目录

[起步 1](#_Toc7835)

[配置文件 1](#_Toc4098)

[applicationContext-eventgateway.xml 2](#_Toc8802)

[BeanPostProcessor 2](#_Toc17976)

[cglib 3](#_Toc24009)

[applicationContext.xml 3](#_Toc4621)

# 起步

目的是想看懂领导给的中间件项目，最终可以凭借自己的记忆画出架构图，说出核心逻辑，亲手实现一个，感觉这个过程会比较坎坷，所以存在被记录的价值。

说是这么说，如何起标题我是一点头绪都没有的，只能想到拿写到拿。

项目不是凭空看的，至少要知道这个项目的用途，涉及哪些框架，接口。最好还是有个牛逼哄哄的人能带着走一遍，我很幸运，peter给我画过流程图。

针对此中间件项目，核心的就是Spring和JMS（这个项目挺老的了，都没用maven打包）。

通过消息队列收发消息，这个中间件项目应该是充当消费者。消息类型多种多样，如何在收到消息后不通过if-else，快速处理该消息，是难点所在。

下面开始看项目。

# 配置文件

但凡java web项目，肯定有web.xml；但凡用SpringMVC的项目必定通过web.xml来管理自己的配置文件，因为这样方便。

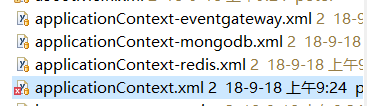
<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath\*:/applicationContext\*.xml</param-value>

</context-param>

resource文件夹下共有这么几个配置文件



MongoDB与Redis，他们属于数据库。这两个配置文件内也确实只记录了连接相关的参数，可以忽略。

## applicationContext-eventgateway.xml

这个配置文件只初始化了一个bean

<bean id=*"EventHandlerListener"* class=*"com.bsmart.eventgateway.core.listener.EventHandlerListener"*></bean>

**public** **class** EventHandlerListener **implements** BeanPostProcessor {

**private** Logger log = LoggerFactory.*getLogger*(**this**.getClass());

@Override

**public** Object postProcessAfterInitialization(Object bean, String beanName) **throws** BeansException {

EventHandlerAnnotation annotation = bean.getClass().getAnnotation(EventHandlerAnnotation.**class**);

String className = bean.getClass().getName();

**if** (className.indexOf("$$EnhancerByCGLIB$$") > -1) {

**try** {

annotation = Class.*forName*(className.substring(0, className.indexOf("$$"))).getAnnotation(EventHandlerAnnotation.**class**);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**if** (annotation != **null**) {

String[] msgTypes = annotation.value();

String msgTypeHex = StringUtils.*arraysToStr*(BatMessage.***MSGTYPE\_SEPARATE***, msgTypes);

EventHandlerFactory.*MSG\_EVENT\_MAP*.put(msgTypeHex, (BaseA500EventHandler) bean);

log.debug("msgTypeHex: " + msgTypeHex + " beanName: " + beanName + " class: " + bean);

}

**return** bean;

}

@Override

**public** Object postProcessBeforeInitialization(Object bean, String beanName) **throws** BeansException {

**return** bean;

}

}

// 这他妈的，第一行就看不懂。。。

### BeanPostProcessor

BeanPostProcessor是Spring提供的接口，内有两个方法，postProcessBeforeInitialization，postProcessAfterInitialization，就是初始化前做什么，初始化后做什么。

也很神奇，实现BeanPostProcessor接口的类一旦被注册到Spring的容器，Spring便会在初始化实例的时候，自动调用BeanPostProcessor接口的方法。

现在看方法，初始化后的操作是空的，忽略。初始化前，简单讲就是看看这个类有没有EventHandlerAnnotation注解，有的话加到EventHandlerFactory.MSG\_EVENT\_MAP。然后就是我不明白的地方，这个初始化前做的检查操作，给一个监听器有什么用，监听器只初始化一次啊，难不成，这个监听器又在某个地方被配置了，所以任何一个类在初始化前都会引用这个监听器内的方法？

这段代码的意思很好懂，但里头的知识点一点不少。

### cglib

**if** (className.indexOf("$$EnhancerByCGLIB$$") > -1) {

**try** {

annotation = Class.*forName*(className.substring(0, className.indexOf("$$"))).getAnnotation(EventHandlerAnnotation.**class**);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

这一段就很精彩，牵扯到cglib动态代理。如果是靠cglib动态代理生成的类，那么类名里会带上$$EnhancerByCGLIB$$这个字符串。我百度了一下，什么字节码之类的东西都出来了，这是个必看知识点，但我现在实在没心思去研究。

就到这了，已经学了30分钟了，再弹吉他30分钟。

### 2018.10.07

bsmart-framework-1.0.jar

bsmart-bmc-entity-1.0.jar

bsmart-bgate-das-3.1.7.26-jdk1.6.jar

没什么心思，就把这3个jar包反编译了，准备看看里头有什么。

然后越看越晕。

我还只看了bsmart-framework-1.0.jar，从结构上看，这个jar包给我的感觉是一个完整的项目

## applicationContext.xml

这个配置文件的内容很多，但基本是数据库相关的。虽然都是些熟悉的东西，可配置方式却很值得我研究，比三合一那种混乱配置肯定要好。

抛开数据库与事务，剩下的配置有：

<bean id=*"ApplicationEventListener"* class=*"com.bsmart.framework.base.service.ApplicationEventListener"*></bean>

<bean id=*"messageSource"* class=*"org.springframework.context.support.ResourceBundleMessageSource"*>

<property name=*"basenames"*>

<list>

<value>base\_message</value>

<value>project\_message</value>

<value>project\_config</value>

</list>

</property>

<property name=*"useCodeAsDefaultMessage"*>

<value>true</value>

</property>

</bean>

<bean id=*"appConfig"* class=*"com.molisc.config.AppConfig"*>

<property name=*"repoPath"*>

<value>${repository.path}</value>

</property>

<property name=*"userAvatarPath"*>

<value>${user.avatar}</value>

</property>

<property name=*"msgBinaryPath"*>

<value>${msg.binary}</value>

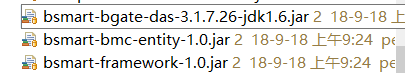
</property>

</bean>

这三项就是我感觉，可能和中间件有关的配置。

此外，配置文件里引用的一些，看着像是项目里的类，却被打在jar包里，并不明白为什么要这么做。

jar包名



类路径

