A GIS-aware Architecture for Tourism Portals

Bernd Müller
University of Applied Studies
and Research
Business Informatics Dept.
38855 Wernigerode, Germany
bmueller@hs-harz.de
Phone +49 3943 659240
Fax +49 3943 199

Harald Wehr
ITGIS
Application Development
38855 Wernigerode, Germany
harald@itgis.com
Phone +49 3943 557807
Fax +49 3943 557808

Björn Voigt
ITGIS
Application Development
38855 Wernigerode, Germany
bjoern@itgis.com
Phone +49 3943 557807
Fax +49 3943 557808

Keywords: Web application architecture, tourism portal, GIS, open source.

Abstract

Cutting-edge tourism portals have to integrate GIS information. However, GIS data formats differ from usual tourism portal data formats. In this paper we propose a web application architecture which uses GIS as a main information resource and therefore diverges from standard web application architectures. GIS data as well as relational data is wrapped into XML and processed by an XSLT processor. Further, the system is built completely on open source software.

1 Introduction

Tourisms portals include many different data formats, mainly relational formats. We propose an architecture based on spatial data provided by geographical information systems (GIS). This kind of data needs very different processing compared to standard relational data. However, the additional expenses seems to be worthwhile because questions like "which are the nearest restaurants to my hotel?", "how long does it take me to the nearest public transport stop?" or "what point of interests are on my trip today?" can be answered easilly.

The architecture we propose was developed in a research program sponsored by the German Federal Ministry of Education and Research, called MobiHarz (mobile in harz, a region in the middle of Germany). It is based completely on open source components, uses XML as the only internal format and is thereof HTML-free. Algorithms are implemented in Java.

2 The overall Architecture

The overall architecture of the MobiHarz system is depicted in figure 1. It consists of many different GIS and non-GIS software components. The heart is Cocoon [2] from the Apache group, which we describe in the next section. Cocoon is augmented with many different sub-systems. The most important systems are the XML parts. But also all business objects like point of interests, restaurants and accomodations live as Java objects in Cocoon. The mapping of the business objects to relational databases is done with Castor [3], an object-relational mapping tool. Departure times of public transport is gathered by HTTP-Unit [7] and integrated into MobiHarz. Mapserver [8], a CGI-Server is responsible for map generation and described in section 6. All generated HTML-pages can also be requested as PDF. The PDF generation is accomplished by FOP [4], also an Apache group project. Some graphics are generated as scalable vector graphics (SVG) and rendered as JPEG for HTML inclusion by Batik [1], one more time a Apache group project.

Some important parts of the system are now explained in more detail.

3 Cocoon

The heart of the system is Cocoon, a publishing framework written in Java. Cocoon can be used as an ordinary Java program via the command line interface but is typically deployed as a servlet in a servlet container. It is therefore responsible for the whole communication with the users. As Cocoon is based on strict separation of content, logic and style integration of external components is easy.
Figure 1: The MobiHarz System Architecture
This is very important as the framework does not contain any GIS functionality from scratch.

The main concept of Cocoon is based on pipelines. According to an incoming request the appropriate pipeline of the system is called. Each pipeline consists of a required generator, optional transformers and finally a required serializer. A generator produces the desired output as normal XML-document. It aggregates content from the filesystem, databases, HTTP-requests or other sources.

Following the pipeline the generated output can be transformed via certain XSL transformations. Finally, a serializer changes the temporary SAX stream to the desired output, usually HTML, XHTML or WML.

One service MobiHarz provides is to calculate a shortest path walking route between two points of interests. A map with this route is build dynamically and included in the HTTP-response. This is also true for an elevation profile of this route. How is this elevation profile generated and integrated into a web page? First of all the user starts a shortest path search between two points of interest. The spatial vector data contains all information about streets and trails for a specific area. The package OpenJGraph [11] is used to build an undirected, weighted graph with these information. The result of the application of Dijkstra's shortest-path algorithm is a set of edges connecting the two points of interest. These edges are temporarily stored in a database identified by the session key of the user's HTTP session. This result set is taken by the profile generator to generate an SVG-document containing the elevation information of the calculated path. Cocoon finally transforms this SVG data into a bitmap file that can be referenced as an image in a web page. This is done by one of the implemented serializers e.g. the SVG/PNG-Serializer.

Of course there are other use cases in the depicted web application that integrate spatial information. Here only a few are mentioned:

- Possibility to geo-code locations of hotels, restaurants and other point of interests via HTML pages,
- generation of maps showing shortest paths between points of interests,
- searching for relevant point of interests within specified distances.

Cocoon can be seen as a mediation between the users spatial interests and different backends described in the next sections.

4 Grass

The Geographic Resources Analysis Support System (Grass, [6]) is one of the biggest and most powerful GIS in the open source world. The program consists of a hundred of modules to process raster, vector and point information. GRASS is completely integrated into the MobiHarz architecture but it is not accessed directly during an incoming user request.

It is rather used to generate, administrate and maintain the spatial information used in MobiHarz:

- Combination of different raster layers (e.g. streets, rivers, forests, residential areas) to produce maps of certain needs,
- Digitizing of vector information that cannot be purchased from external sources,
- Interpolation of grid elevation data to raster information in order to build continuous terrain models.

Grass is currently under heavy development. In future versions it will support PostGIS implementation from OGC's SQL specification. An improved vector library as well as improved database interfaces will manifest Grass as a real alternative to commercial GIS solutions.

5 PostgreSQL/PostGIS

PostgreSQL is an object-relational database system that is enhanced by the PostGIS project [12] to store and retrieve geographical objects. A PostGIS-enabled database is able to store relational attributes as well as spatial properties of objects. Enhanced SQL functions provide a lot of useful methods of analyzing spatial information. In our architecture this database enhancement is used for the following tasks:

- Storage of points that have spatial references like bus and train stops, hotels and all POI,
- Storage of digitized line information like streets and hiking trails,
- Temporary storage of calculated shortest paths to enable an automatic generation of relevant images and elevation profiles.

In the near future PostGIS integrates some important functionality from the GEOS project [5]. Doing this, the implementation comes closer to the mentioned OpenGIS Simple Features Specification for SQL. After that it is planned to commit the implementation for compliance testing to the OGC.
MapServer

MapServer [8] is responsible for dynamic generation of image maps within the MobiHarz architecture. The program can be used as external library within several programming languages like Perl, PHP, Python and Java. Nevertheless it is mostly used as CGI script. The script offers enhanced functionalities and is configured by simple request parameters. Figure 2 shows the steps that are performed when a user searches for a route between two geographic points.

Following steps are performed to give the user the result to his query:

1. The incoming user request is matched by the responsible pipeline of the Cocoon servlet. Start and end point are transferred as HTTP parameters. The user gets a session that identifies him through several HTTP connections.

2. Using these parameters the Dijkstra algorithm of the OpenJGraph package performs a shortest path search4. Each user query results in a minimum spanning tree for the specified points.

3. The result is used to save the set of lines that connect the two points. The identifying key is an aggregation of the user’s session key and current system time.

4. As MapServer is running as CGI script it can only be configured via certain HTTP parameters. This is done by the package HttpUnit. It establishes a HTTP connection between the server Cocoon is deployed on and the web server where MapServer is running.

5. According to the transferred parameters MapServer fetches the information about the shortest path from the database. The program is able to read the PostGIS format.

6. Additionally it reads further raster images produced by Grass. Finally the result image is generated. It can consist of several layers: a normal image map in the background, the shortest path for the query, relevant bus and train stops and further information layers. The image is stored into the file system of the server where MapServer is running.

7. As a result Cocoon gets an XML document about the generated image. It contains information about the image name, the generated image extent, activated layers and current zoom size of the image.

8. In this step Cocoon’s pipeline aggregates the different information about the user’s search into one XML document.

9. Finally this document is serialized as an XHTML page and sent to the user. The referenced image map is fetched by the browser through an additional HTTP connection. This request is simply matched by another pipeline of Cocoon.

MapServer is able to process a lot of different spatial data formats like TIFF/GeoTIFF, PNG, JPEG and several vector encodings like Shape-files or PostGIS information. The product is built on other free software like FreeType, Proj.4 or libTIFF.

7 GSM and GPS Location Based Services

MobiHarz includes LBS for mobile phones and PDAs. The mobile phone application is capable to search for the nearest hotels and restaurants as well as bus stops and railway stations. The technology is GSM cell-based. The provider Vodafone allows requests of geo-coordinates for logged in mobiles. However, because the size of cells can be very large the accuracy is not satisfactory. UTMS will increase accuracy.

We also use Compaq PDAs with GPS. We have built a mobile context-aware tour guide based on interactive Flash. Predefined tours can be selected and consumed in a very entertaining manner. The system is called MAX, the mobile area explorer and described in Müller [9]).
8 Integration of Foreign Data Sources

Many hotels have their own reservation system. Organizers of events (culture, entertainment, tourism) also have their own information systems. A tourism portal has to be able to connect to such systems. We have build a kind of plug-in technique to connect to foreign relational databases. It is based on XML configurations describing a mapping from our business object model to relational tables and vice versa. It is described in Wehr [13].

9 Conclusion

We have introduced a novel architecture for web applications dedicated to GIS data processing. The architecture was developed and realized in a research project sponsored by the German Federal Ministry of Education and Research. The system is called Mobiharz and can be tested at http://www.mobiharz.de. The project started in January 2001 and has reached its end in December 2003. All presented parts of the architecture are implemented and tested. The system behaves very well and was presented at CeBIT 2003. The System is implemented with open source components only, i.e. despite of its complexity and functionality no license fees became due.

Acknowledgments

The authors would like to thank Dirk Gronert for implementing so many Cocoon stylesheets.

References


