ArcGIS Server 9.3 for Java 讲座

1、前言------为什么我们选择 JSF，而不是其它 framework **错误！未定义书签。**

2、ArcGIS Server Java 开发 JSF 基础 1

3、ArcGIS Server Java ADF 开发的 JSF 知识介绍 3

4、ArcGIS Server 开发初步 -- 自定义工具 6

5、使用图片和 TrueType 字体进行标注 8

6、Server Java 讲座-----扩展 Tasks 框架 12

7、Server Java 开发实战---自定义 command 17

8、 ArcGIS Server Java 讲座--ADF 体系结构 第一部分 19

9、ArcGIS Server JAVA 讲座--实现后台 JSF Bean 和前台 Javascript 的联动 22

10、ArcGIS Server JAVA 讲座 --AJAX 基础 23

11、ArcGIS Server JAVA 开发讲座--- ADF 中的 Ajax 函数 25

12、ArcGIS Server Java 讲座--如何在用 JSF 在服务器端处理 Ajax 请求 28

13、ArcGIS Server Java 讲座----利用 Ajax 得到当前地图的比例尺（服务器端） 30

14、ArcGIS Server Java 讲座--用 Ajax 得到地图比例尺(客户端代码) 32

15、ArcGIS Server JAVA 讲座 自定义 Functionality 33

16、ArcGIS Server Java 讲座---用自定义 functionality 实现用户权限控制 35

17、Server Java 讲座-----扩展 Tasks 框架 41

18、ArcGIS Server Java 讲座--自定义 Tools 开发 47

19、ArcGIS Server Java 讲座－空间查询和高亮显示的实现 51

20、Server Java 自定义开发－Network Analysis 53

21、ArcGIS Server 开发——控制图层是否可见 58

22、航线查询问题－Server Java 的实现方法 58

23、ArcGis 航线查询完全例子 62

24、ArcGis Server 中 如何在 Task 中实现 下拉列表和 checkbox 和 radiobutton.（完整例子） 72

25、ArcGIS Server 开发——标注 73

26、ArcGIS Server Java 开发--Born for SOA 系列 前言 73

27、ArcGIS Server Java 开发--Born for SOA 系列 Web Service 基础 74

28、ArcGIS Server Java 讲座:Born For SOA--Server 对于 SOAP 的支持 80

2、ArcGIS Server Java 开发 JSF 基础

有很多开发人员问我，如果我已经有了一个 JSP 的 Application，为了使用 JSF，为了使用我们新版的 ArcGIS Server 的 Java ADF，我需要重新修改写我的程序以让它们适应整个 JSF 框架么？这是一个很有意思的问题，我可以想象出大家头皮发麻，左右为难的感觉，这也引出了今天的话题，Servlet 容器是如何对 JSF 的程序进行处理的呢？

随便找一个 JSF 的 Application，可以是我们的 ArcGIS Server 的 Sample，也可以是任何您从网上下载的 JSF 例子，打开 web.xml 文件，你就可以看到该文件里面有如下的 servlet 映射：

```
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
```

这是 Java 程序员最熟悉不过的 Servlet 配置了，该 Servlet 映射是什么意思呢？将所有对后缀名为 jsf 的请求，都交由 javax.faces.webapp.FacesServlet 来处理。哦，另外插一句，JSF 的标准是公开的，如果你觉得这个 FacesServlet 不好，你大可以自己写一个 JSF 处理 Servlet。将所有对 jsf 的请求自己来处理。事实上，已经有人这么做了，这就是 Myfaces，可以参考 Apache 的网站。

言归正转，如何将 JSF 的应用集成到已经有的应用程序中呢，您一定已经想到了。事实上事情很简单，在访问某个 JSF 页面的时候，映射到某个目录即可。这个目录在 web.xml 进行了设置，比如下面的配置就可以把所有的对 faces 的请求，让 Face Servlet 来处理：

```
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
```

所以，如果你在根目录了放了一个 hello.jsp 页面，这个 jsp 页面使用了 JSF 框架，那么你在访问这个 jsp 页面时用 http://servername/faces/hello.jsp 就可以了，你的容器会把这样的请求提交 Face Servlet 来处理，这个时候，就轮到 Face Servlet 来大显身手了，它会忠实地执行你的请求。

JSF 的 Managed Bean 是保存在 session 中的，所以你在 JSF 页面和普通 JSP 之间切换页面的

时候，不会丢失信息，反过来也一样。JSF 自己有自己的 Page Navigation 系统，但是它仍然可以跟普通的 JSP 页面互相切换，我们在后面的讲座中会讲到。

所以如果你已经存在的一个项目，想上我们的 ArcGIS Server，利用其基于 JSF 的 ADF 进行开发，也是完全可以的。当然，新的应用最好用 JSF 框架开发，开始时可能需要一点时间去学习，对于你后面会用到的强大的 JSF 的功能，这些投资是完全值得的。

欢迎大家进入美妙的 JSF 的世界。

了解了 Servlet 是如何处理你的 JSF 请求的，那么接下去我们来看看，让 JSF 程序跑起来需要什么必要条件，一个 Servlet 容器是必需的，这个一般取决于公司给我们配什么，或者客户要求什么。一般免费的好选择是 Tomcat，价格便宜量又足，我们大家都用它。当然如果项目大，经费充足，IBM 的 J2EE 容器 Websphere 和 Bea 公司的 Weblogic 也是一个好选择。选用大公司 J2EE 容器有一个好处，即使三更或者五更出了问题，你照样可以拍着桌子让他们的工程师过来帮你解决问题，如果你用我们 ESRI 的软件，你也可以这样；而用免费的软件三更出了问题只有我们自己出马了，当然，google 和百度在这个时候看在点击率的份上，还是可以拉我们一把的。除了一个好的容器，还需要一个 JSF 的实现，用 SUN 的 Reference Implementation 总是不会错的，人家是标准的制定者嘛，名字也起做“参考实现”，写来让你参考的。也可以选择功能更加强大的 Myfaces。ESRI 的 ArcGIS Server ADF 遵循标准的 JSF，所以 ADF 可以跑在 Sun 的 JSF 实现，或者 Myfaces 上。我们这次所有的教程都是基于 sun 的标准的 JSF 实现。还有其它的 JSF 实现，大家可以根据自己的爱好，择优选择之。如果你觉得都不好用，大可以参考标准自己写一个实现，除了你的老板（要投入更多的开发经费）和你的 team (要干更多的活) ，是没有人有意见的。

如果您像我一样选择了 SUN 的参考实现，那么去 SUN 的网站上去下载一个 JSF 的开发包吧，如果不想下载，随便找一个 Server Java ADF 的例子，在 WEB-INF/lib 目录里面有 JSF 的包，名为 jsf-api.jar 和 jsf-impl.jar 的两个包就是；注意那个 JSP 的标准标签库 jstl.jar，把它跟 jsf 的包放在一起，你不一定用标准标签库，但是 JSF 用到了它。

我把 JSF 的 doc 放在这里给大家下载，因为这个东西实在不好找，有人知道好的下载 URL，可以在这里贴一个。

把这些包放在一个 web application 的 lib 目录里面有，你的程序你可以使用 JSF 的强大功能了。让我也来俗气一下，写一个简单的 hello world 作为今天的结束吧。注意在你的 web application 里面的 web.xml 配置文件添加了 Face Servlet 的声明和 URI 映射。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page c%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<meta http-equiv="Content-Type" c>
<f:view>
  <html>
    <head>
```

```
            <title>
              欢迎来到 ArcGIS Server ADF 世界
            </title>
          </head>
          <body>
            <h:form id="welcomeForm">
              <h:outputText id="welcomeOutput" value="欢迎来到 ArcGIS Server Java ADF 教程!"
style="color: red;"/>
              <p>
               <h:message id="errors" for="helloInput" style="color: red"/>
              </p>
              <p>
                <h:outputLabel for="helloInput">
                  <h:outputText id="helloInputLabel"
                                value="123"/>
                </h:outputLabel>
            </h:form>
          </body>
        </html>
      </f:view>
```

保存后访问 Http://servername:port/faces/hello.jsp 即可，如果页面能够出来，那么恭喜您，您已经迈入了 JSF wonderland 的大门。

3、ArcGIS Server Java ADF 开发的 JSF 知识介绍

ArcGIS Server 的 Java ADF 开发使用到了 JSF 的知识，下面结合 ArcGIS Server 的开发，介绍一下 JSF 的知识

Struts 和 JSF/Tapestry 都属于表现层框架，这两种分属不同性质的框架，后者是一种事件驱动型的组件模型，而 Struts 只是单纯的 MVC 模式框架。

首先事件是指从客户端页面（浏览器）由用户操作触发的事件，Struts 使用 Action 来接受浏览器表单提交的事件，这里使用了 Command 模式，每个继承 Action 的子类都必须实现一个方法 execute。

在 struts 中，实际是一个表单 Form 对应一个 Action 类(或 DispatchAction)，换一句话说：在 Struts 中实际是一个表单只能对应一个事件，struts 这种事件方式称为 application event，application event 和 component event 相比是一种粗粒度的事件。

struts 重要的表单对象 ActionForm 是一种对象，它代表了一种应用，这个对象中至少包含几个字段，这些字段是 Jsp 页面表单中的 input 字段，因为一个表单对应一个事件，所以，当我们需要将事件粒度细化到表单中这些字段时，也就是说，一个字段对应一个事件时，单纯使用 Struts 就不太可能，当然通过结合 JavaScript 也是可以转弯实现的。

而这种情况使用 JSF 就可以方便实现，

```
<h:inputText id="userId" value="#{login.userId}">
    <f:valueChangeListener type="logindemo.UserLoginChanged" />
</h:inputText>
```

#{login.userId}表示从名为 login 的 JavaBean 的 getUserId 获得的结果，这个功能使用 struts 也可以实现，name="login" property="userId"

关键是第二行，这里表示如果 userId 的值改变并且确定提交后，将触发调用类 UserLoginChanged 的 processValueChanged(...)方法。

JSF 可以为组件提供两种事件：Value Changed 和 Action.相当于 struts 中表单提交 Action 机制，它的 JSF 写法如下：

```
<h:commandButton id="login" commandName="login">
    <f:actionListener type=”logindemo.LoginActionListener” />
</h:commandButton>
```

从代码可以看出，这两种事件是通过 Listerner 这样观察者模式贴在具体组件字段上的，而 Struts 此类事件是原始的一种表单提交 Submit 触发机制。如果说前者比较语言化（编程语言习惯做法类似 Swing 编程）；后者是属于 WEB 化，因为它是来自 Html 表单。

基本配置

Struts 和 JSF 都是一种框架，JSF 必须需要两种包 JSF 核心包、JSTL 包（标签库），此外，JSF 还将使用到 Apache 项目的一些 commons 包，这些 Apache 包只要部署在你的服务器中既可。

JSF 包下载地址：http://java.sun.com/j2ee/javaserverfaces/download.html 选择其中 Reference Implementation。

JSTL 包下载在 http://jakarta.apache.org/site/downloads/downloads_taglibs-standard.cgi

所以，从 JSF 的驱动包组成看，其开源基因也占据很大的比重，JSF 是一个 SUN 伙伴们工业标准和开源之间的一个混血儿。

上述两个地址下载的 jar 合并在一起就是 JSF 所需要的全部驱动包了。在 ArcGIS Server 的项目下的 lib 目录下有相关的库文件如下：commons-beanutils.jar commons-collections.jar commons-digester.jar commons-lang-2.0.jar commons-logging.jar jsf-api.jar jsf-impl.jar jstl.jar standard.jar

与 Struts 的驱动包一样，这些驱动包必须位于 Web 项目的 WEB-INF/lib，和 Struts 一样的是也必须在 web.xml 中有如下配置：

```
<web-app>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.faces</url-pattern>
    </servlet-mapping>
</web-app>
```

这里和 Struts 的 web.xml 配置相似，简直一模一样。

正如 Struts 的 struts-config.xml 一样，JSF 也有类似的 faces-config.xml 配置文件：

```
<faces-config>
    <navigation-rule>
        <from-view-id>/index.jsp</from-view-id>
        <navigation-case>
            <from-outcome>login</from-outcome>
            <to-view-id>/welcome.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <managed-bean>
        <managed-bean-name>user</managed-bean-name>
        <managed-bean-class>com.corejsf.UserBean</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>
</faces-config>
```

在 Struts-config.xml 中有 ActionForm Action 以及 Jsp 之间的流程关系，在 faces-config.xml 中，也有这样的流程，我们具体解释一下 Navigation：

举例：如果在 mapviewer.jsp 中有一个事件：

```
<h:commandButton label="Login" action="login" />
```

action 的值必须匹配 form-outcome 值，上述 Navigation 配置表示：如果在 index.jsp 中有一个 login 事件，那么事件触发后下一个页面将是 welcome.jsp

JSF 有一个独立的事件发生和页面导航的流程安排，这个思路比 struts 要非常清晰。

UI 界面

JSF 和 Struts 一样，除了 JavaBeans 类之外，还有页面表现元素，都是是使用标签完成的，Struts 也提供了 struts-faces.tld 标签库向 JSF 过渡。

使用 Struts 标签库编程复杂页面时，一个最大问题是会大量使用 logic 标签，这个 logic 如同 if 语句，一旦写起来，搞的 JSP 页面象俄罗斯方块一样，但是使用 JSF 标签就简洁优美：

```
<jia:navigatorItem name="inbox" label="InBox"
    icon="/images/inbox.gif"
    action="inbox"
    disabled="#{!authenticationBean.inboxAuthorized}"/>
```

如果 authenticationBean 中 inboxAuthorized 返回是假，那么这一行标签就不用显示，多干净利索！

缺点：

JSF 这类框架面临的最大问题，它大量的使用了标签库，目前流行的网页制作工具(如 deamweaver)又没有提供足够的支持，所以只能依靠挖掘 dataTable 标签的各个属性，并且大量

依赖 css 才能实现页面的美化。如果 java 世界能有一个强大的 JSF IDE，能够提供 vs.net 一样的能力，那么 JSF 也许会更容易流行。

4、ArcGIS Server 开发初步 -- 自定义工具

在 Server 生成的 Web App 中，页面的工具按钮可以分为两类：

l 命令（Command）：A command is an element on a JSP page that triggers a server side action without any further interaction on the client. An example of a command in the sample application is the "zoom to full extent" button. Once the user clicks the button, a method is called on the server。不与用户通过界面交互。

l 工具（Tool）：A tool has further client side interaction before calling a method on the server. An example of a tool in this application is "zoom to rectangle". Once the user clicks the button, drags a rectangle over the map indicating the area they want to zoom to, and then a method is called on the server。与用户通过界面交互。

一、继承接口

public Interface com.esri.adf.web.faces.event.MapToolAction{

void execute(MapEvent event);

}

lMapToolAction 接口代表由 MapControl 控件事件所激活的服务器端工具，系统已预设继承此接口的类：

PanToolAction（平移），

ZoomInToolAction（放大），

ZoomOutToolAction（缩小）

lMapControl 创建 MapEvent 事件并将其传给继承接口的工具类的 execute(MapEvent) 函数，The business logic for the tool should be implemented in this method according to the event。

二、工具在 JSP 页面上的 tag 表达如下：

<ags:tool

serverAction="com.esri.adf.web.faces.event.ZoomInToolAction"

clientAction="EsriMapRectangle"

clientPostBack="true"

/>

三、注册 managed-bean 将所写的类作为一个 managed-bean 注册到 faces-config.xml，并用 WebContext 实例作为其初始化参数：

<managed-bean>

<managed-bean-name>ToolClass</managed-bean-name>

<managed-bean-class>com.brsc.MyToolClass</managed-bean-class>

<managed-bean-scope>request</managed-bean-scope>

<managed-property>

<property-name>webContext</property-name>

<value>#{mapContext}</value>

</managed-property>

</managed-bean>

四、注释：

1． JSP 的 Tag 中 serverAction 写入继承 MapToolAction 接口的类（全称），代表对于此工具服务器端要进行的操作[execute(MapEvent event)]

用户也可以使用任何 Managed Bean 的函数作为工具对应的方法，只要这个函数使用如下声明：

public void anyMethodName(MapEvent event)

JSP 标签使用 serverMethod ，如下：

<ags:tool serverMethod="#{bean.anyMethodName}" ... />

这样，MapControl 也会将适当的 MapEvent 事件传入此函数。

2． JSP 的 Tag 中 clientAction 写入客户端鼠标选择的方式：

EsriMapPoint 点选

EsriMapLine 线

EsriMapRectangle 四边形

EsriMapCircle 圆

EsriMapOval 椭圆

EsriMapPolyline 多线

EsriMapPolygon 多边形

EsriMapPan 移动

对应 Server 端的几何形状（附图）

3． MapEvent 代表客户端进行操作产生的事件，一般会用到 MapEvent 的

public WebGeometry getWebGeometry()函数来得到客户端输入的几何形状

//Returns the WebGeometry in screen coordinates corresponding to

//the client action performed by the user.

来获得客户端产生的形状，这些 Geomentry 一般都是 screen 坐标，需要用 toMapGeometry(WebMap)转换为 地图坐标 。

一般操作如下：

```
public void myToolMethod(MapEvent event) {
   WebContext ctx = event.getWebContext();
   WebGeometry screenGeom = event.getWebGeometry();
   WebGeometry mapGeom = screen.toMapGeometry(ctx.getWebMap());
   ...
}
```

4. JSP 的 Tag 中 clientPostBack

- 设置为 false，刷新地图，并且刷新页面；
- 设置为 true，只刷新地图，不刷新页面；

5、使用图片和 TrueType 字体进行标注

直奔主题吧，接上个主题的讲座的内容，讲讲如何用图片进行标注。在前面一个讲座我们已经说明，如何对选中的物体进行高亮显示。而且这一部分工作是在 ADF 这一端完成的。那么有的时候，我们进行高亮显示的时候，不仅仅希望只是设置颜色，我们希望能够用图片或者 truetype 字体进行标注。比如在犯罪地点放一个坏人之类的功能。

我们来看看实现原理，关键是两个类，WebTrueTypeMarkerSymbol 和 WebPictureMarkerSymbol，没啥好说的，直接用代码来说明吧：

先来看看普通的点标注：

```
WebPointpt=(WebPoint)arg0.getWebGeometry().toMapGeometry(arg0.getWebContext().getWebMap(
));

WebSimpleMarkerSymbol markers =null;

markers = new WebSimpleMarkerSymbol();

markers.setAntialiasing(true);
```

```
markers.setColor("255,0,0");

markers.setWidth(8);

markers.setOutlineColor("255,0,0");

markers.setMarkerType(WebSimpleMarkerSymbol.CIRCLE);

markers.setPicture(bytInput);

GraphicElement ge=new GraphicElement();

ge.setGeometry(pt);

ge.setSymbol(markers);

WebGraphicsgraphics=arg0.getWebContext().getWebGraphics();

graphics.addGraphics(ge);

arg0.getWebContext().refresh();
```

注意 webgraphicsymbol 的 setPicture 方法的参数不是图片目录，而是图片的二进制数组，所以需要用文件 IO 把图片读取进来。当然，如果用户访问量很大，线程就不安全了，大家可以在 application 启动时进行读取，放在 context 的某个 attribute 里面。我原来认为是通过设置路径方式实现，这样又可能可以搞定 gif 图形的闪烁，但是现在实验结果是不行。设置图片标注的代码如下：

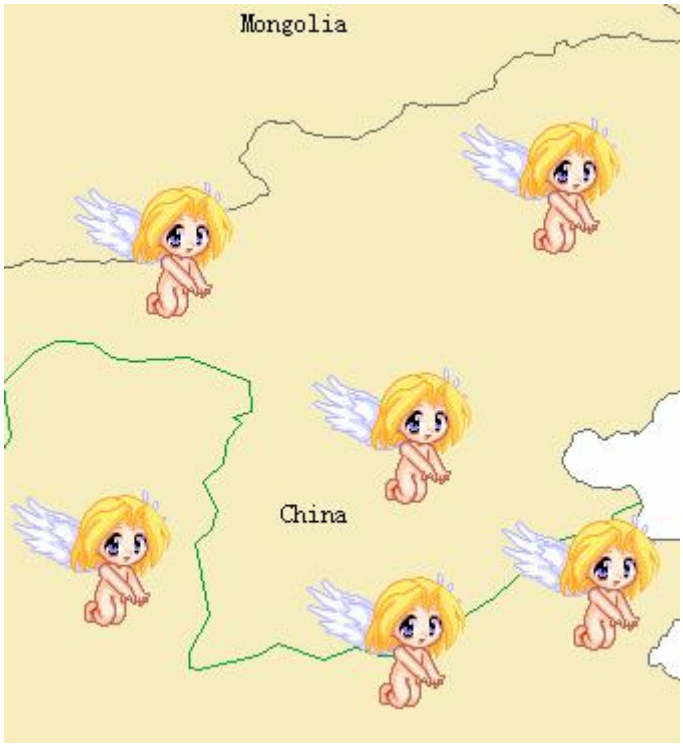
```
    WebPointpt=(WebPoint)arg0.getWebGeometry().toMapGeometry(arg0.getWebContext().getWebMap(
));

//图片在 servelet 容器里面目录

String
picPath=FacesContext.getCurrentInstance().getExternalContext().getRequestContextPath()+"\\images\\angl
e.gif";

File myFile = new File(picPath);

FileInputStream myStream= newFileInputStream(myFile);

BufferedInputStream buf = newBufferedInputStream(myStream);
```

```
byte[] bytInput = newbyte[(int)myFile.length()];

buf.read(bytInput, 0, (int) myFile.length());

buf.close();

myStream.close();

WebPictureMarkerSymbol markers=newWebPictureMarkerSymbol();

markers.setPicture(bytInput);

GraphicElement ge=new GraphicElement();

ge.setGeometry(pt);

ge.setSymbol(markers);

WebGraphics graphics=arg0.getWebContext().getWebGraphics();

graphics.addGraphics(ge);

arg0.getWebContext().refresh();
```

在 jsf 文件里面添加如下代码，调用图片标注工具，进行测试：

```
<a:tool id="pointTest"defaultImage="images/point.gif"hoverImage="images/pointU.gif"

selectedImage="images/pointD.gif"clientAction="EsriMapPoint"
serverAction="com.cj.ucdemo.GifTestTool"clientPostBack="true"/>
```

图片标注的结果如下：



用图片做 markSymbol

我们也可以使用 TrueType 字体里面的矢量字体对图形进行符号化显示，这也是我们经常使用的方法，具体代码如下：

```
webMap=arg0.getWebContext().getWebMap();
WebPointpt=(WebPoint)arg0.getWebGeometry().toMapGeometry(webMap); WebTrueTypeMarkerSymbol
trueMarkerSymbol=new WebTrueTypeMarkerSymbol(); //注意使用系统里面已经安装的字体
trueMarkerSymbol.setFontName("ESRITransportation & Civic");//设置索引
trueMarkerSymbol.setCharacterIndex(8); trueMarkerSymbol.setFontColor("255,0,0");
trueMarkerSymbol.setFontSize(20);
trueMarkerSymbol.setFontStyle(WebTrueTypeMarkerSymbol.BOLD);GraphicElement ge=new
GraphicElement(); ge.setGeometry(pt);
ge.setSymbol(trueMarkerSymbol);WebGraphicsgraphics=arg0.getWebContext().getWebGraphics();
graphics.addGraphics(ge);arg0.getWebContext().refresh();
```

把上面的代码放在一个点击工具里面，在 jsf 文件中用如下代码进行调试：

```
<a:toolid="trueMarkerTest"
defaultImage="images/point.gif"hoverImage="images/pointU.gif"                selectedImage
="images/pointD.gif"clientAction="EsriMapPoint"                serverAction="com.cj.ucdem
o.TrueTypeMarkerTest"clientPostBack="true"/>
```

TrueType 字体标注的结果如下图所示：

6、Server Java 讲座-----扩展 Tasks 框架

Tasks 框架是一个很有意思的东西。如果你对 Tasks 框架还没有概念，你可以参考一下 web map application 那个 Sample，左边浮动的放大工具，查询工具都是用 Tasks 框架构成。有人会问，既然已经有了 command 和 tools，为什么还需要 tasks 呢？事实上 tasks 的目的是不同的；我们可以看一下 web map application，可以看到，相同类型的 button 和 toos 组合在一起，组成一个逻辑单元，这就是 tasks 的选择条件。

我们来看看如何编写 Tasks，编写 Tasks 是非常有意思的事情。Tasks 跟 tools 一样，也是一个普通的 Java 类即可，这个 Java 类可以从 Object 直接继承。请看下面的 Task 的例子：

```
1 public class JSTasks {
2 public void doSomething(){
3 System.out.println("Task button clicked.");
4
5 }
```

```
6 }
```

这个 tasks 实在是简单吧，看看怎么部署上去，我们在 xml 文件中做如下定义：

```
1  <managed-bean>
2      <managed-bean-name>jsTask</managed-bean-name>
3      <managed-bean-class>com.cj.ucdemo.JSTasks</managed-bean-class>
4      <managed-bean-scope>session</managed-bean-scope>
5  </managed-bean>
```

这个配置文件也非常简单，那么看来所有的诀窍在与如何在 JSP 页面里面使用它。我们来看看：

```
1 <a:task  mapId="Map0"  value="#{jsTask}"      rendered="true"  />
```

添加一个 task 标签，把这个标签的 mapId 值指向地图控件"Map0"，讲它的值指向我们刚才定义的 Managed Bean。访问一下，大家看看下面的浮动窗口结果：

是不是很令人惊奇啊？事实上我们这个 Bean 中只有一个方法，但是 Tasks 框架聪明地把这个方法名给提取出来，作为 button 的名字。你点击一下这个按钮，在后台就可以看到执行了这个方法，打印出了一条信息。简单的说：方法就是按钮！！

有了上个 Task 的基础，我们来看其它的 Task 就驾轻就熟了。事实上，整个 Task 就是一个类，Task 的 Caption 就是这个类的类名。里面的每一个 Public 方法都是 Task 浮动条上的一个按钮。那么参数怎么办呢？我们来加一个参数，并且加上一个它的 get 方法和 set 方法：

```
1 String  parameter;
2public  String  getParameter()  {
3return  parameter;
4}
5public  void  setParameter(String  parameter)  {
6this.parameter  =  parameter;
7}
```

看看我们的 task 工具条发生了什么变化，Task 工具条如下图所示：

可以看到，多了一个 Parameter 的输入框，前面还有一个 Label，很有意思吧。那个 DoAnything 是我加的另外一个方法。Task 框架的扩展已经很明显了吧：把相同的功能集合在一个类里面，这个类可以接受参数。例如我们可以利用这个 task 进行 buffer 操作，这个输入框就可以用来输入 buffer 的距离。

看到这里，大家肯定想到了另外的几个问题，比如：怎么把这些参数，按钮的名字修改成中文，上次给人培训，有人说：用中文的变量名？也是一个办法，但是考虑的 JDK 对中文变量的处理，似乎有问题。还有其它的问题包括怎么跟地图进行交互？怎么样设置客户端的动作（画多边形还是画 Polyline）。我们当然有另外的方法。

使用 Taksinfo 建立对 Task 的描述,所有的 Taskinfo 都需要继承自 SimpleTaskInfo 类。大家可以打开 doc 看看 SimpleTaskInfo 的子类有那些。事实上，那些子类都是已经存在的 task 的 taskinfo，如果你需要做汉化，继承这些类并且改写其中的几个方法即可。

我们来看看我们自己的类需要改写 SimpleTaskInfo 的哪几个方法：
TaskActionDescriptorModel[]getActionDescriptors();

此方法用来修改 command 按钮的描述；command 的意思是不需要和地图进行交互而是直接在服务器端执行一个命令。

TaskParamDescriptorModel[]getParamDescriptors()

此方法修改参数的描述；

TaskToolDescriptorModel[]getToolDescriptors();

此方法修改工具的描述，工具的意思是需要和地图进行交互；

那么我们来写一个类，直接说明如何使用这几个方法，修改 task 的表现：

```
    public TaskDescriptor getTaskDescriptor() {

TaskDescriptor td=new TaskDescriptor(JSTasks.class);

td.setDisplayName("我的任务");

return td;

}
```

此方法修改了 task 的标题，将此类作为一个 Managed-bean 添加到 faces-config.xml 文件中，并且在.jsp 页面中修改 task 的使用如下：

```
<a:task mapId="Map0"  value="#{jsTask}" rendered="true" taskInfo="#{jsTaskInfo}"/>
```

修改后的 Task 如下所示：

可以看到 task 的标题已经改变。下面我们来看修改 Action 的标题，代码如下：

```
public TaskActionDescriptorModel[]
getActionDescriptors() {
TaskActionDescriptorModel[] td=new TaskActionDescriptorModel[1];
TaskActionDescriptorModel actionDesc=new TaskActionDescriptor(JSTasks.class,"doSomething","查询");
td[0]=actionDesc;
return td;
}
```

需要说明的是 TaskActionDescriptor(JSTasks.class,"doSomething","查询")构造函数的三个参数，第一个是 task 类，第二个是方法名，第三个是修改后的方法名；构造后放到数组中返回即可，如果有多个方法，可以如法炮制，都放到数组中。修改后的 task 有如下表现：

你会发现另外一个 action 没有了，别着急，因为我们还没有把它放到数组中，而原来默认的显示方式已经被我们修改了。

我们来修改 doAnything 工具，doAnything 工具需要和地图交互，注意它的参数是 MapEvent 而不是 TaskEvent，它完整的代码如下所示：

```
    public void doAnything(MapEvent te){

//通过和地图交互进行放大

WebContext ctx=te.getWebContext();

WebGeometry geom=te.getWebGeometry().toMapGeometry(ctx.getWebMap());

WebExtent ext = (WebExtent)geom;

ctx.getWebMap().setCurrentExtent(ext);

ctx.refresh();

}
```

如果跟地图交互，就要涉及到客户端执行的操作，我们这里没有任何地方指定客户端的操作，这是在 taskinfo 里面指定的，我们来看下面的代码：

```
    public TaskToolDescriptorModel[]

getToolDescriptors() {

TaskToolDescriptor [] toolDesAry=new

TaskToolDescriptor[1];

TaskToolDescriptor toolDesc=new

TaskToolDescriptor(JSTasks.class, "doAnything", "交互放大",ClientActions.MAP_RECTANGLE);

toolDesAry[0]=toolDesc;

return toolDesAry;

}
```

注意 TaskToolDescriptor 构造函数的 4 个参数，第一个是 task 类，第二个是原方法名，第三个是替换后的方法名，第四个是客户端的动作。构造完成后放到数组中，如果有多个 tools，如法炮制即可。修改后的 task 如下所示：

有了上面的解释，我们来看参数的 taskinfo 代码，就非常容易了：

```
public TaskParamDescriptorModel[] getParamDescriptors() {

// TODO Auto-generated method stub

TaskParamDescriptorModel[] td=new TaskParamDescriptorModel[1];

TaskParamDescriptorModel paraDesc=new TaskParamDescriptor(SearchTask.class,"searchText","查询文本");

td[0]=paraDesc;

return td;

}
```

修改后的 task 如下所示：

Task 框架的基本构成就是这样了，祝大家使用愉快。下节将介绍 TaskResult 的使用。

7、Server Java 开发实战---自定义 command

这个文档大家如果喜欢看英文，在 ESRI 的帮助里面就有。本文和 ESRI 的帮助文档类似，当然，我写的会加入自己的一些东西。

咱开发人员拿到一个开发平台，当然是要用来开发的，而且我们最好把它改得面目全非，才能显出开发人员我的水平所在。对于那些拿着模版生成一个 web gis 站点就去问客户要钱的事情，对于一个有自尊的开发人员，一般是不屑做的。

所以，我们今天就讲讲如何自定义命令和工具，先说说自定义命令吧。

命令事实上是 command，也就是平常所在 html 页面里面的 command，这些 command 里面就是命令按钮。在我们的 ArcGIS Server 的 Web ADF 里面，command 被用来做一些不用跟地图交互的工作；比如缩放到全图：你不需要跟地图做任何交互，只要一按按钮，地图就缩放到全图。其它用处，就靠大家去发挥想象了，相信大家在以后的项目中会经常用到。我们以一个例子作为说明：

```
package com.demo;
import javax.faces.event.ActionEvent;
import com.esri.adf.web.data.WebContext;
import com.esri.adf.web.data.geometry.WebExtent;
import com.esri.adf.web.faces.event.MapEvent;
import com.esri.adf.web.faces.event.MapToolAction;
public class MyFullExtent {
WebContext context;
public WebContext getContext() {
return context;
}
public void setContext(WebContext context) {
this.context = context;
}
public void setFullExtent(){
try{
WebExtent ex=context.getWebMap().getFullExtent();
context.getWebMap().setCurrentExtent(ex);
context.refresh();
}catch(Exception ex){
ex.printStackTrace();
}
}
}
```

这是无比简单的一个 command，简单到我都不好意思发出来给大家看了。注意这个类没有实现

任何接口，也没有扩展任何现有的类，除了祖宗 Object。它有一个 WebContext，从上面一个讲座我们可以知道，这个类可是我们整个 Web ADF 的关键，有了它，可以得到所有的东西了。

这个类关键的方法是 setFullExtent，里面的几行代码用来将地图缩放到全图，大家看看代码应该就可以明白。

怎么进行部署呢？就像普通的 managed bean 一样进行部署就可以了，打开 jsf application 的心脏 faces-config.xml 文件，在里面添加如下配置：

```
<managed-bean>
        <managed-bean-name>myFullExtent</managed-bean-name>
        <managed-bean-class>com.demo. MyFullExtent </managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
        <managed-property>
          <property-name>context</property-name>
          <value>#{mapContext}</value>
        </managed-property>
  </managed-bean>
```

注意我们把 web context 作为一个属性，用来初始化 MyFullExtent，#{mapContext}指向了一个在 faces-config.xml 文件中定义的另外一个 Bean，这个 bean 我们上次讲座已经详细解释过了。这里说一句废话，算是提醒一下，有请求过来的时候，先实例化这个类，然后再设置属性，有的工程师想在实例化里面进行一些操作，就必要用 webContext，会报告空指针错误，因为这个时候 webContext 属性还没有被设置。

类写好了，也配置好了，怎么用呢，再给个例子：

```
<a:command id="fullExtent" action="#{myFullExtent. setFullExtent }" toolText="全图显示" />
```

自定义 command 就这样完成了。

8、 ArcGIS Server Java 讲座--ADF 体系结构 第一部分

这两天一直在等待 9.2 Engine 到货，好调试我写的例子，但是看来还得等一下，所以今天的这个讲座将向前跨越一大步。等 Enging 到货后把前面的部分，也就是如何自己写一个 ADF 补上。

有很多人看了前面的讲座问我，怎么讲了半天，还是没有讲到如何对 server 进行开发，就像吃浙大的豆沙包，咬了一大口，发现离馅还有 300 里。这句话提醒了我：那么今天我们直接吃豆沙馅了。

在讲一些 sample 之前，我们来看看整个 ADF 的体系架构，ADF 里面有大部分类是纯 JAVA 的。

它是构建在 JSF 之上的，所以它严格地遵守 JSF 的规范，所有的 Bean 都放在 faces-config.xml 和 web.xml 这样的配置文件中。我们今天吃豆沙馅，所以就讲讲这两个文件里面的配置；

Faces-config.xml 文件中有如下配置：

<managed-bean-name>esriWebSession</managed-bean-name>

<managed-bean-class>com.esri.adf.web.data.WebSession</managed-bean-class>

<managed-bean-scope>session</managed-bean-scope>

<managed-property>

<property-name>webApplication</property-name>

<value>#{esriWebApplication}</value>

</managed-property>

< /managed-bean>

这里定义了一个 managed bean 指向了 WebSession 的类，这个 WebSession 类实现了一个接口，这个接口是 HttpSessionListener，看看这个接口的 Javadoc 吧；这个接口做两件事情：每当往 Session 中存入一个对象（setAttribute）或从 Session 中删除一个对象的时候，如果这个对象实现了此监听器接口，应用服务器将会自动调用接口相应的方法。你有机会在这两个方法里面做一些事情，最好的莫过于初始化存放在 session 中的变量。如果你一定要问我，Managed Bean 的 scope 如果是 session，那么在每个新 session 中，都会初始化这个 managed bean，调用它的 constructor，这样不是也可以实现 managed bean 地初始化么？如果你一定要这样写，也可以，但是系统的可扩展性很差啦。

在 ADF 中，如果我们写了一个类，并且希望这个类能够被 ADF 初始化，我们直接实现 WebContextInitialize 接口就可以了，ADF 会自动初始化这个类，那么 WebContextInitialize 接口是什么呢？

首先，来看看我们的核心中的核心 WebContext，Adf 的广大 Managed bean 们都团结在 WebContext 周围，错了，是团结在 WebContex 里面；这个 WebContext 是这个 Server ADF 的容器。Context 这个词翻译成上下文实在是不合适，我们还是叫它叫容器吧。

这个容器里面有什么？随便举个例子吧，它有很多的 Attribute，比如有一个 WebMap,所有容器里面的 Attribute 都需要实现一个接口，这个接口的名字叫做 WebContextInitialize，这个伟大的接口有两个方法：init 和 destroy。看看这两个方法，我不说你都能想到它们是用来做什么的。初始化这个 attribute，和销毁这个 attribute。看个帮助里面的例子吧：

```
WebContext context;
public void init(WebContext context) {
   this.context = context;
   //初始化其它资源或者变量
}
```

```
    public void destroy() {
        context = null;
        //释放其它资源
    }
}
```

OK，这个 attribute 也将作为一个 managed bean 部署在 faces-config.xml 文件里面。这里引出的问题是，谁来调用这个 init 方法和 destroy 方法呢？对 JSF 来说，这两个只是普通的方法，JSF 不会调用这两个方法来进行资源的初始化和释放。答案是 WebContext 来调用这些初始化和释放的方法，我们可以看 WebContext 的 doc 原文：

WebContext is responsible for making the callback methods implemented by the resources and attributes at appropriate junctures of the ADF application. The callback methods are declared in the WebContextInitialize, WebContextObserver and WebLifecycle interfaces.

原来是 WebContext 负责调用这三个接口的方法，初始化和释放资源。

我们看一下 WebContext，发现它也是需要有人来初始化它的，那么 WebContext 是谁来负责的呢？是 WebSession。WebSession 里面存放了所有的 WebContext，并且保证这些 WebContext 们能够被正常的初始化和 destroy，什么时候初始化，什么时候 Destroy，当然由 WebSession 实现的，大名鼎鼎的 HttpSessionBindingListener 接口来实现，每次有一个用户访问或者用户退出时，该监听器会自动调用两个方法，用来初始化和摧毁 WebContext。另外要说一句地时：一个 session 可以有多个 webContext，你可以用 getWebContexts()得到所有的 WebContext。目前我们看到的所有的配置文件都只使用了一个 WebContext，多个 WebContext 会使用在什么地方呢，这个问题大家先考虑一下，我在后面的讲座中会揭晓答案。

如果有多个用户访问时，就会产生多个并发用户，我们的 Web Server 会为每个用户都分配一个 WebSession，这些 Websession 会放在哪里，答案是放在 WebApplication 中，我们打开 faces-config.xml 文件可以看到里面有一个 webApplication 的定义

```
<referenced-bean-name>esriWebApplication</referenced-bean-name>
<referenced-bean-class>com.esri.adf.web.data.WebApplication</referenced-bean-class>
</referenced-bean>
```

再去看 WebSesstion 的定义，有指向 esriWebApplication 的引用。

现在的问题是，是谁，初始化了 WebApplication，我们祭出最原始也是最强大的武器了：ServletContextListener 出马了，看 web.xml 中下面的项定义，

```
    <listener-class>com.esri.adf.web.util.ADFServletContextListener</listener-class>
  </listener>
```

ADFServletContextListener 实现 ServletContextListener 接口，ServletContextListener 有两个方法：

contextInitialized()和 contextDestroyed(); 分别在 web application 启动的时候和结束的时候调用，你要是觉得好玩，可以写一个 listener 玩玩，我这里写一个简单的 listener 现现丑：

```
public void contextInitialized(ServletContextEvent arg0) {
System.out.println("context object inited.");
}
public void contextDestroyed(ServletContextEvent arg0) {
System.out.println("context object destroyed.");
}
}
```

ESRI 使用了 ADFServletContextListener，它实现了 ServletContextListener 接口，用来启动和 destroy WebApplication。看看 Javadoc 的原文吧：

The ADFServletContextListener initializes a WebApplication object and sets it as an attribute on the ServletContext. When the ServletContext is destroyed, it destroys the associated WebApplication and attempts to clean up all server hosted objects in the current thread.

This class must be registered as a listener-class (in the web.xml file) for the web application.

现在大家都明白了吧，整个调用过程。Servelt 容器启动的时候，会发送启动消息给 ADFServletContextListener，ADFServletContextListener 负责创建 WebApplication，WebApplication 负责创建和存放 WebSession，WebSession 负责创建 WebContext，WebContext 负责创建所有的 Attribute，对资源的使用等等。这一下大家都满意了吧。

谨以此文，纪念在浙大吃过的一千个豆沙包。

9、ArcGIS Server JAVA 讲座--实现后台 JSF Bean 和前台 Javascript 的联动

很多开发人员都梦寐以求的方案，我在论坛里面也看到很多人提出这样的问题，今天集中对这些问题做一个解答，也是作为 Server JAVA 讲座的一个部分。另外，要说明的是，今天的讲座不包括 AJAX 技术。AJAX 技术会另开一个讲座。

先以一个问题开始，如果我们的 webapplication 中对 geodatabase 进行了编辑，大家都知道，这个时候需要一个提交编辑并保存，或者提交编辑并取消的过程，大家都用过 ArcMAP，如果没有用过 ArcMAP，word 总是用过的，编辑过的东西，一关闭，肯定会给你警告。这个问题在桌面程序中简直就不是问题，在 web 上，我们倒是可以来说一说了，因为 server 和你的 browser 可能相距几万公里，你这边要关闭 browser，在美国的服务器怎么能够知道你要关闭呢？服务器上运行的是 JSF 的 Bean，而客户端运行的是 Javascript，怎么让 Javascript 去感知后台的情况，给用户一个提示，这样就会很好地显示我们程序的友好性，对建设以人为本的和谐社会也大有好处。那么我们来看看如

何做这个事情。

Browser 关闭时会产生 onUnload 事件，有了这个事件，使我们有机会做一些事情，比如下面的语句，显示一个信息

<body onUnload="javascript: alert('你要关闭我么？')">

在 body 中添加了这一句后，是不是已经看到了那个温柔的问句呢？如果是，那么你的 Browser 是支持这个 onUnload 事件的。下面来看这段代码：

function checkTime(){

var isModified=<c:out value="${editBean.edited }"/>;

if(isModified ==true)

alert("更新没有保存");

}

这个代码很有意思，其它的好说，<c:out value="${editBean.edited }"/>不好理解，这是 JSTL，如果没有见过 JSTL，google 或者 baidu 一下看看吧。editBean 是一个标准的 JSF 的 bean，它在 faces-config.xml 文件中部署，它有一个属性和属性的 getter,setter 方法，如下：

private boolean edited=false;

public boolean isEdited() {

return edited;

}

public void setEdited(boolean edited) {

this.edited = edited;

}

你在开始编辑的时候，设置 edited 为 true 就可以了。JSTL 能够用过<c:out>标签访问到这个 bean 的值。这样，就实现了前台 Javascript 和后台 JSF bean 的联动。

是不是看了这个文章，马上摩拳擦掌，准备去修改您的程序呢？

10、ArcGIS Server JAVA 讲座 --AJAX 基础

咱也来跟跟潮流，讲一讲 AJAX 和 JSF 如何结合。我相信，这也是将来 SERVER JAVA 的开发人员都会问的一个问题。我们来看看如何能够让 AJAX 能够在服务器上和客户端跑起来。

要应用 AJAX 的条件是：

1、 一个支持 javascript 的浏览器，当然，现在主流的浏览器都是支持 Javascript 的。

2、 浏览器必须支持 XMLHTTP 或者 XMLHttpRequest 对象。

3、 能够以 XML 发送响应的服务器端。服务器端可以有很多的技术来实现，不管你是用 ASP，JAVA，都可以实现。当然，本文要讨论的局限于 JSF 技术。

对于熟悉大多数服务器端程序编写的工程师，比如我，Javascript 完全是另外一个世界。但是了解 Javascript，将使你的程序更加强大，开发起来更加容易。一个很有意思的功能是如果用户在编辑地图的时候，编辑到中途，忘记保存了，这提醒我们可以做一个比较有意思的功能，自动保存。在 google mail 中，你是不是已经感觉到过这个自动保存的功能了呢？反正我当时看到的时候吓了一跳。以一个服务器端程序员的思路去思考，会觉得不可思议。事实上这个功能就可以用 Ajax 技术来实现，用一个定时器，每隔 5 分钟自动发送一个保存的请求就可以实现这样的功能。

再看一个自动查找的例子，比如一个输入一个用户后自动出来和该用户有关的信息，可以通过这样的方法来实现：

<input type="text" id="query" size="20" onkeyup="search(‘query’);">

Search(‘query’)方法可以发送一个 Ajax 请求到客户端，进行查询后，将结果以 XML 请求返回，这样就不需要刷新整个页面而进行部分更新。就这么一个小小的技术，让用户在继续 Web 浏览的时候感觉更好，也成为了各个网站争相模仿的对象，一些开发人员也发现了宝贝一样言必称 Ajax，甚至成为项目选型的关键，这大概也是最初的开发者想象不到的。

来看看 Javascript 如何来发送一个 XMLHttpRequest 请求的。来看一下下面的代码：

```
if (window.XMLHttpRequest) {
    req = new XMLHttpRequest();
}
else if (window.ActiveXObject) {
    req = new ActiveXObject("Microsoft.XMLHTTP");
}
```

这是再简单不过的实例化 XMLHttpRequest 的一段代码，实例化一个之后，你就可以像操作一个普通的对象一样操作它，它有很多的方法，比如下面的方法是建立和服务器端的链接：

```
req.open("GET", url, true);
```

第一个参数是 HTTP 请求的方式：”Get”或者”Post”，第二个参数是服务器的 URL，第三个参数是是否进行非同步的请求。大家如果已经仔细研究了我们 ESRI 的工具标签，你会发现这些标签里面有一个 clientPostBack 属性可以设置是否进行异步请求。我们可以知道，Web ADF 在后面肯定用了 open 方法的第三个参数来进行。要小心的是，如果你要设置 clientPostBack 为 true，你的 javascript 代码可能需要重新构建一下。不然这些代码可能不会被执行。因为异步请求更新的是部分页面。

异步请求的意思是在请求时浏览器可以做其它的事情，那么如何得知服务器端的处理已经结束，可以更新页面上的数据了呢？我们来看看 XMLHttpRequest 对象的方法

```
req.onreadystatechange = processXMLResponse;
```

当服务器端的处理结束时，就可以调用 processXMLResponse 方法，我们就可以通过对这个方法进行编程来处理请求，当然可以是任何的名字的方法，只要将方法注册到 onreadystatechange 方法即可。

XMLHttpRequest 初始化完成后就可以发送到请求到服务器端了，用一个很简单的方法即可：

```
req.send(null);
```

当然，一般 get 用的实在太少，我们最关心的是 Post，看看 Post 应该怎么做：

```
req.open("POST", "/agsviewer", true);

req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded";

req.send("__ADFPostBack__=true&formId=f"+"&mapId=Map0&ajaxdemo=ajaxdemo ");
```

Post 可以发送一些数据到服务器端，这正是我们所需要的。我们来继续看看一个请求当前比例尺的代码：

```
function updateInfoResponse(xmlHttp) {
   if (xmlHttp != null && xmlHttp.readyState == 4 && xmlHttp.status == 200) {
      var xml = xmlHttp.responseXML;
     document.getElementById("scale").value = "1:" +
xml.getElementsByTagName("scale").item(0).firstChild.nodeValue;
   }
}
```

看一下判断了(xmlHttp != null && xmlHttp.readyState == 4 && xmlHttp.status == 200，判断 4 是表示处理已经结束，判断 200 表示 HTTP 请求正常，相信大家都见过 404 的 HTTP 请求，令人讨厌的“page not found”。

判断结束后，就用 xml.getElementsByTagName("scale").item(0).firstChild.nodeValue;语句把比例尺取过来。

相信大家都大致知道 AJAX 是怎么回事了，下一次我会写一个简单的例子。

IBM 网站上有一篇很好的介绍 AJAX 的文章，大家可以去看看：

http://www-128.ibm.com/developerworks/cn/xml/wa-ajaxintro1.html

11、ArcGIS Server JAVA 开发讲座--- ADF 中的 Ajax 函数

ESRI 为了使 Server ADF 能够支持 Ajax，做了大量的工作，Sever ADF 中有很多可以利用的代码，我们来看看：

代码 1：创建 XMLHttpRequest，可以看一下下面的代码：

```
this.createXmlHttpObject = function() {
    if (this.isIE) {
      try {
        return new ActiveXObject("Msxml2.XMLHTTP");
      }
      catch (exception) {
        return new ActiveXObject("Microsoft.XMLHTTP");
      }
    }
    return new XMLHttpRequest();
  }
```

这个代码写得很妙吧，前面两个处理IE浏览器，最后一个支持Firefox，XMLHttpRequest是Firefox的。

发送请求，用如下命令：

```
EsriUtils.sendAjaxRequest(url, params, true, function() { updateInfoResponse(xmlHttp); });
```

解释一下，怎么得到这个 url，ESRI 也提供了办法：

```
var url = EsriUtils.getServerUrl(formId);
```

getServerUrl 的代码如下：

```
this.getServerUrl = function(fId) {
    if (document.forms[fId].action.indexOf(";jsessionid=") != -1)
      return document.forms[fId].action;
    else if (this.getJSessionId())
      return document.forms[fId].action + ";jsessionid=" + this.getJSessionId();
    else
      return document.forms[fId].action;
  }
```

嗯，url 已经解释了，我们看看第二个参数 params，这个参数包含了我们需要传递到服务器端的参数。看一下源代码就可以知道，xh.send(params);将这些参数发送到服务器端。

doGet 参数就比较简单了，用来说明这个请求是 get 还是 post。看看源代码，发送的方法是不一样的，如果是 get 方法，那些参数用?进行连接，传送到服务器端。它返回的仍然是 XMLHttpRequest

对象。

Callback 函数是我们自己写的函数，用来在服务器数据处理完成后，处理服务器端传送过来的数据。大家看着是不是一目了然了呢？

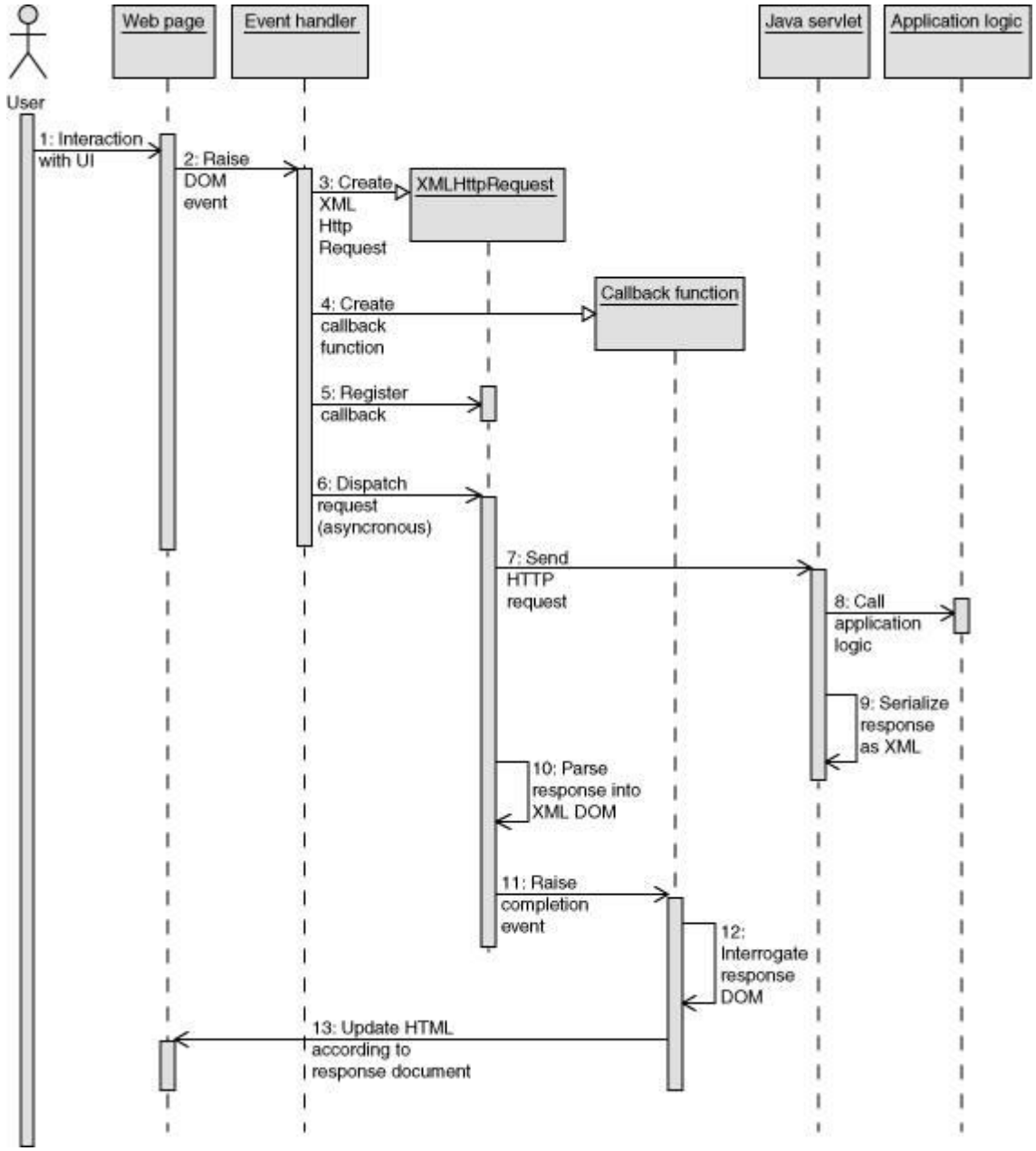
sendAjaxRequest 的源代码

```
this.sendAjaxRequest = function(url, params, doGet, callback) {
    try {
      var xh = this.createXmlHttpObject();
      xh.onreadystatechange = callback;
      if (doGet) {
        xh.open("GET", url + "?" + params, true);
        xh.send(null);
      }
      else {
        xh.open("POST", url, true);
        xh.send(params);
      }
      return xh;
    }
    catch (exception) {
      return null;
    }
  }
}
```

有了这几个强大的函数，我们就可以利用 ESRI 的成果，来进行 Ajax 的开发了。说了半天客户端的东西，下一篇我们来讲服务器端的处理，服务器端的处理需要比较大的篇幅来说。

12、ArcGIS Server Java 讲座--如何在用 JSF 在服务器端处理 Ajax 请求

服务器端的处理才是整个事件的 Key，那么我们来看看整个请求，回复的过程，我们通过下面的图看看我们服务器端的代码怎么样才可以 plug in 进去：



看了这个图您肯定已经明白了，我们有机会得到客户端发送过来的 HTTP 请求，得到这个请求，

得到请求里面的参数，我们就可以捋起袖子，大干一番了。
我们用什么方式去处理呢，最简单的一个方法是自己写一个 servlet，在 servlet 的 Dopost 方法里面，写上处理这个请求的代码，并且把结果以 XML 的方式返回回去，那么我们这里用一个简单的例子：

```
    throws java.io.IOException {
 String action = req.getParameter("action");
 String item = req.getParameter("item");

 if ((action != null)&&(item != null)) {

    // Add or remove items from the Cart
    if ("add".equals(action)) {
      cart.addItem(item);

    } else if ("remove".equals(action)) {
      cart.removeItems(item);

    }
 }

 // Serialize the Cart's state to XML
 String cartXml = cart.toXml();

 // Write XML to response.
 res.setContentType("application/xml");
 res.getWriter().write(cartXml);
}
```

这对于了解服务器端编程的人来说，是最简单不过的了。当然，我们用了 JSF 框架，肯定不会再通过 Servlet 技术去做这件事情。我们这里用 JSF 的 PhaseListener 进行处理。什么是 PhaseListener，我们来看一下 PhaseListener 的 JavaDoc：

extends java.util.EventListener, java.io.Serializable

An interface implemented by objects that wish to be notified at the beginning and ending of processing for each standard phase of the request processing lifecycle.

原来，这个接口捕捉到每个 Request 的声明周期，在请求开始处理和请求处理结束时我们可以插入我们自己的代码。这个接口的两个最重要的方法如下：
Handle a notification that the processing for a particular phase has just been completed. void beforePhase(javax.faces.event.PhaseEvent event)

Handle a notification that the processing for a particular phase of the request processing lifecycle is about to begin.

处理结束和处理开始两个方法，我们可以在这两个方法里面写入我们自己的代码，对用户过来的请求进行处理。

比如，我们可以在 afterPhase 方法里面，插入下面的代码对数据进行处理：

```
ExternalContext externalContext = facesContext.getExternalContext();
Map paramMap = externalContext.getRequestParameterMap();
```

通过 paramMap 里面可以得到所有的参数，当然，我们并不是对每一个请求都会进行处理，所以，在客户端 Javascript 发送 Ajax 请求得时候，可以添加一个自定义的参数，比如我们前面文档中的 ajaxdemo=ajaxdemo 参数，这样我们可以判断一下当前是不是我们要处理的请求：

```
if (! AJAX_DEMO.equals(paramMap.get(AJAX_DEMO)))
return;
```

接下去的代码，就可以处理我们的各种各样的请求了，我们在这里要举的例子是如何得到当前地图的比例尺。下次讲座再深入探讨吧。

13、ArcGIS Server Java 讲座----利用 Ajax 得到当前地图的比例尺（服务器端）

现在我们进入实战阶段，看看我们怎么可以在服务器端得到当前地图的范围，这个代码中你将学到如何从 faces Context 里面得到当前 form 中的控件，以及如何利用 ESRI 提高的工具，简化我们响应 AJAX 请求的过程：

```
UIComponent form = facesContext.getViewRoot().findComponent((String)
paramMap.get(FORMID));
```

首先我们得到页面上的 form，form 是一个 container，通过它我们可以得到 form 中的其它组件，比如我们的地图控件：

```
<a:map id="Map0" value="#{mapContext.webMap}" width="600" height="600"    />
```

我们可以用下面的方法得到这个 Mapcontrol 和 WebMap：

```
MapControl mc = (MapControl) form.findComponent((String) paramMap.get(MAPID));
WebMap wm = mc.getWebMap();
```

得到了这个 WebMap，哈哈，可以得到 Everything 了，因为你可以从 WebMap 中得到 WebContext。

这个时候，想得到那个比例尺，是再简单也不过的事情了，用下面的代码即可：

wm.getMapScale();

得到了比例尺了，接下去怎么办呢？把它返回到客户端，我们的 Ajax 客户端认的是 XML 的流，咋办呢？最笨的办法当然是是一个一个 String 的组装成 XML，然后调用 Response 的 Write 写回到客户端，比如下面的代码：

```
public String toXml() {
    StringBuffer xml = new StringBuffer();
    xml.append("<?xml version=\"1.0\"?>\n");
    xml.append("<scale>");
    xml.append("<value>100</value>");
   xml.append("</scale>\n");
    return xml.toString();
  }
```

生成了 XML 代码，然后用下面的代码把结果返回给客户端：

```
Response res=(Response)externalContext.getResponse();
res.setContentType("application/xml");
res.getWriter().write(scaleXml);
```

当然，各位看官想来都是 JAVA 高手，对于这样的写法实在感觉不舒服，正待想办法用标准的 XML 工具来写时，你会发现，ESRI 已经为您想到了，写好了几个工具类来供你调用了。我们来看看怎么用这些工具：

用下面的代码创建一个标准的 XML 文档：

```
Document doc = XMLUtil.newDocument();
```

用下面的代码新建一个 element：

```
Element responseElement = XMLUtil.createElement(doc, "response", null, null);
```

用下面的代码将我们的比例尺数据写到刚才新建的 Element 里面：

```
XMLUtil.createElement("scale", String.valueOf(wm.getMapScale()), responseElement);
```

然后用下面的语句将 XML 文档写入到客户端：

```
AJAXUtil.writeResponse(facesContext, doc);
```

很可惜的是，ESRI 这些类的代码是不公开的，不过你可以猜一猜，这些方法里面到底发生了什么事情。

OK，皮球又被提到了客户端了，我们下一个讲座来看看，如何解析服务器端返回的 XML 文档，并且更新页面上的输入框。

14、ArcGIS Server Java 讲座--用 Ajax 得到地图比例尺(客户端代码)

上个讲座我们讲到，服务器端已经得到了当前地图的比例尺，我们需要在客户端做一个处理，把这个比例尺数据给用户看。我们在页面上放一个文本框，用来显示比例尺数据，如下面的代码所示：

```
<tr>
  <td>Scale</td>
  <td><input type="text" id="scale" size="36" /></td>
</tr>
```

我们如何将服务器传过来的数据更新这个文本框呢？来看看前面讲座中提到的往服务器端发送请求时候的那个回调函数：

```
var xmlHttp = EsriUtils.sendAjaxRequest(url, params, true, function()
{ updateInfoResponse(xmlHttp); });
```

我们把回调函数的代码也贴在这里：

```
function updateInfoResponse(xmlHttp) {
  if (xmlHttp != null && xmlHttp.readyState == 4 && xmlHttp.status == 200) {
var xml = xmlHttp.responseXML;
    var scale=xml.getElementsByTagName("scale").item(0).firstChild.nodeValue;
    document.getElementById("scale").value = "1:" + scale
  }
}
```

这个代码相比大家都能看明白了，先判断服务器端是不是已经处理结束，并且服务器的状态一切正常，然后得到服务器返回的 xml，通过 getElementsByTagName 把里面包含着的比例尺数据提取出来，组合中我们熟悉的文本，然后把这个文本填充到 ID 为 scale 的文本框中。

我们来总结一下，我们发送了一个请求，这个请求包含了 MapID 参数，和一个标志参数 ajaxdemo，MapID 参数可以让服务器得到 MapControl 的实例和 WebMap，进而得到比例尺。Ajaxdemo 让服务器端的处理程序可以正确的辨识这个请求是我们这个 PhaseListener 需要处理的请求，处理结束后，返回 XML 到客户端。客户端的回调函数处理 XML 文档，提取出比例尺信息，写入到页面的文本框中。撩起 Ajax 神秘的面纱后，发现原来 Ajax 不过如此，老酒装新瓶，重新包装了一下。

讲到这里，大家心中想必还有两个疑点，第一个是说了半天，我们是怎么触发这个请求的呢？通过定时器触发么？不是。我们是通过地图的变化进行触发的，这样是最最合理的，每当地图更新了，客户端会自动发送一个请求，要求地图变化后的比例尺。非常合理，但是怎么进行设置呢？我们经常说，我们的控件是支持 Ajax 的，如何支持？我们支持通过监听器的方式，监听地图的变化，如下面的代码所示：

```
var map = EsriControls.maps["Map0"];
    map.addUpdateListener("request", updateInfoRequest);
```

这样，每次地图更新时，都会触发 updateInfoRequest 函数，该函数用来向服务器端发送请求。

另外一个疑问是 Phaselistener 怎么设置，看一下 JavaDoc 可以知道，我们可以在 faces-config.xml 文件中，用下面的语句进行设置：

```
<lifecycle>
      <phase-listener>demo.AjaxDemoPhaseListener</phase-listener>
  </lifecycle>
```

恩，已经完美了。

Ajax 的讲座到此告一段落了，欢迎大家讨论。

15、ArcGIS Server JAVA 讲座 自定义 Functionality

今天我们来讲讲如何自定义 Functionality，自定义 Functionality 有很多用处。它最大的用处是可以访问数据源。我们来看看已经存在的 Functionality：

AGSMapFunctionality, AIMSMapFunctionality, AWSMapFunctionality, EJBMapFunctionality, WMSMapFunctionality 等等等等，查看 GISFunctionality 接口的子类你可以看到 ADF 中更多的 Functionality。

这些 Functionality 无一不跟后台的资源有关。我们来看看 Functionality 接口的定义：

```
public void initFunctionality(GISResource arg0) {

 }
public void destroyFunctionality() {
}
public GISResource getResource() {
return null;
}
```

每个 functionality 都需要实现 Functionality 这个接口，这个接口里面有三个方法，初始化方法，destroy 方法和得到资源的方法。GISResource 对应于一个特定的资源，比如我们最常用的资源就是 AGSLocalMapResource，当然也可以是 ArcIMS 的 Resource，取决于你把这个 functionality 注册到哪个 Resource 中。我们来看一个例子，虽然 Functionality 一般跟资源有关，但是也可以做其它的事情，比如我们这里举一个非常简单的例子，FixedZoomin：

```
public class DemoFunction implements GISFunctionality {
WebContext ctx;
public void zoomInFunction(){
        WebMap wmap = ctx.getWebMap();
WebExtent ext = wmap.getCurrentExtent();
    ext.expand(0.5);
wmap.setCurrentExtent(ext);
ctx.refresh();
}
public void initFunctionality(GISResource arg0) {
ctx=arg0.getWebContext();
System.out.print("Demo Function inited");
}
public void destroyFunctionality() {
System.out.print("Demo Function destroyed");
}
public GISResource getResource() {
return null;
}
}
```

非常简单的一个例子，从 GISResource 里面得到 WebContext，然后得到 WebMap，修改 WebMap 的 Extent，放大功能就实现了。你说非得跟资源有关系，那么确实也有点关系，因为我们还是使用了后台的数据来进行操作。需要说明的是，我们这个代码对于所有的 Resource 都是适用的，不管你后台使用了 ArcIMS，还是其它的 Resource。ADF 的妙处也在这里。

写好了这个 Funcionality，那么我们来看看如何部署的，每一个 Functionality 都需要部署到某个 Resource 中，那么我来看一下，这个 functionality 如何来部署：

```
<managed-bean>
        <managed-bean-name>ags1</managed-bean-name>

<managed-bean-class>com.esri.adf.web.ags.data.AGSLocalMapResource</managed-bean-class>
<managed-bean-scope>none</managed-bean-scope>
<managed-property>
<property-name>functionalities</property-name>
<map-entries>
    <!--……略去其它的 functionality->
<map-entry>
        <key>demoFunction</key>
<value>#{demoFunction}</value>
    </map-entry>
</map-entries>
</managed-property>
</managed-bean>
```

部署完成了，如何使用呢？随便在你的一个 tools 里面，或者 task 的运行代码里面，添加下面的代码：

```
GISResource rec=(GISResource)evt.getWebContext().getResources().get("ags1");
DemoFunction df=(DemoFunction)rec.getFunctionality("demoFunction");
df.zoomInFunction();
```

就可以了。这只是一个简单的例子，而且是绕了一个远的例子。大家可以发掘一下 Functionlity 的使用方法。在下一次讲座中，我们将详细解释一个对 Resource 进行操作的例子。

16、ArcGIS Server Java 讲座---用自定义 functionality 实现用户权限控制

有了上此讲座的基础，我想理解我们这个功能应该就很容易了。刚刚我们经理问我有用户希望控制 web 登陆用户的权限，那么我们正好利用这个讲座来看看如何实现这个功能。我们新建一个类，functionality 当然要实现 GISFunctionality 接口了：

[Copy to clipboard] [-]

CODE:

```
<?xml:namespace prefix = o ns = "urn:schemas-microsoft-comfficeffice" />
public class SecurityCheckFunctionality implements GISFunctionality {
private AGSMapResource resource;
   public void initFunctionality(GISResource resource) { //empty }
public void destroyFunctionality() { //empty   }
   public GISResource getResource() { //empty   }
}
```

我们在这里放了一个 AGSMapResource 类型的变量，因为我们需要对 Resource 进行很多操作，我们来看一下：

```
public void initFunctionality(GISResource resource) {
this.resource = (AGSMapResource)resource; //得到资源
      //查看用户是否属于"petroEmployee" 角色
     if(!WebUtil.getExternalContext().isUserInRole("pretroEmployee")){
     //从 Resource 中得到 MapFunctionality
    AGSMapFunctionality mapFunc = (AGSMapFunctionality)resource.getFunctionality("map");
     //得到对 MapServerInfo 的引用
     MapServerInfo serverInfo = mapFunc.getMapServerInfo();
     //得到 "Pipeline_Network" 层
MapLayerInfo pipelineLayerInfo = AGSUtil.getLayerInfo("PipeLine_Network",layerInfos);
        if(pipelineLayerInfo==null)
          return; //层不存在，return.
          //从 TOC 中移除
          MapLayerInfo[] newLayerInfos =   removeLayer(pipelineLayerInfo,layerInfos);
          serverInfo.setMapLayerInfos(newLayerInfos);
          //从 MAP 中移除
          LayerDescription[] layerDescriptions
=serverInfo.getDefaultMapDescription().getLayerDescriptions();
```

```
            //新建一个层描述，替换原来的层描述
              LayerDescription[] newLayerDescriptions = new LayerDescription[newLayerInfos.length];
              for(int i=0;i<newLayerInfos.length;i++){

newLayerDescriptions=AGSUtil.getLayerDescription(newLayerInfos.getLayerID(),layerDescriptions);

serverInfo.getDefaultMapDescription().setLayerDescriptions(newLayerDescriptions);
            }
        }
```

上面的代码中用 removeLayer(pipelineLayerInfo,layerInfos);函数移除了要移除的层，然后更新了 TOC 和 MAP，我们来看看这个函数怎么来写：

```
        //Logic to remove a layer and all its descendants
        private MapLayerInfo[] removeLayer(MapLayerInfo unwantedLayerInfo, MapLayerInfo[]
oldLayerInfos) {
                //新建一个 MapLayerInfo
            MapLayerInfo[] newLayerInfos = new MapLayerInfo[oldLayerInfos.length-1];
            ArrayList descendantLayers = new ArrayList();
            for(int i=0,j=0;i<oldLayerInfos.length;i++){
              if(oldLayerInfos.getLayerID()!=unwantedLayerInfo.getLayerID())
                newLayerInfos[j++] = oldLayerInfos;
              if(oldLayerInfos.getParentLayerID()==unwantedLayerInfo.getLayerID())
                descendantLayers.add(oldLayerInfos);
        }
            for(int i=0;i<descendantLayers.size();i++){
      newLayerInfos = removeLayer((MapLayerInfo)descendantLayers.get(i),newLayerInfos);
            }
            return newLayerInfos;
        }
```

我们注意到这是一个递归函数，用于把该层下面所有的其它附属层都删除:

这个 functionality 写完了，我们来看看如何部署，部署的时候，先把它作为一个 Managed Bean

部署，用下面可代码即可，可以在 faces-config.xml 文件里面，也可以在 ags-functionality.xml 里面：

```
<managed-bean>
    <managed-bean-name>securitycheck</managed-bean-name>
     <managed-bean-class>com.mypackage.SecurityCheckFunctionality</managed-bean-class>
    <managed-bean-scope>none</managed-bean-scope>
</managed-bean>
```

将它作为一个 managedBean 部署后，下面就把它部署到 resource 中，注意部署的时候，要将其部署到 map 之后，toc 之前，因为 ADF 初始化是按顺序初始化，我们的权限控制需要从 Map 中读出层的信息，然后修改 MapDesc，从而影响 TOC 的信息，所以，进行如下的部署配置：

```
<managed-bean>
    <managed-bean-name>ags1</managed-bean-name>
    <managed-bean-class>
    com.esri.adf.web.ags.data.AGSLocalMapResource
  </managed-bean-class>
   <managed-bean-scope>none</managed-bean-scope>
  ...
  <managed-property>
    <property-name>functionalities</property-name>
    <map-entries>
      <map-entry>
<key>map</key>
<value>#{agsMap}</value>
      </map-entry>
      <map-entry>
<key>query</key>
<value>#{agsQuery}</value>
      </map-entry>
      <map-entry>
<key>tile</key>
```

<value>#{agsTile}</value>

</map-entry>

<map-entry>

<key>overview</key>

<value>#{agsOverview}</value>

</map-entry>

<map-entry>

<key>pipeline_security_check</key>

<value>#{securitycheck}</value>

</map-entry>

<map-entry>

<key>toc</key>

<value>#{agsToc}</value>

</map-entry>

</map-entries>

</managed-property>

</managed-bean>

OK，这个安全控制的 functionality 已经完成了。但是我们要对整个 WebApplication 加上安全控制，需要在 web.xml 做修改，添加如下的配置：

<login-config>

<auth-method>DIGEST</auth-method>

<realm-name>My_WebApplication</realm-name>

</login-config>

//声明两个组用户

<security-role>

<role-name>petroEmployee</role-name>

</security-role>

<security-role>

<role-name>stateEmployee</role-name>

</security-role>

//声明什么资源将收到安全控制，这里我们把整个目录都控制，当然你也可以控制胆敢目录

<security-constraint>

<web-resource-collection>

<url-pattern>*<url-pattern>

</web-resource-collection>

<auth-constraint>

<role-name>petroEmployee</role-name>

<role-name>stateEmployee<role-name>

</auth-constraint>

</security-constraint>

这两个组在什么地方定义呢？如果是 Tomcat，我们可以在 $TOMCAT_HOME/conf/tomcat-users.xml 中定义，定义如下：

<tomcat-users> ...

<user name="joe" password="joe" roles="petroEmployee">

<user name="daisy" password="daisy" roles="stateEmployee">

...

</tomcat-users>

这样我们就控制了整个 Web Application，并且对于不同的用户组，赋予了不同的层的查看权限。每次用户访问这个网站的时候，都会被提示要求输入用户名和密码，按照我们在 Tomcat-users 里面的定义进行输入即可。

还有用户会问道，如果我希望对属性字段的编辑进行控制怎么办；事实上原理是类似的，你让用户修改属性字段时，肯定要传输一个字段列表，你可以根据不同的用户组，对这个字段列表进行控制。

另外，这里我们用了 WEB 容器的用户和权限进行控制，事实上你不必依赖于 Web 容器，你可以在数据库中存放你自己定义的用户名和组，进行控制。

权限控制简单的讲就是这样子，.net 里面也类似，有机会的话，我会写一个.net 版本的给大家看看。

注：本讲座内容参考了并翻译了 ESRI 的 JAVA 帮助的部分内容。

17、Server Java 讲座-----扩展 Tasks 框架

Tasks 框架是一个很有意思的东西。如果你对 Tasks 框架还没有概念，你可以参考一下 web map application 那个 Sample，左边浮动的放大工具，查询工具都是用 Tasks 框架构成。有人会问，既然已经有了 command 和 tools，为什么还需要 tasks 呢？事实上 tasks 的目的是不同的；我们可以看一下 web map application，可以看到，相同类型的 button 和 toos 组合在一起，组成一个逻辑单元，这就是 tasks 的选择条件。

我们来看看如何编写 Tasks，编写 Tasks 是非常有意思的事情。Tasks 跟 tools 一样，也是一个普通的 Java 类即可，这个 Java 类可以从 Object 直接继承。请看下面的 Task 的例子：

```
1 public class JSTasks {
2 public void doSomething(){
3   System.out.println("Task button clicked.");
4
5 }
6 }
```

这个 tasks 实在是简单吧，看看怎么部署上去，我们在 xml 文件中做如下定义：

```
1  <managed-bean>
2     <managed-bean-name>jsTask</managed-bean-name>
3     <managed-bean-class>com.cj.ucdemo.JSTasks</managed-bean-class>
4     <managed-bean-scope>session</managed-bean-scope>
5  </managed-bean>
```

这个配置文件也非常简单，那么看来所有的诀窍在与如何在 JSP 页面里面使用它。我们来看看：

```
1  <a:task mapId="Map0" value="#{jsTask}"   rendered="true" />
```

添加一个 task 标签，把这个标签的 mapId 值指向地图控件”Map0”，讲它的值指向我们刚才定义的 Managed Bean。访问一下，大家看看下面的浮动窗口结果：

是不是很令人惊奇啊？事实上我们这个 Bean 中只有一个方法，但是 Tasks 框架聪明地把这个方法名给提取出来，作为 button 的名字。你点击一下这个按钮，在后台就可以看到执行了这个方法，打印出了一条信息。简单的说：方法就是按钮!!

有了上个 Task 的基础，我们来看其它的 Task 就驾轻就熟了。事实上，整个 Task 就是一个类，Task 的 Caption 就是这个类的类名。里面的每一个 Public 方法都是 Task

浮动条上的一个按钮。那么参数怎么办呢？我们来加一个参数，并且加上一个它的 get 方法和 set 方法：

```
1 String parameter;
public String getParameter() {
return parameter;
}
public void setParameter(String parameter) {
this.parameter = parameter;
}
```

看看我们的 task 工具条发生了什么变化，Task 工具条如下图所示：

可以看到，多了一个 Parameter 的输入框，前面还有一个 Label，很有意思吧。那个 DoAnything 是我加的另外一个方法。Task 框架的扩展已经很明显了吧：把相同的功能集合在一个类里面，这个类可以接受参数。例如我们可以利用这个 task 进行 buffer 操作，这个输入框就可以用来输入 buffer 的距离。

看到这里，大家肯定想到了另外的几个问题，比如：怎么把这些参数，按钮的名字修改成中文，上次给人培训，有人说：用中文的变量名？也是一个办法，但是考虑的 JDK 对中文变量的处理，似乎有问题。还有其它的问题包括怎么跟地图进行交互？怎么样设置客户端的动作（画多边形还是画 Polyline）。我们当然有另外的方法。

使用 Taksinfo 建立对 Task 的描述,所有的 Taskinfo 都需要继承自 SimpleTaskInfo 类。大家可以打开 doc 看看 SimpleTaskInfo 的子类有那些。事实上，那些子类都是已经存在的 task 的 taskinfo，如果你需要做汉化，继承这些类并且改写其中的几个方法即可。

我们来看看我们自己的类需要改写 SimpleTaskInfo 的哪几个方法：
TaskActionDescriptorModel[]getActionDescriptors();

此方法用来修改 command 按钮的描述；command 的意思是不需要和地图进行交互而是直

接在服务器端执行一个命令。

TaskParamDescriptorModel[]getParamDescriptors()

此方法修改参数的描述；

TaskToolDescriptorModel[]getToolDescriptors()；

此方法修改工具的描述，工具的意思是需要和地图进行交互；

那么我们来写一个类，直接说明如何使用这几个方法，修改 task 的表现：

```
public TaskDescriptor getTaskDescriptor() {
TaskDescriptor td=new TaskDescriptor(JSTasks.class);
td.setDisplayName("我的任务");
return td;
}
```

此方法修改了 task 的标题，将此类作为一个 Managed-bean 添加到 faces-config.xml 文件中，并且在.jsp 页面中修改 task 的使用如下：

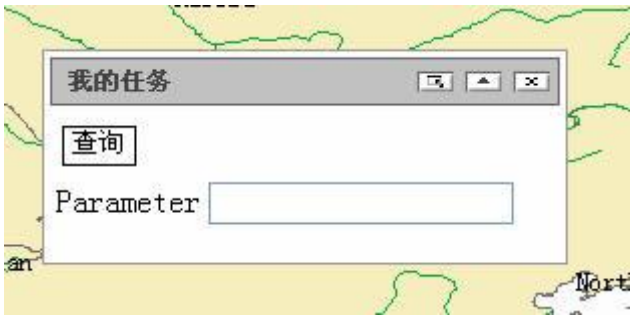
```
<a:task mapId="Map0"  value="#{jsTask}" rendered="true" taskInfo="#{jsTaskInfo}"/>
```

修改后的 Task 如下所示：

可以看到 task 的标题已经改变。下面我们来看修改 Action 的标题，代码如下：

```
public TaskActionDescriptorModel[]

getActionDescriptors() {

TaskActionDescriptorModel[] td=new TaskActionDescriptorModel[1];

TaskActionDescriptorModel actionDesc=new TaskActionDescriptor(JSTasks.class,"doSomething","查询");

td[0]=actionDesc;

return td;

}
```

需要说明的是 TaskActionDescriptor(JSTasks.class,"doSomething","查询")构造函数的三个参数，第一个是 task 类，第二个是方法名，第三个是修改后的方法名；构造后放到数组中返回即可，如果有多个方法，可以如法炮制，都放到数组中。修改后的 task 有如下表现：



你会发现另外一个 action 没有了，别着急，因为我们还没有把它放到数组中，而原来默认的显示方式已经被我们修改了。

我们来修改 doAnything 工具，doAnything 工具需要和地图交互，注意它的参数是 MapEvent 而不是 TaskEvent，它完整的代码如下所示：

```
    public void doAnything(MapEvent te){

//通过和地图交互进行放大

WebContext ctx=te.getWebContext();

WebGeometry geom=te.getWebGeometry().toMapGeometry(ctx.getWebMap());
```

```
WebExtent ext = (WebExtent)geom;

ctx.getWebMap().setCurrentExtent(ext);

ctx.refresh();

}
```

如果跟地图交互，就要涉及到客户端执行的操作，我们这里没有任何地方指定客户端的操作，这是在 taskinfo 里面指定的，我们来看下面的代码：

```
    public TaskToolDescriptorModel[]

getToolDescriptors() {

TaskToolDescriptor [] toolDesAry=new

TaskToolDescriptor[1];

TaskToolDescriptor toolDesc=new

TaskToolDescriptor(JSTasks.class, "doAnything", "交互放大",ClientActions.MAP_RECTANGLE);

toolDesAry[0]=toolDesc;

return toolDesAry;

}
```

注意 TaskToolDescriptor 构造函数的 4 个参数，第一个是 task 类，第二个是原方法名，第三个是替换后的方法名，第四个是客户端的动作。构造完成后放到数组中，如果有多

个 tools，如法炮制即可。修改后的 task 如下所示：

有了上面的解释，我们来看参数的 taskinfo 代码，就非常容易了：

```
public TaskParamDescriptorModel[] getParamDescriptors() {

// TODO Auto-generated method stub

TaskParamDescriptorModel[] td=new TaskParamDescriptorModel[1];

TaskParamDescriptorModel paraDesc=new TaskParamDescriptor(SearchTask.class,"searchText","查询文本");

td[0]=paraDesc;

return td;

}
```

修改后的 task 如下所示：

Task 框架的基本构成就是这样了，祝大家使用愉快。下节将介绍 TaskResult 的使用。

18、ArcGIS Server Java 讲座--自定义 Tools 开发

今天进行整理，发现自定义 tools 的教程忘记写了。今天把这个补全吧。

什么是 tools，tools 就是工具，它跟 command 的区别就是 tools 是要跟地图进行交互后再执行某个命令，而 command 是直接执行某个命令。最简单的 tools 和 command 的对比是放大是一个工具，而全图显示就是一个命令。

ESRI 已经包含的工具包括：放大（zoom in），缩小（zoom out），平移（Pan）。

前面我们都是先说服务器端的处理代码，现在先来看看客户端的处理。跟地图交互，就会产生怎么交互的问题。拉框放大的时候是在地图上画的是一个长方形，测距的时候在地图上画的是一条线。如何控制这些客户端的功能呢？

查看 com.esri.adf.web.faces.event.MapEvnet 的帮助，里面详细说明了如何把客户端的操作如何和服务器端的代码建立联系。已有的客户端支持的操作包括：

EsriMapCircle
EsriMapContinuousPan
EsriMapLine
EsriMapOval
EsriMapPan
EsriMapPoint
EsriMapPolygon
EsriMapPolyline
EsriMapRectangle

如果我们要自己写一个工具，我们可以新建一个普通的类，这个类可以实现 MapToolAction 接口，也可以不实现任何接口。我们先来看一个实现接口的类。

```
public class CJZoomInTool implements MapToolAction {

    public void execute(MapEvent event)  {

WebContext ctx = event.getWebContext();
  WebGeometry screenGeom = event.getWebGeometry();
  WebGeometry mapGeom = screen.toMapGeometry(ctx.getWebMap());
```

}

}

我们可以看到，通过 MapEvent 的 getWebGeometry 方法可以得到客户端所画的多边形，此时得到的是屏幕坐标，然后再通过 toMapGeometry 方法转换为地图坐标。
MapEvent 的 getWebGeometry 方法事实上是指向到 ClientActionArgs 的 getWebGeometry 方法，你可以用下面的代码得到另外一个 WebGeometry 进行比较：

ClientActionArgs clientarg=event.getClientActionArgs();
WebGeometry anotherGeom=clientarg.getWebGeometry();

我们还可以用 ClientActionArcgs 的 getRequestParameter 查看传送过来的参数是什么，在客户端指定长方形的情况下，得到的 RequestParamerter 里面有如下参数：

Map0_maxx:246
Map0_miny:147
Map0_minx:199
Map0_maxy:199

你一定想看看客户端是多边形的情况下，参数是什么情况,我们满足一下您的愿望：

Map0_coords:220:168|264:220|296:186|292:136|258:125

出现了以|分隔的参数集合，这个意思大家都明白了吧，这就是多边形的情况。

ClientActionArgs 是一个基类，它的所有的子类对应于每一个客户端的操作。每个积累需要实现它的抽象方法。最重要的两个方法是 init()和 getWebGeometry，init 从 request 里面得到所有的参数，getWebGeometry 从根据不同的参数，构建不同的 geometry 返回。ADF 会根据不同的客户端操作，创建不同的 ClientActionArgs 类。如果你一定要问我 ADF 是如何根据不同的客户端操作创建 ClientActionArgs 的，我们可以看到 ClientActionArgs 有一个静态方法：

getClientActionArgs(java.lang.String clientAction, java.util.Map requestParameters, java.lang.String controlId)

ADF 就是用这个静态方法，根据不同的 clientAction 可以得到的 ClientActionArgs。

这一下，把来龙去脉都讲了，既让大家看看整个客户端和服务器端操作的来龙去脉，也为大家增加自己的客户端操作热个身。

当然，自定义一个 tool 并不需要这么多的知识。事实上，自定义一个 tools 非常得简单，我们来看看帮助中的一段代码，计算选中了多少个要素。

```
package com.cj.ucdemo;

import java.rmi.RemoteException;

import com.esri.adf.web.ags.ADFAGSException;
import com.esri.adf.web.ags.data.AGSMapResource;
import com.esri.adf.web.data.WebContext;
import com.esri.adf.web.data.geometry.WebExtent;
import com.esri.adf.web.faces.event.MapEvent;
import com.esri.adf.web.faces.event.MapToolAction;
import com.esri.arcgisws.EnvelopeN;
import com.esri.arcgisws.EsriSearchOrder;
import com.esri.arcgisws.EsriSpatialRelEnum;
import com.esri.arcgisws.MapServerPort;
import com.esri.arcgisws.SpatialFilter;

public class CountFeatureTool implements MapToolAction {

    WebContext context=null;
    int countedFeatures;

    private void countFeatures(WebExtent extent){

        //Get the MapServerPort so we can execute methods through ArcGIS Server API

        AGSMapResource agsMap = ((AGSMapResource)context.getResources().get("ags1"));
        MapServerPort mapServer = agsMap.getMapServer();
        //Make a new envelope from the web extent
        EnvelopeN env = new EnvelopeN(extent.getMinX(), extent.getMinY(), extent.getMaxX(),
extent.getMaxY(),
           null, null, null, null, null);

        //Setup a spatial filter for an Intersection relationship
        SpatialFilter spatialFilter = new SpatialFilter();
        spatialFilter.setSpatialRel(EsriSpatialRelEnum.esriSpatialRelIntersects);
        spatialFilter.setWhereClause("");
        spatialFilter.setSearchOrder(EsriSearchOrder.esriSearchOrderSpatial);
        spatialFilter.setSpatialRelDescription("");
        spatialFilter.setGeometryFieldName("");
        //Set the envelope as the geometry
```

```
                spatialFilter.setFilterGeometry(env);

                //MapServer::queryFeatureCount() executes on the server and can throw a
RemoteException
                try{
                  //Count features in 4th layer which intersect with the envelope
                  int layerId = 1;
                  this.countedFeatures =
                    mapServer.queryFeatureCount(mapServer.getDefaultMapName(), layerId,
spatialFilter);

                  System.out.println("你选择了 "+countedFeatures+" 要素");

                }catch(RemoteException rme){
                  //Rethrow this as ADFAGSException so that it can participate in the exception
framework
                  throw new ADFAGSException("Could not execute
MapServer::queryFeatureCount()",rme);
                }
              }


        public void execute(MapEvent arg0) throws Exception {
              // TODO Auto-generated method stub
              try{
                    this.context=arg0.getWebContext();
              WebExtent ex=(WebExtent)arg0.getWebGeometry();
              ex=(WebExtent)ex.toMapGeometry(arg0.getWebContext().getWebMap());
              this.countFeatures(ex);
              }catch(Exception ex){
                    ex.printStackTrace();
              }

        }

}
```

大家注意这里使用了很多 com.esri.arcgisws package 里面的类，这个 package 是通过 axis 调用远程 webservice 的方式去调用服务器端的方法的。我们会在后续的讲座中说明。编译这个类后，直接在 jsp 页面里面用下面的代码调用即可：

```
      <a:tool id="countFeature" defaultImage="images/selection.gif" hoverImage="images/
```

selectionU.gif" selectedImage="images/ selectionD.gif"
clientAction="EsriMapRectangle" serverAction="com.cj.ucdemo.CountFeature
Tool" clientPostBack="true"/>

19、ArcGIS Server Java 讲座－空间查询和高亮显示的实现

ESRI 在 JAVA ADF 中做了很多的工作，其中很重要的一个部分是 WebQuery。WebQuery 用来对 ArcGISServer 和 ArcIMS 进行查询，然后在 ADF 这一端对地图进行渲染。比如说高亮显示，用图片进行渲染等等。

我们来看一个简单的选中，并把选中结果进行高亮显示的开发过程。

事实上选中可以用两种方法，第一种是空间选择，第二种是文本查询，这两种选中都可以。ESRI 也提供了进行这两种查询的两个类，这两个类都继承自同一个接口，我们来看一下：

这个类是 IdentifyCriteria, TextCriteria，接口是 QueryCriteria，QueryCriteria 可以设置一些比如最大返回记录，是否返回记录的详细信息等设置。如名所示，IdentifyCriteria 用来对数据进行空间查询，我们来看一下，如何使用这个类：

IdentifyCriteria ic = new IdentifyCriteria(geom);

geom 是一个 WebGeometry 类的实例，它可以从客户端触发的 MapEvent 类中得到。

创建了这个 IdentifyCriteria 之后，就可以进行查询了，查询是由 WebQuery 来完成的，所以先要得到 WebQuery：

WebQuery query = (WebQuery) context.getAttribute("query");

大家打开 faces-config.xml，mapContext 里面由没有 query，如果有，用上面的代码就可以得到 WebQuery。

得到了 WebQuery，就可以大干一场了，可以用下面的办法进行空间查询：

List results=query.query(ic,lyrList);

ic 就是 IdentifyCriteria，lyrList 是一个查询目标层的列表，目标层可以是一个，也可以是多个，比如可以用下面的方法得到所有可以查到的图层：

public ArrayList getQueryLayers(){

if(layers != null) return layers;

```
layers= new ArrayList();
WebQuery wQuery = this.context.getWebQuery();
WebLayerInfo layerInfo = null;
if(wQuery != null){
List layerList = wQuery.getQueryLayers();
for(Iterator iter = layerList.iterator(); iter.hasNext(); ) {
      Object item = (Object) iter.next();
       if(item instanceof WebLayerInfo){
            layerInfo = (WebLayerInfo)item;
           layers.add(new SelectItem(layerInfo, layerInfo.getName()));
}
        }
     }
return layers;
  }
```

当然，如果你不想查所有的图层，也可以让用户进行选择，选择哪个需要查询的图层。

现在来说说查询返回的结果，我们可以看到结果是一个 List，List 里面是什么？List 里面的对象是 QueryResult.

你可以循环得到里面的每一个 QueryResult，并把它们高亮显示：

```
for(int ii=0;ii<results.size();ii++){
      result = (QueryResult) results.get(ii);
    result.highlight();
}
```

Highlight 采用什么颜色，什么标志进行高亮显示，也是我们可以自己控制的，我个人的喜好是用红色，跟张艺谋一个水准，我们可以把显示符合加到 query 里面：

```
WebSimpleMarkerSymbol markers = query.getPointGraphicSymbol();
if (markers == null ) {
      markers = new WebSimpleMarkerSymbol();
```

```
    query.setPointGraphicSymbol(markers);
}
markers.setAntialiasing(true);
markers.setColor("255,0,0");
markers.setOutlineColor("255,0,0");
markers.setMarkerType(WebSimpleMarkerSymbol.CIRCLE);
```

同样，你可以设置线的颜色和多边形的颜色。这样就可以按照你需要的颜色进行高亮显示了。

当然 QueryResult 的功能不止能高亮显示，你还可以得到查询结果集的每一个字段的值，用 result.getDetails()方法即可得到一个 Map，里面有所有结果的字段名和字段值。

最后有说明的是，以上所有的代码对于 ArcGIS Server 和 ArcIMS 都同样适用，Enjoy。

20、Server Java 自定义开发－Network Analysis

一、目的

Server 9.2 下用户不用写任何代码，就可以用 Manager 或者 Eclipse 的工程模板创建很漂亮的 mapviewer 应用，但是没有可以即拿即用的 network Analysis 的模板。通过 Eclipse 的 Web Samples 中一个 Route Finder 的模板，这个模板创建的 network 应用需要有两个服务，一是带 network 图层的地图文档服务，另一个是 geocode 服务。为什么会同时需要两个服务呢？就是因为 Route Finder 是让用户在输入框中输入起点和终点的地址，然后通过 geocode 转换为地图上的空间点，然后才进行路径查询。

我们可能已经习惯了在 ArcMap 中的操作方式，在地图上添加 stop 和 barrier 点，然后进行路径生成。这种方式也更能为用户接受，毕竟可视化操作的用户体验更好，再说了，谁会愿意记一大堆地址呢？因此我们希望可以通过修改 Route Finder 模板的代码，来实现以下功能：

1、让用户通过点击工具按钮在地图上选择起点、终点以及路障，提交以后生成路径；这样就不需要 geocode 服务了。

2、用户选择的点能够以自定义的符号显示在地图上，达到交互的目的。

就是这两个简单的功能，下面就开始吧 javascript:window.open(this.src); alt=图片点击可在新窗口打开查看 src="/article/UploadPic/2007-6/2007626142947642.gif" onload=imgresize(this); border=0 smilieid="3">

二、数据的准备

这个比较简单，就用 ArcTutor\Network_Analyst\Exercise2\Paris.gdb 中的 streets 数据，因为我觉得 sfo 的图不好看。要先进行一些预处理，这不是本文重点，详细情况可以参考 Documentation 下的文档。

三、修改 faces-config.xml

1、在 Eclipse 下新建一个 ArcGIS Web Samples 工程，选择的模板是 ArcGIS Server Route Finder。

2、修改 faces-config.xml 文档，找到下面这段代码，删之，并且找到定义#{agswsgeo1}的那段<managed-bean>代码，一并删除。这两段代码在 WebContext 中注册一个 Geocode 服务，我们不需要。

```
    <value>#{agswsgeo1}</value>
  </map-entry>
```

3、找到下面这段代码，大家有兴趣可以观察一下它所在的位置，上下打望一番。把其中红色字体部分替换为 Map Server 的 URL 地址，其实就是我们发布的地图服务的地址，我的是 http://wangj:8399/arcgis/services/network/paris/MapServer。其实这个带中括号的部分应该在编译的时候就被替换掉的（有兴趣的可以去看看 sample 里面的那些 build.xml，其中主要功能就是用用户的输入参数替换这些方括号的内容），但实际上需要我们自己改，呵呵，自己动手，丰衣足食。

```
    <!-- AGSWS Map Server URL -->
    <value>[AGSWS_MAPSERVER_ENDPOINT]</value>
  </managed-property>
```

4、好了，最后修改一个地方，在 WebContext 中注册的两个 resources，其中一个是我们熟悉的 ags0，另一个是 agswsExt1，分别是地图服务和 NA 扩展。细心的人也许已经发现了，它们都注册了自己的 functionalities，但是用的是两套变量。ags0 用的是

```
    <value>#{agsMap}</value>
  </map-entry>
```

而 agswsExt1 用的是

```
    <key>map</key>

    <value>#{agswsMap}</value>

  </map-entry>
```

看到区别了吧，其实它们指向的是同一个变量，这个变量是在 WEB-INF/functionalities/ags-functionalities.xml 中定义的：

```
<managed-bean-class>
   com.esri.adf.web.ags.data.AGSMapFunctionality
</managed-bean-class>
<managed-bean-scope>none</managed-bean-scope>
</managed-bean>
```

看到了吧，好大一个坑，被我发现了 javascript:window.open(this.src); alt=图片点击可在新窗口打开查看 src="/article/UploadPic/2007-6/2007626142951155.gif" onload=imgresize(this); border=0 smilieid="9">。原来 ags0 引用错了，要把它改成与 ags-functionalities.xml 中定义的一样。当然，您也可以不修改，先运行看看会有什么样的 Exception 出现，很有意思。

```
selectedImage="images/tasks/routing/addstopD.gif" clientAction="EsriMapPoint"

serverAction="com.wj.adf.web.route.AddOriginPoint" clientPostBack="true"

toolTip="Add start point"/>
```

看这段代码大家应该就知道这个 tool 执行什么任务了 javascript:window.open(this.src); alt=图片点击可在新窗口打开查看 src="/article/UploadPic/2007-6/2007626142951237.gif" onload=imgresize(this); border=0 smilieid="1"> 不错，目的就是让用户选择这个工具在地图上进行选点，执行了 EsriMapPoint 行为。服务器端代码需要执行两个操作，一是保存这个点的坐标（它是一个 WebPoint 类型，下面就称为 originPoint）；二是在地图上用自定义的符号绘制这个点，why？因为默认情况下 EsriMapPoint 这个行为不会在地图上做任何标记，那用户怎么能知道起点是否选中了呢？！没关系，看我的。

先定义一个标记的样式：

```
//定义标记的样式

originMarker = new WebTrueTypeMarkerSymbol();
originMarker.setAntialiasing(true);
originMarker.setMarkerColor("72,145,91");
originMarker.setOutlineColor("5,5,5");
```

```
originMarker.setMarkerType(WebTrueTypeMarkerSymbol.CIRCLE);
originMarker.setGlowingColor("125,125,125");
```

下面揭开 com.wj.adf.web.route.AddOriginPoint 这个类的神秘面纱

```
private WebTrueTypeMarkerSymbol originMarker;
protected GraphicElement stopsElement;
 public AddOriginPoint(){
    //实例化 originMarker 和 stopsElement
   .....
}
public void execute(MapEvent event) throws Exception {
    WebContext ctx = event.getWebContext();



    WebPoint point = (WebPoint)event.getWebGeometry();
    //将屏幕坐标转换为地图坐标
    point = WebPoint.toMapPoint(point, ctx.getWebMap().getCurrentExtent(),
                    ctx.getWebMap().getWidth(), ctx.getWebMap().getHeight());
    WebGraphics graphics = ctx.getWebGraphics();
    if(graphics == null){
        System.out.println("WebGraphics is null");
        return;
    }
    //将这个点显示在地图上
    stopsElement.setGeometry(point);
    //给该点一个编号，因为是起点，所以编号始终为 1
    ArrayList pList = new ArrayList();
    pList.add("1");
    originMarker.setTextValues(pList);
    stopsElement.setSymbol(originMarker);
```

```
                graphics.addGraphics(stopsElement);
                 //保存 point，保存的策略大家可以自由选择，我是将 point 都存到了 WebSession 中
                .....
            }
        }
```

哈哈，这下大家看到 WebGraphics 的妙处了吧，贴个效果图

javascript:window.open(this.src); alt=图片点击可在新窗口打开查看 src="/article/UploadPic/2007-6/2007626142951817.gif" onload=imgresize(this); border=0>图片附件: [图标 1 就是用 WebGraphics 画的] 44.JPG (2006-12-20 03:57 PM, 14.15 K)

javascript:window.open(this.src); alt=图片点击可在新窗口打开查看 src="/article/UploadPic/2007-6/2007626142951712.jpg" onload=imgresize(this); border=0>

在取得 networkAnalystFunc 之后，随便写个类如 RouteFind,注册 faces-config.xml，

RouteFind 里面有这个方法 getSearched，放到 toolbar 里面，

```
public void getSearched(MapEvent event,boolean click){
                        if(!doubleClick){
                                pointStrart= (WebPoint) event.getWebGeometry().toMapGeometry(webMap);
                                networkAnalystFunc.setStopSymbol(originMarker);
                            networkAnalystFunc.addStop(pointStrart);
                        }else{
                                pointEnd = (WebPoint) event.getWebGeometry().toMapGeometry(webMap);
                                networkAnalystFunc.clearRoute();
                                networkAnalystFunc.setRouteSymbol(routeLine);
                                networkAnalystFunc.setStopSymbol(destinationMarker);
                            networkAnalystFunc.addStop(pointEnd);
                            results = networkAnalystFunc.solve();
                        }
                }
```

21、ArcGIS Server 开发——控制图层是否可见

当需要控制图层的可见性时，可以使用

```
IMapDescription mapdescription = webMap.MapDescription;
webMap.ManageLifetime(mapdescription);

            ILayerDescriptions layerdec = mapdescription.LayerDescriptions;

            for(int i=0;i < mapdescription.LayerDescriptions.Count; i++)
{
        ILayerDescription onelayerdesc = layerdec.get_Element(i);
         onelayerdesc.Visible = true;
}
```

22、航线查询问题－Server Java 的实现方法

数据准备

1、航线的相关数据，google 了一部分南航的数据，只要有几十条线路做演示就足够用了。将它们保存在数据库表中，主要有出发城市和到达城市两个字段。

2、用作地图显示的行政区划图以及城市的点要素图。为了简化，只选择了涉及上述航线的若干城市。城市数据是一个 point 的 shapefile 文件，其中有一个属性列为城市名称“name”，为了方便检索，name 属性中保存的城市名称与数据库中航线信息所保存的城市名称是匹配的。

实现方法

ADF 提供了很好用的 task framework，因此这个例子以自定义 task 的形式实现。先来简单了解一下整个操作的流程：

用户选择了自定义工具栏提供的一个工具，与地图进行一个点查询的交互，选择一个起点城市。在服务器端根据用户的操作将得到一个点的坐标，根据这个坐标进行查询，得到这个坐标点所表示的点要素的相关信息，这里我们所关心的只是这个城市的名称。知道了城市名称以后，以它作为起点城市，到数据库中检索所有从该城市出发的航线的目的地城市。这是一个字符串类型的数组，接下来可以根据这些城市名到地图中查找它们对应的点要素，并获得这些点要素的坐标。这样，我们就有了一个起点和若干个终点的坐标，可以绘制航线了。好了，基本思路就是这样，来看看怎么实现

先创建一个自定义 task 类，它有一个方法：

```
public class SearchAirlinesTask{
    public void getAirlines(MapEvent event){
    }
}
```

声明了这个类以后，task framework 就会根据 getAirlines(MapEvent event)方法的参数来判断这个工具的类型。由于 MapEvent 涉及的地图交互操作有很多种，而这里我们需要的是点查询操作，所以接下来我们需要再构造一个类来给自定义 task 加一些说明：

```
public class SearchAirlinesTaskInfo extends SimpleTaskInfo{
    private TaskToolDescriptor[] taskTool = new TaskToolDescriptor[1];
    public SearchAirlinesTaskInfo(){
        taskTool[0] =
            new TaskToolDescriptor(SearchAirlines.class,"getAirlines","选择起点
",ClientActions.MAP_POINT);
        taskTool[0].setToolTip(“从地图上选择一个起点”);
    }
    public TaskToolDescriptorModel[] getToolDescriptors(){
        return taskTool;
    }
}
```

然后将这个 taskInfo 类添加到我们的自定义 task 中：

```
public class SearchAirlinesTask{
    private SearchAirlinesTaskInfo taskInfo = new SearchAirlinesTaskInfo();
    public void getAirlines(MapEvent event){
    }
    public SimpleTaskInfo getTaskInfo(){
        return taskInfo;
    }
}
```

完成了上面的准备步骤以后，我们深入到 getAirlines(MapEvent event)方法的内部，看看它是怎么运行的：

```
public getAirlines(MapEvent event){
    WebContext ctx = event.getWebContext();
    WebGraphics graphics = ctx.getWebGraphics();
    WebQuery query = ctx.getWebQuery();
}
```

所有的查询操作都是通过 WebQuery 来进行的，查询的结果将通过 WebGraphics 绘制到地图上。查询的时候需要提供两个信息，点的坐标以及目标图层：

```
WebPoint point =
    (WebPoint)event.getWebGeometry().toMapGeometry(ctx.getWebMap());
IdentifyCriteria ic = new IdentifyCriteria(point);
ic.setMaxRecordCount(1);
//只需要查询 city 图层，该图层保存的是城市的相关信息
List layers = query.getQueryLayers();
List<WebLayerInfo> queryLayer = new ArrayList<WebLayerInfo>();
for(Iterator iter=layers.iterator(); iter.hasNext();){
    Object item = iter.next();
    if(item instanceof WebLayerInfo){
        WebLayerInfo layerinfo = (WebLayerInfo)item;
        if(layerinfo.getName().equals("city")){
            queryLayer.add(layerinfo);
        }
    }
}

List rs = query.query(ic, queryLayer);
```

ok，用户与地图交互操作的查询就完成了，接下来我们需要从查询结果中得到这个城市的城市名。

```
if(rs.size() > 0){
    Iterator iter = rs.iterator();
```

```
        QueryResult item = (QueryResult)iter.next();
        Object obj = item.getDetails().get("name");
        String startCity = obj.toString();
        WebPoint start = (WebPoint)item.getHighlightGeometry();
    }
```

这里需要解释一下的是最后一句，也许有人会问，刚才我们不是已经获取了一个 point 坐标了吗，那个不就是起点城市么？事实上，我们在地图上点击查询的时候不会恰好就点在表示城市的那个点的中心，WebQuery 查询的时候是有一个距离容错的。而绘制航线的时候要求更精确，所以我们以查询得到的点要素的坐标为准。

费了半天劲总算把城市名给找出来了，擦一把汗，接着查，我们还需要终点城市的坐标呢！刚才我们用的是 WebGeometry 来查询，接下来我们将使用文本进行查询。

```
    DbAirlineSearch dboper = new DbAirlineSearch();
    String[] destinations = dboper.getDestinations(startCity);
    //终点城市的坐标
    WebPoint[] ends = new WebPoint[destinations.length];
    //航线
    WebPath[] airlines = new WebPath[destinations.length];
    //在 WebGraphics 中绘制航线所需的要素
    GraphicElement[] linesElements =
            new GraphicElement[destinations.length];
    for(int i=0;i<destinations.length;i++){
        //设置文本查询的条件
        TextCriteria tc = new TextCriteria();
        List<String> searchFields = new ArrayList<String>();
        searchFields.add("name");
        tc.setSearchFields(searchFields);
        tc.setSearchText(destinations[i]);
        List rs2 = query.query(tc,queryLayer);
        //处理文本查询的结果
        if(rs2.size() > 0){
```

```
                QueryResult destination = (QueryResult)rs2.iterator.next();
                ends[i] = (WebPoint)destination.getHighlightGeometry();
                //得到一个终点坐标以后就可以绘制一条航线了
                List<WebPoint> pointList = new ArrayList<WebPoint>();
                pointList.add(start);
                pointList.add(ends[i]);
                airlines[i] = new WebPath(pointList);
                WebPolyline polyline = new WebPolyline();
                polyline.addPath(airlines[i]);
                linesElement[i] = new GraphicElement();
                linesElement[i].setGeometry(polyline);//这里需要说明一下，如果这里用的是 WebPath，将不会绘制出航线，换成 WebPolyline 以后就可以了。
                linesElement[i].setSymbol(lineSymbol);
                graphics.addGraphics(linesElement[i]);
            }
        }
```

到这里，航线的绘制功能就算完成了。后续的文章中将介绍如何将航线的其他信息添加到 WebResults 中，以及高亮显示某一条航线的功能

23、ArcGis 航线查询完全例子

主要功能

1、可以在地图上点击一个起点城市，自动画出从该城市出发的所有航线；也可以通过文本框输入起点城市进行查询，然后绘制航线；

2、将查询结果显示给用户。当用户选择一条具体的航线时，列出该航线的相关信息，包括起点城市、终点城市、出发时间、到达时间、票价等；

3、根据需要，用户可以高亮显示某条航线，另外还有 ZoomTo 等功能。

数据准备

1、航线的相关数据，google 了一部分南航的数据，虽然少，但是做 demo 也差不多了。其中票

价、起飞时间和到达时间是为了演示需要添加的三个字段。

	出发城市	到达城市	票价	起飞时间	到达时间
1	海口	北京	1201.00	8:35	14:35
2	海口	广州	1202.00	8:35	14:35
3	海口	深圳	1203.00	8:35	14:35
4	海口	南京	1204.00	8:35	14:35
5	海口	贵阳	1205.00	8:35	14:35
6	海口	成都	1206.00	8:35	14:35
7	海口	上海	1207.00	8:35	14:35
8	海口	郑州	1208.00	8:35	14:35
9	海口	长沙	1209.00	8:35	14:35
10	海口	西安	1210.00	8:35	14:35
11	海口	昆明	1211.00	8:35	14:35
12	海口	温州	1212.00	8:35	14:35
13	海口	大连	1213.00	8:35	14:35
14	海口	武汉	1214.00	8:35	14:35
15	海口	沈阳	1215.00	8:35	14:35
16	海口	哈尔滨	1216.00	8:35	14:35
17	海口	重庆	1217.00	8:35	14:35
18	海口	南昌	1218.00	8:35	14:35
19	海口	乌鲁木齐	1219.00	8:35	14:35
20	北京	重庆	1220.00	8:35	14:35
21	北京	成都	1221.00	8:35	14:35
22	北京	南昌	1222.00	8:35	14:35
23	北京	长沙	1223.00	8:35	14:35
24	北京	乌鲁木齐	1224.00	8:35	14:35
25	北京	三亚	1225.00	8:35	14:35
26	北京	延吉	1226.00	8:35	14:35
27	北京	昆明	1227.00	8:35	14:35

28 | 西安 | 榆林 | 1228.00 | 8:35 | 14:35

29 | 西安 | 银川 | 1229.00 | 8:35 | 14:35

30 | 西安 | 南京 | 1230.00 | 8:35 | 14:35

31 | 西安 | 长沙 | 1231.00 | 8:35 | 14:35

32 | 西安 | 海口 | 1232.00 | 8:35 | 14:35

33 | 西安 | 深圳 | 1233.00 | 8:35 | 14:35

34 | 西安 | 广州 | 1234.00 | 8:35 | 14:35

35 | 深圳 | 成都 | 1235.00 | 8:35 | 14:35

36 | 深圳 | 重庆 | 1236.00 | 8:35 | 14:35

37 | 深圳 | 武汉 | 1237.00 | 8:35 | 14:35

38 | 深圳 | 宜昌 | 1238.00 | 8:35 | 14:35

39 | 深圳 | 襄樊 | 1239.00 | 8:35 | 14:35

40 | 温州 | 南京 | 1240.00 | 8:35 | 14:35

41 | 温州 | 烟台 | 1241.00 | 8:35 | 14:35

42 | 温州 | 厦门 | 1242.00 | 8:35 | 14:35

43 | 温州 | 赣州 | 1243.00 | 8:35 | 14:35

把这些数据导入到数据库的 airlines 表中。

2、用作地图显示的行政区划图以及城市的点要素图。为了简化，只选择了涉及上述航线的若干城市。城市数据是一个 point 的 shapefile 文件，其中有一个属性列为城市名称“name”，为了方便检索，name 属性中保存的城市名称与数据库中航线信息所保存的城市名称是匹配的。

实现方法

ADF 提供了很好用的 task framework，因此这个例子以自定义 task 的形式实现。先来简单了解一下整个操作的流程：

用户选择了自定义工具栏提供的一个工具，与地图进行一个点查询的交互，选择一个起点城市。在服务器端根据用户的操作将得到一个点的坐标，根据这个坐标进行查询，得到这个坐标点所表示的点要素的相关信息，这里我们所关心的只是这个城市的名称。知道了城市名称以后，以它作为起点城市，到数据库中检索所有从该城市出发的航线的目的地城市。这是一个字符串类型的数组，接下来可以根据这些城市名到地图中查找它们对应的点要素，并获得这些点要素的坐标。这样，我们就有了一个起点和若干个终点的坐标，可以绘制航线了。好了，基本思路就是这样，来看看怎么实现 J

先创建一个自定义 task 类，它有一个方法：

```
public class SearchAirlinesTask{
public void getAirlines(MapEvent event){
}
}
```

声明了这个类以后，task framework 就会根据 getAirlines(MapEvent event)方法的参数来判断这个工具的类型。由于 MapEvent 涉及的地图交互操作有很多种，而这里我们需要的是点查询操作，所以接下来我们需要再构造一个类来给自定义 task 加一些说明：

```
public class SearchAirlinesTaskInfo extends SimpleTaskInfo{
private TaskToolDescriptor[] taskTool = new TaskToolDescriptor[1];
public SearchAirlinesTaskInfo(){
        taskTool[0] =
new TaskToolDescriptor(SearchAirlines.class,"getAirlines","选择起点",ClientActions.MAP_POINT);
        taskTool[0].setToolTip(“从地图上选择一个起点”);
}
public TaskToolDescriptorModel[] getToolDescriptors(){
        return taskTool;
}
}
```

然后将这个 taskInfo 类添加到我们的自定义 task 中：

```
public class SearchAirlinesTask{
private SearchAirlinesTaskInfo taskInfo = new SearchAirlinesTaskInfo();
public void getAirlines(MapEvent event){
}
public SimpleTaskInfo getTaskInfo(){
      return taskInfo;
}
}
```

完成了上面的准备步骤以后，我们深入到 getAirlines(MapEvent event)方法的内部，看看它是怎么运行的：

```
public getAirlines(MapEvent event){
```

```
WebContext ctx = event.getWebContext();
WebGraphics graphics = ctx.getWebGraphics();
WebQuery query = ctx.getWebQuery();
}
```

所有的查询操作都是通过 WebQuery 来进行的，查询的结果将通过 WebGraphics 绘制到地图上。查询的时候需要提供两个信息，点的坐标以及目标图层：

```
WebPoint point =
(WebPoint)event.getWebGeometry().toMapGeometry(ctx.getWebMap());
IdentifyCriteria ic = new IdentifyCriteria(point);
ic.setMaxRecordCount(1);
//只需要查询 city 图层，该图层保存的是城市的相关信息
List layers = query.getQueryLayers();
List<WebLayerInfo> queryLayer = new ArrayList<WebLayerInfo>();
for(Iterator iter=layers.iterator(); iter.hasNext();){
Object item = iter.next();
if(item instanceof WebLayerInfo){
        WebLayerInfo layerinfo = (WebLayerInfo)item;
        if(layerinfo.getName().equals(“city”)){
              queryLayer.add(layerinfo);
        }
}
}
List rs = query.query(ic, queryLayer);
```

ok，用户与地图交互操作的查询就完成了，接下来我们需要从查询结果中得到这个城市的城市名。

```
if(rs.size() > 0){
Iterator iter = rs.iterator();
QueryResult item = (QueryResult)iter.next();
Object obj = item.getDetails().get(“name”);
```

```
String startCity = obj.toString();
WebPoint start = (WebPoint)item.getHighlightGeometry();
}
```

这里需要解释一下的是最后一句，也许有人会问，刚才我们不是已经获取了一个 point 坐标了吗，那个不就是起点城市么？事实上，我们在地图上点击查询的时候不会恰好就点在表示城市的那个点的中心，WebQuery 查询的时候是有一个距离容错的。而绘制航线的时候要求更精确，所以我们以查询得到的点要素的坐标为准。

费了半天劲总算把城市名给找出来了，擦一把汗，接着查，我们还需要终点城市的坐标呢！刚才我们用的是 WebGeometry 来查询，接下来我们将使用文本进行查询。

```
DbAirlineSearch dboper = new DbAirlineSearch();
String[] destinations = dboper.getDestinations(startCity);
//终点城市的坐标
WebPoint[] ends = new WebPoint[destinations.length];
//航线
WebPath[] airlines = new WebPath[destinations.length];
//在 WebGraphics 中绘制航线所需的要素
GraphicElement[] linesElements =
new GraphicElement[destinations.length];
for(int i=0;i<destinations.length;i++){
//设置文本查询的条件
TextCriteria tc = new TextCriteria();
List<String> searchFields = new ArrayList<String>();
searchFields.add(“name”);
tc.setSearchFields(searchFields);
tc.setSearchText(destinations[i]);
List rs2 = query.query(tc,queryLayer);
//处理文本查询的结果
if(rs2.size() > 0){
    QueryResult destination = (QueryResult)rs2.iterator.next();
    ends[i] = (WebPoint)destination.getHighlightGeometry();
```

```
        //得到一个终点坐标以后就可以绘制一条航线了
        List<WebPoint> pointList = new ArrayList<WebPoint>();
        pointList.add(start);
        pointList.add(ends[i]);
        airlines[i] = new WebPath(pointList);
        WebPolyline polyline = new WebPolyline();
        polyline.addPath(airlines[i]);
        linesElement[i] = new GraphicElement();
        linesElement[i].setGeometry(polyline);//这里需要说明一下，如果这里用的是 WebPath，将不会绘制出航线，换成 WebPolyline 以后就可以了。
        linesElement[i].setSymbol(lineSymbol);
        graphics.addGraphics(linesElement[i]);
    }
    }
```

好了，getAirlines(MapEvent event)方法大致就是这样，完成以后就可以在地图上进行交互查询了，选择某个城市以后就可以自动绘制出该城市出发的航线。

但是玩了几次以后，发现这个 demo 还可以有些扩展。比如可以将查询的结果保存在 WebResults 中，可以让用户点击某个航线记录，然后显示出该航线的相关信息，像起飞时间、到达时间、票价等等；另外，用户可能还有兴趣高亮显示某条记录，甚至用用 ZoomTo 功能啊。

仔细考虑一下，其实刚才我们已经做了很多准备工作了呀。首先，已经查询得到了起点和终点城市，根据这两个城市的名称我们就可以从数据库中查询到这条航线的其他相关信息；其次，绘制航线时创建了 GraphicElement 来表示线要素，这个线要素中包含了 WebPolyline 的信息，可以用来进行高亮和 ZoomTo 等操作。哈哈，现在我们来充分利用这些已有数据。

首先要创建一个辅助类，用来记录每次查询操作的结果：

```
public class SearchAirlinesTaskResult{
private static WebSimpleLineSymbol highlightSymbol;
private WebContext ctx;
private GraphicElement originElement; //航线的原始符号
private WebPolyline polyline;
private GraphicElement highlightElement; //高亮线的符号
private String startCity;
```

```
private String endCity;
private Map<String,String> details; //航线的详细信息
static{
        highlightSymbol = new WebSimpleLineSymbol();
        highlightSymbol.setWidth(2);
        highlightSymbol.setColor("234,244,34");
        highlightSymbol.setLineType(WebSimpleLineSymbol.SOLID);
}
public SearchAirlinesTaskResult(WebContext context, GraphicElement originElement){
      this.ctx = context;
      this.originElement = originElement;
      this.polyline = (WebPolyline)originElement.getGeometry();
}
public Map getDetails(){
if(details == null){
        DbAirlineSearch dboper = new DbAirlineSearch();
        details = dboper.getDetails(startCity, endCity);
 }
return details;
}
//startCity 和 endCity 的 getter 与 setter 方法
……
public String getDescription(){
return startCity + " 至 " + endCity;
}
//highlight 的实现方法就是从 WebGraphics 中删除原有的线型符号，用新的高亮线型符号重绘一次
public void highlight(){
        WebGraphics graphics = ctx.getWebGraphics();
```

```
        graphics.removeGraphics(originElement);
        highlightElement = new GraphicElement();
        highlightElement.setGeometry(polyline);
        highlightElement.setSymbol(highlightSymbol);
        graphics.addGraphics(highlightElement);
        ctx.refresh();
}
public void clearHighlight(){
        WebGraphics graphics = ctx.getWebGraphics();
        if(highlightElement != null){
              graphics.removeGraphics(highlightElement);
        }
        graphics.addGraphics(originElement);
        ctx.refresh();
}
//根据 polyline 获取其外包矩形，然后刷新地图
public void zoomTo(){
        WebExtent ext = getExtent(polyline);
        ctx.getWebMap().setCurrentExtent(ext);
        ctx.refresh();
}
}
```

有了这个辅助类以后，只需要在前面的 getAirlines(MapEvent event)方法中添加以下代码：

```
……
ArrayList<SearchAirlinesTaskResult> results =
new ArrayList<SearchAirlinesTaskResult>();
for(int i=0;i<destinations.length;i++){
//设置文本查询的条件
TextCriteria tc = new TextCriteria();
```

```
List<String> searchFields = new ArrayList<String>();
searchFields.add("name");
tc.setSearchFields(searchFields);
tc.setSearchText(destinations[i]);
List rs2 = query.query(tc,queryLayer);
//处理文本查询的结果
if(rs2.size() > 0){
        QueryResult destination = (QueryResult)rs2.iterator.next();
        ends[i] = (WebPoint)destination.getHighlightGeometry();
        //得到一个终点坐标以后就可以绘制一条航线了
        List<WebPoint> pointList = new ArrayList<WebPoint>();
        pointList.add(start);
        pointList.add(ends[i]);
       airlines[i] = new WebPath(pointList);
        WebPolyline polyline = new WebPolyline();
        polyline.addPath(airlines[i]);
        linesElement[i] = new GraphicElement();
        linesElement[i].setGeometry(polyline);//这里需要说明一下，如果这里用的是 WebPath，将不会绘制出航线，换成 WebPolyline 以后就可以了。
        linesElement[i].setSymbol(lineSymbol);
        graphics.addGraphics(linesElement[i]);
         //将查询结果添加到 QueryResult 中
        SearchAirlinesTaskResult searchRs =
                                new SearchAirlinesTaskResult(ctx,linesElement[i]);
              searchRs.setFromcity(fromcity);
              searchRs.setDestination(destinations[i]);
        results.add(searchRs);
}
}
```

```
HashMap<String,String> actions = new HashMap<String,String>();
actions.put("高亮显示", "highlight");
actions.put("取消高亮", "clearHighlight");
actions.put("ZoomTo", "zoomTo");
    ctx.getWebResults().addResultsWithActionMap("航线查询结果", results,
  "getDescription", "getDetails", actions);
ctx.refresh();
```

最终的效果图如下：

24、ArcGis Server 中 如何在 Task 中实现 下拉列表和 checkbox 和 radiobutton.（完整例子）

如何在 Task 中实现 下拉列表和 checkbox 和 radiobutton？

checkbox 目前还不支持。要实现下拉表和 radiobutton，需要预先设置一个 Map 类型的选择项，此外还需要为 task 设置一个 taskInfo，比如在自定义的 task 中有这样一个参数：

```
----------CustomTask.java----------
private String param;
public void setParam(String param){
this.param = param;
}
public void getParam(){
return this.param;
}
        //设置下拉框选项
public Map getParams(){
java.util.Map result = new java.util.LinkedHashMap();
result.put("key1","value1");
result.put("key2","value2");
return result;
```

}

----------CustomTask.java----------

然后在 taskInfo 中为其设置选择方法：

----------CustomTaskInfo.java----------

TaskParamDescriptor[] taskParams = new TaskParamDescriptor[1];

taskParams[0] = new TaskParamDescriptor(CustomTask.class,"param","参数","getParams",true); //最后一个参数如果设为 true，就是 radioButton，如果设为 false，就是下拉框

----------CustomTaskInfo.java----------

25、ArcGIS Server 开发——标注

[C#写法]

private void LabelField(IFeatureLayer pFeatureLayer,IServerContext pServerContext)

{

IGeoFeatureLayer pGeoFeatureLayer = pFeatureLayer as IGeoFeatureLayer;

pGeoFeatureLayer.AnnotationProperties.Clear();

IAnnotateLayerPropertiesCollection pAnnoLayerPropsColl = pGeoFeatureLayer.AnnotationProperties;

ILabelEngineLayerProperties pLabelEngine;

pLabelEngine = pServerContext.CreateObject("esriCarto.LabelEngineLayerProperties") as ILabelEngineLayerProperties;

pLabelEngine.Expression = "[Field]";

IAnnotateLayerProperties pAnnoLayerProps = pLabelEngine as IAnnotateLayerProperties;

pAnnoLayerPropsColl.Add(pAnnoLayerProps);

pGeoFeatureLayer.DisplayAnnotation = true;

26、ArcGIS Server Java 开发--Born for SOA 系列 前言

在这一章里面，我们将用大量的篇幅，阐述 ArcGIS Server 对 SOA 的支持，我们在这里

所指的 SOA，是基于 Web Serivces 的 SOA，其他架构的 SOA 不在我们这个系列的讨论之列。本章内容包括 Web Service 基础；如何在 J2EE 下环境下构建和部署 Web Service；如何使用 ArcGIS Server 构建 Web Service；如何在 SOA 系统中利用这些 Web Service 等。最后会以一个实际的例子为例，如何在利用 ArcGIS Server 编写 Web Service，并且部署到 SOA 环境中，解决实际中的问题。本章所有的例子都是基于 JDK 1.5,AXIS2,ArcGIS Server 9.2 JAVA ADF 开发。本章也是 CJ 编写的 ESRI 内部教材 Server Java 开发教程的第十五章。

27、ArcGIS Server Java 开发--Born for SOA 系列 Web Service 基础

Web Service 是描述一些操作（利用标准化的 XML 消息传递机制可以通过网络访问这些操作）的接口。Web 服务是用标准的、规范的 XML 概念描述的，称为 Web 服务的服务描述。这一描述囊括了与服务交互需要的全部细节，包括消息格式（详细描述操作）、传输协议和位置。该接口隐藏了实现服务的细节，允许独立于实现服务基于的硬件或软件平台和编写服务所用的编程语言使用服务。这允许并支持基于 Web 服务的应用程序成为松散耦合、面向组件和跨技术实现。Web 服务执行一项特定的任务或一组任务。Web 服务可以单独或同其它 Web 服务一起用于实现复杂的功能或商业交易。

听起来很复杂，但是我们用一个简单的例子加以说明。这个例子跟 GIS 无关，但是可以帮助大家了解在 J2EE 环境中，整个 Web Service 的编写，部署过程。并且如何在.Net 环境和 Java 环境中调用这个 Web Service。

以下的需求来自于一个我原来导师的项目，这虽然不是一个典型的 WebService 充满前景的一个例子，但是确实用 web service 来解决问题的一个好的开始。项目的背景是要帮用户部署多个网站，为了节约成本，两个网站都是使用租用的空间，结果有一个租用的 Linux 操作系统的空间没有安装 XWindows，导致一个使用 JAVA 写的图像缩放程序无法执行，当然，没有 Xwindows，基本上需要访问 AWT 包的程序都会无法执行。怎么解决呢，刚好另外一台服务器上有 Xwindows，可以使用这个功能，所以我们可以在另外一台服务器上编写一个 WebService，发送一个图像 Resize 请求，请求参数是被需要被 Resize 的图片的 URL，以及 Resize 的比例，而返回的结果是 Resize 后的图像。压缩图像的代码如下所示：

```
import java.awt.image.BufferedImage;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.URL;

import javax.imageio.ImageIO;

import com.sun.image.codec.jpeg.JPEGCodec;

import com.sun.image.codec.jpeg.JPEGImageEncoder;

/**
```

```
*
* @author CJ
*
*/
public class CompressLocal {
publicCompressLocal() {
super();
}
/**
*
* @param filename  要被缩放的文件名，如果是 URL，请设置 isURL 为 true
* @param newname  缩放后的新的文件名
* @param std_img_w  目标文件的宽度
* @param std_img_h  目标文件的高度
* @param stretch  是否拉伸，如果 1，则按照参数比例，如果是 2，则按照宽度的比例；如果 3，则取小的那个比例尺
* @param isURL  是否是 URL
*/
public void doScaleOnImage(String filename,String newname,int std_img_w,intstd_img_h,int stretch,boolean isURL)
{
BufferedImageoldimage=null;
try{
intnew_w=-1;int new_h=-1;
if(isURL){
InputStreamin=(new URL(filename)).openStream();
oldimage= ImageIO.read(in);
in.close();
}else{
```

```
oldimage = javax.imageio.ImageIO.read(newjava.io.File(filename));
}
intold_w = oldimage.getWidth(null); //原图像的宽度
intold_h = oldimage.getHeight(null);//原图像的高度
if(stretch==1){
new_w=std_img_w;
new_h=std_img_h;
}elseif(stretch==2){
if((old_w > std_img_w) || (old_h > std_img_h)) {
doubletagSize = (double)old_w / std_img_w; //  按宽度比例确定缩放比例
new_w= (int)(old_w / tagSize); //得到一个新的比例
new_h= (int)(old_h / tagSize);
}
}
else if(stretch==3){
new_w= old_w;
new_h = old_h;
if((old_w > std_img_w) || (old_h > std_img_h)) { //如果原图像高度和宽度都大于目标高度和宽度
floattmpdouble = std_img_w;
floattagSize = old_w / tmpdouble; //  按宽度比例确定缩放比例
tmpdouble= std_img_h;
if(tagSize < (old_h / tmpdouble)) //按高度确定缩放比例
tagSize= old_h / tmpdouble;
new_w= Math.round(old_w / tagSize); //得到一个新的比例
new_h= Math.round(old_h / tagSize);
}
}
BufferedImagetag = new BufferedImage(new_w,new_h,BufferedImage.TYPE_INT_RGB);
tag.getGraphics().drawImage(oldimage,0,0,new_w,new_h,null);
```

```
oldimage.flush();
FileOutputStreamnewfile = new FileOutputStream(newname);
JPEGImageEncoderencoder = JPEGCodec.createJPEGEncoder(newfile);
encoder.encode(tag);
tag.flush();
newfile.close();
}catch (Exception e) {
e.printStackTrace();
}
}
/**
* @param args
*/
public static void main(String[] args) {
CompressLocalcl = new CompressLocal();
cl.doScaleOnImage("D:\\temp\\photo\\0016.jpg","c:\\temp\\cj.jpg", 535, 312,2,false);
}
}
```

缩放图片的代码已经完成，下面来写一个方法，如何调用图片缩放代码：

```
package edu.zju;
/**
* 作为一个 application scope 的 web service，供大家使用
*
* @author CJ
*
*/
public class CompressWebService {
publicCompressWebService() {
super();
}
```

```
/**
*
* @param url  源 jpg 文件的 URL
* @param width  目标 jpg 文件的宽度
* @param length  目标 jpg 文件的高度
* @return  返回目标 jpg 文件的 url
*/
publicString doCompress(String url,int width,int height){
StringrltURL="c:/temp/aa.jpg"; //例子程序使用，正式代码根据 url 生成文件名
CompressLocalcomp=new CompressLocal();
comp.doScaleOnImage(url,rltURL,width,height,1,true);
returnrltURL;
}
/**
* @param args
*/
public static void main(String[] args) {
}
}
```

代码编写完毕后，可以直接部署了，由于有了 AXIS，部署变得非常的简单，我们使用下面的 service.xml 进行部署，具体的部署方法可以参考 AXIS 的部署文档：

```
<description>
ScaleJPEG Service
</description>
<messageReceivers>
<messageReceiver
mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
<messageReceiver
```

mep="http://www.w3.org/2004/08/wsdl/in-out"

class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>

</messageReceivers>

<parameter name="ServiceClass">

edu.zju.CompressWebService

</parameter>

</service>

部署完成后，可以使用下面的 URL 对 WebService 进行访问。

http://chunjie:8088/axis2/services/CompressService/doCompress?url=http://chunjie:8088/jpg/10/deans/0037.JPG&width=55&height=55

返回结果为

<ns:return>c:/temp/aa.jpg</ns:return>

</ns:doCompressResponse>

如果访问 http://chunjie:8088/axis2/services/CompressService?wsdl，可以得到描述当前 WebService 的 wsdl 文件。第一个 WebService 就部署完成了，是不是简单的不可思议？当然，得益于 axis 的伟大工作，隐藏了大量的细节，从而使工作变得如此简单。

刚才我们直接通过 URL 已经可以访问 webservice 了。但是我们一般在 javacode 里面访问，访问后还需要做一些其他的操作，比如马上把图片文件保存到服务器上的某个目录中。如果需要在 Javacode 中访问该 web service，我们需要根据 wsdl 文件生成一个本地的 Java 代理类，通过这个代理类用来访问 WebService 的方法。Axis 提供了 wsdl2java 的工具可以根据 wsdl 的描述生成 java 代理类，可以使用命令行的工具，或者使用 Axis 为 Eclipse 提供的工具生成代理类。生成代理后，就可以用下面的方法调用该 WebService：

DoCompress dc = newDoCompress();

dc.setUrl("http://chunjie:8088/jspsamples/jpg/thanks.jpg");

dc.setHeight(100);

dc.setWidth(100);

CompressWebServiceCompressWebServiceSOAP11PortStubstub = new CompressWebServiceCompressWebServiceSOAP11PortStub(

null,"http://chunjie:8088/axis2/services/CompressService");

DoCompressResponsedcr=stub.doCompress(dc);

System.out.println(dcr.get_return());

} catch (AxisFault e) {

```
e.printStackTrace();
} catch (RemoteException e) {
e.printStackTrace();
}
```

当然，我们也可以用.Net 来调用这个 WebService，这正是 Web Service 的妙处所在，在 VisualStudio 里面新建一个项目，然后添加 Web 引用，在引用地址里面填入 service 的 URL，入下图所示：

添加引用后，就可以用下面的代码来进行访问：

```
dc.url = "http://chunjie:8088/jspsamples/jpg/thanks.jpg";
dc.height = 160;
dc.width = 120;
consumeWS.chunjie.CompressServicecs = new consumeWS.chunjie.CompressService();
consumeWS.chunjie.doCompressResponsedr= cs.doCompress(dc);
```

现在我们已经知道如何编写，部署，在 JAVA 和.NET 中调用 WebService。在下一节中，我们来看看可以通过怎样的方式对 ArcGIS Server 提供的 WebService 的进行访问。

28、ArcGIS Server Java 讲座:Born For SOA--Server 对于 SOAP 的支持

ArcGIS Server 的服务可以通过 Local 或者 Internet 两种方式进行连接，Local 方式直接连接到 SOM 上，本地的对象如何和 SOM 进行交互呢？事实上是通过 AO 进行交互的，所以你必须在本地有 AO 对象才可以进行连接，我们平常使用 ArcCatlog 就可以这样连接，或者通过 Engine 也可以进行这样的连接。而 Internet 方式直接连接到 Web Service 的引用地址，它是通过本地对象连接的，对于 Java ADF 而言，本地对象表示连接 ArcGIS Server 的类存在于本地 JRE 中。Local 连接可以改变 Server Object 的状态，而 Internet 连接由于只是通过 SOAP API 进行访问，所有的交互都是无状态的。

对于 SOAP 的支持是在 AO 这一级，所以处理每一次 SOAP 请求，都会需要 AO 的介入。有两个接口特别要注意一下，一个是 IServiceCatalogAdmin，一个是 IRequestHandler，IRequestHandler 我们已经在上面一节提到过，而 IserviceCatalogAdmin 是用来得到 Server Object 所提供服务的 WSDL 的。

SOM 用来向客户端提供 WSDL，所有支持的服务类型的 WSDL 保存在 ArcGIS Server 的 XMLSchema 目录下。

我们可以使用 IServiceCatalogAdmin 接口得到所有服务类型的 WSDL，如下面的代码所示：

```
private void getWSDL(String domain,String username,String password,String myserver){
ServerInitializer initializer = new ServerInitializer();
```

```
initializer.initializeServer(domain,username, password);
ServerConnection gisconnection;
try {
        gisconnection = new ServerConnection();
        gisconnection.connect(myserver);
// Get reference to ServerObjectManager class.
IServerObjectManager som =gisconnection.getServerObjectManager();
IServerContext serverContext = som.createServerContext("usa", "MapServer");
IServiceCatalogAdmin2 isc =
(IServiceCatalogAdmin2)serverContext.createObject(ServiceCatalog.getClsid());
//          Catalog WSDL
byte[] bitscatalog = isc.getCatalogDescriptionDocument("Catalog","http://localhost");
String catalog_wsdl = new String(bitscatalog,"UTF8");
System.out.println(catalog_wsdl);
//          Service WSDL
byte[] bitsservice =isc.getDescriptionDocument("usa", "MapServer","http://localhost");
String service_wsdl = new String(bitsservice, "UTF8");
System.out.println(service_wsdl);
} catch (UnknownHostException
e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}
```

事实上，ArcGISServer 每一个 ServerObject 都实现了 IRequestHandler 接口，都可以响应 SOAP 的请求，可以使用下面的方法得到 ServerObject 的 IRequestHandler，并且得到当前地图的 Default Map Name.

```
private void getMapName(IServerContext serverContext){
                IRequestHandler irh;
```

```
                try {
                        irh =(IRequestHandler)serverContext.getServerObject();
                       String soap_request ="<?xml version='1.0' encoding='utf-8' ?>";
                      soap_request +="<soap:Envelope
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:tns='http://www.esri.com/schemas/ArcGIS/9.2'>";
                        soap_request +="<soap:Body>";
                      soap_request +="<tns:GetDefaultMapName>";
                      soap_request +="</tns:GetDefaultMapName>";
                      soap_request +="</soap:Body>";
                      soap_request +="</soap:Envelope>";
                    String soap_response =irh.handleStringRequest("map,query,data",
soap_request);
                } catch (AutomationException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
```

所以，ArcGIS Server 处理 SOAP 请求的能力是与生俱来的，从 Server Object 诞生那一天起，它就能处理 SOAP 请求。从而使通过 Web Service 的方式调用 GIS 功能变得异常简单。当然，我们一般都不会想 Server Object 直接发送请求，根据 WSDL 的描述，通过发送和接受 SOAP 请求来访问 ArcGIS Server 工作量十分巨大，那么我们来看看如何在 ADF 中使用 Web Service。

29、Server Java 开发--Born for SOA 系列 通过代理类访问 ArcGIS Server Web Service

我们现在知道，ArcGIS Server 的处理 SOAP 请求的能力在 Server Object 级别就已经实现了，所以我们可以不通过 Web Service 的方式，而通过直接发请求到 Server Object 的方式实现开发，只是这样太过麻烦。

好在 ArcGISServer 在发布每一个 Service 的时候，也同时发布了一个 Web Service，使我们可以通过 Web Services 的方式来直接访问 Service。图 16.1 就演示了一个 Map Service 的 WebService 入口：

web_service_url.jpg (54.67 KB)

2007-5-11 09:26

这是 ArcGISServer 提供的 Web Service 的引用地址，在第一节里面我们已经知道，我们可以通过在 JAVA 中创建代理类和.NET 中添加 Web 引用的方式访问这个 Web 服务。当然，最后对于 SOAP 的请求的处理还是通过 Server Object 来执行的。

在使用 ESRI 提供的类包之前，我们不妨自己来试试，通过特定的工具来根据 WSDL 生成本地代理类。AXIS2 提供的 WSDL2JAVA 是一个很好的工具，生成的 JAVA 类的代码多达 7 万行。里面包含了所有需要访问这个 Web Service 所需要的一切类和方法。这些方法包括 Identify，Find，QueryFeatureCount 等，这些方法的参数和返回值，也作为内部类包含在该方法所在类中。需要特别指出的是：在访问 ArcGIS Server 的 Web Service 的时候，在本地 JRE 环境中，只需要有 AXIS 的类库和标准的 JRE 库即可，不需要任何其他的 AO 类或者代理类，这极大地提高了部署的灵活性。那么我们使用代理类来调用该 Web Services 的一个 QueryFeatureCount 的一个功能：

```
packageedu.zju.esrisvs;

importjava.rmi.RemoteException;

import org.apache.axis2.AxisFault;

import edu.zju.esrisvs.World_MapServerStub.QueryFeatureCount;

import edu.zju.esrisvs. World _MapServerStub.QueryFeatureCountResponse;

import edu.zju.esrisvs. World _MapServerStub.QueryFilter;

public classWebSvsTest {

public WebSvsTest() {

super();
```

```
}

/**

         * @param args

         */

        public static void main(String[] args){

try {

World _MapServerStubstub=new World _MapServerStub();

QueryFilter qf=newQueryFilter();

QueryFeatureCountqfc=new QueryFeatureCount();

qfc.setLayerID(0);

qfc.setMapName("Layers");

qf.setWhereClause("1=1");

qfc.setQueryFilter(qf);

QueryFeatureCountResponseqfr=stub.QueryFeatureCount(qfc);

System.out.println(qfr.getResult());

} catch (AxisFault e) {

e.printStackTrace();

} catch (RemoteException e) {

e.printStackTrace();
```

```
}

}

}
```

这就是使用代理类来访问该 Web Service 的过程，用工具生成的代理类非常长，大概有几万行代码，我就不贴在这里了，大家可以用工具生成。大家可能注意到，很多类的名字如 QueryFitler，和方法，如 QueryFeatureCount，都跟 ADF 里面的一样，事实上 ADF 里面访问 Web Service 的包同样也是用 AXIS 生成的。你也可以在.NET 里面，通过添加 Web 引用的方式来访问该 ServerObject 的 Web Service。