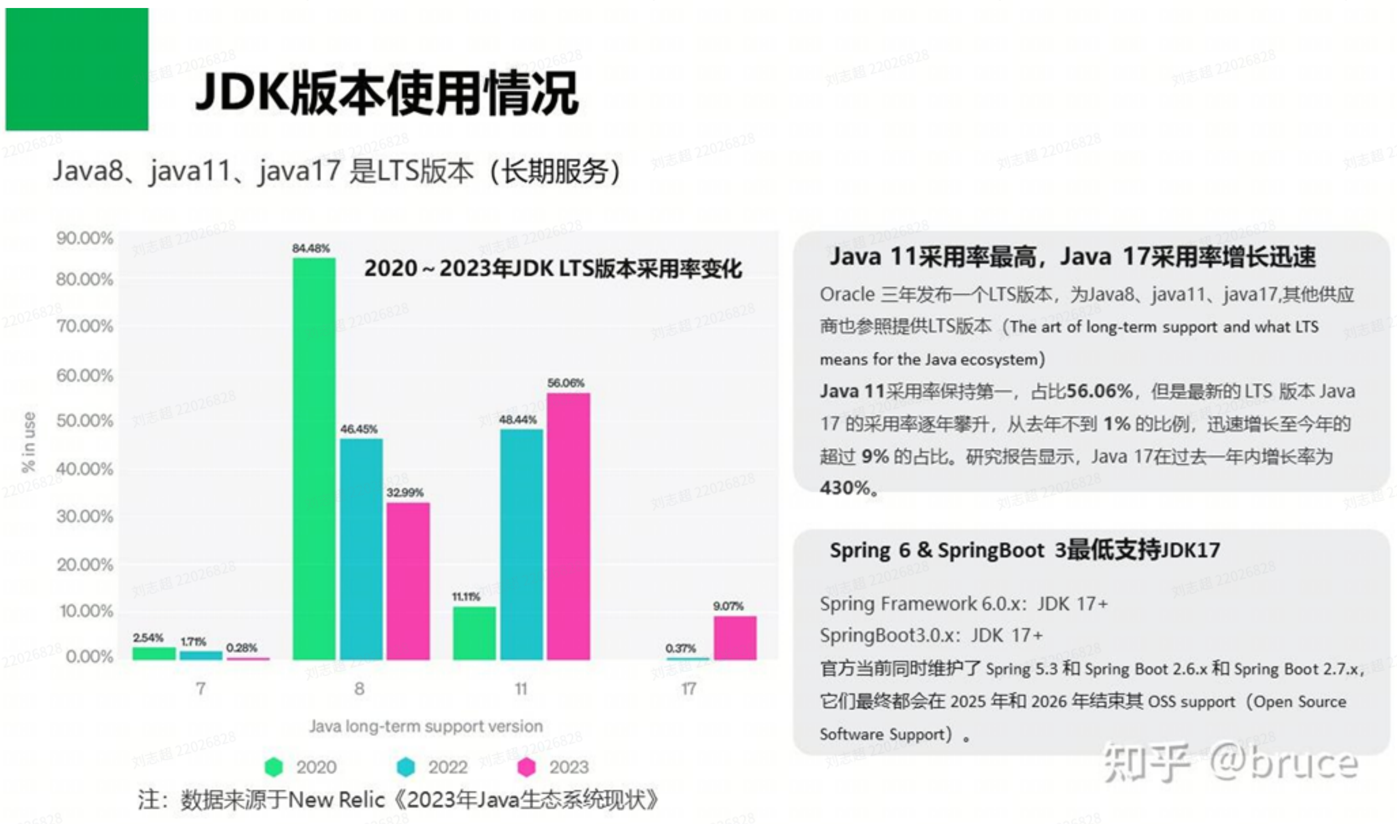
JDK8升级JDK17指引

1. 为什么要升级

①JDK17是LTS版本，采用率增速迅速

②java开源生态发展，逐步要求最低JDK17

③JDK17 ZGC可以控制垃圾回收10ms以内，并且增加了很多新的功能特性



JDK版本使用情况

Java8、java11、java17 是LTS版本（长期服务）

注：数据来源于New Relic《2023年Java生态系统现状》

Java 11采用率最高，Java 17采用率增长迅速

Oracle 三年发布一个LTS版本，为Java8、java11、java17,其他供应商也参照提供LTS版本（The art of long-term support and what LTS means for the Java ecosystem）

Java 11采用率保持第一，占比**56.06%**，但是最新的 LTS 版本 Java 17 的采用率逐年攀升，从去年不到 **1%** 的比例，迅速增长至今年的超过 **9%** 的占比。研究报告显示，Java 17在过去一年内增长率为 **430%**。

Spring 6 & SpringBoot 3最低支持JDK17

Spring Framework 6.0.x：JDK 17+

SpringBoot3.0.x：JDK 17+

官方当前同时维护了 Spring 5.3 和 Spring Boot 2.6.x 和 Spring Boot 2.7.x，它们最终都会在 2025 年和 2026 年结束其 OSS support（Open Source Software Support）。

2. 本地开发环境配置

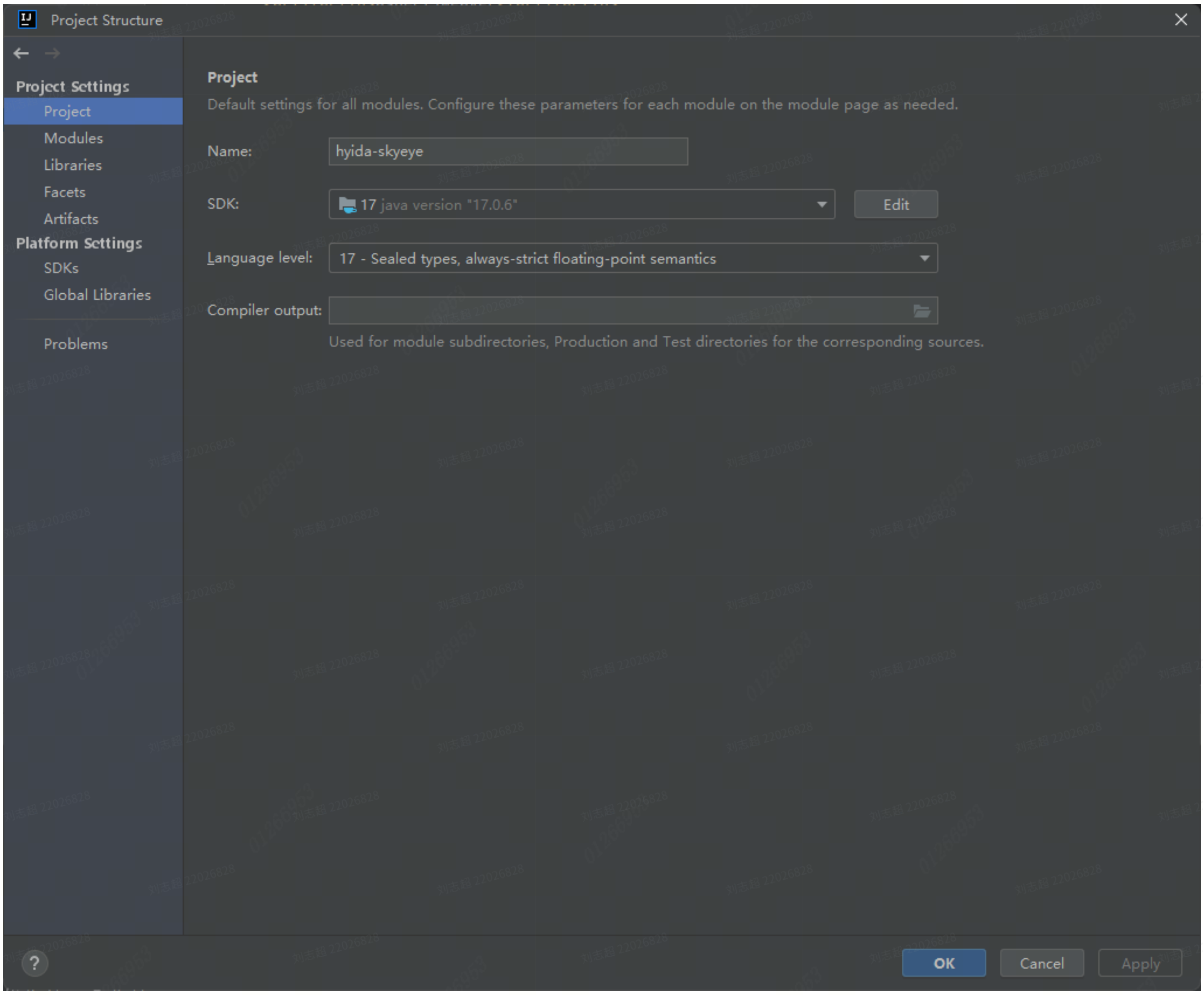
下载地址：https://github.com/adoptium/temurin17-binaries/releases

下载安装OpenJDK17，配置好JAVA_HOME,CLASSPATH,PATH变量

```
C:\Users\01266953>java -version
openjdk version "17.0.10" 2024-01-16
OpenJDK Runtime Environment OpenLogic-OpenJDK (build 17.0.10+7-adhoc..jdk17u)
OpenJDK 64-Bit Server VM OpenLogic-OpenJDK (build 17.0.10+7-adhoc..jdk17u,
mixed mode, sharing)

C:\Users\01266953>
```

把工程的jdk默认选项配置成JDK17



Project Structure

Project Settings
- Project
- Modules
- Libraries
- Facets
- Artifacts

Platform Settings
- SDKs
- Global Libraries

Problems

Project

Default settings for all modules. Configure these parameters for each module on the module page as needed.

Name: hyida-skyeye

SDK: 17 java version "17.0.6"

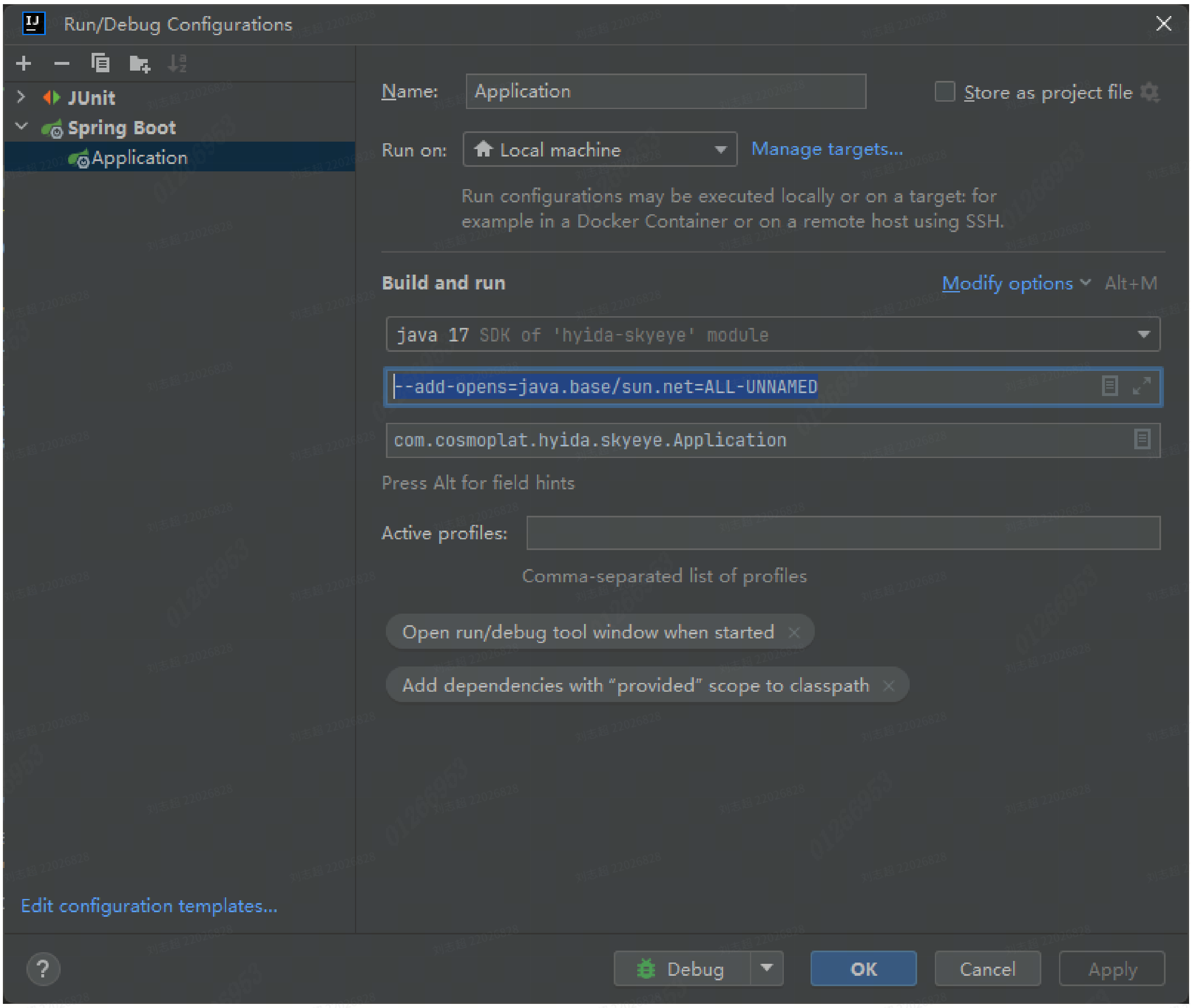
Language level: 17 - Sealed types, always-strict floating-point semantics

Compiler output:

Used for module subdirectories, Production and Test directories for the corresponding sources.

运行main函数所在的主类时，添加上JVM参数

```
--add-opens=java.base/java.time=ALL-UNNAMED --add-opens=java.base/sun.net=ALL-UNNAMED
```


Run/Debug Configurations

Name: Application

Store as project file

Run on: Local machine

Manage targets...

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

Modify options Alt+M

java 17 SDK of 'hyida-skyeye' module

```
--add-opens=java.base/sun.net=ALL-UNNAMED
```

```
com.cosmoplat.hyida.skyeye.Application
```

Press Alt for field hints

Active profiles:

Comma-separated list of profiles

Open run/debug tool window when started

Add dependencies with “provided” scope to classpath

3. 代码修改

3.1 pom文件中修改jdk版本为17

```
<java.version>17</java.version>
```

3.2 使用了druid-spring-boot-starter的，版本需要升级

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid-spring-boot-starter</artifactId>
    <version>1.2.22</version>
</dependency>
```

4. 打包流水线

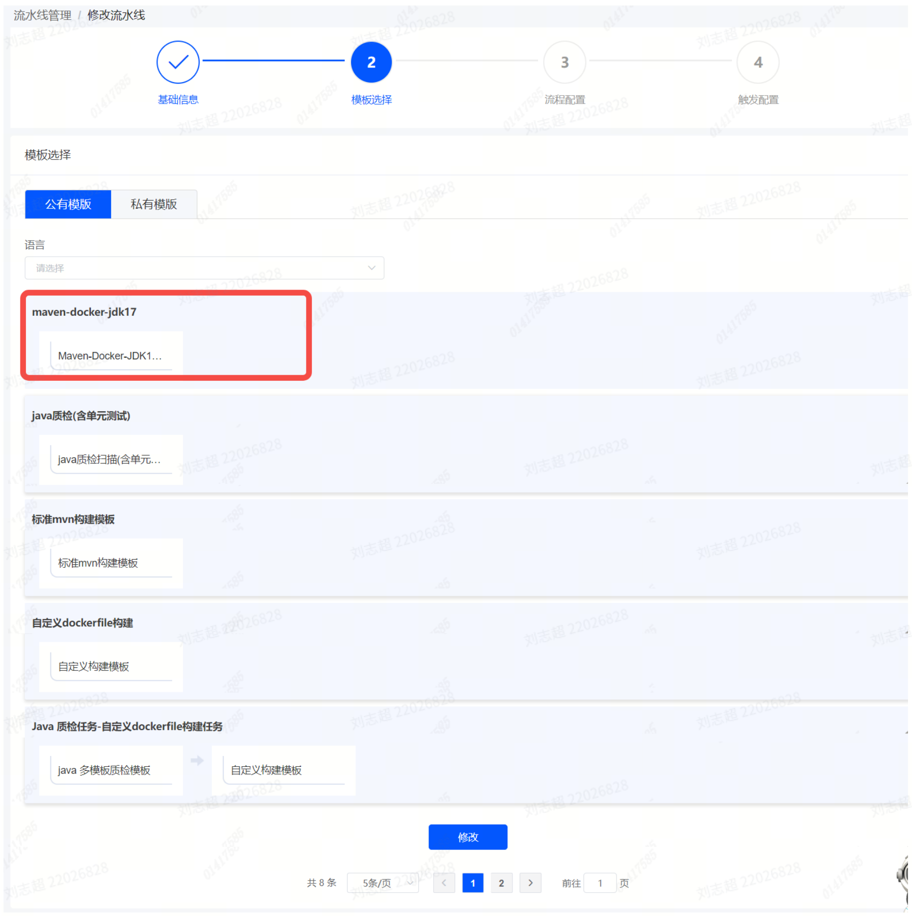
1. 修改Dockerfile（增加红色部分，/opt/app/路径根据实际情况更改）

```
FROM registry2-qingdao.cosmoplat.com/base/jdk:17

MAINTAINER liuxz

COPY ./target/portal-apisix-server-0.0.1.jar portal-apisix-server.jar
ENV ADD_OPENS="--add-opens=java.base/sun.net=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.time=ALL-UNNAMED --add-opens=java.base/java.text=ALL-UNNAMED "
EVN JVM_OPT = ""
EXPOSE 9003
#ENTRYPOINT ["java","-javaagent:/khaos/agent/skywalking-agent.jar","-Dskywalking.agent.service_name=portal-apisix-server","-Dskywalking.collector.backend_service=10.206.128.103:31150","-Xms512m","-Xmx512m","-XX:CompressedClassSpaceSize=256m","-XX:MetaspaceSize=1024M","-jar","portal-apisix-server.jar"]
ENTRYPOINT java ${ADD_OPENS} ${JVM_OPT} -Xms512m -Xmx512m -XX:CompressedClassSpaceSize=256m -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M -jar portal-apisix-server.jar
```

2. 开发者平台->自定义流水线，选择【自定义构建模板】


流水线管理 / 修改流水线

基础信息 → 模板选择 → 流程配置 → 触发配置

模板选择

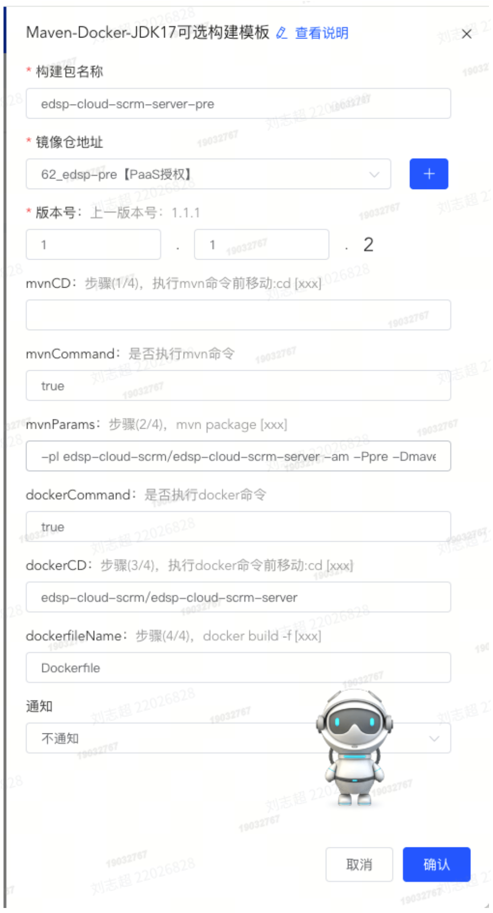
公有模版 | 私有模版

语言: 请选择

- maven-docker-jdk17: Maven-Docker-JDK1...
- java质检(含单元测试): java质检扫描(含单元...
- 标准mvn构建模板: 标准mvn构建模板
- 自定义dockerfile构建: 自定义构建模板
- Java 质检任务-自定义dockerfile构建任务: java 多模板质检模板 → 自定义构建模板

修改

共 8 条 5条/页 < 1 2 > 前往 1 页


Maven-Docker-JDK17可选构建模板 查看说明

* 构建包名称: edsp-cloud-scrm-server-pre

* 镜像仓地址: 62_edsp-pre【PaaS授权】

* 版本号: 上一版本号: 1.1.1
1 . 1 . 2

mvnCD: 步骤(1/4), 执行mvn命令前移动:cd [xxx]

mvnCommand: 是否执行mvn命令
true

mvnParams: 步骤(2/4), mvn package [xxx]
-pl edsp-cloud-scrm/edsp-cloud-scrm-server -am -Ppre -Dmave

dockerCommand: 是否执行docker命令
true

dockerCD: 步骤(3/4), 执行docker命令前移动:cd [xxx]
edsp-cloud-scrm/edsp-cloud-scrm-server

dockerfileName: 步骤(4/4), docker build -f [xxx]
Dockerfile

通知: 不通知

取消 确认

5. 问题

5.1 JDK内部API引用禁用

sun.misc.Base64Encode和sun.misc.Base64Decode替换为：
org.apache.commons.codec.binary.Base64

```
BASE64Decoder decoder = new BASE64Decoder();
byte[] bytes1 = decoder.decodeBuffer(signStr);
变更为:
byte[] bytes1 = Base64.getDecoder().decode(base64string);



new sun.misc.BASE64Encoder().encode(signData);
变更为:
Base64.getEncoder().encodeToString(signData);
```

5.2 循环依赖问题

（不鼓励循环依赖引用，默认情况下是禁止的），放开限制。

2种方法：

1. 配置文件中增加：

```
spring:
  main:
    allow-circular-references: true
```

2. 使用SpringApplicationBuilder来启动Spring Boot应用,并通过allowCircularReferences(true)方法开启了循环依赖支持。

```

public class xxxxxxApplication {
    public static void main(String[] args) {
        new SpringApplicationBuilder(xxxxxxApplication.class).allowCircularReferences(true).run(args);
        System.out.println("----------------------xxxxxxApplication---------------------");
    }
}
```

5.3 SpringBoot版本SpringCloud版本

Springboot	SpringCloud	alibaba-cloud
2.7.14	2021.0.5	2021.0.6.0

5.3.1 衍生cloud负载均衡问题 ribbon弃用

spring-cloud-starter-loadbalancer替换ribbon

1. **nacos 2021 版本已经没有自带 ribbon 的整合，所以需要引入另一个支持的 jar 包 loadbalancer**
2. **nacos 2021 版本已经取消了对 ribbon 的支持，所以无法通过修改 Ribbon 负载均衡的模式来实现 nacos 提供的负载均衡模式**

5.3.2 Junit4 不支持，

5.4 documentationPluginsBootstrapper 错误

配置文件中增加：spring.mvc.pathmatch.matching-strategy=ant-path-matcher

如果使用了swagger，在配置文件中增加上述内容后还报错，需要在swaggerConfig中增加一下代码：

```java
import org.springframework.boot.actuate.autoconfigure.endpoint.web.CorsEndpointProperties;
import org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties;
import org.springframework.boot.actuate.autoconfigure.web.server.ManagementPortType;
import org.springframework.boot.actuate.endpoint.ExposableEndpoint;
import org.springframework.boot.actuate.endpoint.web.EndpointLinksResolver;
import org.springframework.boot.actuate.endpoint.web.EndpointMapping;
import org.springframework.boot.actuate.endpoint.web.EndpointMediaTypes;
import org.springframework.boot.actuate.endpoint.web.ExposableWebEndpoint;
import org.springframework.boot.actuate.endpoint.web.WebEndpointsSupplier;
import org.springframework.boot.actuate.endpoint.web.annotation.ControllerEndpointsSupplier;
import org.springframework.boot.actuate.endpoint.web.annotation.ServletEndpointsSupplier;
import org.springframework.boot.actuate.endpoint.web.servlet.WebMvcEndpointHandlerMapping;
import org.springframework.core.env.Environment;
import org.springframework.util.StringUtils;
import com.google.common.collect.Lists;

@EnableOpenApi
@Configuration
public class Swagger2Config {
    .........省略
        @Bean
        public WebMvcEndpointHandlerMapping webEndpointServletHandlerMapping(WebEndpointsSupplier webEndpointsSupplier,
                        ServletEndpointsSupplier servletEndpointsSupplier, ControllerEndpointsSupplier controllerEndpointsSupplier, EndpointMediaTypes endpointMediaTypes,
                        CorsEndpointProperties corsProperties, WebEndpointProperties webEndpointProperties, Environment environment) {
                List<ExposableEndpoint<?>> allEndpoints = Lists.newArrayList();
                Collection<ExposableWebEndpoint> webEndpoints = webEndpointsSupplier.getEndpoints();
                allEndpoints.addAll(webEndpoints);
                allEndpoints.addAll(servletEndpointsSupplier.getEndpoints());
                allEndpoints.addAll(controllerEndpointsSupplier.getEndpoints());
                String basePath = webEndpointProperties.getBasePath();
                EndpointMapping endpointMapping = new EndpointMapping(basePath);
                boolean shouldRegisterLinksMapping = this.shouldRegisterLinksMapping(webEndpointProperties, environment, basePath);
                return new WebMvcEndpointHandlerMapping(endpointMapping, webEndpoints, endpointMediaTypes, corsProperties.toCorsConfiguration(),
                                new EndpointLinksResolver(allEndpoints, basePath), shouldRegisterLinksMapping, null);
        }

        private boolean shouldRegisterLinksMapping(WebEndpointProperties webEndpointProperties, Environment environment, String basePath) {
                return webEndpointProperties.getDiscovery().isEnabled()
                                && (StringUtils.hasText(basePath) || ManagementPortType.get(environment).equals(ManagementPortType.DIFFERENT));
        }

     。。。。。。。。。省略
 }
```

5.5 错误 No spring.config.import set

```
org.springframework.cloud.commons.ConfigDataMissingEnvironmentPostProcessor$ImportException: No spring.config.import set
        at org.springframework.cloud.commons.ConfigDataMissingEnvironmentPostProcessor.postProcessEnvironment(ConfigDataMissingEnvironmentPostProcessor.java:82)
        at org.springframework.boot.env.EnvironmentPostProcessorApplicationListener.onApplicationEnvironmentPreparedEvent(EnvironmentPostProcessorApplicationListener.java:102)
        at org.springframework.boot.env.EnvironmentPostProcessorApplicationListener.onApplicationEvent(EnvironmentPostProcessorApplicationListener.java:87)
        at org.springframework.context.event.SimpleApplicationEventMulticaster.doInvokeListener(SimpleApplicationEventMulticaster.java:176)
        at org.springframework.context.event.SimpleApplicationEventMulticaster.invokeListener(SimpleApplicationEventMulticaster.java:169)
        at org.springframework.context.event.SimpleApplicationEventMulticaster.multicastEvent(SimpleApplicationEventMulticaster.java:143)
        at org.springframework.context.event.SimpleApplicationEventMulticaster.multicastEvent(SimpleApplicationEventMulticaster.java:131)
        at org.springframework.boot.context.event.EventPublishingRunListener.environmentPrepared(EventPublishingRunListener.java:85)
        at org.springframework.boot.SpringApplicationRunListeners.lambda$environmentPrepared$2(SpringApplicationRunListeners.java:66)
        at java.util.ArrayList.forEach(ArrayList.java:1257)
        at org.springframework.boot.SpringApplicationRunListeners.doWithListeners(SpringApplicationRunListeners.java:120)
        at org.springframework.boot.SpringApplicationRunListeners.doWithListeners(SpringApplicationRunListeners.java:114)
        at org.springframework.boot.SpringApplicationRunListeners.environmentPrepared(SpringApplicationRunListeners.java:65)
        at org.springframework.boot.SpringApplication.prepareEnvironment(SpringApplication.java:343)
        at org.springframework.boot.SpringApplication.run(SpringApplication.java:301)
        at org.springframework.boot.SpringApplication.run(SpringApplication.java:1303)
        at org.springframework.boot.SpringApplication.run(SpringApplication.java:1292)
        at com.cosmoplat.edsp.process.ProcessServerApplication.main(ProcessServerApplication.java:21)
18:20:53.962 [main] ERROR org.springframework.boot.diagnostics.LoggingFailureAnalysisReporter -
```

需要引入依赖：

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-bootstrap</artifactId>
</dependency>
```

5.6 JDK9之后 jeva EE 剥离 javax引用报错

```xml
<jakarta.annotation.version>1.3.5</jakarta.annotation.version>
<javax.jws.version>1.1</javax.jws.version>
<javax.xml.ws.version>2.3.1</javax.xml.ws.version>


<dependency>
    <groupId>jakarta.annotation</groupId>
    <artifactId>jakarta.annotation-api</artifactId>
    <version>${jakarta.annotation.version}</version>
</dependency>
<dependency>
    <groupId>javax.jws</groupId>
    <artifactId>javax.jws-api</artifactId>
    <version>${javax.jws.version}</version>
</dependency>
<dependency>
    <groupId>javax.xml.ws</groupId>
    <artifactId>jaxws-api</artifactId>
    <version>${javax.xml.ws.version}</version>
</dependency>
```

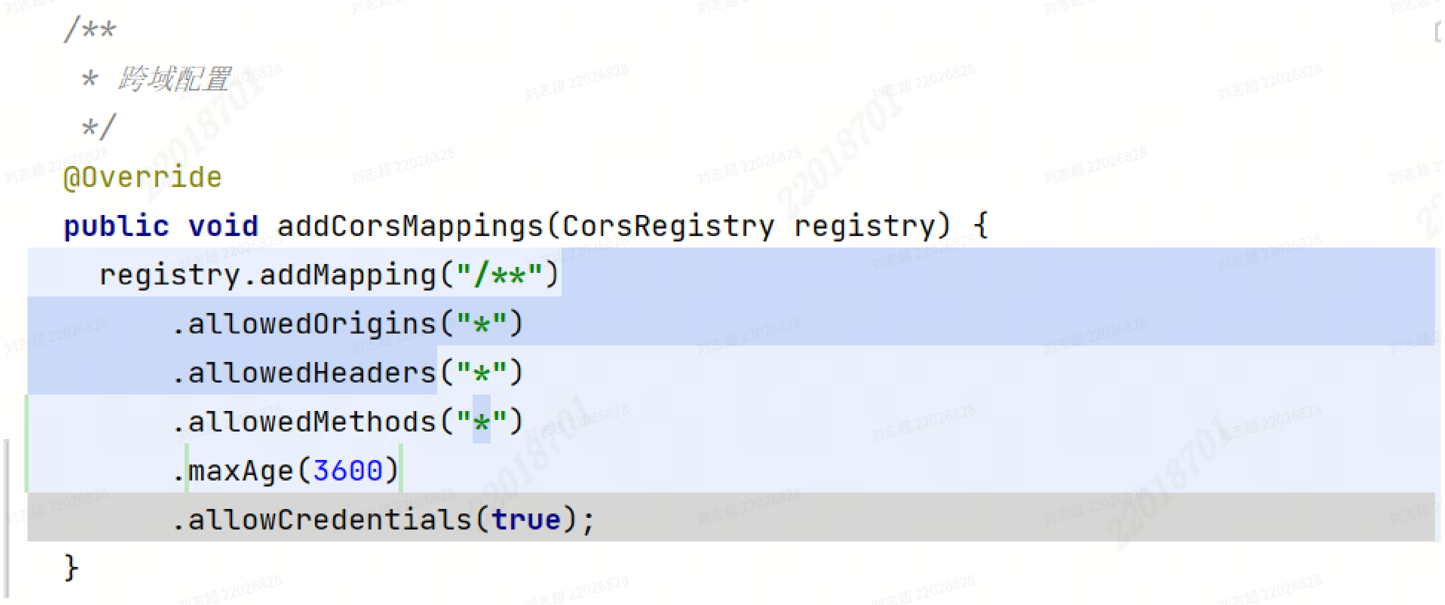
5.7 单元测试 junit

org.junit.Assert不存在，使用org.junit.jupiter.api.Assertions替换

5.8 升级后所有请求跨域问题

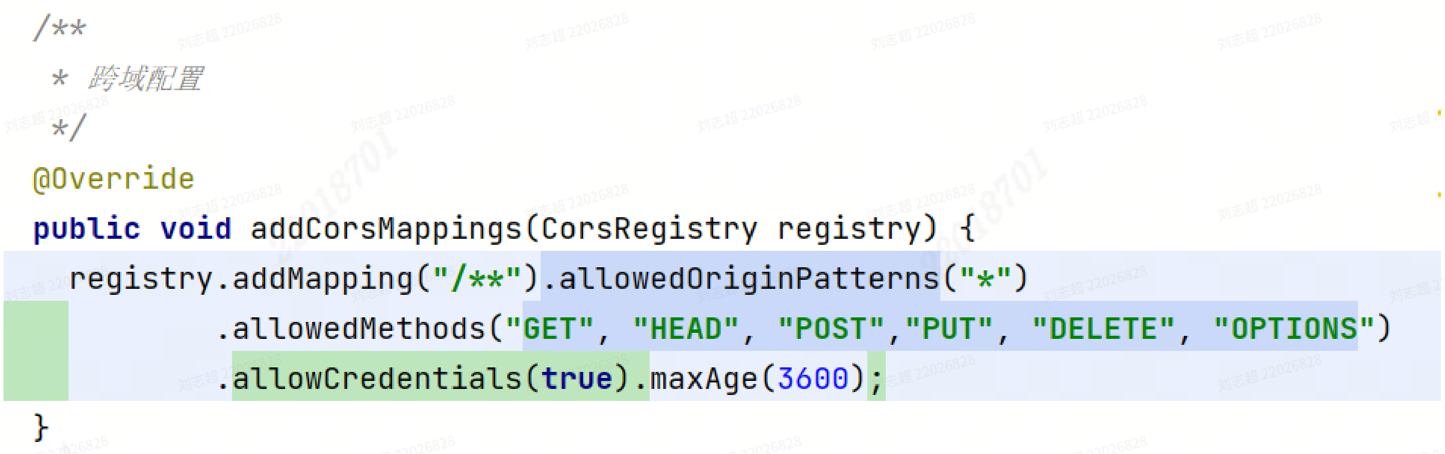
错误信息：java.lang.IllegalArgumentException: When allowCredentials is true, allowedOrigins cannot contain the special value "*" since that cannot be set on the "Access-Control-Allow-Origin" response header. To allow credentials to a set of origins, list them explicitly or consider using "allowedOriginPatterns" instead.

原跨域配置写法：



```
/**
 * 跨域配置
 */
@Override
public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
            .allowedOrigins("*")
            .allowedHeaders("*")
            .allowedMethods("*")
            .maxAge(3600)
            .allowCredentials(true);
}
```

修改后代码如下：



```
/**
 * 跨域配置
 */
@Override
public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**").allowedOriginPatterns("*")
            .allowedMethods("GET", "HEAD", "POST","PUT", "DELETE", "OPTIONS")
            .allowCredentials(true).maxAge(3600);
}
```

5.9

参考

SPring2.2.5升级2.7.2的一些坑坑

https://maimai.cn/article/detail?fid=1761958134&efid=S17pO-SkF9VvGt479Y8ycg