|  |
| --- |
| SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY |
| Modern Topics in Information Technology |
| AOP Assignment-3 Report |
|  |
| **S.A.K.G. Samaraweera (IT13040468) D.D.T.M. Gunathilaka (IT13011130)** |
| **4/11/2016** |

|  |
| --- |
| The following report describes the implementation of Assignment 3 solution. The scenario that has been implemented is a simple Train Seat Reservation application on Java language. In this solution SpringAOP is used to implement the logging aspects in the system. |

Table of Contents

[1. Overview 2](#_Toc448261952)

[2. Class Diagram 4](#_Toc448261953)

[3. Sequence Diagram 6](#_Toc448261954)

[4. Implemented Aspects 7](#_Toc448261955)

[4.1. Point Cut 7](#_Toc448261956)

[4.2. Before Advice 7](#_Toc448261957)

[4.3. After Advice 8](#_Toc448261958)

[4.4. After Returning 8](#_Toc448261959)

[4.5. After Throwing 9](#_Toc448261960)

[4.5.1. No Available Seats Exception 9](#_Toc448261961)

[4.5.2. Roll Back Booking Exception 9](#_Toc448261962)

[4.5.3. Validation Exception 9](#_Toc448261963)

[5. Sample Logger Outputs 10](#_Toc448261964)

[5.1. Before, After and After Returning 10](#_Toc448261965)

[5.1.1. Data Access Layer Advices 10](#_Toc448261966)

[5.1.2. Service Layer Advices 12](#_Toc448261967)

[5.2. After Throwing Advice 13](#_Toc448261968)

# Overview

In this application it will describe a scenario of seat reservation system for trains. The respective application is Java based stand alone application powered by PostgresSQL server taken as the data store. Furthermore this solution consists of Logging functions in order to monitor the runtime behavior of its components. These logging functions are implemented using Aspect Oriented Programming (AOP) paradigm in order to reduce the redundancy and complexity of source code. These AOP features being implemented in this solution through SpringAOP framework.

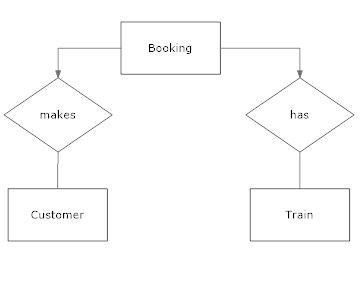


Figure 1‑1.1 Entity relationship diagram for scenario

According to above ER diagram of the system three entities can be identified namely; Customer, Booking and Train. In this solution it is assumed that one booking cannot be made by multiple customers as well as customers cannot reserve seats on different trains using single booking.

Based on above mentioned assumptions the following instance level component diagram being illustrated

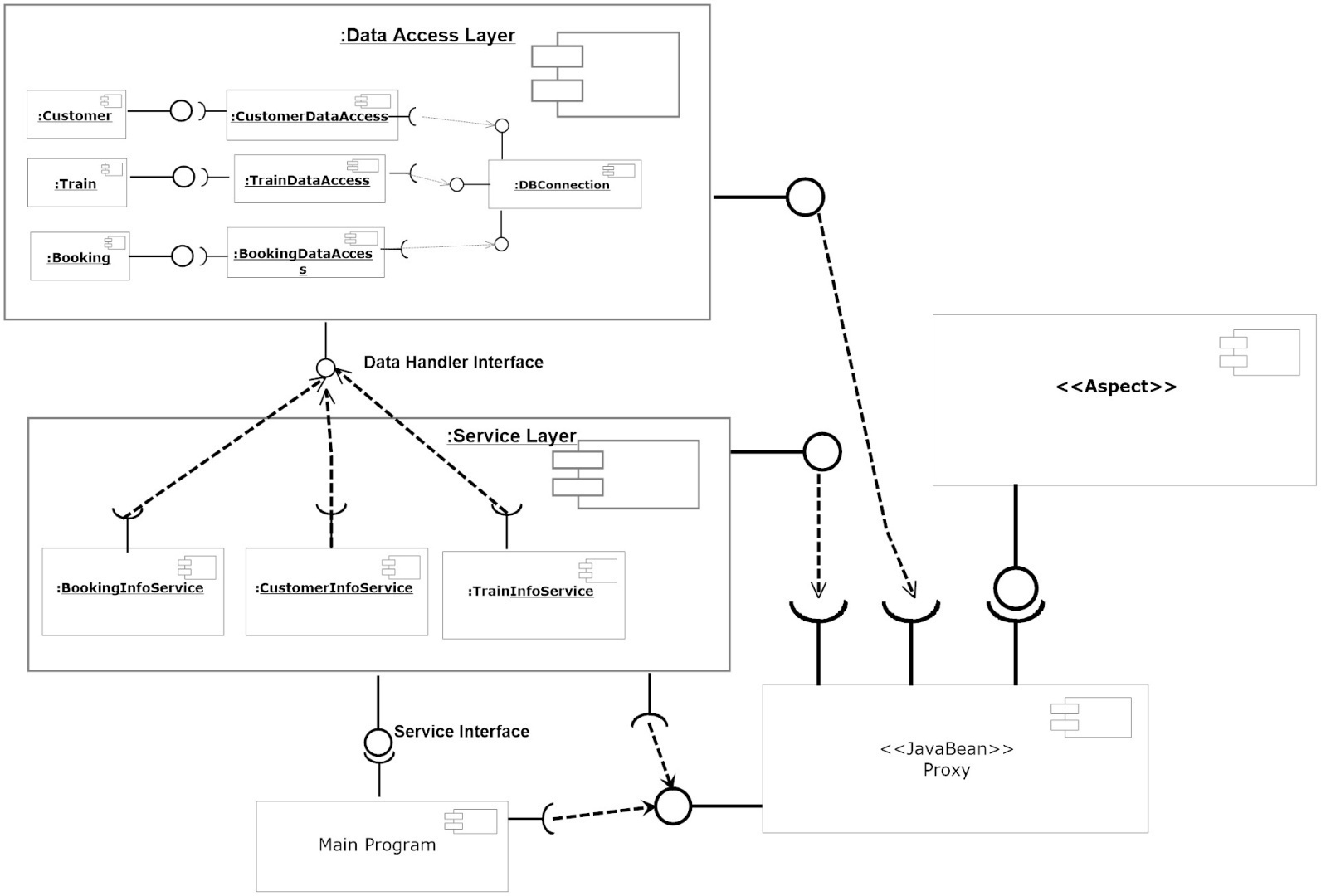


Figure 1.1‑2 Component Diagram for Train Seat Booking System

Since Spring AOP framework acquire the features of Aspect Oriented programming at the runtime rather than mutating the byte code, in here the “config.xml” file of the solution will act as the proxy in runtime. Therefore config.xml file will wrap Service Layer & Data Access Layer objects at runtime. Whenever a function call is made for particular object first proxy gets executed, and then it will invoke all the required advices by considering the point-cuts declared in respective Aspect Layer classes.

# Class Diagram

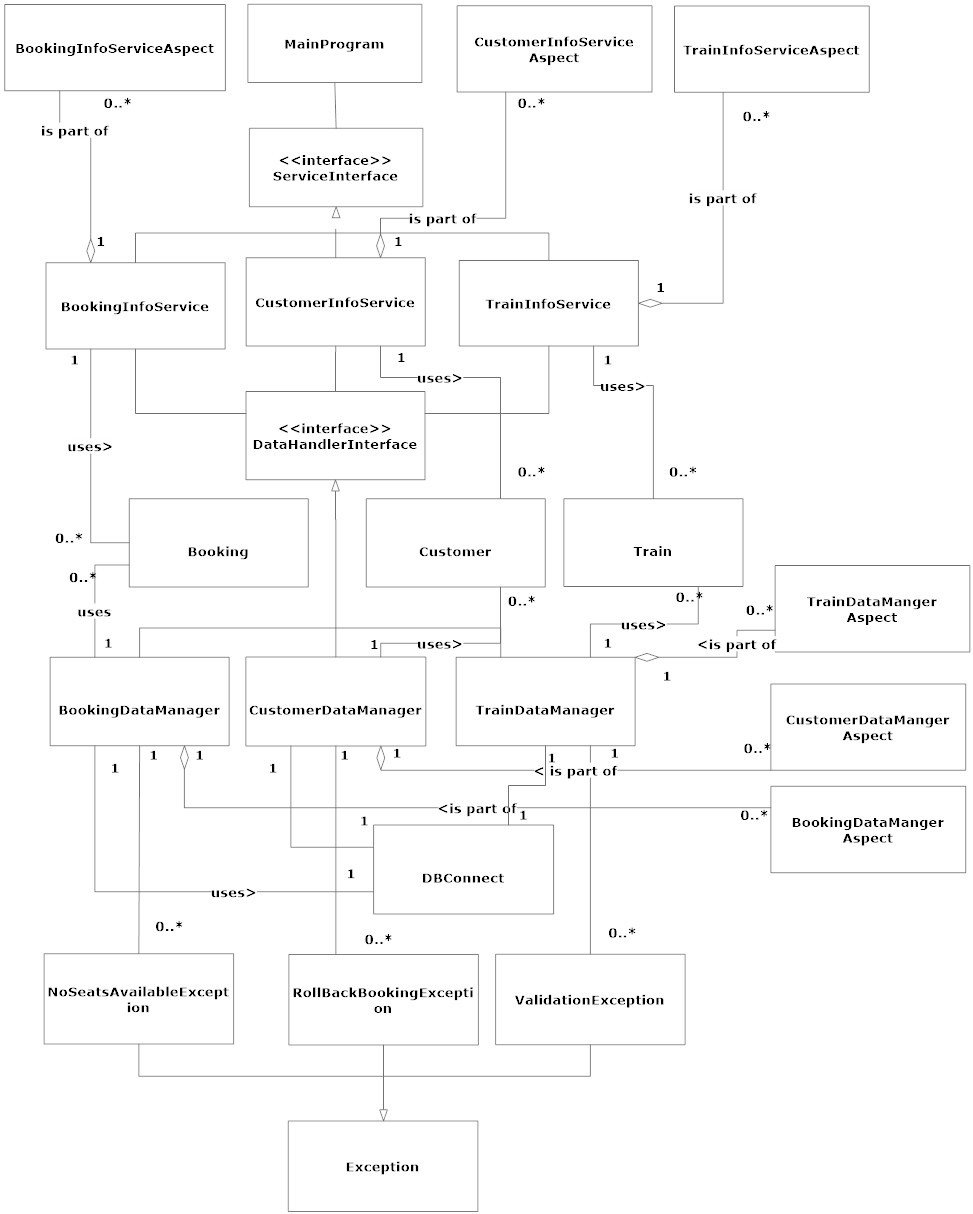


Figure 2 Class diagram for Train Seat Reservation System

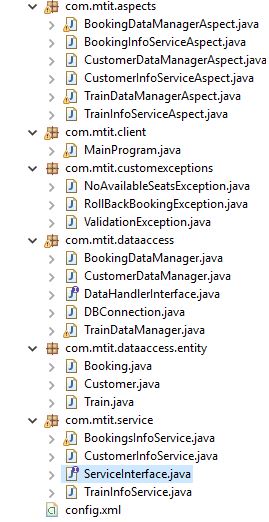


Figure 2.2 Implementation of classes

# Sequence Diagram

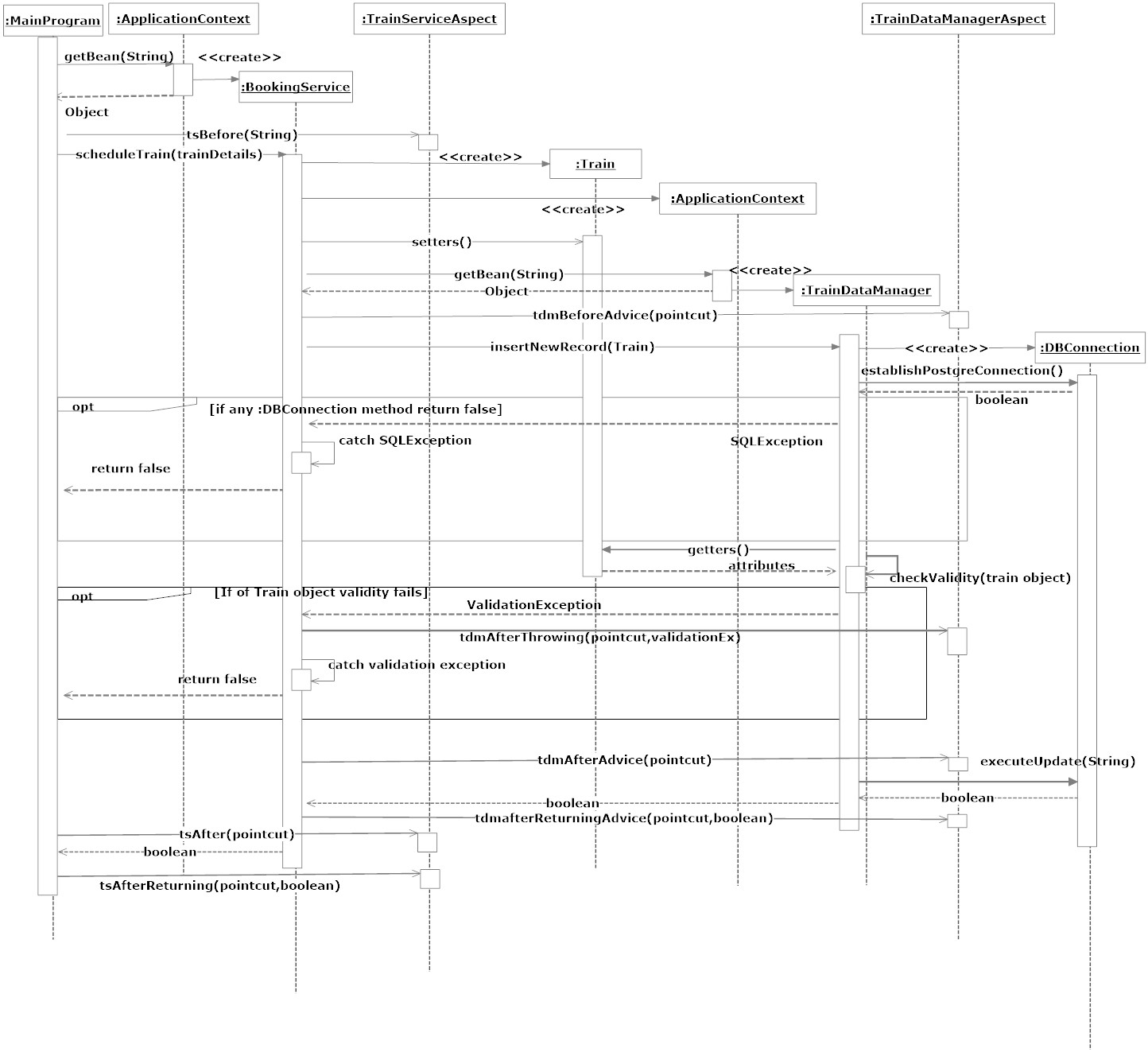


Figure 3‑1 Sequence diagram for scheduling train

# Implemented Aspects

## Point Cut

A point cut is a combination of joint points where advices being invoked at runtime (Spring AOP). The following image illustrates how a point cut is declared in the solution.

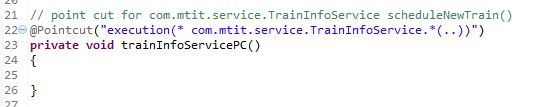


Figure 4‑1.1 point cut for TrainInfoService class in Service Layer

According to above declared point cut it will execute advices for all method invocations in *TrainInfoService* under *com.mtit.service package*

## Before Advice

The before advice will declare a joint point before invoking respective methods. In Spring AOP *@Before* annotation is used to declare a before advice.

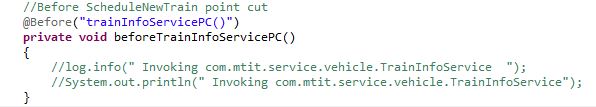


Figure 4.2 Before advice for TrainInfoService class in service layer

## After Advice

Similar to “Before Advice”, the” After Advice “will invoke after executing respective methods in point cut expression. The annotation used for “After Advice” is *@After*

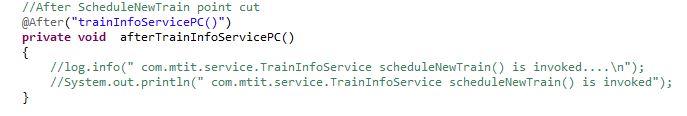


Figure 4.3. After advice for TrainInfoService class point cut

## After Returning

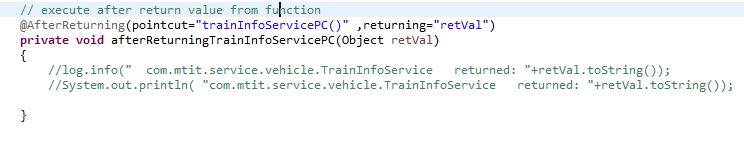
This advice executes when respective callee object in point cut has returned a value after finishing it’s execution to its caller object. The *“@After Returning*” annotation is used to declare these advices.

Figure 4.4. After returning advice for trainInfoServicePC point cut

## After Throwing

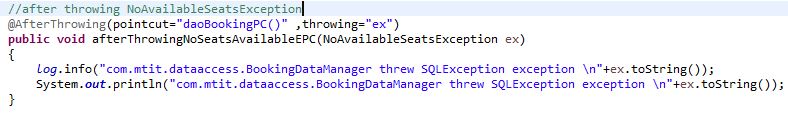
This advice will execute once a particular method under point cut expression scope throws an exception. Also when declaring the advice it should also mention the class type of thrown exception I our solution we have declared below mentioned custom exceptions

Figure 4.5 After throwing advice for doaBookingPC point cut

### No Available Seats Exception

This exception will be thrown when there are not sufficient vacant seats available in train in order to make the reservation

### Roll Back Booking Exception

This exception is thrown when reserved seats were unable to release once a particular seat reservation being cancelled.

### Validation Exception

If a particular Train, Customer or Booking entity object consists of null values in its data this exception will be thrown in order to prevent storing null values in the database.

# Sample Logger Outputs

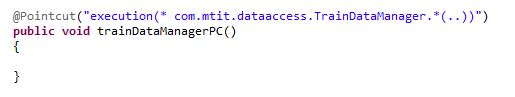


Figure 5 Initiating Spring AOP framework

## Before, After and After Returning

createTrain.JPGIn here we have taken the scheduleNewTrain( service layer) and insertNewRecord( data access layer) to demonstrate Logging advices invocations in point cut expressions

Figure 5.1 Invoke scheduleNewTrain method in main method



### Data Access Layer Advices

Figure 5,1.1 Point cut for TrainDataManager class in data access layer

In above point cut expression it will execute the below advices for all method invocations in TrainDataManager class.



Figure 5.1.1 Before, After, After-Returning advices for above point cut

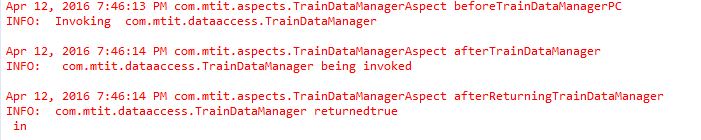


Figure 5.1.1 Log outputs by advices

### servPC.JPGService Layer Advices

Figure 5.1.2 Point cut for Train info service class

The above point cut expression will execute the advices in any method invocation in TrainInfoService class at Service Layer

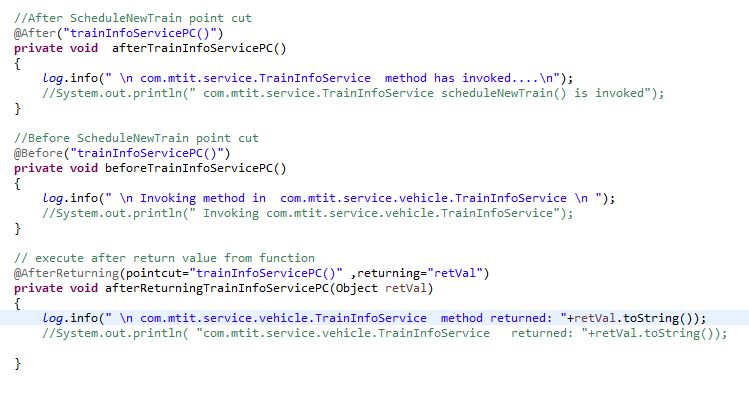


Figure 5.1.2 Advices for TrainInfoService class

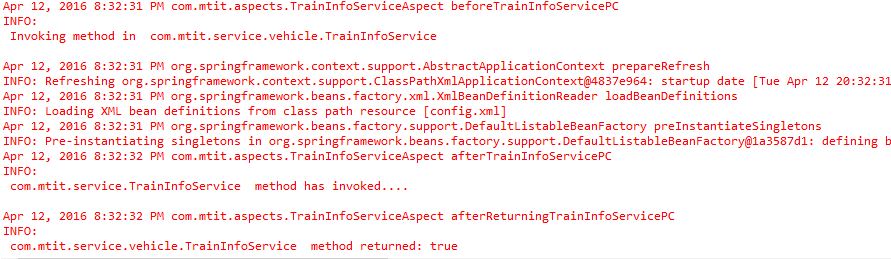


Figure 5.1.2 Log outputs by advices

## After Throwing Advice

In order to demonstrate the point cut for after throwing advice we have selected “NoAvaialabeSeatsException” which have been described Section 4.5. This exception will be thrown by BookingDataManager class in data access layer and caught by BookingInfoService in service layer.

The following images demonstrate how after throwing advice will execute at runtime with respective to NoAvaialabeSeats exception.

This figure illustrates the number of available seats of respective train record in database. However our attempt is to invoke the exception by making a reservation on that particular train where the numbers of reserving seats are higher value than the available seats.

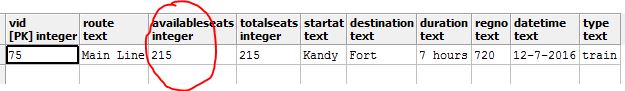


Figure 5.2 Vehicle Record in database

Additionally in order to make a reservation there should be customer id provided with respective Booking object, otherwise it will throw validation exception (Section 4.5) preventing to make the reservation.

Customer rec.JPG

Figure 5.2 Customer record in database

.

Once the above two records are available it is possible to make the reservation.

makeReservation.JPG

Figure 5.2 Reserving seat

Note that the numbers of reserved seats (300) are greater than number of available seats (215). Therefore NoSeatsAvailable exception will be thrown by BookingDataManager object in data access layer.

Now let’s consider the implementation of point cut expression and after throwing advice with respective to BookingDataManager class in data access layer.

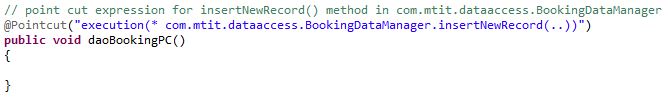


Figure 5.2 Point-cut expression for making reservation method

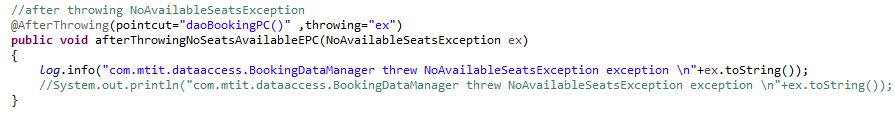
The above point cut will execute advices for insertNewRecord() method in BookingDataManager class at runtime

Figure 5.2 After-throwing advice

Once BookingDataManager object threw NoSeatsAvailable exception the above After-Throwing advice will be triggered by point cut and display the respective log in the console

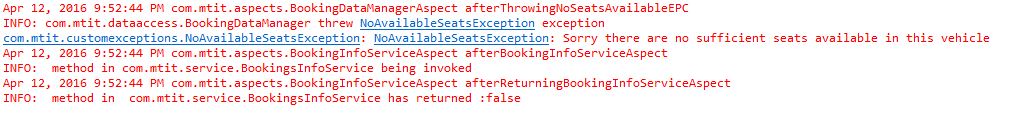


Figure 5.2 Log output for After-throwing advice