Spring MVC 5 + Spring Security 5 + Hibernate 5 example

Posted on January 11, 2018

This post shows you -

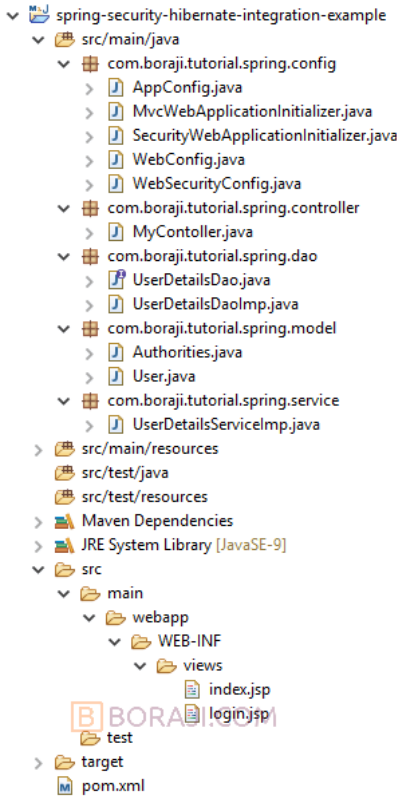
- How to create a custom login form (https://www.boraji.com/spring-security-4-custom-login-from-example) in Spring MVC application with Spring Security.
- How to integrate the Hibernate with Spring security framework to load the user's authentication.
- How to use the `UserDetailsService` interface (https://www.boraji.com/spring-security-5-custom-userdetailsservice-example) to load the user's authentication information from a database.

Tools and technologies used for this application are -

- Spring Security 5.0.0.RELEASE
- Spring MVC 5.0.2.RELEASE
- Spring ORM 5.0.2.RELEASE
- Hibernate 5.2.12.Final
- C3p0 0.9.5.2
- Servlet API 3.1.0
- Common Pool 2.1.1
- Java SE 9
- Maven 3.5.2
- Oxygen.1a Release (4.7.1a)
- Jetty Maven plugin 9.4.8
- MySQL Server 5.7

Project structure

Final project structure of our application will look like as follows.



Read - How to create a web project using maven build tool in eclipse IDE (https://www.boraji.com/how-to-create-a-web-project-using-maven-in-eclipse).

Jar dependencies

Add the following jars dependencies in your `pom.xml` file.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.0.0.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>5.0.0.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.0.2.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.0.2.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.12.Final</version>
  </dependency>

  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-c3p0</artifactId>
    <version>5.2.12.Final</version>
  </dependency>

  <dependency>
    <groupId>com.mchange</groupId>
    <artifactId>c3p0</artifactId>
    <version>0.9.5.2</version>
  </dependency>

  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>6.0.6</version>
  </dependency>

  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
  </dependency>

  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.1</version>
    <scope>provided</scope>
  </dependency>

  <dependency>
    <groupId>javax.servlet.jsp.jstl</groupId>
    <artifactId>javax.servlet.jsp.jstl-api</artifactId>
    <version>1.2.1</version>
  </dependency>

  <dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
  </dependency>

  <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
  </dependency>
</dependencies>
```

Database table creation

Use the following DDL statements to create tables for user's login information in MySQL database.

```
create table users(
        username varchar(50) not null primary key,
        password varchar(100) not null,
        enabled boolean not null
);
create table authorities (
        username varchar(50) not null,
        authority varchar(50) not null,
        constraint fk_authorities_users foreign key(username) references users(username)
);
create unique index ix_auth_username on authorities (username,authority);
```

Insert login information into database tables.

```
insert into users(username,password,enabled)
        values('admin','$2a$10$hbxecwitQQ.dDT4JOFzQAulNySFwEpaFLw38jda6Td.Y/cOiRzDFu',true);
insert into authorities(username,authority)
        values('admin','ROLE_ADMIN');
```

Before inserting data into tables, you can use the following code to encrypt the password.

```
String encoded=new BCryptPasswordEncoder().encode("admin@123");
```

Entity classes

Create two `@Entity` classes, named as `User` and `Authorities`, to map with database tables as follows.

User.java

```
package com.boraji.tutorial.spring.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "USERS")
public class User {
  @Id
  @Column(name = "USERNAME")
  private String username;

  @Column(name = "PASSWORD", nullable = false)
  private String password;

  @Column(name = "ENABLED", nullable = false)
  private boolean enabled;

  @OneToMany(cascade = CascadeType.ALL, mappedBy = "user")
  private Set<Authorities> authorities = new HashSet<>();

  //Getter and Setter methods
}
```

Authorities.java

```
package com.boraji.tutorial.spring.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "AUTHORITIES")
public class Authorities {
  @Id
  @Column(name = "AUTHORITY")
  private String authority;

  @ManyToOne
  @JoinColumn(name = "USERNAME")
  private User user;

  //Getter and Setter methods
}
```


Hibernate configuration

First, create a properties file under `src/main/resources` folder and define the database connection, hibernate and C3P0 properties as follows.

db.properties

```
# MySQL connection properties
mysql.driver=com.mysql.cj.jdbc.Driver
mysql.jdbcUrl=jdbc:mysql://localhost:3306/BORAJI?useSSL=false
mysql.username=root
mysql.password=admin

# Hibernate properties
hibernate.show_sql=true
hibernate.hbm2ddl.auto=validate

#C3P0 properties
hibernate.c3p0.min_size=5
hibernate.c3p0.max_size=20
hibernate.c3p0.acquire_increment=1
hibernate.c3p0.timeout=1800
hibernate.c3p0.max_statements=150
```

Next, create a `@Configuration` class and define the `@Bean` method for `LocalSessionFactoryBean` as follows.

In Spring based application, `LocalSessionFactoryBean` class is used to create a Hibernate `SessionFactory` .

AppConfig.java

```
package com.boraji.tutorial.spring.config;

import java.util.Properties;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.ComponentScans;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import com.boraji.tutorial.spring.model.Authorities;
import com.boraji.tutorial.spring.model.User;

import static org.hibernate.cfg.Environment.*;

@Configuration
@PropertySource("classpath:db.properties")
@EnableTransactionManagement
@ComponentScans(value = { @ComponentScan("com.boraji.tutorial.spring.dao"),
    @ComponentScan("com.boraji.tutorial.spring.service") })
public class AppConfig {

  @Autowired
  private Environment env;

  @Bean
  public LocalSessionFactoryBean getSessionFactory() {
    LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();

    Properties props = new Properties();

    // Setting JDBC properties
    props.put(DRIVER, env.getProperty("mysql.driver"));
    props.put(URL, env.getProperty("mysql.jdbcUrl"));
    props.put(USER, env.getProperty("mysql.username"));
    props.put(PASS, env.getProperty("mysql.password"));

    // Setting Hibernate properties
    props.put(SHOW_SQL, env.getProperty("hibernate.show_sql"));
    props.put(HBM2DDL_AUTO, env.getProperty("hibernate.hbm2ddl.auto"));

    // Setting C3P0 properties
    props.put(C3P0_MIN_SIZE, env.getProperty("hibernate.c3p0.min_size"));
    props.put(C3P0_MAX_SIZE, env.getProperty("hibernate.c3p0.max_size"));
    props.put(C3P0_ACQUIRE_INCREMENT, env.getProperty("hibernate.c3p0.acquire_increment"));
    props.put(C3P0_TIMEOUT, env.getProperty("hibernate.c3p0.timeout"));
    props.put(C3P0_MAX_STATEMENTS, env.getProperty("hibernate.c3p0.max_statements"));

    factoryBean.setHibernateProperties(props);
    factoryBean.setAnnotatedClasses(User.class, Authorities.class);

    return factoryBean;
  }

  @Bean
  public HibernateTransactionManager getTransactionManager() {
    HibernateTransactionManager transactionManager = new HibernateTransactionManager();
    transactionManager.setSessionFactory(getSessionFactory().getObject());
    return transactionManager;
  }
}
```

Repository classes

Create `@Repository` classes under `com.boraji.tutorial.spring.dao` package as follows.

UserDetailsDao.java

```
package com.boraji.tutorial.spring.dao;

import com.boraji.tutorial.spring.model.User;

public interface UserDetailsDao {
  User findUserByUsername(String username);
}
```

UserDetailsDaoImp.java

```
package com.boraji.tutorial.spring.dao;

import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.boraji.tutorial.spring.model.User;

@Repository
public class UserDetailsDaoImp implements UserDetailsDao {

  @Autowired
  private SessionFactory sessionFactory;

  @Override
  public User findUserByUsername(String username) {
    return sessionFactory.getCurrentSession().get(User.class, username);
  }
}
```

UserDetailsService or Service class

To create a custom user service, you need to implement the `UserDetailsService` interface and override the `loadUserByUsername()` method.

Create `@Service` class under `com.boraji.tutorial.spring.service` package as follows.

UserDetailsServiceImp.java

```
package com.boraji.tutorial.spring.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.User.UserBuilder;
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 org.springframework.transaction.annotation.Transactional;

import com.boraji.tutorial.spring.dao.UserDetailsDao;
import com.boraji.tutorial.spring.model.User;

@Service("userDetailsService")
public class UserDetailsServiceImp implements UserDetailsService {

  @Autowired
  private UserDetailsDao userDetailsDao;

  @Transactional(readOnly = true)
  @Override
  public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

    User user = userDetailsDao.findUserByUsername(username);
    UserBuilder builder = null;
    if (user != null) {

      builder = org.springframework.security.core.userdetails.User.withUsername(username);
      builder.disabled(!user.isEnabled());
      builder.password(user.getPassword());
      String[] authorities = user.getAuthorities()
          .stream().map(a -> a.getAuthority()).toArray(String[]::new);

      builder.authorities(authorities);
    } else {
      throw new UsernameNotFoundException("User not found.");
    }
    return builder.build();
  }
}
```

Spring Security configuration

To configure Spring Security in Spring MVC application you need to create a `@Configuration` class by extending the `WebSecurityConfigurerAdapter` class and annotate it with `@EnableWebSecurity` as follows.

WebSecurityConfig.java

```
package com.boraji.tutorial.spring.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
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;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  private UserDetailsService userDetailsService;

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

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
  }

  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests().anyRequest().hasAnyRole("ADMIN", "USER")
    .and()
    .authorizeRequests().antMatchers("/login**").permitAll()
    .and()
    .formLogin().loginPage("/login").loginProcessingUrl("/loginAction").permitAll()
    .and()
    .logout().logoutSuccessUrl("/login").permitAll()
    .and()
    .csrf().disable();

  }
}
```

Next, create `SecurityWebApplicationInitializer` class by extending the `AbstractSecurityWebApplicationInitializer` to register the `springSecurityFilterChain` filter.

SecurityWebApplicationInitializer.java

```
package com.boraji.tutorial.spring.config;

import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SecurityWebApplicationInitializer
  extends AbstractSecurityWebApplicationInitializer {

}
```

Spring MVC configuration

In this example, we are using the JSP views. So create a `@Configuration` class and override the `configureViewResolvers()` method to register the JSP view resolver.

Also, you can override the `addViewControllers()` method to map and render the default login page generated by Spring Security.

WebConfig.java

```
package com.boraji.tutorial.spring.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = { "com.boraji.tutorial.spring.controller" })
public class WebConfig implements WebMvcConfigurer {

  @Override
  public void configureViewResolvers(ViewResolverRegistry registry) {
    registry.jsp().prefix("/WEB-INF/views/").suffix(".jsp");
  }

  @Override
  public void addViewControllers(ViewControllerRegistry registry) {
    registry.addViewController("/login").setViewName("login");
  }
}
```

Servlet container Initialization and configuration

In Spring MVC, The `DispatcherServlet` needs to be declared and mapped for processing all requests either using java or `web.xml` configuration.

In a Servlet 3.0+ environment, you can use `AbstractAnnotationConfigDispatcherServletInitializer` class to register and initialize the `DispatcherServlet` programmatically as follows.

MvcWebApplicationInitializer.java

```
package com.boraji.tutorial.spring.config;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class MvcWebApplicationInitializer
      extends AbstractAnnotationConfigDispatcherServletInitializer {

  // Load database and spring security configuration
  @Override
  protected Class<?>[] getRootConfigClasses() {
    return new Class[] { AppConfig.class, WebSecurityConfig.class };
  }

  // Load spring web configuration
  @Override
  protected Class<?>[] getServletConfigClasses() {
    return new Class[] { WebConfig.class };
  }

  @Override
  protected String[] getServletMappings() {
    return new String[] { "/" };
  }

}
```


Controller class

Create a simple `@Controller` class under `com.boraji.tutorial.spring.controller` package as follows.

MyContoller.java

```
package com.boraji.tutorial.spring.controller;

import java.security.Principal;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MyContoller {

  @GetMapping("/")
  public String index(Model model, Principal principal) {
    model.addAttribute("message", "You are logged in as " + principal.getName());
    return "index";
  }
}
```

JSP views

Create `login.jsp` and `index.jsp` files under `src\main\webapp\WEB-INF\views` folder and write the following code in it.

login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://www.springframework.org/tags (http://www.springframework.org/tags)" prefix="spring"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>BORAJI.COM (http://BORAJI.COM)</title>
</head>
<body>

        <h1>Spring MVC 5 + Spring Security 5 + Hibernate 5 example</h1>
        <h4>Login Form</h4>

        <form action='<spring:url value="/loginAction"/>' method="post">
    <table>
      <tr>
        <td>Username</td>
        <td><input type="text" name="username"></td>
      </tr>
      <tr>
        <td>Password</td>
        <td><input type="password" name="password"></td>
      </tr>
      <tr>
        <td><button type="submit">Login</button></td>
      </tr>
    </table>
  </form>
  <br/>
</body>
</html>
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>BORAJI.COM (http://BORAJI.COM)</title>
</head>
<body>
        <h1>Spring MVC 5 + Spring Security 5 + Hibernate 5 example</h1>
        <h2>${message}</h2>

        <form action="/logout" method="post">
                <input value="Logout" type="submit">
        </form>
</body>
</html>
```

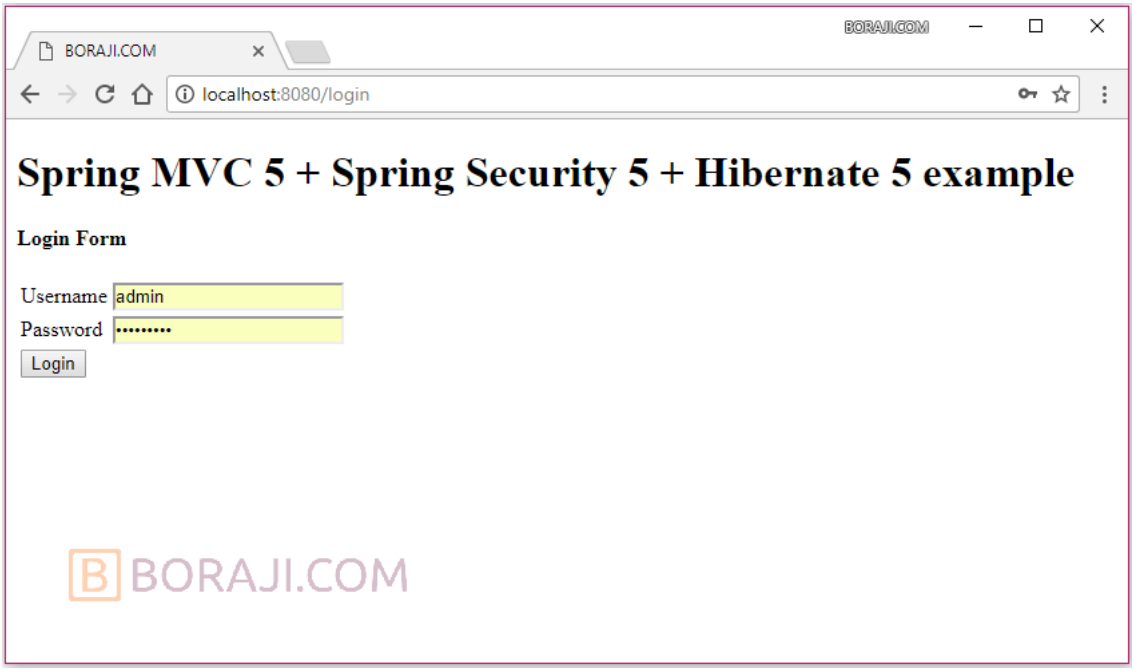
Run application

Use the following maven command to run your application.

`mvn jetty:run` (This command deploy the webapp from its sources, instead of build war).

Enter the **http://localhost:8080** URL in browser's address bar to test our application.

On entering the URL, you will see the login page asking for username and password as follows.



On successful login, you will see the index page as follows.

Download Sources:

spring-security-hibernate-integration-example.zip (https://www.boraji.com/sites/default/files/2018-01/spring-security-hibernate-integration-example.zip) **17.53 KB**

Related Posts

- Spring MVC 5 + Spring Security 5 + Hibernate 5 example (/spring-mvc-5-spring-security-5-hibernate-5-example)
- Spring Security 4 - Hello World example (/spring-security-4-hello-world-example)
- Spring Security 4 - HTTP basic authentication example (/spring-security-4-http-basic-authentication-example)
- Spring Security 4 - Custom login from example (/spring-security-4-custom-login-from-example)
- Spring Security 5 - JDBC based authentication example (/spring-security-5-jdbc-based-authentication-example)
- Spring Security 5 - Custom UserDetailsService example (/spring-security-5-custom-userdetailsservice-example)
- Spring Security 5 - Remember-Me authentication example (/spring-security-5-remember-me-authentication-example)
- Spring MVC 4 + Hibernate 5 + RESTful CRUD operations example (/spring-mvc-4-hibernate-5-restful-crud-operations-example)

Spring Security 5 - Remember Me authentication example with Hibernate 5 (/spring-security-5-remember-me-authentication-example-with-hibernate-5)

Spring MVC 4 + Hibernate 5 integration example (/spring-4-mvc-hibernate-5-integration-example)

Artsiom Bel • 2 years ago

[2018-05-28T14:58:13.957+0200] [glassfish 5.0] [WARN] [] [org.hibernate.engine.jdbc.connections.internal.ConnectionProviderInitiator] [tid: _ThreadID=44 _ThreadName=admin-listener(1)] [timeMillis: 1527512293957] [levelValue: 900] [[
HHH000181: No appropriate connection provider encountered, assuming application will be supplying connections]]

[2018-05-28T14:58:13.959+0200] [glassfish 5.0] [WARN] [] [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator] [tid: _ThreadID=44 _ThreadName=admin-listener(1)] [timeMillis: 1527512293959] [levelValue: 900] [[
HHH000342: Could not obtain connection to query metadata : The application must supply JDBC connections]]

[2018-05-28T14:58:13.965+0200] [glassfish 5.0] [WARNING] [] [org.springframework.web.context.support.AnnotationConfigWebApplicationContext] [tid: _ThreadID=44 _ThreadName=admin-listener(1)] [timeMillis: 1527512293965] [levelValue: 900] [[
Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'webSecurityConfig': Unsatisfied dependency expressed through field 'userDetailsService'; nested exception is org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'userDetailsService': Unsatisfied dependency expressed through field 'userDetailsDao'; nested exception is org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'userDetailsDaoImpl': Unsatisfied dependency expressed through field 'sessionFactory'; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'getSessionFactory' defined in local.vssp.config.AppConfig: Invocation of init method failed; nested exception is org.hibernate.service.spi.ServiceException: Unable to create requested service [org.hibernate.engine.jdbc.env.spi.JdbcEnvironment]]]

1 • Reply • Share ›

tigercomputingsbj • a year ago

java 1.8 error on code (javaserviceimpl.java)
String[] authorities = user.getAuthorities()
.stream().map(a -> a.getAuthority()).toArray(String[]::new);

code is not working.

1 1 • Reply • Share ›

Valentine Golubev • 2 years ago • edited

same error as in all your tutorials: "org.hibernate.HibernateException: Access to DialectResolutionInfo cannot be null when 'hibernate.dialect' not set"
@Table(name = "AUTHORITIES") will not work with 'create table authorities' sql
How do you even run all of your projects?
Do you test it before copy-pasting from other resources?
P.S. to get rid of taglib(JSTL) warning you need to remove one of the depency

1 3 • Reply • Share ›

Coul Youl • a year ago

logout don't work

http://localhost:8080/logout

État HTTP 404 –

• Reply • Share ›

Franky • 2 years ago