

spring5.3.x源码阅读环境搭建-gradle构建编译

spring5.3.x源码阅读环境搭建-gradle构建编译

- 一、依赖工具
- 二、下载源码
- 三、开始构建
- 四、编译源码
- 五、源码测试
- 六、问题及解决方案
- 附：spring源代码各个模块作用
- 结语

码炫课堂技术交流q群：963060292

Spring系列生态十分丰富，涉及到各个方面。但是作为Spring生态的核心基础Spring，是最重要的环节，需要理解Spring的设计原理，就需要深度研读Spring源码。

本文着重阐述当前最新版spring5.3.x的源码构建过程，由于构建工具采用gradle（spring团队已经抛弃maven构建，全面拥抱gradle了），很多小伙伴不太熟悉gradle，所以构建过程有少许困难。本文将带大家手把手的搭建spring源码阅读环境构建。

一、依赖工具

1、git

拉取源码使用

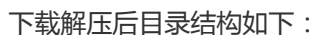
2、jdk8及以上

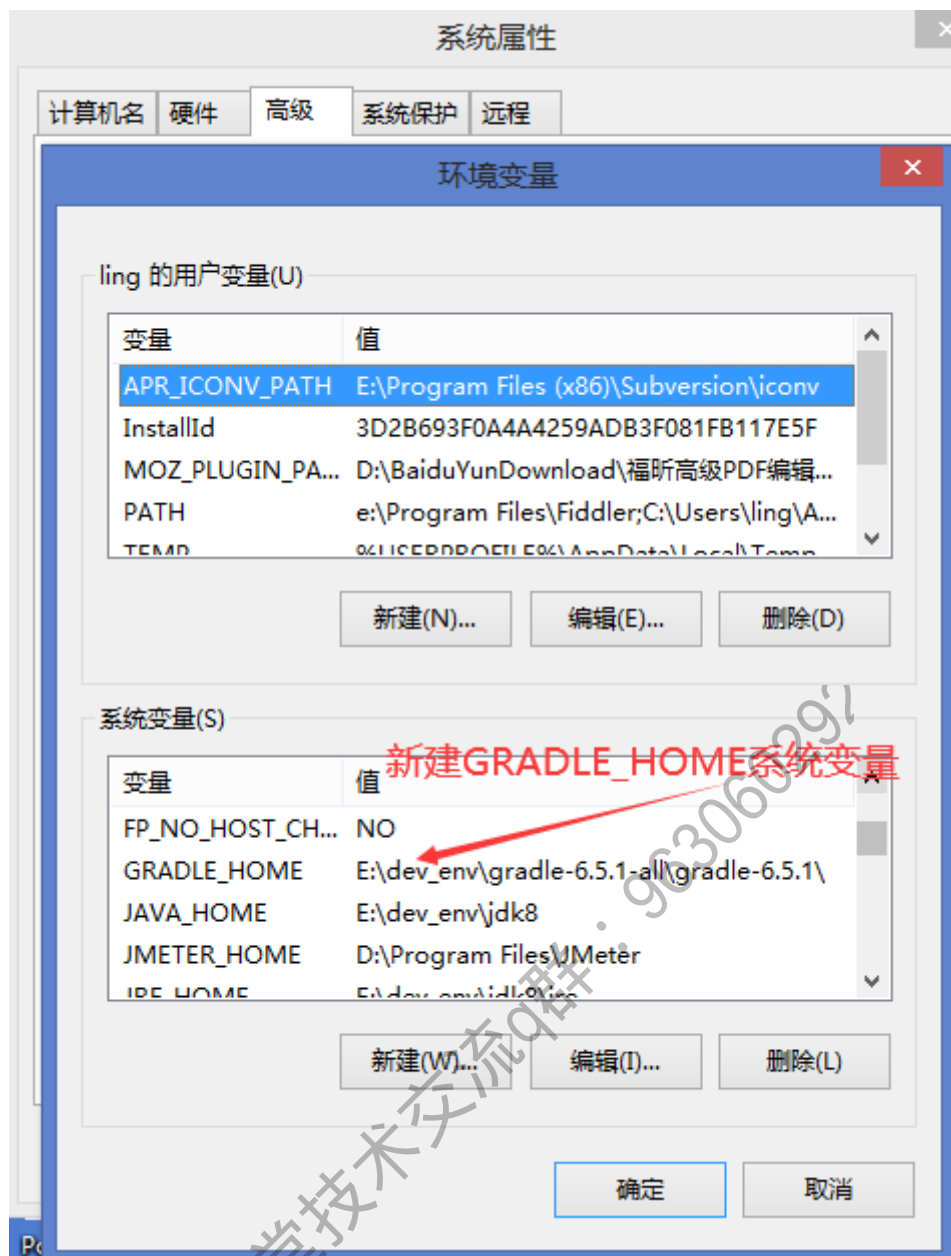
一般小伙伴机器上都已经装好了

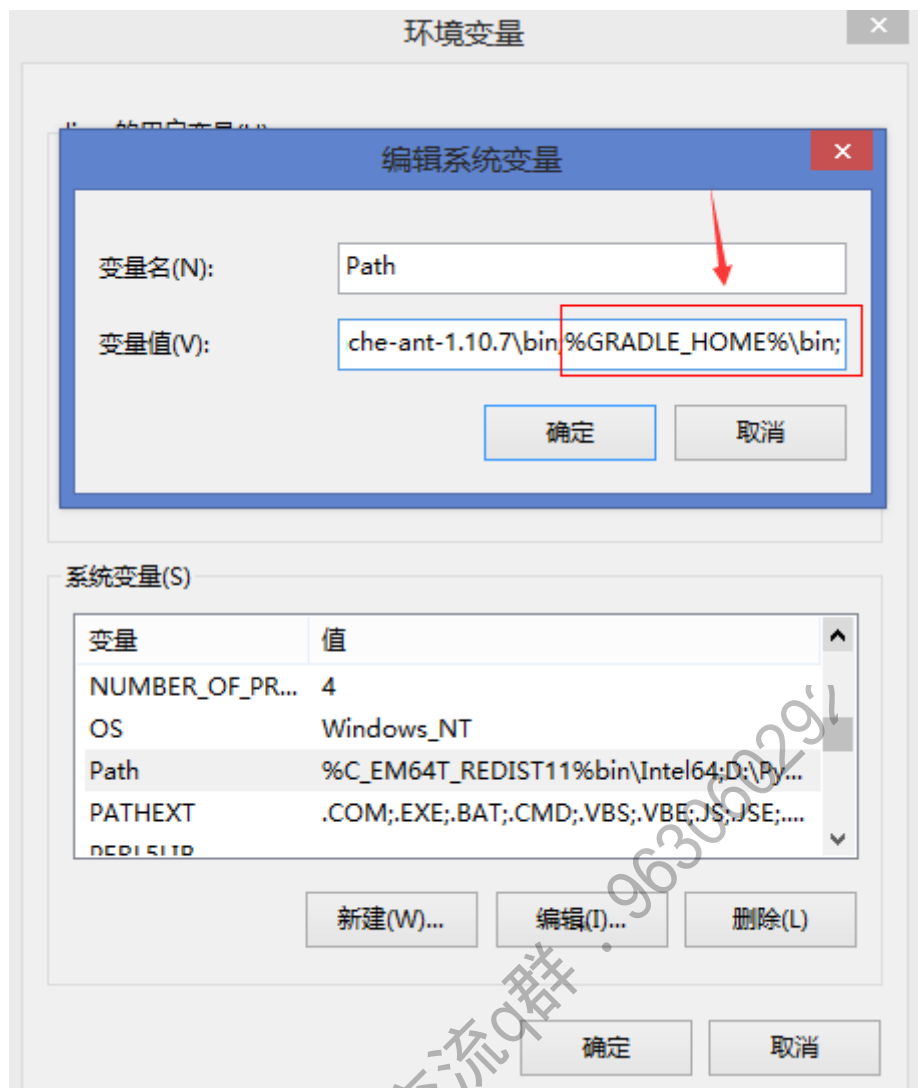
```
E:\mypro\IdeaProjects\spring-boot>java -version
java version "1.8.0_40"
Java(TM) SE Runtime Environment (build 1.8.0_40-b25)
Java HotSpot(TM) 64-Bit Server VM (build 25.40-b25, mixed mode)
```

3、gradle6.5.1

打开 <https://services.gradle.org/distributions/> 选择最新版本：gradle-6.5.1-all.zip(all版本是带源码的)







完成后打开cmd，执行

```
gradle -v
```

```
E:\mypro\IdeaProjects\spring-boot>gradle -v

-----
Gradle 6.5.1
-----

Build time:   2020-06-30 06:32:47 UTC
Revision:     66bc713f7169626a7f0134bf452abde51550ea0a

Kotlin:       1.3.72
Groovy:       2.5.11
Ant:          Apache Ant(TM) version 1.10.7 compiled on September 1 2019
JVM:          1.8.0_40 (Oracle Corporation 25.40-b25)
OS:           Windows 8.1 6.3 amd64

E:\mypro\IdeaProjects\spring-boot>
```

表示已经安装成功，版本为6.5.1

4、idea2020.1.2

(网上很多朋友表示idea2020之前的版本导入时始终有问题，建议升级到2020.1版本，smart哥当前使用的就是2020.1.2版本)

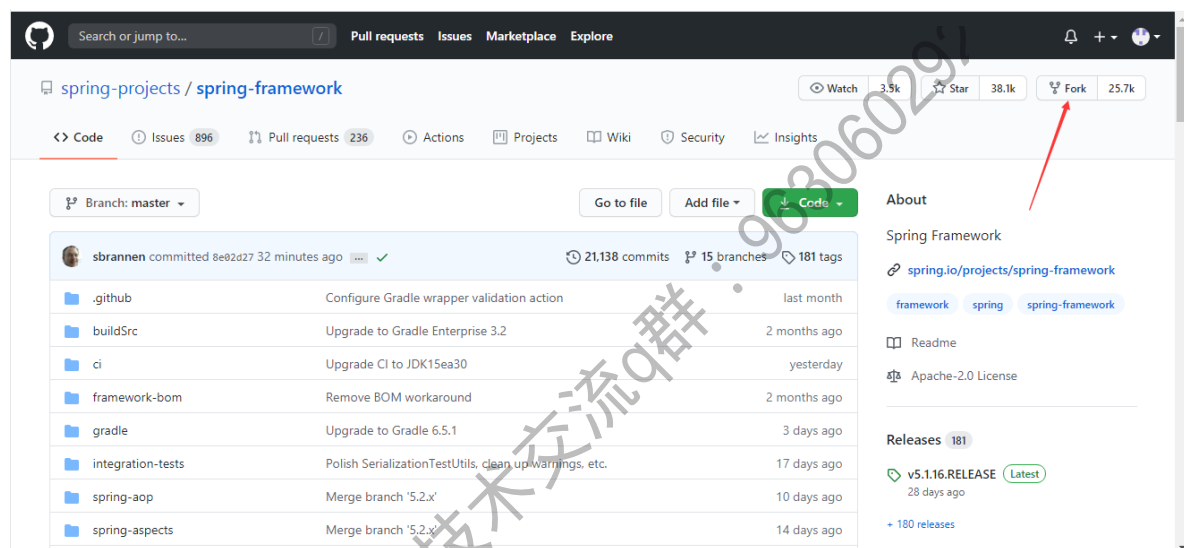
二、下载源码

从官方仓库 <https://github.com/spring-projects/spring-framework> Fork 出属于自己的仓库。

- 为什么要 Fork ? 既然开始阅读、调试源码, 我们可能会写一些注释, 有了自己的仓库, 可以进行自由的提交。
- 本文使用的 Spring 版本为 5.3.x 的 master 分支代码 (5.3.0-SNAPSHOT)。
- 使用 IntelliJ IDEA 从 Fork 出来的仓库拉取代码。因为 Spring 项目比较大, 从仓库中拉取代码的时间会比较长。所以我这边是 git clone 到本地, 然后再导入 idea 中的。

具体过程如下:

1、打开 <https://github.com/spring-projects/spring-framework>, 点击右上角 Fork 即可, 这样就把 spring 仓库 fork 到自己的仓库中了。

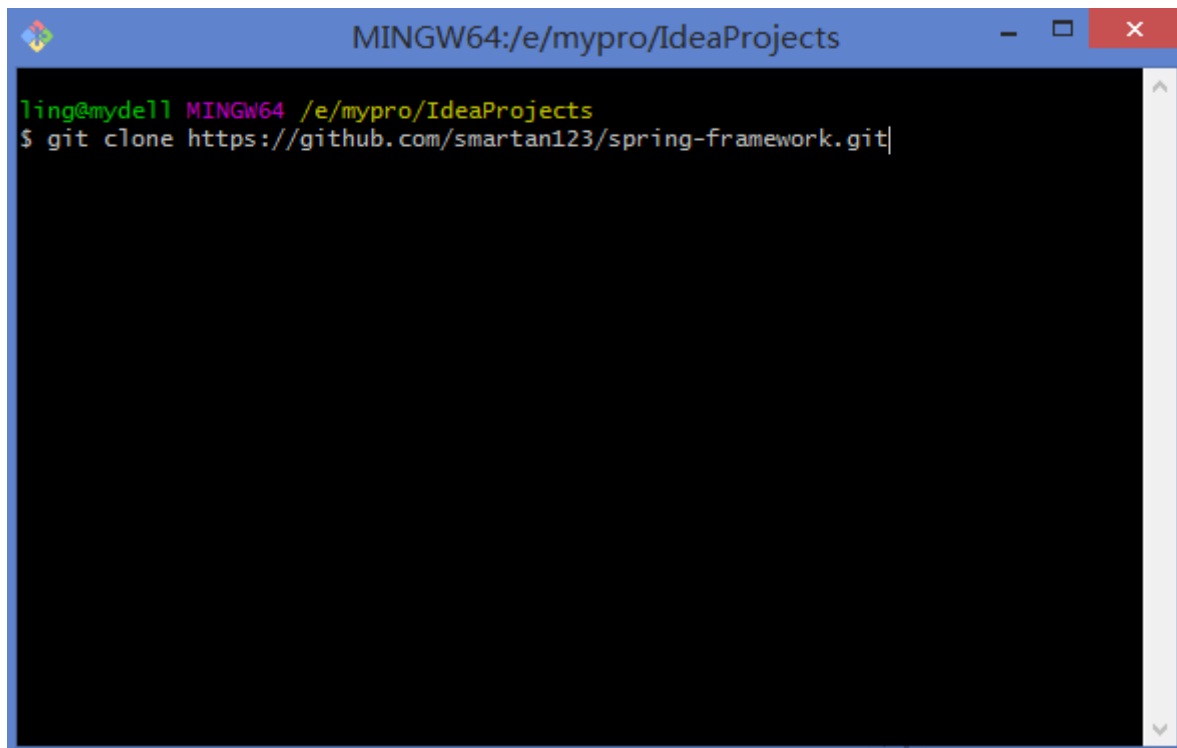


2、选择一个目录, 我的是 E:\myproIdeaProjects, 空白处右击 Git Bash Here

执行:

```
git clone https://github.com/smartan123/spring-framework.git
```

下载到本地



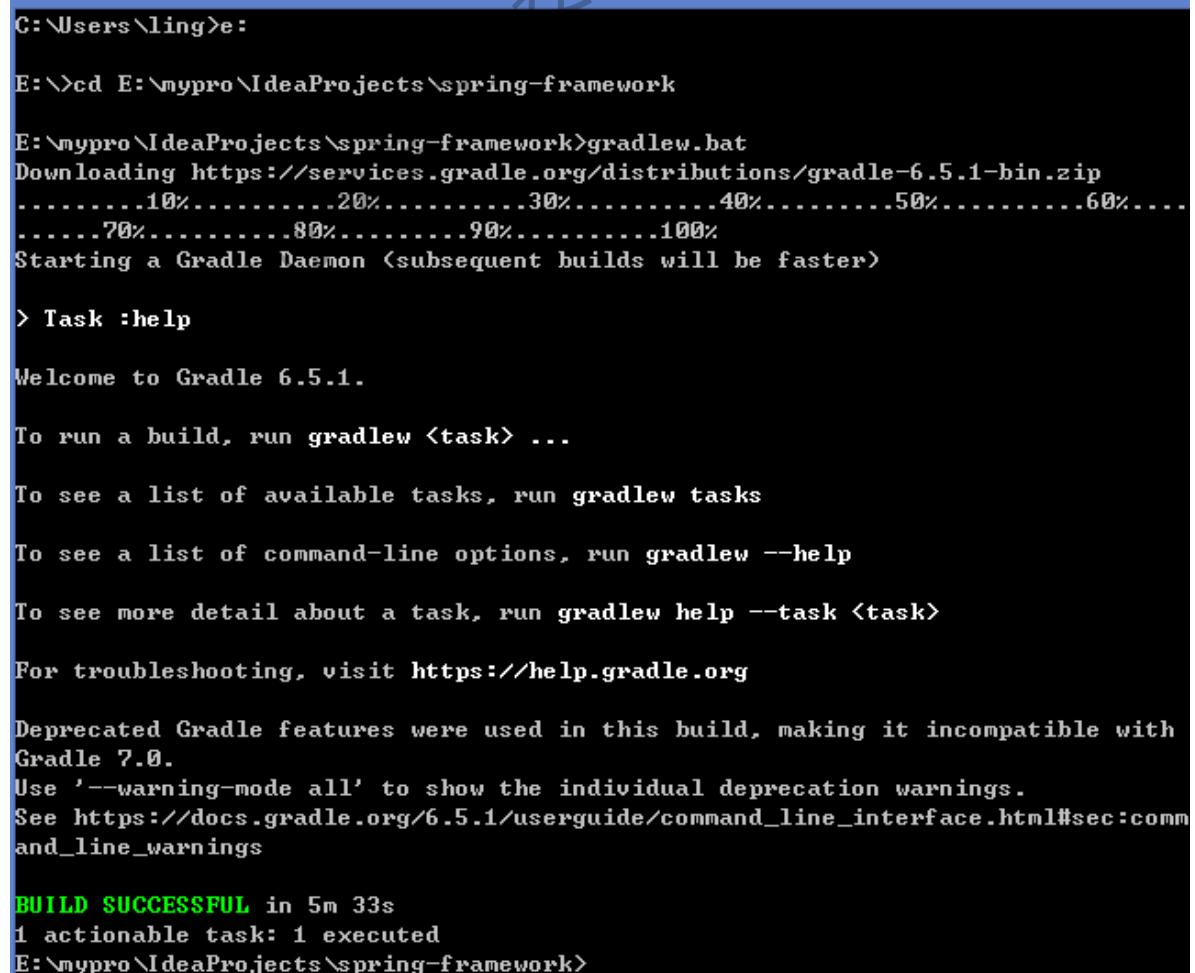
```
ling@mydell MINGW64 /e/mypro/IdeaProjects
$ git clone https://github.com/smartan123/spring-framework.git
```

三、开始构建

1、构建之前先安装gradle，因为spring是gradle构建的。

在cmd中进入源码根目录，输入gradlew.bat命令，脚本将自动下载gradle-6.5.1-bin.zip包

（这一步其实也可以省略，可以直接将源码导入idea中）



```
C:\Users\ling>e:

E:\>cd E:\mypro\IdeaProjects\spring-framework

E:\mypro\IdeaProjects\spring-framework>gradlew.bat
Downloading https://services.gradle.org/distributions/gradle-6.5.1-bin.zip
.....10%.....20%.....30%.....40%.....50%.....60%.....
.....70%.....80%.....90%.....100%
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :help

Welcome to Gradle 6.5.1.

To run a build, run gradlew <task> ...

To see a list of available tasks, run gradlew tasks

To see a list of command-line options, run gradlew --help

To see more detail about a task, run gradlew help --task <task>

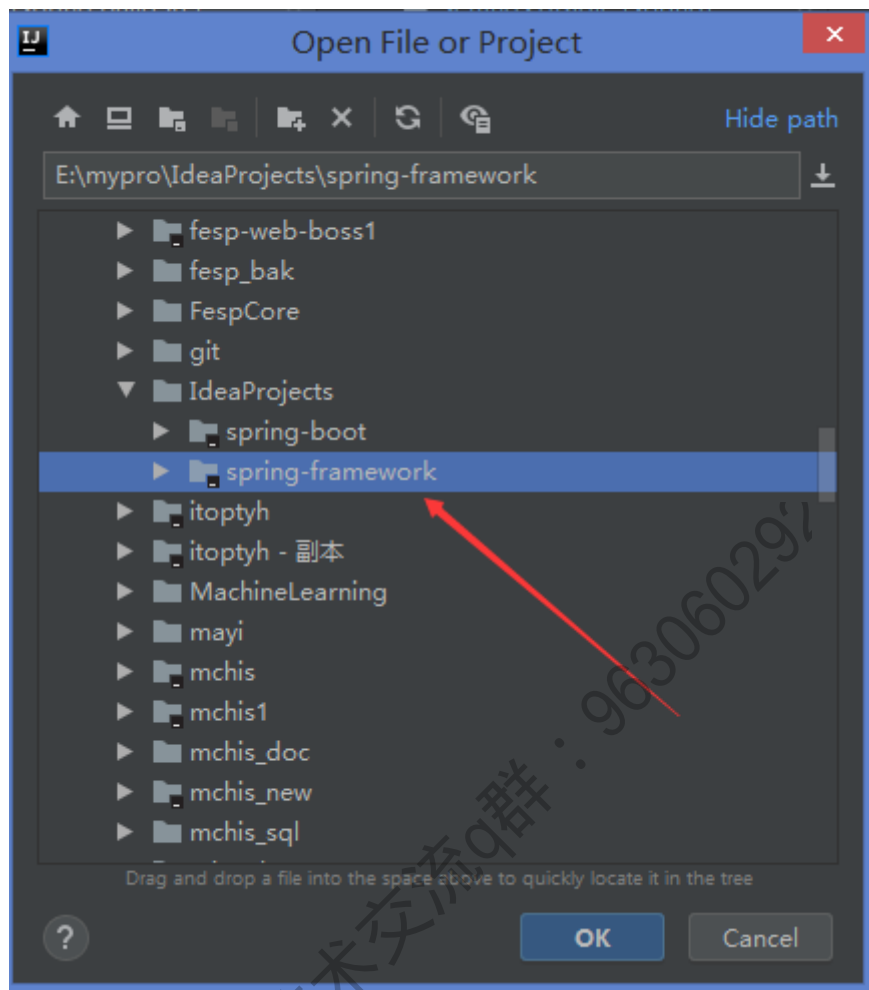
For troubleshooting, visit https://help.gradle.org

Deprecated Gradle features were used in this build, making it incompatible with
Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.5.1/userguide/command_line_interface.html#sec:comm
and_line_warnings

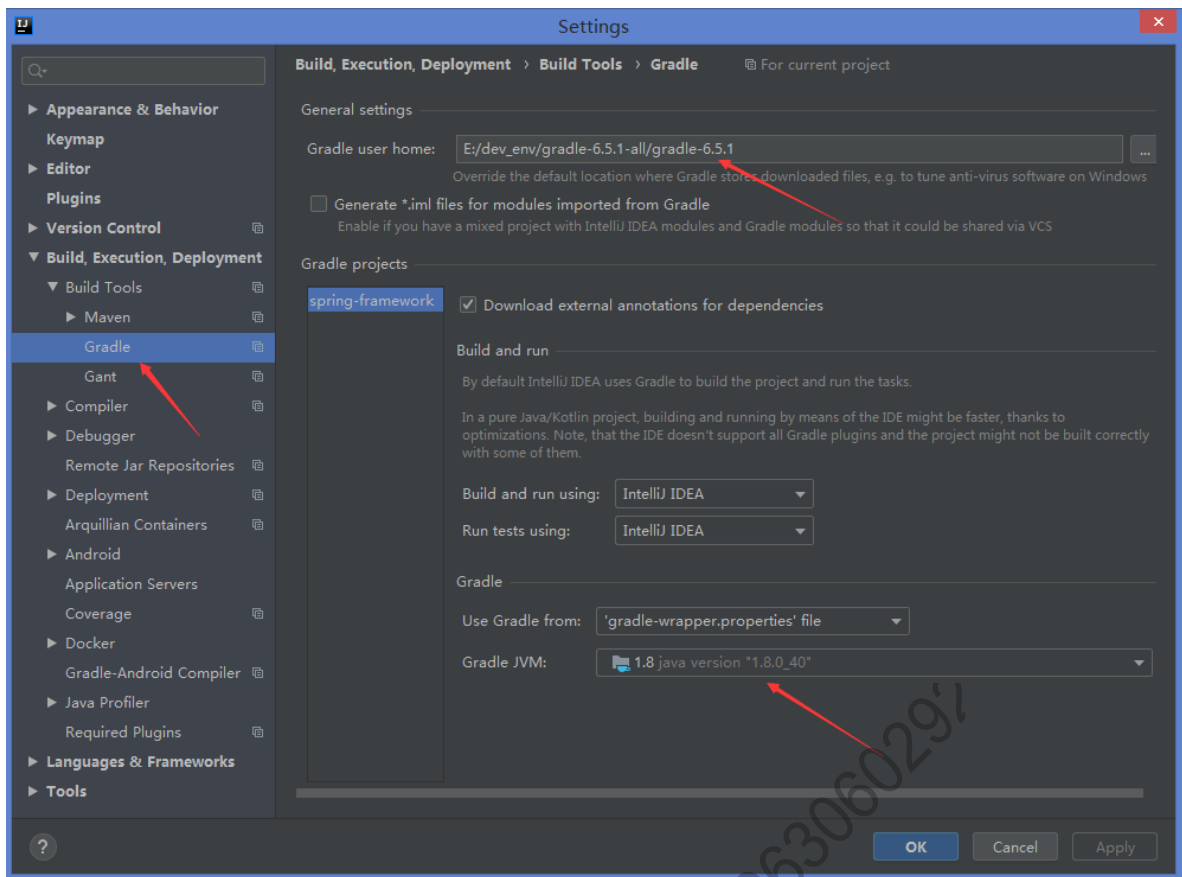
BUILD SUCCESSFUL in 5m 33s
1 actionable task: 1 executed
E:\mypro\IdeaProjects\spring-framework>
```

2、打开IDEA,直接open源码项目

【文件】->【Open】,选择源码根目录, -> 【ok】



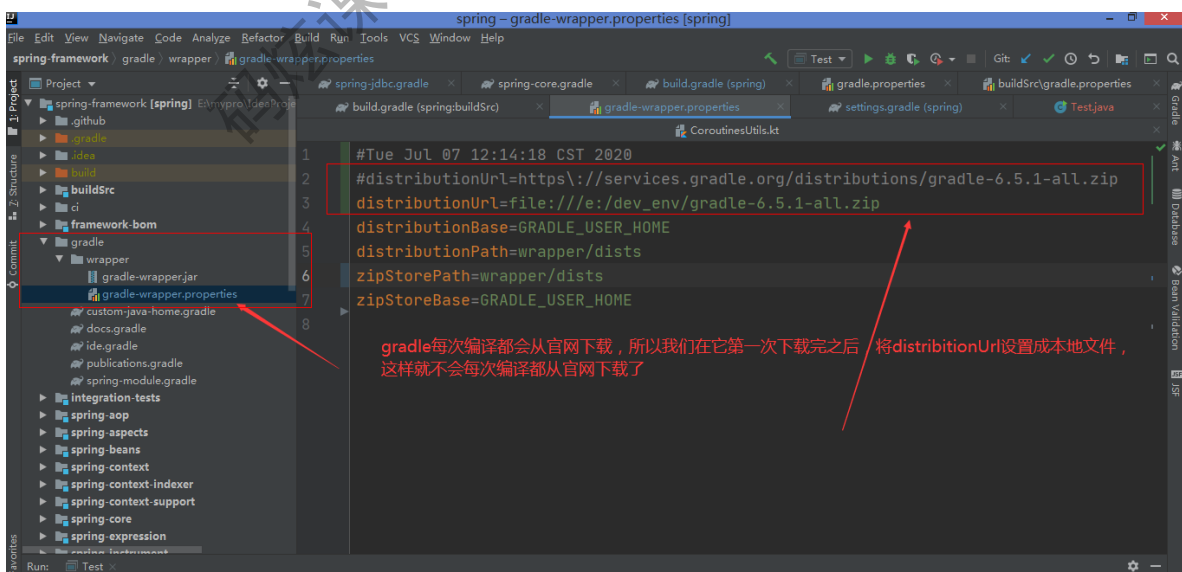
3、打开项目之后,【File】->【Settings】,设置Gradle, Gradle user home这个目录这里我设置的刚安装的gradle目录。如果不设置的话,它默认是C:\Users\ling\.gradle,这个目录你就可以认为相当于我们的本地的maven仓库,gradle编译项目所依赖的jar都会下载后放入这个目录中。jvm默认我这里就是jdk8, jdk至少就是jdk8。



4、设置完毕之后，打开工程下的gradle目录->wrapper目录下的，gradle-wrapper.properties文件。因为gradle每次编译都会从官网下载指定版本（gradle-6.5.1-all.zip），所以我们在它第一次下载完之后，将distributionUrl设置成本地文件，这样就不会每次编译都从官网下载了，如下图：

distributionUrl=file:///e:/dev_env/gradle-6.5.1-all.zip(这里选择gradle的压缩包的全路径地址)

(建议用最新的idea2020版本，最省事，打开后自动编译，免去不必要的麻烦)



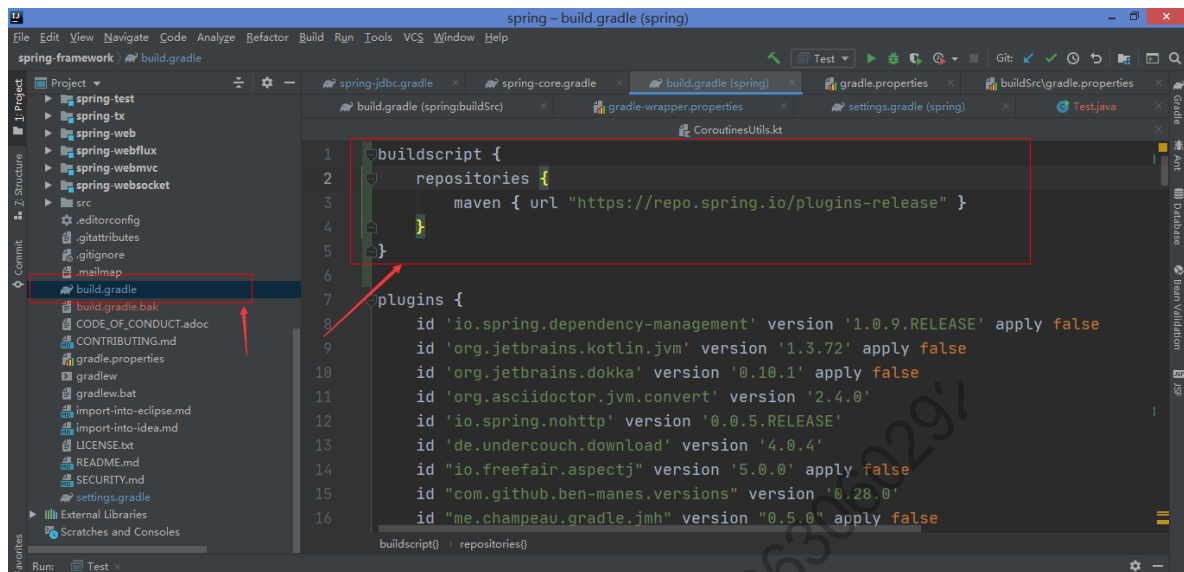
5、打开build.gradle文件（这个就相当于maven的pom文件），在文件头部加上


```

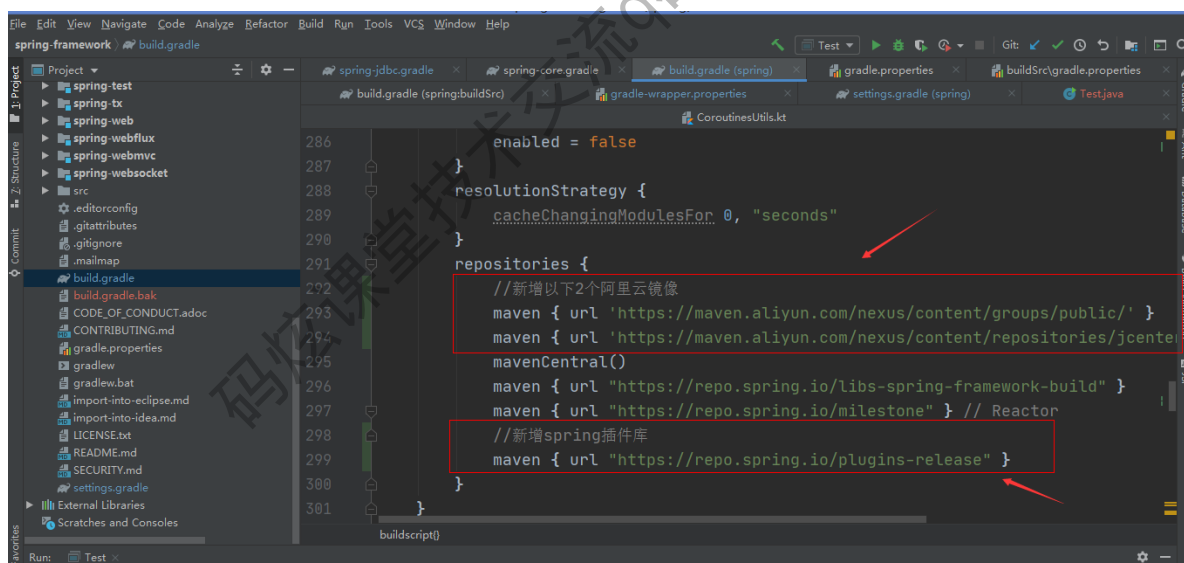
buildscript {
    repositories {
        maven { url "https://repo.spring.io/plugins-release" }
    }
}

```

如下图：



6、然后往下继续寻找，找到如下代码段：



红色线框部分是需要添加的，主要就是添加阿里云镜像和spring的插件库，这里的镜像和我们配置maven的镜像是一样的，目的就是加快依赖包的下载速度，如果不配置镜像的话，可能会编译几个小时。

具体修改如下：

```

repositories {
    //新增以下2个阿里云镜像
    maven { url 'https://maven.aliyun.com/nexus/content/groups/public/'
}

    maven { url
'https://maven.aliyun.com/nexus/content/repositories/jcenter' }
    mavenCentral()
    maven { url "https://repo.spring.io/libs-spring-framework-build" }
    maven { url "https://repo.spring.io/milestone" } // Reactor
    //新增spring插件库
    maven { url "https://repo.spring.io/plugins-release" }
}

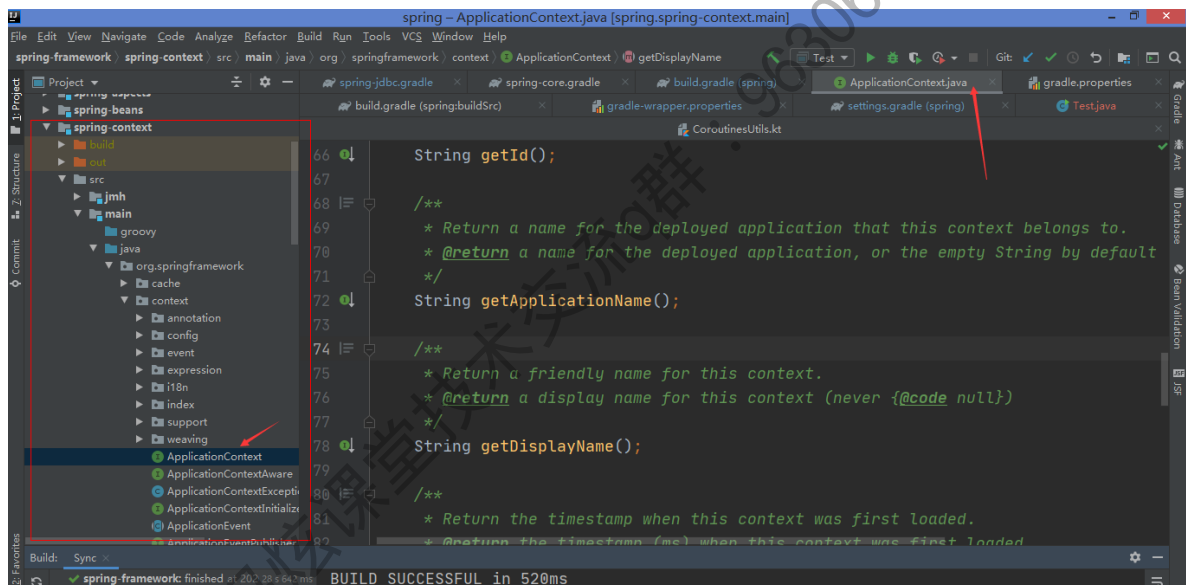
```

修改保存后会自动开始构建。

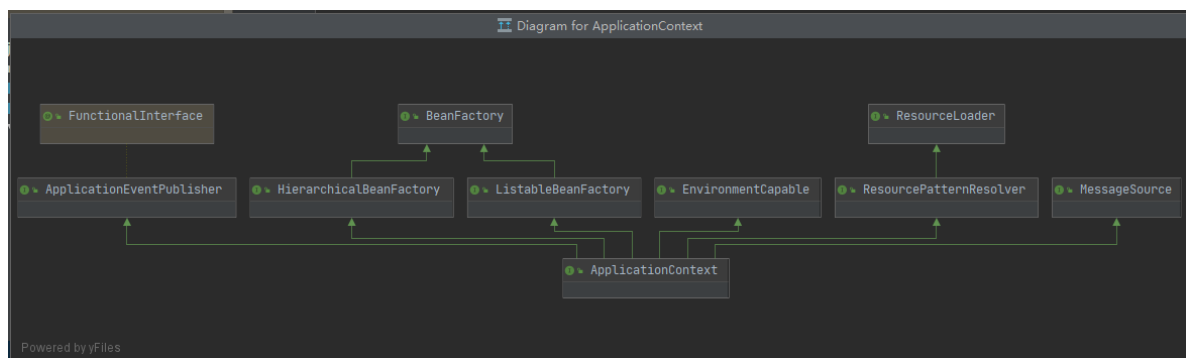
等待一定时间后，构建完毕！（注意：这里只是把依赖包下载下来，其实还没有真正开始编译）

如果构建失败重新refresh几次就行了，一般就是包下载超时之类的错误。

7、构建成功之后，找到ApplicationContext 类



打开后，按下Ctrl+Alt+U键，如果出现下图所示类图界面说明构建成功了！（构建过程就是找依赖对象的过程）

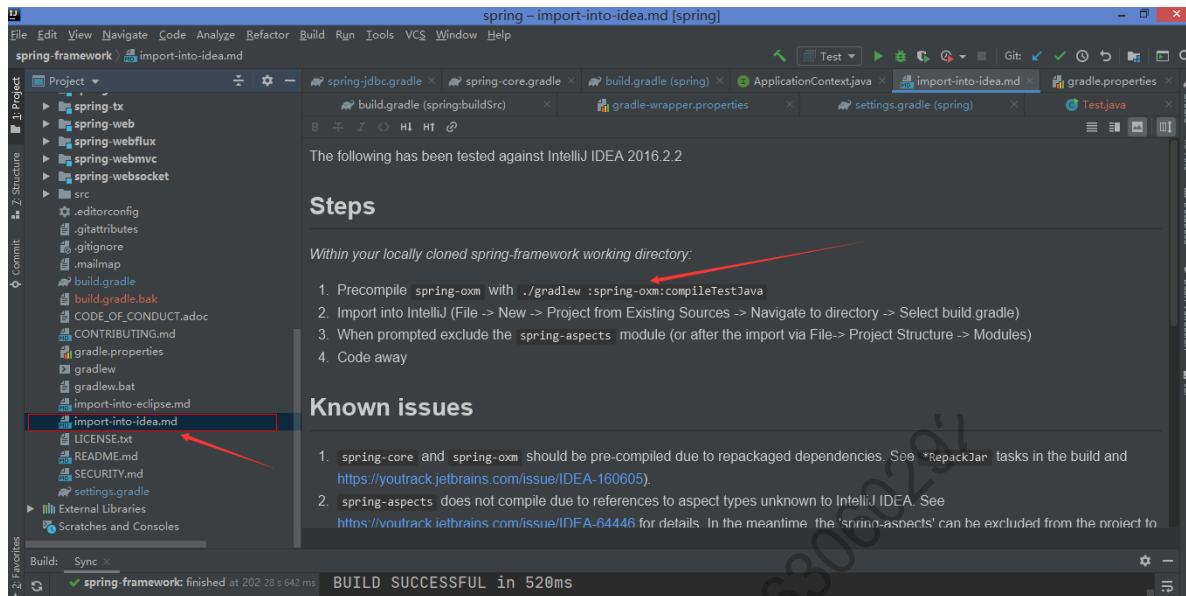


此时可以查看Spring的源码了，但是我们需要在源码的基础上进行修改，开发，调试，添加注释等等操作，所以需要将源码进行编译打包，下面就是将源码编译的过程。

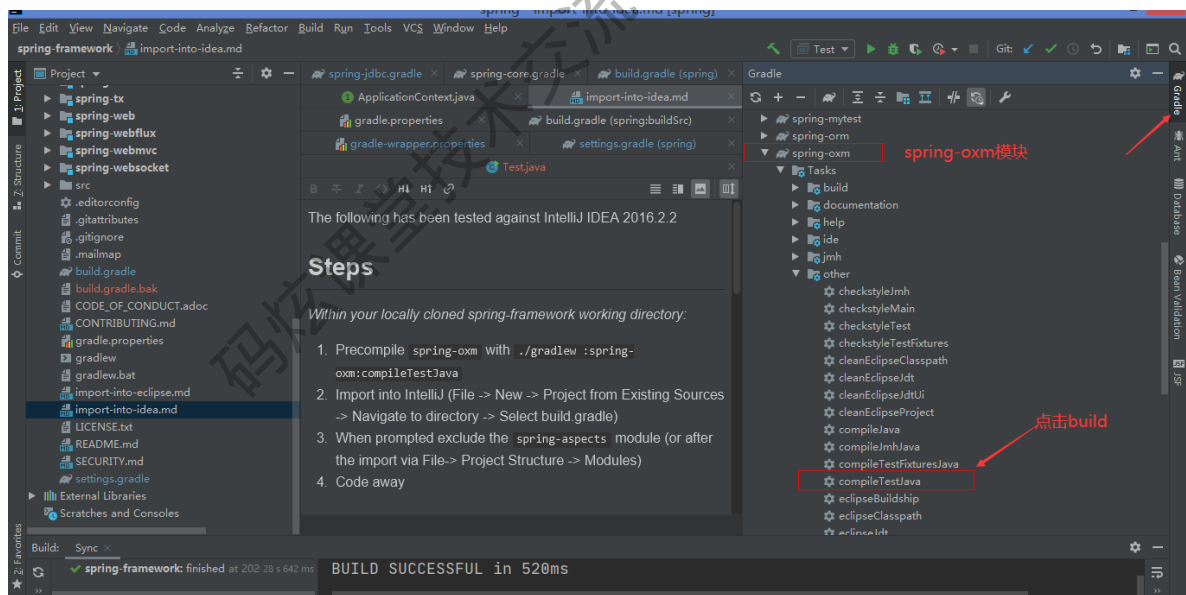
四、编译源码

1、在构建完成源码之后，就搭建好了阅读源码的环境了，此时我们还需要将源码编译打包。

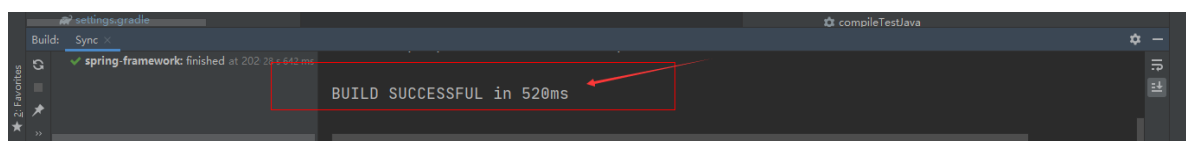
在编译之前需要进行一些配置修改，可以查看import-into-idea.md文档。



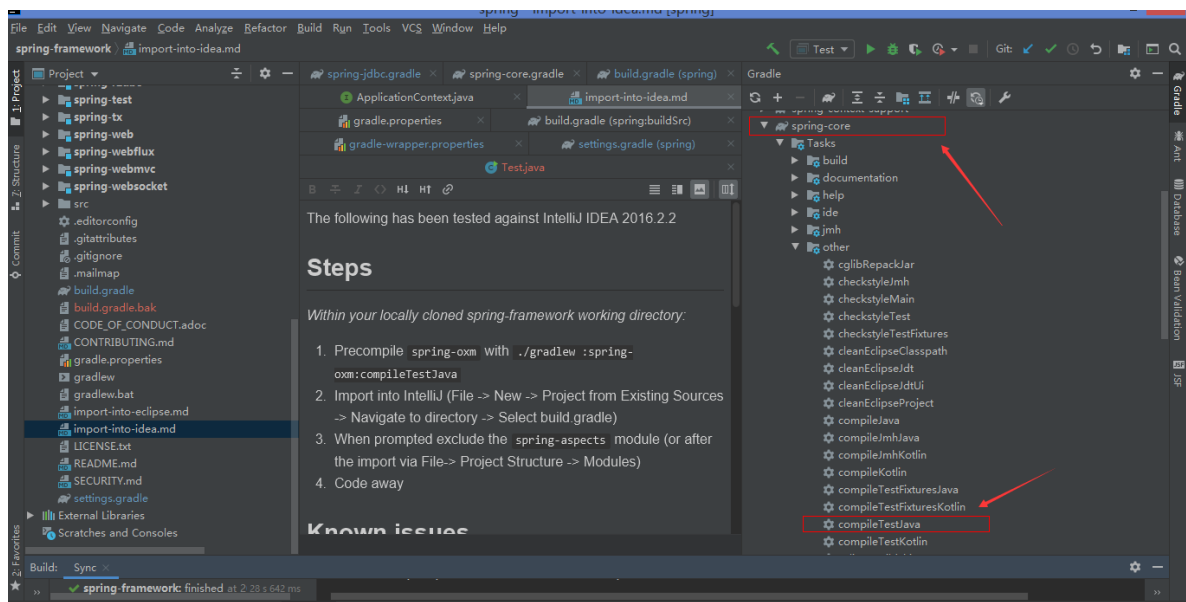
2、文档要求先编译spring-oxm下的compileTestJava，点击右上角gradle打开编译视图，找到spring-oxm模块，然后在other下找到compileTestJava，双击即可！



编译成功！

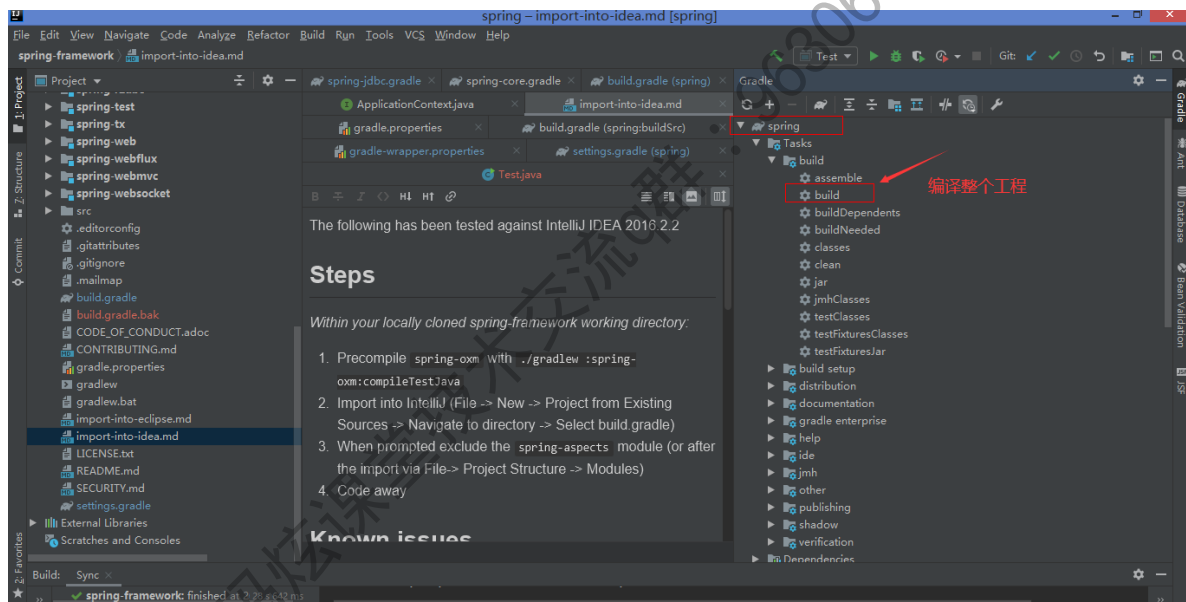


3、保险起见再编译下spring-core模块，因为之后的spring-context依赖于core，方法同上。

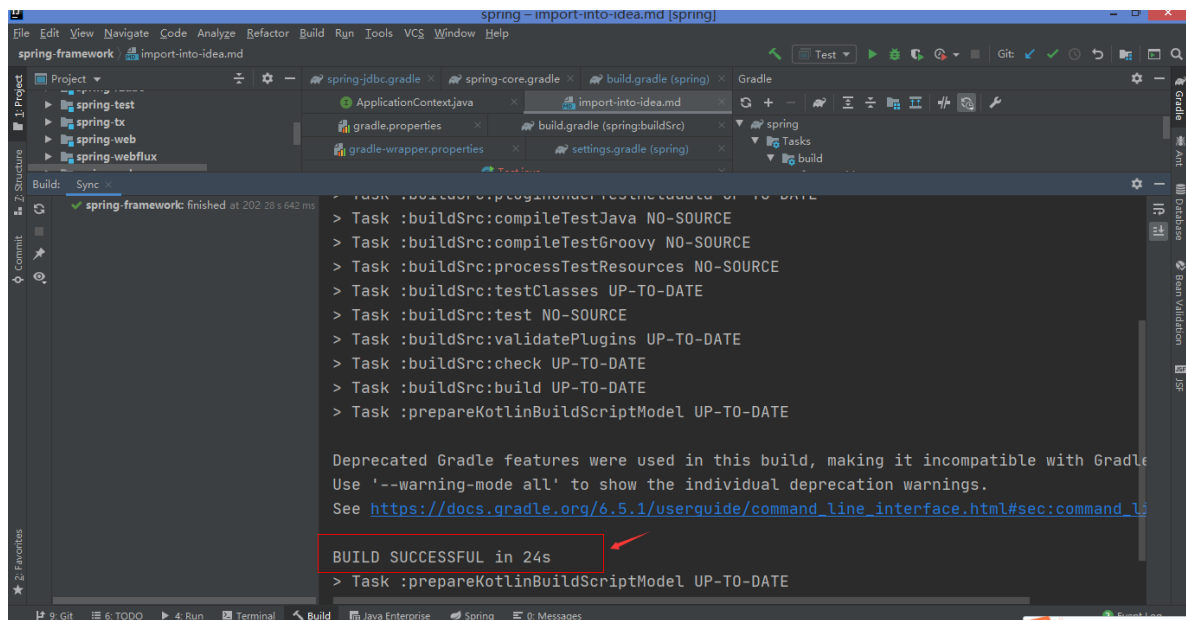


4、都编译完成且成功之后，开始编译整个工程（这个过程非常耗时间，可能10-20分钟！），如下图：

打开顶层spring->build->build



经过一段时间编译，build成功！（每个人电脑的性能不一样，所需时间也不一样）



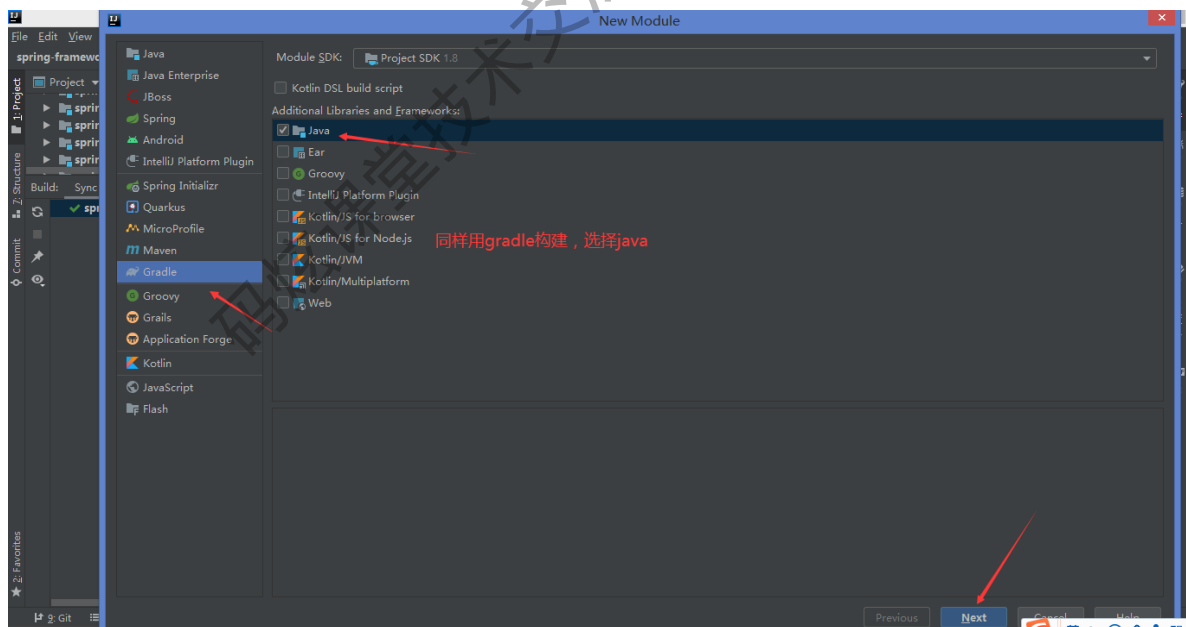
smart哥在编译过程中没有发现问题，非常顺利，但是在下面的测试中发现了问题。

五、源码测试

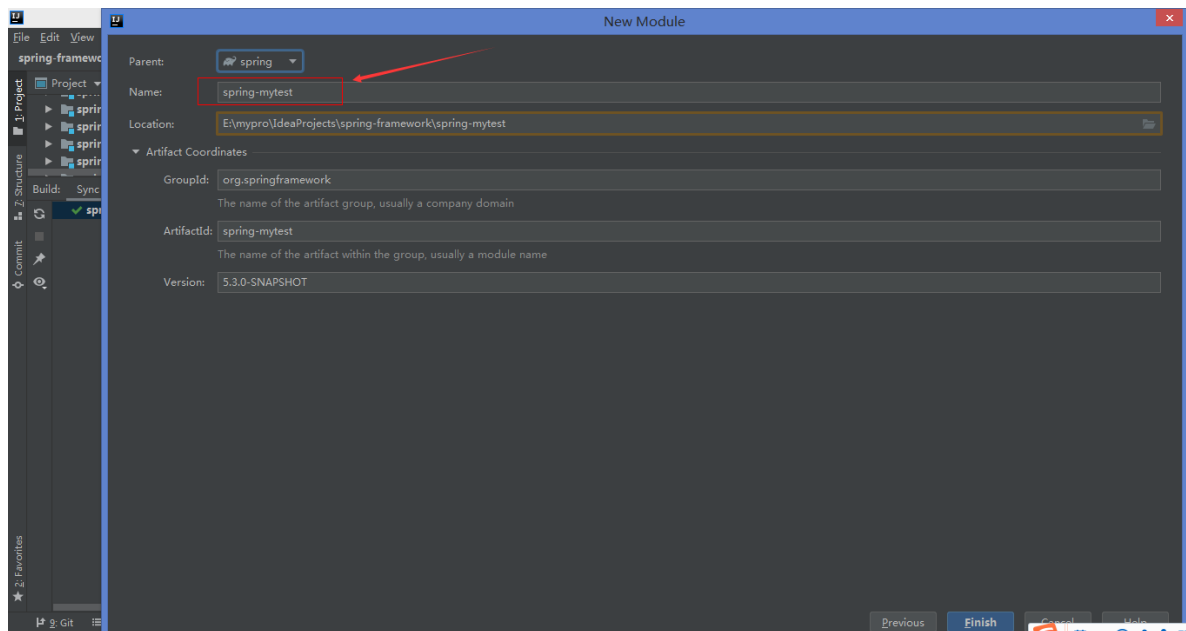
1、完成了上面的过程后，我们可以自己编写一个模块测试该源码构建编译过程是否真正成功完成！

步骤：【File】->【New】->【Module...】

在Spring中添加自己的module模块，同样选择gradle构建。

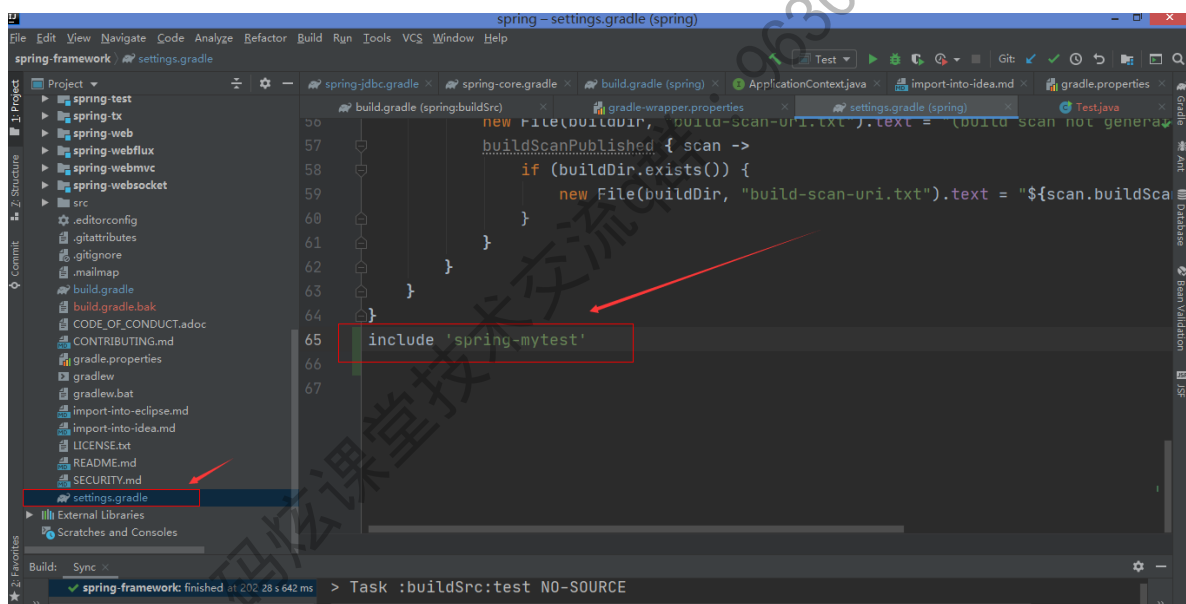


2、新模块Parent：spring，Name：spring-mytest（我们的测试模块作为spring源码的子模块添加进来，下面的配置默认即可）



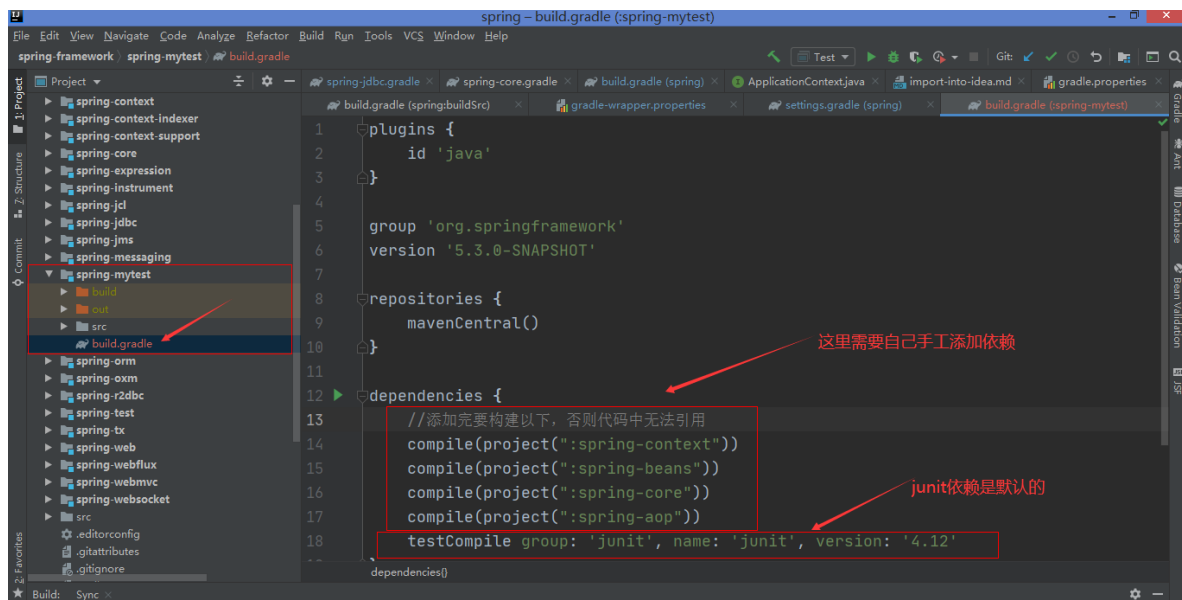
3、Finish，idea会自动帮助我们构建spring-mytest模块

打开全局配置文件：settings.gradle文件，拉到最下面，我们看到系统自动加上了spring-mytest模块。

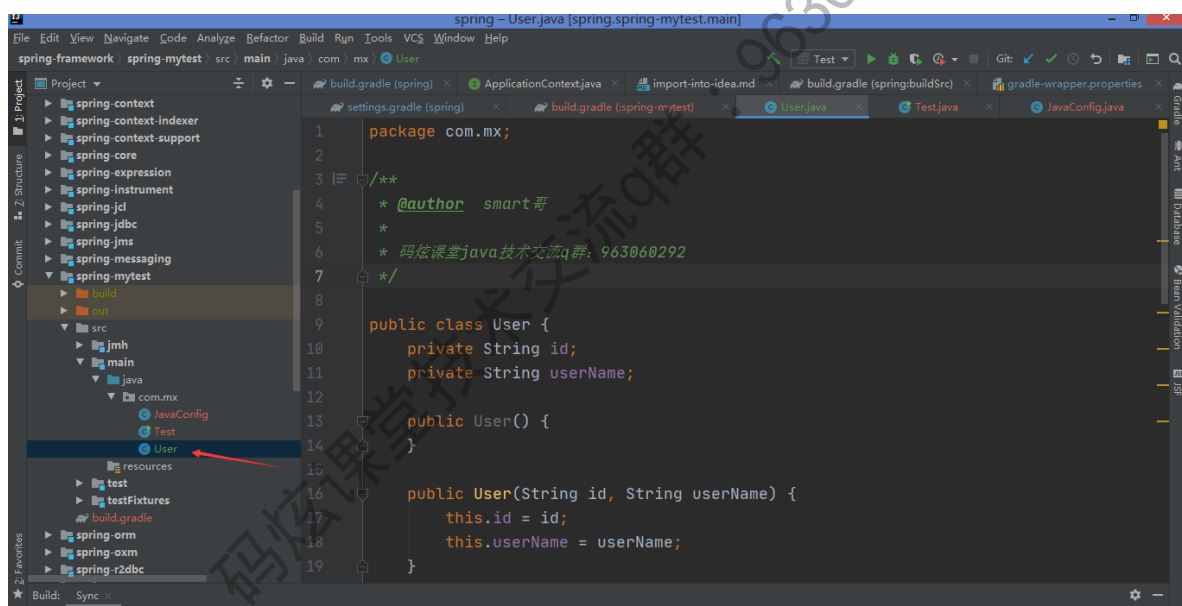


找到我们自己的测试模块spring-mytest，打开build.gradle文件（相当于pom文件），默认dependencies依赖(这里的dependencies和maven里的依赖是一样的)只有一个junit，我们需要手工添加spring-context，spring-beans，spring-core，spring-aop这四个核心模块，具体如下：

```
dependencies {
    //添加完要构建一下，否则代码中无法引用
    compile(project(":spring-context"))
    compile(project(":spring-beans"))
    compile(project(":spring-core"))
    compile(project(":spring-aop"))
    testCompile group: 'junit', name: 'junit', version: '4.12'
}
```



4、下面编写一个简单的applicationContext获取容器用的bean，主要是测试Spring源码构建编译过程是否成功！



新建一个实体类User.java

```
package com.mx;

/**
 * @author smart哥
 *
 * 码炫课堂java技术交流q群: 963060292
 */

public class User {
    private String id;
    private String userName;

    public User() {
    }
}
```

```

public User(String id, String userName) {
    this.id = id;
    this.userName = userName;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}

@Override
public String toString() {
    return "User{" +
        "id='" + id + '\'' +
        ", userName='" + userName + '\'' +
        '}';
}
}

```

新建JavaConfig.java （使用注解的方式声明bean）

```

package com.mx;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

/**
 * @author smart哥
 *
 * 码炫课堂java技术交流q群: 963060292
 */

@Configuration
@ComponentScan
public class JavaConfig {

    @Bean
    public User user(){
        return new User("001", "smart哥");
    }

}

```


最后写一个测试类Test.java

```
package com.mx;

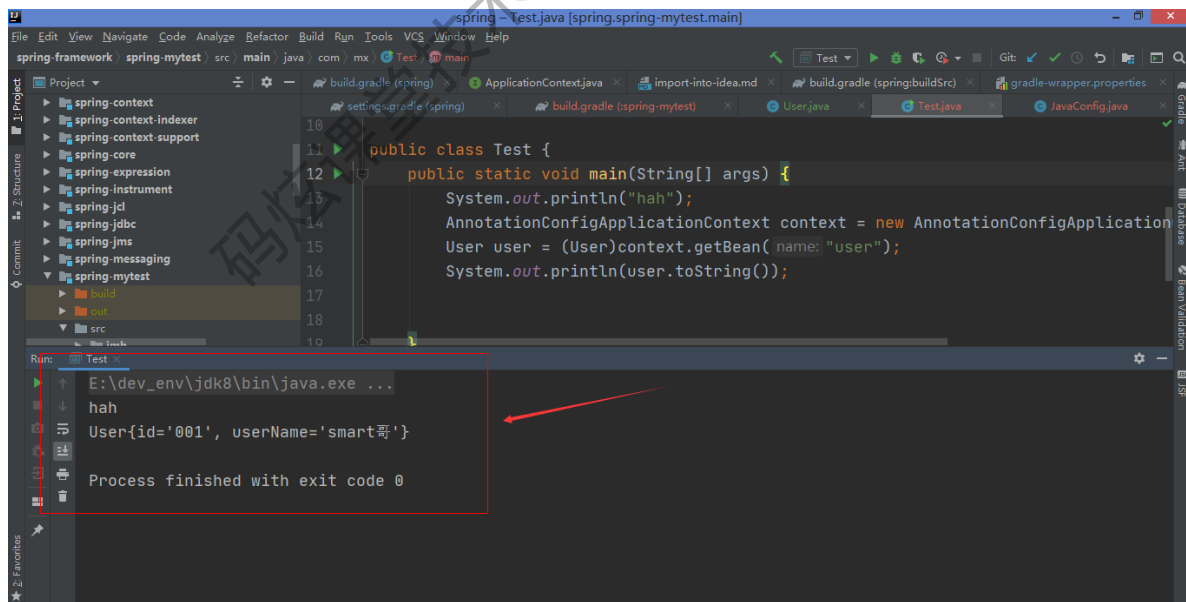
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

/**
 * @author smart哥
 *
 * 码炫课堂java技术交流q群: 963060292
 */

public class Test {
    public static void main(String[] args) {
        System.out.println("hah");
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(JavaConfig.class);
        User user = (User)context.getBean("user");
        System.out.println(user.toString());
    }
}
```

运行, console输出

```
hah
User{id='001', userName='smart哥'}
```



成功! 撒花!!

六、问题及解决方案

1、CoroutinesUtils找不到该类

错误信息:

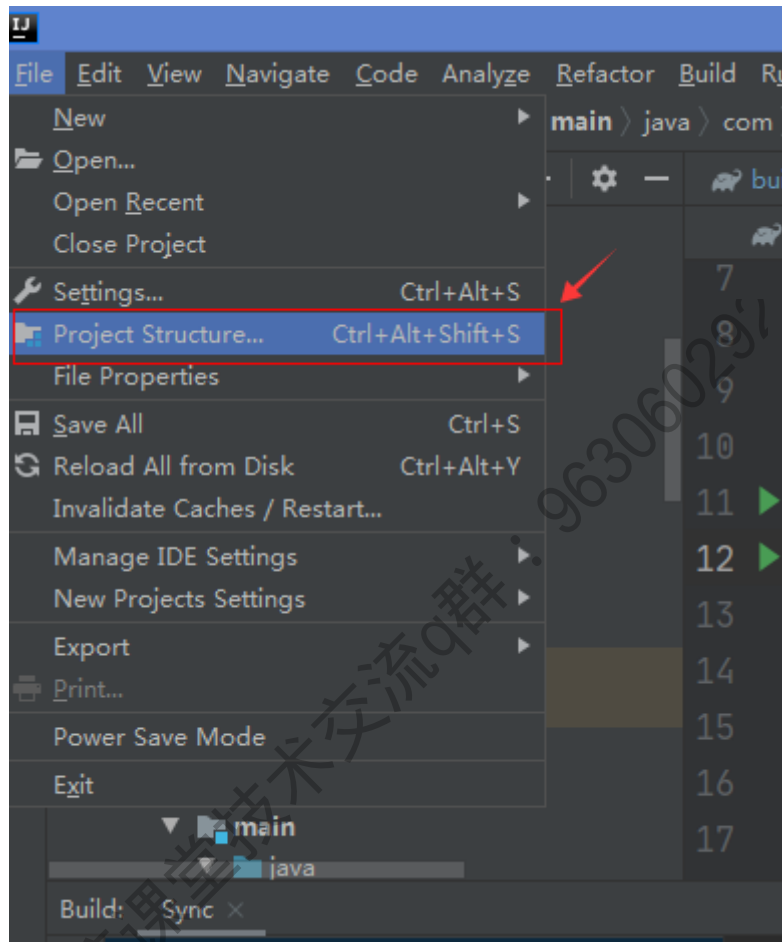
Error:(354, 51) java: 找不到符号

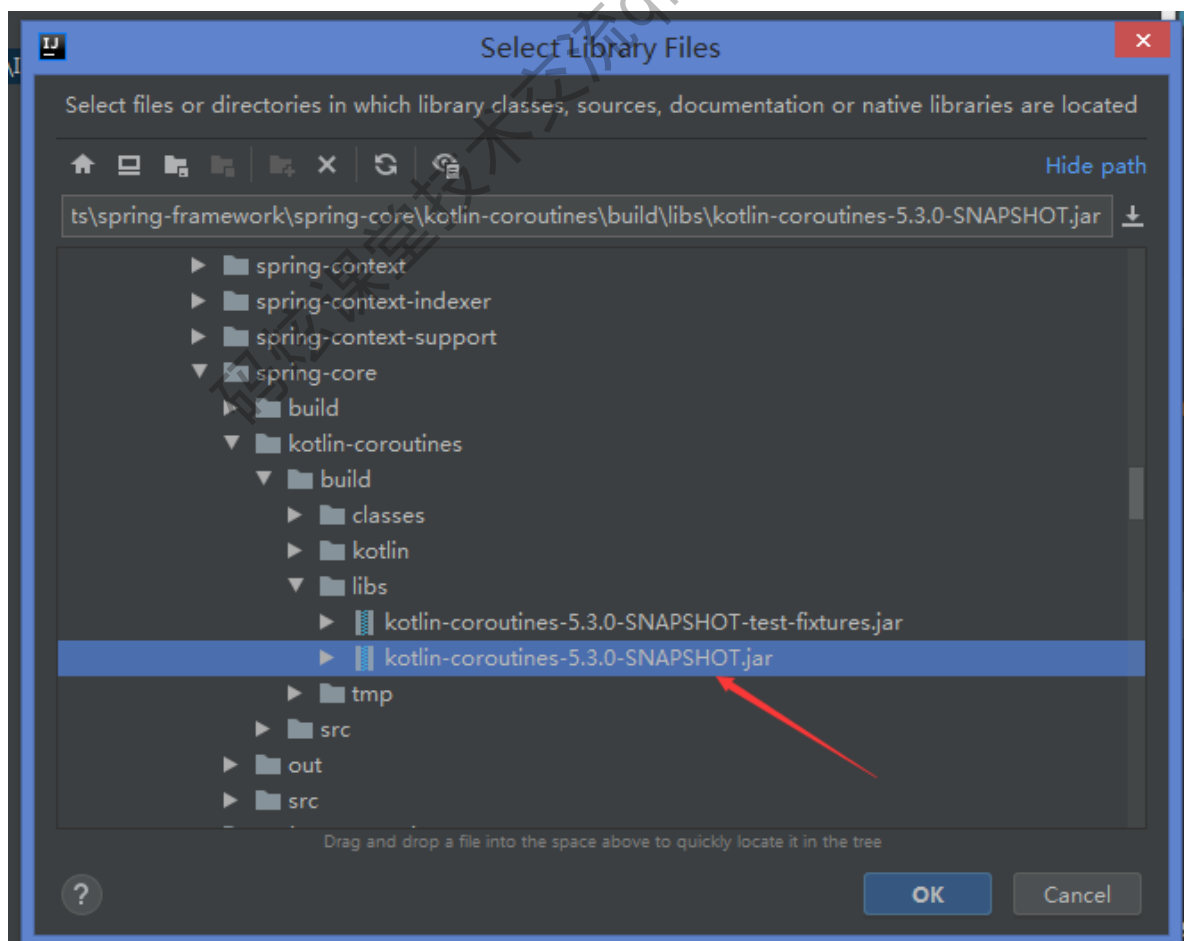
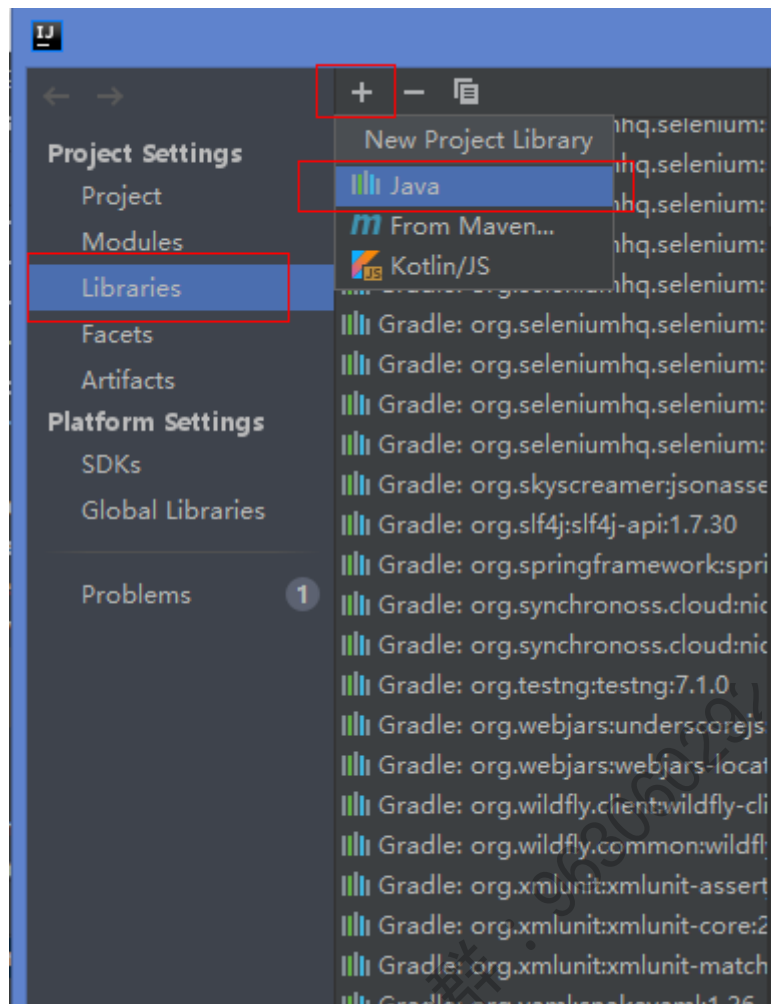
符号: 变量 CoroutinesUtils

位置: 类 org.springframework.core.ReactiveAdapterRegistry.CoroutinesRegistrar

解决方案:

点击【File】->【Project Structure】->【Libraries】->【+】->【Java】, 然后选择spring-framework/spring-core/kotlin-coroutines/build/libs/kotlin-coroutines-5.3.0-SNAPSHOT.jar, 在弹出的对话框中选择spring-core.main, 再重新build项目即可。





2、InstrumentationSavingAgent找不到该类

错误信息：

Error:(26, 38) java: 找不到符号

符号: 类 InstrumentationSavingAgent

位置: 程序包 org.springframework.instrument

解决方案：

修改spring-context模块下的spring-context.gradle文件，找到optional(project(":spring-instrument")), 将optional改为compile

```
//optional改为compile, 否则报错: 找不到InstrumentationSavingAgent
//optional(project(":spring-instrument"))
compile(project(":spring-instrument"))
```

3、H2DatabasePopulatorTests > executesHugeScriptInReasonableTime() FAILED

错误信息：

H2DatabasePopulatorTests > executesHugeScriptInReasonableTime() FAILED

解决方案：

修改spring-jdbc模块下的spring-jdbc.gradle文件，找到optional("com.h2database:h2"), 将optional改成compile

```
//报错: H2DatabasePopulatorTests > executesHugeScriptInReasonableTime() FAILED
//解决方案: 将optional换成compile
//optional("com.h2database:h2")
compile("com.h2database:h2")
```

4、header.mismatch [SpringHeader]

该错误是修改完以上3个问题后重新对整个工程进行重新编译时报的错，这个错是我们新建的spring-mytest模块报的错。

这个错误其实无关紧要，是格式错误，所以可以忽略，不要管它，也用不着重新编译，我们可以直接执行Test.java中的main方法。

如果某些小伙伴有强迫症，非要全部编译成功才肯罢休，那么可以尝试修改全局配置文件，在编译的时候把spring-mytest模块剔除在外。smart哥没有尝试，感兴趣的可以试下。

小伙伴们如果在编译，调试过程中遇到相关问题，可以在[码炫课堂java技术交流q群：963060292](#)中找我交流，共同探讨、研究。

附：spring源代码各个模块作用

主要模块：

spring-core:核心模块 依赖注入IOC和DI的最基本实现
spring-beans:Bean工厂与装配
spring-context:上下文，即IOC容器
spring-context-support:对IOC的扩展，以及IOC子容器
spring-context-indexer:类管理组件和Classpath扫描
spring-expression:表达式语句

切面编程：

spring-aop:面向切面编程，CGLIB,JDKProxy
spring-aspects:集成AspectJ，Aop应用框架
spring-instrument:动态Class Loading模块

数据访问与集成：

spring-jdbc:提供JDBC主要实现模块，用于简化JDBC操作
spring-tx:spring-jdbc事务管理
spring-orm:主要集成Hibernate,jpa,jdo等
spring-oxm:将java对象映射成xml数据或将xml映射为java对象
spring-jms:发送和接受消息

web组件：

spring-web:提供了最基础的web支持，主要建立在核心容器上
spring-webmvc:实现了spring mvc的web应用
spring-websocket:主要与前端页的全双工通讯协议
spring-webflux:一个新的非阻塞函数式Reactive Web框架

报文：

spring-messaging:4.0加入的模块，主要集成基础报文传送应用

测试：

spring-test：测试组件

集成兼容：

spring-framework-bom:解决不同模块依赖版本不同问题

结语

smart哥首创**3位1体**学习法之--**源码篇**-最新【**spring5.3.x源码解析**】课程即将开启，全盘解析spring5.3.x的底层源码，速来学习，机不可失，时不再来！

额~~群在哪？？

码炫课堂java技术交流Q群：963060292