GeoNode Developers Workshop Documentation

Release 2.0

GeoNode

February 17, 2013

CONTENTS

Welcome to the GeoNode Developers Workshop! This workshop will teach how to develop with and for the GeoNode software application.

Introduction to GeoNode development Learn about GeoNode's core components, its Architecture, the tools it is developed with and the standards it supports.

Development Prerequsites and Core Modules Learn about the pre-requisites you will need in order to develop with GeoNode. Take a look at its core modules and how they work together to provide a complete web mapping tool.

Customized GeoNode Projects Learn how existing projects leverage GeoNode and create your own GeoNode based project.

Loading Data into a GeoNode Learn how to load data into a GeoNode with GeoServer, on the command line or programatically with scripts.

GeoNode debugging techniques Learn how to debug GeoNode instances and projects.

Setting up a GeoNode development environment Learn how to set up a GeoNode development environment so you can contribute to GeoNode's core.

GeoNode APIs Learn about the APIs GeoNode leverages and provides.

GeoNode's development process Learn about GeoNode's development process and how to work with the GeoNode community.

CHAPTER

ONE

INTRODUCTION TO GEONODE DEVELOPMENT

This module will introduce you to the components that GeoNode is built with, the standards that it supports and the services it provides based on those standards, and an overview its architecture.

GeoNode is a web based GIS tool, and as such, in order to do development on GeoNode itself or to integrate it into your own application, you should be familiar with basic web development concepts as well as with general GIS concepts.

A set of reference links on these topics is included at the end of this module.

1.1 GeoNode Components

GeoNode's architecture is based on a set of core tools and libraries that provide the building blocks on which the application is built. Having a basic understanding of each of these components is critical to your success as developer working with GeoNode.

Lets look at each of these components and discuss how they are used within the GeoNode application.

1.1.1 Django

GeoNode is based on Django which is a high level Python web development framework that encourages rapid development and clean pragmatic design. Django is based on the Model View Controller (MVC) architecture pattern, and as such, GeoNode models layers, maps and other modules with Django's Model module and and these models are used via Django's ORM in views which contain the business logic of the GeoNode application and are used to drive HTML templates to display the web pages within the application.

1.1.2 GeoServer

GeoServer is a an open source software server written in Java that provides OGC compliant services which publish data from many spatial data sources. GeoServer is used as the core GIS component inside GeoNode and is used to render the layers in a GeoNode instance, create map tiles from the layers, provide for downloading those layers in various formats and to allow for transactional editing of those layers.

1.1.3 GeoExplorer

GeoExplorer is a web application, based on the GeoExt framework, for composing and publishing web maps with OGC and other web based GIS Services. GeoExplorer is used inside GeoNode to provide many of the GIS and

cartography functions that are a core part of the application.

1.1.4 PostgreSQL and PostGIS

PostgreSQL and PostGIS are the database components that store and manage spatial data and information for GeoNode and the django modules that it is composed of, pycsw and GeoServer. All of these tables and data are stored within a geonode database in PostgreSQL. GeoServer uses PostGIS to store and manage spatial vector data for each layer which are stored as a separate table in the database.

1.1.5 pycsw

pycsw is an OGC CSW server implementation written in Python. GeoNode uses pycsw to provide an OGC compliant standards-based CSW metadata and catalogue component of spatial data infrastructures, supporting popular geospatial metadata standards such as Dublin Core, ISO 19115, FGDC and DIF.

1.1.6 Geospatial Python Libraries

GeoNode leverages several geospatial python libraries including gsconfig and OWSLib. gsconfig is used to communicates with GeoServer's REST Configuration API to configure GeoNode layers in GeoServer. OWSLib is used to communicate with GeoServer's OGC services and can be used to communicate with other OGC services.

1.1.7 Django Pluggables

GeoNode uses a set of Django plugins which are usually referred to as pluggables. Each of these pluggables provides a particular set of functionality inside the application from things like Registration and Profiles to interactivity with external sites. Being based on Django enables GeoNode to take advantage of the large ecosystem of these pluggables out there, and while a specific set is included in GeoNode itself, many more are available for use in applications based on GeoNode.

1.1.8 jQuery

jQuery is a feature-rich javascript library that is used within GeoNode to provide an interactive and responsive user interface as part of the application. GeoNode uses several jQuery plugins to provide specific pieces of functionality, and the GeoNode development team often adds new features to the interface by adding additional plugins.

1.1.9 Bootstrap

Bootstrap is a front-end framework for laying out and styling the pages that make up the GeoNode application. It is designed to ensure that the pages render and look and behave the same across all browsers. GeoNode customizes bootstraps default style and its relatively easy for developers to customize their own GeoNode based site using existing Boostrap themes or by customizing the styles directly.

1.2 Standards

GeoNode is based on a set of Open Geospatial Consortium (OGC) standards. These standards enable GeoNode installations to be interoperable with a wide variety of tools that support these OGC standards and enable federation

with other OGC compliant services and infrastructure. Reference links about these standards are also included at the end of this module.

GeoNode is also based on Web Standards ...

1.2.1 Open Geospatial Consortium (OGC) Standards

Web Map Service (WMS)

The Web Map Service (WMS) specification defines an interface for requesting rendered map images across the web. It is used within GeoNode to display maps in the pages of the site and in the GeoExplorer application to display rendered layers based on default or custom styles.

Web Feature Service (WFS)

The Web Feature Service (WFS) specification defines an interface for reading and writing geographic features across the web. It is used within GeoNode to enable downloading of vector layers in various formats and within GeoExplorer to enable editing of Vector Layers that are stored in a GeoNode.

Web Coverage Service (WCS)

The Web Coverage Service (WCS) specification defines an interface for reading and writing geospatial raster data as "coverages" across the web. It is used within GeoNode to enable downloading of raster layers in various formats.

Catalogue Service for Web (CSW)

The Catalogue Service for Web (CSW) specification defines an interface for exposing a catalogue of geospatial metadata across the web. It is used within GeoNode to enable any application to search GeoNode's catalogue or to provide federated search that includes a set of GeoNode layers within another application.

Tile Mapping Service (TMS/WMTS)

The Tile Mapping Service (TMS) specification defines and interface for retrieving rendered map tiles over the web. It is used within geonode to enable serving of a cache of rendered layers to be included in GeoNode's web pages or within the GeoExplorer mapping application. Its purpose is to improve performance on the client vs asking the WMS for rendered images directly.

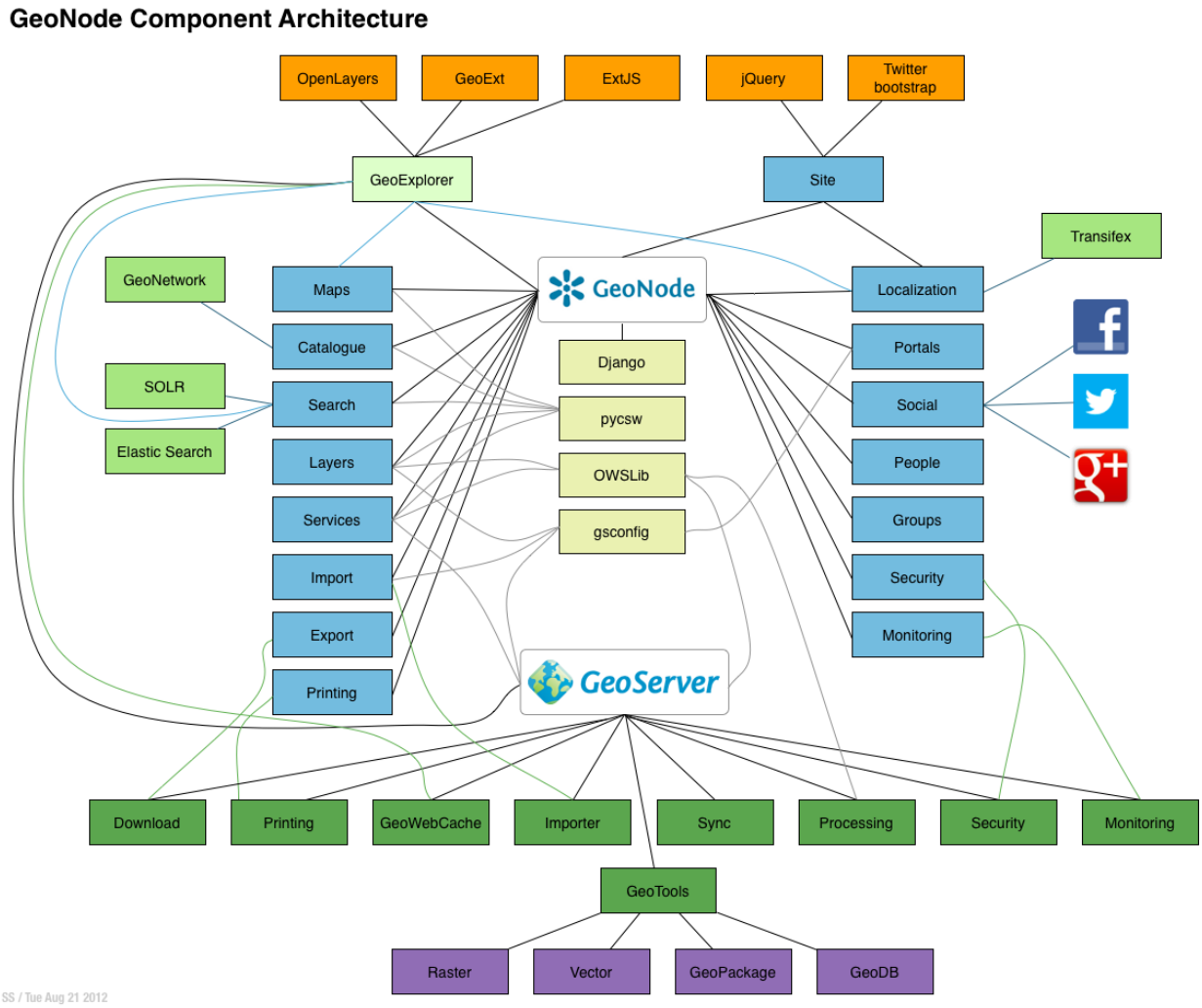


Figure 1.1: GeoNode Component Architecture

1.2.2 Web Standards

HTML

CSS

REST

1.3 GeoNode Architecture

1.3.1 The Big Picture

1.3.2 Django Architecture

MVC/MVT

MVC Model, View, Controller

MVT Model, View, Template

- Model represents application data and provides rich ORM functionality
- Views are a rendering of a Model most often using the Django template engine
- In Django, the controller part of this commonly discussed, layered architecture is a subject of discussion. According to the standard definition, the controller is the layer or component through which the user interacts and model changes occur.

More: http://reinout.vanrees.org/weblog/2011/12/13/django-mvc-explanation.html

WSGI

WSGI Web Server Gateway Interface (whis-gey)

- This is a python specification for supporting a common interface between all of the various web frameworks and an application (Apache, for example) that is 'serving'.
- This allows any WSGI compliant framework to be hosted in any WSGI compliant server.
- For most GeoNode development, the details of this specification may be ignored.

More: http://en.wikipedia.org/wiki/Wsgi

1.3.3 GeoNode and GeoServer

GeoNode uses GeoServer for providing OGC services.

- GeoNode configures GeoServer via the REST API
- GeoNode retreives and caches spatial information from GeoServer. This includes relevant OGC service links, spatial metadata, and attribute information.

  In summary, GeoServer contains the layer's data and GeoNode's layer model extends what metadata is present in GeoServer with it's own.
- GeoNode can discover existing layers published in a GeoServer via the WMS capabilities document

- GeoServer delegates authentication and authorization to GeoNode
- Data uploaded to GeoNode is first processed in GeoNode and finally published to GeoServer (or ingested into the spatial database)

1.3.4 GeoNode and PostgreSQL/PostGIS

In production, GeoNode is configured to use PostgreSQL/PostGIS for it's persistent store. In development and testing mode, often an embedded sqlite database is used. The latter is not suggested for production.

- The database stores configuration and application information. This includes users, layers, maps, etc.
- It is recommended that GeoNode be configured to use PostgresSQL/PostGIS for storing vector data as well. While serving layers directly from shapefile allows for adequate performance in many cases, storing features in the database allows for better performance especially when using complex style rules based on attributes.

1.3.5 GeoNode and pycsw

GeoNode is built with pycsw embedded as the default CSW server component.

Publishing

Since pycsw is embedded in GeoNode, layers published within GeoNode are automatically published to pycsw and discoverable via CSW. No additional configuration or actions are required to publish layers, maps or documents to pycsw.

Discovery

GeoNode's CSW endpoint is deployed available at `http://localhost:8000/catalogue/csw` and is available for clients to use for standards-based discovery. See http://pycsw.org/docs/tools.html for a list of CSW clients and tools.

1.3.6 Javascript in GeoNode

- GeoExplorer runs in the browser and talks with GeoNode and GeoServer's APIs using AJAX.
- jQuery is used for incremental enhancement of many GeoNode HTML interfaces.

1.4 Exercises

1.4.1 Components and Services

Hint, if `bash-completion` is installed, try <TAB><TAB> to get completions.

1. start/stop services

```
$ sudo service apache2
$ sudo service apache2 reload
$ sudo service tomcat7
$ sudo service postgresql
```

1. basic psql interactions

```
$ sudo su - postgres
$ psql
=> help                 # get help
=> \?                   # psql specific commands
=> \l                   # list databases
=> \c geonode           # switch database
=> \ds                  # list tables
=> \dS layers_layer     # describe table
```

1.4.2 OGC Standards

WMS

1. Use the layer preview functionality in GeoServer to bring up a web map.
2. Copy a the URL for the image in the map.
3. Alter URL parameters for the request.
4. Use *curl* to get the capabilities document

```
$ curl 'http://localhost/geoserver/wms?request=getcapabilities'
```

More: http://docs.geoserver.org/stable/en/user/services/wms/index.html

WFS

1. Describe a feature type using curl (replace ws:name with your layer)

```
$ curl 'http://localhost/geoserver/wfs?request=describefeaturetype&name=ws:name
```

More: http://docs.geoserver.org/stable/en/user/services/wfs/reference.html

1.5 Development References

1.5.1 Basic Web based GIS Concepts and Background

- OGC Services
  - http://www.opengeospatial.org/
  - http://en.wikipedia.org/wiki/Open_Geospatial_Consortium
- Web Application Architecture
  - http://en.wikipedia.org/wiki/Web_application
  - http://www.w3.org/2001/tag/2010/05/WebApps.html
  - http://www.amazon.com/Web-Application-Architecture-Principles-Protocols/dp/047051860X
- AJAX and REST
  - http://en.wikipedia.org/wiki/Ajax_(programming)
  - http://en.wikipedia.org/wiki/Representational_state_transfer

- OpenGeo Suite
  - http://workshops.opengeo.org/suiteintro/
  - http://suite.opengeo.org/opengeo-docs/
- GeoServer Administration
  - http://suite.opengeo.org/opengeo-docs/geoserver/
  - https://docs.google.com/a/opengeo.org/presentation/d/15fvUDYg0TO6WGFQlMLM2J1qiTVBYpfjCp0aQBDT0GrM/edit#
  - http://suite.opengeo.org/docs/sysadmin/index.html#sysadmin
- PostgreSQL and PostGIS Administration - http://workshops.opengeo.org/postgis-intro/ - http://workshops.opengeo.org/postgis-spatialdbtips/

1.5.2 Core development tools and libraries

- python
  - http://docs.python.org/2/tutorial/
  - http://www.learnpython.org/
  - http://learnpythonthehardway.org/book/
- django
  - https://docs.djangoproject.com/en/dev/intro/tutorial01/
  - https://code.djangoproject.com/wiki/Tutorials
- javascript
  - http://www.crockford.com/javascript/inheritance.html
  - http://geoext.org/tutorials/quickstart.html
- jquery
  - http://www.w3schools.com/jquery/default.asp
  - http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery
  - http://www.jquery-tutorial.net/
- bootstrap
  - http://twitter.github.com/bootstrap/
  - http://www.w3resource.com/twitter-bootstrap/tutorial.php
- geotools/geoscript/geoserver
  - http://docs.geotools.org/stable/tutorials/feature/csv2shp.html
  - http://geoscript.org/tutorials/index.html
  - http://docs.geotools.org/stable/tutorials/
  - https://github.com/dwins/gsconfig.py/blob/master/README.rst
- geopython
  - http://pycsw.org/docs/documentation.html
  - http://geopython.github.com/OWSLib/

  - https://github.com/toblerity/shapely
  - https://github.com/sgillies/Fiona
  - http://pypi.python.org/pypi/pyproj
- gdal/ogr
  - http://www.gdal.org/gdal_utilities.html
  - http://www.gdal.org/ogr_utilities.html

CHAPTER

TWO

DEVELOPMENT PREREQUSITES AND CORE MODULES

This module will introduce you to the

2.1 GeoNode's Development Prerequisites

2.1.1 Basic Shell Tools

ssh and sudo

ssh and sudo are very basic terminal skills which you will need to deploy, maintain and develop with GeoNode. If you are not already familiar with their usage, you should review the basic descriptions below and follow the external links to learn more about how to use them effectively as part of your development workflow.

ssh is the network protocol used to connect to a remote server where you run your GeoNode instance whether on your own network or on the cloud. You will need to know how to use an the ssh command from the terminal on your unix machine or how to use a ssh client like putty or winscp on windows. You may need to use pki certificates to connect to your remove server, and should be familiar with the steps and options necessary to connect this way. More information about ssh can be found in the links below.

- http://winscp.net/eng/docs/ssh

sudo is the command used to execute a terminal command as the superuser when you are logged in with a normal user. You will to use sudo in order to start, stop and restart key services on your GeoNode instance. If you are not able to grant yourself these privileges on the machine you are using for your GeoNode instance, you may need to consult with your network administrator to arrange for your user to be granted sudo permissions. More information about sudo can be found in the links below.

- http://en.wikipedia.org/wiki/Sudo

bash

Bash is the most common unix shell which will usually be the default on servers where you will be deploying your GeoNode instance. You should be familiar with the most common bash commands in order to be able to deploy, maintain and modify a geonode instance. More information about Bash and common bash commands can be found in the links below.

- http://en.wikipedia.org/wiki/Bash_(Unix_shell)

apt

apt is the packaging tool that is used to install GeoNode on ubuntu and other debian based systems. You will need to be familiar with adding Personal Package Archives to your list of install sources, and will need to be familiar with basic apt commands. More information about apt can be found in the links below.

- http://en.wikipedia.org/wiki/Advanced_Packaging_Tool

2.1.2 Python Development Tools

The GeoNode development process relies on several widely used python development tools in order to make things easier for developers and other users of the systems that GeoNode developers work on or where GeoNodes are deployed. They are considered best practices for modern python development, and you should become familiar with these basic tools and be comfortable using them on your own projects and systems.

virtualenv

virtualenv is a tool used to create isolated python development environments such that the the versions of project dependencies are sandboxed from the system-wide python packages. This eliminates the commonly encountered problem of different projects on the same system using different versions of the same library. You should be familiar with how to create and activate virtual environments for the projects you work on. More information about virtualenv can be found in the links below.

- http://pypi.python.org/pypi/virtualenv
- http://www.virtualenv.org/en/latest/

virtualenvwrapper is a wrapper around the virtualenv package that makes it easier to create and switch between virtual environments as you do development. Using it will make your life much easier, so its recommended that you install and configure it and use its commands as part of your virtualenv workflow. More info about virtualenvwrapper can be found in the links below.

- http://www.doughellmann.com/projects/virtualenvwrapper/

pip

pip is a tool for installing and managing python packages. Specifically it is used to install and upgrade packages found in the Python Pacakge Index. GeoNode uses pip to install itself, and to manage all of the python dependencies that are needed as part of a GeoNode instance. As you learn to add new modules to your geonode, you will need to become familiar with the use of pip and about basic python packaging usage. More information about pip can be found in the links below.

- http://www.pip-installer.org/en/latest/
- http://pypi.python.org/pypi/pip
- http://en.wikipedia.org/wiki/Pip_(Python)

miscellaneous

ipython is a set of tools to make your python development and debugging experience easier. The primary tool you want to use is an interactive shell that adds introspection, integrated help and command completion and more. While not strictly required to do GeoNode development, learning how to use ipython will make your development more productive and pleasant. More information about ipython can be found in the links below.

- http://ipython.org/
- http://pypi.python.org/pypi/ipython
- https://github.com/ipython/ipython
- http://en.wikipedia.org/wiki/IPython

pdb is a standard python module that is used to interactively debug your python code. It supports setting conditional breakpoints so you can step through the code line by line and inspect your variables and perform arbitrary execution of statements. Learning how to effectively use pdb will make the process of debugging your application code significantly easier. More information about pdb can be found in the links below.

- http://docs.python.org/2/library/pdb.html

2.1.3 Django

GeoNode is built on top of the *Django web framework*, and as such, you will need to become generally familiar with Django itself in order to become a productive GeoNode developer. Django has excellent documentation, and you should familiarize yourself with Django by following the Django workshop and reading through its documentation as required.

Model Template View

Django is based on the Model Template View paradigm (more commonly called Model View Controller). Models are used to define objects that you use in your application and Django's ORM is used to map these models to a database. Views are used to implement the business logic of your application and provide objects and other context for the templates. Templates are used to render the context from views into a page for display to the user. You should become familiar with this common paradigm used in most modern web frameworks, and how it is specifically implemented and used in Django. The Django tutorial itself is a great place to start. More information about MTV in Django can be found in the links below.

- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html
- https://docs.djangoproject.com/en/1.4/

HTTP Request Response

Django and all other web frameworks are based on the HTTP Request Response cycle. Requests come in to the server from remote clients which are primarily web browsers, but also can be api clients, and the server returns with a Response. You should be familiar with these very basic HTTP principles and become familiar with the way that Django implements them. More information about HTTP, Requests and Responses and Djangos implementation in the links below.

- http://devhub.fm/http-requestresponse-basics/
- http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://docs.djangoproject.com/en/dev/ref/request-response/

Management Commands

Django projects have access to a set of management commands that are used to manage your project. Django itself provides a set of these commands, and django apps (including GeoNode) can provide their own. Management commands are used to do things like synchronize your models with your database, load data from fixtures or back up your

database with fixtures, start the development server, initiate the debugger and many other things. GeoNode provides management commands for synchronizing with a GeoServer or updating the layers already in your GeoNode. You should become familiar with the basic management commands that come with Django, and specifically with the commands that are part of GeoNode. The GeoNode specific commands are covered in section. More information about management commands can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/django-admin/

Django Admin Interface

Django provides a build-in management console that administrators and developers can use to look at the data in the database that is part of the installed applications. Administrators can use this console to perform many common administration tasks that are a necessary part of running a GeoNode instance, and as a developer, you will use this interface during your development process to inspect the database and the data stored in your models. More information about the django admin interface can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/contrib/admin/

Template Tags

Django templates make use of a set of tags to inject, filte and format content into a rendered HTML page. Django itself includes a set of built-in template tags and filters that you will use in your own templates, and GeoNode provides a geonode specific set of tags that are used in the GeoNode templates. You should become familiar with the built-in tag set and with GeoNode's specific tags as you work on developing your own templates or extending from GeoNode's. More information about Django template tags can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/templates/builtins/

2.2 GeoNode's Core Modules

GeoNode is made up of a set of core Django pluggable modules (known as apps in Django) that provide the functionality of the application. Together they make up the key components of a GeoNode site. While your own use case and implementation may not require that you work directly on these modles, it is important that you become familiar with their layout, structure and the functionality that they provide. You may need to import these apps into your own apps, and as such, becoming familiar with them is an important step in becoming a proficient GeoNode developer.

2.2.1 geonode.layers

geonode.layers is the most key GeoNode module. It is used to represent layers of data stored in a GeoNode's paired GeoServer. The layer model class inherits fields from the ResourceBase class which provides all of the fields necessary for the metadata catalogue, and adds fields that map the object to its corresponding layer in GeoServer. When your users upload a layer via the user interface, the layer is imported to GeoServer and a record is added to GeoNode's database to represent that GeoServer layer within GeoNode itself.

The Layer model class provides a set of helper methods that are used to perform operations on a Layer object, and also to return things such as the list of Download or Metadata links for that layer. Additional classes are used to model the layers Attributes, Styles, Contacts and Links. The Django signals framework is used to invoke specific functions to synchronize with GeoServer before and after the layer is saved.

The views in the layers app are used to perform functions such as uploading, replacing, removing or changing the points of contact for a layer, and views are also used to update layer styles, download layers in bulk or change a layers permissions.

The forms module in the layer app is used to drive the user interface forms necessary for performing the business logic that the views provide.

The Layers app also includes a set of templates that are paired with views and used to drive the user interface. A small set of layer template tags is also used to help drive the layer explore and search pages.

Some helper modules such as geonode.layers.metadata and geonode.layers.ows are used by the layer views to perform specific functions and help keep the main views module more concise and legible.

Additionally, the GeoNode specific management commands are a part of the geonode.layers app.

You should spend some time to review the layers app through GitHubs code browsing interface.

https://github.com/GeoNode/geonode/tree/master/geonode/layers

2.2.2 geonode.maps

The geonode.maps app is used to group together GeoNodes multi layer map functionality. The Map and MapLayer objects are used to model and implement maps created with the GeoExplorer application. The Map object also extends from the ResourceBase class which provides the ability to manage a full set of metadata fields for a Map.

The views in the maps app perform many of the same functions as the views in the layers app such as adding, changing, replacing or removing a map and also provide the endpoints for returning the map configuration from the database that is used to initialize the GeoExplorer app.

The maps app also includes a set of forms, customization of the Django admin, some utility functions and a set of templates and template tags.

You can familiarize yourself with the maps app on GitHub.

https://github.com/GeoNode/geonode/tree/master/geonode/layers

2.2.3 geonode.security

The geonode.security app is used to provide object level permissions within the GeoNode Django application. It is a custom Django authentication backend and is used to assign Generic, User and Group Permissions to Layers, Maps and other objects in the GeoNode system. Generic permissions are used to enable public anonymous or authenticated viewing and/or editing of your data layers and maps, and User and Group specific permissions are used to allow specific users or groups to access and edit your layers.

2.2.4 geonode.search

The geonode.search module provides the search API that is used to drive the GeoNode search pages. It is configured to index layers, maps, documents and profiles, but is extensible to allow you to use it to index your own model classes. This module is currently based on the Django ORM and as such has a limited set of search features, but the GeoNode development team is actively working on making it possible to use this module with more feature-rich search engines.

2.2.5 geonode.catalogue

The geonode.catalogue app provides a key set of metadata catalogue functions within GeoNode itself. GeoNode is configured to use an integrated version of the pycsw library to perform these functions, but can also be configured to use any OGC compliant CS-W implementation such as GeoNetwork or Deegree. The metadata app allows users to import and/or edit metadata for their layers, maps and documents, and it provides an OGC compliant search interface for use in federating with other systems.

2.2.6 geonode.geoserver

The geonode.geoserver module is used to interact with GeoServer from within GeoNode's python code. It relies heavily on the gsconfig library which addresses GeoServer's REST configuration API. Additionally, the geonode.geoserver.uploader module is used to interact with GeoServers Importer API for uploading and configuring layers.

2.2.7 geonode.people

The geonode.people module is used to model and store information about both GeoNode users and people outside of the system who are listed as Points of Contact for particular layers. It is the foundational module for GeoNode's social features. It provides a set of forms for users to edit and manage their own profiles as well as to view and interact with the profiles of other users.

2.2.8 geoexplorer

GeoNode's core GIS client functions are performed by GeoExplorer. The GeoExplorer app is in turn based on GeoExt, OpenLayers and ExtJS. It provides functionality for constructing maps, styling layers and connecting to remote services. GeoExplorer is the reference implementation of the OpenGeo Suite SDK which is based on GXP. GeoNode treats GeoExplorer as an external module that is used out of the box in GeoNode, but it is possible for you to create your own Suite SDK app and integrate it with GeoNode.

2.2.9 Static Site

The front end of GeoNode is composed of a set of core templates, specific templates for each module, cascading style sheets to style those pages and a set of javascript modules that provide the interactive functionality in the site.

Templates

GeoNode includes a basic set of core templates that use Django's template inheritance system to provide a modular system for constructing the web pages in GeoNode's interface. These core templates drive the overall page layout and things like the home page. You will start the process of customizing your GeoNode instance by overriding these templates, so you should familiarize yourself with their tructure and how they inherit from each other to drive the pages.

Additionally, most of the apps described above have their own set of templates that are used to drive the pages for each module. You may also want to override these templates for your own purposes and as such should familiarize yourself with a few of the key ones.

CSS

GeoNode's css is based on Twitter's Bootstrap Library which uses the lessc dynamic stylesheet language. GeoNode extends from the basic Boostrap style and you are able to create your own bootstrap based style to customize the look and feel of your own GeoNode instance. Sites like bootswatch.com also provide ready made styles that you can simply drop in to your project to change the style.

Javascript

The interactive functionality in GeoNode pages is provided by the jQuery javascript framework and a set of jQuery plugins. The core set of GeoNode javascript modules closely aligns with the apps described above, and there are also

a few pieces of functionality provided as javascript modules that are used through out all of the apps. You are able to add your own jQuery code and/or plugins to perform interactive functionality in your own application.

2.3 Exercises

2.3.1 Shell and Utilities

1. *ssh* into your virtual machine or other instance
2. *sudo* to modify the *sshd_config* settings to verify disabling of dns resolution (UseDNS=no)
3. install a command line helper

```
$ sudo apt-get install bash-completion
```

1. exercise command completion

```
$ apt-get install <TAB><TAB>
```

1. activate/deactivate the *virtualenv* on your instance

```
$ source /var/lib/geonode/bin/activate
$ deactivate
```

1. set the *DJANGO_SETTINGS_MODULE* env variable

```
$ export DJANGO_SETTINGS_MODULE=geonode.settings
```

1. install the *httpie* utility via pip

```
$ pip install httpie
$ http http://localhost/geoserver/rest
$ http -a admin http://localhost/geoserver/rest
<type in password - geoserver>
```

2.3.2 Python

1. launch *ipython* and experiment

```
> x = "some text"
> x.<TAB><TAB>
> x.split.__doc__
> ?
```

1. execute a script with *ipython* and open the REPL

```
$ echo "twos = [ x*2 for x in range(5)]" > test.py
$ ipython -i test.py
> twos
```

CHAPTER

THREE

CUSTOMIZED GEONODE PROJECTS

This module will teach you about how to set up and customize your own GeoNode-based project by changing the theme, adding additional modules, and integrating with other systems. When complete, you should understand how Downstream GeoNode projects work, and how to set up a project of your own.

3.1 Introduction to GeoNode Projects

GeoNode enables you to set up a complete site simply by installing the packages and adding your data. If you want to create your own project based on GeoNode, there are a several options available that enable you to customize the look and feel of your GeoNode site. You can add additional modules that are necessary for your own use case and to integrate your GeoNode project with other external sites and services.

This module assumes that you have installed a GeoNode site with the Ubuntu Packages and that you have a working GeoNode based on that setup. If you want to follow this same methodology on a different platform, you can follow this module and adapt as necessary for your environment.

3.1.1 Overview

GeoNode is an out-of-the-box, full-featured Spatial Data Infrastructure solution, but many GeoNode implementations require either customization of the default site or the use of additional modules, whether they be third-party Django Pluggables or modules developed by a GeoNode implementer.

There are quite a few existing Downstream GeoNode projects some of which follow the methodology described in this module. You should familiarize yourself with these projects and how and why they extend GeoNode. You should also carefully think about what customization and additional modules you need for your own GeoNode-based project and research the options that are available to you. The Django Packages site is a great place to start looking for existing modules that may meet your needs.

3.1.2 Existing downstream GeoNode projects

- Harvard Worldmap
- MapStory
- Risiko/SAFE

3.1.3 Django template projects

GeoNode follows the Django template projects paradigm introduced in Django 1.4. At a minimum, a Django project consists of a `settings.py` file and a `urls.py` file; Django apps are used to add specific pieces of functionality. The GeoNode development team has created a template project which contains these required files with all the GeoNode configuration you need to get up and running with your own GeoNode project. If you would like learn more about Django projects and apps, you should consult the Django Documentation

3.2 Setting up your GeoNode project

This section will walk you through the steps necessary to set up your own GeoNode project. It assumes that you have installed GeoNode from the Ubuntu packages and that you have a working GeoNode site.

3.2.1 Setup steps

If you are working remotely, you should first connect to the machine that has your GeoNode installation. You will need to perform the following steps in a directory where you intend to keep your newly created project.

1. Activate GeoNode's Virtual Environment:

   ```
   $ source /var/lib/geonode/bin/activate
   ```

2. Create your GeoNode project from the template:

   ```
   $ django-admin.py startproject --template=https://github.com/GeoNode/geonode-project/zipball/mas
   $ cd my_geonode
   ```

3. Update your `local_settings.py`. You will need to check the `local_settings.py` that is included with the template project and be sure that it reflects your own local environment. You should pay particular attention to the Database settings especially if you intend to reuse the database that was set up with your base GeoNode installation.

4. Synchronize your database:

   ```
   $ python manage.py syncdb --all
   ```

5. Run the test server:

   ```
   $ python manage.py runserver
   ```

6. Visit your new GeoNode site at http://localhost:8000.

3.2.2 Source code revision control

It is recommended that you immediately put your new project under source code revision control. The GeoNode development team uses Git and GitHub and recommends that you do the same. If you do not already have a GitHub account, you can easily set one up. A full review of Git and distributed source code revision control systems is beyond the scope of this tutorial, but you may find the Git Book useful if you are not already familiar with these concepts.

1. Create a new repository in GitHub. You should use the GitHub user interface to create a new repository for your new project.

2. Initialize your own repository:

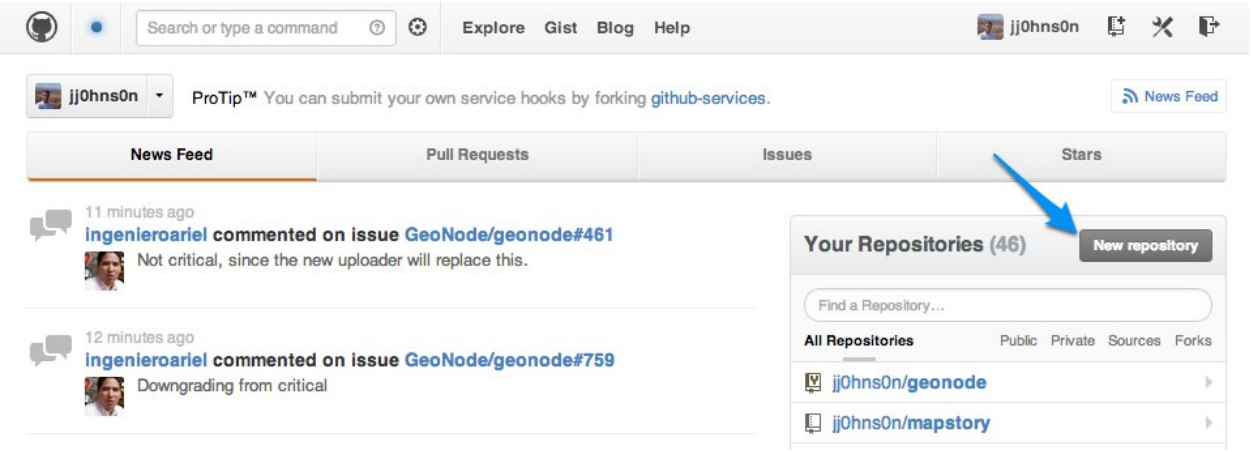


Figure 3.1: *Creating a new GitHub Repository From GitHub's Homepage*

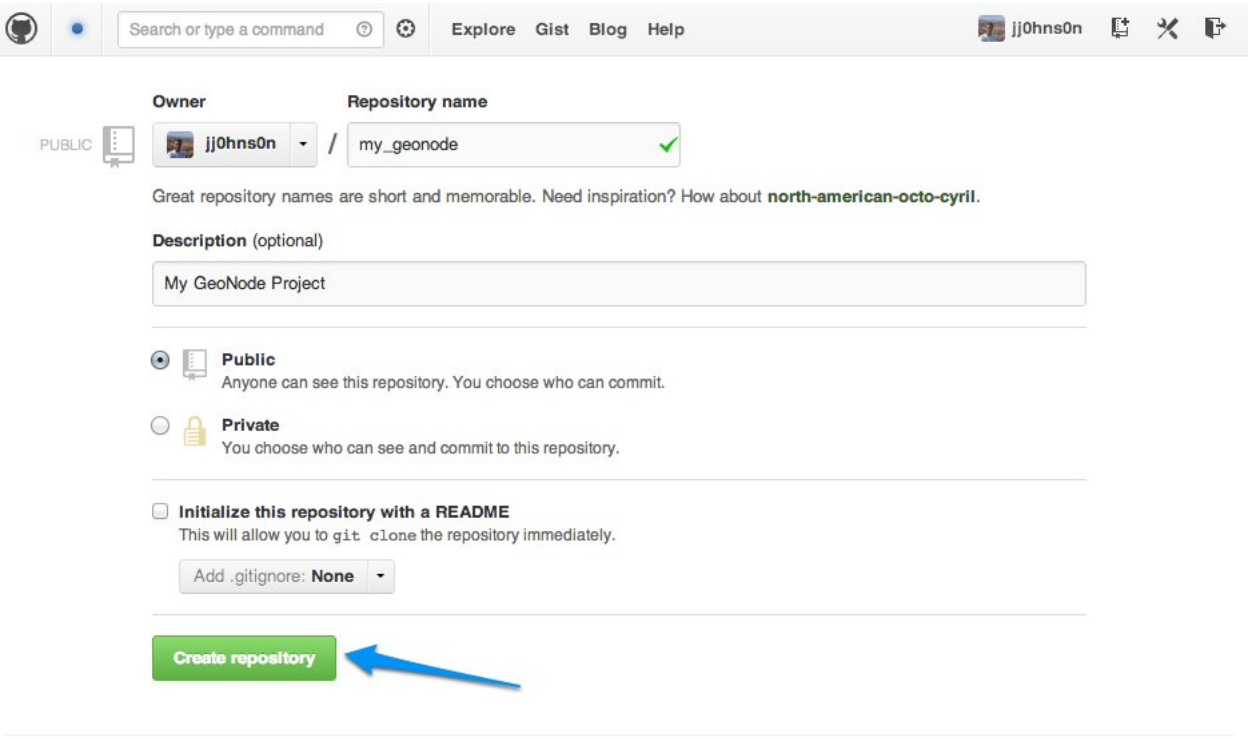


Figure 3.2: *Specifying new GitHub Repository Parameters*

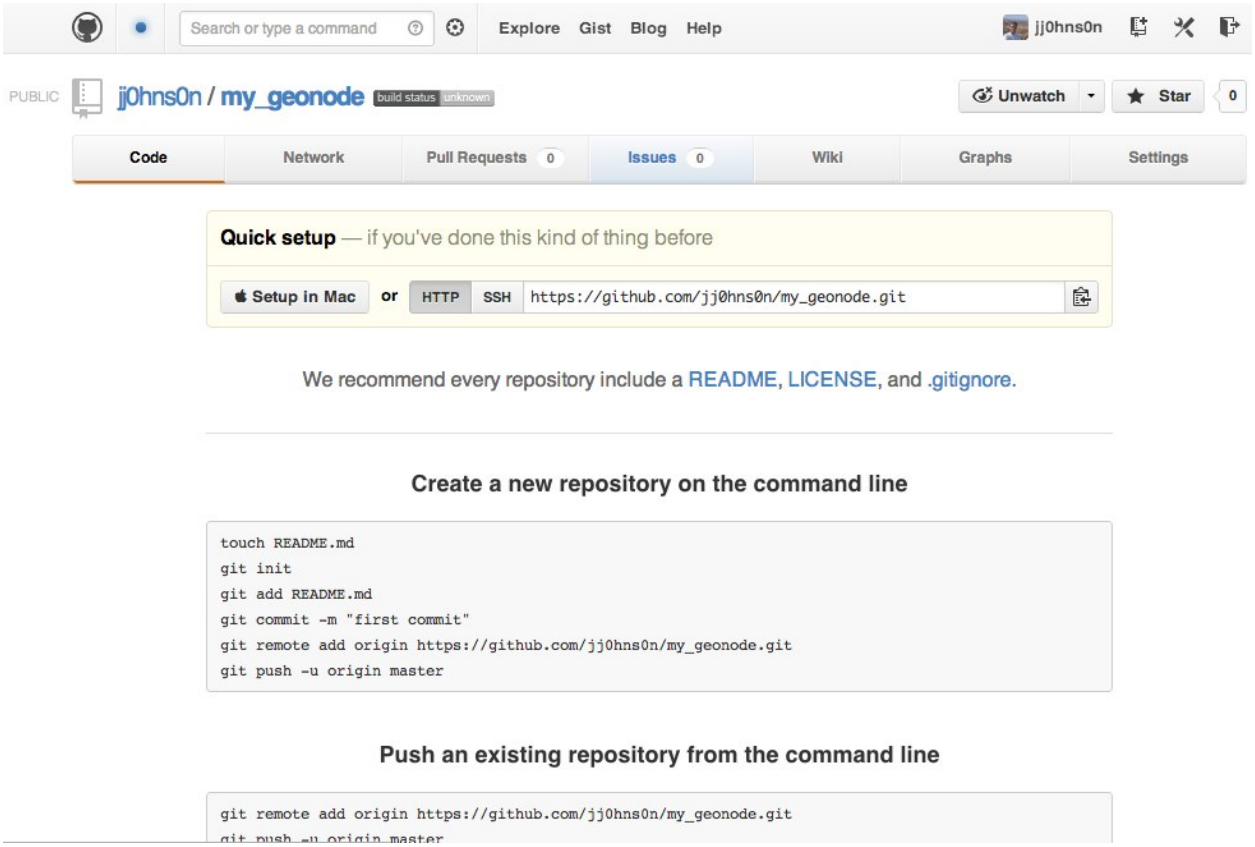


Figure 3.3: *Your new Empty GitHub Repository*

```
$ git init
```

3. Add the remote repository reference to your local git configuration:

```
$ git remote add
```

4. Add your project files to the repository:

```
$ git add .
```

5. Commit your changes:

```
$ git commit -am "Initial commit"
```

6. Push to the remote repository:

```
$ git push origin master
```

3.2.3 Project structure

Your GeoNode project will now be structured as depicted below:

```
|-- README.rst
|-- manage.py
|-- my_geonode
|   |-- __init__.py
|   |-- settings.py
|   |-- static
|   |   |-- README
|   |   |-- css
|   |   |   |-- site_base.css
|   |   |-- img
|   |   |   |-- README
|   |   |-- js
|   |       |-- README
|   |-- templates
|   |   |-- site_base.html
|   |   |-- site_index.html
|   |-- urls.py
|   |-- wsgi.py
|-- setup.py
```

You can also view your project on GitHub.

Each of the key files in your project are described below.

manage.py

`manage.py` is the main entry point for managing your project during development. It allows running all the management commands from each app in your project. When run with no arguments, it will list all of the management commands.

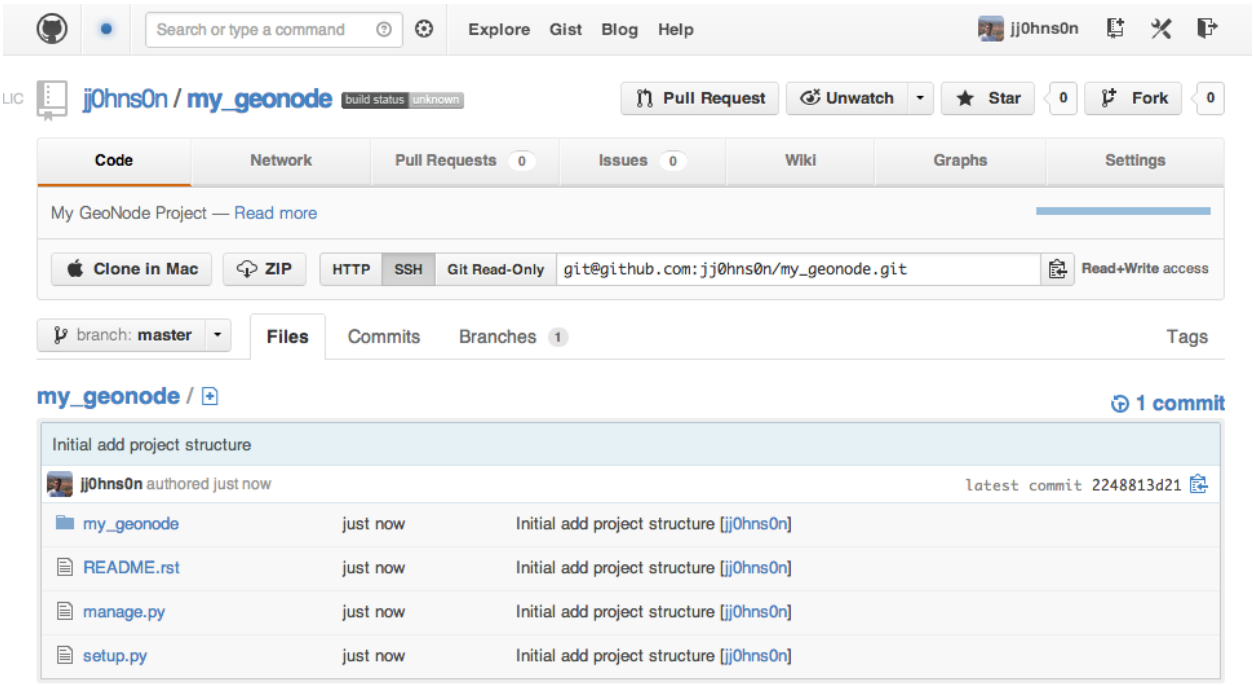


Figure 3.4: *Viewing your project on GitHub*

settings.py

`settings.py` is the primary settings file for your project. It is quite common to put all sensible defaults here and keep deployment specific configuration in the `local_settings.py` file. All of the possible settings values and their meanings are detailed in the Django documentation.

A common paradigm for handing 'local settings' (and in other areas where some python module may not be available) is:

try: from local_settings import *

except: pass

This is not required and there are many other solutions to handling varying deployment configuration requirements.

urls.py

`urls.py` is where your application specific URL routes go. Additionally, any *overrides* can be placed here, too.

wsgi.py

This is a generated file to make deploying your project to a WSGI server easier. Unless there is very specific configuration you need, `wsgi.py` can be left alone.

setup.py

There are several packaging options in python but a common approach is to place your project metadata (version, author, etc.) and dependencies in `setup.py`.

This is a large topic and not necessary to understand while getting started with GeoNode development but will be important for larger projects and to make development easier for other developers.

More: http://docs.python.org/2/distutils/setupscript.html

static

The `static` directory will contain your fixed resources: css, html, images, etc. Everything in this directory will be copied to the final media directory (along with the *static* resources from other apps in your project).

templates

All of your projects templates go in the `templates` directory. While no organization is required for your project specific templates, when overriding or replacing a template from another app, the path must be the same as the template to be replaced.

3.2.4 Deploying your GeoNode Project

Now that your own project is set up, you will need to replace the existing default configuration with configuration for your own project in order to visit your new project site.

1. Update Apache configuration
2. Check GeoServer configuration
3. Check database configuration

3.2.5 Production Ready

While not a complete checklist, some changes should be made prior to deploying to production.

- Ensure DEBUG=False in your effective *settings*.
- Change any admin passwords from any common defaults (i.e. admin/admin)
- Change the geoserver master password from the default
- Ensure GeoServer is ready.

3.2.6 Staying in sync with mainline GeoNode

One of the primary reasons to set up your own GeoNode project using this method is so that you can stay in sync with the mainline GeoNode as the core development team makes new releases. Your own project should not be adversely affected by these changes, but you will receive bug fixes and other improvements by staying in sync.

1. Upgrade GeoNode:

   ```
   $ apt-get update
   $ apt-get install geonode
   ```

2. Verify that your new project works with the upgraded GeoNode:

   ```
   $ python manage.py runserver
   ```

3. Navigate to http://localhost:8000.

3.3 Theming your GeoNode project

There are a range of options available to you if you want to change the default look and feel of your GeoNode project. Since GeoNode's style is based on Bootstrap you will be able to make use of all that Bootstrap has to offer in terms of theme customization. You should consult Bootstrap's documentation as your primary guide once you are familiar with how GeoNode implements Bootstrap and how you can override GeoNode's theme and templates in your own project.

3.3.1 Logos and graphics

GeoNode intentionally does not include a large number of graphics files in its interface. This keeps page loading time to a minimum and makes for a more responsive interface. That said, you are free to customize your GeoNode's interface by simply changing the default logo, or by adding your own images and graphics to deliver a GeoNode experience the way you envision int.

Your GeoNode project has a directory already set up for storing your own images at `<my_geonode>/static/img`. You should place any image files that you intend to use for your project in this directory.

Let's walk through an example of the steps necessary to change the default logo.

1. Change into the `img` directory:

   ```
   $ cd <my_geonode>/static/img
   ```

2. If you haven't already, obtain your logo image. The URL below is just an example, so you will need to change this URL to match the location of your file or copy it to this location:

   ```
   $ wget http://www2.sta.uwi.edu/~anikov/UWI-logo.JPG
   $ cd ../../..
   ```

3. Override the CSS that displays the logo by editing `<my_geonode>/static/css/site_base.css` with your favorite editor and adding the following lines, making sure to update the width, height, and URL to match the specifications of your image.

   ```
   .nav-logo {
       width: 373px;
       height: 79px;
       background: url(../img/UWI-logo.JPG) no-repeat;
   }
   ```

4. Restart your GeoNode project and look at the page in your browser:

   ```
   $ python manage.py runserver
   ```

Visit your site at http://localhost:8000/ or the remote URL for your site.

You can see that the header has been expanded to fit your graphic. In the following sections you will learn how to customize this header to make it look and function the way you want.

Note: You should commit these changes to your repository as you progress through this section, and get in the habit of committing early and often so that you and others can track your project on GitHub. Making many atomic commits and staying in sync with a remote repository makes it easier to collaborate with others on your project.

3.3.2 Cascading Style Sheets

In the last section you already learned how to override GeoNode's default CSS rules to include your own logo. You are able to customize any aspect of GeoNode's appearance this way. In the last screenshot, you saw that the main area

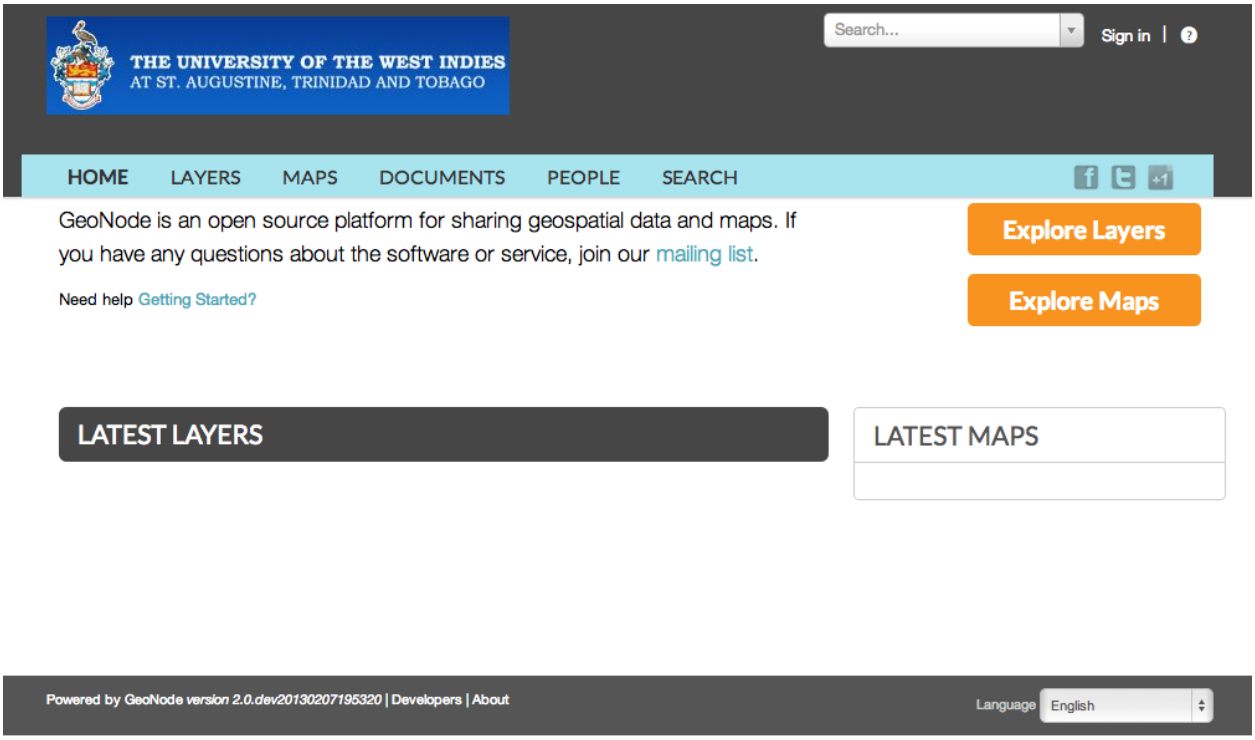


Figure 3.5: *Custom logo*

in the homepage is covered up by the expanded header.

First, we'll walk through the steps necessary to displace it downward so it is no longer hidden, then change the background color of the header to match the color in our logo graphic.

1. Reopen `<my_geonode>/static/css/site_base.css` in your editor and add the following rule after the one added in the previous step:

```
.content-wrap {
    margin: 75px 75px;
}
```

2. Add a rule to change the background color of the header to match the logo graphic we used:

```
.navbar .navbar-inner {
    background: #0e60c3;
}
```

3. Your project CSS file should now look like this:

```
.nav-logo {
    width: 373px;
    height: 79px;
    background: url(../img/UWI-logo.JPG) no-repeat;
}

.content-wrap {
    margin: 75px 75px;
}
```

```
.navbar .navbar-inner {
    background: #0e60c3;
}
```

4. Restart the development server and reload the page:

```
$ python manage.py runserver
```

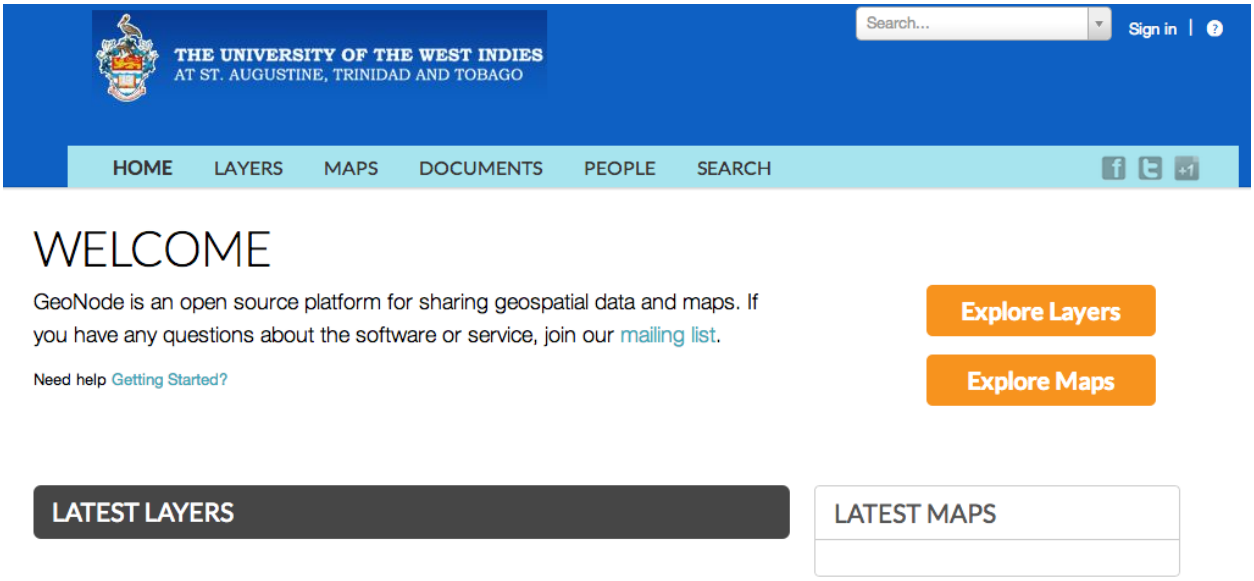


Figure 3.6: *CSS overrides*

Note: You can continue adding rules to this file to override the styles that are in the GeoNode base CSS file which is built from base.less. You may find it helpful to use your browser's development tools to inspect elements of your site that you want to override to determine which rules are already applied. See the screenshot below. Another section of this workshop covers this topic in much more detail.

3.3.3 Templates and static pages

Now that we have changed the default logo and adjusted our main content area to fit the expanded header, the next step is to update the content of the homepage itself. Your GeoNode project includes two basic templates that you will use to change the content of your pages.

The file `site_base.html` (in `<my_geonode>/templates/`) is the basic template that all other templates inherit from and you will use it to update things like the header, navbar, site-wide announcement, footer, and also to include your own JavaScript or other static content included in every page in your site. It's worth taking a look at GeoNode's base file on GitHub. You have several blocks available to you to for overriding, but since we will be revisiting this file in future sections of this workshop, let's just look at it for now and leave it unmodified.

Open `<my_geonode>/templates/site_base.html` in your editor:

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
{% endblock %}
```

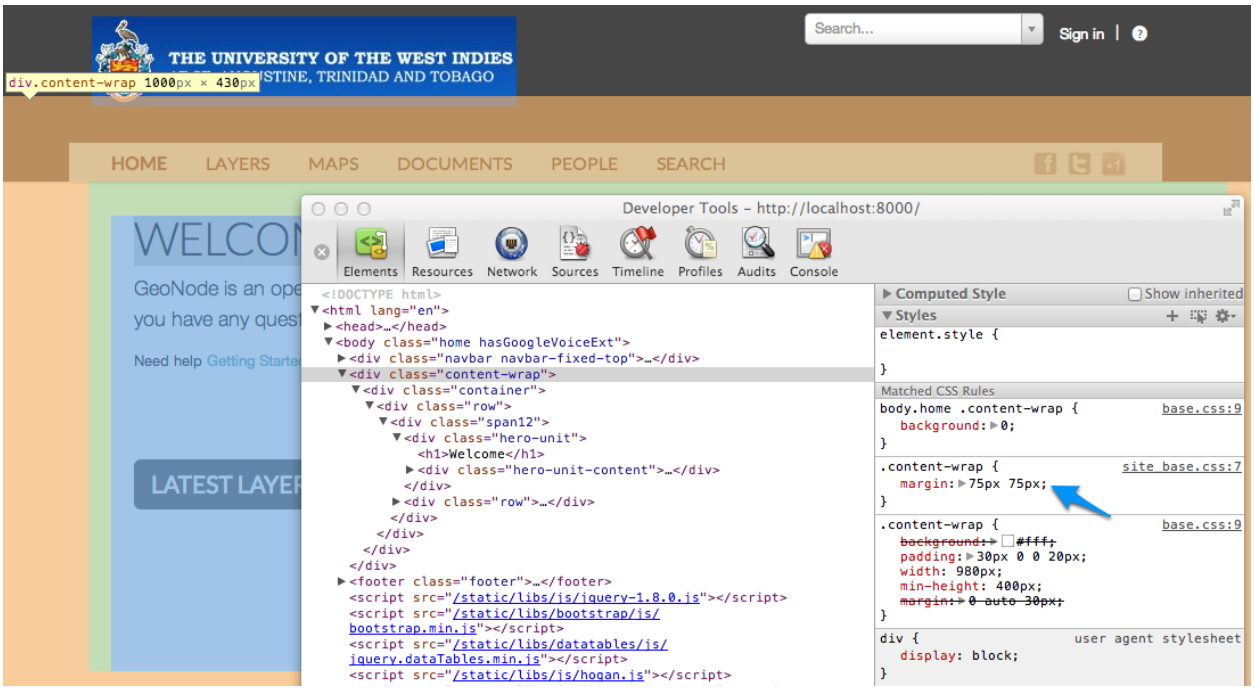


Figure 3.7: *Screenshot of using Chrome's debugger to inspect the CSS overrides*

You will see that it extends from `base.html`, which is the GeoNode template referenced above and it currently only overrides the `extra_head` block to include our project's `site_base.css` which we modified in the previous section. You can see on line 14 of the GeoNode base.html template that this block is included in an empty state and is set up specifically for you to include extra CSS files as your project is already set up to do.

Now that we have looked at `site_base.html`, let's actually override a different template.

The file `site_index.html` is the template used to define your GeoNode project's homepage. It extends GeoNode's default `index.html` template and gives you the option to override specific areas of the homepage like the hero area, but also allows you leave area like the "Latest Layers" and "Maps" and the "Contribute" section as they are. You are of course free to override these sections if you choose and this section shows you the steps necessary to do that below.

1. Open `<my_geonode>/templates/site_index.html` in your editor.
2. Edit the `<h1>` element on line 13 to say something other than "Welcome":

```
<h1>{% trans "UWI GeoNode" %}</h1>
```

3. Edit the introductory paragraph to include something specific about your GeoNode project:

```
<p>
    {% blocktrans %}
    UWI's GeoNode is setup for students and faculty to collaboratively
    create and share maps for their class projects. It is maintained by the
    UWI Geographical Society.
    {% endblocktrans %}
</p>
```

4. Change the *Getting Started* link to point to another website:

```
<span>
    For more information about the UWI Geographical society,
    <a href="http://uwigsmona.weebly.com/">visit our website</a>
```

```
</span>
```

5. Add a graphic to the hero area above the paragraph replaced in step 3:

```
<img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
```

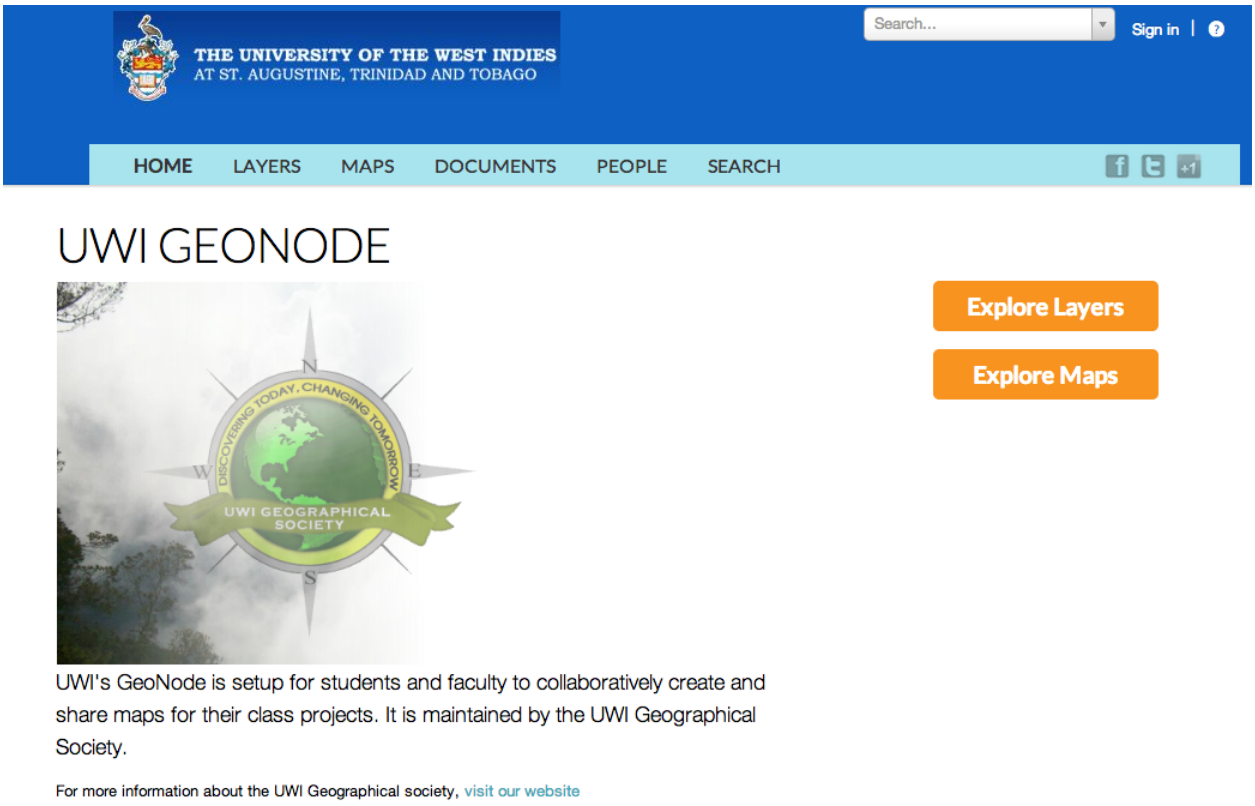
6. Your edited `site_index.html` file should now look like this:

```
{% extends 'index.html' %}
{% load i18n %}
{% load maps_tags %}
{% load layers_tags %}
{% load pagination_tags %}
{% load staticfiles %}
{% load url from future %}
{% comment %}
This is where you can override the hero area block. You can simply modify the content below or r
{% endcomment %}
{% block hero %}
    <div class="hero-unit">
        <h1>{% trans "UWI GeoNode" %}</h1>
        <div class="hero-unit-content">
        <div class="intro">
            <img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
        <p>
            {% blocktrans %}
            UWI's GeoNode is setup for students and faculty to collaboratively
            create and share maps for their class projects. It is maintained by the
            UWI Geographical Society.
            {% endblocktrans %}
        </p>
        <span>
            For more information about the UWI Geographical society,
            <a href="http://uwigsmona.weebly.com/">visit our website</a>
        </span>
    </div>
    <div class="btns">
        <a class="btn btn-large" href="{% url "layer_browse" %}">
        {% trans "Explore Layers" %}
        </a>
        <a class="btn btn-large" href="{% url "maps_browse" %}">
        {% trans "Explore Maps" %}
        </a>
    </div>
</div>
{% endblock %}
```

7. Restart your GeoNode project and view the changes in your browser at http://localhost:8000/ or the remote URL for your site:

```
$ python manage.py runserver
```

From here you can continue to customize your `site_index.html` template to suit your needs. This workshop will also cover how you can add new pages to your GeoNode project site.



3.3.4 Other theming options

You are able to change any specific piece of your GeoNode project's style by adding CSS rules to `site_base.css`, but since GeoNode is based on Bootstrap, there are many pre-defined themes that you can simply drop into your project to get a whole new look. This is very similar to WordPress themes and is a powerful and easy way to change the look of your site without much effort.

Bootswatch

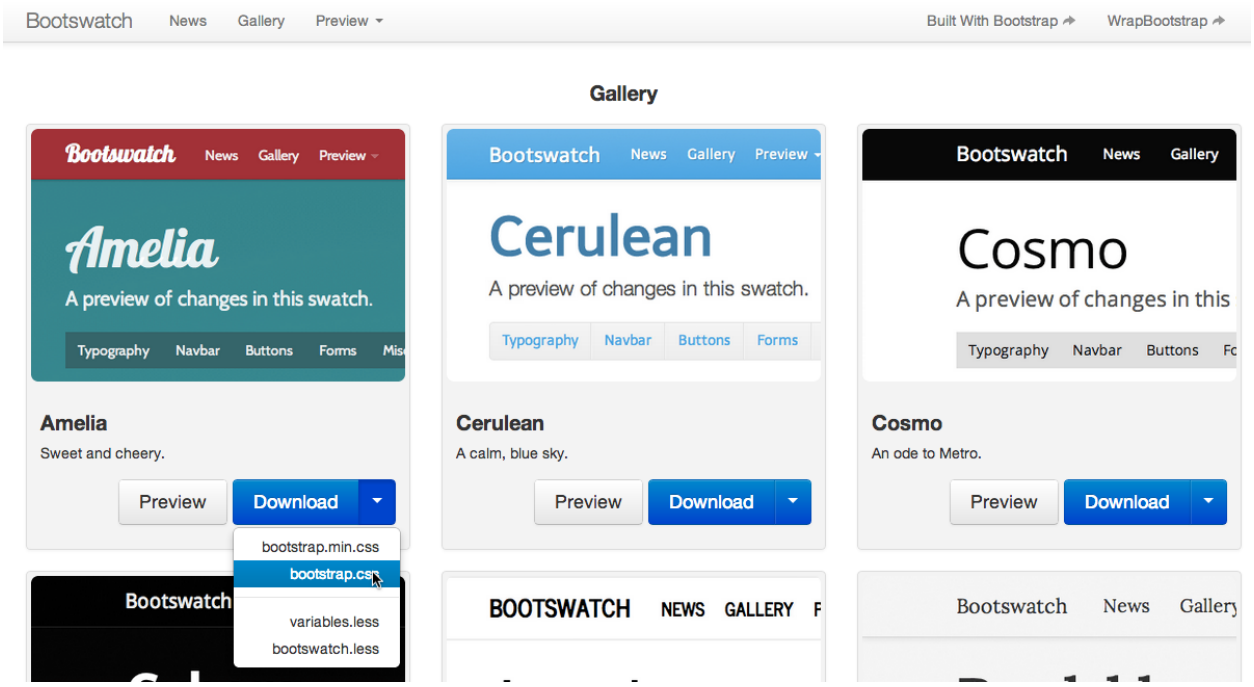
Bootswatch is a site where you can download ready-to-use themes for your GeoNode project site. The following steps will show you how to use a theme from Bootswatch in your own GeoNode site.

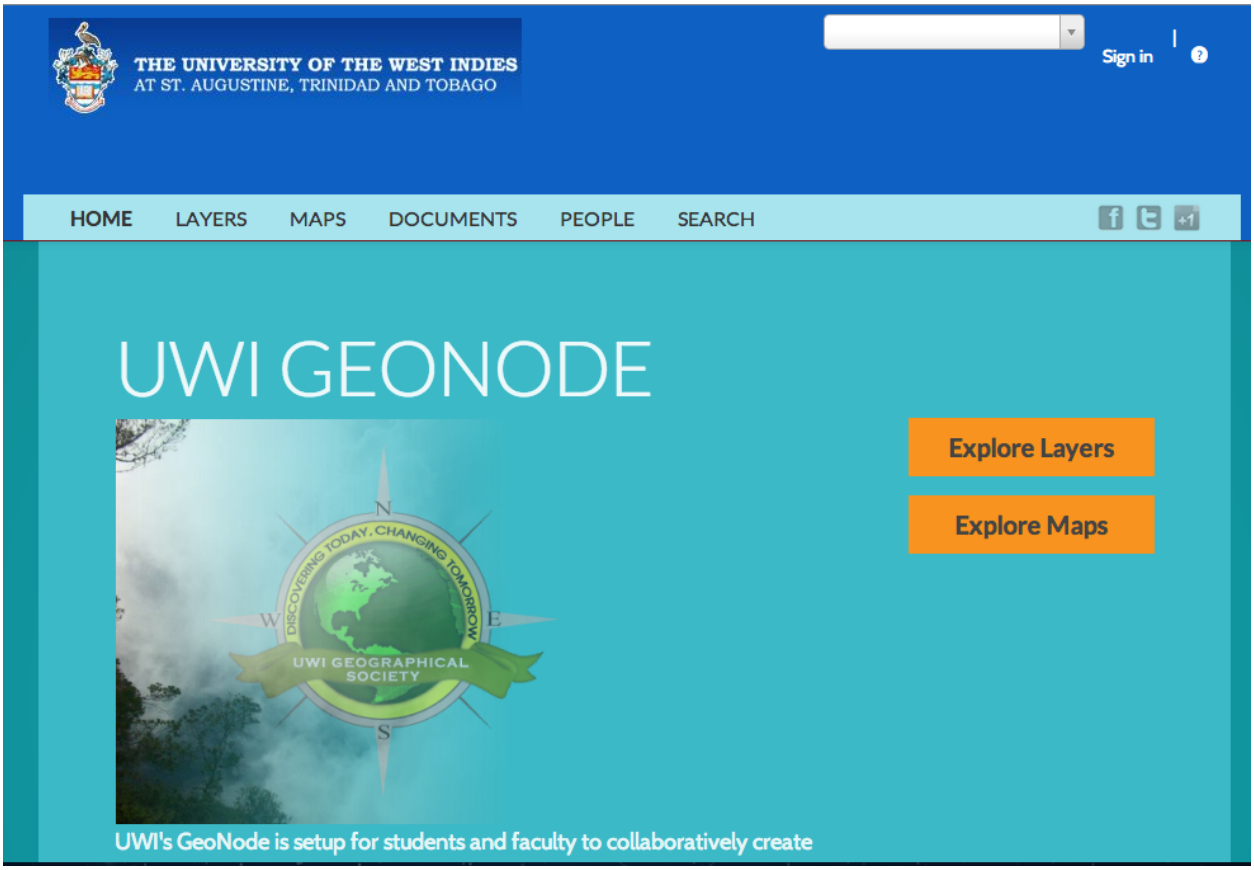
1. Visit http://bootswatch.com and select a theme (we will use Amelia for this example). Select the *download bootstrap.css option* in the menu:
2. Put this file in `<my_geonode>/static/css`.
3. Update the `site_base.html` template to include this file. It should now look like this:

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
    <link href="{{ STATIC_URL }}css/bootstrap.css" rel="stylesheet"/>
{% endblock %}
```

4. Restart the development server and visit your site:

Your GeoNode project site is now using the Amelia theme in addition to the changes you have made.


Bootswatch
News
Gallery
Preview
Built With Bootstrap
WrapBootstrap
Gallery
Amelia
A preview of changes in this swatch.
Typography
Navbar
Buttons
Forms
Amelia
Sweet and cheery.
Preview
Download
bootstrap.min.css
bootstrap.css
variables.less
bootswatch.less
Cerulean
A preview of changes in this swatch.
Cerulean
A calm, blue sky.
Cosmo
A preview of changes in this
Cosmo
An ode to Metro.
BOOTSWATCH
NEWS
GALLERY


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UWI GEONODE
Explore Layers
Explore Maps
DISCOVERING TODAY, CHANGING TOMORROW
UWI GEOGRAPHICAL SOCIETY
UWI's GeoNode is setup for students and faculty to collaboratively create

3.4 Adding additional Django apps to your GeoNode Project

Since GeoNode is based on Django, your GeoNode project can be augmented and enhanced by adding additional third-party pluggable Django apps or by writing an app of your own.

This section of the workshop will introduce you to the Django pluggable app ecosystem, and walk you through the process of writing your own app and adding a blog app to your project.

3.4.1 Django pluggable apps

The Django app ecosystem provides a large number of apps that can be added to your project. Many are mature and used in many existing projects and sites, while others are under active early-stage development. Websites such as Django Packages provide an interface for discovering and comparing all the apps that can plugged in to your Django project. You will find that some can be used with very little effort on your part, and some will take more effort to integrate.

3.4.2 Adding your own Django app

Let's walk through the an example of the steps necessary to create a very basic Django polling app to and add it to your GeoNode project. This section is an abridged version of the Django tutorial itself and it is strongly recommended that you go through this external tutorial along with this section as it provides much more background material and a signficantly higher level of detail. You should become familiar with all of the information in the Django tutorial as it is critical to your success as a GeoNode project developer.

Throughout this section, we will walk through the creation of a basic poll application. It will consist of two parts:

- A public site that lets people view polls and vote in them.
- An admin site that lets you add, change, and delete polls.

Since we have already created our GeoNode project from a template project, we will start by creating our app structure and then adding models:

```
$ python manage.py startapp polls
```

That'll create a directory polls, which is laid out like this:

```
polls/
    __init__.py
    models.py
    tests.py
    views.py
```

This directory structure will house the poll application.

The first step in writing a database web app in Django is to define your models—essentially, your database layout with additional metadata.

In our simple poll app, we'll create two models: polls and choices. A poll has a question and a publication date. A choice has two fields: the text of the choice and a vote tally. Each choice is associated with a poll.

These concepts are represented by simple Python classes.

Edit the `polls/models.py` file so it looks like this:

```
from django.db import models

class Poll(models.Model):
```

```
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __unicode__(self):
        return self.question

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice = models.CharField(max_length=200)
    votes = models.IntegerField()
    def __unicode__(self):
        return self.choice
```

That small bit of model code gives Django a lot of information. With it, Django is able to:

- Create a database schema (CREATE TABLE statements) for this app.
- Create a Python database-access API for accessing Poll and Choice objects.

But first we need to tell our project that the polls app is installed.

Edit the `<my_geonode>/settings.py` file, and update the INSTALLED_APPS setting to include the string "polls". So it will look like this:

```
INSTALLED_APPS = (

    # Apps bundled with Django
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'django.contrib.sitemaps',
    'django.contrib.staticfiles',
    'django.contrib.messages',
    'django.contrib.humanize',

    #Third party apps

    # <snip>

    # GeoNode internal apps
    'geonode.maps',
    'geonode.upload',
    'geonode.layers',
    'geonode.people',
    'geonode.proxy',
    'geonode.security',
    'geonode.search',
    'geonode.catalogue',
    'geonode.documents',

    # My GeoNode apps
    'polls',
)
```

Now Django knows to include the polls app. Let's run another command:

```
$ python manage.py syncdb
```

The `syncdb` command runs the SQL from `sqlall` on your database for all apps in INSTALLED_APPS that don't

already exist in your database. This creates all the tables, initial data, and indexes for any apps you've added to your project since the last time you ran `syncdb`. `syncdb` can be called as often as you like, and it will only ever create the tables that don't exist.

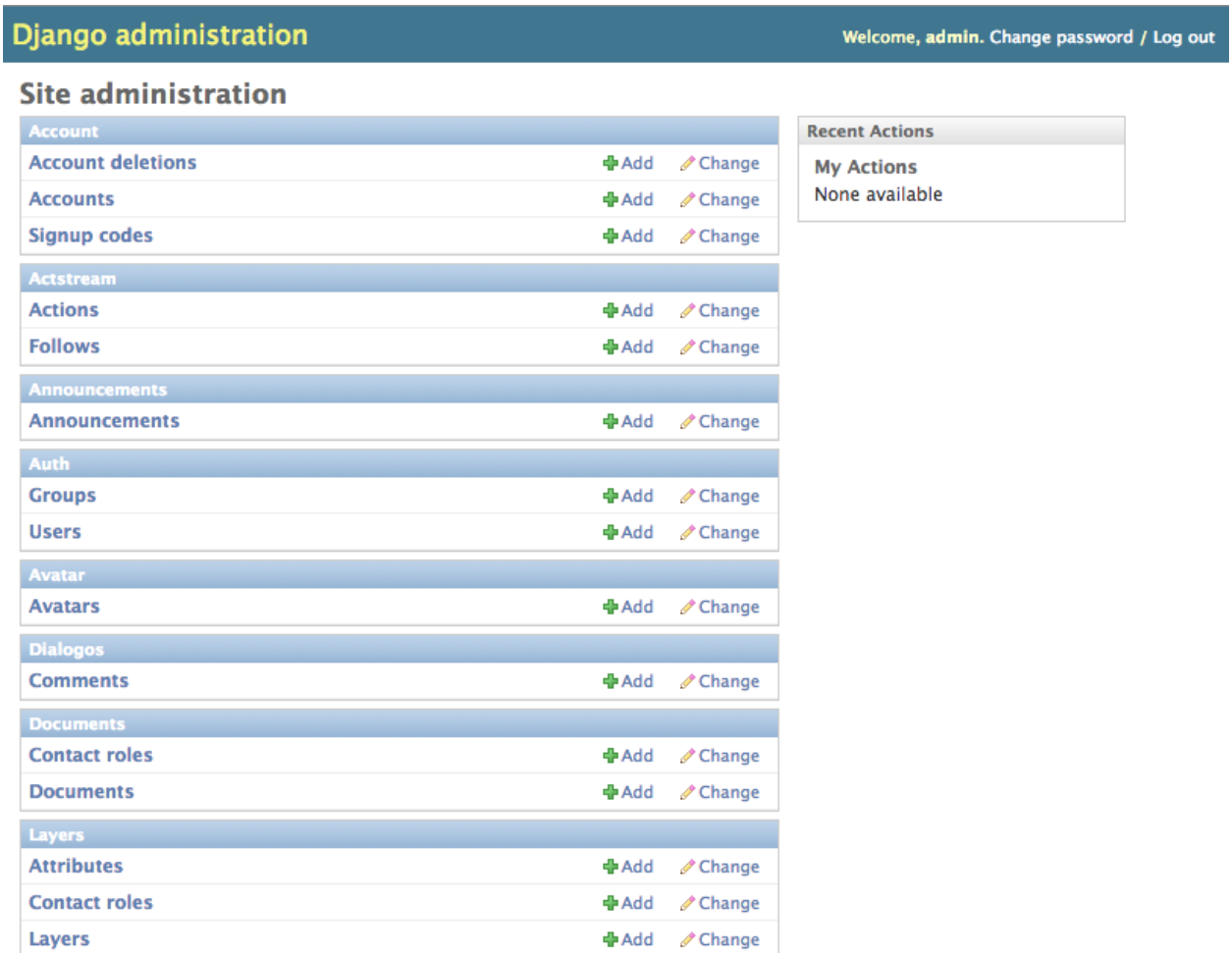
GeoNode uses south for migrations ...

Next, let's add the Django admin configuration for our polls app so that we can use the Django Admin to manage the records in our database. Create and edit a new file called `polls/admin.py` and make it look like the this:

```
from polls.models import Poll
from django.contrib import admin

admin.site.register(Poll)
```

Run the development server and explore the polls app in the Django Admin by pointing your browser to http://localhost:8000/admin/ and logging in with the credentials you specified when you first ran `syncdb`.

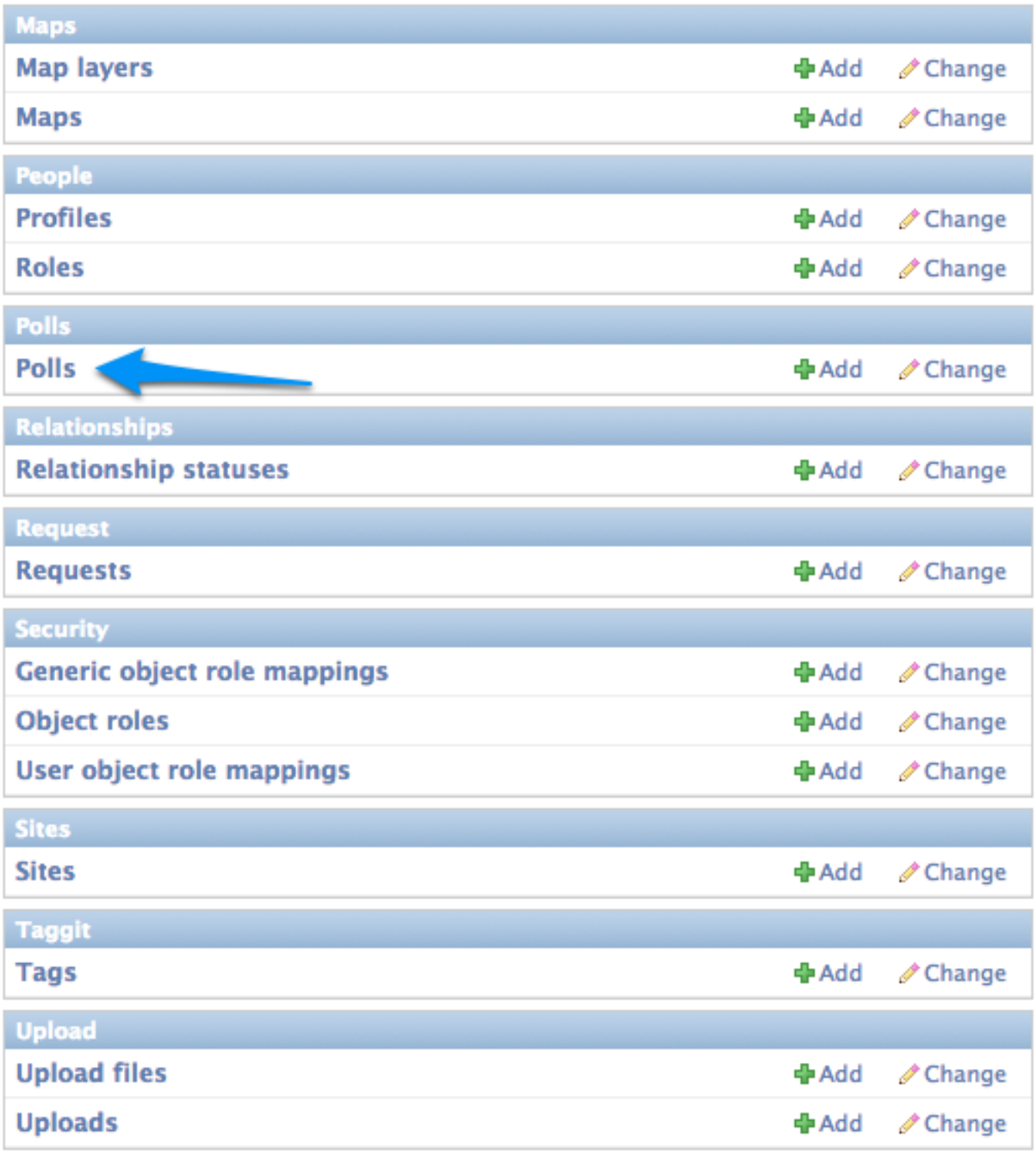


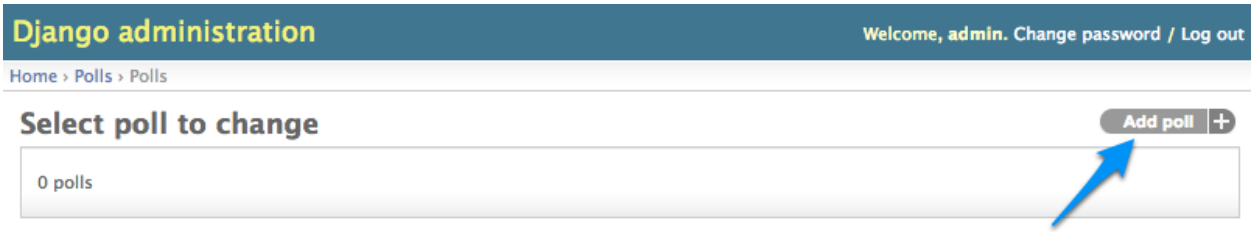
You can see all of the other apps that are installed as part of your GeoNode project, but we are specifically interested in the Polls app for now.

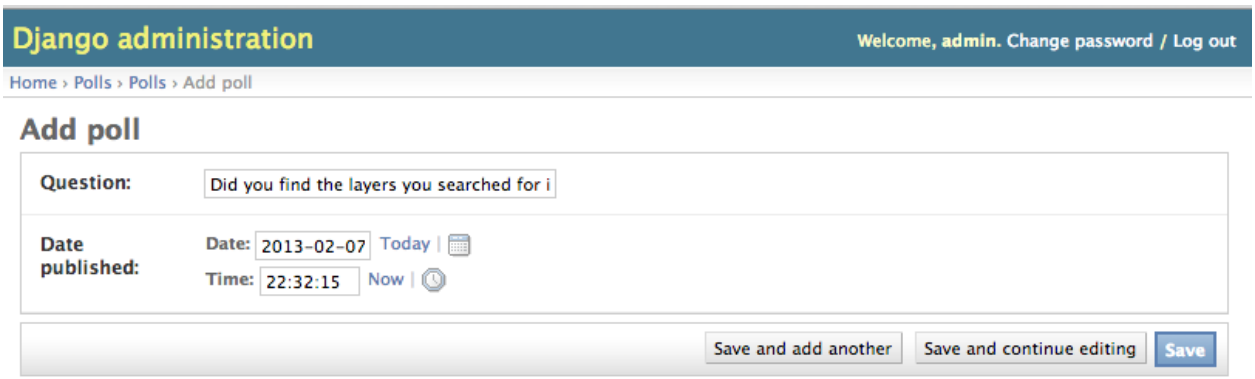
Next we will add a new poll via automatically generated admin form.

You can enter any sort of question you want for initial testing and select today and now for the publication date.

The next step is to configure the Choice model in the admin, but we will configure the choices to be editable in-line with the Poll objects they are attached to. Edit the same `polls/admin.py` so it now looks like the following:


Maps
Map layers Add Change
Maps Add Change
People
Profiles Add Change
Roles Add Change
Polls
Polls Add Change
Relationships
Relationship statuses Add Change
Request
Requests Add Change
Security
Generic object role mappings Add Change
Object roles Add Change
User object role mappings Add Change
Sites
Sites Add Change
Taggit
Tags Add Change
Upload
Upload files Add Change
Uploads Add Change


Django administration
Welcome, admin. Change password / Log out
Home › Polls › Polls
Select poll to change
Add poll
0 polls




```
from polls.models import Poll, Choice
from django.contrib import admin

class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 3

class PollAdmin(admin.ModelAdmin):
    fieldsets = [
        (None,               {'fields': ['question']}),
        ('Date information', {'fields': ['pub_date'], 'classes': ['collapse']}),
    ]
    inlines = [ChoiceInline]

admin.site.register(Poll, PollAdmin)
```

This tells Django that Choice objects are edited on the Poll admin page, and by default, provide enough fields for 3 choices.

You can now return to the Poll admin and either add a new poll or edit the one you already created and see that you can now specify the poll choices inline with the poll itself.

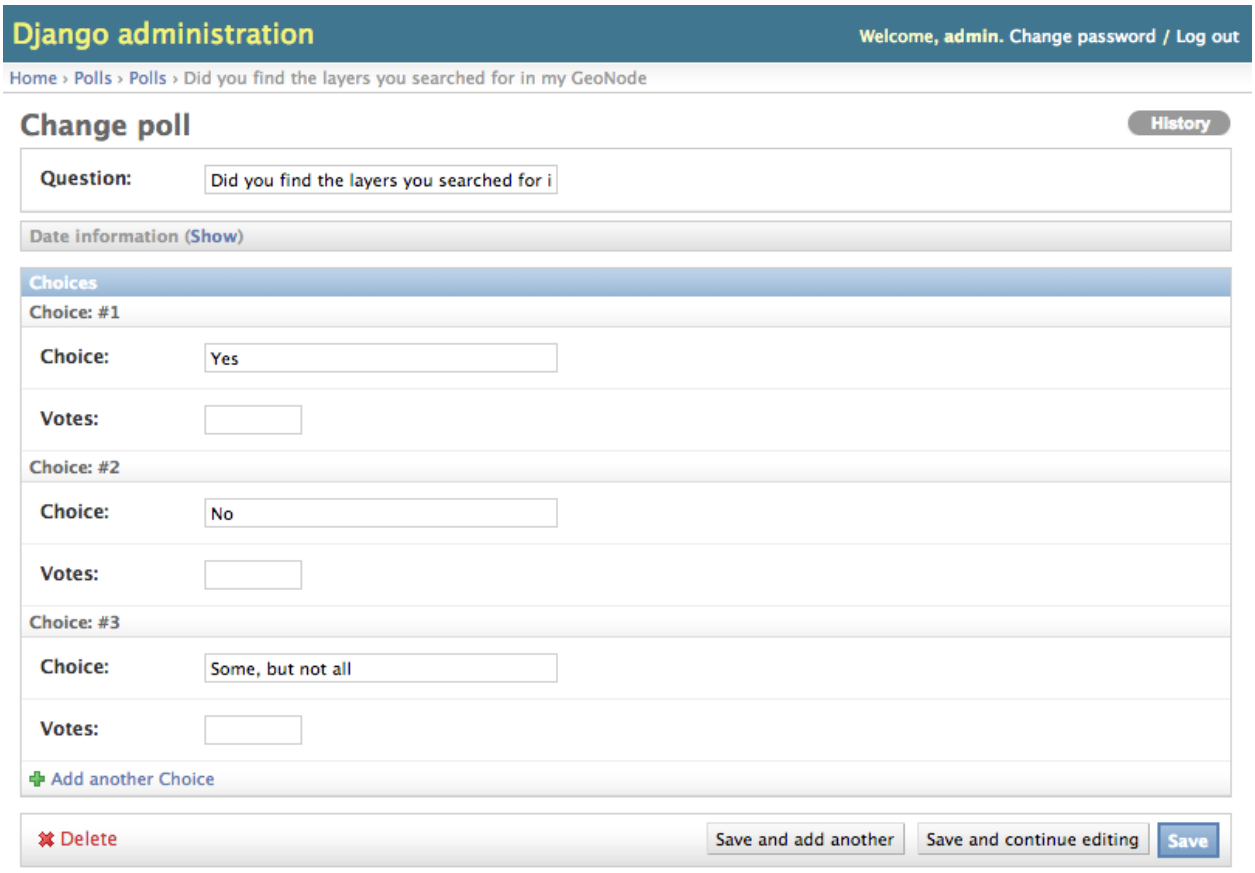
From here, we want to create views to display the polls inside our GeoNode project. A view is a "type" of Web page in your Django application that generally serves a specific function and has a specific template. In our poll application, there will be the following four views:

- Poll "index" page—displays the latest few polls.
- Poll "detail" page—displays a poll question, with no results but with a form to vote.
- Poll "results" page—displays results for a particular poll.
- Vote action—handles voting for a particular choice in a particular poll.

The first step of writing views is to design your URL structure. You do this by creating a Python module called a URLconf. URLconfs are how Django associates a given URL with given Python code.

Let's start by adding our URL configuration directly to the `urls.py` that already exists in your project at `<my_geonode>/urls.py`. Edit this file and add the following lines after the rest of the existing imports around line 80:

```
url(r'^polls/$', 'polls.views.index'),
url(r'^polls/(?P<poll_id>\d+)/$', 'polls.views.detail'),
url(r'^polls/(?P<poll_id>\d+)/results/$', 'polls.views.results'),
url(r'^polls/(?P<poll_id>\d+)/vote/$', 'polls.views.vote'),
```


Django administration
Welcome, admin. Change password / Log out
Home › Polls › Polls › Did you find the layers you searched for in my GeoNode
Change poll
History
Question:
Did you find the layers you searched for i
Date information (Show)
Choices
Choice: #1
Choice:
Yes
Votes:
Choice: #2
Choice:
No
Votes:
Choice: #3
Choice:
Some, but not all
Votes:
Add another Choice
Delete
Save and add another
Save and continue editing
Save

Note: Eventually we will want to move this set of URL configurations inside the URLs app itself, but for the sake of brevity in this workshop, we will put them in the main `urls.py` for now. You can consult the Django tutorial for more information on this topic.

Next write the views to drive the URL patterns we configured above. Edit polls/views.py to that it looks like the following:

```
from django.template import Context, loader
from polls.models import Poll
from django.http import HttpResponse
from django.http import Http404
from django.shortcuts import render_to_response

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    return render_to_response('polls/index.html',
        RequestContext(request, {'latest_poll_list': latest_poll_list}))

def detail(request, poll_id):
    try:
        p = Poll.objects.get(pk=poll_id)
    except Poll.DoesNotExist:
        raise Http404
    return render_to_response('polls/detail.html', RequestContext(request, {'poll': p}))

def results(request, poll_id):
    return HttpResponse("You're looking at the results of poll %s." % poll_id)

def vote(request, poll_id):
    return HttpResponse("You're voting on poll %s." % poll_id)
```

Note: We have only stubbed in the views for the results and vote pages. They are not very useful as-is. We will revisit these later.

Now we have views in place, but we are referencing templates that do not yet exist. Let's add them by first creating a template directory in your polls app at `polls/templates/polls` and creating `polls/templates/polls/index.html` to look like the following:

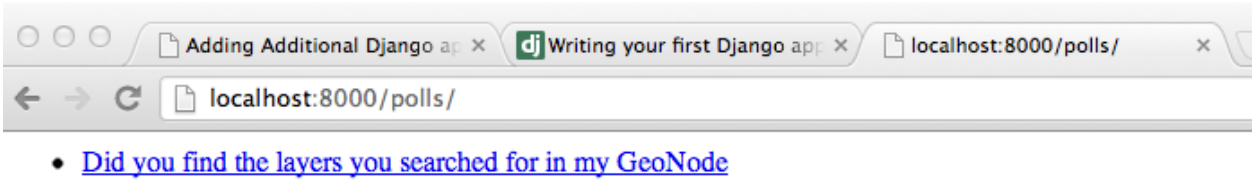
```
{% if latest_poll_list %}
    <ul>
    {% for poll in latest_poll_list %}
        <li><a href="/polls/{{ poll.id }}/">{{ poll.question }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Next we need to create the template for the poll detail page. Create a new file at `polls/templates/polls/detail.html` to look like the following:

```
<h1>{{ poll.question }}</h1>
<ul>
{% for choice in poll.choice_set.all %}
    <li>{{ choice.choice }}</li>
```

```
{% endfor %}
</ul>
```

You can now visit http://localhost:8000/polls/ in your browser and you should see the the poll question you created in the admin presented like this.



We actually want our polls app to display as part of our GeoNode project with the same theme, so let's update the two templates we created above to make them extend from the `site_base.html` template we looked at in the last section. You will need to add the following two lines to the top of each file:

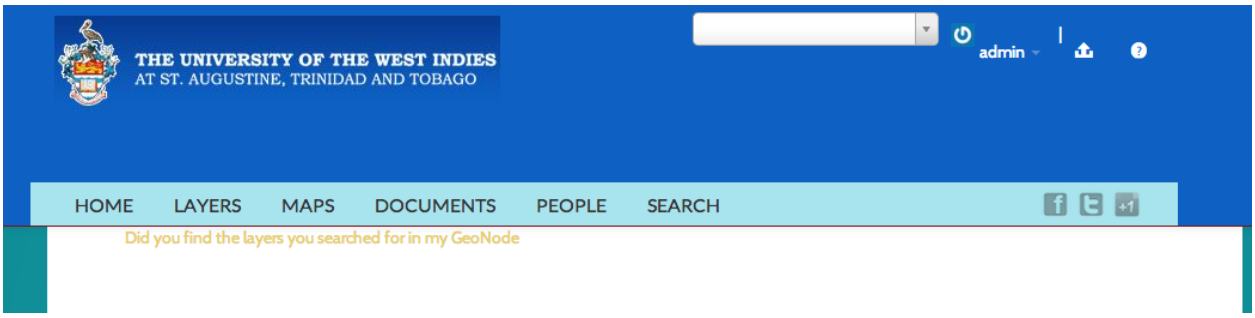
```
{% extends 'site_base.html' %}
{% block body %}
```

And close the block at the bottom of each file with:

```
{% endblock %}
```

This tells Django to extend from the `site_base.html` template so your polls app has the same style as the rest of your GeoNode, and it specifies that the content in these templates should be rendered to the body block defined in GeoNode's `base.html` template that your `site_base.html` extends from.

You can now visit the index page of your polls app and see that it is now wrapped in the same style as the rest of your GeoNode site.

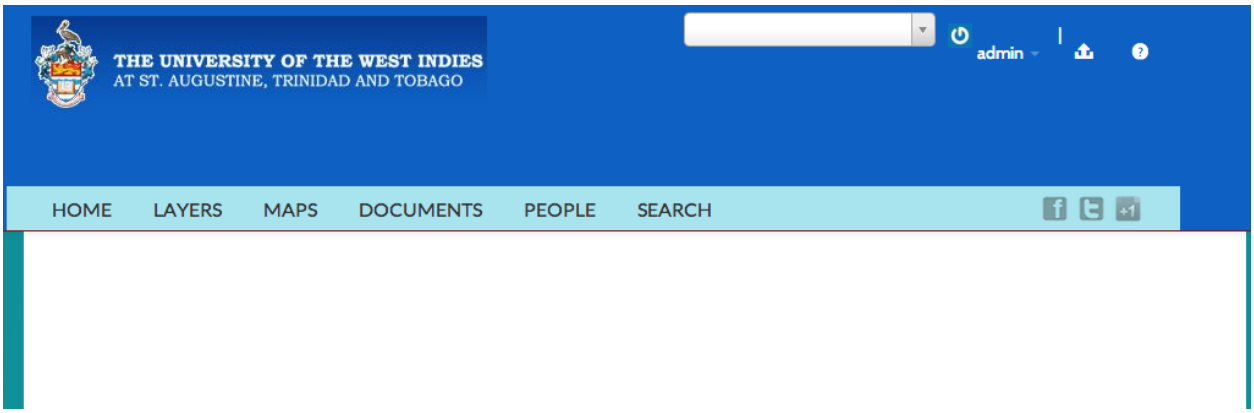


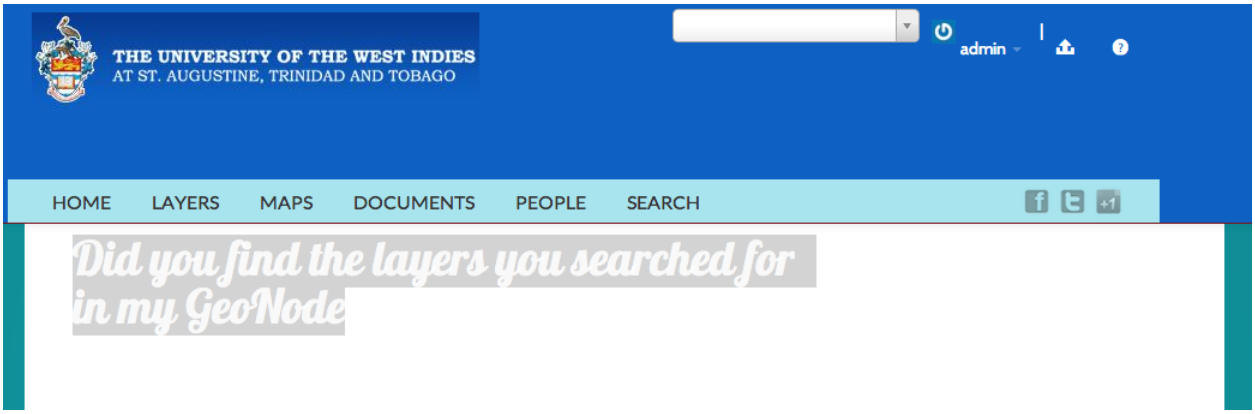
If you click on a question from the list you will be taken to the poll detail page.

It looks like it is empty, but in fact the text is there, but styled to be white by the Bootswatch theme we added in the last section. If you highlight the area where the text is, you will see that it is there.

Now that you have walked through the basic steps to create a very minimal (though not very useful) Django app and integrated it with your GeoNode project, you should pick up the Django tutorial at part 4 and follow it to add the form for actually accepting responses to your poll questions.

We strongly recommend that you spend as much time as you need with the Django tutorial itself until you feel comfortable with all of the concepts. They are the essential building blocks you will need to extend your GeoNode project by adding your own apps.





3.4.3 Adding a 3rd party blog app

Now that we have created our own app and added it to our GeoNode project, the next thing we will work through is adding a 3rd party blog app. There are a number of blog apps that you can use, but for purposes of this workshop, we will use a relatively simple, yet extensible app called Zinnia. You can find out more information about Zinnia on its website or on its GitHub project page or by following its documentation. This section will walk you through the minimal set of steps necessary to add Zinnia to your GeoNode project.

The first thing to do is to install Zinnia into the virtualenv that you are working in. Make sure your virtualenv is activated and execute the following command:

```
$ pip install django-blog-zinnia
```

This will install Zinnia and all of the libraries that it depends on.

Next add Zinnia to the INSTALLED_APPS section of your GeoNode projects `settings.py` file by editing `<my_geonode>/settings.py` and adding 'django.contrib.comments' to the section labeled "Apps Bundled with Django" so that it looks like the following:

```
# Apps bundled with Django
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.sites',
'django.contrib.admin',
'django.contrib.sitemaps',
'django.contrib.staticfiles',
'django.contrib.messages',
```

```
'django.contrib.humanize',
'django.contrib.comments',
```

And then add the `tagging`, `mptt` and `zinnia` apps to the end of the INSTALLED_APPS where we previously added a section labeled "My GeoNode apps". It should like like the following:

```
# My GeoNode apps
'polls',
'tagging',
'mptt',
'zinnia',
```

Next you will need to run `syncdb` again to synchronize the models for the apps we have just added to our project's database. This time we want to pass the `--all` flag to `syncdb` so it ignores the schema migrations. Schema migrations are discussed further in GeoNode's documentation, but it is safe to ignore them here.

```
$ python manage.py syncdb --all
```

You can now restart the development server and visit the Admin interface and scroll to the very bottom of the list to find a section for Zinnia that allows you to manage database records for Categories and Blog Entries.

Next we need to configure our project to add Zinnia's URL configurations. Add the following two URL configuration entries to the end of `<my_geonode>/urls.py`:

```
url(r'^blog/', include('zinnia.urls')),
url(r'^djcomments/', include('django.contrib.comments.urls')),
```

If you visit the main blog page in your browser at http://localhost:8000/blog/ you will find that the blog displays with Zinnia's default theme as shown below.

This page includes some guidance for us on how to change the default theme. The first thing we need to do is to copy Zinnia's `base.html` template into our own project so we can modify it. When you installed Zinnia, templates were installed to `/var/lib/geonode/lib/python2.7/site-packages/zinnia/templates/zinnia/`. You can copy the base template by executing the following commands:

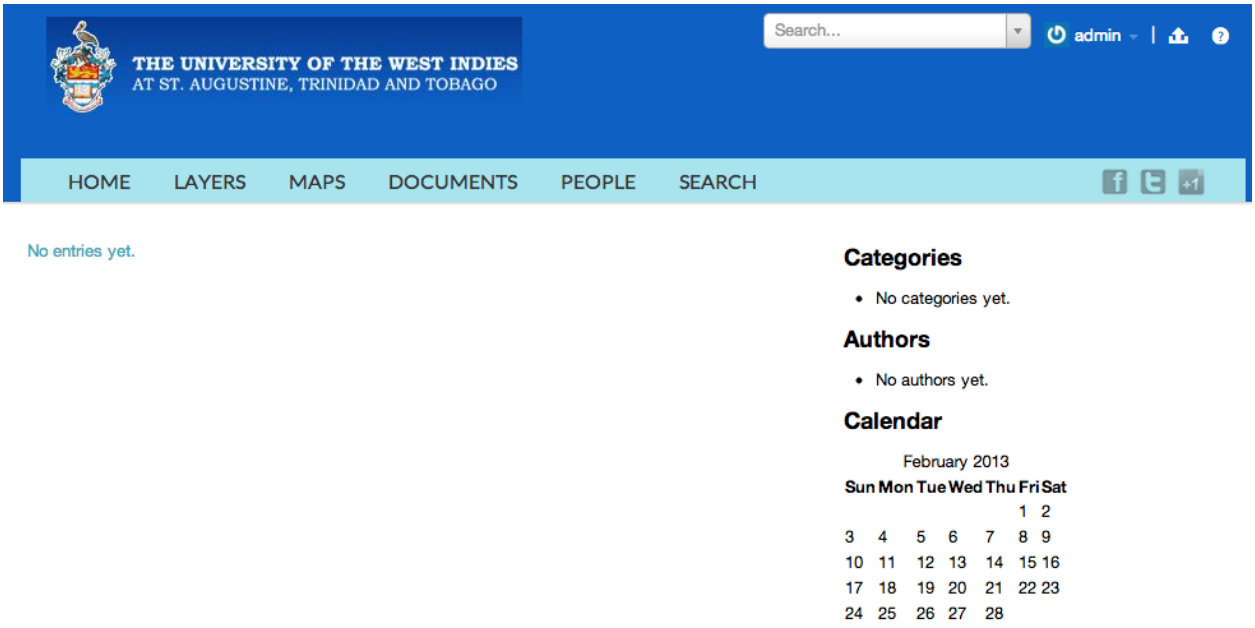
```
$ mkdir <my_geonode>/templates/zinnia
$ cp /var/lib/geonode/lib/python2.7/site-packages/zinnia/templates/zinnia/base.html <my_geonode>/temp
```

Then you need to edit this file and change the topmost line to read as below such that this template extends from our projects `site_base.html` rather than the zinnia `skeleton.html`:

```
{% extends "site_base.html" %}
```

Since Zinnia uses a different block naming scheme than GeoNode does, you need to add the following line to the bottom of your site_base.html file so that the content block gets rendered properly:

```
{% block body %}{% block content %}{% endblock %}{% endblock %}
```



You can see that there are currently no blog entries, so let's add one. Scroll to the bottom of the interface and click the *Post an Entry* link to go to the form in the Admin interface that lets you create a blog post. Go ahead and fill out the form with some information for testing purposes. Make sure that you change the Status dropdown to "published" so the post shows up right away.

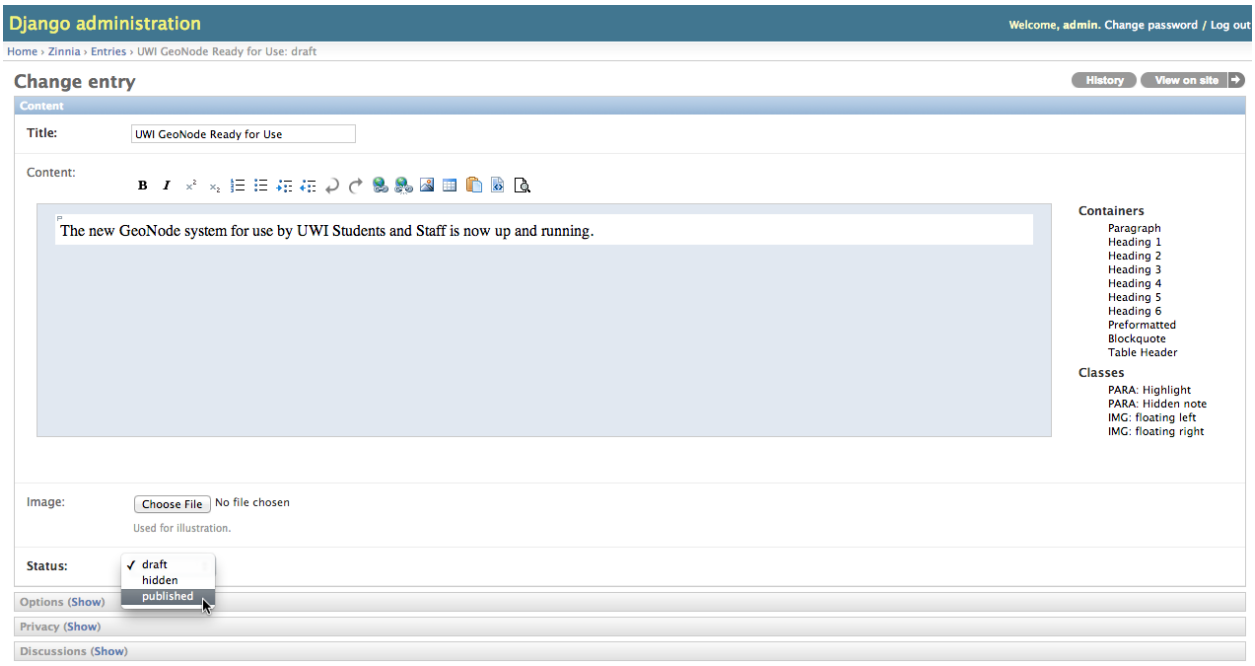
You can explore all of the options available to you as you create your post, and when you are done, click the *Save* button. You will be taken to the page that shows the list of all your blog posts.

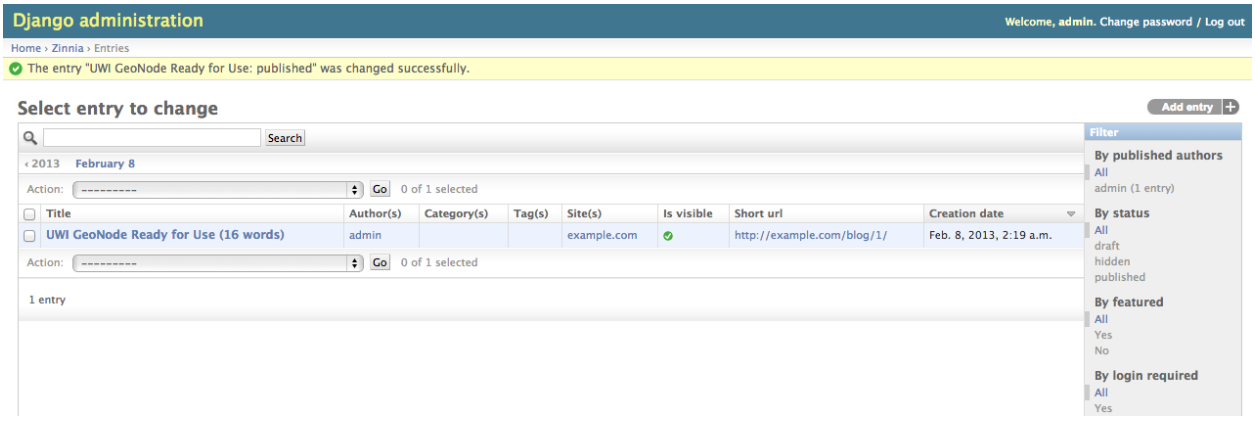
You can then visit your blog post/entry at http://localhost:8000/blog/.

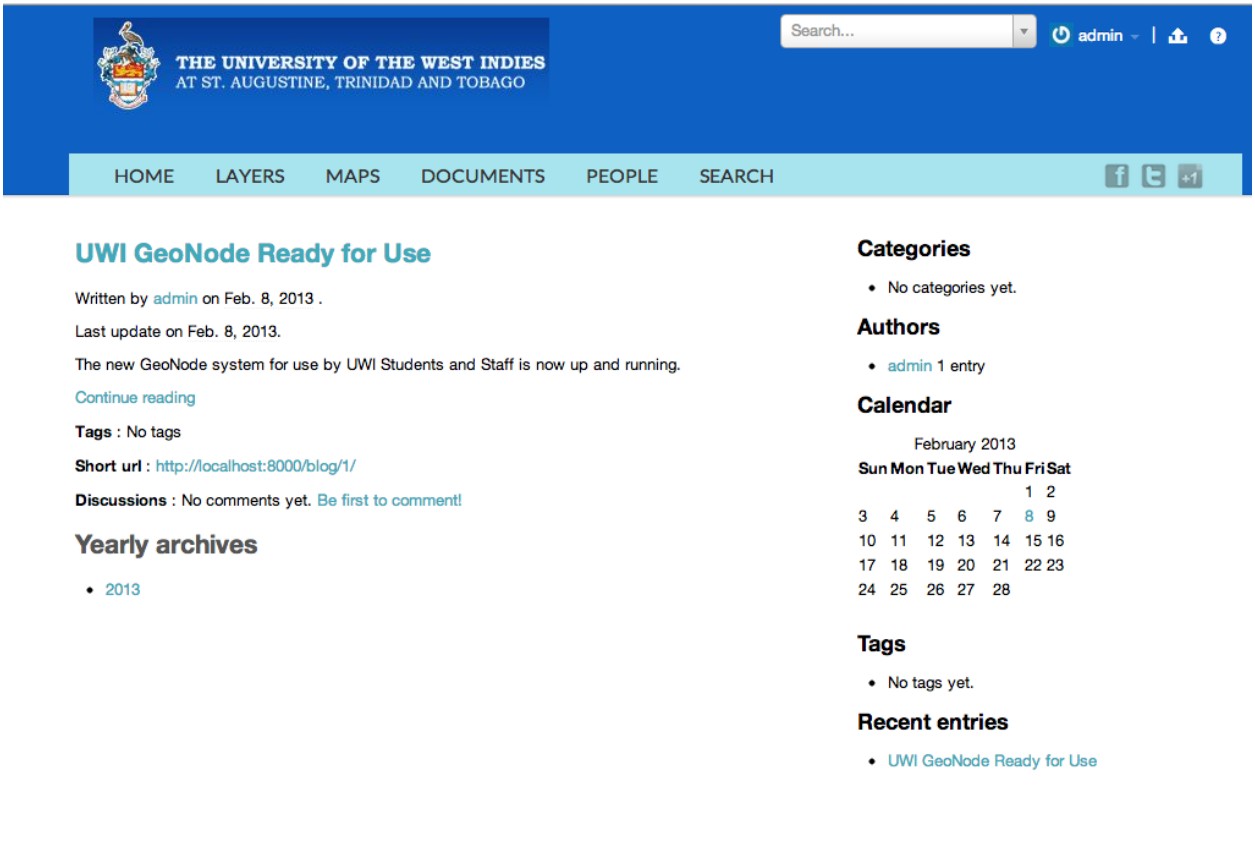
And if you click on the blog post title, you will be taken to the page for the complete blog post. You and your users can leave comments on this post and various other blog features from this page.

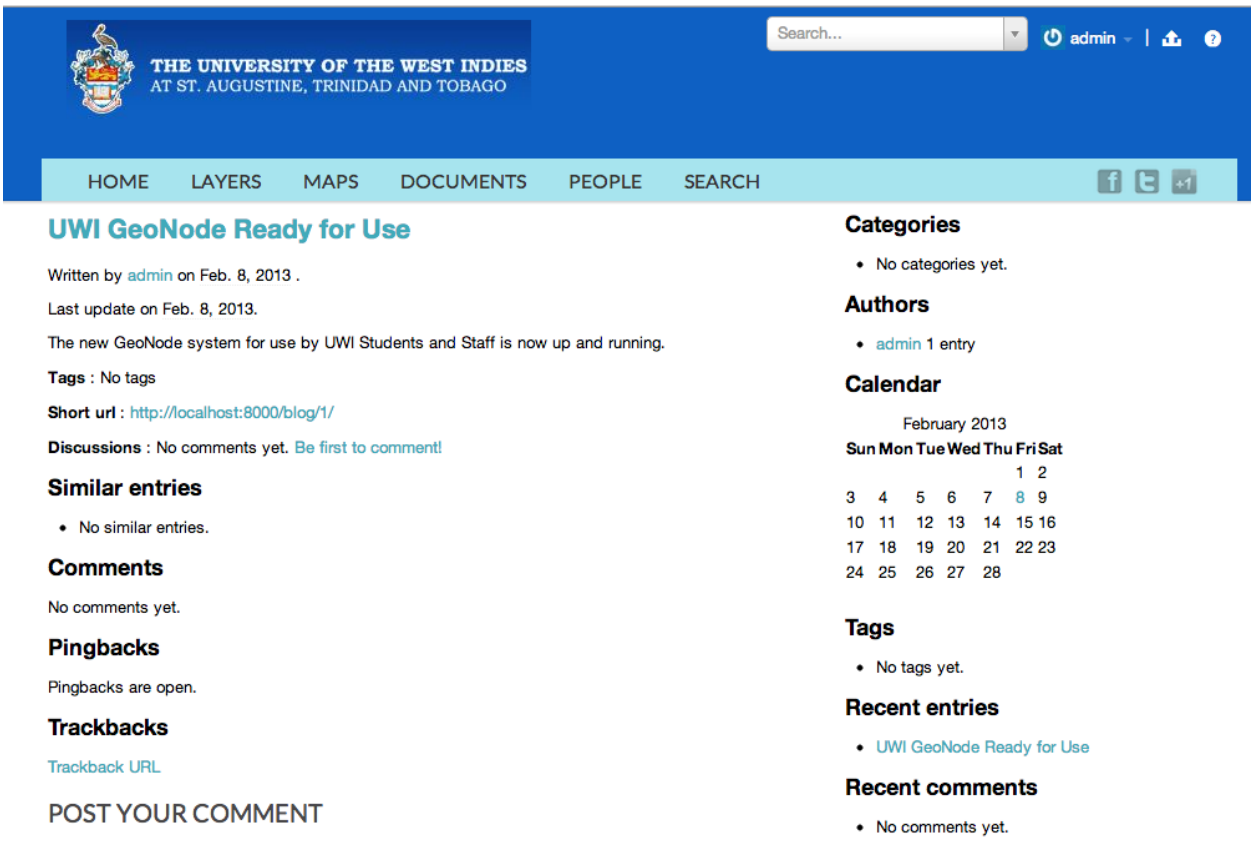
The last thing we need to do to fully integrate this blog app (and our polls app) into our site is to add it to the options on the navbar. To do so, we need to add the following block override to our Projects `site_base.html`:

```
{% block extra-nav %}
<li id="nav_polls">
    <a href="/polls/">Polls</a>
```

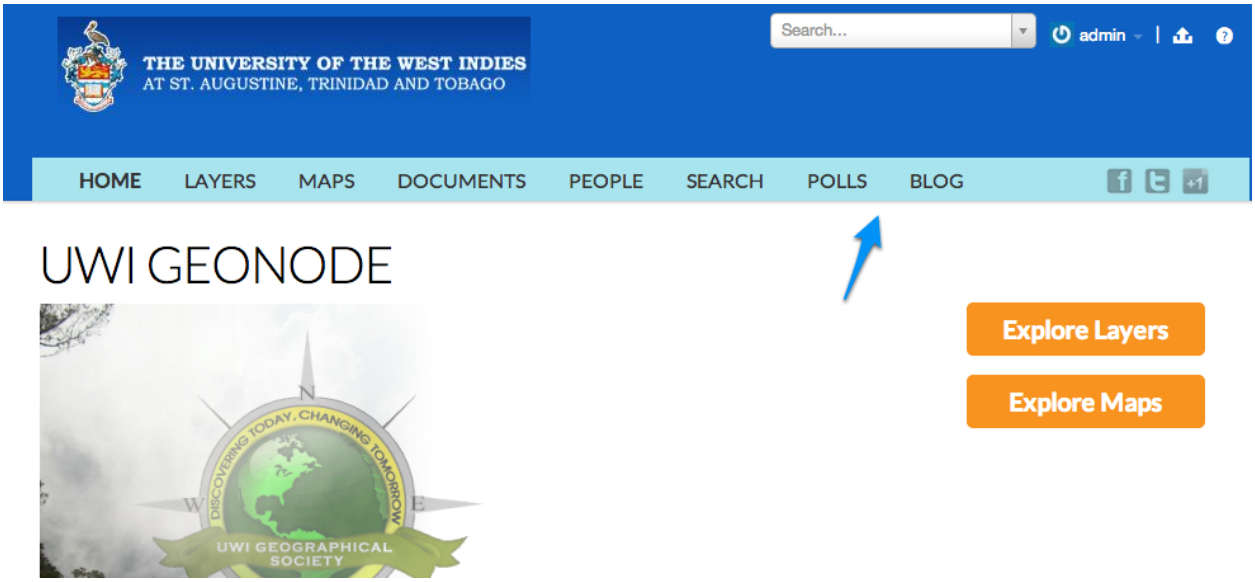

Django administration
Welcome, admin. Change password / Log out
Home › Zinnia › Entries › UWI GeoNode Ready for Use: draft
Change entry
History
View on site
Content
Title:
UWI GeoNode Ready for Use
Content:
The new GeoNode system for use by UWI Students and Staff is now up and running.
Containers
Paragraph
Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6
Preformatted
Blockquote
Table Header
Classes
PARA: Highlight
PARA: Hidden note
IMG: floating left
IMG: floating right
Image:
Choose File
No file chosen
Used for illustration.
Status:
draft
hidden
published
Options (Show)
Privacy (Show)
Discussions (Show)


Django administration
Welcome, admin. Change password / Log out
Home › Zinnia › Entries
The entry "UWI GeoNode Ready for Use: published" was changed successfully.
Select entry to change
Add entry
Search
‹ 2013 February 8
Action:
Go
0 of 1 selected
Title
Author(s)
Category(s)
Tag(s)
Site(s)
Is visible
Short url
Creation date
UWI GeoNode Ready for Use (16 words)
admin
example.com
http://example.com/blog/1/
Feb. 8, 2013, 2:19 a.m.
1 entry
Filter
By published authors
All
admin (1 entry)
By status
All
draft
hidden
published
By featured
All
Yes
No
By login required
All
Yes


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UWI GeoNode Ready for Use
Written by admin on Feb. 8, 2013 .
Last update on Feb. 8, 2013.
The new GeoNode system for use by UWI Students and Staff is now up and running.
Continue reading
Tags : No tags
Short url : http://localhost:8000/blog/1/
Discussions : No comments yet. Be first to comment!
Yearly archives
2013
Categories
No categories yet.
Authors
admin 1 entry
Calendar
February 2013
Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28
Tags
No tags yet.
Recent entries
UWI GeoNode Ready for Use


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UWI GeoNode Ready for Use
Written by admin on Feb. 8, 2013 .
Last update on Feb. 8, 2013.
The new GeoNode system for use by UWI Students and Staff is now up and running.
Tags : No tags
Short url : http://localhost:8000/blog/1/
Discussions : No comments yet. Be first to comment!
Similar entries
No similar entries.
Comments
No comments yet.
Pingbacks
Pingbacks are open.
Trackbacks
Trackback URL
POST YOUR COMMENT
Categories
No categories yet.
Authors
admin 1 entry
Calendar
February 2013
Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28
Tags
No tags yet.
Recent entries
UWI GeoNode Ready for Use
Recent comments
No comments yet.

```
</li>
<li id="nav_blog">
    <a href="{% url zinnia_entry_archive_index %}">Blog</a>
</li>
{% endblock %}
```



At this point, you could explore options for tighter integration between your GeoNode project and Zinnia. Integrating blog posts from Zinnia into your overall search could be useful, as well as including the blog posts a user has written on their Profile Page. You could also explore the additional plugins that go with Zinnia.

3.4.4 Adding other apps

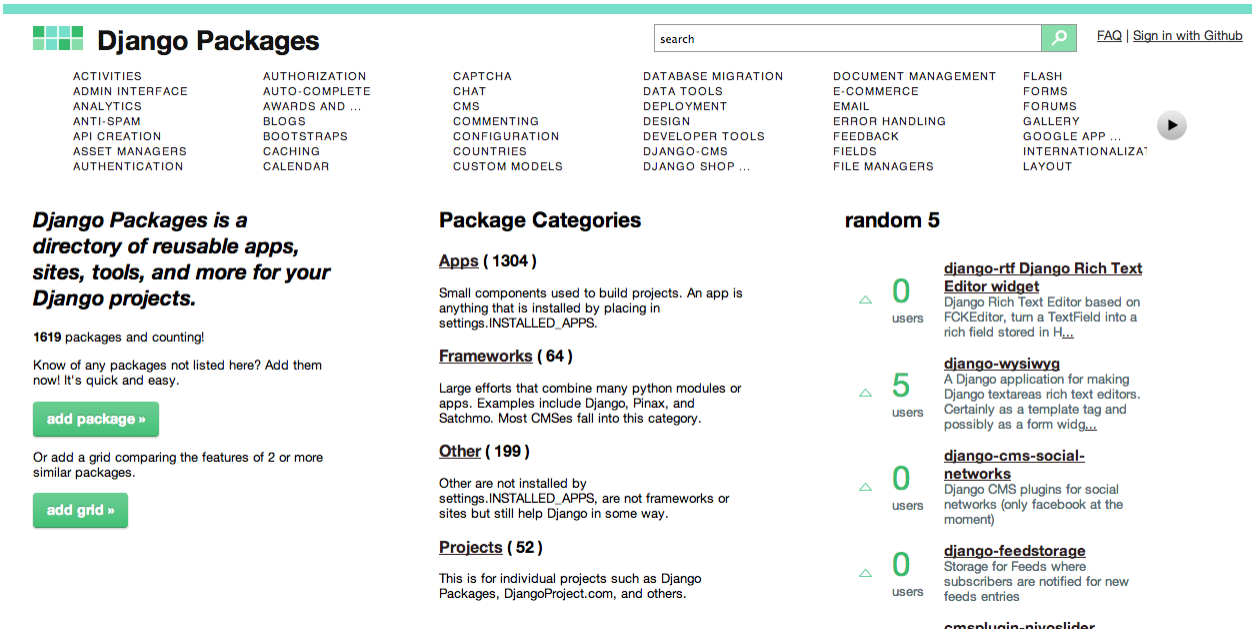
Now that you have both written your own app and plugged in a 3rd party one, you can explore sites like Django Packages to look for other modules that you could plug into your GeoNode project to meet your needs and requirements. For many types of apps, there are several options and Django Packages is a nice way to compare them. You may find that some apps require significantly more work to integrate into your app than others, but reaching out to the app's author and/or developers should help you get over any difficulties you may encounter.

3.5 Integrating your project with other systems

Your GeoNode project is based on core components which are interoperable and as such, it is straightforward for you to integrate with external applications and services. This section will walk you through how to connect to your GeoNode instance from other applications and how to integrate other services into your GeoNode project. When complete, you should have a good idea about the possibilities for integration, and have basic knowledge about how to accomplish it. You may find it necessary to dive deeper into how to do more complex integration in order to accomplish your goals, but you should feel comfortable with the basics, and feel confident reaching out to the wider GeoNode community for help.

3.5.1 OGC services

Since GeoNode is built on GeoServer which is heavily based on OGC services, the main path for integration with external services is via OGC Standards. A large number of systems, applications and services support adding WMS



layers to them, but only a few key ones are covered below. WFS and WCS are also supported in a wide variety of clients and platforms and give you access to the actual data for use in GeoProcessing or to manipulate it to meet your requirements. GeoServer also bundles GeoWebCache which produces map tiles that can be added as layers in many popular web mapping tools including Google Maps, Leaflet, OpenLayers and others. You should review the reference material included in the first chapter to learn more about OGC Services and when evaluating external systems make sure that they are also OGC Compliant in order to integrate as seamlessly as possible.

3.5.2 ArcGIS

ArcGIS Desktop (ArcMap) supports adding WMS layers to your map project. The following set of steps will walk you through how to configure a WMS Layer from your GeoNode within ArcMap.

First, you can start with a new empty project or add these layers to your existing project.

Next click the ArcCatalog button on the toolbar to bring up its interface.

From there, double click the “Add WMS Server” item in the tree to bring up the dialog that lets you enter the details for your WMS.

Next, enter the URL for your GeoNode’s WMS endpoint which is the base url with /geoserver/wms appended to the end of the URL. You can also enter your credentials into the optional Account section of this dialog to gain access to non-public layers that your user may have access to.

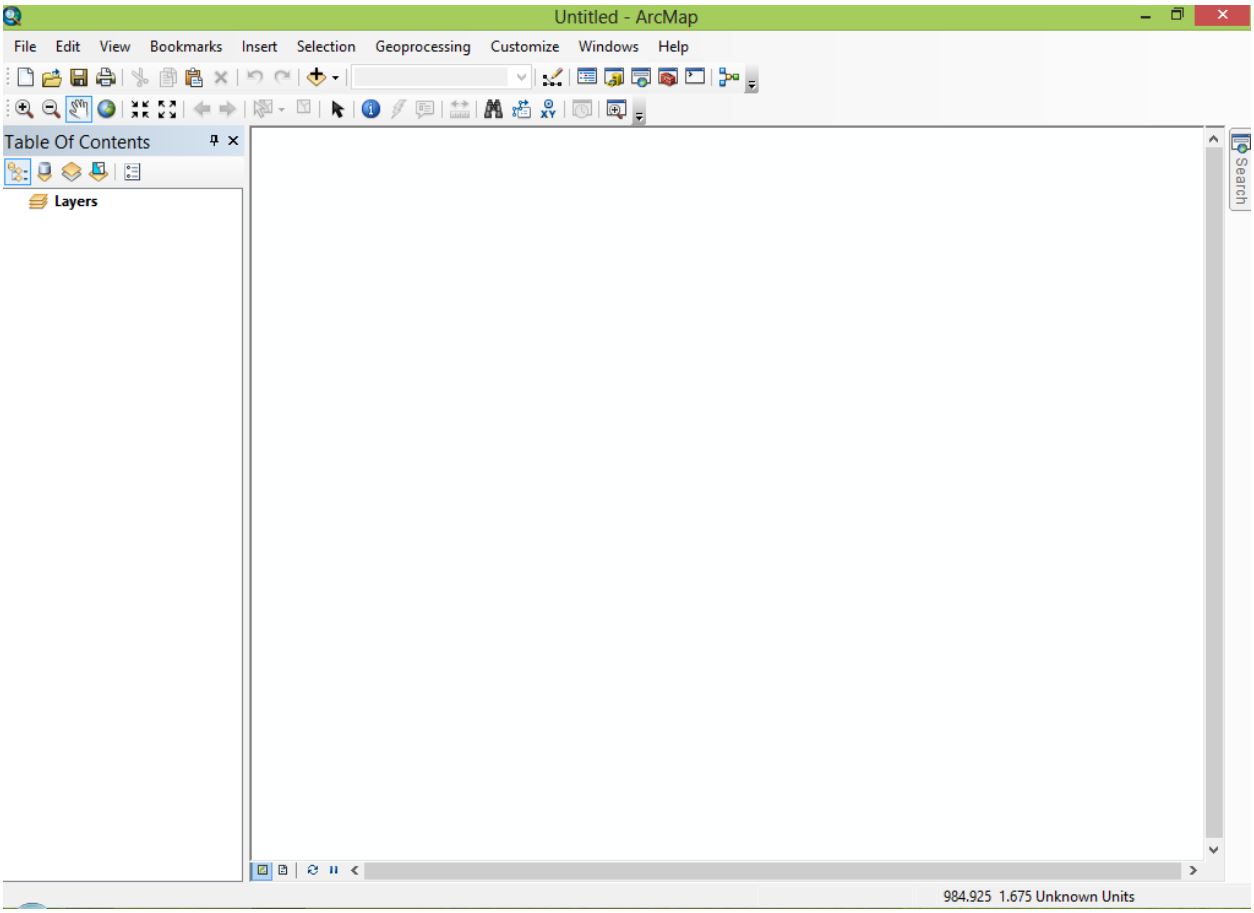
Click the “Get Layers” button to ask ArcMap to query your WMS’s GetCapabilities document to get the list of available layers.

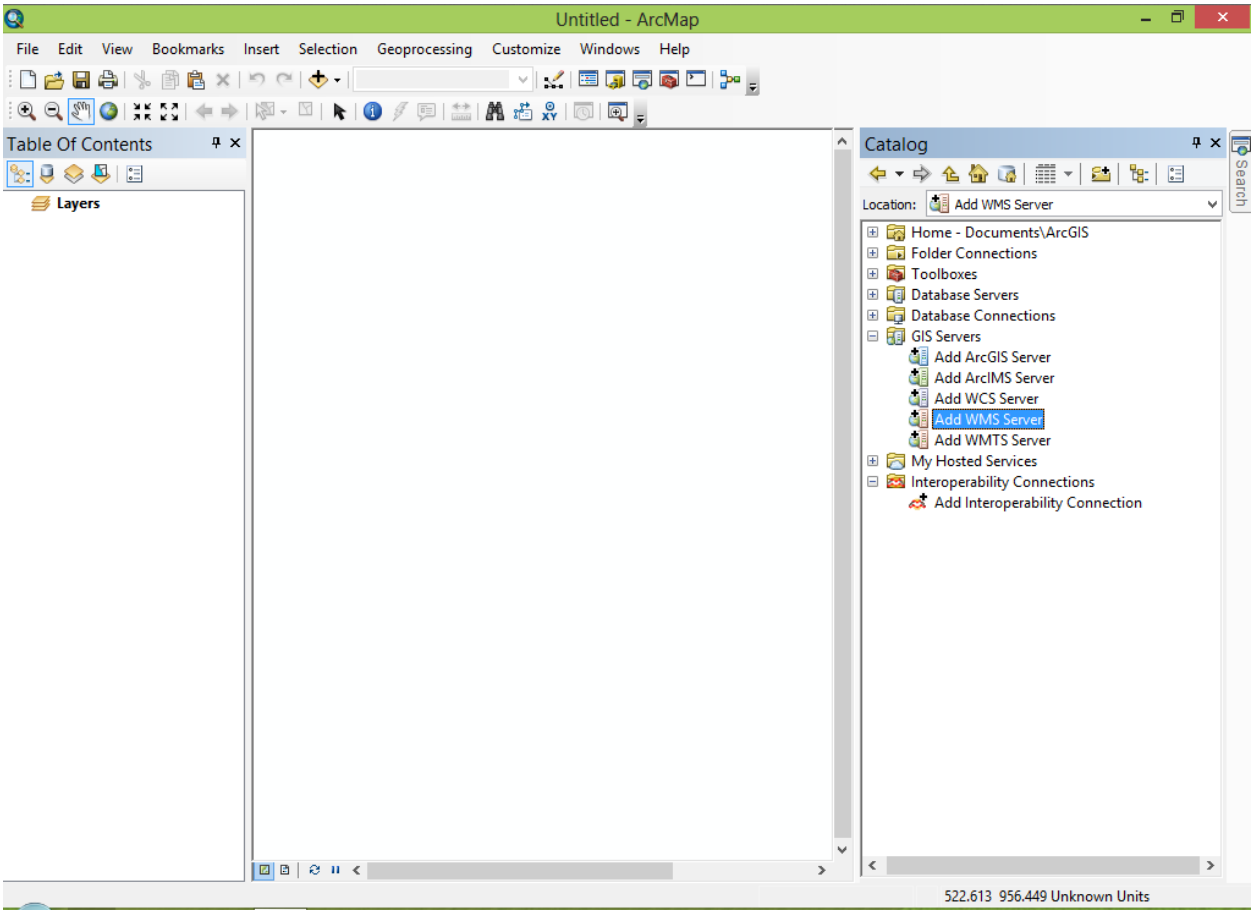
After you click the OK button, your GeoNode layers will appear in the ArcCatalog Interface.

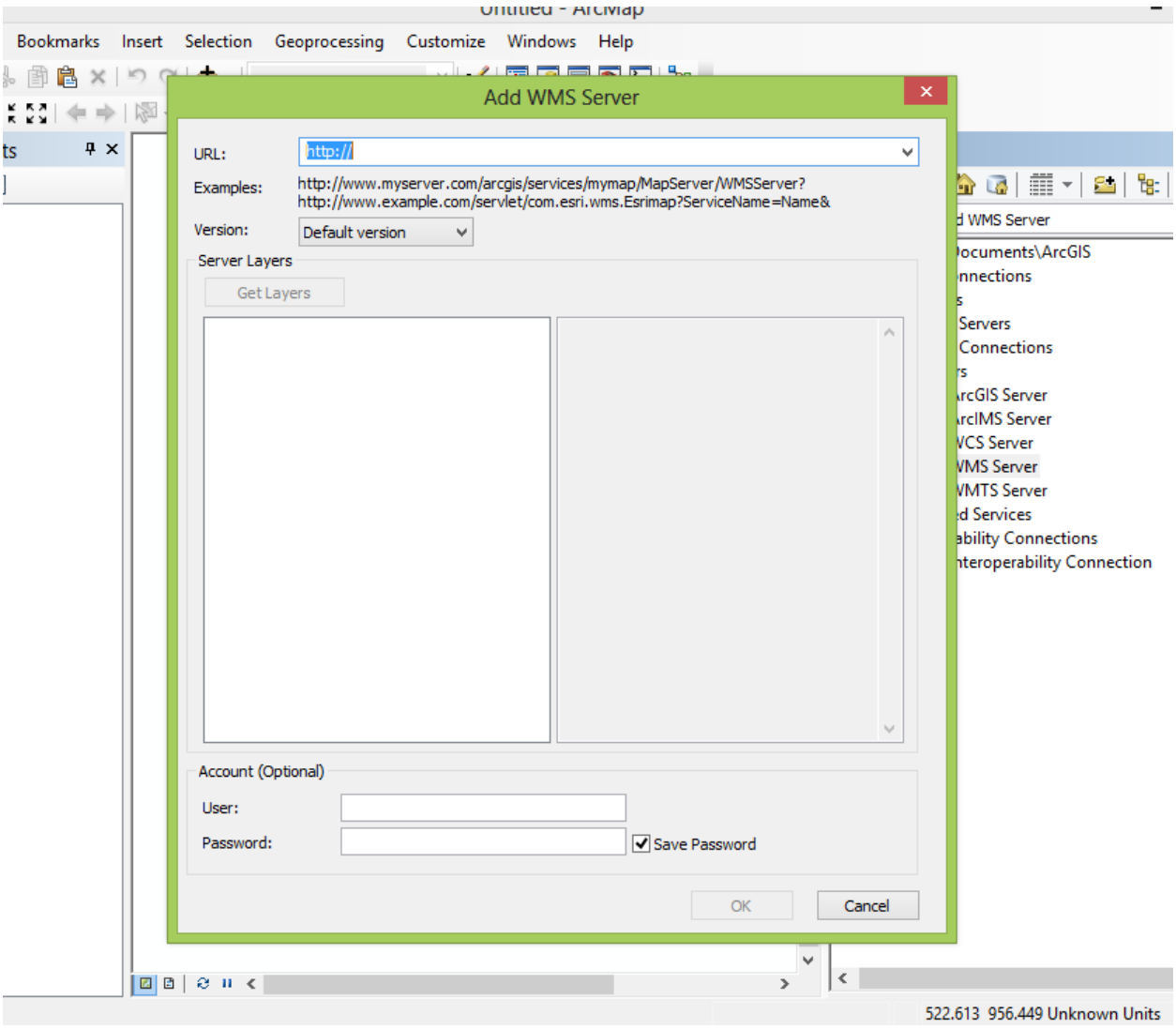
Once your server is configured in ArcMap, you can right click on one of the layers and investigate its properties.

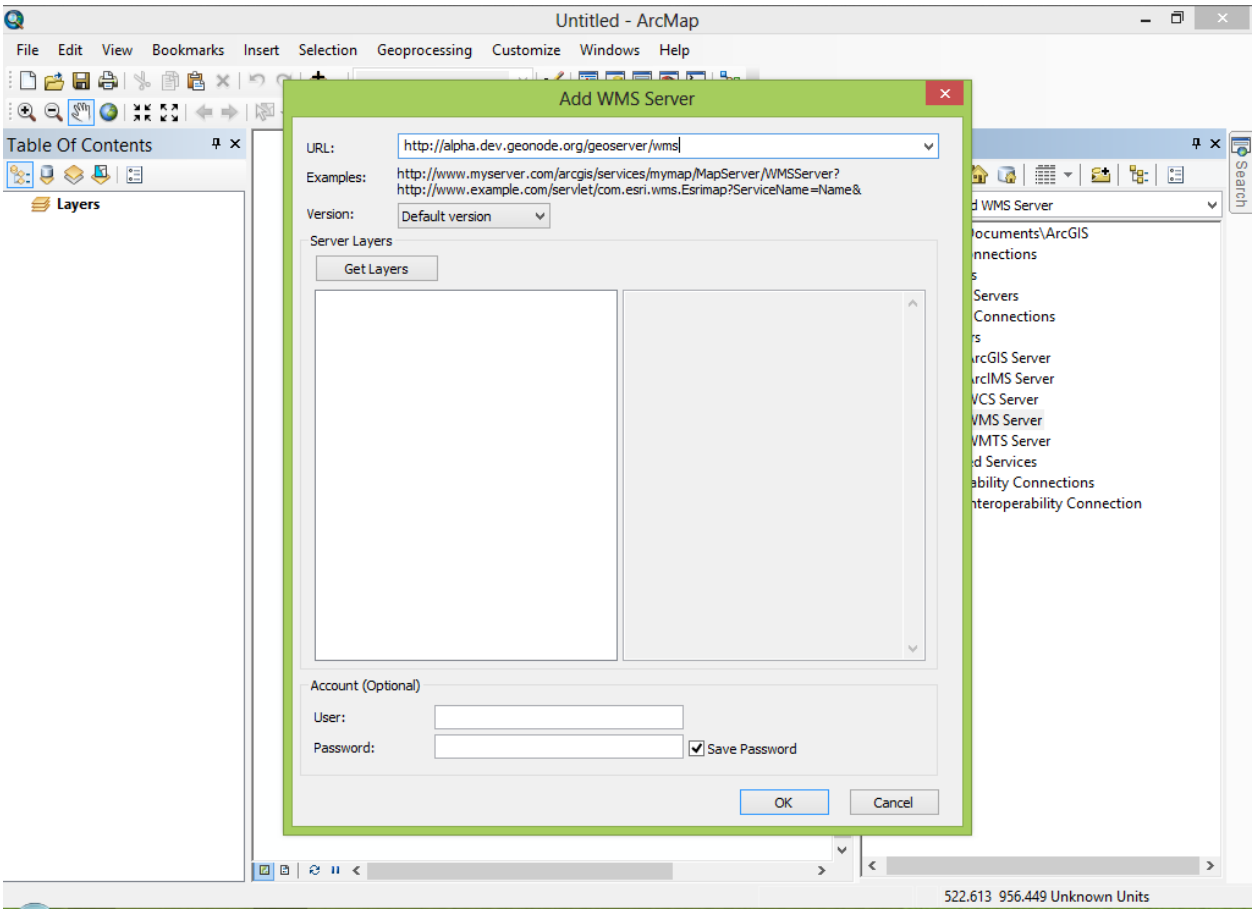
In order to actually add the layer to your project, you can drag and drop it into the Table of Contents, or right click and select “Create Layer”. Your Layer will now be displayed in the map panel of your project.

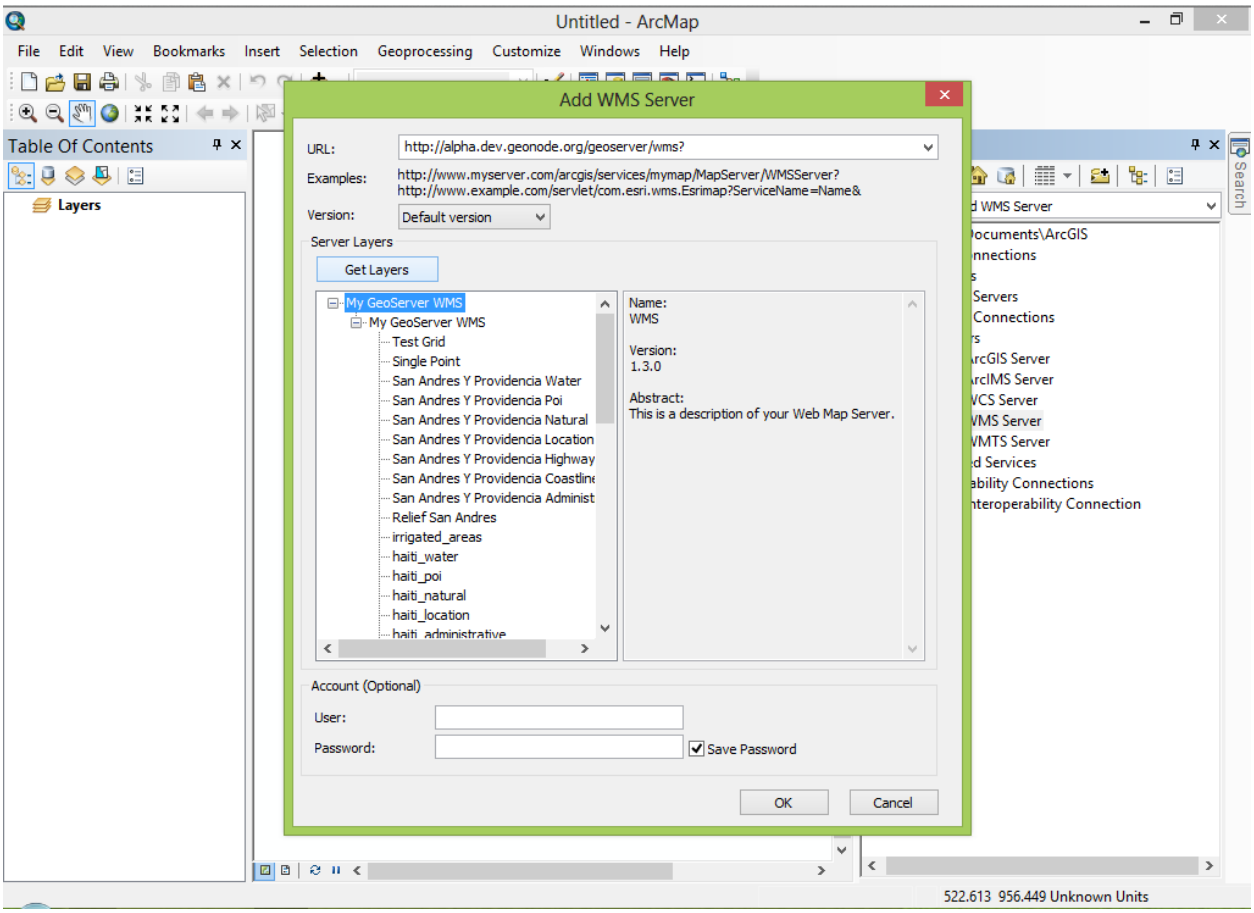
Once the layer is in your projects Table of Contents, you can right click on it and select the Layer Properties option and select the Styles Tab to choose from the available styles for that layer.

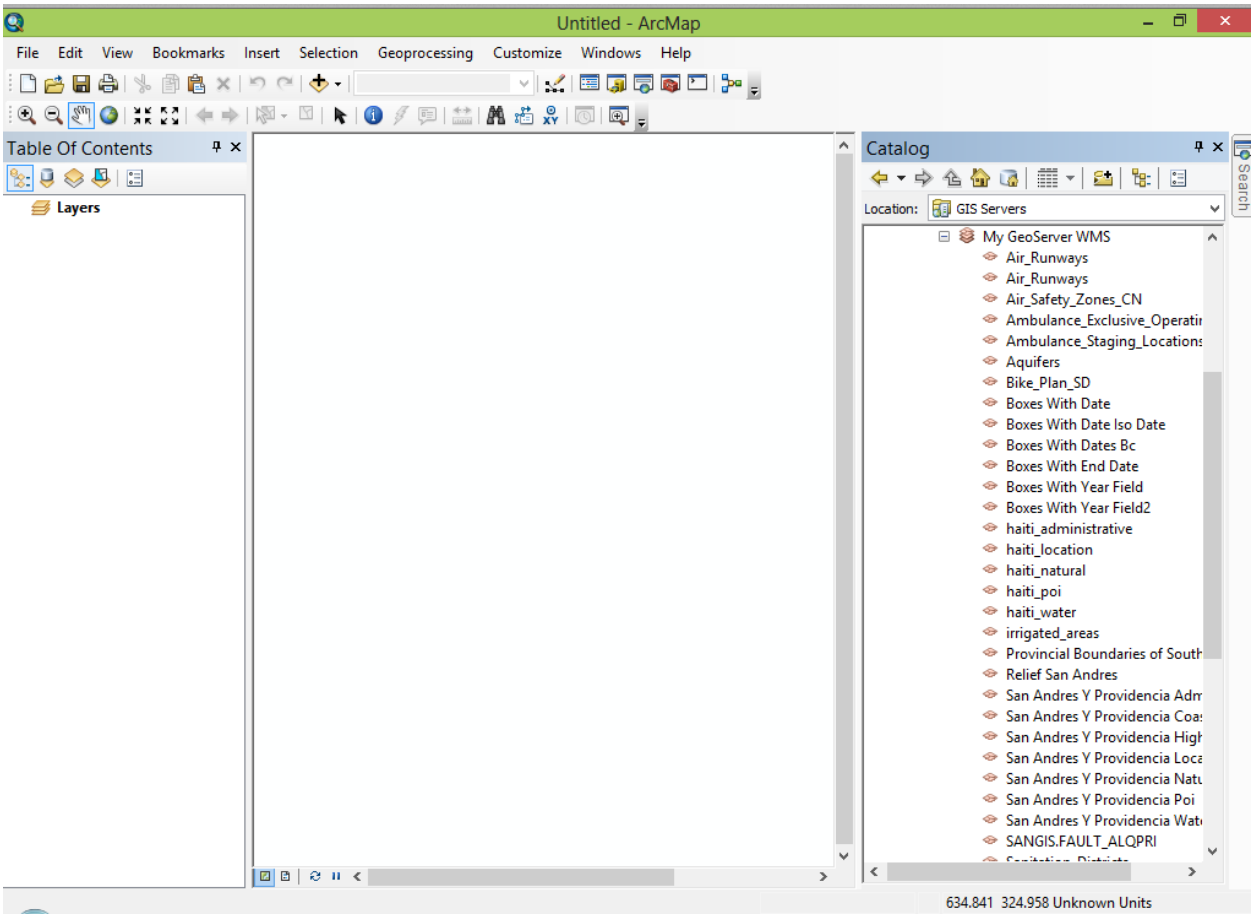

Untitled - ArcMap
File
Edit
View
Bookmarks
Insert
Selection
Geoprocessing
Customize
Windows
Help
Table Of Contents
Layers
Search
984.925 1.675 Unknown Units

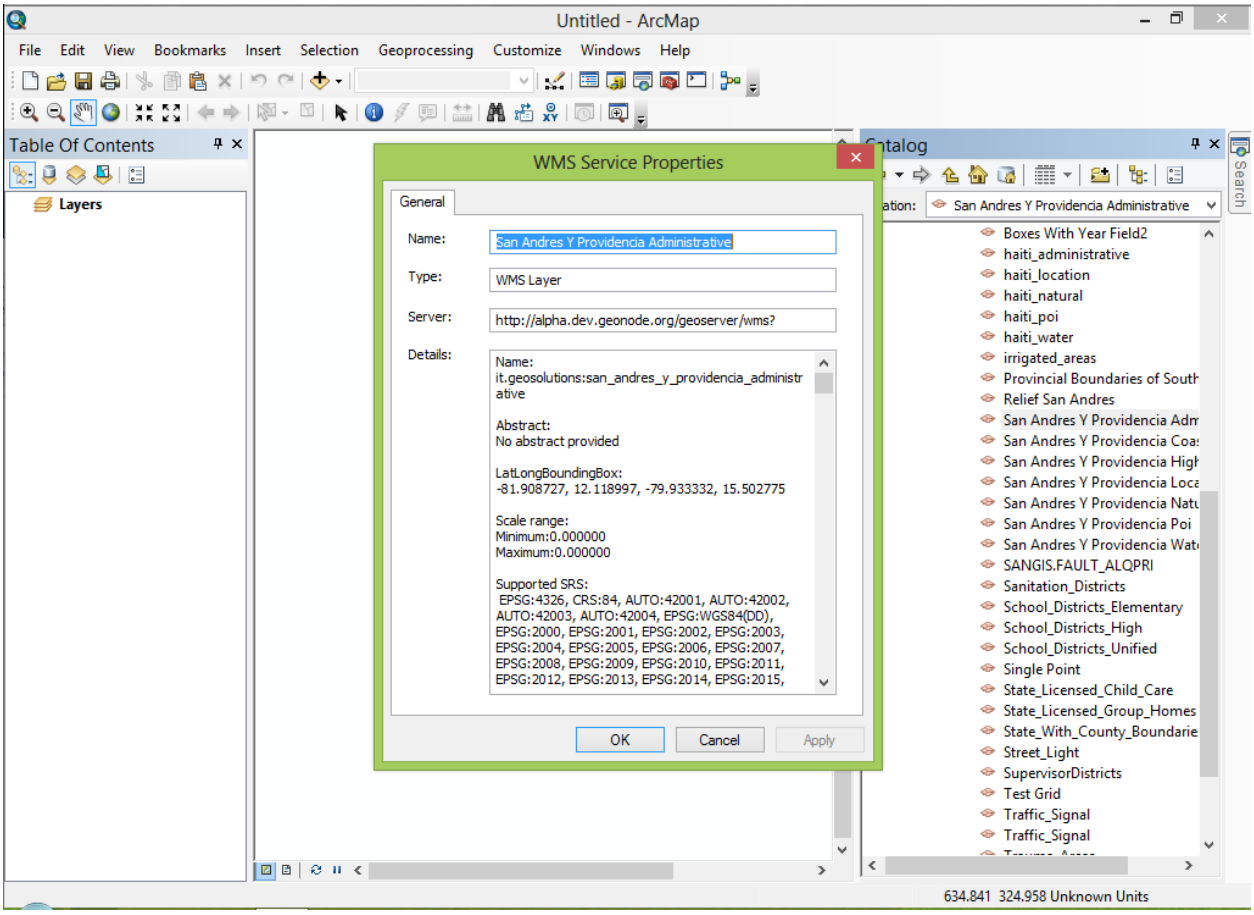

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Catalog
Location: Add WMS Server
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Search
522.613 956.449 Unknown Units

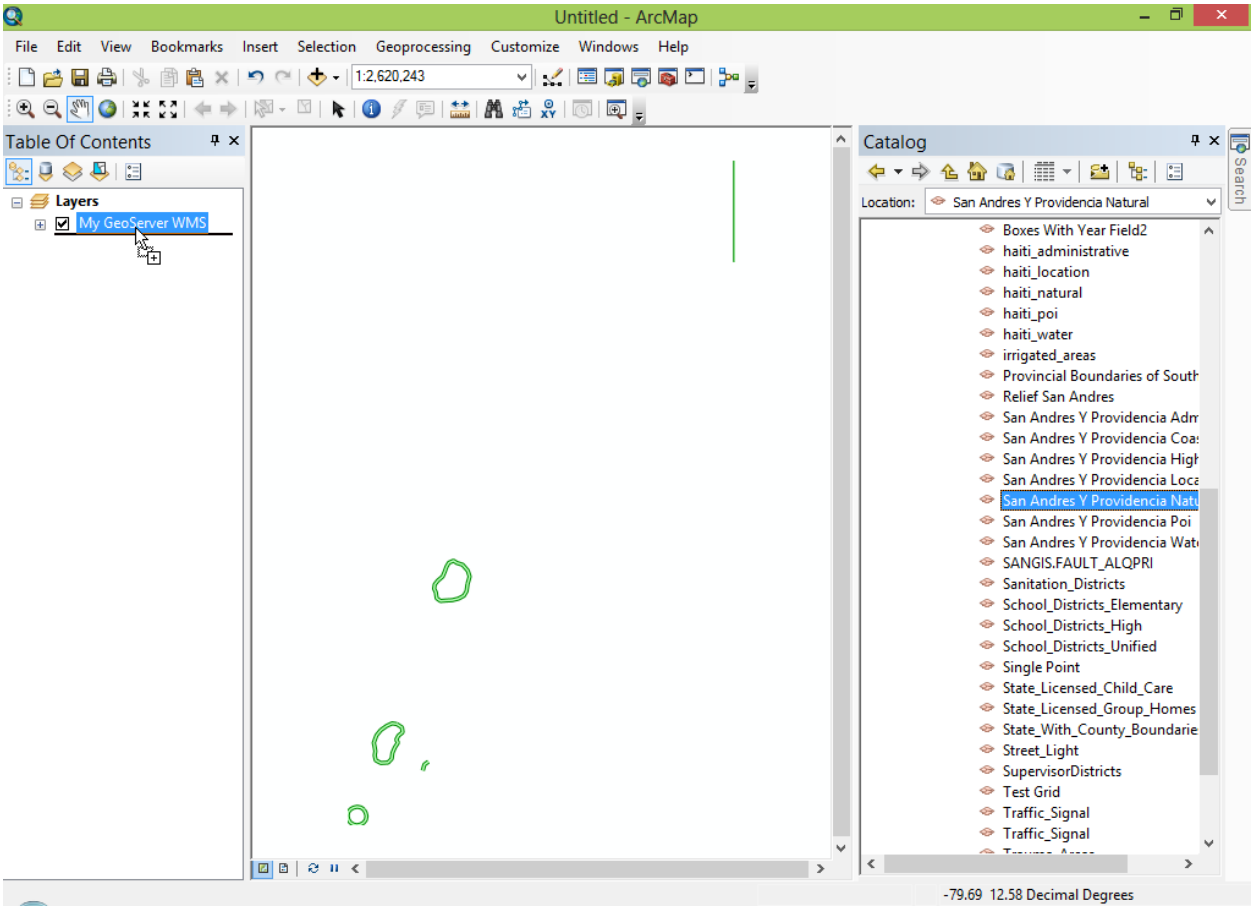

Bookmarks
Insert
Selection
Geoprocessing
Customize
Windows
Help
Add WMS Server
URL:
http://
Examples:
http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&
Version:
Default version
Server Layers
Get Layers
Account (Optional)
User:
Password:
Save Password
OK
Cancel
522.613 956.449 Unknown Units

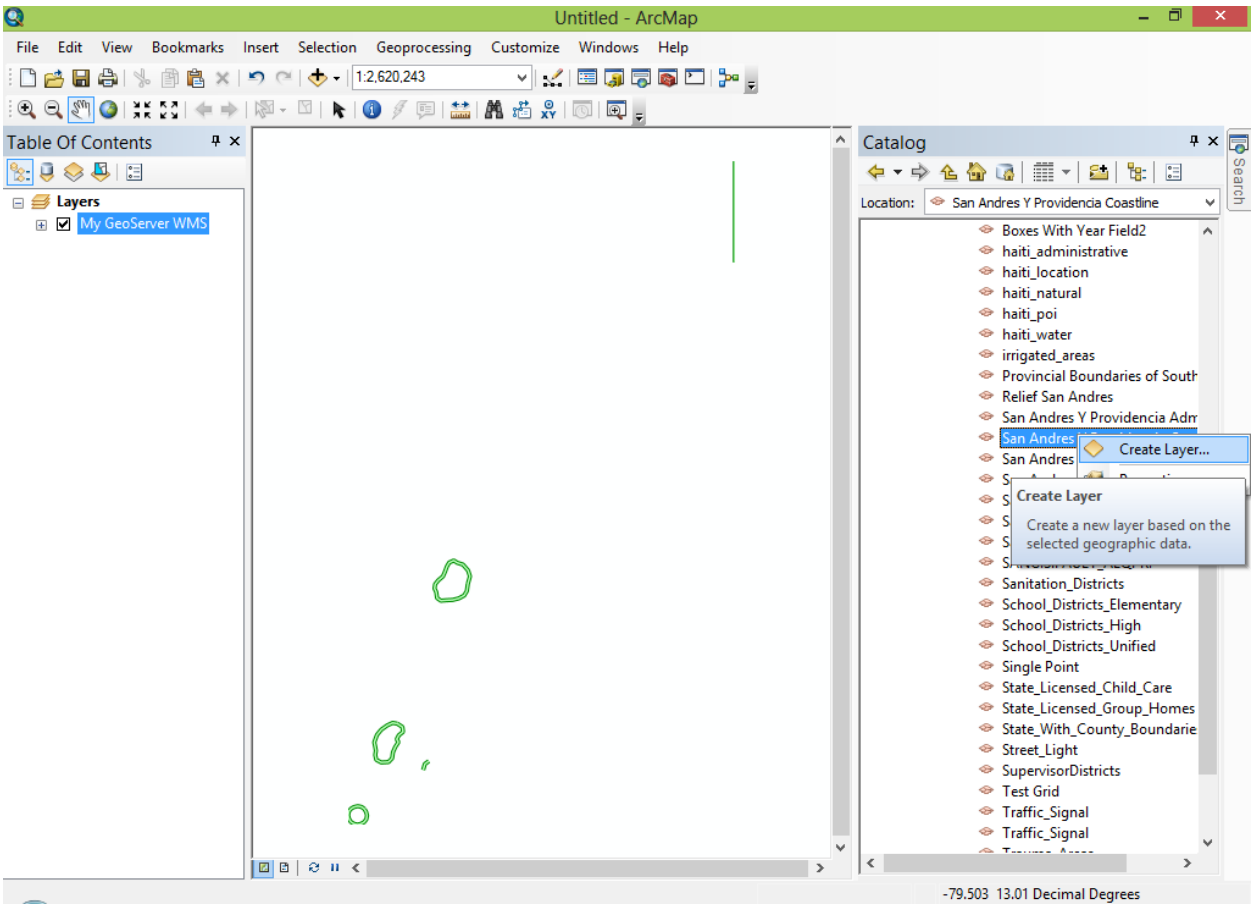

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Add WMS Server
URL:
http://alpha.dev.geonode.org/geoserver/wms
Examples:
http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&
Version:
Default version
Server Layers
Get Layers
Account (Optional)
User:
Password:
Save Password
OK
Cancel
WMS Server
Servers
Connections
ArcGIS Server
ArcIMS Server
WCS Server
WMS Server
WMTS Server
Services
522.613 956.449 Unknown Units

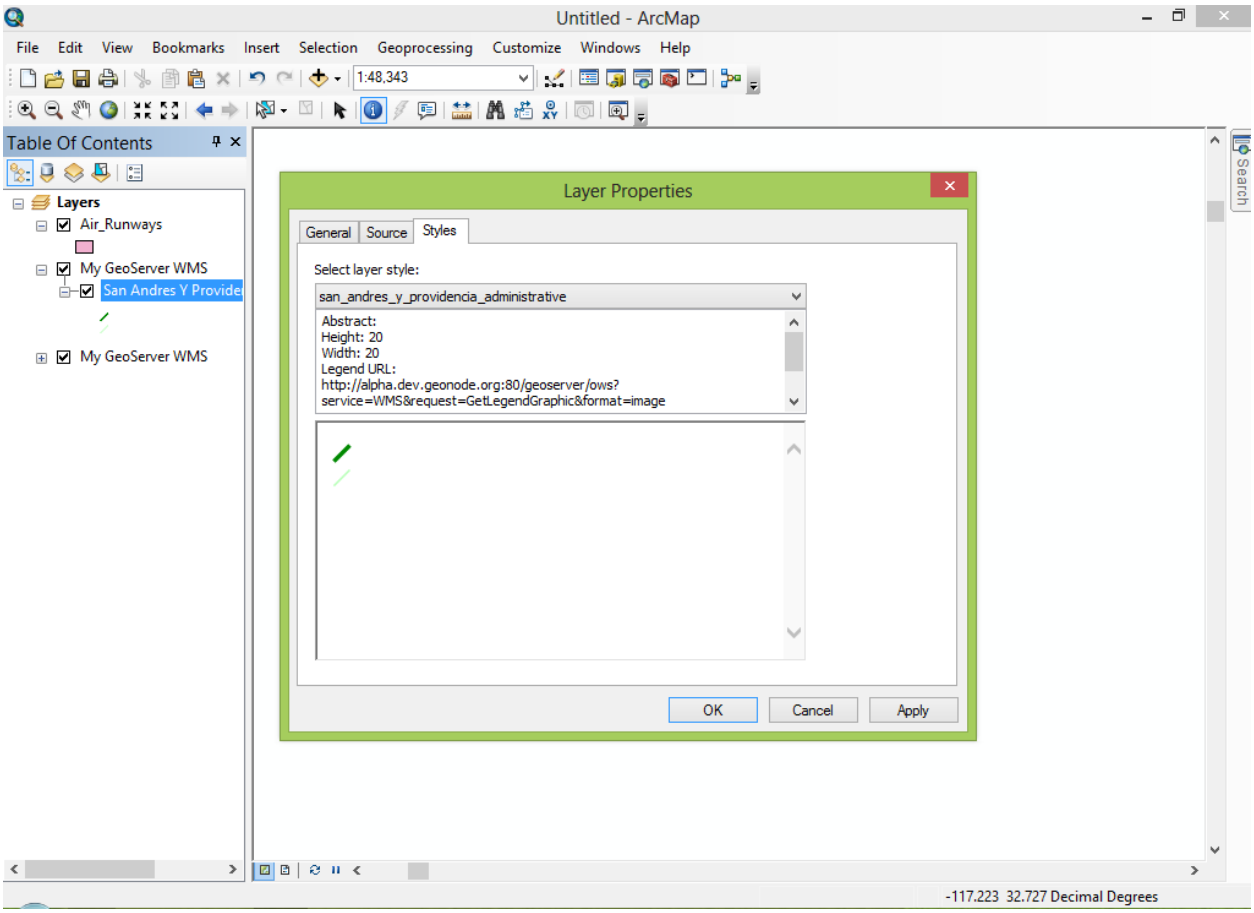

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Add WMS Server
URL:
http://alpha.dev.geonode.org/geoserver/wms?
Examples:
http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&
Version:
Default version
Server Layers
Get Layers
My GeoServer WMS
My GeoServer WMS
Test Grid
Single Point
San Andres Y Providencia Water
San Andres Y Providencia Poi
San Andres Y Providencia Natural
San Andres Y Providencia Location
San Andres Y Providencia Highway
San Andres Y Providencia Coastline
San Andres Y Providencia Administ
Relief San Andres
irrigated_areas
haiti_water
haiti_poi
haiti_natural
haiti_location
haiti_administrative
Name:
WMS
Version:
1.3.0
Abstract:
This is a description of your Web Map Server.
Account (Optional)
User:
Password:
Save Password
OK
Cancel
WMS Server
Documents\ArcGIS
Connections
Servers
Connections
ArcGIS Server
ArcIMS Server
WCS Server
WMS Server
WMTS Server
Services
ability Connections
nteroperability Connection
Search
522.613 956.449 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Catalog
Location: GIS Servers
My GeoServer WMS
Air_Runways
Air_Runways
Air_Safety_Zones_CN
Ambulance_Exclusive_Operati
Ambulance_Staging_Location
Aquifers
Bike_Plan_SD
Boxes With Date
Boxes With Date Iso Date
Boxes With Dates Bc
Boxes With End Date
Boxes With Year Field
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres Y Providencia Coa
San Andres Y Providencia Hig
San Andres Y Providencia Loc
San Andres Y Providencia Nat
San Andres Y Providencia Poi
San Andres Y Providencia Wat
SANGIS.FAULT_ALQPRI
Search
634.841 324.958 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
WMS Service Properties
General
Name:
San Andres Y Providencia Administrative
Type:
WMS Layer
Server:
http://alpha.dev.geonode.org/geoserver/wms?
Details:
Name:
it.geosolutions:san_andres_y_providencia_administrative
Abstract:
No abstract provided
LatLongBoundingBox:
-81.908727, 12.118997, -79.933332, 15.502775
Scale range:
Minimum:0.000000
Maximum:0.000000
Supported SRS:
EPSG:4326, CRS:84, AUTO:42001, AUTO:42002, AUTO:42003, AUTO:42004, EPSG:WGS84(DD), EPSG:2000, EPSG:2001, EPSG:2002, EPSG:2003, EPSG:2004, EPSG:2005, EPSG:2006, EPSG:2007, EPSG:2008, EPSG:2009, EPSG:2010, EPSG:2011, EPSG:2012, EPSG:2013, EPSG:2014, EPSG:2015,
OK
Cancel
Apply
Catalog
San Andres Y Providencia Administrative
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Relief San Andres
Single Point
State_Licensed_Child_Care
Street_Light
SupervisorDistricts
Test Grid
Traffic_Signal
Traffic_Signal
634.841 324.958 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:2,620,243
Table Of Contents
Layers
My GeoServer WMS
Catalog
Location: San Andres Y Providencia Natural
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres Y Providencia Coa
San Andres Y Providencia High
San Andres Y Providencia Loca
San Andres Y Providencia Natu
San Andres Y Providencia Poi
San Andres Y Providencia Wat
SANGIS.FAULT_ALQPRI
Sanitation_Districts
School_Districts_Elementary
School_Districts_High
School_Districts_Unified
Single Point
State_Licensed_Child_Care
State_Licensed_Group_Homes
State_With_County_Boundarie
Street_Light
SupervisorDistricts
Test Grid
Traffic_Signal
Traffic_Signal
Search
-79.69 12.58 Decimal Degrees

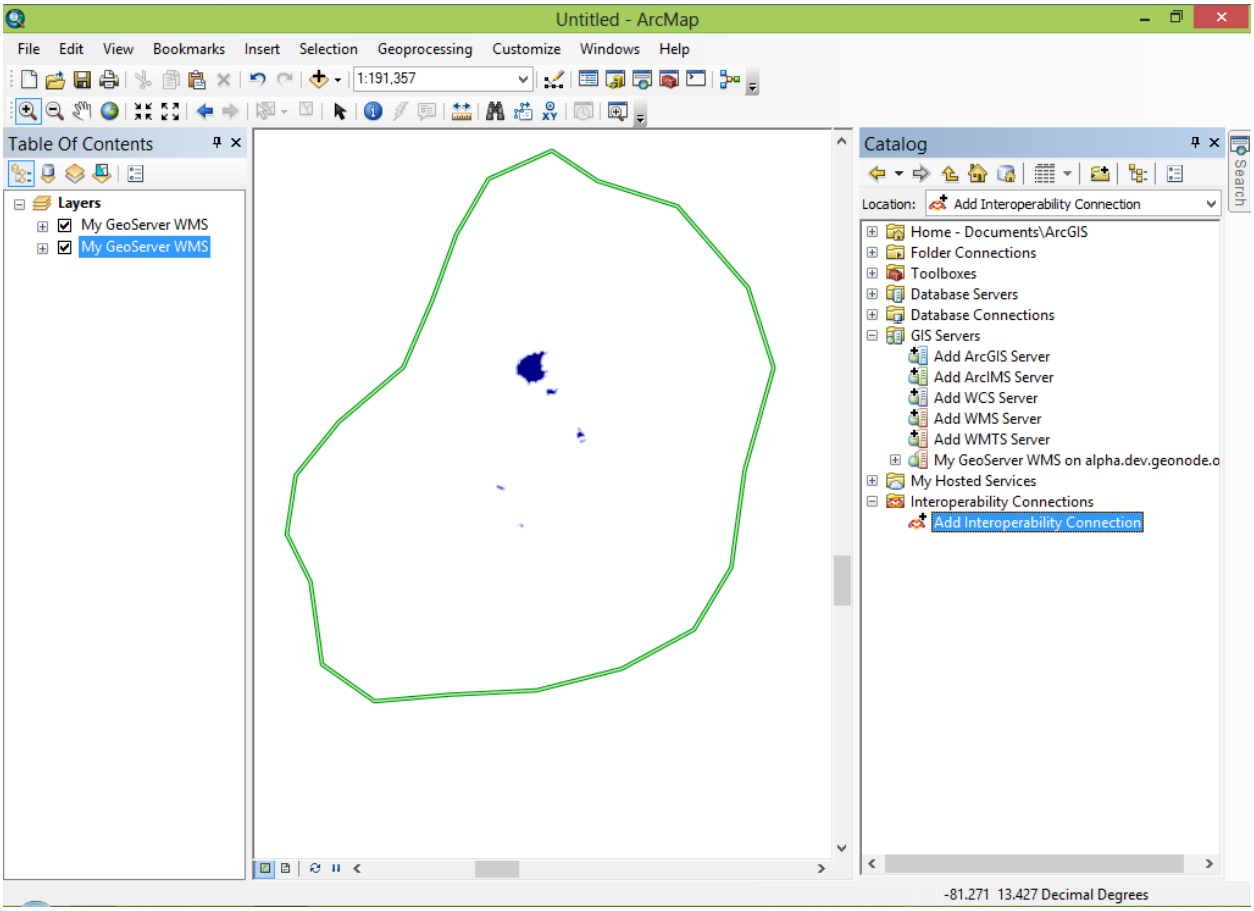

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:2,620,243
Table Of Contents
Layers
My GeoServer WMS
Catalog
Location: San Andres Y Providencia Coastline
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres
San Andres
Create Layer...
Create Layer
Create a new layer based on the selected geographic data.
Sanitation_Districts
School_Districts_Elementary
School_Districts_High
School_Districts_Unified
Single Point
State_Licensed_Child_Care
State_Licensed_Group_Homes
State_With_County_Boundarie
Street_Light
SupervisorDistricts
Test Grid
Traffic_Signal
Traffic_Signal
-79.503 13.01 Decimal Degrees


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:48,343
Table Of Contents
Layers
Air_Runways
My GeoServer WMS
San Andres Y Provide
My GeoServer WMS
Layer Properties
General Source Styles
Select layer style:
san_andres_y_providencia_administrative
Abstract:
Height: 20
Width: 20
Legend URL:
http://alpha.dev.geonode.org:80/geoserver/ows?
service=WMS&request=GetLegendGraphic&format=image
OK Cancel Apply
Search
-117.223 32.727 Decimal Degrees

Now that we have seen how to add a WMS layer to our ArcMap project, lets walk through how to add the same layers as a WFS which retrieves the actual feature data from your GeoNode rather than a rendered map as you get with WMS. Adding layers as a WFS gives you more control over how the layers are styled within ArcMap and makes them available for you to use with other ArcGIS tools like the Geoprocessing toolbox.

Note: Adding WFS layers to ArcMap requires that you have the Data Interoperability Extension installed. This extension is not included in ArcMap by default and is licensed and installed separately.

Start by opening up the ArcCatalog Interface within ArcMap and make sure that you have the "Interoperability Connections" option listed in the list.



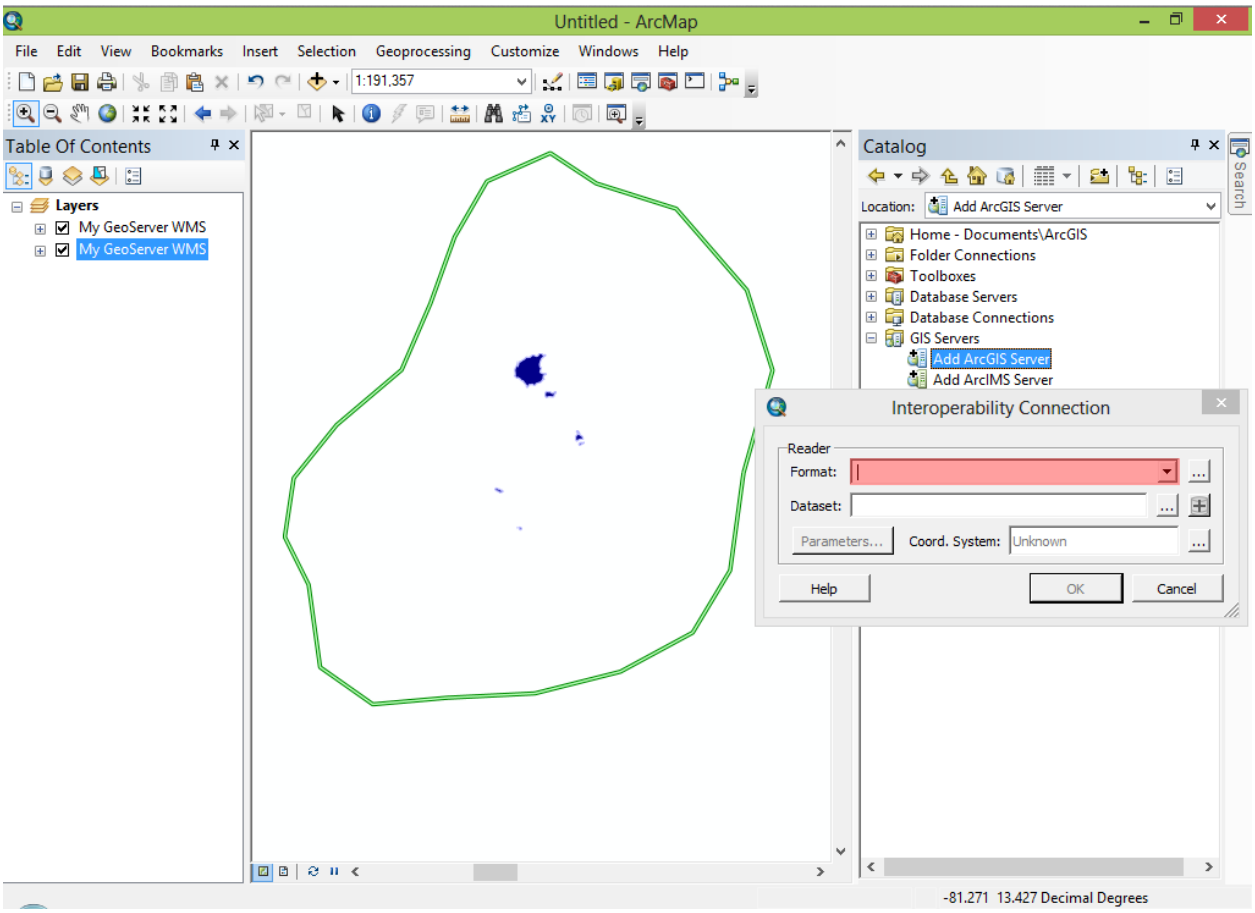
Next select "Add Interoperability Connection" to bring up the dialog that lets you add the WFS endpoint from your GeoNode.

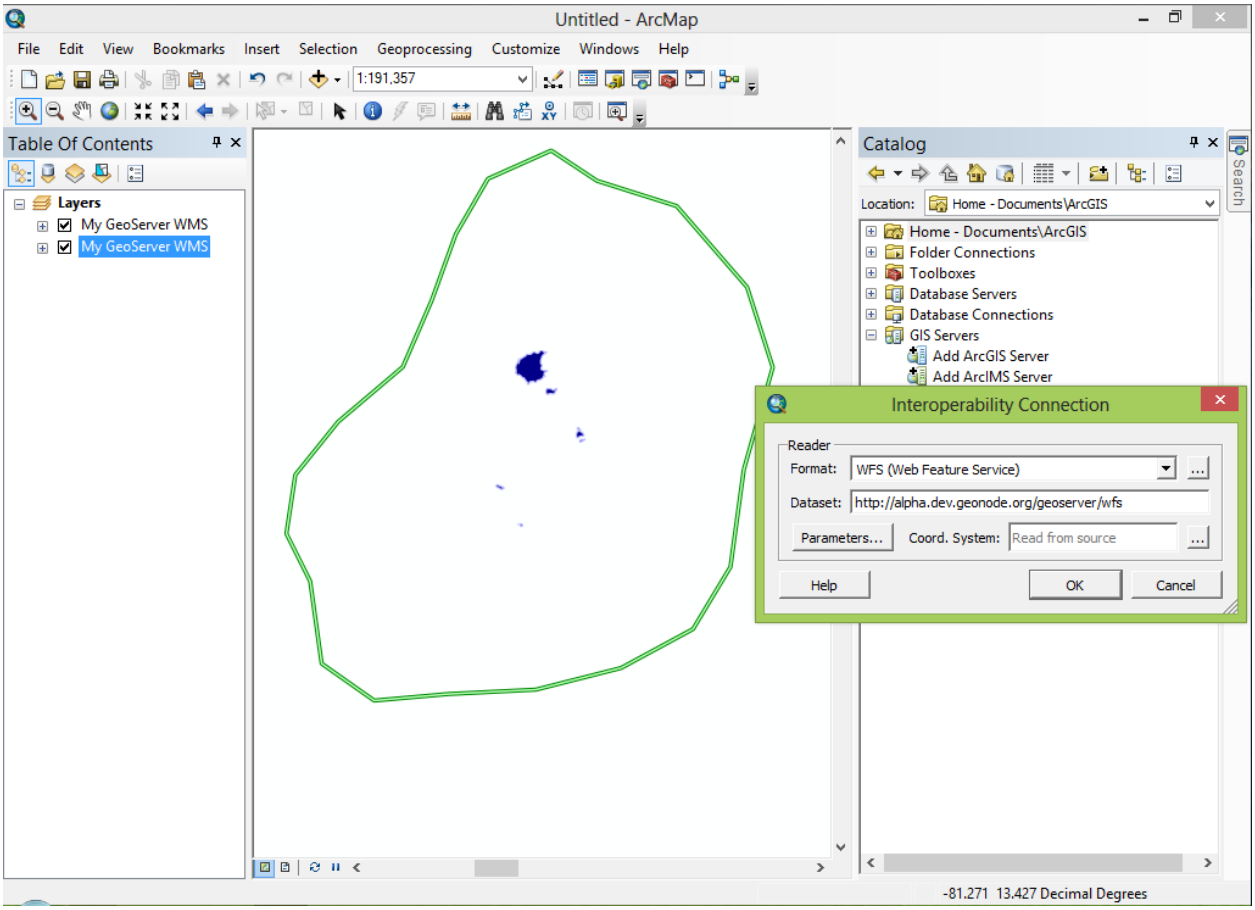
Select "WFS (Web Feature Service)" in the Format dropdown and enter the URL to the WFS endpoint for your GeoNode in the Dataset field. The WFS endpoint is your base URL + /geoserver/wfs

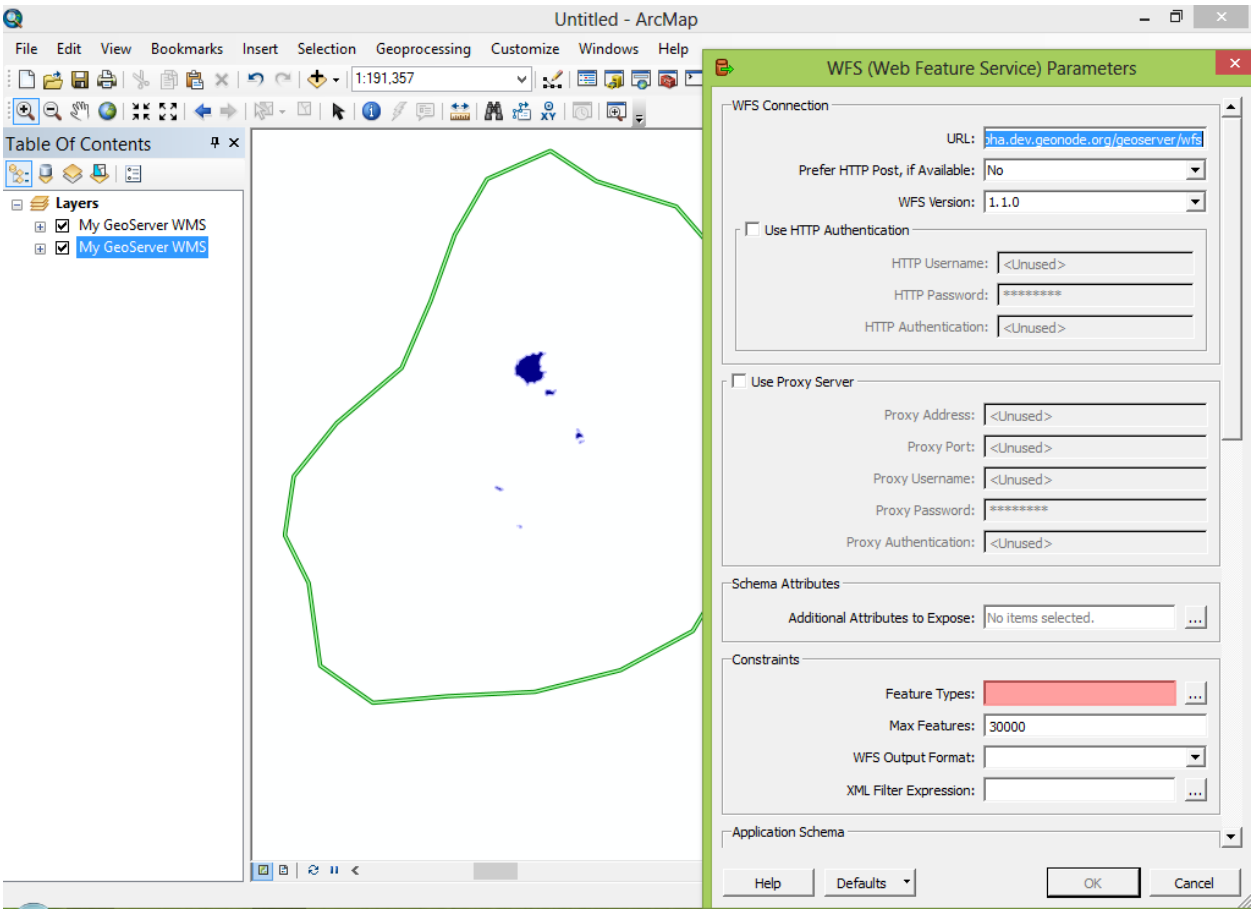
You will need to click the "Parameters" button to supply more connection information including your credentials which will give you the ability to use private layers that you have access to.

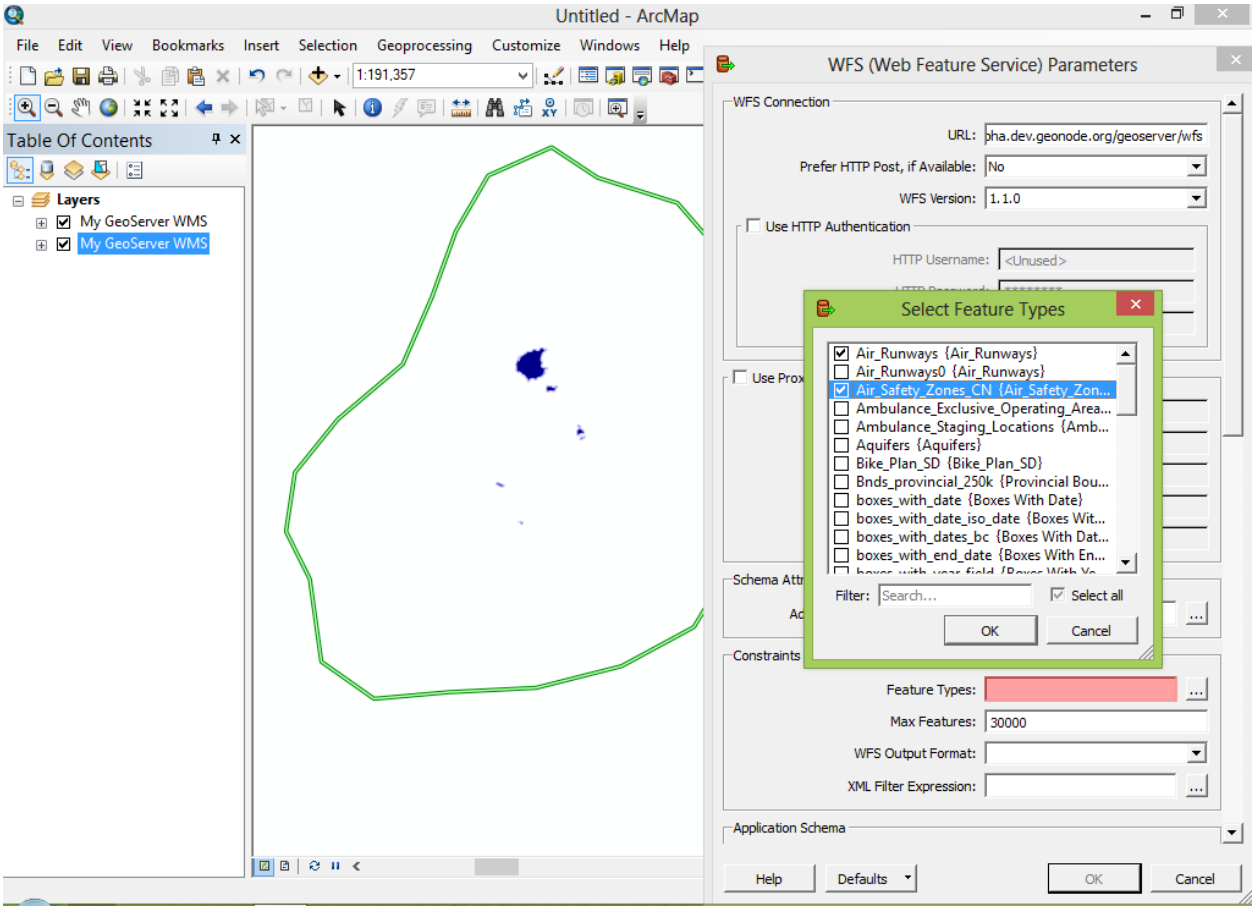
Select the Feature Types button to have ArcMap get a list of layers from the WFS Service of your GeoNode.

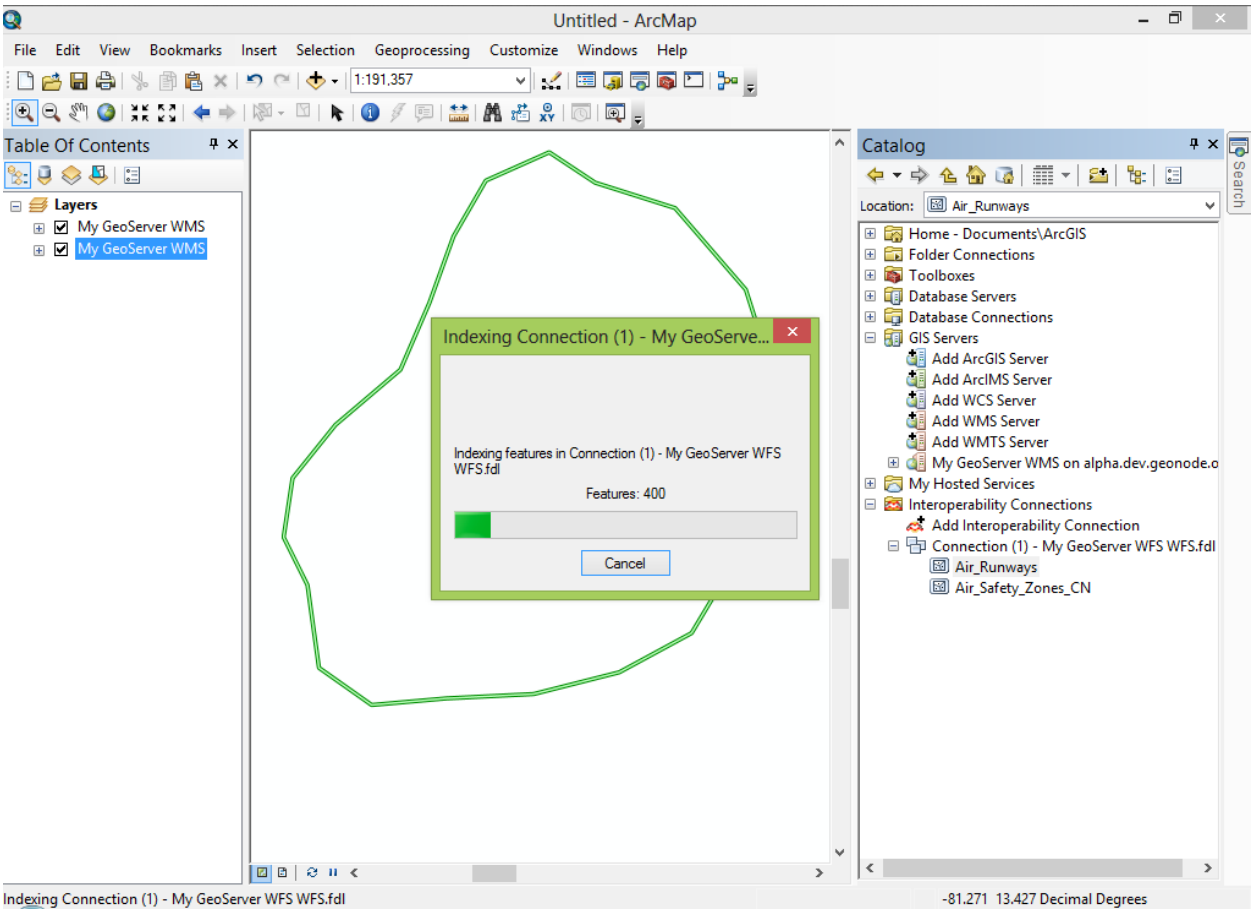
Select the layers that you want to add and click OK and ArcMap will import the features from your GeoNode into the system.


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
Catalog
Location: Add ArcGIS Server
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Search
Interoperability Connection
Reader
Format:
Dataset:
Parameters...
Coord. System:
Unknown
Help
OK
Cancel
-81.271 13.427 Decimal Degrees

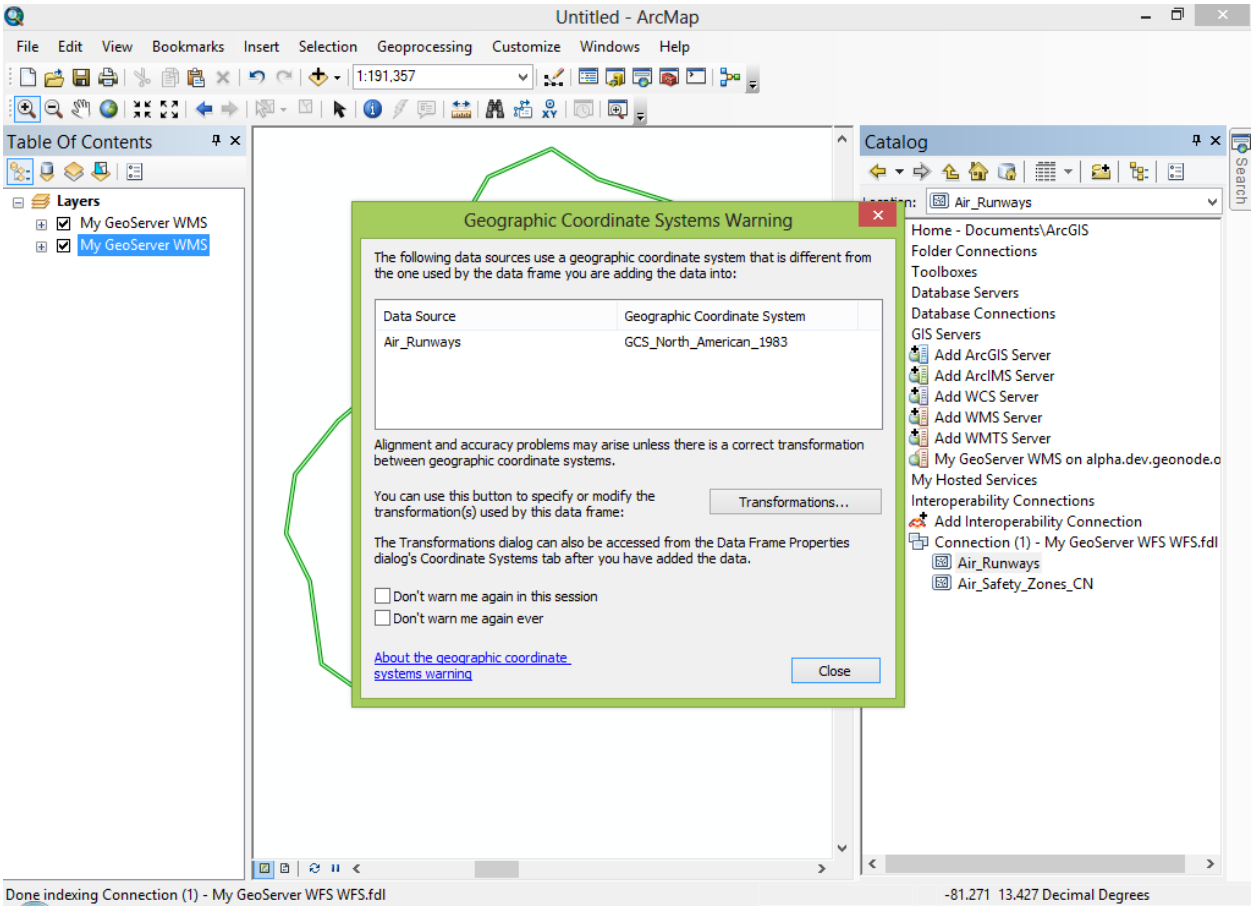

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
Catalog
Location: Home - Documents\ArcGIS
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Search
Interoperability Connection
Reader
Format: WFS (Web Feature Service)
Dataset: http://alpha.dev.geonode.org/geoserver/wfs
Parameters...
Coord. System: Read from source
Help
OK
Cancel
-81.271 13.427 Decimal Degrees


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
WFS (Web Feature Service) Parameters
WFS Connection
URL:
Prefer HTTP Post, if Available:
No
WFS Version:
1.1.0
Use HTTP Authentication
HTTP Username:
<Unused>
HTTP Password:
HTTP Authentication:
Use Proxy Server
Proxy Address:
Proxy Port:
Proxy Username:
Proxy Password:
Schema Attributes
Additional Attributes to Expose:
No items selected.
Constraints
Feature Types:
Max Features:
30000
WFS Output Format:
XML Filter Expression:
Application Schema
Help
Defaults
OK
Cancel


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
WFS (Web Feature Service) Parameters
WFS Connection
URL: pha.dev.geonode.org/geoserver/wfs
Prefer HTTP Post, if Available: No
WFS Version: 1.1.0
Use HTTP Authentication
HTTP Username: <Unused>
Select Feature Types
Air_Runways {Air_Runways}
Air_Runways0 {Air_Runways}
Air_Safety_Zones_CN {Air_Safety_Zon...
Ambulance_Exclusive_Operating_Area...
Ambulance_Staging_Locations {Amb...
Aquifers {Aquifers}
Bike_Plan_SD {Bike_Plan_SD}
Bnds_provincial_250k {Provincial Bou...
boxes_with_date {Boxes With Date}
boxes_with_date_iso_date {Boxes Wit...
boxes_with_dates_bc {Boxes With Dat...
boxes_with_end_date {Boxes With En...
Filter: Search...
Select all
OK
Cancel
Constraints
Feature Types:
Max Features: 30000
WFS Output Format:
XML Filter Expression:
Application Schema
Help
Defaults
OK
Cancel


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
Indexing Connection (1) - My GeoServe...
Indexing features in Connection (1) - My GeoServer WFS WFS.fdl
Features: 400
Cancel
Catalog
Location: Air_Runways
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My GeoServer WMS on alpha.dev.geonode.o
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Connection (1) - My GeoServer WFS WFS.fdl
Air_Runways
Air_Safety_Zones_CN
Search
Indexing Connection (1) - My GeoServer WFS WFS.fdl
-81.271 13.427 Decimal Degrees

Depending on the projection of your data, you may receive a warning about Alignment and Accuracy of data transformations. You can specify the transformation manually or simply hit close to ignore this dialog. If you dont want to be warned again, use the checkboxes in this dialog to hide these warnings temporarily or permanently.



Your WFS Layer will be added to your map and you can view it in the Map Panel. If you need to, use the "Zoom to Layer Extent" or other zoom tools to zoom to the bounds of your layer.

You can now use the identify tool to inspect a feature in your layer, or perform any other function that you can normally use to work with Vector Layers in ArcMap.

Since your layer was imported as actual vector features, you can use normal ArcMap styling tools to style the layer to match how you want it to be displayed.

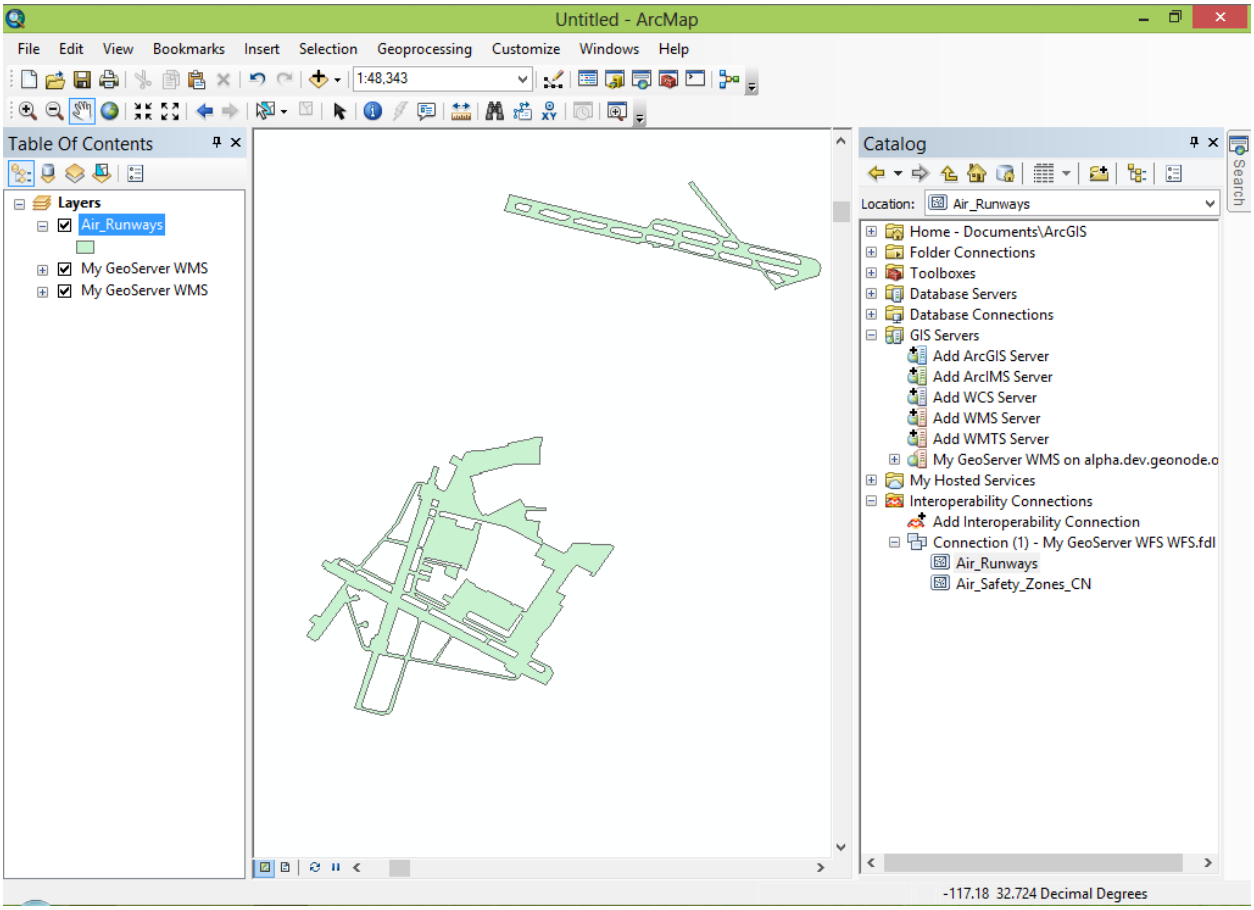
Now that you have added layers from your GeoNode as both WMS and WFS, you can explore the other options available to you with these layers within ArcMap.

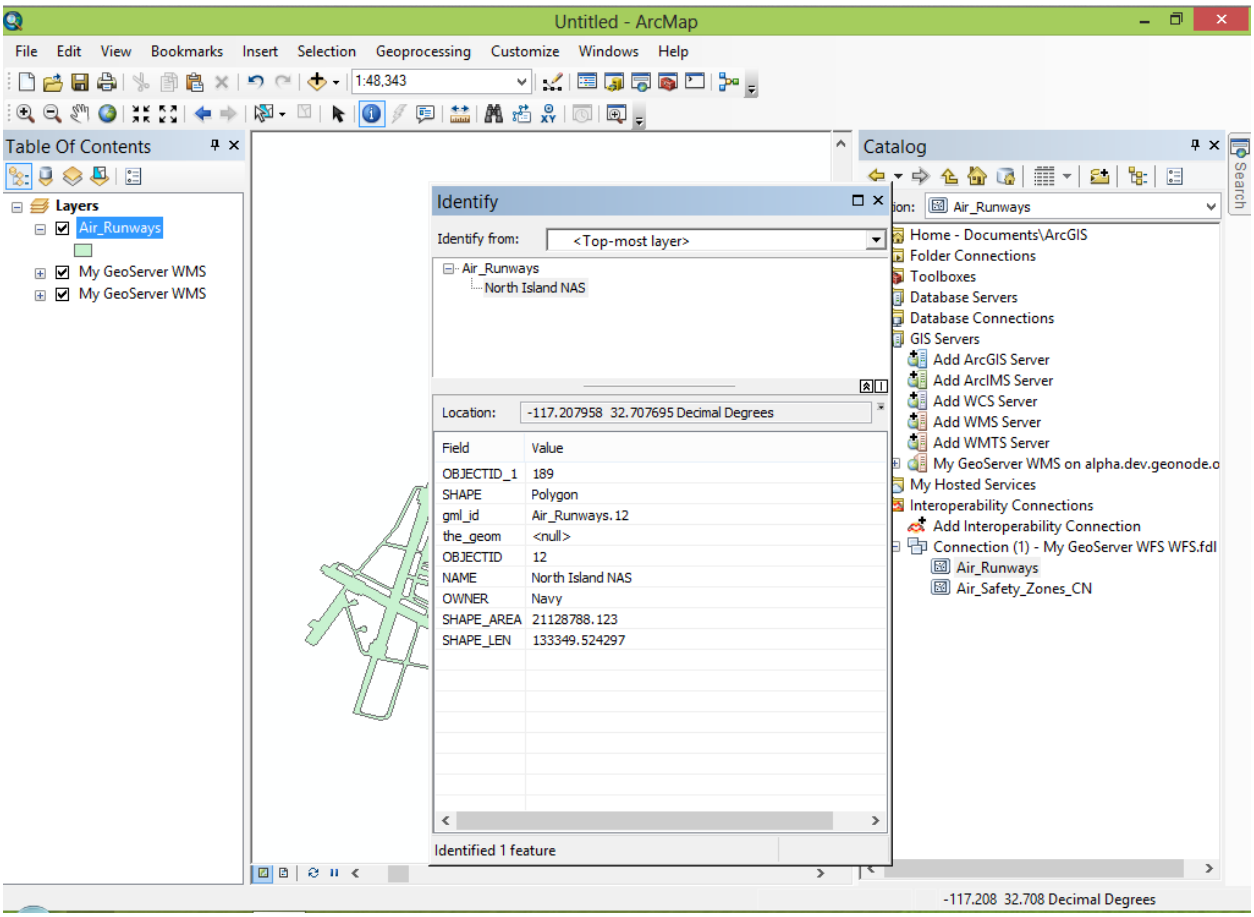
3.5.3 QGIS

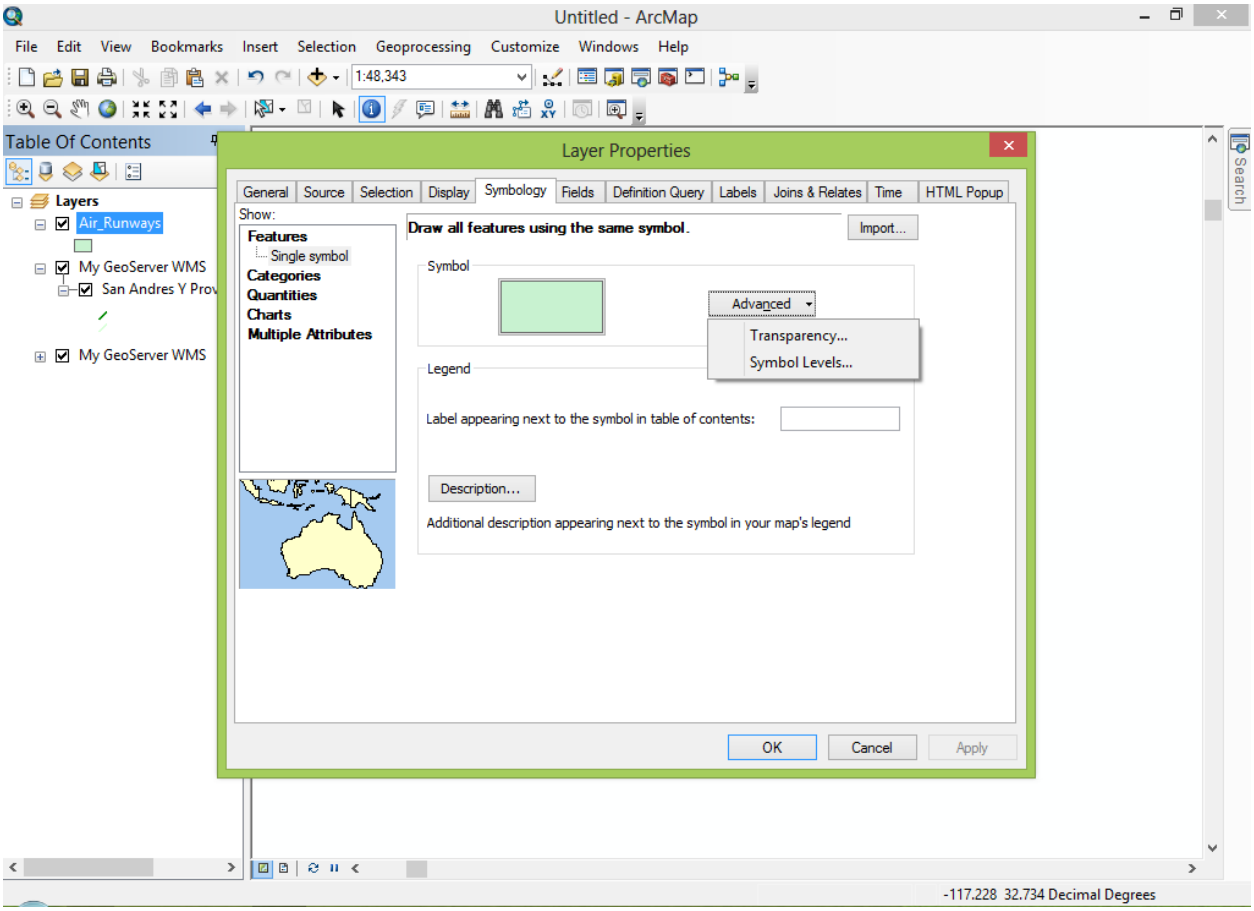
Quantum GIS or qGIS is an open source, cross platform desktop GIS app. It can also be used to add layers from your GeoNode instance as WMS or WFS. The process is very similar to how we add these same layers to ArcMap, and we will walk through the steps necessary in the following section.

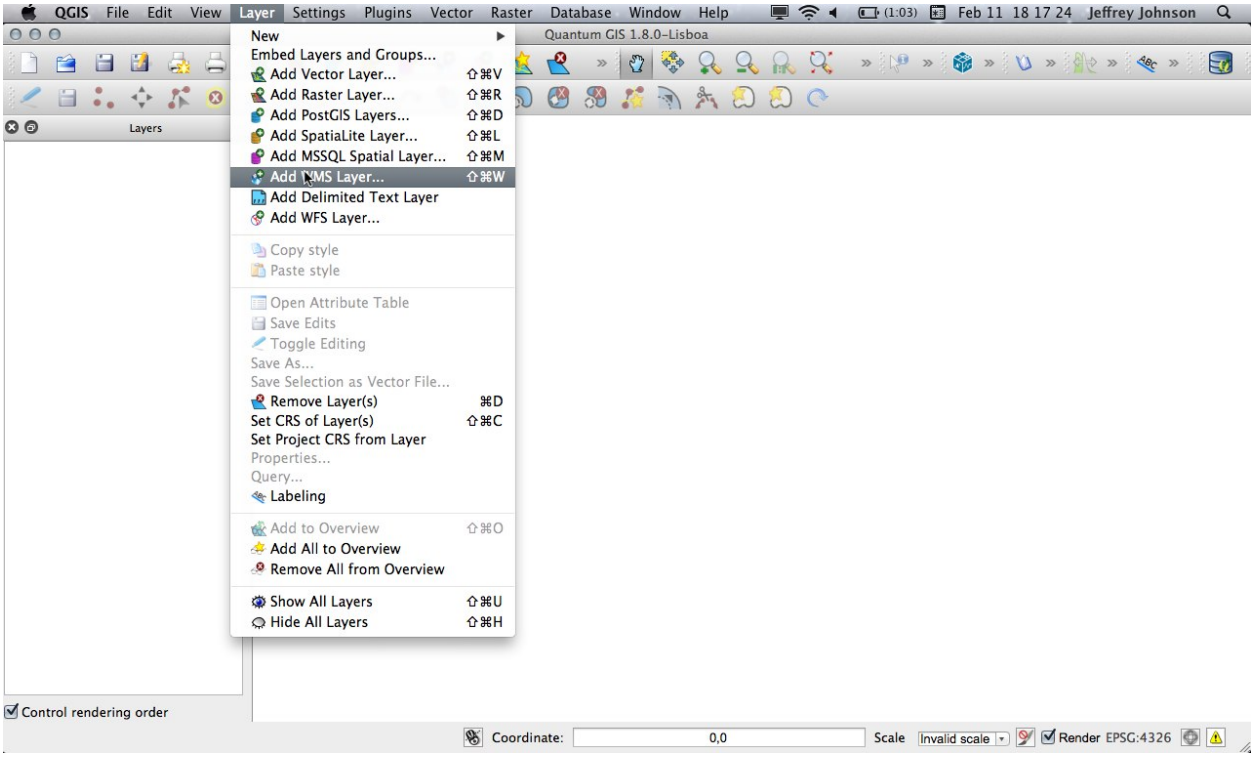
First, select "Add WMS Layer" from the Layer menu.

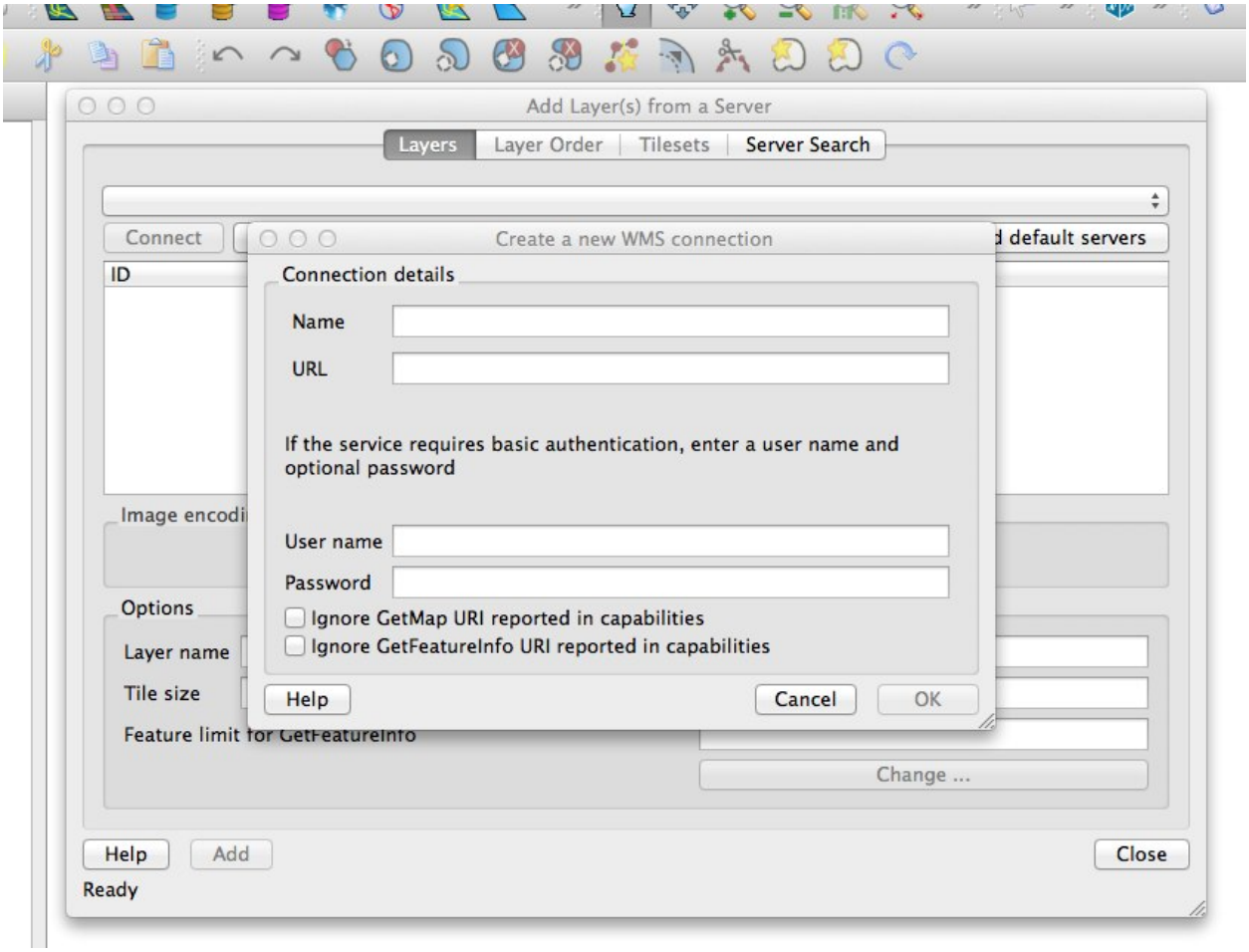
The Add WMS Layer Dialog will be displayed where you are able to specify the parameters to connect to your WMS server.


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:48,343
Table Of Contents
Layers
Air_Runways
My GeoServer WMS
My GeoServer WMS
Catalog
Location: Air_Runways
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My GeoServer WMS on alpha.dev.geonode.o
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Connection (1) - My GeoServer WFS WFS.fdl
Air_Runways
Air_Safety_Zones_CN
Search
-117.18 32.724 Decimal Degrees

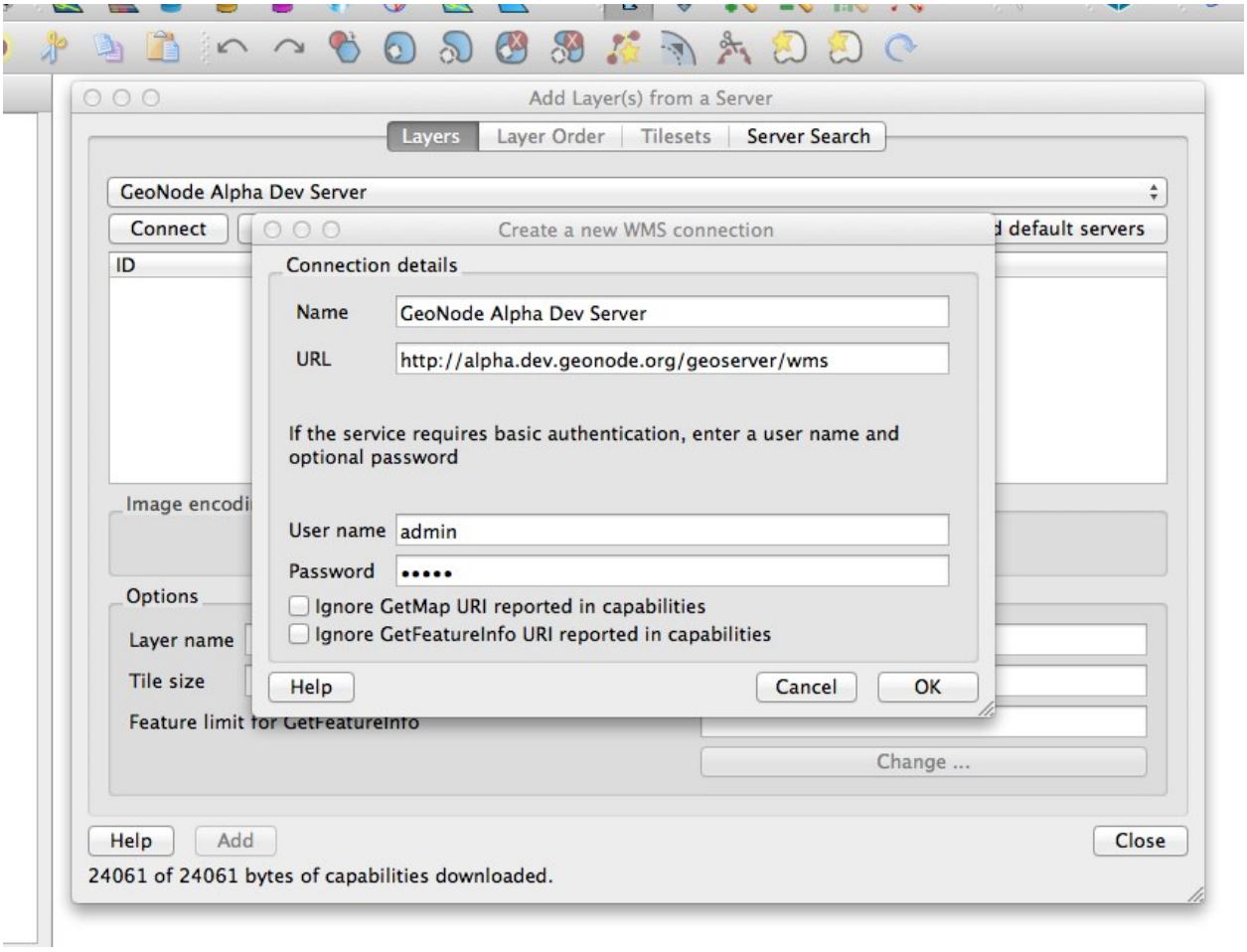

Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:48,343
Table Of Contents
Layers
Air_Runways
My GeoServer WMS
My GeoServer WMS
Identify
Identify from:
<Top-most layer>
Air_Runways
North Island NAS
Location:
-117.207958 32.707695 Decimal Degrees
Field
Value
OBJECTID_1
189
SHAPE
Polygon
gml_id
Air_Runways.12
the_geom
<null>
OBJECTID
12
NAME
North Island NAS
OWNER
Navy
SHAPE_AREA
21128788.123
SHAPE_LEN
133349.524297
Identified 1 feature
Catalog
Air_Runways
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My GeoServer WMS on alpha.dev.geonode.o
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Connection (1) - My GeoServer WFS WFS.fdl
Air_Runways
Air_Safety_Zones_CN
-117.208 32.708 Decimal Degrees


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:48,343
Table Of Contents
Layers
Air_Runways
My GeoServer WMS
San Andres Y Prov
My GeoServer WMS
Layer Properties
General Source Selection Display Symbology Fields Definition Query Labels Joins & Relates Time HTML Popup
Show:
Features
Single symbol
Categories
Quantities
Charts
Multiple Attributes
Draw all features using the same symbol.
Import...
Symbol
Advanced
Transparency...
Symbol Levels...
Legend
Label appearing next to the symbol in table of contents:
Description...
Additional description appearing next to the symbol in your map's legend
OK
Cancel
Apply
-117.228 32.734 Decimal Degrees


QGIS File Edit View Layer Settings Plugins Vector Raster Database Window Help
Feb 11 18 17 24 Jeffrey Johnson
Quantum GIS 1.8.0-Lisboa
Layers
New
Embed Layers and Groups...
Add Vector Layer... ⇧⌘V
Add Raster Layer... ⇧⌘R
Add PostGIS Layers... ⇧⌘D
Add SpatiaLite Layer... ⇧⌘L
Add MSSQL Spatial Layer... ⇧⌘M
Add WMS Layer... ⇧⌘W
Add Delimited Text Layer
Add WFS Layer...
Copy style
Paste style
Open Attribute Table
Save Edits
Toggle Editing
Save As...
Save Selection as Vector File...
Remove Layer(s) ⌘D
Set CRS of Layer(s) ⇧⌘C
Set Project CRS from Layer
Properties...
Query...
Labeling
Add to Overview ⇧⌘O
Add All to Overview
Remove All from Overview
Show All Layers ⇧⌘U
Hide All Layers ⇧⌘H
Control rendering order
Coordinate: 0,0
Scale Invalid scale
Render EPSG:4326


Add Layer(s) from a Server
Layers
Layer Order
Tilesets
Server Search
Connect
default servers
ID
Create a new WMS connection
Connection details
Name
URL
If the service requires basic authentication, enter a user name and optional password
User name
Password
Ignore GetMap URI reported in capabilities
Ignore GetFeatureInfo URI reported in capabilities
Help
Cancel
OK
Image encodi
Options
Layer name
Tile size
Feature limit for GetFeatureInfo
Change ...
Help
Add
Close
Ready

Next, you need to fill in the parameters to connect to your GeoNode instance. The URL for your GeoNode's WMS is the base URL + /geoserver/wms



After clicking the OK button, your server will show up in the list of servers. Make sure its selected, then, click the connect button to have QGIS retrieve the list of layers from your GeoNode.

Select the layers you want to add to your QGIS project and click "Add".

Your layer will be displayed in the map panel.

You can then zoom into your features in the Map.

From there, you can use the identify tool to inspect the attributes of one of the features on the map.

Or, you can look at the layer metadata by right clicking on the layer and selecting Layer Properties and selecting the metadata tab.

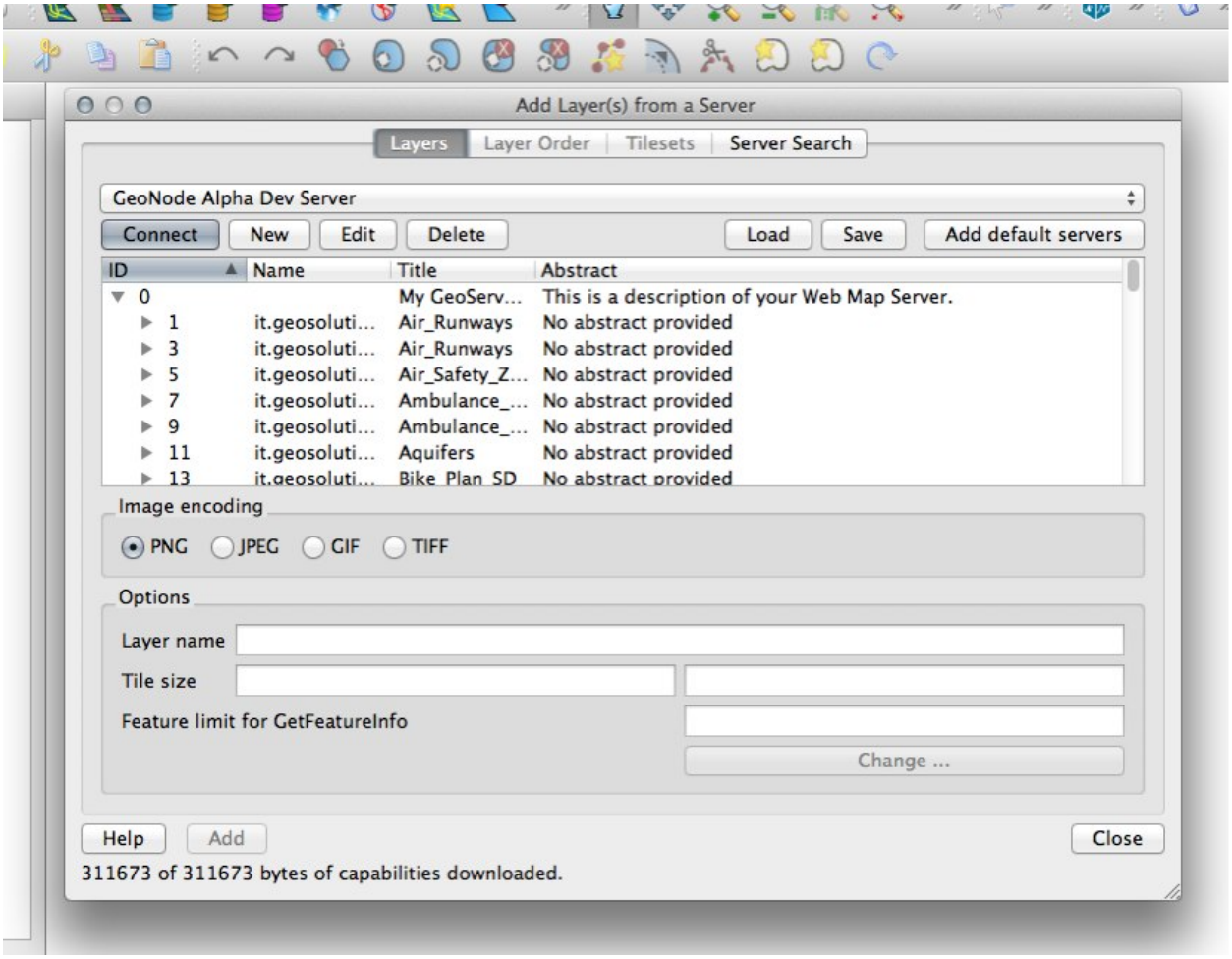
Adding WFS servers and layers to your QGIS project is very similar to adding WMS. Depending on your version of QGIS, you may need to add the WFS plugin. You can use the Plugin manager to add it.

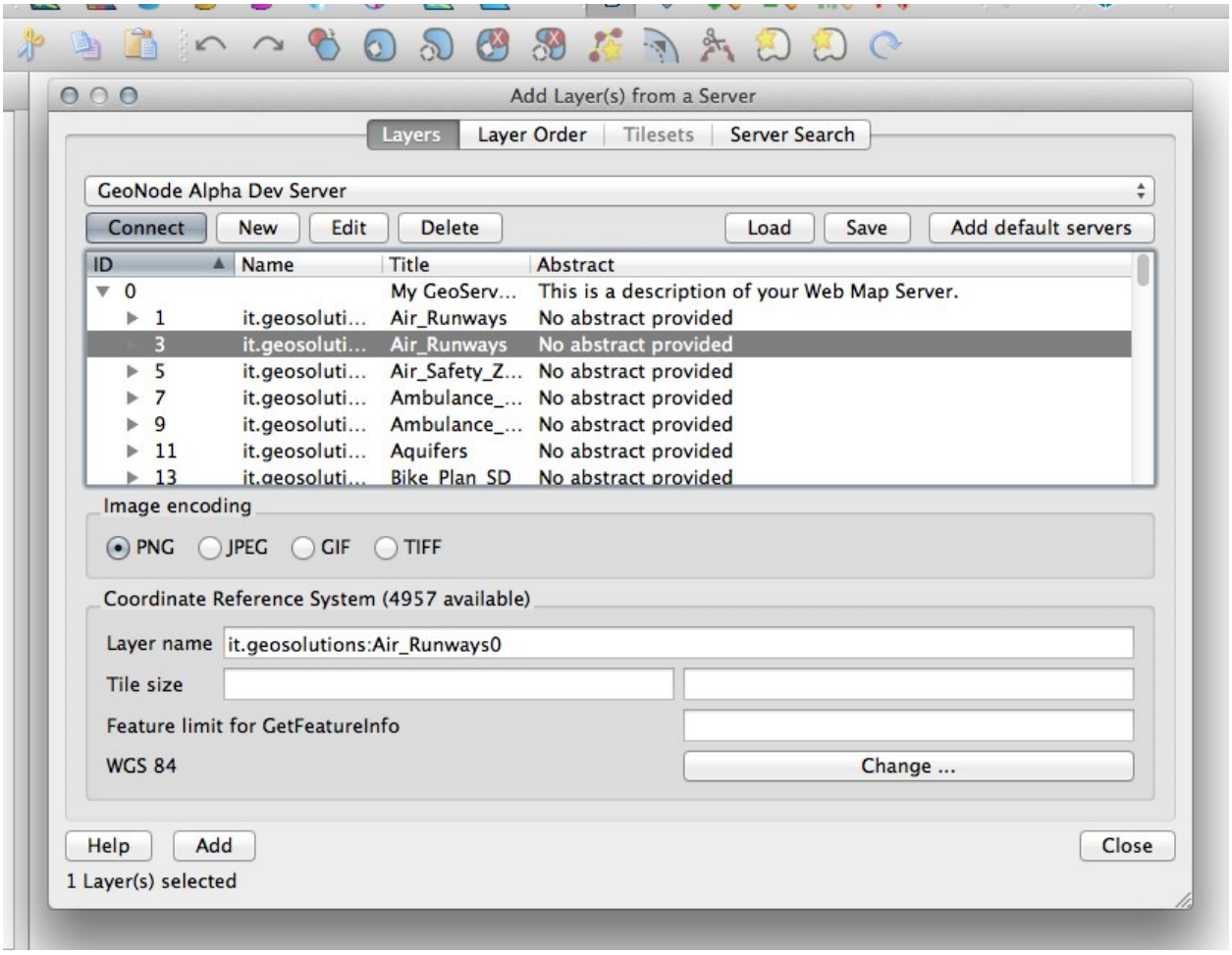
Once the plugin is installed, you can select the "Add WFS Layer" option from the Layer menu.

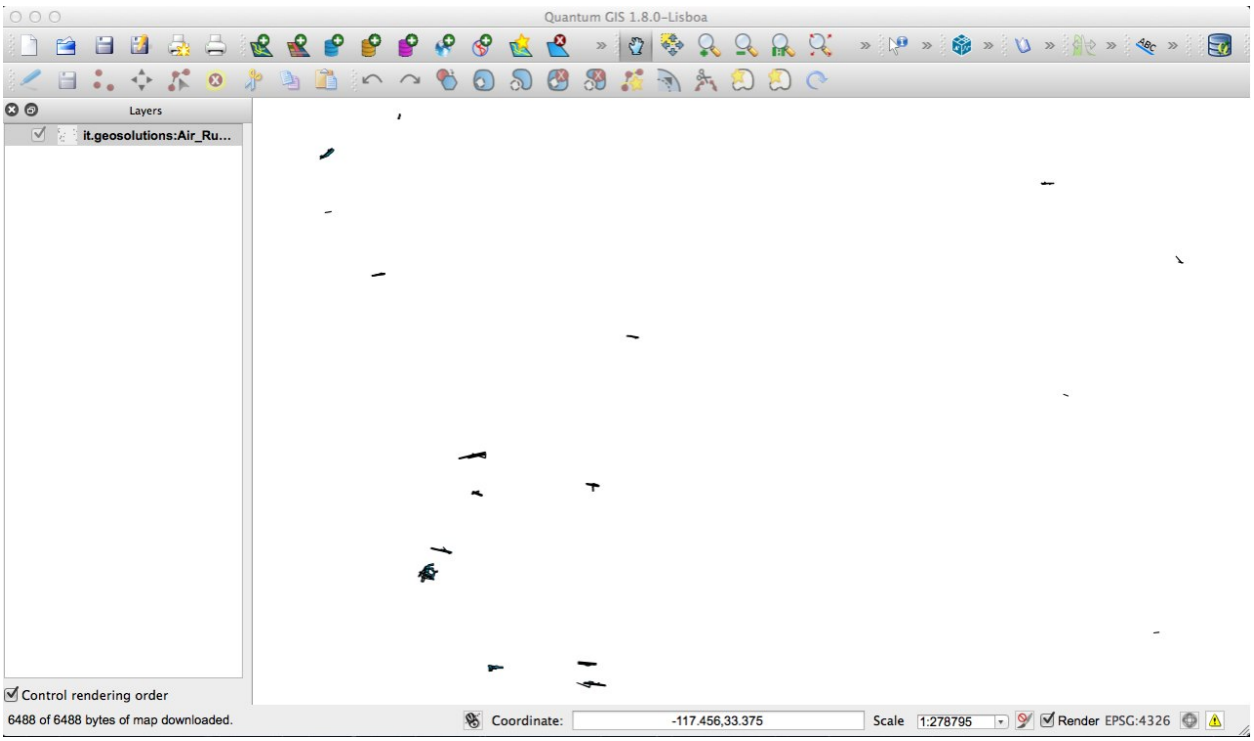
Step through the same process you did for WMS to create a new WFS connection. First specify server parameters and click OK.

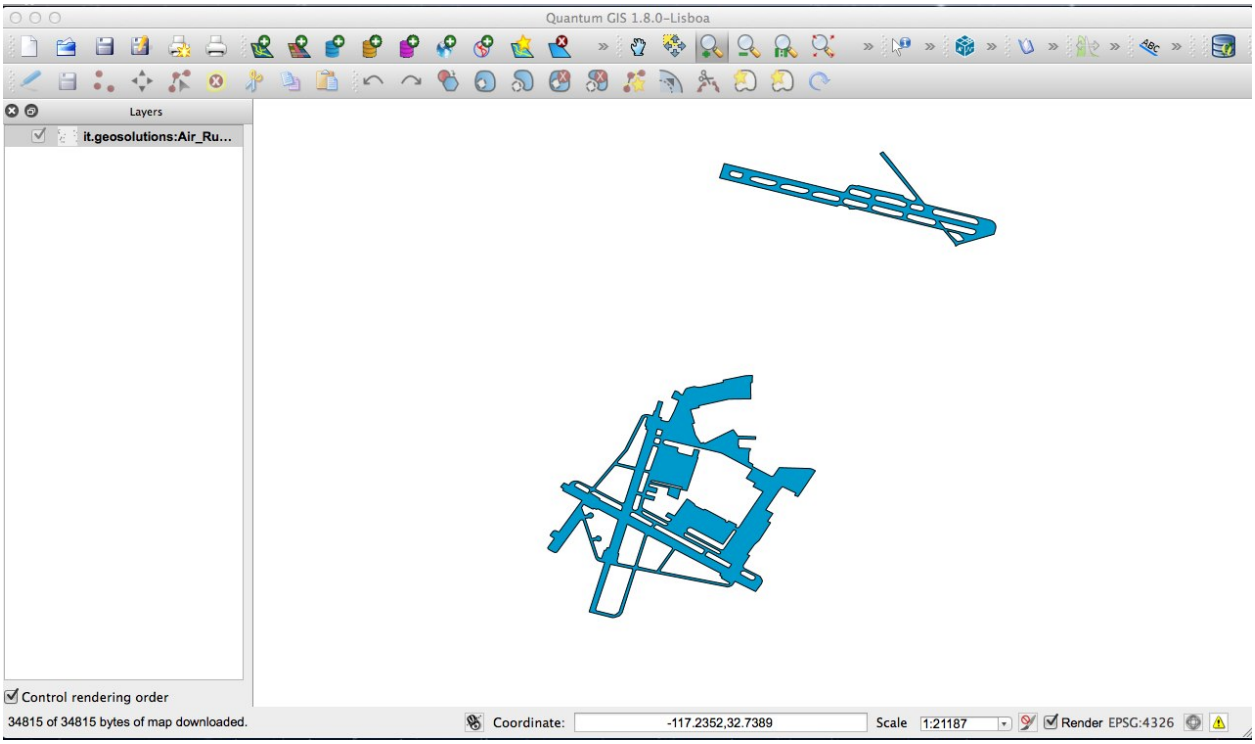
Then click Connect to retrieve the list of layers on the server and select the layers you want to add and click Apply.

The layer(s) you selected will be displayed in the map panel.


Add Layer(s) from a Server
Layers
Layer Order
Tilesets
Server Search
GeoNode Alpha Dev Server
Connect
New
Edit
Delete
Load
Save
Add default servers
ID
Name
Title
Abstract
0
My GeoServ...
This is a description of your Web Map Server.
1 it.geosoluti... Air_Runways No abstract provided
3 it.geosoluti... Air_Runways No abstract provided
5 it.geosoluti... Air_Safety_Z... No abstract provided
7 it.geosoluti... Ambulance_... No abstract provided
9 it.geosoluti... Ambulance_... No abstract provided
11 it.geosoluti... Aquifers No abstract provided
13 it.geosoluti... Bike_Plan_SD No abstract provided
Image encoding
PNG
JPEG
GIF
TIFF
Options
Layer name
Tile size
Feature limit for GetFeatureInfo
Change ...
Help
Add
Close
311673 of 311673 bytes of capabilities downloaded.


Add Layer(s) from a Server
Layers
Layer Order
Tilesets
Server Search
GeoNode Alpha Dev Server
Connect
New
Edit
Delete
Load
Save
Add default servers
ID
Name
Title
Abstract
0
My GeoServ...
This is a description of your Web Map Server.
1
it.geosoluti...
Air_Runways
No abstract provided
3
it.geosoluti...
Air_Runways
No abstract provided
5
it.geosoluti...
Air_Safety_Z...
No abstract provided
7
it.geosoluti...
Ambulance_...
No abstract provided
9
it.geosoluti...
Ambulance_...
No abstract provided
11
it.geosoluti...
Aquifers
No abstract provided
13
it.geosoluti...
Bike_Plan_SD
No abstract provided
Image encoding
PNG
JPEG
GIF
TIFF
Coordinate Reference System (4957 available)
Layer name
it.geosolutions:Air_Runways0
Tile size
Feature limit for GetFeatureInfo
WGS 84
Change ...
Help
Add
Close
1 Layer(s) selected


Quantum GIS 1.8.0-Lisboa
Layers
it.geosolutions:Air_Ru...
Control rendering order
6488 of 6488 bytes of map downloaded.
Coordinate:
-117.456,33.375
Scale
1:278795
Render
EPSG:4326



You can use the same identify tool to inspect features in the map panel.

To look at more information about your layer, right click the layer in the Table of Contents and select Layer Properties. You can look at the list of fields.

... or set a style to match how you want your data to be displayed.

You now know how to add layers from your GeoNode instance to a QGIS project. You can explore all of the other options available to you in QGIS by consulting its documentation.

3.5.4 Google Earth

GeoNode's built in map interface lets you look at your layers and maps in the Google Earth plugin directly in your browser. You can switch to this 3D viewer directly in GeoNode by clicking the google earth icon in the map panel.

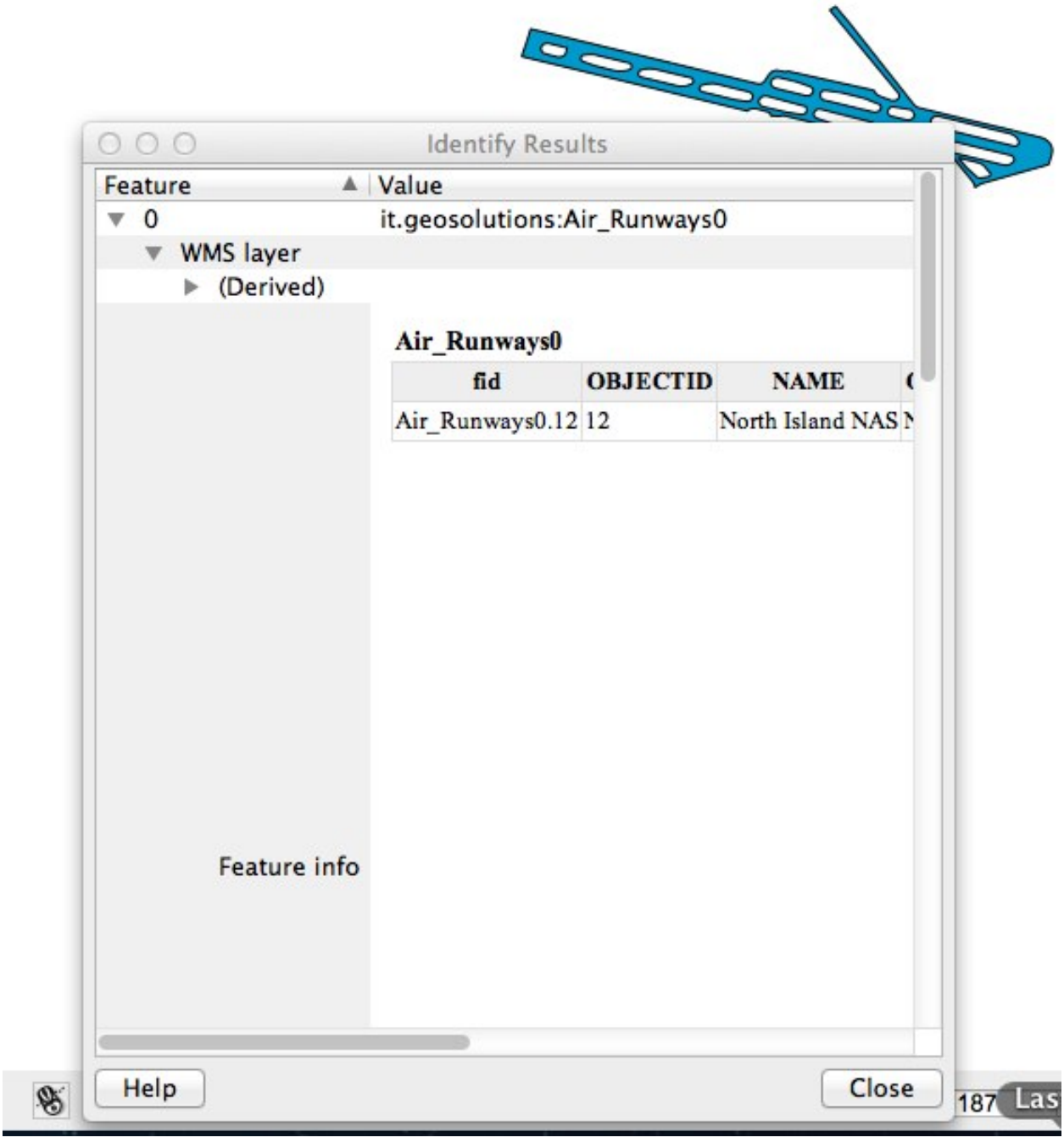
GeoServer will render your layer as an image until you are zoomed in sufficiently, and then it will switch to rendering it as a vector overlay that you can click on to view the attributes for the feature you clicked on.

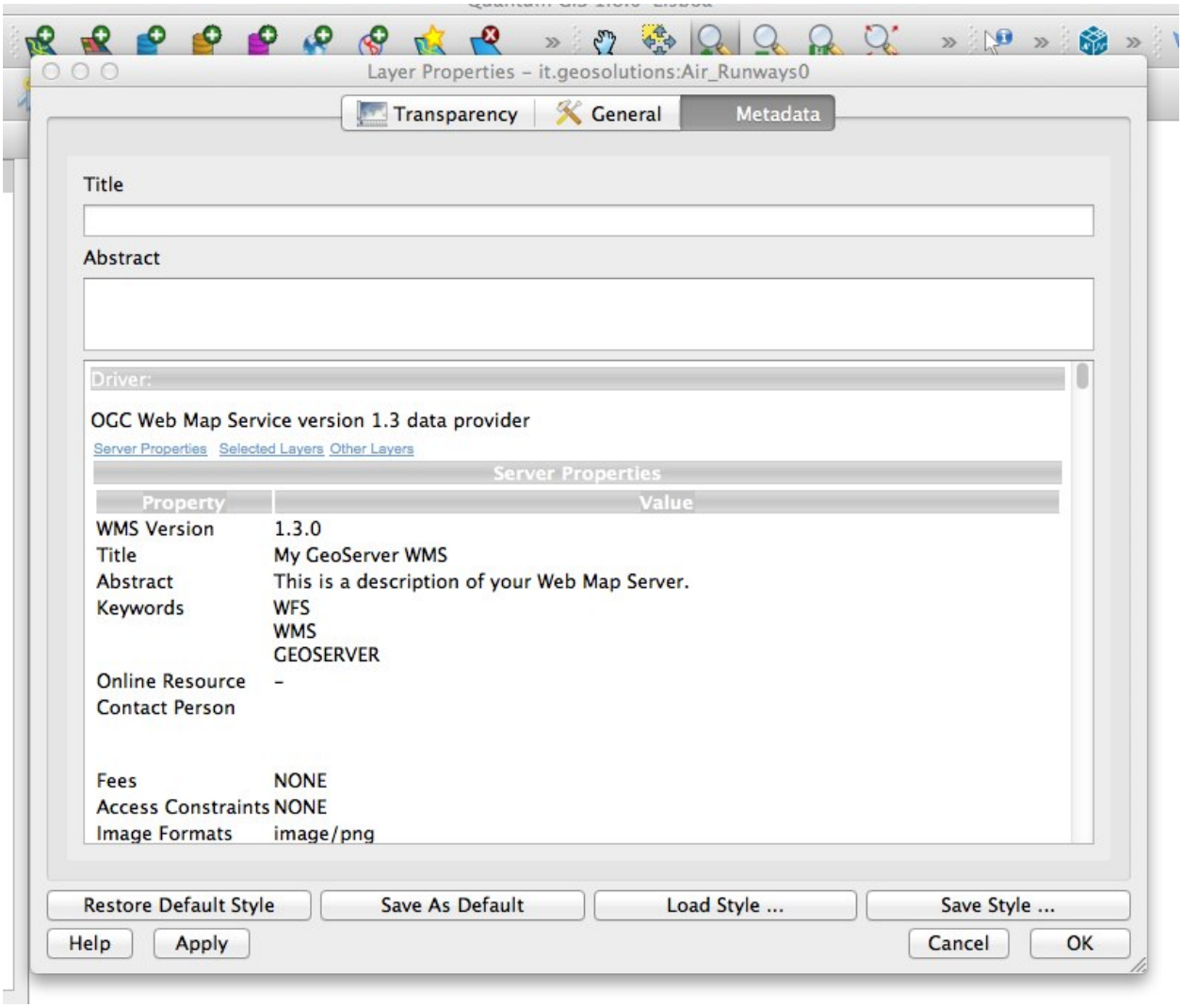
You can also use this option in the GeoExplorer client by clicking the same button.

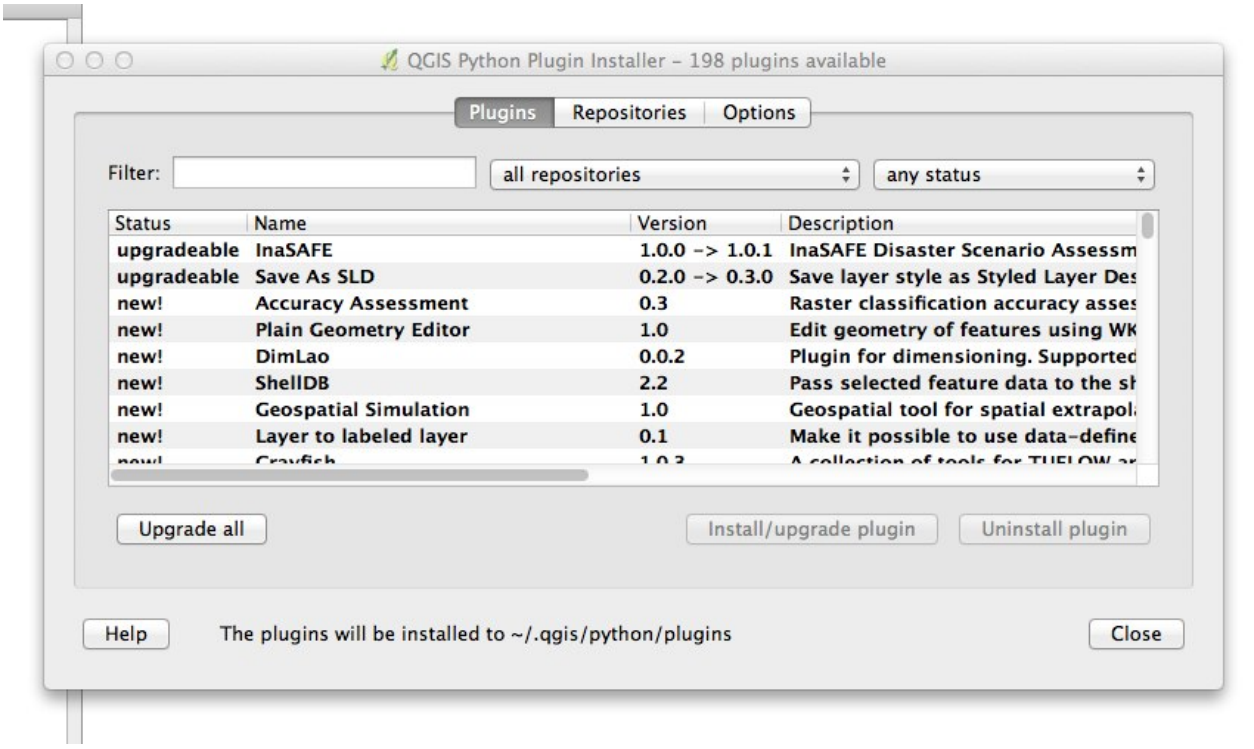
Note: Some of the GeoExplorer options will not be available to you when you are in this mode, they will be grayed out an inaccessible.

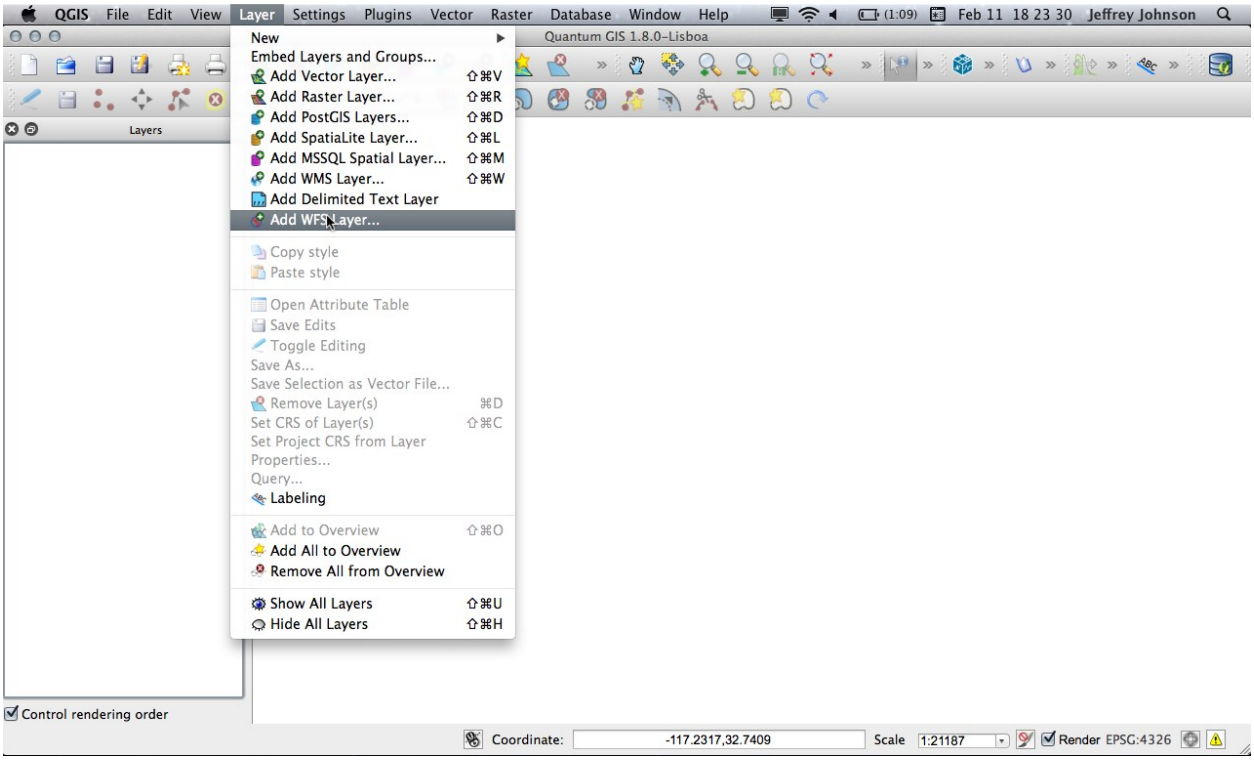
If instead you want to use layers from your GeoNode in the Google Earth client itself, you have a few options available to you.

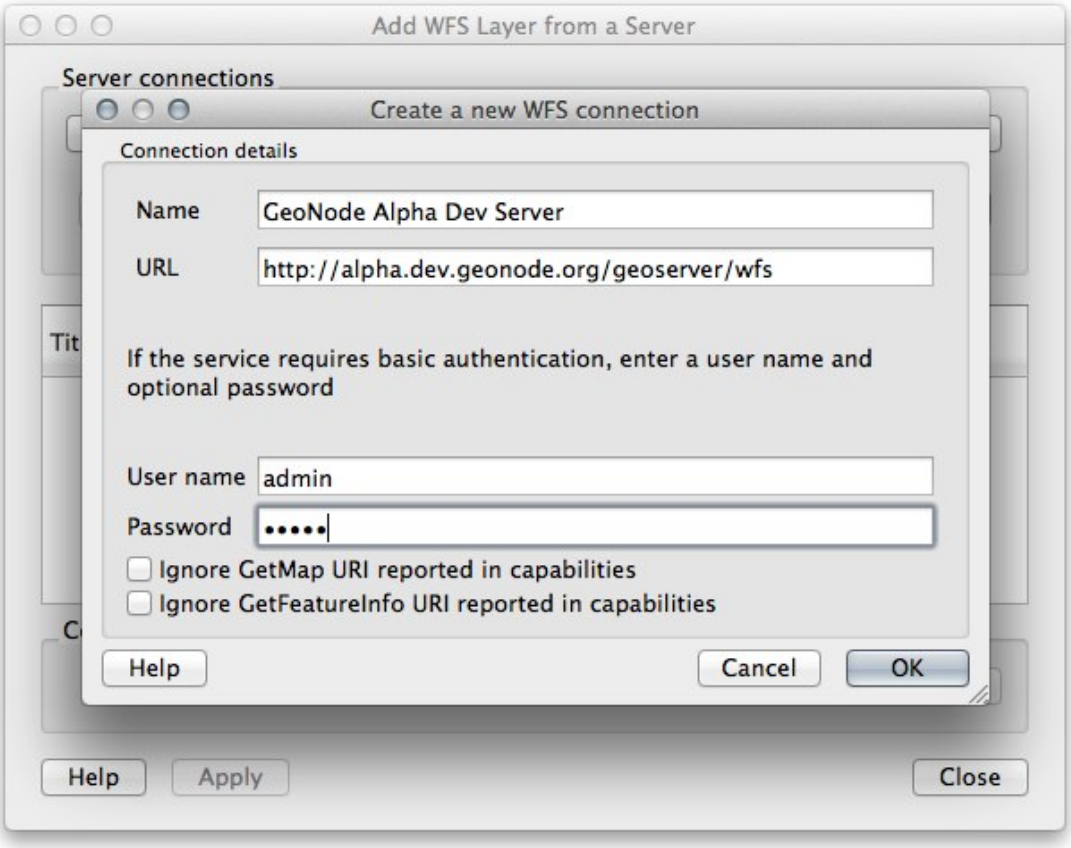
First, you can select the KML option from the Download Layer menu to download the entire layer in a single KML file. Depending on the size of the layer, your GeoNode could take several seconds or longer to generate this KML and return it to you.

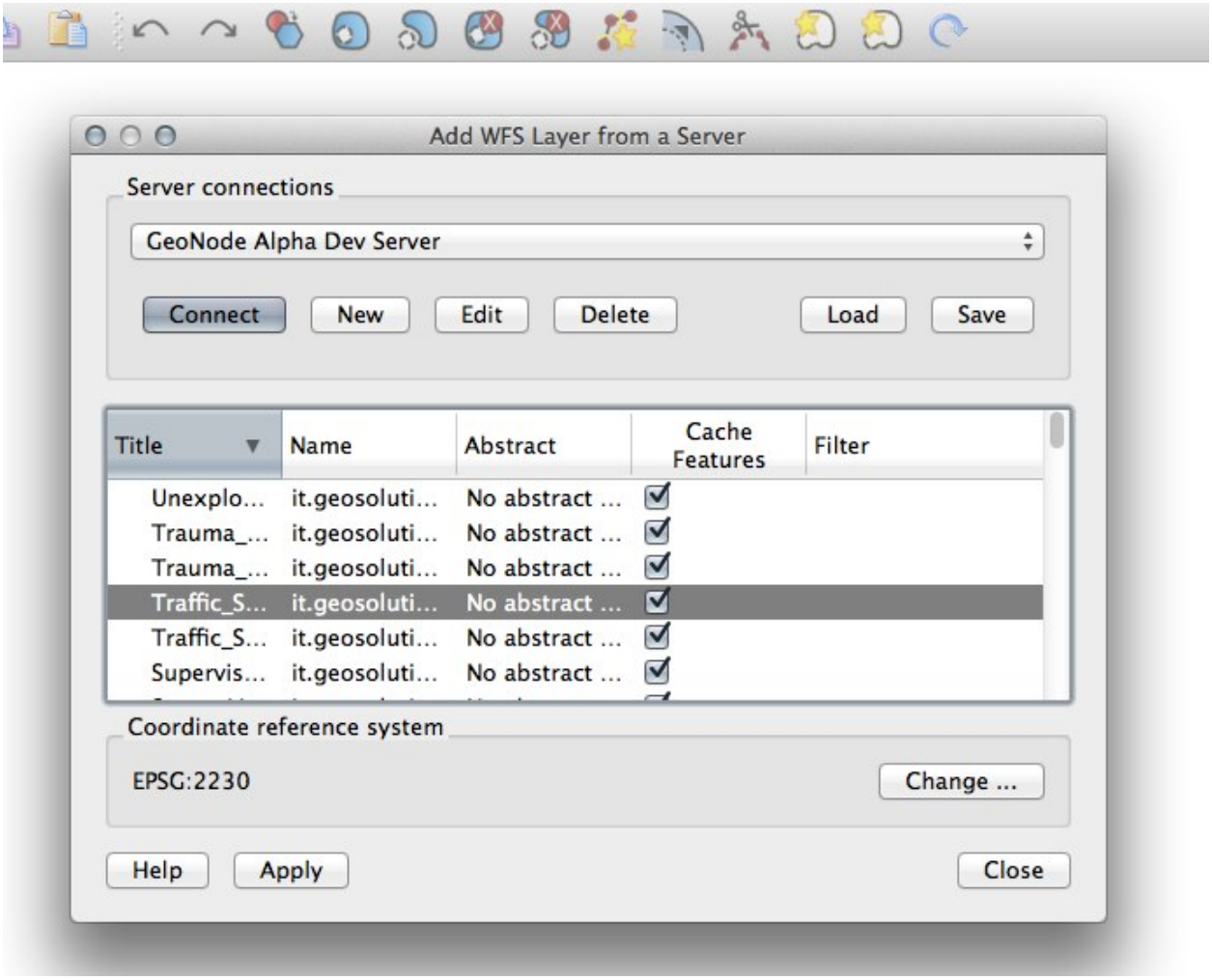

Identify Results
Feature
Value
0
it.geosolutions:Air_Runways0
WMS layer
(Derived)
Air_Runways0
fid
OBJECTID
NAME
Air_Runways0.12
12
North Island NAS
Feature info
Help
Close

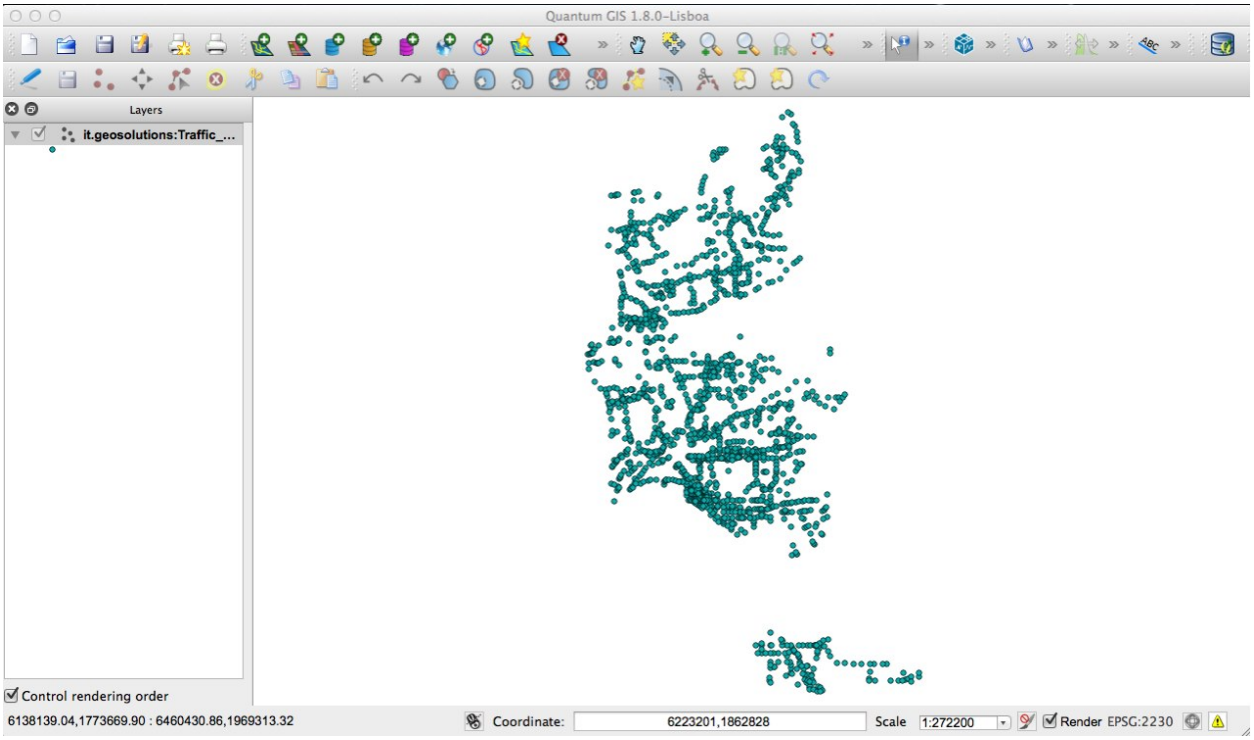

Layer Properties – it.geosolutions:Air_Runways0
Transparency
General
Metadata
Title
Abstract
Driver:
OGC Web Map Service version 1.3 data provider
Server Properties
Selected Layers
Other Layers
Server Properties
Property
Value
WMS Version
1.3.0
Title
My GeoServer WMS
Abstract
This is a description of your Web Map Server.
Keywords
WFS
WMS
GEOSERVER
Online Resource
-
Contact Person
Fees
NONE
Access Constraints
NONE
Image Formats
image/png
Restore Default Style
Save As Default
Load Style ...
Save Style ...
Help
Apply
Cancel
OK


QGIS Python Plugin Installer - 198 plugins available
Plugins
Repositories
Options
Filter:
all repositories
any status
Status
Name
Version
Description
upgradeable
InaSAFE
1.0.0 -> 1.0.1
InaSAFE Disaster Scenario Assessm
upgradeable
Save As SLD
0.2.0 -> 0.3.0
Save layer style as Styled Layer Des
new!
Accuracy Assessment
0.3
Raster classification accuracy asses
new!
Plain Geometry Editor
1.0
Edit geometry of features using WK
new!
DimLao
0.0.2
Plugin for dimensioning. Supported
new!
ShellDB
2.2
Pass selected feature data to the sh
new!
Geospatial Simulation
1.0
Geospatial tool for spatial extrapol
new!
Layer to labeled layer
0.1
Make it possible to use data-define
Upgrade all
Install/upgrade plugin
Uninstall plugin
Help
The plugins will be installed to ~/.qgis/python/plugins
Close


QGIS
File
Edit
View
Layer
Settings
Plugins
Vector
Raster
Database
Window
Help
Feb 11 18 23 30
Jeffrey Johnson
Quantum GIS 1.8.0-Lisboa
Layers
New
Embed Layers and Groups...
Add Vector Layer...
Add Raster Layer...
Add PostGIS Layers...
Add SpatiaLite Layer...
Add MSSQL Spatial Layer...
Add WMS Layer...
Add Delimited Text Layer
Add WFS Layer...
Copy style
Paste style
Open Attribute Table
Save Edits
Toggle Editing
Save As...
Save Selection as Vector File...
Remove Layer(s)
Set CRS of Layer(s)
Set Project CRS from Layer
Properties...
Query...
Labeling
Add to Overview
Add All to Overview
Remove All from Overview
Show All Layers
Hide All Layers
Control rendering order
Coordinate:
-117.2317,32.7409
Scale
1:21187
Render
EPSG:4326


Add WFS Layer from a Server
Server connections
Create a new WFS connection
Connection details
Name
GeoNode Alpha Dev Server
URL
http://alpha.dev.geonode.org/geoserver/wfs
If the service requires basic authentication, enter a user name and optional password
User name
admin
Password
Ignore GetMap URI reported in capabilities
Ignore GetFeatureInfo URI reported in capabilities
Help
Cancel
OK
Help
Apply
Close


Add WFS Layer from a Server
Server connections
GeoNode Alpha Dev Server
Connect
New
Edit
Delete
Load
Save
Title
Name
Abstract
Cache Features
Filter
Unexplo...
it.geosoluti...
No abstract ...
Trauma_...
it.geosoluti...
No abstract ...
Trauma_...
it.geosoluti...
No abstract ...
Traffic_S...
it.geosoluti...
No abstract ...
Traffic_S...
it.geosoluti...
No abstract ...
Supervis...
it.geosoluti...
No abstract ...
Coordinate reference system
EPSG:2230
Change ...
Help
Apply
Close



When the layer is generated, it will be downloaded to your desktop machine and you can simply double click it to open it in Google Earth.

Alternatively, you can use the "View in Google Earth" option in the Layer Download menu to view the layer in Google Earth using the same methodology described above depending on the zoom level.

This will download a small KMZ to your desktop that contains a reference to the layers on the server and you can double click it to open it in Google Earth.

Note: The basic difference between these two options is that the first downloads *all* of the data to your desktop at once and as such, the downloaded file can be used offline while the second is simply a Network Link to the layer on the server. Choose whichever method is best for your own needs and purposes.

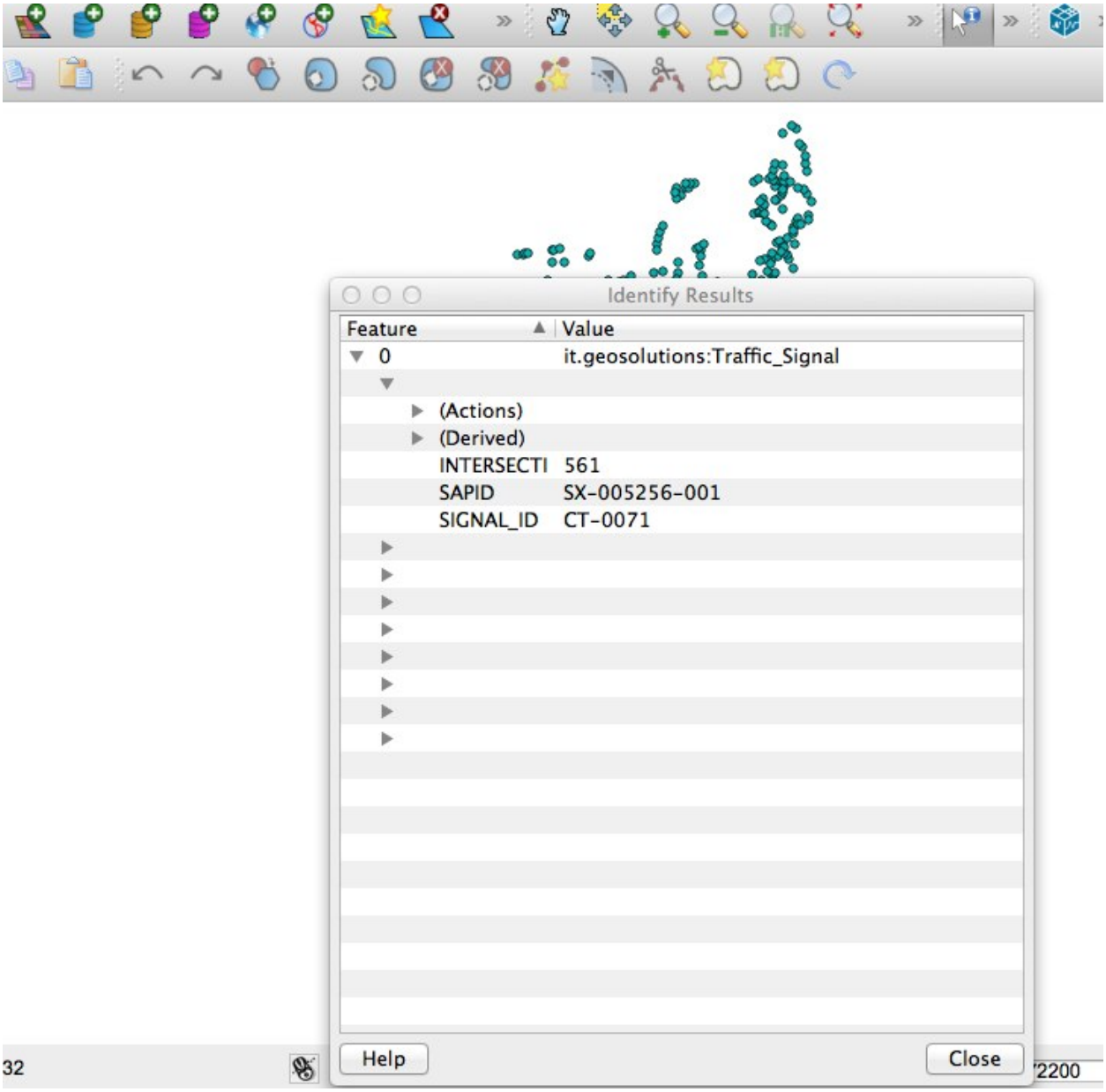
Once you have added your layers to the Places panel in Google Earth, you can move them from the Temporary Places section into My Places if you wish to use them after your current Google Earth session is complete. You can arrange them in folders and use Google Earth functionality to save your project to disk. Consult Google Earths documentation for more information about how to do this.

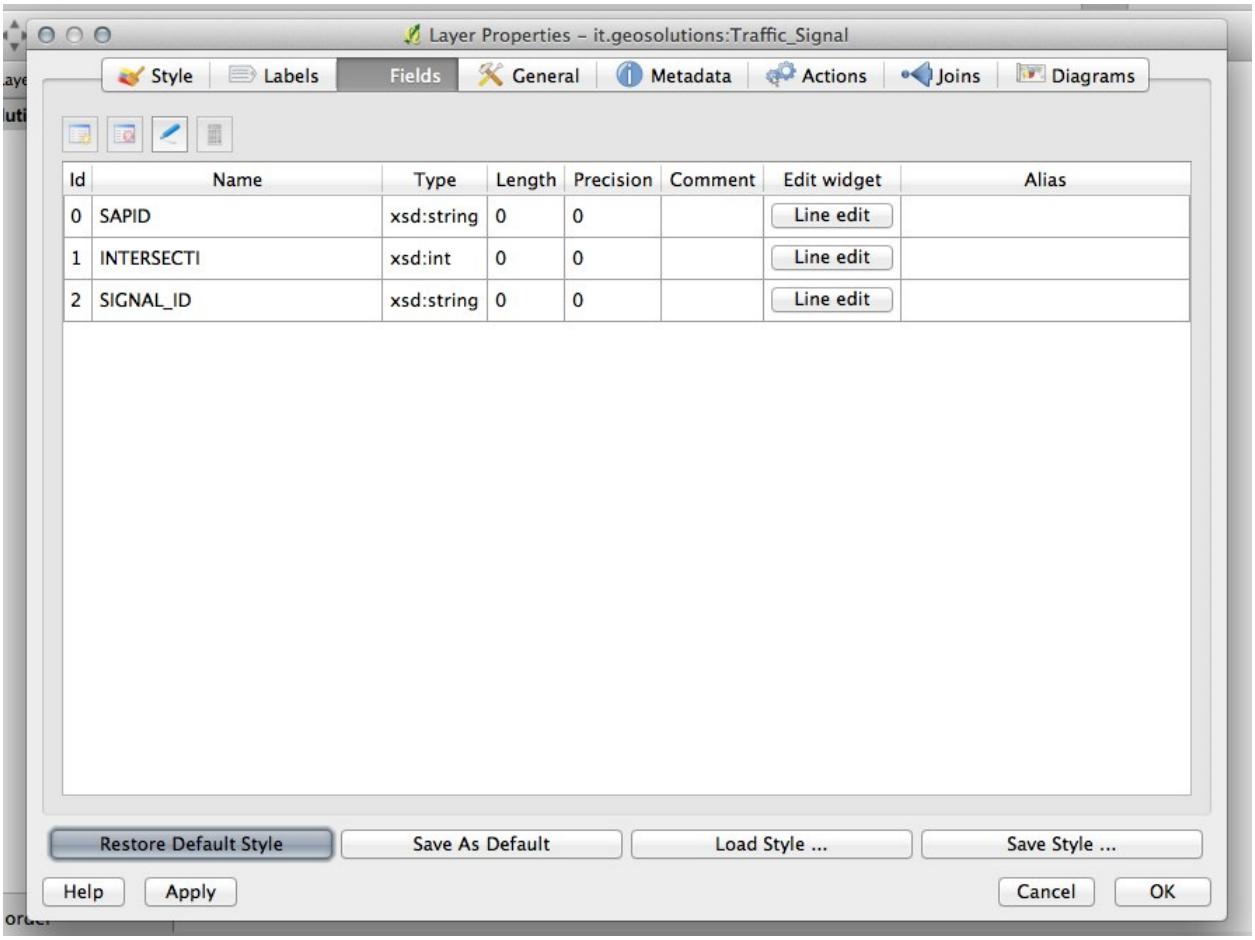
3.5.5 OpenStreetMap

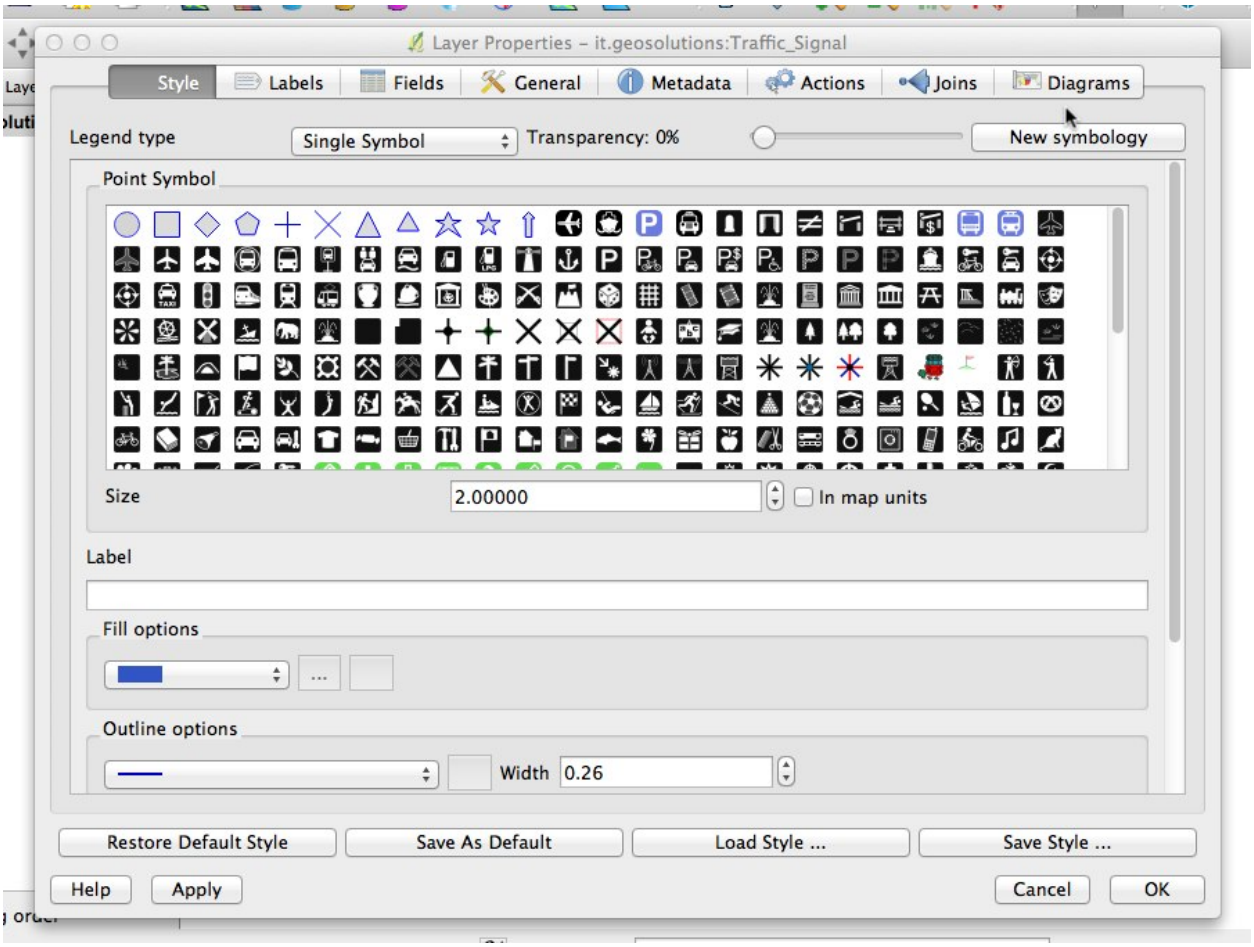
Data from OpenStreetMap is used to generate the map tiles which are the default base layer in your GeoNode. These are provided by MapQuest and are reasonably current with recent edits to the OpenStreetMap database. You can switch to an alternate set of map tiles by changing your base layer.

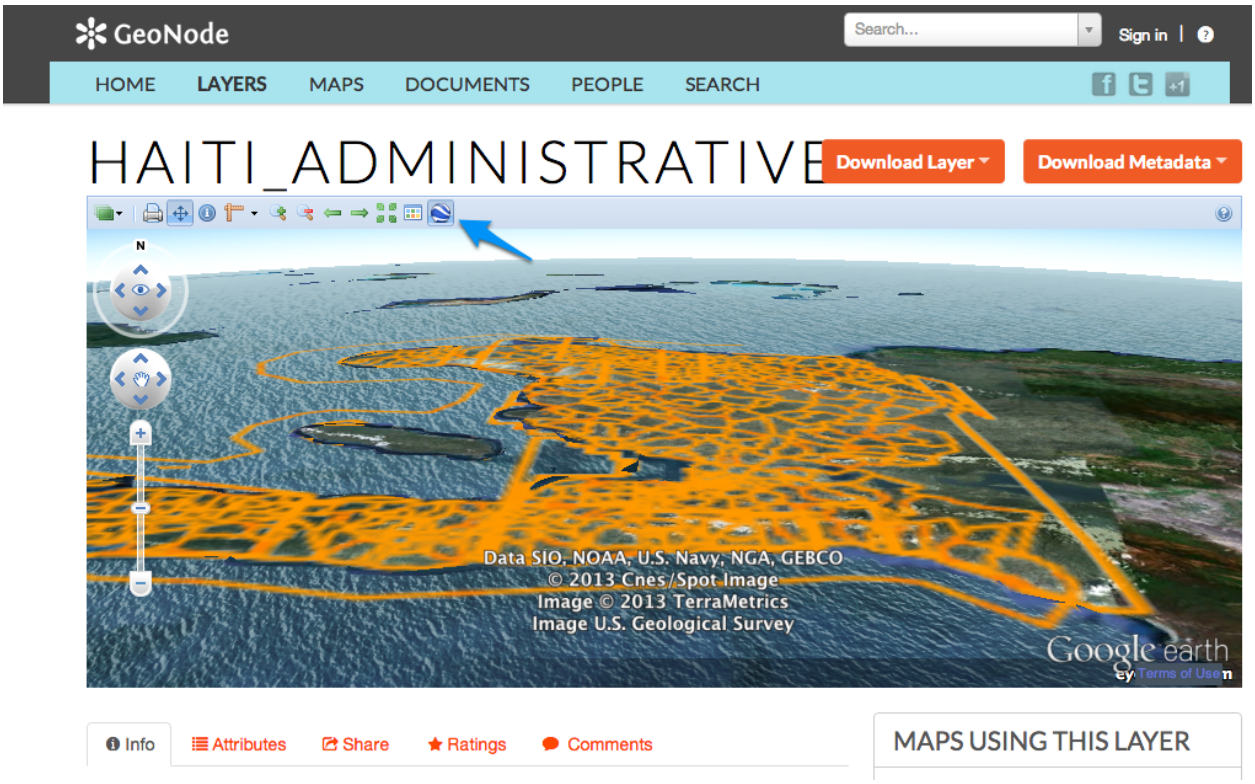
If you would like to change the default base layer to use the alternate OSM tile set, you can change this in the settings.py file for your project. In the MAP_BASELAYERS dictionary, set the visibility setting to True for the gxp_olsource and False for the gxp_mapquestsource.

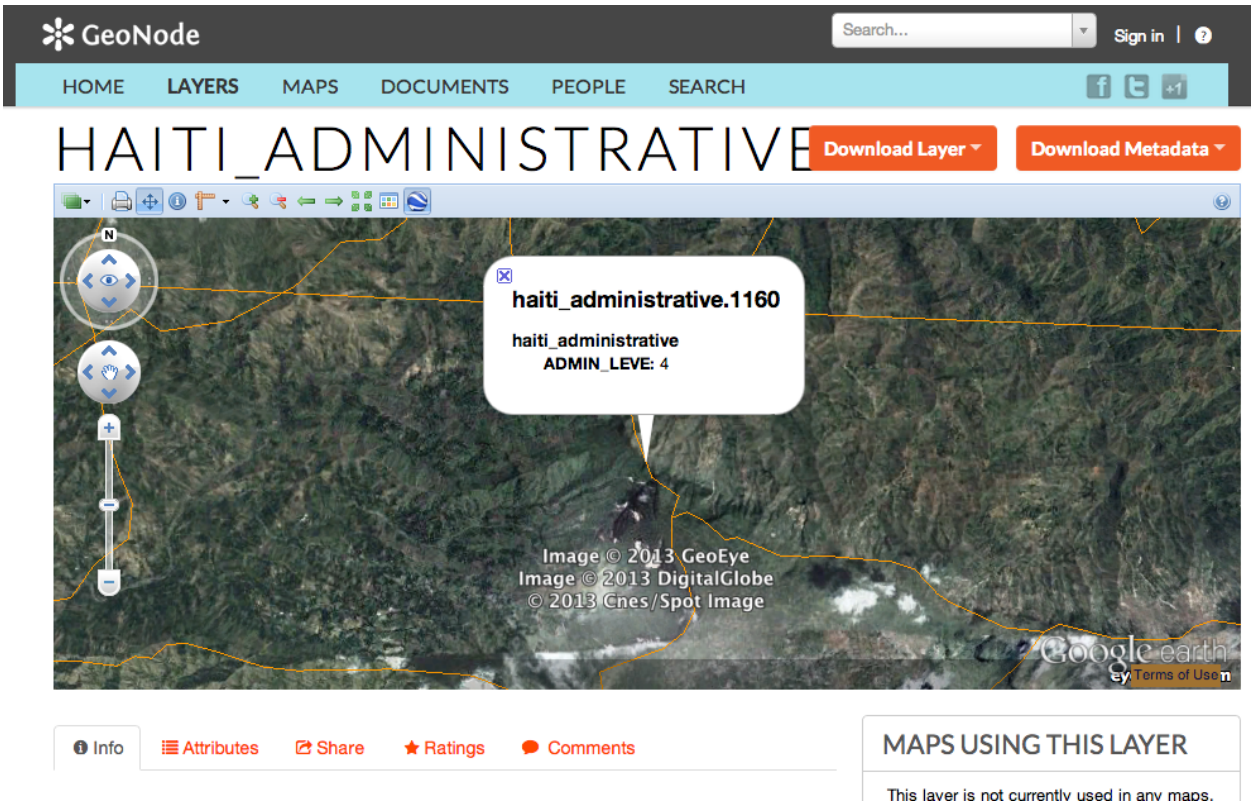
If instead of map tiles, you would like to work with OSM database itself, you should consult the section titled "Loading OSM Data into GeoNode" in this workshop.

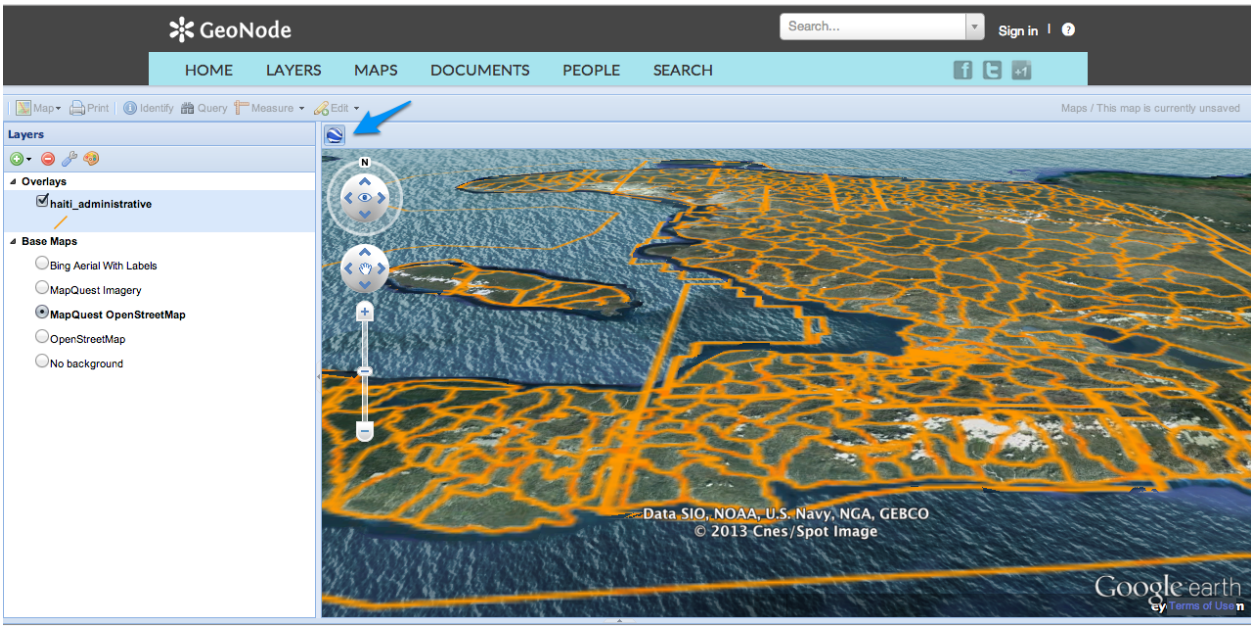

Identify Results
Feature
Value
0
it.geosolutions:Traffic_Signal
(Actions)
(Derived)
INTERSECTI
561
SAPID
SX-005256-001
SIGNAL_ID
CT-0071
Help
Close
32

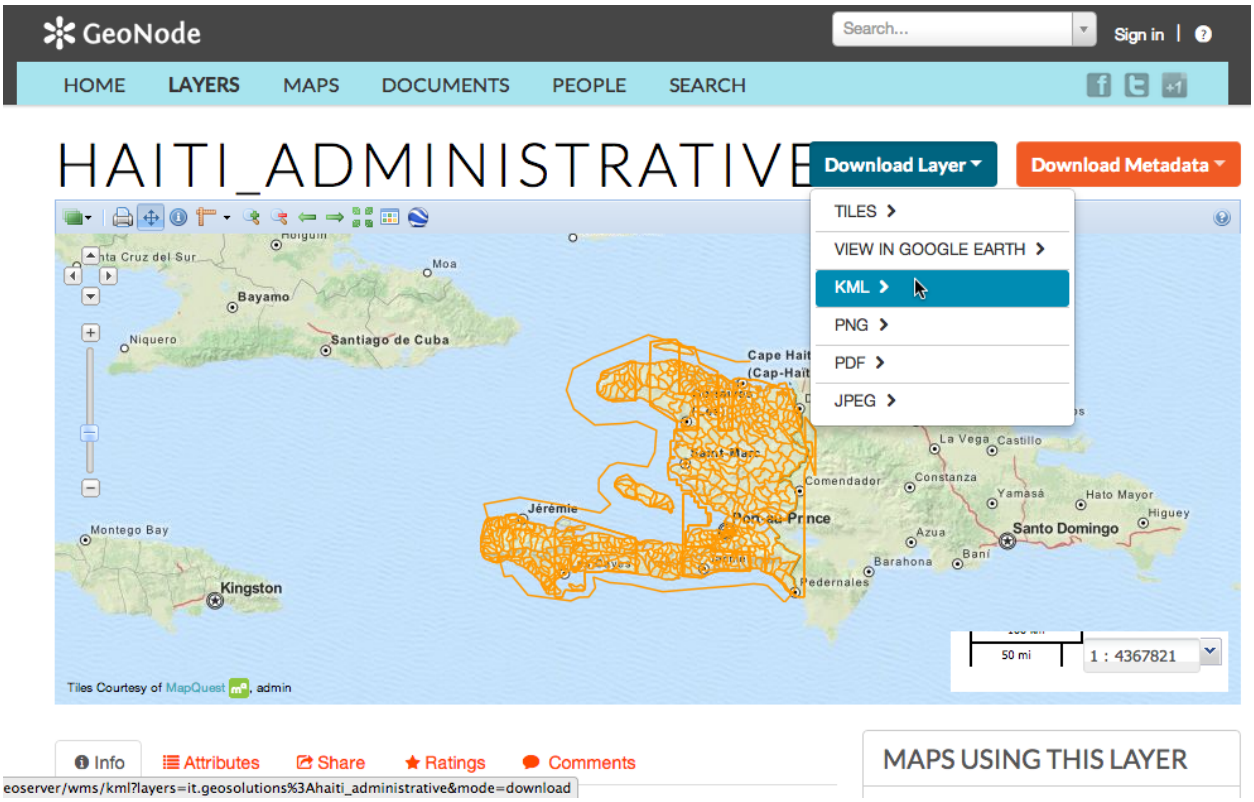

Layer Properties – it.geosolutions:Traffic_Signal
Style
Labels
Fields
General
Metadata
Actions
Joins
Diagrams
Id
Name
Type
Length
Precision
Comment
Edit widget
Alias
0
SAPID
xsd:string
0
0
Line edit
1
INTERSECTI
xsd:int
0
0
Line edit
2
SIGNAL_ID
xsd:string
0
0
Line edit
Restore Default Style
Save As Default
Load Style ...
Save Style ...
Help
Apply
Cancel
OK

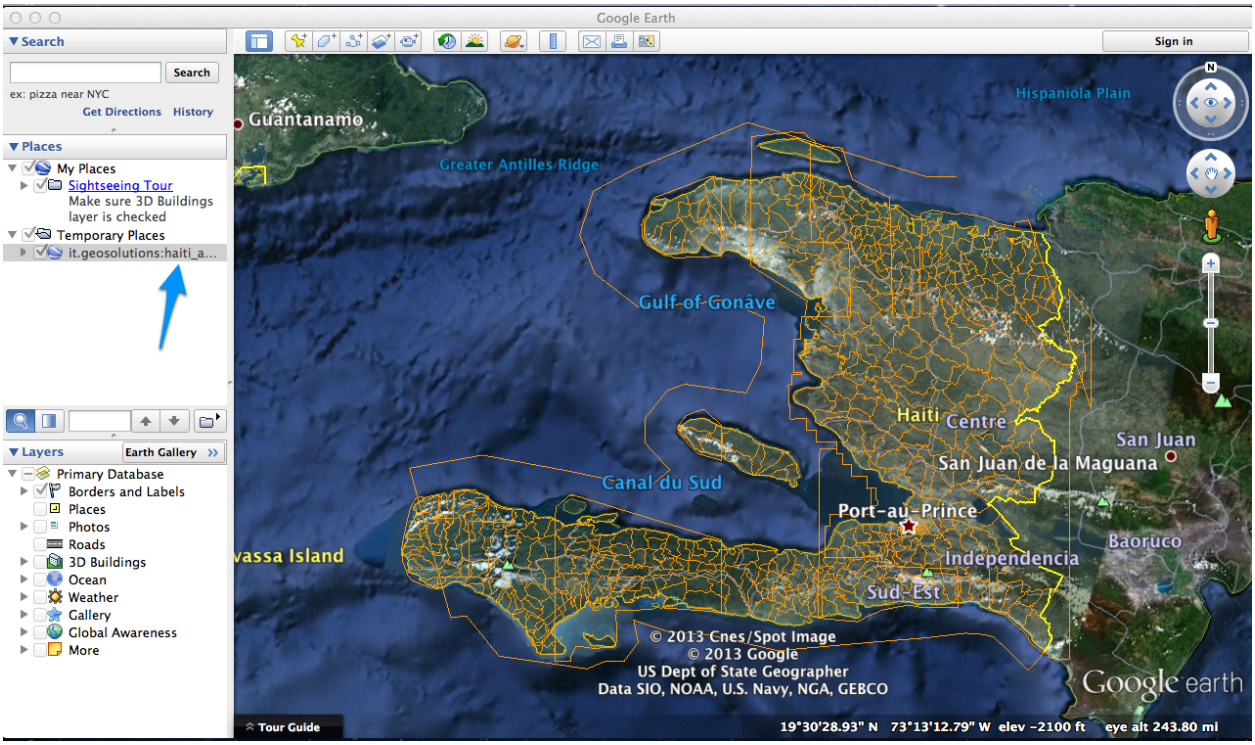

Layer Properties – it.geosolutions:Traffic_Signal
Style
Labels
Fields
General
Metadata
Actions
Joins
Diagrams
Legend type
Single Symbol
Transparency: 0%
New symbology
Point Symbol
Size
2.00000
In map units
Label
Fill options
Outline options
Width
0.26
Restore Default Style
Save As Default
Load Style ...
Save Style ...
Help
Apply
Cancel
OK

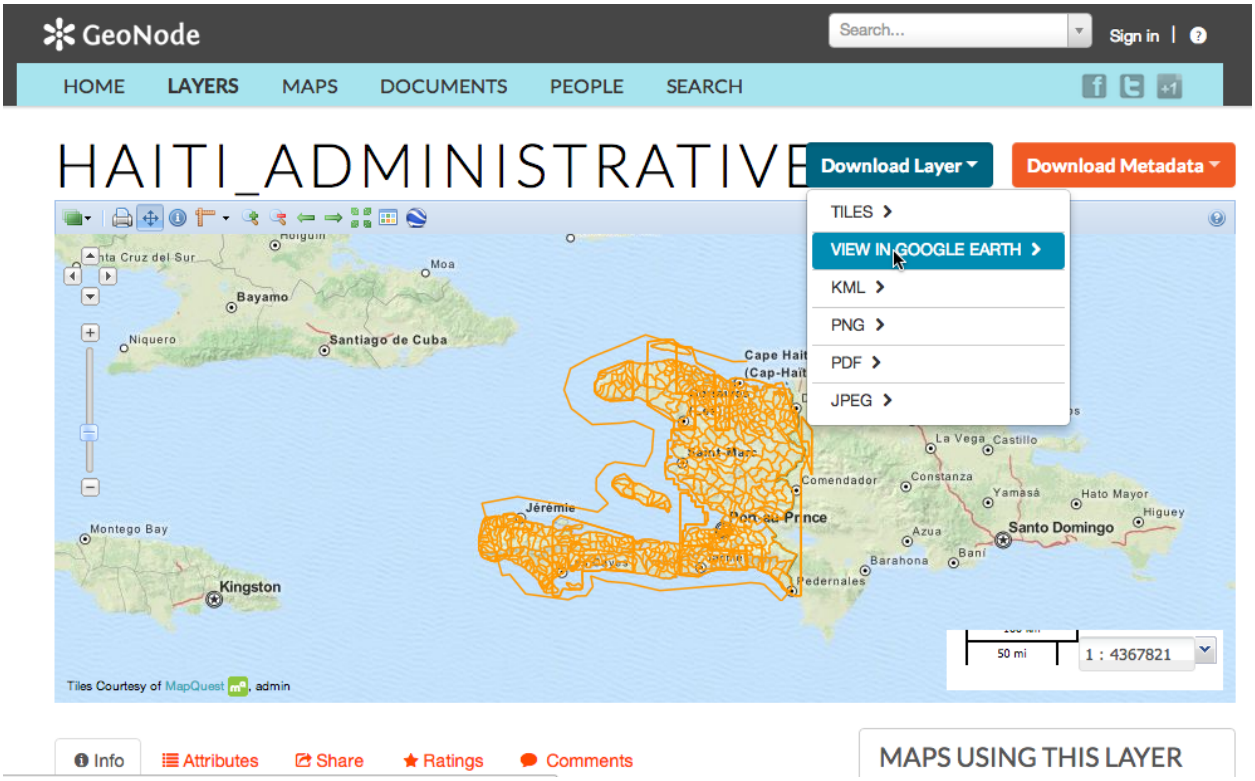

GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2013 Cnes/Spot Image
Image © 2013 TerraMetrics
Image U.S. Geological Survey
Google earth
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER

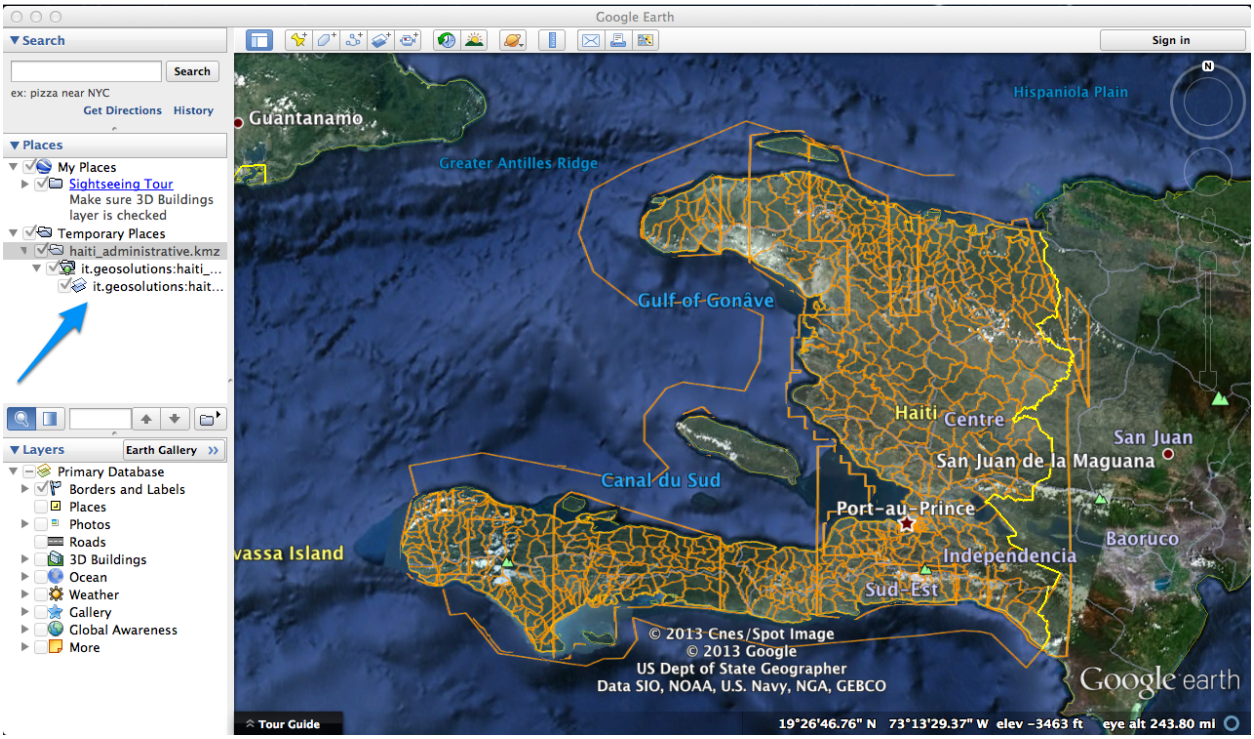

GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
haiti_administrative.1160
haiti_administrative
ADMIN_LEVE: 4
Image © 2013 GeoEye
Image © 2013 DigitalGlobe
© 2013 Cnes/Spot Image
Google earth
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER
This layer is not currently used in any maps.

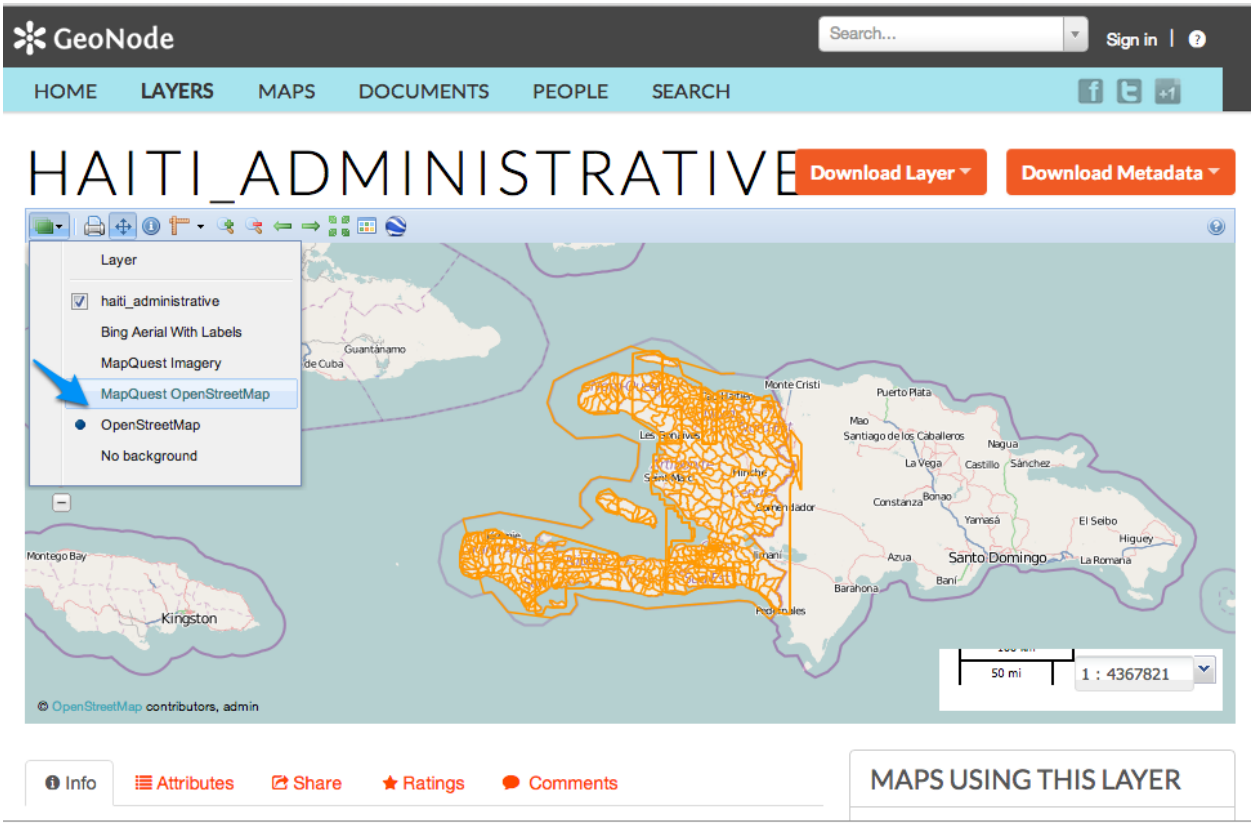

GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Layers
Overlays
haiti_administrative
Base Maps
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2013 Cnes/Spot Image
Google earth


GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
TILES
VIEW IN GOOGLE EARTH
KML
PNG
PDF
JPEG
Bayamo
Moa
Niquero
Santiago de Cuba
Jérémie
Montego Bay
Kingston
Comendador
Constanza
Yamasá
Hato Mayor
Higuey
Azua
Santo Domingo
Barahona
Baní
Pedernales
La Vega
Castillo
50 mi
1 : 4367821
Tiles Courtesy of MapQuest, admin
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER
eoserver/wms/kml?layers=it.geosolutions%3Ahaiti_administrative&mode=download


Google Earth
Search
Search
ex: pizza near NYC
Get Directions
History
Places
My Places
Sightseeing Tour
Make sure 3D Buildings layer is checked
Temporary Places
it.geosolutions:haiti_a...
Layers
Earth Gallery
Primary Database
Borders and Labels
Places
Photos
Roads
3D Buildings
Ocean
Weather
Gallery
Global Awareness
More
Sign in
Guantanamo
Greater Antilles Ridge
Hispaniola Plain
Gulf of Gonâve
Haiti Centre
San Juan
San Juan de la Maguana
Canal du Sud
Port-au-Prince
Baoruco
Independencia
Sud-Est
© 2013 Cnes/Spot Image
© 2013 Google
US Dept of State Geographer
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth
Tour Guide
19°30'28.93" N 73°13'12.79" W elev -2100 ft eye alt 243.80 mi


GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
TILES
VIEW IN GOOGLE EARTH
KML
PNG
PDF
JPEG
Bayamo
Moa
Niquero
Santiago de Cuba
Montego Bay
Kingston
Jérémie
Comendador
Constanza
La Vega
Castillo
Yamasá
Hato Mayor
Higuey
Santo Domingo
Azua
Barahona
Bani
Pedernales
50 mi
1 : 4367821
Tiles Courtesy of MapQuest, admin
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER


Google Earth
Search
Sign in
Guantanamo
Hispaniola Plain
Greater Antilles Ridge
Gulf-of-Gonâve
Haiti Centre
San Juan
San Juan de la Maguana
Canal du Sud
Port-au-Prince
Baoruco
Independencia
Sud-Est
© 2013 Cnes/Spot Image
© 2013 Google
US Dept of State Geographer
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth
Tour Guide
19°26'46.76" N 73°13'29.37" W elev -3463 ft eye alt 243.80 mi
Places
My Places
Sightseeing Tour
Make sure 3D Buildings layer is checked
Temporary Places
haiti_administrative.kmz
Layers
Earth Gallery
Primary Database
Borders and Labels
Places
Photos
Roads
3D Buildings
Ocean
Weather
Gallery
Global Awareness
More


GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
Layer
haiti_administrative
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
Guantánamo
Monte Cristi
Puerto Plata
Mao
Santiago de los Caballeros
Nagua
La Vega
Castillo
Sánchez
Constanza
Bonao
Yamasá
El Seibo
Higuey
Azua
Santo Domingo
La Romana
Baní
Barahona
Montego Bay
Kingston
50 mi
1 : 4367821
© OpenStreetMap contributors, admin
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER


MAP_BASELAYERS = [{
"source": {
"ptype": "gxp_wmscsource",
"url": GEOSERVER_BASE_URL + "wms",
"restUrl": "/gs/rest"
}
},{
"source": {"ptype": "gxp_olsource"},
"type":"OpenLayers.Layer",
"args":["No background"],
"visibility": False,
"fixed": True,
"group":"background"
}, {
"source": {"ptype": "gxp_olsource"},
"type":"OpenLayers.Layer.OSM",
"args":["OpenStreetMap"],
"visibility": True,
"fixed": True,
"group":"background"
}, {
"source": {"ptype": "gxp_mapquestsource"},
"name":"osm",
"group":"background",
"visibility": False
}, {
"source": {"ptype": "gxp_mapquestsource"},
"name":"naip",
"group":"background",
"visibility": False
}, {
"source": {"ptype": "gxp_bingsource"},
"name": "AerialWithLabels",
"fixed": True,
"visibility": False,
"group":"background"
},{
"source": {"ptype": "gxp_mapboxsource"},
}, {
"source": {"ptype": "gxp_olsource"},
"type":"OpenLayers.Layer.WMS",

CHAPTER

FOUR

LOADING DATA INTO A GEONODE

This module will walk you through the various options available to load data into your GeoNode from GeoServer, on the command-line or programatically. You can choose from among these techniques depending on what kind of data you have and how you have your geonode setup.

4.1 Using importlayers to import Data into GeoNode

The geonode.layers app includes 2 management commands that you can use to load or configure data in your GeoNode. Both of these are invoked by using the manage.py script. This section will walk you through how to use the importlayers management command and the subsequent section will lead you through the process of using updatelayers.

First, make sure the virtual environment for your project is activated. If you have installed from packages and followed the steps from Chapter 3, this can be accomplished by issuing the following command:

```
$ source /var/lib/geonode/bin/activate
```

Once your virtualenv is activated, the first thing to do is to use the –help option to the importlayers command to investigate the options to this management command. You can display this help by executing the following command:

```
$ python manage.py importlayers --help
```

This will produce output that looks like the following:

```
Usage: manage.py importlayers [options] path [path...]

Brings a data file or a directory full of data files into aGeoNode site.  Layers are added to the Dja

Options:
  -v VERBOSITY, --verbosity=VERBOSITY
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings=SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath=PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Print traceback on exception
  -u USER, --user=USER  Name of the user account which should own the imported
                        layers
  -i, --ignore-errors   Stop after any errors are encountered.
```

```
  -o, --overwrite       Overwrite existing layers if discovered (defaults
                        False)
  -k KEYWORDS, --keywords=KEYWORDS
                        The default keywords for the imported layer(s). Will
                        be the same for all imported layers if multiple
                        imports are done in one command
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

While the description of most of the options should be self explanatory, its worth reviewing some of the key options in a bit more detail.

- The -i option will force the command to stop when it first encounters an error. Without this option specified, the process will skip over errors that have layers and continue loading the other layers.
- The -o option specifies that layers with the same name as the base name will be loaded and overwrite the existing layer.
- The -k option is used to add keywords for all of the layers imported.

The import layers management command is invoked by specifying options as described above and specifying the path to a single layer file or to a directory that contains multiple files. For purposes of this exercise, lets use the default set of testing layers that ship with geonode. You can replace this path with the directory to your own shapefiles:

```
$ python manage.py importlayers -v 3 /var/lib/geonode/lib/python2.7/site-packages/gisdata/data/good/v
```

This command will produce the following output to your terminal:

```
Verifying that GeoNode is running ...
Found 8 potential layers.
No handlers could be found for logger "pycsw"
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec

Detailed report of failures:


Finished processing 8 layers in 30.0 seconds.

8 Created layers
0 Updated layers
0 Skipped layers
0 Failed layers
3.750000 seconds per layer
```

If you encounter errors while running this command, you can use the -v option to increase the verbosity of the output so you can debug the problem. The verbosity level can be set from 0-3 with 0 being the default. An example of what the output looks like when an error is encountered and the verbosity is set to 3 is shown below:

```
Verifying that GeoNode is running ...
Found 8 potential layers.
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
```

```
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect

Detailed report of failures:


/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vector/san_andres_y_prov
================
Traceback (most recent call last):
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 682, in upload
    keywords=keywords,
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 602, in file_upload
    keywords=keywords, title=title)
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 305, in save
    store = cat.get_store(name)
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 176, i
    for ws in self.get_workspaces():
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 489, i
    description = self.get_xml("%s/workspaces.xml" % self.service_url)
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 136, i
    response, content = self.http.request(rest_url)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1445, in request
    (response, content) = self._request(conn, authority, uri, request_uri, method, body, headers, red
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1197, in _request
    (response, content) = self._conn_request(conn, request_uri, method, body, headers)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1133, in _conn_request
    conn.connect()
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 799, in connect
    raise socket.error, msg
error: [Errno 61] Connection refused
```

Note: This last section of output will be repeated for all layers, and only the first one is show above.

This error indicates that GeoNode was unable to connect to GeoServer to load the layers. To solve this, you should make sure GeoServer is running and re-run the command.

If you encounter errors with this command that you cannot solve, you should bring them up on the geonode users mailing list.

You should now have the knowledge necessary to import layers into your GeoNode project from a directory on the servers filesystem and can use this to load many layers into your GeoNode at once.

Note: The ownership of the imported layers will be assigned to the primary superuser in your system. You can use GeoNodes Django Admin interface to modify this after the fact if you want them to be owned by another user.

4.2 GeoServer Data Configuration

While it is possible to import layers directly from your servers filesystem into your GeoNode, you may have an existing GeoServer that already has data in it, or you may want to configure data from a GeoServer which is not directly supported by uploading data. GeoServer supports a wide range of data formats and connections to database,

and while many of them are not supported as GeoNode upload formats, if they can be configured in GeoServer, you can add them to your GeoNode by following the procedure described below.

GeoServer supprts 3 types of data: Raster, Vector and Databases. For a list of the supported formats for each type of data, consult the following pages.

- http://docs.geoserver.org/latest/en/user/data/vector/index.html#data-vector
- http://docs.geoserver.org/latest/en/user/data/raster/index.html
- http://docs.geoserver.org/latest/en/user/data/database/index.html

---

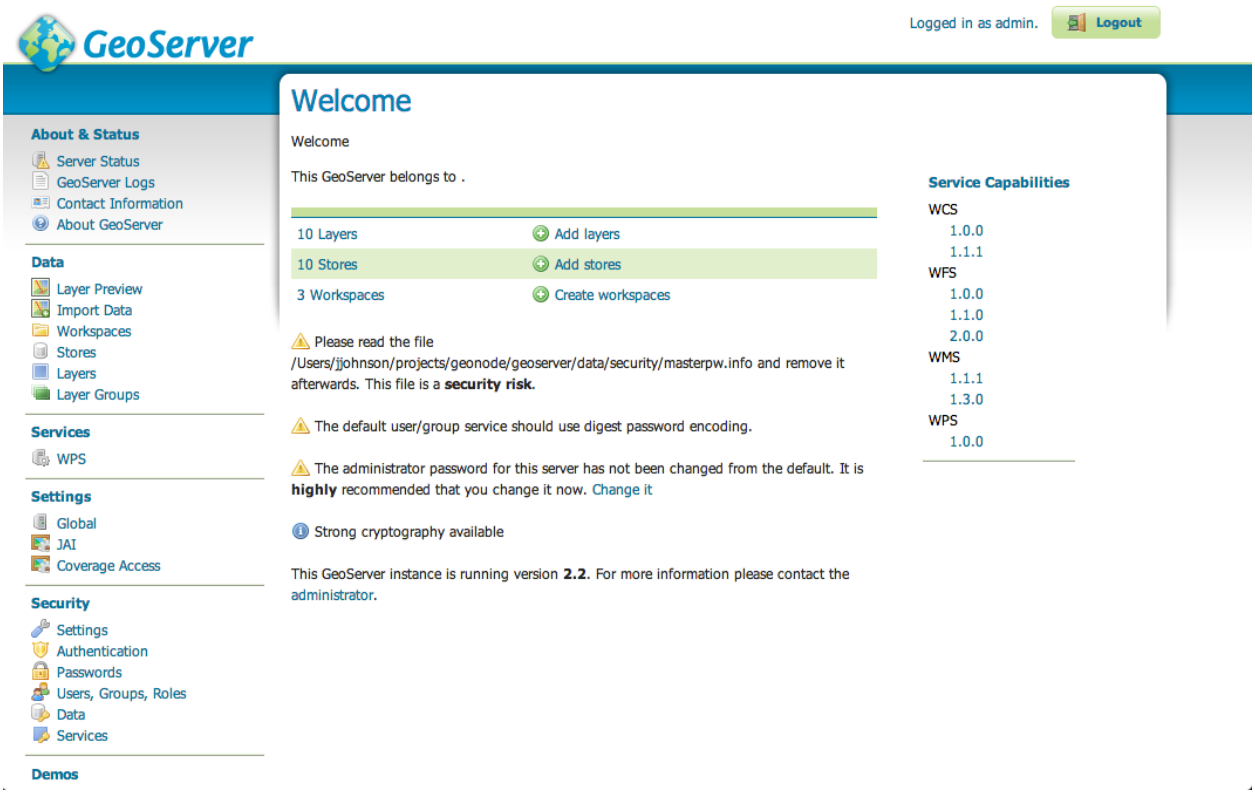
Note: Some of these raster or vector formats or database types require that you install specific plugins in your GeoServer in order to use the. Please consult the GeoServer documentation for more information.

Lets walk through an example of configuring a PostGIS database in GeoServer and then configuring those layers in your GeoNode.

First visit the GeoServer administration interface on your server. This is usually on port 8080 and is available at http://localhost:8080/geoserver/web/

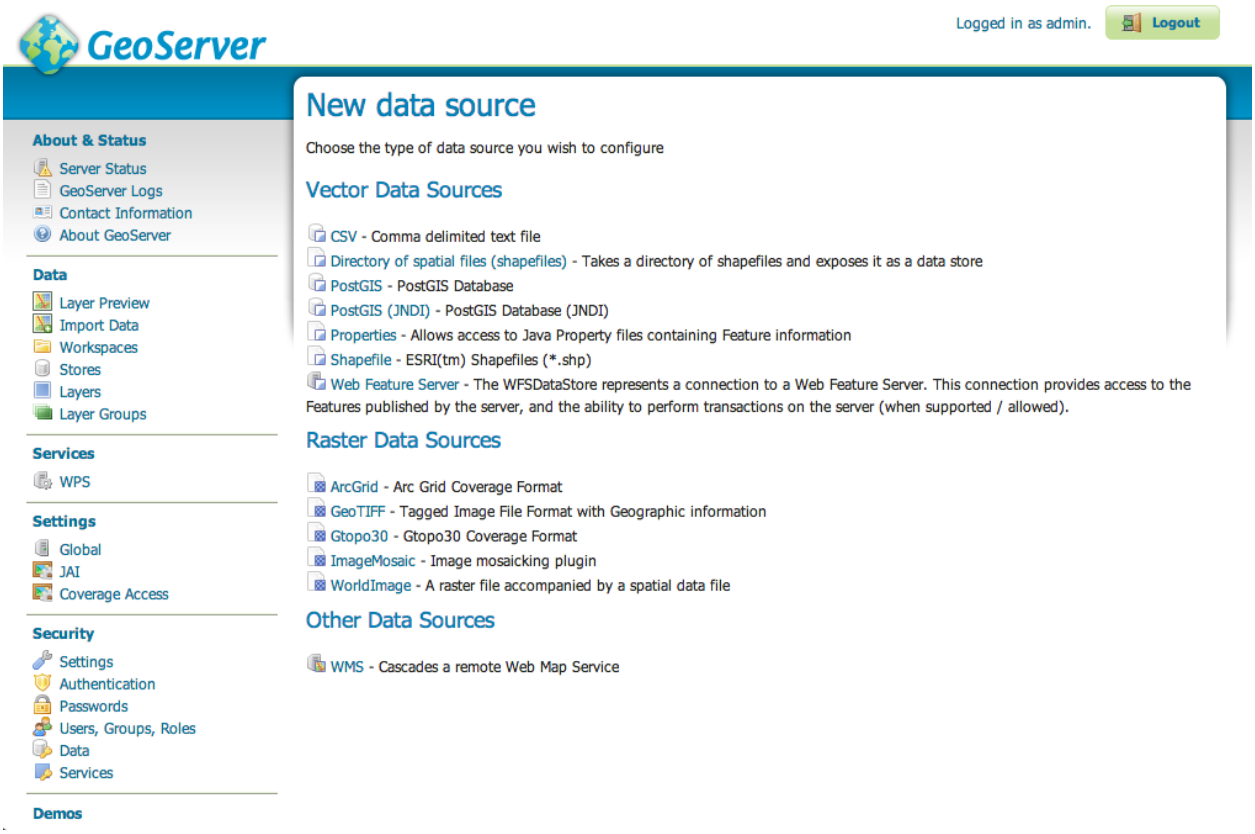
You should login with the superuser credentials you setup when you first configured your GeoNode instance.

Once you are logged in to the GeoServer Admin interface, you should see the following.



Note: The number of stores, layers and workspaces may be different depending on what you already have configured in your GeoServer.

Next you want to select the "Stores" option in the left hand menu, and then the "Add new Store" option. The following screen will be displayed.



In this case, we want to select the PostGIS store type to create a connection to our existing database. On the next screen you will need to enter the parameters to connect to your PostGIS database (alter as necessary for your own database).

Note: If you are unsure about any of the settings, leave them as the default.

The next screen lets you configure the layers in your database. This will of course be different depending on the layers in your database.

Select the "Publish" button for one of the layers and the next screen will be displayed where you can enter metadata for this layer. Since we will be managing this metadata in GeoNode, we can leave these alone for now.

The things that *must* be specified are the Declared SRS and you must select the "Compute from Data" and "Compute from native bounds" links after the SRS is specified.

Click save and this layer will now be configured for use in your GeoServer.

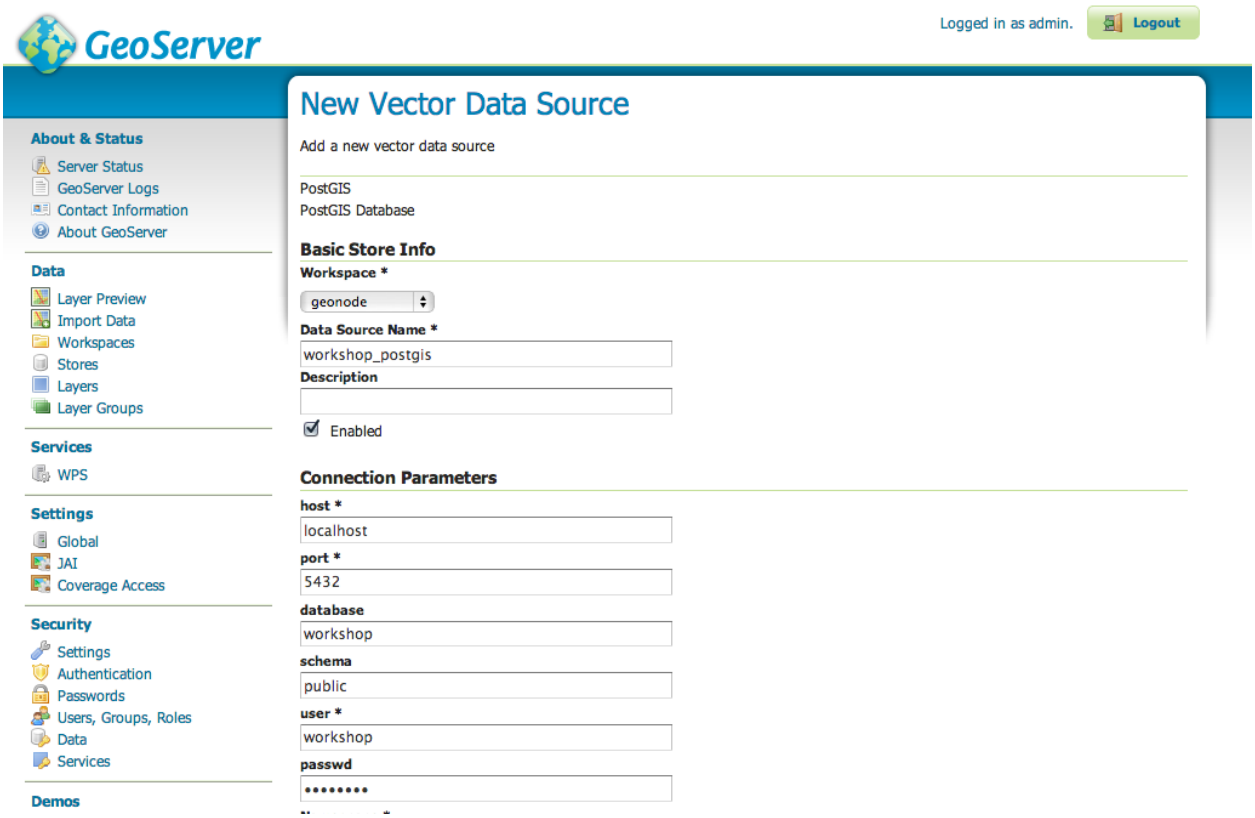
The next step is to configure these layers in GeoNode. The updatelayers management command is used for this purpose. As with importlayers, its useful to look at the command line options for this command by passing the –help option:

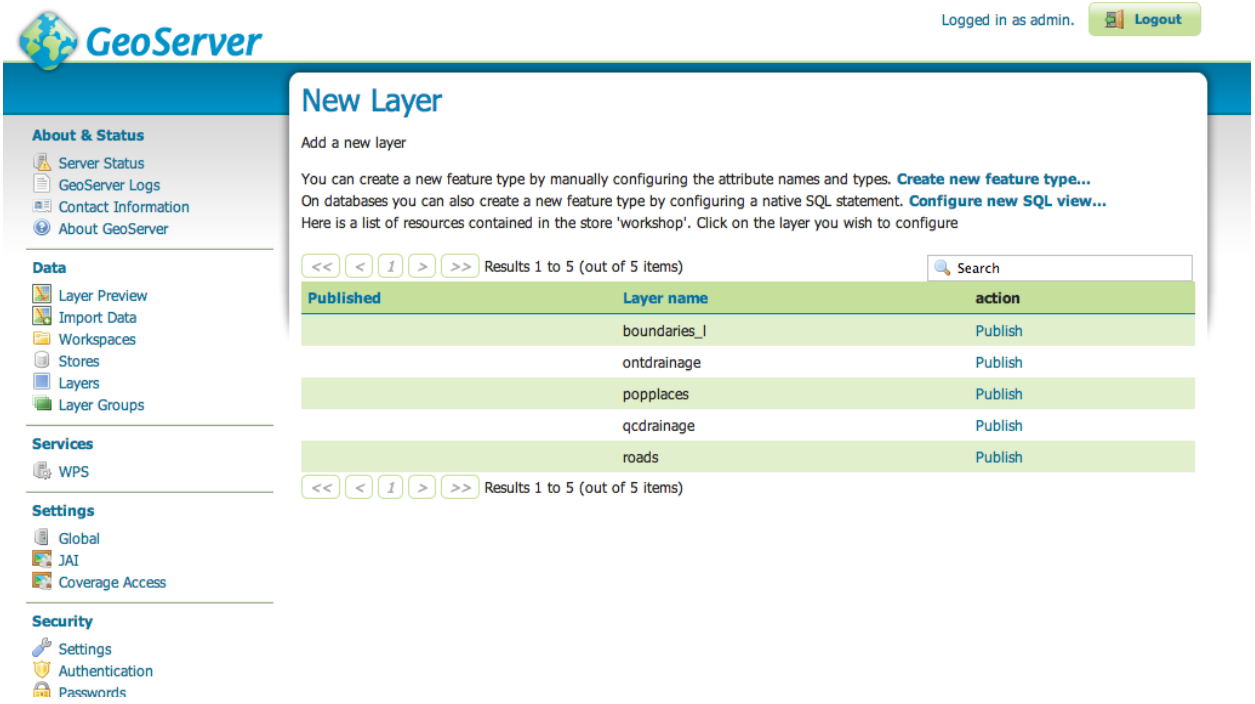
```
$ python manage.py updatelayers --help
```

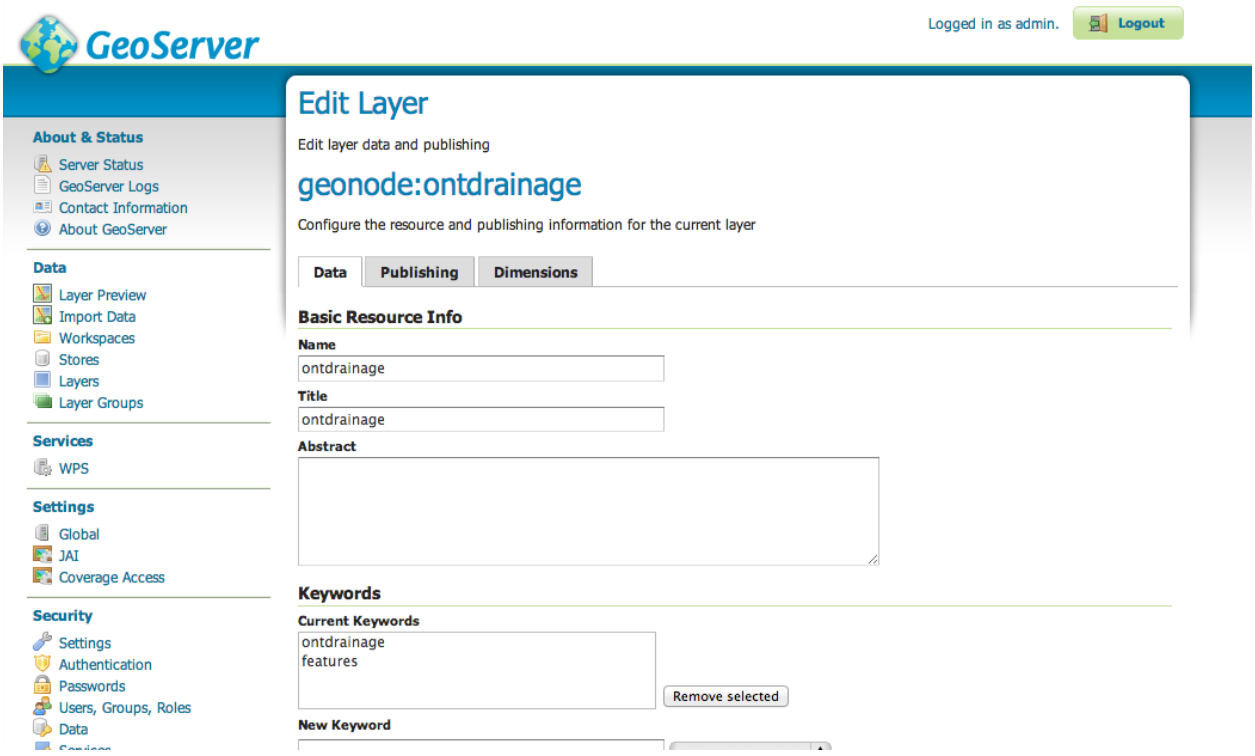
This help option displays the following:

```
Usage: manage.py updatelayers [options]

Update the GeoNode application with data from GeoServer
```


GeoServer
Logged in as admin.
Logout
About & Status
Server Status
GeoServer Logs
Contact Information
About GeoServer
Data
Layer Preview
Import Data
Workspaces
Stores
Layers
Layer Groups
Services
WPS
Settings
Global
JAI
Coverage Access
Security
Settings
Authentication
Passwords
Users, Groups, Roles
Data
Services
Demos
New Vector Data Source
Add a new vector data source
PostGIS
PostGIS Database
Basic Store Info
Workspace *
geonode
Data Source Name *
workshop_postgis
Description
Enabled
Connection Parameters
host *
localhost
port *
5432
database
workshop
schema
public
user *
workshop
passwd
••••••••


GeoServer
Logged in as admin.
Logout
About & Status
Server Status
GeoServer Logs
Contact Information
About GeoServer
Data
Layer Preview
Import Data
Workspaces
Stores
Layers
Layer Groups
Services
WPS
Settings
Global
JAI
Coverage Access
Security
Settings
Authentication
Passwords
New Layer
Add a new layer
You can create a new feature type by manually configuring the attribute names and types. Create new feature type...
On databases you can also create a new feature type by configuring a native SQL statement. Configure new SQL view...
Here is a list of resources contained in the store 'workshop'. Click on the layer you wish to configure
<< < 1 > >> Results 1 to 5 (out of 5 items)
Search
Published
Layer name
action
boundaries_l
Publish
ontdrainage
Publish
popplaces
Publish
qcdrainage
Publish
roads
Publish
<< < 1 > >> Results 1 to 5 (out of 5 items)



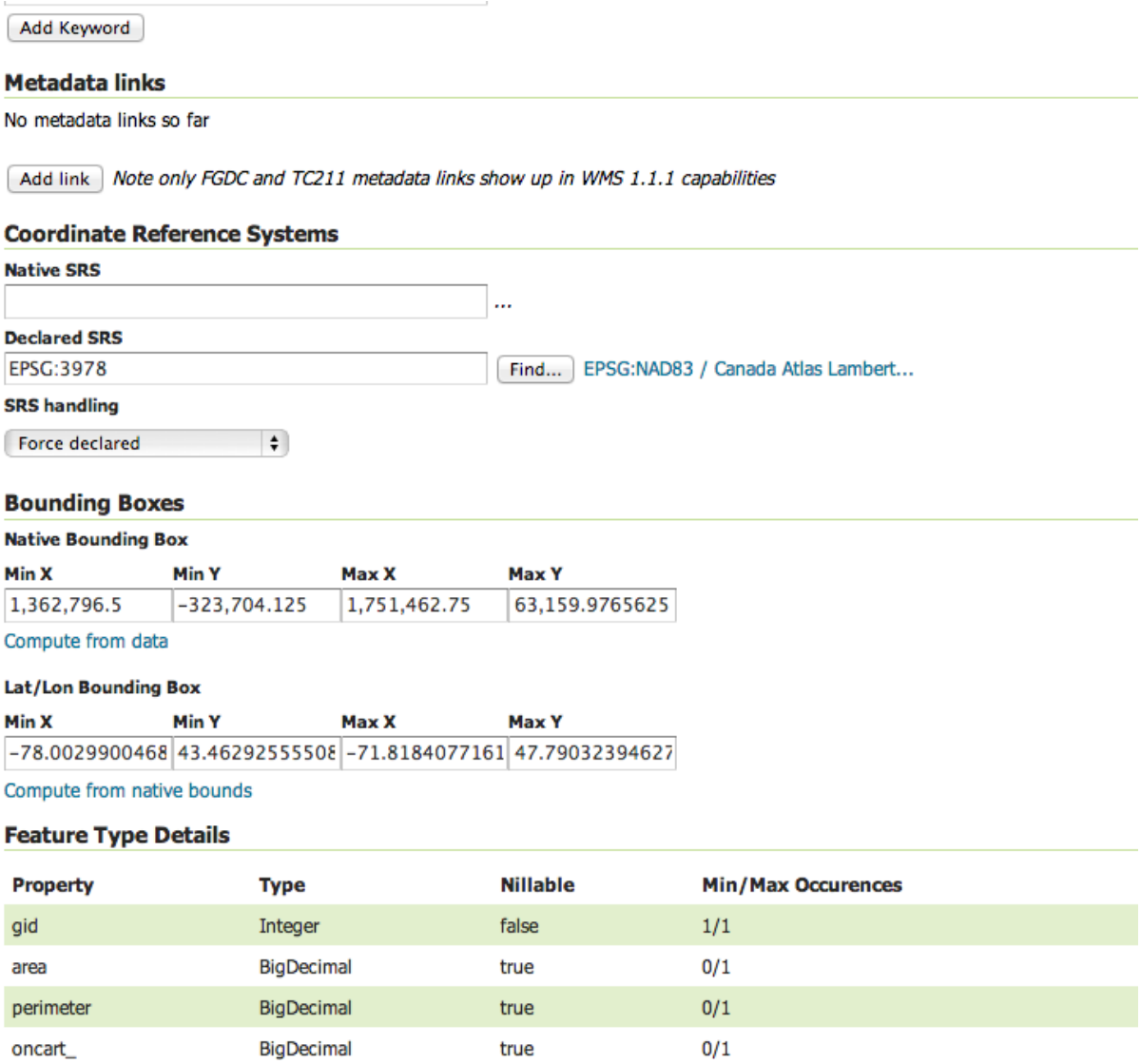

```
Options:
  -v VERBOSITY, --verbosity=VERBOSITY
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings=SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath=PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Print traceback on exception
  -i, --ignore-errors   Stop after any errors are encountered.
  -u USER, --user=USER  Name of the user account which should own the imported
                        layers
  -w WORKSPACE, --workspace=WORKSPACE
                        Only update data on specified workspace
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

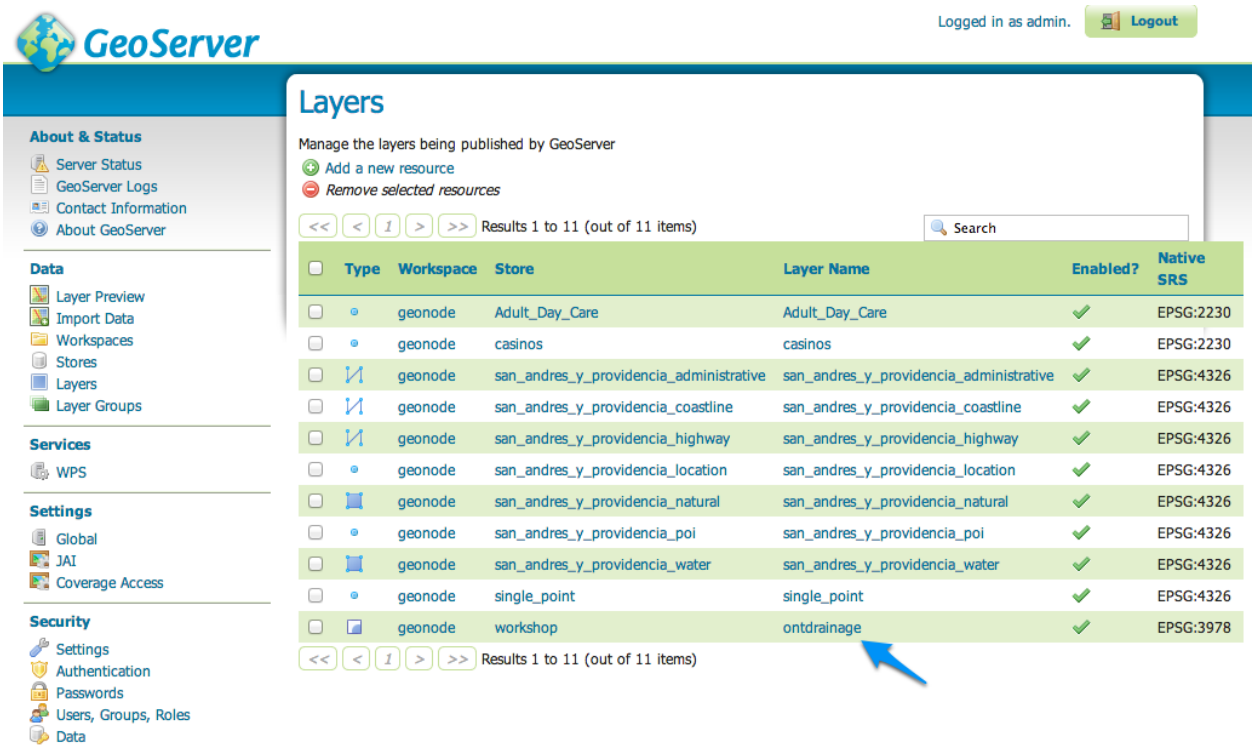
In this case, we can use the default options. So enter the following command to configure the layers from our GeoServer into our GeoNode:

```
$ python manage.py updatelayers
```

The output will look something like the following:

```
No handlers could be found for logger "pycsw"
[created] Layer Adult_Day_Care (1/11)
[created] Layer casinos (2/11)
[updated] Layer san_andres_y_providencia_administrative (3/11)
```


Add Keyword
Metadata links
No metadata links so far
Add link
Note only FGDC and TC211 metadata links show up in WMS 1.1.1 capabilities
Coordinate Reference Systems
Native SRS
...
Declared SRS
EPSG:3978
Find...
EPSG:NAD83 / Canada Atlas Lambert...
SRS handling
Force declared
Bounding Boxes
Native Bounding Box
Min X
Min Y
Max X
Max Y
1,362,796.5
-323,704.125
1,751,462.75
63,159.9765625
Compute from data
Lat/Lon Bounding Box
Min X
Min Y
Max X
Max Y
-78.0029900468
43.46292555508
-71.8184077161
47.79032394627
Compute from native bounds
Feature Type Details
Property
Type
Nillable
Min/Max Occurences
gid
Integer
false
1/1
area
BigDecimal
true
0/1
perimeter
BigDecimal
true
0/1
oncart_
BigDecimal
true
0/1




```
[updated] Layer san_andres_y_providencia_coastline (4/11)
[updated] Layer san_andres_y_providencia_highway (5/11)
[updated] Layer san_andres_y_providencia_location (6/11)
[updated] Layer san_andres_y_providencia_natural (7/11)
[updated] Layer san_andres_y_providencia_poi (8/11)
[updated] Layer san_andres_y_providencia_water (9/11)
[updated] Layer single_point (10/11)
[created] Layer ontdrainage (11/11)


Finished processing 11 layers in 45.0 seconds.

3 Created layers
8 Updated layers
0 Failed layers
4.090909 seconds per layer
```

Note: This example picked up 2 additional layers that were already in our GeoServer, but were not already in our GeoNode.

For layers that already exist in your GeoNode, they will be updated and the configuration synchronized between GeoServer and GeoNode.

You can now view and use these layers in your GeoNode.

4.3 Using GDAL and OGR to convert your Data for use in GeoNode

GeoNode supports uploading data in shapefiles, csv, kml and GeoTiff formats. If your data is in other formats, you will need to convert it into one of these formats for use in GeoNode. This section will walk you through the steps necessary to convert your data into formats suitable for uploading into GeoNode.

You will need to make sure that you have the gdal library installed on your system. On Ubuntu you can install this package with the following command:

```
$ sudo apt-get install gdal-bin
```

4.3.1 OGR (Vector Data)

OGR is used to manipulate vector data. In this example, we will use MapInfo .tab files and convert them to shapefiles with the ogr2ogr command. We will use sample MapInfo files from the website linked below.

http://services.land.vic.gov.au/landchannel/content/help?name=sampledata

You can download the Admin;(Postcode) layer by issuing the following command:

```
$ wget http://services.land.vic.gov.au/sampledata/mif/admin_postcode_vm.zip
```

You will need to unzip this dataset by issuing the following command:

```
$ unzip admin_postcode_vm.zip
```

This will leave you with the following files in the directory where you executed the above commands:

```
|-- ANZVI0803003025.htm
|-- DSE_Data_Access_Licence.pdf
|-- VMADMIN.POSTCODE_POLYGON.xml
|-- admin_postcode_vm.zip
--- vicgrid94
    --- mif
        --- lga_polygon
            --- macedon\ ranges
                |-- EXTRACT_POLYGON.mid
                |-- EXTRACT_POLYGON.mif
                --- VMADMIN
                    |-- POSTCODE_POLYGON.mid
                    --- POSTCODE_POLYGON.mif
```

First, lets inspect this file set using the following command:

```
$ ogrinfo -so vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POSTCODE_POLYGON.mid POSTCODE_POLYGON
```

The output will look like the following:

```
Had to open data source read-only.
INFO: Open of 'vicgrid94/mif/lga_polygon/macedon ranges/VMADMIN/POSTCODE_POLYGON.mid'
    using driver 'MapInfo File' successful.

Layer name: POSTCODE_POLYGON
Geometry: 3D Unknown (any)
Feature Count: 26
Extent: (2413931.249367, 2400162.366186) - (2508952.174431, 2512183.046927)
Layer SRS WKT:
PROJCS["unnamed",
```

```
    GEOGCS["unnamed",
        DATUM["GDA94",
            SPHEROID["GRS 80",6378137,298.257222101],
            TOWGS84[0,0,0,-0,-0,-0,0]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433]],
    PROJECTION["Lambert_Conformal_Conic_2SP"],
    PARAMETER["standard_parallel_1",-36],
    PARAMETER["standard_parallel_2",-38],
    PARAMETER["latitude_of_origin",-37],
    PARAMETER["central_meridian",145],
    PARAMETER["false_easting",2500000],
    PARAMETER["false_northing",2500000],
    UNIT["Meter",1]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TYPE: String (6.0)
FEATURE_QUALITY_ID: String (20.0)
PFI_CREATED: Date (10.0)
UFI: Real (12.0)
UFI_CREATED: Date (10.0)
UFI_OLD: Real (12.0)
```

This gives you information about the number of features, the extent, the projection and the attributes of this layer.

Next, lets go ahead and convert this layer into a shapefile by issuing the following command:

```
$ ogr2ogr postcode_polygon.shp vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POSTCODE_POLYGON.mi
```

The output of this command will look like the following:

```
Warning 6: Normalized/laundered field name: 'FEATURE_TYPE' to 'FEATURE_TY'
Warning 6: Normalized/laundered field name: 'FEATURE_QUALITY_ID' to 'FEATURE_QU'
Warning 6: Normalized/laundered field name: 'PFI_CREATED' to 'PFI_CREATE'
Warning 6: Normalized/laundered field name: 'UFI_CREATED' to 'UFI_CREATE'
```

This output indicates that some of the field names were truncated to fit into the constraint that attributes in shapefiles are only 10 characters long.

You will now have a set of files that make up the postcode_polygon.shp shapefile set. We can inspect them by issuing the following command:

```
$ ogrinfo -so postcode_polygon.shp postcode_polygon
```

The output will look similar to the output we saw above when we inspected the MapInfo file we converted from:

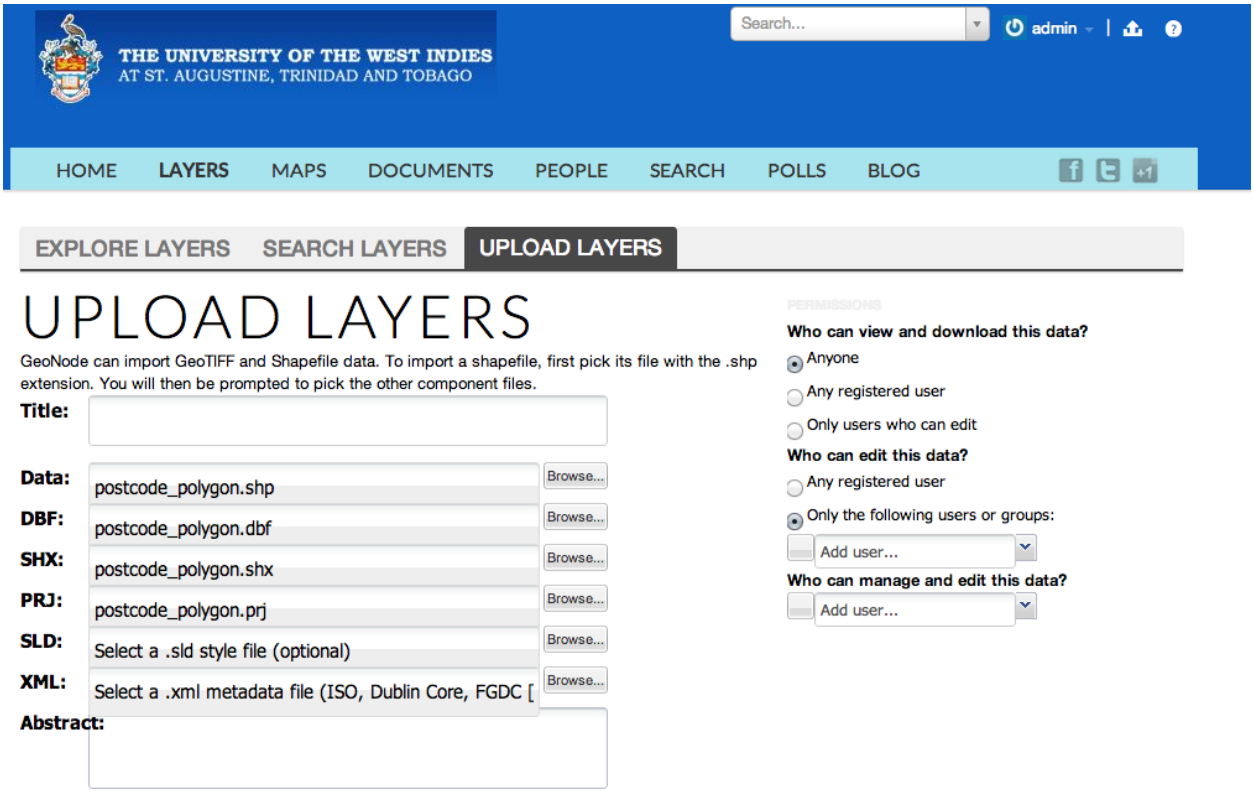
```
INFO: Open of `postcode_polygon.shp'
    using driver `ESRI Shapefile' successful.

Layer name: postcode_polygon
Geometry: 3D Polygon
Feature Count: 26
Extent: (2413931.249367, 2400162.366186) - (2508952.174431, 2512183.046927)
Layer SRS WKT:
PROJCS["Lambert_Conformal_Conic",
    GEOGCS["GCS_Unknown",
        DATUM["GDA94",
            SPHEROID["GRS_80",6378137,298.257222101]],
        PRIMEM["Greenwich",0],
        UNIT["Degree",0.0174532925199433]],
```

```
    PROJECTION["Lambert_Conformal_Conic_2SP"],
    PARAMETER["standard_parallel_1",-36],
    PARAMETER["standard_parallel_2",-38],
    PARAMETER["latitude_of_origin",-37],
    PARAMETER["central_meridian",145],
    PARAMETER["false_easting",2500000],
    PARAMETER["false_northing",2500000],
    UNIT["Meter",1]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TY: String (6.0)
FEATURE_QU: String (20.0)
PFI_CREATE: Date (10.0)
UFI: Real (12.0)
UFI_CREATE: Date (10.0)
UFI_OLD: Real (12.0)
```

These files can now be loaded into your GeoNode instance via the normal uploader.

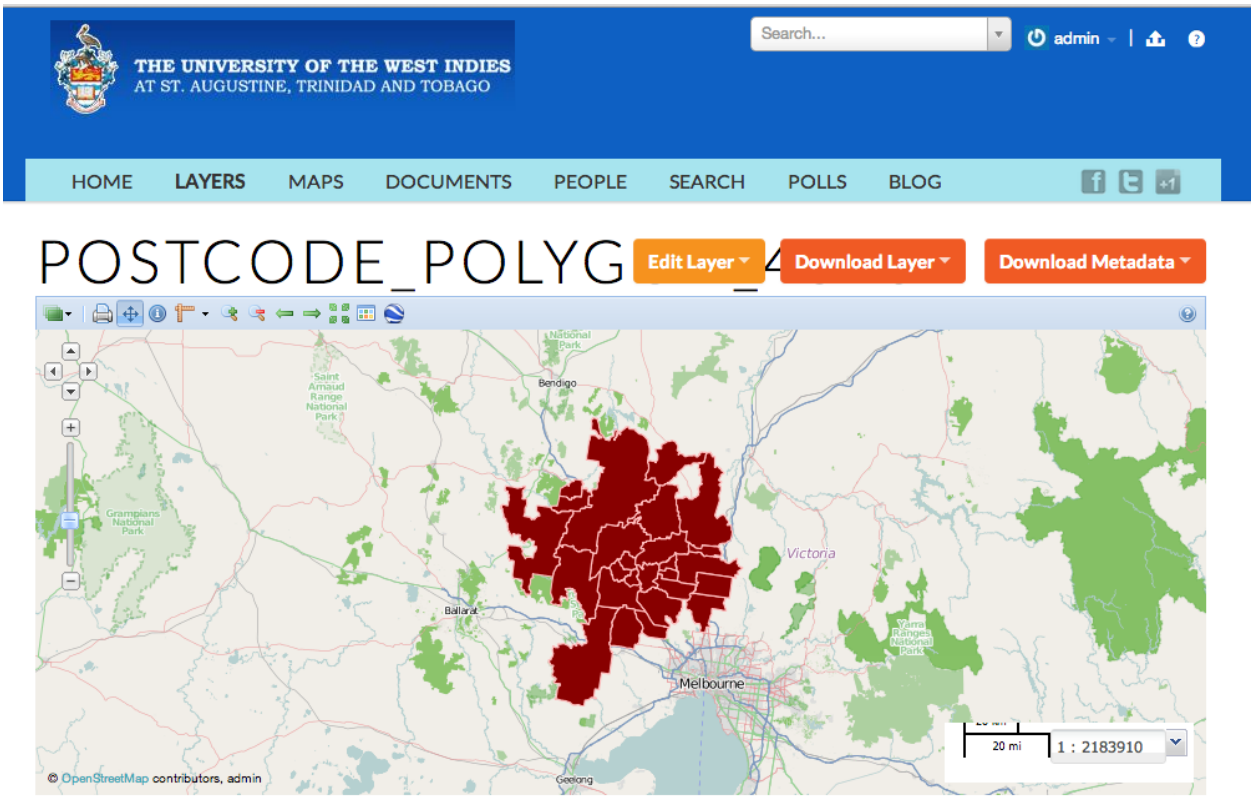
Visit the upload page in your GeoNode.



You can edit the metadata for the layer and then click "Update" and you will see the Layer Info page for this layer.

4.3.2 GDAL (Raster Data)

Now that we have seen how to convert vector layers into shapefiles using ogr2ogr, we will walk through the steps necessary to perform the same operation with Raster layers. For this example, we will work with Arc/Info Binary and ASCII Grid data and convert it into GeoTiff format for use in GeoNode.



First, you need to download the sample data to work with it. You can do this by executing the following command:

```
$ wget http://dev.opengeo.org/~jjohnson/sample_asc.tgz
```

You will need to uncompress this file by executing this command:

```
$ tar -xvzf sample_asc.tgz
```

You will be left with the following files on your filesystem:

```
|-- batemans_ele
|   |-- dblbnd.adf
|   |-- hdr.adf
|   |-- metadata.xml
|   |-- prj.adf
|   |-- sta.adf
|   |-- w001001.adf
|   |-- w001001x.adf
|-- batemans_elevation.asc
```

The file batemans_elevation.asc is an Arc/Info ASCII Grid file and the files in the batemans_ele directory are an Arc/Info Binary Grid file.

You can use the gdalinfo command to inspect both of these files by executing the following command:

```
$ gdalinfo batemans_elevation.asc
```

The output should look like the following:

```
Driver: AAIGrid/Arc/Info ASCII Grid
Files: batemans_elevation.asc
Size is 155, 142
Coordinate System is ''
Origin = (239681.000000000000000,6050551.000000000000000)
Pixel Size = (100.000000000000000,-100.000000000000000)
Corner Coordinates:
Upper Left  (  239681.000, 6050551.000)
Lower Left  (  239681.000, 6036351.000)
Upper Right (  255181.000, 6050551.000)
Lower Right (  255181.000, 6036351.000)
Center      (  247431.000, 6043451.000)
Band 1 Block=155x1 Type=Float32, ColorInterp=Undefined
    NoData Value=-9999
```

You can then inspect the batemans_ele files by executing the following command:

```
Driver: AIG/Arc/Info Binary Grid
Files: batemans_ele
    batemans_ele/dblbnd.adf
    batemans_ele/hdr.adf
    batemans_ele/metadata.xml
    batemans_ele/prj.adf
    batemans_ele/sta.adf
    batemans_ele/w001001.adf
    batemans_ele/w001001x.adf
Size is 155, 142
Coordinate System is:
PROJCS["unnamed",
    GEOGCS["GDA94",
        DATUM["Geocentric_Datum_of_Australia_1994",
            SPHEROID["GRS 1980",6378137,298.257222101,
                AUTHORITY["EPSG","7019"]],
            TOWGS84[0,0,0,0,0,0,0],
            AUTHORITY["EPSG","6283"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4283"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",153],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",10000000],
    UNIT["METERS",1]]
Origin = (239681.000000000000000,6050551.000000000000000)
Pixel Size = (100.000000000000000,-100.000000000000000)
Corner Coordinates:
Upper Left  (  239681.000, 6050551.000) (150d 7'28.35"E, 35d39'16.56"S)
Lower Left  (  239681.000, 6036351.000) (150d 7'11.78"E, 35d46'56.89"S)
Upper Right (  255181.000, 6050551.000) (150d17'44.07"E, 35d39'30.83"S)
Lower Right (  255181.000, 6036351.000) (150d17'28.49"E, 35d47'11.23"S)
Center      (  247431.000, 6043451.000) (150d12'28.17"E, 35d43'13.99"S)
Band 1 Block=256x4 Type=Float32, ColorInterp=Undefined
    Min=-62.102 Max=142.917
NoData Value=-3.4028234663852886e+38
```

You will notice that the batemans_elevation.asc file does *not* contain projection information while the batemans_ele file does. Because of this, lets use the batemans_ele files for this exercise and convert them to a GeoTiff for use in GeoNode. We will also reproject this file into WGS84 in the process. This can be accomplished with the following command.

```
$ gdalwarp -t_srs EPSG:4326 batemans_ele batemans_ele.tif
```

The output will show you the progress of the conversion and when it is complete, you will be left with a batemans_ele.tif file that you can upload to your GeoNode.

You can inspect this file with the gdalinfo command:

```
$ gdalinfo batemans_ele.tif
```

Which will produce the following output:

```
Driver: GTiff/GeoTIFF
Files: batemans_ele.tif
Size is 174, 130
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
Origin = (150.119938943722502,-35.654598806259330)
Pixel Size = (0.001011114155919,-0.001011114155919)
Metadata:
    AREA_OR_POINT=Area
Image Structure Metadata:
    INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 150.1199389, -35.6545988) (150d 7'11.78"E, 35d39'16.56"S)
Lower Left  ( 150.1199389, -35.7860436) (150d 7'11.78"E, 35d47' 9.76"S)
Upper Right ( 150.2958728, -35.6545988) (150d17'45.14"E, 35d39'16.56"S)
Lower Right ( 150.2958728, -35.7860436) (150d17'45.14"E, 35d47' 9.76"S)
Center      ( 150.2079059, -35.7203212) (150d12'28.46"E, 35d43'13.16"S)
Band 1 Block=174x11 Type=Float32, ColorInterp=Gray
```

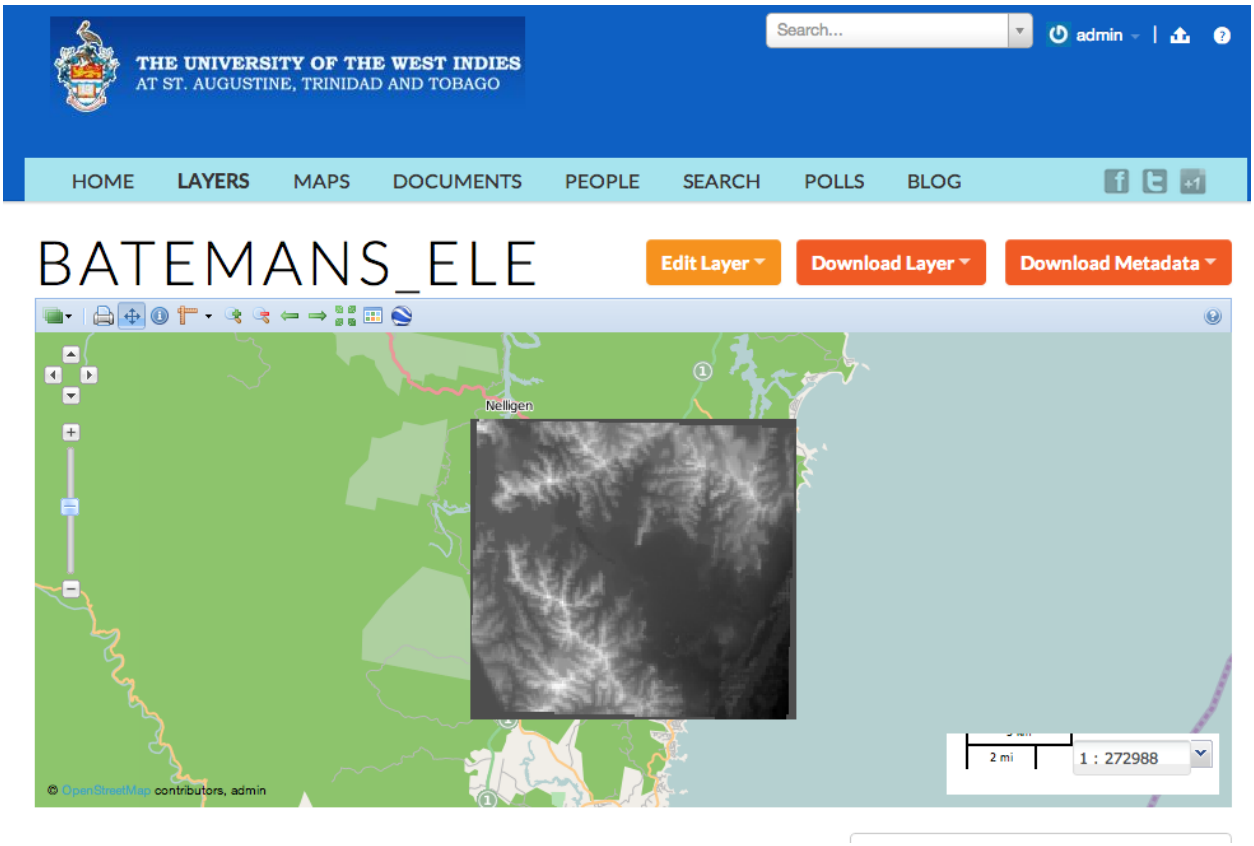
You can then follow the same steps we used above to upload the GeoTiff file we created into the GeoNode, and you will see your layer displayed in the Layer Info page.

Now that you have seen how to convert layers with both OGR and GDAL, you can use these techniques to work with your own data and get it prepared for inclusion in your own GeoNode.

4.4 Loading OSM Data into GeoNode

In this section, we will walk through the steps necessary to load OSM data into your GeoNode project. As discussed in previous sections, your GeoNode already uses OSM tiles from MapQuest and the main OSM servers as some of the available base layers. This session is specifically about extracting actual data from OSM and converting it for use in your project and potentially for Geoprocessing tasks.

The first step in this process is to get the data from OSM. We will be using the OSM Overpass API since it lets us do more complex queries than the OSM API itself. You should refer to the OSM Overpass API documentation to learn



about all of its features. It is an extremely powerful API that lets you extract data from OSM using a very sophisticated API.

- http://wiki.openstreetmap.org/wiki/Overpass_API
- http://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide

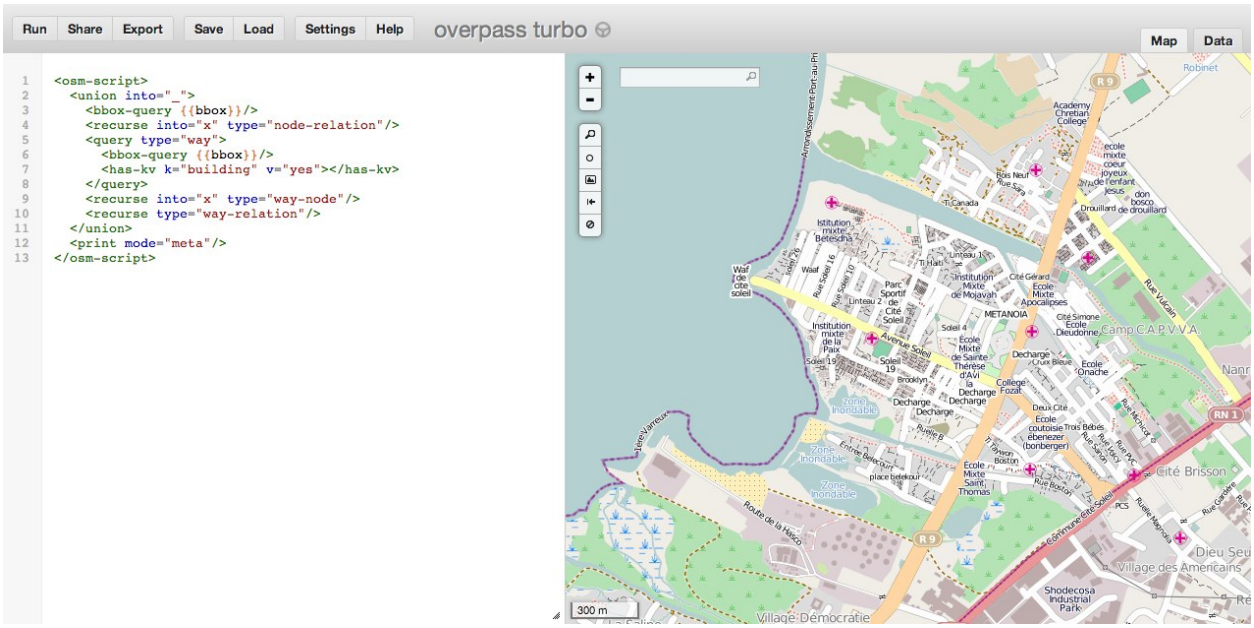
In this example, we will be extracting building footprint data around Port au Prince in Haiti. To do this we will use an interactive tool that makes it easy construct a Query against the Overpass API. Point your browser at http://overpass-turbo.eu/ and use the search tools to zoom into Port Au Prince and Cite Soleil specifically.

You will need to cut and paste the query specified below to get all of the appropriate data under the bbox:

```
<osm-script>
  <union into="_">
    <bbox-query {{bbox}}/>
    <recurse into="x" type="node-relation"/>
    <query type="way">
      <bbox-query {{bbox}}/>
      <has-kv k="building" v="yes"></has-kv>
    </query>
    <recurse into="x" type="way-node"/>
    <recurse type="way-relation"/>
  </union>
  <print mode="meta"/>
</osm-script>
```

This should look like the following.

When you have the bbox and query set correctly, click the “Export” button on the menu to bring up the export menu,



and then click the API interpreter link to download the OSM data base on the query you have specified.

This will download a file named 'interpreter' on your file system. You will probably want to rename it something else more specific. You can do that by issuing the following command in the directory where it was downloaded:

```
$ mv interpreter cite_soleil_buildings.osm
```

Note: You can also rename the file in your Operating Systems File managmenet tool (Windows Explorer, Finder etc).

Now that we have osm data on our filesystem, we will need to convert it into a format suitable for uploading into your GeoNode. There are many ways to accomplish this, but for purposes of this example, we will use an OSM QGIS plugin that makes if fairly easy. Please consult the wiki page that explains how to install this plugin and make sure it is installed in your QGIS instance. Once its installed, you can use the Web Menu to load your file.

This will bring up a dialog box that you can use to find and convert the osm file we downloaded.

When the process has completed, you will see your layers in the Layer Tree in QGIS.

Since we are only interested in the polygons, we can turn the other 2 layers off in the Layer Tree.

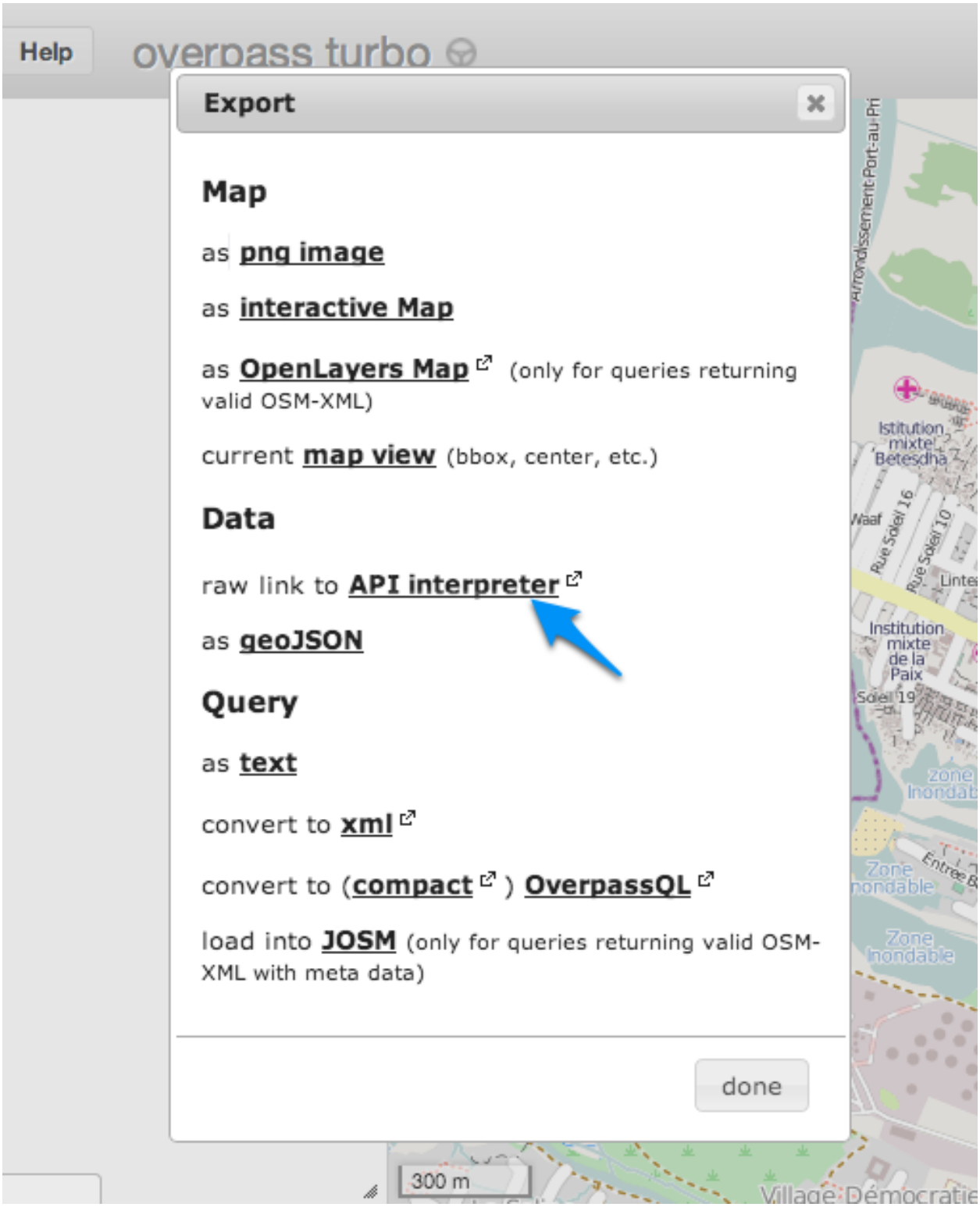
The next step is to use QGIS to convert this layer into a Shapefile so we can upload it into GeoNode. To do this, select the layer in the Layer tree, right click and then select the Save As option.

This will bring up the Save Vector Layer as Dialog.

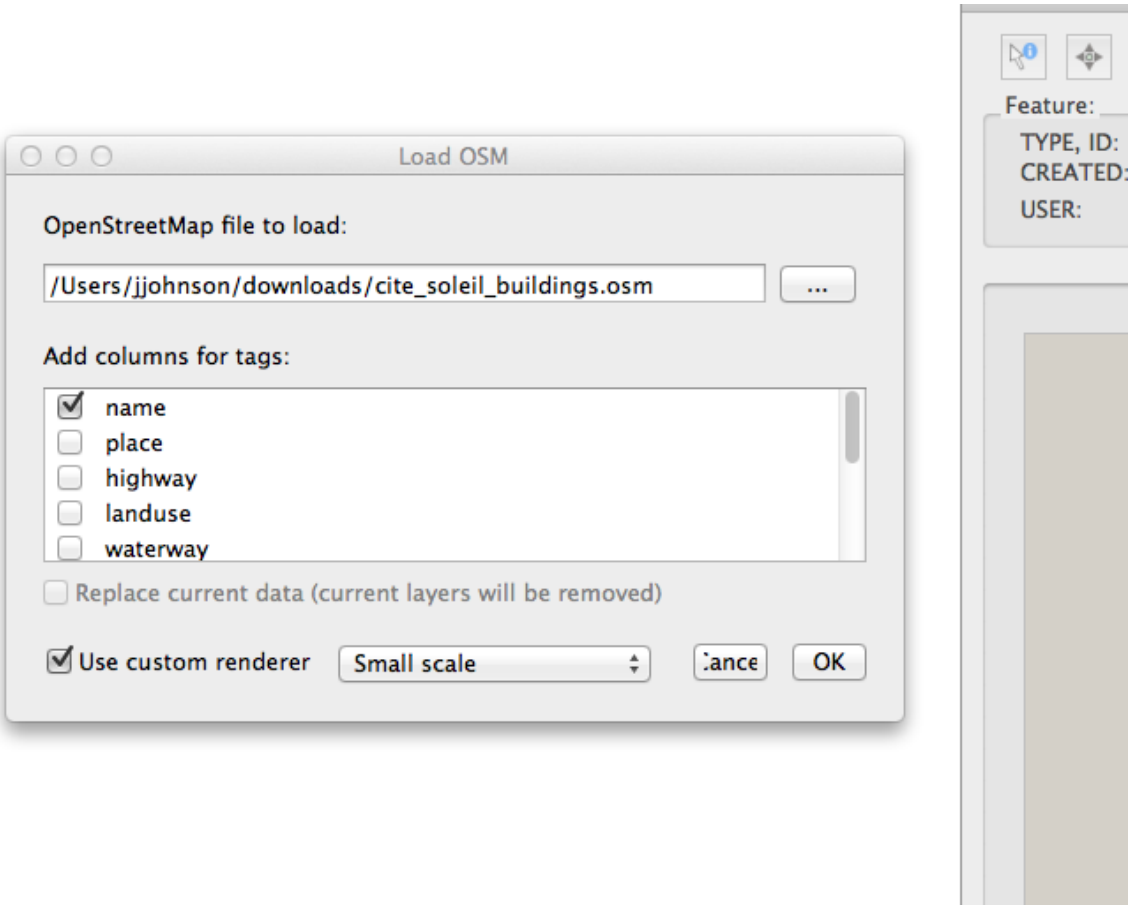
Specify where on disk you want your file saved, and hit Save then OK.

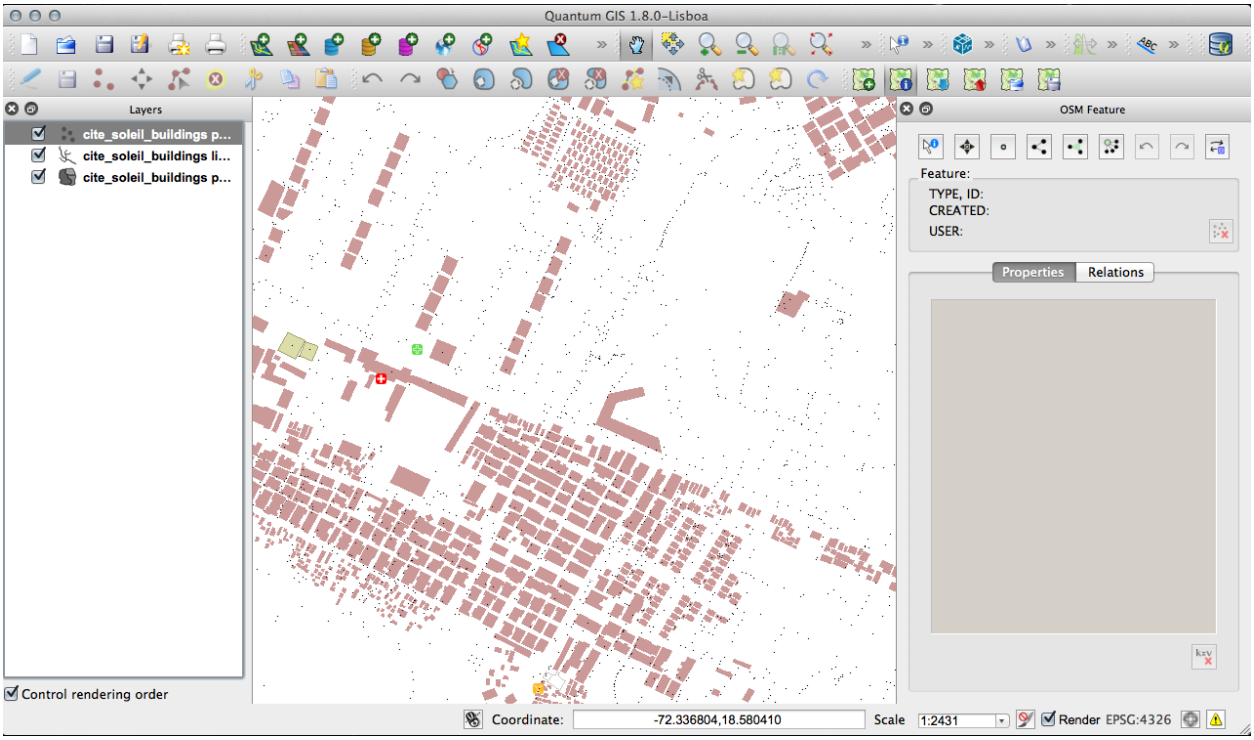
You now have a shapefile of the data you extracted from OSM that you can use to load into GeoNode. Use the GeoNode Layer Upload form to load the Shapefile parts into your GeoNode, and optionally edit the metadata and then you can view your layer in the Layer Info page in your geonode.

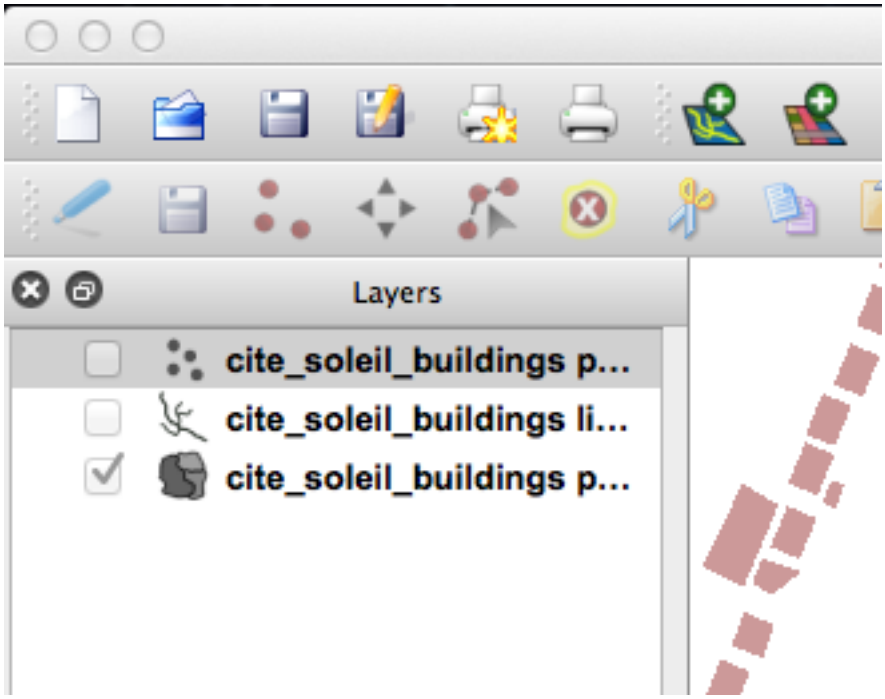
Note: You may want to switch to an imagery layer in order to more easily see the buildings on the OSM background.

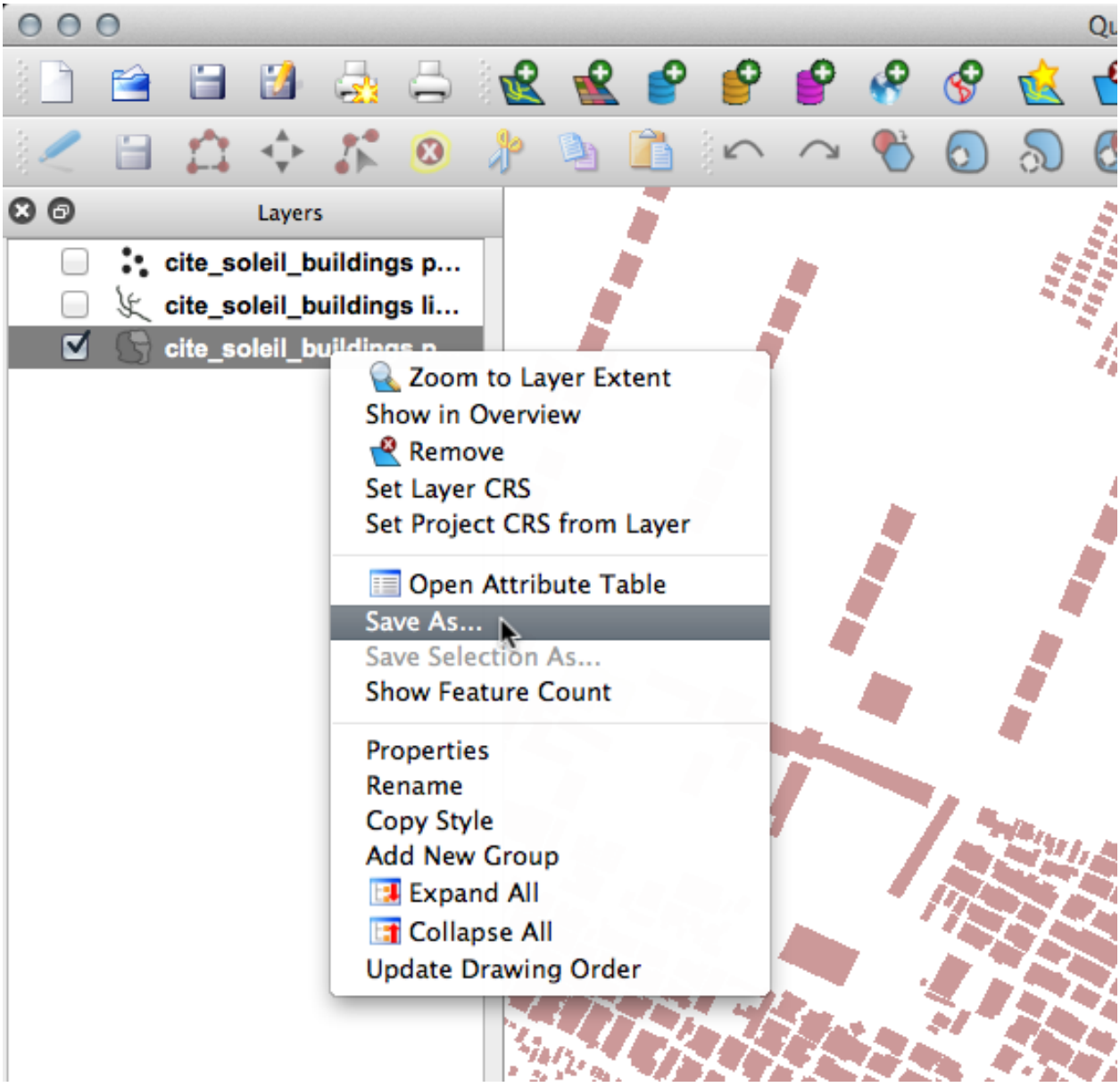

Help
overpass turbo
Export
Map
as png image
as interactive Map
as OpenLayers Map (only for queries returning valid OSM-XML)
current map view (bbox, center, etc.)
Data
raw link to API interpreter
as geoJSON
Query
as text
convert to xml
convert to (compact) OverpassQL
load into JOSM (only for queries returning valid OSM-XML with meta data)
done
300 m

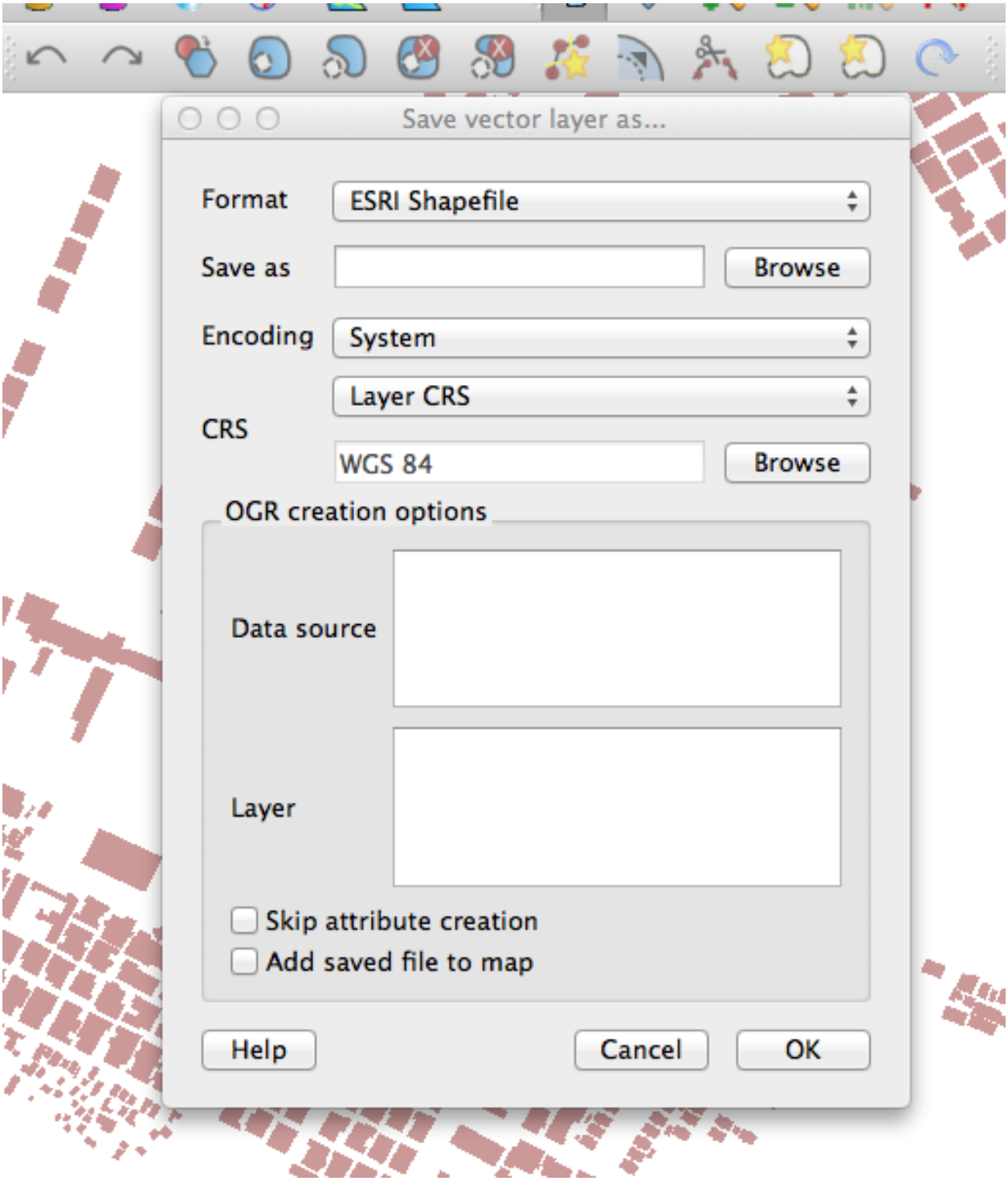

Settings
Plugins
Vector
Raster
Database
Web
Window
Help
Feb 13
OpenStreetMap
Load OSM from file
Show/Hide OSM Feature Manag
Download OSM data
Upload OSM data
Import data from a layer
Save OSM to file
Feature:
TYPE, ID:
CREATED:
USER:

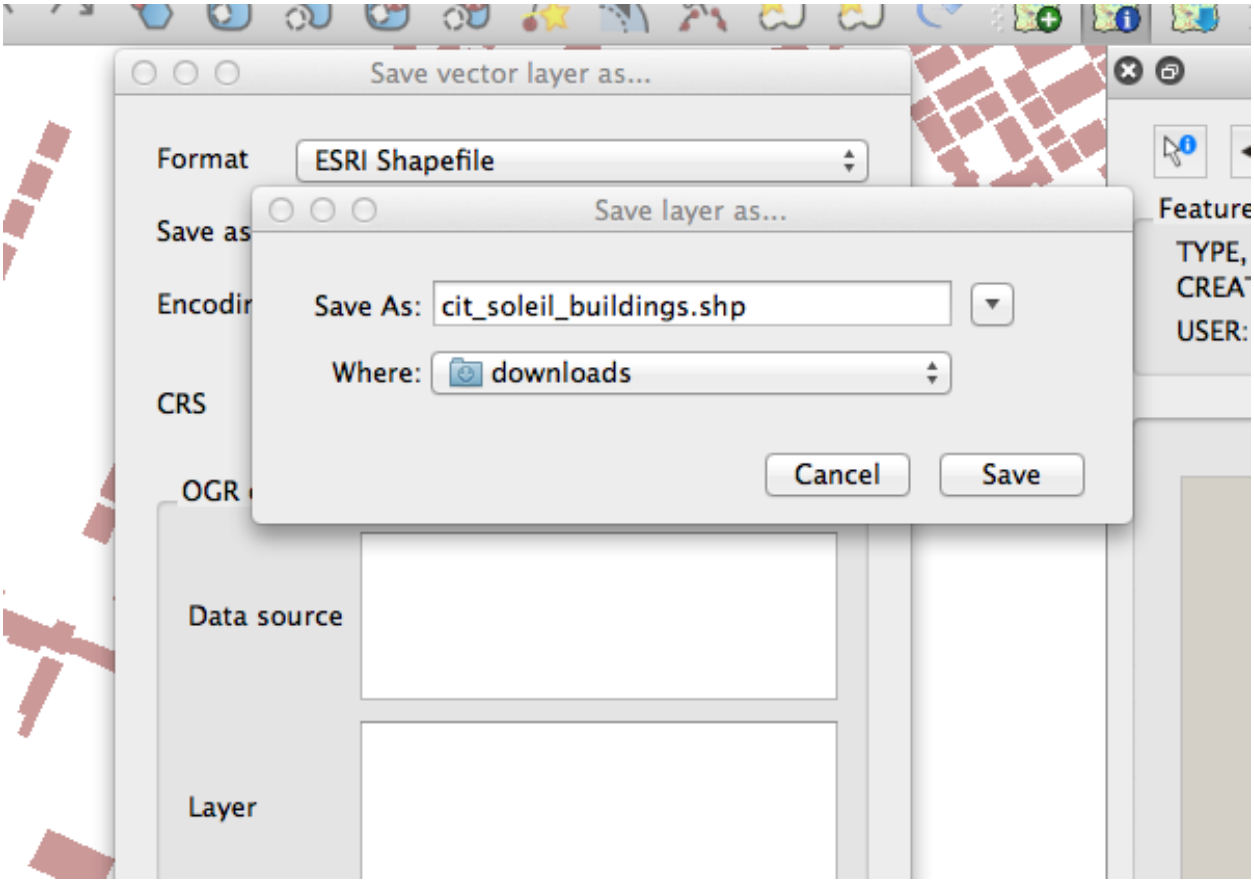

Load OSM
OpenStreetMap file to load:
/Users/jjohnson/downloads/cite_soleil_buildings.osm
...
Add columns for tags:
name
place
highway
landuse
waterway
Replace current data (current layers will be removed)
Use custom renderer
Small scale
OK
Feature:
TYPE, ID:
USER:

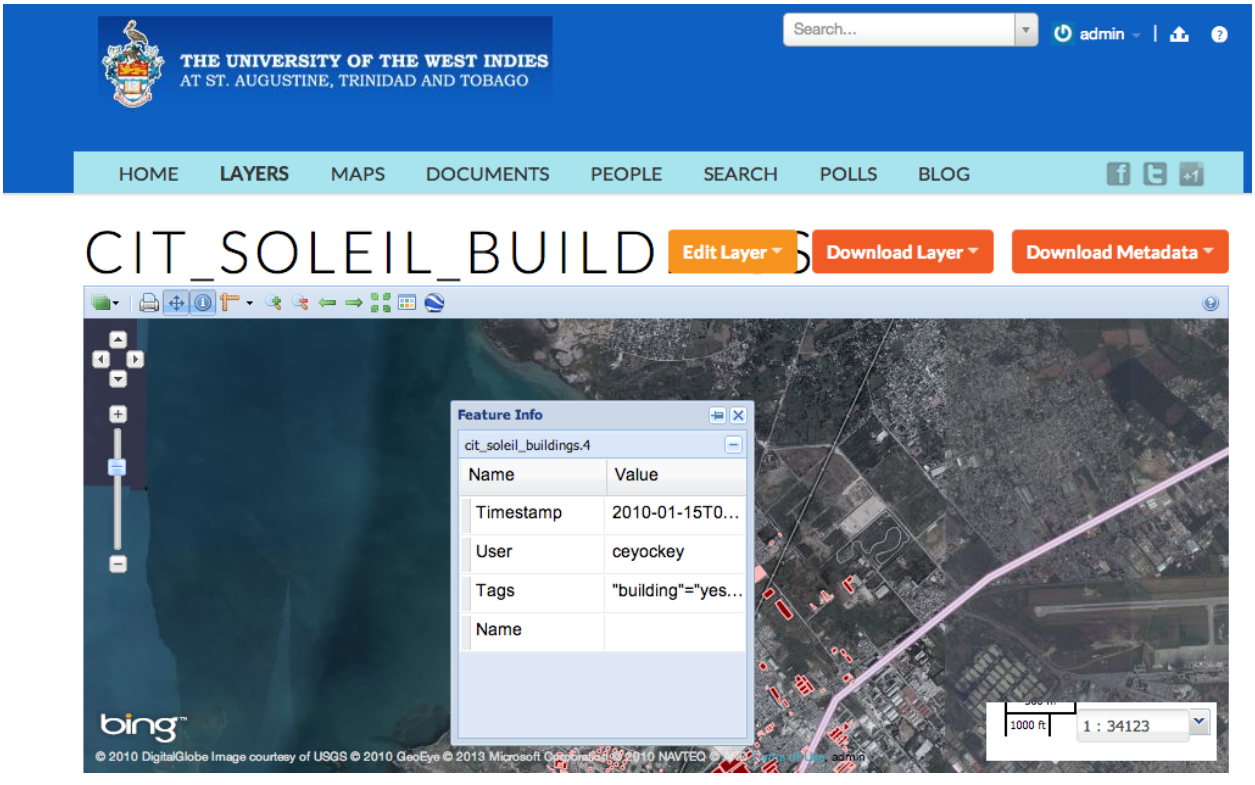

Quantum GIS 1.8.0-Lisboa
Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
cite_soleil_buildings p...
Control rendering order
OSM Feature
Feature:
TYPE, ID:
CREATED:
USER:
Properties
Relations
Coordinate:
-72.336804,18.580410
Scale
1:2431
Render
EPSG:4326


Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
cite_soleil_buildings p...


Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
Zoom to Layer Extent
Show in Overview
Remove
Set Layer CRS
Set Project CRS from Layer
Open Attribute Table
Save As...
Save Selection As...
Show Feature Count
Properties
Rename
Copy Style
Add New Group
Expand All
Collapse All
Update Drawing Order


Save vector layer as...
Format
ESRI Shapefile
Save as
Browse
Encoding
System
Layer CRS
CRS
WGS 84
Browse
OGR creation options
Data source
Layer
Skip attribute creation
Add saved file to map
Help
Cancel
OK


Save vector layer as...
Format
ESRI Shapefile
Save as
Encodir
CRS
OGR
Data source
Layer
Save layer as...
Save As:
cit_soleil_buildings.shp
Where:
downloads
Cancel
Save
Feature
TYPE,
CREA
USER:


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
POLLS
BLOG
CIT_SOLEIL_BUILD
Edit Layer
Download Layer
Download Metadata
Feature Info
cit_soleil_buildings.4
Name
Value
Timestamp
2010-01-15T0...
User
ceyockey
Tags
"building"="yes...
Name
bing
1000 ft
1 : 34123

CHAPTER

FIVE

GEONODE DEBUGGING TECHNIQUES

GeoNode can be difficult to debug as there are several different components involved:

- Browser - includes HTML/CSS issues, JavaScript, etc.
- Django - GeoNode HTML views and web APIs
- GeoServer - Core Wxx services and Platform REST APIs

When attempting to diagnose a specific problem, often the order of investigation mirrors the order above - that is, start with the client: Is this a bug in code running on the browser. If not, step to the next level: the Django responses to client requests. Often this is possible via the browser using the correct tools. Many requests require Django communications with GeoServer. This is the next stage of investigation if a specific bug does not appear to originate in Django or the client.

The following section covers techniques to help diagnose and debug errors.

5.1 Debugging GeoNode in the Browser

This section covers some techniques for debugging browser and Django related response bugs using the Firefox web browser extension named Firebug. The concepts covered apply to other browser's tools but may vary in terminology.

Another Firefox extension worth noting is 'jsonview'. This extension supports formatted viewing of JSON responses and integrates well with Firebug.

References:

- https://getfirebug.com/faq/
- http://jsonview.com/

5.1.1 Net Tab

The net tab allows viewing all of the network traffic from the browser. The subtabs (like the selected "Images" tab) allow filtering by the type of traffic.

In this screen-shot, the mouse hover displays the image content and the full URL requested. One can right-click to copy-paste the URL or view in a separate tab. This is useful for obtaining test URLs. The grayed out entries show that the resource was cached via conditional-get (the 304 not modified). Other very useful advanced information includes the size of the response and the loading indicator graphics on the right. At the bottom, note the total size and timing information.

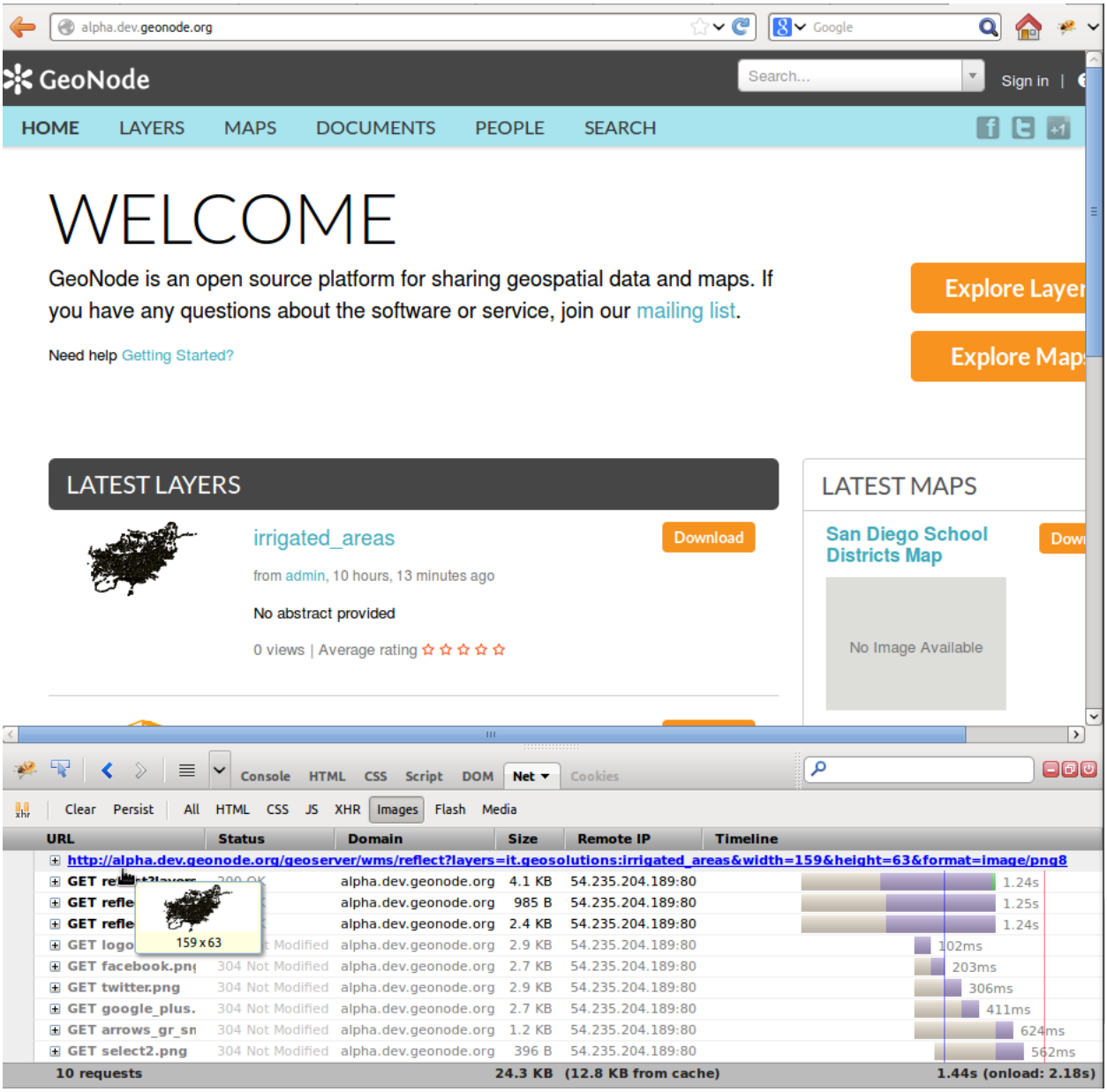


Figure 5.1: Firebug Net Tab

Net Tab Exercises

1. Go to layers/maps/search pages and look at the various requests. Note

   the XHR subtab. Look at the various request specific tabs: headers, params, etc.

1. Use the 'disable browser cache' option and see how it affects page

   loads. Discuss advantages/challenges of caching.

5.1.2 DOM Tab

The DOM tab displays all of the top-level window objects. By drilling down, this can be a useful way to find out what's going on in a page.

In this example, the mouse is hovering over the `app` object. Note the high level view of objects and their fields. The console tab allows interacting with the objects.

DOM Tab Exercises

1. Drill down in the DOM tab.
2. Use the console to interactively exercise jquery.
3. Use the console to interact with the app/map or other page objects

5.1.3 Script Tab

The script tab allows viewing scripts and debugging.

The screen-shot displays a breakpoint set at line 3, the current code is stopped at line 8 and the mouse hover is displaying the value of the variable 'class_list'. On the right, the 'Watch' tab displays the various variables and scopes and offers a drill down view similar to the DOM view. The stack tab displays the execution stack context outside the current frame.

Script Tab Exercises

1. Step through some code
2. Look at various features: variables, scopes, DOM drill-down

5.1.4 HTML Tab

The HTML tag allows viewing and drilling down into the DOM. This is an incredibly useful feature when doing CSS or HTML work.

The screen-shot displays a search result 'article' element highlighted with padding and margin in yellow and purple. The DOM structure is displayed on the left and the right panel displays the specific style rules while the computed tab displays the effective style rules. The layout tab displays rulers and property values while the DOM tab displays a debug/DOM-like view of the actual object's properties.

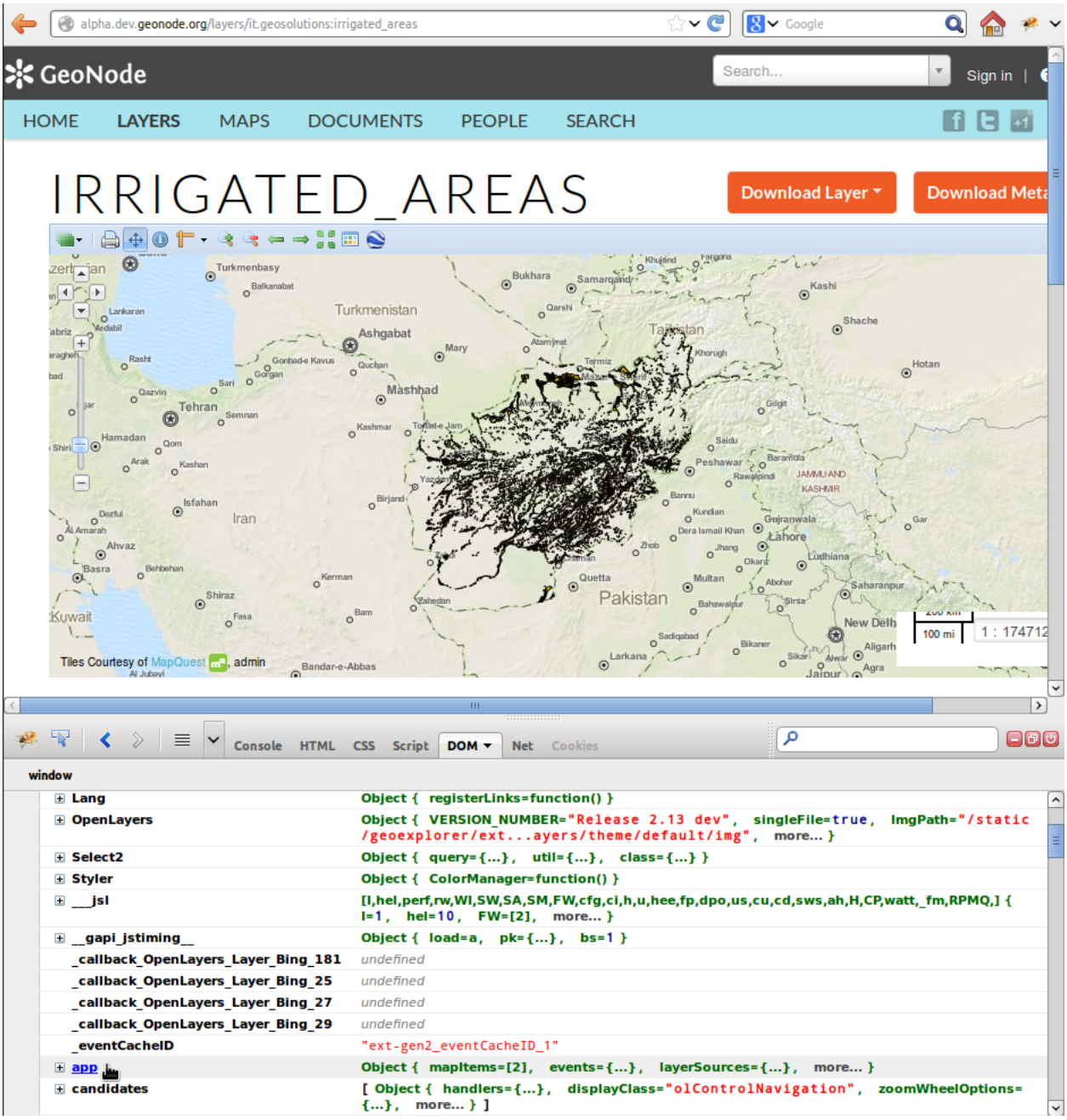
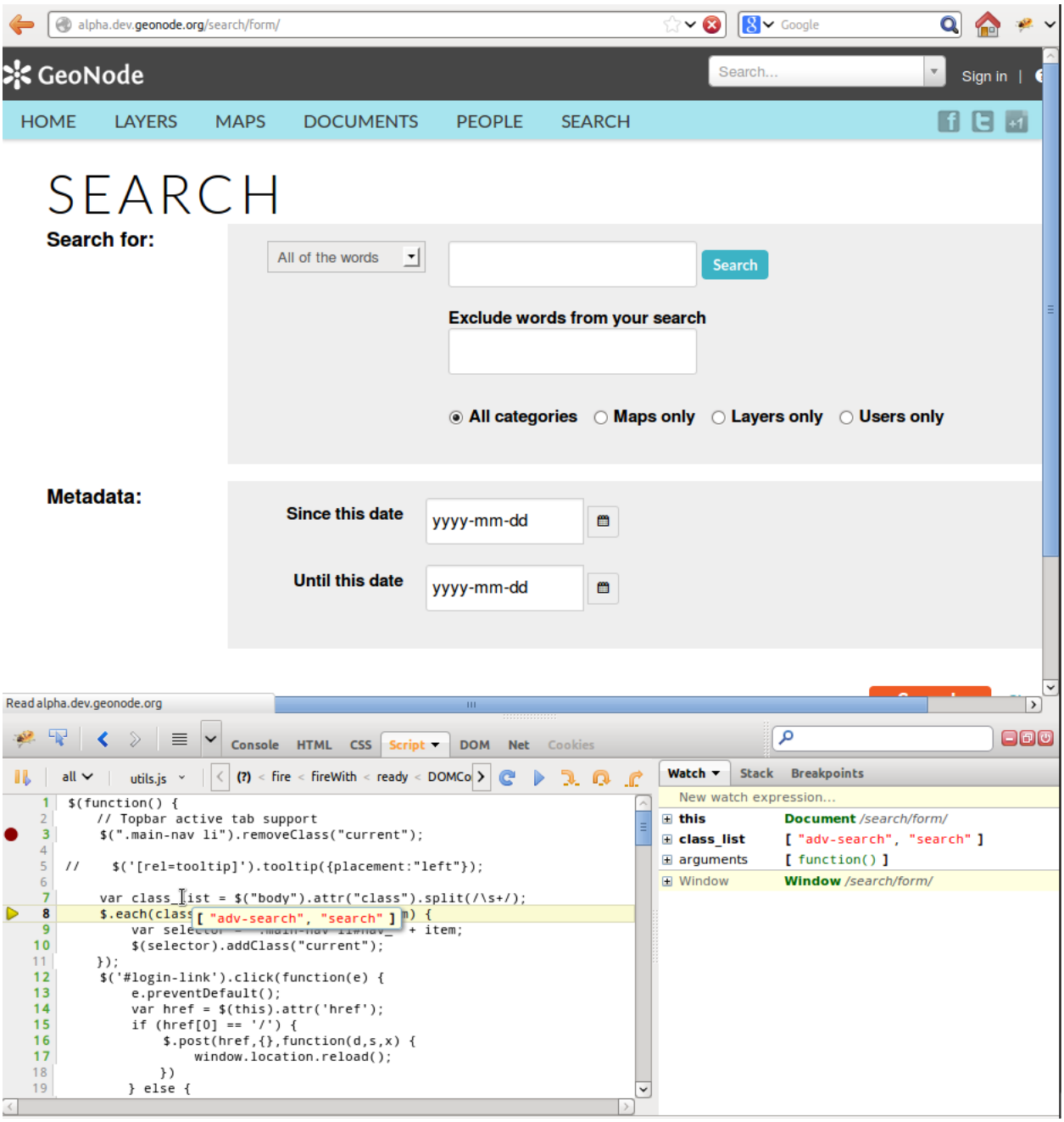
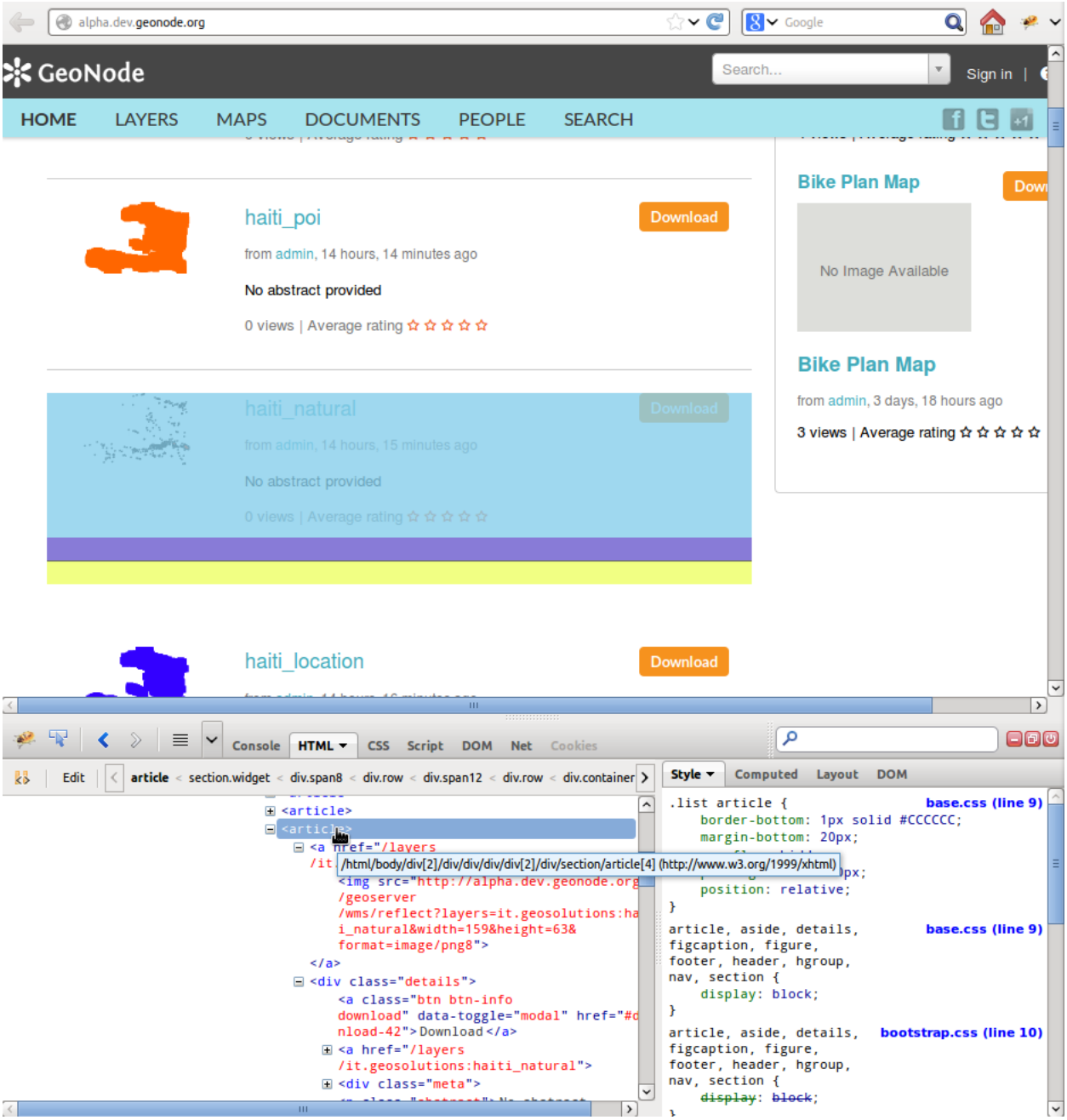


Figure 5.2: Firebug DOM Tab


alpha.dev.geonode.org/search/form/
GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
SEARCH
Search for:
All of the words
Search
Exclude words from your search
All categories
Maps only
Layers only
Users only
Metadata:
Since this date
yyyy-mm-dd
Until this date
yyyy-mm-dd
Console
HTML
CSS
Script
DOM
Net
Cookies
Watch
Stack
Breakpoints
New watch expression...
this
Document /search/form/
class_list
["adv-search", "search"]
arguments
[function()]
Window
Window /search/form/
["adv-search", "search"]


alpha.dev.geonode.org
GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
haiti_poi
Download
from admin, 14 hours, 14 minutes ago
No abstract provided
0 views | Average rating
Bike Plan Map
No Image Available
Bike Plan Map
from admin, 3 days, 18 hours ago
3 views | Average rating
haiti_natural
Download
from admin, 14 hours, 15 minutes ago
No abstract provided
0 views | Average rating
haiti_location
Download
Console
HTML
CSS
Script
DOM
Net
Cookies
Edit
article
section.widget
div.span8
div.row
div.span12
div.row
div.container
Style
Computed
Layout
DOM
<article>
<a href="/layers
/html/body/div[2]/div/div/div/div[2]/div/section/article[4] (http://www.w3.org/1999/xhtml)
<img src="http://alpha.dev.geonode.org
/geoserver
/wms/reflect?layers=it.geosolutions:ha
i_natural&width=159&height=63&
format=image/png8">

<a class="btn btn-info
download" data-toggle="modal" href="#d
nload-42">Download
<a href="/layers
/it.geosolutions:haiti_natural">

.list article {
base.css (line 9)
border-bottom: 1px solid #CCCCCC;
margin-bottom: 20px;
position: relative;
}
article, aside, details,
figcaption, figure,
footer, header, hgroup,
nav, section {
base.css (line 9)
display: block;
}
article, aside, details,
figcaption, figure,
footer, header, hgroup,
nav, section {
bootstrap.css (line 10)
display: block;

HTML Tab Exercises

1. Identify elements, look at the tabs on the right
2. Change styles, add new rules and styles
3. Edit existing HTML elements via the raw and tree-view

5.2 Debugging GeoNode's Python Components

5.2.1 Logging

References:

- http://docs.python.org/2/library/logging.html
- https://docs.djangoproject.com/en/1.4/topics/logging/

Logging is controlled by the contents of the logging data structure defined in the `settings.py`. The default settings distributed with GeoNode are configured to only log errors. During development, it's a good idea to override the logging data structure with something a bit more verbose.

Output

In production, logging output will go into the apache error log. This is located in `/var/log/apache2/error.log`. During development, logging output will, by default, go to standard error.

Configuring

- Ensure the 'console' handler is at the appropriate level. It will ignore log messages below the set level.
- Ensure the specific logger you'd like to use is set at the correct level.
- If attempting to log SQL, ensure `DEBUG=True` in your `local_settings.py`.

Debugging SQL

- To trace all SQL in django, configure the `django.db.backends` logger to `DEBUG`
- To examine a specific query object, you can use the `query` field: str(Layer.objects.all().query)
- You can gather more information by using `django.db.connection.queries`. When `DEBUG` is enabled, query SQL and timing information is stored in this list.

Hints

- Don't use print statements. They are easy to use in development mode but in production they will cause failure.
- Take advantage of python. Instead of:

```
logging.info('some var ' + x + ' is not = ' + y)
```

  Use:

```
logging.info('some var %s is not = %s', x, y)
```

Excercises:

1. Enable logging of all SQL statements. Visit some pages and view the logging output.
2. Using the python shell, use the `queries` object to demonstrate the results of specific queries.

5.2.2 PDB

Reference:

- http://docs.python.org/2/library/pdb.html

For the adventurous, `pdb` allows for an interactive debugging session. This is only possible when running in a shell via `manage.py runserver` or `paver runserver`.

To set a breakpoint, insert the following code before the code to debug.

..code-block:: python

import pdb; pdb.set_strace()

When execution reaches this statement, the debugger will activate. The commands are noted in the link above. In addition to those debugger specific commands, general python statements are supported. For example, typing the name of a variable in scope will yield the value via string coersion.

5.3 Debugging GeoServer

Resources:

- http://docs.geoserver.org/stable/en/user/advanced/logging.html
- http://docs.geoserver.org/stable/en/user/production/troubleshooting.html

This section does not attempt to cover developer-level debugging in GeoServer as this is a much larger topic involving many more tools. The goal here is to provide 'black-box' techniques to help resolve and report problems.

5.3.1 Logging

GeoServer logging, while sometimes containing too much information, is the best way to start diagnosing an issue in GeoNode once the other. To create a proper error report for use in requesting support, providing any contextual logging information is critical.

When using a standard geoserver installation, the GeoServer logs are located at `/usr/share/geoserver/data/logs/geoserver.log`. The properties files that control the varying rules are also located here.

Exercises

1. Switch logging levels for various loggers.
2. Look at the different logging profiles and discuss the loggers and levels.
3. Learn how to read stacktraces, nested or otherwise.

5.3.2 Advanced Troubleshooting

JVM diagnostics and advanced troubleshooting techniques are covered in the GeoServer documents linked to above. When providing information for a bug report, these can be helpful but in-depth Java knowledge is required to fully comprehend the output from some of these tools.

Exercises

1. Look at jstack output

5.3.3 Using Django to Help Debug

The gsconfig library provides a rich interface to interacting with GeoServer's REST API. This allows high-level functions as well as viewing raw REST responses.

```
cat = Layer.objects.gs_catalog
cat.get_layers() # list of gsconfig layer objects
# OR, for a specific layer
lyr = Layer.objects.get(id=1)
lyr.resource # specfic gsconfig layer object
lyr.resource.fetch() # get the XML from REST
lyr.resource.dom # reference to the parsed XML
from xml.etree.ElementTree import tostring
tostring(lyr.resource.dom)
```

CHAPTER

SIX

GEONODE APIS

CHAPTER

SEVEN

SETTING UP A GEONODE DEVELOPMENT ENVIRONMENT

This module will lead you through the steps necessary to install a GeoNode development environment.

CHAPTER

EIGHT

GEONODE'S DEVELOPMENT PROCESS