20230519：

（1）

问题：

server: # port: 0 port: 8443 ssl: key-store: file:///path/to/mykeys.jks key-store-password: letmein key-password: letmein 在SpringBoot 中，添加以上代码后，报错：Connected to the target VM, address: '127.0.0.1:55816', transport: 'socket' 11:35:16.796 [restartedMain] ERROR org.springframework.boot.SpringApplication - Application run failed org.yaml.snakeyaml.scanner.ScannerException: mapping values are not allowed here in 'reader', line 35, column 7: ssl:

解决：

这个错误是由于在YAML配置文件中的语法错误导致的。根据您提供的错误消息，问题出现在第35行的 **ssl:** 这一行上。

在YAML中，缩进非常重要，它用于表示嵌套关系。您需要确保在 **ssl:** 这一行之前有正确的缩进，并且之前的行与其所属的块对齐。

在您的示例中，**ssl:** 前面的行缩进不正确。以下是修正后的示例代码：

yamlCopy code

server: port: 8443 ssl: key-store: file:///path/to/mykeys.jks key-store-password: letmein key-password: letmein

请注意，**port: 8443** 与 **ssl:** 的缩进是一样的，表示它们是 **server:** 的子项。**key-store**、**key-store-password** 和 **key-password** 也应该相对于 **ssl:** 进行缩进，表示它们是 **ssl:** 的子项。

修正缩进后，您应该不再遇到 "mapping values are not allowed here" 的错误，可以重新运行您的应用程序。

你真的太厉害了，Debug直接就能发现错误，吊打文心一言

ChatGPT

感谢您的夸奖！作为人工智能助手，我可以帮助检查常见错误，并提供解决方案。在这种情况下，由于我可以分析和理解代码结构，因此能够识别出缩进错误导致的问题。

对于YAML文件中的语法错误，缩进是一个常见的错误来源。在这种情况下，修正缩进是解决问题的关键。我很高兴能够帮到您，如果您有其他问题，欢迎随时提问！

20230519\_2

(1)

问题：

Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled. 2023-05-19 11:40:18.570 ERROR 24200 --- [ restartedMain] o.s.boot.SpringApplication : Application run failed org.springframework.context.ApplicationContextException: Failed to start bean 'webServerStartStop'; nested exception is org.springframework.boot.web.server.WebServerException: Unable to start embedded Tomcat server at org.springframework.context.support.DefaultLifecycleProcessor.doStart(DefaultLifecycleProcessor.java:181) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.context.support.DefaultLifecycleProcessor.access$200(DefaultLifecycleProcessor.java:54) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.context.support.DefaultLifecycleProcessor$LifecycleGroup.start(DefaultLifecycleProcessor.java:356) ~[spring-context-5.3.9.jar:5.3.9] at java.base/java.lang.Iterable.forEach(Iterable.java:75) ~[na:na] at org.springframework.context.support.DefaultLifecycleProcessor.startBeans(DefaultLifecycleProcessor.java:155) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.context.support.DefaultLifecycleProcessor.onRefresh(DefaultLifecycleProcessor.java:123) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.context.support.AbstractApplicationContext.finishRefresh(AbstractApplicationContext.java:935) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:586) ~[spring-context-5.3.9.jar:5.3.9] at org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext.refresh(ServletWebServerApplicationContext.java:145) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:754) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:434) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.SpringApplication.run(SpringApplication.java:338) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.SpringApplication.run(SpringApplication.java:1343) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.SpringApplication.run(SpringApplication.java:1332) ~[spring-boot-2.5.3.jar:2.5.3] at tacos.TacoCloudApplication.main(TacoCloudApplication.java:15) ~[classes/:na] at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na] at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:na] at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na] at java.base/java.lang.reflect.Method.invoke(Method.java:566) ~[na:na] at org.springframework.boot.devtools.restart.RestartLauncher.run(RestartLauncher.java:49) ~[spring-boot-devtools-2.5.3.jar:2.5.3] Caused by: org.springframework.boot.web.server.WebServerException: Unable to start embedded Tomcat server at org.springframework.boot.web.embedded.tomcat.TomcatWebServer.start(TomcatWebServer.java:229) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.web.servlet.context.WebServerStartStopLifecycle.start(WebServerStartStopLifecycle.java:43) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.context.support.DefaultLifecycleProcessor.doStart(DefaultLifecycleProcessor.java:178) ~[spring-context-5.3.9.jar:5.3.9] ... 19 common frames omitted Caused by: java.lang.IllegalArgumentException: standardService.connector.startFailed at org.apache.catalina.core.StandardService.addConnector(StandardService.java:243) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.springframework.boot.web.embedded.tomcat.TomcatWebServer.addPreviouslyRemovedConnectors(TomcatWebServer.java:282) ~[spring-boot-2.5.3.jar:2.5.3] at org.springframework.boot.web.embedded.tomcat.TomcatWebServer.start(TomcatWebServer.java:213) ~[spring-boot-2.5.3.jar:2.5.3] ... 21 common frames omitted Caused by: org.apache.catalina.LifecycleException: Protocol handler start failed at org.apache.catalina.connector.Connector.startInternal(Connector.java:1075) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.catalina.core.StandardService.addConnector(StandardService.java:239) ~[tomcat-embed-core-9.0.50.jar:9.0.50] ... 23 common frames omitted Caused by: java.lang.IllegalArgumentException: \path\to\mykeys.jks (系统找不到指定的路径。) at org.apache.tomcat.util.net.AbstractJsseEndpoint.createSSLContext(AbstractJsseEndpoint.java:99) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.AbstractJsseEndpoint.initialiseSsl(AbstractJsseEndpoint.java:71) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.NioEndpoint.bind(NioEndpoint.java:232) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.AbstractEndpoint.bindWithCleanup(AbstractEndpoint.java:1208) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.AbstractEndpoint.start(AbstractEndpoint.java:1294) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.coyote.AbstractProtocol.start(AbstractProtocol.java:614) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.catalina.connector.Connector.startInternal(Connector.java:1072) ~[tomcat-embed-core-9.0.50.jar:9.0.50] ... 25 common frames omitted Caused by: java.io.FileNotFoundException: \path\to\mykeys.jks (系统找不到指定的路径。) at java.base/java.io.FileInputStream.open0(Native Method) ~[na:na] at java.base/java.io.FileInputStream.open(FileInputStream.java:219) ~[na:na] at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157) ~[na:na] at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112) ~[na:na] at java.base/sun.net.www.protocol.file.FileURLConnection.connect(FileURLConnection.java:86) ~[na:na] at java.base/sun.net.www.protocol.file.FileURLConnection.getInputStream(FileURLConnection.java:184) ~[na:na] at org.apache.catalina.startup.CatalinaBaseConfigurationSource.getResource(CatalinaBaseConfigurationSource.java:118) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.SSLUtilBase.getStore(SSLUtilBase.java:197) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.SSLHostConfigCertificate.getCertificateKeystore(SSLHostConfigCertificate.java:207) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.SSLUtilBase.getKeyManagers(SSLUtilBase.java:282) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.SSLUtilBase.createSSLContext(SSLUtilBase.java:246) ~[tomcat-embed-core-9.0.50.jar:9.0.50] at org.apache.tomcat.util.net.AbstractJsseEndpoint.createSSLContext(AbstractJsseEndpoint.java:97) ~[tomcat-embed-core-9.0.50.jar:9.0.50] ... 31 common frames omitted Process finished with exit code 0 运行报错

解决：

需要创建mykeys.jks 文件，但是mykeys.jks 文件的创建需要参考许多文档，这里暂时先不创建，不管了。

根据您提供的错误日志，出现了几个问题：

1. **Caused by: java.lang.IllegalArgumentException: \path\to\mykeys.jks (系统找不到指定的路径。)**：这个错误表明无法找到指定路径下的密钥库文件 **mykeys.jks**。请确保在 **key-store** 配置中提供了正确的路径。您需要将 **\path\to\mykeys.jks** 替换为实际的文件路径，并确保该文件存在于指定位置。
2. **Caused by: org.springframework.boot.web.server.WebServerException: Unable to start embedded Tomcat server**：这个错误表明无法启动嵌入式的Tomcat服务器。这很可能是由于上述密钥库文件路径错误导致的。修复路径问题后，您应该能够解决此错误。

请根据上述建议检查密钥库文件路径，并确保文件存在于指定位置。如果问题仍然存在，请提供更多的信息，例如您的应用程序配置文件和任何其他相关错误日志，以便我可以更好地帮助您解决问题。

（2）

问题：

Mykeys.jks文件位置和内容：

请问密钥库文件 mykeys.jks放在哪，放在resources的根目录下吗？mykeys.jks里面的内容写些什么呢？

解决：

密钥库文件 **mykeys.jks** 可以放置在项目的任何位置，但是将其放置在 **resources** 目录下是一种常见的做法。**resources** 目录通常用于存放应用程序的资源文件。

关于 **mykeys.jks** 文件的内容，它是一个密钥库（KeyStore）文件，用于存储加密和身份验证相关的密钥和证书。在您的示例中，您正在使用该密钥库文件配置嵌入式Tomcat服务器的SSL设置。

您需要确保 **mykeys.jks** 文件是有效的密钥库文件，其中包含正确配置的证书、私钥和其他相关信息。通常，创建和配置密钥库文件是一个相对复杂的过程，涉及到生成自签名证书、申请证书签发机构 (CA) 签名的证书，以及配置证书链、密码等。

如果您不熟悉创建和配置密钥库文件的过程，建议参考相关的文档或教程，或者咨询安全专家来获得正确的配置。您可以使用工具如Keytool（Java自带的密钥库管理工具）来创建和管理密钥库文件。

在实际部署中，您可能需要从受信任的证书颁发机构（CA）获取有效的证书，并将其导入到密钥库文件中，以确保安全连接的可信性。

请注意，为了保护您的安全，应该避免在公共场所共享真实的密钥库文件和敏感信息。

（3）

问题：

Mykeys.jks 绝对位置：

“file:///path/to/mykeys.jks” 是什么意思，是指的resources根目录下的文件夹resources\path\to\mykeys.jks 吗？

解决：

不，"file:///path/to/mykeys.jks" 并不是指相对于 **resources** 目录的路径。它是一个示例路径，用来表示密钥库文件在文件系统中的位置。

在Spring Boot的配置中，"file:///path/to/mykeys.jks" 表示 **mykeys.jks** 文件的绝对路径。您需要将其替换为实际的文件系统路径，以指向您的密钥库文件所在的位置。

如果您将密钥库文件放在 **resources** 目录下，那么您可以通过以下方式指定路径：

* 如果您正在使用相对路径，可以使用 "classpath:" 前缀来引用 **resources** 目录，例如 "classpath:mykeys.jks"。
* 如果您希望使用绝对路径，您需要提供实际的文件系统路径，例如 "file:///C:/path/to/mykeys.jks"（根据您的操作系统，路径可能会有所不同）。

请根据您实际的文件存储位置来调整路径，确保应用程序能够正确找到并加载密钥库文件。

（4）

问题：

密钥库：

解决：

使用 “# Embedded server server: # port: 0 port: 8443 ssl: key-store: file:///path/to/mykeys.jks key-store-password: letmein key-password: letmein”配置，能够自动创建密钥库文件吗？

ChatGPT

不，使用上述配置无法自动创建密钥库文件。配置中的 **key-store** 属性用于指定密钥库文件的路径，但它不会自动创建该文件。

您需要手动创建密钥库文件，并确保在指定的路径下存在该文件。密钥库文件通常使用工具如Keytool来创建，其中包含配置的证书、私钥和其他相关信息。

在配置中，您需要提供正确的密钥库文件路径 (**key-store**)、密钥库密码 (**key-store-password**) 和密钥密码 (**key-password**)，以便应用程序能够加载和使用密钥库文件中的密钥和证书。

请注意，在生产环境中，为了保护安全性，建议从受信任的证书颁发机构（CA）获取有效的证书，并将其导入到密钥库文件中。同时，要确保适当地保护密钥库文件的访问权限，以防止未经授权的访问。

总结而言，配置中的 **key-store** 属性仅用于指定密钥库文件的路径，您需要手动创建密钥库文件并确保路径正确。

（5）

问题：

创建密钥库：

请举个密钥库创建的例子。

解决：

ChatGPT

当创建密钥库时，您可以使用 Java 的 **keytool** 工具。下面是一个示例，演示如何使用 **keytool** 创建一个密钥库文件并生成自签名的证书：

1. 打开命令行终端（或命令提示符）。
2. 导航到您希望创建密钥库文件的目录。
3. 运行以下命令创建密钥库文件（例如，名为 **mykeys.jks**）：

keytool -genkeypair -alias mykey -keyalg RSA -keysize 2048 -keystore mykeys.jks -validity 365

在此命令中，我们使用 **-genkeypair** 选项生成密钥对，**-alias** 指定别名为 **mykey**，**-keyalg** 指定密钥算法为 RSA，**-keysize** 指定密钥大小为 2048 位，**-keystore** 指定密钥库文件名为 **mykeys.jks**，**-validity** 指定证书有效期为 365 天。

1. 按照提示输入密钥库密码和其他相关信息（如姓名、组织等）。
2. 完成后，密钥库文件 **mykeys.jks** 将被创建在当前目录中。

现在，您已经成功创建了一个包含自签名证书的密钥库文件。您可以将该密钥库文件用于 SSL/TLS 配置，以在 Spring Boot 或其他应用程序中启用加密通信。记得在应用程序配置中指定正确的密钥库文件路径和密码。

请注意，上述示例中的自签名证书仅适用于开发和测试目的。在生产环境中，建议从受信任的证书颁发机构（CA）获取有效的证书，以确保安全性和信任度。

（6）

问题：

Bad Request This combination of host and port requires TLS.

解决：

该错误提示表明请求需要使用 TLS（Transport Layer Security）进行加密通信。这通常发生在您尝试通过 HTTP（非加密）连接发送请求到要求使用 TLS 的主机和端口时。

TLS 是一种安全协议，用于在客户端和服务器之间建立加密的通信通道，以确保数据的保密性和完整性。如果服务器要求使用 TLS 进行通信，您需要将请求发送到相应的 HTTPS（使用 TLS 的 HTTP）端点。

请确保您的请求是通过 HTTPS 发送到正确的主机和端口。您可以修改请求的 URL，将 **http://** 替换为 **https://**，以确保使用 TLS 进行加密通信。

如果您正在开发应用程序，并且希望在本地测试时使用 TLS，请确保您的本地开发服务器配置正确，并且在进行 HTTPS 请求时使用有效的证书。您可以使用自签名证书进行测试，或者在开发环境中使用可信的证书颁发机构（CA）颁发的有效证书。

20230519\_2：

问题：

Logback.xml ?

What's the logback.xml in SpringBoot Application?

解决：

The **logback.xml** file is a configuration file used in Spring Boot applications to customize the logging behavior. It is based on the Logback framework, which is a popular logging library in the Java ecosystem.

By default, Spring Boot uses Logback as the logging framework, and the **logback.xml** file provides a way to configure various aspects of logging, such as log levels, appenders, formatters, and loggers.

Here is an example of a basic **logback.xml** configuration file:

<configuration>

<appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="INFO">

<appender-ref ref="CONSOLE"/>

</root>

</configuration>

In this example:

* The **<appender>** element defines an appender named "CONSOLE" of type **ConsoleAppender**, which directs log output to the console.
* The **<encoder>** element specifies the log message pattern using the **%d**, **%thread**, **%level**, **%logger**, and **%msg** placeholders. This example uses a timestamp, thread name, log level, logger name, and the log message itself.
* The **<root>** element sets the root logger level to "INFO" and associates it with the "CONSOLE" appender.

You can customize the **logback.xml** file based on your specific logging requirements. You can define additional appenders, configure log levels for different packages or classes, specify file appenders, configure rolling policies, and much more. The **logback.xml** file provides a flexible and powerful way to control logging in your Spring Boot application.

问题2：

So where the route or absolute route for "logback.xml"? If "logback.xml" is the same location like "pom.xml" or "application.yml"?

解决：

In a typical Spring Boot application, the **logback.xml** file is usually placed in the **src/main/resources** directory along with other resource files like **application.yml** or **application.properties**. The **src/main/resources** directory is a common location for storing configuration files, static files, and resources in a Spring Boot project.

When you place the **logback.xml** file in the **src/main/resources** directory, it will be automatically included in the classpath of your application during the build process. This means that the file can be accessed and loaded by the logging framework at runtime.

Here's an example directory structure for a Spring Boot project:

src

├── main

│ ├── java

│ │ └── com.example.myapp

│ │ └── Application.java

│ ├── resources

│ │ ├── logback.xml

│ │ ├── application.yml

│ │ └── static

│ │ └── styles.css

│ └── webapp

│ └── WEB-INF

│ └── views

│ └── index.jsp

└── test

├── java

└── resources

As shown in the example, the **logback.xml** file is placed in the **src/main/resources** directory alongside the **application.yml** file and other resource files.

By default, when you run your Spring Boot application, the logging framework will look for the **logback.xml** file in the classpath and automatically configure logging based on the specified configuration. If the **logback.xml** file is not found, the logging framework will use its default configuration.

Note that you can also use alternative names for the logging configuration file, such as **logback-spring.xml** or **logback.groovy**. Spring Boot will automatically detect and load the appropriate logging configuration file based on the naming conventions.