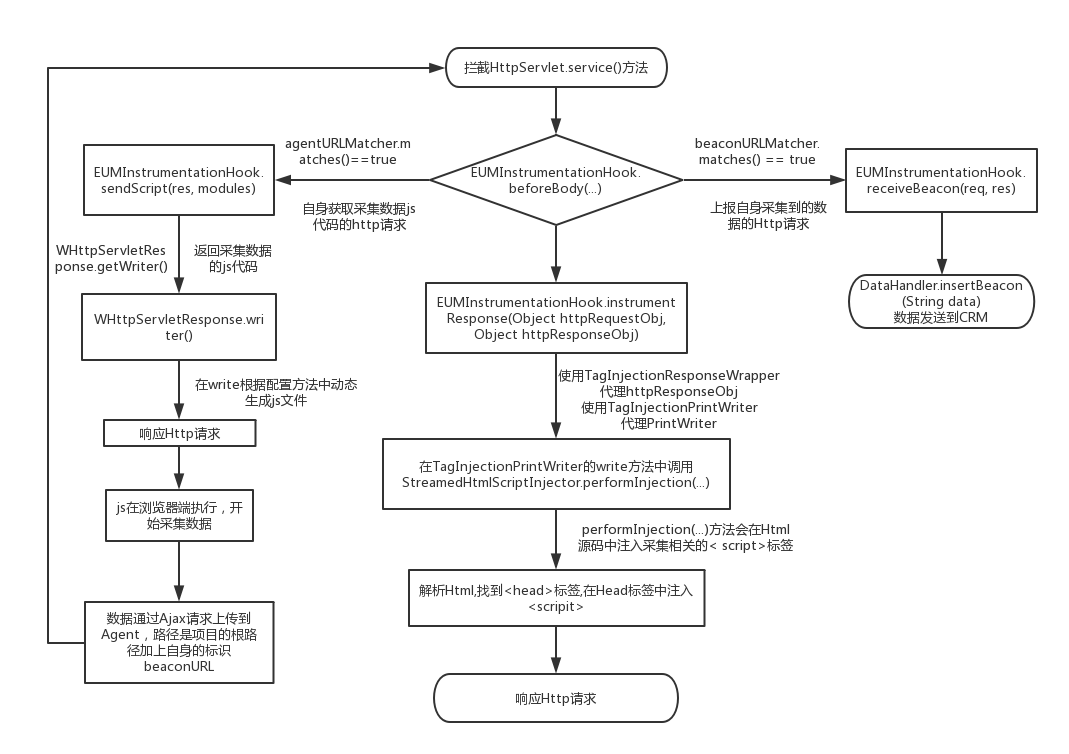
InspectIT\_EUM实现调研

1.实现原理概述



2.实现原理详解

2.1.如何注入的

通过拦截HttpService.service()方法，拦截所有的Http请求，对每一个Http请求进行判断：

A：如果是自生上报采集到的数据的Http请求，择将请求转发到数据处理的方法。

B: 如果是自身的获取采集数据的js代码的请求，择将请求转发的对应的方法，根据配置动态生成对应的js代码，并返回。

C：如果是正常的业务相关的Http请求，择转入到注入相关的逻辑

注入的基本步骤：

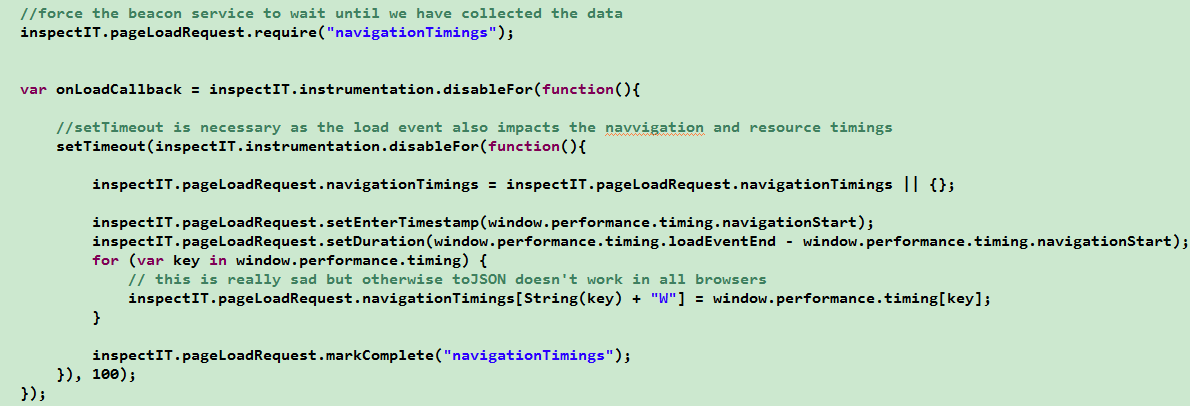
通过代理包装HttpReponse，以及HttpResponse中的输出流PrintWriter和OutputStream重写write方法，在wirite方法中调用StreamedHtmlScriptInjector.performInjection(CharSequence htmlData)，在该方法中解析要输出的Html代码，找到<head>标签作为插入点，插入<scrpt>标签：



<script>标签会根据src属性的值，发起http请求从服务端获取js资源，这个请求会分发到EUMInstrumentationHook.sendScript(WHttpServletResponse res, String activeModules)，该方法会根据配置信息生成js代码，并返回给浏览器，至此js注入完成。

2.2.如何采集信息：

Js代码注入之后，会直接开始执行采集的方法，主要的方式为调用浏览器相关的API来获取对应的信息，以页面加在时间为例：



调用了浏览器 Navigation Timing 系列中的方法，获取到对应的信息，其它的实现类似都是通过调用浏览器对应的API进行简单的计算获取。

2.3.采集那些指标

采集的指标分为几个系列，集体的代码分别在不同的js文件中：

ajax.js：监听ajax请求the time between sending the request 和the time between send and finish of all

async.js：监控异步Javascript函数setTimeout和setInterval 的开始时间，函数名，间隔时间，迭代次数

browsermetainfo.js：获取主机类型，浏览器类型

listener.js：用于监听附加到DOM元素上的事件的类型，和方法名

restimings.js：用于获取资源资源加载的时间

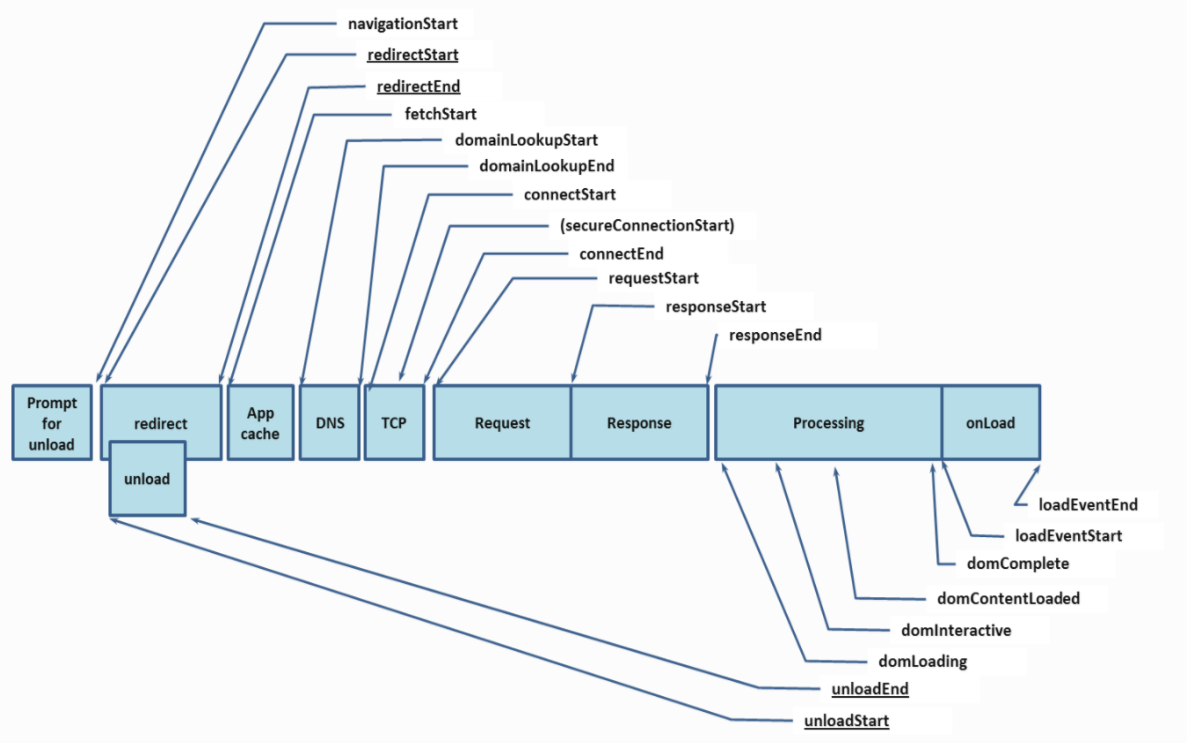
rum-speedindex.js：用于获取页面加载速度，也就是首屏时间

navtimings.js：获取window.performance.timing中的所有属性，可以计算得到页面请求和响应的相关时间

timing中的属性：



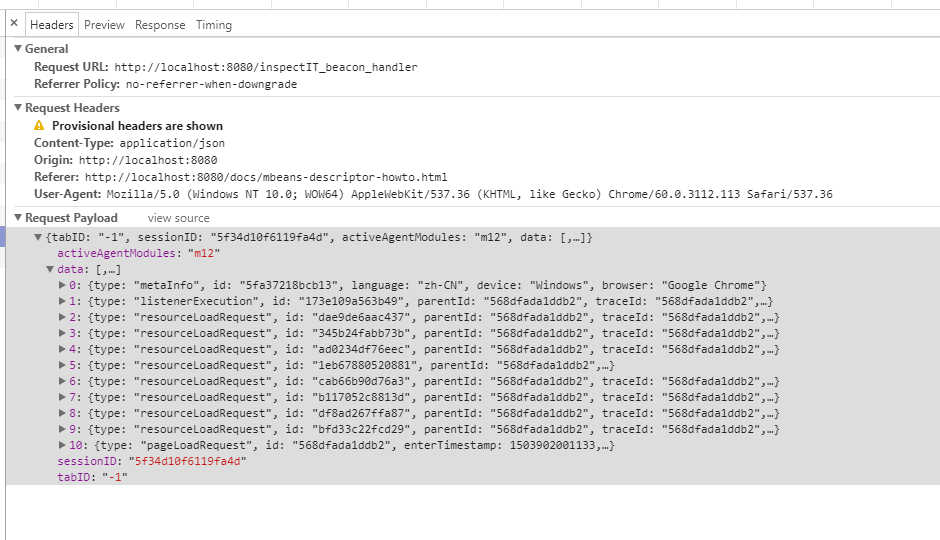
timing中属性的含义：



2.4.数据如何上报

浏览器端采集到的数据，通过Ajax异步请求发送到Agent端，发送的路径是WAS的根路径加上在<script>标签中在注入的具体的路径，这个请求在HttpServlet.service()方法中被拦截，转入数据上报的处理逻辑，最终发送到CMR。

数据上报的格式：



具体格式：

