GeoNode Developers Workshop Documentation

Release 2.0

GeoNode

February 06, 2013

CONTENTS

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

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.

1.2.2 Web Standards

HTML

CSS

REST

1.3 GeoNode Architecture

1.3.1 Django Architecture

- MVC
- WSGI

1.3.2 GeoNode and GeoServer

- Configuration via the REST API
- Authentication and Authorization

1.3.3 GeoNode and PostgreSQL/PostGIS

- Configuration and Application Information
- Vector Data Layer Storage

1.3.4 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.

1.3.5 Javascript in GeoNode

- GeoExplorer
- jQuery Functionality

1.4 Development References

1.4.1 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/

1.4.2 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

- **ssh is the network protocol used to connect to a remote server where you run your GeoNode instance whether on your ow**
  - 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. Y**
  - 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 depende**
  - 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 e**
  - 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 t**
  - 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**
  - 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 brea**
  - 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 def

- 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

Management Commands

Django Admin Interface

Template Tags

2.2 GeoNode's Core Modules

2.2.1 layers

2.2.2 maps

2.2.3 security

2.2.4 search

2.2.5 catalogue

2.2.6 geoserver

2.2.7 geoexplorer

2.2.8 Static Site

Templates

CSS

Javascript

CHAPTER

THREE

CUSTOMIZED GEONODE PROJECTS

This module will teach you about how to setup your own GeoNode based project and how to customize your project by changing the theme, adding additional modules and how to integrate your project with other systems. When complete, you should understand how Downstream GeoNode projects work, and how to setup your own for your own use cases and needs.

3.1 Setting up your GeoNode Project

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

3.1.1 Setup Steps

If you are working remotely, You should first connect to the machine that has your GeoNode installed on it. 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 setup 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.

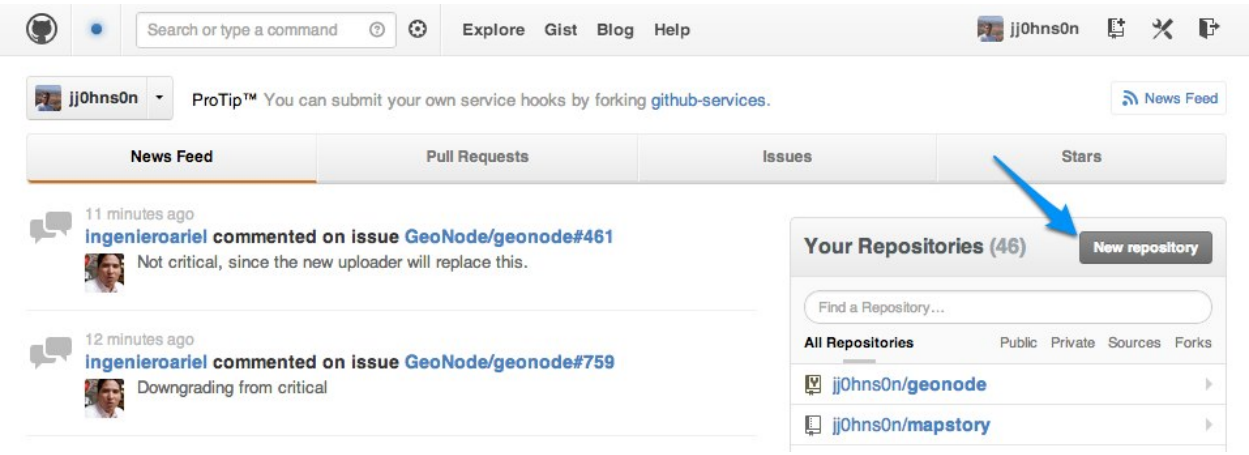
   http://localhost:8000

3.1.2 Source Code Revision Control

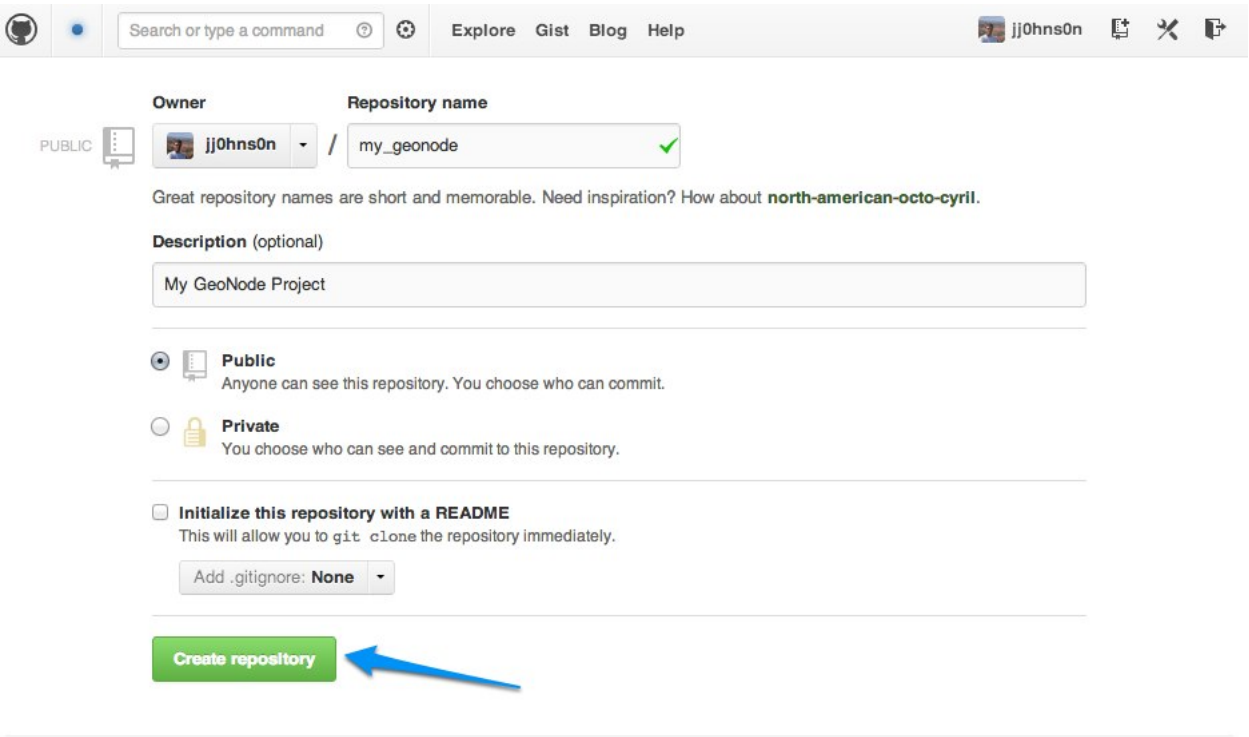
It is recommended that you immediately put your new project under source code revision control. The GeoNode development team uses Git and GitHub purpose and recommends that you do the same. If you do not already have a GitHub account, it is recommended that you set one up. A full review of Git and distributed 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:

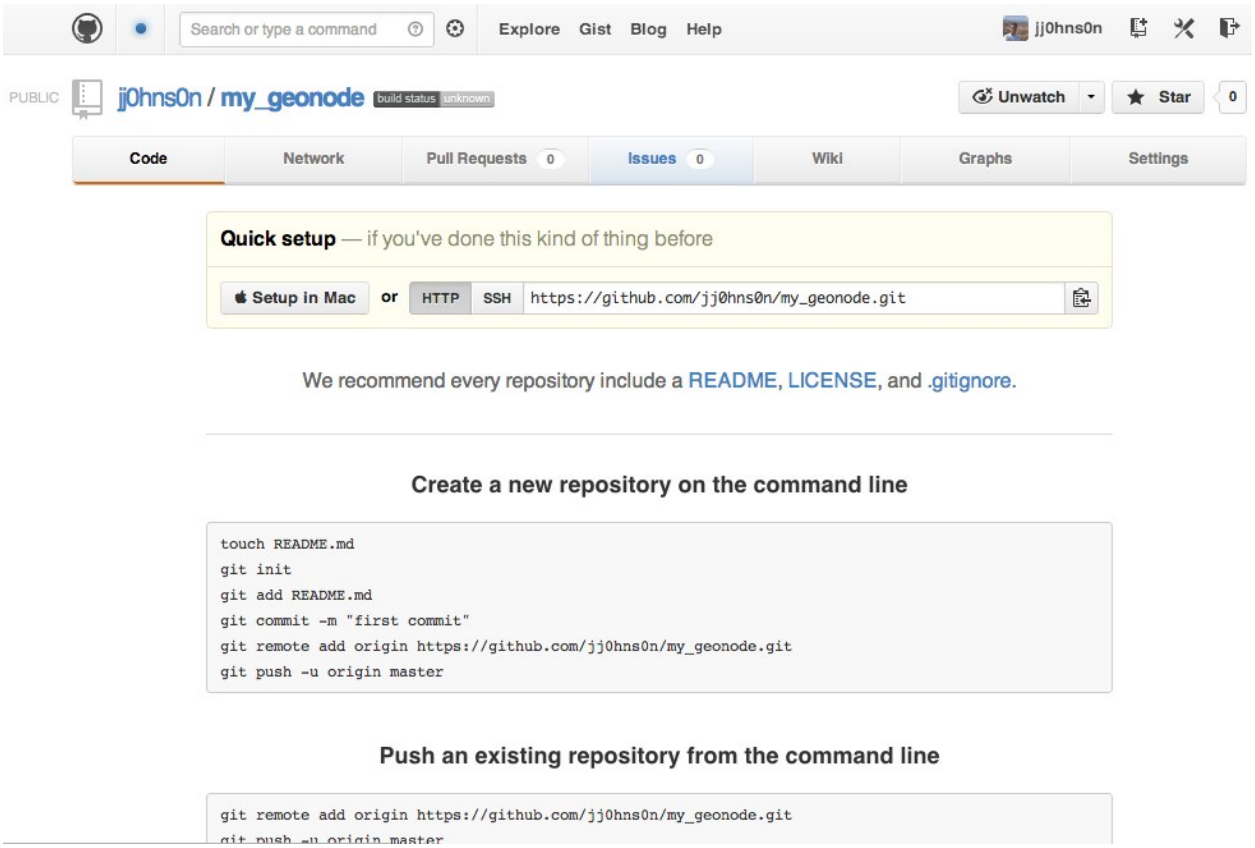
   You should use the GitHub user interface to create a new repository for your new project.



Creating a new GitHub Repository From GitHub's Homepage



Specifying new GitHub Repository Parameters



Your new Empty GitHub Repository

2. Initialize your own 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
   ```

3.1.3 Deploying your GeoNode Project

Now that your own project is setup, 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

3.1.4 Staying in Sync with Mainline GeoNode

One of the primary reasons that we setup your own GeoNode project using this method is so that you can stay in sync with mainline geonode as the core GeoNode development team makes new releases. Your own project should not be adversely affected by these upstream 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
   ```

   Visit http://localhost:8000/

3.2 Theming your GeoNode Project

3.2.1 Logos and Graphics

3.2.2 Templates and Static Pages

3.2.3 Cascading Style Sheets

3.2.4 Other Theming Options

Bootswatch

3.3 Adding Additional Modules to your GeoNode Project

3.3.1 Intro to Django Pluggables

3.3.2 Adding your own Django Module

3.3.3 Adding a 3rd Party Blog Module

3.3.4 Adding Other Modules

3.4 Integrating your Project with other Systems

3.4.1 Third Party Integration Overview

3.4.2 OGC Services

3.4.3 Google Earth

3.4.4 qGIS

3.4.5 OpenStreetMap

3.4.6 Wordpress

3.4.7 MapBox

CHAPTER

FOUR

SETTING UP A GEONODE DEVELOPMENT ENVIRONMENT

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

CHAPTER

FIVE

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.

5.1 GeoServer Data Configuration

5.2 Using ogr2ogr to load data into GeoNode

5.3 Loading OSM Data into GeoNode

CHAPTER

SIX

GEONODE APIS

CHAPTER

SEVEN

GEONODE DEBUGGING TECHNIQUES

CHAPTER

EIGHT

GEONODE'S DEVELOPMENT PROCESS

8.1 GeoNode's Issue Tracking System

8.2 Testing in GeoNode

8.2.1 Unit Tests

8.2.2 Integration Tests

8.2.3 Javascript Tests

8.3 GeoNode's Patch Review Process

8.4 GeoNode Improvement Proposals

8.5 GeoNode's Roadmap Process

8.6 Development Resources

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.

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

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

GeoNode APIs Learn about the APIs GeoNode leverages and provides.

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

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