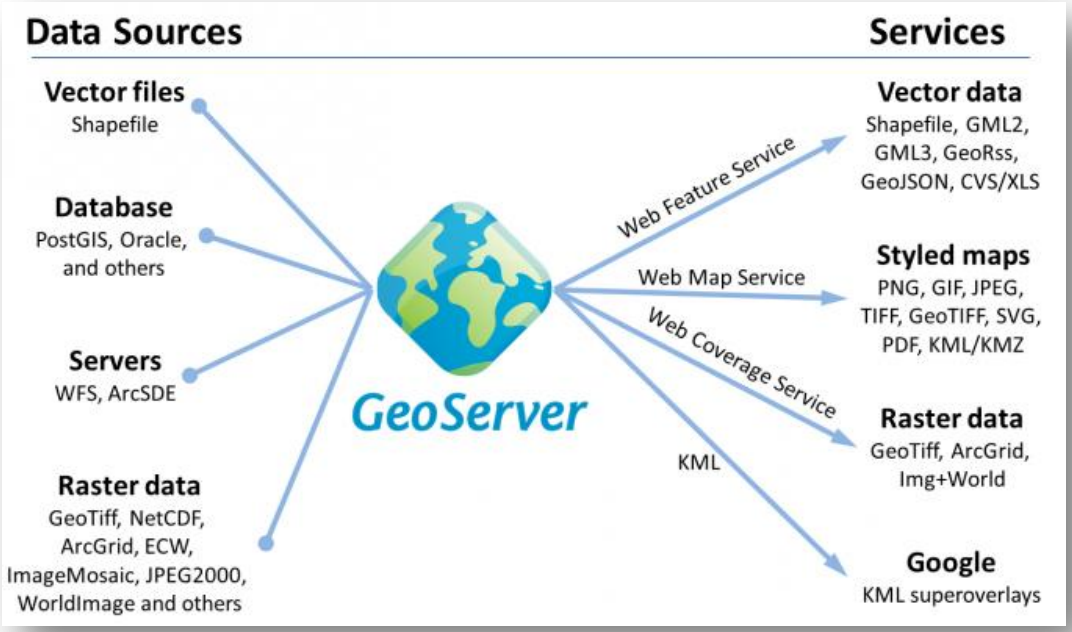
Introduction to Open Source GIS Server for Web Mapping – A Practical Approach

What you'll be learning

- Basic Understanding of OWS Servers
- Installing and Setting Up GeoServer
- Publishing of OGC service Using GeoServer
- Understanding WMS service
- Map Querying Using GeoServer CQL/ECQL
- MapServer for Windows
  - Understanding Mapfile
  - Publishing of OGC service Using MapServer

What is GeoServer? Why would You Use it?

- GeoServer is an open source **OWS** server for **sharing geospatial data**
- Server Side application publish and serves maps from any spatial data source using OGC open standards such as WMS, WFS, WPS, WCS, TMS, WMTS etc.
- Excels at **handling very large datasets**, both raster and vector.
- Produces high quality rendering of maps and can handle hundreds to thousands of map layers easily.



Walkthrough : Setting Up GeoServer

Setting Up GeoServer

1. Install the latest version of Java JRE
2. Download **GeoServer** and choose/download appropriate Version
3. Indicate the path of JRE subfolder. If you're using Windows, this is probably **C:\Program Files (x86)\Java\jre8** or **C:\Program Files (x86)\Java\jre1.8.x** or something similar.
4. Once GeoServer is installed, start it by clicking **Start > All Programs > GeoServer 2.x.x > Start GeoServer**. You'll see a bunch of status messages appearing in a black and white console, like the following.

If you get a Windows Security Alert that Windows Firewall has blocked some features of the program, check the top box to allow it to run on private networks and click **Allow access**. Uncheck the bottom box, as public access will not be needed in this course.

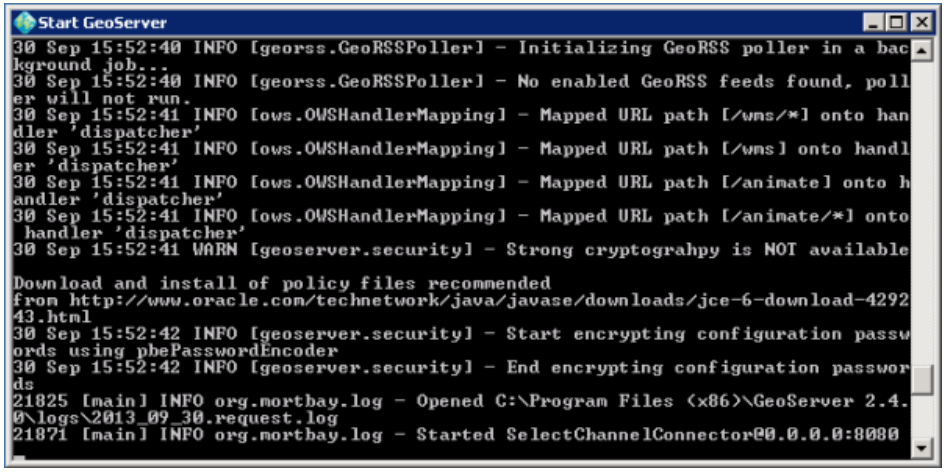


Figure: Status messages during GeoServer startup.

5. Click **Start > All Programs > GeoServer 2.14.0 > GeoServer Web Admin Page** (or go to your browser and enter the address **localhost:8080/geoserver/web**).
   This is a web page that you can use to administer GeoServer from this or any other computer in your network. GeoServer includes a servlet called **Jetty**, for serving web service and web page requests

Setting Up GeoServer

6. Type a GeoServer username and password in the upper boxes and click **Login**. You may remember that the installation created an administrative user with the username **admin** and the password **geoserver**. You must use this the first time you log in. You will see a welcome page similar to the following

7. GeoServer works with the concept of layers and layer groups. You define a set of datasets that you want to have exposed on your server

8. In the left-hand menu, click **Layer Preview**.

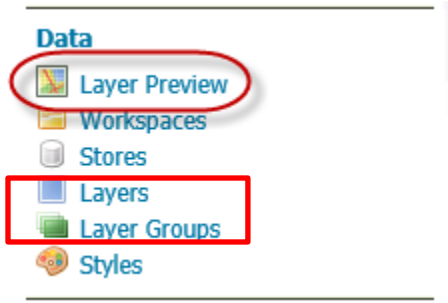


Figure : Layer Preview

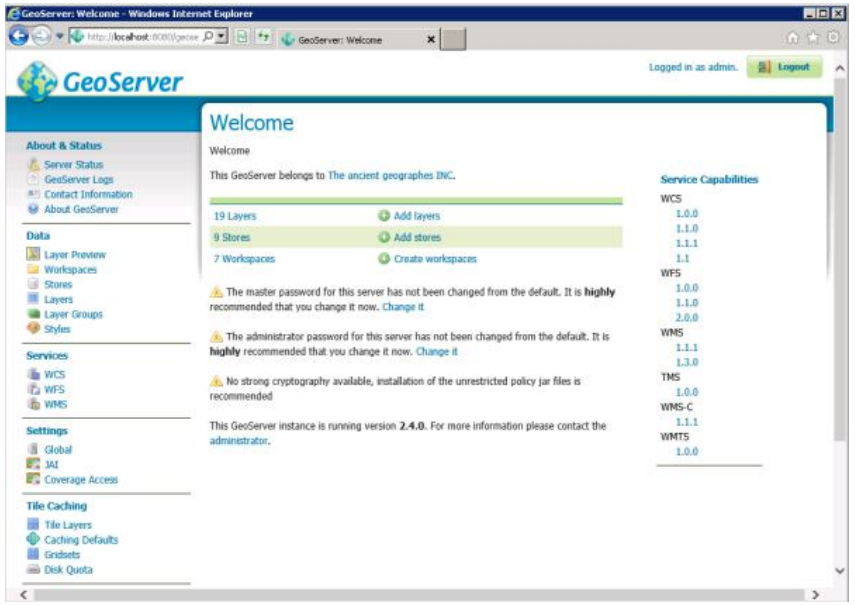


Figure: Geoserver startup webpage

Setting Up GeoServer

9. Scroll down to the Tasmania state boundaries layer and click the **OpenLayers** link.

This displays your map as a web service that you can navigate. The web service was delivered through the Open Geospatial Consortium (OGC) Web Map Service (WMS) specification, an openly documented way of serving web maps. The map frame and navigation buttons were created through the OpenLayers JavaScript framework.

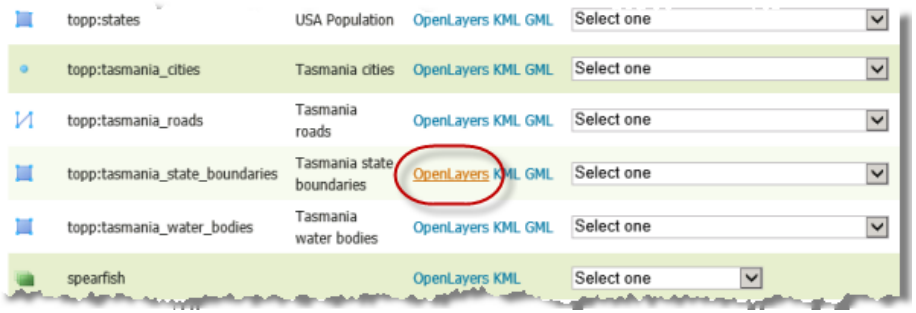


Figure : Find Tasmania in the list of layers

It's important to understand that you also could have done this by clicking the dropdown list and choosing **WMS > OpenLayers**. Looking at this list, you get a better idea of the many different output formats supported by GeoServer.

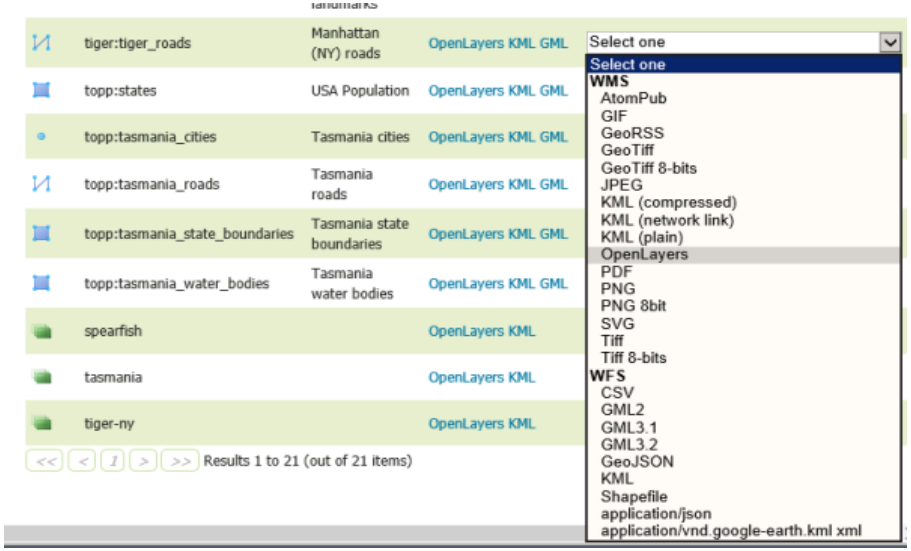


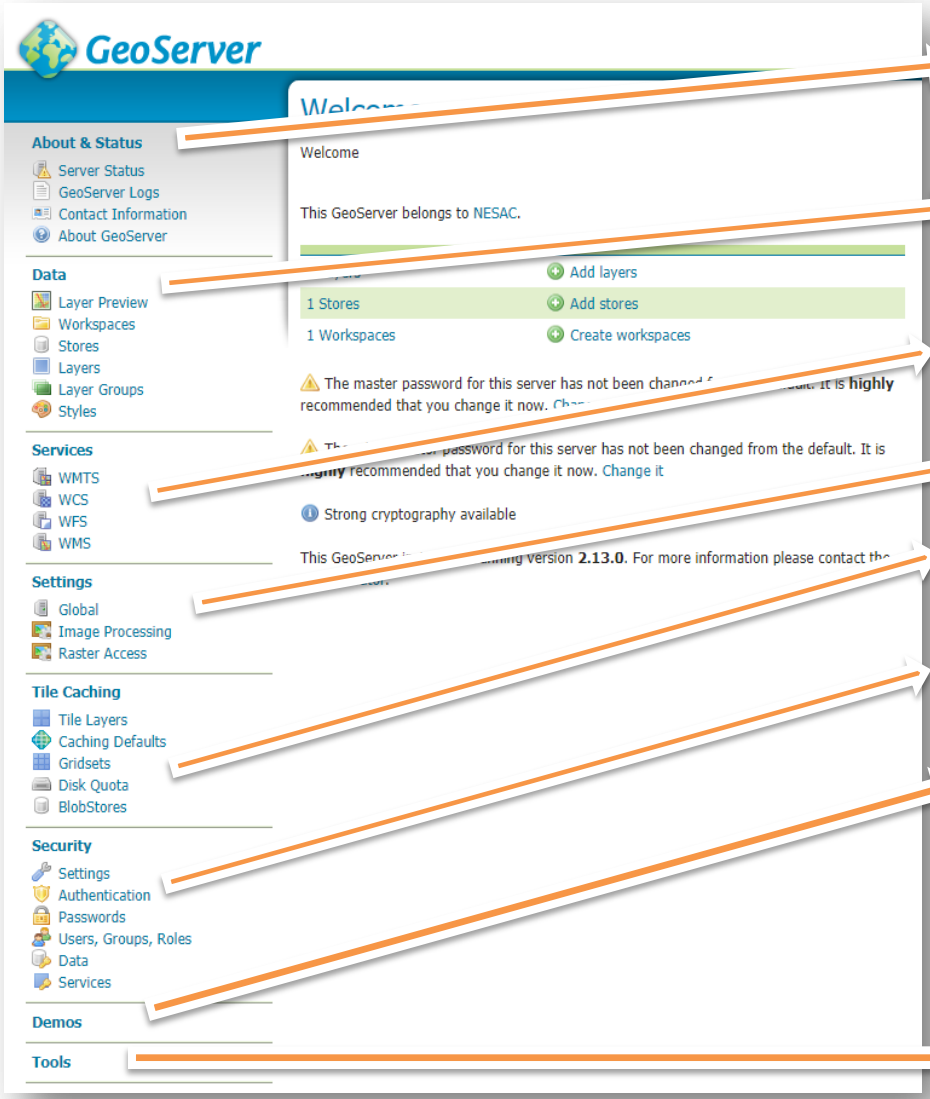
Figure : OpenLayers selection in GeoServer

Setting Up GeoServer

10. Close the map preview window and return to the GeoServer layer preview list. This time, click the **KML** link to get the layer as KML, and XML-based open specification for geographic data, often used by Google. If you have Google Earth installed, the layer should open there. If you don't have Google Earth, you can open the layer in Notepad and just examine the raw KML.

This confirms that, your GeoServer has been Set Up and Configured properly!

Understanding GeoServer : Web administration interface



The **About & Status** section provides access to GeoServer **diagnostic and configuration tools**, and can be particularly useful for debugging.

The **Data** management section contains configuration options for all the **different data-related settings**.

The **Services** section is for **configuring the services** published by GeoServer.

The **Settings** section contains **configuration settings** that apply to **the entire server**.

The **Tile Caching** section configures the embedded **GeoWebCache**.

The **Security** section configures the **built-in security** subsystem.

The **Demos** section contains links to example WMS, WCS, and WFS requests for GeoServer as well as a listing all SRS info known to GeoServer. In addition, there is a reprojection console for converting coordinates between spatial reference systems, and a request builder for WCS requests

The **Tools** section **contains administrative tools**. By default, the only tool is the Catalog Bulk Load Tool, which can bulk copy test data into the catalog.

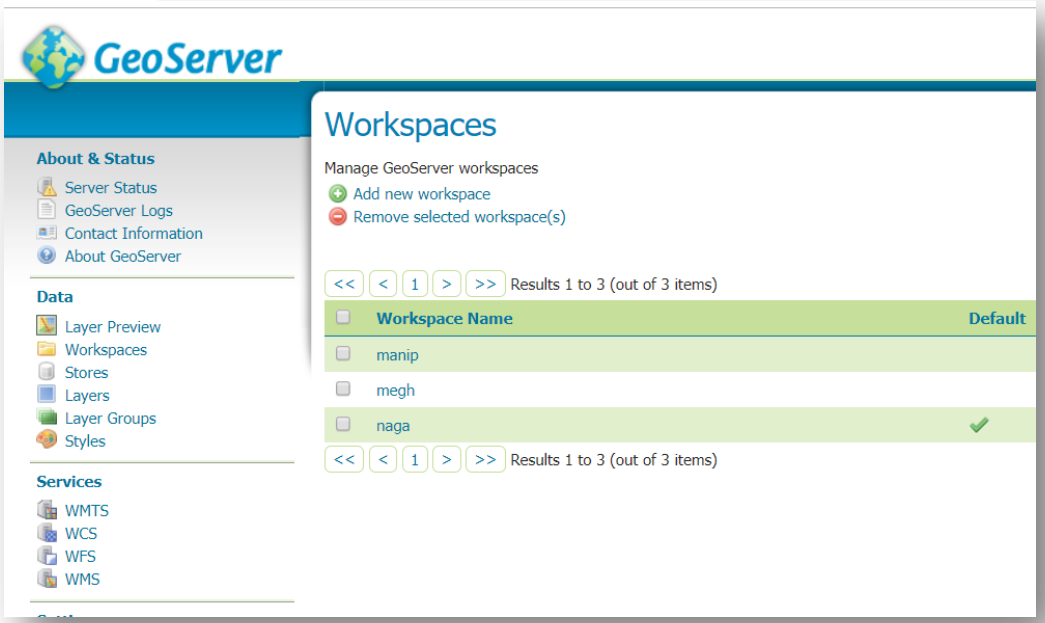
Publishing Data in GeoServer

Publishing Data in GeoServer

Eg. Publishing a Shapefile

1. Move the shapefile directory, naga_data into <GEOSERVER_DATA_DIR>/data, where <GEOSERVER_DATA_DIR> is the root of the GeoServer data directory. If no changes have been made to the GeoServer file structure, the path is geoserver/data_dir/data/nagaland_data
2. Create a new workspace
    1. In a web browser, navigate to http://localhost:8080/geoserver
    2. Login and Navigate to **Data ▸ Workspaces**.

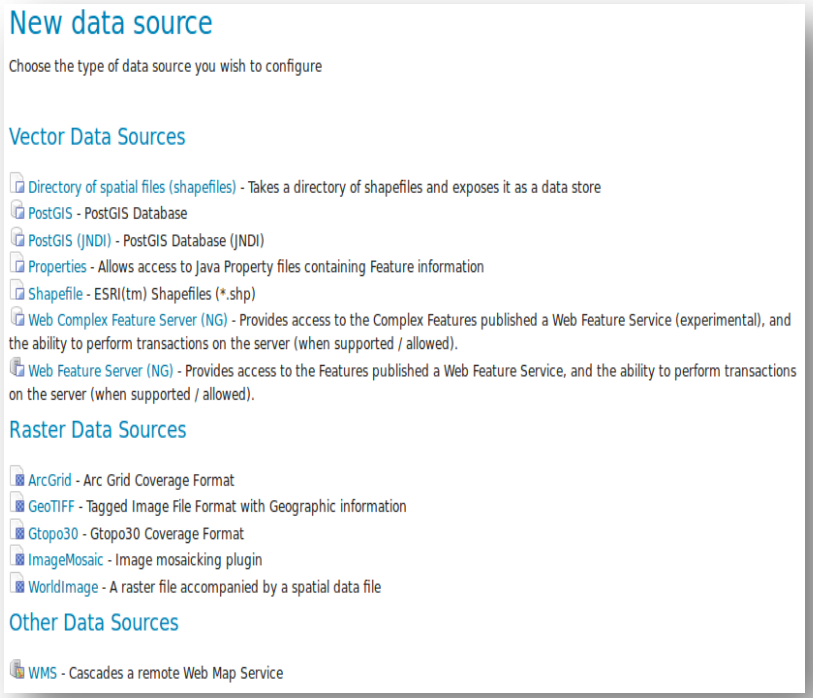
A workspace is often used to group similar layers together. Layers may be referred to by their workspace name, colon, layer name (for example **naga**:circle_boundary).



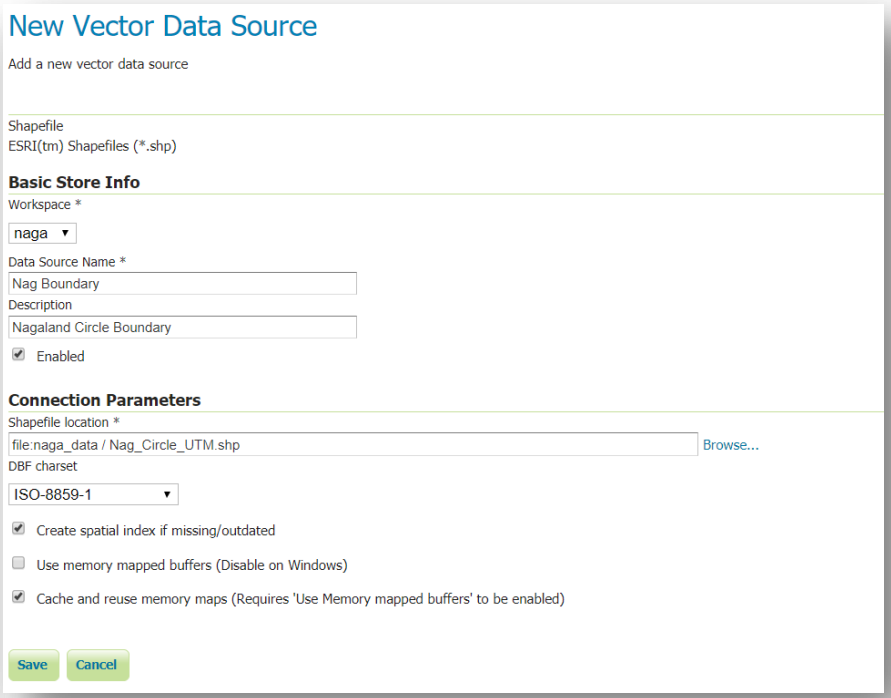
Publishing Data in GeoServer : Creating a Store

3. Create a Store under the workspace. The store tells GeoServer **how to connect to the shapefile.**
    1. Navigate to **Data▸Stores**.
    2. You should see a list of stores, including the type of store and the workspace that the store belongs to.
    3. In order to add the shapefile, you need to create a new store. Click the **Add new Store** button. You will be redirected to a list of the data sources supported by GeoServer.
    4. Click **Shapefile**. The **New Vector Data Source** page will display.
    5. Begin by configuring the **Basic Store Info**.
        - Select the workspace **naga** from the drop down menu.
        - Enter the **Data Source Name** as Nag Boundary
        - Enter a brief **Description** (such as “Nagaland Circle Boundary”).
    6. Under **Connection Parameters**, browse to the location **URL** of the shapefile, typically nagaland_data / Nag_Circle_UTM.shp.
    7. Click **Save**. You will be redirected to the **New Layer** page in order to configure the Nag_Circle_UTM layer

Publishing Data in GeoServer : Creating a Store



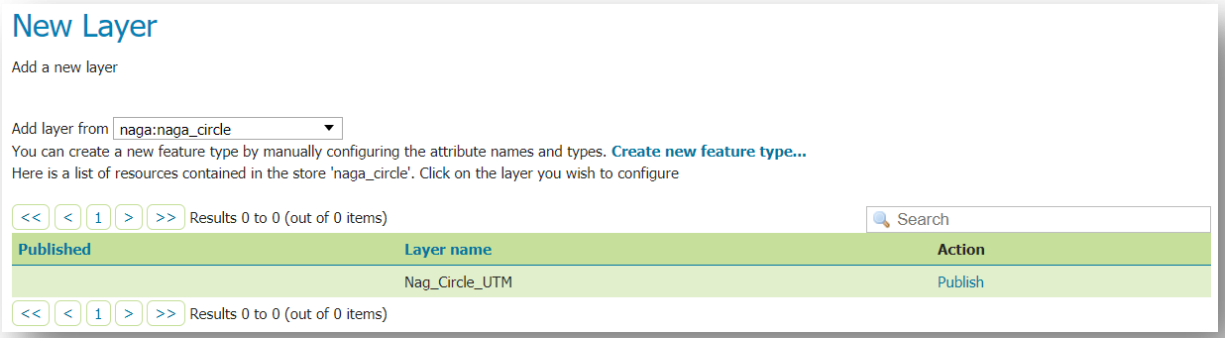
Data Stores



Basic Store Info and Connection Parameters

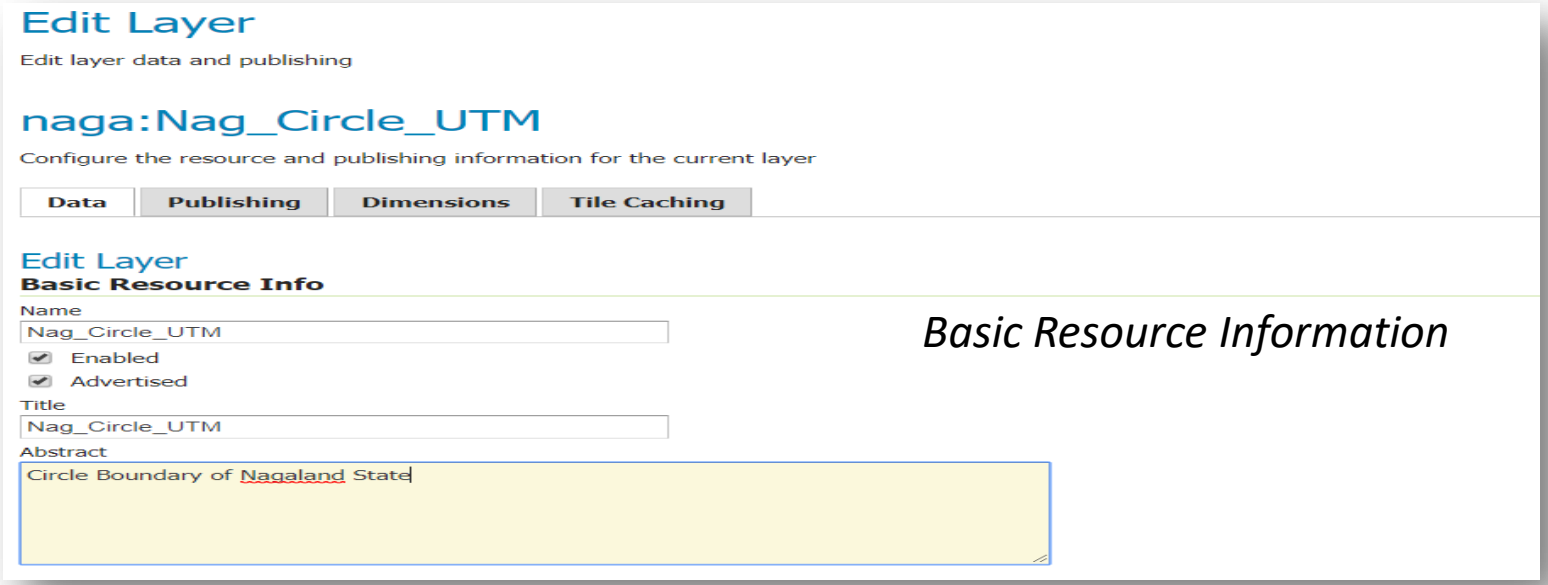
Publishing Data in GeoServer : Creating a Layer

1. On the **New Layer** page, click **Publish** beside the **Nag_Circle_UTM** layer name



New layer

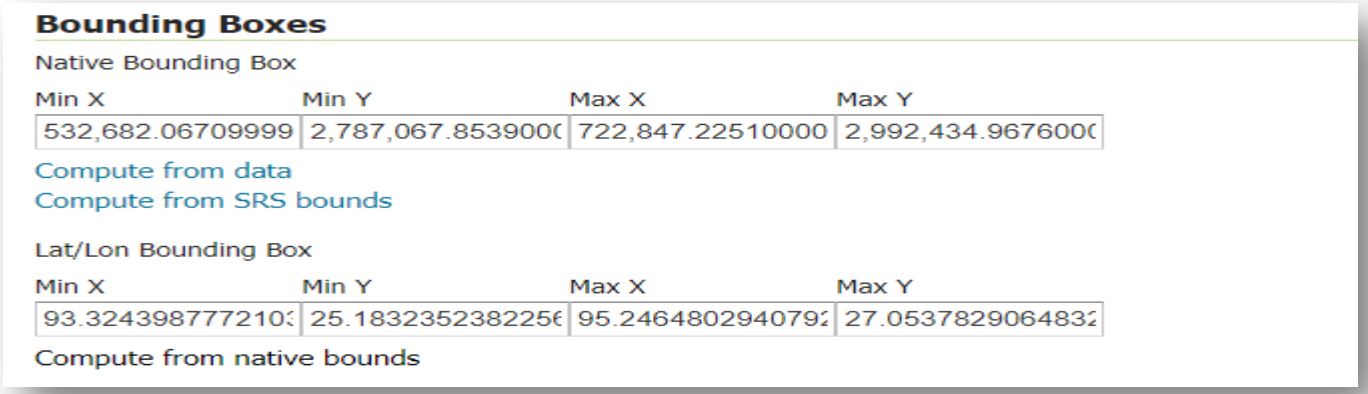
2. The **Edit Layer** page defines the data and publishing parameters for a layer. Enter a short **Title** and an **Abstract** for the Nag_Circle_UTM layer.



Basic Resource Information

Publishing Data in GeoServer : Publishing a Layer

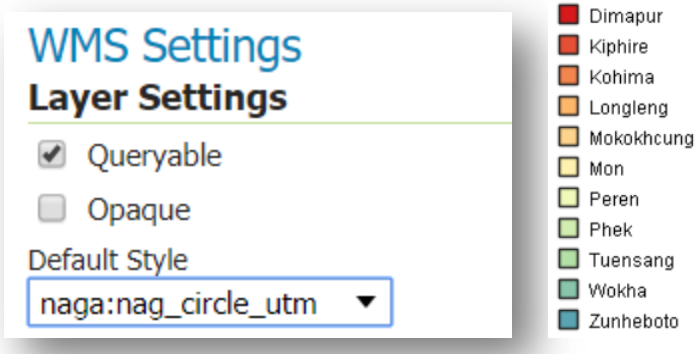
3. Generate the layer's bounding boxes by clicking the **Compute from data** and then **Compute from native bounds** links



Generating bounding boxes

4. Click the **Publishing** tab at the top of the page.

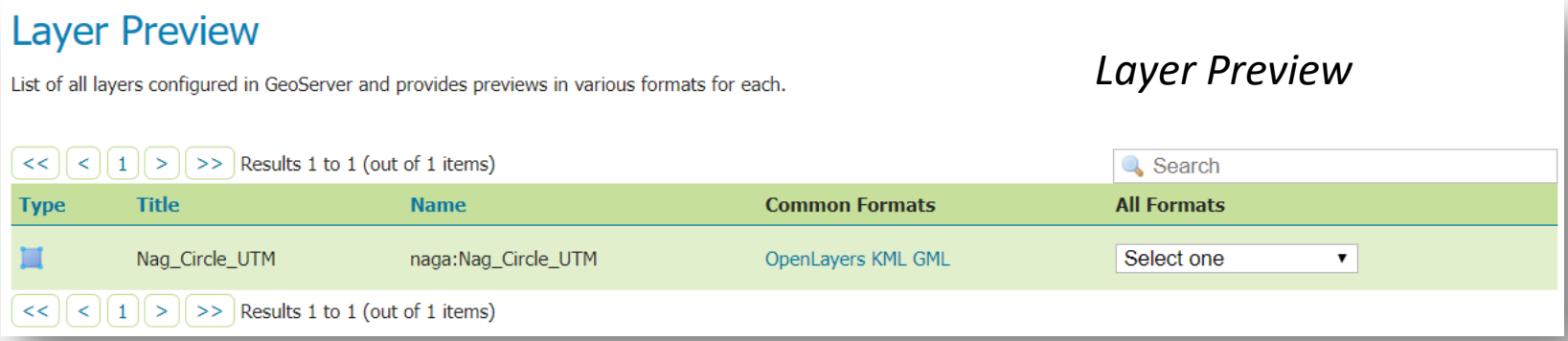
5. We can set the layer's style here. Under **WMS Settings**, ensure that the **Default Style** is set to **line**.



6. F
scr

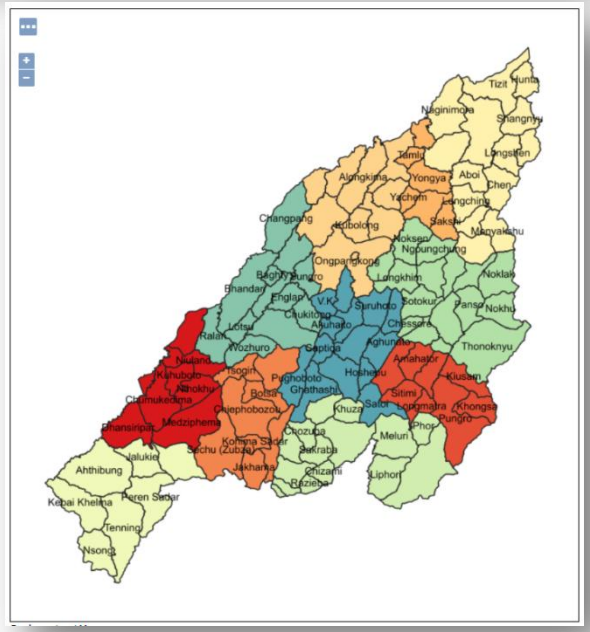
Publishing Data in GeoServer : Previewing a Layer

1. In order to verify that the Nag_Circle_UTM layer is published correctly, we can preview the layer.



Layer Preview

2. Click the **OpenLayers** link in the **Common Formats** column.
3. An OpenLayers map will load in a new tab and display the shapefile data with our custom style for the layer. You can use this preview map to zoom and pan around the dataset, as well as display the attributes of features.

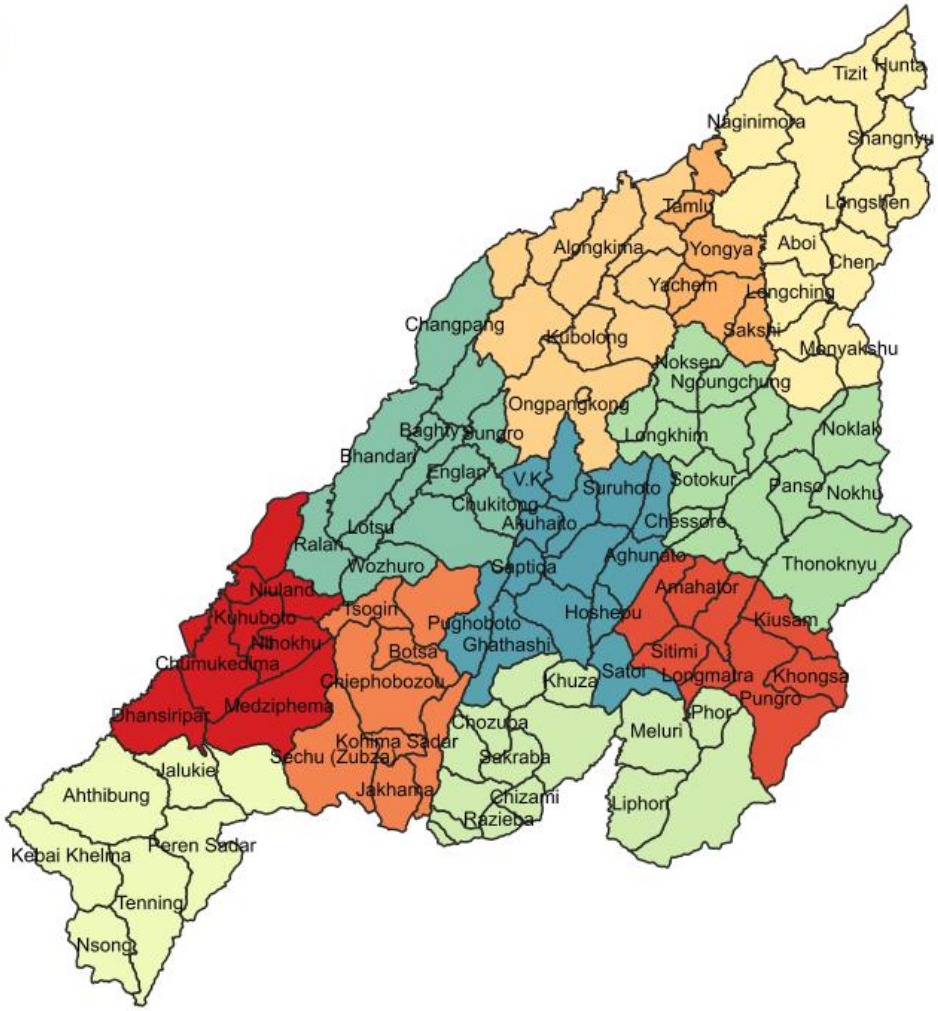


Understanding a WMS Service

Understanding a WMS Service: ‘GetMap’ Request

The request
The following URL is a WMS 'GetMap' request:

http://localhost:8080/geoserver/naga/wms?service=WMS&version=1.1.0&request=GetMap&layers=naga:Nag_Circle_UTM&styles=&bbox=532682.0670999996,2787067.8539000005,722847.2251000003,2992434.9676000006&width=711&height=768&srs=EPSG:32646&format=image/png&TRANSPARENT=TRUE



Understanding a WMS Service: ‘GetMap’ Request

Breaking down the request

That big, long URL is actually made up of many small bits, separated by '**&**' characters. Here is the request, broken up so that each bit is on its own line, and with bits re-arranged so they flow better:

http://localhost:8080/geoserver/naga/wms?
service=WMS&
version=1.1.0&
request=GetMap&
layers=naga:Nag_Circle_UTM&
styles=&
bbox=532682.0670999996,2787067.8539000005,722847.2251000003,2992434.9676000006&
width=711&
height=768&
srs=EPSG:32646&
format=image/png
TRANSPARENT=TRUE

Understanding a WMS Service: ‘GetMap’ Request

http://localhost:8080/geoserver/naga/wms?
The protocol, host and path

service=WMS&
The *SERVICE* parameter tells the server which exact service you're sending your message to. In some cases, the service endpoint might work for multiple services, and this parameter could be used to specify whether you're sending your 'GetMap' request to the WMS or the WFS.

version=1.1.0&
The 'VERSION' parameter

request=GetMap&
The *REQUEST* parameter tells the server which operation you'd like to perform. The **'GetMap' request tells the server you want to fetch a map image**. Other request types include 'GetLegendImage', 'GetFeatureInfo', 'GetCapabilities, ‘GetLegendGraphic’

Understanding a WMS Service: ‘GetMap’ Request

layers=nesac:Nag_Circle_UTM&
The *LAYERS* parameter **lists** for the WMS, the **exact layers you wish to have drawn**, as a comma-separated list. The order in which you list the layers is the //order in which they're drawn//. So layers *listed first* are *drawn first*.

styles=&
The *STYLES* parameter tells the server //how// to draw the layers you've specified in the *LAYERS* parameter.

bbox=532682.0670999996,2787067.8539000005,722847.2251000003,2992434.9676000006&
The *BBOX* parameter tells the WMS what spatial extent to use for this map. It is specified as four ordered spatial points: //minx,miny,maxx,maxy//

width=711&height=768&
The *WIDTH* and *HEIGHT* parameters specify the width and height of the created image

Understanding a WMS Service: ‘GetMap’ Request

srs=EPSG:32646&
The *SRS* parameter tells the WMS which coordinate system the *BBOX* parameter is expressed in. The SRS parameter is written as an EPSG code.

format=image/png
The *FORMAT* parameter specifies the format of the returned image

TRANSPARENT=TRUE
The *TRANSPARENT* parameter specifies whether areas of the map which are not otherwise drawn should be marked as transparent in the response image

Other Parameters:
http://www.opengeospatial.org/standards/wms

GeoServer CQL/ECQL

GeoServer CQL/ECQL

- CQL (Common Query Language) is a **query language created by the OGC** for the Catalogue Web Services specification. CQL is written using a text-based syntax. It is thus more readable and better-suited for manual authoring.

- Common Query Language or CQL can be a **shorter** (as compared to other filters), more readable way to put a "filter" or SQL-like "where" statement into a URL.

- ECQL is an extension of CQL and thus provides a more flexible language with stronger similarities with SQL.

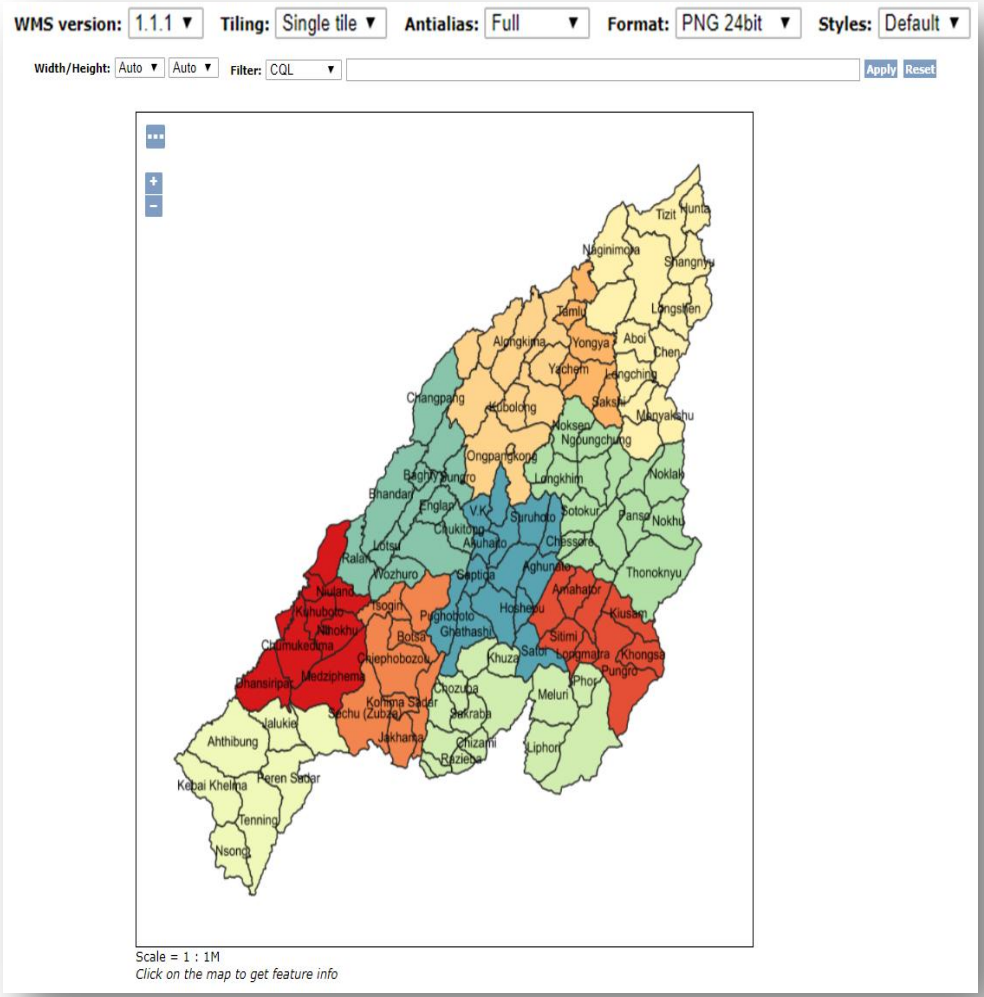
- GeoServer supports the use of both CQL and ECQL in WMS and WFS requests, as well as in GeoServer's SLD dynamic symbolizers. Requests can contain attribute statements or spatial requests.

Use **cql_filter** to define filter parameters in your application (OL/Leaflet)

GeoServer CQL/ECQL :Opening and Using CQL Filter box

The following examples use the **naga:Circle Boundary of Nagaland** . It demonstrates how CQL filters work by using the WMS CQL_FILTER vendor parameter to alter the data displayed by WMS requests.

- Open the GeoServer Map Preview for the **naga:Circle Boundary of Nagaland** layer.
- Click on the *Options* button at the top of the map preview to open the advanced options toolbar.
- The example filters can be entered in the *Filter: CQL* box



Attributes Names

Circle Boundary of Nagaland

fid	OBJECTID_1	OBJECTID	Id	District_N	Circle_N	Shape_Leng	Shape_Le_1	Shape_Area

GeoServer CQL/ECQL :Comparisons

Layer : Load Nagaland block boundary in Geoserver

Simple comparisons
The full list of comparison operators is: **=, <>, >, >=, <, <=**

CQL query:

```
District_N ='Zunheboto'
Circle_N LIKE 'A%'
Shape_Area > 300
```

list comparisons
If instead we want to extract the states whose name is in a given list we can use the IN operator specifying an attribute name, as in

```
Circle_N IN ('Meluri','Liphori','Lotsu')
```

GeoServer CQL/ECQL :Filter

Filter functions

CQL/ECQL can use any of the filter functions available in GeoServer. This greatly increases the power of CQL expressions.

For example, suppose we want to find all circles whose name contains an “a”, regardless of letter case. We can use the `strToLowerCase` to turn all the circle names to lowercase and then use a like comparison: `strToLowerCase(Circle_N)` like `'%a%'`

```
strToLowerCase(Circle_N) LIKE '%a%'
```

GeoServer CQL/ECQL : Geometric Filter

Geometric filters
CQL provides a full set of geometric filter capabilities. Say, for example, you want to display only the district boundary that intersect the (x,x,x,x) bounding box
Check the current BBOX values from URL and change the bbox and supply it to BBOX query

```
BBOX(the_geom,
532682.0670999996,2787067.8539000005,722847.2251000003,2992434.9676000006)
```

The full list of geometric predicates
is: `EQUALS, DISJOINT, INTERSECTS, TOUCHES, CROSSES, WITHIN, CONTAINS, OVERLAPS, RELATE, DWITHIN, BEYOND`

Eg Using CQL filter for WFS GetFeature as URL.

```
http://host:port/geoserver/wfs?service=WFS&version=1.0&request=GetFeature&typeName=myLayer&CQL_FILTER=INTERSECTS(the_geom,
POLYGON((...)))&propertyName=data1,data2,data3
```

More Ref : https://docs.geoserver.org/2.7.1/user/filter/ecql_reference.html#spatial-predicate