**PHASE3-FINALPROJECT-WRITEUP**

**Make an E-commerce Website for Sporty Shoes .**

**Project objective:**

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository. 

**Background of the problem statement:**

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their e-commerce portal sportyshoes.com.

You’re asked to develop a prototype of the application. It will be then presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you’re asked to do the following:

● Presenting the specification document which has the product’s capabilities, appearance, and user interactions  
● Setting up Git and GitHub account to store and track your enhancements of the prototype   
● Explaining the Java concepts used in the project   
● Discussing the generic features of the product:  
● There will be an admin to manage the website. An administrator login will be required to access the admin page.

**The admin should be able to change his password if he wants, he should be able to:**

● Manage the products in the store including categorizing them  
● Browse the list of users who have signed up and be able to search users  
● See purchase reports filtered by date and category

This project has following subsections, namely:

1. Creating a Spring Boot Starter Project which is web enabled
2. Building the project
3. Publishing and starting the project
4. Pushing the code to your GitHub repositories

**1. Creating a Spring Boot Starter Project which is web enabled**

* Open Eclipse
* Go to the **File** menu. Choose **New->Other**
* In the **Wizards** list, select **Spring Boot->Spring Starter Project**
* In **Name,** enter sportyshoes, **Type** as Maven, **Packaging** as Jar, **Group** as com.ecommerce, and **Package** as com.ecommerce Click on **Next**
* In the list of **Available** dependencies, scroll down to select starter-web, h2database, Lombok, starter-data-jpa, spring-security

**Dependencies** and add swagger2-ui dependency from maven repositories

* Click on **Next**
* Click on **Finish**
* This will create the project files in the Project Explorer

**sportyshoesApplication**

**package** com.ecommerce.sportyshoes;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration;

@SpringBootApplication(exclude = { SecurityAutoConfiguration.**class** })

**public** **class** sportyshoesApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(sportyshoesApplication.**class**, args);

}

}

**ServletInitializer**

**package** com.ecommerce.sportyshoes;

**import** org.springframework.boot.builder.SpringApplicationBuilder;

**import** org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

**public** **class** ServletInitializer **extends** SpringBootServletInitializer {

@Override

**protected** SpringApplicationBuilder configure(SpringApplicationBuilder application) {

**return** application.sources(sportyshoesApplication.**class**);

}

}

**SpringSecurityConfig**

**package** com.ecommerce.sportyshoes.config;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.~~WebSecurityConfigurerAdapter~~;

@Configuration

@EnableWebSecurity

**public** **class** SpringSecurityConfig **extends** ~~WebSecurityConfigurerAdapter~~ {

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http

.csrf().disable()

.authorizeRequests()

.anyRequest()

.authenticated()

.and()

.httpBasic();

}

}

**SwaggerConfig**

**package** com.ecommerce.sportyshoes.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** springfox.documentation.builders.RequestHandlerSelectors;

**import** springfox.documentation.spi.DocumentationType;

**import** springfox.documentation.spring.web.plugins.Docket;

**import** springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableSwagger2

**public** **class** SwaggerConfig {

@Bean

**public** Docket superHeroApiDoc() {

**return** **new** Docket(DocumentationType.***SWAGGER\_2***).select()

.apis(RequestHandlerSelectors.*basePackage*("com.ecommerce.sportyshoes")).build();

}

}

**CRUDController**

**package** com.ecommerce.sportyshoes.controller;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.util.LinkedMultiValueMap;

**import** org.springframework.util.MultiValueMap;

**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.ecommerce.sportyshoes.exceptionHandler.BusinessException;

**import** com.ecommerce.sportyshoes.model.PurchaseReport;

**import** com.ecommerce.sportyshoes.model.Shoe;

**import** com.ecommerce.sportyshoes.service.SportyShoesService;

@RestController

**public** **class** CRUDController {

@Autowired

**private** SportyShoesService service;

**private** MultiValueMap<String, String> errorMap;

/\*\*

\* Shoe post request controller

\*

\* **@param** shoe

\* **@return** ResponseEntity<Shoe> with newly created Shoe

\*/

@PostMapping("/admin/shoe")

**public** ResponseEntity<Shoe> createShoe(@RequestBody Shoe shoe) {

**try** {

**return** **new** ResponseEntity<>(service.createShoe(shoe), HttpStatus.***OK***);

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(**null**, errorMap, HttpStatus.***BAD\_REQUEST***);

}

}

/\*\*

\* Shoe get request controller

\*

\* **@param** id

\* **@return** ResponseEntity<Shoe> with the given id

\*/

@GetMapping("/admin/shoe/{id}")

**public** ResponseEntity<Shoe> getShoeById(@PathVariable **int** id) {

**try** {

**return** **new** ResponseEntity<>(service.getShoeById(id), HttpStatus.***OK***);

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(**null**, errorMap, HttpStatus.***NOT\_FOUND***);

}

}

/\*\*

\* Shoe put(update) request controller

\*

\* **@param** shoe

\* **@return** ResponseEntity<Shoe> with updated shoe

\*/

@PutMapping("/admin/shoe")

**public** ResponseEntity<Shoe> updateShoe(@RequestBody Shoe shoe) {

**return** **new** ResponseEntity<>(service.updateShoe(shoe), HttpStatus.***OK***);

}

/\*\*

\* Shoe delete request controller

\*

\* **@param** id

\* **@return** ResponseEntity<String> containing the status of delete operation

\*/

@DeleteMapping("/admin/shoe/{id}")

**public** ResponseEntity<String> deleteShoeById(@PathVariable **int** id) {

**try** {

service.deleteShoeById(id);

**return** **new** ResponseEntity<>("Succesfully deleted shoe with id: " + id, HttpStatus.***OK***);

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(e.getMessage(), errorMap, HttpStatus.***BAD\_REQUEST***);

}

}

/\*\*

\* Purchase Report post request controller

\*

\* **@param** pr - Purchase Report

\* **@return** ResponseEntity<PurchaseReport> with newly created Purchase Report

\*/

@PostMapping("/admin/purchaseReport")

**public** ResponseEntity<PurchaseReport> createPurchaseReport(@RequestBody PurchaseReport pr) {

**try** {

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

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(**null**, errorMap, HttpStatus.***BAD\_REQUEST***);

}

}

/\*\*

\* Purchase Report get request controller

\*

\* **@param** id

\* **@return** ResponseEntity<PurchaseReport> with given id

\*/

@GetMapping("/admin/purchaseReport/id/{id}")

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

**try** {

**return** **new** ResponseEntity<>(service.getPurchaseReportById(id), HttpStatus.***OK***);

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(**null**, errorMap, HttpStatus.***NOT\_FOUND***);

}

}

/\*\*

\* Purchase Report put(update) request controller

\*

\* **@param** pr

\* **@return** ResponseEntity<PurchaseReport> containing updated Purchase Report

\*/

@PutMapping("/admin/purchaseReport")

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

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

}

/\*\*

\* Purchase Report delete request controller

\*

\* **@param** id

\* **@return** ResponseEntity<String> containing the status of delete request.

\*/

@DeleteMapping("/admin/purchaseReport/{id}")

**public** ResponseEntity<String> deletePurchaseReportById(@PathVariable **int** id) {

**try** {

service.deletePurchaseReportById(id);

**return** **new** ResponseEntity<>("Succesfully deleted Purchase Report with id: " + id, HttpStatus.***OK***);

} **catch** (BusinessException e) {

errorMap = **new** LinkedMultiValueMap<>();

errorMap.add("errorMessage:", e.getMessage());

**return** **new** ResponseEntity<>(e.getMessage(), errorMap, HttpStatus.***BAD\_REQUEST***);

}

}

}

**SearchController**

**package** com.ecommerce.sportyshoes.controller;

**import** java.util.Date;

**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.GetMapping;

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

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

**import** com.ecommerce.sportyshoes.model.PurchaseReport;

**import** com.ecommerce.sportyshoes.model.Shoe;

**import** com.ecommerce.sportyshoes.service.SportyShoesService;

@RestController

**public** **class** SearchController {

@Autowired

**private** SportyShoesService service;

/\*\*

\* Shoe search controller

\* **@return** all shoe list

\*/

@GetMapping("/admin/shoe/all")

**public** ResponseEntity<List<Shoe>> getAllShoes(){

**return** **new** ResponseEntity<List<Shoe>>(service.getAllShoes(), HttpStatus.***OK***);

}

/\*\*

\* Purchase Report Search Controller

\* **@param** category

\* **@return** purchase reports filtered by the category

\*/

@GetMapping("/admin/purchaseReport/category/{category}")

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

**return** **new** ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReportsByCategory(category), HttpStatus.***OK***);

}

/\*\*

\* Purchase Report Search Controller

\* **@param** dateInMs

\* **@return** purchase reports filtered by date of purchase(in millisecond time)

\*/

@GetMapping("/admin/purchaseReport/date/{dateInMs}")

**public** ResponseEntity<List<PurchaseReport>> getAllPurchaseReportsByDop(@PathVariable Long dateInMs){

Date dop = **new** Date(dateInMs);

**return** **new** ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReportsByDOP(dop), HttpStatus.***OK***);

}

/\*\*

\* Purchase Report Search Controller

\* **@return** all purchase reports

\*/

@GetMapping("/admin/purchaseReport/all")

**public** ResponseEntity<List<PurchaseReport>> getAllPurchaseReport(){

**return** **new** ResponseEntity<List<PurchaseReport>>(service.getAllPurchaseReports(), HttpStatus.***OK***);

}

}

**ExceptionController**

**package** com.ecommerce.sportyshoes.controller;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

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

**import** com.ecommerce.sportyshoes.exceptionHandler.BusinessException;

@ControllerAdvice

**public** **class** ExceptionController {

@ExceptionHandler(value = BusinessException.**class**)

**public** ResponseEntity<Object> exception(BusinessException exception){

**return** **new** ResponseEntity<>(exception.getMessage(), HttpStatus.***EXPECTATION\_FAILED***);

}

}

**BusinessException**

**package** com.ecommerce.sportyshoes.exceptionHandler;

**public** **class** BusinessException **extends** Exception{

**private** String message;

**public** BusinessException(String message) {

**this**.message=message;

}

@Override

**public** String getMessage() {

**return** message;

}

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = -851150852212525045L;

}

**Shoe.java**

**package** com.ecommerce.sportyshoes.model;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@Entity

@Table

@Getter

@Setter

@NoArgsConstructor

@ToString

**public** **class** Shoe {

**public** Shoe(**int** id, String name, String category, **double** price) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.category = category;

**this**.price = price;

}

@Id

@GeneratedValue

**private** **int** id;

**private** String name;

**private** String category;

**private** **double** price;

}

**PurchaseReport.java**

**package** com.ecommerce.sportyshoes.model;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** javax.persistence.Temporal;

**import** javax.persistence.TemporalType;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@Entity

@Table

@Setter

@Getter

@NoArgsConstructor

@ToString

**public** **class** PurchaseReport {

**public** PurchaseReport(**int** id, String purchasedBy, String category, Date dop, String orderList) {

**super**();

**this**.id = id;

**this**.purchasedBy = purchasedBy;

**this**.category = category;

**this**.dop = dop;

**this**.orderList = orderList;

}

@Id

@GeneratedValue

**private** **int** id;

**private** String purchasedBy; // This can be extended to utilize one to one relation with User Table [Future Implemetations]

**private** String category;

@Temporal(TemporalType.***DATE***)

**private** Date dop;

/\*\*

\* This can be used for storing orderlist as <Qty, Shoe>

\* Here implementation is made simple by using shoeId instead

\* of shoe in string format.

\*/

// @ManyToMany(cascade = CascadeType.ALL)

// Map<Integer,Shoe> orderList = new HashMap<Integer,Shoe>();

// OR

// Map<Integer,Integer> orderList = new HashMap<Integer,Integer>();

String orderList;

}

**ShoesRepository**

**package** com.ecommerce.sportyshoes.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

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

**import** com.ecommerce.sportyshoes.model.Shoe;

@Repository

**public** **interface** ShoesRepository **extends** JpaRepository<Shoe, Integer>{

}

**PurchaseReportRepository**

**package** com.ecommerce.sportyshoes.repository;

**import** java.util.Date;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

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

**import** com.ecommerce.sportyshoes.model.PurchaseReport;

@Repository

**public** **interface** PurchaseReportRepository **extends** JpaRepository<PurchaseReport, Integer>{

**public** List<PurchaseReport> findByDop(Date dop);

**public** List<PurchaseReport> findByCategory(String category);

}

**SportyShoesService.java**

**package** com.ecommerce.sportyshoes.service;

**import** java.util.Date;

**import** java.util.List;

**import** com.ecommerce.sportyshoes.exceptionHandler.BusinessException;

**import** com.ecommerce.sportyshoes.model.PurchaseReport;

**import** com.ecommerce.sportyshoes.model.Shoe;

**public** **interface** SportyShoesService {

**public** Shoe createShoe(Shoe shoe) **throws** BusinessException;

**public** Shoe getShoeById(**int** id) **throws** BusinessException;

**public** Shoe updateShoe(Shoe shoe);

**public** **void** deleteShoeById(**int** id) **throws** BusinessException;

**public** List<Shoe> getAllShoes();

**public** PurchaseReport createPurchaseReport(PurchaseReport pr) **throws** BusinessException;

**public** PurchaseReport getPurchaseReportById(**int** id) **throws** BusinessException;

**public** PurchaseReport updatePurchaseReport(PurchaseReport pr);

**public** **void** deletePurchaseReportById(**int** id) **throws** BusinessException;

**public** List<PurchaseReport> getAllPurchaseReports();

**public** List<PurchaseReport> getAllPurchaseReportsByCategory(String category);

**public** List<PurchaseReport> getAllPurchaseReportsByDOP(Date dop);

}

**SportyShoesServiceImpl.java**

**package** com.ecommerce.sportyshoes.service.impl;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.NoSuchElementException;

**import** javax.annotation.PostConstruct;

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

**import** org.springframework.dao.EmptyResultDataAccessException;

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

**import** com.ecommerce.sportyshoes.exceptionHandler.BusinessException;

**import** com.ecommerce.sportyshoes.model.PurchaseReport;

**import** com.ecommerce.sportyshoes.model.Shoe;

**import** com.ecommerce.sportyshoes.repository.PurchaseReportRepository;

**import** com.ecommerce.sportyshoes.repository.ShoesRepository;

**import** com.ecommerce.sportyshoes.service.SportyShoesService;

**import** lombok.NoArgsConstructor;

@Service

@NoArgsConstructor

**public** **class** SportyShoesServiceImpl **implements** SportyShoesService{

@Autowired

**private** ShoesRepository shoesRepo;

@Autowired

**private** PurchaseReportRepository prRepo;

@PostConstruct

**public** **void** init() {

Shoe s1 = **new** Shoe(1,"Shoe Name 1","Basketball",1000.24);

Shoe s2 = **new** Shoe(2,"Shoe Name 2","Cricket",1100.24);

Shoe s3 = **new** Shoe(3,"Shoe Name 3","Running",900.24);

Shoe s4 = **new** Shoe(4,"Shoe Name 4","Football",1900.24);

shoesRepo.save(s1);

shoesRepo.save(s2);

shoesRepo.save(s3);

shoesRepo.save(s4);

Date d = **new** Date(0);

PurchaseReport pr1 = **new** PurchaseReport(5,"user\_1","Running",d,"adidas\_runner:5,nike\_airmax:10");

PurchaseReport pr2 = **new** PurchaseReport(6,"user\_2","Cricket",d,"adidas\_cricket:5,nike\_cricket:10");

PurchaseReport pr3 = **new** PurchaseReport(7,"user\_3","Basketball",d,"adidas\_basketball:5,nike\_basketball:10");

PurchaseReport pr4 = **new** PurchaseReport(8,"user\_4","Football",d,"adidas\_football:5,nike\_football:10");

prRepo.save(pr1);

prRepo.save(pr2);

prRepo.save(pr3);

prRepo.save(pr4);

}

**public** Shoe createShoe(Shoe shoe) **throws** BusinessException {

**int** id = shoe.getId();

Shoe oldShoe = **null**;

**try** {

oldShoe = shoesRepo.findById(id).get();

}**catch**(NoSuchElementException e) {

}

**if**(oldShoe!=**null**) **throw** **new** BusinessException("Shoe already exists with id: "+id);

**return** shoesRepo.save(shoe);

}

**public** Shoe getShoeById(**int** id) **throws** BusinessException {

Shoe shoe = **null**;

**try** {

**if**(id<=0) **throw** **new** BusinessException("Shoe Id can not be negative or zero");

shoe = shoesRepo.findById(id).get();

}**catch**(NoSuchElementException e) {

**throw** **new** BusinessException("Shoe not found with Id: "+id);

}

**return** shoe;

}

**public** Shoe updateShoe(Shoe shoe) {

**return** shoesRepo.save(shoe);

}

**public** **void** deleteShoeById(**int** id) **throws** BusinessException {

**try** {

shoesRepo.deleteById(id);

}**catch**(IllegalArgumentException e) {

**throw** **new** BusinessException("Invalid id: "+id);

}**catch**(EmptyResultDataAccessException e) {

**throw** **new** BusinessException("SHoe does not exist with id: "+id);

}

}

**public** List<Shoe> getAllShoes() {

**return** shoesRepo.findAll();

}

**public** PurchaseReport createPurchaseReport(PurchaseReport pr) **throws** BusinessException {

**int** id = pr.getId();

PurchaseReport oldPr = **null**;

**try** {

oldPr = prRepo.findById(id).get();

}**catch**(NoSuchElementException e) {

}

**if**(oldPr!=**null**) **throw** **new** BusinessException("Purchase report already exists with id: "+id);

**return** prRepo.save(pr);

}

**public** PurchaseReport getPurchaseReportById(**int** id) **throws** BusinessException {

PurchaseReport pr = **null**;

**try** {

**if**(id<=0) **throw** **new** BusinessException("Purchase Report Id can not be negative or zero");

pr = prRepo.findById(id).get();

}**catch**(NoSuchElementException e) {

**throw** **new** BusinessException("Purchase Report not found with Id: "+id);

}

**return** pr;

}

**public** PurchaseReport updatePurchaseReport(PurchaseReport pr) {

**return** prRepo.save(pr);

}

**public** **void** deletePurchaseReportById(**int** id) **throws** BusinessException {

**try** {

prRepo.deleteById(id);

}**catch**(IllegalArgumentException e) {

**throw** **new** BusinessException("Invalid id: "+id);

}**catch**(EmptyResultDataAccessException e) {

**throw** **new** BusinessException("Puchase Report does not exist with Id: "+id);

}

}

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

**return** prRepo.findAll();

}

**public** List<PurchaseReport> getAllPurchaseReportsByCategory(String category) {

**return** prRepo.findByCategory(category);

}

**public** List<PurchaseReport> getAllPurchaseReportsByDOP(Date dop) {

**return** prRepo.findByDop(dop);

}

}

**Application.properties**

spring.application.name=sportyshoes

server.port=4000

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.h2.console.enabled=true

spring.h2.console.path=/h2

spring.security.user.name=admin

spring.security.user.password=adminPass

spring.security.user.roles=ADMIN

**SportyshoesApplicationTests**

**package** com.ecommerce.sportyshoes;

**import** org.junit.jupiter.api.Test;

**import** org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

**class** SportyshoesApplicationTests {

@Test

**void** contextLoads() {

}

**Sportyshoes/pom.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.1</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.ecommerce</groupId>

<artifactId>sportyshoes</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>sportyshoes</name>

<description>Phase 3 Final project sporty shoes</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**2. Building the project**

* From the **Project** menu at the top, click on **Build**
* If any compile errors are shown, fix them as required

**3. Publishing and starting the project**

* In the Project Explorer, right click on **SpringBootStarter->Run As->Spring Boot App**
* Check in the Eclipse Console for the message **Started SpringBootStarterApplication**

**4. Pushing 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 initially created using the following command:

**git push -u origin master**