

ECP 2005 GEO 038026 EGN

EuroGeoNames

Report on potential Web Client Application

Deliverable number	<i>D-7.5</i>
Dissemination level	<i>Public</i>
Delivery date	<i>14 January 2009</i>
Status	<i>Final</i>
Author(s)	<i>Stéphane Haby, Jochen Schach</i>



eContentplus

This project is funded under the *eContentplus* programme¹,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

¹ OJ L 79, 24.3.2005, p. 1.

Table of Contents

1	PURPOSE OF THE DOCUMENT	3
2	ABBREVIATIONS	3
3	SYSTEM OVERVIEW OF THE REFERENCE APPLICATION I	4
4	DEVELOPMENT OF THE REFERENCE APPLICATION – LESSONS LEARNED	5
4.1	WFS INTERFACE	5
4.2	GOOGLE WEB TOOLKIT	6
5	EUROGEONAMES FUNCTIONS	8
6	CONCLUSION: POTENTIAL CLIENT WEB APPLICATION	9

1 Purpose of the document

In order to showcase the functionality of the EuroGeoNames (EGN) infrastructure and services, the so-called Reference Application was developed in WP7 "EGN Web GIS Reference Application" (RA). This application illustrates the functions of the EGN Central Service.

This paper describes the experiences and insight gained from the development of the RA. It should help web application developers to benefit from our lessons learned. The authors intend to transfer these experiences about some rather technological aspects and about the use of the EuroGeoNames Central Service (CS) functions needed for the various fields of application. The discussion result in recommendation for a potential web client application that may succeed the RA.

The developers experience of using the CS' WFS interface and the experience to use the GWT for web development are described in chapter 4. In which extend the EuroGeoNames functions that have been elaborated in WP2 are available in the CS will be discussed in chapter 5. Chapter 6 lists recommendations for a potential web client application.

2 Abbreviations

Abbreviations used in this document are listed and explained below, in alphabetical order.

API: Application Programming Interface

CS: EuroGeoNames Central Service

CSS: Cascading Style Sheets

EGN: EuroGeoNames

ERM: EuroRegionalMap

GWT: Google Web Toolkit

RA: EuroGeoNames Web GIS Reference Application

WFS: Web Feature Service

3 System overview of the Reference Application

The Reference Application is implemented as Web Application using the Google Web Toolkit (GWT). GWT is an open source Java development framework that allows developing AJAX applications in the Java language using Java development tools.

The following figure shows the architecture of the Reference Application and its interfaces to other systems.

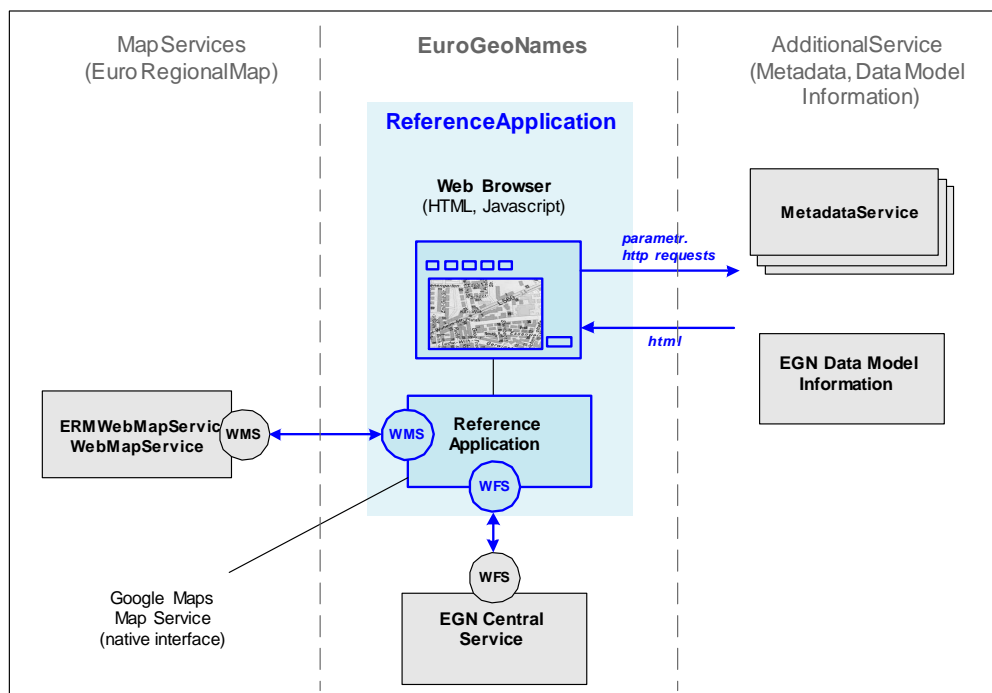


Figure 1: System Architecture

The WFS interface is used to query the EGN Central Service. It is a crucial element of the entire EGN architecture as it is the only communication mean that an application developer can use to query the data.

For more information about the system and its functions, please read the Document D7.1 “Concept and requirements for the EGN Web GIS Reference Application” and the document D7.4 “Documentation: EGN Web GIS Reference Application”.

4 Development of the Reference Application – Lessons Learned

4.1 WFS Interface

The OpenGIS Web Feature Service Interface Standard (WFS) defines an interface for specifying requests for retrieving geographic objects across the Web using platform-independent calls. The WFS standard defines interfaces and operations for data access and manipulation on a set of geographic features. Among many possibilities WFS offers, in the context of the project only the “Getting” or “Querying” of features based on spatial and non-spatial constraints is used. The queries are formulated in SQL-like language encoded in XML. The specified feature encoding for input and output is the Geography Markup Language (GML)

WFS provides a simple standard for serving geolocated vector data such as points and polygons. The request is expressed entirely in the URL, and the response is delivered as a XML. WFS can provide extensive information for each feature instance including identifier, location, time, elevation, URL (i.e. reference for further information), and any number of additional data attributes. As a lot of GIS vendors support the WFS interface, it is getting quite popular in web application development.

For developers it is relatively easy with WFS services. One reason is that the WFS specification clearly describes the requests and responses that a WFS service should support. This way it is possible to 'decouple' client and server and e.g. build an application-specific Web client that still can communicate with (open source or commercial) server software developed by others. Another reason is that WFS uses standard web technologies as HTTP and XML (GML and WFS-requests/responses). Common web technologies can be used, like Servlets/Java Server Pages for application logic.

As WFS usage increases, several performance issues have become apparent. A request for a large number of feature instances can return a prohibitively large result, exhausting network or memory resources. This happens with a rather low number of results: You have to take into account that a part of the results are geometries where i.e. every point of a borderline having an X and Y coordinates with ten digits needs to be transferred from the client to server.

For those developers not familiar with WFS and for those that don't want to get too deep inside WFS possibilities, the Reference Application's tab “Query” may be used to obtain the syntax of certain calls and the format of the results. For simple use cases, these examples are absolutely sufficient to succeed with a rapidly created implementation.

Experiences with EGN Central Service

Knowing the CS implements the WFS Server of deegree (deegree - Free Software for Spatial Data Infrastructures, <http://www.deegree.org>), we decided to use the deegree WFS Client as we wanted to assure to exploit rapidly and easily the object of the CS. This was a good decision: we never had any difficulties on that level.

Performance issues were coming up only at a later state of the development as the data volume was not available earlier. We needed to optimize the queries in order to alleviate the effects. After these enhancements, the response times were in the range where we had wanted

them, at least as to the part of the WFS. It is in this context important to know that the RA only receives a maximum of 50 records. But the performance problem is not only due to the WFS technology. The complexity of the transferred objects is another cause: Even if the client application would only need to process the coordinate(s) of the place name, it has to read and process the full description having information about feature classification, language, gender and others in it.

What we missed for the development was the possibility to make a request that only returns the number of hits for a specific request. Having such a function available at the CS would help developers to implement a smarter treatment of the results according to the number of hits.

Apart from the performance issues, that can be solved through further system optimisation on both client and server side, the interface was well appropriate for our purpose.

4.2 Google Web Toolkit

Google Web Toolkit (GWT) is an open source Java development framework that lets you escape the matrix of technologies that make writing AJAX applications so difficult and error prone. With GWT, you can develop and debug AJAX applications in the Java language using the Java development tools of your choice. When you deploy your application to production, the GWT compiler translates your Java application to browser-compliant JavaScript and HTML.

The GWT development cycle is like this:

- Use your favourite a Java integrated development environment to write and debug an application in the Java language, using as many (or as few) GWT libraries as you find useful.
- Use GWT's Java-to-JavaScript compiler to distill your application into a set of JavaScript and HTML files that you can serve with any web server.
- Confirm that your application works in each browser that you want to support.

Experiences in developing with GWT

That fact that we could use the same tools than we were used to in the pure Java Development was one of the major reasons to chose GWT. With this choice, we could get a very good JavaScript solution without the pain of dealing with the cross-browser JavaScript coding. But we still needed to invest into a fair amount of Cascading Style Sheets (CSS) configuration to create nice look and layout of the RA. Layout problems different browser came up frequently. As the code is entirely JavaScript and you always have to recompile in order to have a new version, solving those issues was sometimes a bit more time consuming than with a simple HTML page, but the more experience we got with the RA's layout, the better we knew how to cope with layout issues.

One conceptual problem of AJAX problems remains the fact that everything is executed on the client side, which means that there a lot of data transfer is needed between the client and

the server. It is also GWT that is generating the code, which on the one hand side is intended, but considering performance issues is a disadvantage as we have less configuration options to make things faster.

As we made a ranking of the results on the client, we had to interpret the entire result XML to determine the best results and display the first page of the result list. The interpretation of the result XML was very time consuming and we didn't have any means to accelerate this process.

Another disadvantage due to the general concept of GWT is the configurability with property files. If you want to change a text or add a language, it would be, at least in Java, easy to do so: All the user interface items that change frequently would be stored into property files and so such usual changes would be implemented very rapidly (change text in configuration file, than restart application server). With GWT, each change in these files requires a new compilation of the entire projects to come into effect. The process is heavier and so changes need to be organized well.

5 EuroGeoNames functions

The goal of the research done in WP 2 was to analyse the market in terms of potential applications and related business actors and stakeholders from both public and private sectors. The work package provides a detailed description of user and businesses that may use the EGN Central Service. The result of the analysis was one major input for the development of the data model and associated services.

In this chapter, the potential application functions elaborated in WP2 will again be briefly discussed against the background of the technical components resulting from the EGN project.

Function / description	Availability for Web Application developers / experiences with usage
<p>Place name normalisation A check to verify whether the place name has been spelled properly.</p>	<p>As analysis of WP2, this function was one the most important the CS should offer. The CS hat two functions to allow “fuzzy” search: The WFS’s “like” and the search option called “SoundsLike”. Both are tools that can be used for the normalisation. The RA integrates both.</p> <p>The fuzzy search implemented with “like” some frequent cases of normalisation, but further rules should be implemented (i.e. replace if double consonants with simple ones in a normalized search string).</p> <p>“SoundsLike” is a good conceptual basis, but needs a lot of configuration (and before maybe even some research) to work really with all languages.</p>
<p>Place name translation Checking how a place name is written in another language.</p>	<p>The “AlternativeGeographicIdentifier” links endonym and exonyms. This is enough to identify translations of a place name. Already implemented in the RA.</p> <p>If translation is the main focus of a potential usage of the CS, the potential application should offer a special funtion “translate” that only outputs place names in defined languages and applies translation if relevant.</p>
<p>Place indexing Checking the position of a place within an administrative hierarchy.</p>	<p>Possible with the name properties "parentGeographicIdentifier" and "childGeographicIdentifier". Currently no data is available in the central service.</p> <p>The information about the place names and the functions the CS offers is sufficient for a lot of use cases but it needs some special “business logic” in a potential web application if you really want to exploit the value of this information</p>
<p>Geocoding Looking up the coordinates of a place.</p>	<p>Function available and integrated in the RA.</p>
<p>Geoindexing Looking up places located near a specific place.</p>	<p>Function available through bounding box search. RA currently doesn't implement it. Would be relevant i.e. in Hotel reservation applications.</p>
<p>Reverse look-up Finding what place or address is valid at a specific location</p>	<p>Function available in the Central Service . RA currently doesn't implement it.</p> <p>The information about the place names and the functions the CS offers is sufficient for reverse look-up, but it needs some special “business logic” in a potential web application if you really want to exploit this function.</p>

6 Conclusion: Potential Client Web Application

As stated in chapter 5.3 of the document D2.1 – Final report on user/business requirements, most commercial applications make use of their own Gazetteer Service & Database. The use of external services is not very popular yet. One major reason for this: the gazetteer database always has to be complete in respect of the area covered by the application and the data has always to be up-to-date. So even if the table in chapter 5 shows that most of the functions are already available, potential application developers always will check first whether the aforementioned data quality is achieved before they start to analyse how they can use the EGN and its functions for their own projects. So achieving a good data quality and coverage shall be the main focus of EGN.

The CS offers already in its first version the variety of functions that was elaborated as being necessary in WP2. Only as to the function “normalisation”, there is direct need to continue the development. The others are already fully operational.

All those functions that can be accessed via WFS. Being a protocol using standard technologies such as http and XML, it can be easily used by application developers even not very familiar with geographic information. The fact that geometric data is big and that xml is easy interpretable has the price of heavy web traffic and calculation time for both the CS and potential applications. Application developers should take this into account by creating intelligent client applications that only request and process data when really needed.

As use cases vary considerably as to the functions they demand, we recommend to have a few small applications that offer specific functionalities rather than a single powerful central application. These applications should be easily integrated into other web applications (i.e. “EGN Translation application” as “mashup” for applications such as a hotel booking system or a travelling portal).

The GWT is an appropriate toolkit for application developers. Java developers will appreciate it but have to take into account that the “real” code is generated by GWT and hence some optimisation possibilities will be lost. But the advantages outweigh the limitations due to the GWT concept: We would anytime use it again.