Module No:	1	IU No:	6	Exercise No.	1

Lab Assessment Statement

Assignment 5 - Spring Security, Error Handling, and Logging

This assignment is about implementing Authentication & Authorization solution for Know-Your-Neighborhood application.

Spring Security

1. Create user, role and user_role tables in the database.

    a. *[Note: The words user & role are reserved in some databases.. Use some other name, if so.]*

2. Add entries into user, role and user_role tables.

    a. Add two entries into role table. One role is to view the store info (VIEW_STORE) and the other role is to add/modify store info (ADD_STORE).
    b. Add two entries into user table. Let one user have play VIEW_STORE role, another user play only both VIEW_STORE & ADD_STORE roles.

3. Enhance Spring Data JPA enabled version of 'Know-Your-Neighborhood' application to support authentication and authorization.
4. Authentication:

    a. Create entities for User and Role.
    b. Create a login page for the application. User should be forced to login before visiting any of the application page.
    c. Authenticate user with the credentials entered against the database entries added in the previous step.
    d. Show a message if user enters invalid user and password. And then allow him to re-enter the credentials.

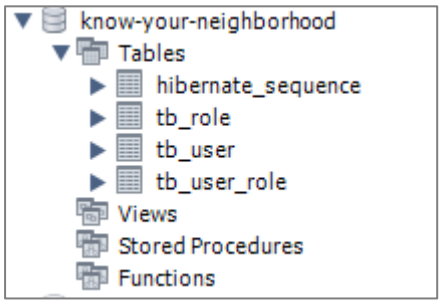
5. Authorization:

    a. Once the user loggedin, make sure that only the user with ADD_STORE role can add/modify store data. The user with VIEW_STORE data can only view the stores entered.
    b. If there are no stores in the system, show appropriate message to the user.

6. Add CSRF security to the application

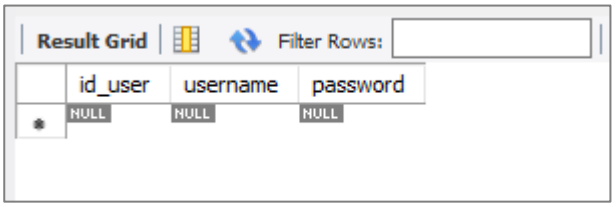
	Logging 1. Develop 5 unit test cases to validate the role based authorization **Logging** 1. Implement the rolling file appender and log at the class level **Error Handling** 1. Create Global Exception Handler that can handle an exception. Add method to handle run away exception. [Run away exception is a RuntimeException that is not handled by any exception handler]. 2. Create an exception class StoreNotFoundException. 3. Create an end point to search store by email. 4. Create exception handler specific to the controller. 5. Let the service throw StoreNotFoundException if no store is found for given email. 6. Let the StoreNotFoundException handler route to view stores page and show the error message in view stores page. 7. Enter the url http://<host>:<port>/stores?email@test.com. This should route to view stores page and show the error message.
Technical Environment	-
Guidelines	-
Duration	180 mins

1. Create user, role and user_role tables in the database.



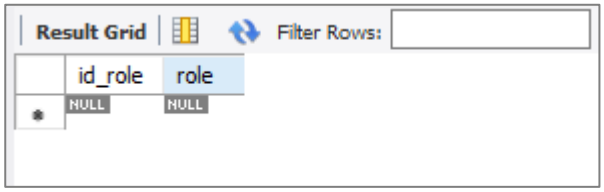
- **tb_user table**

```
CREATE TABLE tb_user (
`id_user` INT NOT NULL AUTO_INCREMENT,
`username` VARCHAR(45) NOT NULL,
`password` VARCHAR(45) NOT NULL,
PRIMARY KEY (`id_user`)
) ENGINE = InnoDB;
```



- **tb_role table**

```
CREATE TABLE tb_role (
`id_role` INT NOT NULL AUTO_INCREMENT,
`role` VARCHAR(45) NOT NULL,
PRIMARY KEY (`id_role`)
) ENGINE = InnoDB;
```



- **tb_user_role table**

```
CREATE TABLE tb_user_role (
`id_user_role` INT NOT NULL AUTO_INCREMENT,
`id_user` INT NOT NULL,
`id_role` INT NOT NULL,
PRIMARY KEY (`id_user_role`),
FOREIGN KEY (`id_user`) REFERENCES tb_user (`id_user`),
FOREIGN KEY (`id_role`) REFERENCES tb_role (`id_role`)
) ENGINE = InnoDB;
```

2. Add entries into user, role, and user_role tables.

   a. Add two entries to the role table. One role is to view the store info (VIEW_STORE) and the other role is to add/modify store info (ADD_STORE).

```
INSERT INTO tb_role (role) VALUES ("ROLE_VIEW_STORE"), ("ROLE_ADD_STORE");
```

Result Grid | Filter Rows:

id_role	role
1	ROLE_VIEW_STORE
2	ROLE_ADD_STORE
NULL	NULL

   b. Add two entries into user table. Let one user have play VIEW_STORE role, another user play only both VIEW_STORE & ADD_STORE roles.

```
INSERT INTO tb_user_role (id_user, id_role) VALUES
("1", "1"),
("2", "1"),
("2", "2");

SELECT UR.id_user, U.username, R.role FROM tb_user as U
INNER JOIN tb_user_role as UR
ON (U.id_user = UR.id_user)
INNER JOIN tb_role as R
ON (UR.id_role = R.id_role);
```

Result Grid | Filter Rows:

id_user	username	password
1	viewstore	ROLE_VIEW_STORE
2	addstore	ROLE_VIEW_STORE
2	addstore	ROLE_ADD_STORE

3. Enhance Spring Data JPA enabled version of 'Know-Your-Neighborhood' application to support authentication and authorization

   ✓ Add maven Dependency

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

   ✓ Create security config class

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

  @Bean
  public static PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }

  @Bean
  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

    // Authorize
    http.authorizeRequests(configurer -> configurer
        .antMatchers("/").authenticated()
        .antMatchers("/stores").hasRole("VIEW_STORE")
        .antMatchers("/addStore/**").hasRole("ADD_STORE")
        .antMatchers("/editStore/**").hasRole("ADD_STORE")
        .antMatchers("/deleteStore").hasRole("ADD_STORE"));

    // Form Login
    http.formLogin(form -> form
        .loginPage("/login")
        .loginProcessingUrl("/loginUser")
        .permitAll());

    // Logout
    http.logout(logout -> logout
        .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
        .permitAll());

    return http.build();
  }

}
```

4. Authentication

✓ Create entities for User and Role.

Users

```
@Entity
@Table(name = "tb_user")
public class Users {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "id_user")
  private int idUser;

  private String username;
  private String password;

  @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
  @JoinTable(name = "tb_user_role", joinColumns = @JoinColumn(name = "id_user"),
inverseJoinColumns = @JoinColumn(name = "id_role"))
  private List<Roles> roles = new ArrayList<>();

  public Users() {
  }

  public Users(String username, String password) {
    this.username = username;
    this.password = password;
  }

  public int getIdUser() {
    return idUser;
  }

  public void setIdUser(int idUser) {
    this.idUser = idUser;
  }

  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 List<Roles> getRoles() {
    return roles;
  }

  public void setRoles(List<Roles> roles) {
    this.roles = roles;
  }

}
```

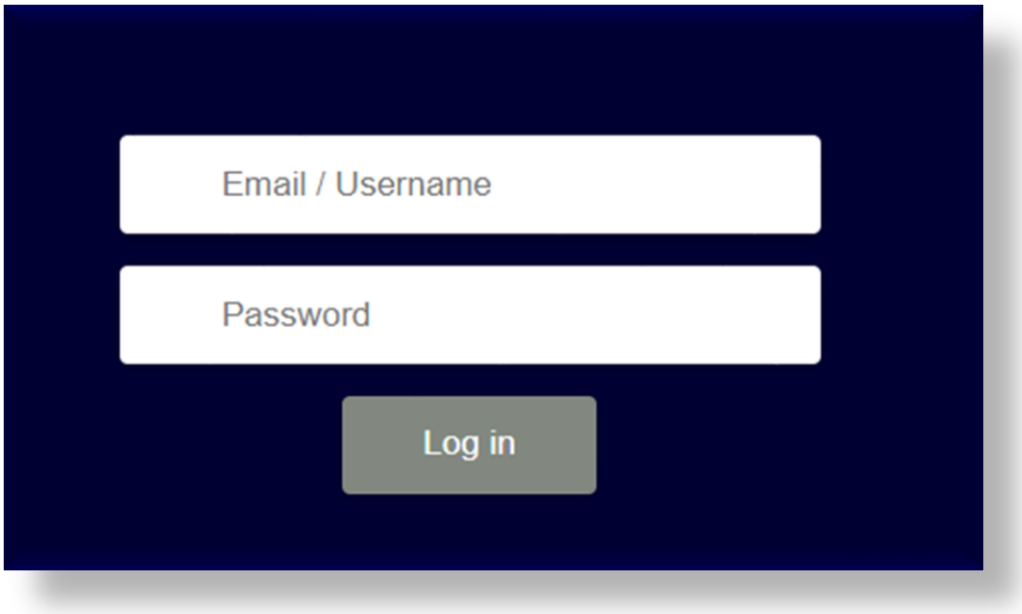
Roles

```
@Entity
@Table(name = "tb_role")
public class Roles {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "id_role")
  private int idRole;

  @Column(name = "role")
  private String role;

  @ManyToMany(mappedBy = "roles")
  private List<Users> users = new ArrayList<>();

  public Roles() {
  }

  public int getIdRole() {
    return idRole;
  }

  public void setIdRole(int idRole) {
    this.idRole = idRole;
  }

  public String getRole() {
    return role;
  }

  public void setRole(String role) {
    this.role = role;
  }

  public List<Users> getUsers() {
    return users;
  }

  public void setUsers(List<Users> users) {
    this.users = users;
  }

}
```

- ✓ Create a login page for the application. User should be forced to login before visiting any of the application page.

Login.jsp

```
<%@ include file="jsp-tags.jsp" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1" />
    <title>Know Your Neighborhood</title>
    <style type="text/css">
      label,
      input {
        display: block;
      }
      .error {
        color: red;
      }
    </style>
  </head>
  <body>
    <form:form action="loginUser" modelAttribute="user">
      <c:if test="${param.error != null}">
        <p class="error">Invalid username or password</p>
      </c:if>
      <label>Username</label>
      <form:input type="text" path="username" />

      <label>Password</label>
      <form:input type="password" path="password" />

      <input
        type="hidden"
        name="${_csrf.parameterName}"
        value="${_csrf.token}"
      />

      <button type="submit">Login</button>
    </form:form>
  </body>
</html>
```

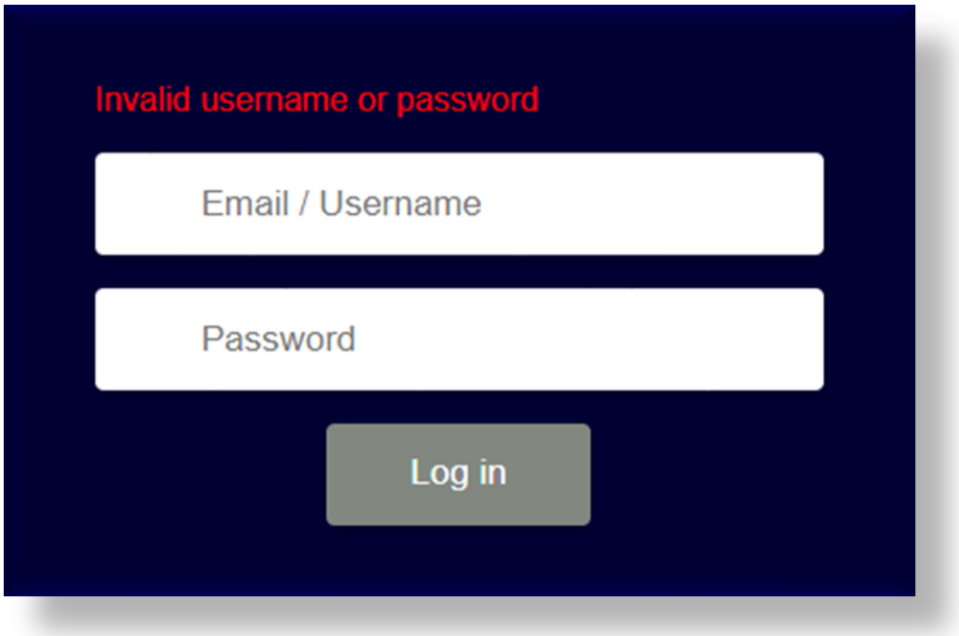

Email / Username
Password
Log in

- ✓ Authenticate user with the credentials entered against the database entries added in the previous step.

```
package com.lithan.a5.config;

import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import com.lithan.a5.entity.Users;
import com.lithan.a5.repository.UsersRepository;

@Service
public class CustomUserDetailsService implements UserDetailsService {

  @Autowired
  private UsersRepository userRepo;

  @Override
  public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

    Users user = userRepo.findByUsername(username);

    if (user != null) {
      return new User(user.getUsername(), user.getPassword(),
          user.getRoles().stream().map((userRole -> new
SimpleGrantedAuthority(userRole.getRole())))
              .collect(Collectors.toList()));
    } else {
      throw new UsernameNotFoundException("Invalid username or password");
    }
  }
}
```

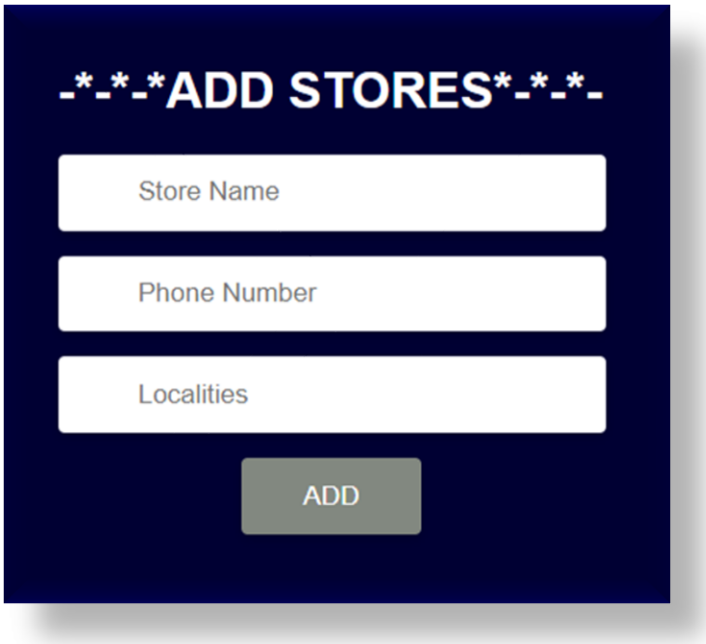
✓ Show a message if user enters invalid user and password. And then allow him to re-enter the credentials.

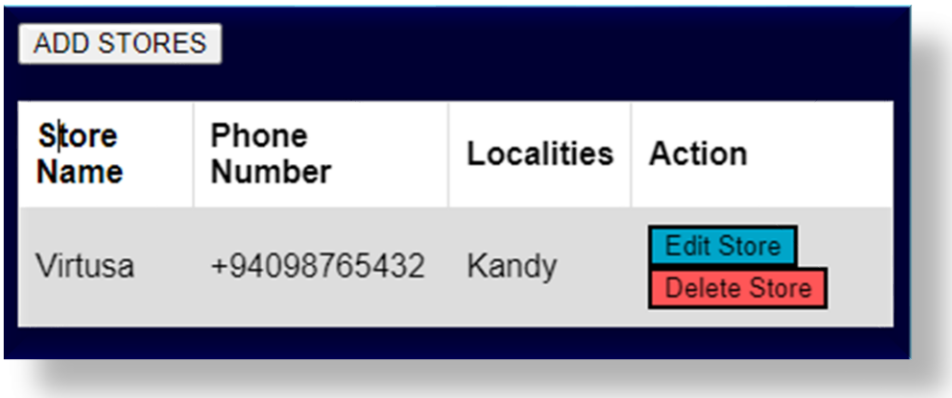
```
<c:if test="${param.error != null}">
  <p class="error">Invalid username or password</p>
</c:if>
```



5. Authorization

✓ Once the user logged in, make sure that only the user with ADD_STORE role can add/modify store data. The user with VIEW_STORE data can only view the stores entered.






```
@Bean
public static PasswordEncoder passwordEncoder() {
  return new BCryptPasswordEncoder();
}

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

  // Authorize
  http.authorizeRequests(configurer -> configurer
      .antMatchers("/").authenticated()
      .antMatchers("/stores").hasRole("VIEW_STORE")
      .antMatchers("/addStore/**").hasRole("ADD_STORE")
      .antMatchers("/editStore/**").hasRole("ADD_STORE")
      .antMatchers("/deleteStore").hasRole("ADD_STORE"));
```

- ✓ If there are no stores in the system, show appropriate message to the user.

```
<c:if test="${stores.isEmpty()}">
  <tr>
    <td colspan="4" style="text-align: center">No stores</td>
  </tr>
</c:if>
```

- ✓ Add CSRF security to the application

```
<input
type="hidden"
name="${_csrf.parameterName}"
value="${_csrf.token}"/>
```

Unit Test

```
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = SecurityConfigTest.class)
@WebAppConfiguration
class SecurityTest {

  @Autowired
  private WebApplicationContext context;

  private MockMvc mvc;

  @BeforeEach
  public void setup() {
    mvc = MockMvcBuilders
        .webAppContextSetup(context)
        .apply(springSecurity())
        .build();
  }
```

```
@Test
  @WithMockUser(value = "user")
  public void testHomeWithUser() throws Exception {
    mvc.perform(get("/"));
  }

  @Test
  @WithMockUser(value = "viewstore", roles = "VIEW_STORE")
  public void testViewStoreRole() throws Exception {
    mvc.perform(get("/stores"));
  }

  @Test
  @WithMockUser(value = "addstore", roles = "ADD_STORE")
  public void testAddStoreRole() throws Exception {
    mvc.perform(get("/addStore"));
  }

  @Test
  @WithMockUser()
  public void testNotFoundURL() throws Exception {
    mvc.perform(get("/bad-url")).andExpect(status().isNotFound());
  }

  @Test
  public void testWithCSRF() throws Exception {
    mvc.perform(post("/").with(csrf()));
  }

}
```

✓ Develop 5-unit test cases to validate the role-based authorization

```
package com.lithan.a5;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.test.context.support.WithMockUser;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import static
org.springframework.security.test.web.servlet.setup.SecurityMockMvcConfigurers.*;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
import static
org.springframework.security.test.web.servlet.request.SecurityMockMvcRequestPostProce
ssors.*;

@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = SecurityConfigTest.class)
@WebAppConfiguration
class SecurityTest {

  @Autowired
  private WebApplicationContext context;

  private MockMvc mvc;

  @BeforeEach
  public void setup() {
    mvc = MockMvcBuilders
        .webAppContextSetup(context)
        .apply(springSecurity())
        .build();
  }

  @Test
  @WithMockUser(value = "user")
  public void testHomeWithUser() throws Exception {
    mvc.perform(get("/"));
  }

  @Test
  @WithMockUser(value = "viewstore", roles = "VIEW_STORE")
  public void testViewStoreRole() throws Exception {
    mvc.perform(get("/stores"));
  }

  @Test
  @WithMockUser(value = "addstore", roles = "ADD_STORE")
  public void testAddStoreRole() throws Exception {
    mvc.perform(get("/addStore"));
  }

  @Test
  @WithMockUser()
  public void testNotFoundURL() throws Exception {
    mvc.perform(get("/bad-url")).andExpect(status().isNotFound());
  }

  @Test
  public void testWithCSRF() throws Exception {
    mvc.perform(post("/").with(csrf()));
  }

}
```

Logging

- ✓ Implement the rolling file appender and log at the class level

StoreController

```
@Controller
public class StoreController {

  private static final Logger LOGGER =
LoggerFactory.getLogger(StoreController.class);

  @Autowired
  StoreService storeService;

  @GetMapping("/")
  public ModelAndView home() {
    LOGGER.info("Inside home of StoreController");

    ModelAndView mv = new ModelAndView("store");

    List<Store> stores = storeService.listStore();

    mv.addObject("stores", stores);

    return mv;
  }
```

application.properties

```
spring.mvc.view.prefix=/WEB-INF/
spring.mvc.view.suffix=.jsp

##Database
spring.datasource.url=jdbc:mysql://localhost:3306/a5_sts
spring.datasource.username=root
spring.datasource.password=root123

##JPA-HIBERNATE
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
## CREATE,UPDATE,CREATE-DROP
spring.jpa.hibernate.ddl-auto = update

logging.level.some.package.path=DEBUG
logging.level.some.other.package.path=ERROR
logging.file.name=logfile.log
```

logfile.log

```
logfile.log ×
logfile.log
2022-12-02 15:17:54.684  INFO 7620 --- [restartedMain] c.l.a5.KnowYourNeighborhoodApplication   : Starting KnowYourNeighborhoodApplicatior
2022-12-02 15:17:54.686  INFO 7620 --- [restartedMain] c.l.a5.KnowYourNeighborhoodApplication   : No active profile set, falling back to 1
2022-12-02 15:17:54.751  INFO 7620 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set '
2022-12-02 15:17:54.752  INFO 7620 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consi
2022-12-02 15:17:55.368  INFO 7620 --- [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositori
2022-12-02 15:17:55.435  INFO 7620 --- [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning
2022-12-02 15:17:56.144  INFO 7620 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (h
2022-12-02 15:17:56.155  INFO 7620 --- [restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-12-02 15:17:56.156  INFO 7620 --- [restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/
2022-12-02 15:17:56.408  INFO 7620 --- [restartedMain] org.apache.jasper.servlet.TldScanner     : At least one JAR was scanned for TLDs ye
2022-12-02 15:17:56.421  INFO 7620 --- [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicat
2022-12-02 15:17:56.421  INFO 7620 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializati
2022-12-02 15:17:56.631  INFO 7620 --- [restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInf
2022-12-02 15:17:56.677  INFO 7620 --- [restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.
2022-12-02 15:17:56.839  INFO 7620 --- [restartedMain] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotatic
2022-12-02 15:17:56.934  INFO 7620 --- [restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2022-12-02 15:17:57.106  INFO 7620 --- [restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2022-12-02 15:17:57.122  INFO 7620 --- [restartedMain] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.
2022-12-02 15:17:57.736  INFO 7620 --- [restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementat
2022-12-02 15:17:57.744  INFO 7620 --- [restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for
2022-12-02 15:17:58.286  INFO 7620 --- [restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.spring
2022-12-02 15:17:58.336  WARN 7620 --- [restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by de
2022-12-02 15:17:58.653  INFO 7620 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 357
2022-12-02 15:17:58.690  INFO 7620 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) w
2022-12-02 15:17:58.700  INFO 7620 --- [restartedMain] c.l.a5.KnowYourNeighborhoodApplication   : Started KnowYourNeighborhoodApplication
```

Error Handling

1. Create Global Exception Handler that can handle an exception. Add method to handle run away exception. [Run away exception is a RuntimeException that is not handled by any exception handler].

   Global Exception Handler & Store Not Found Exception Handler

```
package com.lithan.a5.controller;

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import com.lithan.a5.exception.StoreNotFoundException;

@ControllerAdvice
public class GlobalExceptionHandler {

  @ExceptionHandler(StoreNotFoundException.class)
  public String handleNoStoreException(StoreNotFoundException e, Model model) {
    model.addAttribute("message", e.getMessage());

    return "search";
  }

  @ExceptionHandler(Exception.class)
  public String handleGeneralException(Exception e, Model model) {
    model.addAttribute("message", e.getMessage());

    return "global-error";
  }
}
```

2. Create an exception class StoreNotFoundException.

```
package com.lithan.a5.exception;

public class StoreNotFoundException extends Exception {
  public StoreNotFoundException(String errorMessage) {
    super(errorMessage);
  }
}
```

3. Create an end point to search store by email.

StoreRepository.java

```
package com.lithan.a5.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.lithan.a5.entity.Store;

@Repository
public interface StoreRepository extends JpaRepository<Store, Integer> {
  @Query(value = "SELECT * FROM tb_store WHERE name LIKE '%' :keyword '%'",
nativeQuery = true)
  public List<Store> search(@Param("keyword") String keyword);
}
```

StoreServiceImpl.java

```
@Override
  public List<Store> search(String keyword) throws StoreNotFoundException {
    List<Store> listStore = storeRepo.search(keyword);

    return listStore;
  }
```

Source Code

Assignment 5.rar

StoreController.java

```
@GetMapping("/stores")
  public ModelAndView searchStore(@RequestParam("name") String keyword) throws
StoreNotFoundException {
    LOGGER.info("Inside searchStore of StoreController");

    ModelAndView mv = new ModelAndView();

    List<Store> stores = storeService.search(keyword);

    if (stores.isEmpty()) {
      throw new StoreNotFoundException("Store not found");
    }

    mv.addObject("stores", stores);
    mv.setViewName("search");
    return mv;
  }
```

Testing the error