THE ROLE OF OPEN-SOURCE PYTHON PACKAGE GEOSERVER-REST IN WEB-GIS DEVELOPMENT

Tek Bahadur Kshetri[1], Angsana Chaksan[1], Shraddha Sharma[1],

[1]Geoinformatics Center (GIC), Asian Institute of Technology (AIT), Thailand

(iamtekson@gmail.com, angsana@ait.asia, shraddhas@ait.asia)

KEY WORDS: geoserver, geospatial data, geoserver style, web GIS, geospatial science, python package.

ABSTRACT:

This work describes geoserver-rest, an open-source python package that can manage the geospatial data in a geoserver which is helpful for uploading, editing, and deleting the raster/vector layers from various sources. It is also useful for generating the style/legend from the uploaded geospatial data. Thus, generated legend can be used for visualization of maps in the web-GIS platform. The package is successfully used to build the web-GIS portal for agricultural datasets of Afghanistan, which has around 6000 map layers. The main benefit that geoserver-rest provides to this project is the ability to upload the data to the geoserver and create the styles file dynamically. Thus, created style file are directly linked to the corresponding layer and provide the Web Mapping Service (WMS) standard and visualize in an interactive way.

1. INTRODUCTION

The common procedure for publishing the geospatial data on web is by using the map servers such as MapServer (Kropla, 2005), GeoServer (Iacovella, 2014), ArcMap Server, ncWMS (Blower et al. 2013) etc. Such servers provide the interactive interface to upload, update and delete the geospatial data. Some of them also provide the interface to generate the legend for the uploaded dataset. However, there is not, to the best of my knowledge, any open-source program that can automatically generate the legends for uploaded dataset dynamically.

Sometimes the complexities occur on publishing the geospatial data due to size, file format, source, and amount of data. Such complexities can be solved using geoserver. The geoserver is the powerful open-source server for storing, managing, and presenting the geospatial data (Iacovella, 2017). The data can be published from any major data sources like shapefile, GeoTiff, PostGIS etc. The data from database, API and directly as a file can be published in the geoserver and share. The geoserver contain vector as well as raster data optimization techniques. Generally, vector data can be optimized by storing it in the spatial database, by using spatial indexing. In case of raster, image mosaic and image pyramid technique can be used. This technique will help to solve the problem of size of data.

The geoserver allows us to share spatial information to the public. Such data needs to be visualized in a cartographic way so that general people also can understand it. Thus, generated maps can be share to the public in a web. The Open Geospatial Consortium (OGC) provides the lots of standards for sharing such maps. One of the commonly used standards is Web Map Service (WMS) (de La Beaujardiere, 2006). WMS standards provide the geo-registered map image that can be displayed in a browser application. The implementation of WMS standard in geoserver can create the maps in a variety of output formats. It also supports the web Feature Service (WFS) (Vretanos, 2005) and Web Coverage Service (WCS) (Baumann, 2010) standard which allow sharing and editing the data that are used to generate the maps.

The geoserver-rest is the open-source spatial data management library written in python based on the geoserver API. Python is the chosen language, because it is easy to use, object oriented, runs on all the major Operating Systems (OS) and already has thousands of libraries and frameworks. It is one of the leading technologies for web development (Taneja and Gupta, 2014). Implementation of geoserver-rest in python will significantly enhance the approach of GIS application development. Since the package is developed in the python library, it can be useful for the web platform as well as desktop applications development. Rather than serving the data in client and server, it also generates the style files and creates the legend dynamically by reading the uploaded dataset. The automation for geospatial data sharing and dynamic style creation feature makes it unique and more popular and useful in the latest web-GIS development works. It will be very helpful for editing, updating, and deleting large sets of data.

The geoserver-rest helps to upload the raster as well as vector layers from various sources to the geoserver. The package also helps to connect the multiple spatial databases and store the layers in geoserver. It supports the standard HTTP methods GET, POST, PUT, DELETE. The web-GIS application having uploading, managing and presenting the geospatial data functions can be integrated with the help of this library.

2. ARCHITECTURE AND IMPLEMENTATION

The package is developed using the python library. It has four dependencies, that are pycurl, GDAL, seaborn and psycopg2. The pycurl library is used for the client-side URL transform. The pycurl library can communicate the geoserver API URL and transfer and manage the file. Psycopg2 is the most powerful python postgreSQL adapter. This library is used for communication between PostGIS and geoserver. Generally, in geoserver, the vector data is stored in the postGIS database and linked to the geoserver. The geospatial data can be read and write using the gdal. The seaborn library is used for creating the color ramp. Depending on the user case, a discrete or continuous color ramp will be created. Thus, created ramp will be used in the WMS styling.

The geoserver-rest library is still in the development mode. It already supports commonly used functionalities from geoserver API. Create, update, get and delete the geospatial data from geoserver are the most used geoserver functionalities in broad term. The settings for WMS, WFS, WCS standards, geoserver user, groups and roles are still under development.

The overview of the geoserver-rest architecture is shown in Fig. 1. The library will check the authentication of the user by simply username and password. If the user is not the authenticated user, it will throw the 403-error code, else the user can proceed with the request. Sometimes while calling some function there might be some problem. Not passing the path to the raster layer during create_coveragestore call, trying to delete the non-existing layer, trying to delete non existing style file, trying to update the non-existing layer etc are some of the examples of request failure. If the request does not have any issue, it will return the 2xx status code, which means request success.

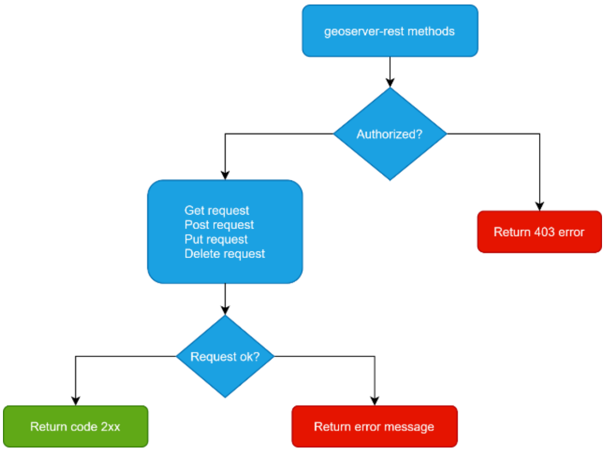


Figure 1. Architecture of the geoserver-rest

2.1. Geoserver-rest requests

The functionalities of this library can be further categorized into the four types, 1. GET request, 2. Create request, 3. Update request, and 4. Delete request.

2.1.1 GET request

The get requests are generally used to check whether instances exist or not. Depending on the function type it also provides the information about the workspace, layer, or style file in JSON format. Some of the get request examples are as below,

API functions	**Description**
get_workspace	Get/check the geoserver workspace
get_coveragestore	Get/check whether raster datastore exist or not
get_featurestore	Get/check the postGIS connection
get_featuretype	Get the name of features in specific featurestore
get_feature_attribute	Get the name of columns in specific featuretype

Table1. Some of the commonly used get request on geoserver-rest

2.1.2 Create request

The create requests are generally used for uploading the data to the geoserver. Some of the most used create request examples are as below,

API functions	**Description**
create_workspace	Create the new workspace to the geoserver
create_coveragestore	Upload raster data to the geoserver
create_featurestore	Connect the postGIS database with geoserver
publish_featurestore	Publish the vector dataset from PostGIS featurestore
upload_style	Upload the style file (generally SLD) to the geoserver
create_coveragestyle	Dynamically create the style file for raster layer
create_outline_featurestyle	Dynamically create outline only style file for vector layer
create_categorized_featurestyle	Dynamically create the categorized legend based on the one of the columns of the vector layer
create_classified_featurestyle	Dynamically create the classified style legend based on one column of the vector data
publish_style	Connect the style file to the required layer

Table2. Some of the most used create request on geoserver-rest

2.1.3 Update request

The update request is used for updating the instance in geoserver. Generally, by passing “overwrite=True” parameter on the create request to update the instance. For example, if the user wants to update the raster dataset, then he/she must pass one extra parameter i.e., “overwrite=True” in create_coveragestore function.

2.1.4 Delete request

The delete request is used to delete the instance/data from the geoserver. Some of the most used delete request examples are as below,

API functions	Description
delete_workspace	Delete the workspace from geoserver
delete_layer	Delete the layer from geoserver
delete_coveragestore	Delete the raster layer store from geoserver
delete_style	Delete the style file from geoserver
delete_featurestore	Delete the featurestore from geoserver

Table3. Some of the commonly used delete request on geoserver-rest.

2.2 Geoserver-rest style

There are basically two types of style files in geoserver-rest library, 1. vector data style, 2. Raster data style. Creating the dynamic styles based on the uploaded dataset makes this library unique from others.

The “create_coveragestore” function will be used for creating the style file for the geotiff image. The full process for uploading raster data can be seen in fig 2. The library will read the uploaded raster file and take the min and max values from the uploaded raster. Based on the color palette and number of classes from the client, it will classify the raster and create the Styled Layer Descriptor (SLD) file (Lupp, 2007). The SLD file will be linked to the uploaded raster. Finally, the raster map as a output will be available through WMS from geoserver. The geoserver is also able to provide the legend of the map from getLegendGraphics request.

The geoserver-rest supports three types of styles available for vector data, 1. Outline border only 2. Categorized and 3. Classified. For vector data, the first step is to upload the data to the PostGIS database. After that, the PostGIS database will be connected to the geoserver using the create_featurestore function from geoserver-rest. The style function can be chosen based on the type of visualization need. Based on the color palette and attribute field name from the client, it will generate the SLD file. The SLD file will be linked to the same PostGIS layer. Finally, vector map as an output will be available through WMS from geoserver.

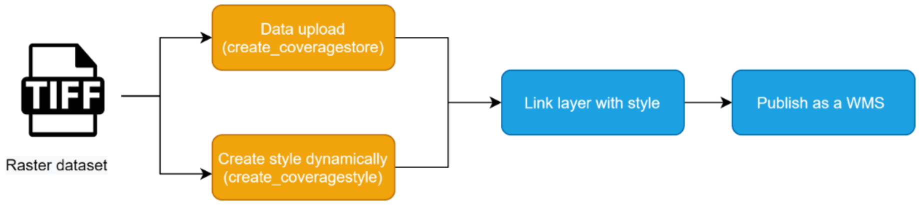


Figure 2. workflow for publishing the WMS standard for raster dataset

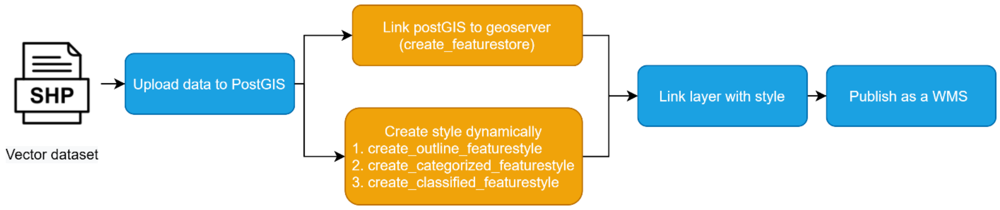
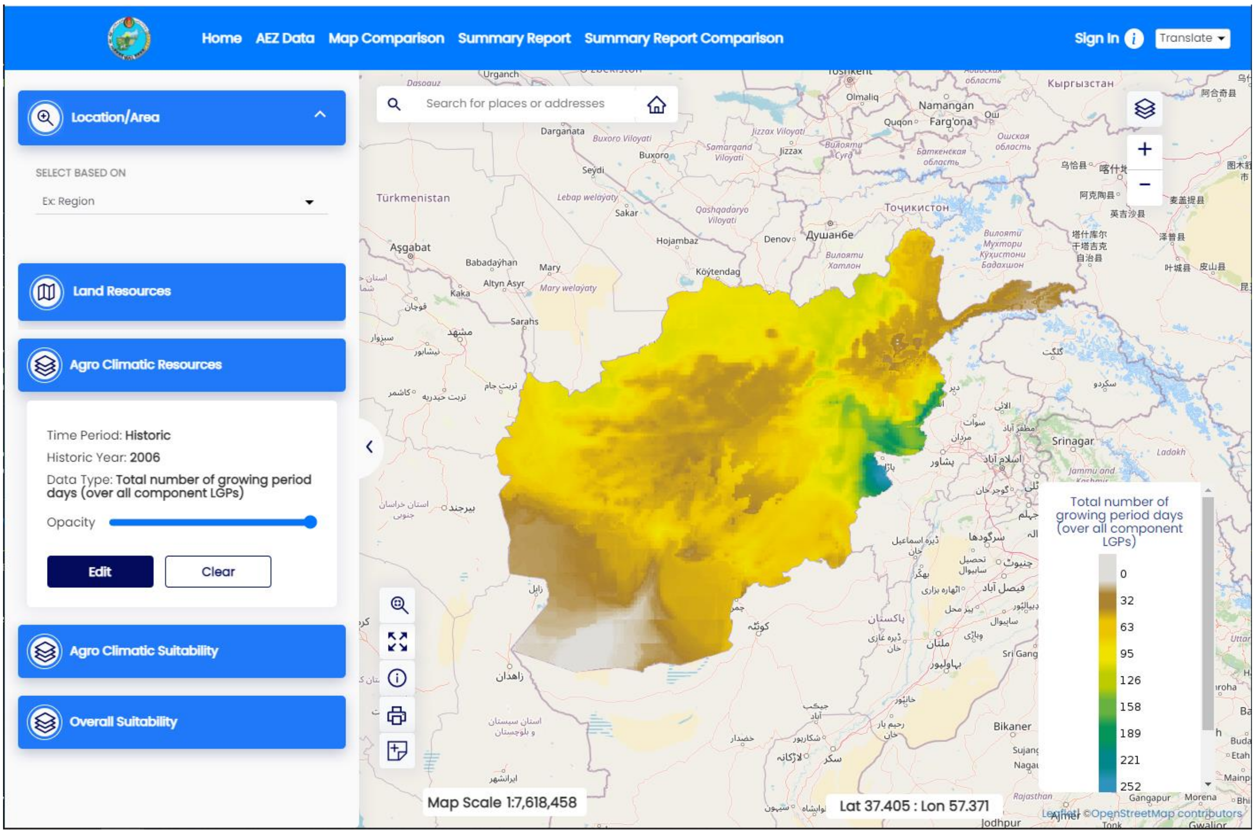

SHP
Vector dataset
Upload data to PostGIS
Link postGIS to geoserver
(create_featurestore)
Create style dynamically
1. create_outline_featurestyle
2. create_categorized_featurestyle
3. create_classified_featurestyle
Link layer with style
Publish as a WMS

Figure 3. workflow for publishing the WMS standard for vector dataset

3. CASE STUDY: DEVELOPMENT OF LAND RESOURCES INFORMATION MANAGEMENT SYSTEM (LRIMS) FOR AFGHANISTAN

This section describes the successful example geoserver-rest where it was used for management of all the agricultural related geospatial data in the geoserver. The LRIMS web-portal was developed for the land resource information management and Agro-Ecological Zones (AEZ) to support agricultural policies and climate change adaptation in the country. The web-GIS portal is used for the visualization of land resources, agro climatic resources, climate and agro climatic suitability results in the form of maps. Based on the authorization level, users will be able to upload, visualize, update, download and delete the datasets and maps.

The LRIMS contains mainly three types of datasets, land resources, agro climatic resources, and suitability and potential yield. The land resources include multiple spatial layers of climate, terrain, water, land cover, etc. which are used for estimating crop suitability and potential yields. The Agro Climatic resources category includes relevant characteristics of prevailing thermal and moisture regimes and growing periods which are used for estimating crop suitability and potential yield in intermediate levels. And the suitability and potential yield includes the possible upper limit to produce individual crops under given agro-climatic, soil and terrain conditions for a specific management and input levels. In each category further the datasets are further categorized based on the time, rcp and the data type.



SELECT BASED ON
Ex: Region

Time Period: Historic
Historic Year: 2006
Data Type: Total number of growing period days (over all component LGPs)
Opacity

Map Scale 1:7,618,458
Lat 37.405 : Lon 57.371

Figure 4. Interface of the web-GIS portal of LRIMS

In this portal, there were nearly 6000 raster layers and nearly 100 vector layers. Managing this much huge raster dataset was a challenging task. Also, the biggest challenge was producing the proper cartographic maps from the grayscale raster. We could visualize the data in QGIS and produce a nice map and upload it to the geoserver with reference style was an easy option. But since it is the manual process, creating the map from a dataset, uploading to the geoserver, and connecting the layer with the associate style file would consume lots of time.

The main benefit that geoserver-rest provides to this project is the ability to upload the data to the geoserver and create the styles file dynamically. The fig. 5 and fig. 6 shows the overall process WMS publishes in geoserver. The raster data upload function named create_coveragestore from geoserver-rest library is used to upload the raster dataset. And for creating the style file, a raster style function named create_coveragestyle from geoserver-rest library is used. Later the uploaded data and the dynamically created style file of the raster is linked together to produce the WMS in geoserver. For vector data, the first step is to upload the data to the PostGIS database. After that, the PostGIS database will be connected to the geoserver using the create_featurestore function from geoserver-rest. There are three types of styles available for vector data, 1. Outline border only 2. Categorized and 3. Classified. The style function can be chosen based on the type of visualization need. Thus, generated style will be linked to the uploaded shapefile and produce the WMS in geoserver.

The functions were iterated and run over each dataset by applying the python for loop to upload and create style.

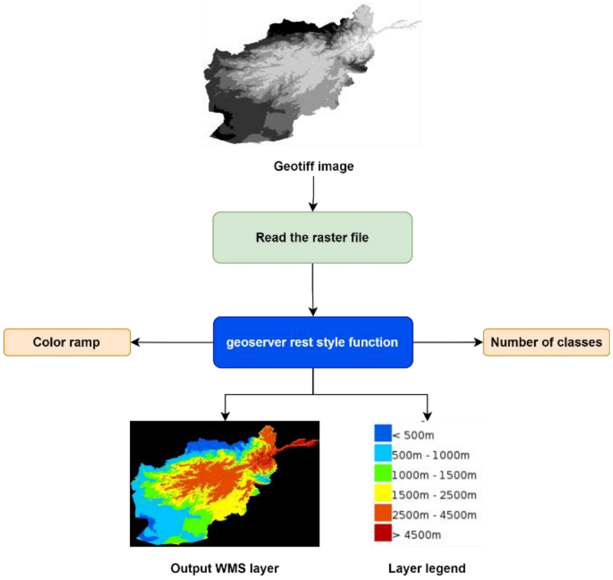


Figure 5. Process of generation of WMS layer of raster dataset using geoserver-rest. The input data is the altitude zones of Afghanistan

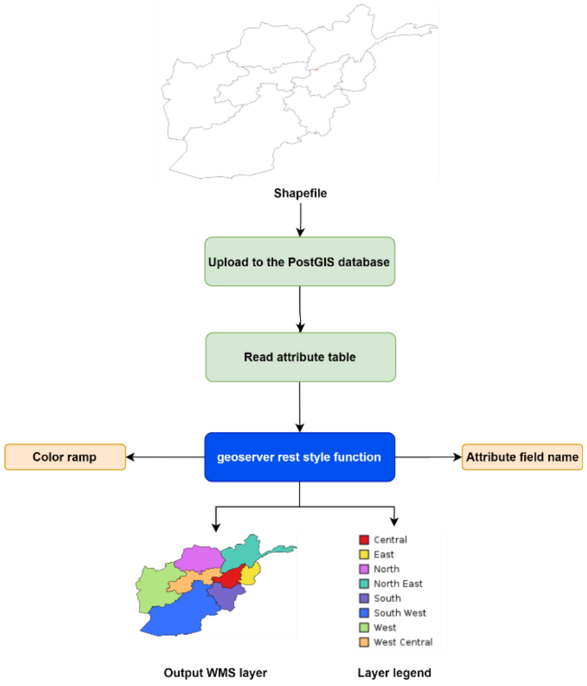


Figure 7. Process of generation of WMS layer of raster dataset using geoserver-rest. The input data is the administrative regions of Afghanistan

4. DISCUSSION

The geoserver-rest is the first open-source python library that can manage the geospatial data and can generate the style/legends of geospatial data dynamically on geoserver. The main features of the geoserver-rest provide are python implementation of geoserver API, simple, easy to use, clear documentation, ability to get, create, delete, and update the geospatial data on geoserver using python.

There is also the PHP implementation of the geoserver API named as “GeoServer PHP Client” and “geoserver-rest-api”. Both libraries have very limited functionalities. The library can create workspace, datastore, coveragestore, upload files, manage styles. The libraries are not maintained anymore. These libraries are not tested for the geoserver 2.15+ versions. The “gsconfig” is also the python 2 implementation of the geoserver API. The gsconfig project was initiated 7 years ago and it has not been maintained since the last 2 years. Since python 2 is no longer supported by the Python Software Foundation, the gsconfig is also about to die. Due to this reason, the python 3 version of the same library was initiated in July 2019. It was named as “geoserver-restconfig” and is available in github. The library is currently maintained by the geonode community. The library is used in geonode. But this library does not provide proper documentation.

All the libraries, methods described above lack the ability to generate the style/legend dynamically.

Since the geoserver-rest uses the gdal as the dev dependencies, the installation of geoserver-rest will be a little bit more complex than other libraries. Some advanced features are still under development. These are a few minor drawbacks of this library.

The library is recently developed, it can guarantee that it works perfectly with the geoserver 2.15+ versions. The current development of geoserver-rest is focused on the new web technology that allows the visualization more easily and effectively. The geoserver-rest became a popular open-source library and standard tool for managing the geospatial data and producing the interactive maps.

5. ACKNOWLEDGEMENT

We would like to acknowledge the Geoinformatics Center (GIC), AIT, Thailand for the guidance and motivation for working on this package. We would like to thank Rabin Ojha and Trevor James Smith for their valuable contribution on the documentation. We want to show our gratitude for Pablo Zader, Nicola Jordan, Lisa Stillwell for their contribution on new features implementation.

6. REFERENCES

Baumann, P., 2010: Beyond rasters: introducing the new OGC web coverage service 2.0. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 320-329. doi.org/10.1145/1869790.1869835.

Blower, J., Gemmell, A., Griffiths, G., Haines, K., Santokhee, A., Yang, X., 2013: A Web Map Service implementation for the visualization of multidimensional gridded environmental data. *Environmental Modelling & Software, 47*, 218-224. doi.org/10.1016/j.envsoft.2013.04.002.

de La Beaujardiere, J., 2006: *OpenGIS® Web Map Server Implementation Specification. Version 1.3. 0.* Wayland, MA: Open Geospatial Consortium. doi.org/10.25607/OBP-656.

Iacovella, S., 2014: *Geoserver Cookbook.* Livery place: Packet Publishing Ltd.

Iacovella, S., 2017: *GeoServer Beginner's Guide: Share Geospatial Data Using Open Source Standards.* Livery Place: Packt Publishing Ltd.

Kolokotsa, K. G., Mavrigiannaki, A., Kalaitzakis, K., Dionysia-Denia., 2017. Design and development of a Web based GIS platform for zero energy settlements monitoring. *Energy Procedia, 134*, 48-60. doi.org/10.1016/j.egypro.2017.09.598.

Kropla, B., 2005: *Beginning MapServer.* United State of America: Apress. doi:10.1007/978-1-4302-0053-6.

Lupp, M., 2007: Styled Layer Descriptor profile of the Web Map Service Implementation Specification [Corrigendum]. Version 1.1. 0 (revision 4). Open Geospatial Consortium.

Mohan, A. K., Mohanty, J., Eldho, T., B.K., E. R. 2014: A web GIS based integrated flood assessment modeling tool for coastal urban watersheds. *Computers & Geosciences, 64*, 7-14. doi.org/10.1016/j.cageo.2013.11.002.

Nugroho Joshua, J. W., Agus Swastika, I. P., Windha Daniaty, T. O., 2017: E-Government Integration through Implementation of web-based GIS on Community Health monitoring in Jembrana Regency, Bali. *Procedia Computer Science, 124*, 552-559. doi.org/10.1016/j.procs.2017.12.189.

Sánchez-Aparicio, L. J., Masciotta, M.-G., García-Alvarez, J., Ramos, L. F., Oliveira, D. V., Martín-Jiménez, Monteiro, P., 2020: Web-GIS approach to preventive conservation of heritage buildings. *Automation in Construction, 118*, 103304. doi.org/10.1016/j.autcon.2020.103304.

Taneja, S., Gupta, P. R., 2014: Python as a tool for web server application development. *Int. J. Information, Commun. Comput. Technol, 2*, 2347-7202.

Vretanos, P., 2005. Web Feature Service Implementation Specification, Verson 1.1. 0. Open Geospatial Consortium.