# Jwebap User Guide

leadyu (yu-lead@163.com)

# 目录

# 1 介绍

Jwebap是一个用于java web application 的profiler工具。它不采用JVMPI提供的特性实现监控，是一个纯粹的JAVA应用，不依赖于OS，JVM，JDK1.4以上用户都可以使用。

它的目的是希望能够安全高效的部署于生产以及测试应用系统，及时地发现应用系统中存在的性能瓶颈，以及为一些动态性很高难于调试和维护的应用系统提供帮助。目前它主要致力于性能分析和优化方面，以后可能往其他监控方面发展（比如业务数据的监控）。因此，组件提供基本的SDK，提供基本的分析功能，不断丰富的分析功能可以通过统一的方式以插件的形式加入Jwebap组件。

以下是Jwebap的特性:

- Jwebap是纯java应用，可以方便的部署于JDK14和以上，各种中间件环境。
- 执行非常高效，几乎不给系统带来更多的开销，目前已经应用于中国电信数个省级大型业务系统。
- 提供http监控分析，时间阀值设置，平均时间，最长最小时间等等统计；jdbc执行监控，连接泄漏，sql监控分析，帮助找出执行sql的代码行数;程序执行方法监控,时间阀值过滤，方法调用jdbc监听，方法调用栈等等。

- 包含三大基础组件：轨迹收集，轨迹容器，Jwebap Console
- 基于 plugin 架构进行扩展，所有的功能都是通过 plugin 方式加入，方便按需使用和加载，默认提供 Tracer 监控插件，完成上述 http，jdbc，method 三块的监控功能。

# 2 什么是轨迹

在 Jwebap 组件的设计中包含一个重要的概念——轨迹。那么什么是轨迹？

- 轨迹的定义

在系统中，任何程序的执行都有可能留下轨迹（数据库的调用会留下 SQL 轨迹，事务的轨迹，http 请求会留下访问的轨迹，程序的方法执行也可以留下轨迹）。

轨迹包含了对性能分析有用的信息（比如上下文的信息，线程堆栈的信息，执行时间的信息等等），轨迹之间存在关系，比如数据库事务的轨迹和 SQL 的轨迹，http 请求和类方法执行的轨迹之间都有一定的关系。在性能分析中，我们可能存在几个关注点，比如数据库，远程调用，方法调用，组件等等，通过这些关注点相关程序留下的轨迹，我们进行性能分析。

- 轨迹的管理

轨迹是错综复杂的，零碎地，关系模糊的，所以要真正有助于性能分析，还需要相应的支持。所以，轨迹又需要生命周期管理，和统计管理。

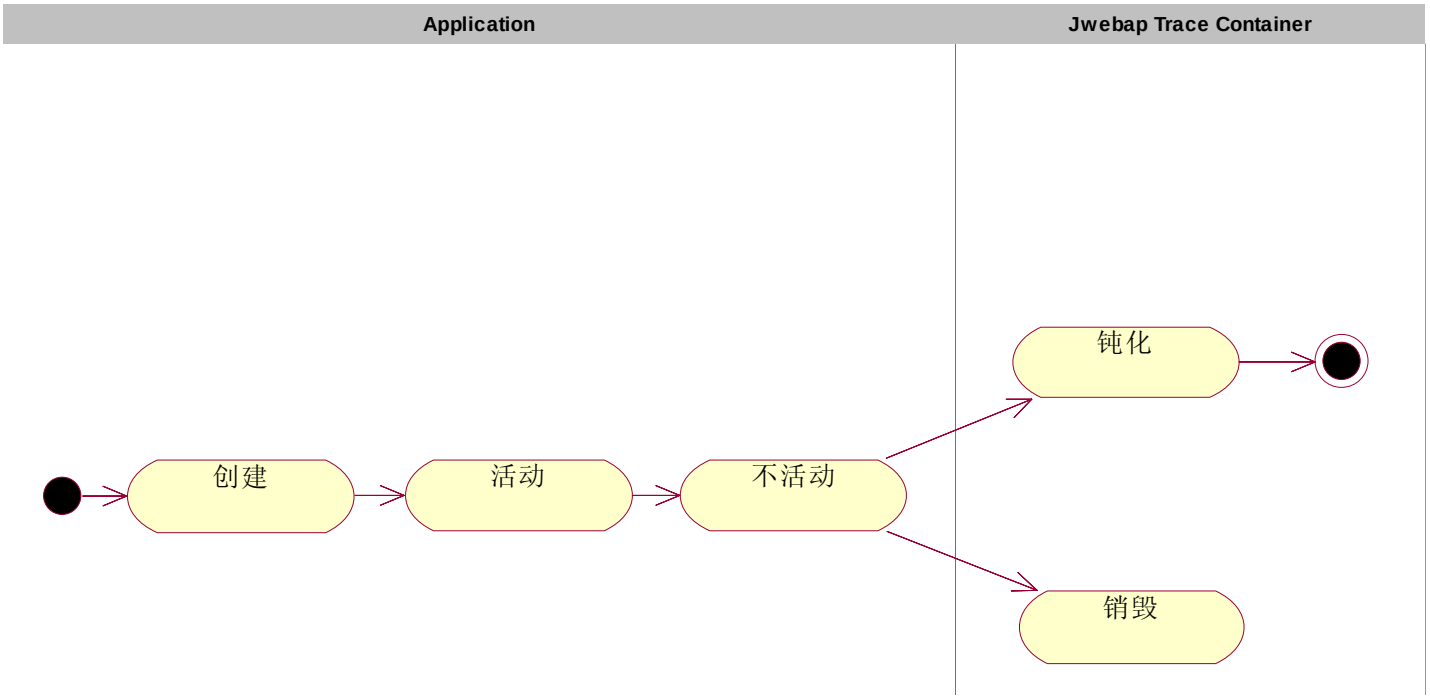
轨迹的生命周期管理：轨迹创建，运行，销毁，钝化的各个状态。通过分析工具把有问题的轨迹（比如执行时间过长）找到，通过视图暴露出来。

轨迹的统计：不同关注点的轨迹，存在不同的统计逻辑，通过不同的分析插件，展现给性能分析者。

- 轨迹的注入

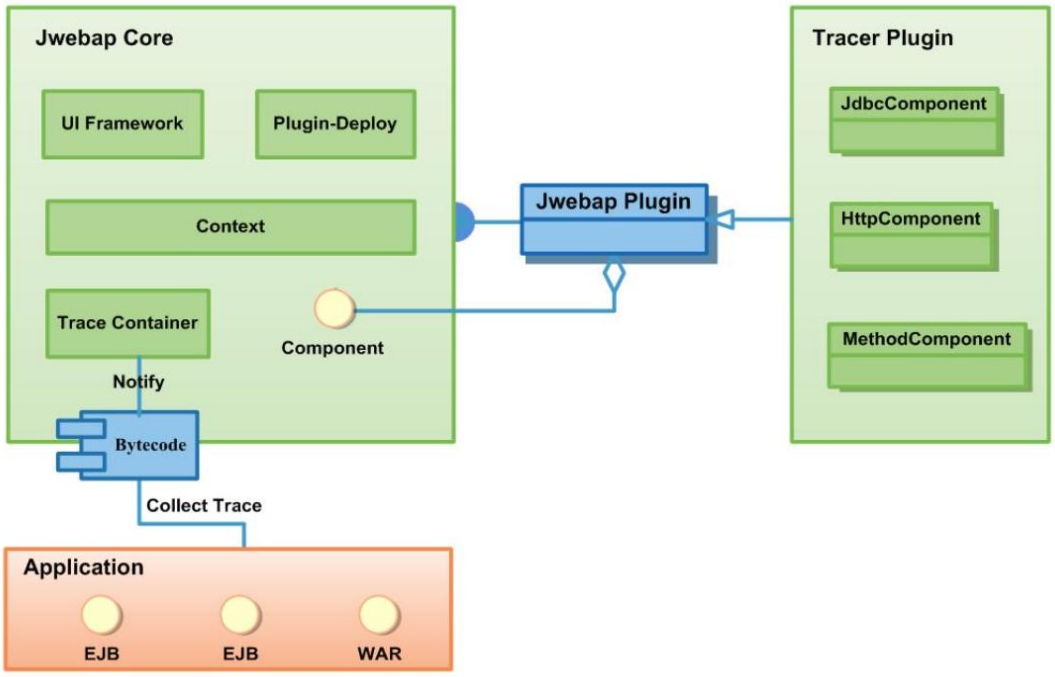
轨迹不会自己产生，我们也不可能修改系统的代码。而我又不想像其他的一些分析工具一样，通过本地接口获取 JVM 的状态从而进行统计（那样就得依赖 JVM，依赖 OS）。而现在有一个不错的办法摆在我们的面前，那就是采用 ClassWorking。目前，我已经实现了一个 ClassEnhancer 以及一个 PackageEnhancer，它能够对指定的类和包进行注入。

- 轨迹的生命周期




# 3 Jwebap 插件

Jwebap 在模型上采用了插件的设计，从而达到对轨迹的收集管理，和分析统计的解耦。模型如下图：



Jwebap 分为两大部分，Jwebap Core 和 Jwebap Plugin。

Jwebap Core 实现了对 Component 的管理，轨迹的收集，轨迹容器，UI Framework 集成了模板引擎以实现视图等等，另外，还提供了 ByteCode 包以实现对代码的注入。

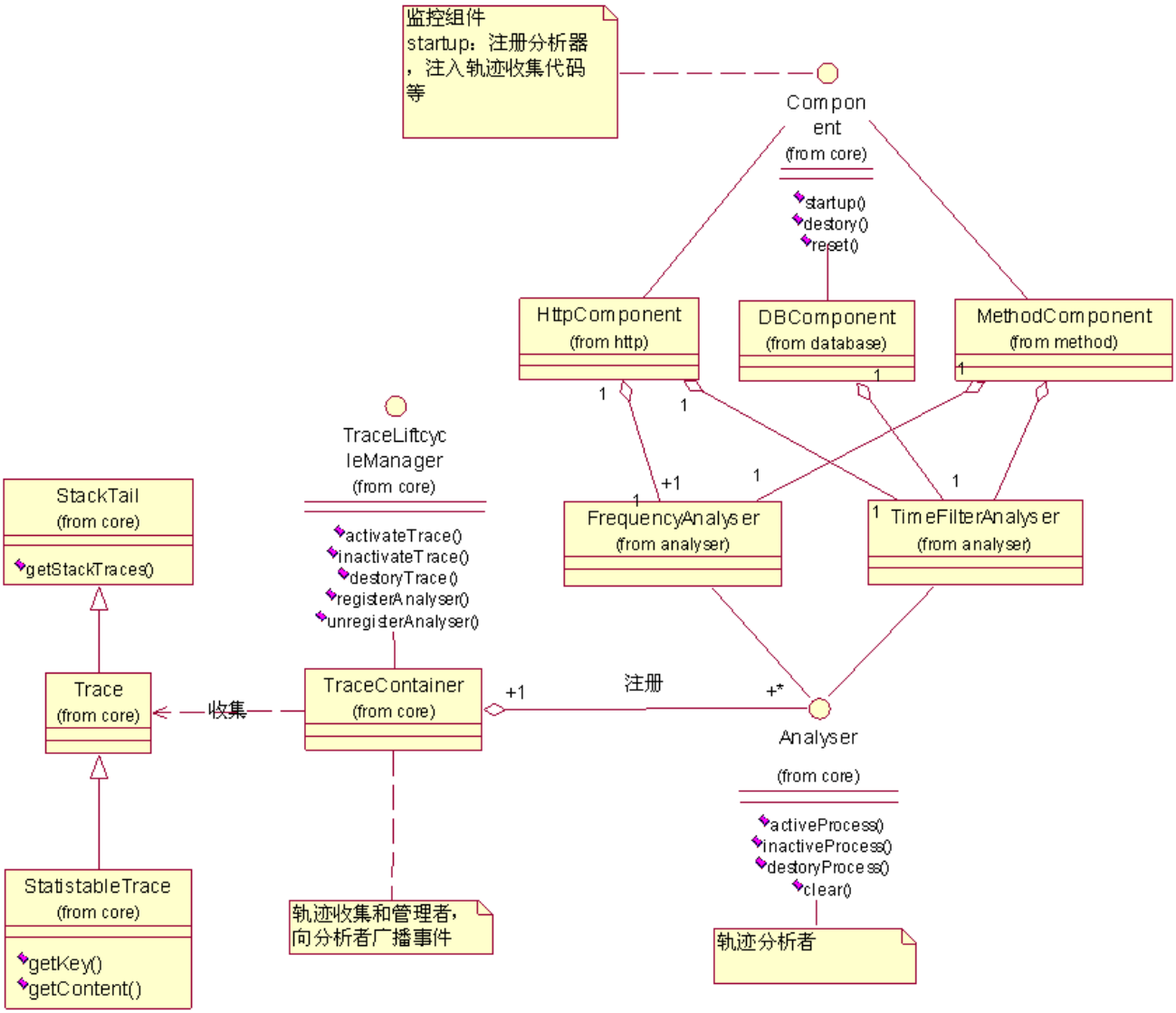
Jwebap Plugin 包含多个 Component 实现，实现各种各样的分析功能。Component 作为 Plugin 之间可复用的功能单元组件集成于 Plugin 内部，而 Plugin 则作为部署和加载意义上的模块，以插件的方式集成于 Jwebap,当不需要时，随时可以去掉。

Plugin 可以通过插件 jar 包中的 plugin.xml 定义，对于插件参数的修改以及插件的部署，Jwebap 提供 Jwebap Console 进行界面化的配置。Plugin.xml 如下例:

```
<component>
    <class>org.jwebap.method.MethodComponent</class>
    <trace-filter-active-time>-1</trace-filter-active-time>
    <trace-max-size>1000</trace-max-size>
    <!--要监控的类，目前还不可以配包名，如test.*，以后会增加这个功能，多个类名之间通过;分隔-->
    <detect-clazzs>
    test.*;
    </detect-clazzs>
</component>
<component>
    <class>org.jwebap.database.DBComponent</class>
    <trace-filter-active-time>-1</trace-filter-active-time><!--ms -->
    <trace-max-size>1300</trace-max-size>
    <connection-listener><!--监听器，可配置多个,以;分隔-->
       org.jwebap.http.ServletOpenedConnectionListener;org.jwebap.method.MethodOpenedConnectionListener
    </connection-listener>
    <!--
      1)本地连接池:可以采取配置驱动的方式，动态监控，比如应用采用oracle,mysql
      两种数据库，驱动分别是orcale.jdbc.driver.OracleDeriver和
      com.mysql.jdbc.Driver，那么在DBComponent配置项中配置
      driver-clazzs=orcale.jdbc.driver.OracleDeriver;com.mysql.jdbc.Driver

      2)jndi远程数据源:可以配置Application中用于获取连接的类作为驱动(比如
      com.ConnectionManager).Jweabp启动时对这些驱动类进行注入，使得所有
      该类方法返回的Connection具有监控的功能.
    -->
    <driver-clazzs></driver-clazzs>
</component>
```

Jwebap 内部对 Component 的管理模型如下:

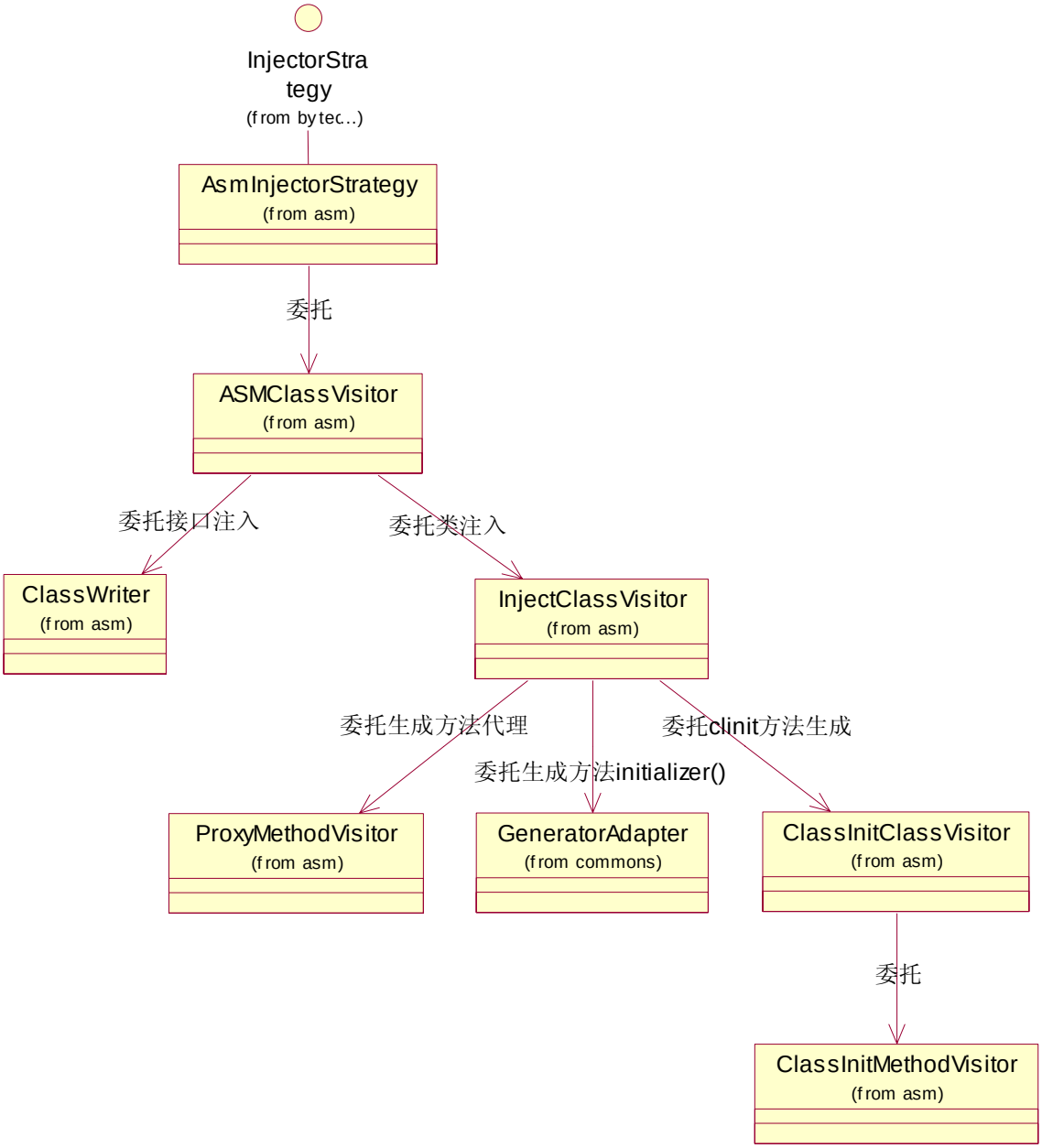



# 4 Class-Working

Class-Working 在 Jwebap 里面是一个很重要的基础构件。轨迹的收集，目前都是通过字节码操作注入到目标类里面的,同时 Jwebap 对类的增强，不同于各种热加载或者 agent 代理的方式，能够实现让增强的字节码更安全的加载于中间件内部，而不需要为 JVM 或者中间件做任何配置。

Jwebap默认实现了基于ASM-2.1( asm.objectweb.org )Class-Working

构件。结合ASM的访问者模式，Jwebap的注入模型如下：




# 5 轨迹钝化（未实现）

对于不活动的轨迹（参照轨迹生命周期），轨迹容器按照一定的算法钝化轨迹到存储设备（可能是磁盘，可能是数据库）。钝化的轨

迹文件用作一些特定的统计分析。

也可以日后开发客户端分析工具，进行全量信息的分析。

# 6 钝化文件分析（未实现）

客户端钝化文件分析工具。进行全量的数据分析。

# 7 Tracer 插件

Tracer 插件包含了 3 个 Component。实现包含 http 请求监控统计，method 调用的监控和统计，jdbc 执行的监控和统计。可以设定阀值，帮助找到系统的瓶颈。

## 7.1 HttpComponent

### 7.1.1概述

HttpComponent 在 Jwebap 中是个非常简单的组件，但是却非常有用。它可以抓取所有的 http 请求，进行分析，包括时间阀值，执行次数，平均执行时间，请求清单，对于不想记录的请求，可以配置 exclude url pattern，进行滤除。

同时，HttpComponent 可以结合 JdbcComponent，监听请求内执行的 sql，进行统计。

### 7.1.2配置参数

trace-filter-active-time: (ms)超过该毫秒数的执行轨迹将被记录

trace-max-size: 最大记录的超时轨迹数

## 7.2 MethodComponent

### 7.2.1概述

MethodComponent 与其说是一项功能，不如说是一种机制。灵活的使用,可以达到更多的监控效果。组件本身可以通过配置 java 包名或类名，实现对这些包和类的方法进行监控。

一般来说，应用系统实现，会采取分层结构，使用 MethodComponent 有针对性的对某个层次或模块进行监控，更能有效帮助找到系统瓶颈，同时又不会因为监控范围过大带来不必要的开销。一般可以使用它，对服务层（比如 EJB, WebService），或者业务层，甚至 DAO 层进行监控，根据系统实际情况，选取合适的粒度进行分析监控，比泛泛的全量监控，更有助于找到系统瓶颈。

和 HttpComponent 一样，MethodComponent 也可以结合 JdbcComponent，实现对方法内执行的 sql 进行跟踪统计。

### 7.2.2配置参数

trace-filter-active-time: (ms)超过该毫秒数的执行轨迹将被记录

trace-max-size: 最大记录的超时轨迹数

detect-clazzs: 要监控的类，可以配置包名和类名，如test.*;test.Test,但不支持通配符,多个类名之间通过;分隔

## 7.3 JdbcComponent

### 7.3.1概述

JdbcComponent 是 Jwebap 中，最常用的组件。对于一般的企业应用，数据库分析往往能够最直接找到瓶颈所在。JdbcComponent 组件能够实现对连接的监控，设置时间阀值，监控连接泄漏，监控连接打开的 Statement，和所有执行的 sql，**同时可以找出执行 sql 的代码行数**，这点非常有用。

组件对于 jdbc 的监控，是全范围的，可以支持多个数据源的应用，也可以支持非本地数据源(jndi 数据源)。

### 7.3.2配置参数

trace-filter-active-time: (ms)超过该毫秒数的执行轨迹将被记录

trace-max-size:最大记录的超时轨迹数

connection-listener: Connection 监听器，可配置多个,以;分

隔

**driver-clazzs:**

- 对于本地数据源的配置，**driver-clazzs** 参数可以配置自己应用的 ConnectionManager 类，也可以配置连接池的 Datasource 类，c3p0: com.mchange.v2.c3p0.ComboPooledDataSource; dbcp: org.apache.commons.dbcp.BasicDataSource。可以配置，任何用于管理连接的类，jwebap 会对 driver-clazzs 方法中返回的所有 Connection 和 Datasource 对象进行监控。**值得注意的是：本地数据源不建议直接配置数据库驱动作为 driver-clazzs**，由于连接池的缘故，配置数据库驱动作为 driver-clazzs 会造成 jwebap 误以为所有的连接都泄漏了。
- 对于 jndi 数据源，driver-clazzs 参数配置（很多 Jwebap 的使用者无法正确配置这点，特此注意）需要配置本地用于获取连接或者数据源的类，举例，应用采用 spring 的 JndiObjectFactory 获取 jndi 数据源对象，那么 **driver-clazzs** 就可以设为 org.springframework.jndi.JndiObjectFactoryBean; 再比如，应用是自己封装获取 jndi 数据源的方法，那么也可以配置 **driver-clazzs** 为那个类。**Jwebap 会在启动时，对 driver-clazzs 做字节码注入，对 driver-clazzs 所有返回 Connection 或者 Datasource 对象的方法进行监控以获取 sql 信息。**

# 8 模版引擎

为了部署方便，Jwebap 开发 Web 统计视图，通过集成第三方的模版引擎实现。在集成层，Jwebap 实现客户端 MVC 框架,模版引擎实现模版语言与容器的实现。

Jwebap 通过 Servlet 作为输出的控制器。

# 9 部署

## 9.1 Step I

1) 把 Jwebap_*.*.*_*.jar 放到应用的 ClassPath 下。
2) 把依赖的jar包也放到应用的ClassPath下(参照11依赖性说明)。
3) 把 jwebap.xml 放到工程 web module 任意目录中。

## 9.2 Step II

修改 web.xml,增加全局参数 jwebap-config,指向 jwebap.xml 的位置

```
<context-param>
        <param-name>jwebap-config</param-name>
        <param-value>/WEB-INF/jwebap.xml</param-value>
</context-param>
```

配置 Jwebap 启动类

**注意：配置在所有 listener 之前，以保证 Jwebap 最先启动，这点对于类增强很重要。**

```
<listener>
        <listener-class>org.jwebap.startup.JwebapListener</listener-class>
</listener>
```

配置 HttpComponent 的过滤器。

```
<filter>
        <filter-name>PageDetectFilter</filter-name>
        <filter-class>org.jwebap.plugin.tracer.http.DetectFilter</filter-class>
        <init-param>
            <param-name>excludeUrls</param-name>
            <param-value>/detect;/detect/*;*.js;*.jpg;*.htm;*.html;*.gif;*.png;*.css;*.sw
        </init-param>
</filter>
<filter-mapping>
        <filter-name>PageDetectFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

配置 Jwebap Console 视图 Servlet。

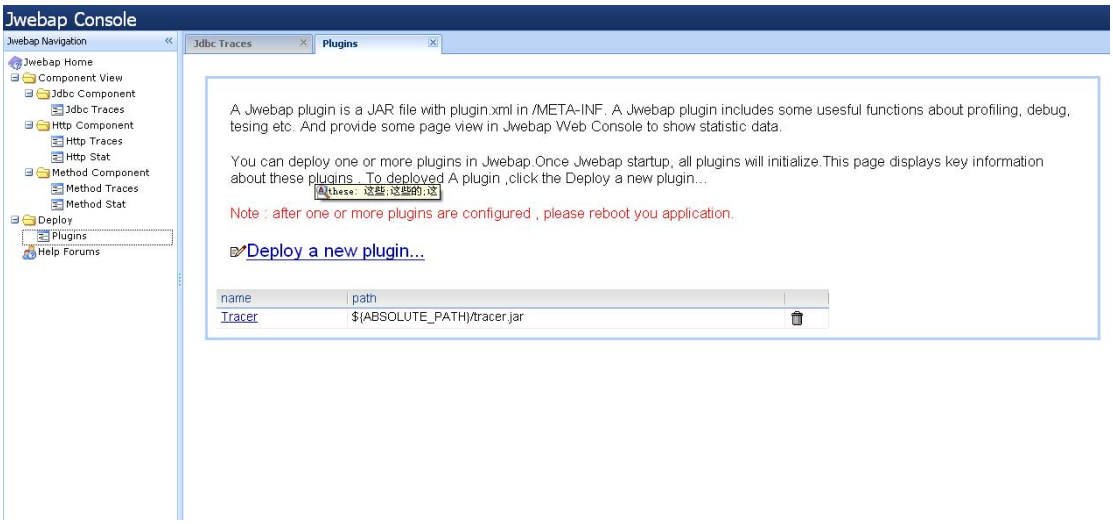
```
<servlet>
        <servlet-name>detect</servlet-name>
        <servlet-class>org.jwebap.ui.controler.JwebapServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>detect</servlet-name>
        <url-pattern>/detect/*</url-pattern>
```

```
</servlet-mapping>
```

## 9.3 Step III

1) 把 tracer.jar（**默认的 jwebap 插件，配置在 jwebap.xml 的 plugin-ref 里**）放到应用的 ClassPath 下。

2) 启动应用，如果配置无误，这时候，通过访问 JwebapServlet 就可以打开 Jwebap Console 界面了！



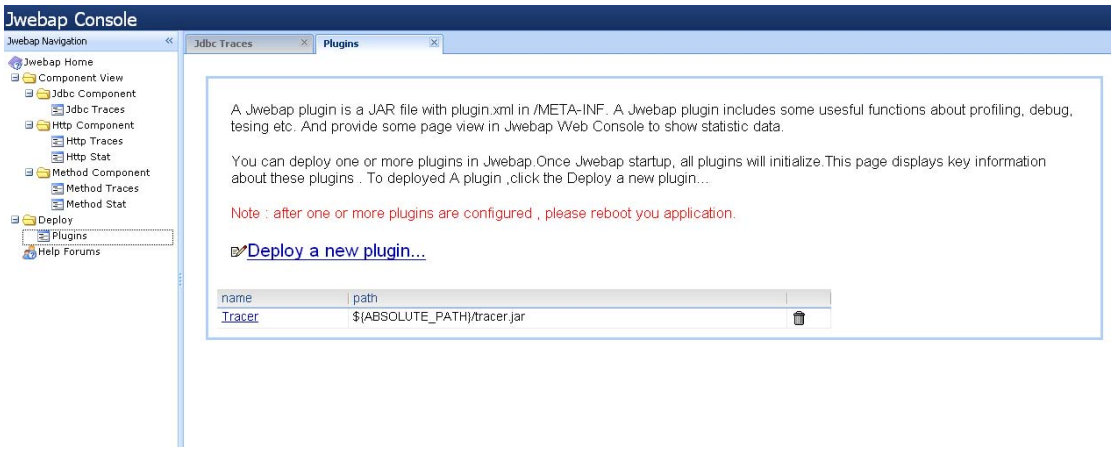

## 9.4 Step IV

从 0.6 版开始，jwebap 已经采用完全的 plugin 插件设计，所有的功能在部署上都采用插件方式进行集成。

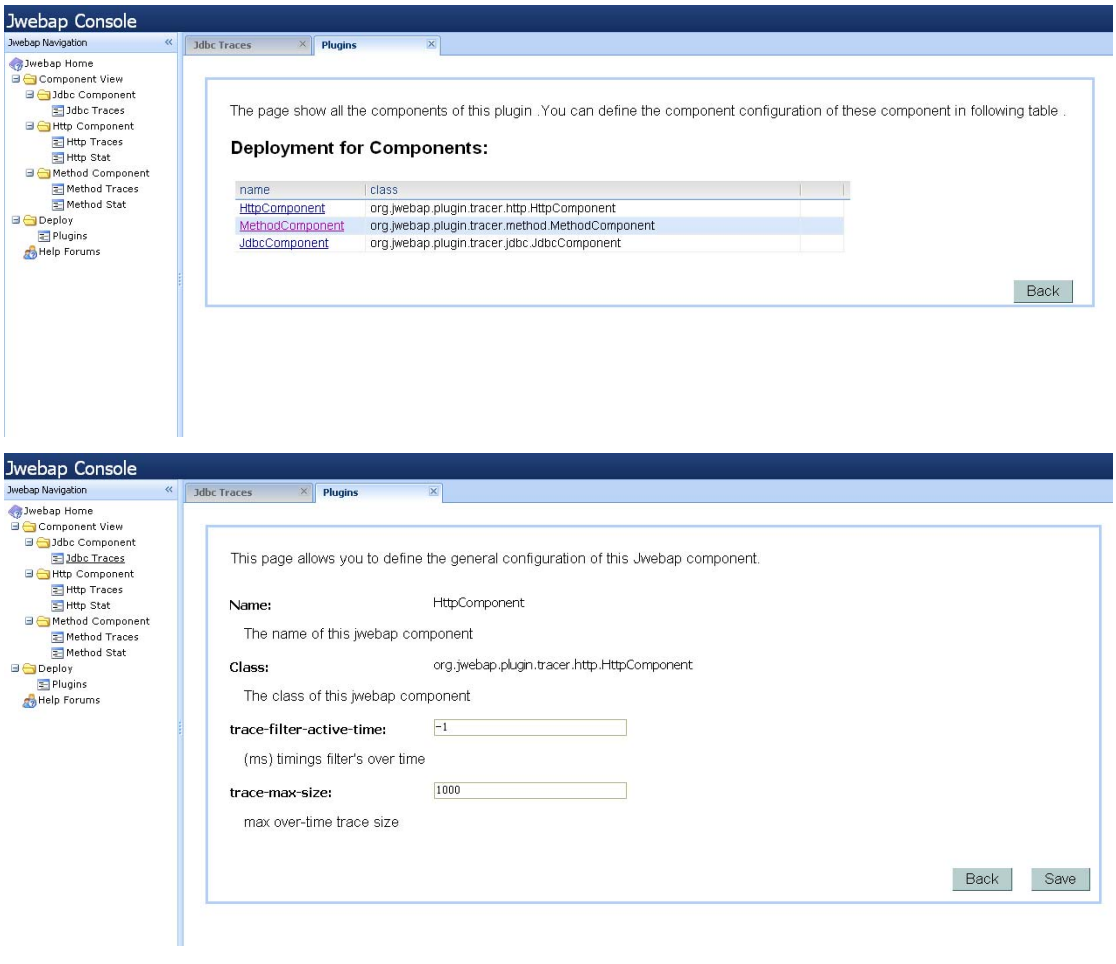
Jwebap 默认提供了 Tracer 插件，实现了 jdbc，method，http 三方面的监控统计功能。

对于插件的部署和参数配置，Jwebap Console 提供了可视化界面进行配置：

- 部署插件：



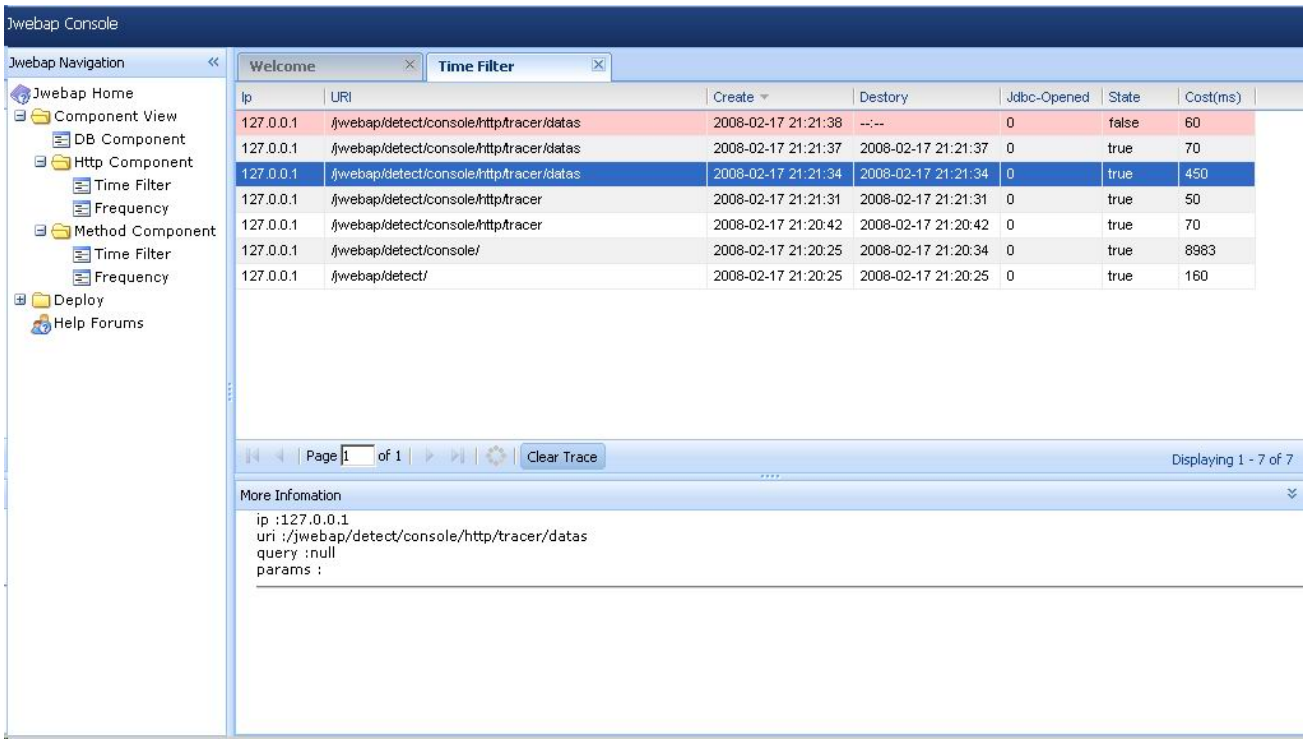

- 配置插件的运行参数（对于Tracer插件的参数配置说明参照7Tracer插件）:



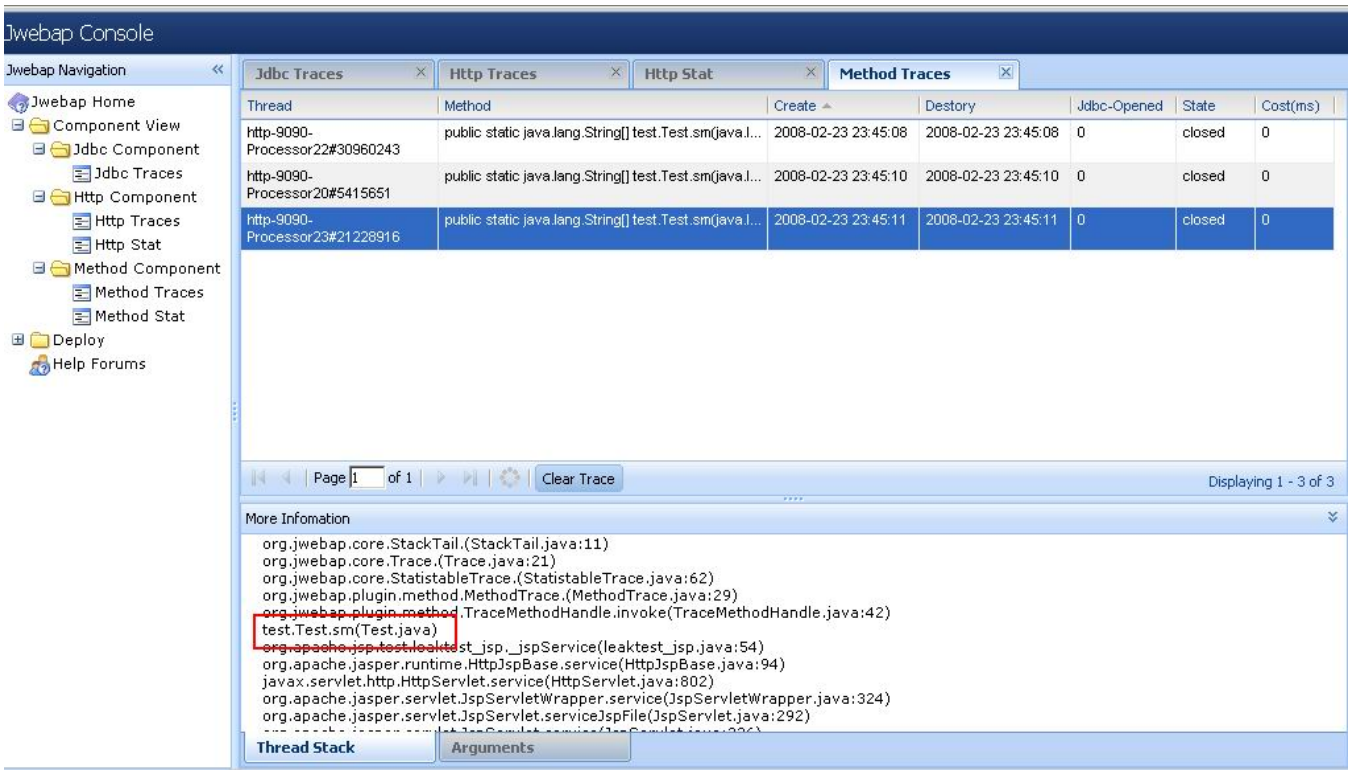

**配置完插件后，重启应用，以使更改生效。**

# 10 ScreenShot

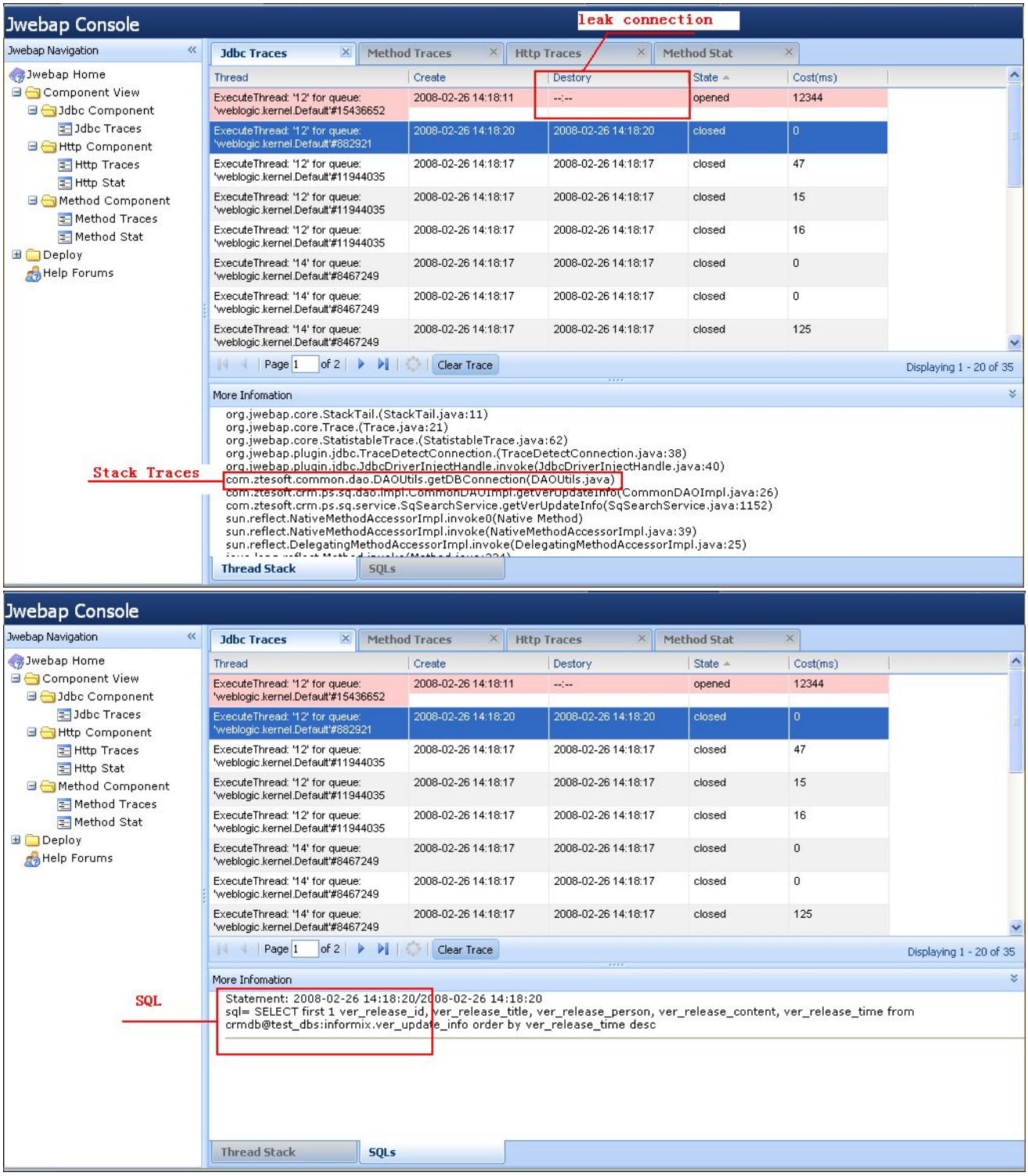
## 10.1 Http Traces



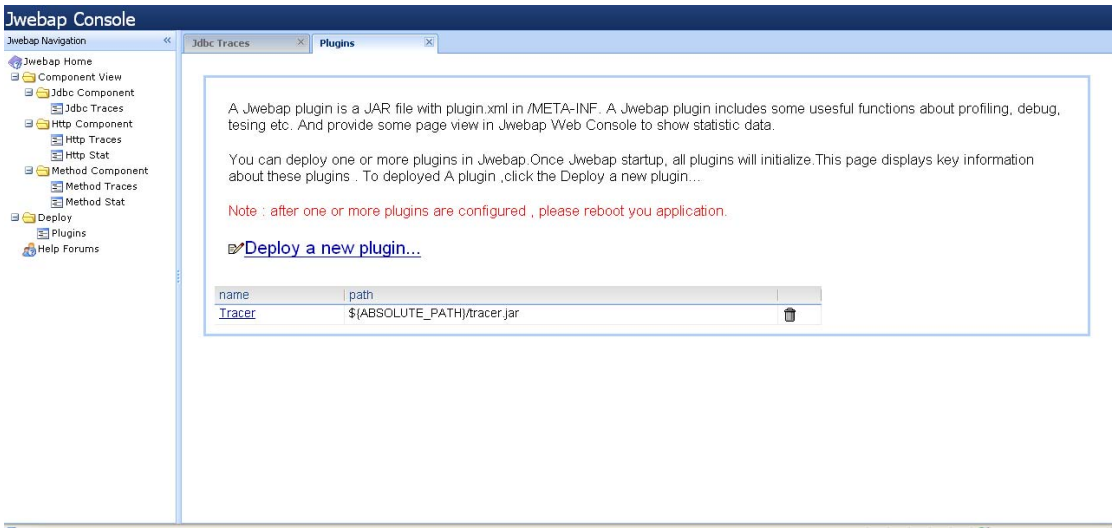
# 10.2 Method Traces



# 10.3 Jdbc Traces



## 10.4 Plugin Deploy




# 11 依赖性说明

1) JDK1.4 或以上版本

2) commons-digester-1.7.jar

3) commons-betwixt-0.8.jar

4) commons-beanutils.jar

5) commontemplate-0.8.1.jar

6) commons-logging.jar

配置文件有:

1) jwebap.xml

中间件支持:

Tomcat4+, WebLogic8+, WAS6+, JBoss

# 12 licence 声明

Jwebap 采用 Apache Licence,内部采用了 ASM-2.1RC 组件，出于 ASM-2.1RC 兼容性的考虑，把 ASM-2.1RC 集成到了组件内部，特此声明。

# 13 Jwebap-SDK(开发自己的 Plug-in)