Soil Data Portal

Application Config Documentation

AAFC - Science and Technology

Winter 2017

Author: Andrew Roberts

Overview

The purpose of this documentation is to support the maintenance, transport, and deployment of the Agriculture Canada - Soil Data Portal web map application, developed by Andrew Roberts, Max Guo, and Anne Hagerman during the 2017 co-op work term. This document will provide an overview of environment settings, dependencies, data structure, and important data migration steps to ensure the stable reproduction and implementation of the application.

This document will not provide step by step instructions. For a detailed walkthrough of the development environment setup and installation, please refer to setup and configuration.

For detailed documentation about the Plant Disease Model, a python based modeler supported by the application, please refer to this document.

*All support documentation can be found at ulysses.agr.gc.ca/docs

Scope

The Soil Data Portal was developed to provide in-house access to Canadian soil attribute information, stored as Canada wide coverages in .tif format on a local server database. Utilizing current WMS (Web Map Service) and WCS (Web Coverage Service) technologies, the web based portal establishes the database as an authoritative source for soil information, facilitating cohesion and consistency in future research and studies. After selecting the soil attribute layers they require, the portal allows users to select an area of interest by three means; 1) drawing tools provided on the map, 2) defining a bounding box and projection, and 3) uploading a shapefile and selecting specific features. The extent of the users area of interest is then used to subset their selected layers, after which a python script is run to extract metadata, zip them up, then offer to download.

The development of this application completes two objectives. First, it provides wide and simple access to in-house information, establishing the database as a single, authoritative source. And second, it provides an opensource framework on which future applications can be developed for any department or organization whose day to day operations or project goals might be facilitated by customized web processing technologies.

Team - HR

Lead Supervision - Xiaoyuan Geng, Head Soil Scientist at Landscape Analysis and Applications, AAFC-STB
xiaoguon.geng@agr.gc.ca

Tech Guru - Tim Martin, Sustainability Metrics, AA
martin.tim@agr.gc.ca

Developer - Andrew Roberts, Algonquin College, co-op
andrew.ed.roberts@gmail.com

Developer - Max Guo, Lethbridge University, co-op
maxlovesbbq@gmail.com

Developer - Anne Hagerman, Algonquin College, co-op
annehargerman@hotmail.com

Abbreviations

AAFC - Agriculture and Agri-Food Canada

AOI - Area of Interest

OL - OpenLayers

OSM - Open Street Map

WCS - Web Coverage Service

WMS - Web Mapping Service

WPS - Web Processing Service

Architecture

The basic architecture of the application is client -> workhorse (application server) -> data-server (see Figure 1 below). The client interacts with the application on the front end - html, css, and javascript handle all interactions including drawing, shapefile uploads, and feature selection. All front end application code sits on the application server.

The application server (for the Soil Data Portal at AAFC this is Daedalus) is setup to handle most of the data processing. Any dynamic content relevant to the data being served is pulled from the data server to the application server via REST interface. This is then displayed for the client. For example, all soil attribute layers available at AAFC on the data server are dynamically accessed anytime the client visits the page. Any new or deleted files will be updated automatically on the web portal.

PHP processing is done on the application server. The user's selected AOI is sent to a PHP script via AJAX. From here, the application server connects to the data server and requests information based on user specs. These requests will result in a response from the data server, usually in the form of the requested data (if something went wrong, error messages are received and handled). This data is saved and on success, made available to the user in the form of a download button. Any python processing is done here, prior to the download option for the client. The Soil Data Portal will create a metadata file with statistics for each subset the client has requested. This file is sent to the client along with the requested .tifs as a .zip.

The data-server (for the Soil Data Portal at AAFC this is Ulysses) houses the data and serves is through GeoServer as WCS services. GeoServer receives the user's AOI, subsets the specified layer, and returns the subset TIFF.

Soil Data Portal Architecture

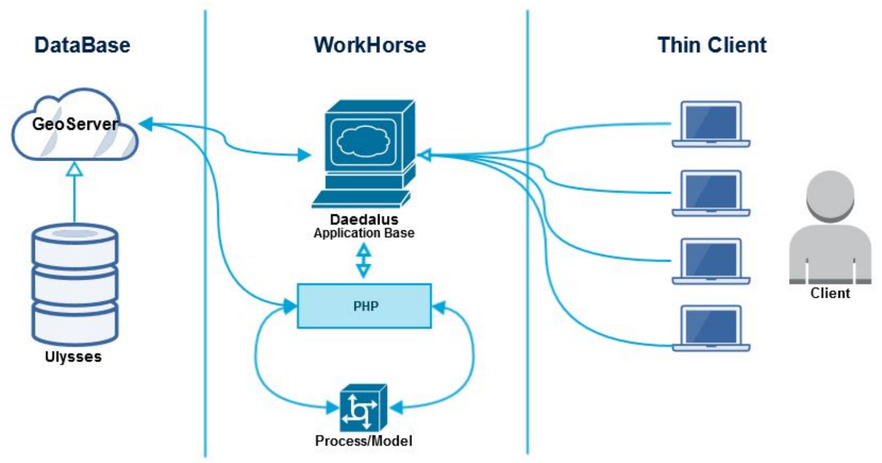

Figure 1. Three tiered platform

Environment

(See appendix 1 for complete list) The application was developed on Ubuntu v16.04 LTS running Apache Web Server v2.4.18. Tomcat v9.0.0 sat on Apache to handle Java requirements. GeoServer v2.9.1, an opensource Java based GeoSpatial data server, was used to serve up the soil data through WCS. PHP v7.0.8 handled server side scripting. Python 3 was used for image processing and plant disease modeling. OpenLayers v3.19.1 handled the web map interaction. Javascript, jQuery 3.1.1 and CSS framework Semantic UI v2.2.9 handled the application GUI.

In house (AAFC) it runs on Ubuntu Server v16.04. All other environment specs remain the same.

Dependencies

The application assumes a strict folder structure. All relative paths and download links require placement of second level directories (one down from src, as found on github). Any changes to folder structure will require modification of code.

Apache/Apache-Tomcat

Significant configuration file modifications must be made in order to effectively run Tomcat on Apache. The mod.jk package must be installed, and workers.properties config file must include workers for java enabled pages. Detailed steps for this can be found in section 5 and 6 of the Setup_and_Config document.

GeoServer

- **Cross Origin Request**

  To allow cross origin resource sharing and avoid "Same Origin Policy" errors when requesting server pages through OpenLayers, configurations are made to the Geoserver web.xml config file to include "Allow Cross Origin" headers. See section 7.7 of Setup_and_Config document.

- **Heap Memory Increase**

  To avoid potential memory shortage and "Out Of Memory" errors, modifications are made to the Tomcat context.xml file and a custom environment settings file is created. See section 7.9 of Setup_and_Config document.

- **Enabling REST Services for GET Request**

  In order to access workspaces, coverage stores, and layer groups, rest.properties config file is modified to include anonymous GET requests to GeoServer's REST interface. See section 7.11 of Setup_and_Config document.

- **GeoServer Data Structure**

  This section assumes a good understanding of GeoServer. For more information about the service and terms used here, please check out the GeoServer support documentation here.

  The application assumes a strict standard for loading new data onto GeoServer for the purpose of inclusion with the Soil Data Portal. All soil attribute files represent a Canada wide coverage and are stored under a single workspace titled 'Canada'. All files are assigned a layer group, each group a single soil attribute at varying layer depths. The application uses the REST interface to fetch specific layers from all layer-groups found within this workspace. Layer-groups, file descriptions, and file titles are formatted to create consistent data display for the client.

  *These standards must be followed in order for the data to display correctly.
  ** Any changes to file names without correction on GeoServer will result in critical failure of the application.

When creating a new data store, steps to follow are:

- Add new data-store -> GeoTIFF
  - Workspace: Canada
  - Connection Parameters URL: <path to you data> (at AAFC this is */mnt/fourTb/spatial_data/soils/canadasoils/LL*)
  - Data Source Name: name/depth/resolution/coords (ex. *SNDPPT_M_sl3_250m_ll*)
  - Description: Layer description from META_GEOTIFF_1B_2017.csv, found at */mnt/fourTb/spatial_data/docs*
- Save -> Publish Layer
  - Title: Remove 'Canada_' from title. GeoServer supports same layer names as long as they belong to different workspaces. Should be same as data-store name.
  - Abstract:
    - Data available through Agriculture and AgriFood Canada, STB. 2017. Global GeoTiffs available at ftp://ftp.soilgrids.org/data/recent/ ISRIC - World Soil Information
- Publishing -> WMS settings -> default style -> associated style for that soil attribute

When creating a new layer group, steps to follow are:

- Layer groups -> Add New Layer group
  - Name: Soil grid attribute title (ex. SNDPPT_M_250m)
  - Title: Same as above, soil grid attribute title (ex. SNDPPT_M_250m)
  - Abstract: description of layers from META_GEOTIFF_1B_2017.csv, found at */mnt/fourTb/spatial_data/docs*
  - Workspace: Canada
- Add Layer: Add from loaded layers. Arrange in descending order.
- SAVE

When creating a new SLD style, steps to follow are:

- Styles -> Add new style
  - Workspace: empty
  - Title: <attribute to style> i.e. SNDPPT. For styles from SoilGrids or ulysses, <soil attribute>_SoilGrids
  - Browse for style -> Styles from Ulysses have to be transferred locally prior to upload. They can be found at */mnt/fourTb/spatial_data/soils/globalsoils/sld_files*
  - Validate
  - Upload

PHP

- ### Installed Modules

  PHP requires several modules not included on a standard install. See section 11.3 of Setup_and_Config document for specific install steps. These modules are:

  - Curl
  - Xml
  - Zip
  - Json

- ### Allow File Upload

  Modifications are made to the php.ini file to allow file uploads. See section 11.7 of Setup_and_Config document for steps.

- ### Directory Permission

  If a web page is to interact with directories on the server with PHP, Apache needs to have the appropriate permissions to those directories. Instead of making the entire web root folder available to Apache (security concern), specified directories are provided for web pages to interact with the server.

  The Apache user and group are both *www-data*. Any directories that need to allow access to Apache should be changed from *root* to *www-data*. The Soil Data Portal allows access to php and python directories.

Python

Metadata statistics files and Plant Disease Model processing requires python 3.0. Required libraries are GDAL and numpy.

Cron-Job

A cron-job, or chronological job, is setup to clear the *temp* file (location of client's download data) at noon each day. Cron-jobs are specific to Ubuntu with equivalent tasks available on different OSs. See section 13.2 of the Setup_and_Config document for specific steps.

This method of data handling isn't desireable for scaling. For the scope of this project it was an acceptable solution, but services with higher volume will experience memory issues if the *temp* file gets too large. Deletion of the .zip directly after download would be a favorable solution.

Final Notes

Browser Compatibility

The Soil Data Portal was created using modern web browsers. Chrome, Firefox, and Explorer 10 will display the application nicely. Older browsers may experience issues. The responsive design will allow for tablet and computer screen use, though mobile was not integrated. These features are room for development in the future.

Deployment

Specific to the AAFC Soil Data Portal, deployment of the application was achieved by cloning the source code from github. All second level directories were moved up into */HTML*, proving a clean url to the app; http://www.daedalus.agr.gc.ca/soil-data-portal. When moving second level directories up, attention must be paid to changes in user for directories requiring PHP/APACHE access. As specified in this document, all directories/files requiring PHP/APACHE access must have user www:data.

Semantic UI Overrides

Semantic UI provides sections in the source code where overrides to the base framework code can be place. Sidebar and Container modules have overrides located in the source css. Most notably is 'overflow-y: visible'. This is noted as it could be a difficult to find answer for unexpected behavior.

References

Example Requests

REST - Layer-groups from Canada workspace in json format

```
http://ulysses.agr.gc.ca:8080/geoserver/rest/workspaces/Canada/layergroups.json
```

REST - Full list of coverage stores from Canada workspace

```
http://ulysses.agr.gc.ca:8080/geoserver/rest/workspaces/Canada/coveragestores/
```

WCS - Subset of layer from Canada workspace (percentage of sand at 1st depth layer)

```
http://ulysses.agr.gc.ca:8080/geoserver/ows?service=WCS&version=2.0.1&request=GetCovera
ge&CoverageId=Canada_SNDPPT_M_sl1_250m_LL
```

Appendix 1

System

- Ubuntu v16.04 LTS
- Apache Web Server v2.4.18
- Tomcat v9.0.0
- GeoServer v2.9.1
- PHP v7.0.8
- Semantic UI v2.2.9
- Python 3

Modules

- PHP - zip
- PHP - curl
- PHP - xml
- Python - GDAL

Libraries

- OpenLayers v3.19.1
- jQuery 3.1.1
- Proj4.js - Javascript reprojection library
- Shp2Geojson.js - Shapefile upload to OpenLayers