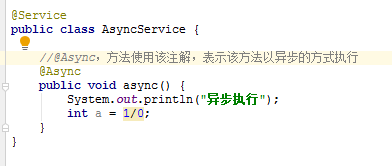
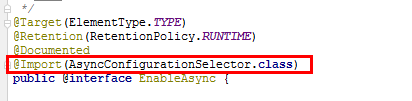
Sprinboot 异步框架加载流程

使用@EnableAsync注解开启异步功能，然后在业务中使用了@Async注解的方法在执行时，就是以异步的方式执行的，也就是会开启一个新线程去执行我们的业务方法。代码如下：

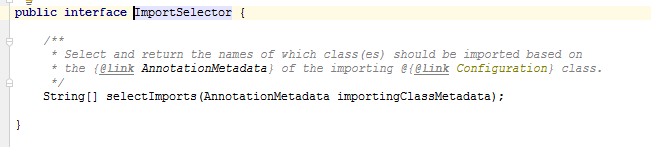


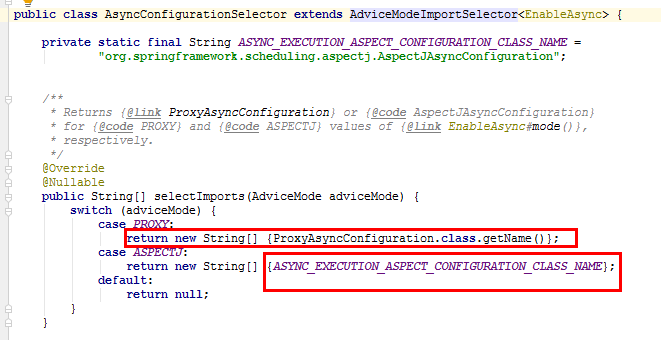


下面就是以@EnableAsync注解入手，看其具体做了哪些工作



可以看到，该注解导入了AsyncConfigurationSelector这个类，该类实现ImportSelector接口，AsyncConfigurationSelector被注入到spring容器中后，spring容器会调用ImportSelector接口的selectImports()方法，该方法会返回一个Class的完全限定名的数组，然后spring容器会将这个数组中的Class注入到容器中，代码如下：



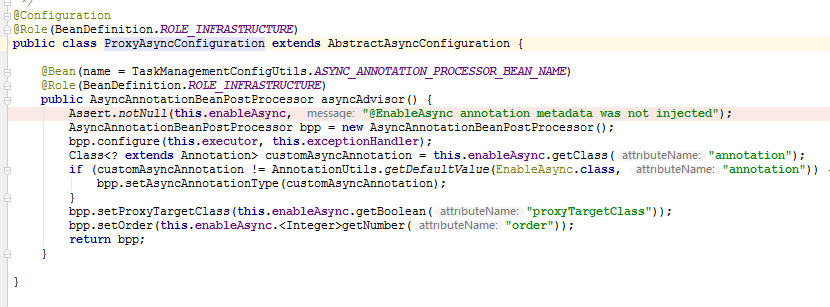


根据@EnableAsync注解的mode()方法的值，来判断是注入ProxyAsyncConfiguration

还是**AspectJAsyncConfiguration，默认是**ProxyAsyncConfiguration，@EnableAsync代码如下

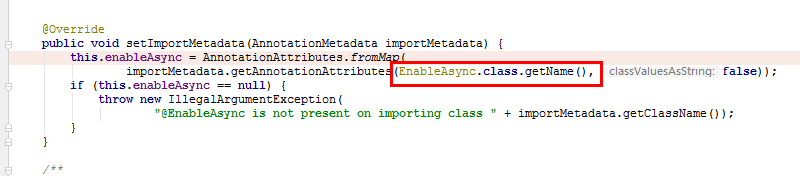


ProxyAsyncConfiguration代码如下：

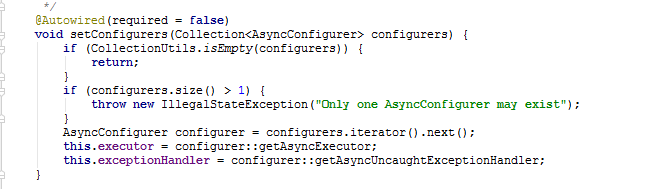


ProxyAsyncConfiguration是一个配置类，会往spring容器中注入AsyncAnnotationBeanPostProcessor这个后置处理器。后置处理器一会在说。

ProxyAsyncConfiguration继承AbstractAsyncConfiguration，AbstractAsyncConfiguration实现ImportAware接口，ImportAware接口作用是获取所有注解信息，然后根据注解信息获取我们想要的信息。AbstractAsyncConfiguration是获取了@EnableAsync注解的信息，代码如下：



AbstractAsyncConfiguration也会注入AsyncConfigurer如果有的话。其作用就是获取我们的异步执行的线程池管理类，和异常处理类。默认是非自动配置的，需要我们手动提供。代码如下：



下面我们来看重点AsyncAnnotationBeanPostProcessor这个后置处理器，后置处理器的作用如下：  
1：后置处理器的postProcessorBeforeInitailization方法是在bean实例化，依赖注入之后及自定义初始化方法(例如：配置文件中bean标签添加init-method属性指定Java类中初始化方法、  
@PostConstruct注解指定初始化方法，Java类实现InitailztingBean接口)之前调用  
2：后置处理器的postProcessorAfterInitailization方法是在bean实例化、依赖注入及自定义初始化方法之后调用

AsyncAnnotationBeanPostProcessor继承了AbstractAdvisingBeanPostProcessor这个类，bean实例化完后就会调用postProcessAfterInitialization方法，代码如下：

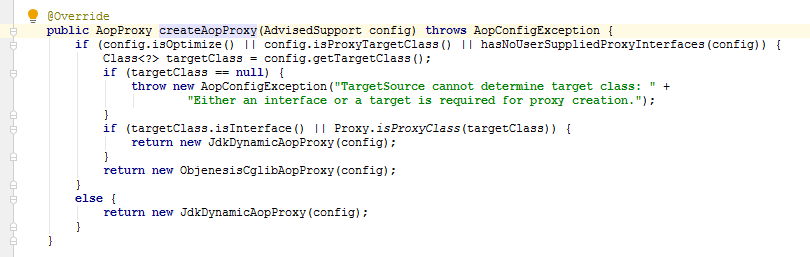


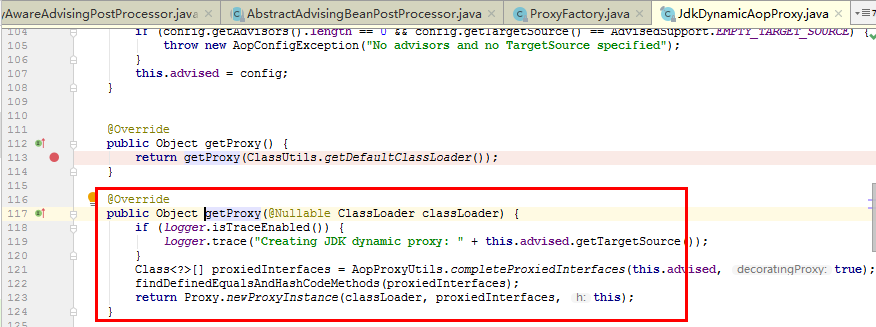
大致流程就是先判断这个bean是否是Advised，如果是，则再判断advised是否被冻结，然后判断bean的方法使用有使用@Async注解。如果没有被冻结，且方法上存在@Async注解，则往bean中添加一个advidor，并将其级别设置为最高。

如果bean不是advised，则调用prepareProxyFactory()方法生成JdkDynamicAopProxy

或者ObjenesisCglibAopProxy，根据bean的class是否实现了接口，如果实现了接口则是JdkDynamicAopProxy（jdk动态代理），否则ObjenesisCglibAopProxy（cglib动态代理）

代码如下：



最后调用JdkDynamicAopProxy或者ObjenesisCglibAopProxy的getProxy方法生成bean的代理类，然后注入到容器中。代码如下：

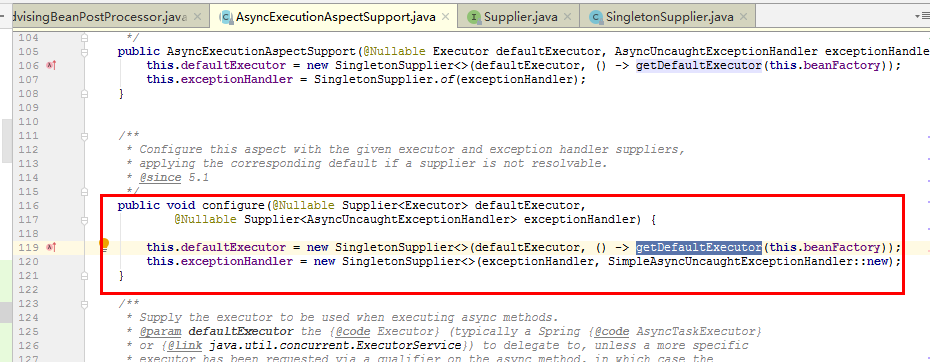
然后我们业务代码自动注入的实例则是这个代理类：

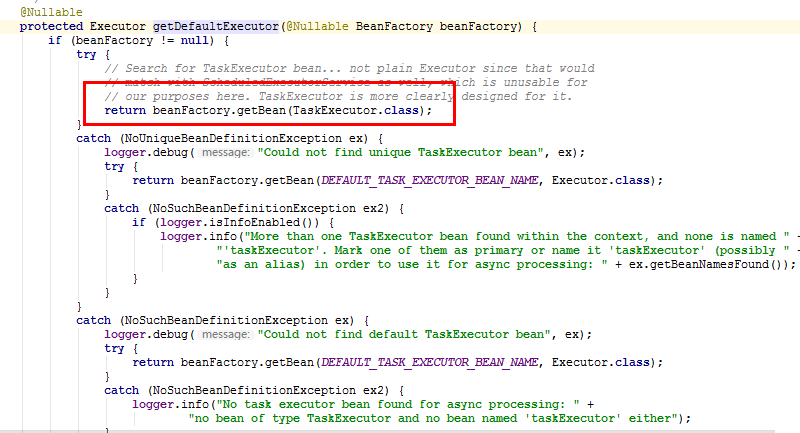


我们在执行业务方法时，就会调到JdkDynamicAopProxy，ObjenesisCglibAopProxy的invoke方法中，最终调用到AsyncExecutionInterceptor的invoke方法进行异步执行代码如下：

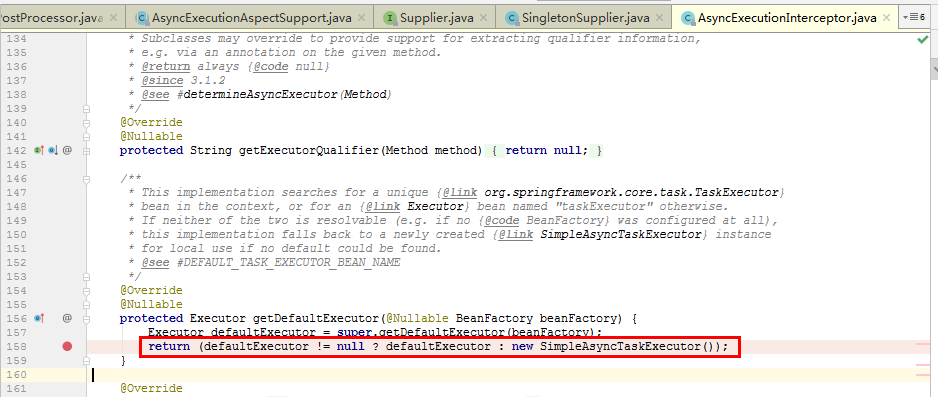


线程池，异常处理获取：

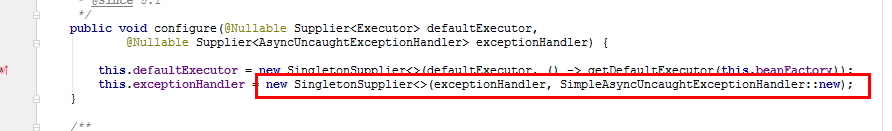




线程池在beanFactory中获取，由于开启自动配置了，这里可以获取到，如果获取不到则创建一个简单的线程池，代码如下：



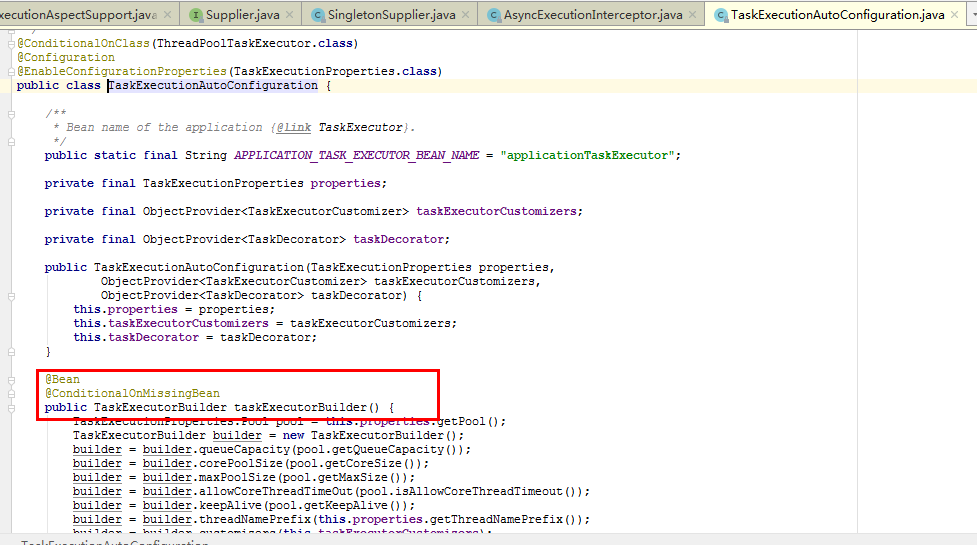
异常处理，默认提供一个简单的处理器，代码如下：



获取规则是有自定义的就使用自定义的，否则使用默认提供的。

线程池管理自动配置地方TaskExecutionAutoConfiguration

：



线程池核心线程数默认为8