《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

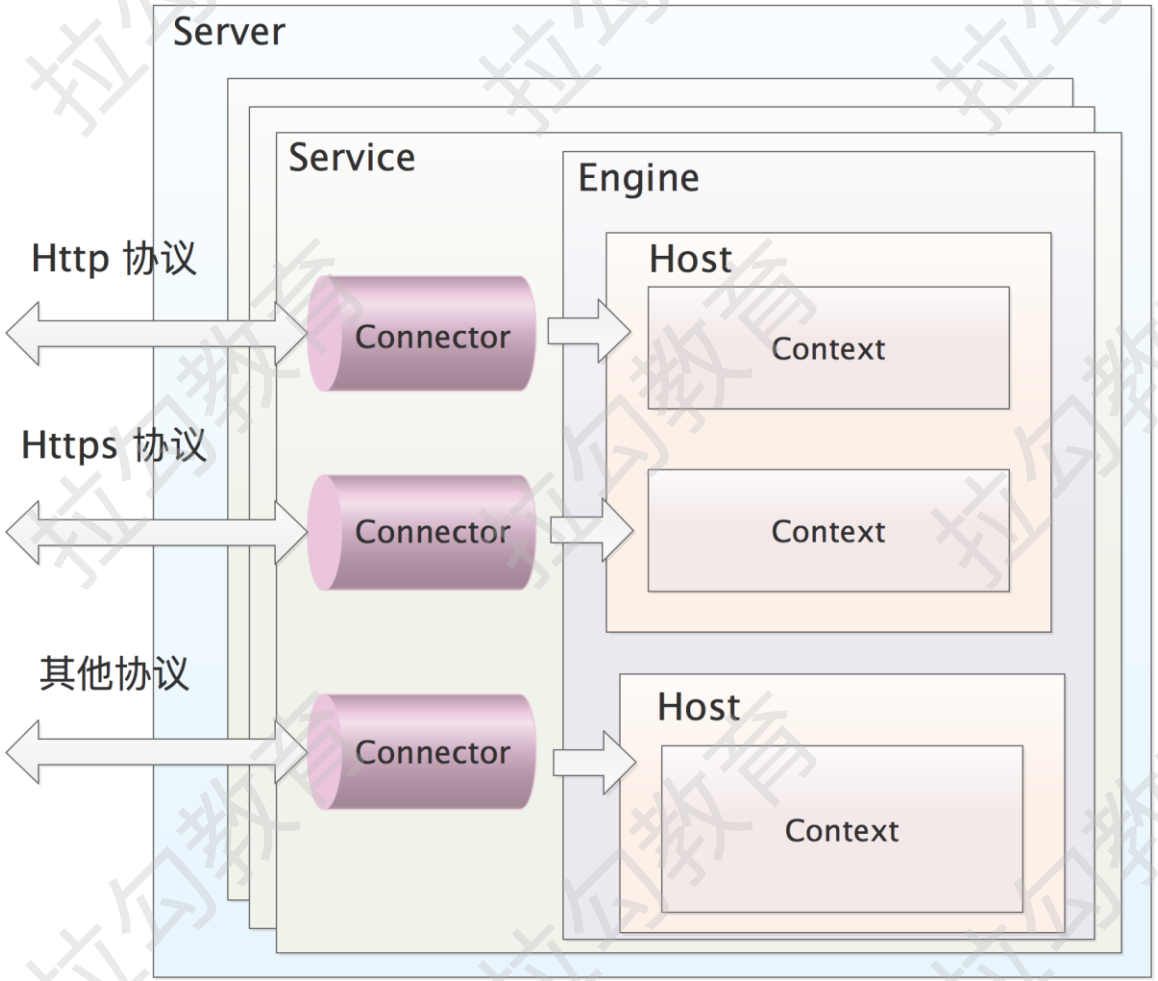
— 拉勾教育出品 —

第15讲：Tomcat 插件原理精析
看 SkyWalking 如何增强这只 Cat（上）

前言

- SkyWalking Agent 的整体架构、启动流程
- 插件埋点的基本原理，其中深入讲解了对静态方法、构造方法以及实例方法的拦截和增强并结合 mysql-8.x 插件进行串讲
- Trace 基本概念在 SkyWalking 中的落地，其中讲解了 Trace ID、Span、TraceSegment、TracingContext 等核心组件的实现，并结合 demo-webapp 进行了分析
- 核心 BootService 实现的深入剖析，其中包括了网络连接的封装和管理、服务以及服务实例的注册流程定期心跳、EndpointName、NetworkAddress 定期同步、Context 生成与管理、客户端采样的功能 Trace 的收集与发送
- DataCarrier 核心原理的深入剖析

Tomcat 架构基础



Tomcat 架构基础

Container 是容器的父接口，所有子容器都必须实现这个接口

Tomcat 中有四个子容器组件：Engine、Host、Context、Wrapper（四个组件是父子关系）

Engine 包含 Host

Host 包含 Context

Context 包含 Wrapper

Tomcat 架构基础

四个 Container 的核心功能：

- Engine：用于管理多个站点，一个 Service 最多只能有一个 Engine
- Host：代表一个站点，也可以叫虚拟主机，通过在 server.xml 配置文件就可以添加 Host

  一个 Host 下可以运行多个 Context，但是在实践中，单 JVM 的处理能力有限

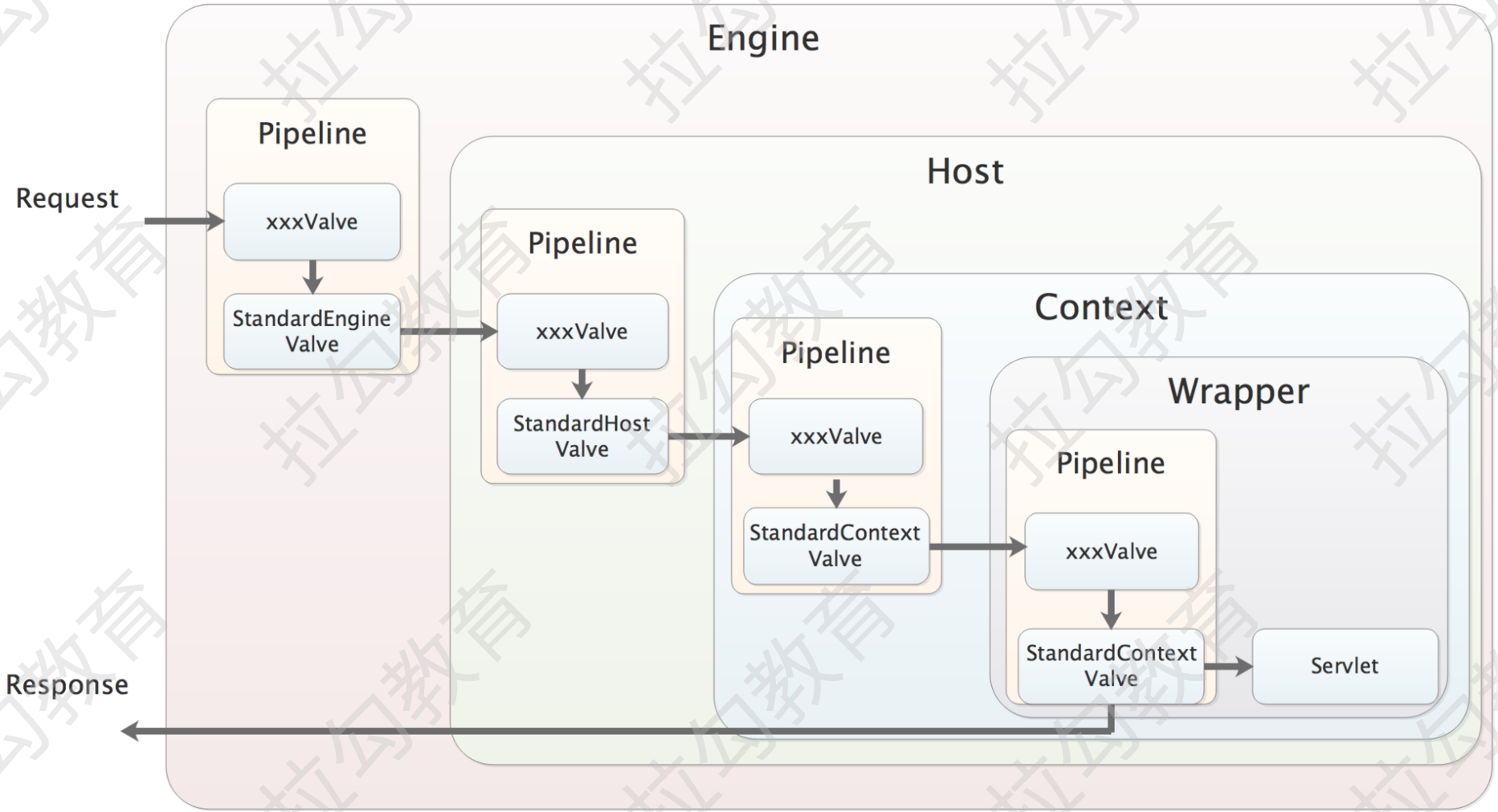
  一般一个 Tomcat 实例只会配置一个 Host，也只会配置一个 Context
- Context：代表一个应用程序，对应你在日常开发的一个 Web 应用

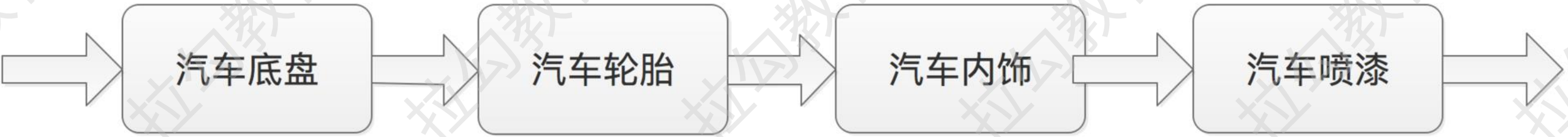
  Context 最重要的功能就是管理它里面的 Servlet 实例，并为 Request 匹配正确的 Servlet

  Servlet 实例在 Context 中是以 Wrapper 出现的
- Wrapper：一个 Wrapper 负责管理一个 Servlet，包括 Servlet 的装载、初始化、执行以及资源回收

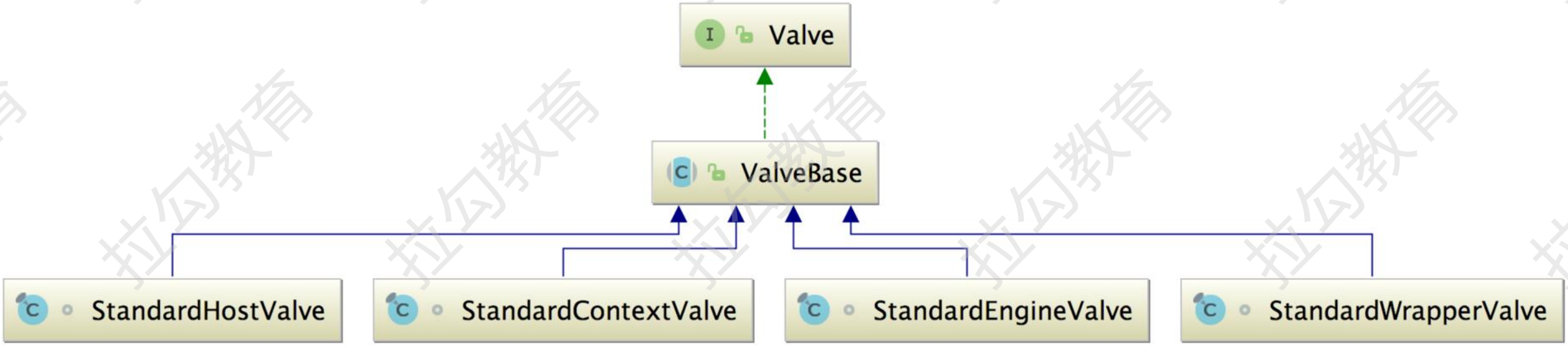
  Wrapper 是最底层的容器，没有子容器

Tomcat 架构基础





Tomcat 架构基础



tomcat-7.x-8.x-plugin 插件

tomcat-7.x-8.x-plugin 插件要做的事情也比较明确：

1. 在请求进入 Web 项目之前进行拦截
2. 检测当前请求是否处于一个 Trace 之中，也就是检测当前请求是否携带了 ContextCarrier

   如果携带了 ContextCarrier，则在创建 TracingContext 时恢复上下文信息，保持实现 Trace 跨进程传递

   如果没携带 ContextCarrier，则会开启一个全新的 TracingContext
3. 创建（或 restart） EntrySpan
4. 记录一些额外的信息

   例如请求相关的 Tags 信息（请求的 URL、Method 信息等）

   记录当前组件的类型（即 Tomcat）等

tomcat-7.x-8.x-plugin 插件

Valve 接口中定义的 invoke(Request request, Response response) 方法是每个 Valve 的核心逻辑

例如根据请求信息进行过滤、修改请求的特殊字段、打印 access log 等

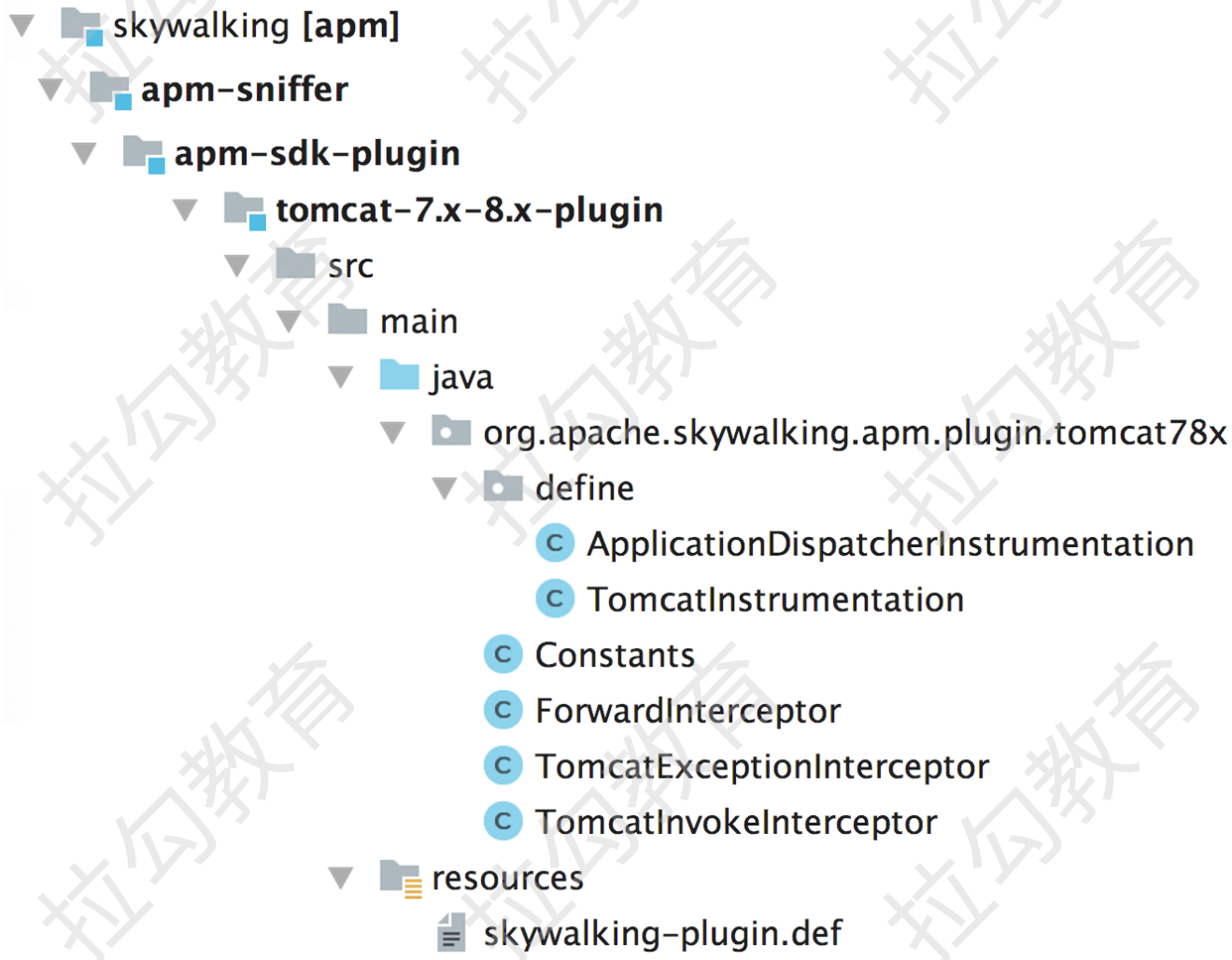
特殊功能的 Valve 实现是可插拔的

标准 Valve 实现不可删除

tomcat-7.x-8.x-plugin 插件


```
public final void invoke(Request request, Response response){
    // 根据请求选择Context
    Context context = request.getContext();
    // 获取 Context 中第一个 Valve，并调用其invoke()方法
    context.getPipeline().getFirst().invoke(request, response);
    Throwable t = (Throwable) request
        .getAttribute(RequestDispatcher.ERROR_EXCEPTION);
    if (response.isErrorReportRequired()) {
      if (t != null) {//出现异常的话，会调用throwable()方法处理
        throwable(request, response, t);
      }
    }
}
```

tomcat-7.x-8.x-plugin 插件

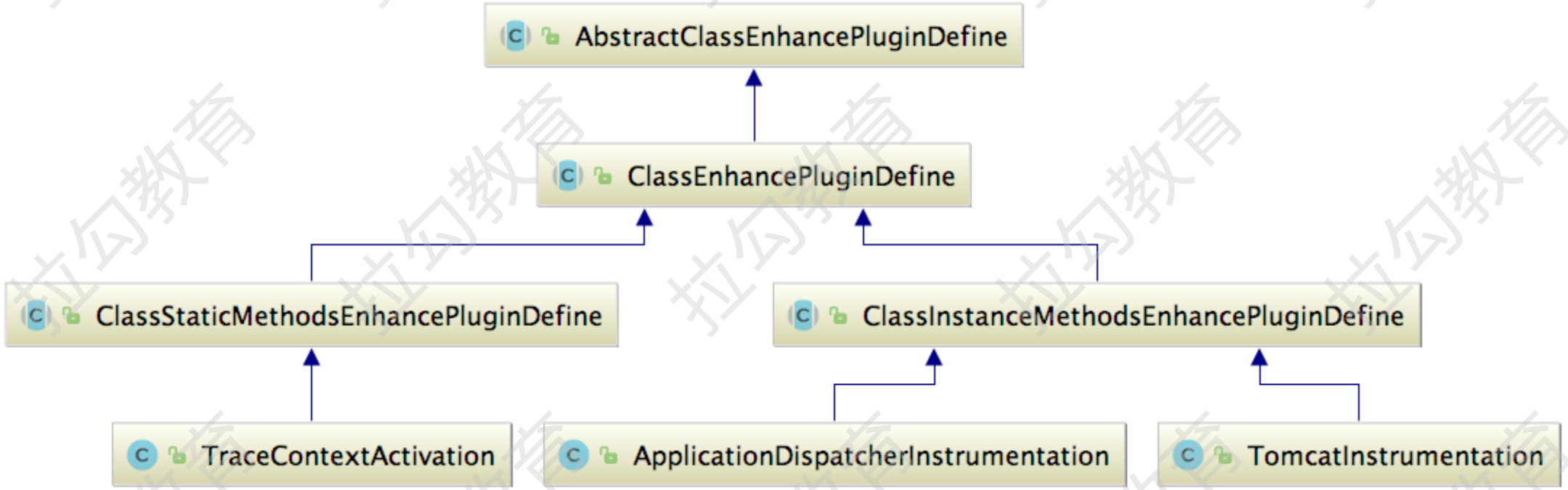


tomcat-7.x-8.x-plugin 插件


```
tomcat-7.x/8.x=org.apache.skywalking.apm.plugin.tomcat78x.define
.TomcatInstrumentation

tomcat-7.x/8.x=org.apache.skywalking.apm.plugin.tomcat78x.define
.ApplicationDispatcherInstrumentation
```

tomcat-7.x-8.x-plugin 插件



tomcat-7.x-8.x-plugin 插件

ClassEnhancePluginDefine 的子类需要实现下面三个方法：

- getStaticMethodsInterceptPoints()方法

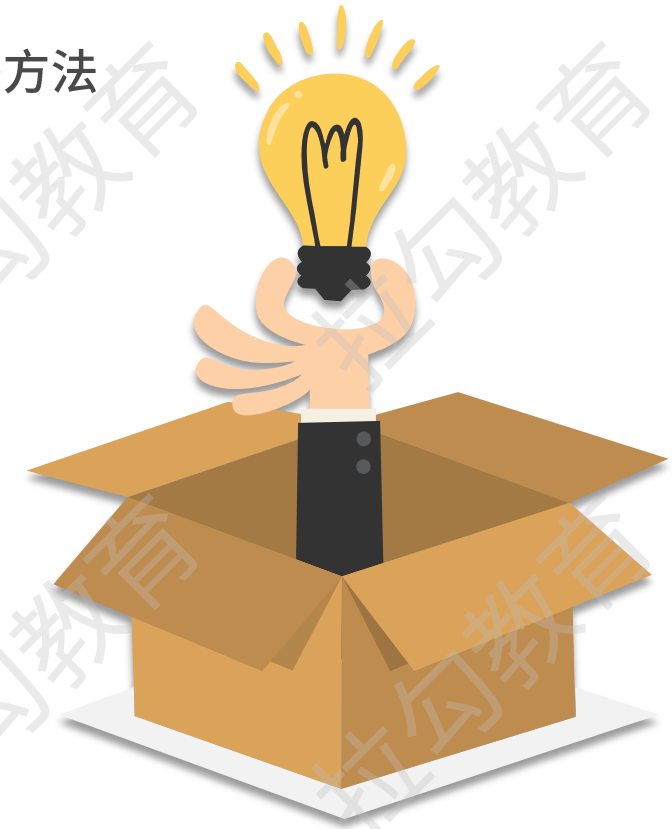
  用于获取 static 静态方法增强点，指定了增强哪些类的哪些 static 静态方法

- getConstructorsInterceptPoints()方法

  用于获取构造方法增强点，指定增强哪些类的哪些构造方法

- getInstanceMethodsInterceptPoints()方法

  用于获取实例方法增强点，指定增强哪些类的哪些实例方法



tomcat-7.x-8.x-plugin 插件

ClassInstanceMethodsEnhancePluginDefine

只实现了 getStaticMethodsInterceptPoints() 方法，具体实现为空实现

ClassStaticMethodsEnhancePluginDefine 正好相反

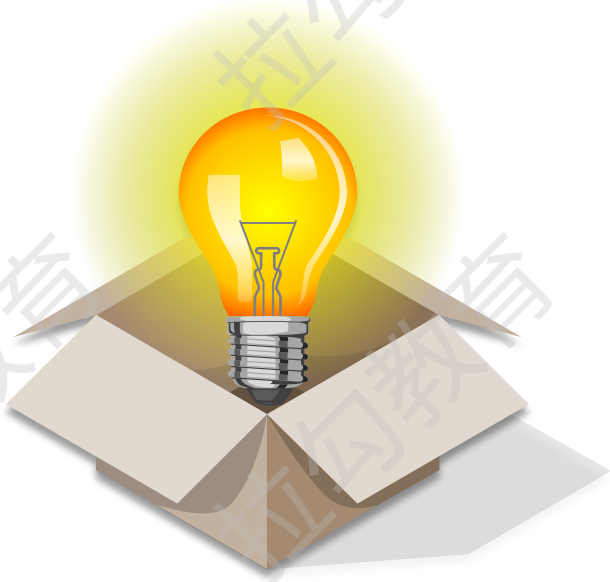
实现了 getConstructorsInterceptPoints() 和 getInstanceMethodsInterceptPoints() 两个方法

且这两个方法都是空实现

AbstractMysqlInstrumentation

同时实现上述三个方法（且三个方法都是空实现）

然后由子类根据具体情况进行覆盖



tomcat-7.x-8.x-plugin 插件

在实践中可以比较

AbstractMysqlInstrumentation

ClassInstanceMethodsEnhancePluginDefine

ClassStaticMethodsEnhancePluginDefine 的设计方式

根据实际情况进行折中选择

Next：第14讲《Tomcat 插件原理精析，看 SkyWalking 如何增强这只 Cat（下）》

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