**OGCServer的支持的URL分析**

支持的功能：ＷＭＳ三大功能：GetCapabilities（获取服务能力）, GetMap（获取地图）和GetFeatureInfo（获取对象信息）。其中GetMap为核心操作，此操作得到一幅地图图像。

参考网址：<http://www.cnblogs.com/yinxiangpei/articles/2600796.html>

提供较为详细的参数说明：

<http://blog.csdn.net/drr789/article/details/8998638>

！！！ＯＧＣ服务参考：

<http://support.supermap.com.cn:8090/iserver/help/html/API/OGC_intro.htm>

根据上述内容，展开实验：

请求元数据，获取服务能力信息：

<http://121.49.97.58:8000/?SERVICE=WMS&VERSION=version&REQUEST=GetCapabilities>

得到一个xml文档

分析xml文档，得到wms服务参数的可选值

1. 我们使用github上ogcserver给出的测试url进行实验，实验服务器是教研室架设的服务器

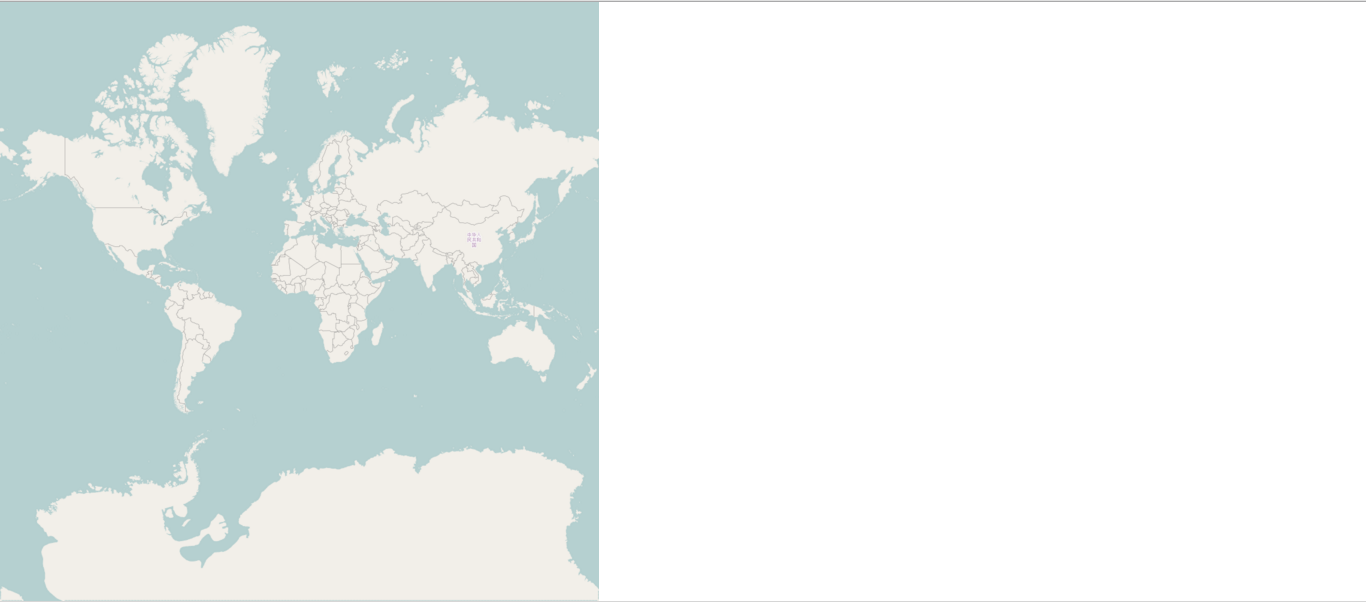
<http://121.49.97.58:8000/?LAYERS=__all__&STYLES=&FORMAT=image%2Fpng&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A3857&BBOX=-20037508.34,-20037508.34,20037508.3384,20037508.3384&WIDTH=256&HEIGHT=256>

输出：



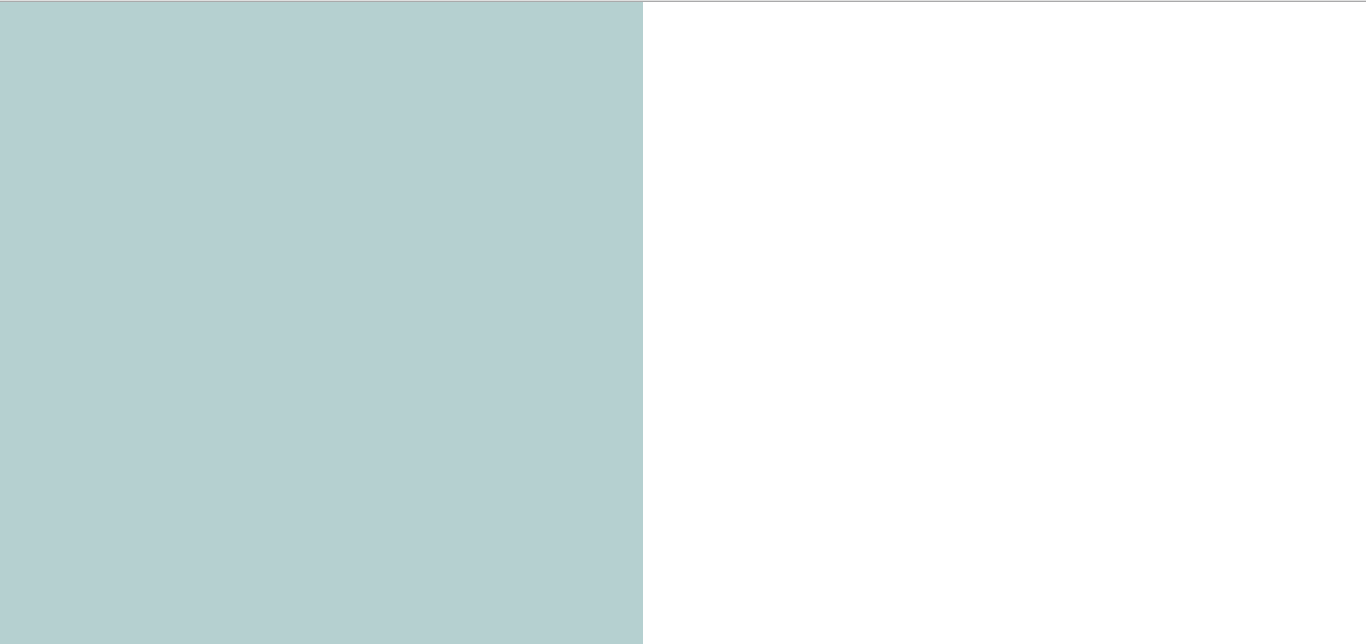
改变像素宽度和高度：

<http://121.49.97.58:8000/?LAYERS=__all__&STYLES=&FORMAT=image%2Fpng&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A3857&BBOX=-20037508.34,-20037508.34,20037508.3384,20037508.3384&WIDTH=1000&HEIGHT=1000>



更改投影方式：

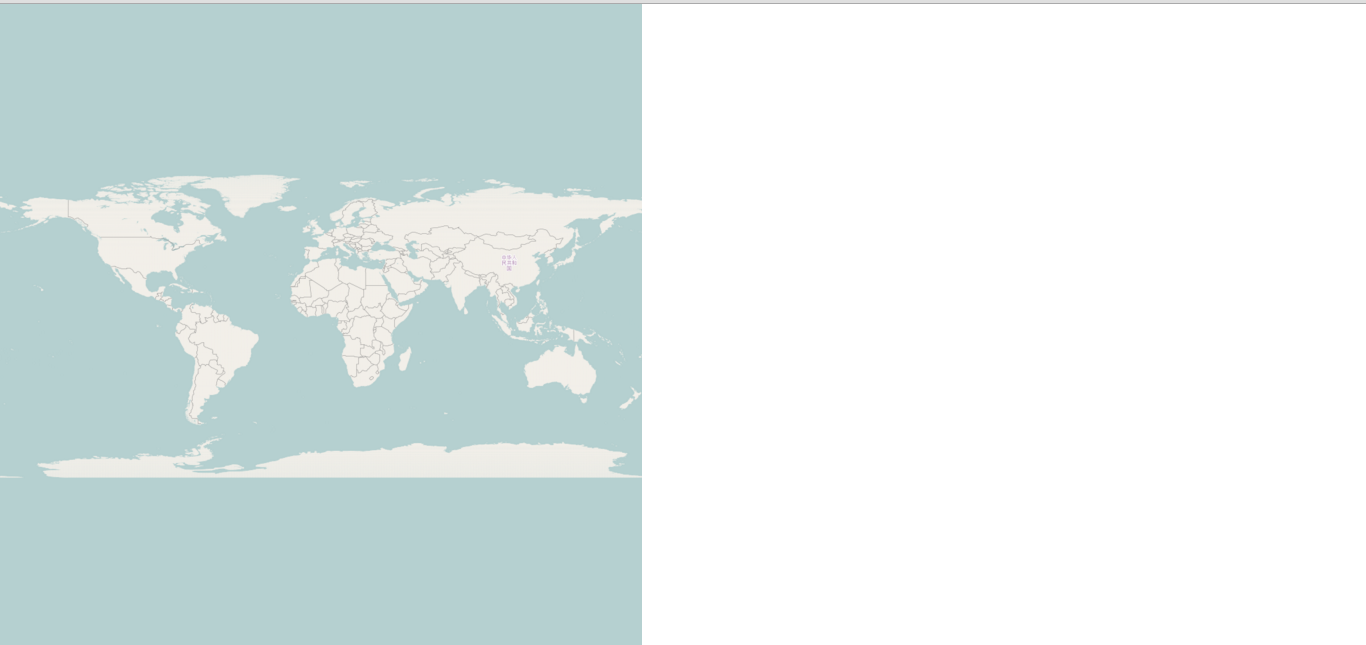
http://121.49.97.58:8000/?LAYERS=\_\_all\_\_&STYLES=&FORMAT=image%2Fpng&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=-20037508.34,-20037508.34,20037508.3384,20037508.3384&WIDTH=1000&HEIGHT=1000



猜想是投影方式变了，地图显示范围不对，即BBOX

于是，根据获得的服务信息元数据，对BBOX进行调整：

<http://121.49.97.58:8000/?LAYERS=__all__&STYLES=&FORMAT=image%2Fpng&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=-179,-89,179,89&WIDTH=1000&HEIGHT=1000>



接下来对图片的格式进行实验，

我们将之前的图片另存为，发现格式为png

更改format参数：

<http://121.49.97.58:8000/?LAYERS=__all__&STYLES=&FORMAT=image%2Fjpeg&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=-179,-89,179,89&WIDTH=1000&HEIGHT=1000>

重复上步操作，发现图片格式为jpeg



由于style和layer需要一一对应，接下来我们只对layer进行测试

为了更清楚地进行观察比较，我们首先调整BBOX 参数，使地图较为清晰的显示局部内容。

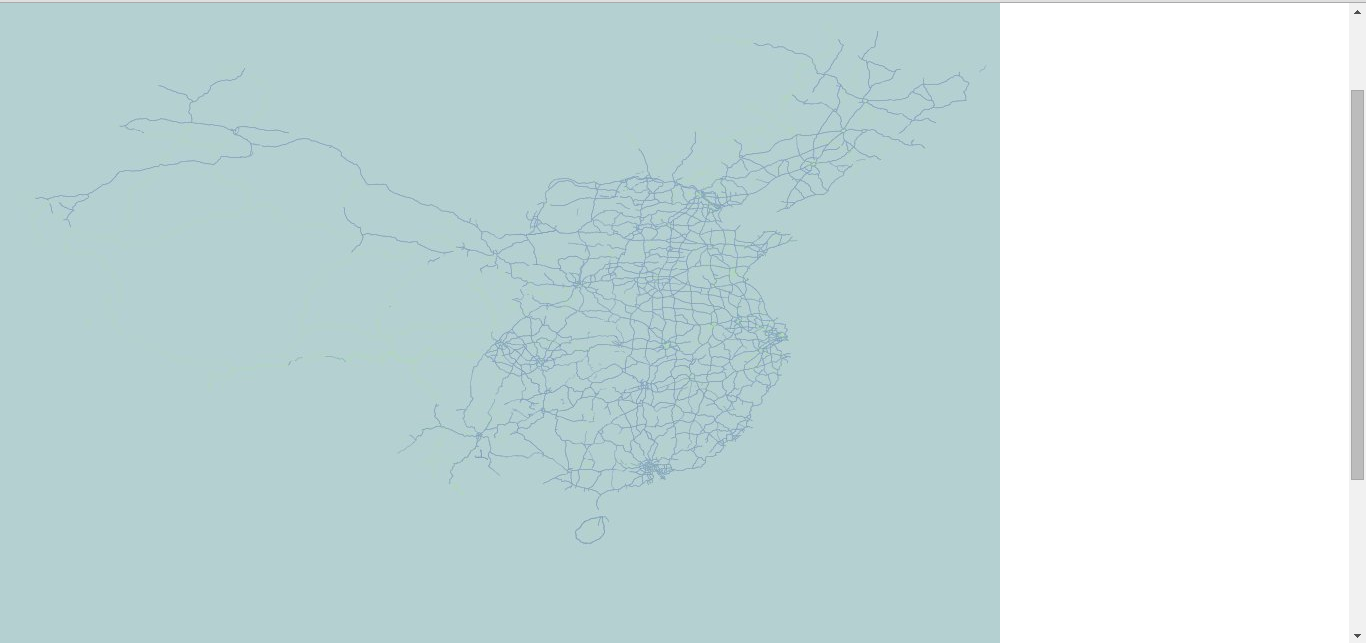
http://121.49.97.58:8000/?LAYERS=\_\_all\_\_&STYLES=&FORMAT=image%2Fjpeg&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=73,3,135,53&WIDTH=1000&HEIGHT=1000

同时在浏览器看全大小显示的图片：



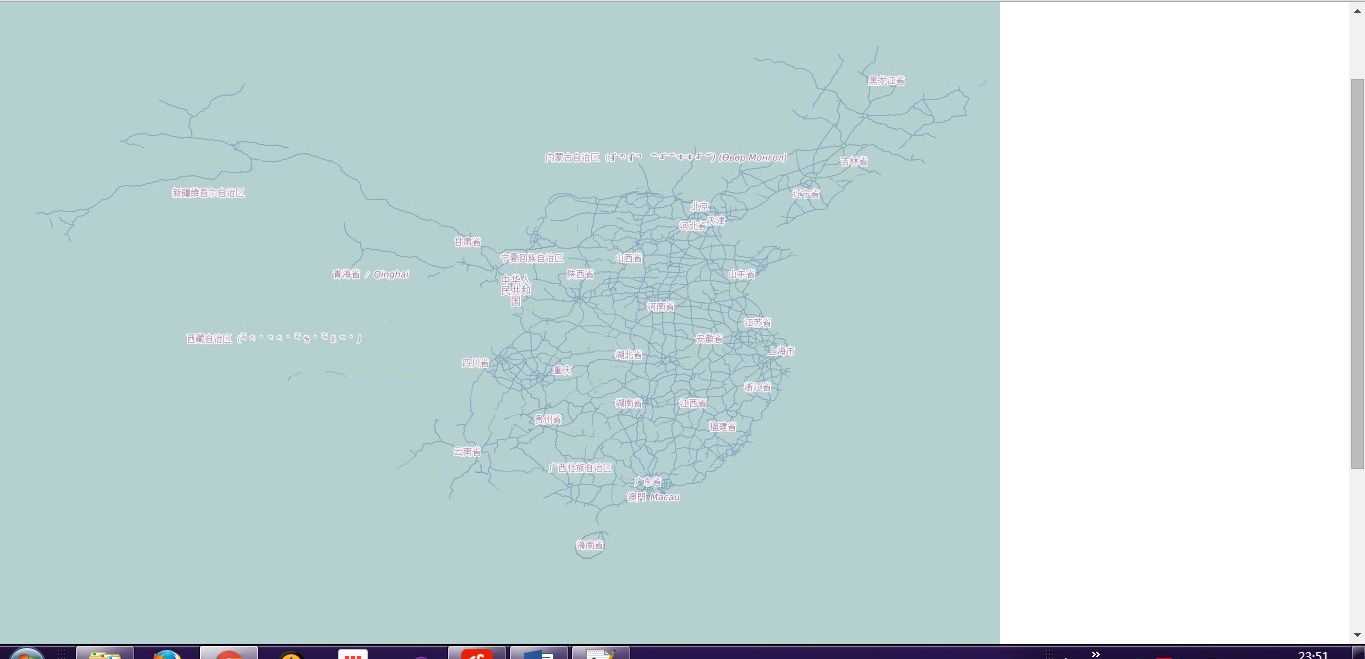
我们选择显示roads：

<http://121.49.97.58:8000/?LAYERS=roads&STYLES=&FORMAT=image%2Fjpeg&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=73,3,135,53&WIDTH=1000&HEIGHT=1000>



我们选择roads和placenames-large：

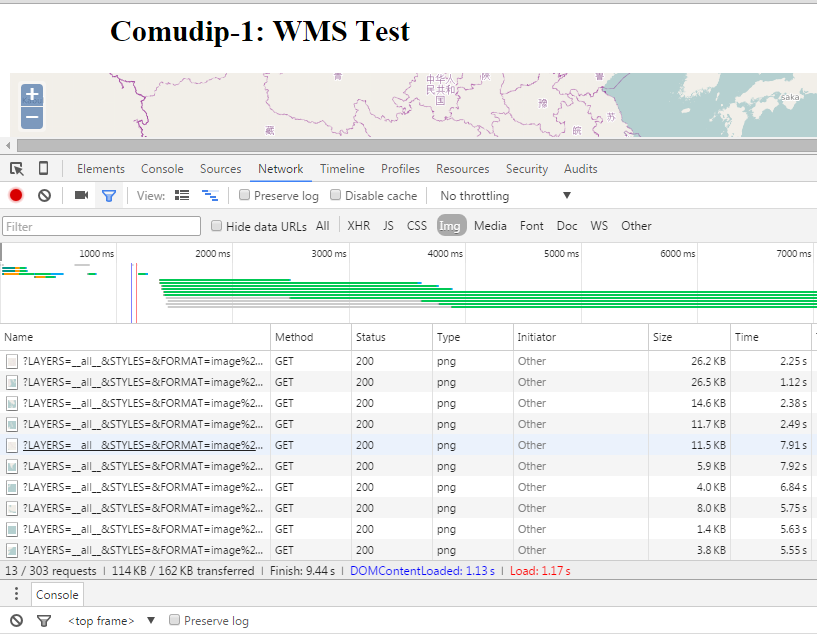
<http://121.49.97.58:8000/?LAYERS=roads,placenames-large&STYLES=&FORMAT=image%2Fjpeg&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&SRS=EPSG%3A4326&BBOX=73,3,135,53&WIDTH=1000&HEIGHT=1000>



其他还可以缩小范围，或是使用更大的像素来进行查看测试，此处不再一一实验。

在实验中可以明显的感受到，这样直接请求较大尺寸的图片，需要教长的渲染等待时间，所以openlayer在请求的时候是进行小尺寸的多次请求，

我们对使用了openlayer的页面进行观测，观察network的内容，会发现其发送了多个小尺寸的getmap请求。



由此，我们也更加明确了地图服务前后端的分工，也就是说，ogcserver只需根据客户端的Url提供相应的数据即可。而不必关心其他。

**Ogcserver改进前准备工作，思路分析**

我们之后要对ogcserver进行改进，添加我们需要的功能。要完成改进，我们需要先了解现有的系统，只有在弄清了现有的系统组成，工作原理，数据流向后，我们才有可能有的放矢的对其进行修改，添加相应的功能。

现在我们已经了解到，ogcserver运作的基本流程是：

解析Url ——》将解析的请求转化为对mapnik的命令——》mapnik进行渲染——》传输数据到客户端。

我们添加功能之后，会增加相应的字段，以便客户端通过Url对服务器发起相应的请求。所以需要了解Url的解析过程。同时需要构思如何将解析得到的请求进行进一步的实现。这就涉及到如何将解析得到的参数转换我我们的下一步操作的命令或是动作，这实际对应的已有系统的将请求转换为mapnik命令的过程，所以这一过程需要研究，作为示例参考。况且，我们添加的功能可能会直接对这一过程进行修改，扩展。所以对其的学习是必须的。同时，由于我们需要将服务结果传回客户端，也就必须了解最后数据传回的阶段如何实现。

总的来说，要解构这个系统，需要抓住数据的流向。主要是

1．由URL中解析出相关参数;

2．将相关请求参数转换为mapnik命令或是实际的动作。

3．数据传回客户端。

我们考虑到数据的量可能会比较大，要全部跟踪会比较困难，为了避免遗漏，我们以一二步的共同部分，亦即解析出的参数，作为切入点。因为这部分参数是这两个步骤的纽带，我们可以从第一部正向追逐，同时可以从第二步逆向追踪。这样可以尽可能不遗漏的将其全部找出，也就可以保证对其他数据追踪的完整性。

同时，这些参数处于一个接口位置，便于我们锁定，进行验证。验证思路是，对我们找到的参数进行直接赋值，然后更改客户端的url请求，看服务器的返回数据是否不再受所接收的url的影响。如果我们对参数的锁定成功的屏蔽了所有的url请求，就表示我们的参数已经全部找出。关于url请求的问题，我们前面已经进行了讨论，此处不再展开讨论。