1. Open Spring Boot App:

2. Create Simple Maven project

3. Skip architype

4. Select package name/Groupid/ArchitypeID as sportyshoe

5. Edit the pom.xml file to add the necessary dependencies

<?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 http://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>2.2.5.RELEASE</version>

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

</parent>

<groupId>net.codejava</groupId>

<artifactId>sportyshoe</artifactId>

<version>2.0</version>

<name>sportyshoe</name>

<description>assessment phase3</description>

<packaging>war</packaging>

<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-thymeleaf</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.thymeleaf.extras</groupId>

<artifactId>thymeleaf-extras-springsecurity5</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

6. Edit the application.properties file:

server.port = 9096

spring.jpa.hibernate.ddl-auto=none

spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce

spring.datasource.username=root

spring.datasource.password=mysql

logging.level.root=WARN

7. Create User Table:

CREATE TABLE `users` (

-> `user\_id` int(11) NOT NULL AUTO\_INCREMENT,

-> `username` varchar(45) NOT NULL,

-> `password` varchar(64) NOT NULL,

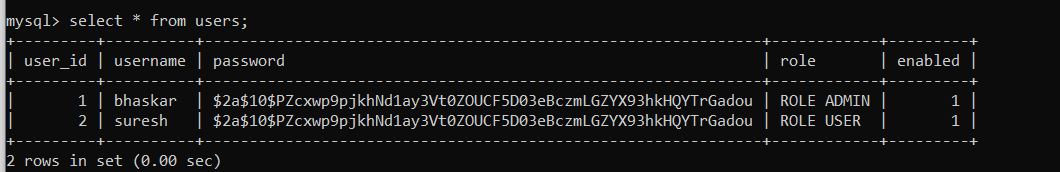
-> `role` varchar(45) NOT NULL,

-> `enabled` tinyint(4) DEFAULT NULL,

-> PRIMARY KEY (`user\_id`)

-> );

8. select \* from users;



The password is encrypted using Bcrypt API . clean text password is “singh”

Have Column Role for identifying which type user : USER or ADMIN

Enabled column will help to enable or disable the user

9. Create Shoes tables to manage shoes:

CREATE TABLE IF NOT EXISTS shoes (

id INT AUTO\_INCREMENT,

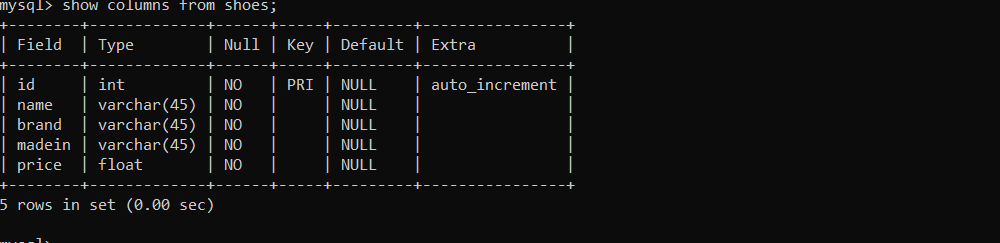
brand VARCHAR(255) NOT NULL,

model\_name VARCHAR(255) NOT NULL,

price long,

PRIMARY KEY (id)

);



10. Create the HTML pages:

Index.html

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"https://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Shoes Manager</title>

</head>

<body>

<div align=*"center"*>

<div >

<h3 th:inline=*"text"*>Welcome [[${#httpServletRequest.remoteUser}]]</h3>

</div>

<form th:action=*"@{/logout}"* method=*"post"*>

<input type=*"submit"* value=*"Logout"* />

</form>

<div>

<h1>List of Users</h1>

</div>

<h3><a th:href=*"@{/users}"*>List of Users</a></h3>

<h1>Shoes Manager</h1>

<div sec:authorize=*"hasRole('ADMIN')"*>

<a href=*"new"*>Create New Shoes</a>

</div>

<br/><br/>

<table border=*"1"* cellpadding=*"10"*>

<thead>

<tr>

<th>Shoes ID</th>

<th>Name</th>

<th>Brand</th>

<th>Made In</th>

<th>Price</th>

<th >Actions</th>

</tr>

</thead>

<tbody>

<tr th:each=*"shoes : ${listShoes}"*>

<td th:text=*"${shoes.id}"*>Shoes ID</td>

<td th:text=*"${shoes.name}"*>Name</td>

<td th:text=*"${shoes.brand}"*>Brand</td>

<td th:text=*"${shoes.madein}"*>Made in</td>

<td th:text=*"${shoes.price}"*>Price</td>

<td >

<a th:href=*"@{'/edit/' + ${shoes.id}}"*>Edit</a>

&nbsp;&nbsp;&nbsp;&nbsp;

<a th:href=*"@{'/delete/' + ${shoes.id}}"*>Delete</a>

</td>

</tr>

</tbody>

</table>

</div>

</body>

</html>

2. ListUser.html

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"ISO-8859-1"*>

<title>List Users</title>

<link rel=*"stylesheet"* type=*"text/css"* href=*"/webjars/bootstrap/css/bootstrap.min.css"* />

<script type=*"text/javascript"* src=*"/webjars/jquery/jquery.min.js"*></script>

<script type=*"text/javascript"* src=*"/webjars/bootstrap/js/bootstrap.min.js"*></script>

</head>

<title>Users List:</title>

</head>

<body>

<div>

<table class=*"table table-striped table-bordered"*>

<thead class=*"thead-dark"*>

<tr>

<th>User ID</th>

<th>User name</th>

<th>Role</th>

</tr>

</thead>

<tbody>

<tr th:each=*"users: ${listUsers}"*>

<td th:text=*"${users.user\_id}"*>User ID</td>

<td th:text=*"${users.username}"*>E-mail</td>

<td th:text=*"${users.role}"*>First Name</td>

</tr>

</tbody>

</table>

</div>

</body>

</html>

3. new-shoe.html

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Create Shoe</title>

</head>

<body>

<div align=*"center"*>

<h1>Create New Shoe</h1>

<br />

<form action=*"#"* th:action=*"@{/save}"* th:object=*"${shoes}"*

method=*"post"*>

<table border=*"0"* cellpadding=*"10"*>

<tr>

<td>Shoe Name:</td>

<td><input type=*"text"* th:field=*"\*{name}"* /></td>

</tr>

<tr>

<td>Brand:</td>

<td><input type=*"text"* th:field=*"\*{brand}"* /></td>

</tr>

<tr>

<td>Made In:</td>

<td><input type=*"text"* th:field=*"\*{madein}"* /></td>

</tr>

<tr>

<td>Price:</td>

<td><input type=*"text"* th:field=*"\*{price}"* /></td>

</tr>

<tr>

<td colspan=*"2"*><button type=*"submit"*>Save</button> </td>

</tr>

</table>

</form>

</div>

</body>

</html>

4. edit-shoe.html

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Edit Shoes</title>

</head>

<body>

<div align=*"center"*>

<h1>Edit Shoes</h1>

<br />

<form action=*"#"* th:action=*"@{/save}"* th:object=*"${shoes}"*

method=*"post"*>

<table border=*"0"* cellpadding=*"10"*>

<tr>

<td>Shoes ID:</td>

<td>

<input type=*"text"* th:field=*"\*{id}"* readonly=*"readonly"* />

</td>

</tr>

<tr>

<td>Shoes Name:</td>

<td>

<input type=*"text"* th:field=*"\*{name}"* />

</td>

</tr>

<tr>

<td>Brand:</td>

<td><input type=*"text"* th:field=*"\*{brand}"* /></td>

</tr>

<tr>

<td>Made In:</td>

<td><input type=*"text"* th:field=*"\*{madein}"* /></td>

</tr>

<tr>

<td>Price:</td>

<td><input type=*"text"* th:field=*"\*{price}"* /></td>

</tr>

<tr>

<td colspan=*"2"*><button type=*"submit"*>Save</button> </td>

</tr>

</table>

</form>

</div>

</body>

</html>

5. 403.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Access Denied</title>

</head>

<body>

<h1>Sorry, you don't have permission</h1>

</body>

</html>

11. Create the Controller Classes to map the html pages

a. ShoeController.java

package sportyshoe;

import java.util.List;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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.servlet.ModelAndView;

@Controller

public class ShoeController {

@Autowired

private ShoeService service;

@RequestMapping("/")

public String viewHomePage(Model model) {

List<Shoes> listShoes = service.listAll();

model.addAttribute("listShoes", listShoes);

return "index";

}

@RequestMapping("/new")

public String showNewShoesPage(Model model) {

Shoes shoe = new Shoes();

model.addAttribute("shoes", shoe);

return "new-shoe";

}

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

public String saveShoe(@ModelAttribute("shoes") Shoes shoe) {

service.save(shoe);

return "redirect:/";

}

@RequestMapping("/edit/{id}")

public ModelAndView showEditShoesPage(@PathVariable(name = "id") int id) {

ModelAndView mav = new ModelAndView("edit-shoe");

Shoes shoe = service.get(id);

mav.addObject("shoes", shoe);

return mav;

}

@RequestMapping("/delete/{id}")

public String deleteShoe(@PathVariable(name = "id") int id) {

service.delete(id);

return "redirect:/";

}

}

b. UsersController.java

**package** sportyshoe;

**import** java.util.List;

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

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

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

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

@Controller

**public** **class** UsersController{

@Autowired

UsersService service;

@GetMapping("/users")

**public** String listUsers(Model model) {

List<users> listUsers = service.listAll();

model.addAttribute("listUsers", listUsers);

**return** "ListUsers";

}

}

12. Create the POJO classes:

package sportyshoe;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Shoes {

private Long id;

private String name;

private String brand;

private String madein;

private float price;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

public String getMadein() {

return madein;

}

public void setMadein(String madein) {

this.madein = madein;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

protected Shoes(Long id, String name, String brand, String madein, float price) {

super();

this.id = id;

this.name = name;

this.brand = brand;

this.madein = madein;

this.price = price;

}

protected Shoes() {

}

}

**package** sportyshoe;

**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** users {

@Id

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

**private** Long user\_id;

@Column

**private** String username;

@Column

**private** String password;

@Column

**private** String role;

@Column

**private** **short** enabled;

**public** Long getUser\_id() {

**return** user\_id;

}

**public** **void** setUser\_id(Long user\_id) {

**this**.user\_id = user\_id;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getRole() {

**return** role;

}

**public** **void** setRole(String role) {

**this**.role = role;

}

**public** **short** getEnabled() {

**return** enabled;

}

**public** **void** setEnabled(**short** enabled) {

**this**.enabled = enabled;

}

}

13. Create the Service classes:

package sportyshoe;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

@Transactional

public class ShoeService {

@Autowired

private ShoeRepository repo;

public List<Shoes> listAll() {

return repo.findAll();

}

public void save(Shoes shoe) {

repo.save(shoe);

}

public Shoes get(long id) {

return repo.findById(id).get();

}

public void delete(long id) {

repo.deleteById(id);

}

}

**package** sportyshoe;

**import** java.util.List;

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

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

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

@Service

@Transactional

**public** **class** UsersService {

@Autowired

**private** UsersRepository repo;

**public** List<users> listAll() {

**return** repo.findAll();

}

}

14. Create the repository classes:

**package** sportyshoe;

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

**public** **interface** UsersRepository **extends** JpaRepository<users, Long> {

}

**package** sportyshoe;

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

**public** **interface** ShoeRepository **extends** JpaRepository<Shoes, Long> {

}

15 Password Generator main class for generating the passwords using Bcrypt api

**package** sportyshoe;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**public** **class** PasswordGenerator {

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

// **TODO** Auto-generated method stub

BCryptPasswordEncoder encoder = **new** BCryptPasswordEncoder();

String rawpassword = "password";

String encodedpassword =encoder.encode(rawpassword);

System.***out***.println(encodedpassword);

}

}

16. WebSecurity Config class to use form based authentication and JDBC

package sportyshoe;

import javax.sql.DataSource;

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

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.\*;

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

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private DataSource dataSource;

@Autowired

public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication().passwordEncoder(new BCryptPasswordEncoder())

.dataSource(dataSource)

.usersByUsernameQuery("select username, password, enabled from users where username=?")

.authoritiesByUsernameQuery("select username, role from users where username=?")

;

}

@Override

protected void configure(HttpSecurity http) throws Exception {

// TODO Auto-generated method stub

http.authorizeRequests()

.antMatchers("/new").permitAll()

.antMatchers("/edit/\*", "/delete/\*").permitAll()

.anyRequest().authenticated()

.and()

.formLogin().permitAll()

.and()

.logout().permitAll()

.and()

.exceptionHandling().accessDeniedPage("/403")

;

}

}

17. MVC config class to handle exceptions for 403 unauthorized usage

**package** sportyshoe;

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

**import** org.springframework.web.servlet.config.annotation.ViewControllerRegistry;

**import** org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

**public** **class** MvcConfig **implements** WebMvcConfigurer {

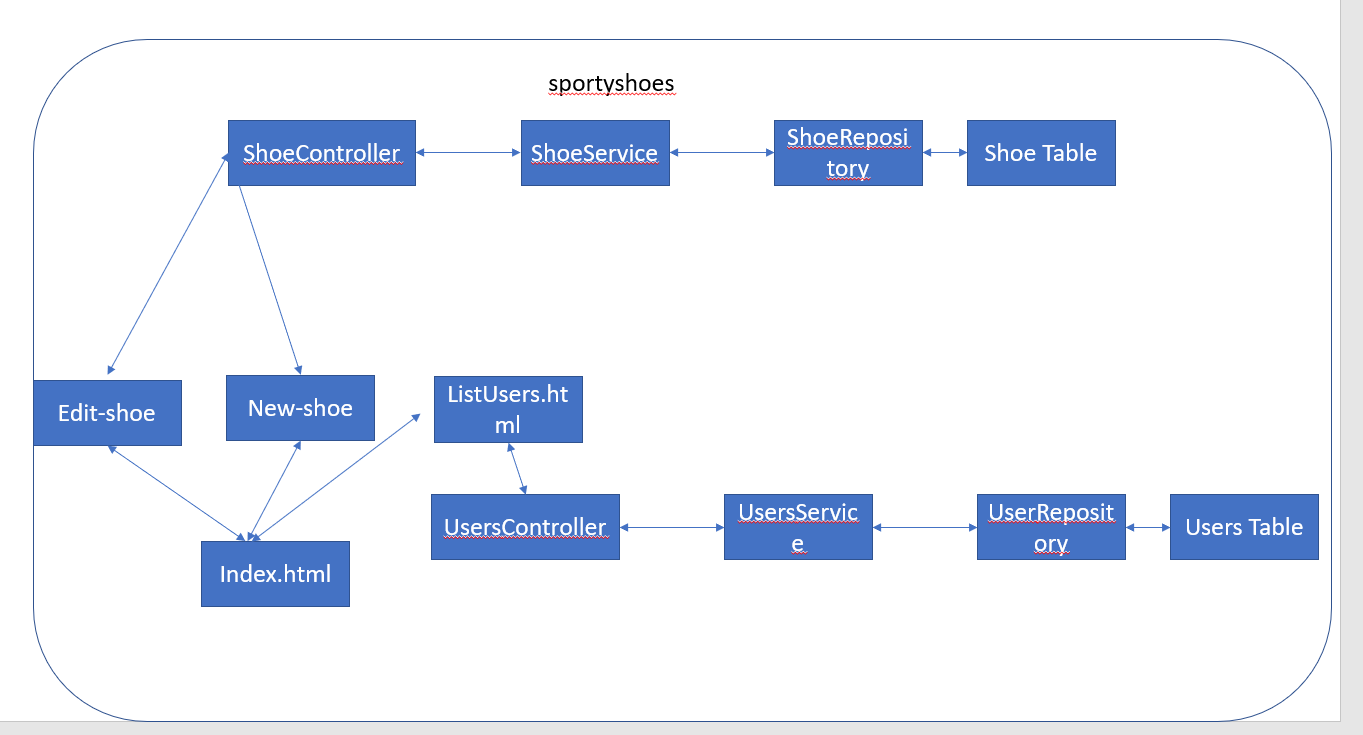
**public** **void** addViewControllers(ViewControllerRegistry registry) {

registry.addViewController("/403").setViewName("403");

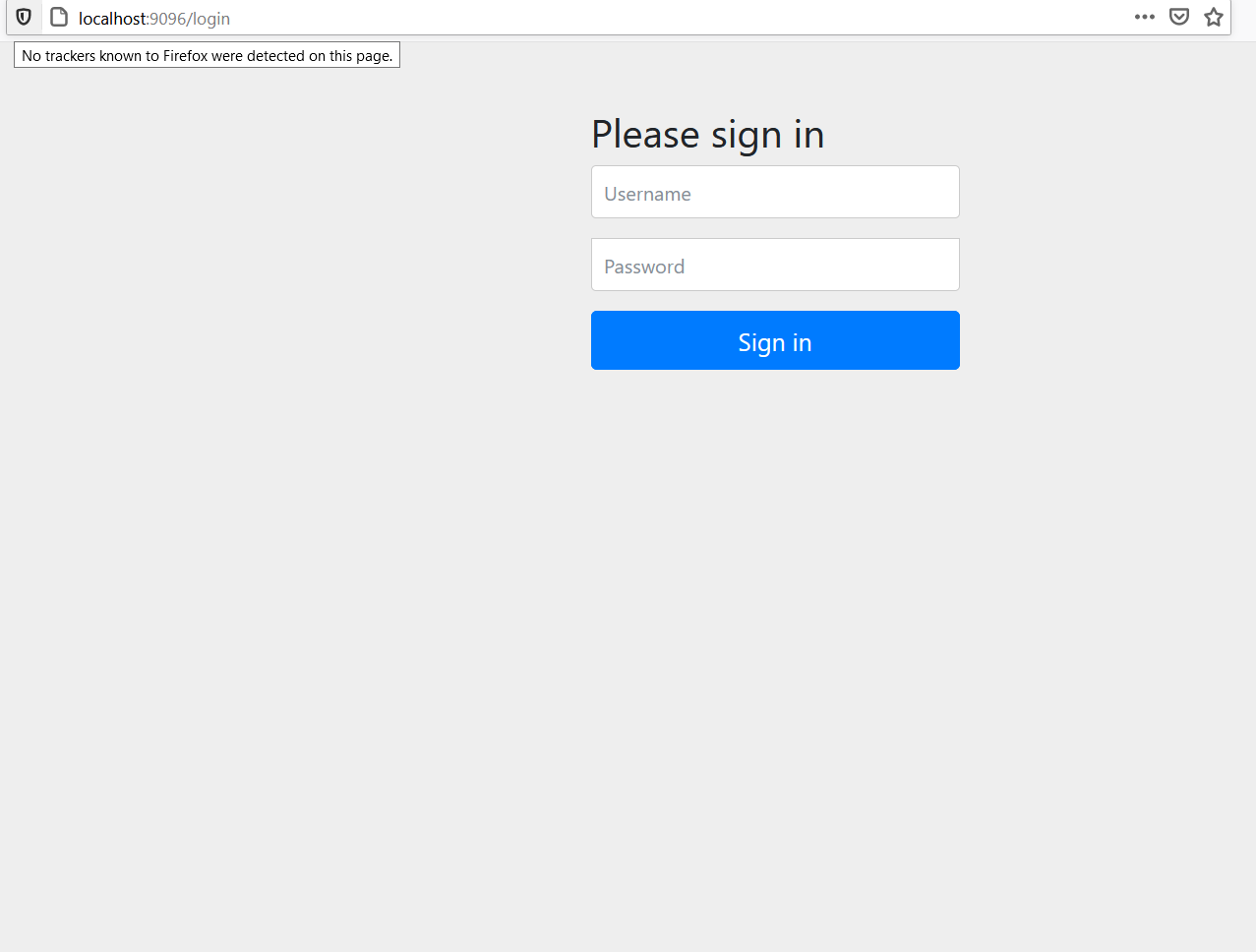
}

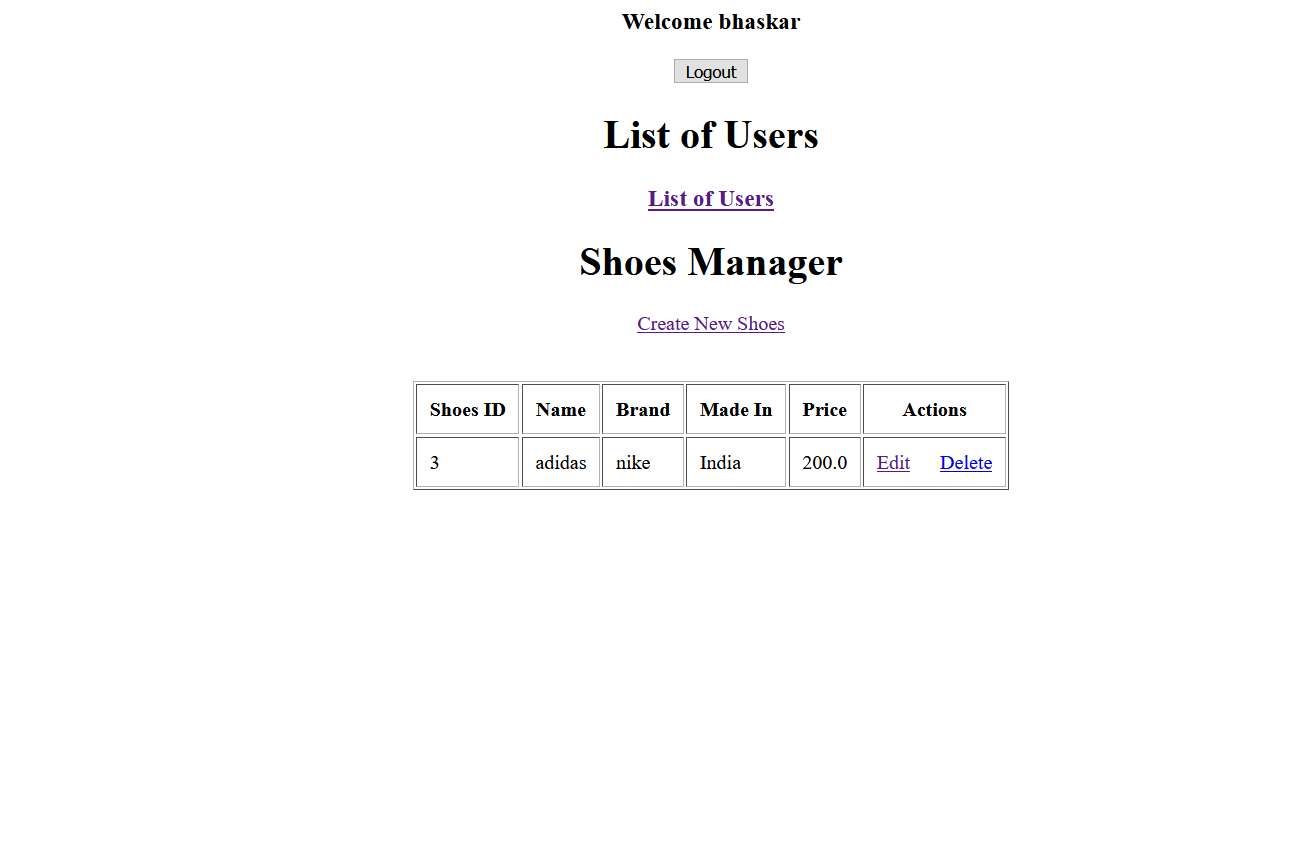
}

18.

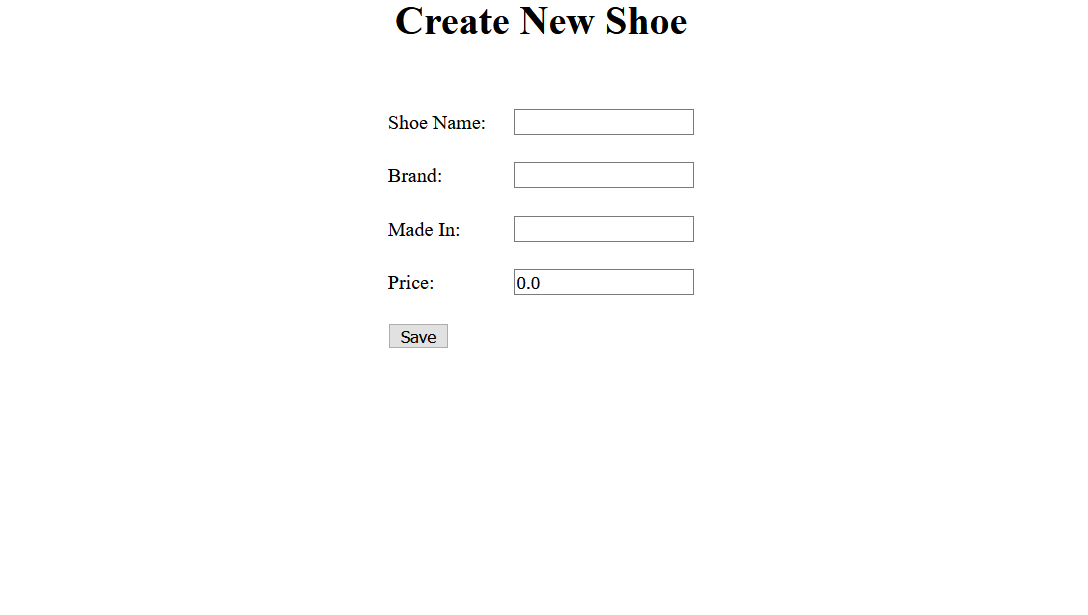


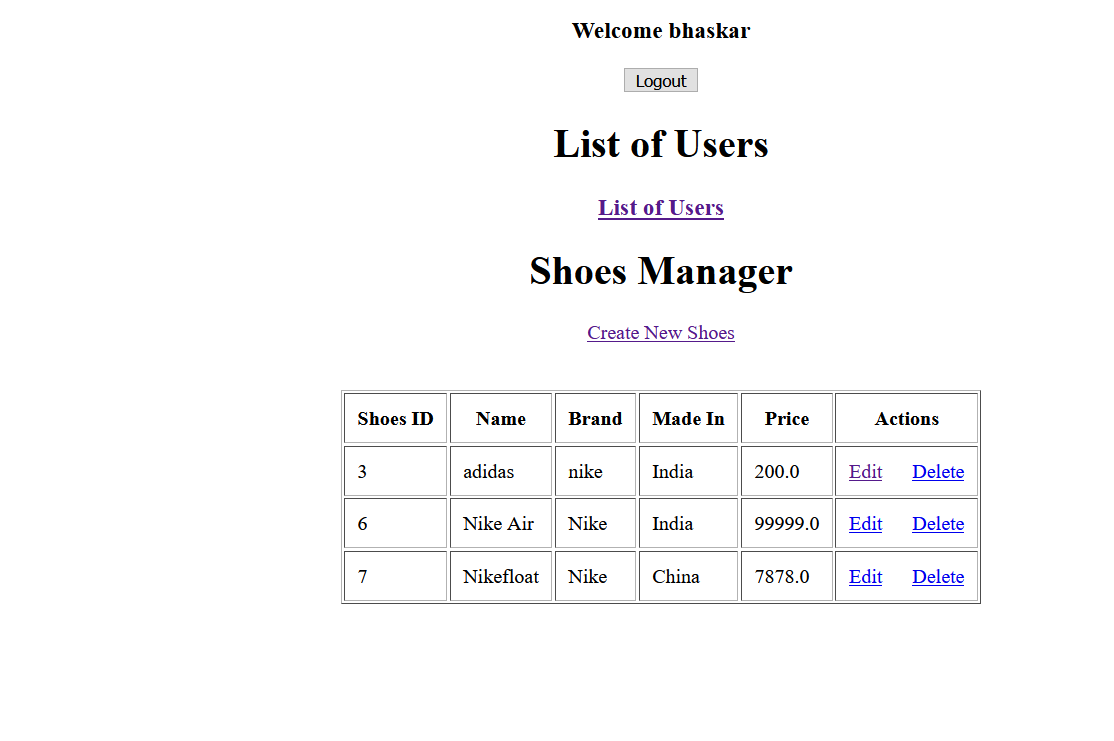
19 Pictures for App running:

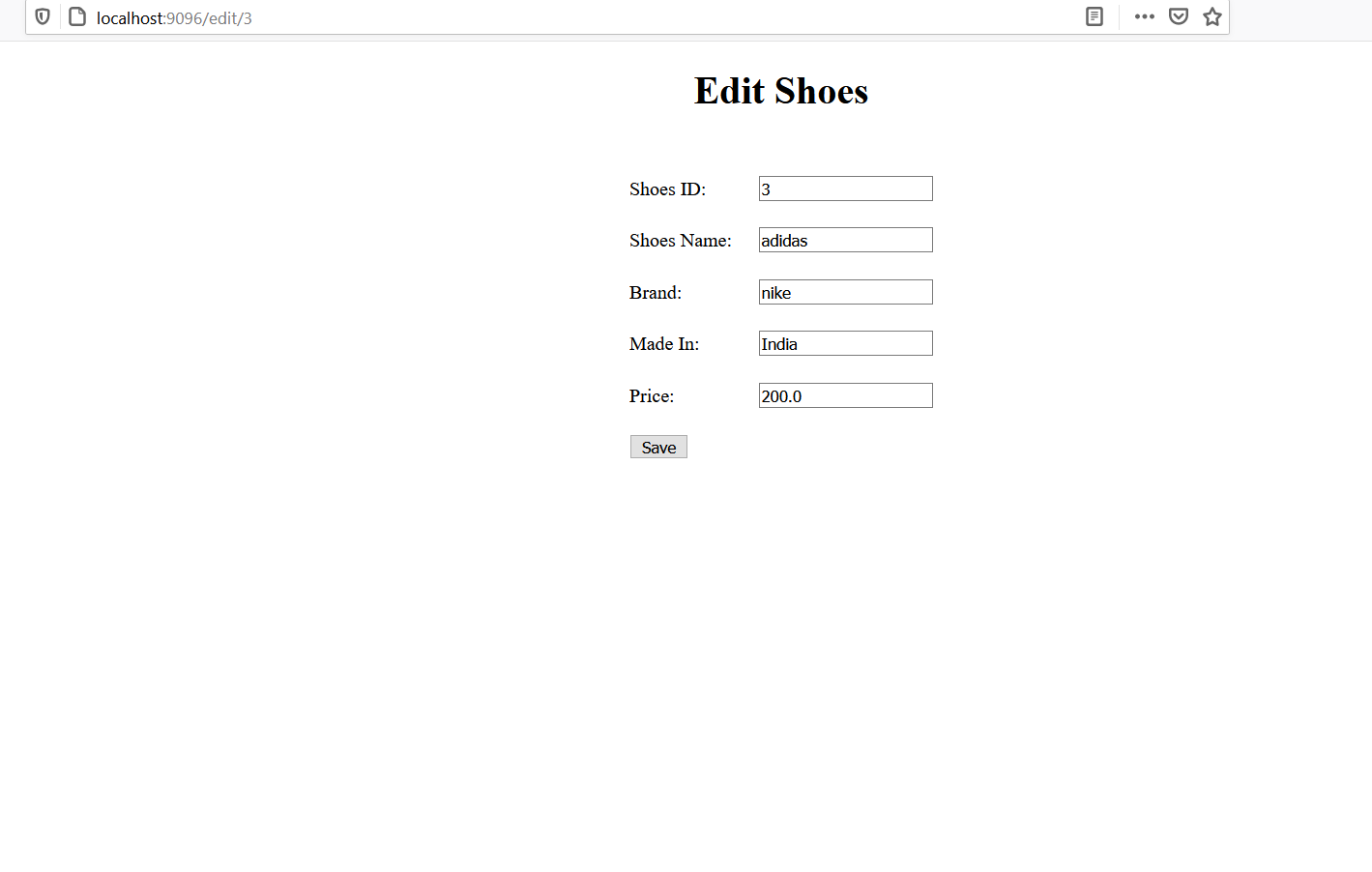


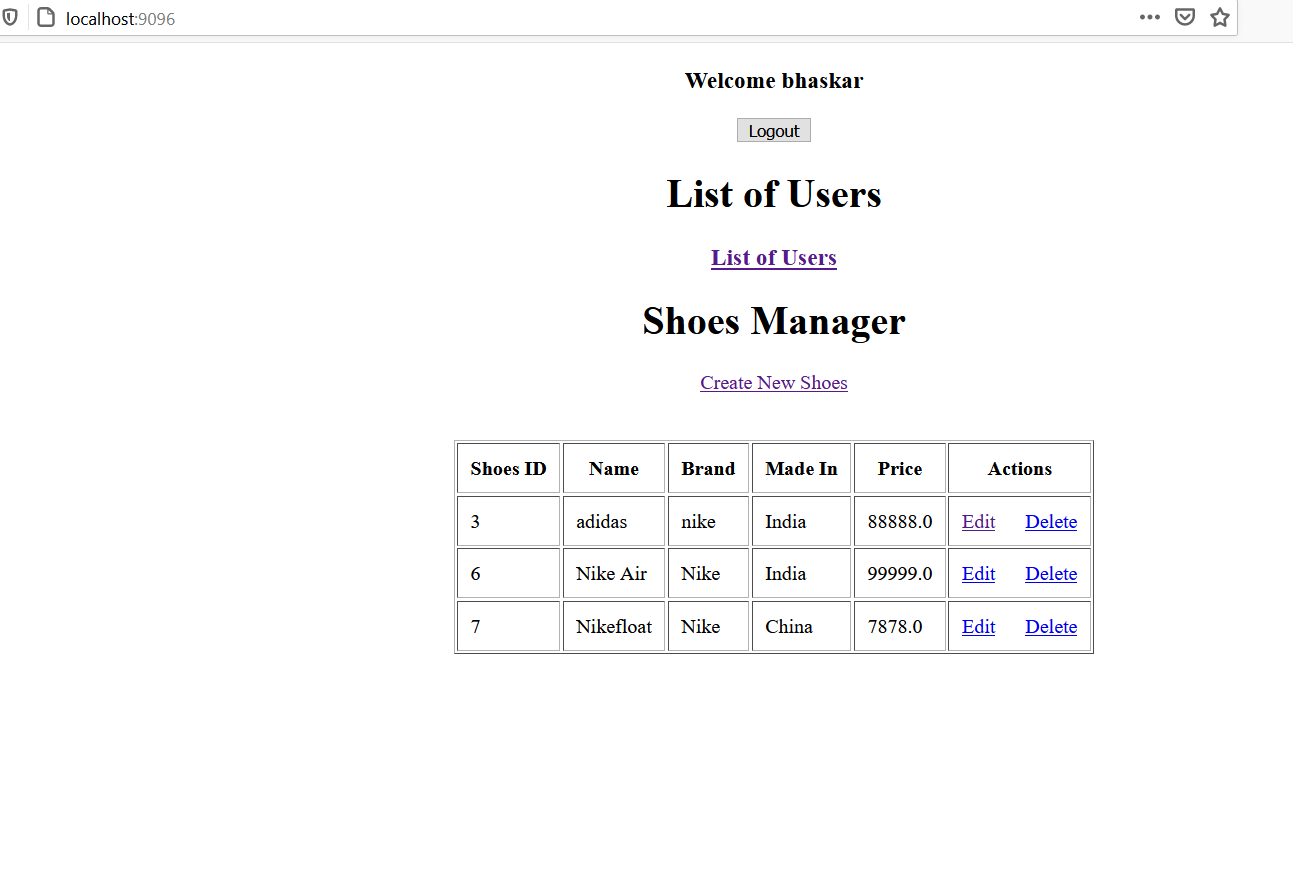


After clicking create new shoe

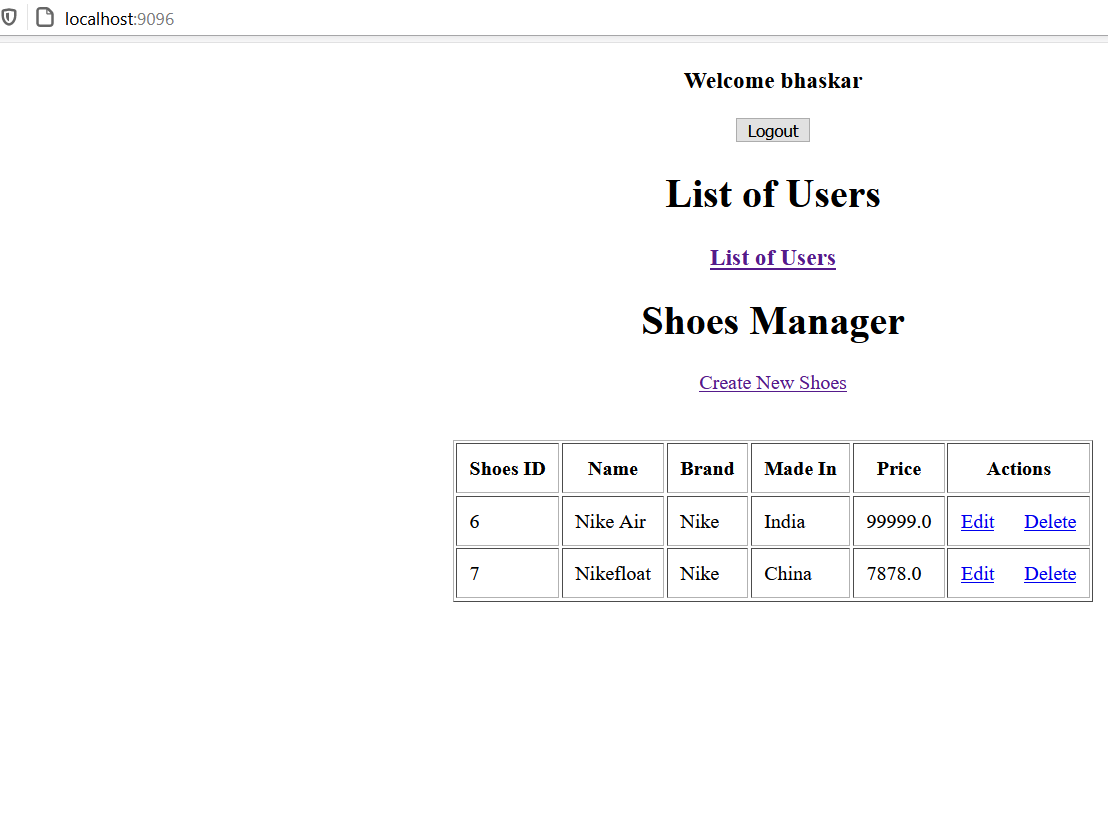








After clicking delete



After clicking list users



After press logout

