Mule ESB Studio v3.3 安装使用手册

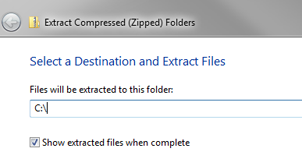
# \*\*\*初级教程\*\*\*

如果你还没有做好准备，请到<http://www.mulesoft.org>下载免费的社区版Mule ESB，按照网站上的说明启动**Mule Studio**，并且选择一个工作区（另外，你还可以下载30天免费试用的企业版Mule ESB）

# 安装Mule Studio

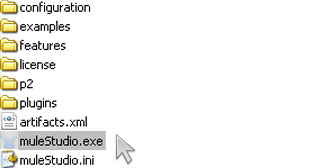
安装前，请确认你的机器上已经安装了1.6版本的JDK。最后请确认你的JDK环境变量配置是否正确

2.1 导出



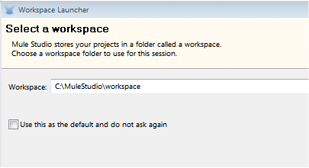
将下载的文件解压到你的硬盘分区的**根目录**下，例如：C:\

1. 执行

****

找到**C:\MuleStudio**目录，运行**muleStudio.exe**启动**Studio**

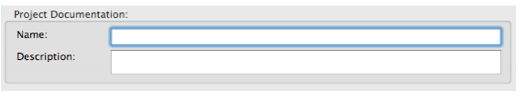
2. 选择工作区

****

点击**OK**使用默认的工作区

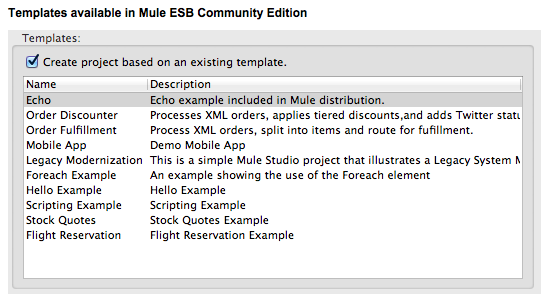
# 使用Studio模板

1. 点击**File** 菜单，选择**New > Mule Project**

2. 出现**New Mule Project**面板后，为你的项目输入名称和一个简短的说明，如图：

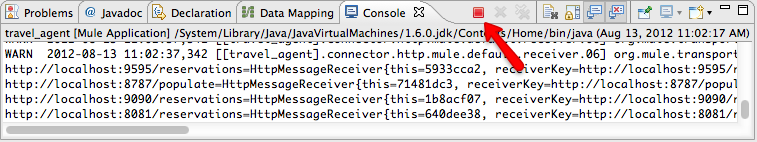
3. 在**Server Runtime**选项上选择你将要使用的Mule运行时版本，如图：server_settings.png

4. 点击旁边的复选框，根据现有的模板创建项目，单击项目，选择你想要使用的模板创建项目，如图：



5. 点击**Finish**按钮，Mule Studio会创建并打开一个新的项目，完成预创建和预配置的流程

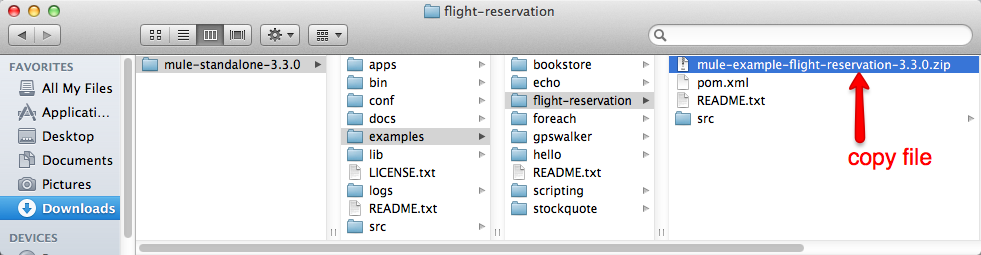
6. 在Mule Studio的**Package Explorer**栏中，右键点击**mule-config.mflow**文件，选择**Run As > Mule Application**

7. 停止运行该项目，请在Mule Studio控制台点击红色的**Terminate**按钮，如图：

# 运行独立的例子

1. 到**Mule ESB Standalone**目录下，找到**Examples**目录下你想运行的例子

2. 拷贝.zip文件的例子到**$MULE\_HOME/apps**目录下，例如：运行**Flight Reservation example**的例子，拷贝**mule-example-flight-reservation-3.3.0.zip**到**$MULE\_HOME/apps**目录下，如图：



3. 启动Mule，运行这个例子

# 启动Mule Studio

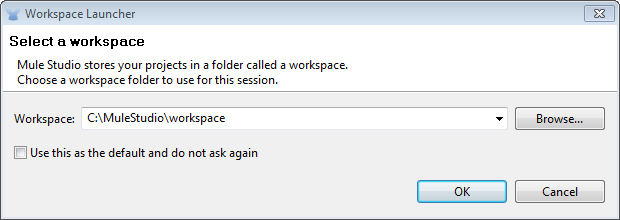
如果你在安装过程中启动了Mule Studio，并且已经在运行了，请跳过本节的其余部分，直接进行：[创建新项目](#_创建新项目)

如果当前Mule Studio没有启动，通过完成下面的步骤启动应用程序

1. 找到Mule Studio安装目录

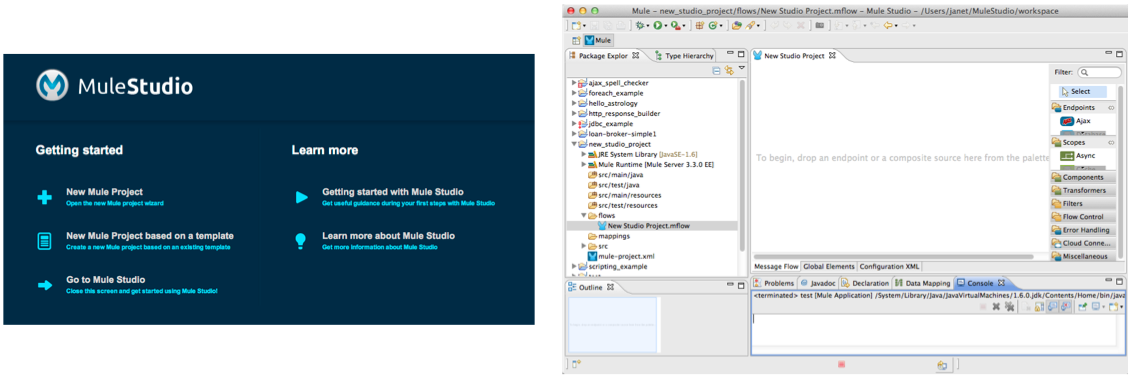
2. 执行**muleStudio.exe**

3. 点击**OK**使用默认的工作区



# 创建新项目

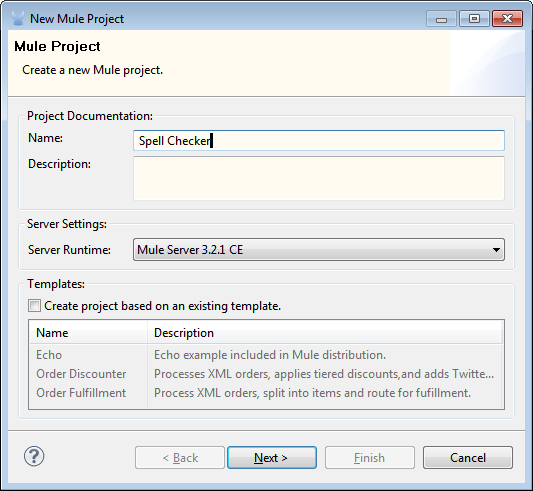
1. 如果你看到是各种控制组件的应用程序窗口（右下图），请直接进入第2节。相反（左下图），这是你第一次运行Studio。单击**Go to Mule Studio**，然后进入第2节



2. 点击**File > New > Mule Project**

3. 在弹出的**New Mule Project**窗口中输入你的项目名称

4. 点击描述文本框中的任意位置，然后键入与项目有关的描述，需要注意的是**Server Settings**显示的是 **Mule Server3.2.1** 社区版

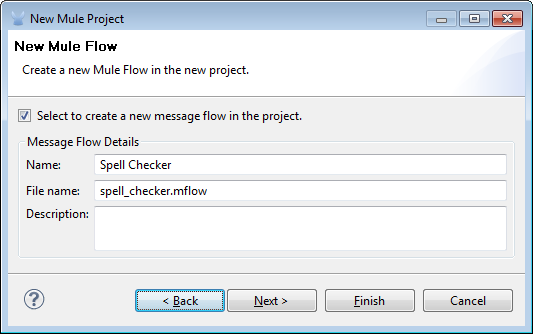


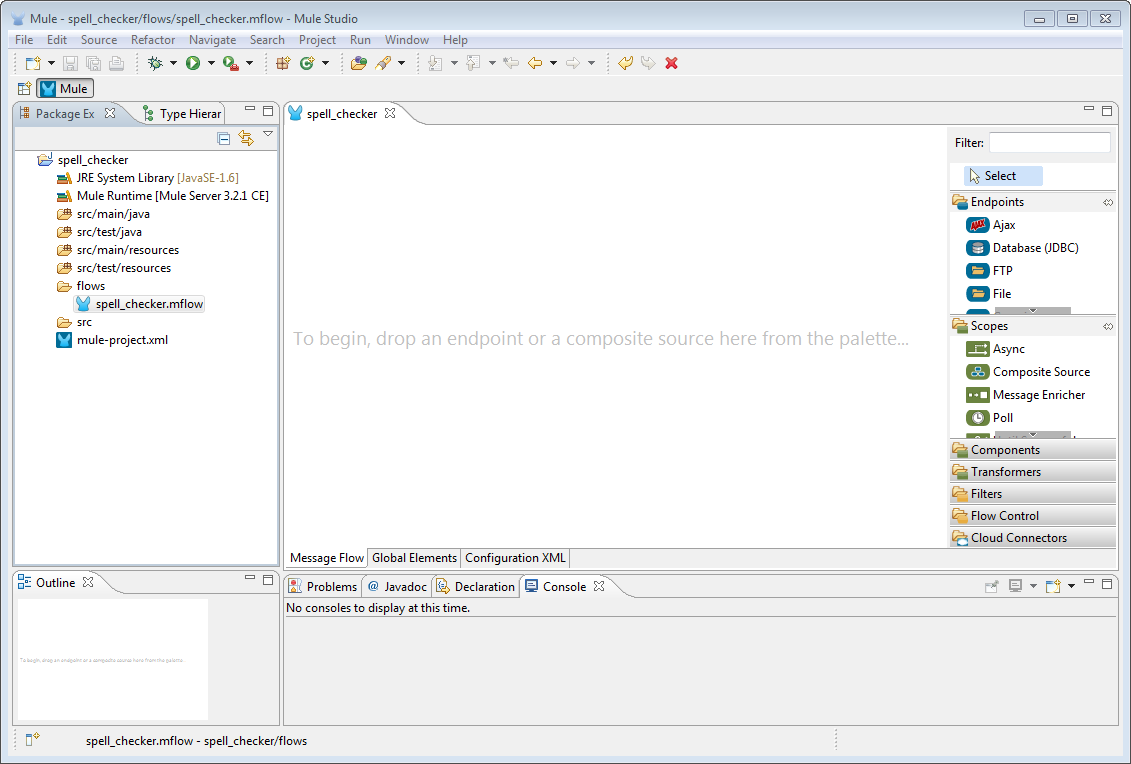
5. 点击**Next**

所有项目都包含至少一个消息流程，由于是一个简单的项目，所以只有一个消息流程。我们将把应用程序的名称用作流程的名称，所以请在**New Mule Flow**面板中输入名称：“Spell Checker”，然后点击描述文本框中的任意位置。

注意，Studio会自动将大写字母转换为小写，使用“\_”字符替换单词之间的空格，并在结尾追加“. mflow”。结果显示在**File name**字段中。当你建立Spell Checker应用程序的拖放和拖放界面时，. mflow文件会一直跟踪你的工作。

在**Description**中输入任意描述，点击**Finish**完成创建

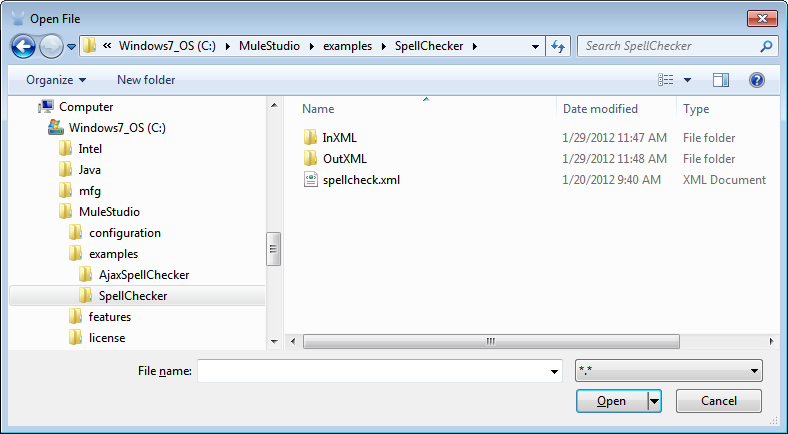




# 文件夹设置

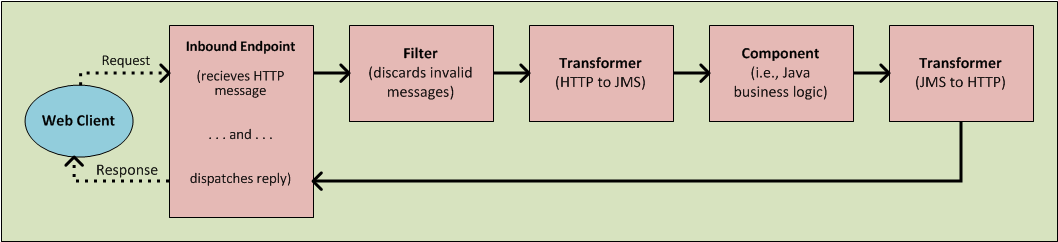
**Spell Checker**应用程序需要一对特殊的文件夹，方便数据的输入和输出。可以通过完成以下步骤来创建它们：

1. 找到**Studio菜单> File > Open File >**
2. 然后在**Open File**的窗口中找到你的Studio安装目录，例如：C:\MuleStudio > Examples > Spell Checker
3. 在该目录下创建两个分别以**InXML**和**OutXML**命名的文件夹



# 关于Mule流程

在我们的例子中（如下图），Web客户端发送HTTP消息的**Inbound Endpoint**设置为“request-response”的交换模式。换句话说，用户期望应用程序能够响应他们的请求。



过滤器判断收到的消息是否有效，然后决定丢弃或者传递给流程中的下一个模块（将HTTP请求按照JMS规范进行转换）

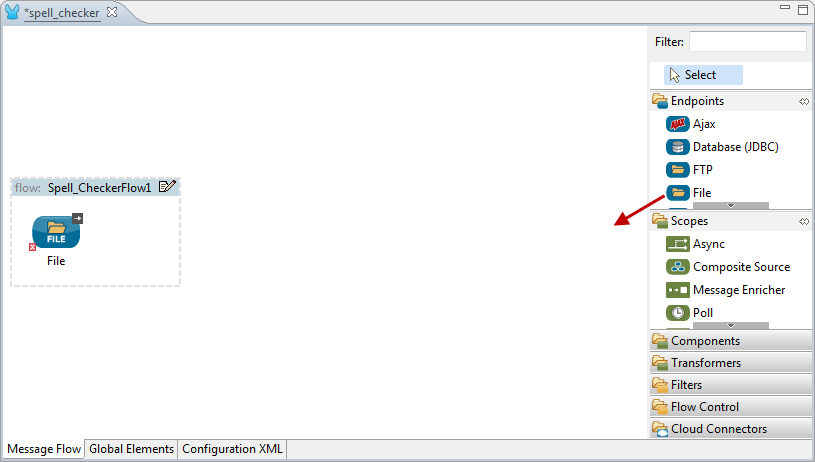
接着，用Java编写的一个业务逻辑组件的消息进行处理，然后生成一个回答，传递到另一个转换器，将回答从JMS转回HTTP响应，然后把响应返回给正在等待的Web客户端。

# 模块部署

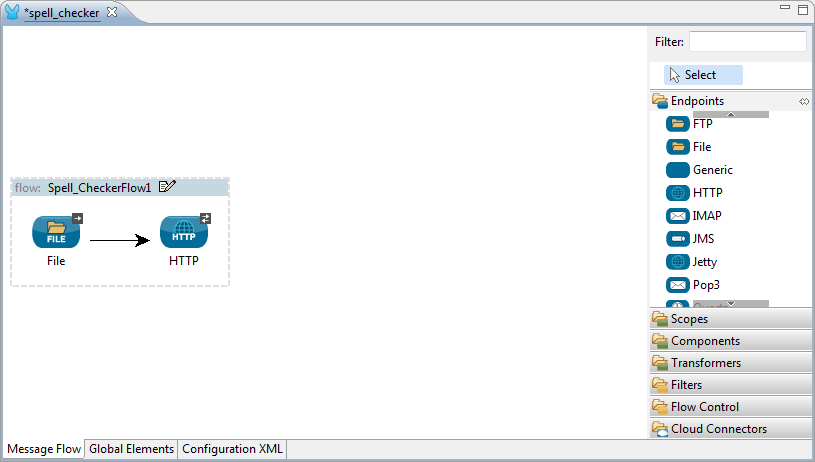
你可以从右侧组件栏中拖拽组件到画板上，从而创建一个Mule 流程，组件栏中一共包含七个类别的组件：**Endpoints**、**Scopes**、**Components**、**Transformers**、**Filters**、**Flow Control**、和**Cloud Connectors**

# 创建Spell Checker流程

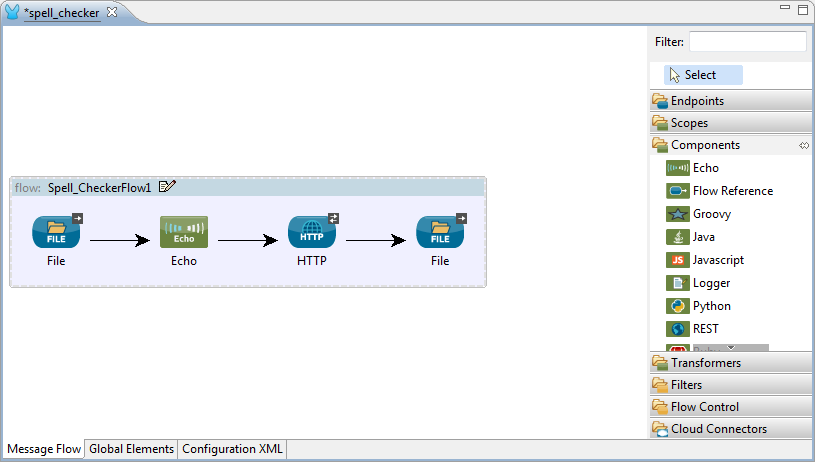
1. 在**Endpoints**类组件栏中找到**File**，单击并拖放到左侧画板中相应的位置



1. 单击并拖动**HTTP**，也放在**Endpoints**组中，但要确保你把它放到**File**图标的右侧，因此，它会出现在用虚线表示的边框内



1. 松开鼠标按钮后，会有一条箭头线从**File**组件指向**HTTP**组件
2. 添加另一个**File**组件，将**Endpoints**类组件栏中的**File**组件拖放到**HTTP**组件的右侧，必须确保它也出现在用虚线表示的边框内
3. 最后，拖放**Ehco**组件，将它放到第一个**File**组件和**HTTP**组件中间，你的画板展示了一个完整的流程，如下图：

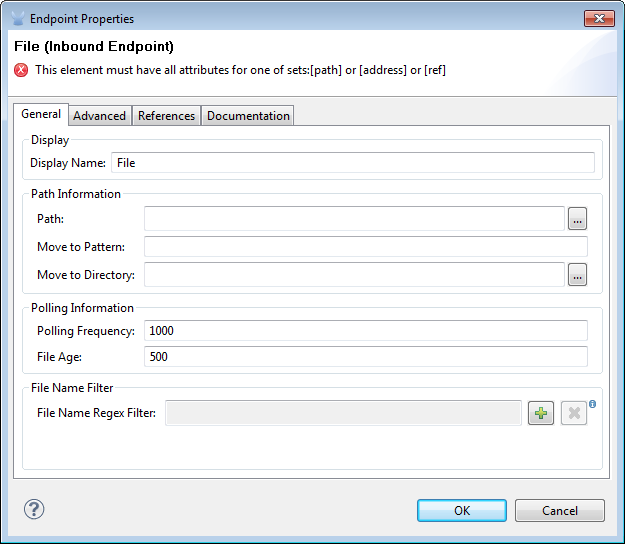


# 配置流程组件

几乎所有的Mule组件提供的配置选项，你都可以通过对话框设置，无需编写XML

# File入站端

1. 双击流程中最左侧的**File**组件，会弹出一个**Endpoint Properties**对话框，默认显示**General**选项卡，需要注意的是，Mule自动识别这个文件组件为入站端点（Inbound Endpoint）。
2. 在**Display Name**输入框中，你必须输入一个具有很强描述性的名字，比如：“Incomming data File”
3. 在**Path Information**面板中，在**Path**字段输入InXML所在项目的路径，整体路径看起来像这样：C:\MuleStudio\Examples\SpellChecker\InXML
4. 点击**OK**，完成**Incoming Endpoint**的配置并返回画板



# Echo组件

应用启动时，在**Studio**控制台输出消息的组件，无需特殊配置

1. 你也可以在**General**选项卡的**Display Name**字段输入名字，在**Documentation**选项卡的**descriptive comments**字段输入一些描述信息，当你的鼠标移到该组件上面，会展示出这些信息
2. 点击**OK**，完成**Echo**组件的配置并返回画板

# HTTP Endpoint

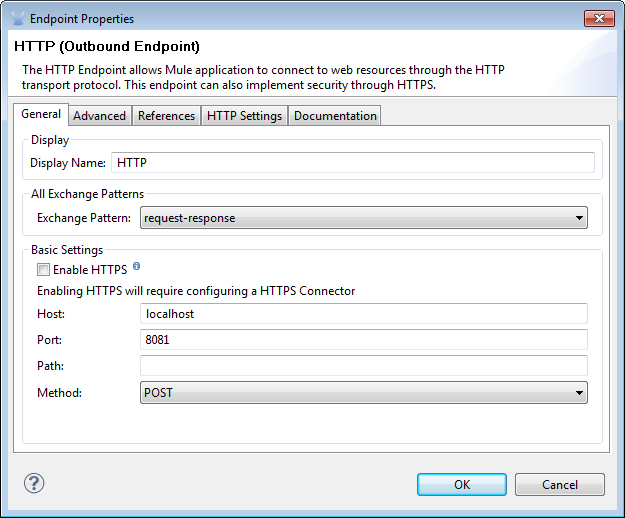
这个出站端从后台检索信息（这例子中的后台是**Google Spell Checker API**）

1. 双击**HTTP**组件，在**General**选项卡中完成配置

在**Host**字段输入[www.google.com/tbproxy/spell?lang=en](http://www.google.com/tbproxy/spell?lang=en)

在**Port**字段输入Google Spell Checker Server使用的端口：80

在**Method**字段输入Google Spell Checker Server使用的方法类型：POST



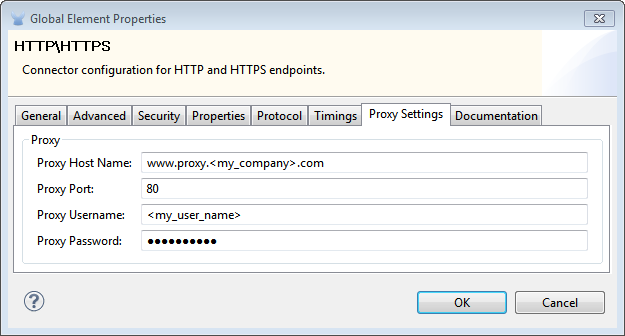
1. 点击**OK**完成配置

如果你打算在公司防火墙后面运行本教程中的应用程序，需要另外配置：

双击**HTTP Endpoint**，点击**References**选项卡

点击**Connector Reference**选项卡右边的"+"图标

点击**Proxy Settings**，进行如下图配置：



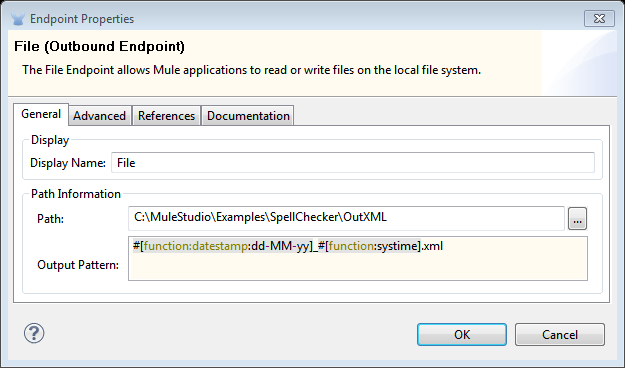
# File 出站端

1. 双击配置流程右边的**File**组件
2. 在**Path Information**的**Path**字段输入OutXML所在项目的路径，整体路径看起来像这样：C:\MuleStudio\Examples\SpellChecker\OutXML
3. 在**Output pattern**栏中配置以下输出格式：

#[function:datestamp:dd-MM-yy]\_#[function:systime].xml

这会让Mule在OutXML目录下以时间戳的格式给生成数据的文件命名

1. 点击**OK**完成配置

****

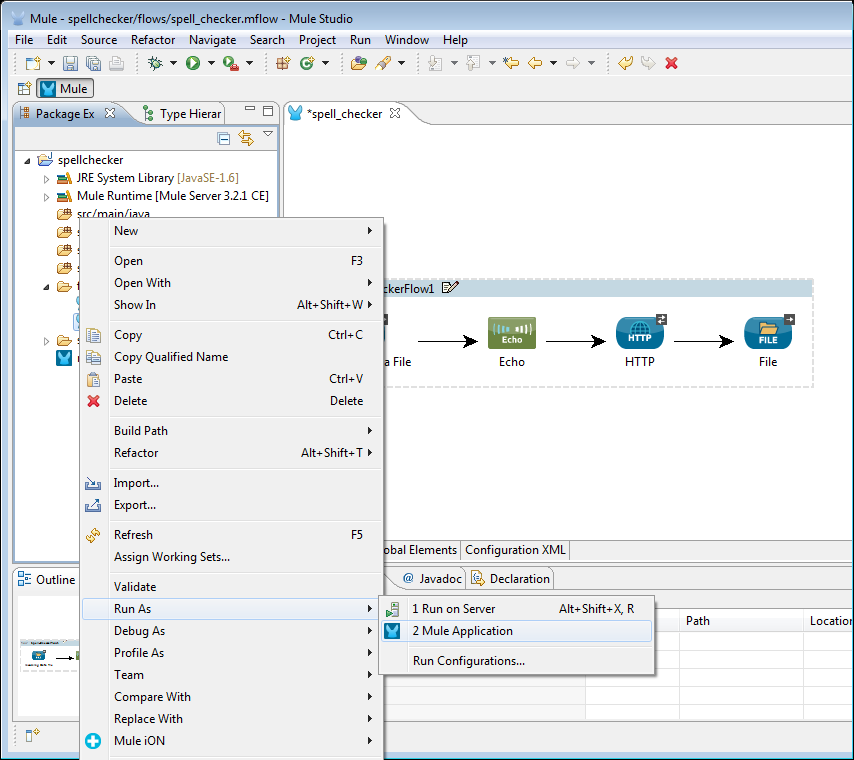
祝贺你！你已经完成了你的第一个Mule ESB应用

可选：点击控制台下方的**Configuration XML**选项卡切换到XML视图，你可以看到XML格式的流程信息，你可以直接修改这些XML内容，只要切回流程视图，你的修改会很快体现在视图中，在XML视图中你应该看到如下内容：

****

# 运行Spell Checker

在**Package Explorer**栏中，鼠标右键**spell\_checker.mflow**，在弹出的菜单中点击**Run As > Mule Application**

****

如果你没有保存该应用，在弹出的**Save and launch**对话框中点击**OK**，Mule会保存并自动运行该应用

随后控制台会打印出Started app 'spell\_checker'等信息

# 测试Spell Checker

你的项目正在运行，**Mule**正在检查**InXML**目录下的**XML**文件，如果要启动一个流程，你需要拷贝一个**XML**文件到这个目录下

1. 到Spell Checker 目录下（例如：C:\MuleStudio\Examples\SpellChecker）
2. 找到**spellcheck.xml**文件，这个文件中包含一些拼写错误的内容："Creeating mi fiirst Mulle Studio aplication!"
3. 将这个文件拷贝到InXML目录下，请注意，你只需要做这一步

Studio控制台会输出一些信息，最后的内容是："Writing file to..."

随后**spellcheck.xml**文件不再保留在InXML目录下

一个新的文件在OutXML目录下产生

这个新产生的文件名字包含我们在**File出站端**指定的时间戳格式，如：

21-01-12\_1327187012641.xml

新产生的文件中包含拼写错误的单词的更正

Spell Checker应用会一直运行，等待处理新的文件

下面是你拷贝到InXML目录下的**spellcheck.xml**文件内容：

<?xml version="1.0" encoding="utf-8" ?>

<spellrequest textalreadyclipped="0" ignoredups="0" ignoredigits="1" ignoreallcaps="1">

<text>Creeating mi fiirst Mulle Studio aplication!</text>

</spellrequest>

下面是Google API返回的修改建议，这部分内容保存在以时间戳格式命名的文件中，放在OutXML目录下

<?xml version="1.0" encoding="UTF-8"?>

<spellresult error="0" clipped="0" charschecked="44">

<c o="0" l="9" s="0">Creating Crating Greeting Cremating Curating</c>

<c o="13" l="6" s="0">first foist fist fiesta fest</c>

<c o="20" l="5" s="0">Muller Mlle Mule Mull Miller</c>

<c o="33" l="10" s="0">application applications allocation placation implication</c>

</spellresult>

# \*\*\*中级教程\*\*\*

初级教程演示了如何创建简单的拼写检查的应用程序。中级教程将会演示如何使用Studio拖放接口去完成更复杂的**Ajax Spell Checker**，它包含了以下几项内容：

1. Google Spell Checker的web服务
2. 一个HTML页面
3. 预先排好序的Mule组件

本教程覆盖了以下任务：

1. 创建一个有Ajax出站端和Ajax连接器的流程
2. 将网页上的数据转换后传递到Mule 消息处理器的流程
3. 使用XSL和Mule转换器预处理用户输入的内容，变成Google API能够理解的XML格式
4. 创建一个Google Spell Checker API的连接，以便它可以处理网页上输入的内容
5. 通过Ajax返回XML数据给原网页
6. 在流程里实现一个Jetty HTTP server

如果你已经完成了初级教程，最好关闭之前教程中所做的项目，否则可能会在接下来的教程中引起混乱，你需要做好以下步骤：

1. 鼠标右键spell\_checker或者你在基本教程中使用的其他命名的项目
2. 选择**Close Project**选项

这个Ajax Spell Checker流程包含五个组件和一个连接器

本节覆盖了前面基础教程中创建项目的步骤：

1. 点击**File > New > Mule Project**
2. 在弹出的**New Mule Project**窗口中输入Ajax Spell Checker作为项目名称，点击**Next**
3. 当**New Mule Flow**面板出现之后，在**Name**字段输入ajax flow或者其他比较有意义的名字，名字中的空格会被替换成下划线，比如：ajax flow会最终以**ajax\_flow.mflow**呈现，

点击**Finish**

# 向流程中添加组件

如果你已经创建了Ajax Spell Checker，请双击ajax\_flow.mflow文件，它在**Package Explorer**栏的flows目录下

接下来在**Global Elements**视图中创建**Ajax connector**组件，在**Message Flow**视图中创建**Ajax endpoint**组件，本教程后面会演示这两个组件如何一起实现Jetty HTTP 服务器的Ajax传输通道

将以下组件按照编号的顺序从左至右拖放到画板中，如下图所示，这些组件分别在右侧组件栏的**Endpoint**、**Transformer**、**Components**类别中：

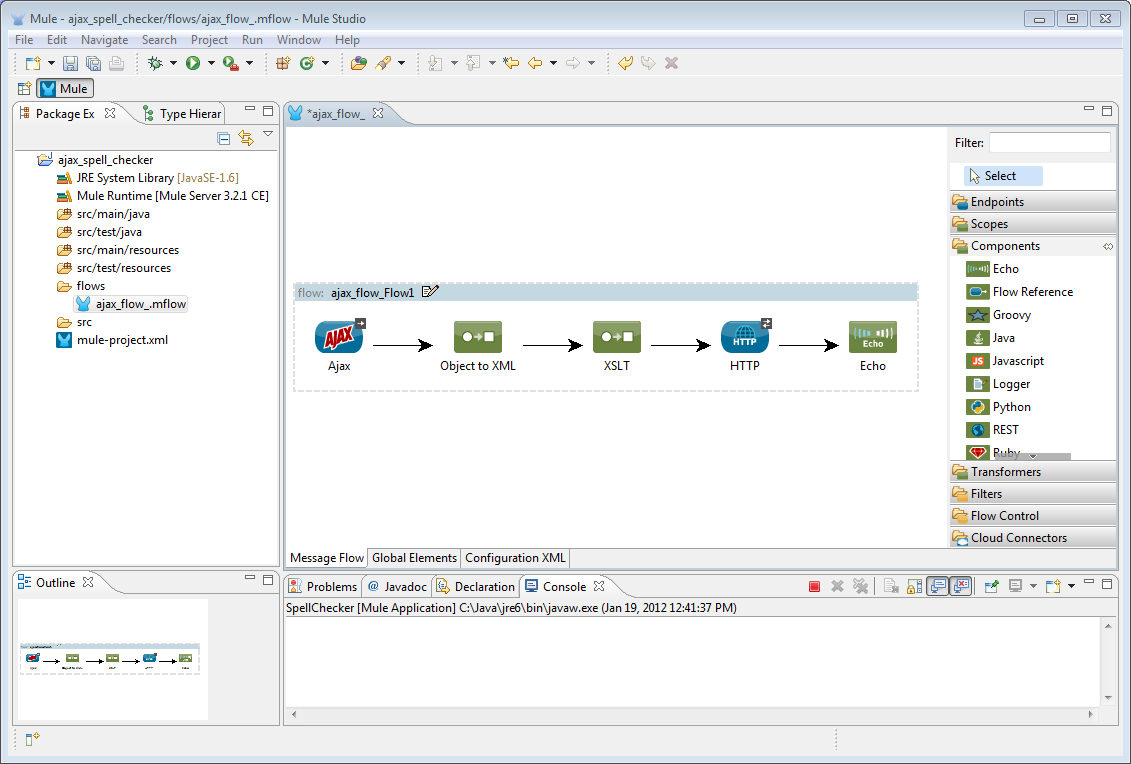
1. **Ajax Endpoint**

2. **Object-to-XML Transformer**

3. **XSLT Transformer**

4. **HTTP Endpoint**

5. **Echo Component**

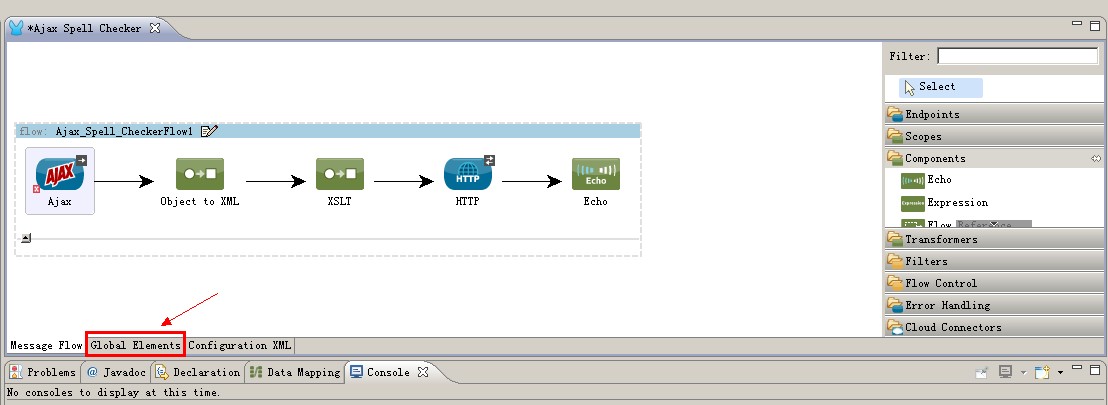


如果你创建的组件不小心放错了顺序，只需要删除它，然后从组件栏中拖放另一个放到你的画板中

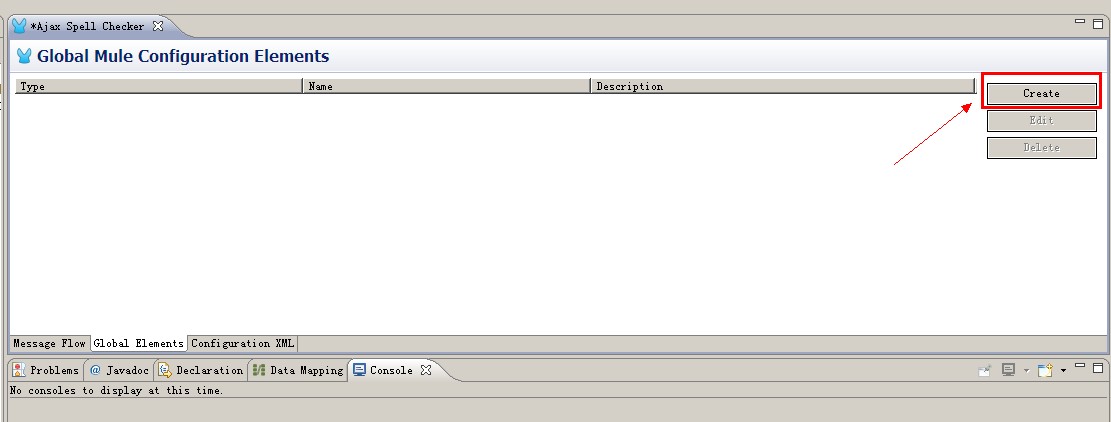
# 添加和配置连接器

接下来，添加并且配置Ajax连接器，它是一个全局元素（即模板），它向**Ajax Endpoint**提供服务器连接信息

1. 点击画板底部的**Global Elements**选项卡，打开Global Elements视图



1. 点击**Create**按钮打开**Choose Global Type**面板



1. 展开**Connectors**节点
2. 选择Ajax，点击**OK**

a. 在 **Name**字段输入AjaxConnector

b. 在**Server URL**字段输入http://127.0.0.1:8090/Ajax

c. 在**Resource Base**字段输入src/main/app/docroot

5. 点击**OK**，关闭窗口

现在**Global Elements**视图列出了一个Ajax连接器

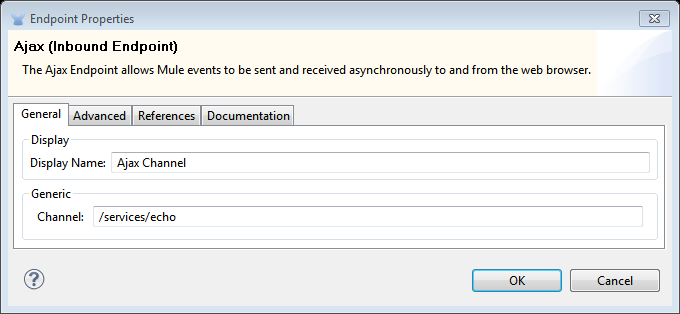
# 配置模块

现在，Ajax连接器已经创建好了，接下来需要将这个连接器引用到**Ajax Endpoint**

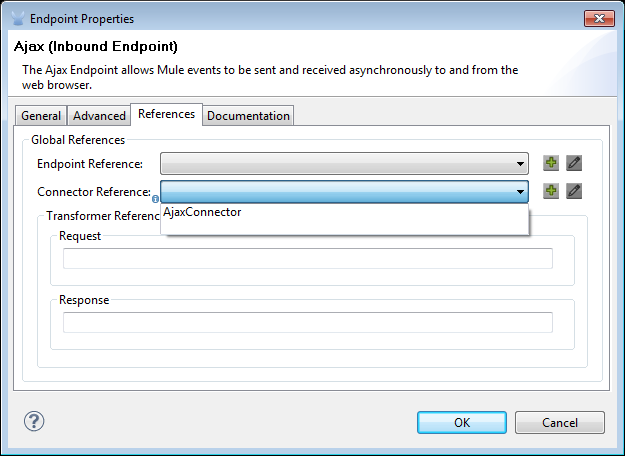
# 配置Ajax Endpoint

**Ajax Endpoint**是流程的**Message Source**（即入站端），按照下面的步骤完成配置：

1. 点击**Message Flow**选项卡
2. 双击**Ajax**组件，打开属性配置面板
3. 在**General**选项卡的**Display Name**字段输入Ajax Channel
4. 在**Channel**字段输入/services/echo，这个路径用于和web页面通讯



1. 点击**References**选项卡，点击**Connector Reference**下拉菜单，选择**AjaxConnector**，这个就是刚刚在**Global Elements**视图中创建的Ajax连接器（windows系统的面板可能和下面的截图有些差异）



点击**Connector Reference**右边的“增加”或“修改”按钮，可以在**References**选项卡中编辑这个Ajax连接器，无论你在哪里创建它，都可以在**Global Elements**视图中看到它

1. 点击**Documentation**选项卡，在**Description**字段输入一个富有意义的描述，点击**OK**关闭**Endpoint Properties**面板

# 配置Object-to-XML转换器

这个组件将消息转换成XML代码，不需要特别的配置，但是**Display** **Name**和**Description**字段会在排除故障时提供有用的信息

1. 双击流程画板中的**Object-to-XML**组件打开属性面板
2. 在**General**选项卡中的**Display Name**字段输入Convert JS Object to XML
3. 在**Documentation**选项卡中的**Description**字段输入Convert JS Object to XML，然后点击**OK**关闭属性面板

# 配置XSLT转换器

**transform.xsl**文件为你的应用程序在消息中提取数据并且翻译成能够识别的内容提供指示，**Mule Studio**中包含很多你在完成教程时需要的文件，按照这些步骤在你的机器上添加多个文件，然后导入到你的**Mule Studio**应用程序中

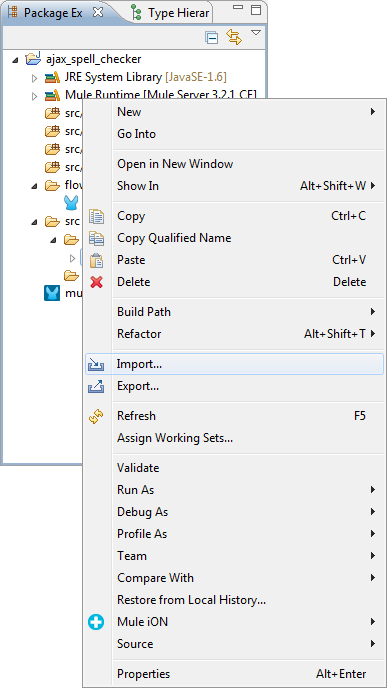
# 下载和导入XSLT转换文件



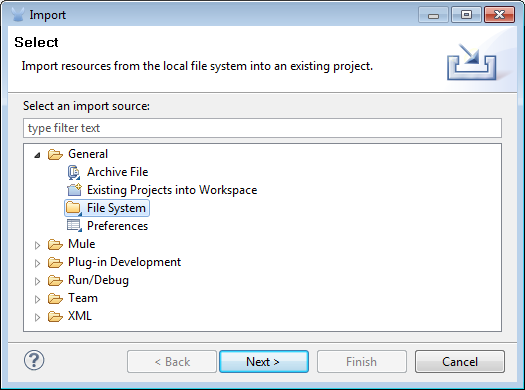
1. 你可以双击打开上面的文件，或者按住ctrl键，点击下面的链接来下载这个文件

[transform.xsl file](http://www.mulesoft.org/documentation/download/attachments/57049117/transform.xsl?version=2&modificationDate=1341008752741)

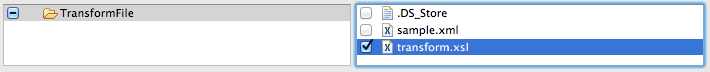
1. 将这个文件保存到：**MuleStudio目录> examples > AjaxSpellChecker > TransformFile**目录下，以**transform.xsl**命名
2. 在Studio的左侧**Package Explorer**栏中，找到应用的**src>main>app**目录，鼠标右键**app**目录，选择**Import**



1. 在导入窗口中，选择**General>File System**，点击**Next**



1. 在弹出的面板中，选择**Browse…**按钮，找到刚刚保存的**transform.xsl**文件，点击**Finish**



1. 检查导入的文件是否出现在了**src>main>app**目录下

# 配置XSLT消息处理器

这个组件按照**transform.xsl**中配置的说明来解析实际的消息，以便后面的流程能够继续执行，按照下面的步骤来完成XSLT消息处理器的配置：

1. 双击XSLT转换器，打开**Pattern Properties**面板
2. 在**General**选项卡的**Display Name**字段输入“Convert XML Structure”
3. 在**XSL File**字段输入“src/main/app/transform.xsl”
4. 点击**Documentation**选项卡，在**Description**字段输入“Produces compliant XML for the Google API”
5. 点击**OK**关闭**Pattern Properties**面板

# 配置HTTP Endpoint

HTTP Endpoint负责向Google Web Service发送XML请求和接收XML响应

1. 双击**HTTP Endpoint**组件打开**Endpoint Properties**面板
2. 在**General**选项卡的**Display Name**字段输入“Google API”
3. 在**Host**字段输入[www.google.com/tbproxy/spell?lang=en](http://www.google.com/tbproxy/spell?lang=en)
4. 在**Port**字段输入80
5. 在**Method**字段输入POST
6. 点击**HTTP Settings**选项卡，在**Content type**下拉列表中选择text/xml
7. 点击**Documentation**选项卡，在**Description**字段输入“Sends XML requests to the Google Web service and receives XML responses”
8. 点击**OK**保存并返回**Endpoint Properties**面板

# 配置Echo组件

在**Studio**控制台输出请求和响应消息的组件，无需特殊的配置，但你也可以配置**Display Name**和**Description**

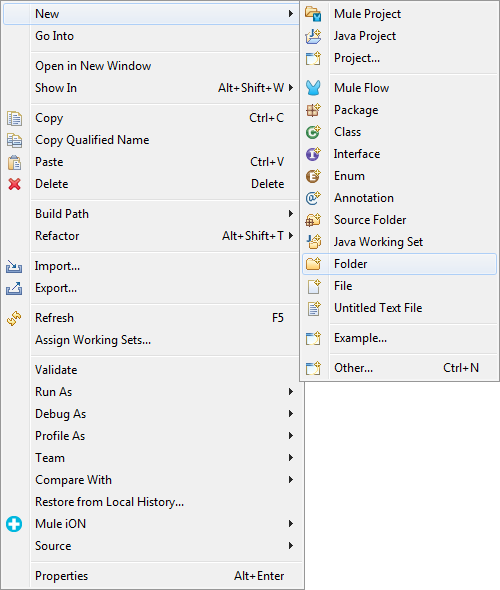
1. 双击**Echo**组件打开属性面板
2. 在**General**选项卡的**Display Name**字段输入“Echo to Console”
3. 在**Documentation**选项卡的**Description**字段输入“Terminates the flow and logs results to the console”
4. 点击**OK**

# 导入HTML页面

这个应用需要一个包含JavaScript代码的HTML页面，用来调用Mule的类，从而完成与流程的通信

# 为HTML文件创建文档根目录

1. 在**Package Explorer**栏中找到应用的**src/main/app**目录
2. 鼠标右键点击**app**目录，选择**New > Folder**

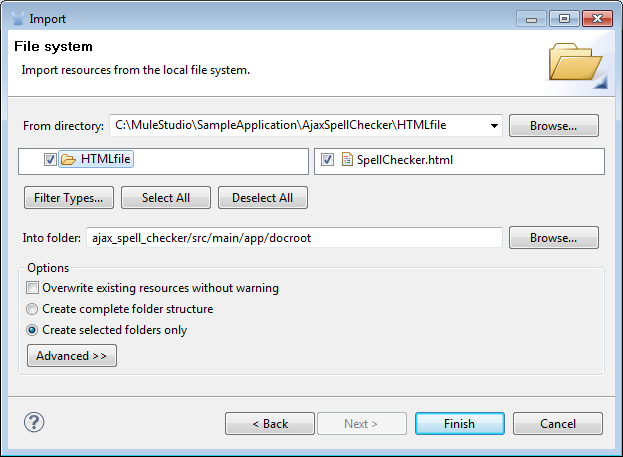


新建的目录以**docroot**命名

1. 鼠标右键点击**docroot**目录，选择**Import > File System**，点击**Next**
2. 点击**From directory** 字段右边的**Browse**按钮，

找到<Mule Studio目录>\Examples\AjaxSpellChecker\HTMLfile，点击**OK**

1. 在**Import**面板的右边，勾选**SpellChecker.html**左边的复选框，点击**Finish**



1. 检查**SpellChecker.html**文件是否成功导入到项目的**src/main/app/docroot**目录下
2. 双击**src/main/app/docroot**目录下的**SpellChecker.html**文件，可以看到文件内容

祝贺你！你已经使用Mule Studio创建了一个高级的Mule ESB应用

# 运行项目

在Mule Studio中运行该项目，观察它究竟做了些什么

1. 在**Package Explorer**栏中，鼠标右键**Ajax Spell Checker.mflow**，在弹出的菜单中点击**Run As > Mule Application**，控制台会输出以下信息：

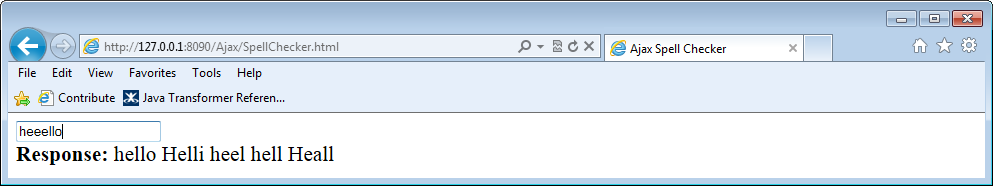
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

+ Started app 'ajax\_spell\_checker' +

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1. Mule Studio在创建项目时，做了如下设置：
2. 你在本地计算机上创建了一个Jetty HTTP服务器，**Ajax connector**的**Server URL** 被设置为<http://127.0.0.1:8090/Ajax>，你可以通过这个URL访问这个服务
3. 一个**SpellChecker.html**文件放到了**docroot**目录下，**src/main/app/docroot**将被指定为Jetty HTTP服务器的基础资源
4. 通过<http://127.0.0.1:8090/Ajax/SpellChecker.html>可以访问你的应用

通过浏览器访问<http://127.0.0.1:8090/Ajax/SpellChecker.html>，页面会出现一个空的文本框，输入heeello，服务的响应会很快出现在下方：



请注意：当你每次输入一个字符的时候，Google的web服务会返回一系列的拼写建议，一般情况下，第一个建议是正确的拼写

**Echo**组件可以让你查看到控制台输出的信息，大致内容如下：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Message received in service: ajax\_flow\_Flow1. Content is: '<?xml \*

\* version="1.0" encoding="UTF-8"?><spellresult error="0" clipped="0" \*

\* charschecked="7" suggestedl...[100 of 175]' \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*高级教程\*\*\*

初级教程演示了如何创建简单的拼写检查的应用程序。中级教程演示了如何使用Studio拖放接口去完成更复杂的**Ajax Spell Checker**

回顾一下中级教程，这里可以使用数据映射器（**DataMapper**）替代中级教程中使用的转换器（**Transformers**），虽然许多步骤重复、相似，但可以借此机会反复实践，更加熟练。往往只是一些比较细微的差异，可以帮助你更快速的掌握**Mule Studio**的界面操作

**DataMapper**是Mule ESB 企业版中特有的功能，作为回报，MuleSoft欢迎社区版用户开发带有**DataMapper**功能的应用程序，通过Studio图形界面可以预览数据映射关系。然而，实际部署、运行这样的应用程序，需要向MuleSoft申请一个有效的企业版license

**由于最近工作比较忙，高级教程暂时没有整理进来。水平有限，有翻译不到位的地方请与我联系：**[**sk.he@hotmail.com**](mailto:sk.he@hotmail.com)**，谢谢！**

参考资料：

<http://www.mulesoft.org/documentation/display/MULE3START/Basic+Studio+Tutorial>

<http://www.mulesoft.org/documentation/display/MULE3START/Intermediate+Studio+Tutorial>

<http://www.mulesoft.org/documentation/display/MULE3START/Advanced+DataMapper+Tutorial>