Spring Security implementation in the given code

1. Adding dependency.

a.In the pom.xml file of the backendproject , add the property in the propertieslist.

<springsecurity.version>4.2.2.RELEASE</springsecurity.version>

b.Add the following dependencies in the depenedencies tag.

<!-- Spring Security Config Dependency -->

<dependency>

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

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

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

</dependency>

<!-- Spring Security Taglibs Dependency -->

<dependency>

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

<artifactId>spring-security-taglibs</artifactId>

<version>${springsecurity.version}</version>

</dependency>

2. Save the project to download dependency files.

3. Modify the existing model class , User to accommodate the convention defaults , here the data of table has to have columns with the name password and role which will be used by spring security , hence that has to be updated.

a. Goto the user.java in the backendproject model folder ,

@Size(min=1, max=30,message="user password should be between 1 to 30 characters long")

private String password;

@NotBlank

private String role;

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

b. Update the spring form of registration to match the same.

c. Also the testcases for userdaoimpl methods have to be altered.

4. Update web.xml to make all requests pass through this.Add:

<!-- Loading the configuration file -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/applicationContext.xml,

/WEB-INF/application-security.xml,

/WEB-INF/dispatcher-servlet.xml

</param-value>

</context-param>

<!-- Spring Security Filter -->

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

5. Add a xml file , applicationContext.xml ,at same level of the dispatcher-servlet.xml .

a. Cut the mvc:annotationdriver,context-scan and mvc:resources tags from dispatcher-servlet.

b. Add these in the applicationContext.xml.Add:

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:webflow="http://www.springframework.org/schema/webflow-config"

xsi:schemaLocation="

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/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/webflow-config

http://www.springframework.org/schema/webflow-config/spring-webflow-config.xsd">

<context:component-scan base-package="net.sports.ZenSportsFrontEnd" />

<context:component-scan base-package="net.sports.ZenSportsBackEnd" />

<!--Loading static resources -->

<mvc:annotation-driven />

<mvc:resources location="/assets/" mapping="/resources/\*\*" />

</beans>

6. In the HibernateConfig.java file of the backendproject , inside the config folder , add the bean declaration at the top of the getDataSource() method.Add:

@Bean(name="dataSource")

public DataSource getDataSource() {

BasicDataSource dataSource = new BasicDataSource();

// Providing the database connection information

dataSource.setDriverClassName(DATABASE\_DRIVER);

dataSource.setUrl(DATABASE\_URL);

dataSource.setUsername(DATABASE\_USERNAME);

dataSource.setPassword(DATABASE\_PASSWORD);

return dataSource;

}

7.Add a xml file , application-security.xml , at same level of applicationContext and dispatcher-servlet.xml.

a.Add the following inside it :

<beans:beans xmlns="http://www.springframework.org/schema/security"

xmlns:beans="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/security

http://www.springframework.org/schema/security/spring-security.xsd">

<!-- No security for static resources -->

<http pattern="/resources/\*\*" security="none" />

<!-- Security for the rest of my mappings -->

<http>

<intercept-url pattern="/admin/\*\*" access="hasAuthority('ADMIN')" />

<intercept-url pattern="/user/\*\*" access="hasAuthority('User')" />

<!-- <intercept-url pattern="/supplier/\*\*" access="hasAuthority('SUPPLIER')"

/> -->

<intercept-url pattern="/\*\*" access="permitAll" />

<form-login login-page="/login" default-target-url="/"

authentication-failure-url="/login?error" />

<logout logout-success-url="/" />

<csrf disabled="true" />

<!--this is to avoid redirecting to other websites (cross site request

forgery) -->

</http>

<authentication-manager>

<authentication-provider>

<jdbc-user-service data-source-ref="dataSource"

users-by-username-query="SELECT username, password, enabled FROM user WHERE username = ?"

authorities-by-username-query="SELECT username, role FROM user WHERE username = ?" />

</authentication-provider>

<!--name of the bean has to be renamed to dataSource -->

</authentication-manager>

</beans:beans>

7.Inside the navbar.jsp of the shared folder of view,update the code:

<%@taglib prefix="security" uri="http://www.springframework.org/security/tags"%>

<nav class="navbar navbar-default" style="font-size: 8pt">

<div class="collapse navbar-collapse" id="menuBar">

<!-- Navbar-fixed-top -->

<ul class="nav navbar-nav" style="font-size: 8pt">

<li id="home"><a href="${contextRoot}/index"><img

src="${images}/zensports.png" style="height: 20pt; width: 100pt" /></a></li>

<li id="about"><a href="${contextRoot}/about"><span

class="glyphicon glyphicon-info-sign"></span> About Us</a></li>

<li id="contact"><a href="${contextRoot}/contact"><span

class="glyphicon glyphicon-envelope"></span> Contact Us</a></li>

<security:authorize access="hasAuthority('User')">

<li id="cart"><a href="/#/"><span

class="glyphicon glyphicon-shopping-cart"></span> Cart</a></li>

<li id="products"><a href="${contextRoot}/user/products"><span

class="glyphicon glyphicon-list"></span> View All</a></li>

</security:authorize>

<security:authorize access="hasAuthority('ADMIN')">

<li class="dropdown"><a class="dropdown-toggle"

data-toggle="dropdown" href="/#/"><span

class="glyphicon glyphicon-user"></span> ADMIN<span class="caret"></span></a>

<ul class="dropdown-menu">

<li><a href="${contextRoot}/admin/productCRUD"><span

class="glyphicon glyphicon-list"></span>Product Management</a></li>

</ul></li>

</security:authorize>

</ul>

<ul class="nav navbar-nav navbar-right" style="font-size: 8pt">

<security:authorize access="isAnonymous()">

<li id="register"><a href="register"><span

class="glyphicon glyphicon-plus"></span> Register</a></li>

<li id="login"><a href="${contextRoot}/login"><span

class="glyphicon glyphicon-user"></span> Login</a></li>

</security:authorize>

<security:authorize access="isAuthenticated()">

<li id="logout"><a href="${contextRoot}/logout">Logout</a></li>

</security:authorize>

</ul>

</div>

</nav>

8.Inside the views folder , add a jsp file , login.jsp , inside it , add:

<%@ page language="java" isELIgnored="false"%>

<%@taglib prefix="s" uri="http://www.springframework.org/tags"%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Login Page</title>

<s:url value="/resources/images" var="images" />

<s:url value="/resources/js" var="js" />

<s:url value="/resources/css" var="css" />

<link rel="stylesheet" href="${css}/bootstrap.min.css" />

<link rel="stylesheet" href="${css}/style.css" />

<script src="${js}/jquery.js"></script>

<script src="${js}/bootstrap.min.js"></script>

</head>

<body>

<div class="wrapper">

<div class="content">

<div class="col-md-2"></div>

<div class="col-md-8 jumbotron">

<h1>Welcome to Login</h1>

<form class="form" action="login" method="POST">

<c:if test="${error}">

<p style="color:red;font-size:20px">${error}</h1>

</c:if>

<div class="form-group">

<label> <span class="glyphicon glyphicon-user"></span>

USERNAME

</label> <input type="text" class="form-control" name="username" required>

</div>

<div class="form-group">

<br /> <label> <span class="glyphicon glyphicon-pencil"></span>

PASSWORD

</label> <input type="password" class="form-control" name="password"

required>

</div>

<div>

<input type="submit" class="btn btn-default" value="Login"/>

</div>

</form>

<br /> <br /> <a href="#/register"><span

class="glyphicon glyphicon-wrench"></span> Signup</a>

</div>

<div class="col-md-2"></div>

</div>

<div class="row footer">

<div class="col-md-12">

<nav class="navbar navbar">

<center>

<p>Copyright 2017 © ZenSports.com | All Rights Reserved By

Neha@NIIT</p>

</center>

</nav>

</div>

</div>

</div>

</body>

9.In the frontendproject , goto the frontendproductcontroller , and update the mappings to accommodate roles , admin and user, update the following in the bosy of the class:

@Autowired

private IProductDAO productDAO;

@Autowired

private HttpServletRequest request;

@RequestMapping(value = { "/user/products" })

public ModelAndView about() {

ModelAndView model = new ModelAndView("page");

model.addObject("title", "Products");

model.addObject("userClickProducts", true);

model.addObject("list", productDAO.getProducts());

return model;

}

@RequestMapping(value = { "/product" })

public @ResponseBody List<Product> product() {

return productDAO.getProducts();

}

@RequestMapping("/user/showProduct")

public ModelAndView showProduct() {

ModelAndView model = new ModelAndView("Products");

model.addObject("list", productDAO.getProducts());

return model;

}

@RequestMapping("/admin/productCRUD")

public ModelAndView productcrud() {

ModelAndView model = new ModelAndView("page");

model.addObject("title", "Product Management");

model.addObject("userClickProductCRUD", true);

model.addObject("product", new Product());

model.addObject("productList", productDAO.getProducts());

return model;

}

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

public ModelAndView doActions(@ModelAttribute("product") @Valid Product product, BindingResult result,

@RequestParam String action) {

Product productResult = new Product();

if (result.hasErrors()) {

ModelAndView model1 = new ModelAndView("page");

model1.addObject("title", "Product Management");

model1.addObject("userClickProductCRUD", true);

return model1;

}

switch (action.toLowerCase()) {

case "add":

if(!(product.getFile().getOriginalFilename().equals("")))

{

product.setProductImage(uploadImage(product.getFile()));

}

productDAO.addProducts(product);

productResult = product;

break;

case "edit":

productDAO.updateProduct(product);

productResult = product;

break;

case "delete":

productDAO.deleteProduct(product.getProductId());

productResult = product;

break;

case "search":

Product searchedProduct = productDAO.getProduct(product.getProductId());

productResult = searchedProduct != null ? searchedProduct : new Product();

break;

}

ModelAndView model1 = new ModelAndView("page");

model1.addObject("product", productResult);

model1.addObject("productList", productDAO.getProducts());

model1.addObject("title", "Product Management");

model1.addObject("userClickProductCRUD", true);

return model1;

}

protected String uploadImage(MultipartFile multipart)

{

String folderToUpload="/resources/images/";

String filename=multipart.getOriginalFilename();

String realPath=request.getServletContext().getRealPath(folderToUpload);

if(!new File(realPath).exists())

{

new File(realPath).mkdirs();

}

String filePath=realPath+filename;

File destination=new File(filePath);

try

{

multipart.transferTo(destination);

}

catch(Exception ex){}

return filename;

}

10.Goto the frontendpagecontroller of the frontendproject , isnde it add one more mapping :

@GetMapping("/login")

public String login(@RequestParam(value = "error", required = false)String error, Model model) {

if(error!=null) {

model.addAttribute("error","Authentication Failed - Invalid credentials!");

}

model.addAttribute("title", "page");

return "login";

}