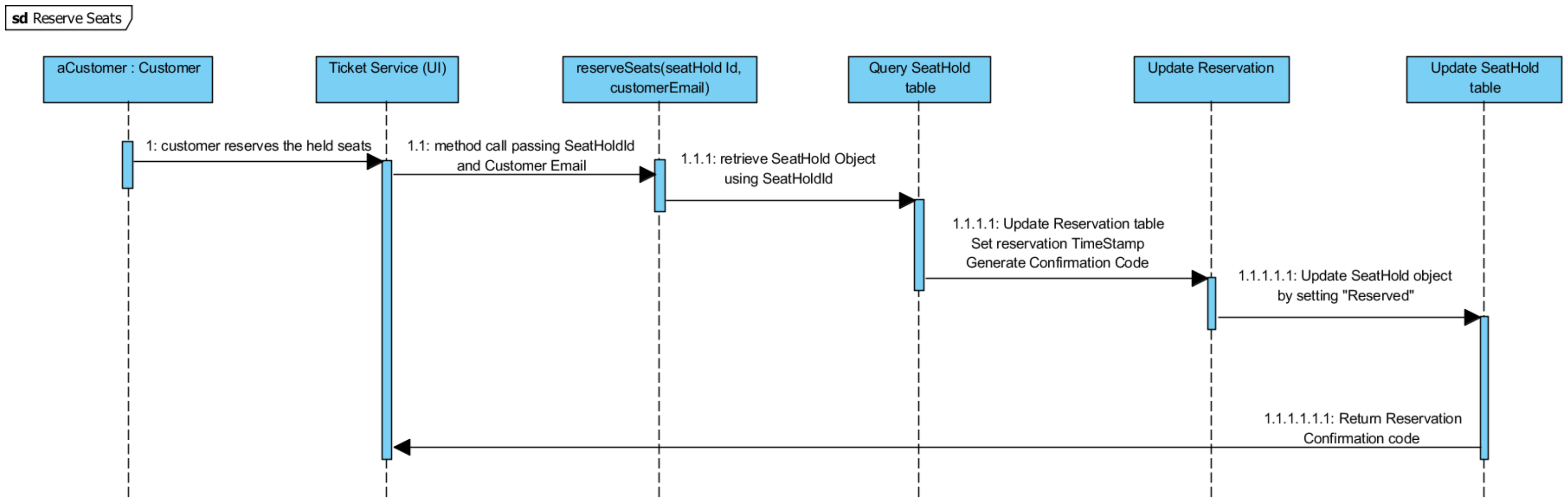

sd Reserve Seats
aCustomer : Customer
Ticket Service (UI)
reserveSeats(seatHold Id, customerEmail)
Query SeatHold table
Update Reservation
Update SeatHold table
1: customer reserves the held seats
1.1: method call passing SeatHoldId and Customer Email
1.1.1: retrieve SeatHold Object using SeatHoldId
1.1.1.1: Update Reservation table Set reservation TimeStamp Generate Confirmation Code
1.1.1.1.1: Update SeatHold object by setting "Reserved"
1.1.1.1.1.1: Return Reservation Confirmation code

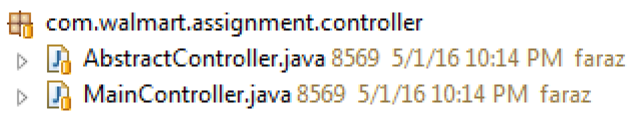
Figure 17: Sequence Diagram for Seat Reservation

Package diagrams

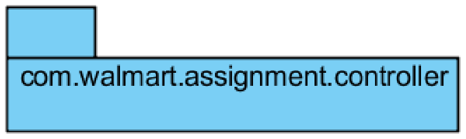
The following packages are created in Ticket Service Application:

Controller

This package contains one MainController that contains all GET/POST methods for all the web pages. This controller extends an AbstractController that houses the session variables all model object instantiations and service interface instantiations.

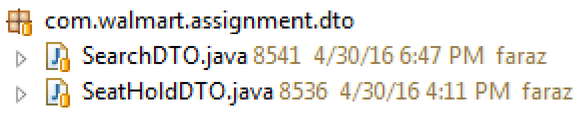


```
com.walmart.assignment.controller
  AbstractController.java 8569  5/1/16 10:14 PM  faraz
  MainController.java 8569  5/1/16 10:14 PM  faraz
```



DTO

This package contains DTO objects to hold user data that's used to query tables and store information.

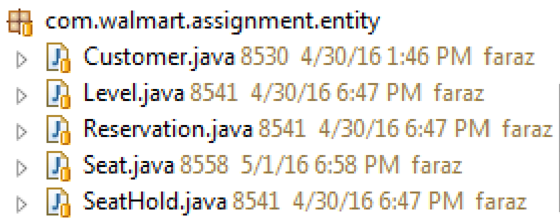


```
com.walmart.assignment.dto
  SearchDTO.java 8541  4/30/16 6:47 PM  faraz
  SeatHoldDTO.java 8536  4/30/16 4:11 PM  faraz
```

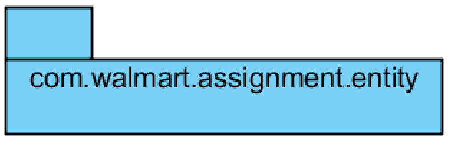
com.walmart.assignment.dto

Entity

This package contains the ORM classes for all tables.



```
com.walmart.assignment.entity
  Customer.java 8530  4/30/16 1:46 PM  faraz
  Level.java 8541  4/30/16 6:47 PM  faraz
  Reservation.java 8541  4/30/16 6:47 PM  faraz
  Seat.java 8558  5/1/16 6:58 PM  faraz
  SeatHold.java 8541  4/30/16 6:47 PM  faraz
```

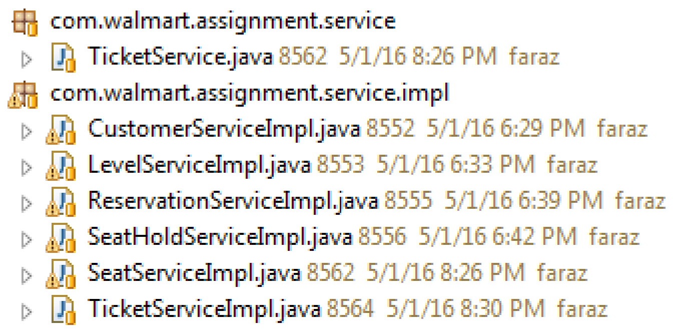


Service Interface

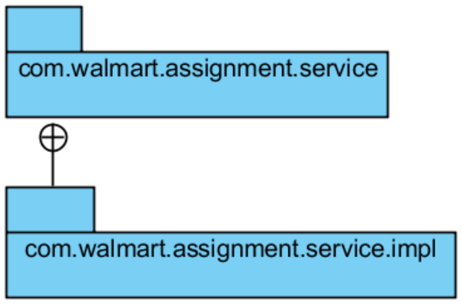
This package contains the interface definition for the TicketService that has the three method signatures.

Service Implementation

This package contains the service implementations that interact with the database. It also contains the service class that implements the TicketService interface.

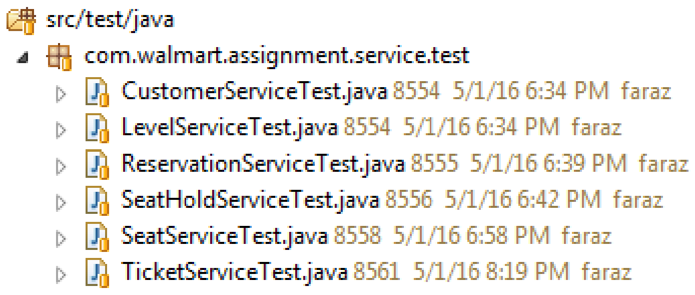


```
com.walmart.assignment.service
  TicketService.java 8562  5/1/16 8:26 PM  faraz
com.walmart.assignment.service.impl
  CustomerServiceImpl.java 8552  5/1/16 6:29 PM  faraz
  LevelServiceImpl.java 8553  5/1/16 6:33 PM  faraz
  ReservationServiceImpl.java 8555  5/1/16 6:39 PM  faraz
  SeatHoldServiceImpl.java 8556  5/1/16 6:42 PM  faraz
  SeatServiceImpl.java 8562  5/1/16 8:26 PM  faraz
  TicketServiceImpl.java 8564  5/1/16 8:30 PM  faraz
```

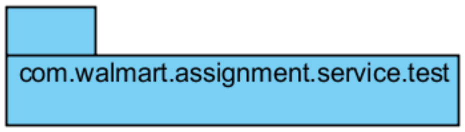

com.walmart.assignment.service
com.walmart.assignment.service.impl

jUnit Test Cases

This package contains all junit test cases that test all the service class/interface implementations. The jUnit "TicketService.java" class tests the complete application by executing individual test cases for all three use cases explained in this document using sample test input data.



```
src/test/java
  com.walmart.assignment.service.test
    CustomerServiceTest.java 8554  5/1/16 6:34 PM  faraz
    LevelServiceTest.java 8554  5/1/16 6:34 PM  faraz
    ReservationServiceTest.java 8555  5/1/16 6:39 PM  faraz
    SeatHoldServiceTest.java 8556  5/1/16 6:42 PM  faraz
    SeatServiceTest.java 8558  5/1/16 6:58 PM  faraz
    TicketServiceTest.java 8561  5/1/16 8:19 PM  faraz
```



Limitations

Due to time limit and scheduling constraints – the ticket expiry feature didn't get completed. However, if time permitted – this feature can be added.