An HTML tool for exploiting geospatial web services

Theofilos Papadopoulos[1] and Konstantinos Evangelidis[2]
1. Research Associate, priestont@gmail.com
2. Associate Professor, kevan70@gmail.com

Technological Educational Institute of Central Macedonia
Department of Civil Engineering, Surveying Engineering and Geoinformatics
Terma Magnisias str. 62124, Serres, GREECE

Abstract

In the present work we made an attempt to develop a generic end user web interface with typical GIS functionalities, based on the specifications of selected OGC standards, with the ability to be executed locally as a simple html application on the client side. The tool is developed on top of open Javascript libraries and exploits already available Geospatial Web Services. The user is capable of parameterizing the capabilities provided by geospatial service providers and visualize the results of the submitted requests. This software prototype may act as a forerunner for future complex geospatial applications providing high level capabilities for putting together different Geospatial Web Services to satisfy specific geo-related user needs.

Keywords: Geospatial Web Services, OGC WFS, WMS, Javascript, Openlayers

1 Introduction

Geospatial web services (GWS) are becoming the dominant technology for data sharing and exchange between geospatial stakeholders. The prevailing GWSs standards have been introduced by the Open Geospatial Consortium (OGC) during last decade, and include the web map service (WMS) [1], the web feature service (WFS) [2] and the web coverage service (WCS) [3] for sharing, the catalog service for the web (CSW) [4] for discovering, and also the web processing service (WPS) [5] for processing, geospatial data. In this respect, numerous research projects and business solutions rely on the above standards to achieve geospatial data interoperability between custom applications and satisfy project-specific needs.

Most applications are nowadays based on Web Services, use data provided over the Web or generated by users [6], and are executed on cross-platform browser-based interfaces. In the geospatial community, GWSs and XML-based open geospatial data formats, such as Geography MarkUp Language (GML), and Keyhole MarkUp Language (KML) have become basic components of desktop and web GIS software solutions. For example the ESRI's ArcGIS commercial product supports WMS connections through its popular "Add data" interface[1]. On the other side QGIS open solution also supports connection to GWSs through appropriate plug-ins. Of course they both support geospatial data interoperability. For individual web-based applications it is possible to develop a custom GIS capability through open Javascript libraries (e.g. Openlayers[2]) and have it executable in the browser without the need of installing anything on the client-side but an updated modern browser.

In the present work we have developed a simple web interface that provides principle GIS functionalities exclusively based on open standards and services. The tool is an HTML page locally launched making use of Javascript capabilities to perform a) data discovery from public data providers, b) layer-based data view, c) data selection by attributes and d) feature data acquisition and preview.

2 Review of Existing GWS Implementations

As mentioned above GWSs have been incorporated into known desktop GIS solutions. As a result it is already a routine job to add WMS and WFS services into existing map projects. For example ESRI's ArcGIS 9.3 version supported satisfactory OGC services[3] in 2010, while at the same time QGIS released stable versions of Python plugins offering WMS/WFS Clients[4]. The above services could be also

---

[1] http://desktop.arcgis.com/en/arcmap/10.3/map/web-maps-and-services/adding-wms-services.htm
[2] http://openlayers.org/
[3]
http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Adding_a_WFS_service_layer_to_a rcmap
[4] https://plugins.qgis.org/plugins/wfsclient/

incorporated in custom desktop GIS applications developed under the APIs of the above GIS packages.

On the server side, it has become an easy, wizard driven, procedure to publish WMS and WFS services. In ArcGIS Sever for example[5], since 9.3.1 version, a WFS service could be created, parameterized and accessed through, for example, web interfaces developed in .NET framework. In the open-source geospatial software community, OSGeo[6], GeoServer[7], is considered as a reliable OGC compliant server implementation for publishing WMS/WFS/WCS standards. Being an open solution, it is accessible by open web interfaces developed on top of open Javascript libraries such as OpenLayers.

Many significant GWSs implementations concern the WPS standard such as that of 52º North[8] an open source software initiative in the field of Geoinformatics, PyWPS[9] and ZOO[10], open WPS platforms which are incubated in OSGeo foundation. A complete list of recent WPS implementations is listed by [7].

Having mentioned the above, one realizes the maturity of utilizing GWSs in various GIS developments [9], [10], [11], [12], [13], [14], however, it seems that an integrated environment including all stages of a GIS development project, from data discovery and acquisition to data processing and manipulation remains a worth of dealing with, task.

3   Design and Development

3.1   Functional Requirements
The major functional requirement of the proposed software prototype web interface is to exploit GWSs. Therefore a request and response interaction between the user and the available OGC web services (e.g. WMS, WFS, WPS) takes place. Various types of data should be the result of the above interaction as shown in figure 1 and described below:

---

[5] http://webhelp.esri.com/ARCGISSERVER/9.3.1/dotNET/index.htm#wfs_service.htm

[6] http://www.osgeo.org/

[7] http://geoserver.org/

[8] http://geoprocessing.demo.52north.org/

[9] http://pywps.org/
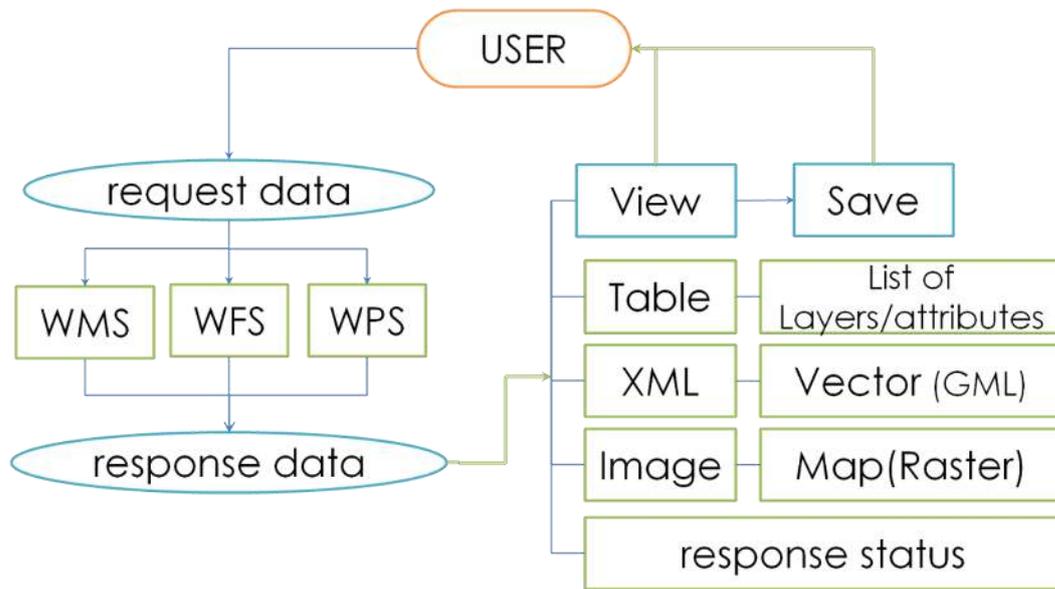
[10] http://www.zoo-project.org/

Figure 1: User interaction and data type results

- Tabular data containing the available layers offered by the spatial data providers. In a higher level of detail tables may contain attributes of selected layers
- XML data containing features in GML encoding
- Images representing maps of the available raster data

Details of the server responses should be separately provided in order to identify server response failures, errors in request syntax etc.

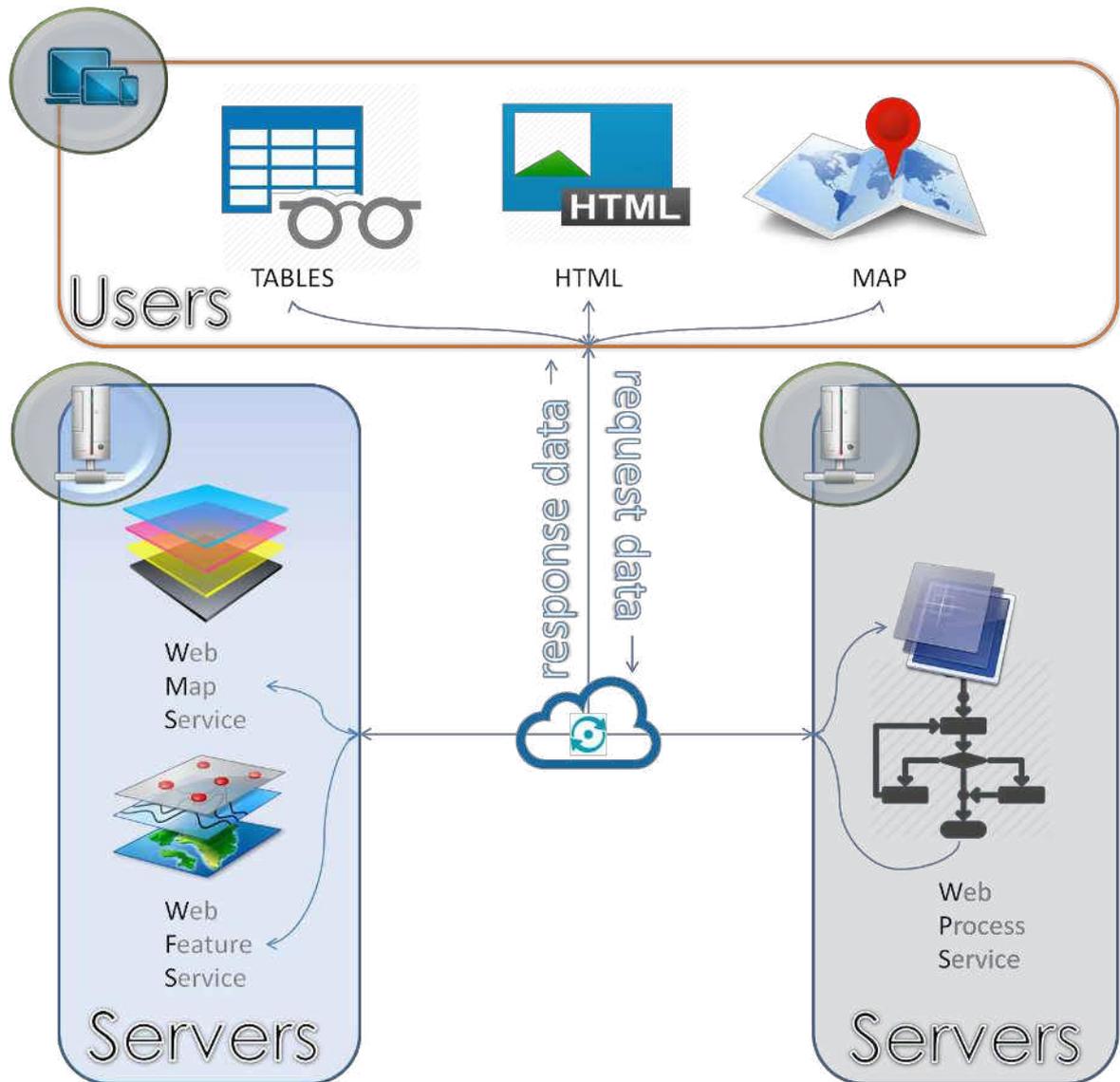Figure 2 represents graphically the above interactions and data types.

Figure 2: Functional architecture of a system exploiting GWSs

3.2 Software Prototype Design

To achieve a typical desktop GIS-based "look and feel" interface, five discrete operational areas, each one represented by the respective interface pane were defined, as shown in Figure 3. A brief description of the processes and interactions occurring in every pane is depicted in Table 1 and analyzed in the following paragraphs:

Table 1: Interface panes of the proposed software prototype

Pane	Comment
Main tools	The main tools of the application (settings, raster layer, vector layer, upload file)
Data interaction	Pop-up windows to interact and manipulate data
Layers	Saved layers
Data visualization	Data visualization panel with visualization tools and map view
Messages feedback	Feedback messages from the application (e.g. logging errors)

Figure 3: Panes - Operational areas of the proposed software prototype

3.2.1 Main Tools

Main tools pane offers typical capabilities for configuring and managing the spatial content and more specifically:

- declaring the service providers
- requesting a map (raster layer) through *WMS/GetMap* request
- requesting features (vector layer) through *WFS/GetFeatures* request
- uploading a layer
- saving/removing a layer to/from the local environment
- zooming to map/full extend
- defining the under underlying texture

3.2.2 Data interaction

Data Interaction pane displays paged tables of the involved spatial data, containing valuable information with most significant ones a) lists of offered layers from the service providers and b) lists of layers' attributes.

At this pane, two data interaction capabilities are offered:

- The capability of previewing a selected layer (*Preview Command*) over a list of either raster or vector layers: the available layers are retrieved directly from the XML file returned after the *WMS-WFS/GetCapabilities* requests
- The capability of applying filters in the attributes of a selected layer (*Filter Command*): the attributes of the layer are retrieved directly from the XML file returned after the *WMS/GetMap* or the *WFS/GetFeatures* requests

3.2.3 Layers

The layers retrieved in the interaction pane remain in the remote servers and they are dynamically rendered in the user interface through GWSs OGC requests. This pane provides the capability of creating a separate local environment of selected geospatial datasets copied in the local disk.

3.2.4 Data Visualization

At Data Visualization pane the user can view data at their default format (e.g image, kml) if the 'Preview' tab is selected. At this tab, multilayer view is not available so the user can view one layer at a time. If the 'Map' tab is selected, using the openlayers 3 library a map interface is created and can be manipulated by the user. The map is using Bing maps as base layer.

Over the map and next of the tabs there is a toolbar that perform action to manipulate layer data and the map. The user can add a layer from the cloud into the software and then it can be accessed via the Layers pane, or remove it. When at 'Map' tab the user can select a layer and zoom to the layer extend or change the order of the layers and change the z index of the layers to bring one above another.

3.2.5 Messages Feedback

Because of the software being developed web based it uses requests and responses. Any request on the internet may fail for many reasons so the user must be informed each time what action the software is performing and what was the result status (success or fail). Also, the data which the server responds may include errors that the software is not aware of. For these reasons there is a message pane that inform the user what is going on in the software.

Working with WMS and WFS is not an easy task for the common user owning to the fact that it requires basic knowledge of handling XML files. On the other hand, paged

tables are best for reading and manipulating big data and for this scope is the best tool to explore layers that many servers provide together.

3.3 Software Demo Development

For demonstration purposes the following Geospatial Servers were declared:
1. The ArcGIS online sample server (http://sampleserver1.arcgisonline.com/) offering WMS service
2. The Boundless demo Geoserver (http://demo.boundlessgeo.com/geoserver/web/) offering WMS/WFS services

The implementation of the proposed software prototype passed through a series of tasks developed in corresponding methods as follows:

3.3.1 Get XML data

In order to get all the available data from the declared servers an ajax request is created with the 'GetCapabilities' REST Service GET method
The Messages pane displays the following information:

```
[27-04-2016|20:38:30] Request data from:
http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_U
SA/MapServer/WMSServer?request%3DGetCapabilities'
[27-04-2016|20:38:30] Request data from:
http://demo.boundlessgeo.com/geoserver/wms?request%3DGetCapabilities'
[27-04-2016|20:38:32] Response HTTP status 200 [OK] from:
http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_U
SA/MapServer/WMSServer?request%3DGetCapabilities'
[27-04-2016|20:38:34] Response HTTP status 200 [OK] from:
http://demo.boundlessgeo.com/geoserver/wms?request%3DGetCapabilities'
```

The response data is an XML file from each server.

3.3.2 Create JavaScript layer objects

An XML file is not the easiest form of data to manipulate with JavaScript, and for this reason the layers are transformed into JavaScript objects with properties based on the OGC standard for WMS [8].

3.3.3 Create paged tables

After creating an object that stores the layers and their properties (e.g. bounding box, EPSG etc.), a paged table is constructed and can be viewed and manipulated via the *Data interaction pane*. Each row of the table is a layer and includes an inner table as tree view that stores the layer details. The user can manipulate the details (e.g. change

the EPSG, change the background color) and a differently parameterized REST Service is created. Figure 4 depicts an instance of the table for the attributes of layer named "2" offered by the 1st server.

3.3.4 Visualize the preferable layer

The result of the user request previously parameterized may be an image in case of a WMS/GETMap request or an XML file in case of a WFS/GetFeatures request. The server responds accordingly and in the first of the above cases an image (raster layer) if the format of the layer at the request is set of that type, is displayed at the *Data visualization pane*, as shown in figure 4.
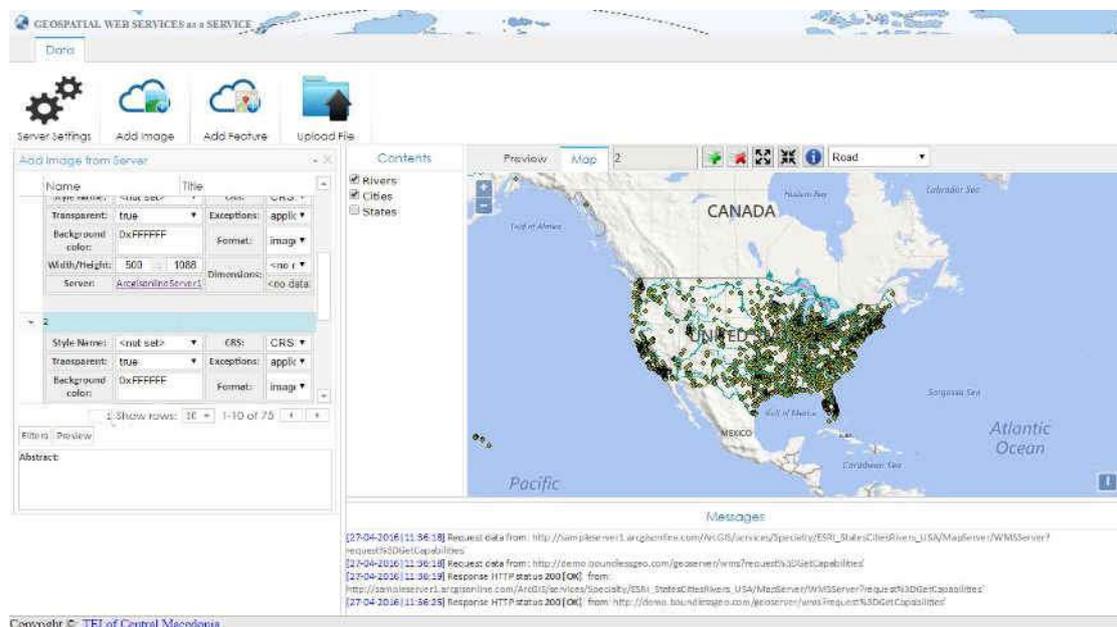

Figure 4: A Demo Implementation of the proposed software prototype

4   Further Developments

The presented development provides a simple interface with layer-based viewing functionalities over map and feature servicing capabilities of public geospatial servers. The end user only needs an Internet browser in order to execute a simple HTML web-page and explore the available geospatial data. Depending on the capabilities offered it is possible to perform simple filtering tasks.

For specific geospatial processes it could be of great interest to develop individual specialized HTML tools by composing only the appropriate geospatial web services

and incorporating the required functionality to perform the desired geo-processing tasks.

Collecting all available services generates an obvious challenge of the presented work: to further extend the present development into a fully functional Cloud-GIS environment offering complex geospatial processes.

5 Acknowledgements

6 References

1. de la Beaujardie` re, J., 2004. Web map service (WMS). Version 1.3. OGC 04-024. Open Geospatial Consortium, Inc., 85pp.
2. Vretanos, P., 2005. Web feature service (WFS) implementation specification. Version 1.1.0. OGC 04-094. Open Geospatial Consortium, Inc., 131pp.
3. Evans, J., 2003. Web coverage service (WCS). Version 1.0.0. OGC 03-065r6. Open Geospatial Consortium, Inc., 67pp.
4. Martell, R., 2004. OGC™ catalogue services—ebRIM(ISO/TS 15000-3) profile of CSW. Version 0.9.1. OGC 04-017rl. Open Geospatial Consortium, Inc., 87pp.
5. Zhao, P., Foerster, T., & Yue, P. (2012). The geoprocessing web. Computers & Geosciences, 47, 3-12.
6. Haklay, M., & Weber, P. (2008). Openstreetmap: User-generated street maps. Pervasive Computing, IEEE, 7(4), 12-18
7. Evangelidis, K., Ntouros, K., Makridis, S., & Papatheodorou, C. (2014). Geospatial services in the Cloud. Computers & Geosciences, 63, 116-122.
8. OGC Inc., 2006. OpenGIS® Web Map Server Implementation Specification, Table: 7, pg:32 Online, available at: http://portal.opengeospatial.org/files/?artifact_id=14416
9. Granell, C., Díaz, L., & Gould, M. (2010). Service-oriented applications for environmental models: Reusable geospatial services. Environmental Modelling & Software, 25(2), 182-198.

10. Foerster, T., Schaeffer, B., Brauner, J. & Jirka, S. (2009). Integrating OGC Web Processing Services into Geospatial Mass-market Application. IEEE GeoWS 2009

11. Stollberg, B. and Zipf, A., 2007, OGC Web Processing Service Interface for Web Service Orchestration – Aggregating Geo-Processing Services in a Bomb Threat Scenario, pp. 239–251 (Cardiff, UK: Springer).

12. Lapierre, A., & Cote, P. (2007, October). Using Open Web Services for urban data management: A testbed resulting from an OGC initiative for offering standard CAD/GIS/BIM services. In Urban and Regional Data Management. Annual Symposium of the Urban Data Management Society (pp. 381-393).

13. Meng, X., Xie, Y., & Bian, F. (2010). Distributed Geospatial Analysis through Web Processing Service: A Case Study of Earthquake Disaster Assessment.Journal of Software, 5(6), 671-679.

14. Evangelidis, K., Ntouros, K., & Makridis, S. (2012). Geoprocessing Services over the Web. In Proceedings of the 32nd EARSeL Symposium, Mykonos, Greece (pp. 344-349).