The method requestMatchers(RequestMatcher...) in the type AbstractRequestMatcherRegistry<AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizedUrl> is not applicable for the arguments (String[])Java(67108979) package org.studyeasy.SpringBlog.security; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.security.config.Customizer; import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity; import org.springframework.security.config.annotation.web.builders.HttpSecurity; import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity; import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.security.web.SecurityFilterChain; @Configuration @EnableWebSecurity @EnableMethodSecurity(prePostEnabled = true, securedEnabled = true) public class WebSecurityConfig { private static final String[] WHITELIST = { "/", "/login", "/register", "/db-console/\*\*", // Allow all the css and other resources "/css/\*\*", "/fonts/\*\*", "/images/\*\*", "/js/\*\*" }; @Bean public static PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); } @Bean public SecurityFilterChain filterChain(HttpSecurity http) throws Exception { http .authorizeHttpRequests(auth -> auth .requestMatchers(WHITELIST) .permitAll() .anyRequest() .authenticated()) .formLogin(formLogin -> formLogin .loginPage("/login") .loginProcessingUrl("/login") .usernameParameter("email") .passwordParameter("password") .defaultSuccessUrl("/", true) .failureUrl("/login?error") .permitAll()) .logout(logout -> logout .logoutUrl("/logout") .logoutSuccessUrl("/logout?success") .permitAll()) .httpBasic(Customizer.withDefaults()); // Enable HTTP Basic authentication // TODO: remove these files after upgrading the DB from h2 infile DB http.csrf(csrf -> csrf.disable()); // Disable frame options http.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.disable())); return http.build(); } }

ChatGPT

The error you are encountering is because the requestMatchers method in the version of Spring Security you are using does not accept a String[] as an argument directly. Instead, it expects multiple String arguments or an array of RequestMatcher objects.

To fix this, you can use a method reference or lambda expression to pass the WHITELIST array correctly. Here's how you can do it:

java

Copy code

package org.studyeasy.SpringBlog.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

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

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true)

public class WebSecurityConfig {

private static final String[] WHITELIST = {

"/",

"/login",

"/register",

"/db-console/\*\*",

// Allow all the css and other resources

"/css/\*\*",

"/fonts/\*\*",

"/images/\*\*",

"/js/\*\*"

};

@Bean

public static PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> {

for (String pattern : WHITELIST) {

auth.requestMatchers(pattern).permitAll();

}

auth.anyRequest().authenticated();

})

.formLogin(formLogin -> formLogin

.loginPage("/login")

.loginProcessingUrl("/login")

.usernameParameter("email")

.passwordParameter("password")

.defaultSuccessUrl("/", true)

.failureUrl("/login?error")

.permitAll())

.logout(logout -> logout

.logoutUrl("/logout")

.logoutSuccessUrl("/logout?success")

.permitAll())

.httpBasic(Customizer.withDefaults()); // Enable HTTP Basic authentication

// TODO: remove these files after upgrading the DB from h2 infile DB

http.csrf(csrf -> csrf.disable());

// Disable frame options

http.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.disable()));

return http.build();

}

}

In this modification, the WHITELIST array is iterated over, and each pattern is passed to requestMatchers individually. This ensures that each whitelist pattern is properly registered with Spring Security.

4o

The method requestMatchers(RequestMatcher...) in the type AbstractRequestMatcherRegistry<AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizedUrl> is not applicable for the arguments (String)Java(67108979)

ChatGPT

It seems that the requestMatchers method in your version of Spring Security is not accepting individual String arguments. This issue is likely due to the method's signature expecting RequestMatcher objects or a different way to specify multiple matchers.

Let's correct this by using the requestMatchers method correctly. We will use the mvcMatchers method if available, which can accept string patterns directly. If not, we will convert the string patterns into RequestMatcher objects.

Here's how you can modify your code to handle the whitelist correctly:

java

Copy code

package org.studyeasy.SpringBlog.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

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

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true)

public class WebSecurityConfig {

private static final String[] WHITELIST = {

"/",

"/login",

"/register",

"/db-console/\*\*",

// Allow all the css and other resources

"/css/\*\*",

"/fonts/\*\*",

"/images/\*\*",

"/js/\*\*"

};

@Bean

public static PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.requestMatchers(AntPathRequestMatcher::new, WHITELIST)

.permitAll()

.anyRequest()

.authenticated())

.formLogin(formLogin -> formLogin

.loginPage("/login")

.loginProcessingUrl("/login")

.usernameParameter("email")

.passwordParameter("password")

.defaultSuccessUrl("/", true)

.failureUrl("/login?error")

.permitAll())

.logout(logout -> logout

.logoutUrl("/logout")

.logoutSuccessUrl("/logout?success")

.permitAll())

.httpBasic(Customizer.withDefaults()); // Enable HTTP Basic authentication

// TODO: remove these files after upgrading the DB from h2 infile DB

http.csrf(csrf -> csrf.disable());

// Disable frame options

http.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.disable()));

return http.build();

}

}

In this code, requestMatchers is used with a method reference to create AntPathRequestMatcher objects from the whitelist array. This should resolve the issue with the argument type mismatch.

4o

The method requestMatchers(RequestMatcher...) in the type AbstractRequestMatcherRegistry<AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizedUrl> is not applicable for the arguments (AntPathRequestMatcher::new, String[])Java(67108979)

ChatGPT

It appears that the correct approach would be to use mvcMatchers instead of requestMatchers, which directly accepts String patterns. Here is how you can modify your filterChain method:

java

Copy code

package org.studyeasy.SpringBlog.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

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

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true)

public class WebSecurityConfig {

private static final String[] WHITELIST = {

"/",

"/login",

"/register",

"/db-console/\*\*",

// Allow all the css and other resources

"/css/\*\*",

"/fonts/\*\*",

"/images/\*\*",

"/js/\*\*"

};

@Bean

public static PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.mvcMatchers(WHITELIST)

.permitAll()

.anyRequest()

.authenticated())

.formLogin(formLogin -> formLogin

.loginPage("/login")

.loginProcessingUrl("/login")

.usernameParameter("email")

.passwordParameter("password")

.defaultSuccessUrl("/", true)

.failureUrl("/login?error")

.permitAll())

.logout(logout -> logout

.logoutUrl("/logout")

.logoutSuccessUrl("/logout?success")

.permitAll())

.httpBasic(Customizer.withDefaults()); // Enable HTTP Basic authentication

// TODO: remove these files after upgrading the DB from h2 infile DB

http.csrf(csrf -> csrf.disable());

// Disable frame options

http.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.disable()));

return http.build();

}

}

In this modification, the mvcMatchers method is used to handle the WHITELIST array. This method is designed to accept an array of String patterns, which should resolve the issue you're encountering. If mvcMatchers is not available in your Spring Security version, the fallback is to create individual matchers for each path. Here’s how you can do it:

java

Copy code

package org.studyeasy.SpringBlog.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

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

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

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration

@EnableWebSecurity

@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true)

public class WebSecurityConfig {

private static final String[] WHITELIST = {

"/",

"/login",

"/register",

"/db-console/\*\*",

// Allow all the css and other resources

"/css/\*\*",

"/fonts/\*\*",

"/images/\*\*",

"/js/\*\*"

};

@Bean

public static PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> {

for (String pattern : WHITELIST) {

auth.requestMatchers(new AntPathRequestMatcher(pattern)).permitAll();

}

auth.anyRequest().authenticated();

})

.formLogin(formLogin -> formLogin

.loginPage("/login")

.loginProcessingUrl("/login")

.usernameParameter("email")

.passwordParameter("password")

.defaultSuccessUrl("/", true)

.failureUrl("/login?error")

.permitAll())

.logout(logout -> logout

.logoutUrl("/logout")

.logoutSuccessUrl("/logout?success")

.permitAll())

.httpBasic(Customizer.withDefaults()); // Enable HTTP Basic authentication

// TODO: remove these files after upgrading the DB from h2 infile DB

http.csrf(csrf -> csrf.disable());

// Disable frame options

http.headers(headers -> headers.frameOptions(frameOptions -> frameOptions.disable()));

return http.build();

}

}

In this approach, AntPathRequestMatcher objects are created for each pattern in the WHITELIST array, ensuring compatibility with the requestMatchers method.

4o

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'entityManagerFactory' defined in class path resource [org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaConfiguration.class]: Invocation of init method failed; nested exception is javax.persistence.PersistenceException: [PersistenceUnit: default] Unable to build Hibernate SessionFactory; nested exception is org.hibernate.exception.JDBCConnectionException: Unable to open JDBC Connection for DDL execution at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1804) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:620) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:542) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:335) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:234) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:333) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:208) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.context.support.AbstractApplicationContext.getBean(AbstractApplicationContext.java:1154) ~[spring-context-5.3.23.jar:5.3.23] at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:908) ~[spring-context-5.3.23.jar:5.3.23] at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:583) ~[spring-context-5.3.23.jar:5.3.23] at org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext.refresh(ServletWebServerApplicationContext.java:147) ~[spring-boot-2.7.5.jar:2.7.5] at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:734) ~[spring-boot-2.7.5.jar:2.7.5] at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:408) ~[spring-boot-2.7.5.jar:2.7.5] at org.springframework.boot.SpringApplication.run(SpringApplication.java:308) ~[spring-boot-2.7.5.jar:2.7.5] at org.springframework.boot.SpringApplication.run(SpringApplication.java:1306) ~[spring-boot-2.7.5.jar:2.7.5] at org.springframework.boot.SpringApplication.run(SpringApplication.java:1295) ~[spring-boot-2.7.5.jar:2.7.5] at org.studyeasy.SpringBlog.SpringStarterApplication.main(SpringStarterApplication.java:10) ~[classes/:na] at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na] at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Unknown Source) ~[na:na] at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source) ~[na:na] at java.base/java.lang.reflect.Method.invoke(Unknown Source) ~[na:na] at org.springframework.boot.devtools.restart.RestartLauncher.run(RestartLauncher.java:49) ~[spring-boot-devtools-2.7.5.jar:2.7.5] Caused by: javax.persistence.PersistenceException: [PersistenceUnit: default] Unable to build Hibernate SessionFactory; nested exception is org.hibernate.exception.JDBCConnectionException: Unable to open JDBC Connection for DDL execution at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.buildNativeEntityManagerFactory(AbstractEntityManagerFactoryBean.java:421) ~[spring-orm-5.3.23.jar:5.3.23] at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.afterPropertiesSet(AbstractEntityManagerFactoryBean.java:396) ~[spring-orm-5.3.23.jar:5.3.23] at org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean.afterPropertiesSet(LocalContainerEntityManagerFactoryBean.java:341) ~[spring-orm-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1863) ~[spring-beans-5.3.23.jar:5.3.23] at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1800) ~[spring-beans-5.3.23.jar:5.3.23] ... 21 common frames omitted Caused by: org.hibernate.exception.JDBCConnectionException: Unable to open JDBC Connection for DDL execution at org.hibernate.exception.internal.SQLExceptionTypeDelegate.convert(SQLExceptionTypeDelegate.java:48) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.exception.internal.StandardSQLExceptionConverter.convert(StandardSQLExceptionConverter.java:37) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.engine.jdbc.spi.SqlExceptionHelper.convert(SqlExceptionHelper.java:113) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.engine.jdbc.spi.SqlExceptionHelper.convert(SqlExceptionHelper.java:99) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl.getIsolatedConnection(DdlTransactionIsolatorNonJtaImpl.java:71) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.exec.GenerationTargetToDatabase.jdbcStatement(GenerationTargetToDatabase.java:77) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.exec.GenerationTargetToDatabase.accept(GenerationTargetToDatabase.java:53) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.applySqlString(SchemaDropperImpl.java:387) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.applySqlStrings(SchemaDropperImpl.java:371) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.dropFromMetadata(SchemaDropperImpl.java:246) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.performDrop(SchemaDropperImpl.java:156) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.doDrop(SchemaDropperImpl.java:128) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.internal.SchemaDropperImpl.doDrop(SchemaDropperImpl.java:114) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.spi.SchemaManagementToolCoordinator.performDatabaseAction(SchemaManagementToolCoordinator.java:157) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.tool.schema.spi.SchemaManagementToolCoordinator.process(SchemaManagementToolCoordinator.java:85) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.internal.SessionFactoryImpl.<init>(SessionFactoryImpl.java:335) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.boot.internal.SessionFactoryBuilderImpl.build(SessionFactoryBuilderImpl.java:471) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.jpa.boot.internal.EntityManagerFactoryBuilderImpl.build(EntityManagerFactoryBuilderImpl.java:1498) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.springframework.orm.jpa.vendor.SpringHibernateJpaPersistenceProvider.createContainerEntityManagerFactory(SpringHibernateJpaPersistenceProvider.java:58) ~[spring-orm-5.3.23.jar:5.3.23] at org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean.createNativeEntityManagerFactory(LocalContainerEntityManagerFactoryBean.java:365) ~[spring-orm-5.3.23.jar:5.3.23] at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.buildNativeEntityManagerFactory(AbstractEntityManagerFactoryBean.java:409) ~[spring-orm-5.3.23.jar:5.3.23] ... 25 common frames omitted Caused by: org.h2.jdbc.JdbcSQLNonTransientConnectionException: Unsupported database file version or invalid file header in file "C:/Users/dinit/db/blogdb.mv.db" [90048-214] at org.h2.message.DbException.getJdbcSQLException(DbException.java:678) ~[h2-2.1.214.jar:2.1.214] at org.h2.message.DbException.getJdbcSQLException(DbException.java:477) ~[h2-2.1.214.jar:2.1.214] at org.h2.message.DbException.get(DbException.java:212) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.db.Store.convertMVStoreException(Store.java:161) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.db.Store.<init>(Store.java:145) ~[h2-2.1.214.jar:2.1.214] at org.h2.engine.Database.<init>(Database.java:324) ~[h2-2.1.214.jar:2.1.214] at org.h2.engine.Engine.openSession(Engine.java:92) ~[h2-2.1.214.jar:2.1.214] at org.h2.engine.Engine.openSession(Engine.java:222) ~[h2-2.1.214.jar:2.1.214] at org.h2.engine.Engine.createSession(Engine.java:201) ~[h2-2.1.214.jar:2.1.214] at org.h2.engine.SessionRemote.connectEmbeddedOrServer(SessionRemote.java:338) ~[h2-2.1.214.jar:2.1.214] at org.h2.jdbc.JdbcConnection.<init>(JdbcConnection.java:122) ~[h2-2.1.214.jar:2.1.214] at org.h2.Driver.connect(Driver.java:59) ~[h2-2.1.214.jar:2.1.214] at com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.java:138) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:364) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:206) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:476) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:561) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:115) ~[HikariCP-4.0.3.jar:na] at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:112) ~[HikariCP-4.0.3.jar:na] at org.hibernate.engine.jdbc.connections.internal.DatasourceConnectionProviderImpl.getConnection(DatasourceConnectionProviderImpl.java:122) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess.obtainConnection(JdbcEnvironmentInitiator.java:181) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] at org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl.getIsolatedConnection(DdlTransactionIsolatorNonJtaImpl.java:44) ~[hibernate-core-5.6.12.Final.jar:5.6.12.Final] ... 41 common frames omitted Caused by: org.h2.mvstore.MVStoreException: The write format 3 is larger than the supported format 2 [2.1.214/5] at org.h2.mvstore.DataUtils.newMVStoreException(DataUtils.java:1004) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.MVStore.getUnsupportedWriteFormatException(MVStore.java:1066) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.MVStore.readStoreHeader(MVStore.java:883) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.MVStore.<init>(MVStore.java:463) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.MVStore$Builder.open(MVStore.java:4082) ~[h2-2.1.214.jar:2.1.214] at org.h2.mvstore.db.Store.<init>(Store.java:136) ~[h2-2.1.214.jar:2.1.214] ... 58 common frames omitted

ChatGPT

The error you're encountering is due to an incompatibility between the H2 database file version and the version of H2 you're using in your project. The H2 database file at C:/Users/dinit/db/blogdb.mv.db is created with a newer or different version of H2 than the one you are currently using.

Here are a few steps you can take to resolve this issue:

**Option 1: Update H2 Version**

Ensure that you are using the same version of H2 that was used to create the database file. You can do this by updating your pom.xml (for Maven) or build.gradle (for Gradle) to use the correct H2 version.

For Maven:

xml

Copy code

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version>

</dependency>

For Gradle:

groovy

Copy code

implementation 'com.h2database:h2:2.1.214'

**Option 2: Downgrade H2 Version**

If updating the H2 version is not feasible, you may need to downgrade your H2 version to match the format supported by the existing database file. However, this approach might not be ideal if you need features or bug fixes from the newer version.

**Option 3: Migrate Data**

If you need to upgrade to the latest H2 version but still want to keep your data, you can try migrating the data. This involves exporting the data from the old H2 database file and importing it into a new database file created with the current version of H2.

**Steps for Migration:**

1. **Create a backup** of your existing database file.
2. **Use an H2 console** with the older version of H2 to export your data to a SQL script or CSV files.
3. **Update your H2 dependency** to the new version.
4. **Use the H2 console** with the new version to create a new database file and import your data.

**Option 4: Start with a Fresh Database**

If possible, you can start with a fresh database. This will involve deleting the existing blogdb.mv.db file and allowing your application to create a new one. Note that this will result in the loss of existing data.

**Implementation Example:**

Here's an example of how to configure your pom.xml for Maven to use the correct version of H2:

xml

Copy code

<dependencies>

<!-- Other dependencies -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version> <!-- Ensure this matches the version used to create the database file -->

</dependency>

</dependencies>

**Configuration Changes:**

Ensure that your application.properties or application.yml is correctly configured to point to the H2 database file:

**application.properties:**

properties

Copy code

spring.datasource.url=jdbc:h2:file:C:/Users/dinit/db/blogdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

**application.yml:**

yaml

Copy code

spring:

datasource:

url: jdbc:h2:file:C:/Users/dinit/db/blogdb

driverClassName: org.h2.Driver

username: sa

password: password

jpa:

database-platform: org.hibernate.dialect.H2Dialect

By ensuring compatibility between the H2 version and the database file format, you should be able to resolve the JDBCConnectionException.

4o

<?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>2.7.5</version> <relativePath /> <!-- lookup parent from repository --> </parent> <groupId>org.studyeasy</groupId> <artifactId>SpringStarter</artifactId> <version>0.0.1-SNAPSHOT</version> <name>SpringStarter</name> <description>Spring Starter project</description> <properties> <java.version>17</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> <scope>runtime</scope> <optional>true</optional> </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> </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> <configuration> <excludes> <exclude> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> </exclude> </excludes> </configuration> </plugin> </plugins> </build> </project>

ChatGPT

To ensure that your project is using the correct version of the H2 database, you should explicitly specify the version in your pom.xml. Given the error you're seeing, it looks like you might need to align the H2 database version with the version used to create the database file.

First, let's add the specific version of the H2 database dependency in your pom.xml. Update the H2 dependency as follows:

xml

Copy code

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version> <!-- Specify the exact version here -->

<scope>runtime</scope>

</dependency>

Here is your complete pom.xml with the updated H2 dependency:

xml

Copy code

<?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>2.7.5</version>

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

</parent>

<groupId>org.studyeasy</groupId>

<artifactId>SpringStarter</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringStarter</name>

<description>Spring Starter project</description>

<properties>

<java.version>17</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>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version> <!-- Specify the exact version here -->

<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>

</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>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

Next, ensure that your application.properties is correctly configured to point to the H2 database file. Here's an example:

properties

Copy code

spring.datasource.url=jdbc:h2:file:C:/Users/dinit/db/blogdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

This configuration should help align the versions and allow Hibernate to connect to your H2 database. If you still encounter issues, consider creating a new database file compatible with the version of H2 you're using. This might involve exporting data from the old database and importing it into a new one as discussed previously.

4o

<parent> <version>2.7.18</version> <artifactId>spring-boot-starter-parent</artifactId> <version>2.7.5</version> <relativePath /> <!-- lookup parent from repository --> </parent> The version of the parent project to inherit.

ChatGPT

It looks like you have a conflicting version in your parent tag. The <parent> tag should only have one <version> element specifying the version of the parent project. Here is the corrected pom.xml snippet:

xml

Copy code

<parent>

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

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

<version>2.7.18</version> <!-- Make sure to use the correct version here -->

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

</parent>

You should choose either version 2.7.18 or 2.7.5, but not both. Ensure the correct and consistent version throughout your project. Here's the updated pom.xml with version 2.7.18 for the parent:

xml

Copy code

<?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>2.7.18</version> <!-- Make sure to use the correct version here -->

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

</parent>

<groupId>org.studyeasy</groupId>

<artifactId>SpringStarter</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringStarter</name>

<description>Spring Starter project</description>

<properties>

<java.version>17</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>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version> <!-- Specify the exact version here -->

<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>

</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>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

Make sure to align all versions correctly and consistently throughout your pom.xml file.

4o

OSS support for Spring Boot 2.7.x ended on 2023-11-24, get commercial support until 2025-08-24 via Tanz

ChatGPT

It seems like you are using Spring Boot 2.7.x, which no longer has open-source support. If you want to continue using an officially supported version, you should consider upgrading to a newer version of Spring Boot. The latest stable version is Spring Boot 3.x. Here's how you can update your pom.xml file to use Spring Boot 3.x:

First, update the parent section to use the latest version of Spring Boot:

xml

Copy code

<parent>

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

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

<version>3.1.2</version> <!-- Update to the latest 3.x version -->

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

</parent>

Make sure to check the Spring Boot [release notes](https://github.com/spring-projects/spring-boot/releases) for the latest stable version and any potential migration issues.

Here is the updated pom.xml file with the version updated to 3.1.2:

xml

Copy code

<?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>3.1.2</version> <!-- Update to the latest 3.x version -->

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

</parent>

<groupId>org.studyeasy</groupId>

<artifactId>SpringStarter</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringStarter</name>

<description>Spring Starter project</description>

<properties>

<java.version>17</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>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>2.1.214</version> <!-- Specify the exact version here -->

<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>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>thymeleaf-extras-springsecurity6</artifactId> <!-- Update to springsecurity6 for compatibility -->

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

Make sure to check for any breaking changes or necessary code updates when migrating from Spring Boot 2.7.x to 3.x. The official [Spring Boot migration guide](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#migration) can help you with this process.