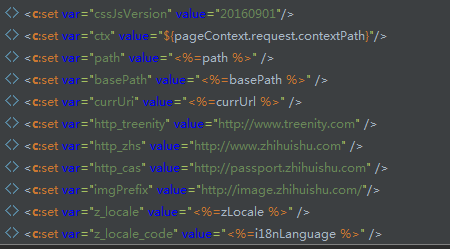
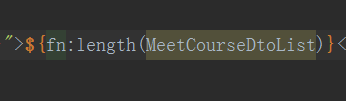
<%@**taglib prefix**="c" **uri**="http://java.sun.com/jsp/jstl/core" %>

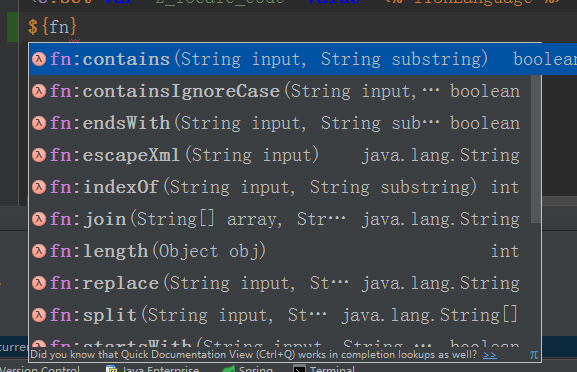


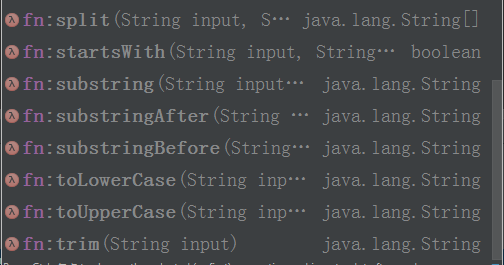




<%@**taglib prefix**="fn" **uri**="http://java.sun.com/jsp/jstl/functions" %>





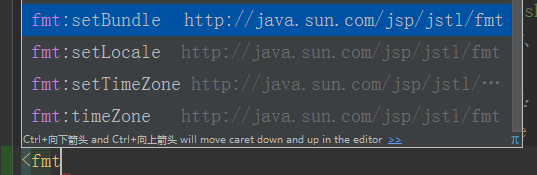


使用fmt 对时间进行格式化的时候需要引入如下

<%@**taglib prefix**="fmt" **uri**="http://java.sun.com/jsp/jstl/fmt" %>

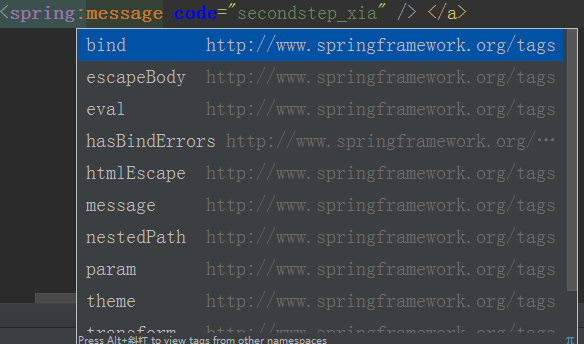
1. **<input** type="text" class="easyui-my97" id="replyTime"
2. value="<fmt:formatDate value="${rb.replyTime}" pattern="yyyy-MM-dd"**/>**"
3. name="replyTime"
4. data-options="disabled:false,startDate:'%y-%M-%d',dateFmt:'yyyy-MM-dd'"**>**

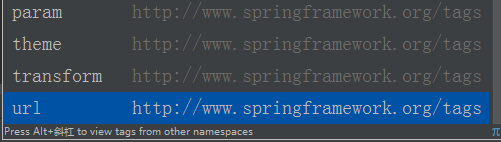




<%@**taglib prefix**="spring" **uri**="http://www.springframework.org/tags" %>

<**spring:message code**="secondstep\_xia" />





xmlns:tx="http://www.springframework.org/schema/tx"

http://www.springframework.org/schema/tx   
http://www.springframework.org/schema/tx/spring-tx.xsd

<tx:annotation-driven />

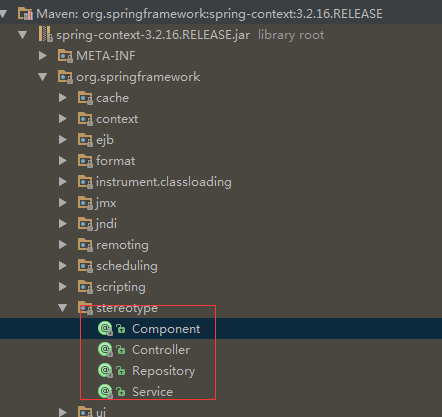
读取资源文件配置

<context:property-placeholder location="classpath:jdbc.properties,classpath:config.properties  
 ,http://conf.zhihuishu.com/redis.uri.properties" />

可以直接读取外部url

<context:component-scan base-package="com.zhihuishu.micro" />

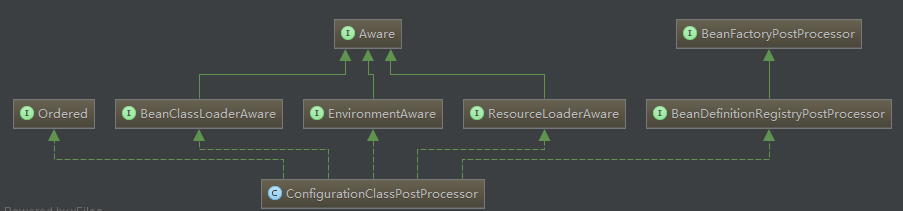
注册以下注解配置了的bean

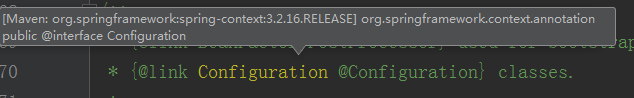


<context:annotation-config />

定义了如下bean

ConfigurationClassPostProcessor.class

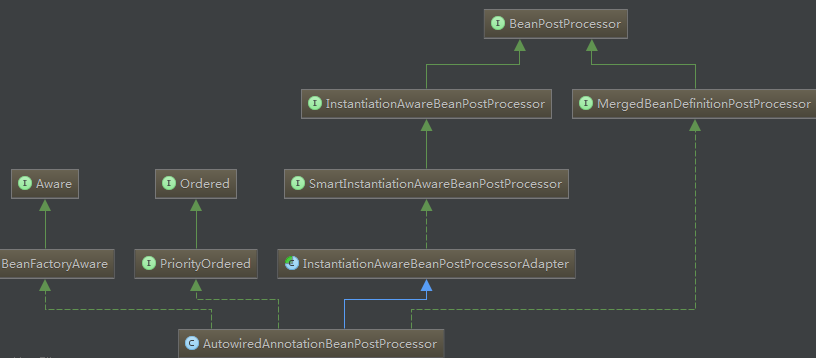




处理@ Configuration注解的，<context:component-scan/>

也定义了该BeanFactoryPostProcessor

AutowiredAnnotationBeanPostProcessor.class



处理 Autowired @Autowired ，Value @Value 注解

同时支持 JSR-330's avax.inject.Inject @Inject，作为@Autowired的替代。

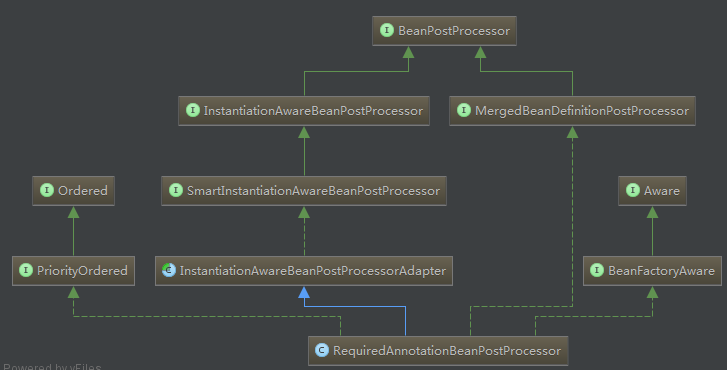
Autowired可被用于构造函数上，不过只能有一个的required被set true，其它注解了的是候选项，在spring做选择时选择一个能提供最多依赖的构造函数执行

Autowired标注构造函数和字段不必为public

<context:component-scan/>也定义了该BeanPostProcessor

注解优先于xml执行，所以后者能覆盖前者

RequiredAnnotationBeanPostProcessor.class



处理 Required @Required (org.springframework.beans.factory.annotation)

<context:component-scan/>也定义了该BeanPostProcessor

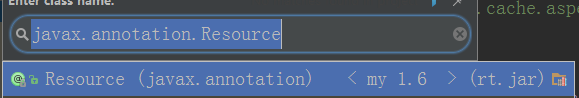
1. @Required
2. **public** **void** setProduct(Product product) {
3. **this**.product = product;
4. }

如果任何带有@Required的属性未设置的话 将会抛出BeanInitializationException异常

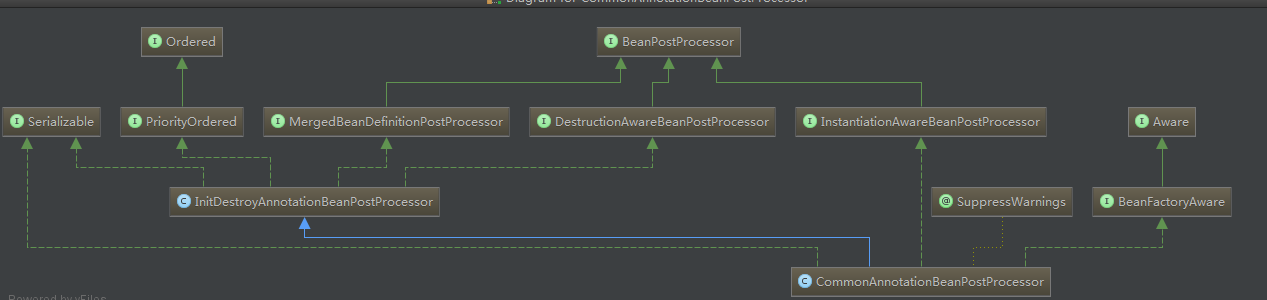
when *jsr250Present(*

ClassUtils.*isPresent*("javax.annotation.Resource", AnnotationConfigUtils.class.getClassLoader());

*)*



CommonAnnotationBeanPostProcessor.class



主要是支持javax.annotation包中的JSR-250注解

javax.annotation.PostConstruct

javax.annotation.PreDestroy

javax.annotation.Resource

javax.xml.ws.WebServiceRef（跟Resource类似，用来创建特殊的JAX-WS service endpoints）

EJB 3 javax.ejb.EJB （跟Resource类似，但能在获取失败时查询global JNDI name，目标bean可以是POJO或者EJB Session Bean）

JDK1.6开箱即用，1.5需要JSR-250 api jar。还有单独的JAX-WS

EJB3 API jar.

For direct JNDI access 查询空间"java:comp/env/"

需要配置property name="alwaysUseJndiLookup" value="true"

包括全局JNDI命名（including "java:" prefix）

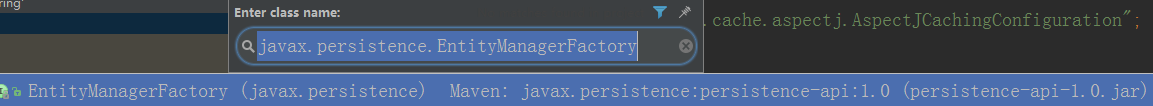
<context:component-scan/>也定义了该BeanPostProcessor

注解优先于xml执行，所以后者能覆盖前者

when *jpaPresent(*

ClassUtils.*isPresent*("javax.persistence.EntityManagerFactory", AnnotationConfigUtils.class.getClassLoader()) &&  
ClassUtils.*isPresent*(*"*org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor*"*, AnnotationConfigUtils.class.getClassLoader());

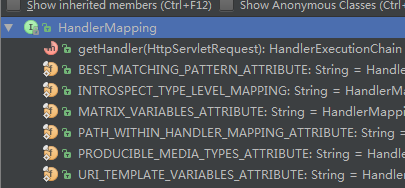
*)*



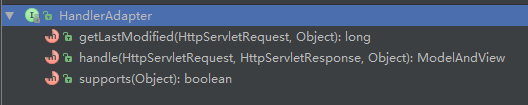
org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor class

<mvc:annotation-driven/>

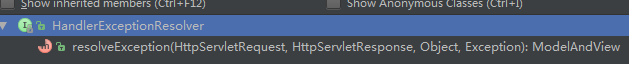
注册HandlerMapping



HandlerAdapter



HandlerExceptionResolver



HandlerMapping包括：

RequestMappingHandlerMapping（mapping requests to annotated controller methods.）

BeanNameUrlHandlerMapping（map URL paths to controller bean names.）

HandlerAdapter包括：

RequestMappingHandlerAdapter（processing requests with annotated controller methods）

HttpRequestHandlerAdapter（processing requests with {@link HttpRequestHandler}）

SimpleControllerHandlerAdapter（processing requests with interface-based {@link Controller}s）

HandlerExceptionResolver包括：

ExceptionHandlerExceptionResolver（handling exceptions  
through @{@link ExceptionHandler} methods）

ResponseStatusExceptionResolver（exceptions annotated  
with @{@link ResponseStatus}）

DefaultHandlerExceptionResolver（resolving known Spring  
 exception types）

