1. 约定 > 配置 > 编码

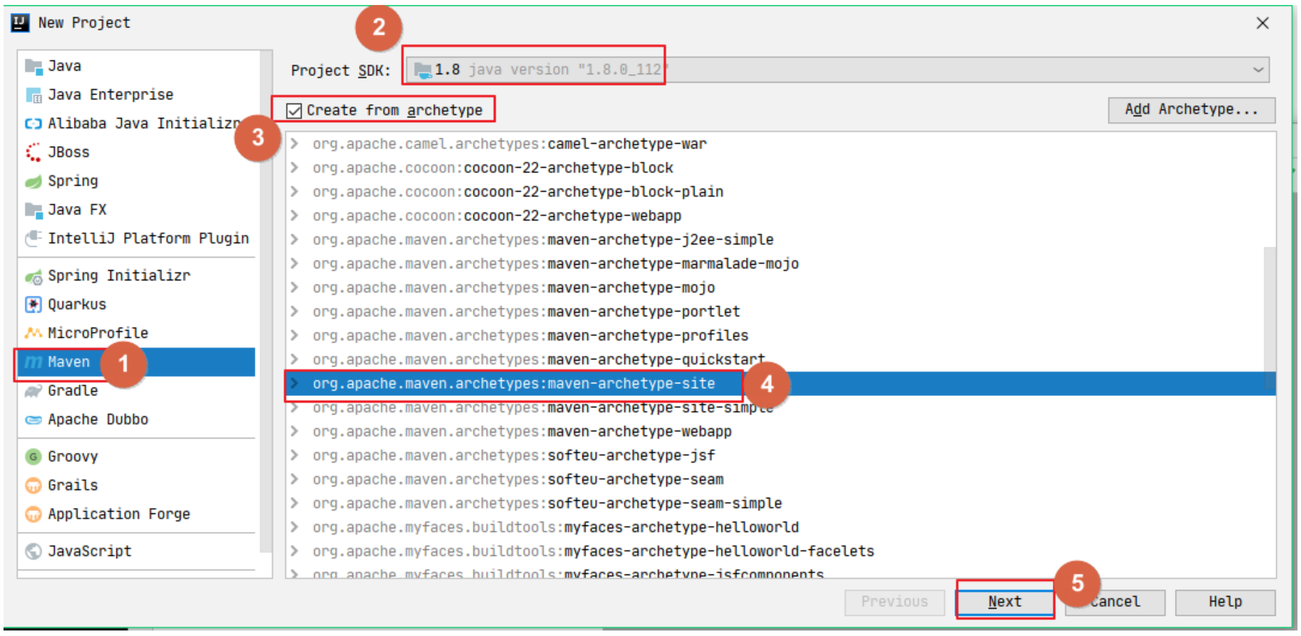
2. IDEA新建project工作空间

2.1. 微服务Cloud整体聚合工程

父工程步骤:

2.1.1. New Project

New Project -> Maven -> Create from archetype -> maven-archetype-site


New Project

Java
Java Enterprise
Alibaba Java Initializr
JBoss
Spring
Java FX
IntelliJ Platform Plugin
Spring Initializr
Quarkus
MicroProfile
Maven
Gradle
Apache Dubbo
Groovy
Grails
Application Forge
JavaScript

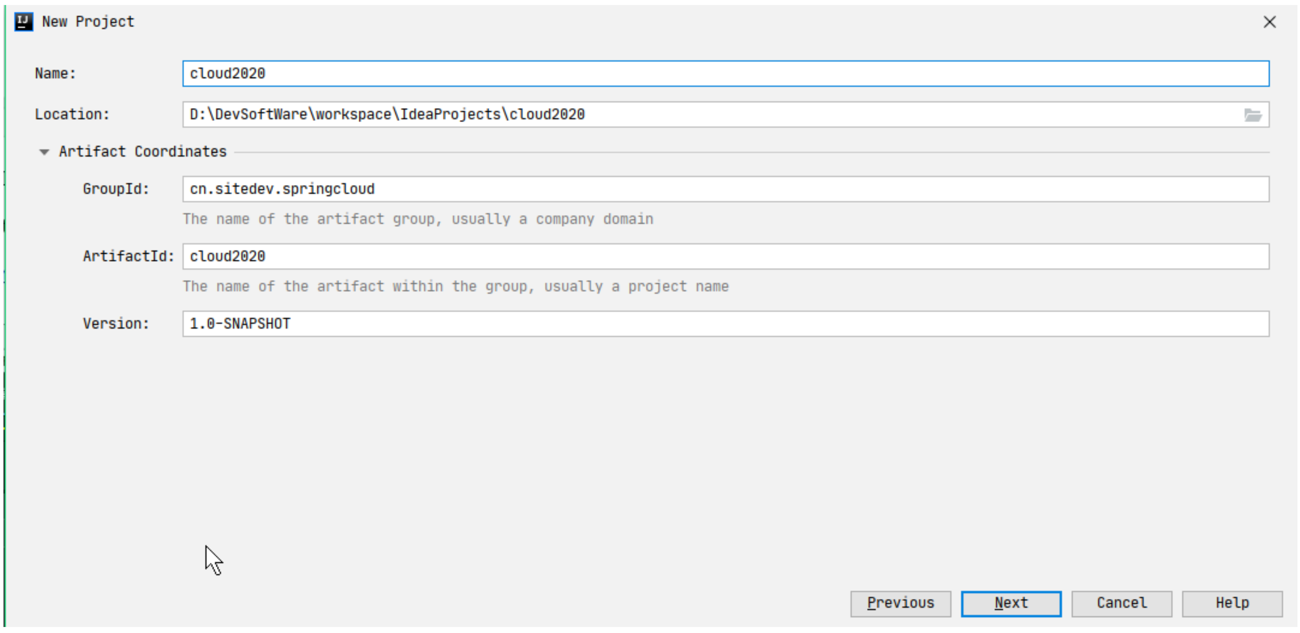
Project SDK: 1.8 java version "1.8.0_112"

☑ Create from archetype

Add Archetype...

```
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
org.apache.maven.archetypes:maven-archetype-mojo
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles
org.apache.maven.archetypes:maven-archetype-quickstart
org.apache.maven.archetypes:maven-archetype-site
org.apache.maven.archetypes:maven-archetype-site-simple
org.apache.maven.archetypes:maven-archetype-webapp
org.apache.maven.archetypes:softeu-archetype-jsf
org.apache.maven.archetypes:softeu-archetype-seam
org.apache.maven.archetypes:softeu-archetype-seam-simple
org.apache.myfaces.buildtools:myfaces-archetype-helloworld
org.apache.myfaces.buildtools:myfaces-archetype-helloworld-facelets
org.apache.myfaces.buildtools:myfaces-archetype-jsfcomponents
```

2.1.2. 聚合总父工程名字


New Project

Name:	cloud2020
Location:	D:\DevSoftWare\workspace\IdeaProjects\cloud2020

Artifact Coordinates

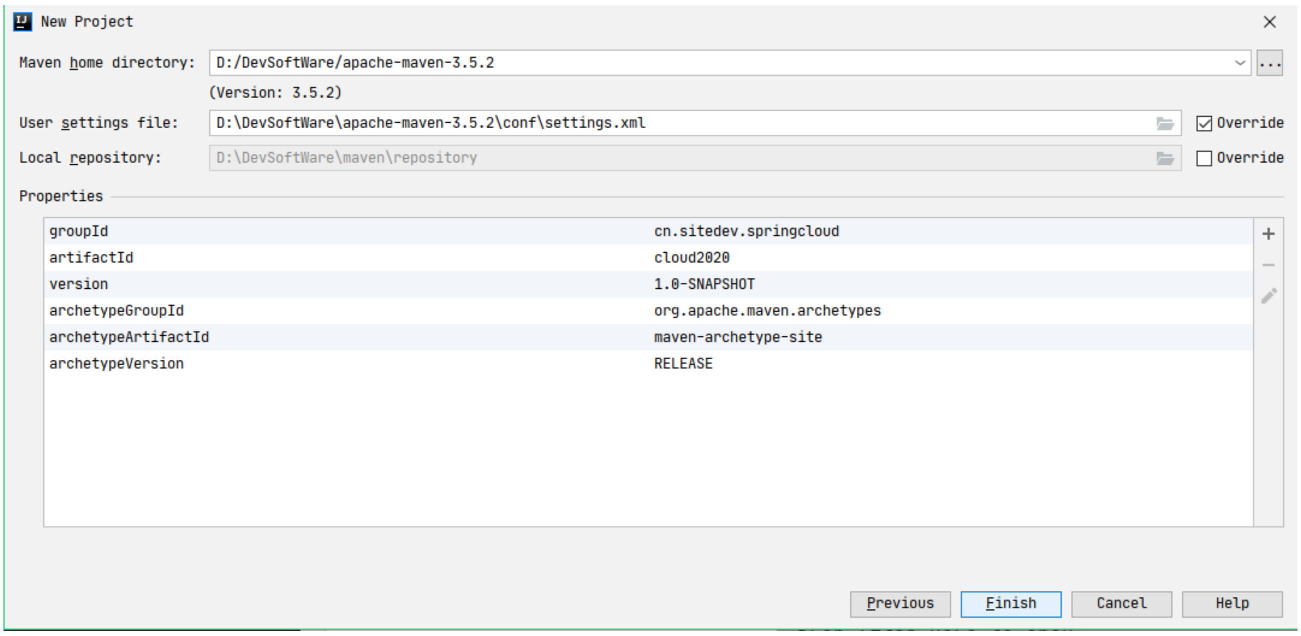
GroupId:	cn.sitedev.springcloud
	The name of the artifact group, usually a company domain
ArtifactId:	cloud2020
	The name of the artifact within the group, usually a project name
Version:	1.0-SNAPSHOT

Name : cloud2020

GroupId : cn.sitedev.springcloud

ArtifactId : cloud2020

2.1.3. Maven 选版本


New Project

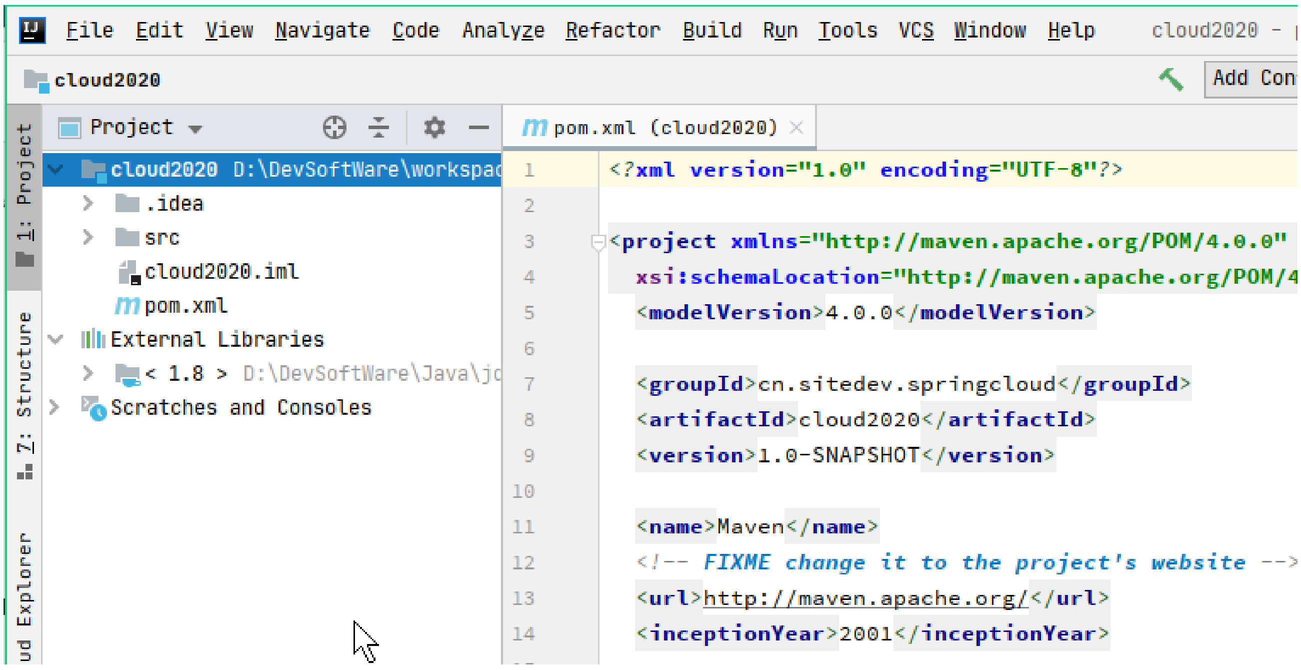
Maven home directory: D:/DevSoftWare/apache-maven-3.5.2
(Version: 3.5.2)
User settings file: D:\DevSoftWare\apache-maven-3.5.2\conf\settings.xml [✓ Override]
Local repository: D:\DevSoftWare\maven\repository [Override]

Properties

groupId	cn.sitedev.springcloud
artifactId	cloud2020
version	1.0-SNAPSHOT
archetypeGroupId	org.apache.maven.archetypes
archetypeArtifactId	maven-archetype-site
archetypeVersion	RELEASE

选择自定义版本: Maven 3.5.2

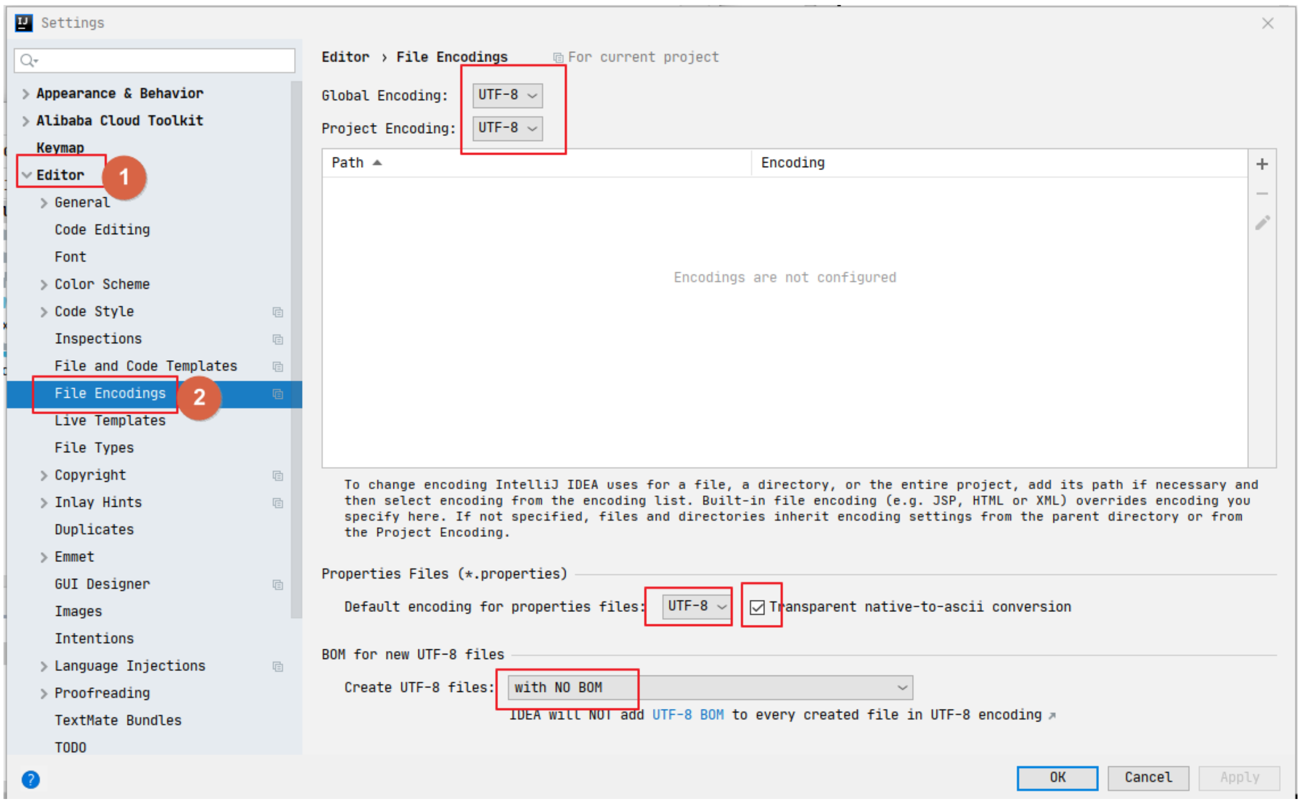
2.1.4. 工程名字



```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xsi:schemaLocation="http://maven.apache.org/POM/4
  <modelVersion>4.0.0</modelVersion>

  <groupId>cn.sitedev.springcloud</groupId>
  <artifactId>cloud2020</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>Maven</name>
  <!-- FIXME change it to the project's website -->
  <url>http://maven.apache.org/</url>
  <inceptionYear>2001</inceptionYear>
```

2.1.5. 字符编码

File -> Settings -> Editor -> File Encodings



Global Encoding: UTF-8
Project Encoding: UTF-8

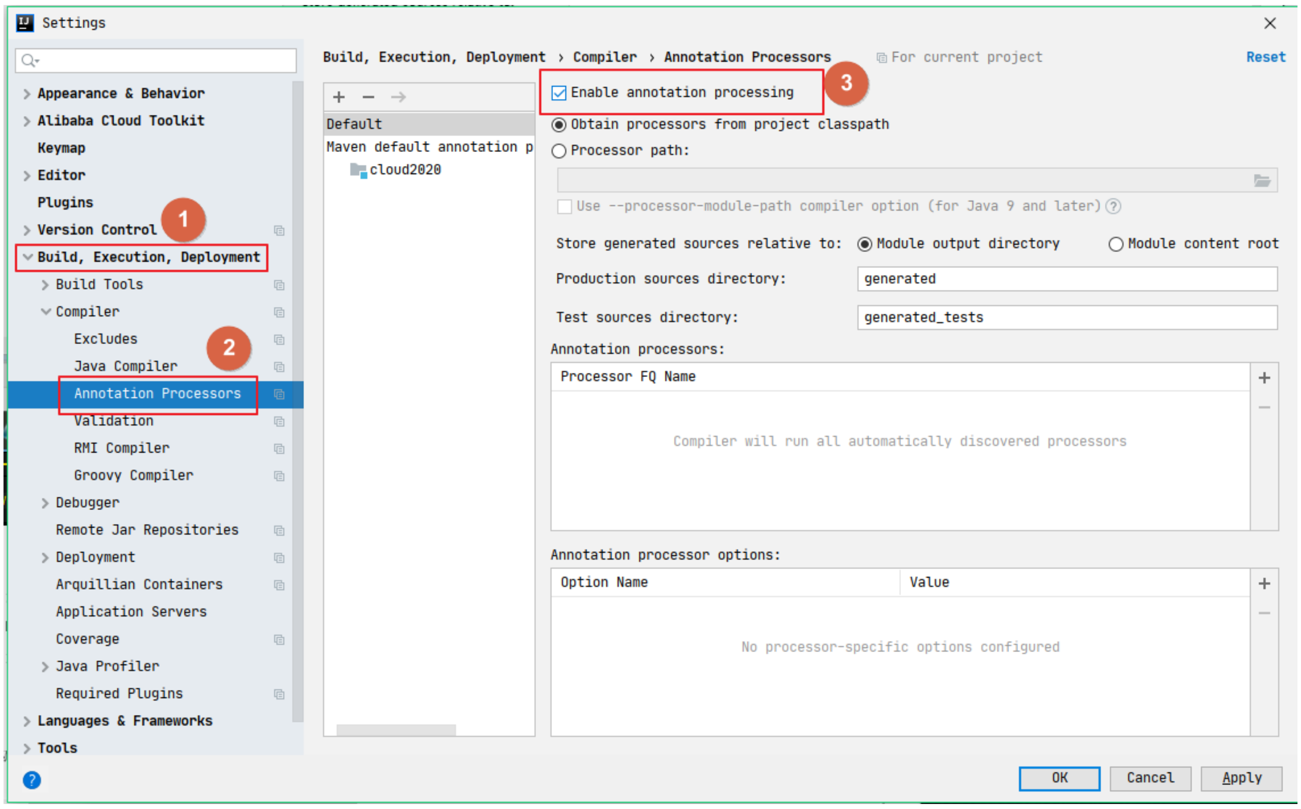
To change encoding IntelliJ IDEA uses for a file, a directory, or the entire project, add its path if necessary and then select encoding from the encoding list. Built-in file encoding (e.g. JSP, HTML or XML) overrides encoding you specify here. If not specified, files and directories inherit encoding settings from the parent directory or from the Project Encoding.

Properties Files (*.properties)
Default encoding for properties files: UTF-8 [✓] Transparent native-to-ascii conversion

BOM for new UTF-8 files
Create UTF-8 files: with NO BOM
IDEA will NOT add UTF-8 BOM to every created file in UTF-8 encoding

2.1.6. 注解生效激活

File -> Settins -> Build, Execution, Deployment -> Compiler -> Annotation Processors -> 勾选 Enable annoatation processing



[✓] Enable annotation processing
● Obtain processors from project classpath
○ Processor path:
☐ Use --processor-module-path compiler option (for Java 9 and later)

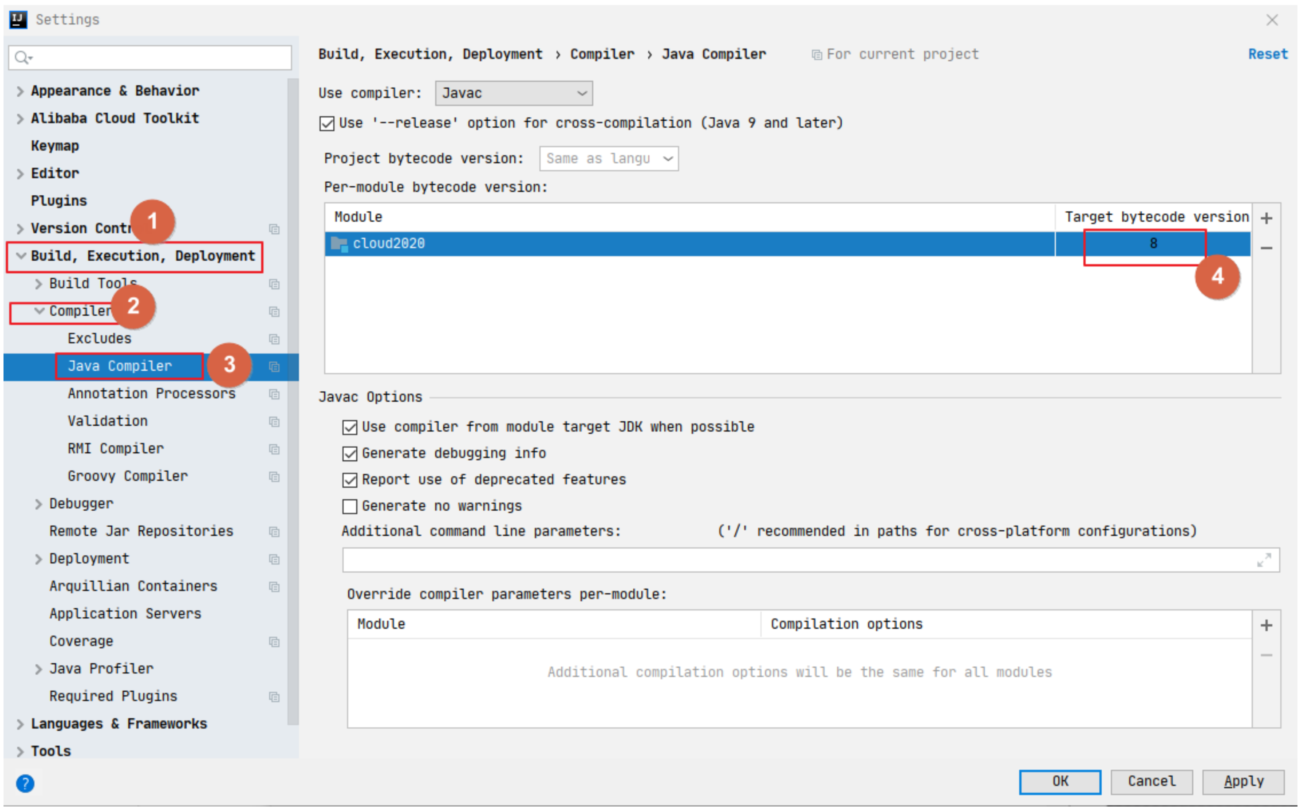
Store generated sources relative to: ● Module output directory ○ Module content root
Production sources directory: generated
Test sources directory: generated_tests

Annotation processors:
Processor FQ Name
Compiler will run all automatically discovered processors

Annotation processor options:
Option Name | Value
No processor-specific options configured

2.1.7. Java编译版本选8

File -> Settins -> Build, Execution, Deployment -> Compiler -> Java Compiler -> Target bytecode version 由 1.5 改为 8


Settings

Appearance & Behavior
Alibaba Cloud Toolkit
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Compiler
Excludes
Java Compiler
Annotation Processors
Validation
RMI Compiler
Groovy Compiler
Debugger
Remote Jar Repositories
Deployment
Arquillian Containers
Application Servers
Coverage
Java Profiler
Required Plugins
Languages & Frameworks
Tools

Build, Execution, Deployment › Compiler › Java Compiler   For current project   Reset

Use compiler: Javac
☑ Use '--release' option for cross-compilation (Java 9 and later)
Project bytecode version: Same as langu
Per-module bytecode version:

Module	Target bytecode version
cloud2020	8

Javac Options
☑ Use compiler from module target JDK when possible
☑ Generate debugging info
☑ Report use of deprecated features
☐ Generate no warnings
Additional command line parameters: ('/' recommended in paths for cross-platform configurations)

Override compiler parameters per-module:

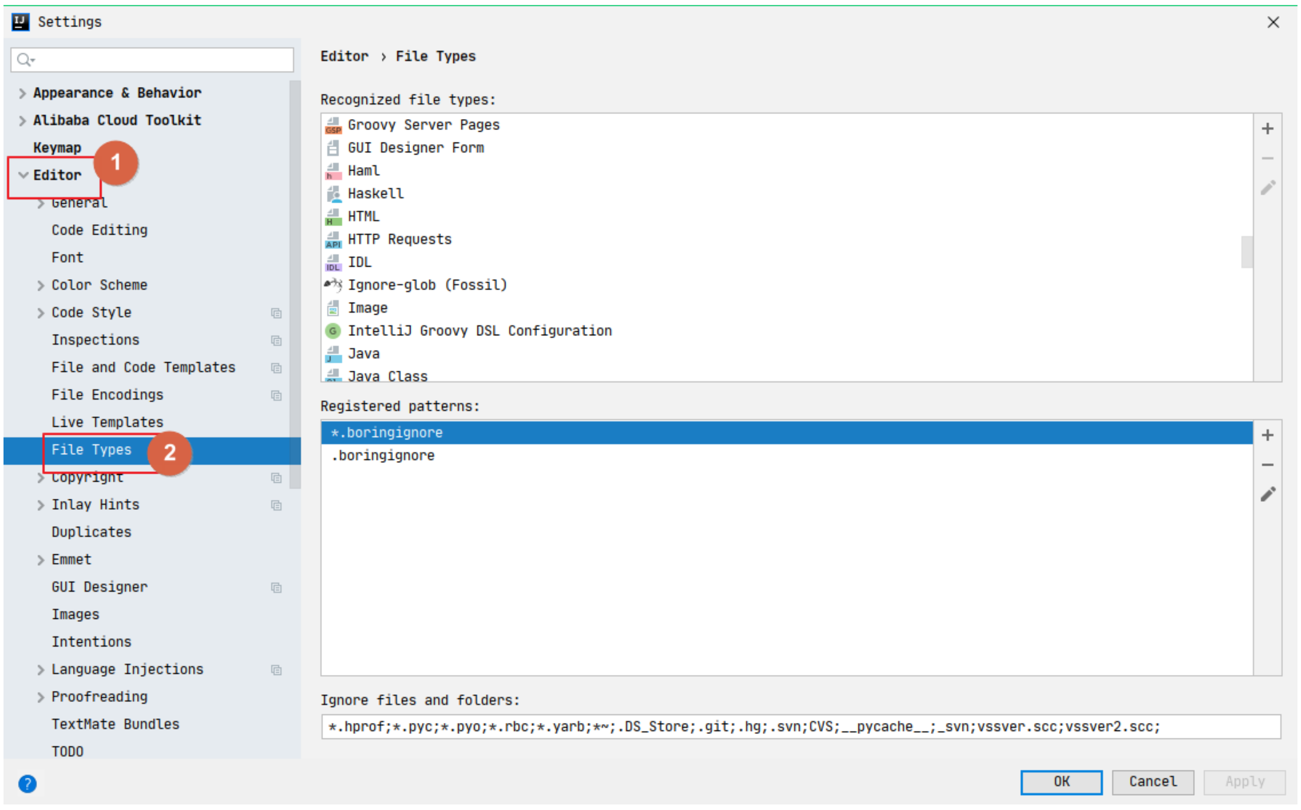
Module	Compilation options

Additional compilation options will be the same for all modules

OK Cancel Apply

2.1.8. File Type 过滤

File -> Settings -> Editor -> File Types


Settings

Appearance & Behavior
Alibaba Cloud Toolkit
Keymap
Editor
General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Inlay Hints
Duplicates
Emmet
GUI Designer
Images
Intentions
Language Injections
Proofreading
TextMate Bundles
TODO

Editor › File Types

Recognized file types:
Groovy Server Pages
GUI Designer Form
Haml
Haskell
HTML
HTTP Requests
IDL
Ignore-glob (Fossil)
Image
IntelliJ Groovy DSL Configuration
Java
Java Class

Registered patterns:
*.boringignore
.boringignore

Ignore files and folders:
.hprof;.pyc;*.pyo;*.rbc;*.yarb;*~;.DS_Store;.git;.hg;.svn;CVS;__pycache__;_svn;vssver.scc;vssver2.scc;

OK Cancel Apply

> 该项依据个人习惯而定, 这里使用默认配置, 不进行任何变更

2.2. 父工程POM

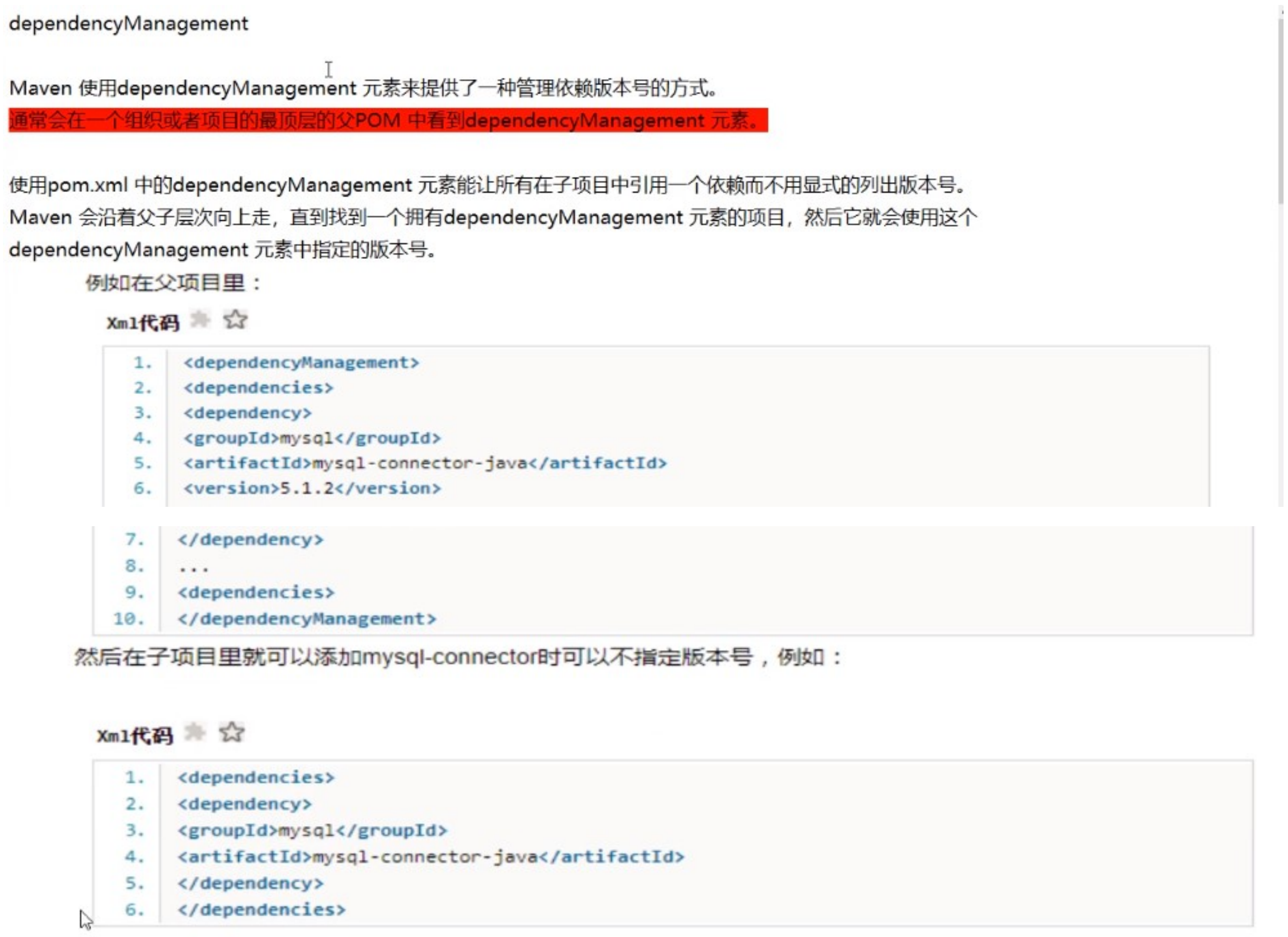
```xml
<?xml version="1.0" encoding="UTF-8"?>

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.sitedev.springcloud</groupId>
    <artifactId>cloud2020</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <modules>
        <module>cloud-provider-payment8001</module>
    </modules>

    <!--统一管理jar包版本-->
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <junit.version>4.12</junit.version>
        <log4j.version>1.2.17</log4j.version>
        <lombok.version>1.16.18</lombok.version>
        <mysql.version>5.1.47</mysql.version>
        <druid.version>1.1.16</druid.version>
        <spring.boot.version>2.2.2.RELEASE</spring.boot.version>
        <spring.cloud.version>Hoxton.SR1</spring.cloud.version>
        <spring.cloud.alibaba.version>2.1.0.RELEASE</spring.cloud.alibaba.version>
        <mybatis.spring.boot.version>1.3.0</mybatis.spring.boot.version>
    </properties>

    <!--子模块继承后,提供作用:锁定版本+子module不用groupId和version-->
    <dependencyManagement>
        <dependencies>
            <!--springboot 2.2.2-->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>${spring.boot.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!--Spring cloud Hoxton.SR1-->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring.cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <!--Spring cloud alibaba 2.1.0.RELEASE-->
            <dependency>
                <groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>
                <version>${spring.cloud.alibaba.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>${log4j.version}</version>
            </dependency>
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>${mysql.version}</version>
            </dependency>
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>druid</artifactId>
                <version>${druid.version}</version>
            </dependency>
            <dependency>
```

```xml
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>${lombok.version}</version>
            </dependency>
            <dependency>
                <groupId>org.mybatis.spring.boot</groupId>
                <artifactId>mybatis-spring-boot-starter</artifactId>
                <version>${mybatis.spring.boot.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>${spring.boot.version}</version>
                <configuration>
                    <fork>true</fork>
                    <addResources>true</addResources>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <!--第三方maven私服-->
    <repositories>
        <repository>
            <id>nexus-aliyun</id>
            <name>Nexus aliyun</name>
            <url>http://maven.aliyun.com/nexus/content/groups/public</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
</project>
```

> 注意事项: 如果遇到某个版本的依赖无法找到的情况, 可以尝试将 `<dependencyManagement>` 和 `</dependencyManagement>` 节点先注释掉, 然后下载依赖.依赖下载成功后, 改回之前的配置

2.3. Maven工程落地细节复习

2.3.1. Maven中的DependencyManagement和Dependencies



dependencyManagement

Maven 使用dependencyManagement 元素来提供了一种管理依赖版本号的方式。
通常会在一个组织或者项目的最顶层的父POM 中看到dependencyManagement 元素。

使用pom.xml 中的dependencyManagement 元素能让所有在子项目中引用一个依赖而不用显式的列出版本号。
Maven 会沿着父子层次向上走，直到找到一个拥有dependencyManagement 元素的项目，然后它就会使用这个
dependencyManagement 元素中指定的版本号。

例如在父项目里：

Xml代码

```xml
<dependencyManagement>
<dependencies>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.2</version>
</dependency>
...
<dependencies>
</dependencyManagement>
```

然后在子项目里就可以添加mysql-connector时可以不指定版本号，例如：

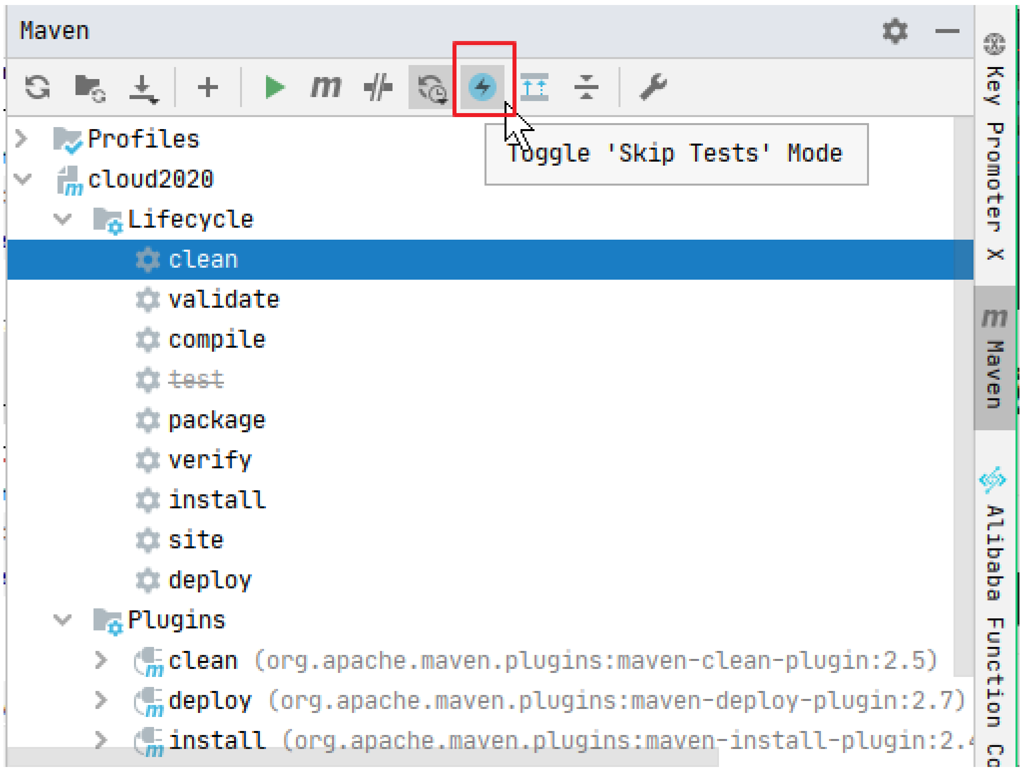
Xml代码

```xml
<dependencies>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
</dependency>
</dependencies>
```

这样做的好处就是: 如果有多个子项目都引用同一样的依赖,则可以避免在每个使用的子项目里都声明一个版本号,这样想升级或切换到另一个版本时,只需在顶层父容器里更新,而不需要一个一个子项目的修改;另外如果某个子项目需要另外的一个版本,只需声明version版本

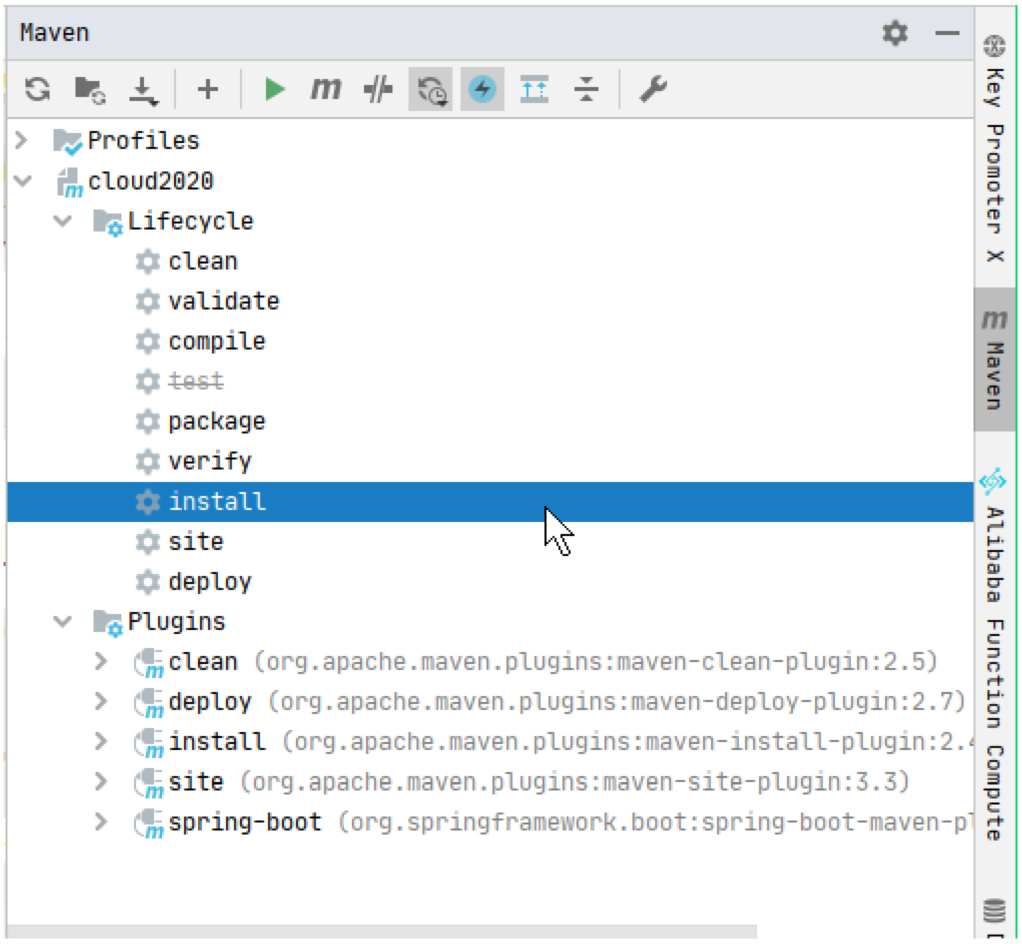
dependencyManagement里只是声明依赖，并不实现引入，因此子项目需要显式的声明需要用的依赖。

2.3.2. maven中跳过单元测试



```
Profiles
cloud2020
  Lifecycle
    clean
    validate
    compile
    ~~test~~
    package
    verify
    install
    site
    deploy
  Plugins
    clean (org.apache.maven.plugins:maven-clean-plugin:2.5)
    deploy (org.apache.maven.plugins:maven-deploy-plugin:2.7)
    install (org.apache.maven.plugins:maven-install-plugin:2.
```

2.4. 父工程创建完成执行mvn:install将父工程发布到仓库方便子工程继承



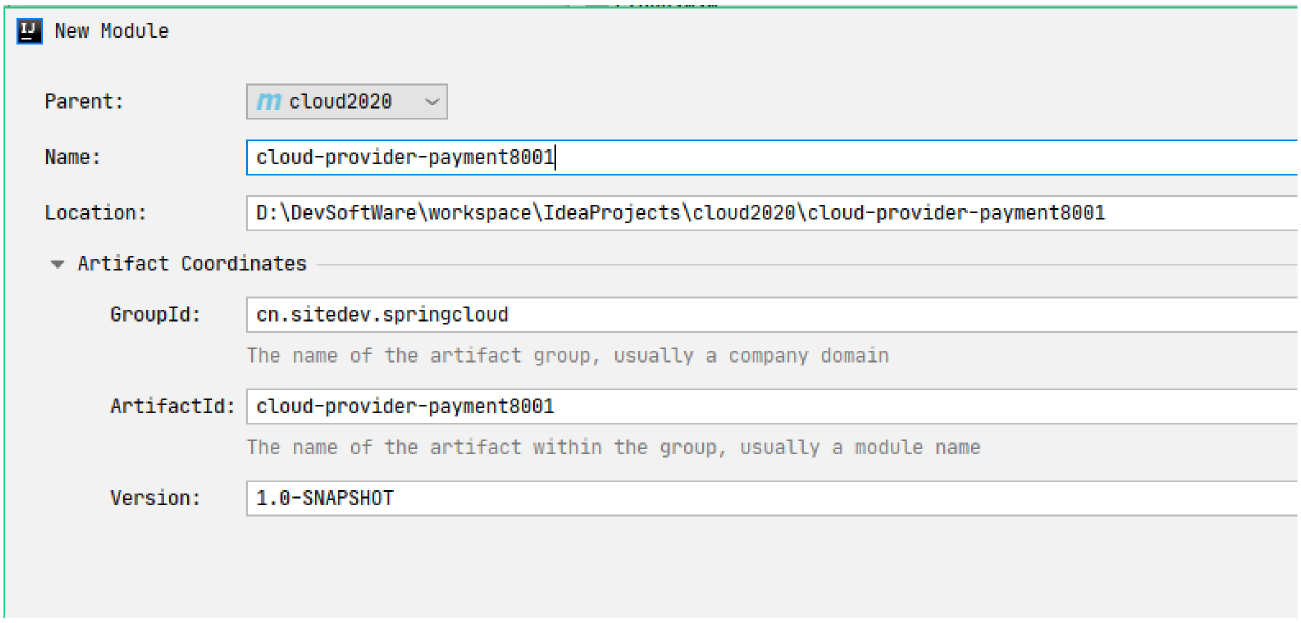
```
Profiles
cloud2020
  Lifecycle
    clean
    validate
    compile
    ~~test~~
    package
    verify
    install
    site
    deploy
  Plugins
    clean (org.apache.maven.plugins:maven-clean-plugin:2.5)
    deploy (org.apache.maven.plugins:maven-deploy-plugin:2.7)
    install (org.apache.maven.plugins:maven-install-plugin:2.
    site (org.apache.maven.plugins:maven-site-plugin:3.3)
    spring-boot (org.springframework.boot:spring-boot-maven-pl
```

3. Rest微服务工程搭建

3.1. 搭建步骤

3.1.1. cloud-provider-payment8001微服务提供者Module模块

3.1.1.1. 建Module



New Module

Parent: cloud2020

Name: cloud-provider-payment8001

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-provider-payment8001

Artifact Coordinates

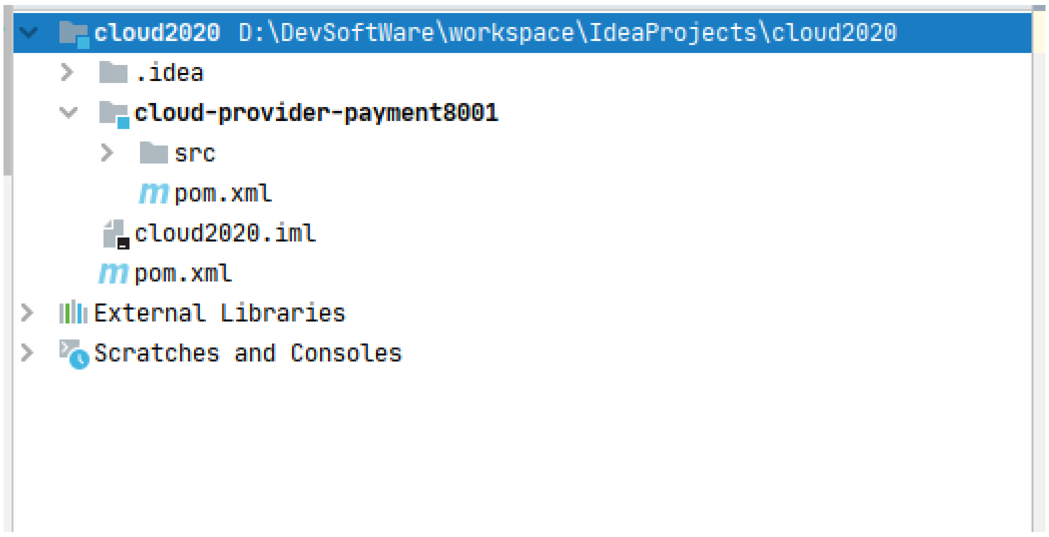
GroupId: cn.sitedev.springcloud

The name of the artifact group, usually a company domain

ArtifactId: cloud-provider-payment8001

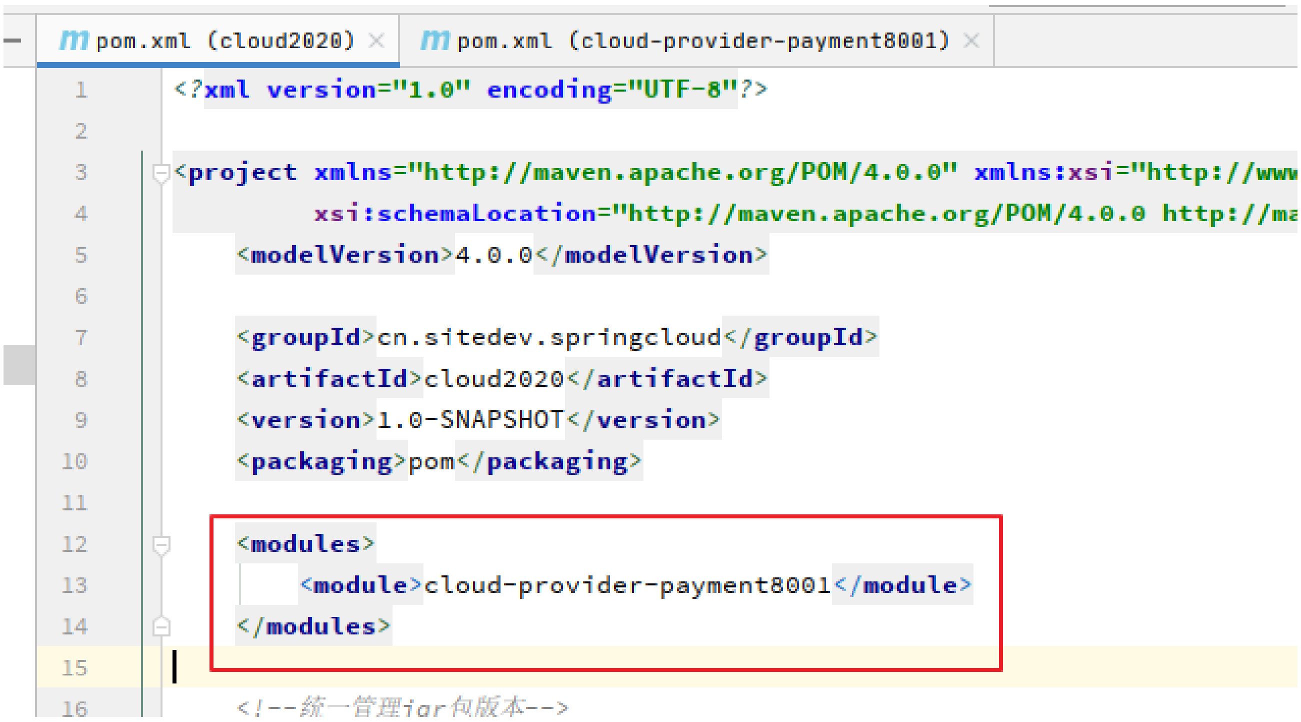
The name of the artifact within the group, usually a module name

Version: 1.0-SNAPSHOT



```
cloud2020  D:\DevSoftWare\workspace\IdeaProjects\cloud2020
  .idea
  cloud-provider-payment8001
    src
    pom.xml
  cloud2020.iml
  pom.xml
External Libraries
Scratches and Consoles
```

创建完成后回到父工程查看pom文件变化



pom.xml (cloud2020) × | pom.xml (cloud-provider-payment8001) ×

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://ma
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.sitedev.springcloud</groupId>
    <artifactId>cloud2020</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <modules>
        <module>cloud-provider-payment8001</module>
    </modules>

    <!--统一管理jar包版本-->
```

3.1.1.2. 改POM

```
<?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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-provider-payment8001</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

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

```xml

        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid-spring-boot-starter</artifactId>
            <version>1.1.10</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>

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

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </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>

    </dependencies>

</project>
```

3.1.1.3. 写YML

```yaml
server:
  port: 8001

spring:
  application:
    name: cloud-payment-service
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource # 当前数据源操作类型
    driver-class-name: org.gjt.mm.mysql.Driver # mysql驱动包
    url: jdbc:mysql://localhost:3306/db2019?useUnicode=true&characterEncoding=utf-8&useSSL=false
    username: root
    password: root

mybatis:
  mapperLocations: classpath:mapper/*.xml
  type-aliases-package: cn.sitedev.springcloud.entities # 所有Entity别名类所在包
```

3.1.1.4. 主启动

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PaymentMain8001 {
    public static void main(String[] args) {
        SpringApplication.run(PaymentMain8001.class, args);
    }
}
```

3.1.1.5. 业务类

3.1.1.5.1. 建表SQL

```
CREATE TABLE `payment`  (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键',
  `serial` varchar(200) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT '支付流水号',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci COMMENT = '支付表' ROW_FORMAT = Dynamic;
```

3.1.1.5.2. entities

主实体Payment

```java
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Payment implements Serializable {
    private Long id;
    private String serial;
}
```

Json封装体CommonResult

```java
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CommonResult<T> {
    private Integer code;
    private String message;
    private T data;

    public CommonResult(Integer code, String message) {
        this(code, message, null);
    }
}
```

3.1.1.5.3. dao

接口PaymentDao

```java
package cn.sitedev.springcloud.dao;

import cn.sitedev.springcloud.entities.Payment;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Param;

@Mapper
public interface PaymentDao {
    int create(Payment payment);

    Payment getPaymentById(@Param("id") Long id);
}
```

mybatis的映射文件PaymentMapper.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="cn.sitedev.springcloud.dao.PaymentDao">

    <insert id="create" parameterType="Payment" useGeneratedKeys="true"
keyProperty="id">
        INSERT INTO payment(serial) VALUES (#{serial});
    </insert>

    <select id="getPaymentById" parameterType="Long" resultMap="BaseResultMap">
        SELECT * FROM payment WHERE id = #{id};
    </select>

    <resultMap id="BaseResultMap" type="Payment">
        <id column="id" property="id" jdbcType="BIGINT"/>
        <id column="serial" property="serial" jdbcType="VARCHAR"/>
    </resultMap>
</mapper>
```

3.1.1.5.4. service

接口PaymentService

```java
package cn.sitedev.springcloud.service;

import cn.sitedev.springcloud.entities.Payment;

public interface PaymentService {
    int create(Payment payment);

    Payment getPaymentById(Long id);
}
```

实现类

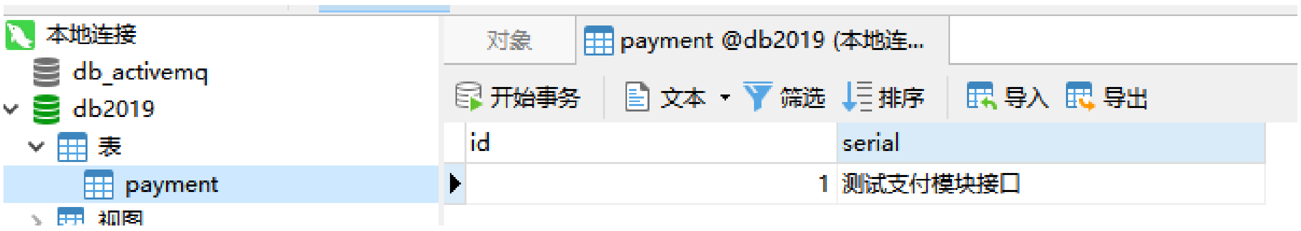
```java
package cn.sitedev.springcloud.service;

import cn.sitedev.springcloud.dao.PaymentDao;
import cn.sitedev.springcloud.entities.Payment;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;

@Service
public class PaymentServiceImpl implements PaymentService {
    @Resource
    private PaymentDao paymentDao;

    @Override
    public int create(Payment payment) {
        return paymentDao.create(payment);
    }

    @Override
    public Payment getPaymentById(Long id) {
        return paymentDao.getPaymentById(id);
    }
}
```

3.1.1.5.5. controller

```java

package cn.sitedev.springcloud.controller;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import cn.sitedev.springcloud.service.PaymentService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;

@RestController
@Slf4j
@RequestMapping("/payment")
public class PaymentController {
    @Resource
    private PaymentService paymentService;

    @PostMapping(value = "/create")
    public CommonResult create(Payment payment) {
        int result = paymentService.create(payment);
        log.info("*****插入结果: " + result);
        if (result > 0) {
            return new CommonResult(200, "插入数据库成功", result);
        } else {
            return new CommonResult(444, "插入数据库失败", null);
        }
    }

    @GetMapping(value = "/get/{id}")
    public CommonResult getPaymentById(@PathVariable("id") Long id) {
        Payment payment = paymentService.getPaymentById(id);
        log.info("*****查询结果: " + payment);
        if (payment != null) {
            return new CommonResult(200, "查询成功", payment);
        } else {
            return new CommonResult(444, "没有对应记录, 查询id: " + id, null);
        }
    }
}
```

3.1.1.6. 测试

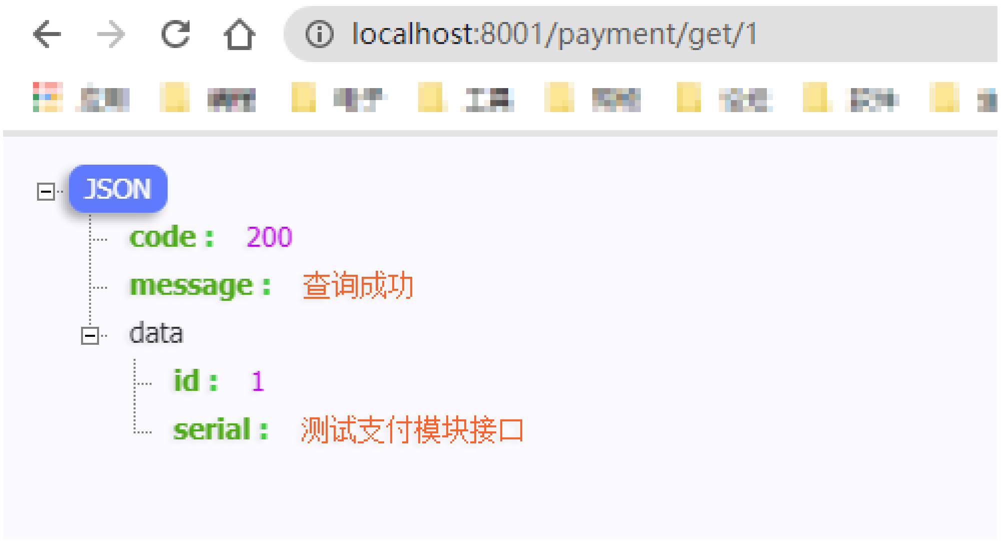
数据库中插入一条测试数据



id	serial
1	测试支付模块接口

3.1.1.6.1. 测试查询接口

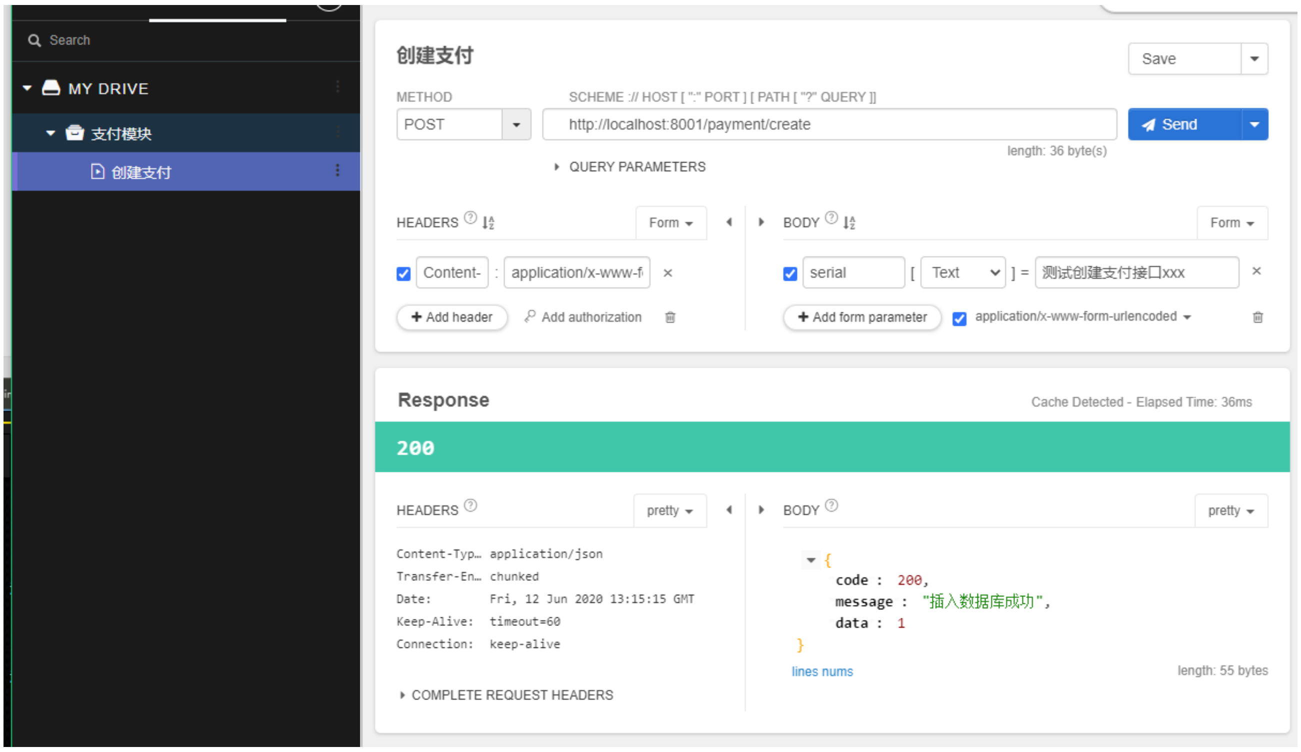
启动微服务, 浏览器访问http://localhost:8001/payment/get/1



```
code : 200
message : 查询成功
data
    id : 1
    serial : 测试支付模块接口
```

3.1.1.6.2. 测试创建接口

使用PostMan等工具(这里使用Chrome浏览器插件 Talend API Tester)模拟POST请求接口 http://localhost:8001/payment/create



创建支付

METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8001/payment/create
length: 36 byte(s)

HEADERS: Content- : application/x-www-f
BODY: serial [Text] = 测试创建支付接口xxx
application/x-www-form-urlencoded

Response
Cache Detected - Elapsed Time: 36ms

200

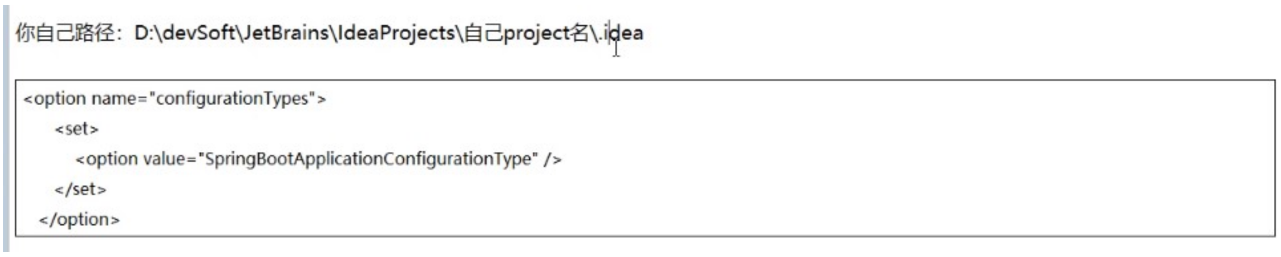
```
Content-Typ…  application/json
Transfer-En…  chunked
Date:         Fri, 12 Jun 2020 13:15:15 GMT
Keep-Alive:   timeout=60
Connection:   keep-alive
```

```
{
    code :  200,
    message :  "插入数据库成功",
    data :  1
}
```

length: 55 bytes

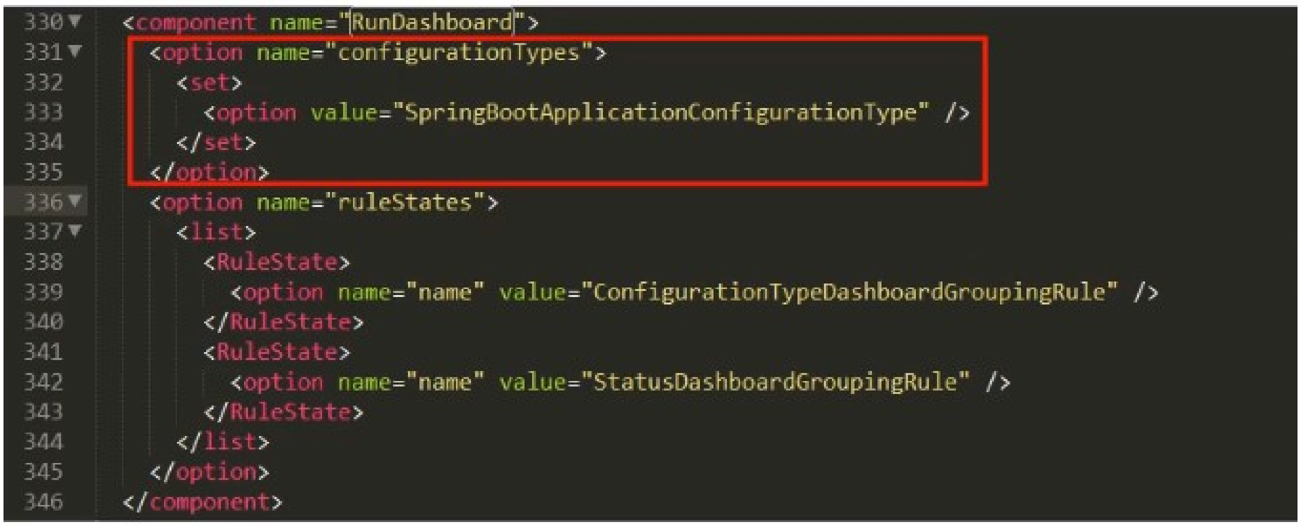
3.1.1.6.3. 运行

开启Run DashBoard(旧版IDEA)(通过修改idea的workspace.xml的方式快速打开Run Dashboard窗口)

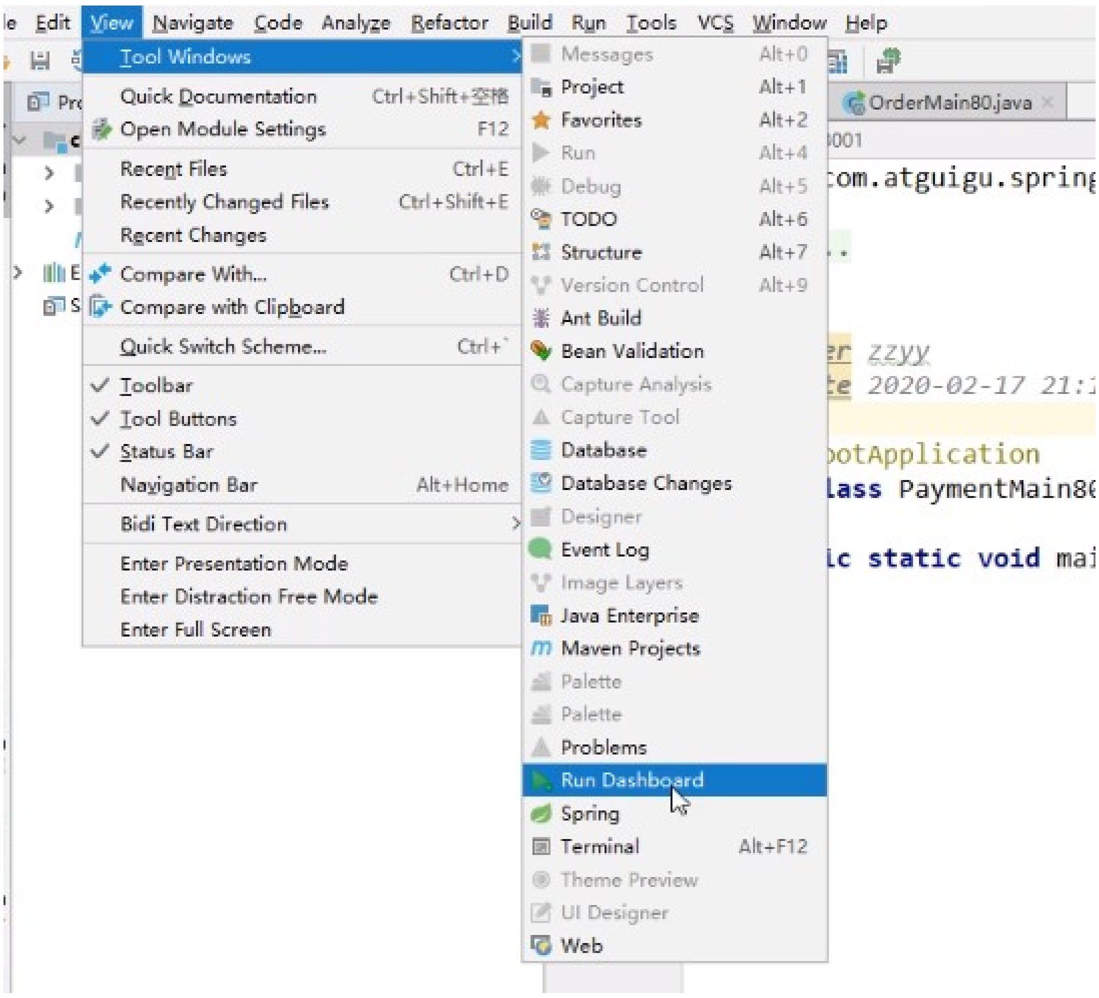

你自己路径：D:\devSoft\JetBrains\IdeaProjects\自己project名\.idea

```
<option name="configurationTypes">
    <set>
      <option value="SpringBootApplicationConfigurationType" />
    </set>
  </option>
```

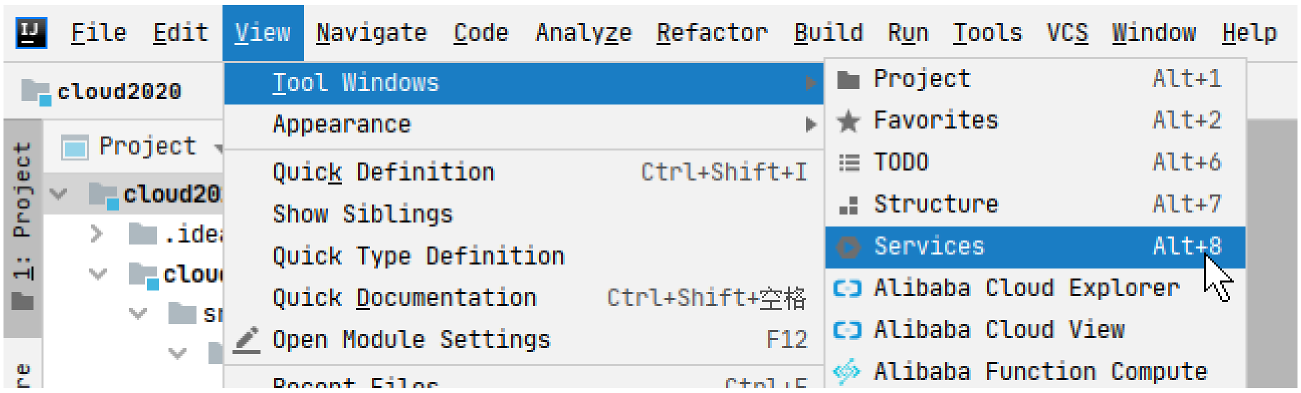
```
<component name="RunDashboard">
    <option name="configurationTypes">
      <set>
        <option value="SpringBootApplicationConfigurationType" />
      </set>
    </option>
  </component>
```

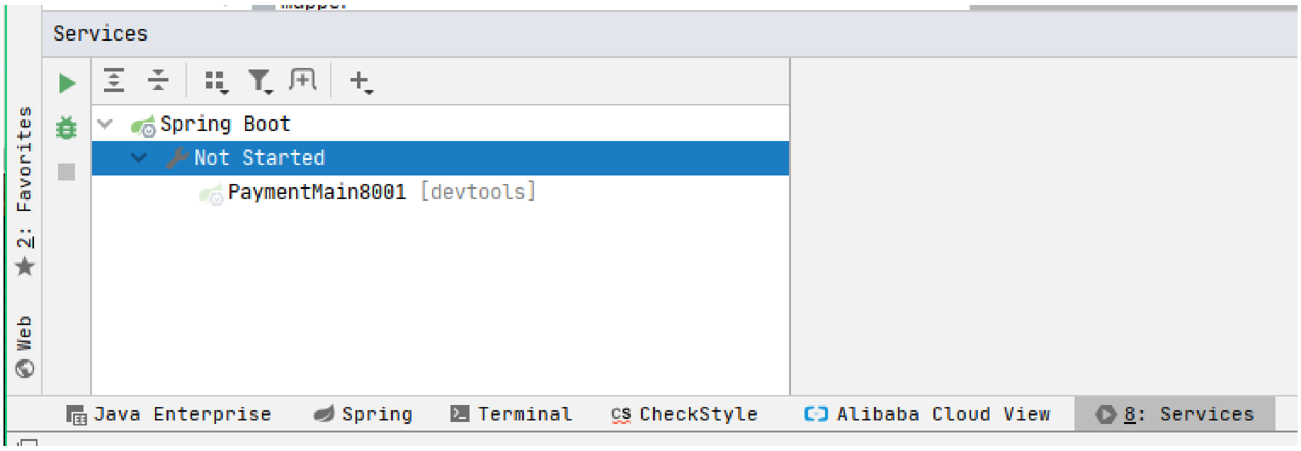


```
<component name="RunDashboard">
  <option name="configurationTypes">
    <set>
      <option value="SpringBootApplicationConfigurationType" />
    </set>
  </option>
  <option name="ruleStates">
    <list>
      <RuleState>
        <option name="name" value="ConfigurationTypeDashboardGroupingRule" />
      </RuleState>
      <RuleState>
        <option name="name" value="StatusDashboardGroupingRule" />
      </RuleState>
    </list>
  </option>
</component>
```



开启Run DashBoard(新版IDEA 2020.1.2)





3.1.1.7. 小总结

创建支付模块的步骤总结:

- 建module
- 改POM
- 写YML
- 主启动
- 业务类

3.1.2. 热部署Devtools

3.1.2.1. 添加devtools依赖

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
```

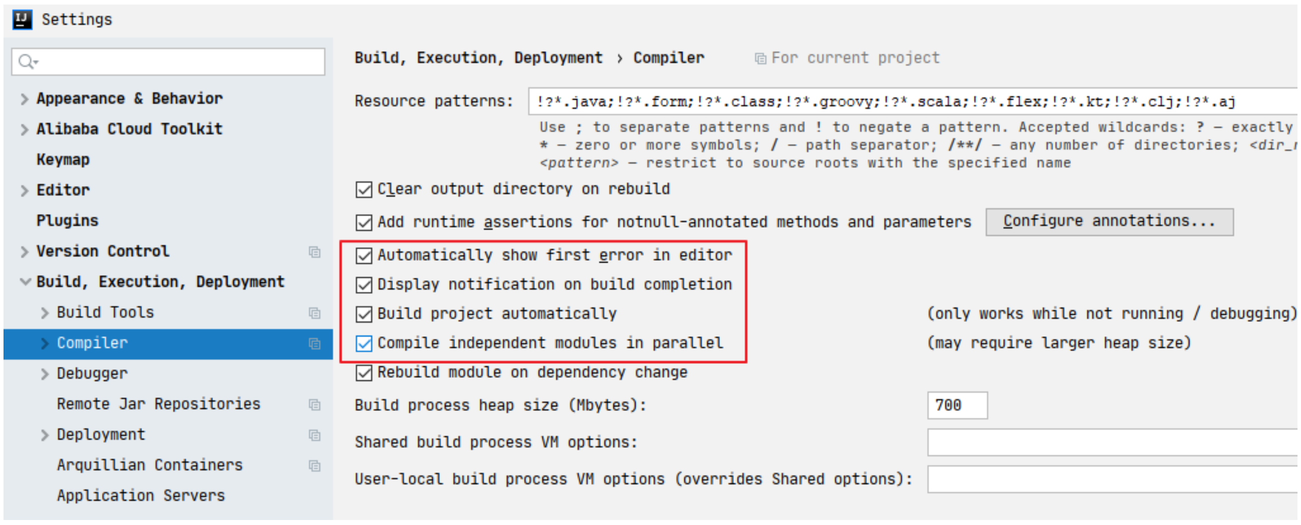
3.1.2.2. 添加插件配置

下面配置我们粘贴进聚合父类总工程的pom.xml里

```xml
<build>
    <fileName>你自己的工程名字<fileName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <fork>true</fork>
                <addResources>true</addResources>
            </configuration>
        </plugin>
    </plugins>
</build>
```

3.1.2.3. 开启自动编译

Settings -> Build,Execution, Deployment -> Compiler


Settings

Build, Execution, Deployment › Compiler   For current project

Resource patterns: !?*.java;!?*.form;!?*.class;!?*.groovy;!?*.scala;!?*.flex;!?*.kt;!?*.clj;!?*.aj

Use ; to separate patterns and ! to negate a pattern. Accepted wildcards: ? – exactly
* – zero or more symbols; / – path separator; /**/ – any number of directories; <dir_
<pattern> – restrict to source roots with the specified name

- [✓] Clear output directory on rebuild
- [✓] Add runtime assertions for notnull-annotated methods and parameters   Configure annotations...
- [✓] Automatically show first error in editor
- [✓] Display notification on build completion
- [✓] Build project automatically   (only works while not running / debugging)
- [✓] Compile independent modules in parallel   (may require larger heap size)
- [✓] Rebuild module on dependency change

Build process heap size (Mbytes): 700

Shared build process VM options:

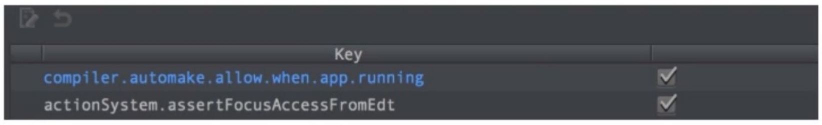
User-local build process VM options (overrides Shared options):

3.1.2.4. 更新值

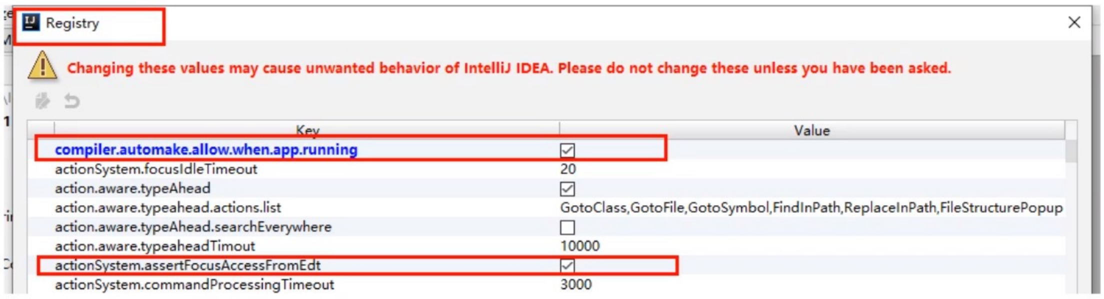
简要说明:

press `ctrl+shift+Alt+/` and search for the registry. In the `Registry`, enable :

- ☑ compiler.automake.allow.when.app.running



Key	
compiler.automake.allow.when.app.running	☑
actionSystem.assertFocusAccessFromEdt	☑



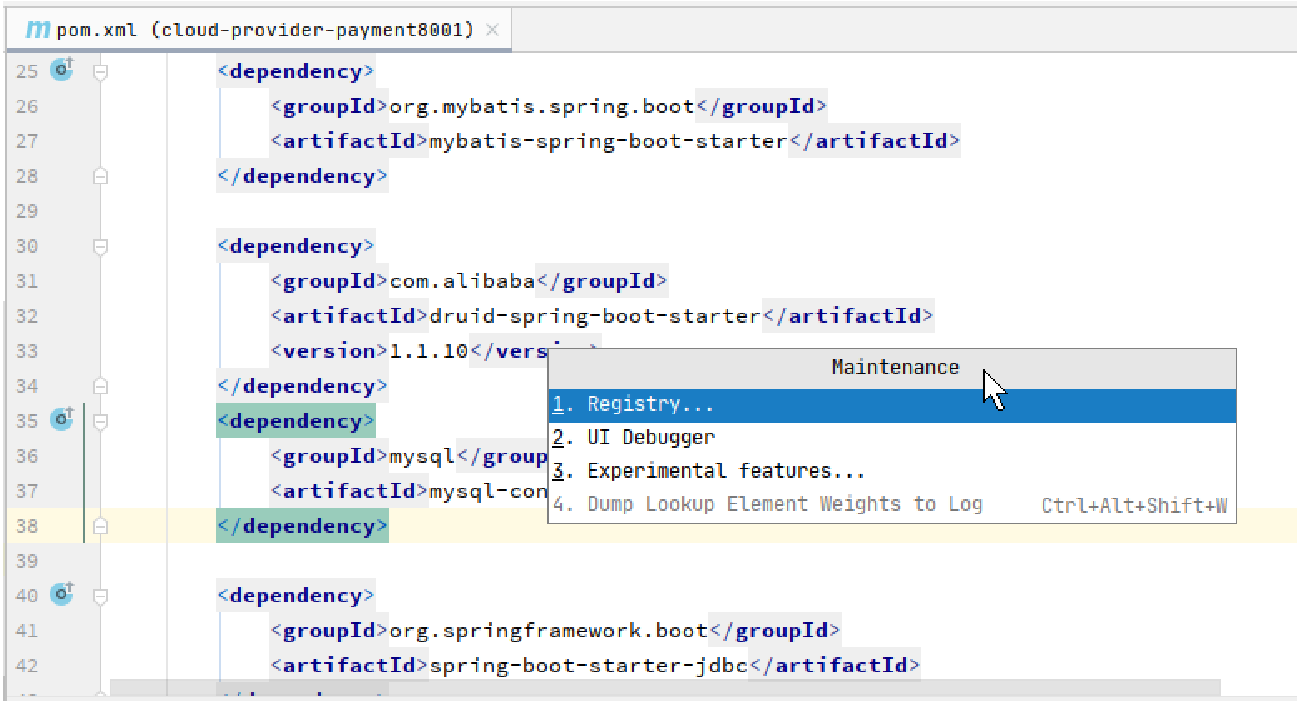
Registry

Changing these values may cause unwanted behavior of IntelliJ IDEA. Please do not change these unless you have been asked.

Key	Value
compiler.automake.allow.when.app.running	☑
actionSystem.focusIdleTimeout	20
action.aware.typeAhead	☑
action.aware.typeahead.actions.list	GotoClass,GotoFile,GotoSymbol,FindInPath,ReplaceInPath,FileStructurePopup
action.aware.typeAhead.searchEverywhere	☐
action.aware.typeaheadTimout	10000
actionSystem.assertFocusAccessFromEdt	☑
actionSystem.commandProcessingTimeout	3000

Ctrl+Shift+Alt+/ -> 1. Registry... -> 勾选下述选项:

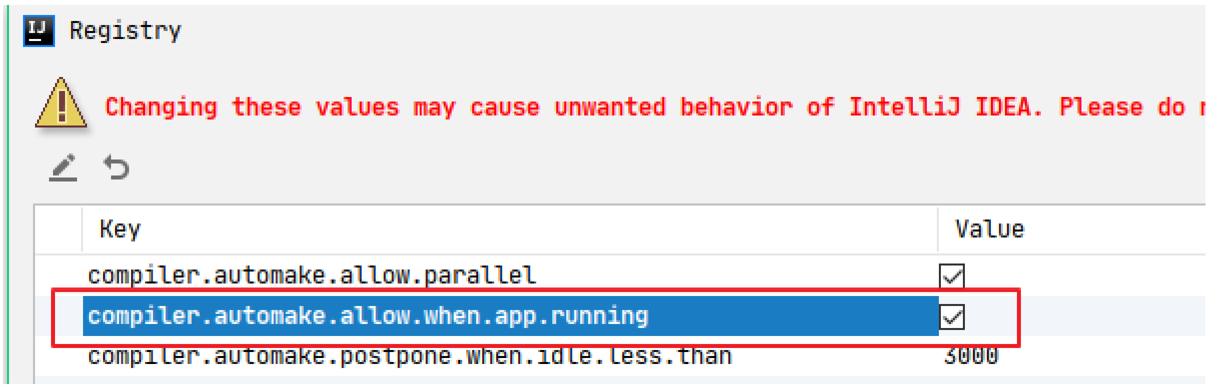
- compiler.automake.allow.when.app.running
- actionSystem.assertFocusAccessFromEdt



pom.xml (cloud-provider-payment8001)

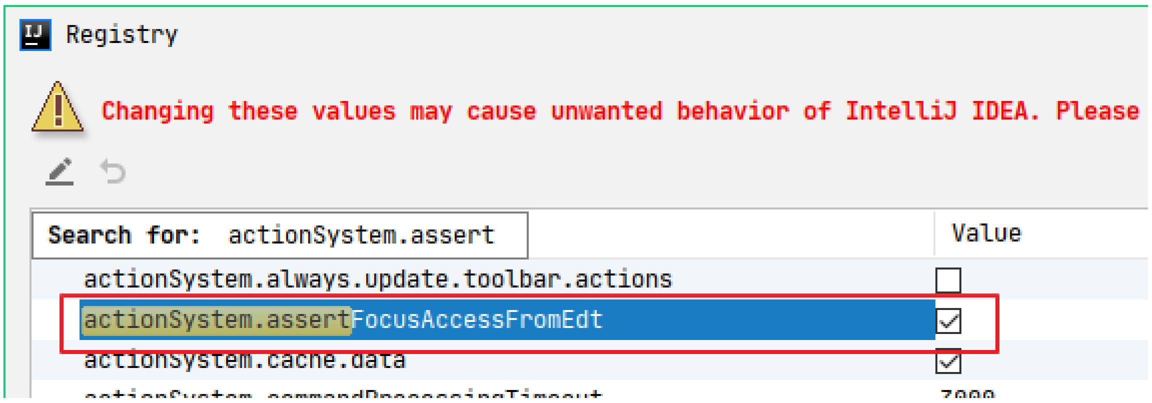
```
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid-spring-boot-starter</artifactId>
            <version>1.1.10</vers
        </dependency>
        <dependency>
            <groupId>mysql</group
            <artifactId>mysql-con
        </dependency>

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

Maintenance
1. Registry...
2. UI Debugger
3. Experimental features...
4. Dump Lookup Element Weights to Log   Ctrl+Alt+Shift+W


Registry

Changing these values may cause unwanted behavior of IntelliJ IDEA. Please do

Key	Value
compiler.automake.allow.parallel	✓
compiler.automake.allow.when.app.running	✓
compiler.automake.postpone.when.idle.less.than	3000


Registry

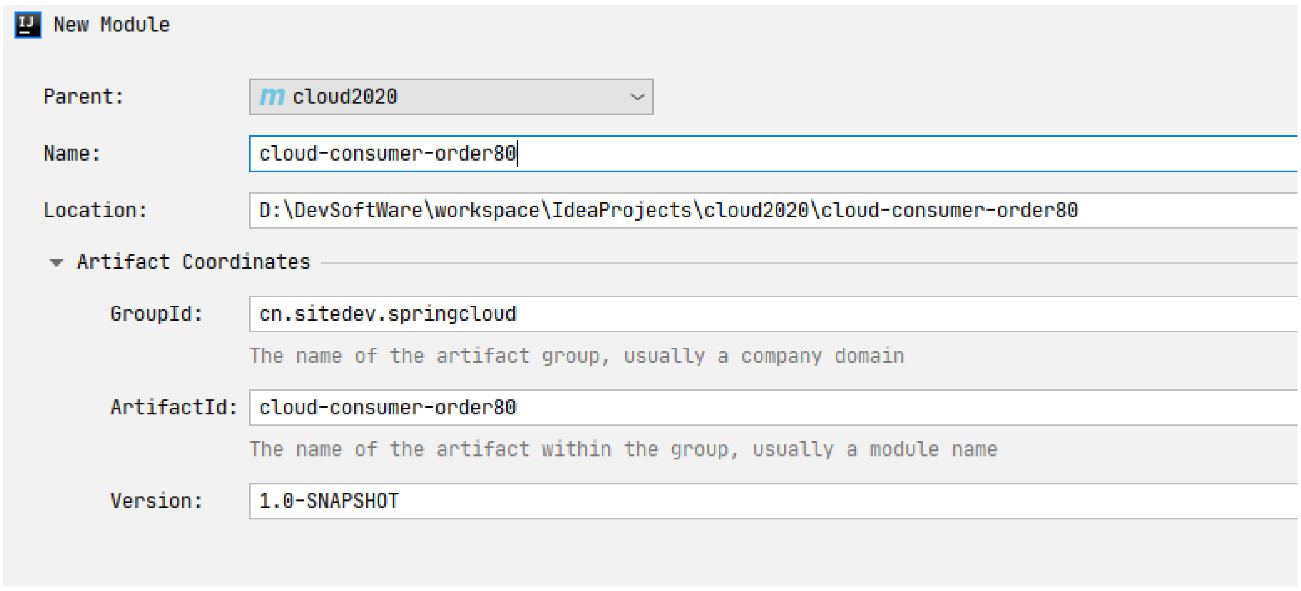
Changing these values may cause unwanted behavior of IntelliJ IDEA. Please

Search for: actionSystem.assert	Value
actionSystem.always.update.toolbar.actions	☐
actionSystem.assertFocusAccessFromEdt	✓
actionSystem.cache.data	✓

3.1.2.5. 重启IDEA

3.1.3. cloud-consumer-order80微服务消费者订单Module模块

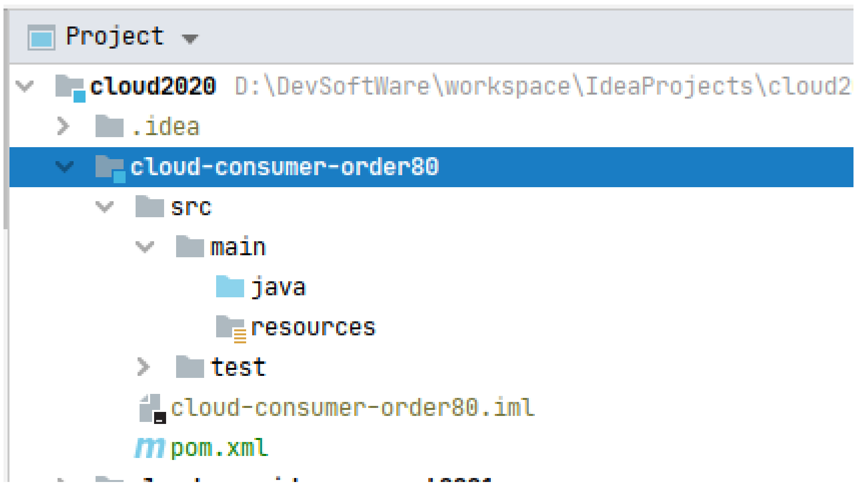
3.1.3.1. 建Module


New Module

Parent:	cloud2020
Name:	cloud-consumer-order80
Location:	D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-consumer-order80

Artifact Coordinates

GroupId:	cn.sitedev.springcloud
	The name of the artifact group, usually a company domain
ArtifactId:	cloud-consumer-order80
	The name of the artifact within the group, usually a module name
Version:	1.0-SNAPSHOT


Project
cloud2020 D:\DevSoftWare\workspace\IdeaProjects\cloud2
.idea
cloud-consumer-order80
src
main
java
resources
test
cloud-consumer-order80.iml
pom.xml

3.1.3.2. 改POM

```xml
<?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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-consumer-order80</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

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

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </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>

    </dependencies>
</project>
```

3.1.3.3. 写YML

```yaml
server:
  port: 80
```

3.1.3.4. 主启动

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OrderMain80 {
    public static void main(String[] args) {
        SpringApplication.run(OrderMain80.class, args);
    }
}
```

3.1.3.5. 业务类

3.1.3.5.1. entities

```java
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CommonResult<T> {
    private Integer code;
    private String message;
    private T data;

    public CommonResult(Integer code, String message) {
        this(code, message, null);
    }
}
////////////////////////////////
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Payment implements Serializable {
    private Long id;
    private String serial;
}
```

3.1.3.5.2. 首说RestTemplate

- 是什么

> RestTemplate提供了多种便捷访问远程Http服务的方法，是一种简单便捷的访问restful服务模板类，是Spring提供的用于访问Rest服务的客户端模板工具集

- 官方使用

  > https://docs.spring.io/spring-framework/docs/5.2.2.RELEASE/javadoc-api/org/springframework/web/client/RestTemplate.html
  >
  >
  > 官网地址
  >
  > https://docs.spring.io/spring-framework/docs/5.2.2.RELEASE/javadoc-api/org/springframework/web/client/RestTemplate.html
  >
  > 使用
  >
  > 使用restTemplate访问restful接口非常的简单粗暴无脑。
  > (url, requestMap, ResponseBean.class)这三个参数分别代表
  > REST请求地址、请求参数、HTTP响应转换被转换成的对象类型。

3.1.3.5.3. config配置类

```java
package cn.sitedev.springcloud.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class ApplicationContextConfig {
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

3.1.3.5.4. controller

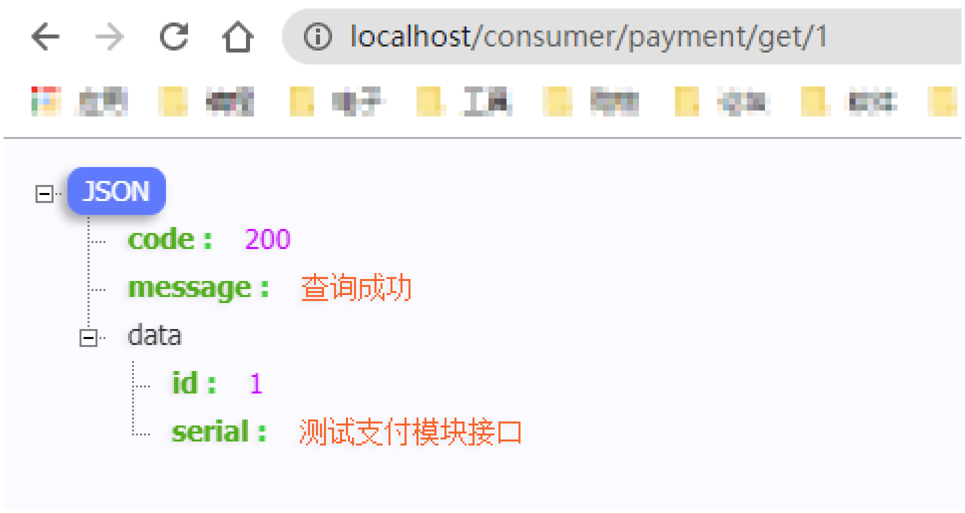
```java
package cn.sitedev.springcloud.controller;

import cn.sitedev.springcloud.entities.CommonResult;
import cn.sitedev.springcloud.entities.Payment;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

import javax.annotation.Resource;
```

```java

@RestController
@RequestMapping("/consumer/payment")
@Slf4j
public class OrderController {

    public static final String PAYMENT_URL = "http://localhost:8001";

    @Resource
    private RestTemplate restTemplate;

    @PostMapping(value = "/create")
    public CommonResult<Payment> create(Payment payment) {
        return restTemplate.postForObject(PAYMENT_URL + "/payment/create", payment, CommonResult.class);
    }

    @GetMapping(value = "/get/{id}")
    public CommonResult<Payment> getPayment(@PathVariable("id") Long id) {
        return restTemplate.getForObject(PAYMENT_URL + "/payment/get/" + id, CommonResult.class);
    }
}
```

3.1.3.6. 测试

3.1.3.6.1. 测试查询接口

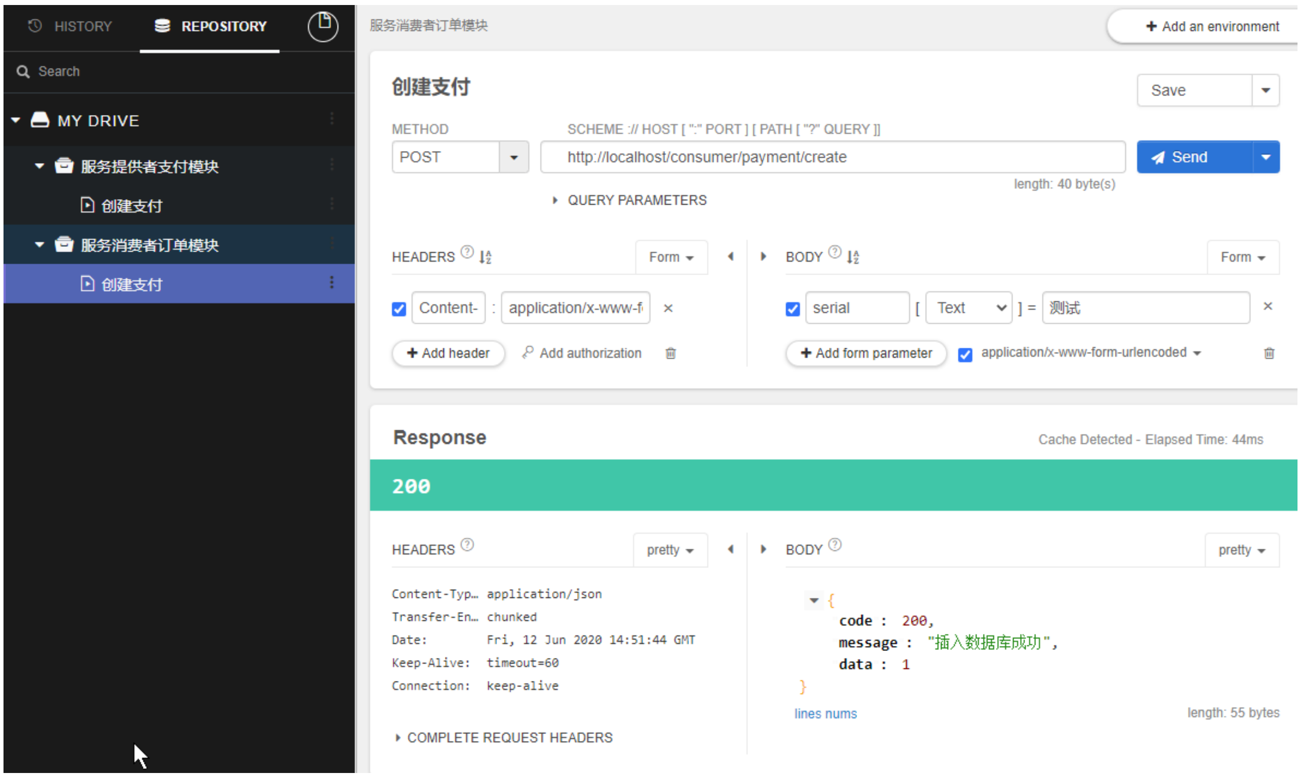
启动服务, 浏览器访问http://localhost/consumer/payment/get/1



```
code : 200
message : 查询成功
data
    id : 1
    serial : 测试支付模块接口
```

3.1.3.6.2. 测试创建接口

使用插件向http://localhost/consumer/payment/create发起POST请求



METHOD: POST
URL: http://localhost/consumer/payment/create
length: 40 byte(s)

HEADERS: Content- : application/x-www-f
BODY: serial [Text] = 测试
application/x-www-form-urlencoded

Response — Cache Detected - Elapsed Time: 44ms

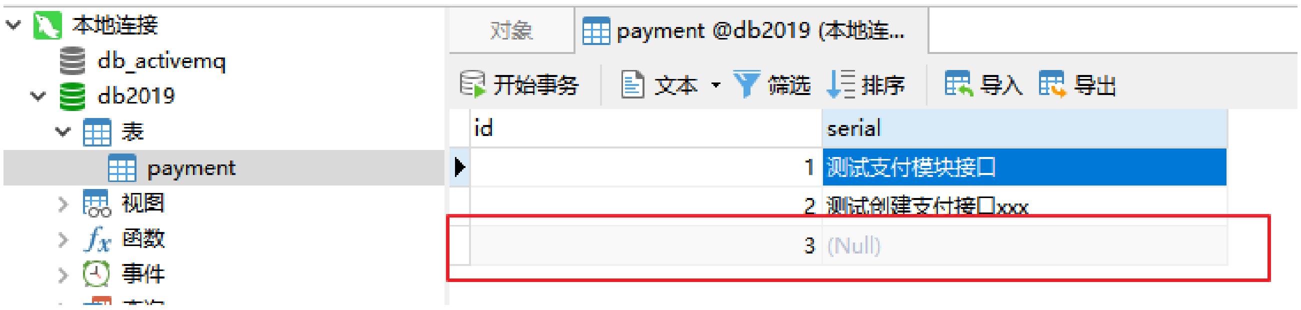
200

```
Content-Typ… application/json
Transfer-En… chunked
Date:        Fri, 12 Jun 2020 14:51:44 GMT
Keep-Alive:  timeout=60
Connection:  keep-alive
```

```
{
   code : 200,
   message : "插入数据库成功",
   data : 1
}
```

length: 55 bytes

虽然表面上看起来, 接口已调用成功, 数据应该成功插入数据库, 但是我们通过查看数据库记录, 可以发现, 新增的记录中serial字段没有值



id	serial
1	测试支付模块接口
2	测试创建支付接口xxx
3	(Null)

因此, 我们需要对代码进行一些修改

3.1.3.6.3. 修改cloud-provider-payment8001模块的PaymentController

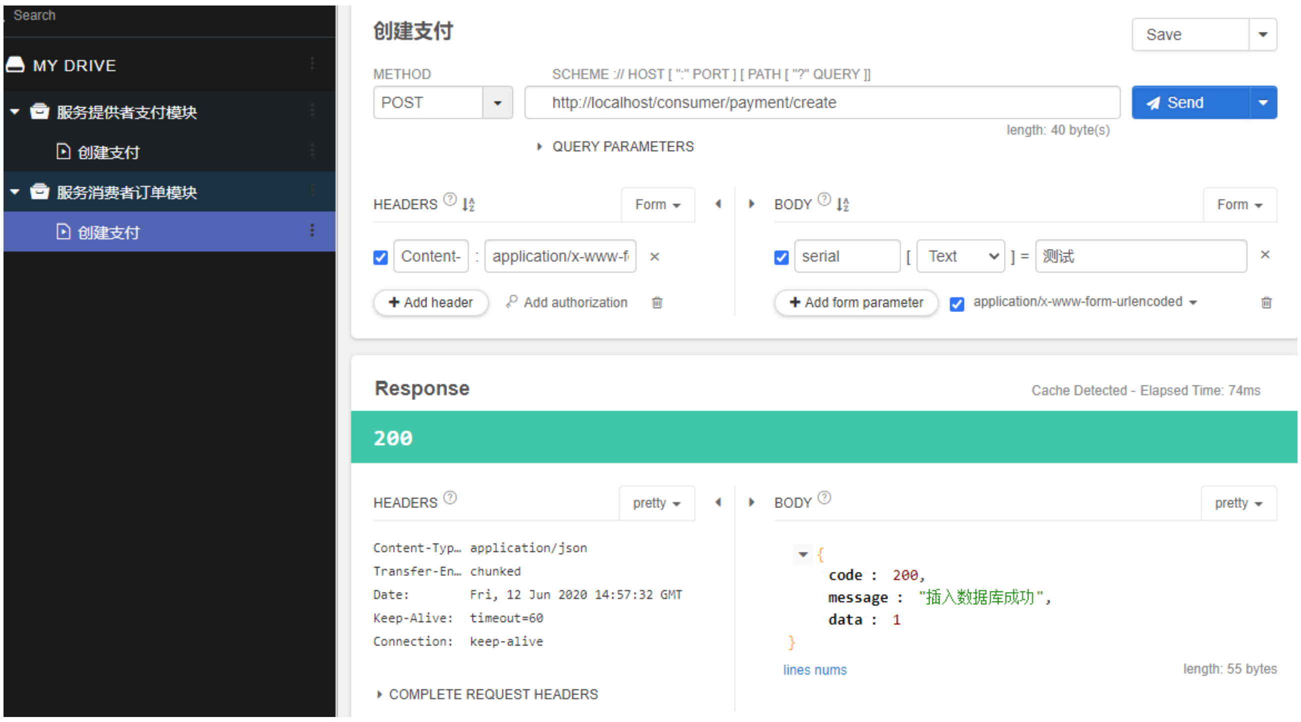
给create方法的payment入参添加@RequestBody注解

```java
@RestController
@Slf4j
@RequestMapping("/payment")
public class PaymentController {
    @Resource
    private PaymentService paymentService;

    @PostMapping(value = "/create")
    public CommonResult create(@RequestBody Payment payment) {
        ...
    }
```

修改完毕后, 重启cloud-provider-payment8001模块对应服务

3.1.3.6.3. 再次测试创建接口

使用插件向http://localhost/consumer/payment/create发起POST请求



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost/consumer/payment/create
length: 40 byte(s)

HEADERS (Form): Content- : application/x-www-f

BODY (Form): serial [Text] = 测试 ; application/x-www-form-urlencoded

Response — Cache Detected - Elapsed Time: 74ms

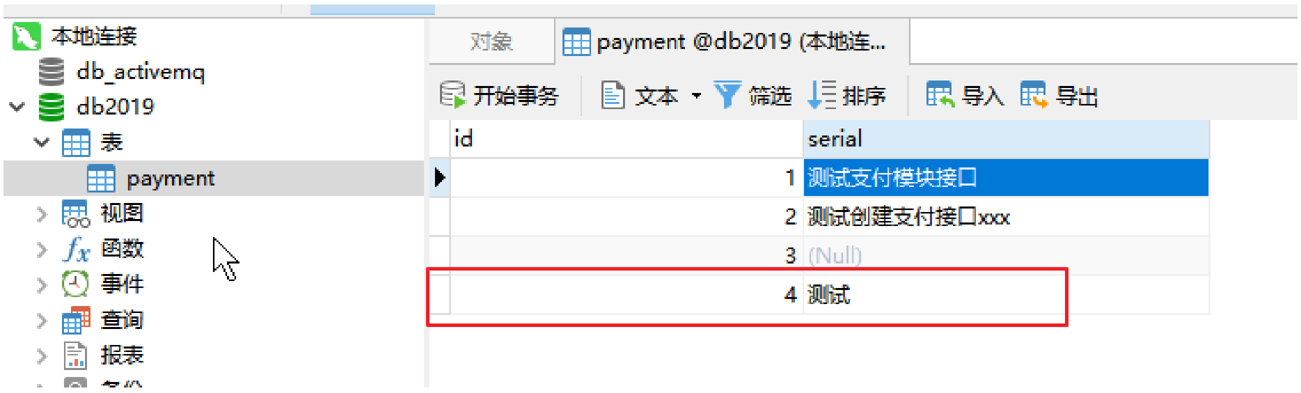
200

```
Content-Typ… application/json
Transfer-En… chunked
Date:        Fri, 12 Jun 2020 14:57:32 GMT
Keep-Alive:  timeout=60
Connection:  keep-alive
```

```
{
    code : 200,
    message : "插入数据库成功",
    data : 1
}
```

length: 55 bytes

查看数据库记录, 可以看到新插入记录中的serial字段已经有值了

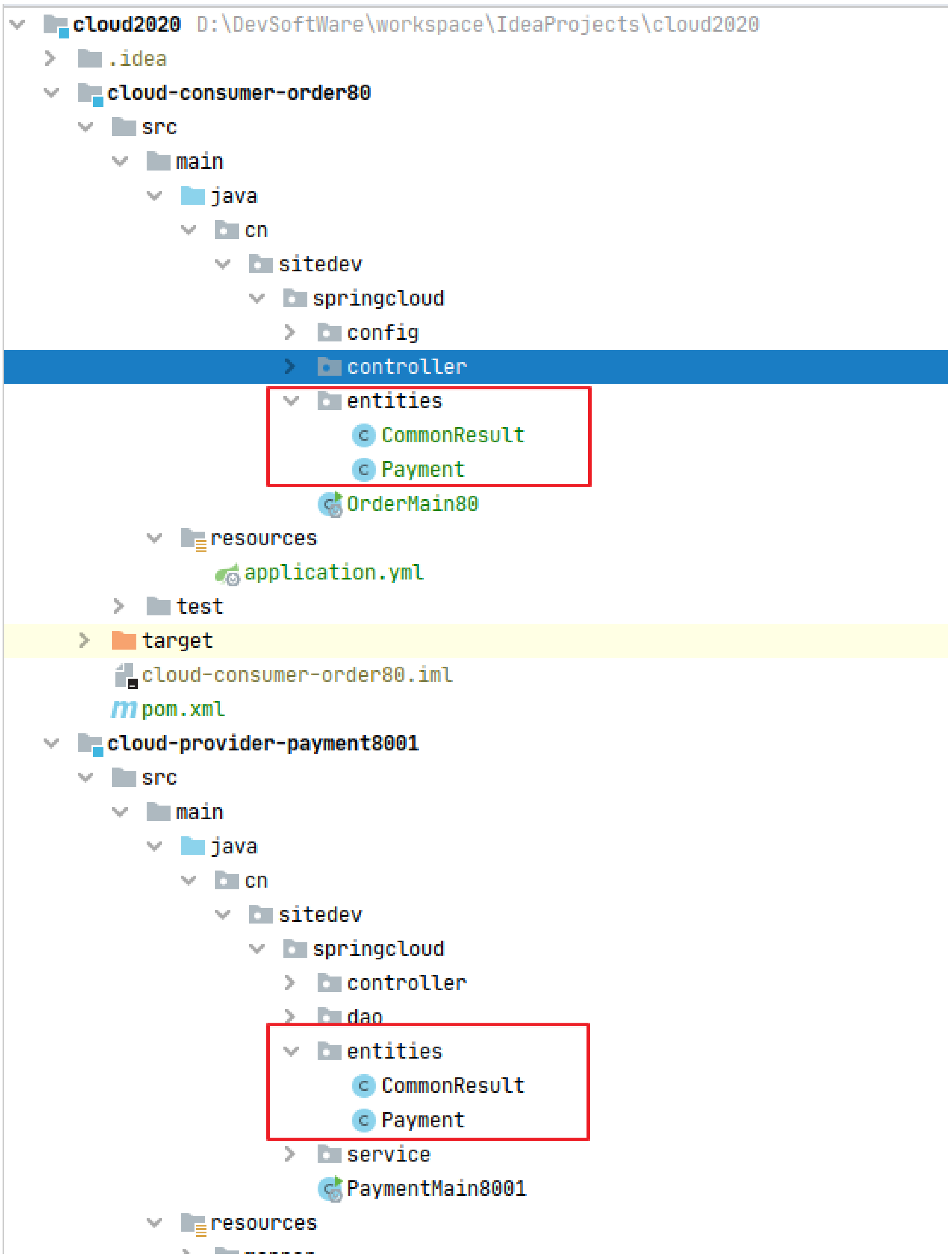


id	serial
1	测试支付模块接口
2	测试创建支付接口xxx
3	(Null)
4	测试

3.1.4. 工程重构

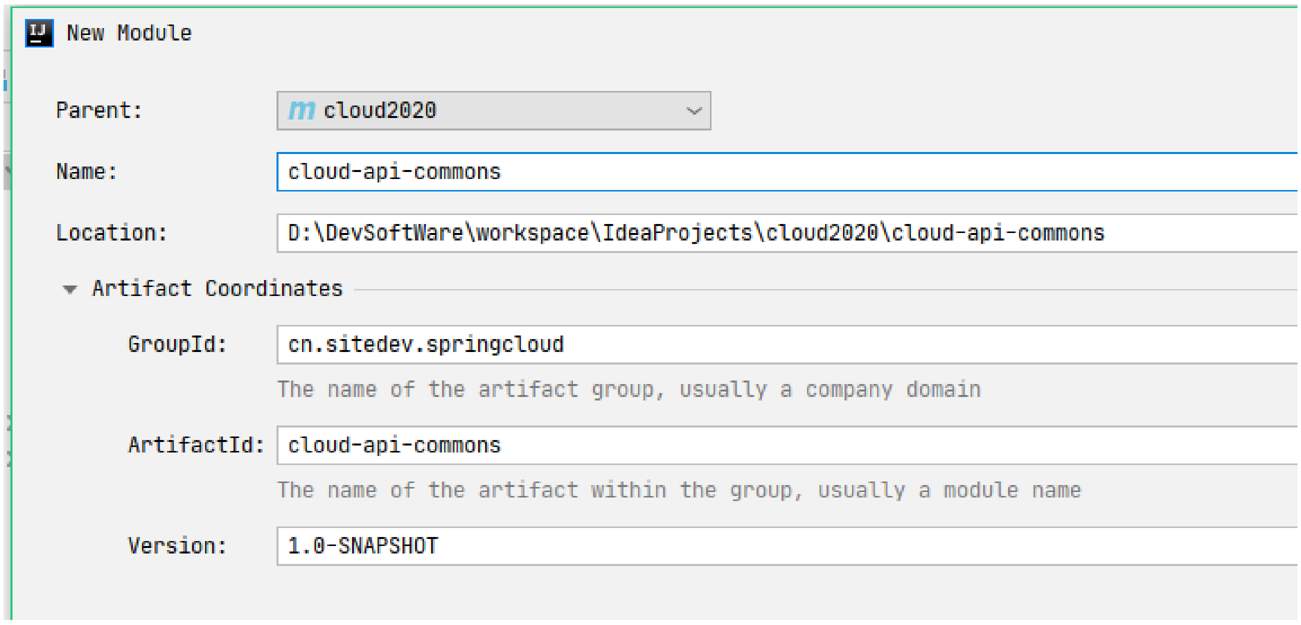
3.1.4.1. 观察问题

系统中有重复部分, 需要进行重构



```
cloud2020  D:\DevSoftWare\workspace\IdeaProjects\cloud2020
  .idea
  cloud-consumer-order80
    src
      main
        java
          cn
            sitedev
              springcloud
                config
                controller
                entities
                  CommonResult
                  Payment
                OrderMain80
        resources
          application.yml
      test
    target
    cloud-consumer-order80.iml
    pom.xml
  cloud-provider-payment8001
    src
      main
        java
          cn
            sitedev
              springcloud
                controller
                dao
                entities
                  CommonResult
                  Payment
                service
                PaymentMain8001
        resources
```

3.1.4.2. 新建Module


New Module

Parent: cloud2020

Name: cloud-api-commons

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-api-commons

Artifact Coordinates

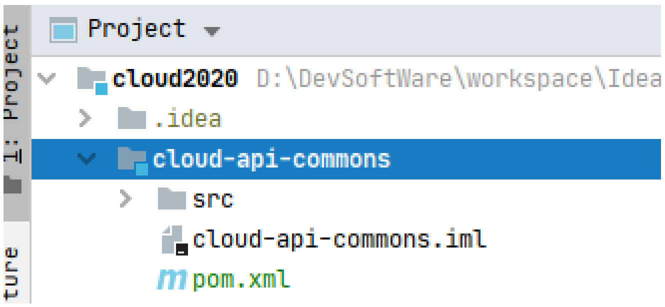
GroupId: cn.sitedev.springcloud

The name of the artifact group, usually a company domain

ArtifactId: cloud-api-commons

The name of the artifact within the group, usually a module name

Version: 1.0-SNAPSHOT


```
Project
cloud2020  D:\DevSoftWare\workspace\Idea
  .idea
  cloud-api-commons
    src
    cloud-api-commons.iml
    pom.xml
```

3.1.4.3. 修改POM

```xml
<?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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-api-commons</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.1.0</version>
        </dependency>
    </dependencies>

</project>
```

3.1.4.4. entities

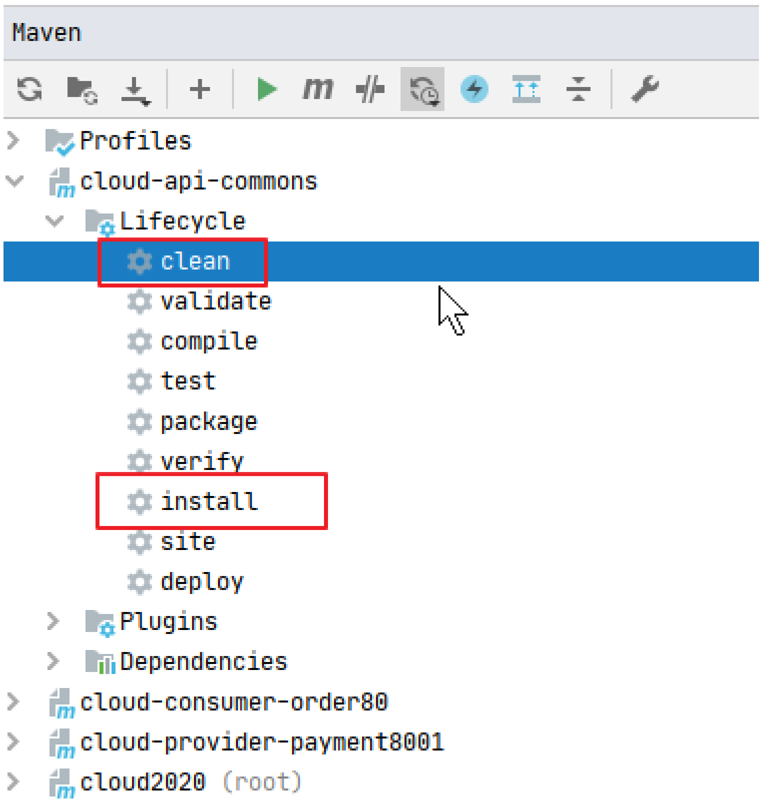
CommonResult通用封装类

```java
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CommonResult<T> {
    private Integer code;
    private String message;
    private T data;

    public CommonResult(Integer code, String message) {
        this(code, message, null);
    }
}
```

Payment实体

```java
package cn.sitedev.springcloud.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Payment implements Serializable {
    private Long id;
    private String serial;
}
```

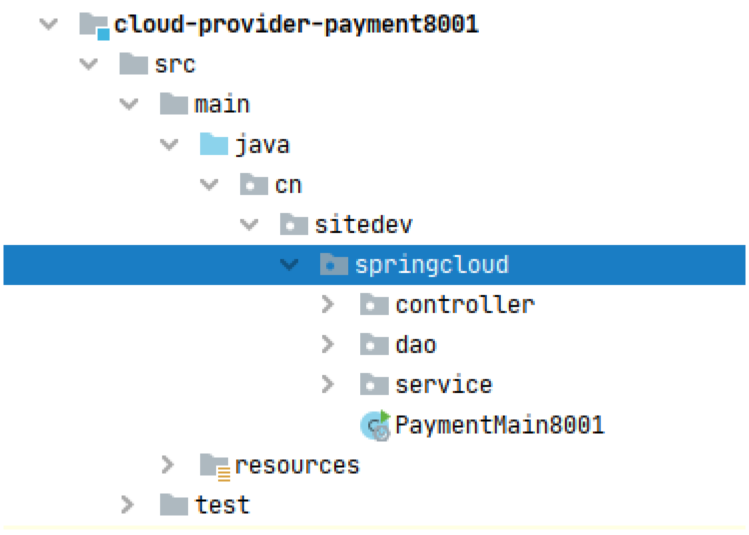
3.1.4.5. 执行maven 命令 clean install



3.1.4.6. 订单80和支付8001模块分别进行改造

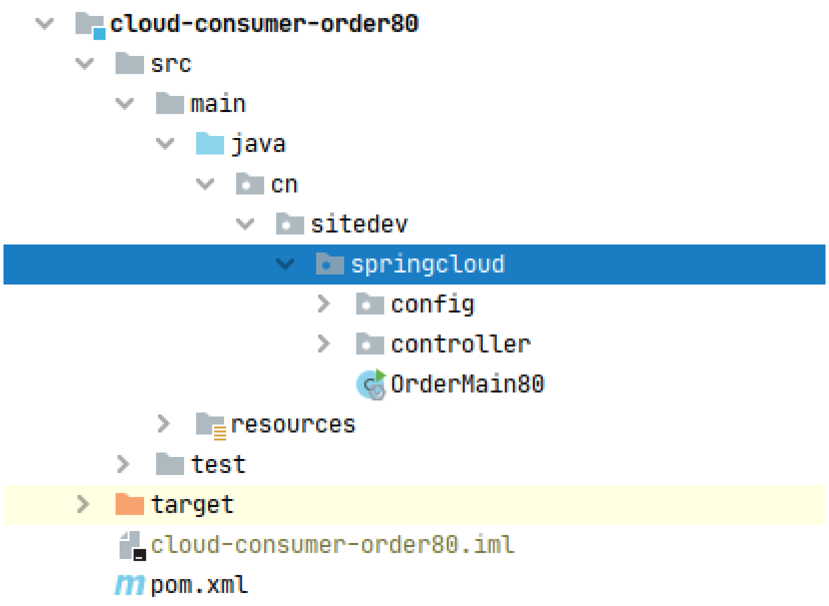
3.1.4.6.1. 删除各自的原先的entities文件夹

cloud-provider-payment8001模块:



```
cloud-provider-payment8001
  src
    main
      java
        cn
          sitedev
            springcloud
              controller
              dao
              service
              PaymentMain8001
      resources
    test
```

cloud-consumer-order80模块:



```
cloud-consumer-order80
  src
    main
      java
        cn
          sitedev
            springcloud
              config
              controller
              OrderMain80
      resources
    test
  target
  cloud-consumer-order80.iml
  pom.xml
```

3.1.4.6.2. 修改各自POM内容

cloud-provider-payment8001模块:

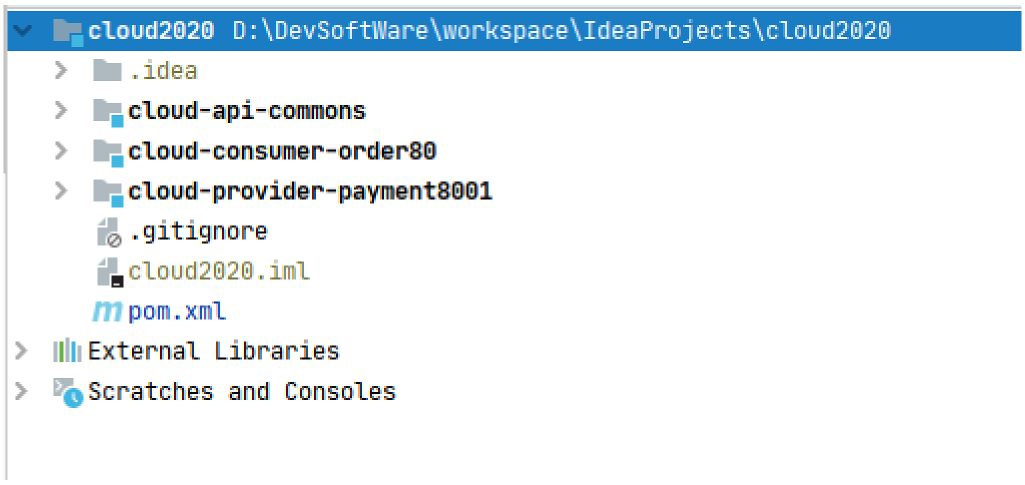
```xml
        <!--引入自己定义的api通用包，可以使用Payment支付Entity-->
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
```

cloud-consumer-order80模块:

```xml
        <!--引入自己定义的api通用包，可以使用Payment支付Entity-->
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
```

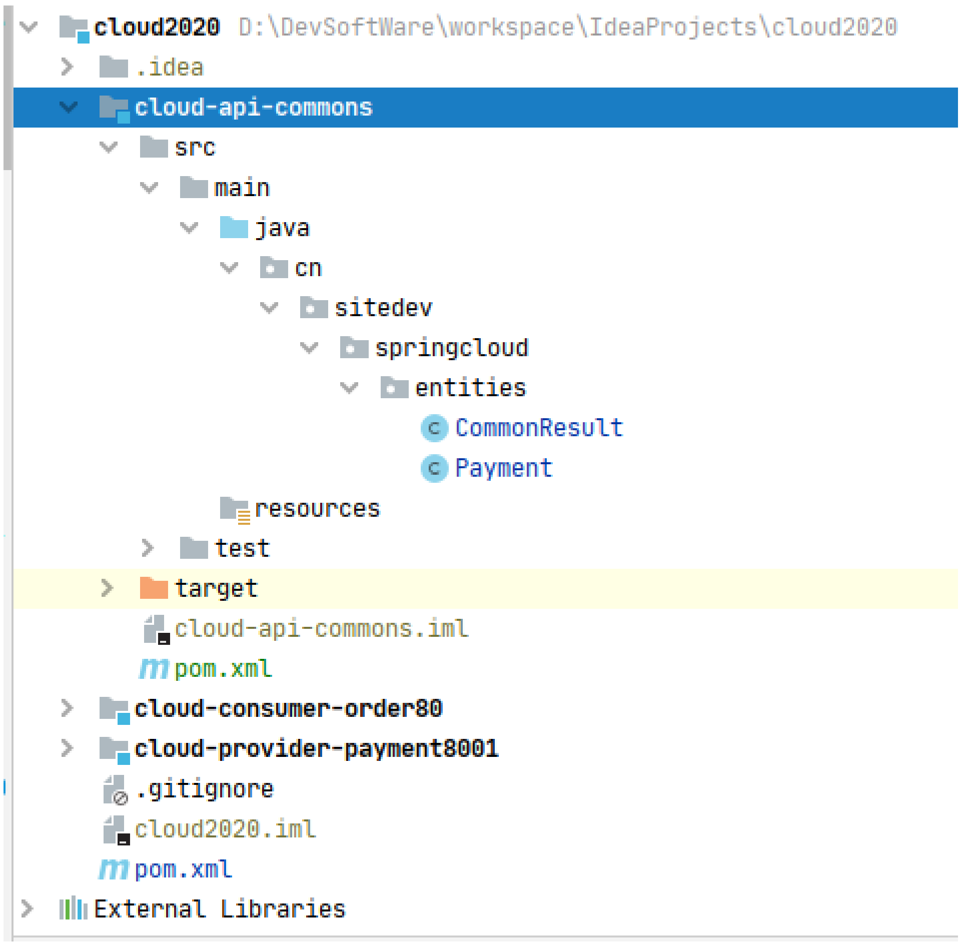
3.2. 目前工程样图

总体一览:



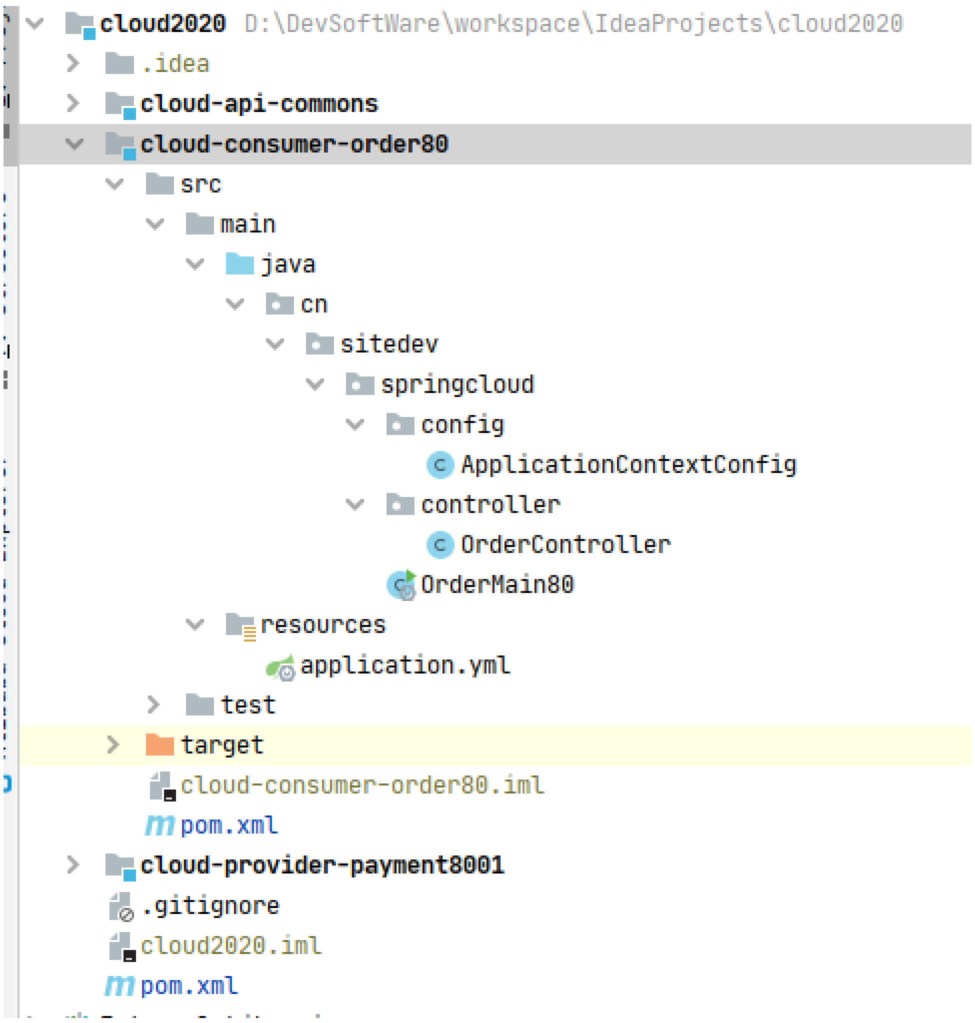
```
cloud2020  D:\DevSoftWare\workspace\IdeaProjects\cloud2020
    .idea
    cloud-api-commons
    cloud-consumer-order80
    cloud-provider-payment8001
    .gitignore
    cloud2020.iml
    pom.xml
External Libraries
Scratches and Consoles
```

cloud-api-commons模块:

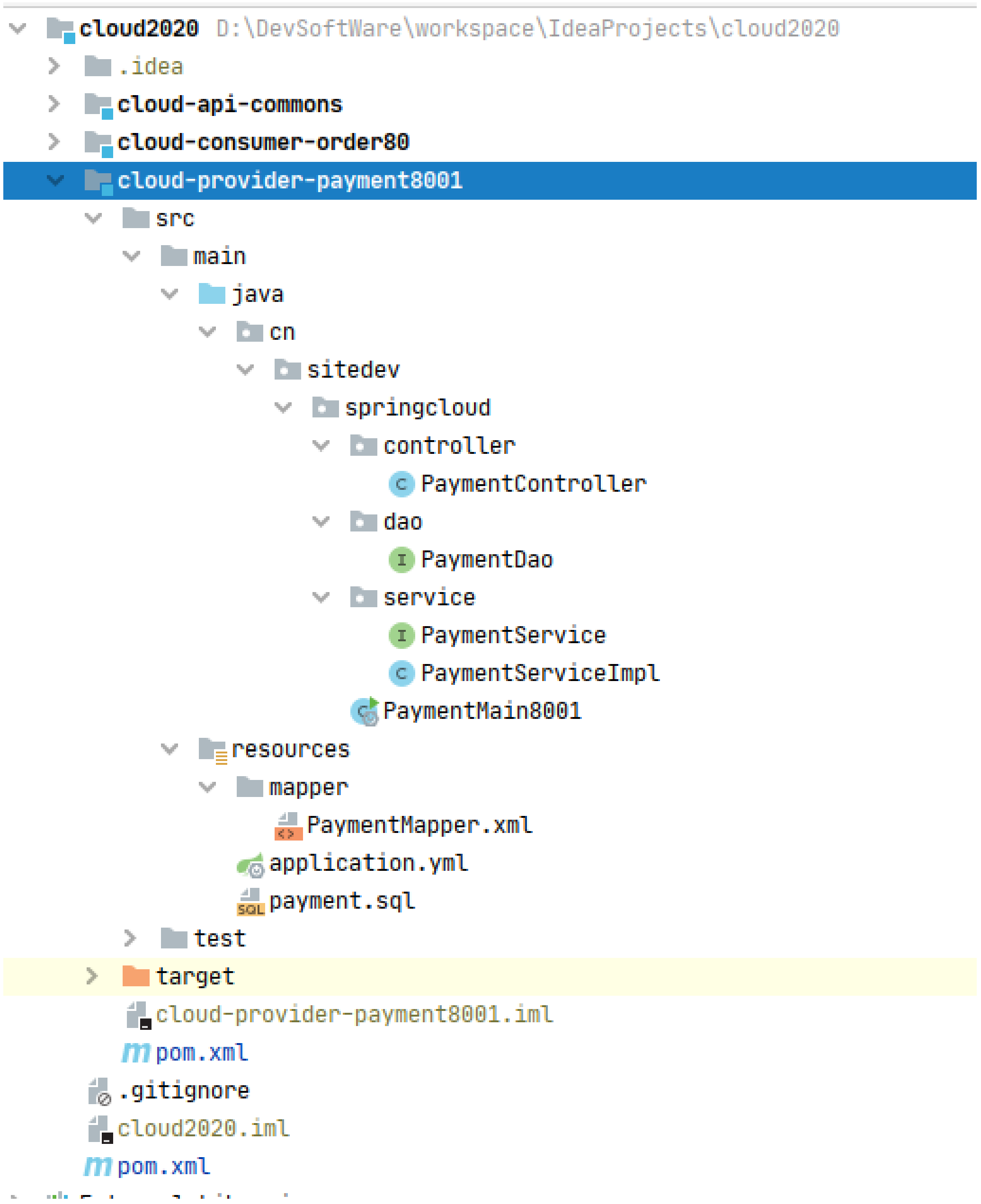


```
cloud2020  D:\DevSoftWare\workspace\IdeaProjects\cloud2020
    .idea
    cloud-api-commons
        src
            main
                java
                    cn
                        sitedev
                            springcloud
                                entities
                                    CommonResult
                                    Payment
                resources
            test
        target
        cloud-api-commons.iml
        pom.xml
    cloud-consumer-order80
    cloud-provider-payment8001
    .gitignore
    cloud2020.iml
    pom.xml
External Libraries
```

cloud-consumer-order80模块:


```
cloud2020  D:\DevSoftWare\workspace\IdeaProjects\cloud2020
  .idea
  cloud-api-commons
  cloud-consumer-order80
    src
      main
        java
          cn
            sitedev
              springcloud
                config
                  ApplicationContextConfig
                controller
                  OrderController
                OrderMain80
        resources
          application.yml
      test
    target
    cloud-consumer-order80.iml
    pom.xml
  cloud-provider-payment8001
  .gitignore
  cloud2020.iml
  pom.xml
```

cloud-provider-payment8001模块:



```
cloud2020 D:\DevSoftWare\workspace\IdeaProjects\cloud2020
  .idea
  cloud-api-commons
  cloud-consumer-order80
  cloud-provider-payment8001
    src
      main
        java
          cn
            sitedev
              springcloud
                controller
                  PaymentController
                dao
                  PaymentDao
                service
                  PaymentService
                  PaymentServiceImpl
                PaymentMain8001
        resources
          mapper
            PaymentMapper.xml
          application.yml
          payment.sql
      test
    target
    cloud-provider-payment8001.iml
    pom.xml
  .gitignore
  cloud2020.iml
  pom.xml
```