Create a Testing Framework for Sporty Shoes Website



This section will cover the following high-level tasks:

1. Create a web-enabled ecommerce website for SportyShoes. (This website will be then tested using JUnit, Selenium, and JMeter.)
2. Do Unit Testing with JUnit.
3. Host the website on AWS EC2 cloud.
4. Do website end-user testing using Selenium WebDriver.
5. Do load testing using JMeter.
6. Automate build and test processes using Jenkins.
7. Use Postman to test the API endpoints.

**Step 1: Create a web-enabled ecommerce website for SportyShoes.**

1. Create a Maven project which is web-enabled.
2. Create pom.xml for including the required components.
3. Create tables in MySQL: admin, category, eproduct, purchase\_items, purchases, and users.
4. Create an entity class Admin for admin table.
5. Create an entity class CartItem.
6. Create an entity class Category for category table.
7. Create an entity class Product for e-product table.
8. Create an entity class Purchase for purchases table.
9. Create an entity class Purchase Item for purchase\_items table.
10. Create an entity class User for users table.
11. Create a DAO class AdminDAO.
12. Create a DAO class CategoryDAO.
13. Create a DAO class ProductDAO.
14. Create a DAO class PurchaseDAO.
15. Create a DAO class PurchaseItemDAO.
16. Create a DAO class UserDAO.
17. Create a service class AdminService.
18. Create a service class CategoryService.
19. Create a service class ProductService.
20. Create a service class PurchaseItemService.
21. Create a service class PurchaseService.
22. Create a service class UserService.
23. Create a controller AdminController to handle admin pages.
24. Create a controller CartController to handle all cart-related pages.
25. Create a controller DashboardController to handle the member dashboard.
26. Create a controller HomeController to handle the home page.
27. Create a controller MemberController to handle all member pages.
28. Create a controller PurchaseController to handle member purchases page.
29. Create application.properties file to configure database access.
30. Configure Hibernate with hibernate.cfg.xml.
31. Create a JSP page cart.jsp to display the user’s cart.
32. Create a JSP page checkout.jsp to do checkout.
33. Create a JSP page confirm.jsp to show checkout confirmation.
34. Create a JSP dashboard.jsp to show member dashboard.
35. Create a JSP page edit-profile.jsp to let members edit their profiles.
36. Create a JSP page gateway.jsp to show a dummy payment gateway.
37. Create a JSP page index.jsp to display the homepage.
38. Create a JSP page login.jsp to do user login.
39. Create a JSP page purchases.jsp to show member’s order history.
40. Create a JSP page register.jsp to do member registration.
41. Create a JSP page register-confirm.jsp to show registration confirmation.
42. Create a JSP page admin/categories.jsp to let admin manage product categories.
43. Create a JSP page admin/change-password.jsp to let the admin change his password.
44. Create a JSP page admin/dashboard.jsp to show the admin dashboard.
45. Create a JSP page admin/edit-category.jsp to let admin add or edit a product category.
46. Create a JSP page admin/edit-product.jsp to let admin add or edit a product.
47. Create a JSP page admin/login.jsp to do admin login.
48. Create a JSP page admin/members.jsp to let admin see the members list.
49. Create a JSP page admin/products.jsp to let admin manage products.
50. Create a JSP page admin/purchases.jsp to see order history.
51. Create a JSP page components/admin-footer.jsp to show footer in admin pages.
52. Create a JSP page components/admin-header.jsp to show header in admin pages.
53. Create a JSP page components/admin-topbar.jsp to show menu options in admin pages.
54. Create a JSP page components/footer.jsp to show footer in public pages.
55. Create a JSP page components/header.jsp to show header in public pages.
56. Create a JSP page components/topbar.jsp to show menu for logged in members.
57. Set up jdbc.properties for Hibernate.
58. Set up main-servlet.xml for Spring MVC.
59. Configure web.xml.
60. Build the project.
61. Publish and start the project.
62. Run the project.
63. Push the code to your GitHub repositories.

**1.1: Create a Maven Project which is web-enabled**

* Go to the **File** menu. Choose **New->Maven Project.**
* Uncheck **Create a Simple Project** and check **Use Default Workspace Location** and click on **Next.**
* From the **archetype** list, choose the row that has **artifactId** as **maven-archetype-webapp** and click on **Next.**
* Enter **groupId** and **artifactId** as **SportyShoes** and click on **Finish.**
* This will create the project files in the Project Explorer.

**1.2: Create pom.xml for including the required components**

* In the Project Explorer, expand **SportyShoes** and double click **pom.xml.**
* Add the following entries:

**<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SportyShoes</groupId>

<artifactId>SportyShoes</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>Section7LEP Maven Webapp</name>

<url>http://maven.apache.org</url>

<!-- JBoss repository for Hibernate -->

<repositories>

<repository>

<id>JBoss repository</id>

<url>http://repository.jboss.org/nexus/content/groups/public/</url>

</repository>

</repositories>

<properties>

<org.springframework.version>5.1.6.RELEASE</org.springframework.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.1-api</artifactId>

<version>1.0.0.Final</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>5.1.6.RELEASE</version>

<scope>test</scope>

</dependency>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.0.5.Final</version>

</dependency>

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.12.1.GA</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.9</version>

</dependency>

<dependency>

<groupId>javax.transaction</groupId>

<artifactId>jta</artifactId>

<version>1.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>javax.activation</groupId>

<artifactId>activation</artifactId>

<version>1.1.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>2.21.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>Section7LEP2</finalName>

<sourceDirectory>src/main</sourceDirectory>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<**source**>1.8</**source**>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<configuration>

<webXml>src\main\webapp\WEB-INF\web.xml</webXml>

</configuration>

<version>2.4</version>

<executions>

<execution>

<id>default-war</id>

<phase>package</phase>

<goals>

<goal>war</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**1.3: Create tables in MySQL: admin, category, eproduct, purchase\_items, purchases, and users**

* Login to the MySQL command line console.
* Type **CREATE DATABASE ecommerce** and press **Enter.**
* Type **USE ecommerce** and press **Enter.**
* Enter the following script and execute it:

--

-- Host: localhost Database: ecommerce

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

-- Server version 5.7.24-0ubuntu0.18.04.1

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `admin`

--

**DROPTABLEIFEXISTS**`admin`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`admin`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`admin\_id`varchar(20) DEFAULT NULL,

`admin\_pwd`varchar(10) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `admin`

--

**LOCK TABLES**`admin`**WRITE**;

/\*!40000 ALTER TABLE `admin` DISABLE KEYS \*/;

**INSERT INTO**`admin`**VALUES**(1,'admin','aaaaaa');

/\*!40000 ALTER TABLE `admin` ENABLE KEYS \*/;

**UNLOCK TABLES**;

--

-- Table structure for table `category`

--

**DROP TABLE IF EXISTS**`category`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`category`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`name`varchar(50) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `category`

--

**LOCK TABLES**`category`**WRITE**;

/\*!40000 ALTER TABLE `category` DISABLE KEYS \*/;

**INSERT INTO**`category`**VALUES**(1,'Laptops '),(2,'Mobiles'),(3,'Pen Drives'),(4,'Mouse'),(7,'Keyboards'),(9,'Cameras'),(10,'Motherboards');

/\*!40000 ALTER TABLE `category` ENABLE KEYS \*/;

**UNLOCK TABLES**;

--

-- Table structure for table `eproduct`

--

**DROP TABLE IF EXISTS**`eproduct`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`eproduct`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`name`varchar(100) DEFAULT NULL,

`price`decimal(10,2) DEFAULT NULL,

`date\_added`timestampNOTNULL DEFAULT **CURRENT\_TIMESTAMP**,

`parts\_hdd`varchar(10) DEFAULT NULL,

`parts\_cpu`varchar(10) DEFAULT NULL,

`parts\_ram`varchar(10) DEFAULT NULL,

`category\_id`bigint(20) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=108 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `eproduct`

--

**LOCK TABLES**`eproduct`**WRITE**;

/\*!40000 ALTER TABLE `eproduct` DISABLE KEYS \*/;

**INSERT INTO**`eproduct`**VALUES**(1,'HP Laptop ABC',21900.00,'2019-06-04 07:18:57','2 Gb HDD','AMD Phenom','4 Gb',1),(2,'Acer Laptop ABC',23300.00,'2019-06-04 07:19:07','500 Gb HDD','Core-i7','4 Gb',1),(3,'Lenovo Laptop ABC',33322.00,'2019-06-04 07:19:19','1 Tb HDD','Core-i7','8 Gb',1),(4,'Redmi 128 Gb',10000.00,'2019-06-16 06:38:38',NULL,NULL,NULL,2),(5,'iPhone x',10000.00,'2019-06-16 06:38:38',NULL,NULL,NULL,2),(105,'Asus Motherboard Intel icore7',13400.00,'2019-06-22 07:44:55',NULL,NULL,NULL,10),(106,'OnePlus 16 Gb ',18200.00,'2019-06-22 07:45:26',NULL,NULL,NULL,2),(107,'ABC Product',134.00,'2019-06-26 05:12:50',NULL,NULL,NULL,1);

/\*!40000 ALTER TABLE `eproduct` ENABLE KEYS \*/;

**UNLOCK TABLES**;

--

-- Table structure for table `purchase\_items`

--

**DROP TABLE IF EXISTS**`purchase\_items`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`purchase\_items`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`purchase\_id`bigint(20)NOTNULL,

`product\_id`bigint(20)NOTNULL,

`user\_id`bigint(20)NOTNULL,

`rate`decimal(10,2) DEFAULT NULL,

`qty`smallint(6) DEFAULT NULL,

`price`decimal(10,2) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `purchase\_items`

--

**LOCK TABLES**`purchase\_items`**WRITE**;

/\*!40000 ALTER TABLE `purchase\_items` DISABLE KEYS \*/;

**INSERT INTO**`purchase\_items`**VALUES**(1,3,110,24,56000.00,1,56000.00),(2,4,3,24,33322.00,1,33322.00),(3,4,2,24,23300.00,1,23300.00),(4,5,5,24,10000.00,1,10000.00);

/\*!40000 ALTER TABLE `purchase\_items` ENABLE KEYS \*/;

**UNLOCK TABLES**;

--

-- Table structure for table `purchases`

--

**DROP TABLE IF EXISTS**`purchases`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`purchases`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`date`datetime DEFAULT NULL,

`user\_id`bigint(20)NOTNULL,

`gross\_total`decimal(10,2) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `purchases`

--

**LOCK TABLES**`purchases`**WRITE**;

/\*!40000 ALTER TABLE `purchases` DISABLE KEYS \*/;

**INSERT INTO**`purchases`**VALUES**(2,'2019-06-28 00:57:24',24,10000.00),(3,'2019-06-28 00:59:03',24,56000.00),(4,'2019-06-28 00:59:41',24,56622.00),(5,'2019-06-28 02:23:03',24,10000.00);

/\*!40000 ALTER TABLE `purchases` ENABLE KEYS \*/;

**UNLOCK TABLES**;

--

-- Table structure for table `users`

--

**DROP TABLE IF EXISTS**`users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

**CREATE TABLE**`users`(

`ID`bigint(20)NOTNULL AUTO\_INCREMENT,

`fname`varchar(25) DEFAULT NULL,

`lname`varchar(25) DEFAULT NULL,

`address`varchar(100) DEFAULT NULL,

`age`int(11) DEFAULT NULL,

`date\_added`datetime DEFAULT NULL,

`emailid`varchar(50) DEFAULT NULL,

`pwd`varchar(10) DEFAULT NULL,

**PRIMARY KEY**(`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=26 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `users`

--

**LOCK TABLES**`users`**WRITE**;

/\*!40000 ALTER TABLE `users` DISABLE KEYS \*/;

**INSERT INTO**`users`**VALUES**(1,'John ','Doe ','127 Park Avenue NYC 3817',20,'2019-06-23 17:59:13',NULL,NULL),(2,'Jack','Jones','13 Maple Street Washington DC 7615',22,'2019-06-23 17:59:55',NULL,NULL),(3,'John','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(4,'John2','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(5,'John3','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(6,'John 4','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(7,'John 5','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(8,'John 6','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(9,'John 6','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(10,'John 8','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(11,'John 9','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(12,'John 10','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(13,'John 11','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(14,'John 12','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(15,'John 13','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(16,'John 14','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(17,'John 15','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(18,'John 16','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(19,'John 17','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(20,'John 18','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(21,'John 19','Doe','127 Park Avenue NYC 3817',25,'2019-06-23 17:59:13',NULL,NULL),(24,'amit','sengupta','some address',29,'2019-06-27 22:47:45','amit@amit.com','aaaaaa'),(25,'amit','hotmail','',0,'2019-06-28 01:12:29','sengupta\_amit@hotmail.com','aaaaaa');

/\*!40000 ALTER TABLE `users` ENABLE KEYS \*/;

**UNLOCK TABLES**;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2019-06-28 12:32:50

**1.4: Create an entity class Admin for admin table**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **Admin** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "admin")**

**public class** Admin {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private long** ID;

**@Column(name = "admin\_id")**

**private String** adminId;

**@Column(name = "admin\_pwd")**

**privateString** pwd;

**public** long getID(){**return this.ID**;}

**public String** getAdminId(){**return this**.adminId;}

**publicString** getAdminPwd(){**returnthis**.pwd;}

**public void** setID(long id){**this**.ID = id;}

**public void** setAdminId(**String** value){**this**.adminId= value;}

**public void** setAdminPwd(**String** value){**this**.pwd = value;}

}

**1.5: Create an entity class CartItem**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **CartItem** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

/\*\*

**\*** This is NOT a Hibernate class or a table class. This is a POJO which is used internally within the app

**\*@authoroem**

**\***

\*/

**public class** CartItem {

**private** long productId;

**private String** name;

**private BigDecimal** rate;

**private BigDecimal** price;

**private int** qty;

**public** long getProductId(){**return this**.productId;}

**public String** getName(){**return this.name**;}

**public BigDecimal** getPrice(){**return this.price**;}

**public BigDecimal** getRate(){**return this.rate**;}

**public** int getQty(){**return this.qty**;}

**public void** setProductId(long id){**this**.productId= id;}

**public void** setName(**String** value){**this**.name = value;}

**public void** setPrice(**BigDecimal** value){**this**.price = value;}

**public void** setRate(**BigDecimal** value){**this**.rate = value;}

**public void** setQty(int value){**this**.qty = value;}

}

**1.6: Create an entity class Category for category table**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **Category** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "category")**

**public class** Category {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private long** ID;

**@Column(name = "name")**

**private String** name;

**public** long getID(){**return this.ID**;}

**public String** getName(){**return this.name**;}

**public void** setID(long id){**this**.ID = id;}

**public void** setName(**String** value){**this**.name = value;}

}

**1.7: Create an entity class Product for an e-product table**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **Product** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.OneToOne;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "eproduct")**

**public class** Product {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private long** ID;

**@Column(name = "name")**

**private String** name;

**@Column(name = "price")**

**private BigDecimal** price;

**@Column(name = "date\_added")**

**private Date** dateAdded;

**@Column(name = "category\_id")**

**private** long categoryId;

**public** long getID(){**return this.ID**;}

**public String** getName(){**return this.name**;}

**publicBigDecimal** getPrice(){**return this.price**;}

**public long** getCategoryId(){**return this**.categoryId;}

**public Date** getDateAdded(){**return this**.dateAdded;}

**public void** setID(long id){**this**.ID = id;}

**public void** setName(**String** value){**this**.name = value;}

**public void** setPrice(**BigDecimal** value){**this**.price = value;}

**public void** setCategoryId(long value){**this**.categoryId = value;}

**public void** setDateAdded(**Date** date){**this**.dateAdded = date;}

}

**1.8: Create an entity class Purchase**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **Purchase,** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "purchases")**

**publicclass** Purchase {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private** long ID;

**@Column(name = "user\_id")**

**private** long userId;

**@Column(name = "date")**

**private Date** date;

**@Column(name = "gross\_total")**

**private BigDecimal** total;

**public** long getID(){**return this.ID**;}

**public** long getUserId(){**returnthis**.userId;}

**public BigDecimal** getTotal(){**return this.total**;}

**public Date** getDate(){**return this.date**;}

**public** void setID(long id){**this**.ID = id;}

**public** void setUserId(long value){**this**.userId = value;}

**public** void setTotal(**BigDecimal** value){**this**.total = value;}

**public** void setDate(**Date** date){**this**.date = date;}

}

**1.9: Create an entity class PurchaseItem for purchase\_items table**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class,** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **PurchaseItem,** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "purchase\_items")**

**public class** PurchaseItem {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private** long ID;

**@Column(name = "purchase\_id")**

**private** long purchaseId;

**@Column(name = "product\_id")**

**private** long productId;

**@Column(name = "user\_id")**

**private** long userId;

**@Column(name = "rate")**

**private BigDecimal** rate;

**@Column(name = "qty")**

**private** int qty;

**@Column(name = "price")**

**private BigDecimal** price;

**public** long getID(){**return this.ID**;}

**public** long getPurchaseId(){**return this**.purchaseId;}

**public** long getProductId(){**return this**.productId;}

**public** long getUserId(){**return this**.userId;}

**public BigDecimal** getRate(){**return this.rate**;}

**public** int getQty(){**return this.qty**;}

**public BigDecimal** getPrice(){**return this.price**;}

**public** void setID(long id){**this**.ID = id;}

**public** void setPurchaseId(long value){**this**.purchaseId = value;}

**public** void setProductId(long value){**this**.productId = value;}

**public** void setUserId(long value){**this**.userId = value;}

**public** void setRate(**BigDecimal** value){**this**.rate = value;}

**public** void setQty(int value){**this**.qty= value;}

**public** void setPrice(**BigDecimal** value){**this**.price= value;}

}

**1.10: Create an entity class User for users table**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **User** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "users")**

**public class** User {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "ID")**

**private** long ID;

**@Column(name = "fname")**

**private String** fname;

**@Column(name = "lname")**

**private String** lname;

**@Column(name = "address")**

**private String** address;

**@Column(name = "age")**

**private** int age;

**@Column(name = "date\_added")**

**private Date** dateAdded;

**@Column(name = "emailid")**

**private String** emailId;

**@Column(name = "pwd")**

**private String** pwd;

**public** long getID(){**return this**.ID;}

**public String** getEmail(){**return this**.emailId;}

**public String** getFname(){**return this**.fname;}

**public String** getLname(){**return this.name**;}

**public String** getAddress(){**return this.address**;}

**public String** getPwd(){**return this.pwd**;}

**public** int getAge(){**return this.age**;}

**public Date** getDateAdded(){**return this**.dateAdded;}

**public** void setID(long id){**this**.ID = id;}

**public** void setEmail(**String** value){**this**.emailId = value;}

**public** void setFname(**String** value){**this**.fname = value;}

**public** void setLname(**String** value){**this**.lname = value;}

**public** void setAddress(**String** value){**this**.address= value;}

**public** void setPwd(**String** value){**this**.pwd= value;}

**public** void setAge(int value){**this**.age= value;}

**public** void setDateAdded(**Date** date){**this**.dateAdded = date;}

}

**1.11: Create a DAO class AdminDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **AdminDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Admin;

**@Repository**

**@Component**

**public class** AdminDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public** Admin authenticate(**String** adminId,**String** pwd){

**List**<Admin> list =**this**.sessionFactory.getCurrentSession().createQuery("from Admin where admin\_id=:admin\_id and admin\_pwd=:admin\_pwd")

.setParameter("admin\_id", adminId)

.setParameter("admin\_pwd", pwd)

.list();

**if**(list.size()>0)

**return**(Admin) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public** Admin getAdminById(long id){

**List**<Admin> list =**this**.sessionFactory.getCurrentSession().createQuery("from Admin where ID=:admin\_id")

.setParameter("admin\_id", id)

.list();

**if**(list.size()>0)

**return**(Admin) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public** void updatePwd(Admin admin){

**String** sql ="update Admin set admin\_pwd=:admin\_pwd where ID=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("admin\_pwd", admin.getAdminPwd());

query.setParameter("id", admin.getID());

query.executeUpdate();

}

}

**1.12: Create a DAO class CategoryDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **CategoryDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Category;

**@Repository**

**@Component**

**public class** CategoryDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public Category** getCategoryById(long id){

**String** strId =**String**.valueOf(id);

**List**<**Category**> list =**this**.sessionFactory.getCurrentSession().createQuery("from Category where id="+ strId).list();

**if**(list.size()>0)

**return**(**Category**) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public** void updateCategory(**Category** category){

**String** sql ="";

**if**(category.getID()==0)

**this**.sessionFactory.getCurrentSession().save(category);

**elseif**(category.getID()>0){

sql ="update Category set name=:name where "+

" ID=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("name", category.getName());

**if**(category.getID()>0)

query.setParameter("id", category.getID());

query.executeUpdate();

}

}

**@SuppressWarnings("unchecked")**

**public**void deleteCategory(long id){

// mark all category\_id of products with this category to zero before deleting the category row

**String** sql ="";

sql ="update Product set category\_id=0 where category\_id=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("id", id);

query.executeUpdate();

sql ="delete from Category where ID=:id";

query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("id", id);

query.executeUpdate();

}

**@SuppressWarnings("unchecked")**

**publicList**<**Category**> getAllCategories(){

**List**<**Category**> list =**this**.sessionFactory.getCurrentSession().createQuery("from Category order by name").list();

**return** list;

}

}

**1.13: Create a DAO class ProductDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **ProductDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Product;

**@Repository**

**@Component**

**public class** ProductDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public** Product getProductById(long id){

**String** strId =**String**.valueOf(id);

**List**<Product> list =**this**.sessionFactory.getCurrentSession().createQuery("from Product where id="+ strId).list();

**if**(list.size()>0)

**return**(Product) list.get(0);

**else**

**returnnull**;

}

**@SuppressWarnings("unchecked")**

**public** void updateProduct(Product product){

**String** sql ="";

**if**(product.getID()==0)

**this**.sessionFactory.getCurrentSession().save(product);

**elseif**(product.getID()>0){

sql ="update Product set name=:name, price=:price, category\_id=:category\_id where "+

" ID=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("name", product.getName());

query.setParameter("price", product.getPrice());

query.setParameter("category\_id", product.getCategoryId());

query.setParameter("id", product.getID());

query.executeUpdate();

}

}

**@SuppressWarnings("unchecked")**

**public** void deleteProduct(long id){

// delete all purchase items for this product before deleting this product

// Pending:Purchase total is not updated in the purchase total - it shows the old value

**String** sql ="";

sql ="delete from PurchaseItem where product\_id=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("id", id);

sql ="delete from Product where ID=:id";

query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("id", id);

query.executeUpdate();

}

**@SuppressWarnings("unchecked")**

**public List**<Product> getAllProducts(){

**List**<Product> list =**this**.sessionFactory.getCurrentSession().createQuery("from Product order by name").list();

**return** list;

}

}

**1.14: Create a DAO class PurchaseDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **PurchaseDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Purchase;

**@Repository**

**@Component**

**public class** PurchaseDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public** Purchase getPurchaseById(long id){

**String** strId =**String**.valueOf(id);

**List**<Purchase> list =**this**.sessionFactory.getCurrentSession().createQuery("from Purchase where id="+ strId).list();

**if**(list.size()>0)

**return**(Purchase) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public List**<Purchase> getAllItems(){

**List**<Purchase> list =**this**.sessionFactory.getCurrentSession().createQuery("from Purchase order by ID desc").list();

**return** list;

}

**@SuppressWarnings("unchecked")**

**public List**<Purchase> getAllItemsByUserId(long userId){

**String** strId =**String**.valueOf(userId);

**List**<Purchase> list =**this**.sessionFactory.getCurrentSession().createQuery("from Purchase where user\_id="+ strId +" order by ID desc").list();

**return** list;

}

**@SuppressWarnings("unchecked")**

**public** long updatePurchase(Purchase purchase){

**String** sql ="";

long newId =0;

**if**(purchase.getID()==0){

**this**.sessionFactory.getCurrentSession().save(purchase);

newId = purchase.getID();

}**else**{

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("user\_id", purchase.getUserId());

query.setParameter("gross\_total", purchase.getTotal());

query.executeUpdate();

}

**return** newId;

}

}

**1.15: Create a DAO class PurchaseItemDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **PurchaseItemDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.PurchaseItem;

**@Repository**

**@Component**

**public class** PurchaseItemDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public** PurchaseItem getItemById(long id){

**String** strId =**String**.valueOf(id);

**List**<PurchaseItem> list =**this**.sessionFactory.getCurrentSession().createQuery("from PurchaseItem where id="+ strId).list();

**if**(list.size()>0)

**return**(PurchaseItem) list.get(0);

**else**

**returnnull**;

}

**@SuppressWarnings("unchecked")**

**public List**<PurchaseItem> getAllItemsByPurchaseId(long purchaseId){

**String** strId =**String**.valueOf(purchaseId);

**List**<PurchaseItem> list =**this**.sessionFactory.getCurrentSession().createQuery("from PurchaseItem where purchase\_id="+ strId).list();

**return** list;

}

**@SuppressWarnings("unchecked")**

**public**void updateItem(PurchaseItem item){

**this**.sessionFactory.getCurrentSession().save(item);

}

**@SuppressWarnings("unchecked")**

**public** void deleteItem(long id){

//Pending: Purchase total in purchase table should be updated after this

**String** sql ="";

sql +="delete from PurchaseItem where ID=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("id", id);

query.executeUpdate();

}

**@SuppressWarnings("unchecked")**

**public** void deleteAllItemsForPurchaseId(long purchaseId){

**String** sql ="";

sql +="delete from PurchaseItem where purchase\_id=:purchase\_id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("purchase\_id", purchaseId);

query.executeUpdate();

}

}

**1.16: Create a DAO class UserDAO**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.dao** and in **Name,** enter **UserDAO** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.dao;

**import** java.util.Calendar;

**import** java.util.List;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Session;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.entity.Admin;

**import** com.ecommerce.entity.User;

**@Repository**

**@Component**

**public class** UserDAO {

**@Autowired**

**private** SessionFactory sessionFactory;

**@SuppressWarnings("unchecked")**

**public** User authenticate(**String** emailId,**String** pwd){

**List**<User> list =**this**.sessionFactory.getCurrentSession().createQuery("from User where emailid=:emailid and pwd=:pwd")

.setParameter("emailid", emailId)

.setParameter("pwd", pwd)

.list();

**if**(list.size()>0)

**return**(User) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public** User getUserById(long id){

**String** strId =**String**.valueOf(id);

**List**<User> list =**this**.sessionFactory.getCurrentSession().createQuery("from User where id="+ strId).list();

**if**(list.size()>0)

**return**(User) list.get(0);

**else**

**return null**;

}

**@SuppressWarnings("unchecked")**

**public** User getUserByEmailId(**String** emailId){

**List**<User> list =**this**.sessionFactory.getCurrentSession().createQuery("from User where emailid='"+ emailId +"'").list();

**if**(list.size()>0)

**return**(User) list.get(0);

**else**

**returnnull**;

}

**@SuppressWarnings("unchecked")**

**public** void updateUser(User user){

**String** sql ="";

**if**(user.getID()==0){

user.setDateAdded(**Calendar**.getInstance().getTime());

**this**.sessionFactory.getCurrentSession().save(user);

}**elseif**(user.getID()>0){

sql ="update User set fname=:fname, lname=:lname, address=:address, age=:age, pwd=:pwd where "+

" ID=:id";

Query query =**this**.sessionFactory.getCurrentSession().createQuery(sql);

query.setParameter("fname", user.getFname());

query.setParameter("lname", user.getLname());

query.setParameter("address", user.getAddress());

query.setParameter("age", user.getAge());

query.setParameter("pwd", user.getPwd());

**if**(user.getID()>0)

query.setParameter("id", user.getID());

query.executeUpdate();

}

}

**@SuppressWarnings("unchecked")**

**public List**<User> getAllUsers(){

**List**<User> list =**this**.sessionFactory.getCurrentSession().createQuery("from User order by date\_added").list();

**return** list;

}

}

**1.17: Create a service class AdminService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **AdminService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.AdminDAO;

**import** com.ecommerce.entity.Admin;

**@Component**

**publicclass** AdminService {

**@Autowired**

**private** AdminDAO adminDAO;

**@Transactional**

**public** Admin authenticate(**String** adminId,**String** pwd){

**return** adminDAO.authenticate(adminId, pwd);

}

**@Transactional**

**public** Admin getAdminById(long id){

**return** adminDAO.getAdminById(id);

}

**@Transactional**

**public** void updatePwd(Admin admin){

adminDAO.updatePwd(admin);

}

}

**1.18: Create a service class CategoryService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **CategoryService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.CategoryDAO;

**import** com.ecommerce.entity.Category;

**@Component**

**public class** CategoryService {

**@Autowired**

**private** CategoryDAO categoryDAO;

**@Transactional**

**public Category** getCategoryById(long id){

**return** categoryDAO.getCategoryById(id);

}

**@Transactional**

**public** void updateCategory(**Category** category){

categoryDAO.updateCategory(category);

}

**@Transactional**

**public** void deleteCategory(long id){

categoryDAO.deleteCategory(id);

}

**@Transactional**

**public List**<**Category**> getAllCategories(){

**return** categoryDAO.getAllCategories();

}

**@Transactional**

**public String** getCategoriesDropDown(long id){

**StringBuilder** sb =**newStringBuilder**("");

**List**<**Category**> list = categoryDAO.getAllCategories();

**for**(**Category** cat: list){

**if**(cat.getID()== id)

sb.append("<option value="+**String**.valueOf(cat.getID())+" selected>"+ cat.getName()+"</option>");

**else**

sb.append("<option value="+**String**.valueOf(cat.getID())+">"+ cat.getName()+"</option>");

}

**return** sb.toString();

}

}

**1.19: Create a service class ProductService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **ProductService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.ProductDAO;

**import** com.ecommerce.entity.Product;

**@Component**

**public class** ProductService {

**@Autowired**

**private** ProductDAO productDAO;

**@Transactional**

**public** Product getProductById(long id){

**return** productDAO.getProductById(id);

}

**@Transactional**

**public** void updateProduct(Product product){

productDAO.updateProduct(product);

}

**@Transactional**

**public** void deleteProduct(long id){

productDAO.deleteProduct(id);

}

**@Transactional**

**public List**<Product> getAllProducts(){

**return** productDAO.getAllProducts();

}

**@Transactional**

**public List**<**Object**> getAllProductsWithJoin(){

**return** productDAO.getAllProductsWithJoin();

}

}

**1.20: Create a service class PurchaseItemService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* In the Wizard list choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **PurchaseItemService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.PurchaseItemDAO;

**import** com.ecommerce.entity.PurchaseItem;

**@Component**

**public class** PurchaseItemService {

**@Autowired**

**private** PurchaseItemDAO purchaseItemDAO;

**@Transactional**

**public** PurchaseItem getItemById(long id){

**return** purchaseItemDAO.getItemById(id);

}

**@Transactional**

**publicList**<PurchaseItem> getAllItemsByPurchaseId(long purchaseId){

**return** purchaseItemDAO.getAllItemsByPurchaseId(purchaseId);

}

**@Transactional**

**public**void updateItem(PurchaseItem item){

purchaseItemDAO.updateItem(item);

}

**@Transactional**

**public**void deleteItem(long id){

purchaseItemDAO.deleteAllItemsForPurchaseId(id);

}

**@Transactional**

**public**void deleteAllItemsForPurchaseId(long purchaseId){

purchaseItemDAO.deleteAllItemsForPurchaseId(purchaseId);

}

}

**1.21: Create a service class PurchaseService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* In the Wizard list choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **PurchaseService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.PurchaseDAO;

**import** com.ecommerce.entity.Purchase;

**@Component**

**public class** PurchaseService {

**@Autowired**

**private** PurchaseDAO purchaseDAO;

**@Transactional**

**public** Purchase getPurchaseById(long id){

**return** purchaseDAO.getPurchaseById(id);

}

**@Transactional**

**public List**<Purchase> getAllItems(){

**return** purchaseDAO.getAllItems();

}

**@Transactional**

**public List**<Purchase> getAllItemsByUserId(long userId){

**return** purchaseDAO.getAllItemsByUserId(userId);

}

**@Transactional**

**public** long updatePurchase(Purchase purchase){

**return** purchaseDAO.updatePurchase(purchase);

}

**@Transactional**

**public** void deletePurchase(long id){

purchaseDAO.deletePurchase(id);

}

}

**1.22: Create a service class UserService**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **UserService** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.ecommerce.dao.UserDAO;

**import** com.ecommerce.entity.Admin;

**import** com.ecommerce.entity.User;

**@Component**

**public class** UserService {

**@Autowired**

**private** UserDAO userDAO;

**@Transactional**

**public** User authenticate(**String** userId,**String** pwd){

**return** userDAO.authenticate(userId, pwd);

}

**@Transactional**

**public** User getUserById(long id){

**return** userDAO.getUserById(id);

}

**@Transactional**

**public** User getUserByEmailId(**String** emailId){

**return** userDAO.getUserByEmailId(emailId);

}

**@Transactional**

**public** void updateUser(User user){

userDAO.updateUser(user);

}

**@Transactional**

**publicList**<User> getAllUsers(){

**return** userDAO.getAllUsers();

}

}

**1.23: Create a controller AdminController to handle admin pages**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **AdminController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.ecommerce.entity.**\***;

**import** com.ecommerce.service.**\***;

**@Controller**

**public class** AdminController {

**@Autowired**

**private** AdminService adminService;

**@Autowired**

**private** CategoryService categoryService;

**@Autowired**

**private** ProductService productService;

**@Autowired**

**private** PurchaseService purchaseService;

**@Autowired**

**private** PurchaseItemService purchaseItemService;

**@Autowired**

**private** UserService userService;

**@RequestMapping(value = "/adminlogin", method = RequestMethod.GET)**

**public String** login(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

map.addAttribute("pageTitle","ADMIN LOGIN");

**return**"admin/login";

}

**@RequestMapping(value = "/adminloginaction", method = RequestMethod.POST)**

**public String** loginAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="admin\_id", required=true)String** adminId,

**@RequestParam(value="admin\_pwd", required=true)String** adminPwd)

{

Admin admin = adminService.authenticate(adminId, adminPwd);

**if**(admin ==**null**){

map.addAttribute("error","Admin login failed");

**return**"admin/login";

}

// store admin id in session

HttpSession session = request.getSession();

session.setAttribute("admin\_id", admin.getID());

**return**"admin/dashboard";

}

**@RequestMapping(value = "/admindashboard", method = RequestMethod.GET)**

**public String** dashboard(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

map.addAttribute("pageTitle","ADMIN DASHBOARD");

**return**"admin/dashboard";

}

**@RequestMapping(value = "/adminchangepassword", method = RequestMethod.GET)**

**public String** changePwd(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

Admin admin = adminService.getAdminById((**Long**) session.getAttribute("admin\_id"));

map.addAttribute("admin", admin);

map.addAttribute("pageTitle","ADMIN CHANGE PASSWORD");

**return**"admin/change-password";

}

**@RequestMapping(value = "/adminchangepwdaction", method = RequestMethod.POST)**

**public String** updatePassword(ModelMap map,**@RequestParam(value="pwd", required=true)String** pwd,

**@RequestParam(value="pwd2", required=true)String** pwd2,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

**if**(pwd ==**null**|| pwd2 ==**null**|| pwd.equals("")|| pwd2.equals("")){

map.addAttribute("error","Error , Incomplete passwords submitted.");

**return**"admin/change-password";

}

**if**(!pwd.equals(pwd2)){

map.addAttribute("error","Error , Passwords do not match.");

**return**"admin/change-password";

}

Admin admin = adminService.getAdminById((**Long**) session.getAttribute("admin\_id"));

admin.setAdminPwd(pwd);

adminService.updatePwd(admin);

**return**"admin/dashboard";

}

**@RequestMapping(value = "/adminproducts", method = RequestMethod.GET)**

**public String** products(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

**List**<Product> list = productService.getAllProducts();

// use a MAP to link category names to each product in list

**HashMap**<**Long**,**String**> mapCats =**newHashMap**<**Long**,**String**>();

**for**(Product product: list){

**Category** category = categoryService.getCategoryById(product.getCategoryId());

**if**(category !=**null**)

mapCats.put(product.getID(), category.getName());

}

map.addAttribute("list", list);

map.addAttribute("mapCats", mapCats);

map.addAttribute("pageTitle","ADMIN SETUP PRODUCTS");

**return**"admin/products";

}

**@RequestMapping(value = "/admincategories", method = RequestMethod.GET)**

**public String** categories(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

**List**<**Category**> list = categoryService.getAllCategories();

map.addAttribute("list", list);

map.addAttribute("pageTitle","ADMIN SETUP PRODUCT CATEGORIES");

**return**"admin/categories";

}

**@RequestMapping(value = "/adminmembers", method = RequestMethod.GET)**

**public String** members(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

**List**<User> list = userService.getAllUsers();

map.addAttribute("list", list);

map.addAttribute("pageTitle","ADMIN BROWSE MEMBERS");

**return**"admin/members";

}

**@RequestMapping(value = "/adminpurchases", method = RequestMethod.GET)**

**publicString** purchases(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

**List**<Purchase> list = purchaseService.getAllItems();

**BigDecimal** total =**newBigDecimal**(0.0);

**for**(Purchase purchase: list){

total = total.add(purchase.getTotal());

}

// use MAPs to mape users to each purchase and item names to each purchase item row

**HashMap**<**Long**,**String**> mapItems =**newHashMap**<**Long**,**String**>();

**HashMap**<**Long**,**String**> mapUsers =**newHashMap**<**Long**,**String**>();

**for**(Purchase purchase: list){

total = total.add(purchase.getTotal());

User user = userService.getUserById(purchase.getUserId());

**if**(user !=**null**)

mapUsers.put(purchase.getID(), user.getFname()+""+ user.getLname());

**List**<PurchaseItem> itemList = purchaseItemService.getAllItemsByPurchaseId(purchase.getID());

**StringBuilder** sb =**newStringBuilder**("");

**for**(PurchaseItem item: itemList){

Product product = productService.getProductById(item.getProductId());

**if**(product !=**null**)

sb.append(product.getName()+", "+

item.getQty()+" units @"+ item.getRate()+" = "

+ item.getPrice()+"<br>");

}

mapItems.put(purchase.getID(), sb.toString());

}

map.addAttribute("totalAmount", total);

map.addAttribute("list", list);

map.addAttribute("mapItems", mapItems);

map.addAttribute("mapUsers", mapUsers);

map.addAttribute("pageTitle","ADMIN PURCHASES REPORT");

**return**"admin/purchases";

}

**@RequestMapping(value = "/admindeletecat", method = RequestMethod.GET)**

**publicString** deleteCategory(ModelMap map,**@RequestParam(value="id", required=true)String** id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

long idValue =**Long**.parseLong(id);

**Category** category =**newCategory**();

**if**(idValue >0){

categoryService.deleteCategory(idValue);

}

**return**"redirect:admincategories";

}

**@RequestMapping(value = "/admineditcat", method = RequestMethod.GET)**

**public String** editCategory(ModelMap map,**@RequestParam(value="id", required=true)String** id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

long idValue =**Long**.parseLong(id);

**Category** category =**newCategory**();

**if**(idValue >0){

category = categoryService.getCategoryById(idValue);

}**else**{

category.setID(0);

}

map.addAttribute("category", category);

map.addAttribute("pageTitle","ADMIN EDIT PRODUCT CATEGORY");

**return**"admin/edit-category";

}

**@RequestMapping(value = "/admineditcataction", method = RequestMethod.POST)**

**publicString** updateCategory(ModelMap map,**@RequestParam(value="id", required=true)String** id,

**@RequestParam(value="name", required=true)String** name,

javax.servlet.http.HttpServletRequest request)

{

long idValue =**Long**.parseLong(id);

**if**(name ==**null**|| name.equals("")){

map.addAttribute("error","Error, A category name must be specified");

**return**"redirect:admineditcat?id="+id;

}

**Category** category =**newCategory**();

category.setID(idValue);

category.setName(name);

categoryService.updateCategory(category);

**return**"redirect:admincategories";

}

**@RequestMapping(value = "/admindeleteproduct", method = RequestMethod.GET)**

**publicString** deleteProduct(ModelMap map,**@RequestParam(value="id", required=true)String** id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

long idValue =**Long**.parseLong(id);

Product product =**new** Product();

**if**(idValue >0){

productService.deleteProduct(idValue);

}

**return**"redirect:adminproducts";

}

**@RequestMapping(value = "/admineditproduct", method = RequestMethod.GET)**

**public String** editProduct(ModelMap map,**@RequestParam(value="id", required=true)String** id,

javax.servlet.http.HttpServletRequest request)

{

// check if session is still alive

HttpSession session = request.getSession();

**if**(session.getAttribute("admin\_id")==**null**){

**return**"admin/login";

}

long idValue =**Long**.parseLong(id);

Product product =**new** Product();

**if**(idValue >0){

product = productService.getProductById(idValue);

}**else**{

product.setID(0);

product.setCategoryId(0);

}

**String** dropDown = categoryService.getCategoriesDropDown(product.getCategoryId());

map.addAttribute("product", product);

map.addAttribute("catDropdown", dropDown);

map.addAttribute("pageTitle","ADMIN EDIT PRODUCT");

**return**"admin/edit-product";

}

**@RequestMapping(value = "/admineditproductaction", method = RequestMethod.POST)**

**public String** updateProduct(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="id", required=true)String** id,

**@RequestParam(value="name", required=true)String** name,

**@RequestParam(value="price", required=true)String** price,

**@RequestParam(value="category", required=true)String** categoryId)

{

long idValue =**Long**.parseLong(id);

long categoryIdValue =**Long**.parseLong(categoryId);

**BigDecimal** priceValue =**newBigDecimal**(0.0);

**if**(name ==**null**|| name.equals("")){

map.addAttribute("error","Error, A product name must be specified");

**return**"redirect:admineditproduct?id="+id;

}

**try**{

priceValue =**newBigDecimal**(price);

}**catch**(**Exception** ex){

map.addAttribute("error","Error, Price is invalid");

**return**"redirect:admineditproduct?id="+id;

}

Product product =**new** Product();

product.setID(idValue);

product.setCategoryId(**Long**.parseLong(categoryId));

product.setName(name);

product.setPrice(priceValue);

productService.updateProduct(product);

**return**"redirect:adminproducts";

}

**@RequestMapping(value = "/adminlogout", method = RequestMethod.GET)**

**publicString** logout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

session.invalidate();

**return**"admin/login";

}

}

**1.24: Create a controller CartController to handle all cart-related pages**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **CartController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.ecommerce.service.**\***;

**import** com.ecommerce.entity.**\***;

**@Controller**

**public class** CartController {

**@Autowired**

**private** ProductService productService;

**@Autowired**

**private** PurchaseService purchaseService;

**@Autowired**

**private** PurchaseItemService purchaseItemService;

**@SuppressWarnings("unchecked")**

**@RequestMapping(value = "/cart", method = RequestMethod.GET)**

**public String** cart(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before adding items to cart");

}**else**{

// if cart is already in session then retrieve it else create a new cart list and

// save it to session

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

// get total of all cart items

**BigDecimal** totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle","SPORTY SHOES - YOUR CART");

**return**"cart";

}

**@SuppressWarnings("unchecked")**

**@RequestMapping(value = "/cartadditem", method = RequestMethod.GET)**

**publicString** cartAddItem(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="id", required=true)String** productId)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before adding items to cart");

}**else**{

long idValue =**Long**.parseLong(productId);

// if cart is already in session then retrieve it else create a new cart list and

// save it to session

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**if**(isItemInCart(cartItems, idValue)){

map.addAttribute("error","This item is already in your cart");

}**else**{

Product product = productService.getProductById(idValue);

CartItem item =**new** CartItem();

item.setProductId(idValue);

item.setQty(1);

item.setRate(product.getPrice());

**BigDecimal** dprice = item.getRate().multiply(**newBigDecimal**(item.getQty()));

item.setPrice(dprice);

item.setName(product.getName());

cartItems.add(item);

session.setAttribute("cart\_items", cartItems);

}

}

**return**"redirect:cart";

}

**@RequestMapping(value = "/cartdeleteitem", method = RequestMethod.GET)**

**public String** cartDeleteItem(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="id", required=true)String** id)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before deleting items from cart");

}**else**{

long idValue =**Long**.parseLong(id);

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**for**(CartItem item: cartItems){

**if**(item.getProductId()== idValue){

cartItems.remove(item);

session.setAttribute("cart\_items", cartItems);

**break**;

}

}

}

**return**"redirect:cart";

}

**@RequestMapping(value = "/checkout", method = RequestMethod.GET)**

**public String** checkout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before checking out");

}**else**{

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**BigDecimal** totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle","SPORTY SHOES - CHECKOUT");

**return**"checkout";

}

**@RequestMapping(value = "/completepurchase", method = RequestMethod.GET)**

**public String** completePurchase(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before completing purchase");

}**else**{

// take items from cart and update the databae

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**BigDecimal** totalValue = getCartValue(cartItems);

long userId =(**Long**) session.getAttribute("user\_id");

Purchase purchase =**new** Purchase();

purchase.setUserId(userId);

purchase.setDate(**Calendar**.getInstance().getTime());

purchase.setTotal(totalValue);

long purchaseId = purchaseService.updatePurchase(purchase);

**for**(CartItem item: cartItems){

PurchaseItem pItem =**new** PurchaseItem();

pItem.setPurchaseId(purchaseId);

pItem.setProductId(item.getProductId());

pItem.setUserId(userId);

pItem.setRate(item.getRate());

pItem.setQty(item.getQty());

pItem.setPrice(item.getPrice());

purchaseItemService.updateItem(pItem);

}

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

**return**"redirect:confirm";

}

**@RequestMapping(value = "/gateway", method = RequestMethod.GET)**

**public String** gateway(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before making payment");

}**else**{

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**BigDecimal** totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

map.addAttribute("cartItems", cartItems);

}

map.addAttribute("pageTitle","SPORTY SHOES - PAYMENT GATEWAY");

**return**"gateway";

}

**@RequestMapping(value = "/confirm", method = RequestMethod.GET)**

**public String** confirm(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

// check if user is logged in

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

map.addAttribute("error","Error, You need to login before completing the purchase");

}**else**{

// clear items from cart as order has completed

**List**<CartItem> cartItems =**newArrayList**<CartItem>();

**if**(session.getAttribute("cart\_items")!=**null**)

cartItems =(**List**<CartItem>) session.getAttribute("cart\_items");

**BigDecimal** totalValue = getCartValue(cartItems);

map.addAttribute("cartValue", totalValue);

cartItems.clear();

session.setAttribute("cart\_items",**null**);

}

map.addAttribute("pageTitle","SPORTY SHOES - PURCHASE CONFIRMATION");

**return**"confirm";

}

/\*\*

**\*** Check if an item is already in the cart

**\*@param** list

**\*@param** item

**\*@return**

\*/

**private** Boolean isItemInCart(**List**<CartItem> list,long item){

boolean retVal =**false**;

**for**(CartItem thisItem: list){

**if**(item == thisItem.getProductId()){

retVal =**true**;

**break**;

}

}

**return** retVal;

}

/\*\*

**\*** Get total value of items in cart

**\*@param** list

**\*@return**

\*/

**privateBigDecimal** getCartValue(**List**<CartItem> list){

**BigDecimal** total =**newBigDecimal**(0.0);

**for**(CartItem item: list){

**BigDecimal** dprice = item.getRate().multiply(**newBigDecimal**(item.getQty()));

total= total.add(dprice);

}

**return** total;

}

}

**1.25: Create a controller DashboardController to handle the member dashboard**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **DashboardController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**@Controller**

**public class** DashboardController {

**@RequestMapping(value = "/dashboard", method = RequestMethod.GET)**

**publicString** dashboard(ModelMap map)

{

map.addAttribute("pageTitle","SPORTY SHOES - DASHBOARD");

**return**"dashboard";

}

}

**1.26: Create a controller HomeController to handle the home page**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **HomeController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.ecommerce.entity.**\***;

**import** com.ecommerce.service.**\***;

**@Controller**

**public class** HomeController {

**@Autowired**

**private** CategoryService categoryService;

**@Autowired**

**private** ProductService productService;

**@RequestMapping(value = {"/", "/home"}, method = RequestMethod.GET)**

**public String** home(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

**List**<Product> list = productService.getAllProducts();

// use MAP to map the category names to product rows

**HashMap**<**Long**,**String**> mapCats =**newHashMap**<**Long**,**String**>();

**for**(Product product: list){

**Category** category = categoryService.getCategoryById(product.getCategoryId());

**if**(category !=**null**)

mapCats.put(product.getID(), category.getName());

}

map.addAttribute("list", list);

map.addAttribute("mapCats", mapCats);

map.addAttribute("pageTitle","SPORTY SHOES - HOMEPAGE");

**return**"index";

}

}

**1.27: Create a controller MemberController to handle all member pages**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **MemberController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.ecommerce.entity.**\***;

**import** com.ecommerce.service.**\***;

**@Controller**

**public class** MemberController {

**@Autowired**

**private** UserService userService;

**@RequestMapping(value = "/login", method = RequestMethod.GET)**

**public String** login(ModelMap map)

{

map.addAttribute("pageTitle","SPORTY SHOES - MEMBER LOGIN");

**return**"login";

}

**@RequestMapping(value = "/loginaction", method = RequestMethod.POST)**

**public String** loginAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="email\_id", required=true)String** emailId,

**@RequestParam(value="pwd", required=true)String** pwd)

{

User user = userService.authenticate(emailId, pwd);

**if**(user ==**null**){

map.addAttribute("error","Login failed");

**return**"login";

}

HttpSession session = request.getSession();

session.setAttribute("user\_id", user.getID());

**return**"redirect:dashboard";

}

**@RequestMapping(value = "/signup", method = RequestMethod.GET)**

**publicString** signup(ModelMap map)

{

map.addAttribute("pageTitle","SPORTY SHOES - MEMBER REGISTRATION");

**return**"register";

}

**@RequestMapping(value = "/signupaction", method = RequestMethod.POST)**

**public String** signupAction(ModelMap map, javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="email\_id", required=true)String** emailId,

**@RequestParam(value="pwd", required=true)String** pwd,

**@RequestParam(value="pwd2", required=true)String** pwd2,

**@RequestParam(value="fname", required=true)String** fname,

**@RequestParam(value="lname", required=true)String** lname,

**@RequestParam(value="age", required=true)String** age,

**@RequestParam(value="address", required=true)String** address)

{

**if**(emailId ==**null**|| emailId.equals("")){

map.addAttribute("error","Email id is required.");

**return**"register";

}

**if**(pwd ==**null**|| pwd2 ==**null**|| pwd.equals("")|| pwd2.equals("")){

map.addAttribute("error","Error , Incomplete passwords submitted.");

**return**"register";

}

**if**(!pwd.equals(pwd2)){

map.addAttribute("error","Error , Passwords do not match.");

**return**"register";

}

**if**(fname ==**null**|| fname.equals("")){

map.addAttribute("error","First name is required.");

**return**"register";

}

**if**(lname ==**null**|| lname.equals("")){

map.addAttribute("error","Last name is required.");

**return**"register";

}

**if**(age ==**null**|| age.equals("")){

age ="0";

}

User user = userService.getUserByEmailId(emailId);

**if**(user !=**null**){

map.addAttribute("error","This email id already exists.");

**return**"register";

}

user =**new** User();

user.setID(0);

user.setEmail(emailId);

user.setFname(fname);

user.setLname(lname);

user.setAge(**Integer**.parseInt(age));

user.setAddress(address);

user.setPwd(pwd);

userService.updateUser(user);

**return**"redirect:registerconfirm";

}

**@RequestMapping(value = "/registerconfirm", method = RequestMethod.GET)**

**public String** registerConfirm(ModelMap map)

{

**return**"register-confirm";

}

**@RequestMapping(value = "/editprofile", method = RequestMethod.GET)**

**public String** editProfile(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

**return**"login";

}

User user = userService.getUserById((**Long**) session.getAttribute("user\_id"));

map.addAttribute("pageTitle","SPORTY SHOES - MEMBER EDIT PROFILE");

map.addAttribute("user", user);

**return**"edit-profile";

}

**@RequestMapping(value = "/editprofileaction", method = RequestMethod.POST)**

**public String** editProfileAction(ModelMap map,

javax.servlet.http.HttpServletRequest request,

**@RequestParam(value="user\_id", required=true)String** userId,

**@RequestParam(value="pwd", required=true)String** pwd,

**@RequestParam(value="pwd2", required=true)String** pwd2,

**@RequestParam(value="fname", required=true)String** fname,

**@RequestParam(value="lname", required=true)String** lname,

**@RequestParam(value="age", required=true)String** age,

**@RequestParam(value="address", required=true)String** address)

{

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

**return**"login";

}

User user = userService.getUserById((**Long**) session.getAttribute("user\_id"));

map.addAttribute("user", user);

**if**(pwd ==**null**|| pwd2 ==**null**|| pwd.equals("")|| pwd2.equals("")){

map.addAttribute("error","Error , Incomplete passwords submitted.");

**return**"edit-profile";

}

**if**(!pwd.equals(pwd2)){

map.addAttribute("error","Error , Passwords do not match.");

**return**"edit-profile";

}

**if**(fname ==**null**|| fname.equals("")){

map.addAttribute("error","First name is required.");

**return**"edit-profile";

}

**if**(lname ==**null**|| lname.equals("")){

map.addAttribute("error","Last name is required.");

**return**"edit-profile";

}

**if**(age ==**null**|| age.equals("")){

age ="0";

}

user.setFname(fname);

user.setLname(lname);

user.setAge(**Integer**.parseInt(age));

user.setAddress(address);

user.setPwd(pwd);

userService.updateUser(user);

**return**"redirect:dashboard";

}

**@RequestMapping(value = "/logout", method = RequestMethod.GET)**

**public String** logout(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

session.invalidate();

**return**"redirect:home";

}

}

**1.28: Create a controller PurchaseController to handle member purchases page**

* In the Project Explorer, expand **SportyShoes->src->main->java**
* Right click on **src** and choose **New->Other**
* From the list of Wizards, choose **Class** and click on **Finish.**
* In **Package,** enter **com.ecommerce.controller** and in **Name,** enter **PurchaseController** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.controller;

**import** java.math.BigDecimal;

**import** java.util.HashMap;

**import** java.util.List;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.ecommerce.service.**\***;

**import** com.ecommerce.entity.**\***;

**@Controller**

**public class** PurchaseController {

**@Autowired**

**private** ProductService productService;

**@Autowired**

**private** PurchaseService purchaseService;

**@Autowired**

**private** PurchaseItemService purchaseItemService;

**@RequestMapping(value = "/memberpurchases", method = RequestMethod.GET)**

**public String** memberpurchases(ModelMap map, javax.servlet.http.HttpServletRequest request)

{

HttpSession session = request.getSession();

**if**(session.getAttribute("user\_id")==**null**){

**return**"login";

}

long userId =(**Long**) session.getAttribute("user\_id");

**List**<Purchase> list = purchaseService.getAllItemsByUserId(userId);

**BigDecimal** total =**new BigDecimal**(0.0);

// map purchase items to each purchase for display

**HashMap**<**Long**,**String**> mapItems =**new HashMap**<**Long**,**String**>();

**for**(Purchase purchase: list){

total = total.add(purchase.getTotal());

**List**<PurchaseItem> itemList = purchaseItemService.getAllItemsByPurchaseId(purchase.getID());

**StringBuilder** sb =**new StringBuilder**("");

**for**(PurchaseItem item: itemList){

Product product = productService.getProductById(item.getProductId());

**if**(product !=**null**){

sb.append(product.getName()+", "+

item.getQty()+" units @"+ item.getRate()+" = "

+ item.getPrice()+"<br>");

}

}

mapItems.put(purchase.getID(), sb.toString());

}

map.addAttribute("totalAmount", total);

map.addAttribute("list", list);

map.addAttribute("mapItems", mapItems);

map.addAttribute("pageTitle","SPORTY SHOES - YOUR ORDERS");

**return**"purchases";

}

}

**1.29: Create application.properties file to configure database access**

* In the Project Explorer, expand **SportyShoes->src->main**
* Right click on **resources** and choose **New->File**
* In filename, enter **application.properties** and click on **Finish.**
* Enter the following code:

spring.jpa.hibernate.use-new-id-generator-mappings=false

**1.30: Configure Hibernate with hibernate.cfg.xml**

* In the Project Explorer, expand **SportyShoes->src->main**
* Right click on **resources** and choose **New->File**
* In filename, enter **hibernate.cfg.xml** and click on **Finish.**
* Enter the following code:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.id.new\_generator\_mappings">false</property>

</session-factory>

</hibernate-configuration>

**1.31: Create a JSP page cart.jsp to display the user’s cart**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **cart.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Your Cart</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<%

if (request.getParameter("error") != null)

out.print(request.getParameter("error") + "<**br**>");

%>

<**br**><**br**>Total Cart Value: ${cartValue }<**br**>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Product</**b**></**td**>

<**td**><**b**>Qty</**b**></**td**>

<**td**><**b**>Rate</**b**></**td**>

<**td**><**b**>Price</**b**></**td**>

<**td**></**td**>

</**tr**>

<c:forEach items="${cartItems}" var="item">

<**tr**>

<**td**>${item.name }</**td**>

<**td**>${item.rate }</**td**>

<**td**>${item.qty}</**td**>

<**td**>${item.price}</**td**>

<**td**>

<**a** href="cartdeleteitem?id=${item.productId}">Remove</**a**>

</**td**>

</**tr**>

</c:forEach>

</**table**>

<**br**>

<c:if test ="${cartItems.size() > 0}">

<**a** href="checkout">Checkout Now</**a**>

</c:if>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.32: Create a JSP page checkout.jsp to do checkout**

* In the Project Explorer, expand the project **SportyShoes**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File.**
* Enter the filename as **checkout.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Checkout</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<**br**><**br**>

Your Total Order is worth ${cartValue}<**br**><**br**>

<**a** href="gateway">Pay via secure Payment Gateway</**a**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.33: Create a JSP page confirm.jsp to show checkout confirmation**

* In the Project Explorer, expand the project **SportyShoes**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **confirm.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Purchase Confirmation</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<**br**><**br**>

Your Order worth ${cartValue} has been completed successfully.<**br**><**br**>

Check your <**A** href="memberpurchases">Order History</**A**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.34: Create a JSP page dashboard.jsp to show member dashboard**

* In the Project Explorer, expand the project **SportyShoes**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **dashboard.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Dashboard</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**135: Create a JSP page edit-profile.jsp to let members edit their profiles**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **edit-profile.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Edit Profile</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

${error }

<**form** name=frmEdit action="editprofileaction" method="post">

<**input** type=hidden name=user\_id value="${user.ID}">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Password\*</**td**>

<**td**><**input** name=pwd maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Confirm Password\*</**td**>

<**td**><**input** name=pwd2 maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>First name\*</**td**>

<**td**><**input** name=fname maxlength=25 value="${user.fname }"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Last name\*</**td**>

<**td**><**input** name=lname maxlength=25 value="${user.lname }"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Age\*</**td**>

<**td**><**input** name=age maxlength=3 type="numeric" value="${user.age }"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Address</**td**>

<**td**><**input** name=address maxlength=100 value="${user.address }"></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Update</**button**><**br**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.36: Create a JSP page gateway.jsp to show a dummy payment gateway**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File.**
* Enter the filename as **gateway.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Make Payment</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<**br**><**br**>

Your card will be charged an amount of ${cartValue}<**br**><**br**>

<**a** href="completepurchase">Click to complete checkout</**a**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.37: Create a JSP page index.jsp to display the homepage**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File.**
* Enter the filename as **index.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Product</**b**></**td**>

<**td**><**b**>Price</**b**></**td**>

<**td**><**b**>Category</**b**></**td**>

<**td**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.name }</**td**>

<**td**>${item.price }</**td**>

<**td**>${mapCats.get(item.ID)}</**td**>

<**td**>

<**a** href="cartadditem?id=${item.ID}">Add To Cart</**a**>

</**td**>

</**tr**>

</c:forEach>

</**table**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.38: Create a JSP page login.jsp to do user login**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File.**
* Enter the filename as **login.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Login</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

${error}

<**form** name=frmLogin action="loginaction" method="post">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Email id\*</**td**>

<**td**><**input** name=email\_id maxlength=50></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Password\*</**td**>

<**td**><**input** name=pwd maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Login</**button**><**br**>

<**a** href="signup">Not a member? Signup</**a**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.39: Create a JSP page purchases.jsp to show member’s order history**

* In the Project Explorer, expand the project **SportyShoes**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **purchases.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Your Purchases</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

<**br**><**br**>Total ${list.size()} Orders: ${totalAmount }<**br**>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Order ID</**b**></**td**>

<**td**><**b**>Date</**b**></**td**>

<**td**><**b**>Total</**b**></**td**>

<**td**><**b**>Items</**b**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.ID }</**td**>

<**td**>${item.date }</**td**>

<**td**>${item.total}</**td**>

<**td**>

${mapItems.get(item.ID)}

</**td**>

</**tr**>

</c:forEach>

</**table**>

<**br**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.40: Create a JSP page register.jsp to do member registration**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **register.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Register</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

${error}

<**form** name=frmReg action="signupaction" method="post">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Email id\*</**td**>

<**td**><**input** name=email\_id type="email" maxlength=50></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Password\*</**td**>

<**td**><**input** name=pwd maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Confirm Password\*</**td**>

<**td**><**input** name=pwd2 maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>First name\*</**td**>

<**td**><**input** name=fname maxlength=25></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Last name\*</**td**>

<**td**><**input** name=lname maxlength=25></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Age\*</**td**>

<**td**><**input** name=age maxlength=3 type="number"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Address</**td**>

<**td**><**input** name=address maxlength=100></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Signup</**button**><**br**>

<**a** href="login">Already a member? Login</**a**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.41: Create a JSP register-confirm.jsp to show registration confirmation**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->JSP File**
* Enter the filename as **register-confirm.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Sporty Shoes - Registration Confirmation</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/topbar.jsp"></jsp:include>

Your registration is confirmed.<**a** href="login">Login to continue shopping</**a**><**br**>

<jsp:include page="/WEB-INF/view/components/footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.42: Create a JSP admin/categories.jsp to let admin manage product categories**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF-view**. Right click on **view**. Choose **New->Folder**
* Enter folder name as admin and click on **Finish.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **categories.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Setup Product Categories</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

Total Categories: ${list.size()} &nbsp;&nbsp; <**a** href="admineditcat?id=0">Add Category</**a**><**br**>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Category</**b**></**td**>

<**td**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.name }</**td**>

<**td**>

<**a** href="admineditcat?id=${item.ID}">Edit</**a**> | <**a** href="admindeletecat?id=${item.ID}">Delete</**a**>

</**td**>

</**tr**>

</c:forEach>

</**table**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.43: Create a JSP admin/change-password.jsp to let admin change his password**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as change-password.jsp and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Change Password</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

${error }

<**form** name=frmPwd method=post action="adminchangepwdaction">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Enter new password\*</**td**>

<**td**><**input** name=pwd maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Confirm new Password\*</**td**>

<**td**><**input** name=pwd2 maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Login</**button**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.44: Create a JSP admin/dashboard.jsp to show admin dashboard**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin** . Choose **New->JSP File.**
* Enter the filename as **dashboard.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin Dashboard</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

Welcome, Admin.<**br**>Choose an option from the menu above

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.45: Create a JSP admin/edit-category.jsp to let the admin add or edit a product category**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **edit-category.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Add/Edit Product Category</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

<%

if (request.getParameter("error") != null)

out.print(request.getParameter("error"));

%>

<**form** name=frmCat method=post action="admineditcataction">

<**input** type=hidden name=id value=${category.ID}>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Category name\*</**td**>

<**td**><**input** name=name maxlength=50 value="${category.name}"></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Save</**button**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.46: Create a JSP qdmin/edit-product.jsp to let admin add or edit a product.**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **edit-product.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Add/Edit Product</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

<%

if (request.getParameter("error") != null)

out.print(request.getParameter("error"));

%>

<**form** name=frmProduct method=post action="admineditproductaction">

<**input** type=hidden name=id value="${product.ID }">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Product name\*</**td**>

<**td**><**input** name=name maxlength=100 value="${product.name }"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Price\*</**td**>

<**td**><**input** name=price type="numeric" maxlength=6 value="${product.price }"></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Category\*</**td**>

<**td**>

<**select** name=category>

<**option** value="0">Select category</**option**>

${catDropdown}

</**select**>

</**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Save</**button**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.47: Create a JSP admin/login.jsp to do admin login**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **login.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin Login</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

${error}<**br**>

<**form** name=frmLogin action="adminloginaction" method="post">

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td** width=25%>Admin id\*</**td**>

<**td**><**input** name=admin\_id maxlength=20></**td**>

</**tr**>

<**tr**>

<**td** width=25%>Admin Password\*</**td**>

<**td**><**input** name=admin\_pwd maxlength=10 type="password"></**td**>

</**tr**>

<**tr**>

<**td** colspan=2>

<**button**>Login</**button**>

</**td**>

</**tr**>

</**table**>

</**form**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.48: Create a JSP admin/members.jsp to let the admin see the members list**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **members.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Browse Members</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

Total Members: ${list.size() }

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Name</**b**></**td**>

<**td**><**b**>Email</**b**></**td**>

<**td**><**b**>Signup Date</**b**></**td**>

<**td**><**b**>Age</**b**></**td**>

<**td**><**b**>Address</**b**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.fname}&nbsp;${item.lname }</**td**>

<**td**>${item.email}</**td**>

<**td**>${item.dateAdded}</**td**>

<**td**>${item.age}</**td**>

<**td**>${item.address}</**td**>

</**tr**>

</c:forEach>

</**table**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.49: Create a JSP admin/products.jsp to let the admin manage products**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **products.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin Setup Products</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

Total Products: ${list.size()} &nbsp;&nbsp; <**a** href="admineditproduct?id=0">Add Product</**a**><**br**>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Product</**b**></**td**>

<**td**><**b**>Price</**b**></**td**>

<**td**><**b**>Added On</**b**></**td**>

<**td**><**b**>Category</**b**></**td**>

<**td**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.name }</**td**>

<**td**>${item.price }</**td**>

<**td**>${item.dateAdded}</**td**>

<**td**>${mapCats.get(item.ID)}</**td**>

<**td**>

<**a** href="admineditproduct?id=${item.ID}">Edit</**a**> | <**a** href="admindeleteproduct?id=${item.ID}">Delete</**a**>

</**td**>

</**tr**>

</c:forEach>

</**table**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.50: Create a JSP page admin/purchases.jsp to see order history**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->admin**. Right click on **admin**. Choose **New->JSP File.**
* Enter the filename as **purchases.jsp** and click on **Finish.**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Admin - Purchases Report</**title**>

</**head**>

<**body**>

<jsp:include page="/WEB-INF/view/components/admin-header.jsp"></jsp:include>

<jsp:include page="/WEB-INF/view/components/admin-topbar.jsp"></jsp:include>

<**br**><**br**>Total ${list.size()} Orders: ${totalAmount }<**br**>

<**table** border=1 cellspacing=2 cellpadding=4>

<**tr**>

<**td**><**b**>Order ID</**b**></**td**>

<**td**><**b**>User</**b**>

<**td**><**b**>Date</**b**></**td**>

<**td**><**b**>Total</**b**></**td**>

<**td**><**b**>Items</**b**></**td**>

</**tr**>

<c:forEach items="${list}" var="item">

<**tr**>

<**td**>${item.ID }</**td**>

<**td**>

${mapUsers.get(item.ID)}

</**td**>

<**td**>${item.date }</**td**>

<**td**>${item.total}</**td**>

<**td**>

${mapItems.get(item.ID)}

</**td**>

</**tr**>

</c:forEach>

</**table**>

<jsp:include page="/WEB-INF/view/components/admin-footer.jsp"></jsp:include>

</**body**>

</**html**>

**1.51: Create a JSP page components/admin-footer.jsp to show footer in admin pages**

* In the Project Explorer, expand the project **SportyShoes.**
* **Expand src->main->webapp->WEB-INF->view.** Right click on **view.** Choose **New->Folder.**
* Enter folder name as **components** and click on **Finish.**
* Expand **src->main->webapp->WEB-INF->view->components**. Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **admin-footer.jsp** and click on **Finish.**
* Enter the following code:

<**hr**>

(C) Sporty Shoes 2019

**1.52: Create a JSP page components/admin-header.jsp to show header in admin pages**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->components**. Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **admin-header.jsp** and click on **Finish.**
* Enter the following code:

<%@ page isELIgnored="false" %>

<**h3**> ${pageTitle} </**h3**>

**1.53: Create a JSP components/admin-topbar.jsp to show menu options in admin pages**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->components**. Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **admin-topbar.jsp** and click on **Finish.**
* Enter the following code:

<**a** href="admindashboard">Dashboard</**a**> |

<**a** href="adminproducts">Setup Products</**a**> |

<**a** href="admincategories">Setup Product Categories</**a**> |

<**a** href="adminmembers">Browse Members</**a**> |

<**a** href="adminpurchases">Purchase Report</**a**> |

<**a** href="adminchangepassword">Change Password</**a**> |

<**a** href="adminlogout">Logout</**a**>

<**br**><**br**>

**1.54: Create a JSP components/footer.jsp to show footer in public pages**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->components**.
* Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **footer.jsp** and click on **Finish.**
* Enter the following code:

<**hr**>

(C) Sporty Shoes 2019

**1.55: Create a JSP components/header.jsp to show header in public pages**

* In the Project Explorer, expand the project **SportyShoes.**
* Expand **src->main->webapp->WEB-INF->view->components**. Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **header.jsp** and click on **Finish.**
* Enter the following code:

<%@ page isELIgnored="false" %>

<**h3**> ${pageTitle} </**h3**>

**1.56: Create a JSP components/topbar.jsp to show menu for logged in members**

* In the Project Explorer, expand the project **SportyShoes**
* Expand **src->main->webapp->WEB-INF->view->components**. Right click on **components**. Choose **New->JSP File.**
* Enter the filename as **topbar.jsp** and click on **Finish.**
* Enter the following code:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<%@ page isELIgnored="false" %>

<**a** href="home">Home</**a**> |

<% if (session.getAttribute("user\_id") == null ) { %>

<**a** href="login">Login/Signup</**a**> |

<% } else { %>

<**a** href="dashboard">Dashboard</**a**> |

<**a** href="cart">Cart</**a**> |

<**a** href="logout">Logout</**a**>

<**br**>

<**a** href="editprofile">Edit Profile</**a**> |

<**a** href="memberpurchases">Your Orders</**a**>

<% } %>

<**br**><**br**>

**1.57: Set up jdbc.properties for Hibernate**

* In the Project Explorer, expand **SportyShoes->src->main->webapp->WEB-INF**
* Right click on **WEB-INF** and choose **New->File.**
* In **Name,** enter **jdbc.properties** and click on **Finish.**
* Enter the following data:

jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.dialect=org.hibernate.dialect.MySQLDialect

jdbc.databaseurl=jdbc:mysql://127.0.0.1:3306/ecommerce

jdbc.username=root

jdbc.password=master

**1.58: Set up main-servlet.xml for Spring MVC**

* In the Project Explorer, expand **SportyShoes->src->main->webapp->WEB-INF**
* Right click on **WEB-INF** and choose **New->File.**
* In **Name,** enter **main-servlet.xml** and click on **Finish.**
* Enter the following data:

<?xml version="1.0" encoding="UTF-8"?>

<beans:beans xmlns="http://www.springframework.org/schema/mvc"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:beans="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">

<context:annotation-config/>

<context:component-scan base-package="com.ecommerce"/>

<beans:bean id="jspViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<beans:property name="viewClass"

value="org.springframework.web.servlet.view.JstlView"></beans:property>

<beans:property name="prefix" value="/WEB-INF/view/"></beans:property>

<beans:property name="suffix" value=".jsp"></beans:property>

</beans:bean>

<beans:bean id="messageSource"

class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<beans:property name="basename" value="classpath:messages"></beans:property>

<beans:property name="defaultEncoding" value="UTF-8"></beans:property>

</beans:bean>

<beans:bean id="propertyConfigurer"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<beans:property name="location" value="/WEB-INF/jdbc.properties"></beans:property>

</beans:bean>

<beans:bean id="dataSource"

class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">

<beans:property name="driverClassName" value="${jdbc.driverClassName}"></beans:property>

<beans:property name="url" value="${jdbc.databaseurl}"></beans:property>

<beans:property name="username" value="${jdbc.username}"></beans:property>

<beans:property name="password" value="${jdbc.password}"></beans:property>

</beans:bean>

<beans:bean id="sessionFactory"

class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<beans:property name="dataSource" ref="dataSource"></beans:property>

<beans:property name="packagesToScan" value="com.ecommerce.entity"></beans:property>

<beans:property name="configLocation">

<beans:value>classpath:hibernate.cfg.xml</beans:value>

</beans:property>

<beans:property name="hibernateProperties">

<beans:props>

<beans:prop key="hibernate.dialect">${jdbc.dialect}</beans:prop>

<beans:prop key="hibernate.show\_sql">true</beans:prop>

</beans:props>

</beans:property>

</beans:bean>

<tx:annotation-driven/>

<beans:bean id="transactionManager"

class="org.springframework.orm.hibernate5.HibernateTransactionManager">

<beans:property name="sessionFactory" ref="sessionFactory"></beans:property>

</beans:bean>

</beans:beans>

**1.59: Configure web.xml**

* In the Project Explorer, expand **SportyShoes->WebContent->WEB-INF**
* Double click on **web.xml** to open it in the editor.
* Enter the following script:

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd">

<web-app>

<display-name>Archetype Created Web Application</display-name>

<welcome-file-list>

<welcome-file>/WEB-INF/view/index.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>main</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/main-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>main</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/main-servlet.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

**1.60: Build the project**

* From the **Project** menu at the top, click on **Build.**
* If any compile errors are shown, fix them as required.

**1.61: Publish and start the project**

* If you do not see the **Servers** tab near the bottom of the IDE, go to **Window** menu and click on **Show View->Servers.**
* Right click on the **Server** entry and choose **Add and Remove.**
* Click the **Add** button to move **SportyShoes** from the **Available** list to the **Configured** list.
* Click on **Finish.**
* Right click the **Server** entry and click on **Publish.**
* Right click the **Server** entry and click on **Start.**
* This will start the server.

**1.62: Run the project**

* To run the project, open a web browser and type [**http://localhost:8080/SportyShoes**](http://localhost:8080/Section1LEP)

**1.63: Push the code to your GitHub repositories**

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master

**Step 2: Do Unit Testing with JUnit**

1. Configure JUnit 5 in Eclipse and add it to the Project.
2. Create AdminTest class to test entity class Admin.
3. Create CartItemTest class to test entity class CartItem.
4. Create CategoryTest class to test entity class Category.
5. Create ProductTest class to test entity class Product.
6. Create PurchaseItemTest class to test entity class PurchaseItem.
7. Create PurchaseTest class to test entity class Purchase.
8. Create UserTest class to test entity class User.
9. Create AdminServiceTest class to test service class AdminService.
10. Create CategoryServiceTest class to test service class CategoryService.
11. Create ProductServiceTest class to test service class ProductService.
12. Create PurchaseItemServiceTest class to test service class PurchaseItemService.
13. Create PurchaseServiceTest class to test service class PurchaseService.
14. Create UserServiceTest class to test service class UserService.
15. Run the JUnit test classes.

**2.1: Configure JUnit 5 in Eclipse and add it to the Project**

* Eclipse Oxygen 1a onwards, JUnit is already a part of its deployment. So there is no need to set up JUnit 5 separately. The steps below are for the older version of Eclipse:
  + Create a folder **Junit5** anywhere in the system.
  + From the terminal, go to that folder and clone the JUnit repo from git with the command **git clone** [**https://github.com/noopur2507/eclipse-junit5.git**](https://github.com/noopur2507/eclipse-junit5.git)
  + Copy all the extracted jar files to the **dropins** folder of your Eclipse installation.
  + Create another folder **eclipse\_git** anywhere in the system.
  + From the terminal, go to that folder and clone the following git repo: **git clone** [git://git.eclipse.org/gitroot/jdt/eclipse.jdt.ui.git](about:blank)
  + Checkout the BETA\_JUNIT5 branch.
  + In Eclipse, import the following projects into your workspace: **org.eclipse.jdt.ui, org.eclipse.jdt.junit, org.eclipse.jdt.junit.core, org.eclipse.jdt.junit.runtime, org.eclipse.jdt.junit4.runtime, org.eclipse.jdt.junit5.runtime.**
  + Restart Eclipse. JUnit 5 should work with all your projects now.

**2.2: Create AdminTest class to test entity class *Admin***

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **AdminTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**class** AdminTest {

**@BeforeAll**

**static**void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

int id =1;

**String** adminId ="admin";

**String** pwd ="password";

Admin admin =**new** Admin();

admin.setID(id);

admin.setAdminId(adminId);

admin.setAdminPwd(pwd);

assertEquals(admin.getID(), id);

assertEquals(admin.getAdminId(), adminId);

assertEquals(admin.getAdminPwd(), pwd);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.3: Create CartItemTest class to test entity class CartItem**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other.**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **CartItemTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**class** CartItemTest {

**@BeforeAll**

**static** void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long productId =1;

**String** name ="product name";

**BigDecimal** rate=**newBigDecimal**(10.00);

**BigDecimal** price =**newBigDecimal**(20.00);

int qty =2;

CartItem cartItem =**new** CartItem();

cartItem.setProductId(productId);

cartItem.setName(name);

cartItem.setRate(rate);

cartItem.setPrice(price);

cartItem.setQty(qty);

assertEquals(cartItem.getProductId(), productId);

assertEquals(cartItem.getName(), name);

assertEquals(cartItem.getRate(), rate);

assertEquals(cartItem.getPrice(), price);

assertEquals(cartItem.getQty(), qty);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.4: Create CategoryTest class to test entity class Category**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **CategoryTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**class** CategoryTest {

**@BeforeAll**

**static** void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long id =1;

**String** name ="category name";

**Category** category =**new Category**();

category.setID(id);

category.setName(name);

assertEquals(category.getID(), id);

assertEquals(category.getName(), name);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.5: Create ProductTest class to test entity class Product**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **ProductTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**class** ProductTest {

**@BeforeAll**

**static**void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long ID =1;

**String** name ="product name";

**BigDecimal** price =**newBigDecimal**(10.00);

**Date** dateAdded =**Calendar**.getInstance().getTime();

long categoryId =1;

Product product =**new** Product();

product.setID(ID);

product.setName(name);

product.setPrice(price);

product.setDateAdded(dateAdded);

product.setCategoryId(categoryId);

assertEquals(product.getID(), ID);

assertEquals(product.getName(), name);

assertEquals(product.getPrice(), price);

assertEquals(product.getDateAdded(), dateAdded);

assertEquals(product.getCategoryId(), categoryId);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.6: Create PurchaseItemTest class to test entity class PurchaseItem**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **PurchaseItemTest** and click on **Finish.**
* Enter the following code:

**package com.ecommerce.entity;**

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**class** PurchaseItemTest {

**@BeforeAll**

**static**void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long ID =1;

long userId =10;

long purchaseId =1;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setID(ID);

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

assertEquals(pi.getID(), ID);

assertEquals(pi.getUserId(), userId);

assertEquals(pi.getPurchaseId(), purchaseId);

assertEquals(pi.getProductId(), productId);

assertEquals(pi.getRate(), rate);

assertEquals(pi.getQty(), qty);

assertEquals(pi.getPrice(), price);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.7: Create PurchaseTest class to test entity class Purchase**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **PurchaseTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**class** PurchaseTest {

**@BeforeAll**

**static**void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long ID =1;

long userId =10;

**BigDecimal** grossTotal =**newBigDecimal**(10.00);

**Date** date =**Calendar**.getInstance().getTime();

Purchase purchase =**new** Purchase();

purchase.setID(ID);

purchase.setUserId(userId);

purchase.setTotal(grossTotal);

purchase.setDate(date);

assertEquals(purchase.getID(), ID);

assertEquals(purchase.getUserId(), userId);

assertEquals(purchase.getTotal(), grossTotal);

assertEquals(purchase.getDate(), date);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.8: Create UserTest class to test entity class User**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.entity** and in **Name,** enter **UserTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.entity;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**import** javax.persistence.Column;

**class** UserTest {

**@BeforeAll**

**static**void beforeAll(){

}

**@BeforeEach**

void beforeEach(){

}

**@DisplayName("Test set and get methods")**

**@Test**

void testValues(){

long ID =1;

**String** fname ="fname";

**String** lname ="lname";

**String** address ="some address";

int age =20;

**Date** dateAdded =**Calendar**.getInstance().getTime();

**String** emailId ="email@email.com";

**String** pwd ="password";

User user =**new** User();

user.setID(ID);

user.setFname(fname);

user.setLname(lname);

user.setAddress(address);

user.setAge(age);

user.setDateAdded(dateAdded);

user.setEmail(emailId);

user.setPwd(pwd);

assertEquals(user.getID(), ID);

assertEquals(user.getFname(), fname);

assertEquals(user.getLname(), lname);

assertEquals(user.getAddress(), address);

assertEquals(user.getAge(), age);

assertEquals(user.getDateAdded(), dateAdded);

assertEquals(user.getEmail(), emailId);

assertEquals(user.getPwd(), pwd);

}

**@AfterEach**

void afterEach(){

}

**@AfterAll**

**static**void afterAll(){

}

}

**2.9: Create AdminServiceTest class to test service class AdminService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **AdminServiceTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.Admin;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** AdminServiceTest {

**@Autowired**

**private** AdminService adminService;

**@DisplayName("Login Authentication")**

**@Test**

void authenticateTest(){

**String** adminId ="admin";

**String** pwd ="aaaaaa";

Admin admin = adminService.authenticate(adminId, pwd);

assertNotNull(admin);

assertTrue(admin.getID()>0);

}

**@DisplayName("Get Admin by Id")**

**@Test**

void getAdminByIdTest(){

**String** adminId ="admin";

**String** pwd ="aaaaaa";

Admin admin = adminService.authenticate(adminId, pwd);

assertNotNull(admin);

assertTrue(admin.getID()>0);

long ID = admin.getID();

Admin newAdmin = adminService.getAdminById(ID);

assertNotNull(newAdmin);

assertEquals(admin.getID(), ID);

}

**@DisplayName("Update Password")**

**@Test**

void updatePwdTest(){

**String** adminId ="admin";

**String** pwd ="aaaaaa";

Admin admin = adminService.authenticate(adminId, pwd);

assertNotNull(admin);

long ID = admin.getID();

admin.setAdminPwd(pwd);

adminService.updatePwd(admin);

Admin newAdmin = adminService.getAdminById(ID);

assertNotNull(newAdmin);

assertEquals(admin.getAdminPwd(), pwd);

}

}

**2.10: Create CategoryServiceTest class to test service class CategoryService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **CategoryServiceTest**and click on **Finish**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.Category;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** CategoryServiceTest {

**@Autowired**

**private** CategoryService categoryService;

**@DisplayName("Get Category By Id")**

**@Test**

void getCategoryByIdTest(){

long id =1;

**Category** cat = categoryService.getCategoryById(id);

assertNotNull(cat);

assertEquals(cat.getID(), id);

}

**@DisplayName("Update Category")**

**@Test**

void updateCategoryTest(){

**String** name =**String**.valueOf(**System**.currentTimeMillis());

**Category** cat =**newCategory**();

cat.setName(name);

categoryService.updateCategory(cat);

**List**<**Category**> list = categoryService.getAllCategories();

long newId =0;

**for**(**Category** c: list ){

**if**(c.getName().equals(name)){

newId = c.getID();

**break**;

}

}

assertTrue(newId >0);

categoryService.deleteCategory(newId);

}

**@DisplayName("Delete Category")**

**@Test**

void deleteCategoryTest(){

**String** name =**String**.valueOf(**System**.currentTimeMillis());

**Category** cat =**newCategory**();

cat.setName(name);

categoryService.updateCategory(cat);

**List**<**Category**> list = categoryService.getAllCategories();

long newId =0;

**for**(**Category** c: list ){

**if**(c.getName().equals(name)){

newId = c.getID();

**break**;

}

}

assertTrue(newId >0);

categoryService.deleteCategory(newId);

list = categoryService.getAllCategories();

boolean wasFound =**false**;

**for**(**Category** c: list ){

**if**(c.getID()== newId){

wasFound =**true**;

**break**;

}

}

assertFalse(wasFound);

}

**@DisplayName("Get all Categories")**

**@Test**

void getAllCategoriesTest(){

**List**<**Category**> list = categoryService.getAllCategories();

assertTrue(list.size()>0);

}

**@DisplayName("Get Categories Dropdown")**

**@Test**

void getCategoriesDropDownTest(){

**String** dropdown = categoryService.getCategoriesDropDown(0);

assertTrue(dropdown.indexOf("<option value=")>-1);

}

}

**2.11: Create ProductServiceTest class to test service class ProductService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **ProductServiceTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.Category;

**import** com.ecommerce.entity.Product;

**import** org.junit.jupiter.api.Test;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** ProductServiceTest {

**@Autowired**

**private** ProductService productService;

**@DisplayName("Get Product By Id")**

**@Test**

void getProductByIdTest(){

long id =2;

Product product = productService.getProductById(id);

assertNotNull(product);

assertEquals(product.getID(), id);

}

**@DisplayName("Update Product")**

**@Test**

void updateProductTest(){

**String** name ="product name";

**BigDecimal** price =**newBigDecimal**(10.00);

**Date** dateAdded =**Calendar**.getInstance().getTime();

long categoryId =1;

Product product =**new** Product();

product.setName(name);

product.setPrice(price);

product.setDateAdded(dateAdded);

product.setCategoryId(categoryId);

productService.updateProduct(product);

**List**<Product> list = productService.getAllProducts();

long newId =0;

**for**(Product p: list ){

**if**(p.getName().equals(name)){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

productService.deleteProduct(newId);

}

**@DisplayName("Delete Product")**

**@Test**

void deleteProductTest(){

**String** name ="product name";

**BigDecimal** price =**newBigDecimal**(10.00);

**Date** dateAdded =**Calendar**.getInstance().getTime();

long categoryId =1;

Product product =**new** Product();

product.setName(name);

product.setPrice(price);

product.setDateAdded(dateAdded);

product.setCategoryId(categoryId);

productService.updateProduct(product);

**List**<Product> list = productService.getAllProducts();

long newId =0;

**for**(Product p: list ){

**if**(p.getName().equals(name)){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

productService.deleteProduct(newId);

list = productService.getAllProducts();

boolean wasFound =**false**;

**for**(Product p: list ){

**if**(p.getID()== newId){

wasFound =**true**;

**break**;

}

}

assertFalse(wasFound);

}

**@DisplayName("Get all Products")**

**@Test**

void getAllProductsTest(){

**List**<Product> list = productService.getAllProducts();

assertTrue(list.size()>0);

}

}

**2.12: Create PurchaseItemServiceTest class to test service class PurchaseItemService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **PurchaseItemServiceTest**and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.Purchase;

**import** com.ecommerce.entity.Product;

**import** com.ecommerce.entity.PurchaseItem;

**import** org.junit.jupiter.api.Test;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** PurchaseItemServiceTest {

**@Autowired**

**private** PurchaseItemService purchaseItemService;

**@DisplayName("Get Purchase Item By Id")**

**@Test**

void getItembyIdTest(){

long userId =10;

long purchaseId =1;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

purchaseItemService.updateItem(pi);

**List**<PurchaseItem> list = purchaseItemService.getAllItemsByPurchaseId(1);

long newId =0;

**for**(PurchaseItem p: list){

**if**(p.getUserId()== userId && p.getQty()==2){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

pi = purchaseItemService.getItemById(newId);

assertTrue(pi !=**null**);

assertEquals(pi.getID(), newId);

purchaseItemService.deleteItem(newId);

}

**@DisplayName("Get All Items By Purchase Id")**

**@Test**

void getAllItemsByPurchaseItemTest(){

long userId =10;

long purchaseId =1;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

purchaseItemService.updateItem(pi);

**List**<PurchaseItem> list = purchaseItemService.getAllItemsByPurchaseId(1);

assertTrue(list.size()>0);

long newId =0;

**for**(PurchaseItem p: list){

**if**(p.getUserId()== userId && p.getQty()==2){

newId = p.getID();

**break**;

}

}

purchaseItemService.deleteItem(newId);

}

**@DisplayName("Update Purchase Item")**

**@Test**

void updateItemTest(){

long userId =10;

long purchaseId =111;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

purchaseItemService.updateItem(pi);

**List**<PurchaseItem> list = purchaseItemService.getAllItemsByPurchaseId(purchaseId);

long newId =0;

int newQty =100;

**for**(PurchaseItem p: list){

**if**(p.getUserId()== userId && p.getQty()==2){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

purchaseItemService.deleteItem(newId);

}

**@DisplayName("Delete Purchase Item")**

**@Test**

void deleteItemTest(){

long userId =10;

long purchaseId =1;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

purchaseItemService.updateItem(pi);

**List**<PurchaseItem> list = purchaseItemService.getAllItemsByPurchaseId(purchaseId);

long newId =0;

int newQty =100;

**for**(PurchaseItem p: list){

**if**(p.getUserId()== userId && p.getQty()==2){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

purchaseItemService.deleteItem(newId);

list = purchaseItemService.getAllItemsByPurchaseId(purchaseId);

boolean wasFound =**false**;

**for**(PurchaseItem p: list){

**if**(p.getID()== newId){

wasFound =**true**;

**break**;

}

}

assertFalse(wasFound);

}

**@DisplayName("Delete All Purchase Items for a Purchase Id")**

**@Test**

void deleteAllsItemForPurchaseIdTest(){

long userId =10;

long purchaseId =1;

long productId =1;

**BigDecimal** rate =**newBigDecimal**(10.00);

int qty =2;

**BigDecimal** price =**newBigDecimal**(20.00);

PurchaseItem pi =**new** PurchaseItem();

pi.setUserId(userId);

pi.setPurchaseId(purchaseId);

pi.setProductId(productId);

pi.setPrice(price);

pi.setRate(rate);

pi.setQty(qty);

purchaseItemService.updateItem(pi);

**List**<PurchaseItem> list = purchaseItemService.getAllItemsByPurchaseId(1);

long newId =0;

int newQty =100;

**for**(PurchaseItem p: list){

**if**(p.getUserId()== userId && p.getQty()==2){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

purchaseItemService.deleteAllItemsForPurchaseId(purchaseId);

list = purchaseItemService.getAllItemsByPurchaseId(1);

assertTrue(list.size()==0);

}

}

**2.13: Create PurchaseServiceTest class to test service class PurchaseService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **PurchaseServiceTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.Purchase;

**import** com.ecommerce.entity.Product;

**import** com.ecommerce.entity.PurchaseItem;

**import** org.junit.jupiter.api.Test;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** PurchaseServiceTest {

**@Autowired**

**private** PurchaseItemService purchaseItemService;

**@Autowired**

**private** PurchaseService purchaseService;

**@DisplayName("Get Purchase By Id")**

**@Test**

void getPurchaseByIdTest(){

long userId =10;

**BigDecimal** grossTotal =**newBigDecimal**(10.00);

**Date** date =**Calendar**.getInstance().getTime();

Purchase purchase =**new** Purchase();

purchase.setUserId(userId);

purchase.setTotal(grossTotal);

purchase.setDate(date);

purchaseService.updatePurchase(purchase);

**List**<Purchase> list = purchaseService.getAllItemsByUserId(userId);

long newId =0;

**for**(Purchase p: list){

**if**(p.getUserId()== userId && p.getTotal().compareTo(grossTotal)==0){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

purchase = purchaseService.getPurchaseById(newId);

assertTrue(purchase !=**null**);

assertEquals(purchase.getID(), newId);

}

**@DisplayName("Get All Purchases")**

**@Test**

void getAllItemsTest(){

**List**<Purchase> list = purchaseService.getAllItems();

assertTrue(list.size()>0);

}

**@DisplayName("Get All Purchases By User Id")**

**@Test**

void getAllItemsByUserIdTest(){

long userId =10;

**BigDecimal** grossTotal =**newBigDecimal**(10.00);

**Date** date =**Calendar**.getInstance().getTime();

Purchase purchase =**new** Purchase();

purchase.setUserId(userId);

purchase.setTotal(grossTotal);

purchase.setDate(date);

purchaseService.updatePurchase(purchase);

**List**<Purchase> list = purchaseService.getAllItemsByUserId(userId);

assertTrue(list.size()>0);

}

**@DisplayName("Update Purchase")**

**@Test**

void updatePurchaseTest(){

long userId =10;

**BigDecimal** grossTotal =**newBigDecimal**(10.00);

**Date** date =**Calendar**.getInstance().getTime();

Purchase purchase =**new** Purchase();

purchase.setUserId(userId);

purchase.setTotal(grossTotal);

purchase.setDate(date);

purchaseService.updatePurchase(purchase);

**List**<Purchase> list = purchaseService.getAllItemsByUserId(userId);

long newId =0;

**for**(Purchase p: list){

**if**(p.getUserId()== userId && p.getTotal().compareTo(grossTotal)==0){

newId = p.getID();

**break**;

}

}

assertTrue(newId >0);

purchase = purchaseService.getPurchaseById(newId);

assertTrue(purchase !=**null**);

assertEquals(purchase.getID(), newId);

}

}

**2.14: Create UserServiceTest class to test service class UserService**

* In the Project Explorer, expand **SportyShoes->src->test->java**
* Right click on **java** and click **New->Other**
* From the list of Wizards, choose **JUnit->Test Case** and click on **Next.**
* In **Package,** enter **com.ecommerce.service** and in **Name,** enter **UserServiceTest** and click on **Finish.**
* Enter the following code:

**package** com.ecommerce.service;

**importstatic** org.junit.jupiter.api.Assertions.**\***;

**import** java.math.BigDecimal;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.junit.runner.RunWith;

**import** org.junit.jupiter.api.AfterAll;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.BeforeAll;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Disabled;

**import** org.junit.jupiter.api.DisplayName;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit.jupiter.SpringExtension;

**import** com.ecommerce.entity.User;

**import** org.junit.jupiter.api.Test;

**@ExtendWith(SpringExtension.class)**

**@ContextConfiguration(locations = {"classpath:configuration.xml"})**

**class** UserServiceTest {

**@Autowired**

**private** UserService userService;

**@DisplayName("Login Authentication")**

**@Test**

void authenticateTest(){

**String** emailId ="email@email.com";

**String** pwd ="password";

User user = userService.authenticate(emailId, pwd);

assertTrue(user !=**null**);

assertEquals(user.getEmail(), emailId);

}

**@DisplayName("Get User By Id")**

**@Test**

void getUserByIdTest(){

**String** emailId ="email@email.com";

**String** pwd ="password";

User user = userService.authenticate(emailId, pwd);

assertTrue(user !=**null**);

long id = user.getID();

user = userService.getUserById(id);

assertTrue(user !=**null**);

assertEquals(user.getID(), id);

}

**@DisplayName("Get User By EmailId")**

**@Test**

void getUserByEmailIdTest(){

**String** emailId ="email@email.com";

User user = userService.getUserByEmailId(emailId);

assertTrue(user !=**null**);

assertEquals(user.getEmail(), emailId);

}

**@DisplayName("Update User")**

**@Test**

void updateUserTest(){

**String** fname ="fname";

**String** lname ="lname";

**String** address ="some address";

int age =20;

**Date** dateAdded =**Calendar**.getInstance().getTime();

**String** emailId ="email@email.com";

**String** pwd ="password";

User user =**new** User();

user.setFname(fname);

user.setLname(lname);

user.setAddress(address);

user.setAge(age);

user.setDateAdded(dateAdded);

user.setEmail(emailId);

user.setPwd(pwd);

userService.updateUser(user);

user = userService.getUserByEmailId(emailId);

assertTrue(user !=**null**);

assertEquals(user.getEmail(), emailId);

}

**@DisplayName("Get All Usersd")**

**@Test**

void getAllUsersTest(){

**List**<User> list = userService.getAllUsers();

assertTrue(list.size()>0);

}

}

**2.15: Run the JUnit test classes**

In the Project Explorer, expand **SportyShoes.**

* Right click on **SportyShoes** and click **Run As->JUnit Test Case**
* This will run all the JUnit tests and show the results in the JUnit console in Eclipse.

**Step 3: Host the website on AWS EC2 cloud**

1. Log in to your AWS account.
2. Create and configure an EC2 instance.
3. Set up Apache tomcat webserver.
4. Set up the MySQL database on Amazon RDS service.
5. Deploy the website on the EC2 instance.

**3.1: Log in to AWS account**

* Go to <https://aws.amazon.com/console/> and click on **Sign in To The Console.**
* In the **Find Services** search box, type EC2 and click on the matching EC2 item from the dropdown list.

**3.2: Create and configure an EC2 instance**

* Click on **Launch Instance.**
* From the list of AMI items, choose **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type**, and 64-bit (x86) and click on **Select**
* In the next screen, **Choose an Instance Type and** choose **General Purpose – t2.micro** from the list of instances.
* Click on **Next.**
* For **Configure Security Group**, it will show one item in the list for SSH. For SSH change the source to **Anywhere.**
* Click on **Review And Launch.**
* In the next screen, click on **Launch.**
* In the **Select Key Pair** popup, choose **Create A New Key Pair** from the dropdown.
* In **Key Pair Name** enter **SportyShoes** and click **Download Key pair.**
* A file called SportyShoes.pem will be downloaded into your local system.
* In your local system, using the terminal, go to the folder where the file was downloaded and change its access permissions: sudo **chmod 400 sportyshoes.pem**
* Click on **Launch Instance.**
* In the EC2 Console, wait until the **Instance State** changes to **running.**
* In the Description tab, make a note of the Public DNS Ipv4 value displayed. This is the url of the EC2 instance.
* In the instance row displayed, go to the **Security Groups** column and click on the Security Group displayed.
* In the **Security Group** screen, go to the **Inbound** tab and make sure the following rules are in place after clicking on **Edit:**

Custom TCP Rule, TCP, 8080, 0.0.0./0

SSH, TCP, 22, 0.0.0./0

Custom UDP Rule, UDP, 8080, ::/0

* Go back to the EC2 Console and select the instance in the list.
* Click on **Connect.**
* In the pop-up, copy the ssh command displayed in the form: **ssh -i "sportyshoes.pem" ubuntu@ecxxx.amazonaws.com.**
* In your local system, using the terminal, go to the folder where SportyShoes.pem is saved, and paste the ssh command there and press Enter.
* You are now able to access your EC2 instance. Type **exit** to log out of it.

**3.3: Setup Apache tomcat webserver**

* Log in via SSH into the EC2 instance.
* Type **sudo apt update** to update packages.
* Type **sudo apt install default-jdk** to install JDK.
* Type **sudo useradd -r -m -U -d /opt/tomcat -s /bin/false tomcat** to set up a tomcat user.
* Download tomcat by typing **wget** **http://www-eu.apache.org/dist/tomcat/tomcat-9/v9.0.27/bin/apache-tomcat-9.0.27.tar.gz -P /tmp**
* After downloading extract the files: **sudo tar xf /tmp/apache-tomcat-9\*.tar.gz -C /opt/tomcat**
* Set the group ownership of the tomcat directory: **sudo chown -RH tomcat: /opt/tomcat/latest**
* Give execute permissions to all the files in the bin folder: **sudo sh -c 'chmod +x /opt/tomcat/latest/bin/\*.sh'**
* Set up tomcat as a service by creating a systemd file: **sudo nano /etc/systemd/system/tomcat.service**
* Put the following configuration data into the file:

[Unit]

Description=Tomcat 9 servlet container

After=network.target

[Service]

Type=forking

User=tomcat

Group=tomcat

Environment="JAVA\_HOME=/usr/lib/jvm/default-java"

Environment="JAVA\_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true"

Environment="CATALINA\_BASE=/opt/tomcat/latest"

Environment="CATALINA\_HOME=/opt/tomcat/latest"

Environment="CATALINA\_PID=/opt/tomcat/latest/temp/tomcat.pid"

Environment="CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/latest/bin/startup.sh

ExecStop=/opt/tomcat/latest/bin/shutdown.sh

[Install]

WantedBy=multi-user.target

* Change the value of JAVA\_HOME if the location is different in your EC2 instance.
* Reload the system daemon services: **sudo systemctl daemon-reload**
* Start the tomcat service: **sudo systemctl start tomcat**
* Check whether tomcat is running by typing **sudo systemctl status tomcat**
* Enable tomcat to auto-start on boot: **sudo systemctl enable tomcat**
* Exit the SSH session.
* To test the EC2 instance now, open a browser and type in http://<your ec2 instance IP4 name>:8080
* You should see the Apache Tomcat homepage.

**3.4: Set up the MySQL database on Amazon RDS service**

* Go to the AWS Console and search for RDS in the Services search box. Click on **Amazon RDS.**
* In the RDS Console, click on **Create Database.**
* Select Creation Method as **Standard Create,** Engine Options as **MySQL,** and **Templates** as Dev/Test.
* Under **Settings->Credential Settings,** choose a Master username and Master Password.
* Under **Network and Security,** make sure **Public Accessibility** is set to **Yes.**
* Click on **Create Database** at the bottom of the page.
* In the RDS Console, wait till the Status of the RDS instance becomes Available.
* Click on the RDS instance name to see the RDS detailed info.
* Under **Connectivity and Security** tab, copy the value of **Endpoint.**
* From your system, you should now be able to access the Mysql server with the MySQL command line tool: mysql -h <your rds endpoint> --user=<your rds username> --password=<your rds password>
* Transfer the ecommerce database from your local MySQL server to RDS by exporting and importing the sql dump.
* Change the settings in jdbc.properties of the SportyShoes webapp to point to the RDS instance.
* Your webapp should now run with the database hosted in Amazon RDS.

**3.5: Deploy the website on the EC2 instance**

* In Eclipse Project Explorer, right click on **SportyShoes** and choose **Export->WAR file.**
* Upload the WAR file to a publicly accessible repo like Dropbox or Google Drive. Get a shareable link to the file.
* Log in to your EC2 instance using SSH. Make sure **wget** is installed by typing: **sudo apt install wget**
* Go to the webapps folder in the Tomcat installation and download the war file using **wget <your shareable link to the war file>**
* Exit the ec2 instance.
* In your browser enter the url **http://<your ec2 instance url>:8080/SportyShoes**
* You should see the Sporty Shoes homepage now.

**Step 4: Do website end-user testing using Selenium WebDriver**

1. Install Selenium for Java and WebDriver for the Firefox browser.
2. Create a Selenium Project in Eclipse.
3. Create Cart.java to test cart functions.
4. Create Checkout.java to test cart checkout flow.
5. Create Dashboard.java to check logging into dashboard.
6. Create EditProfile.java to check edit profile function.
7. Create Globals.java to create global variables for use in the classes.
8. Create Homepage.java to test homepage loading.
9. Create Login.java to test login features.
10. Create OrderHistory.java to test display of Order History.
11. Create Signup.java to test signup feature.
12. Run all the Selenium tests.

**4.1: Install Selenium for Java and WebDriver for the Firefox browser**

* Go to <https://selenium.dev/downloads/>
* Click on **Download** for the Java language driver.
* Unzip the contents of the file into a folder in your local system.
* Scroll down and expand the Browsers section.
* Go to <https://github.com/mozilla/geckodriver/releases> and download the latest tar.gz of geckodriver.
* Unzip the contents of the file into a folder in your local system.

**4.2: Create a Selenium Project in Eclipse**

* In Eclipse, go to **File->New->Project->Java Project**
* Put **Project Name** as **RSTest**
* Select **Execution Environment JRE** as JavaSE-1.8.
* Click on **Finish.**
* In the Project Explorer, right click **RSTest** and click **Properties.**
* In the Propertieswindow, click on **Java Build Path** in the left pane and then click on the **Libraries** tab.
* Click on **Add External JARs** and select all the jar files in the selenium-driver **libs** folder.

**4.3: Create Cart.java to test cart functions**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as Cart and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.sql.Timestamp;

**import** java.util.Calendar;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public class** Cart {

**private** WebDriver mDriver;

**public static void** main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Cart cart =**new** Cart();

//cart.addToCartWithoutLogin();

//cart.addItemToCart();

//cart.addDuplicateItemToCart();

cart.removeItemFromCart();

**System**.exit(0);

}

**public** Boolean addToCartWithoutLogin(){

**System**.out.println("=====Cart - Add To Cart W/o Login=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.HOMEPAGE;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.linkText("Add To Cart")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Error, You need to login before adding items to cart")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/cart-addcartwithoutlogin.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public** Boolean addItemToCart(){

**System**.out.println("=====Cart - Add To Cart With Login=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Add To Cart")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR CART")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/cart-additemtocart.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public**boolean addDuplicateItemToCart(){

**System**.out.println("=====Cart - Add Duplicate Item To Cart With Login=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Add To Cart")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR CART")){

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Add To Cart")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR CART")){

**if**(mDriver.getPageSource().contains("This item is already in your cart")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/cart-adddupitemtocart.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public** Boolean removeItemFromCart(){

**System**.out.println("=====Cart - Remove Item From Cart =====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Add To Cart")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR CART")){

mDriver.findElement(By.linkText("Remove")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("Total Cart Value: 0")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

**System**.out.println("PASSED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/cart-removeitemfromocart.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.4: Create Checkout.java to test cart checkout flow**

* In Project Explorer, right click **RSTest->src->rstest**
* Right click on rstest and click **New->Class**
* Put **Name** as Checkout and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.sql.Timestamp;

**import** java.util.Calendar;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public class** Checkout {

**private** WebDriver mDriver;

**public static** void main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Checkout checkout =**new** Checkout();

checkout.doCheckout();

**System**.exit(0);

}

**public** boolean doCheckout(){

**System**.out.println("=====Checkout - Do Checkout=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Add To Cart")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR CART")){

mDriver.findElement(By.linkText("Checkout Now")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("CHECKOUT")){

mDriver.findElement(By.linkText("Pay via secure Payment Gateway")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("PAYMENT GATEWAY")){

mDriver.findElement(By.linkText("Click to complete checkout")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("PURCHASE CONFIRMATION")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/checkout-docheckout.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.5: Create Dashboard.java to check logging into dashboard**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click on rstest and click **New->Class**
* Put **Name** as Dashboard and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public class** Dashboard {

**private** WebDriver mDriver;

**public static void** main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Dashboard dashboard =**new** Dashboard();

dashboard.showDashboard();

**System**.exit(0);

}

**public** boolean showDashboard(){

**System**.out.println("=====Dashboard - Show Dashboard=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("DASHBOARD")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/dashboard-showdashboard.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.6: Create EditProfile.java to check edit profile function**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as EditProfile and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**publicclass** EditProfile {

**private** WebDriver mDriver;

**public static**void main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

EditProfile ep =**new** EditProfile();

//ep.invalidData();

ep.validData();

**System**.exit(0);

}

**public** boolean invalidData(){

**System**.out.println("=====Edit Profile - Invalid Data=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Edit Profile")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("MEMBER EDIT PROFILE")){

mDriver.findElement(By.xpath("//button[text()='Update']")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Error , Incomplete passwords submitted.")){

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.name("pwd2")).sendKeys("aaaaaa");

mDriver.findElement(By.name("fname")).clear();

mDriver.findElement(By.xpath("//button[text()='Update']")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("First name is required.")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.name("pwd2")).sendKeys("aaaaaa");

mDriver.findElement(By.name("fname")).sendKeys("fname");

mDriver.findElement(By.name("lname")).clear();

mDriver.findElement(By.xpath("//button[text()='Update']")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("Last name is required.")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

**System**.out.println("PASSED");

}

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/editprofile-invaliddata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public** boolean validData(){

**System**.out.println("=====Edit Profile - Valid Data=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("DASHBOARD")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Home")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(!mDriver.getPageSource().contains("HOMEPAGE")){

retVal =**false**;

**System**.out.println("FAILED");

}**else**{

mDriver.findElement(By.linkText("Edit Profile")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("MEMBER EDIT PROFILE")){

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.name("pwd2")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Update']")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("DASHBOARD")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/editprofile-validdata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.7: Create Globals.java to create global variables for use in the classes**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as Gobals and click **Finish.**
* Add the following code:

**package** rstest;

**public class** Globals {

**public staticString** BASE\_DOMAIN ="http://ec2-13-58-250-84.us-east-2.compute.amazonaws.com:8080/SportyShoes";

**public static String** HOMEPAGE = BASE\_DOMAIN +"/";

**public static String** LOGIN = BASE\_DOMAIN +"/login";

**public static String** SIGNUP = BASE\_DOMAIN +"/signup";

**public static String** SIGNUP\_CONFIRMATION = BASE\_DOMAIN +"/registerconfirm";

**public static String** DASHBOARD = BASE\_DOMAIN +"/dashboard";

**public static String** EDIT\_PROFILE = BASE\_DOMAIN +"/editprofile";

**public static String** CART = BASE\_DOMAIN +"/cart";

**public static String** ORDERS = BASE\_DOMAIN +"/memberpurchases";

**public static String** LOGOUT = BASE\_DOMAIN +"/logout";

**public static String** SCREENSHOT\_DIR ="/home/oem/Downloads";

}

**4.8: Create Homepage.java to test homepage loading**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as Homepage and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.openqa.selenium.TakesScreenshot;

**public class** Homepage {

**private** WebDriver mDriver;

**public static void** main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Homepage homepage =**new** Homepage();

homepage.loadHomepage();

**System**.exit(0);

}

**public** Boolean loadHomepage(){

**System**.out.println("=====Homepage Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.HOMEPAGE;

mDriver.get(url);

mDriver.manage().window().maximize();

**String** appTitle=mDriver.getTitle();

**String** expTitle="Sporty Shoes";

**if**(appTitle.equals (expTitle)){

**System**.out.println("PASSED");

}

**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/homepage.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.9: Create Login.java to test login features**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as Login and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.JavascriptExecutor;

**public class** Login {

**private** WebDriver mDriver;

**public static** void main(**String**[] args)**throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Login login =**new** Login();

//login.emptyForm();

//login.invalidData();

//login.validData();

login.logout();

**System**.exit(0);

}

**public** Boolean emptyForm(){

**System**.out.println("=====Login - Empty Form Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Login failed")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/login-emptyform.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public Boolean** invalidData(){

**System**.out.println("=====Login - Invalid Data Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("dummy@dummy.com");

mDriver.findElement(By.name("pwd")).sendKeys("dummy");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Login failed")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/login-invaliddata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public Boolean** validData(){

**System**.out.println("=====Login - Valid Data Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("DASHBOARD")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/login-validdata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public boolean** logout(){

**System**.out.println("=====Login - Do Logout=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**publicBoolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("DASHBOARD")){

mDriver.findElement(By.linkText("Logout")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("HOME")&& mDriver.getPageSource().contains("Login/Signup")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**newFile**(Globals.SCREENSHOT\_DIR +"/login-logout.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.10: Create OrderHistory.java to test display of Order History**

* In Project Explorer, right click **RSTest->src->rstest**
* Right click rstest and click **New->Class**
* Put **Name** as OrderHistory and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public class** OrderHistory {

**private** WebDriver mDriver;

**public static void** main(**String**[] args) **throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

OrderHistory orders =**new** OrderHistory();

orders.showOrders();

**System**.exit(0);

}

**public boolean** showOrders(){

**System**.out.println("=====Order History - Show Orders=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.LOGIN;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.xpath("//button[text()='Login']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("DASHBOARD")){

mDriver.findElement(By.linkText("Your Orders")).click();

wait =**new** WebDriverWait(mDriver,40);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("YOUR ORDERS")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/orderhistory-showorders.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.11.Create Signup.java to test signup feature**

* In Project Explorer, right click **RSTest->src->rstest.**
* Right click rstest and click **New->Class**
* Put **Name** as Signup and click **Finish.**
* Add the following code:

**package** rstest;

**import** java.io.File;

**import** java.io.IOException;

**import** java.nio.file.Files;

**import** java.sql.Timestamp;

**import** java.util.Calendar;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.JavascriptExecutor;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedCondition;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public class** Signup {

**private** WebDriver mDriver;

**public static** void main(**String**[] args) **throws InterruptedException**{

**System**.setProperty("webdriver.gecko.driver","/home/oem/workspace/geckowebdriver/geckodriver");

**System**.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE,"/dev/null");

Signup signup=**new** Signup();

signup.emptyForm();

signup.duplicateData();

signup.validData();

**System**.exit(0);

}

**public boolean** emptyForm(){

**System**.out.println("=====Signup - Empty Form Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.SIGNUP;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.xpath("//button[text()='Signup']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Email id is required.")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/signup-emptyform.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public boolean** duplicateData(){

**System**.out.println("=====Signup - Invalid Data Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.SIGNUP;

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys("amit@amit.com");

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.name("pwd2")).sendKeys("aaaaaa");

mDriver.findElement(By.name("fname")).sendKeys("fname");

mDriver.findElement(By.name("lname")).sendKeys("lname");

mDriver.findElement(By.name("age")).sendKeys("20");

mDriver.findElement(By.xpath("//button[text()='Signup']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("This email id already exists.")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/signup-duplicatedata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

**public boolean** validData(){

**System**.out.println("=====Signup - Valid Data Test=====");

boolean retVal =**true**;

mDriver=**new** FirefoxDriver();

**String** url = Globals.SIGNUP;

**String** emailId =**String**.valueOf(**System**.currentTimeMillis())+"@dummy.com";

mDriver.get(url);

mDriver.manage().window().maximize();

mDriver.findElement(By.name("email\_id")).sendKeys(emailId);

mDriver.findElement(By.name("pwd")).sendKeys("aaaaaa");

mDriver.findElement(By.name("pwd2")).sendKeys("aaaaaa");

mDriver.findElement(By.name("fname")).sendKeys("fname");

mDriver.findElement(By.name("lname")).sendKeys("lname");

mDriver.findElement(By.name("age")).sendKeys("20");

mDriver.findElement(By.name("address")).sendKeys("some address");

mDriver.findElement(By.xpath("//button[text()='Signup']")).click();

ExpectedCondition<**Boolean**> pageLoadCondition =**new**

ExpectedCondition<**Boolean**>(){

**public Boolean** apply(WebDriver driver){

**return**((JavascriptExecutor)driver).executeScript("return document.readyState").equals("complete");

}

};

WebDriverWait wait =**new** WebDriverWait(mDriver,20);

wait.until(pageLoadCondition);

**if**(mDriver.getPageSource().contains("Your registration is confirmed.")){

**System**.out.println("PASSED");

}**else**{

retVal =**false**;

**System**.out.println("FAILED");

}

TakesScreenshot ts =(TakesScreenshot) mDriver;

**File** source = ts.getScreenshotAs(OutputType.FILE);

**File** dest =**new File**(Globals.SCREENSHOT\_DIR +"/signup-validdata.png");

**try**{

**Files**.copy(source.toPath(), dest.toPath(), java.nio.file.**StandardCopyOption**.REPLACE\_EXISTING);

}**catch**(**IOException** iex){

iex.printStackTrace();

}

mDriver.close();

**return** retVal;

}

}

**4.12: Run the Selenium tests**

* In Project Explorer, click **RSTest** and press **Ctrl F11.**
* All the test classes will be with the automated Firefox webdriver. The output for each test can be seen in the Eclipse Console.

**Step 5: Do load testing using JMeter**

1. Download and install Apache JMeter
2. Create a SportyShoes Test Plan
3. Create a Homepage Thread Group
4. Create a Signup Thread Group
5. Create a Login Thread Group
6. Create an Add To Cart and Checkout Thread Group
7. Create an Edit ProfileThread Group
8. Create a View Orders Thread Group
9. Run the Jmeter test plan

**5.1: Download and install Apache Jmeter**

* Go to <https://jmeter.apache.org/download_jmeter.cgi> and download the latest tgz binary.
* Unzip the contents into a folder in your local system.
* Go to the /bin folder of jmeter and run the **jmeter** program file.
* This will load the Jmeter application.

**5.2: Create a SportyShoes Test Plan**

* Click on **File->New.** This will open the screen to enter a new Test Plan.
* Put **Name** as Sporty Shoes.
* Click on **Add** under **User Defined Variables.**
* Double click the entry in **Name** column and enter uid
* Double click the entry in **Value** column and enter ${\_\_UUID()}

**5.3: Create a Homepage Thread Group**

* Right click **SportyShoes** in the sidebar and click **Add->Threads->ThreadGroup**
* Enter **Name** as TG Homepage.
* Right click **TG Homepage** in sidebar click **Add->Sampler->HTTP Request**
* Enter Homepage in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter GET.
* In **Path** enter /SportyShoes.
* Right click **Homepage** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **Homepage** in the sidebar and click **Add->Listener->View Results Tree.**

**5.4: Create a Signup Thread Group**

* Right click **SportyShoes** in the sidebar and click **Add->Threads->ThreadGroup.**
* Enter **Name** as TG Signup.
* Right click **TG Signup** in sidebar click **Add->Sampler->HTTP Request.**
* Enter Show Registration Form in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** put 8080.
* In **Method** put GET.
* In **Path** put /*SportyShoes/*signup.
* Right click **Show Registration Form** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **TG Signup** in sidebar and click **Add->Sampler->HTTP Request.**
* Enter Submit Signup Form in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter POST.
* In **Path** enter /*SportyShoes/*signupaction.
* In **Parameters Tab,** add the following parameters:

email\_id, ${\_\_UUID()@dummy.com,

pwd, aaaaaa

pwd2, aaaaaa

fname, ${\_\_UUID() fname

lname, ${\_\_UUID() lname

age, 20

address, some address

* Right click **Submit Signup Form** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **Submit Signup Form** in the sidebar and click **Add->Listener->View Results Tree.**

**5.5: Create a Login Thread Group**

* Right click **SportyShoes** in the sidebar and click **Add->Threads->ThreadGroup.**
* Enter **Name** as TG Login.
* Right click **TG Login** in sidebar and click **Add->Config Element->HTTP Cookie Manager.**
* In the **Cookie Manager** screen check **Use Thread Group Configuration to control cookie clearing.**
* Right click **TG Login** in sidebar click **Add->Sampler->HTTP Request.**
* Enter Show Login Form in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter GET.
* In **Path** enter /*SportyShoes/*login.
* Right click **Show Login Form** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **TG Login** in sidebar click **Add->Sampler->HTTP Request.**
* Enter Submit Login Form in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter POST.
* In **Path** enter /*SportyShoes/*loginaction.
* In **Parameters,** enter the following parameters:

email\_id, [amit@amit.com](mailto:amit@amit.com)

pwd, aaaaaa

* Right click on **Submit Login Form** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click on **Submit Signup Form** in the sidebar and click **Add->Listener->View Results Tree.**

**5.6: Create an Add To Cart and Checkout Thread Group**

* Right click on **SportyShoes** in the sidebar and click **Add->Threads->ThreadGroup.**
* Enter **Name** as TG Add To Cart and Checkout.
* Right click **TG Add To Cart & Checkout** in sidebar and click **Add->Config Element->HTTP Cookie Manager.**
* In the **Cookie Manager** screen, check **Use Thread Group Configuration to control cookie clearing.**
* Right click **TG Add To Cart & Checkout** in sidebar and click **Add->Sampler->HTTP Request.**
* Enter Show Login Form in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number** put 8080.
* In **Method** enter GET.
* In **Path** enter /*SportyShoes/*login.
* Right click **Show Login Form** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **TG Add To Cart & Checkout** in side bar click **Add->Sampler->HTTP Request.**
* Enter Submit Login Form in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter POST.
* In **Path,** enter /*SportyShoes/*loginaction.
* In **Parameters,** enter the following parameters:

email\_id, [amit@amit.com](mailto:amit@amit.com)

pwd, aaaaaa

* Right click **Submit Login Form** in the sidebar and click **Add->Timer->Constant Timer**
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter Show Homepage in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter GET.
* In **Path** enter /*SportyShoes.*
* Right click **Show Homepage** in the sidebar and click **Add->Timer->Constant Timer.**
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter Add Item 1 to Cart in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number** enter 8080.
* In **Method** enter GET.
* In **Path** enter /SportyShoes/cartadditem?id=107
* Right click **Add Item 1 to Cart** in the sidebar and click Add->Timer->Constant Timer.
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter Add Item 2 to Cart in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /SportyShoes/cartadditem?id=2
* Right click **Add Item 2 to Cart** in the sidebar and click Add->Timer->Constant Timer.
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter View Cart in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /SportyShoes/cart.
* Right click **View Cart** in the sidebar and click Add->Timer->Constant Timer.
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter **Show Checkout Page** in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /SportyShoes/checkout.
* Right click **Show Checkout Page** in the sidebar and click Add->Timer->Constant Timer.
* Right click **TG Add To Cart & Checkout** in the sidebar and click **Add->Sampler->HTTP Request.**
* Enter **Show Payment Gateway** in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /SportyShoes/gateway.
* Right click **Show Payment Gateway** in the sidebar and click on Add->Timer->Constant Timer.
* Right click on **TG Add To Cart & Checkout** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter **Complete Checkout** in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /SportyShoes/completepurchase.
* Right click on **Complete Checkout** in the sidebar and click on Add->Timer->Constant Timer.
* Right click on **Complete Checkout**  in the sidebar and click on **Add->Listener->View Results Tree.**

**5.7: Create an Edit ProfileThread Group**

* Right click on **SportyShoes** in the sidebar and click on **Add->Threads->ThreadGroup.**
* Enter **Name** as TG Edit Profile.
* Right click on **TG Edit Profile** in the sidebar and click on **Add->Config Element->HTTP Cookie Manager.**
* In the **Cookie Manager** screen, check **Use Thread Group Configuration** to control cookie clearing.
* Right click on **TG Edit Profile** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter Show Login Form in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /*SportyShoes/*login.
* Right clickon **Show Login Form** in the sidebar and click on **Add->Timer->Constant Timer.**
* Right click on **TG Edit Profile** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter “Submit Login Form” in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter POST.
* In **Path,** enter /*SportyShoes/*loginaction.
* In **Parameters,** enter the following parameters:

email\_id, [amit@amit.com](mailto:amit@amit.com)

pwd, aaaaaa

* Right click on **Submit Login Form** in the sidebar and click on **Add->Timer->Constant Timer.**
* Right click on **TG Edit Profile** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter “Show Edit Profile Form” in **Name.**
* In **Servername** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /*SportyShoes/*editprofile.
* Right click on **Show Edit Profile Form** in the sidebar and click on Add->Timer->Constant Timer.
* Right click on **TG Edit Profile**  in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter “Submit Edit Profile Form” in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter POST.
* In **Path,** enter /*SportyShoes/*signupaction.
* In **Parameters Tab,** add the following parameters:

pwd, aaaaaa

pwd2, aaaaaa

fname, ${\_\_UUID() fname

lname, ${\_\_UUID() lname

age, 20

address, some address

user\_id, 24

* Right click on **Submit Edit Profile** in the sidebar and click on **Add->Timer->Constant Timer.**
* Right click on **Submit Edit Profile** in the sidebar and click on **Add->Listener->View Results Tree.**

**5.8: Create a View Orders Thread Group**

* Right click on **Sporty Shoes** in the sidebar and click on **Add->Threads->ThreadGroup.**
* Enter **Name** as TG View Orders.
* Right click on **TG View Orders** in the sidebar and click on **Add->Config Element->HTTP Cookie Manager.**
* In the **Cookie Manager** screen, check **Use Thread Group Configuration to control cookie clearing**
* Right click on **TG View Orders** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter “Show Login Form” in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter /*SportyShoes/*login.
* Right click **on Show Login Form** in the sidebar and click on **Add->Timer->Constant Timer.**
* Right click on **TG Edit Profile** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter “Submit Login Form” in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter POST.
* In **Path,** enter/*SportyShoes/*loginaction.
* In **Parameters,** put the following parameters:

email\_id, [amit@amit.com](mailto:amit@amit.com)

pwd, aaaaaa

* Right click on **Submit Login Form** in the sidebar and click on **Add->Timer->Constant Timer.**
* Right click on **TG View Orders** in the sidebar and click on **Add->Sampler->HTTP Request.**
* Enter Show Orders Page in **Name.**
* In **Servername,** paste the EC2 instance url.
* In **Port Number,** enter 8080.
* In **Method,** enter GET.
* In **Path,** enter/*SportyShoes/*memberpurchases.
* Right click on **Show Orders Page** in the sidebar and click on **Add->Timer->Constant Timer**.
* Right clickon **Show Orders Page** in the sidebar and click on **Add->Listener->View Results Tree.**

**5.9: Run the JMeter test plan**

* Press Ctrl + R to run all the thread groups in the Sporty Shoes Test plan.

**Step 6: Automate build and test processes using Jenkins**

1. Download and install Jenkins
2. Automate the build process of SportyShoes webapp
3. Automate JUnit tests of SportyShoes webapp
4. Automate Selenium Scripts Execution
5. Add a Report plugin to see job results

**6.1: Download and install Jenkins**

* Go to <https://jenkins.io/download/> . Under **Long Term Support (LTS),** click on **Ubuntu/Debian.**
* In the next page under **Individual Package Downloads,** download the latest deb package.
* Execute the downloaded deb package and install Jenkins.
* After installation, open your browser and type [http://localhost:8080](http://localhost:8080/) . This should start the configuration of Jenkins.
* Set up the admin username and password.
* Jenkins is now ready for use.

**6.2: Automate the build process of SportyShoes webapp**

* From the dashboard click **New Item.**
* Enter **Item Name** as “Build Build” and click **Ok.**
* In the **Configuration** page, under **Build** section in the **Add Build Step** dropdown, click **Execute Shell.**
* Type in the following script:

#!/bin/bash

cd <path to your pom.xml for SportyShoes>

mvn clean build

* Under **Build Triggers** section, check **Build Periodically.**
* In the Schedule, add the following script:

# once in every two hours slot between 9 AM and 5 PM every weekday

H H(9-16)/2 \* \* 1-5

* Click **Save.**

**6.3: Automate JUnit tests of SportyShoes webapp**

* Before JUnit tests can be automated, a JUnit Platform Runner has to be downloaded.
* Go to <https://jar-download.com/maven-repository-class-search.php?search_box=org.junit.platform.runner.JUnitPlatform> and download **Download junit-platform-runner.jar (1.5.2)**
* Copy this jar into a folder which is in the JAVA classpath.
* From the Jenkins dashboard, click on **New Item.**
* Enter **Item Name** as JUnit Tests and click **Ok.**
* In the **Configuration** page, under **Build** section in the **Add Build Step** dropdown, click **Execute Shell.**
* Type in the following script:

#!/bin/bash

cd <project path of SportyShoes>/target/test-classes/com/ecommerce/entity

java -jar junit-platform-console-standalone-1.5.2.jar --class-path <project path of SportyShoes>/target/test-classes/com/ecommerce/entity –scan-class-path

cd <project path of SportyShoes>/target/test-classes/com/ecommerce/service

java -jar junit-platform-console-standalone-1.5.2.jar --class-path <project path of SportyShoes>/target/test-classes/com/ecommerce/service –scan-class-path

* Under **Build Triggers** section, check **Build Periodically.**
* In the Schedule add the following script:

# once in every three hours slot between 9 AM and 5 PM every weekday

H H(9-16)/3 \* \* 1-5

* Click **Save**

**6.4: Automate Selenium Scripts Execution**

* Before we can execute Selenium tasks from Jenkins, we have to convert each of the classes into an executable format.
* In Eclipse, open the RSTest project in **Project Explorer.**
* **Expand RSTest->src->rstest.** Right click on Cart.java and click **Export.** In the **Export Wizard,** select Java->Runnable JAR File, change the name of the jar in **Export Destination** to Cart.jar and click **Next.**
* From the Launch Configuration dropdown, select **Cart RSTest** and click on **Finish.**
* This will create a jar called Cart.jar in the destination folder.
* To test it, run the jar from the terminal: **java -jar Cart.jar**
* The jar file should execute exactly as it does in Eclipse.
* Repeat the above steps for all the java classes in rstest package.
* From the Jenkins dashboard, click on **New Item.**
* Enter **Item Name** as Selenium Tests and click **Ok.**
* In the **Configuration** page, under **Build** section in the **Add Build Step** dropdown, click on **Execute Shell.**
* Type in the following script:

#!/bin/bash

cd <path to the executable jars folder>

java -jar <name of jar file>

java -jar <name of jar file>

java -jar <name of jar file>

* Under **Build Triggers** section, check **Build Periodically.**
* In the Schedule, add the following script:

# once in every three hours slot between 9 AM and 5 PM every weekday

H H(9-16)/3 \* \* 1-5

* Click **Save.**

**6.5: Add a Report plugin to see job results**

* Go to the Jenkins Dashboard.
* Click on the link that says "Manage Jenkins."
* On the Plugin Manager page, go to the "Available" tab next to Updates tab.
* Look for the html publisher plugin, select the checkbox, and click **install**. Wait for it to come back with the status "Success."
* Restart Jenkins.
* For each of the jobs created, go into the **Configure** page and go to the **Post Build Actions** section.
* Select **Publish HTML Reports** and click the checkbox.
* Fill the path to the directory containing the html reports in the **HTML directory to the archive** field.
* Specify the **Index page(s)** to display (default index.html). You can specify multiple comma-separated pages and each will be a tab on the report page.
* You can also specify **Index Page titles** for the report page that appears on the tab. By default, the file name will be taken as title.
* Add a name in the **Report Title** field, which will be used to provide a link to the report. By default, only the most recent HTML report will be saved, but if you'd like to be able to view HTML reports for each past build, select **Keep past HTML reports.**
* Click **Save.**

**Step 7: Use Postman to test the API endpoints**

1. Download and Install Postman.
2. Create a New Collection with Requests.
3. Run the Postman Collection.

**7.1: Download and Install Postman**

* Go to <https://www.getpostman.com/downloads/> and click on **Download.** Select Linux 64-bit.
* This will download the tar file for Postman. Unzip the contents in a folder in your local system.
* Double click on the Postman link in the folder. This will launch Postman.
* It will ask you to do a login/signup before you can get started. If you already have a Postman account then you can log in, otherwise you have to go through the login process.
* Once the initial loading of Postman is done, we can set up Postman Collections.

**7.2: Create a New Collection with Requests**

* Click on **New.** Enter Sporty Shoes REST API as the **Name** and click **Create.**
* Right click on **Sporty Shoes** in the sidebar and click on **Add Request.**
* Enter Fetch Products List as **Request Name** and click on **Save.**
* Click on **Fetch Products** in the sidebar. In the right panel enter **<EC2 URL>:8080/RESTService/rest/products** for **Enter request URL.**
* Click on **Send.**
* Once the data has been fetched, click **Save.**
* Right click on **Sporty Shoes** in the sidebar and click on **Add Request.**
* Enter Fetch User List as Request Name and click on Save.
* Click on **Fetch User List** in the sidebar. In the right panel enter **<EC2 URL>:8080/RESTService/rest/users** for **Enter request URL.**
* Click on **Send.**
* Once the data has been fetched, click **Save.**

**7.3. Run the Postman Collection**

* To run all the Requests in the Collection, click on the right arrow next to **Sporty Shoes REST API** in the sidebar.
* In the expanded second panel, click **Run.**