svip.iocoder.cn /Spring/build-debugging-environment/

精尽 Spring 源码分析 —— 调试环境搭建（Spring 5.1.1 版本）

2019-01-01
Spring

1. 依赖工具

- Gradle
- Git
- JDK1.8+
- IntelliJ IDEA

笔者目前使用的系统版本是 macOS Mojave 10.14 。所以，如果胖友是 Windows 环境，胖到一些问题，请在星球给我留言。

> 懵逼的芮芮：根据现在收到的信息，貌似主要是 Windows 环境会搭建失败。如果胖友真的搭建不起来，建议可以先新建一个项目，搭建一个 Spring Demo 来调试。
>
> 😈 总之，我们的目的是，一定一定一定要调试。酱紫，才能更好的阅读 Spring 的代码。

另外，本文参考官方提供的文档 《import-into-idea》 。

补充说明 1 ：IntelliJ IDEA 请使用 2018 版本，之前有胖友反馈搭建不起来，因为 IDEA 版本过低。

2. 源码拉取

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 `5.1.1.BUILD-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。

拉取完成后，Gradle 会开始自动 **Build** 项目。因为 Build 的过程中，会下载非常多的依赖，请耐心等待。

- 😈 不过笔者有点不太确定，Gradle 是否会自动 **Build** 项目，反正我的会。如果此处碰到问题，请给我留言。

3. 预编译 `spring-oxm` 项目

打开 IDEA Terminal ，输入如下命令，预编译 `spring-oxm` 项目：

```
./gradlew :spring-oxm:compileTestJava
```

当看到 `BUILD SUCCESSFUL` ，说明编译成功。

~~😈 另外，笔者有点不确定，Gradle 在上面已经自动 **Build** 项目，这个步骤是否还需要。但是笔者不熟悉 Gradle 的机制，官方文档又要求这么做，所以做下也没什么影响。哈哈哈哈~~。

感谢【蝴蝶落于指尖】同学，经过测试，这个是必须的操作。

4. 运行示例

在 `spring-context` 项目中的 `src/test/java/example` 目录下，已经提供了一些示例。

① 解析 XML 配置文件成对应的 BeanDefinition 们的流程

可调试 `org.springframework.beans.factory.xml.XmlBeanDefinitionReaderTests` 的 `#withFreshInputStream()` 和 `#withImport()` 这两个单元测试。

相比来说，后者比前者多了一个 `<import />` 标签的解析。当然，XmlBeanDefinitionReaderTests 类中，其它方法也可以简单调试下。看胖友的兴趣哈。

- `#factorySingleton()` 方法，单例的 FactoryBean 的单元测试。

② 加载 Bean 的流程

可调试 `org.springframework.beans.factory.xml.AbstractBeanFactoryTests` 这个单元测试类里的方法。

实际上，AbstractBeanFactoryTests 是一个**抽象**类，所以在运行时，需要选择对应的子类，例如 XmlListableBeanFactoryTests 类。

③ ClassPathXmlApplicationContext 的流程

可调试 `org.springframework.context.support.ClassPathXmlApplicationContextTests` 这个单元测试类里的方法。例如 `#testResourceAndInputStream()` 方法。

④ 解析 Properties 配置文件成对应的 BeanDefinition 们的流程

选读，实际使用非常少。主要目的是为了更深入的理解 BeanDefinitionReader 的设计。

可调试
`org.springframework.beans.factory.support.PropertiesBeanDefinitionReaderTests` 这个单元测试里的方法。

另外，也推荐阅读下 《spring beans源码解读之 – BeanDefinition 解析器》 一文。

⑤ 调试 Spring AOP 相关的流程

参见 《精尽 Spring 源码分析 —— AOP 源码简单导读》

⑥ 调试 Spring Transaction 相关的流程

参见 《精尽 Spring 源码分析 —— Transaction 源码简单导读》

⑦ 调试 Spring MVC 相关的流程

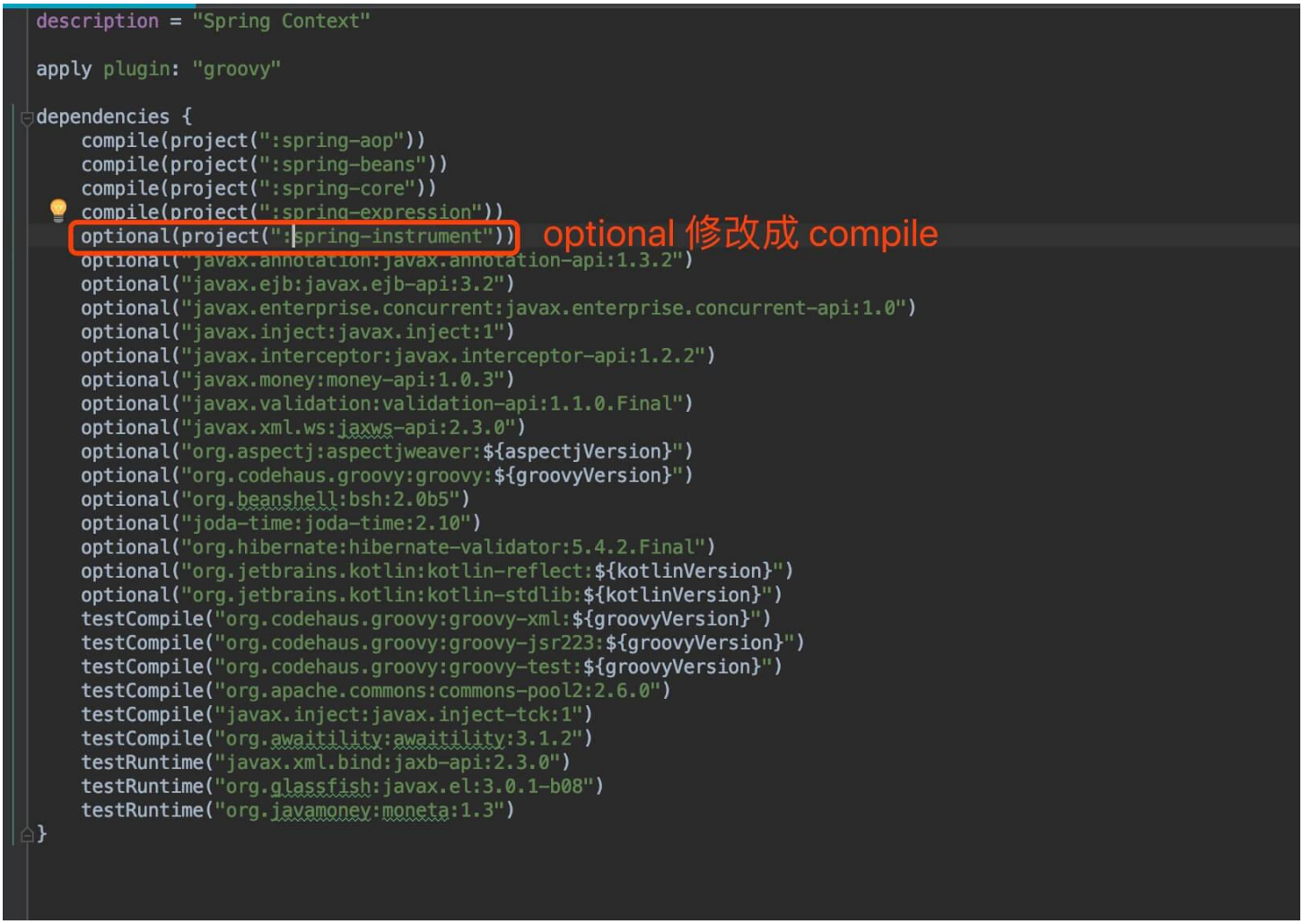
参见 《精尽 Spring MVC 源码分析 —— 调试环境搭建》

⑧ TODO 芋艿，补充一些推荐阅读的示例

5. 可能碰到的问题

5.1 报 InstrumentationSavingAgent 不存在的错误

例如说，在运行 `spring-context` 项目中的单元测试时，会报 InstrumentationSavingAgent 存在的错误。此时，我们将 `spring-context.gradle` 修改如下：



`spring-context.gradle` 修改图

修改完成后，Gradle 又会自动 **Build** 项目，下载相关依赖。完成后，再次运行 `spring-context` 项目中的单元测试，顺利通过。

> 根据胖友【Tomy_Rich】测试的说法，凡是报 XXX 不存在的，就和 InstrumentationSavingAgent 的处理方式一样就可以了。

5.2 报 ‘io.spring.dependency-management’ 插件不存在

可参考 《Plugin [id: ‘io.spring.dependency-management’, version: ‘1.0.5.RELEASE’, apply: false] was not found in any of the following sources:》 一文进行解决。

来自胖友【贾鹤鸣】的提供。

5.3 找不到符号，变量 CoroutinessUtils

可参考 传送门 疑问进行解决。

5.4 其它

如果胖友你在搭建调试环境的过程中，如果碰到任何问题，可以在星球给我留言。

666. 彩蛋

笔者开始更新 Spring 源码解析系列，让我们在 2019 一起**精尽** Spring 。

另外，笔者的好基友小明哥，已经在死磕 Spring 源码，并更新相应的文章。所以对于这个系列，如果小明哥已经写了，会直接进行引用。感谢小明哥的文章授权。美滋滋。

那么，就开始干吧。

重要的友情提示一：

《Spring 源码深度解析》，基于 Spring 3 版本的源码解析。虽然版本有点老，但是内容的流畅性很不错，特别是 Spring IoC 部分。芳芳自己也阅读了一遍，点赞。