

拉勾教育

— 互联网人实战大学 —

《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

— 拉勾教育出品 —

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

TomcatInstrumentation

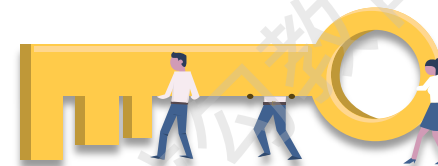
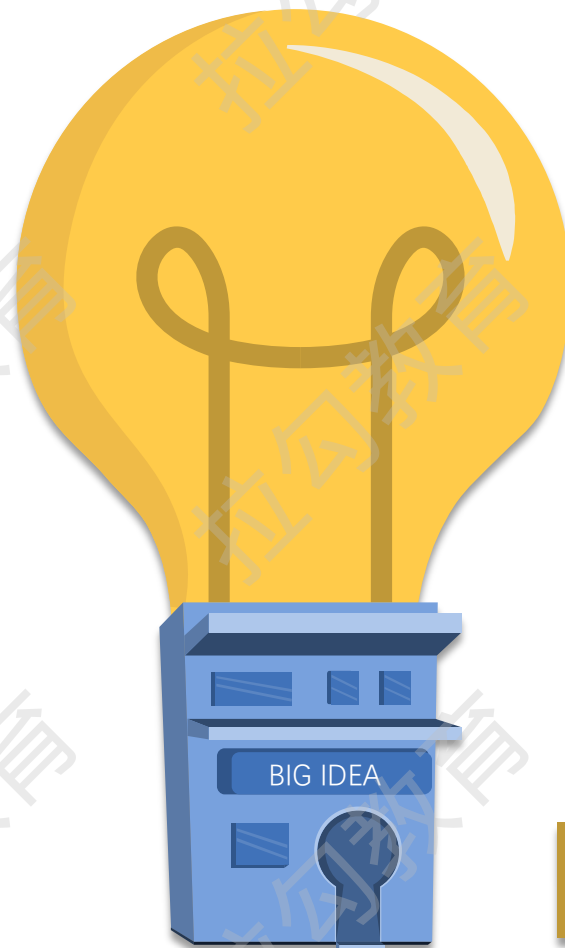
拉勾教育

— 互联网人实战大学 —

TomcatInstrumentation 插件类

重点关注四个问题：

1. 拦截哪个类
2. 拦截哪个方法
3. 由谁进行增强
4. 具体增强逻辑

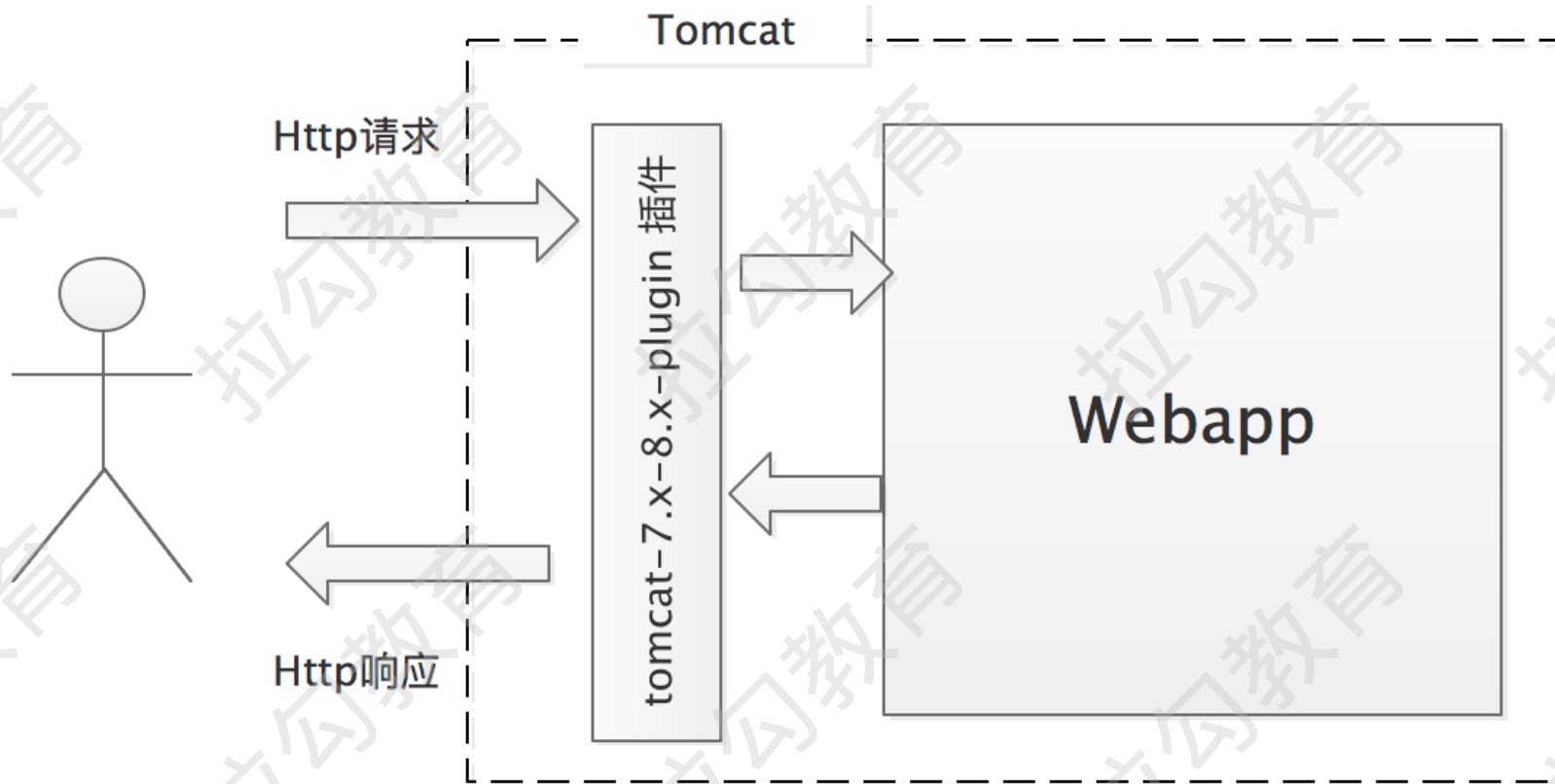


enhanceClass()方法返回的 ClassMatch 匹配了拦截的类名

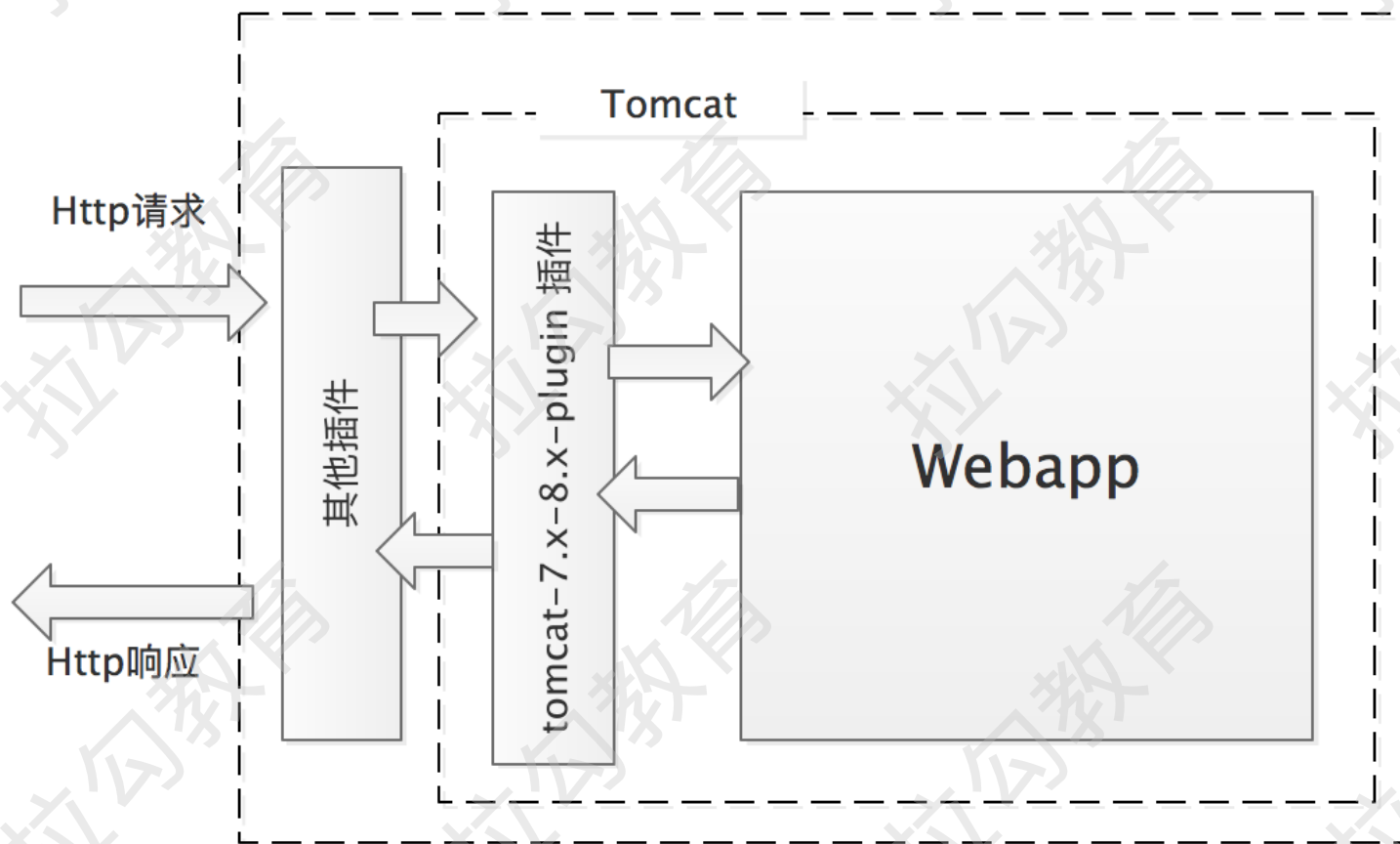
```
protected ClassMatch enhanceClass() { // 拦截Tomcat?StandardHostValve类  
    return byName("org.apache.catalina.core.StandardHostValve");  
}
```

```
new InstanceMethodsInterceptPoint() {  
    public ElementMatcher<MethodDescription> getMethodsMatcher() {  
        return named( "invoke" ); // 拦截名为 invoke 的方法  
    }  
  
    public String getMethodsInterceptor() {  
        return "org.apache.skywalking.apm.plugin.tomcat78x  
            .TomcatInvokeInterceptor"; // 拦截后的增强逻辑  
    }  
  
    public boolean isOverrideArgs() {  
        return false; // 不修改 invoke()方法的参数  
    }  
}
```

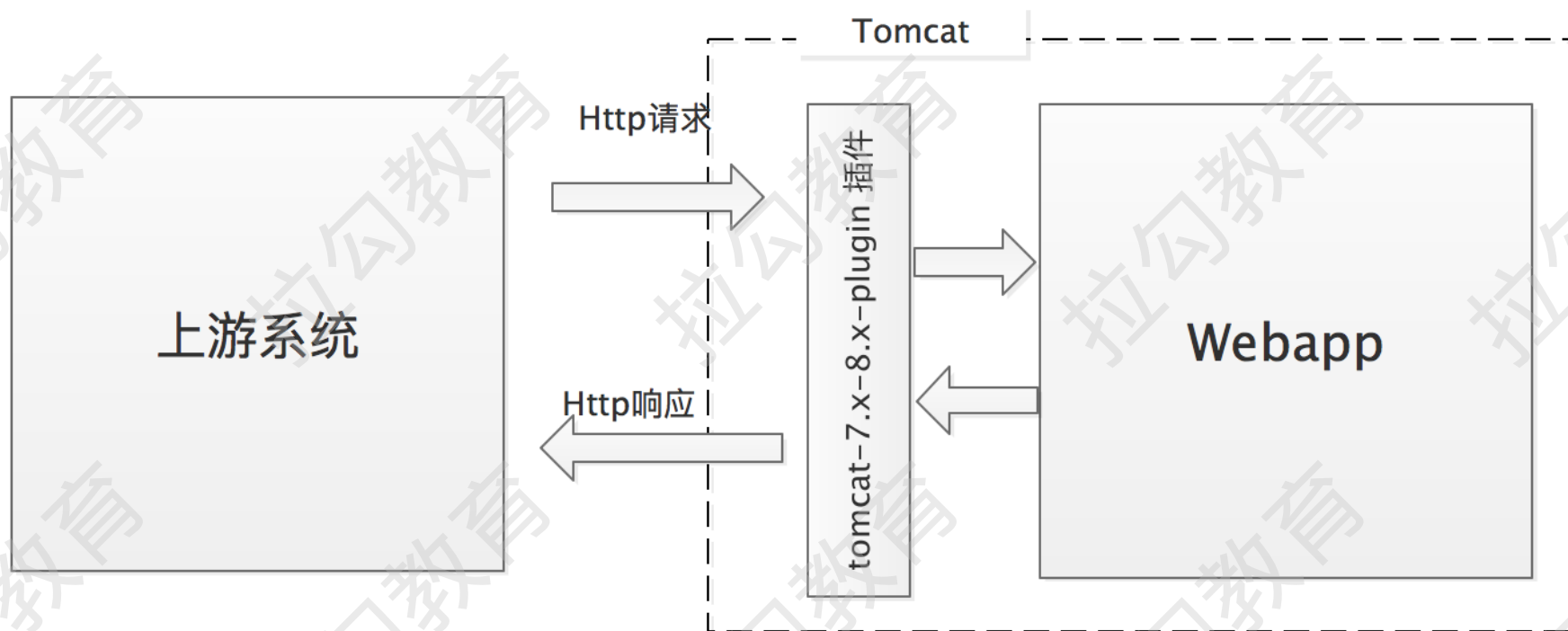
当 Tomcat 作为用户请求接入层的场景时



tomcat-7.x-8.x-plugin 插件被嵌套在其他插件之后的场景



Tomcat 作为下游系统被其他系统调用的场景



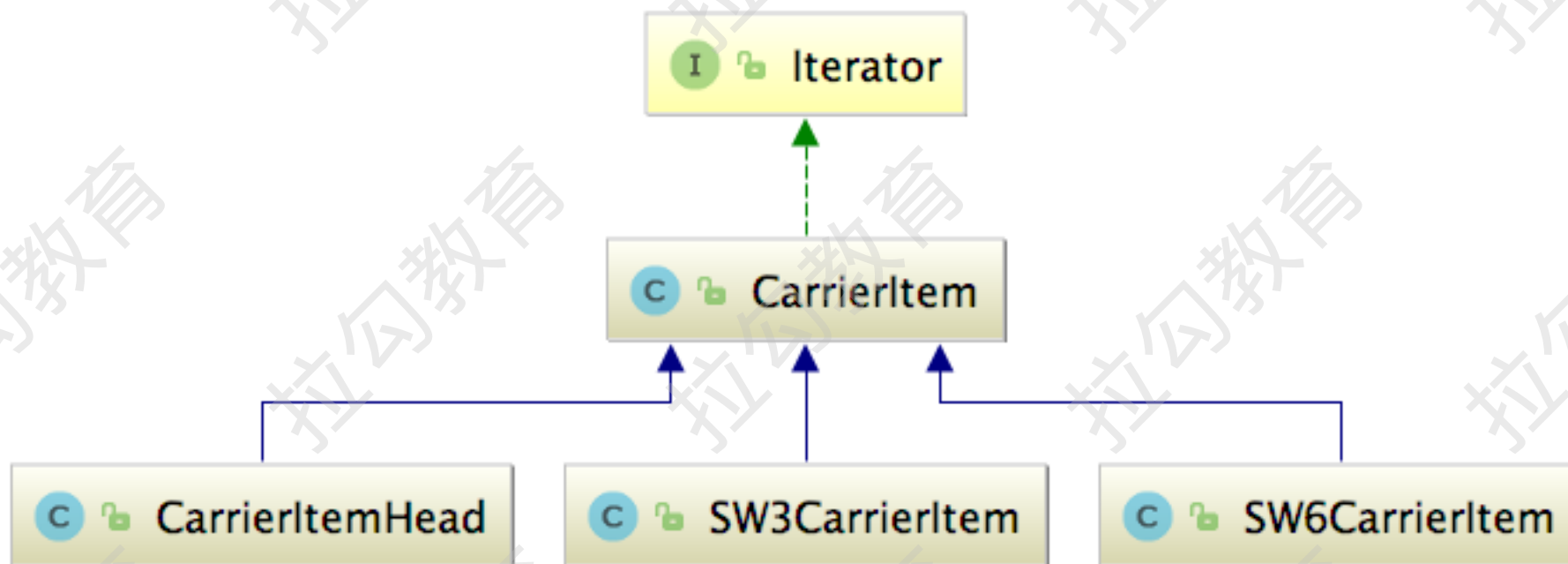

```
public void beforeMethod(EnhancedInstance objInst, Method method,
    Object[] allArguments, Class<?>[] argumentsTypes,
    MethodInterceptorResult result) throws Throwable {
    // invoke()方法的第一个参数就是HttpServletRequest对象
    HttpServletRequest request = (HttpServletRequest)allArguments[0];
    //创建一个空的ContextCarrier对象
    ContextCarrier contextCarrier = new ContextCarrier();
    //从Http请求头中反序列化ContextCarrier
    CarrierItem next = contextCarrier.items();
    while (next.hasNext()) {
        next = next.next();
        next.setHeadValue(request.getHeader(next.getHeadKey()));
    }
    //获取当前线程绑定的TracingContext，如果未绑定则会创建新TracingContext并
    //绑定，同时还会创建EntrySpan，如果已存在EntrySpan，则再次调用其start()方
    //法。这里的第一个参数是operationName(即EndpointName)，Tomcat的场景下
    //就是请求的URI
    AbstractSpan span = ContextManager.createEntrySpan(
        request.getRequestURI(), contextCarrier);
}
```

```
ContextCarrier contextCarrier = new ContextCarrier();
//从Http请求头中反序列化ContextCarrier
CarrierItem next = contextCarrier.items();
while (next.hasNext()) {
    next = next.next();
    next.setHeadValue(request.getHeader(next.getHeadKey()));
}
//获取当前线程绑定的TracingContext, 如果未绑定则会创建新TracingContext并
//绑定, 同时还会创建EntrySpan, 如果已存在EntrySpan, 则再次调用其start()方
//法。这里的第一个参数是operationName(即EndpointName), Tomcat的场景下
//就是请求的URI
AbstractSpan span = ContextManager.createEntrySpan(
    request.getRequestURI(), contextCarrier);
//为EntrySpan添加Tags, 记录请求的URL以及Method信息
Tags.URL.set(span, request.getRequestURL().toString());
Tags.HTTP.METHOD.set(span, request.getMethod());
span.setComponent(ComponentsDefine.TOMCAT); //设置component字段
SpanLayer.asHttp(span); //匹配layer字段
}
```

再探 ContextCarrier

拉勾教育

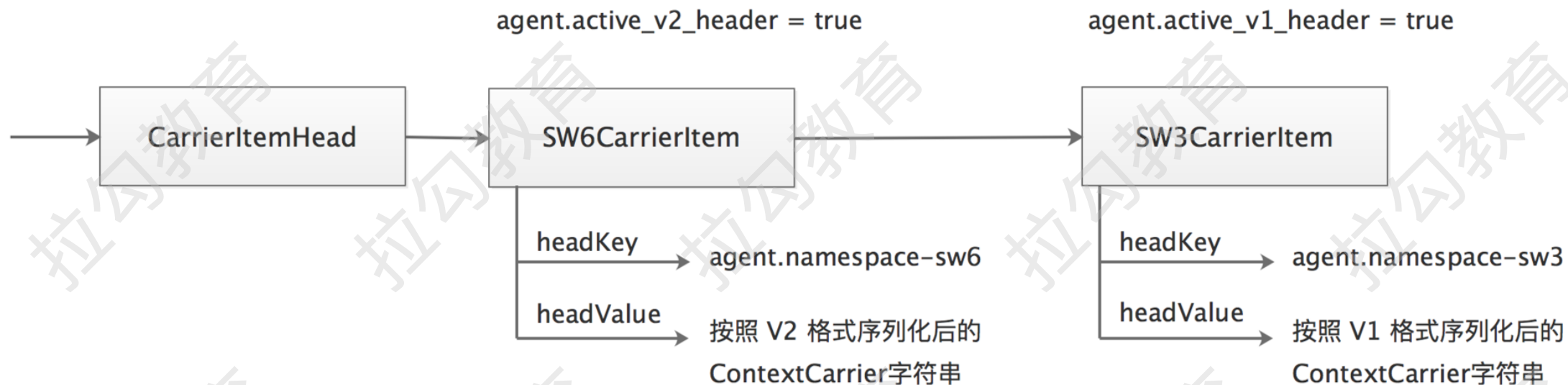
— 互联网人实战大学 —



再探 ContextCarrier

拉勾教育

— 互联网人实战大学 —



```
public SW6CarrierItem(ContextCarrier carrier, CarrierItem next) {  
    super(HEADER_NAME, // headKey  
        // 按照V2版本序列化得到headValue  
        carrier.serialize(ContextCarrier.HeaderVersion.v2),  
        next); // 下一个CarrierItem节点  
    this.carrier = carrier; // 记录关联的ContextCarrier对象  
}
```

```
//创建空的ContextCarrier对象
ContextCarrier contextCarrier = new ContextCarrier();
//创建CarrierItem链表，因为ContextCarrier对象是空的，所以链表也是空的
CarrierItem next = contextCarrier.items();
while (next.hasNext()) {
    next = next.next();
    //拿到HttpHeader的Value，即对应版本的ContextCarrier序列化字符串
    next.setHeadValue(request.getHeader(next.getHeadKey()));
}
```

```
public Object afterMethod(EnhancedInstance objInst, Method method,  
    Object[] allArguments, Class<?>[] argumentsTypes,  
    Object ret) throws Throwable {
```

```
// invoke()方法的第二个参数是HttpServletResponse
```

```
HttpServletResponse response =
```

```
    (HttpServletResponse)allArguments[1];
```

```
// 获取当前Span，因为TracingContext是栈的形式管理Span，当前Span即为
```

```
// beforeMethod()方法中创建的EntrySpan
```

```
AbstractSpan span = ContextManager.activeSpan();
```

```
if (response.getStatus() >= 400) {
```

```
    // 如果响应码是4xx或是5xx，则表示Http相应异常，标记当前Span的
```

```
    // errorOccurred字段，并记录一个Key为status_code的Tag
```

```
// errorOccurred字段，并记录一个Key为status_code的Tag
span.errorOccurred();
Tags.STATUS_CODE.set(span,
    Integer.toString(response.getStatus()));
}
//关闭当前EntrySpan，如果EntrySpan完全关闭，则整个Span栈为空
//所在的TraceSegment也将随之关闭，这些逻辑在前面已经详细介绍过了
ContextManager.stopSpan();
//从RuntimeContext中清理FORWARD_REQUEST_FLAG信息，其含义后面再说
ContextManager.getRuntimeContext().remove(
    Constants.FORWARD_REQUEST_FLAG);
return ret;
}
```


ApplicationDispatcherInstrumentation

拉勾教育

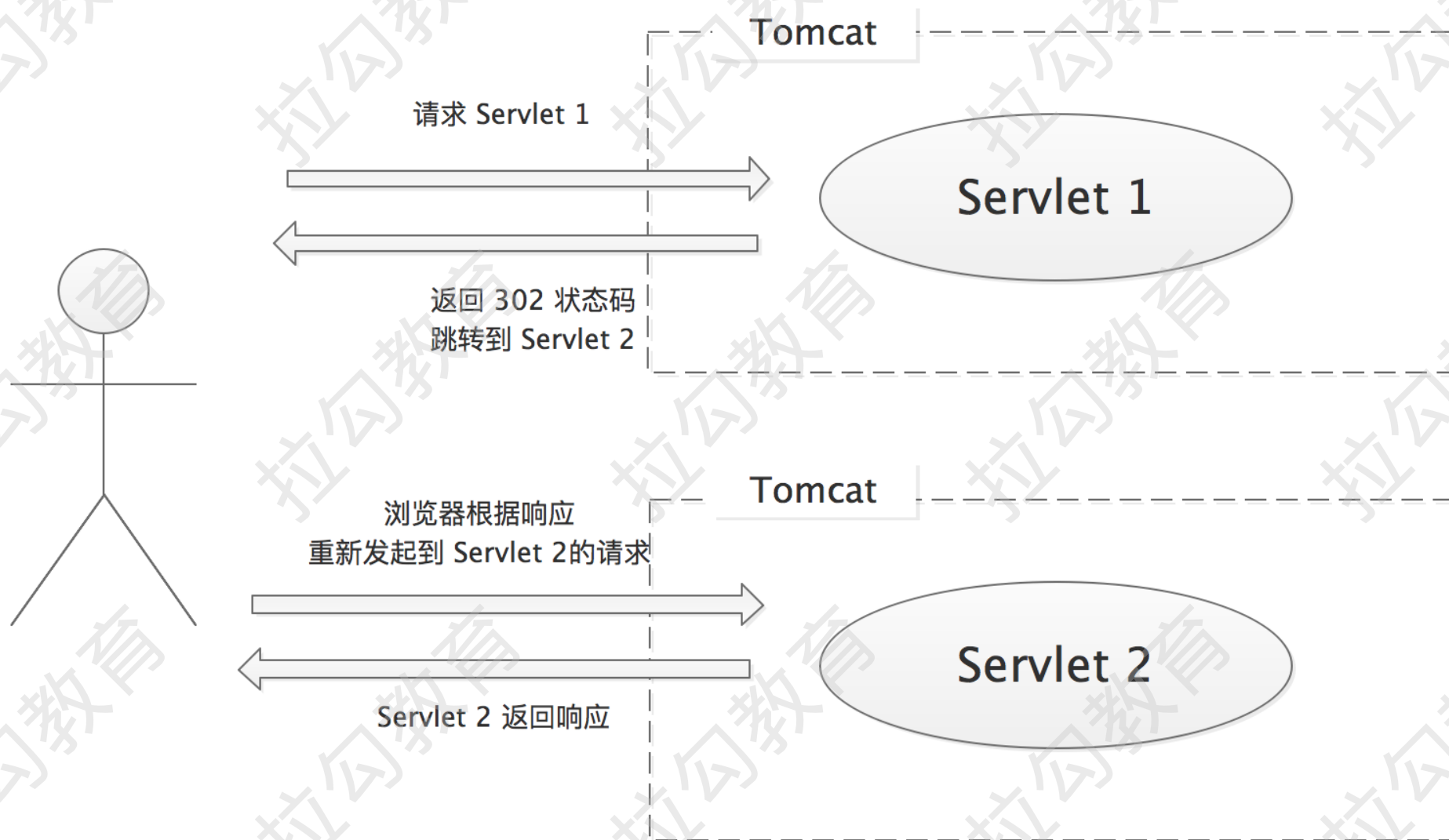
— 互联网人实战大学 —

```
public void doGet(HttpServletRequest request,  
    HttpServletResponse response){  
    response.sendRedirect("跳转到的目标URL");  
}
```

ApplicationDispatcherInstrumentation

拉勾教育

互联网人实战大学

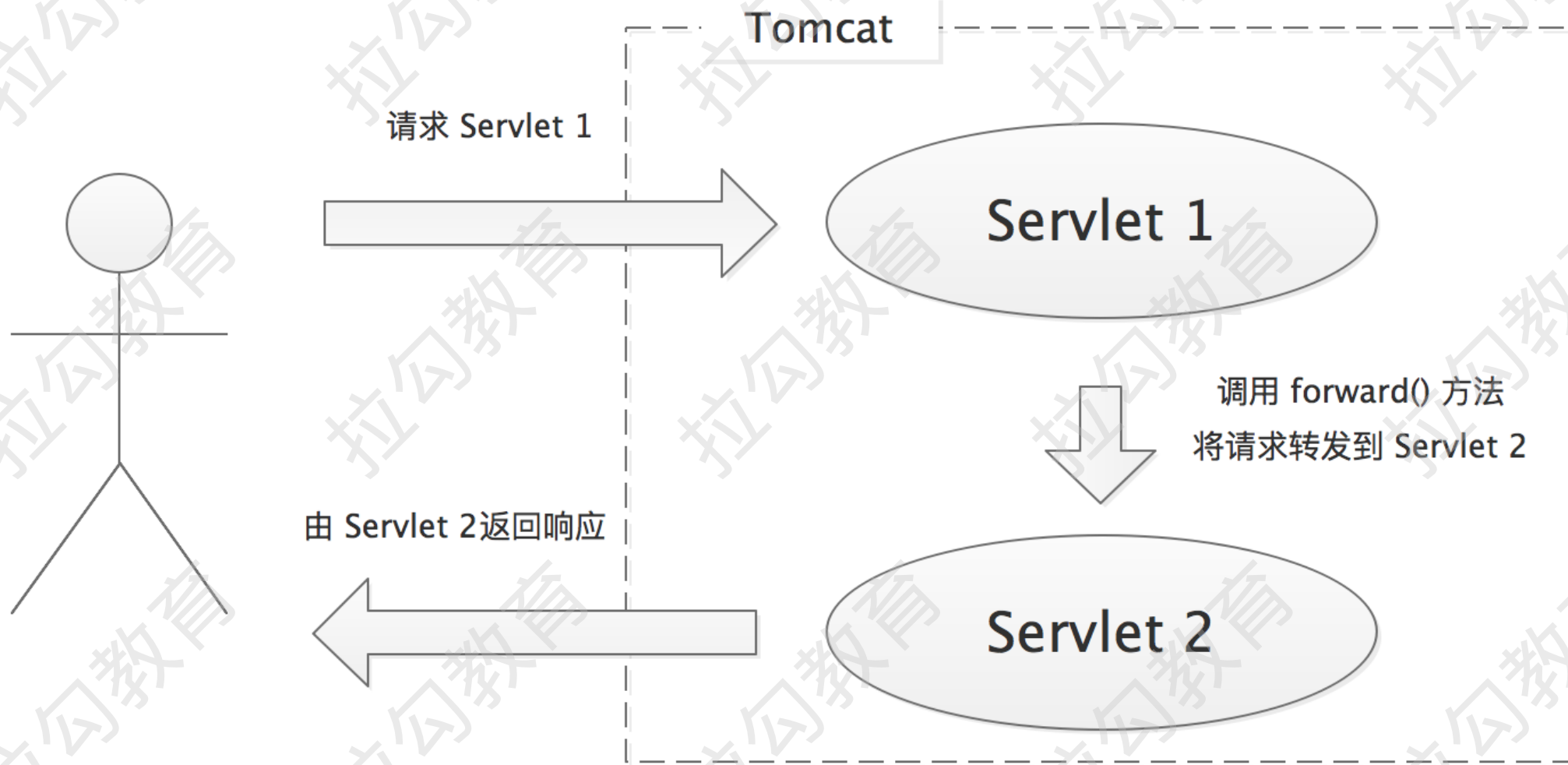


```
public void sendRedirect(String location, int status) {  
    try {  
        String locationUri = ...; // 获取 redirectUrl  
        setStatus(status); // 状态码设置为302或是307  
        setHeader("Location", locationUri);  
        if (getContext().getSendRedirectBody()) { // 返回ResponseBody  
            ...  
        }  
    } catch (IllegalArgumentException e) {  
        setStatus("404");  
    }  
    setSuspended(true); // Cause the response to be finished  
}
```

ApplicationDispatcherInstrumentation

拉勾教育

— 互联网人实战大学 —



```
public void doGet(HttpServletRequest request ,  
    HttpServletResponse response){  
    //获取请求转发器对象，该转发器的指向通过getRequestDispatcher()的参数设置  
    RequestDispatcher requestDispatcher =  
        request.getRequestDispatcher("Servlet2???");  
    // 调用forward()方法，转发请求  
    requestDispatcher.forward(request,response);  
}
```

ApplicationDispatcherInstrumentation

拉勾教育

— 互联网人实战大学 —

```
public void onConstruct(EnhancedInstance objInst,  
    Object[] allArguments) {  
    // ApplicationDispatcher构造方法的第二个参数为跳转的目标地址，下图所示  
    objInst.setSkyWalkingDynamicField(allArguments[1]);  
}
```

```
public void onConstruct(EnhancedInstance objInst,  
    Object[] allArguments) {
```

// ApplicationDispatcher构造方法的第二个参数为跳转的目标地址，下图所示

```
    objInst.setSkyWalkingDynamicField(allArguments[1]);
```

```
}
```

```
public ApplicationDispatcher  
    (Wrapper wrapper, String requestURI, String servletPath,  
     String pathInfo, String queryString, HttpServletMapping mapping, String name) {
```

ApplicationDispatcherInstrumentation

拉勾教育

— 互联网人实战大学 —

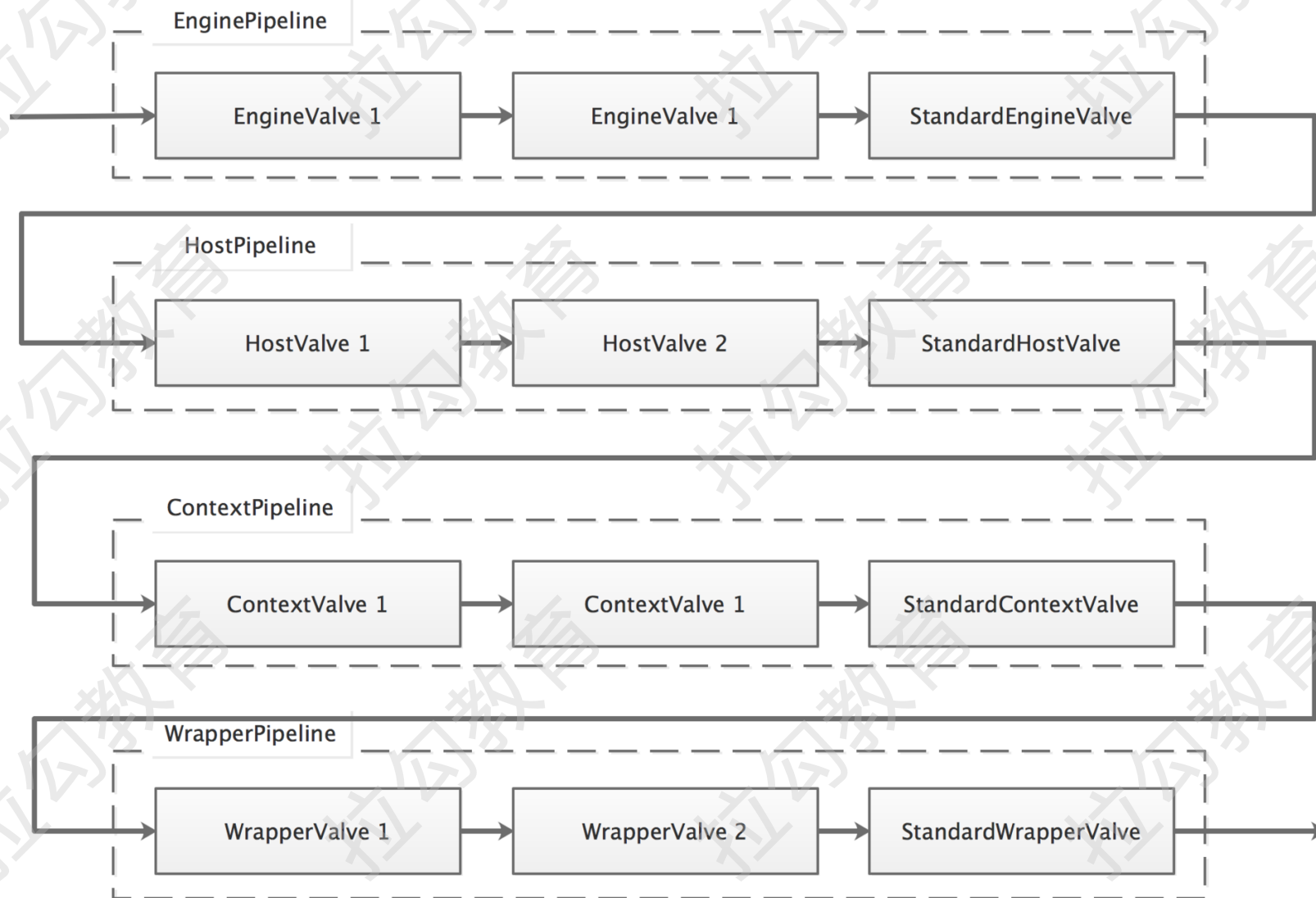
```
public void beforeMethod(EnhancedInstance objInst, Method method,
    Object[] allArguments, Class<?>[] argumentsTypes,
    MethodInterceptResult result) throws Throwable {
    AbstractSpan abstractTracingSpan =
        ContextManager.activeSpan();
    Map<String, String> eventMap = new HashMap<String, String>();
    eventMap.put("forward-url",
        objInst.getSkyWalkingDynamicField() == null ? "" :
        String.valueOf(objInst.getSkyWalkingDynamicField()));
    // 通过Log的方式记录将跳转URL
    abstractTracingSpan.log(System.currentTimeMillis(), eventMap);
    ContextManager.getRuntimeContext() // 记录forward标记
        .put(Constants.FORWARD_REQUEST_FLAG, true);
}
```


总结

拉勾教育

— 互联网人实战大学 —

整个调用流程



Next: 第15讲 《Dubbo 插件核心剖析，Trace 是这样跨服务传播的》

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