THE POWERFUL WPS PLATFORM

Venkatesh RAGHAVAN, OCU
Gérald FENOY, GeoLabs
Nicolas BOZON, 3LIZ

GWF 2011 Hyderabad, India
WHAT IS WPS?

What does WPS means?

Why Pay for Software?

Not really ....

but it is something similar to that if you think twice after the end of my talk!!!
WHAT IS WPS?

**Web Processing Service** is designed to standardize the way that GIS algorithms are made available on the Internet.

WPS specifies a mean for a client to request the execution of a spatial calculation from a service.

WPS intends to automate geoprocessing by employing geospatial semantics in a service-oriented architecture (SOA).

Open Geospatial Consortium interoperability standards since 2005

WPS Standard was published and accepted as Standard in 2007

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform
Short WPS Overview

Web Processing Service defines three requests:

- **GetCapabilities** (as almost all the OWS Standards)
  Provides a full overview of all the processes available on the requested server.

- **DescribeProcess**
  Provides for a unique set of services Identifiers the specific description of a set of Service.

- **Execute**
  Provides a way to run the Service and ask for the ResponseDocument or RawdataOutput.
WHY USING WPS?

Web Processing Service allows to deploy and to orchestrate advanced GIS processes on the server-side.

WPS can connect to any cartographic engine and spatial databases and thus drive complex spatial data infrastructures (SDI).

WPS is a generic and standardized way to use GIS on the Cloud.

WPS can run FOSS4G and OSGeo tools!
WHAT IS ZOO?

ZOO is a **WPS (Web Processing Service)** compliant server-side platform based on OGC's WPS 1.0.0.

ZOO is **open source software**

ZOO is designed to create and chain web processing services easily, using **FOSS4G libs or existing code**.

ZOO is based on a C Kernel (**ZOO Kernel**) able to load dynamic libraries and to orchestrate Webservices coded in several programming languages.
ZOO PROJECT GOALS

Make FOSS4G libs communicate in a standardized way using WPS

Make GIS-based Web Services deployment and chaining easier and faster

Create a ready-to-use Webservices suite based on stable libs

Enable advanced Web GIS functionalities such as:

- Conversion, reprojection and processing of data
- Advanced GIS algorithms
- External spatial-based scientific models
Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform
ZOO-Project PSC Members

- Nicolas BOZON (3LIZ), FR
- Maria BROVELLI (Politecnico di Milano), IT
- Massimiliano CANNATA (SUPSI), CH
- Gerald FENOY (GeoLabs), FR (Chair)
- Hirofumi HAYASHI (AppTech), JP
- Daniel KASTL (Georepublic), DE/JP
- Jeff McKENNA (Gateway Geomatics), CA
- Markus NETELER (Fondazione Edmund Mach), IT
- Venkatesh RAGHAVAN (Osaka City University), JP
- Satoshi SEKIGUCHI (AIST GEO Grid), JP
HOW DOES ZOO WORK? (1)

ZOO Kernel handles and chains ZOO Services

A ZOO service is composed of:

- A metadata file `.zcfg` (Title, Metadata, Inputs, Output…)

- A Service Provider: « Service Shared Object » (SSO) (Dynamic library, Python modules, JAVA Class, PHP script …)

WPS GetCapabilities and DescribeProcess requests requires only to parse .zcfg file using Flex and Bison

ZOO Kernel is able to load SSO dynamically, to extract specific functions and to execute them, answering the WPS Execute requests

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform'
Does this look like mess to you? Please Take a look

HelloWorld.py

def HelloPy(conf, inputs, outputs):
    outputs['output1'] = {'value': 'helloworld' + inputs['Name']['value'], 'datatype': 'string'}
    return SERVICE_SUCCEEDED

HelloWorld.zcfg overview

[HelloWorld]
Title = Get the HelloWorld string.
serviceProvider = test_service
serviceType = Python
<DataInputs>
    [Name]
    Title = HelloString
    <LiteralData>
        DataType = string
    </LiteralData>
</DataInputs>
<DataOutputs>
    [output1]
    Title = HelloString
    <LiteralData>
        DataType = string
    </LiteralData>
</DataOutputs>
HOW DOES ZOO WORK? (2)

- **Execute request** parsing (XML / KVP) to fill internal data structure

- **Xlink:href checking**, conditional data download to provide input data value

- **Dynamic loading** of the Service Provider Shared Objects

- **Specific service function call**, passing internal data structure by reference

- **ResponseDocument / Raw data output**, using the previous data structure (modified by the service itself)
SUPPORTED LANGUAGES

**ZOO Kernel** supports several programming languages

- **C / C++** Native support
- **Python** Optionnal support (Python interpreter)
- **Fortran** Optionnal support (F77, F90)
- **PHP** Optionnal support (PHP embedded)
- **Java** Optionnal support (Java SDK)
- **Perl** Optionnal support (Perl interpreter)
- **Javascript** Optionnal support (SpiderMonkey)
**ZOO Kernel** supports several natural languages:

- English
- French
- Japanese

Workshop Material available on the ZOO-Project web site in English, Japanese and Italian

http://www.zoo-project.org/
ZOO PROJECT STATUS(1)

ZOO 1.0 released under MIT/X-11 license since April 2010:

- ZOO Kernel ( C )
- ZOO Services ( C and Python)
- ZOO API ( Javascript )

svn checkout http://svn.zoo-project.org/svn/trunk zoo

ZOO 1.0 documentation and packaging in progress

Install guide and ZOO Workshop available on the wiki
ZOO 1.0 available on OSGeo-Live since version 4.0
ZOO PROJECT STATUS(2)

ZOO Services

Ogr2Ogr (C)
GEOS/OGP (C)
GdalTranslate (C)
GdalGrid (C)
OpenOffice Document Text Converter (Python)

http://zoo-project.org/trac/browser/trunk/zoo-services/

ZOO API

Based on SpiderMonkey and ZOO's optional JavaScript support
Uses server-side JavaScript and a Proj4js adaptation
Add Logic in WPS chaining
ZOO PROJECT PLATFORM

Mapserver (WMS, WFS, WCS, SOS)
+ Apache Server (HTTP)
+ Open Office Server (ODF)
+ ZOO Kernel (WPS)

ZOO OWS Platform
ZOO PROJECT STATUS
Simple Vector Features operations OGR Based

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform'
ZOO PROJECT STATUS
Simple Voronoi triangulation CGAL Based

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform'
ZOO PROJECT EXPERIMENTS(1)

WFS-T support implementation  (Fenoy, G.)

MapServer 5.6.5
ZOO 1.0
WFS-T proxy

Provides a Transactional support to MapServer WFS
Edits the input data source geometry and attributes
Allows any data source input supported by OGR

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform'
ZOO PROJECT EXPERIMENTS(2)

GRASS GIS implementation (Gebbert, S.)

GRASS GIS 7
ZOO 1.0
ZOOGrassModuleStarter.py

Fenoy, Bozon, Raghavan, 2011. 'ZOO: The powerful WPS platform'

www.zoo-project.org
OSM import (ZOOSM) (Delluchi, L.)

PythonOSMApi
OGR python
ZOO 1.0
ZOOSM.py

Provides a new way to import data into OSM using WPS
Could be used for massive imports and updates in OSM
Could be extended to Export from OSM in OGR supported formats
Drift-X WPS  (Bozon, N. and Fenoy, G)

ZOO 1.0
driftx.f
GdalTranslate
Services.py

Provides a webmapping application for Pesticide atmospheric dispersion calculation.

WPS for scientific computing
goGPS  
(Realini, Yoshida, Hayashi and Venka)

GPS Data processing to obtain accurate positioning from raw GPS observations.
ZOO PROJECT NEXT STEPS

Supported languages:

- C# (.NET)
- Ruby

What about using ZOO-Project without Web Server?

- Mozilla XPCOM Technology = one dev step to get both Server side and Desktop application, using Services locally or remotely!

Making Documentation better:

- Publishing the Workshop Material
Merci beaucoup pour votre temps
ありがとうございました
THANKS FOR YOUR TIME
zoo-discuss@gisws.media.osaka-cu.ac.jp

Many thanks to FOSS4G to make this story happen!!!!