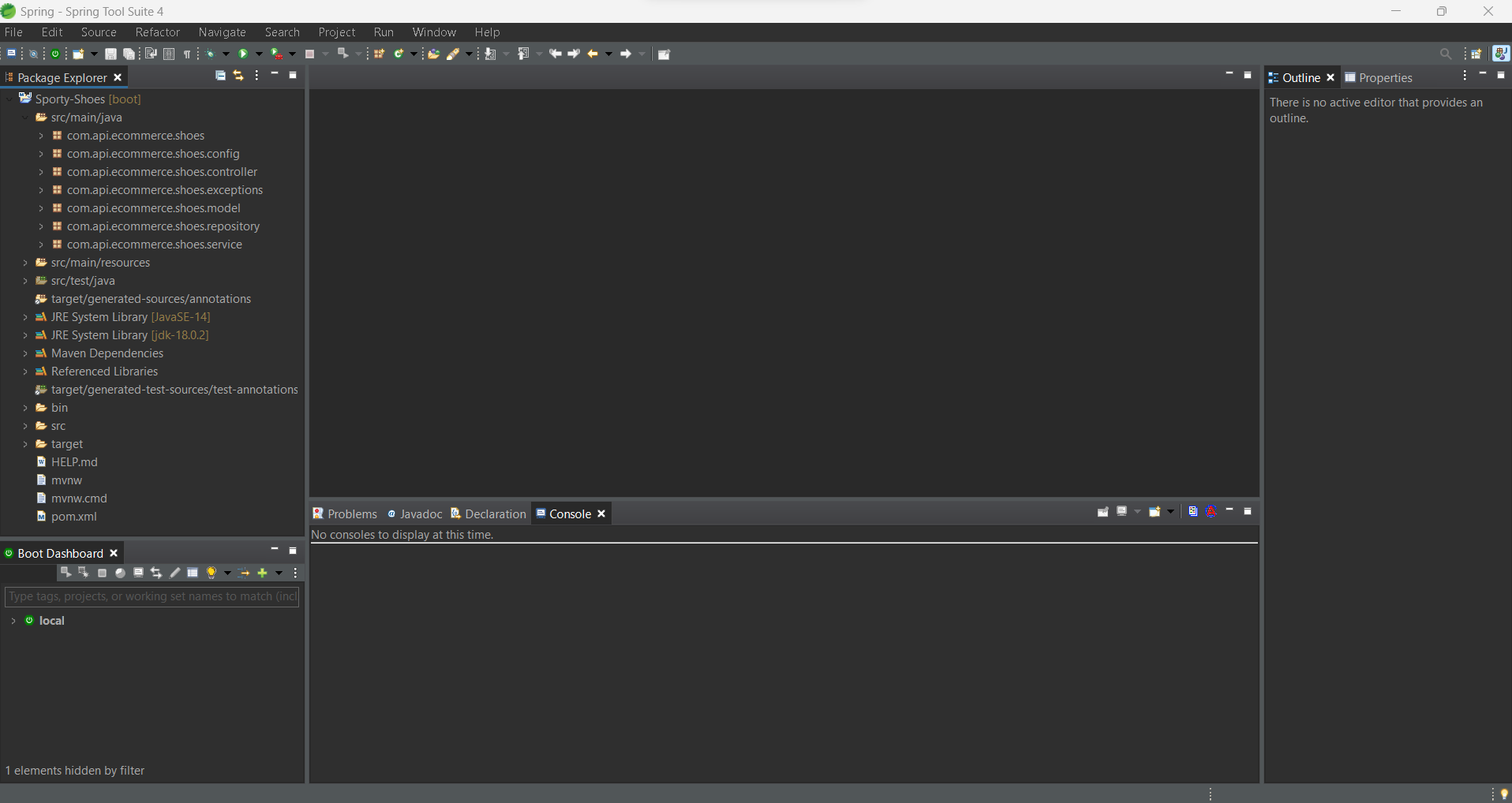
E-Commerce Application for Sporty Shoes

Source Code

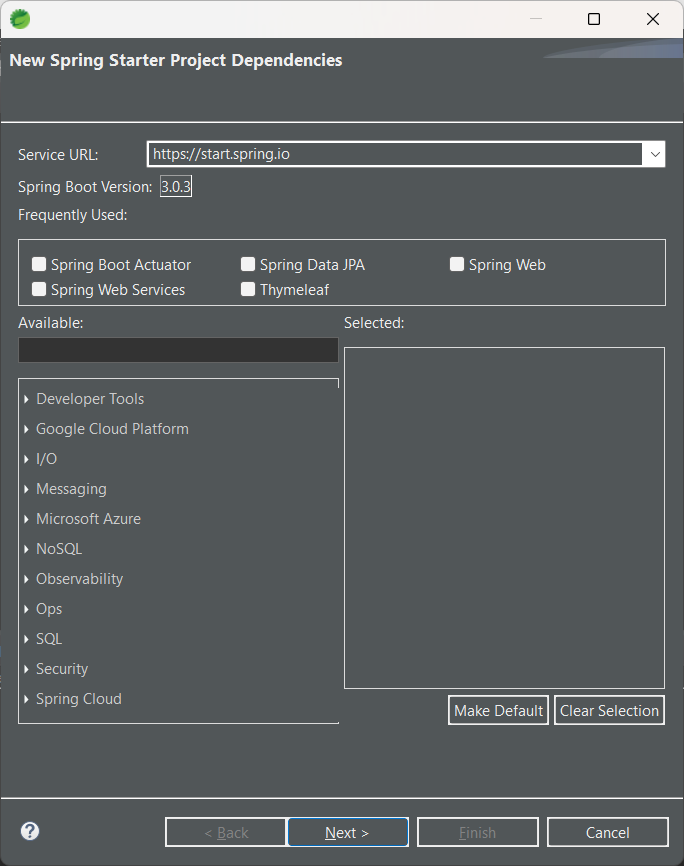
Step – 1: Create a new Spring Boot Project

* **Open the Spring.io website and use spring Initialize to create a new Spring Boot project using the settings and select “Maven” as build and add the required Dependencies to generate the Project.**
* **Open Spring Tool Suite**
* **Go to File -> Open Projects from File System -> Load Project from Folder/Directory and click on the ok button.**
* **Now the Project will be loaded into your IDE then go to the Pom.xml file and check whether the dependencies added or not.**
* **Now If not Right click on the Project and Select Spring -> Add Staters -> A new window will pop up add the required dependencies then click on ok it will automatically add them to the Pom.xml file.**
* **After that Right click on the Project -> Run as -> Maven clean -> Maven Install -> Maven Build it will add the newly added dependencies to the project.**
* **Then Right click on Project -> Run as -> Spring Boot App it should launch the application with the default port number as “8080” if this port number is not available or we want to use some other port then go to the application. properties file in the Project and configure it like**

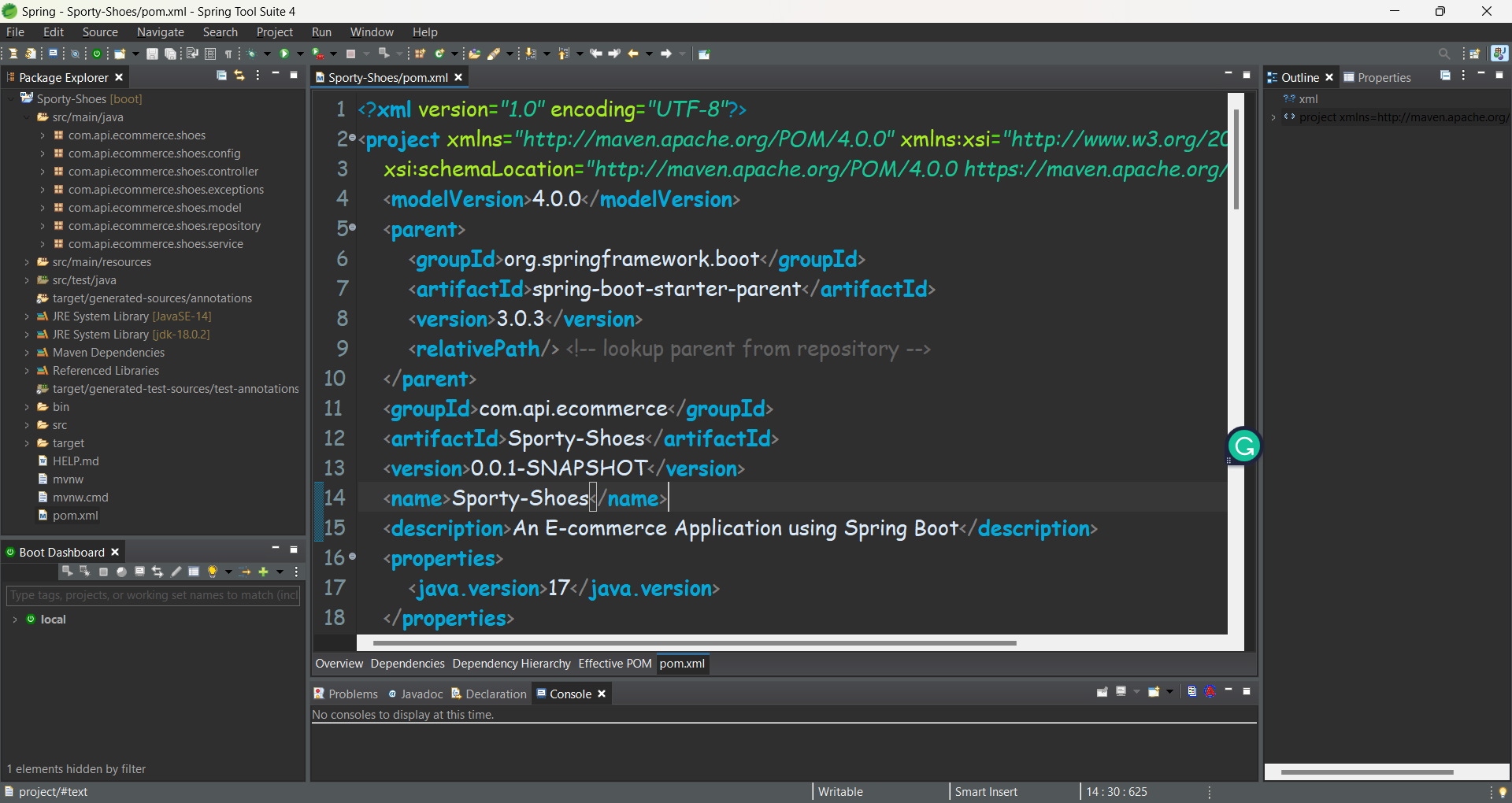
**“Server. Port = 8081”**

****

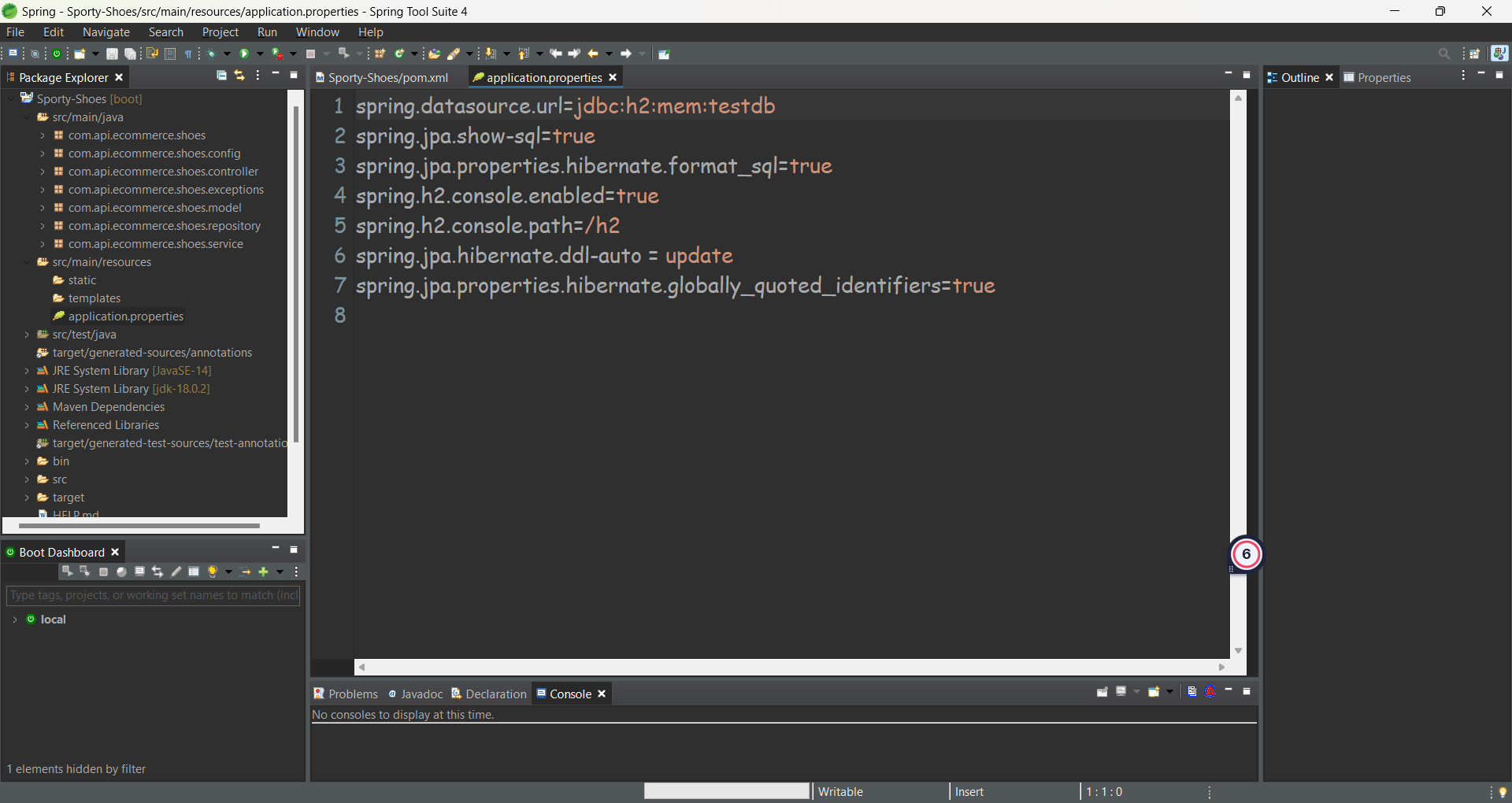
Step - 2: Adding the Dependencies



Step – 4: Configuring Pom. XML file



Step – 5: Configuring Application. Properties file with SQL



Step – 6: Creating the Model Classes (data) as Entities

Admin.java:

package com.api.ecommerce.shoes.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@Entity*

*@Table*

public class Admin {

public Admin() {

super();

}

public Admin(int id, String adminfirstName, String adminlastName, String adminUserName, String adminPassword,

String adminContactNo) {

super();

this.id = id;

this.adminfirstName = adminfirstName;

this.adminlastName = adminlastName;

this.adminUserName = adminUserName;

this.adminPassword = adminPassword;

this.adminContactNo = adminContactNo;

}

*@Id* //defining the id as primary key

*@Column*

private int id;

*@Column*

private String adminfirstName;

*@Column*

private String adminlastName;

*@Column*

private String adminUserName;

*@Column*

private String adminPassword;

*@Column*

private String adminContactNo;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAdminfirstName() {

return adminfirstName;

}

public void setAdminfirstName(String adminfirstName) {

this.adminfirstName = adminfirstName;

}

public String getAdminlastName() {

return adminlastName;

}

public void setAdminlastName(String adminlastName) {

this.adminlastName = adminlastName;

}

public String getAdminUserName() {

return adminUserName;

}

public void setAdminUserName(String adminUserName) {

this.adminUserName = adminUserName;

}

public String getAdminPassword() {

return adminPassword;

}

public void setAdminPassword(String adminPassword) {

this.adminPassword = adminPassword;

}

public String getAdminContactNo() {

return adminContactNo;

}

public void setAdminContactNo(String adminContactNo) {

this.adminContactNo = adminContactNo;

}

}

User.java:

package com.api.ecommerce.shoes.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table

public class User {

@Id

@Column

private int id;

@Column

private String userFirstName;

@Column

private String userLastName;

@Column

private String userEmailId;

@Column

private String userPassword;

@Column

private String userContactNo;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUserFirstName() {

return userFirstName;

}

public void setUserFirstName(String userFirstName) {

this.userFirstName = userFirstName;

}

public String getUserLastName() {

return userLastName;

}

public void setUserLastName(String userLastName) {

this.userLastName = userLastName;

}

public String getUserEmailId() {

return userEmailId;

}

public void setUserEmailId(String userEmailId) {

this.userEmailId = userEmailId;

}

public String getUserPassword() {

return userPassword;

}

public void setUserPassword(String userPassword) {

this.userPassword = userPassword;

}

public String getUserContactNo() {

return userContactNo;

}

public void setUserContactNo(String userContactNo) {

this.userContactNo = userContactNo;

}

}

Product.java:

package com.api.ecommerce.shoes.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table

public class Product {

@Id

@Column

private int pid;

@Column

private String pname;

@Column

private String pdesc;

@Column

private String pcategory;

@Column

private String color;

@Column

private String labelCode;

public String getLabelCode() {

return labelCode;

}

public void setLabelCode(String labelCode) {

this.labelCode = labelCode;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

@Column

private int price;

public int getPid() {

return pid;

}

public void setPid(int pid) {

this.pid = pid;

}

public String getPname() {

return pname;

}

public void setPname(String pname) {

this.pname = pname;

}

public String getPdesc() {

return pdesc;

}

public void setPdesc(String pdesc) {

this.pdesc = pdesc;

}

public String getPcategory() {

return pcategory;

}

public void setPcategory(String pcategory) {

this.pcategory = pcategory;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

}

PurchaseReport.java:

package com.api.ecommerce.shoes.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table

public class PurchaseReport {

@Id

@Column

private int id;

@Column

private String purchasedCustomerName;

@Column

private String category;

@Column

private String dateOfPurchase;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getPurchasedCustomerName() {

return purchasedCustomerName;

}

public void setPurchasedCustomerName(String purchasedCustomerName) {

this.purchasedCustomerName = purchasedCustomerName;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getDateOfPurchase() {

return dateOfPurchase;

}

public void setDateOfPurchase(String dateOfPurchase) {

this.dateOfPurchase = dateOfPurchase;

}

}

Step – 7: Creating a Repository File to perform JPA Operations

AdminRepository.java:

package com.api.ecommerce.shoes.repository;

import org.springframework.data.repository.CrudRepository;

import com.api.ecommerce.shoes.model.Admin;

public interface AdminRepository extends CrudRepository<Admin, Integer>{

public Admin findByAdminUserName(String name);

public Admin findByAdminUserNameAndAdminPassword(String userName, String password);

}

UserRepository.java:

package com.api.ecommerce.shoes.repository;

import org.springframework.data.repository.CrudRepository;

import com.api.ecommerce.shoes.model.User;

public interface UserRepository extends CrudRepository<User, Integer>{

User findByUserEmailId(String name);

User findByUserEmailIdAndUserPassword(String emailId, String password);

}

ProductRepository.java:

package com.api.ecommerce.shoes.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.api.ecommerce.shoes.model.Product;

*@Repository*

public interface ProductRepository extends CrudRepository<Product, Integer>{

List<Product> findAll();

Product getByLabelCode(String name);

}

PurchaseReportRepository.java:

package com.api.ecommerce.shoes.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.api.ecommerce.shoes.model.PurchaseReport;

*@Repository*

public interface PurchaseReportRepository extends CrudRepository<PurchaseReport, Integer> {

PurchaseReport findByPurchasedCustomerName(String name);

List<PurchaseReport> findAll();

List<PurchaseReport> findByCategory(String name);

}

Step – 8: Adding the Service class for implementing the JPA Methods

AdminService.java:

package com.api.ecommerce.shoes.service;

import com.api.ecommerce.shoes.model.Admin;

public interface AdminService {

public Admin addAdminUser(Admin admin);

public Admin getAdminUser(String name);

public Admin getAdminUserByUserNameAndAdminPassword(String tempUserName, String tempPass);

}

UserService.java:

package com.api.ecommerce.shoes.service;

import org.springframework.stereotype.Component;

import com.api.ecommerce.shoes.model.User;

*@Component*

public interface UserService {

User addUser(User user);

User getByUserEmailId(String name);

User getByUserEmailIdAndUserPassword(String email, String pass);

}

ProductService.java:

package com.api.ecommerce.shoes.service;

import java.util.List;

import org.springframework.stereotype.Component;

import com.api.ecommerce.shoes.exceptions.ProductNotFoundException;

import com.api.ecommerce.shoes.model.Product;

*@Component*

public interface ProductService {

Product addProduct(Product product);

Product getById(int id) throws ProductNotFoundException;

Product updateProduct(Product product);

List<Product> deleteProductById(int id);

List<Product> getAllProducts();

List<Product> getProductByPrice(int price);

Product getByLabelCode(String name);

}

PurchaseReportService.java:

package com.api.ecommerce.shoes.service;

import java.util.List;

import org.springframework.stereotype.Component;

import com.api.ecommerce.shoes.exceptions.PurchaseReportNotFoundException;

import com.api.ecommerce.shoes.model.PurchaseReport;

*@Component*

public interface PurchaseReportService {

PurchaseReport createPurchaseReport(PurchaseReport purchaseReport);

PurchaseReport getPurchaseReportByPurchasedCustomerName(String name);

PurchaseReport getPurchaseReportById(int id) throws PurchaseReportNotFoundException;

PurchaseReport updatePurchaseReport(PurchaseReport purchaseReport);

List<PurchaseReport> deletePurchaseReportById(int id);

List<PurchaseReport> getAllPurchaseReports();

List<PurchaseReport> getAllPurchaseReportsByCategory(String name);

}

Step – 9: Implementing the Service Layer and performing all the operations

AdminServiceImpl.java:

package com.api.ecommerce.shoes.service;

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

import org.springframework.stereotype.Service;

import com.api.ecommerce.shoes.model.Admin;

import com.api.ecommerce.shoes.repository.AdminRepository;

*@Service*

public class AdminServiceImpl implements AdminService {

*@Autowired*

private AdminRepository adminRepository;

/\* This method will add new admin user \*/

public Admin addAdminUser(Admin admin) {

return adminRepository.save(admin);

}

public Admin getAdminUser(String name) {

return adminRepository.findByAdminUserName(name);

}

public Admin getAdminUserByUserNameAndAdminPassword(String uname, String pwd) {

return adminRepository.findByAdminUserNameAndAdminPassword(uname, pwd);

}

}

UserServiceImpl.java

package com.api.ecommerce.shoes.service;

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

import org.springframework.stereotype.Service;

import com.api.ecommerce.shoes.model.User;

import com.api.ecommerce.shoes.repository.UserRepository;

*@Service*

public class UserServiceImpl implements UserService{

*@Autowired*

private UserRepository userRepo;

public User addUser(User user) {

return userRepo.save(user);

}

public User getByUserEmailId(String name) {

return userRepo.findByUserEmailId(name);

}

public User getByUserEmailIdAndUserPassword(String eid, String pwd) {

return userRepo.findByUserEmailIdAndUserPassword(eid, pwd);

}

}

ProductServiceImpl.java:

package com.api.ecommerce.shoes.service;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

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

import org.springframework.stereotype.Service;

import com.api.ecommerce.shoes.exceptions.ProductNotFoundException;

import com.api.ecommerce.shoes.model.Product;

import com.api.ecommerce.shoes.repository.ProductRepository;

*@Service*

public class ProductServiceImpl implements ProductService {

*@Autowired*

private ProductRepository productRepo;

public Product addProduct(Product product) {

return productRepo.save(product);

}

public Product getById(int id) throws ProductNotFoundException {

return productRepo.findById(id).orElseThrow(() -> new ProductNotFoundException());

}

public Product updateProduct(Product product) {

return productRepo.save(product);

}

public List<Product> deleteProductById(int id) {

List<Product> productList = new ArrayList<Product>();

productRepo.deleteById(id);

productList = productRepo.findAll().stream().collect(Collectors.*toList*());

return productList;

}

public List<Product> getAllProducts() {

List<Product> productList = new ArrayList<Product>();

productList = productRepo.findAll().stream().collect(Collectors.*toList*());

return productList;

}

public List<Product> getProductByPrice(int price) {

List<Product> productList = new ArrayList<Product>();

productList = productRepo.findAll().stream().filter(i -> i.getPrice() <= price).collect(Collectors.*toList*());

return productList;

}

public Product getByLabelCode(String name) {

return productRepo.getByLabelCode(name);

}

}

PurchaseReportServiceImpl.java:

package com.api.ecommerce.shoes.service;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

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

import org.springframework.stereotype.Service;

import com.api.ecommerce.shoes.exceptions.PurchaseReportNotFoundException;

import com.api.ecommerce.shoes.model.PurchaseReport;

import com.api.ecommerce.shoes.repository.PurchaseReportRepository;

*@Service*

public class PurchaseReportServiceImpl implements PurchaseReportService{

*@Autowired*

private PurchaseReportRepository prRepo;

public PurchaseReport createPurchaseReport(PurchaseReport purchaseReport) {

return prRepo.save(purchaseReport);

}

public PurchaseReport getPurchaseReportByPurchasedCustomerName(String name) {

return prRepo.findByPurchasedCustomerName(name);

}

public PurchaseReport getPurchaseReportById(int id) throws PurchaseReportNotFoundException {

return prRepo.findById(id).orElseThrow(()-> new PurchaseReportNotFoundException());

}

public PurchaseReport updatePurchaseReport(PurchaseReport purchaseReport) {

return prRepo.save(purchaseReport);

}

public List<PurchaseReport> deletePurchaseReportById(int id) {

List<PurchaseReport> purchaseReportList = new ArrayList<PurchaseReport>();

prRepo.deleteById(id);

purchaseReportList = prRepo.findAll().stream().collect(Collectors.*toList*());

return purchaseReportList;

}

public List<PurchaseReport> getAllPurchaseReports() {

List<PurchaseReport> purchaseReportList = new ArrayList<PurchaseReport>();

purchaseReportList = prRepo.findAll().stream().collect(Collectors.*toList*());

return purchaseReportList;

}

public List<PurchaseReport> getAllPurchaseReportsByCategory(String name) {

return prRepo.findByCategory(name);

}

}

Step – 10: Create Controller classes to handle the web request methods using Rest API

AdminController.java:

package com.api.ecommerce.shoes.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.api.ecommerce.shoes.exceptions.AdminException;

import com.api.ecommerce.shoes.exceptions.AdminUserNotFoundException;

import com.api.ecommerce.shoes.exceptions.LoginException;

import com.api.ecommerce.shoes.model.Admin;

import com.api.ecommerce.shoes.service.AdminService;

*@RestController*

public class AdminController {

*@Autowired*

private AdminService adminService;

*@PostMapping*(value="/adminUsers")

public ResponseEntity<Admin> addAdminUser(*@RequestBody* Admin admin) throws Exception {

String tempUserName = admin.getAdminUserName();

if (tempUserName != null && ! "".equals(tempUserName)) {

Admin adminobj = adminService.getAdminUser(tempUserName);

if (adminobj != null) {

throw new AdminException();

}

}

Admin adminObj = null;

adminObj = adminService.addAdminUser(admin);

return new ResponseEntity<Admin>(adminObj, *HttpStatus*.***CREATED***);

}

*@GetMapping*(value = "/adminUsers/{adminUserName}")

public ResponseEntity<Admin> getAdminUser(*@PathVariable*("adminUserName") String adminUserName) throws Exception{

Admin admin = adminService.getAdminUser(adminUserName);

if(admin == null) {

throw new AdminUserNotFoundException();

}

return new ResponseEntity<Admin>(admin, *HttpStatus*.***OK***);

}

*@PostMapping*(value="/login")

public ResponseEntity<Admin> getAdminUserByUserNameAndPassword(*@RequestBody* Admin admin) throws Exception {

String tempUserName = admin.getAdminUserName();

String tempPass = admin.getAdminPassword();

Admin adminUser = null;

if (tempUserName != null && tempPass != null) {

adminUser = adminService.getAdminUserByUserNameAndAdminPassword(tempUserName, tempPass);

}

if (adminUser == null) {

throw new LoginException();

}

return new ResponseEntity<Admin>(adminUser, *HttpStatus*.***OK***);

}

}

UserController.java:

package com.api.ecommerce.shoes.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.api.ecommerce.shoes.exceptions.LoginException;

import com.api.ecommerce.shoes.exceptions.NoUserFoundException;

import com.api.ecommerce.shoes.exceptions.UserException;

import com.api.ecommerce.shoes.model.User;

import com.api.ecommerce.shoes.service.UserService;

*@RestController*

public class UserController {

*@Autowired*

private UserService userService;

*@PostMapping*(value="/users")

public ResponseEntity<User> addUser(*@RequestBody* User user) throws Exception{

String tempEmailId = user.getUserEmailId();

if(tempEmailId != null && !"".equals(tempEmailId)) {

User userobj = userService.getByUserEmailId(tempEmailId);

if(userobj != null) {

throw new UserException();

}

}

User userObj = null;

userObj = userService.addUser(user);

return new ResponseEntity<User>(userObj, *HttpStatus*.***CREATED***);

}

*@GetMapping*(value="/users/{emailId}")

public ResponseEntity<User> getByUserEmailId(*@PathVariable*("emailId") String emailId) throws Exception{

User user = userService.getByUserEmailId(emailId);

if(user == null) {

throw new NoUserFoundException();

}

return new ResponseEntity<User>(user, *HttpStatus*.***OK***);

}

*@PostMapping*(value="/loginUser")

public ResponseEntity<User> getByUserEmailIdAndUserPassword(*@RequestBody* User user) throws Exception {

String tempEmailId = user.getUserEmailId();

String tempPass = user.getUserPassword();

User userObj = null;

if (tempEmailId != null && tempPass != null) {

userObj = userService.getByUserEmailIdAndUserPassword(tempEmailId, tempPass);

}

if (userObj == null) {

throw new LoginException();

}

return new ResponseEntity<User>(userObj, *HttpStatus*.***OK***);

}

}

ProductController.java:

package com.api.ecommerce.shoes.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.api.ecommerce.shoes.exceptions.ProductException;

import com.api.ecommerce.shoes.exceptions.ProductNotFoundException;

import com.api.ecommerce.shoes.model.Product;

import com.api.ecommerce.shoes.service.ProductService;

*@RestController*

public class ProductController {

*@Autowired*

ProductService productService;

*@PostMapping*(value="/products")

public ResponseEntity<Product> addProduct(*@RequestBody* Product product) throws Exception{

String tempLabelName = product.getLabelCode();

if(tempLabelName != null && !"".equals(tempLabelName)) {

Product productObj = productService.getByLabelCode(tempLabelName);

if(productObj != null ) {

throw new ProductException();

}

}

Product productobj = null;

productobj = productService.addProduct(product);

return new ResponseEntity<Product>(productobj, *HttpStatus*.***CREATED***);

}

*@GetMapping*(value="/products/{id}")

public ResponseEntity<Product> getByProductId(*@PathVariable*("id") int id) throws ProductNotFoundException {

Product product = productService.getById(id);

return new ResponseEntity<Product>(product, *HttpStatus*.***OK***);

}

*@PutMapping*(value="/products")

public ResponseEntity<Product> updateProduct(*@RequestBody* Product product) {

Product pro= productService.updateProduct(product);

return new ResponseEntity<Product>(pro,*HttpStatus*.***CREATED***);

}

*@DeleteMapping*(value="/products/{id}")

public ResponseEntity<List<Product>> deleteProductById(*@PathVariable*("id") int id) {

List<Product> productList = productService.deleteProductById(id);

return new ResponseEntity<>(productList, *HttpStatus*.***ACCEPTED***);

}

*@GetMapping*(value="/products")

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

List<Product> productList = productService.getAllProducts();

return new ResponseEntity<>(productList, *HttpStatus*.***OK***);

}

*@GetMapping*(value="/products/price/{price}")

public ResponseEntity<List<Product>> getProductByPrice(*@PathVariable*("price") int price) {

List<Product> productList = productService.getProductByPrice(price);

return new ResponseEntity<>(productList, *HttpStatus*.***OK***);

}

}

PurchaseReportController.java:

package com.api.ecommerce.shoes.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.api.ecommerce.shoes.exceptions.PurchaseReportException;

import com.api.ecommerce.shoes.exceptions.PurchaseReportNotFoundException;

import com.api.ecommerce.shoes.model.PurchaseReport;

import com.api.ecommerce.shoes.service.PurchaseReportService;

*@RestController*

public class PurchaseReportController {

*@Autowired*

private PurchaseReportService prService;

*@PostMapping*(value="/admin/purchaseReport")

public ResponseEntity<PurchaseReport> createPurchaseReport(*@RequestBody* PurchaseReport purchaseReport) throws Exception{

String tempCustomerName = purchaseReport.getPurchasedCustomerName();

if(tempCustomerName != null && !"".equals(tempCustomerName)) {

PurchaseReport prObject = prService.getPurchaseReportByPurchasedCustomerName(tempCustomerName);

if(prObject != null) {

throw new PurchaseReportException();

}

}

PurchaseReport pr = null;

pr = prService.createPurchaseReport(purchaseReport);

return new ResponseEntity<PurchaseReport>(pr, *HttpStatus*.***CREATED***);

}

*@GetMapping*(value="admin/purchaseReport/{id}")

public ResponseEntity<PurchaseReport> getPurchaseReportById(*@PathVariable*("id") int id) throws PurchaseReportNotFoundException{

PurchaseReport pr = prService.getPurchaseReportById(id);

return new ResponseEntity<PurchaseReport>(pr, *HttpStatus*.***OK***);

}

*@PutMapping*(value="admin/purchaseReport")

public ResponseEntity<PurchaseReport> updatePurchaseReport(*@RequestBody* PurchaseReport purchaseReport) {

PurchaseReport pr = prService.updatePurchaseReport(purchaseReport);

return new ResponseEntity<PurchaseReport>(pr, *HttpStatus*.***CREATED***);

}

*@DeleteMapping*(value="admin/purchaseReports/{id}")

public ResponseEntity<List<PurchaseReport>> deletePurchaseReposeReportById(*@PathVariable*("id") int id) {

List<PurchaseReport> prList = prService.deletePurchaseReportById(id);

return new ResponseEntity<>(prList, *HttpStatus*.***OK***);

}

*@GetMapping*(value="/admin/purchaseReports")

public ResponseEntity<List<PurchaseReport>> getAllPurchaseReports() {

List<PurchaseReport> prList = prService.getAllPurchaseReports();

return new ResponseEntity<>(prList, *HttpStatus*.***OK***);

}

*@GetMapping*(value="/admin/purchaseReports/{category}")

public ResponseEntity<List<PurchaseReport>> getAllPurchaseReportsByCategory(*@PathVariable*("category") String name) {

List<PurchaseReport> prList = prService.getAllPurchaseReportsByCategory(name);

return new ResponseEntity<>(prList, *HttpStatus*.***OK***);

}

}

Step – 10: Adding the Exception Controller to handle the Exceptions custom Exception Handler

ExceptionController.java:

package com.api.ecommerce.shoes.exceptions;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

*@ControllerAdvice*

public class ExceptionController {

*@ExceptionHandler*(value= AdminException.class)

public ResponseEntity<Object> exception(AdminException exception) {

return new ResponseEntity<>("User already Exists", *HttpStatus*.***CONFLICT***);

}

*@ExceptionHandler*(value= AdminUserNotFoundException.class)

public ResponseEntity<Object> exception(AdminUserNotFoundException exception) {

return new ResponseEntity<>("User Not found", *HttpStatus*.***NOT\_FOUND***);

}

*@ExceptionHandler*(value= ProductException.class)

public ResponseEntity<Object> exception(ProductException exception) {

return new ResponseEntity<>("Product already exist", *HttpStatus*.***CONFLICT***);

}

*@ExceptionHandler*(value= LoginException.class)

public ResponseEntity<Object> exception(LoginException exception) {

return new ResponseEntity<>("Bad Credentials Try Again", *HttpStatus*.***UNAUTHORIZED***);

}

*@ExceptionHandler*(value= UserException.class)

public ResponseEntity<Object> exception(UserException exception) {

return new ResponseEntity<>("You are already registered", *HttpStatus*.***CONFLICT***);

}

*@ExceptionHandler*(value= NoUserFoundException.class)

public ResponseEntity<Object> exception(NoUserFoundException exception) {

return new ResponseEntity<>("you are not registered user", *HttpStatus*.***NOT\_FOUND***);

}

*@ExceptionHandler*(value= ProductNotFoundException.class)

public ResponseEntity<Object> exception(ProductNotFoundException exception) {

return new ResponseEntity<>("Product not found", *HttpStatus*.***NOT\_FOUND***);

}

*@ExceptionHandler*(value= PurchaseReportException.class)

public ResponseEntity<Object> exception(PurchaseReportException exception) {

return new ResponseEntity<>("Purchase Report already exists", *HttpStatus*.***CONFLICT***);

}

*@ExceptionHandler*(value= PurchaseReportNotFoundException.class)

public ResponseEntity<Object> exception(PurchaseReportNotFoundException exception) {

return new ResponseEntity<>("Purchase Report doesn't exist", *HttpStatus*.***NOT\_FOUND***);

}

}