《31 讲带你搞懂 SkyWalking》

徐郡明　前搜狗资深技术专家、源码剖析系列畅销书作者

第10讲：深入剖析 Agent 插件原理
无侵入性埋点

课程回顾

- 介绍 SkyWalking Agent 启动流程时
- 介绍了插件中 skywalking-agent.def 文件的查找、解析流程
- AbstractClassEnhancePluginDefine 抽象类的核心定义
- 插件类与 AgentBuilder 配合为目标类动态添加埋点功能的核心流程

本课时深入介绍

AbstractClassEnhancePluginDefine 抽象类以及其子类的运行原理

AbstractClassEnhancePluginDefine 核心实现

AbstractClassEnhancePluginDefine 是所有插件的父类

SkywalkingAgent.Transformer

会通过其 enhanceClass() 方法返回的 ClassMatch 对象，匹配到要增强的目标类

在不同的插件实现类中，enhanceClass() 方法返回的 ClassMatch 对象不同

例如：

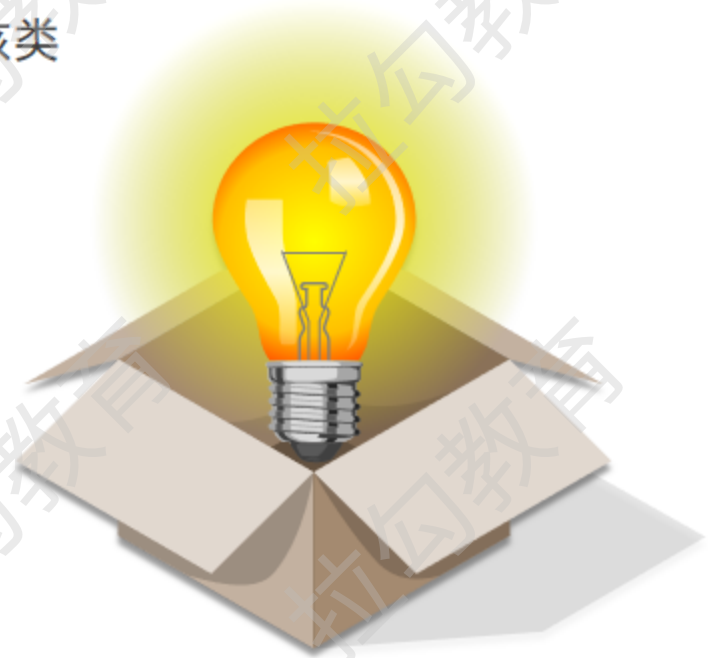
- Dubbo 插件拦截的是 com.alibaba.dubbo.monitor.support.MonitorFilter 这个类
- Tomcat 插件拦截的是 org.apache.catalina.core.StandardHostValve 这个类

AbstractClassEnhancePluginDefine 核心实现

完成目标类和插件类的匹配之后，会进入 define() 方法

其核心逻辑如下：

1. 通过 witnessClass() 方法确定当前插件与当前拦截到的目标类的版本是否匹配
   - 若版本不匹配，则 define() 方法直接结束，当前插件类不会增强该类
   - 若版本匹配，则继续后续逻辑
2. 进入 enhance() 方法执行增强逻辑
3. 设置插件增强标识



witnessClass() 方法

很多开源组件和工具类库的功能会不断增加，架构也会随之重构

导致不同版本的兼容性得不到很好的保证

例如，MySQL 常用的版本有 5.6、5.7、8.0 多个版本

在使用 JDBC 连接 MySQL 时

使用的 mysql-connector-java.jar 包也分为 5.x、6.x、8.x 等版本

对应的 JDBC 协议的版本也各不相同

witnessClass() 方法

mysql-connector-java.jar 版本	类名
5.x	com.mysql.jdbc.ConnectionImpl
8.x	com.mysql.cj.jdbc.ConnectionImpl

witnessClass() 方法

通过一个 SkyWalking Agent 插件完成对一个开源组件所有版本的增强，是非常难实现的

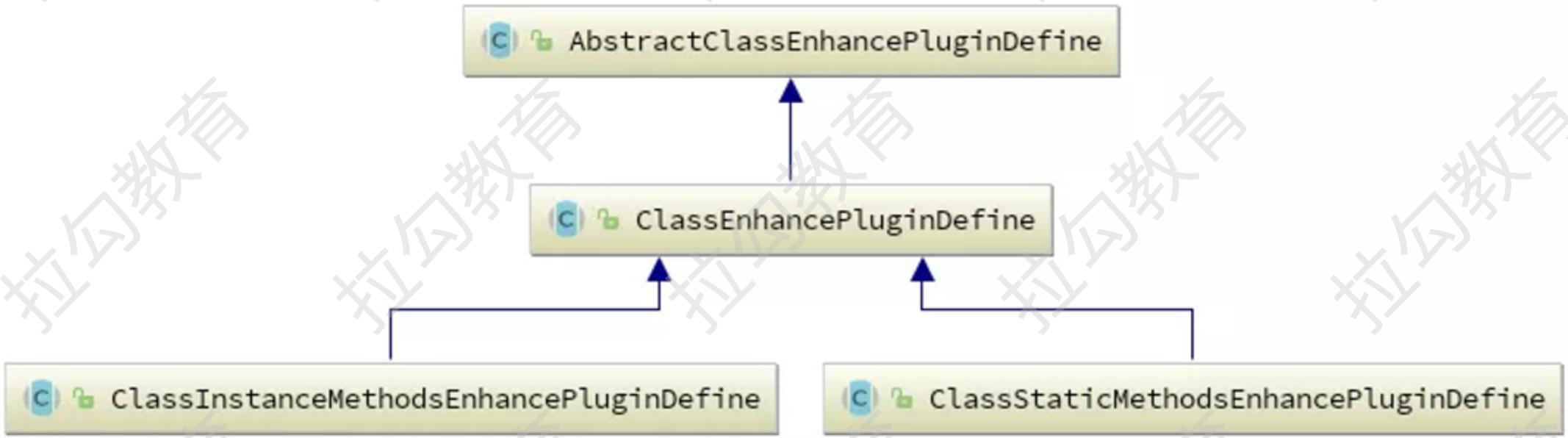
即使勉强能够实现，该插件的实现也会变的非常臃肿，扩展性也会成问题

SkyWalking 怎么解决这个问题呢 ？

witnessClass() 方法

插件版本	witnessClass() 方法返回值	mysql-connector-java.jar 版本
mysql-5.x-plugin	com.mysql.jdbc.ConnectionImpl	5.x
mysql-6.x-plugin	com.mysql.cj.api.MysqlConnection	6.x
mysql-8.x-plugin	com.mysql.cj.interceptors.QueryInterceptor	8.x

增强 static 静态方法



增强 static 静态方法

如果不需要修改方法参数，则会通过 StaticMethodsInter 对象进行增强

其实现与 StaticMethodsInterWithOverrideArgs 类似，唯一区别在于调用目标方法时无法修改参数

上面使用的 StaticMethodsAroundInterceptor 是个接口

其中定义了如下三个方法：

- before()：在目标方法之前调用
- after()：在目标方法之后调用
- handleMethodException()：在目标方法抛出异常时调用

增强 static 静态方法

mysql-8.x-plugin 中的 ConnectionImplCreateInstrumentation 自然也实现了该接口

通过对 StaticMethodsInterWithOverrideArgs 以及 StaticMethodsAroundInterceptor 接口的介绍

发现 Agent 插件对静态方法的增强逻辑与 Spring AOP 中环绕通知的逻辑非常类似

设计模式 TIP

拉勾教育

ClassEnhancePluginDefine 是个典型的模板方法模式的使用场景

其 enhanceClass() 方法只实现了增强静态方法的基本流程

真正的增强逻辑全部通过 getStaticMethodsInterceptPoints() 抽象方法推迟到子类实现

在后面增强对象的构造方法和实例方法时，同样会看到类似的实现

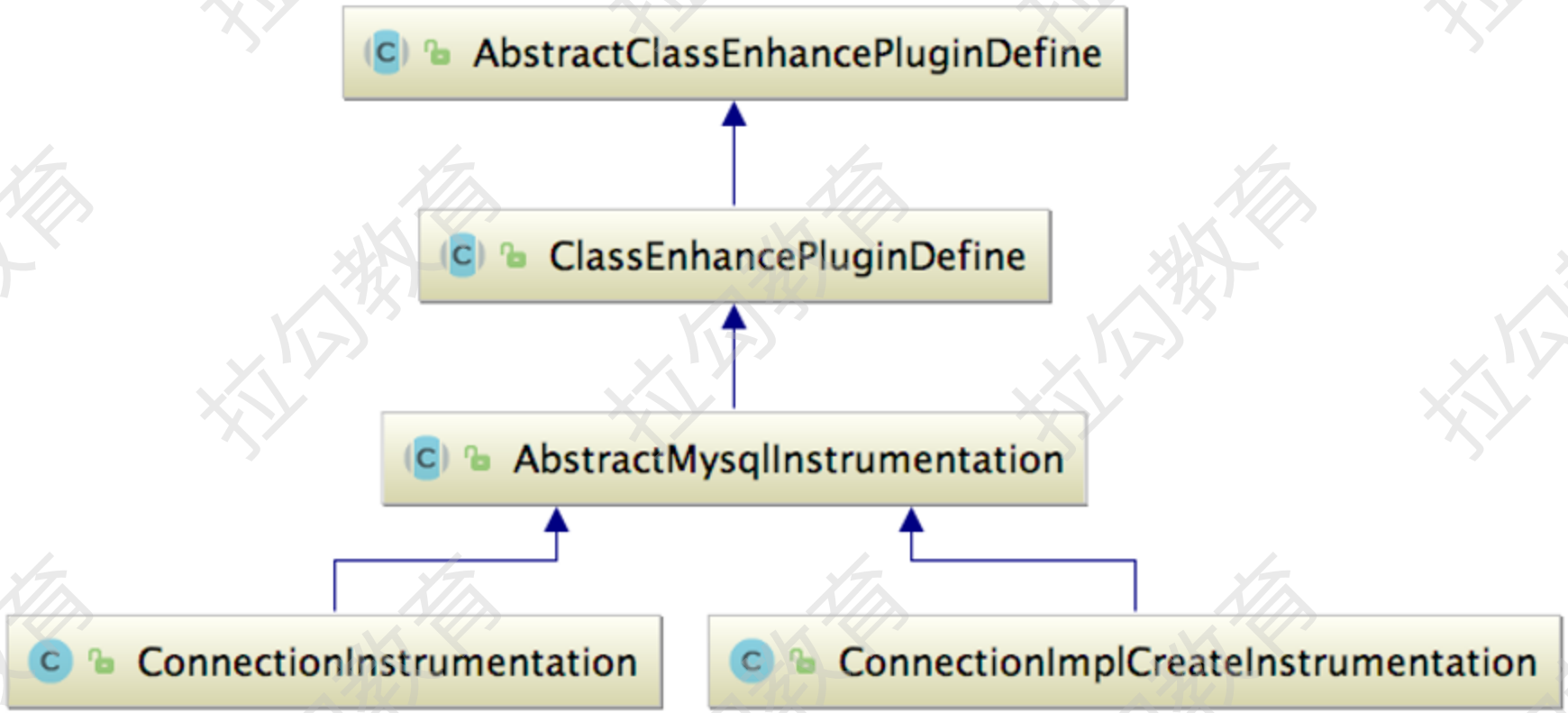
增强实例对象

增强一个 Java 实例对象的相关逻辑 —— 入口是 enhanceInstance() 方法

enhanceInstance() 方法将分成三个部分来分析其实现：

- 实现 EnhancedInstance 接口
- 增强构造方法
- 增强实例方法

AbstractMysqlInstrumentation 抽象类



增强实例方法

增强实例方法	增强静态方法
InstanceMethodsInterceptPoint	StaticMethodsInterceptPoint
InstMethodsInterWithOverrideArgs	StaticMethodsInterWithOverrideArgs
InstMethodsInter	StaticMethodsInter
InstanceMethodsAroundInterceptor	StaticMethodsAroundInterceptor

InterceptorInstanceLoader

以 demo-webapp 为例，其类加载器的结构如下：

总结

本课时深入介绍了 Agent 插件增强目标类的实现，这是 Agent 最核心功能

其中深入分析了增强静态方法、构造方法、实例方法的原理

以及插件如何让目标实例对象实现 EnhanceInstance 接口，如何为目标实例对象添加新字段等

分析的过程中以 mysql-8.x-plugin 插件为例将上述核心逻辑串连起来

Next：第11讲《BootService 核心实现解析》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U