# [Ext.Direct之DirectJNgine使用（一）](http://blog.csdn.net/plfl520/article/details/7361480)

链接地址：<http://blog.csdn.net/plfl520/article/details/7361480>

1. [ExtDirect之DirectJNgine使用一](http://blog.csdn.net/plfl520/article/details/7361480" \l "t0)
2. [一什么是DirectJNgine](http://blog.csdn.net/plfl520/article/details/7361480#t1)
3. [二下载DirectJNgine](http://blog.csdn.net/plfl520/article/details/7361480#t2)
4. [三项目中使用DirectJNgine依赖文件](http://blog.csdn.net/plfl520/article/details/7361480#t3)
5. [四DirectJNing实例](http://blog.csdn.net/plfl520/article/details/7361480#t4)
   1. [1实例说明](http://blog.csdn.net/plfl520/article/details/7361480#t5)
   2. [2 在webxml中配置DirectJNgineServlet](http://blog.csdn.net/plfl520/article/details/7361480#t6)
   3. [3 设计服务器端Java方法](http://blog.csdn.net/plfl520/article/details/7361480#t7)
   4. [3 让JavaScript能够使用服务器端方法](http://blog.csdn.net/plfl520/article/details/7361480#t8)
   5. [4 ExtDirect调用服务器端方法](http://blog.csdn.net/plfl520/article/details/7361480#t9)
      1. [41 引入自动生成的js文件](http://blog.csdn.net/plfl520/article/details/7361480#t10)
      2. [42通过ExtDrect注册服务器端方法](http://blog.csdn.net/plfl520/article/details/7361480#t11)
      3. [43调用服务器端方法](http://blog.csdn.net/plfl520/article/details/7361480#t12)
   6. [7 ExtDirect调用过程分析](http://blog.csdn.net/plfl520/article/details/7361480#t13)
6. [五总结](http://blog.csdn.net/plfl520/article/details/7361480#t14)

**[Ext.Direct之DirectJNgine使用（一）](http://blog.csdn.net/puppet_love/article/details/6424624)**

分类： [ExtJS](http://blog.csdn.net/puppet_love/article/category/822431) [web](http://blog.csdn.net/puppet_love/article/category/541285) [JavaScript](http://blog.csdn.net/puppet_love/article/category/822435) 2011-05-16 15:59 412人阅读 [评论](http://blog.csdn.net/puppet_love/article/details/6424624#comments)(2) 收藏 [举报](http://blog.csdn.net/puppet_love/article/details/6424624#report)

目录

一、         什么是DirectJNgine. 1

二、         下载DirectJNgine. 1

三、         项目中使用DirectJNgine依赖文件... 2

四、DirectJNing实例... 2

4、1实例说明... 2

4、2 在web.xml中配置DirectJNgineServlet. 3

4、3 设计服务器端Java方法... 3

4、3 让JavaScript能够使用服务器端方法... 4

4、4Ext.Direct调用服务器端方法... 6

4.4.1 引入自动生成的js文件... 6

4.4.2通过Ext.Drect注册服务器端方法... 7

4.4.3调用服务器端方法... 7

4、7Ext.Direct调用过程分析... 8

五、总结... 9

**一、什么是DirectJNgine**

DirectJNgine是[**Java**](http://lib.csdn.net/base/javase)版Ext Direct API的实现，它允许应用程序使用ExtJS调用Java方法，从而使[**JavaScript**](http://lib.csdn.net/base/javascript)调用Java方法变得透明，摆脱过去繁琐费时的调用方法。

**二、下载DirectJNgine**

DirectJNgine下载地址：<http://code.google.com/p/directjngine/>，目前最新版本为1.3，下载后解压即可，里面包含有DirectJNgine类库（lib）、帮助文档(doc)、实例(demo)等。要使用DirectJNgine当然少不了Extjs类库支持，Ext类库在http://www.sencha.com很容易找到，不再啰嗦。下载完毕将ExtJS类文件拷贝至项目的WebContent目录下。

**三、项目中使用DirectJNgine依赖文件**

项目中使用DirectJNgine需要将以下jar文件添加至web应用程序的WEB-INF/lib目录下：

（1）DirectJNgine.jar：位于deliverables/directingine.xxx.jar；

（2）DirectJNgine使用第三方类库：①位于lib/\*.jar下的所用文件，子目录除外；②位于lib/runtimeonly/\*.jar

如果需要启用客户端参数调试模式，需要在web app项目中加入下面两个Js文件：

（3）djn-remote-call-support.js：位于deliverables/ djn-remote-call-support.js；

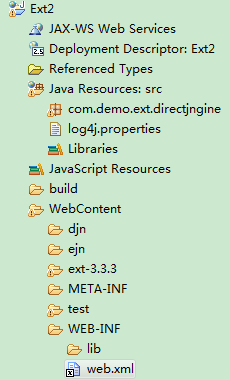
（4）ejn-assert.js：位于deliverables/ejn-assert.js。

**四、DirectJNing实例**

在Extjs/examples/direct目录下可以查看到ExtJS提供的一些使用DirectAPI的例子，这些例子都采用[**PHP**](http://lib.csdn.net/base/php)作为服务器端处理。当然在服务器端采用JAVA也很容易，不用担心，下面就来看以实际例子看DirectJNgine如何运作。

**4、1实例说明**

本例按照DirectJNing官方提供文档实例进行演示，项目目录结构图一：



图一

说明：

（1）      [**Java**](http://lib.csdn.net/base/java) Resources:src目录存放服务器端Java文件，其中log4j.properties可以参考DirectJNgine下载目录中提供的配置信息根据自己项目情况做配置即可；

（2）       WebContent/djn和WebContent/ejn存放djn-remote-call-support.js、deliverables/ejn-assert.js，这步不是必须的；

（3）       Ext-3.3.3：存放ExtJs类库文件；

（4）       test：存放实例源码，本例静态代码文件为testAction.html，下面会用到。

其他文件内容就不详述。

**4、2 在web.xml中配置DirectJNgineServlet**

在web.xml文件中添加如下配置：

<servlet>

<servlet-name>DjnServlet</servlet-name>

<servlet-class>

com.softwarementors.extjs.djn.servlet.DirectJNgineServlet

</servlet-class>

       <init-param>

           <param-name>debug</param-name>

           <param-value>true</param-value>

       </init-param>

       <init-param>

           <param-name>providersUrl</param-name>

           <param-value>djn/directprovider</param-value>

       </init-param>

       <!—moreparameters –->

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

       <servlet-name>DjnServlet</servlet-name>

       <url-pattern>/djn/directprovider/\*</url-pattern>

</servlet-mapping>

com.softwarementors.extjs.djn.servlet.DirectJNgineServlet即为DierctJNgine的Servlet引擎。初始化参数debug设置为true，DirectJNgine会为我们动态生成-dubug.js文件，不懂先不管后边一看便知。Servlet url-pattern必须使用“/\*”结尾，官方推荐使用/djn/directprovider/\*就别改了。

当然还用其他参数，别急慢慢来，这步是使用DirectJNgine的最基础配置，接着往下看。

**4、3 设计服务器端Java方法**

服务器端代码如下：

**package** com.demo.ext.directjngine;

**import**com.softwarementors.extjs.djn.config.annotations.DirectMethod;

**public class** TestAction {

    @DirectMethod

**public** StringdoEcho(String data){

**return** data;

    }

    @DirectMethod

**public double** multiply(String num){

**try**{

**double** num\_ = Double.*parseDouble*(num);

**return** num\_\*0.8;

       }**catch**( NumberFormatException e){

**throw** e;

       }

    }

}

说明：

（1）  TestAction实现两个方法：deEcho()实现返回输入参数，multiply()则对输入的参数做八折操作并返回折扣价。

（2）  @DirectMethod则对以上两个方法做注解，表明这些方法会被客户端调用，这两个方法会被DirectJNgine引擎自动生成客户端代码（可能不理解，没管理往下看，实践完后自然明白）。

EirectJNgine还有其他注解参数，先不管，目前掌握这个就够了，待原理搞懂了自己便可尝试使用其他参数。

**4、3 让JavaScript能够使用服务器端方法**

接下来就要配置4.2中的DirectMethod，使它能够暴露给js客户端，从而调用服务器端方法，很简单，只需要把4.2中的TestAction类在web.xml中配置下即可。web.xml配置代码如下：

<servlet>

……

<init-param>

         <param-name>apis</param-name>

         <param-value>test</param-value>

</init-param>

<init-param>

         <param-name>test.apiFile</param-name>

         <param-value>test/testApi.js</param-value>

</init-param>

<init-param>

         <param-name>test.apiNamespace</param-name>

         <param-value>Ext.test</param-value>

</init-param>

<init-param>

         <param-name>test.classes</param-name>

         <param-value>com.demo.ext.directjngine.TestAction </param-value>

</init-param>

……

</servlet>

参数都是键值对形式，以下配置说明：

键apis：可以理解把本模块应用程序起个别名，以便组织模块（个人理解）。本模块命名为test，如果有多个用逗号分隔即可；

键%apiName%.apiFile：本例中%apiName%=test（test.apiFile）即在apis中定义的别名，用于定义DirectJNgine自动生成的js代码存放位置。在本例存放到test/testApi.js中，这个文件服务器端会自动生成，注意这里的生成路径相对于webapp的跟目录。%apiName%.apiFile后面的“.apiFile”为固定模式，我们只需要根据项目配置%apiName%即可；

键%apiName%.aipNamespace：用于定义自动生成js代码的命名空间。同上我们只需要替换%apiName%为test即可。

键%apiName%.classes：为本模块依赖的后端java类，它的值可以有多个类，用逗号分隔即可。本例中就是com.demo.ext.directjngine.TestAction，这样客户端js就可以调用TestAction中所有使用@DirectMethod注解的方法。

通过以上配置服务器端Java类的方法就暴露给客户端JavaScript了，下面说下如何调用这些方法。发布项目重启web服务，可以查看到test文件夹下多了三个文件分别是：testApi.js、testApi-debug.js和testApi-min.js。打开testApi.js代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \*

 \* Codegenerated automatically by DirectJNgine

 \* Copyright(c) 2009, Pedro Agulló Soliveres

 \*

 \* DO NOTMODIFY MANUALLY!!

 \*

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Ext.namespace( 'Ext.test');

Ext.test.PROVIDER\_BASE\_URL=window.location.protocol+ '//' + window.location.host + '/' +(window.location.pathname.split('/').length>2 ?window.location.pathname.split('/')[1]+ '/' : '')  + 'djn/directprovider';

Ext.test.POLLING\_URLS = {

  message :Ext.test.PROVIDER\_BASE\_URL + '/poll/message' /\* () => String -- calls com.demo.ext.directjnging.TestAction.handlMessagePoll\*/

}

Ext.test.REMOTING\_API = {

  url:Ext.test.PROVIDER\_BASE\_URL,

  type:'remoting',

  actions: {

   TestAction: [

      {

        name:'doEcho'/\*(String) => String \*/,

        len: 1,

       formHandler: false

      },

      {

        name:'multiply'/\*(String) => double \*/,

        len:1,

       formHandler: false

      }

    ]

  }

}

有点感觉了吧，接着往下看。

**4、4 Ext.Direct调用服务器端方法**

**4.4.1 引入自动生成的js文件**

在testAction.html<head></head>引入如下代码：

<head>

……

<link rel=*"stylesheet"*type=*"text/css"*

    href=*"../ext-3.3.3/resources/css/ext-all.css"*/>

<script type=*"text/javascript"*

    src=*"../ext-3.3.3/adapter/ext/ext-base.js"*></script>

<script type=*"text/javascript"* src=*"../ext-3.3.3/ext-all.js"*></script>

<script type=*"text/javascript"* src=*"testApi.js"*></script>

……

<script type=*"text/javascript"*>

Ext.onReady(**function**(){

//下面再扩充

});

</script>

</head>

说明：

（1）       引入ext基础文件包含css和js文件；

（2）       引入testApi.js：这个文件时自动生成的，具体引用路径就是我们在test.apiFile中定义的路径，由于本例直接定义在test/testApi.js，与testAction.html同一级目录所以就直接引用没有加入目录结构，根据自己定义路径引用相应的文件

**4.4.2通过Ext.Drect注册服务器端方法**

为使Extjs能够调用java方法，需要注册一个远程供应器。扩充Ext.onReady以便注册服务器端方法，代码如下：

Ext.onReady(**function**(){

Ext.Direct.addProvider(

        Ext.test.REMOTING\_API

);

});

这里Ext.test是我们在web.xml定义Servlet参数的test.apiNamespace的值，REMOTING\_API是自动生成的testApi.js中定义的供应器配置。

**4.4.3调用服务器端方法**

我们通过构造一个TextField和一个DisplayField组件用于接收参数并输出处理后内容，通过Button组件调用事件，我们先看运行效果，再分析代码原理。继续扩充Ext.onReady代码如下：

Ext.onReady(**function**(){

Ext.Direct.addProvider(

       Ext.test.REMOTING\_API

);

**var** text = **new** Ext.form.TextField({//输入参数容器

    fieldLabel:"输入内容",

    width:230

});

**var** out = **new**Ext.form.DisplayField({ //输出内容容器

        cls: 'x-form-text',

        fieldLabel:"输出内容",

        width:230

});

**var** btn =  **new** Ext.Button({ //触发调用事件

    text:"调用Java方法",

    handler:**function**(){

**var** value = text.getValue();

        TestAction.doEcho(value,**function**(result ,e){  //这里是调用方法

**var** t = e.getTransaction();

**var** content = String.format('成功调用{0}.{1} ，返回数据为:{2}',t.action,t.method,result);

           out.setValue(content);

        });

    }

});

**var** panel = **new**Ext.form.FormPanel({

    title:"简单Ext.Direct例子",

    width:400,

    autoScroll:**true**,

    applyTo:Ext.getBody(),

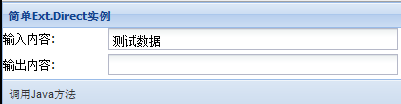
    items:[text,out],

    bbar:[btn]

});

});

运行页面数据数据点击按钮前如下图二：



图二

点击“调用Java方法”按钮后，如图三所示：



图三

从上图可以看出成功调用java方法，并返回数据。

**4、7 Ext.Direct调用过程分析**

（1）在4.3中我们看到，DirectJNgine通过web.xml配自动生成了testApi.js文件，通过查看Ext.Direct源码发现，Ext.Direct把我们使用@DirectMethod注册的方法都按照事件方法进行注册，事件名默认就是我们的函数名，如：”doEcho()等”。

（2）我们使用“类名.方法名”进行调用，方法参数依次传入，最后传入JavaScript函数作回调函数，用于接收服务端返回结果，如：TestAction.doEcho(value,function(result,e){……});如果有三个输入参数则按照如下形式调用：TestAction.doEcho(value1, value2, value3,function(result,e){……})。下面看下本例中调用过程：

TestAction.doEcho(value,**function**(result ,e){

**var** t = e.getTransaction();

**var** content = String.format('成功调用{0}.{1} ，返回数据为:{2}',t.action,t.method,result);

     out.setValue(content);

});

回调函数通过参数result接收返回数据，附加参数e为事件对象，它包含调用的Action和method名等其他信息。

**五、总结**

到这里DirectJNgine的简单使用方法基本介绍完毕，不知道你有没有为Ext.Direct这种数据通讯方案感到兴奋，至少我稍微兴奋了一把。它可以：后端实现和前台调用彻底分离，同样的后台实现可以很快换成其他方案，如Java换成php或.net，前端调用不需要任何改变，这一特性体现在它采用JSON作为数据交换格式，即testApi.js中自动生成的Ext.test.REMOTING\_API。