**Developing a PyQGIS Script to Automate the Process of Ingestion of KoboToolbox and ODK Central Data into GeoServer**

Appalla Sai Manoj

September 2022

SUPERVISOR:

Mr. Koti Shiva Reddy



***iirs***



**Developing a PyQGIS Script to Automate the Process of Ingestion of KoboToolbox and ODK Central Data into GeoServer**

Appalla Sai Manoj

IIRS, Dehradun, September 2022

Final report submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente and Indian Institute of Remote Sensing (IIRS) in partial fulfilment of the requirements for the Postgraduate Diploma course in Geo-information Science and Earth Observation.

Specialization: Geoinformatics

SUPERVISOR:

Mr. Koti Shiva Reddy

FINAL ASSIGNMENT ASSESSMENT BOARD:

[Title, Initials, Name (Chair)]

[Title, Initials, Name (Examiner, Name of Institute)]

etc



***iirs***

Disclaimer

This document describes work undertaken as part of a Postgraduate Diploma (PGD) programme in Geo-information Science and Earth Observation (specialisation: Geoinformatics) offered within the framework of Joint Education Programme (JEP) of the Indian Institute of Remote Sensing (IIRS) and the Faculty of Geo-information Science and Earth Observation (ITC) of the University of Twente (UT), The Netherlands. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the IIRS and ITC.

Abstract

Type the text here.

Acknowledgements

Type the text here.

Table of contents

[1. Introduction 1](#_Toc113438806)

[1.1. Background 1](#_Toc113438807)

[1.2. Problem Statement 1](#_Toc113438808)

[1.3. Innovation 1](#_Toc113438809)

[1.4. Objectives 2](#_Toc113438810)

[1.5. Software and Hardware Used 2](#_Toc113438811)

[2. Chapter title 3](#_Toc113438812)

[2.1. Title section 3](#_Toc113438813)

[3. Methodolgy 3](#_Toc113438814)

[3.1. Flowchart 3](#_Toc113438815)

[3.2. Create ini files 3](#_Toc113438816)

[3.3. Use the geoserver-rest API library 5](#_Toc113438817)

[3.4. Code to update the project and form names in the ini files 5](#_Toc113438818)

[3.5. Define Import class 5](#_Toc113438819)

[3.6. Define importData function 5](#_Toc113438820)

[3.7. Define updateLayerXML and updateFields functions 5](#_Toc113438821)

[3.8. Define collectData function 5](#_Toc113438822)

[3.9. Test the script 5](#_Toc113438823)

List of figures

List of tables

# Introduction

## Background

Data collection and analysis have always been essential phases in any research process. Traditionally, GIS data was collected using various surveying techniques, which is a complex process and, at the same time, requires trained personnel (Nowak, Dziób, Ludwisiak, & Chmiel, 2020). However, with the advancement in technology, the development of the GNSS, and the introduction of GPS chips in mobile phones, today, every individual is directly or indirectly contributing to GIS data collection. The data collected in the field are brought back to labs, stored, and analyzed to elicit knowledge. If this collection, storage, and analysis process happen simultaneously, it saves a lot of time for the researchers. This is where automation plays a prominent role.

QGIS is an open-source GIS software used to handle the GIS data and perform spatial analysis on the data (QGIS Development Team, 2022). QGIS also provides PyQGIS python API, which can perform all the tasks performed by QGIS and is also used to automate the GIS processes. PyQGIS scripts can be executed directly in the QGIS python console or standalone. PyQGIS standalone script can automate the ingestion of the data collected from mobile apps to GeoServer, a map server that can be used to share the GIS data with the public in the form of WFS or WMS layers. These WFS or WMS layers, when accessed by the analysts, help them to comprehend the data in real time.

## Problem Statement

One way to approach GIS data collection is by preparing a form, uploading it to a mobile GIS data collection app like KoboCollect or ODK Central, downloading the form in the app, collecting the data, downloading the collected data in the format of a shapefile, and finally upload

the same for GeoServer.

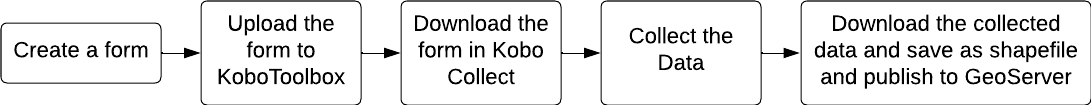


Figure 1: Data collection process without automation

The project focuses on automating the last two processes where data collection and ingestion into GeoServer co-occur. This helps us to analyze the collected data in real time.

## Innovation

A plugin QRealTime does a similar thing where an empty shapefile is created and uploaded to the Kobo and ODK server along with the configuration. Later, the forms can be accessed through a mobile application like ODK Collect and Kobo Collect to collect the data in the field. Finally, the data collected is imported into the shapefile depending on the sync time set by the user. The innovation in this project is that the standalone script runs independently from QGIS, imports the data into the shapefile or PostGIS database, and publishes the same to GeoServer in real-time.

## Objectives

### Main Objective

To develop a python script to automate the ingestion of Kobo Toolbox and ODK Central Data into GeoServer.

### Sub Objectives

1. Develop a standalone script using PyQGIS API to import the data collected using KoboCollect and ODK Collect apps into a shapefile or a PostGIS Database.
2. Develop a python script to publish the shapefile layer or PostGIS database to GeoServer and sync the data to initiate the automation process.
3. Develop a use case where this script can be used.

## Software and Hardware Used

* **Software used**
  + KoboCollect to collect the data
  + QGIS to create the shapefile and upload it as a form to KoboCollect and the PostGIS database
  + PostgreSQL
  + Geoserver
* **Hardware used**
  + MacBook Pro (2.6 GHz 6-Core Intel Core i7, 16 GB 2667 MHz DDR4, AMD Radeon Pro 5300M 4 GB, Intel UHD Graphics 630 1536 MB)
  + A Mobile phone to collect a few points

# Chapter title

## Title section

### Title sub-section

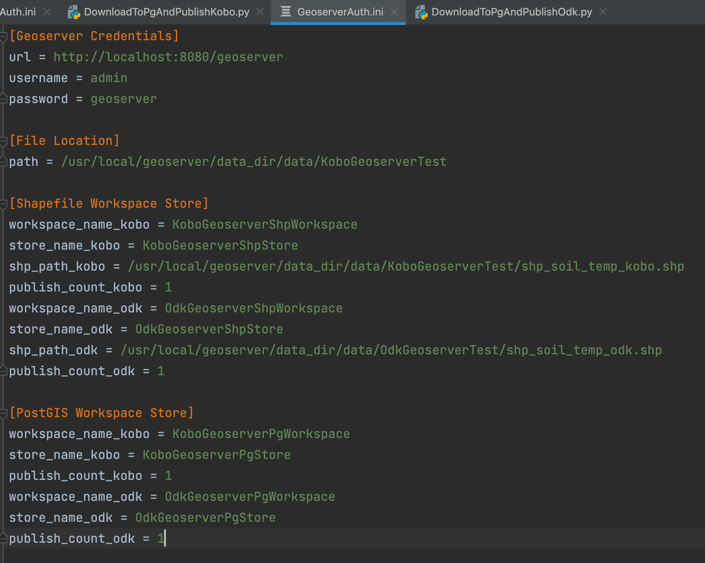
# Methodolgy

## Flowchart

## Create ini files

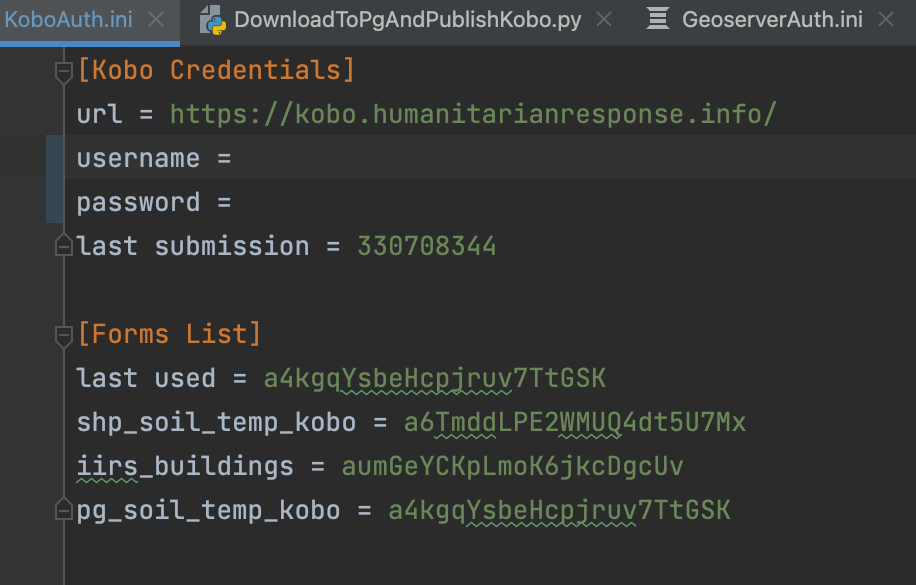
The configuration files have been created to store some information generated by the script and information entered by the user, like credentials, last-used forms, file paths, etc. Four .ini files were created for GeoServer, Kobo, ODK, and PostGIS.

* **GeoserverAuth.ini**: This file has three sections; where the first section stores the credentials required to authenticate the GeoServer. The second section deals with shapefiles and stores the workspace name, store name, shapefile path, and publish count. Finally, the third section deals with the PostGIS database and will store the workspace name, store name, and publish count information.

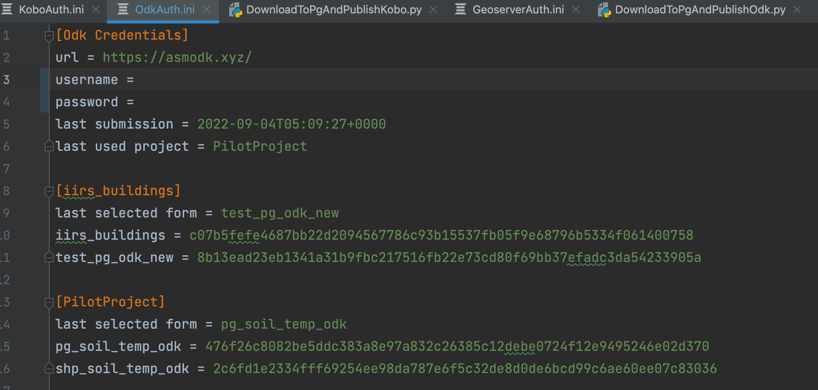


As seen in the above screenshot, only the publish\_count\_kobo and publish\_count\_odk variables are updated by the code, and the rest of the variables are changed by the user.

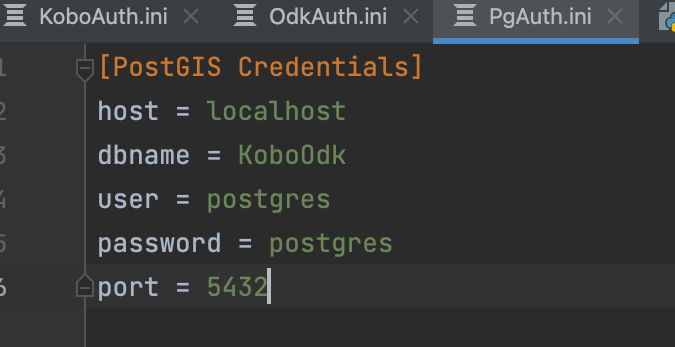
* **KoboAuth.ini**: This ini file has two sections. The first section stores the Kobo Collect credentials of the user. The second section gets updated when the user runs the python code used to update the forms list. Finally, the last used form is updated by the user.



* **OdkAuth.ini**: This ini file has one section where the authentication information entered by the user, the last used project, and the last submission reside. The other sections appear when the user runs the code to update the project and form data.



* **PgAuth.ini**: This ini file has only one section where the user enters the credentials to connect to the PostGIS database, database name, and port number.



## Use the geoserver-rest API library

This library was developed by Tek Bahadur Kshetri (Tek Bahadur Kshetri, 2021). Only the functions necessary to publish the shapefile and the PostGIS database were extracted and put in a separate python file named Geoserver.py. This file has three essential methods used to publish the data to GeoServer.

* **create\_datastore <method>:** This method will accept the store name, workspace name, and the shapefile path as an argument and creates a store in GeoServer by making an API call.
* **publish\_featurestore <method>:** This method takes the workspace name, store name, and shapefile name as an argument and then makes an API call to publish the shapefile to GeoServer
* **create\_featurestore <method>:** This method is used to create a store for the PostGIS database, and the workspace name, store name, and the PostGIS credential are supplied as arguments. This method makes an API call to GeoServer to create a data store for the PostGIS Layer.

## Code to update the project and form names in the ini files

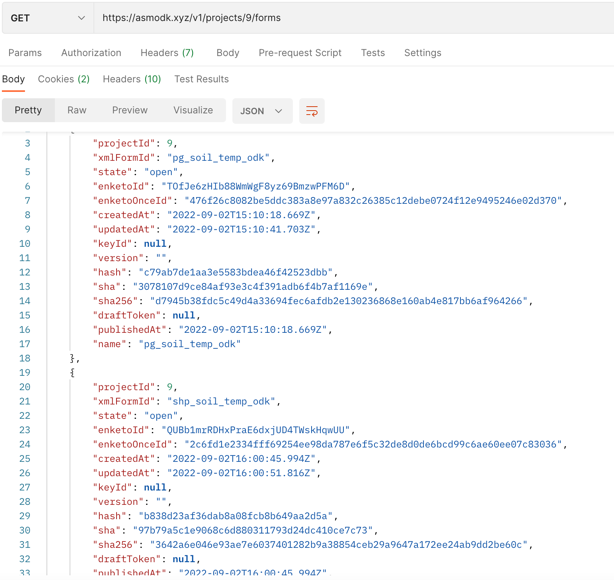
Two python files with the names KoboFormsList.py and OdkFormsList.py were created to extract all the project names and form names along with their ids and update them in their respective ini files.

* **KoboFormsList.py:** This python file has the getFormsList function, which is used to fetch the form names and their ids in the selected project in the Kobo Server. An API call is made to get the forms list. The following is the request made and the response returned.

Graphical user interface, text, application, Teams

Description automatically generated

* **ODKFormsList.py:** This python file has the getFormsList function, which is used to fetch the form names and their ids in the selected project in ODK Central Server. An API call is made to get the forms list. The following is the request made and the response returned.



## Define Import class

This Import class is defined in four python files, namely, DownloadToShpAndPublishkobo.py, DownloadToShpAndPublishOdk.py, DownloadToPgAndPublishKobo.py, and DownloadToPgAndPublishOdk.py. This ImportKobo and ImportOdk class hosts all the necessary methods to import the data into the shapefile and PostGIS database and publish it to GeoServer.

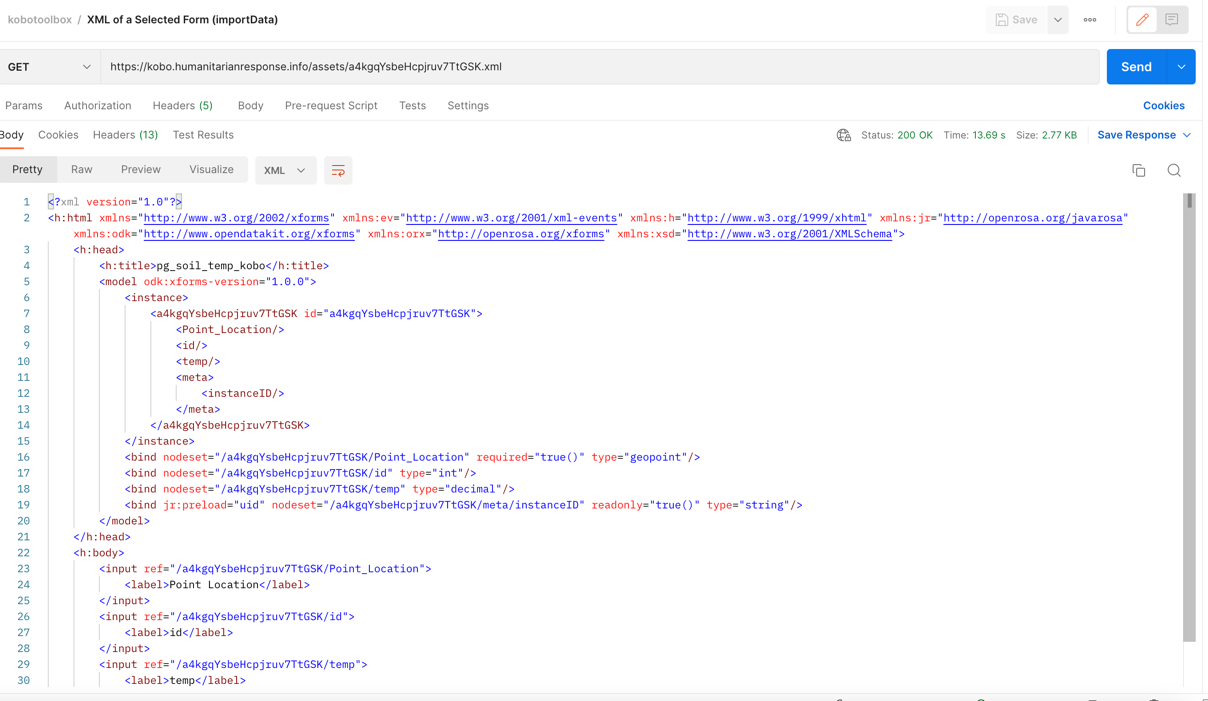
* **Kobo:** The ImportKobo class initializes the Kobo URL, username, and password parameters when the object is created. These will be later used to make the API requests to the Kobo Server.
* **ODK:** The ImportOdk class initializes the user token, project id, and form name. The other functions later change these variables when the script is run.

## Define importData function

This function passes the XML response from the request sent to the Kobo Server using the Kobo API and ODK Central server using the ODK Central API to the updateLayerXML function, and the return value of the updateLayerXML function is passed as an argument to collectData function. The following are screenshots of the XML response from which the other functions will extract the information.

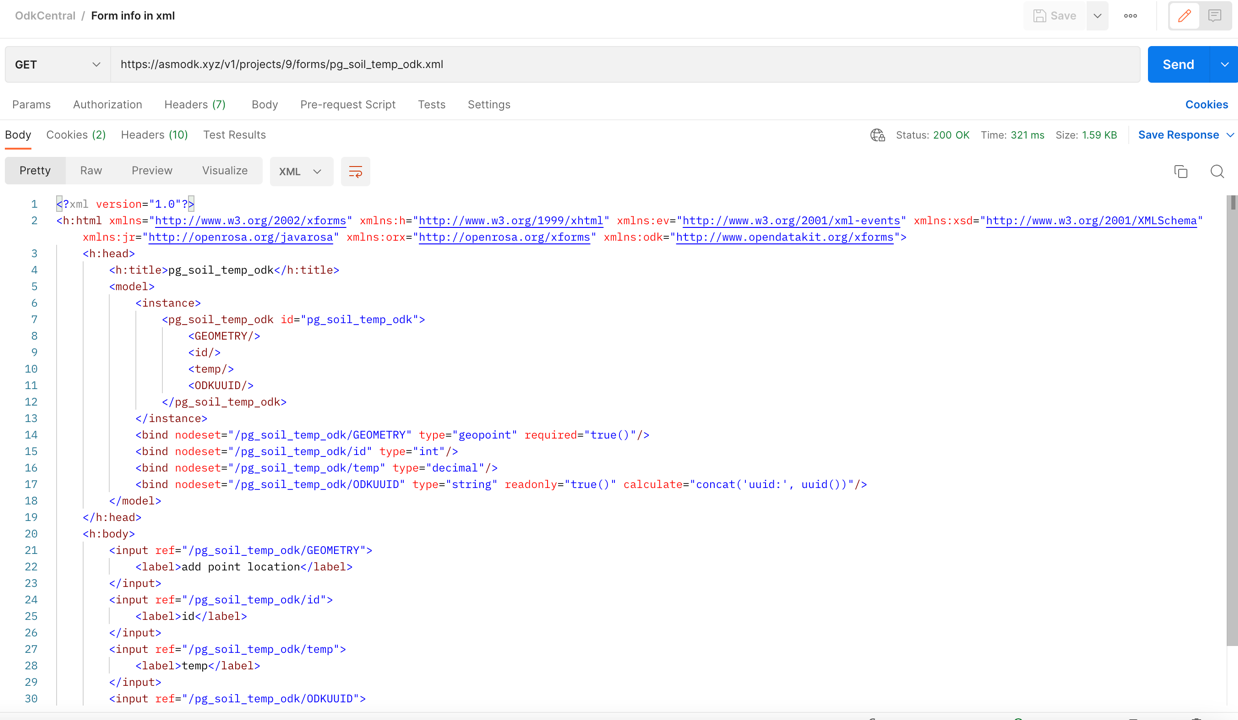
* **Kobo**

The XML response returned by Kobo Server is shown in the following screenshot.



* **ODK**

The XML response returned by the ODK Central server is shown in the following screenshot.



## Define updateLayerXML and updateFields functions

* **updateLayerXML**

This function extracts the following from the XML response returned above:

1. Title of the selected form
2. The instance of the selected form
3. The version of the selected form is available.
4. Field names and field types from the <bind> tag in the XML response.

After extracting the above information, qgstype and config of the attribute type are set using a user-defined function. Next, we pass the layer, field names, qgstypes, and configuration as an argument to the updateFields function.

* **updateFields**

This function will add the field names and types to the open shapefile or the PostGIS layer. qgstype is the field type, for example, string, integer, etc., and the config tells what type of value it is, for example, whether it is an attachment, value map, hidden, etc. The configuration of the attribute does not get updated in the standalone scripts. So, every field will be given the standard datatypes like string, integer, float, etc.,

## Define collectData function

This is the function where the

## Test the script

List of references