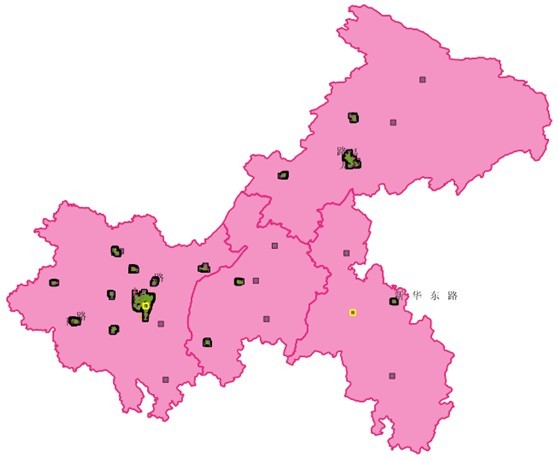
[**GeoServer地图开发解决方案（一）：环境搭建篇**](http://www.cnblogs.com/beniao/archive/2011/01/08/1930822.html)

　　GeoServer 是 OpenGIS Web 服务器规范的 J2EE 实现的社区开源项目，利用 GeoServer 可以方便的发布地图数据，允许用户对特征数据进行更新、删除、插入操作，通过 GeoServer 可以比较容易的在用户之间迅速共享空间地理信息。本系列博文提供全面、完善的GeoServer部署解决方案，包括GeoServer环境搭建、地图数据处理、部署地图数据、发布地图服务等功能的详细介绍。文中内容来自本人工作中通过网络学习后总结而成，如有类同纯属巧合，同时欢迎广大网友前来交流。



　　搭建GeoServer环境需要安装GeoServer、Java JDK 1.5/1.6、Tomcat 5.0/6.0（老版本的GeoServer需要Tomcat的支持，最新版本的GeoServer 2.0.2版已经内置了HTTP服务器）。

**一、下载、安装Tomcat、Java JDK1.5/1.6**

　　Tomcat和Java jdk都可以通过其官方网站下载到，详细请查看本文末的资源表。如本人电脑中就是将JDK安装在D盘下的，详细如下图：



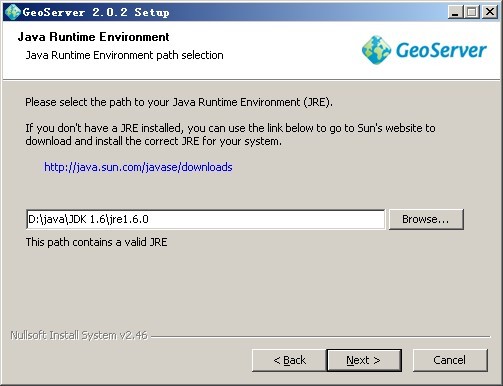
　　如果是使用最新版本的GeoServer 2.0.2版本的就不需要安装Tomcat，GeoServer 2.0.2已经内置了HTTP服务器。

二**、下载、安装GeoServer**

　　GeoServer 是 OpenGIS Web 服务器规范的 J2EE 实现，利用 GeoServer 可以方便的发布地图数据，允许用户对特征数据进行更新、删除、插入操作，通过 GeoServer 可以比较容易的在用户之间迅速共享空间地理信息。GeoServer是社区开源项目，可以直接通过社区网站下载，详细请查看本文档末的资源表。

　　GeoServer兼容 WMS 和 WFS 特性；支持 PostGIS 、 Shapefile 、 ArcSDE 、 Oracle 、 VPF 、 MySQL 、 MapInfo ；支持上百种投影；能够将网络地图输出为 jpeg 、 gif 、 png 、 SVG 、 KML 等格式；能够运行在任何基于 J2EE/Servlet 容器之上；嵌入 MapBuilder 支持 AJAX 的地图客户端OpenLayers；除此之外还包括许多其他的特性。关于GeoServer更多信息可以访问百度百科了解：<http://baike.baidu.com/view/871332.htm> 或官网了解：<http://geoserver.org/display/GEOS/Welcome> 。

　　安装GeoServer非常简单，由于GeoServer是Java编写的开源项目，故GeoServer的运行需要有Java JDK的支持，也就是上面一步所安装Java JDK，在安装GeoServer的同时就必须提供Java JDK的jre才能完成GeoServer的安装。

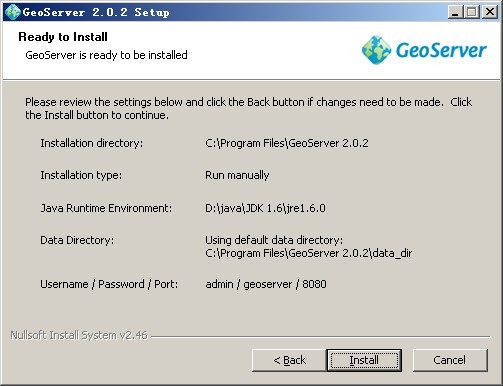


　　使用GeoServer来部署发布shp格式地图数据为WMS服务，需要将shp 格式地图文件复制到GeoServer指定的地图数据目录下面，在最新版的GeoServer安装中就可以指定地图数据的存放目录，其默认放置在GeoServer的安装目录下。



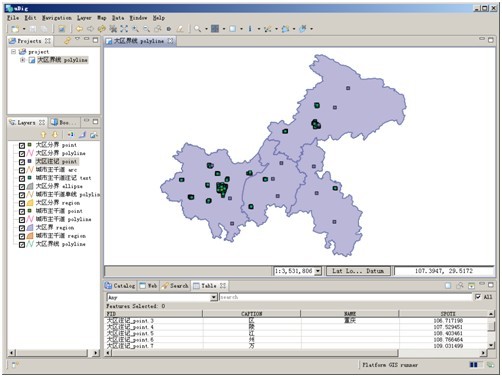
　　GeoServer安装的时候提供默认的用户名(admini)和密码(geoserver)，以及GeoServer管理系统的访问端口号(8080)。





**三、下载、安装uDig**

　　uDig是一款开源桌面GIS软件，基于Java和Eclipse平台，可以进行shp格式地图文件的编辑和查看。其下载地址：<http://udig.refractions.net/files/downloads/udig-1.2.0.exe>



到此为止，基于GeoServer的地图部署环境基本搭建完成，下一篇我将详细介绍如何基于uDig进行地图数据查看、编辑以及地图样式导出等功能。

# [GeoServer地图开发解决方案（二）：地图数据处理篇](http://www.cnblogs.com/beniao/archive/2011/01/09/1930932.html)

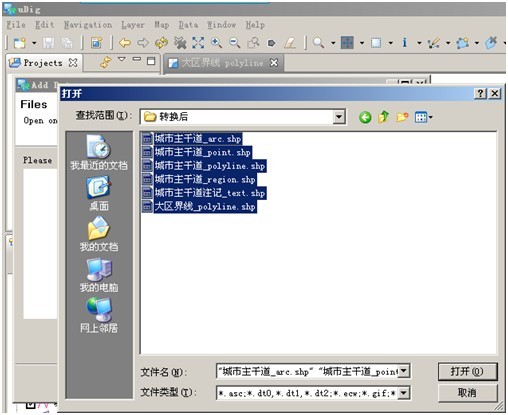
　　由于GeoServer只支持发布ArcGIS格式的地图数据，既.shp格式的地图数据。如果是别的厂商提供的地图数据则需要进行一次地图数据格式转化。本篇主要介绍如何查看、编辑地图数据的应用。

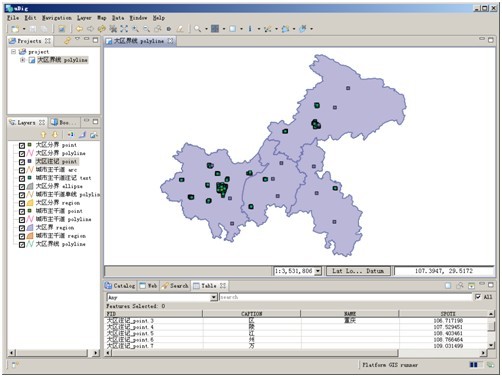
一、**地图数据转为shp数据格式**

　　由于GeoServer只支持发布ArcGIS格式的地图数据，既.shp格式的地图数据。在使用GeoServer部署地图数据之前则需要针对不同地图厂商提供的地图数据进行数据格式转化。比如说将MapInfo地图数据转为ArcGIS的shp格式地图数据，需要使用MapInfo提供的工具进行装换。操作非常简单，这里不做详细介绍。

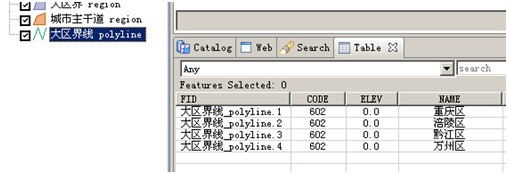
**二、使用uDig编辑SHP格式数据**

　　首先使用uDig打开shp格式地图文件，支持同时打开多个shp格式地图数据文件。打开单个地图数据文件则以地图数据文件的地图风格呈现出来，如果是打开多个地图数据文件则是以按照地理坐标确定图层位置的图层重叠呈现。





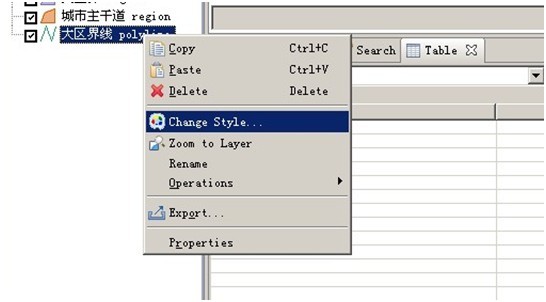
　　在打开的shp格式地图数据文件后，通过uDig的图层面板就可以查看当所选择的shp地图数据文件中所包含的地图图层信息，当选中某个图层后就可以通过Table面板查看到该图层所对应的一些详细数据信息。如下图所示：



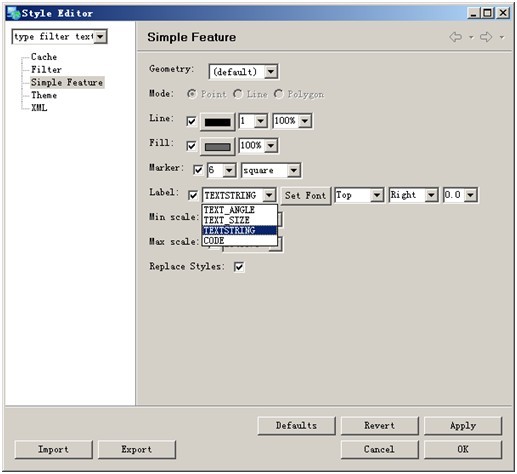
　　通过uDig工具可以对地图数据进行查看，编辑，是地图数据部署中对地图数据进行纠错处理不可缺少的一个工具，很给力的一个工具。

**三、从uDig导出SHP格式数据样式**

　　uDig提供了非常强大的地图数据编辑功能，实际上应用最多的就是使用uDig编辑shp格式的地图数据，从中提起样式文件。目的其实很简单就是为了修改编码，通常默认的文字编码发布出的WMS服务呈现出来的地图标签中午是乱码。通过图层的“Change Style”功能选项就可以进入图层的样式编辑器。



　　通过可视化编辑器将修改后的样式导出为样式文件(.sld)，在发布地图数据的时候就可以使用编辑后的样式到对应的地图图层，从而解决中文标签乱码问题。





　　到此为止，基于uDig进行地图数据编辑处理介绍完毕，当然本文只是简单的介绍，希望可以通过本文抛砖引玉，期待关于uDig的更多精彩文章出现。下一篇我将介绍如何使用GeoServer进行地图数据部署相关的知识点。

# [GeoServer地图开发解决方案（三）：部署地图数据篇](http://www.cnblogs.com/beniao/archive/2011/01/10/1930995.html)

　　本篇文章正式介绍基于GeoServer的地图数据部署实现，前提条件为成功搭建有GeoServer环境。实际上基于GeoServer部署shp格式的地图数据非常简单，对于GeoServer对于的磁盘物理层主要就是一个地图数据目录的概念，对于GeoServer应用系统来说则有三大重要知识点，分别为：工作空间（workspace）、存储器（store）和地图图层（layer），这些概念的详细都将在本文中逐一的介绍。

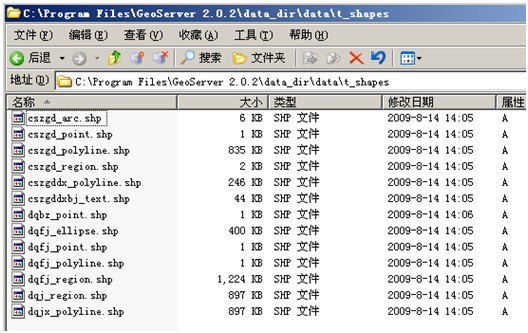
**一、GeoServer地图数据目录（Data Directory）**

　　所谓地图数据目录既地图数据的存放目录，在《[GeoServer地图数据部署解决方案（一）：环境搭建篇](http://www.cnblogs.com/beniao/archive/2011/01/08/1930822.html)》中介绍环境搭建的时候就提到过地图数据目录这个概念，安装GeoServer的时候就已经指定了地图数据目录的位置。

　　部署地图数据非常简单，首先需要将地图数据文件(shp地图数据文件)复制到GeoServer的数据目录（安装GeoServer的时候所指定的地图数据目录）下面，因为只有将地图数据放到此目录下，GeoServer后台才能发现shp的地图数据文件（建议使用英文命名地图数据文件）。



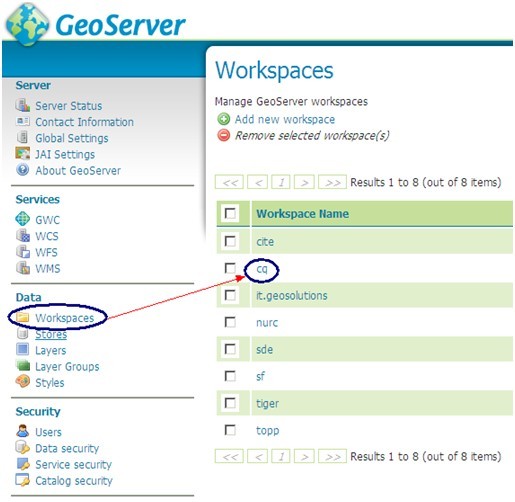
　　需要部署的地图数据必须放置在上面GeoServer提供的地图数据目录之下，如下图示就是将待部署发布的shp格式地图数据放在在GeoServer地图数据目录下名为“t\_shapes”的目录中的。这里的“t\_shapes”需要特别注意，在使用GeoServer进行地图数据部署发布的时候（创建存储器）需要使用到此名字（t\_shapes）。



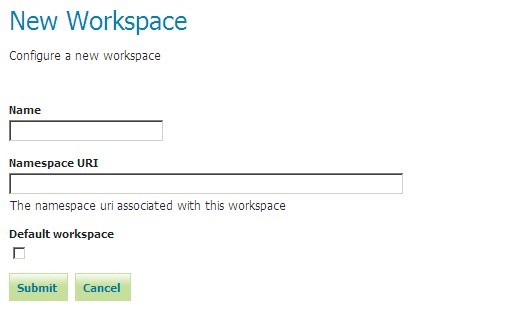
　　最新版(版本号：2.0.2)的GeoServer全面改善了shp格式地图数据的发布，并增加了OpenLayers方式的地图数据发布预览功能，以及将地图数据导出KML或GML等格式数据。下面将详细介绍这些功能点。

**二、工作空间（workspace）**

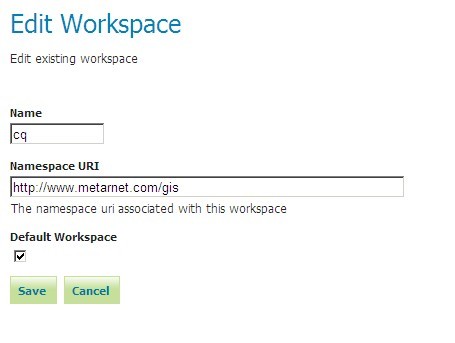
　　最新版的GeoServer里面修改了老版本的“目录”为“工作空间”，工作空间存放着多个数据存储器。成功登陆GeoServer管理系统后就可以从左边的功能导航处看到“工作空间”选项。工作空间管理平台详细如下图所示：



　　要发布地图数据为WMS服务，首先得建立工作空间（也可以使用现有的工作空间），然后建立数据存储器，最后在存储器里面发布地图数据。要想创建工作空间可以直接通过管理平台界面的“Add new workspace”进入工作空间创建界面，如下图所示：



　　编辑工作空间和创建工作空间一样，非常的简单，可以直接从工作空间管理列表界面进入工作空间编辑界面。

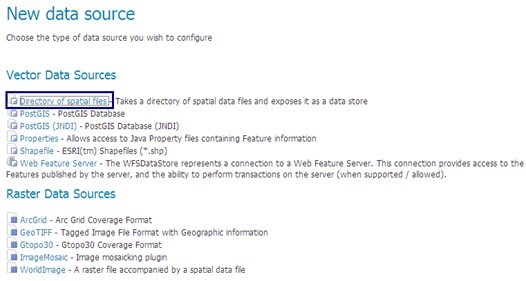


**三、存储器（store）**

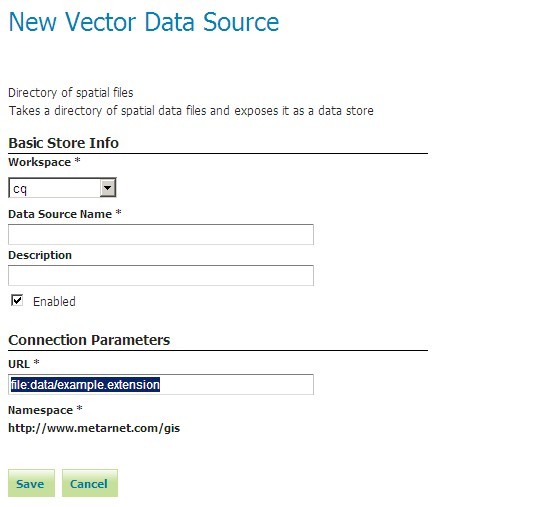
 　　基于工作空间只上的存储器，维护着和地图数据目录的映射关系。可以直接通过GeoServer左边的功能导航进入存储器管理界面，如下图所示：



　　如上图所示，在存储器管理界面中点击“Add new Store”就可以导航到创建存储器界面，可以完成存储器到地图数据目录的映射，为后面发布地图图层做准备。如下图所示：

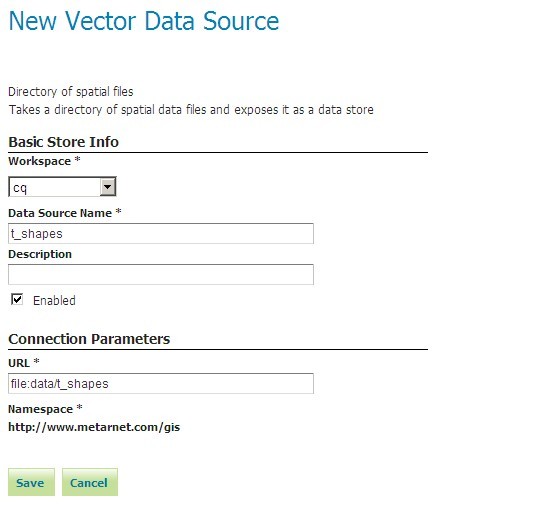


　　通常我们使用的是“Directory of Spatial files”，也就是以目录为单位进行shp格式地图数据的部署。下图为添加存储器的界面：



　　按照GeoServer的约定一个数据存储器可以部署一个独立的shp格式地图数据文件，也可以部署一个目录的shp格式地图数据格式。

　　需要特别注意就是“URL”，这里指向的就是存放需要部署的地图数据的目录，此地址的格式为：“file:” + “GeoServer的地图数据目录”，比如说地图数据放在C:\Program Files\GeoServer 2.0.2\data\_dir\data\t\_shapes 下，那么在建立存储器的时候的URL应该为file:data/t\_shapes 。最终的配置如下图所示：



**四、地图图层（layer）**

　　地图图层主要就是管理部署在GeoServer里面的地图数据中的图层元素，通过地图图层管理列表可以非常清楚的看到地图图层的类型、所属工作空间、所属存储器、图层名称以及采用的SRS标准等相关信息。

　　通过图层管理界面的“Add a new resource”可以进入图层类型(工作空间：存储器)选择界面，详细如下图：



　　地图数据存储器管理维护着与地图数据目录对应的地图图层数据，选择了对应的图层存储器类型，就会列表处该图层类型下面的所有图层元素，如下列表所示：



　　使用图层列表管理维护着地图数据图层的发布，图上图所示成功发布地图数据图层为地图服务的前面有“V”标记，如果要修改已发布的地图图层可以使用再次发布功能完成。

　　本篇就介绍到这里，希望可以通过本文抛砖引玉，期待关于GeoServer地图数据部署更加精彩的文章。下一篇将详细介绍如何发布地图图层、应用地图样式以及通过Openlayers的方式预览地图数据。

[**GeoServer地图开发解决方案（四）：发布Web地图服务（WMS）篇**](http://www.cnblogs.com/beniao/archive/2011/01/11/1931028.html)

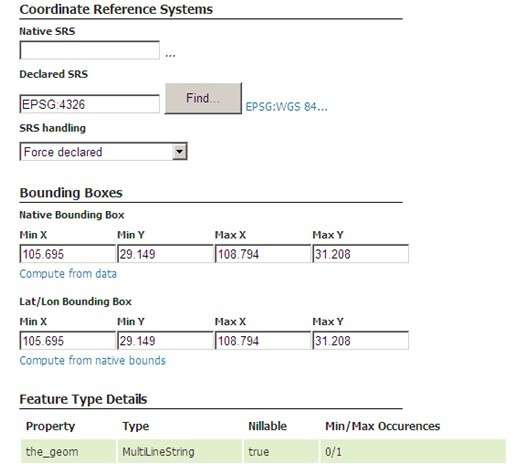
**一、发布地图图层**

　　在上一篇《[GeoServer地图开发解决方案（三）：部署地图数据篇](http://www.cnblogs.com/beniao/archive/2011/01/10/1930995.html)》中大家都看到基于工作空间的数据存储器管理的地图图层，还提到可在图层列表中发布地图图层为WMS服务，对于已经发布过的地图数据还可以进行修改发布。



　　从上图上可以看到每个图层有“Published”属性，表示当前图层是否发布，其后还有操作连接“Publish again”或“Publish”，表示对以及发布的图层进行再次发布，或者对没有发布的图层进行发布。

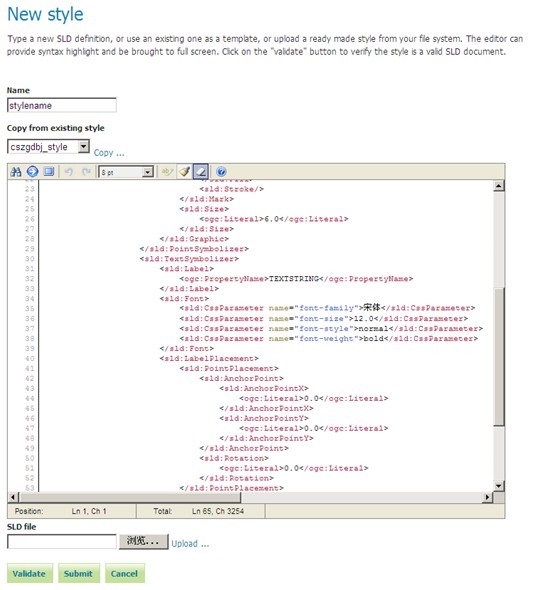
　　要发布地图图层可以在图层列表中点击“Publish”进入图层发布界面，这里需要注意的就是在进行图层发布的时候有几个必填的参数，分别为Declared SRS、Bounding Boxes等。Declared SRS表示当前发布的地图图层将采用何种地理空间引用标准，这里通常都是使用的EPSG:4326标准；Bounding Boxes表示当前图层的经度、纬度范围，这两项值可以直接通过GeoServer提供的工具自动获取到。详细如下图所示：



**二、发布地图样式**

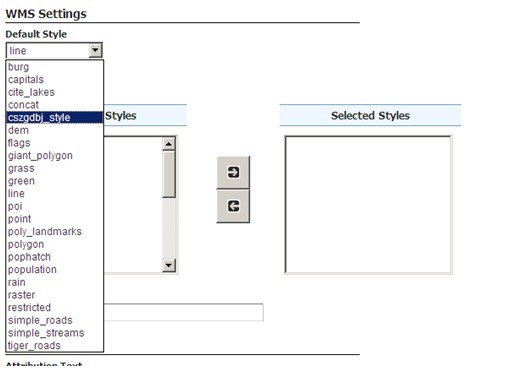
　　之所以要应用样式，主要是解决中文编码的问题以及某些地图数据需要做特别的处理，通常可以修改样式文件实现，这里大多数的图层使用默认的样式基本不会有问题，某些图层使用默认的样式却不能正确的发布为WMS服务，就需要使用前面所介绍的通过uDig对地图数据的样式进行编辑并导出样式文件（.sld），然后再GeoServer中发布新的样式，并在图层发布或者是编辑图层的时候就使用自己发布的样式。

　　通过样式列表界面的“Add a new style”就可进入下面的样式发布界面。如下图所示：



**三、应用地图样式**

　　应用地图样式非常简单，在添加地图图层或者编辑地图图层的时候，通过“Default style”功能选项进行地图图层的样式设置。详细如下图所示：

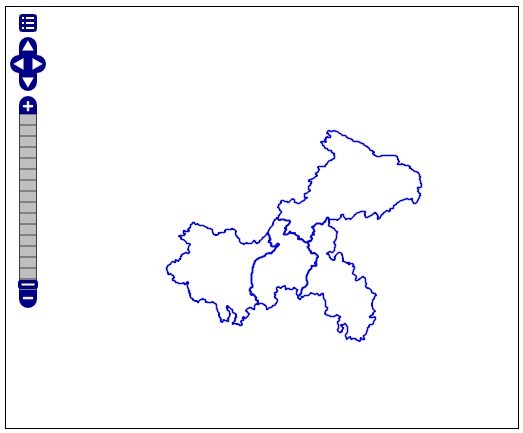


**四、预览地图图层**

　　通过GeoServer管理台左边功能导航“Layer Preview”可进入图层阅览列表，在此列表中部分类型地图图层还提供导出为KML和GML格式的地图数据。



　　从上图所示中点击图层所对应的“OpenLayers”就可以打开所对应地图图层的预览界面，上图图层列表的地图图层是为了测试而部署的，比如需要查看重庆大区界地图图层，那么可以直接预览大区边界地图图层就可以了，也就是上图中的名为“cq:dqjx\_polyline”的图层。



　　如上面预览地图图层，实际上就是加载了单个地图图层，其对应的URL地址如下表格所示：

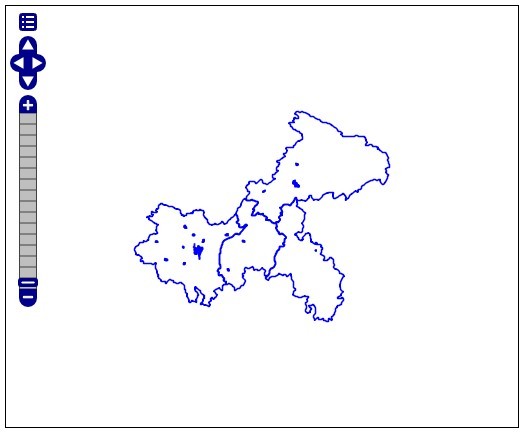
|  |  |  |
| --- | --- | --- |
| **WMS服务的单个地图图层加载分析表** | | |
| <http://localhost:8080/geoserver/wms?service=WMS&version=1.1.0&request=GetMap&layers=cq:dqjx_polyline&styles=&bbox=105.286,28.163,110.194,32.203>  &width=512&height=421&srs=EPSG:4326&format=application/openlayers | | |
| **参数名** | **参数值** | **参数说明** |
| service | WMS | 表示该服务为WMS服务 |
| Version | 1.0.1 | WMS服务版本 |
| request | GetMap | WMS服务请求接口类型,GetMap表示请求地图数据 |
| layers | cq:dqjx\_polyline | WMS所请求的地图图层 |
| Styles |  | 图层样式 |
| bbox | 105.286,28.163,110.194,32.203 | 请求的地图图层范围 |
| width | 512 | 宽度 |
| height | 512 | 高度 |
| srs | EPSG:4326 | 地理空间引用标准 |
| format | application/openlayers | 地图响应格式(openlayers) |

**五、多图层叠加呈现**

　　在访问WMS服务的时候通常都是将多个地图图层进行叠加组合，才能完成一个相对完整的地图界面呈现。在访问WMS服务的时候图层叠加非常简单，就是在WMS请求地址的layers参数后面用逗号（“,”）将多个图层进行分割就可以了。下面的WMS服务请求地址就叠加了3个地图图层：  
<http://localhost:8080/geoserver/wms?service=WMS&version=1.1.0&request=GetMap&layers=cq:dqjx_polyline,cq:cszgddx_polyline,cq:cszgd_polyline>

&styles=&bbox=105.286,28.163,110.194,32.203&width=512&height=421&srs=EPSG:4326&format=application/openlayers

　　layers=cq:dqjx\_polyline,cq:cszgddx\_polyline,cq:cszgd\_polyline，表示此WMS服务请求是由三个地图图层叠加组合而成的。最终的预览效果如下图：



[**GeoServer地图开发解决方案（五）：基于Silverlight技术的地图客户端实现**](http://www.cnblogs.com/beniao/archive/2011/01/12/1931029.html)

　　我曾经写作过一篇关于微软Bing Maps的客户端实现的博文：《[基于DeepZoom技术的Bing Maps客户端实现研究](http://www.cnblogs.com/beniao/archive/2010/05/21/1740480.html)》，详细介绍了如何使用Silverlight中的DeepZoom技术实现Bing Maps的客户端。本篇介绍的内容则为基于Web地图服务（Web Map Service，简称：WMS）的Silverlight地图客户端实现。

**一、DeepZoom简介**

　　DeepZoom技术以MultiScaleImage控件为核心，其内部有一个MultiScaleTileSource类型的源属性，主要用于设置MultiScaleImage控件所要呈现的数据源。基于Silverlight的Web GIS客户端实现也是通MultiScaleImage控件来实现，核心就在于通过MultiScaleTileSource属性针对不同的Web GIS地图瓦片数据（Image Tiles）提供商为MultiScaleImage控件实现一个数据源。因此本篇所需要做的工作就是针对WMS服务为MultiScaleImage控件实现一套加载数据源的算法。

**二、WMS服务加载实现**

　　实现WMS服务加载的算法其实非常简单，只需要了解WMS发布的方式、WMS地址的参数组成结构以及地图瓦片的投影原理就可以了，首先需要定义一个盒子对象作为访问WMS的边界参数对象。

[复制代码](javascript:void(0);)

public class BBox  
{  
    public int X { get; set; }  
    public int Y { get; set; }  
    public int Width { get; set; }  
    public int Height { get; set; }  
  
    public BBox(int x, int y, int w, int h)  
    {  
        this.X = x;  
        this.Y = y;  
        this.Width = w;  
        this.Height = h;  
    }  
}

[复制代码](javascript:void(0);)

　　关于WMS服务加载的详细算法需要一些GIS理论基础才能够知道具体的实现原理，这里我就不逐一介绍，直接贴代码：

[复制代码](javascript:void(0);)

public class WMSTileSource : MultiScaleTileSource   
{  
    public WMSTileSource()  
        : base(int.MaxValue, int.MaxValue, 0x100, 0x100, 0)  
    { }  
  
    public const int TILE\_SIZE = 256;  
    /// <summary>  
    /// 地球半径  
    /// </summary>  
    public const double EARTH\_RADIUS = 6378137;  
    /// <summary>  
    /// 地球周长  
    /// </summary>  
    public const double EARTH\_CIRCUMFERENCE = EARTH\_RADIUS \* 2 \* Math.PI;  
    public const double HALF\_EARTH\_CIRCUMFERENCE = EARTH\_CIRCUMFERENCE / 2;  
  
    /// <summary>  
    /// WMS服务地址  
    /// </summary>  
    private const string TilePath = @"http://localhost:8080/geoserver/wms?service=WMS&version=1.1.0&request=GetMap&layers=cq:CQ\_County\_region,cq:CQ\_County\_region\_level&styles=&bbox={0},{1},{2},{3}&width=512&height=421&srs=EPSG:4326&&Format=image/png";  
  
    public string GetQuadKey(string url)  
    {  
        var regex = new Regex(".\*tiles/(.+)[.].\*");  
        Match match = regex.Match(url);  
  
        return match.Groups[1].ToString();  
    }  
  
    public BBox QuadKeyToBBox(string quadKey, int x, int y, int zoomLevel)  
    {  
        char c = quadKey[0];  
  
        int tileSize = 2 << (18 - zoomLevel - 1);  
  
        if (c == '0')  
        {  
            y = y - tileSize;  
        }  
  
        else if (c == '1')  
        {  
            y = y - tileSize;  
            x = x + tileSize;  
        }  
  
        else if (c == '3')  
        {  
            x = x + tileSize;  
        }  
  
        if (quadKey.Length > 1)  
        {  
            return QuadKeyToBBox(quadKey.Substring(1), x, y, zoomLevel + 1);  
        }  
        return new BBox(x, y, tileSize, tileSize);  
    }  
  
    public BBox QuadKeyToBBox(string quadKey)  
    {  
        const int x = 0;  
        const int y = 262144;  
        return QuadKeyToBBox(quadKey, x, y, 1);  
    }  
  
    public double XToLongitudeAtZoom(int x, int zoom)  
    {  
        double arc = EARTH\_CIRCUMFERENCE / ((1 << zoom) \* TILE\_SIZE);  
        double metersX = (x \* arc) - HALF\_EARTH\_CIRCUMFERENCE;  
        double result = RadToDeg(metersX / EARTH\_RADIUS);  
        return result;  
    }  
  
    public double YToLatitudeAtZoom(int y, int zoom)  
    {  
        double arc = EARTH\_CIRCUMFERENCE / ((1 << zoom) \* TILE\_SIZE);  
        double metersY = HALF\_EARTH\_CIRCUMFERENCE - (y \* arc);  
        double a = Math.Exp(metersY \* 2 / EARTH\_RADIUS);  
        double result = RadToDeg(Math.Asin((a - 1) / (a + 1)));  
        return result;  
    }  
  
    public double RadToDeg(double d)  
    {  
        return d / Math.PI \* 180.0;  
    }  
  
    private static string TileXYToQuadKey(int tileX, int tileY, int levelOfDetail)  
    {  
        var quadKey = new StringBuilder();  
        for (int i = levelOfDetail; i > 0; i--)  
        {  
            char digit = '0';  
            int mask = 1 << (i - 1);  
            if ((tileX & mask) != 0)  
            {  
                digit++;  
            }  
            if ((tileY & mask) != 0)  
            {  
                digit++;  
                digit++;  
            }  
            quadKey.Append(digit);  
        }  
        return quadKey.ToString();  
    }  
  
    protected override void GetTileLayers(int tileLevel, int tilePositionX, int tilePositionY, System.Collections.Generic.IList<object> tileImageLayerSources)  
    {  
        int zoom = tileLevel - 8;  
        if (zoom > 0)  
        {  
            string quadKey = TileXYToQuadKey(tilePositionX, tilePositionY, zoom);  
            BBox boundingBox = QuadKeyToBBox(quadKey);  
  
            double lon = XToLongitudeAtZoom(boundingBox.X \* TILE\_SIZE, 18);  
            double lat = YToLatitudeAtZoom(boundingBox.Y \* TILE\_SIZE, 18);  
  
            double lon2 = XToLongitudeAtZoom((boundingBox.X + boundingBox.Width) \* TILE\_SIZE, 18);  
            double lat2 = YToLatitudeAtZoom((boundingBox.Y - boundingBox.Height) \* TILE\_SIZE, 18);  
  
            string wmsUrl = string.Format(TilePath, lon, lat, lon2, lat2, TILE\_SIZE);  
  
            var veUri = new Uri(wmsUrl);  
            tileImageLayerSources.Add(veUri);  
        }  
    }  
}

[复制代码](javascript:void(0);)

　　前端通过一个按钮事件驱动触发加载WMS服务，按钮的XAML代码如下：

<Button Content="WMS图层" Height="30" Width="80" Name="btnWms" Click="btnWms\_Click"/>

　　示例我就直接基于《[基于DeepZoom技术的Bing Maps客户端实现研究](http://www.cnblogs.com/beniao/archive/2010/05/21/1740480.html)》一文中的示例扩展，对应的后台代码为如下代码块：

private void btnWms\_Click(object sender, RoutedEventArgs e)  
{  
    msi.Source = new WMSTileSource();  
}

