Mule

我的第一次骑骡子旅行

Starup for Mule

Eric Liu(铁手)
www.blogjava.net/steelhand

1 环境需求

1.1 操作系统

目前 Mule 可支持多种操作系统，包括：

- Windows XP SP2 and Windows 2000.
- Linux, Solaris, AIX and HP-UX.
- Mac OSX

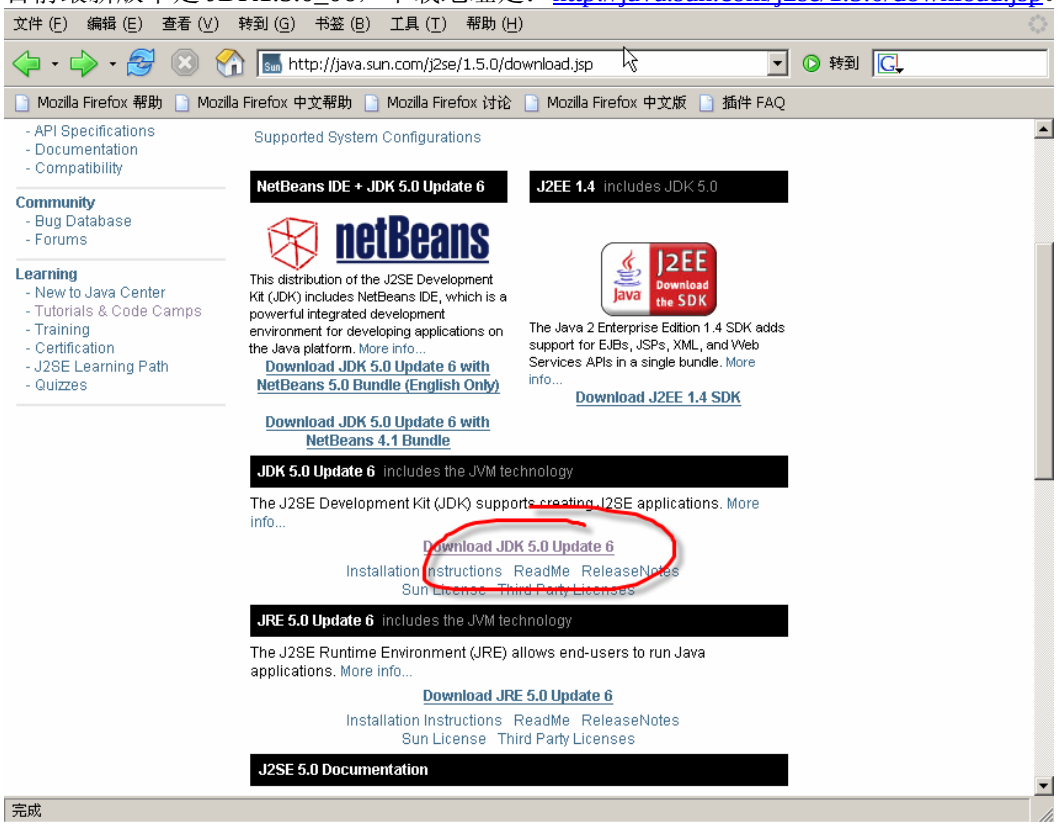
我的机器是 Windows XP SP2，所以以此为例。

1.2 Java 环境

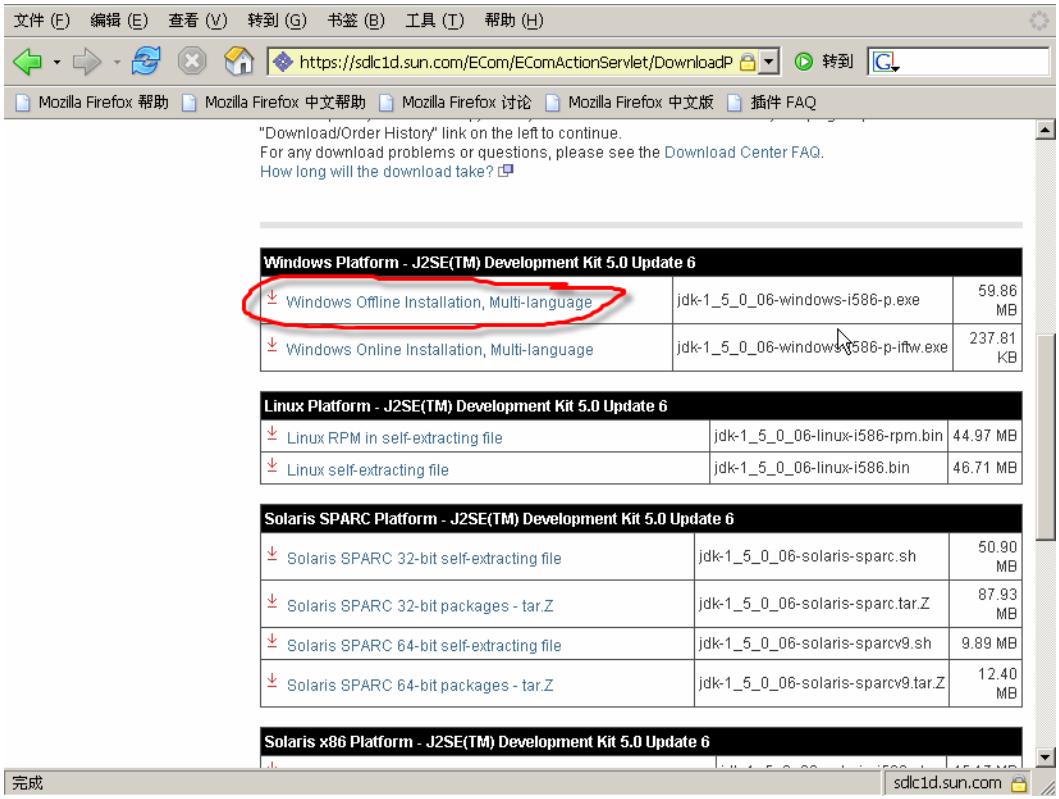
目前可支持 Java 1.4+以上的运行环境，如果需要从源代码自行编译构建，需要安装 JDK1.5+。

1.2.1 下载 JDK1.5。

目前最新版本是 JDK1.5.0_06，下载地址是：http://java.sun.com/j2se/1.5.0/download.jsp。



选择 Offline 安装的版本，它是一个自动的 Windows EXE 安装文件。

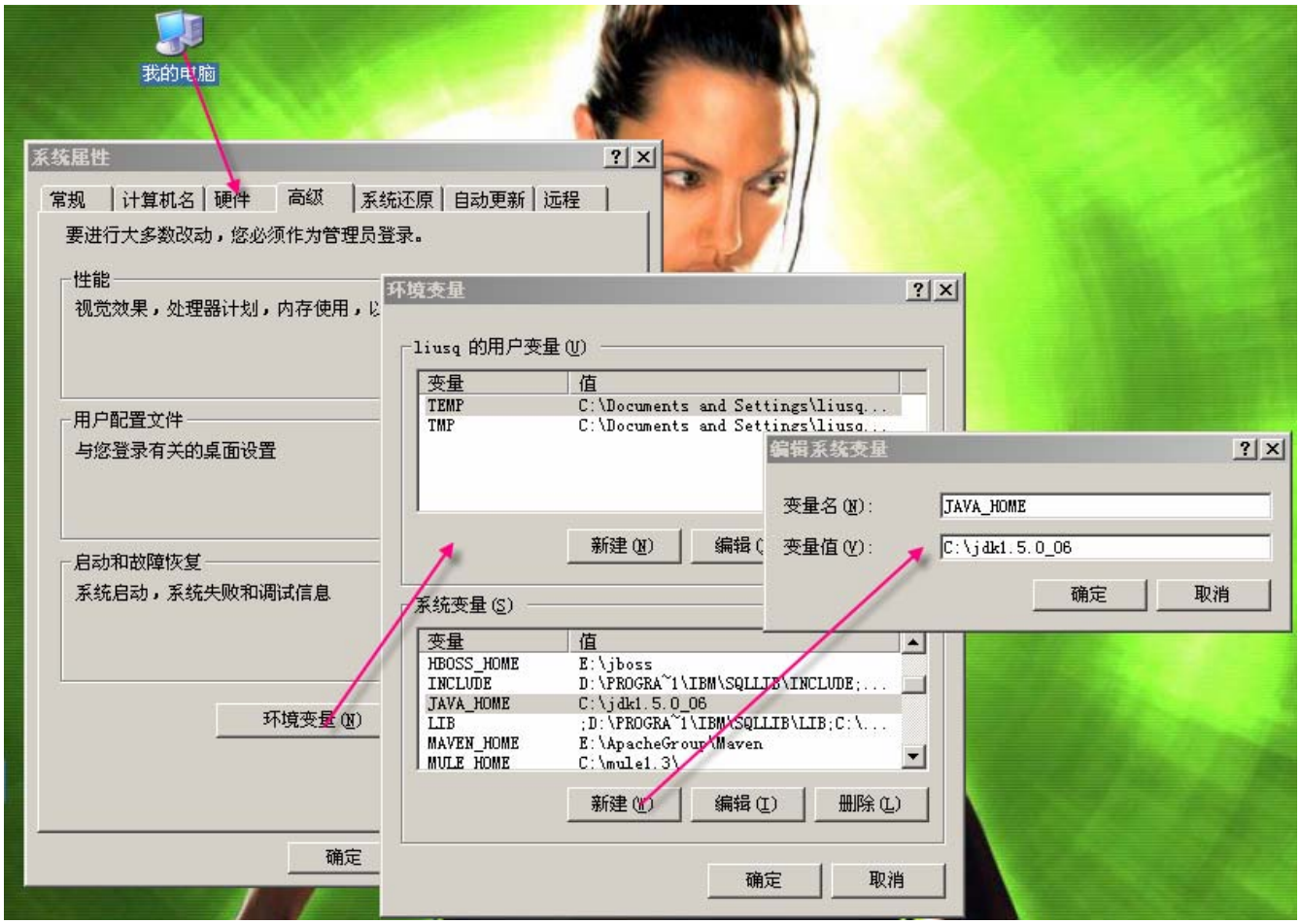


1.2.2 安装

将 JDK 安装在机器上，可选择你想要安装的目录，我们这里设置为：C:\jdk1.5.0_06。
安装时它还会自动安装一个 JRE。

1.2.3 设置环境变量

选择“我的电脑”，在快捷菜单中选择“属性”。在“属性”对话框中选择“高级”页面，选择“环境变量”按钮。在环境变量中选择“新建”，设置 JAVA_HOME 环境变量，值为你安装的 JDK 路径。

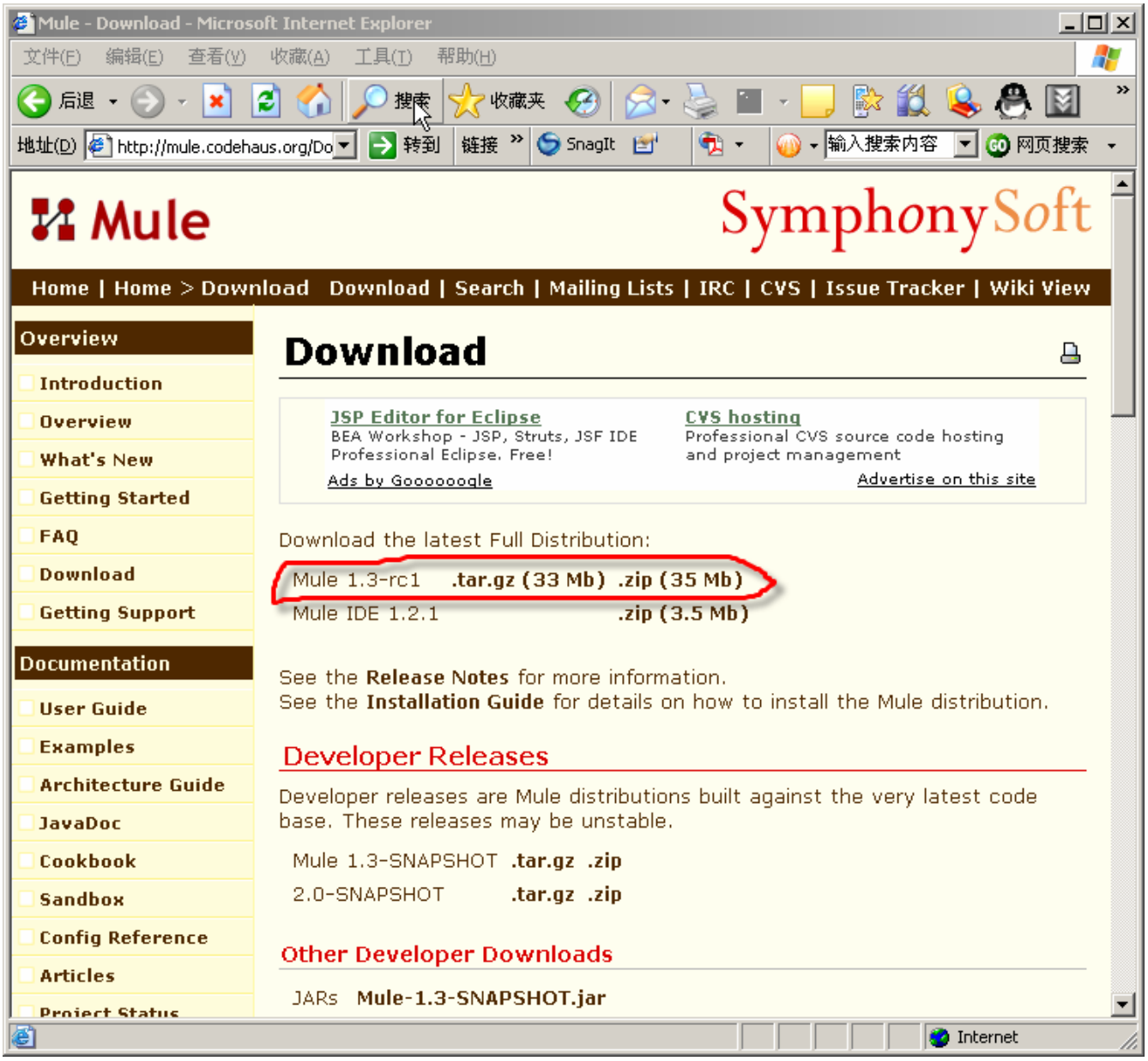


同理，在 Path 环境变量中加入以下路径　%JAVA_HOME%\bin，注意分号，你可以将 JAVA_HOME 修改为你安装的路径。

2 下载和安装 Mule 发行包

2.1 下载

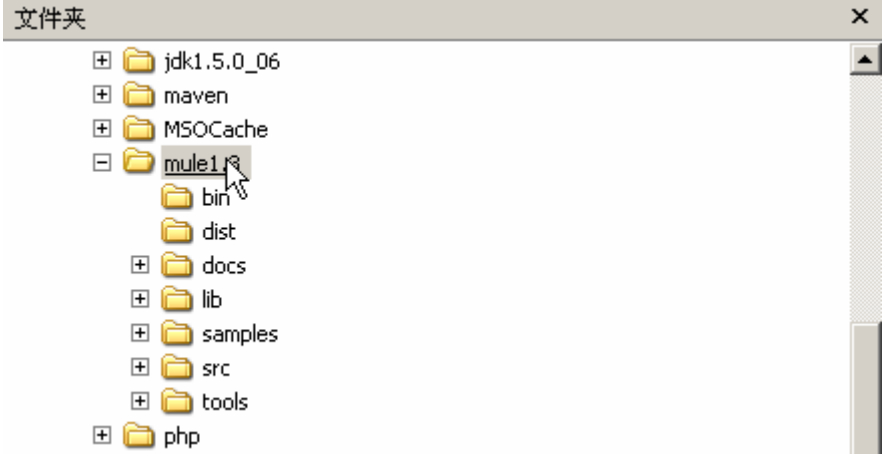
在 Mule 的网站下载 Mule 的发行包。地址是：http://mule.codehaus.org/Download。目前版本是 1.3RC。



有多个版本的发行包可选择，这里我们选择完整的正式发行包。

2.2 安装

安装 Mule 非常简单，将发行包的压缩文件解压到硬盘上即可。我们设置的路径为：C:\MULE1.3。



其中，bin 文件夹包含启动所需的脚本和载入工具。Dist 编译目标文件夹；docs 为 API 和相关文档；lib 为所需的类库；samples 包含附带的实例；src 是 mule 的源代码；tools 包含一些工具。

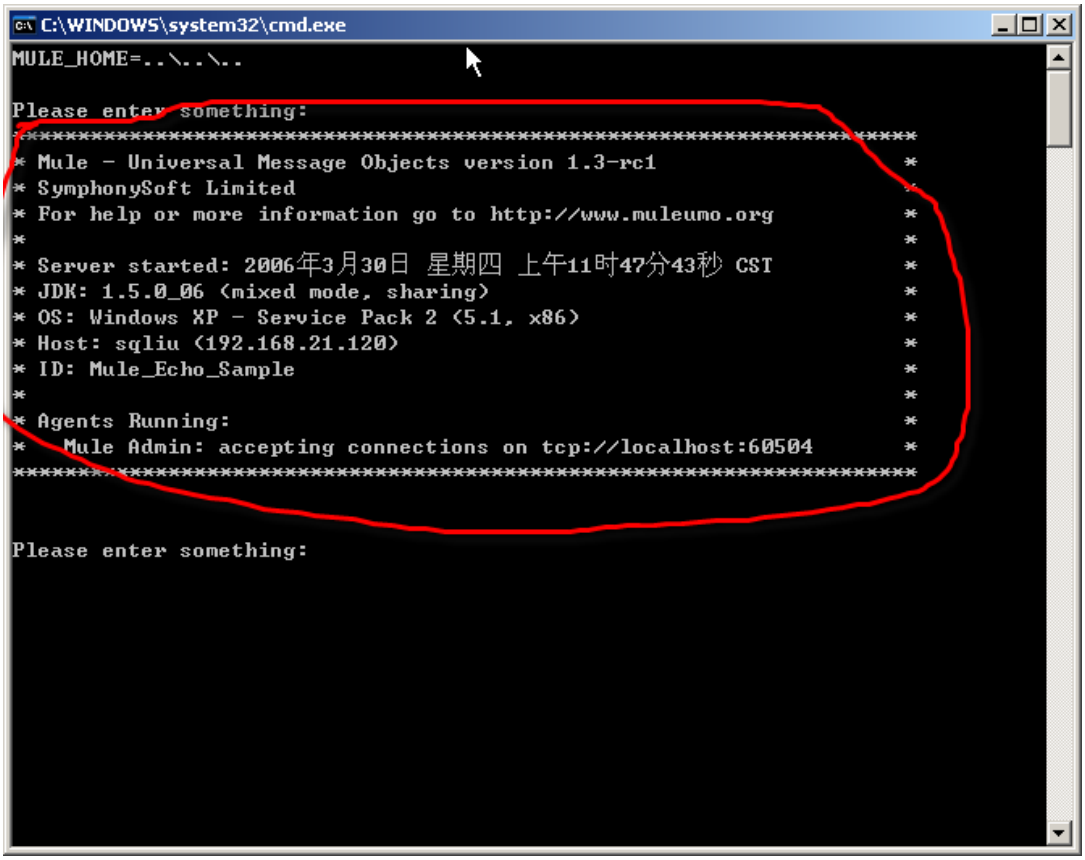
3 运行测试

我们通过一个基本示例来测试是否安装成功。
Mule 发行包中附带了一些基本例子，我们使用 echo 这个例子。它从控制台接受用户输入，然后将信息输出到屏幕上。虽然非常简单，但是信息从输入到反馈输出经过了一个典型的消息路径（我们下面会详细介绍）。

打开一个命令窗口，进入 Mule 安装文件夹下面的 samples\echo\bin 文件夹，运行 mule-echo.bat:

```
Mule-echo
```

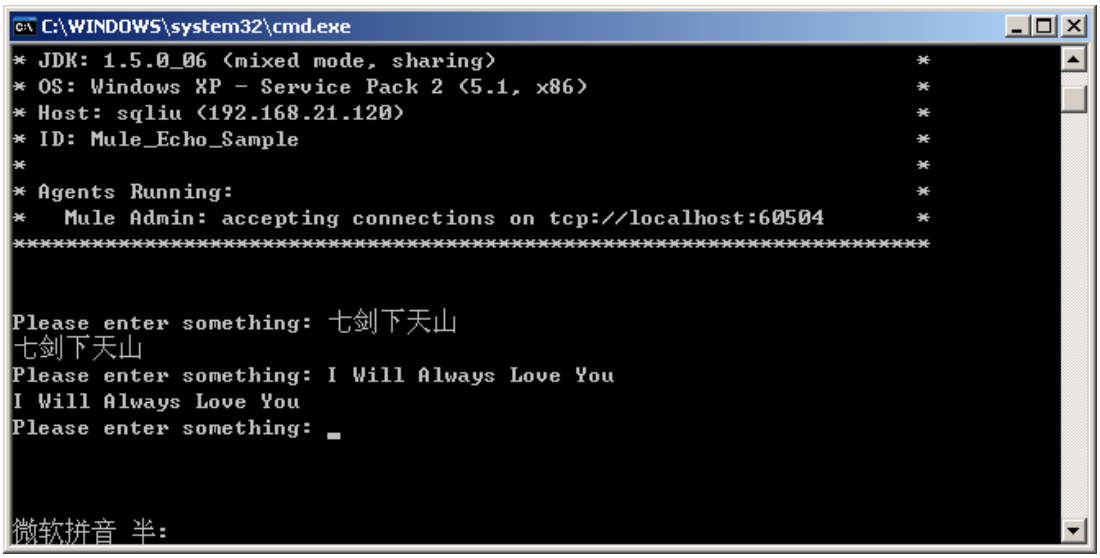
则 Mule Server 会启动，出现一些启示信息，如下图：表示 Mule 可以在 TCP:60504 端口接受连接。



你可以使用命令来检查该端口：

```
netstat -an|find "60504"
```

可以看到，echo 也已经启动，并且出现了提示信息。我们输入任何信息，他都会原样输出到屏幕上。

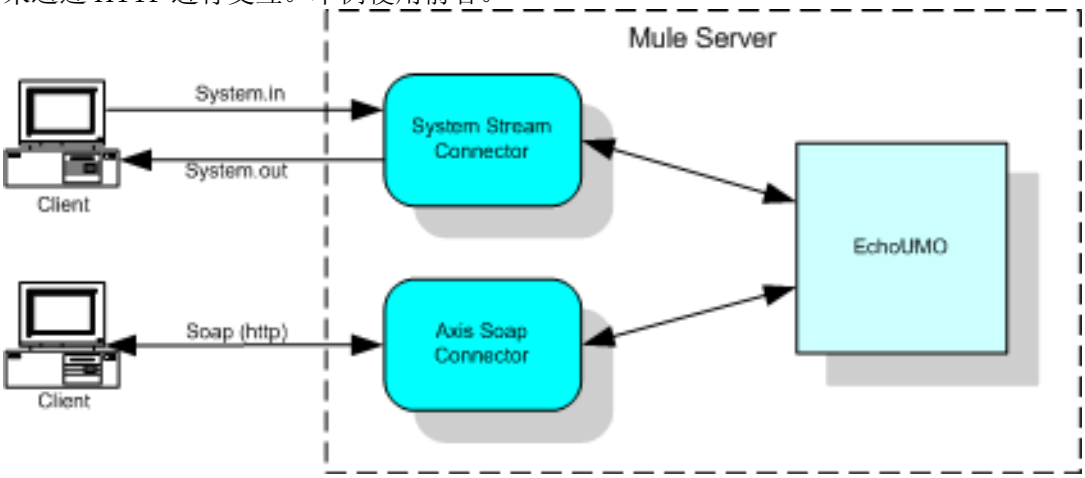


这表示 MULE Server 已经成功运行。
关闭 Mule 只需要关闭控制台窗口，或者 Ctrl+C 即可。

4 Echo 应用之体验

4.1 概述

整个应用的结构如下，Echo 的核心业务功能实现为 EchoUMO(UMO 在 MULE 中被称为统一消息对象)。客户可以通过不同的协议方法与之进行交互。最基本的是通过系统的 Stream 连接器来交互。当然也可以通过 SOAP 来通过 HTTP 进行交互。本例使用前者。



4.2 服务实现

服务首先被定义为一个接口，只有一个方法 Echo（）：

```
public interface EchoService
{
```

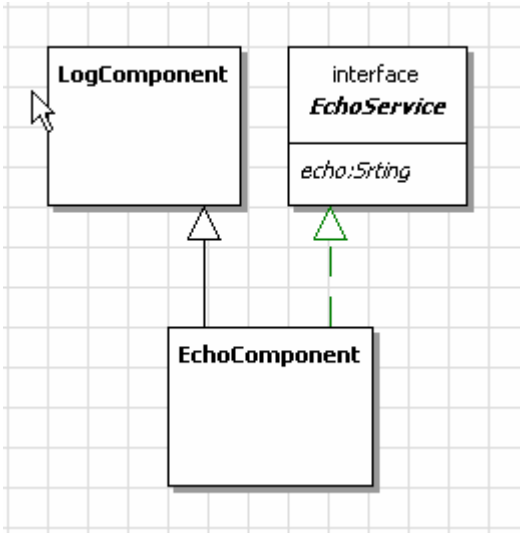
```
    public String echo(String echo);
}
```

具体的实现为 EchoComponent:

```
    package org.mule.components.simple;

    public class EchoComponent extends LogComponent implements EchoService
    {
        public String echo(String echo) {
            return echo;
        }
}
```

EchoComponent 扩展了一个 MULE 的类 logCompoment，以让其具有 log 能力。然后实现了 echo 方法。该方法很简单，紧紧是将输入的参数原样返回。
你可以修改实现来修改其具体行为。
所以，整个类图十分简单，如下所示：



4.3 服务定义

MULE 通过 XML 配置文件来定义服务（这已经是通用方法了☺）。

```
<mule-descriptor name="EchoUMO"
      implementation="org.mule.components.simple.EchoComponent">
</mule-descriptor>
```

定义的元素为 <mule-descriptor>，其中的 name 属性制定了该服务的名称，implementation 指定了具体实现该服务的类。你可以调整着个属性来改变不同的实现版本。
这也是一种 DI（IoC），呵呵。

4.4 定义服务的调用 I/O

定义了服务，还需要定义对其调用的方式，包括进入方式（inbound）和出站方式（outbound）。这实际上是定义服务的端点（EndPoint）。服务端点可以通过多种方式来进行，比如 WS， JMS 等。这里我们首先通过系统的 I/O Stream 来进行。

```
    <mule-descriptor name="EchoUMO"
```

```
        implementation="org.mule.components.simple.EchoComponent">

        <inbound-router>
            <endpoint address="stream://System.in"/>
        </inbound-router>

        <outbound-router>
            <router className="org.mule.routing.outbound.FilteringOutboundRouter">
                <endpoint address="stream://System.out"/>
            </router>
        </outbound-router>
</mule-descriptor>
```

<inbound-router>(入站路由器)元素定义了服务向外暴露的 Endpoint，这里，我们定义的是一个 System Stream，通过地址 Stream://来引用。还有其它一些端点方法，比如，通过 SOAP 暴露为 WS，这样，就可以增加另外的端点定义：

```
    <mule-descriptor name="EchoUMO"
        implementation="org.mule.components.simple.EchoComponent">

        <inbound-router>
            <endpoint address="stream://System.in"/>
            <endpoint address="axis:http://localhost:8081/services"/>
        </inbound-router>

        <outbound-router>
            <router className="org.mule.routing.outbound.FilteringOutboundRouter">
                <endpoint address="stream://System.out"/>
            </router>
        </outbound-router>
</mule-descriptor>
```

斜体部分就将这个服务暴露为一个 Axis WebService，可以通过地址 http://localhost:8081/services 进行访问。

出站端点也定义为一个 System Stream，但是这里使用了一个 Mule 的内建类型，具有额外的过滤能力。

最后就是定义连接 system stream 的 connector,以便可以通过 console 来访问之：：

```
    <connector name="SystemStreamConnector"
        className="org.mule.providers.stream.SystemStreamConnector">
        <properties>
            <property name="promptMessage" value="Please enter something: "/>
            <property name="messageDelayTime" value="1000"/>
        </properties>
</connector>
```

其中 promptMessage 是可以定制的提示信息，出现在 console 中。这里还定义了一个消息的延迟时间，虽然没什么用。

对于暴露为 WS 的服务，这里无须针对其定义专门的 outbound router，它会自动通过 HTTP 返回 SOAP 消息。

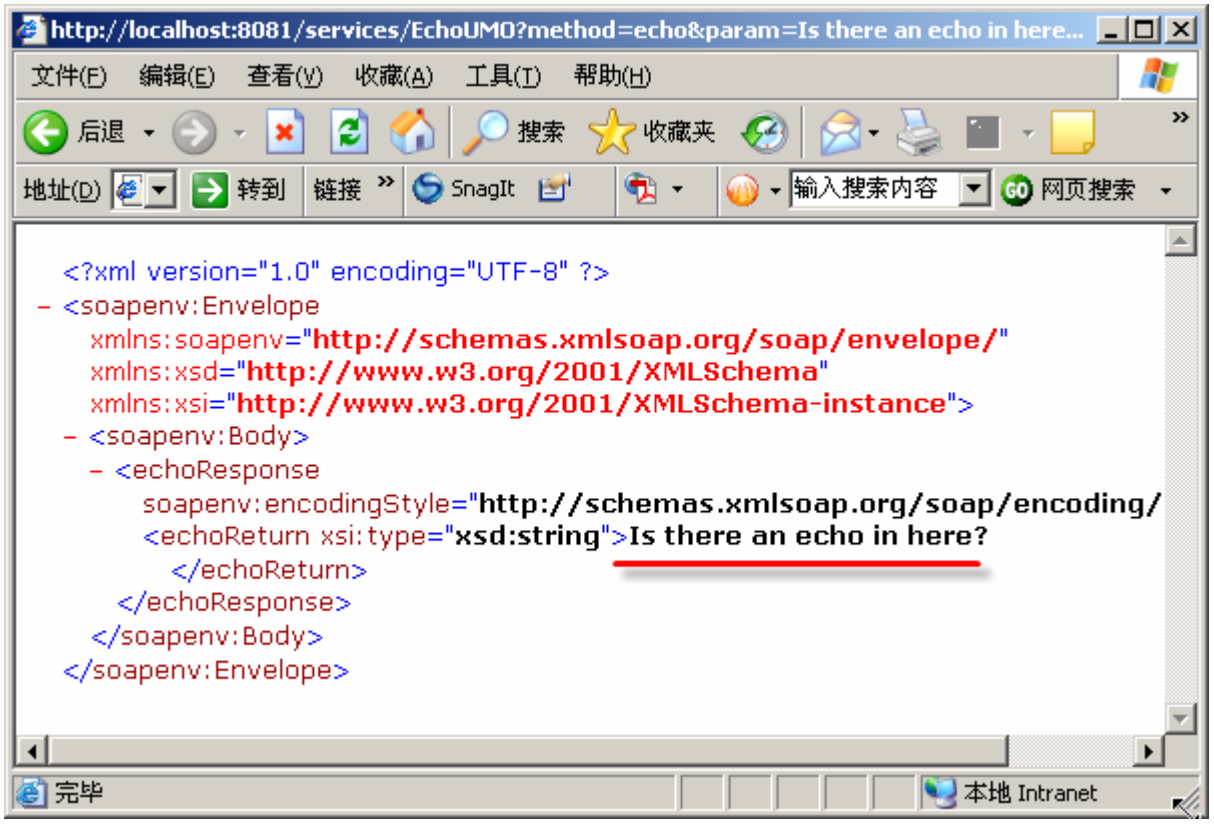
4.5 运行

运行和通过控制台调用的情形我们前面已经看到，这里只是说明一下通过 SOAP 的调用。
在浏览器地址栏输入：

```
http://localhost:8081/services/EchoUMO?method=echo&param=Is%20there%20an%20echo%20in%20here?
```

其中：

http://localhost:8081/services 是调用服务的地址， EchoUMO 是服务名称，mothod 是服务中需要调用的方法，而 param 则是需要传递的参数。
浏览器中返回的是 SOAP 封装的响应消息，我们看到在地址栏中输入的参数，又被反馈回来了！



5 结束语

完。