PUBLICAMUNDI

SCALABLE
REUSABLE
OPEN
GEOSPATIAL
DATA

REPORT FOR

DELIVERABLE D4.1

1 INTRODUCTION

This report provides an overview of the technical characteristics and functionality of deliverable D4.1 “Processing Engine”. Its purpose is to provide a short introduction to the ZOO-Project software and to present the major functionalities and improvements introduced by the PublicaMundi project.

The reader is encouraged to visit the software repository (github.com/PublicaMundi) to receive:

- Up-to-date versions of the software, along with documentation targeted to developers
- Detailed information regarding all development effort (commits, activity, issues)
- Instructions regarding the installation of the software and its dependencies

2 ZOO-PROJECT

In the following we present the open source WPS implementation ZOO-Project which is used at different levels of the PublicaMundi project, especially to address the WP4 research goals and to provide a reliable and scalable Processing Engine to PublicaMundi architecture, as well as a collection of generic and user centric WPS services.

According to the Open Geospatial Consortium, the OpenGIS® WPS Interface Standard provides a set of rules for standardizing inputs and outputs (requests and responses) of geospatial processing services The standard defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of geospatial processes and clients discovery of and binding to those processes.

WPS thus allows to process geospatial data over the Internet and ZOO-Project implements it through an open architecture and a module source code mostly written in C, Python and JavaScript.

ZOO-Project is an open source Web Processing Service (WPS) platform initiated by GeoLabs and maintained by an international community of GIS users and developers. It is composed of three main parts since its creation, which are presented in the following sections.

2.1 ZOO KERNEL: THE WPS SERVER

ZOO-Kernel is an implementation of the OGC WPS 1.0.0 specification written in C language. It is based on a powerful server-side Kernel able to manage and chain WPS services, by loading dynamic libraries and code written in different programing languages. The WPS server can thus execute WPS Services written in C/C++, Fortran 77/90, Ruby, Perl, Java, PHP, Python or JavaScript. This multi-languages support allows the user to select the most suited languages according to his needs, but also to reuse existing libraries and source codes and to turn them into standard-compliant Web Services.

The ability of using different programming languages also addresses several identified issues and needs regarding the common architecture for Spatial Data Infrastructures, which are most of the time composed of several software or libraries not necessarily implemented in the same languages. ZOO-Kernel can indeed be compared as a software bus allowing to use open source geospatial libraries in a generic manner, and to make them communicate through standardized WPS requests. The latter can handle Services input/output data as raw data and/or as other standard

Web Services such as WMS, WFS or WCS, and this last point makes ZOO Kernel able to work side-by-side with traditional cartographic engines and other server-side geospatial technologies.

2.2 ZOO-SERVICES

ZOO-Services is a growing collection of ready to use WPS services built on top of reliable open source libraries such as GDAL, CGAL, GRASS, R and others. They are included in the project source code tree and provide basic examples for creating new Services.

A ZOO Service is a couple composed of the source code to execute and a configuration file which describes the Service. The Service configuration file can be edited using the ZOO Service Configuration File (ZCFG) or the « YAML Ain't Markup Language » (YAML) syntaxes. It describes a WPS service and is automatically parsed by ZOO-Kernel for every GetCapabilities, DescribeProcess and Execute requests. A ZOO Configuration file is composed of fthree distinct sections which are here explained The Main section contains general metadata of the WPS Service. The second and third section are the list of Inputs and the list of output which respectively provide metadata information on supported input/output and types of data nodes such as LiteralData, BoundingBoxData and CoomplexData.

2.3 ZOO-API

ZOO API is a Javascript library designed to make the WPS Process creation and chaining easier. It is based on the server-side interpretation of JavaScript using SpiderMonkey, the Mozilla foundation JavaScript engine. The API allows to create processes using the rather simple JavaScript syntax, and consequently to add logic and common conditionnal statments. The output result of a service can be the input of another one, and complex processing chains can thus be easily powered. The ZOO API provides simple classes and functions to instantiate processes, as well as a lightweight JavaScript utility to convert and reproject vector formats such as GML, KML, and GeoJSON directly into the processing flow.

3 CONTRIBUTIONS IN PUBLICAMUNDI

3.1 GENERAL ENHANCEMENTS

The various efforts accomplished by GeoLabs regarding the ZOO-Project source code leads to a suitable Processing Engine able to reach the goals of the PublicaMundi WP4, but also benefit to the ZOO-Project itself. The latter is indeed welcoming more commits, numerous corrections and optimizations as well as various new features and additional documentation, which bring more overall activity to ZOO-Project lately, and more generally since the beginning of the PublicaMundi project.

As an example and according to the OpenHub platform, the bigger number of contributions commited lately by a growing number of developers helped the project to raise from a « Low activity » status to a « Moderate activity », as shown at https://www.openhub.net/p/zoo-project.

Another clue of this higher activity induced by the PublicaMundi activities lies in the official release of ZOO-Project 1.4, available at http://zoo-project.org/site/Downloads since November 2014. Further development tasks are being pursued and a new ZOO-Project release procedure is already planned for 2015 and the second phase of the PublicaMundi project.

3.2 CONFORMANCE AND PERFORMANCE TESTING

Prior to the numerous modifications and enhancements introduced in ZOO-Project for the need of PublicaMundi, an important work was accomplished on ZOO-Project conformance and performance testing. This initial step could take place through a participative exercise called "WPS Benchmarking" which gathered several individuals from various WPS implementation projects (namely 52°North WPS, GeoServer, PyWPS and ZOO-Project) and allowed to compare software conformity and performance according to different scenarios and for the different type of WPS requests.

In order to perform such test, GeoLabs first provided a server infrastructure to host several instance of an identical system with exact same resources (chrooted environments), and to make them available for each WPS server. Gerald Fenoy from GeoLabs also actively participated in setting up the tests procedures and scenarios and a test suite based on the use of Apache benchmark was also made available for automating the performance tests. The testing scenarios were run against each WPS server and each type of request could be analyzed and compared. The OGC Compliance tools were also used to run conformity tests as detailed in the presentation. The detailed presentation of the benchmarking exercise and the results for the

GetCapabilities, DescribeProcess, Synchrone Execute, Asynchrone Execute requests scenarios are presented and compared in a WPS Benchmarking presentation available here.

This first step was important and allowed to detect, correct and optimize some of the core functionalities of the ZOO-Kernel server. Further, it was helpful for strengthening the foundations of the software and enhance its general capability, stability and conformity. The very satisfying results obtained by ZOO-Project for both performances and conformity tests confirmed that the software foundation is strong and stable enough to introduce new optimization and to implement scalable processing functionalities.

3.3 TOWARDS THE SUPPORT OF WPS 2.0

As stated in the Description of Work, another crucial aspect of the research related to the Processing Engine is to implement the needed changes in ZOO-Kernel in order to respect the WPS 2.0 specification and to provide a fully compliant Processing Engine to the PublicaMundi platform.

According to ZOO-Project source code actual status and a detailed study of the WPS 2.0 draft specification, GeoLabs identified that the two main missing features were the newly introduced 'GetStatus' and 'Dismiss' parameters. The first one has already been introduced in the ZOO source code as a WPS service written in C language allowing to poll a WPS Execute Process and to return the progress of the computation under the form of its status values. The source code of this service is currently being integrated as a core feature of ZOO-Kernel server which will soon provide a proper GetStatus request implementation. The Dismiss missing request is also actually tackled directly at the core of the WPS server but necessitates more development effort to be properly implemented. ZOO-Kernel needs to be able to cancel or 'dismiss' a process and delete the corresponding allocated resources to fully support this parameter and this implies additional core modifications which are already planned for the next ZOO-Project release.

Once the support of both the GetStatus and Dismiss requests will be effective and released, ZOO-Project will be candidate for the WPS 2.0 conformance tests and compliancy. This will subsequently benefit to the PublicaMundi Processing Engine which will soon implement the most recent WPS specification.

3.4 PERFORMANCE AND SCALABILITY ENHANCEMENTS

An important part of the research was also accomplished regarding the WPS server scalability during several development phases which occurred during the first year. More stability and robustness were first obtained by adding numerous code optimizations in order to provide a "leak-free" ZOO-Kernel. There are very few memory leaks when using latest versions of ZOO-Project and this brings better and faster results.

Other performance enhancements were introduced by defining methods allowing to compile and use the ZOO-Kernel server as FastCGI. The latter was compared with the traditional CGI environment and provides much faster results for both GetCapabilities, DescribeProcess and Execute requests. Once the FastCGI was supported, every ZOO-Service was also tested and executed as FastCGI and the source code of some of them was corrected and optimized for FastCGI. This work was somehow very healthy for the collection of ZOO Services which could benefit of more tests and optimizations.

At last but not least, some additional in depth modifications have been carried out in order reach even more scalability. Some caching mechanisms are indeed currently being implemented at the core of ZOO-Kernel allowing to save/cache WPS requests efficiently in the first hand, and to load WPS Services in memory in the other hand. The latter drastically improves response times and allows invoking the ZOO-Services in a much scalable way.

The various performance and scalability related enhancements are finally being tested on the project infrastructure, using several ZOO-Kernel at the same time and scenario running numerous WPS concurrent requests. The results of such development are very promising and will be released in the next ZOO-Project release and will subsequently benefit to the PublicaMundi Processing Engine.

3.5 CKAN WPS EXTENSION

The previously presented results related to the Processing Engine and its exploitation within the PublicaMundi platform lately allowed GeoLabs to tackle another underlying objective of WP4. A WPS extension for the CKAN software was indeed initiated in order to provide a bridge between the Data Catalog and the Processing Engine.

The so called CKAN-ext WPS indeed brings WPS capabilities to the CKAN powered open data catalogs by introducing the possibility of adding an

existing WPS server as a new CKAN resource, from the CKAN administration panel. The support for WPS getCapabilities, DescribeProcess and Execute requests support was implemented using the Python language and the CKAN logic in order to create a suitable extension. The Processing Engine can thus be directly exploited from CKAN and this allows to directly process open data and create, store and reference new information directly from the Catalog. Every WPS service returned by the GetCapabilities request can then be configured through appropriate CKAN forms (i.e generated according to the DescribeProcess response document) and Executed using any suitable CKAN geospatial resource available from the CKAN Raster and Vector data stores as input data. The output data finally generates a new CKAN resource and add it to the targeted data store.

The first version of the CKAN-ext WPS source code is available in the project repository at https://github.com/PublicaMundi/ckanext-wps. Further improvements are still under development and their release is already planned for the next phase of the project.

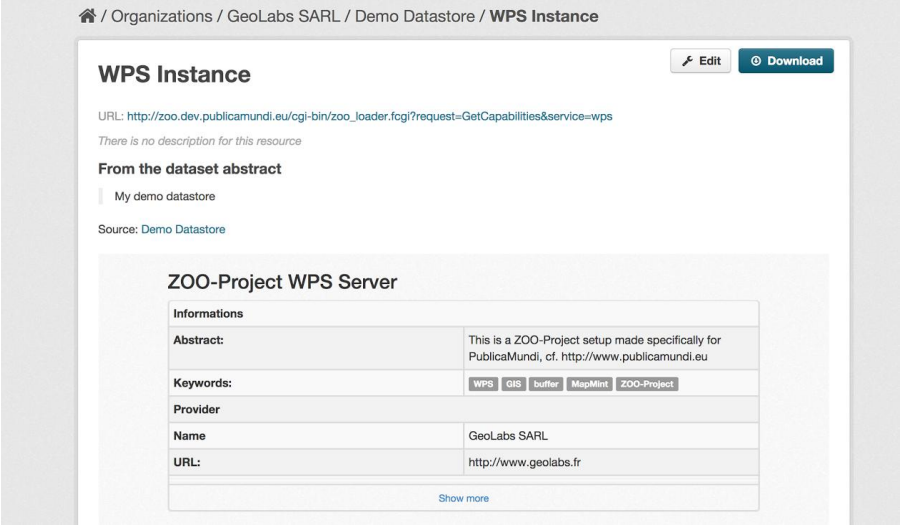

Figure 1: Linking the Processing Engine to the CKAN open data catalog