# 启动加载类

@Service，@Mapper等这些注解在项目启动时将对应的类注入到Spring容器中.

# 启动执行方法

其实从依赖注入的字面意思就可以知道，要将对象p注入到对象a，那么首先就必须得生成对象p与对象a，才能执行注入。所以，如果一个类A中有个成员变量p被@Autowired注解，那么@Autowired注入是发生在A的构造方法执行完之后的。

如果想在生成对象时候完成某些初始化操作，而偏偏这些初始化操作又依赖于依赖注入，那么就无法在构造函数中实现。为此，可以使用@PostConstruct注解一个方法来完成初始化，@PostConstruct注解的方法将会在依赖注入完成后被自动调用。

@PostConstruct依赖注入完成后被自动调用.

@PreDestroy销毁时执行的方法.

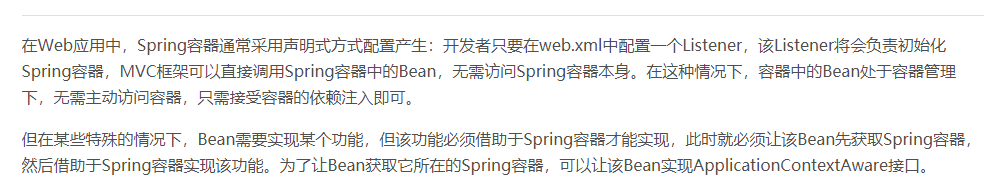
这种注解用在方法上,项目启动时就会执行该方法.

 afterPropertiesSet()方法是因为类实现了InitializingBean接口,所以该方法也会执行.

# Aware

Aware接口就是为了感知自身的一些属性，比如实现了ApplicationContextAware接口的类，能够获取到ApplicationContext对象，实现了BeanFactoryAware接口的类，能够获取BeanFactory对象。

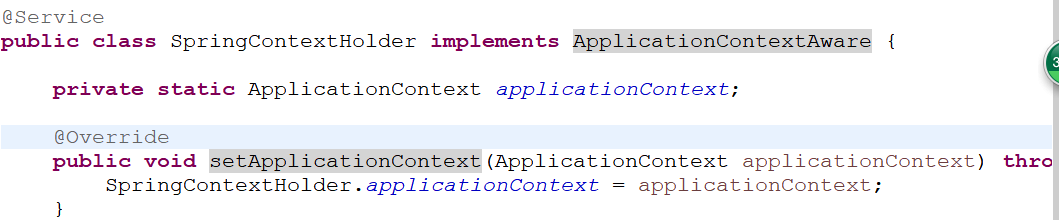
# ApplicationContextAware接口的作用



为了让Bean获取他所在的Spring容器,就让该Bean实现ApplicationContextAware接口.这里的Bean可以是一个@Service注解的类.

Spring容器会检测容器中的所有Bean，如果发现某个Bean实现了ApplicationContextAware接口，Spring容器会在创建该Bean之后，自动调用该Bean的setApplicationContext()方法，调用该方法时，会将容器本身作为参数传给该方法——该方法中的实现部分将Spring传入的参数（容器本身）赋给该类对象的applicationContext实例变量，因此接下来可以通过该applicationContext实例变量来访问容器本身。

上面红色部分就是说:将Spring容器作为参数给了*applicationContext*

**

这样applicationContext就有了Spring容器对象.

具体代码:

|  |
| --- |
| package com.thunisoft.jy.exchange.process.common;  import org.springframework.beans.BeansException;  import org.springframework.context.ApplicationContext;  import org.springframework.context.ApplicationContextAware;  import org.springframework.stereotype.Service;  @Service  public class SpringContextHolder implements ApplicationContextAware {    private static ApplicationContext applicationContext;  @Override  public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  SpringContextHolder.applicationContext = applicationContext;  }  public static ApplicationContext getApplicationContext() {  assertApplicationContext();  return applicationContext;  }  @SuppressWarnings("unchecked")  public static <T> T getBean(String beanName) {  assertApplicationContext();  return (T) applicationContext.getBean(beanName);  }  public static <T> T getBean(Class<T> requiredType) {  assertApplicationContext();  return applicationContext.getBean(requiredType);  }  private static void assertApplicationContext() {  if (SpringContextHolder.applicationContext == null) {  throw new RuntimeException("applicaitonContext属性为null,请检查是否注入了SpringContextHolder!");  }  }  } |

此时在任何地方都可以通过这个类中的getBean方法去Spring容器中获取对象.

例如:获取SourceService对象.

|  |
| --- |
| 8be89e69ef4244879cab7e553a5ae7db |

# ApplicationContext

这个就是Spring容器.

# Spring Bean 初始化/销毁的三种方式

前提是这些类要有@Service等注解，首先要让Spring扫描到该类才有后续的操作。

1. 通过实现 InitializingBean/DisposableBean 接口来定制初始化之后/销毁之前的操作方法；  
   2.通过 <bean> 元素的 init-method/destroy-method属性指定初始化之后 /销毁之前调用的操作方法；  
   3.在指定方法上加上@PostConstruct 或@PreDestroy注解来制定该方法是在初始化之后还是销毁之前调用。

# BeanFactoryAware

这里和ApplicationContextAware是同理的。