基于 JMX 的 HTML 适配器的设计与实现

Design and Implementation of HTML Adaptor Base on JMX

吕 律　刘光昌　(广州 暨南大学 电子工程系 510632)

摘要:JMX(Java Management Extension)是 Sun 公司推出的一套网络资源管理的标准,由三层组成,即设备层,代理层和分布式服务层。本文主要侧重于分布式服务层中的 HTML 适配器的实现。

关键词:JMX HTML 适配器 MBean 动态管理

1 引言

JMX(Java Management Extension)是 Sun 公司推出的一套网络资源管理的标准。JMX 提供了一个可扩展的管理体系结构,每一个 JMX 代理都是相互独立的,而且可以根据需求随时在管理器中注册或注销。目前大部分的 J2EE 应用服务器,如 IBM 的 WEBSPHERE、BEA 的 WEBLOGIC、JBOSS 等都是建立在 JMX 的基础上的。不仅如此,JMX 还支持其他的管理协议,如 SNMP、WBEM、TMN,这大大加强了 JMX 的可扩展性,使得 JMX 更适应异构网络的管理。

JMX 标准定义了三层,设备层和代理层的源代码是公开的,为学习和完善 JMX 提供很大的便利。而第三层分布式服务层的实现则留给第三方软件开发者,目前主要的分布式服务层应用软件有 Sun 的 JDMK(Java Dynamic Management Kit), AdventNet 的 Agent Toolkit, ManageEngine? JMX Studio 等。这些商用软件的特点都是基于 HTML 适配器,也就是现在流行的通过网页浏览器的方式进行管理。本文将对 JMX 的 HTML 适配器的关键技术的设计和实现进行探讨。

2 关键技术的设计和实现

JMX 是由设备层、代理层和分布式服务层三层组成。Sun 在 JMX 标准中实际上只是对前面两层,即设备层和代理层给出详细的定义。而第三层分布式服务层主要由第三方厂商提供基于各种网络协议的适配器。目前主要的商用软件有 Sun 的 JDMK(Java Dynamic Management Kit),AdventNet 的 Agent Toolkit,ManageEngine? JMX Studio 等。以下将主要介绍 HTML 适配器的实现。

根据 JMX 标准,HTML 适配器的任务是监听客户端的连接和操作,实现远程管理网络资源的方式。所以 HTML 适配器主要是由两个部分组成:

(1) HTML 服务器,用于监听用户的连接,产生新的线程处理用户的操作。

(2) HTTP 报头处理器,因为用户不同的操作会产生不同的 HTTP 报头,所以根据这些不同的 HTTP 报头再对 JMX 的设备层和代理层进行操作。

2.1 HTML 服务器的实现

(1) 首先实例化 HTML 适配器。根据 JMX 标准,代理层的对象并不直接跟其他的管理应用程序通信,所以 HttpAdaptorServer 的实现是要继承 JMX 提供的动态 MBean 的接口。然后通过注册这个动态 Mbean 来和代理层通信

```
public class HttpAdaptorServer extends
implements MBeanRegistration, DynamicMBean{
。。。。。}
```

实例化 HTML 适配器时有两个关键的参数,port 和 notificationBroadcaster。Port 是 HTML 适配器的监听端口,用于监听客户端的连接,notifBroadcaster,就是 2.1 中提到的在设备层中的 Notification 机制,它是对 JMX 的 javax. management. NotificationBroadcasterSupport 的具体实现。

```
。。。。。。
notificationBroadcaster = new NotificationBroadcasterSupport();
    port = 8088;
。。。。。。
```

(2) 根据 JMX 的标准为(1)中构造的 HTML 适配器实例化一个名字 name=http,port=8088

```
http_name = new ObjectName("Adaptor:name=http,port=8088");
```

ObjectName 将调用 JMX 中的 javax. management. ObjectName. java,ObjectName 的作用是产生一个符合 JMX 标准的 Mbean 的名字。

(3) 根据 JMX 的标准注册 HTML 适配器

```
server. registerMBean(http, http_name)
```

; registerMBean 是 JMX 提供的注册函数,它包含两个

参数，一个是实现 HTML 适配器的 Mbean 的类的对象，另一个是符合 JMX 标准的 HTML 适配器的名字。

(4) 最后启动 HTML 适配器

启动时将产生一个新的线程，监听客户端的连接

```
……
mainThread = new Thread(this, "Adaptor:name=http,port=8088");
mainThread.start();
……
```

在新的线程中，将实现端口的监听，并等待客户端的连接

```
run(){……
    sockListening = new ServerSocket(8088); //初始化要监听的端口
    ……
    sock = sockListening.accept(); //监听端口，等待连接
    ……}
```

以上就是 HTML 服务器的实现，下面接着就是监听客户端的连接和操作。

2.2 HTTP 报头处理器的实现

(1) 当用户通过浏览器管理时，将唤醒 2.1 中(4)里提到的正在等待的线程，实例化 HTTP 的报头处理：

```
HttpRequestHandler httprequesthandler = new HttpRequestHandler(sock, this, objectName);
```

在实例化 HTTP 报头处理器时，关键是产生一个新的线程，并由这个新的线程处理 HTTP 报头，然后结束旧的线程，以便让 HTML 适配器继续监听其他的连接。在 HTTP 报头处理器实例化的同时通过 JAVA 的 synchronize 机制保证 HTML 适配器的同步。

(2) 读取 HTTP 报头。客户端不同的操作将产生不同的 HTTP 报头。

```
httprequesthandler.readFrom(sock.getInputStream());
```

在读取 HTTP 报文的过程中，一个很重要的参数 request，request 将记录 HTTP 报头中用户操作的参数。Request 的格式为起始字符加上操作的参数。由 request 的起始字符决定 HTTP 报头处理的操作。

(3) 处理 HTTP 报头，在(2)中读取 HTTP 报头的基础上，对 HTTP 报头的处理将根据 request 的起始字符来决定。request 有以下 6 种起始字符：

① "ObjectRes"：客户端查看具体某个网络资源的参数；

② "Refresh"：客户端将要设置的自动加载网页的时间；

③ "SetForm"：客户端将要设置的网络资源的属性的参数；

④ "Action"：客户端将对网络资源进行的操作；

⑤ "Admin"：客户端对网络资源的管理、产生或注销；

⑥ "Index"：客户端首次启动 HTTP 适配器时，查看到的网络资源。

2.3 6 种 HTTP 报头的处理

(1) 当 request.startsWith("ObjectRes ")时，即客户端要查看某个具体的 MBean 的参数。首先获得所要参看的 MBean 对象，然后再由 JMX 提供的接口查询该 MBean 所有的属性和操作。

```
……
mbeanObj = getObjectNameByObjNameStr(sMBean);
//sMBean 是所要查看的 MBean 对象的名字，例如 2.1(2)中"Adaptor:name=http,port=8088"
//mbeanObj 是所要查看的网络资源对象(即 MBean，根据 JMX 标准 MBean 代表网络资源)，
attributeinfo[] = mbeanObj.getAttributes();
//attributeinfo 将得到所查看 Mbean 的所有的 Attributes。
propertyName = new String[attributeinfo.length];
for(int i = 0; i < propertyName.length; i++) //propertyName.length 表示所查看的 Mbean 的个数
{……
String sType = usualType(attributeinfo[i].getType()); //是 propertyName 对应的参数的类型
String sDescribtion = attributeinfo[i].getDescription(); //是 propertyName 对应的参数的描述
……}
MBeanOperationInfo mbeanoperationinfo[] = mbeanObj.getOperations();
//mbeanoperationinfo 将得到所查看 Mbean 的所有的操作的属性
for(int i = mbeanoperationinfo.length - 1; i >= 0; i--)
{……
String sOperationDescription = ambeanoperationinfo[i].getDescription();
//将得到所查看 Mbean 的所有的操作的属性的描述
……}
```

(2) 当 request.startsWith("Refresh ")时，即客户端将要设置的自动加载网页的时间。这里不需要对代理层进行任何的操作，只要修改返回给客户端的 HTTP 报头

```
httphead = "<META HTTP-EQUIV=REFRESH CONTENT="";
httpheaf += autoRefreshTime + "; URL=" + "/
```

Refresh" + sMBean + ""＞";

//autoRefreshTime 是自动加载网页时间的值，sMBean 是所要自动加载的 Mbean 的名字

(3) 当 request. startsWith("SetForm ")时，即客户端将要设置的网络资源的属性的参数。

首先获得所要修改的 MBean 的对象，再用新值替换旧的值。最后根据 JMX 标准将新的值加入 MBean 的属性表。

mbeanObj = getObjectNameByObjNameStr (sMBean);

// sMBean 是所要查看的 MBean 对象的名字，例如 2.1(2)中"Adaptor:name=http,port=8088"

//mbeanObj 是所要查看的网络资源对象(即 MBean，根据 JMX 标准 MBean 代表网络资源) Attribute attribute = new Attribute(sOldValue, sNewValue);

// sOldValue 是原来的值，sNewValue 是新的值，Attribute 将设置新的值

attributelist. add(attribute);

attributelist1 = mbs. setAttributes (mbeanObj, attributelist);

//mbs 是整个代理层的 MBean 对象，mbeanObj 则是所要进行处理的 MBean 对象

(4) 当 request. startsWith("Action ")时，即客户端将对网络资源进行的操作。正如 2.1 所述，HTML 适配器本身也是一个动态 MBean。所以通过 JMX 的 invoke()实现对所要管理的资源，即 MBean 的操作。

mbeanObj = getObjectNameByObjNameStr (sMBean);

// sMBean 是所要查看的 MBean 对象的名字，例如 2.1(2)中"Adaptor:name=http,port=8088"

//mbeanObj 是所要查看的网络资源对象(即 MBean，根据 JMX 标准 MBean 代表网络资源)

obj = mbs. invoke(mbeanObj, sOperationName);

//mbs 是整个代理层的 MBean 对象，mbeanObj 则是所要进行处理的 MBean 对象

//sOperationName 是所要操作的名字，如"stop"

(5) 当 request. startsWith("Admin ")时，即客户端对网络资源的管理、产生或注销。在获取所要注销或新注册的 MBean 后，通过 JMX 的 unregisterMBean 和 createMBean 实现

mbeanObj = getObjectNameByObjNameStr (sMBean);

// sMBean 是所要查看的 MBean 对象的名字，例如 2.1(2)中"Adaptor:name=http,port=8088"

//mbeanObj 是所要查看的网络资源对象(即 MBean，根据 JMX 标准 MBean 代表网络资源)

当注销一个 Mbean 时 mbs. unregisterMBean (mbeanObj);

当新注册一个 Mbean 时 mbs. createMBean (mbeanObj);

//mbs 是整个代理层的 MBean 对象，mbeanObj 则是所要进行处理的 MBean 对象

// unregisterMBean，和 createMBean 是 JMX 提供的产生或注销 MBean 的函数

(6) 当 request. startsWith("Index")时，即客户端首次启动 HTML 适配器时，查看到的网络资源。它将获得代理层的资源，即所用的 MBean，并输出到客户端。

set = mbs. queryNames();

//mbs 是整个代理层的 MBean 对象，queryNames()将得到代理层中所有的 MBean 对象

//set 将保存获得已经注册并启动的 Mbean

for(int k = set. size() - 1; k >= 0; k--)

sMBean[k] = set[k]. toString();

//sMBean[k]将保存所有 MBean 的描述，以输出到客户端

3 结束语

随着网络技术的不断发展，如何管理日益庞大的网络资源和网络服务是摆在我们面前的一个难题，JMX 正是解决这一难题的很好的方案。本文侧重于 JMX 中的 HTML 适配器的实现，使得客户能够通过 HTML 网页很方便的对网络资源、网络服务进行管理。

参考文献

1 Sun Microsystems, Inc. JMX Instrumentation and Agent Specification, V1.0. 2000,7. http://www.java.sun.com.

2 Sun Microsystems, Inc. Java Management Extensions White Paper. 2001, 5. http://www.java.sun.com.

3 J. Steven Perry Java Management Extension. O'Reilly June 2002.