

Using WFS-T in combination with geolocation for online transactional editing of geographic features through mobile browsers

Yaowapa Pornsamutsin



Master's Thesis
Master of Science in Media Technology
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2010

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Using WFS-T in combination with geolocation for
online transactional editing of geographic features
through mobile browsers

Yaowapa Pornsamutsin

2010/06/20

Abstract

While online maps have been widely used in the society, the number of providers who develop applications to serve those users needs have also been increasing. To make use of data or exchange data from each provider, some standards are involved. The OpenGIS Consortium (OGC) is an association offering certain standards for geoprocessing which can be helped in developing and implementing software.

The purpose of this project is to use WFS-T in combination with geolocation for online transactional editing of geographic features through mobile browsers on mobile phones based on one of OGC standards called WFS (Web Feature Service), which WFS-T (Transactional Web Feature Service) is a part of WFS. With WFS-T, users can perform transactional operations such as create, update and delete geographic features. Geographic features can be described as a set of properties which may include a name, type and value. For example, one country can be defined as a geographic feature containing the following properties: $\{country_name, country_capital, country_lonlat, country_population\}$; each property can have different type and value such as *country_name* has a type as string and has a value as Norway while *country_population* has a type as number and has a value as 4882600.

In this implementation, GeoServer is selected to be used as GIS server software, PostgreSQL with PostGIS is used as datastore and OpenLayers is used in creating web-based geographic application. Additionally, W3C Geolocation API is used to detect users' locations from using their devices, in this case, an Android phone is used as a test device during the implementation. Once users launch their mobile browsers to connect to the Internet. The application will figure the users' locations and show the position as a point on a map through their mobile browsers. Moreover, the amenity places around their area will be displayed. The prototype allows users to make online transactions such as insert a new point as a new place. When users finish and save their work, all data will be recorded in the database via WFS-T operations.

The project report will give readers some background information about geolocation, the OpenGIS Standards including concepts and technologies of GIS and mobile browsers, how the prototype is created, and also the results of the implementation to provide information for those who are interested in using WFS-T to edit geographic features through web browsers on mobile phones.

Acknowledgements



Contents

Abstract	iii
Acknowledgements	v
Contents	vii
1 Introduction	1
1.1 Topic covered by the Thesis	3
1.2 Keywords	4
1.3 Problem description	4
1.4 Justification, Motivation and Benefits	4
1.5 Research Questions	5
1.6 Methodology	5
2 Background	7
2.1 OGC Web Services	7
2.1.1 WMS	9
2.1.2 WFS	9
2.1.3 GML	9
2.2 Mobile Browsers	9
2.3 Geolocation	12
3 Related Work	13
4 Research and Implementation	19
4.1 Investigated platforms	19
4.1.1 Server-side	19
4.1.2 Client-side	24
4.2 Implementation	27
4.2.1 Preparation	27
4.2.2 Testing	32
5 Results and conclusion	37
5.1 Results	37
5.2 Conclusion	39
6 Future Work	41
Bibliography	43
A GeoServer Issues	47
B PostgreSQL with PostGIS Issues	49
C Data preparation	51
C.1 Export data from OpenStreetMap (OSM)	51
C.2 Convert and Import data using FME Universal Translator	53
C.3 Manipulate and optimize data with PostgreSQL 8.4 / pgAdmin III	54

C.4 Create 'Workspace' and 'Store' in GeoServer	55
D OpenLayers Library	57
E W3C Geolocation API	61
F A client application	63

1 Introduction

If someone asks you what map is, how do you answer the question? You may think about an image of the world including some regions, each region contains many countries. Or, you may think of the shape of your country, pinning the city you live on that. Moreover, if you are a traveler, you may think that map is something that can provide you information about sight-seeings including how to be there. We can see that *maps are a powerful way of thinking about the earth* [1]. Figure 1 is the map drawn on birch bark presenting the migration legend of the native American called Ojibwe around 1820. It illustrates the lives of Ojibwe origins since they were born until moved to the new places which the spiritual realities had taken part. Moreover, it also flashes the essential of making maps related to their lives and thoughts as the symbols of significant spiritual guides were used along the route (from the right to the left).

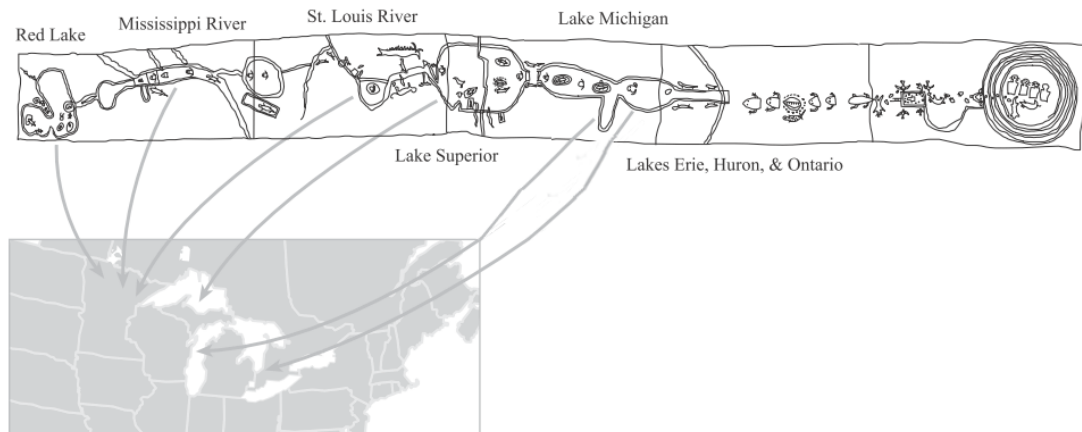


Figure 1: The Native map shows the migration legend of the Ojibwe ca. 1820 [1]

As time goes by, from drawing pictures on the rock in the past until now using maps on digital devices, the changes of technology have been taking part of how we understand the world including the ways to create maps. *GIS (Geographic Information System or Geographical Information System)* is the combination of cartography (maps) and database technology which we may be familiar with and use it in our daily lives but many of us do not realize about it. As the Internet today has played an important role providing us huge information, we probably find all information we want over the Web including spatial information. *Spatial Information* informs the physical location of objects and the relationship between objects. For simple example, when we are online and want to find directions from one place to the other places, we open Google Maps and input texts to get the directions and map. Nowadays, there are lot of providers who have developed and provided this kind of information in the market; *how can we combine data from*

different sources? As same as in our society that we have some rules we need to follow to make us can live happily together, in GIS world, to make one product/datum to be used with the others, the implementation has been developed by following the OpenGIS®Standards - OpenGIS® is a Registered Trademark of the Open Geospatial consortium, Inc (OGC). This organization is similar to the World Wide Web Consortium (W3C) which has developed Web standards; both OGC and W3C may be comprised of many organizations such as commercial, government, research, university and so forth. The difference between the OGC and ISO (International Organization for Standardization) is that ISO has provided of International Standards (supported by law) and its organization is only at the National level without membership by any organizations. The OGC can submit OGC standards for approval as ISO standards under the terms of the agreement with TC211, Geographic information/Geomatics [2]. Recently, the OGC standards baseline comprises 30 standards [3] and for some standards such as WMS, SF and GML are already complied with the requirements of ISO standard.

GIS allows us not only to create maps but also to manipulate important information on maps. Information, in this case, can be any things in the real-world, for instance, parks, rivers, schools, houses, and so far. Each category can be stored separately in different layers as shown in figure 2. When we move to a new house, we may want to know the nearby places. We open a GIS application and send a request to a server asking for a map of our area. Then, the server finds geographic data as we request, renders all together as one image and finally sends a respond back to us. This is an example of using *Web Map Service (WMS)*, one of the most famous OGC standards which has been widely used for some years. Anyway, while all we get from using WMS is a picture, using *Web Feature Service (WFS)* offers us a chance to do something more than just to look at the map; all we get from using WFS is actual vector data (as XML). It can be seen that WFS provides us greater control for the ways we are able to play with the raw geographic data including query data and/or manipulate (insert, update and delete) data. From the previous example, when we move to a new house, we may want to know nearby supermarkets. Then, we submit a query to a server. Once the server finds related data to our request, it compiles the data and sends it back to us as GML file (in this case, the GML file contains information about nearby shops). After that, the application validates the file and draws vector for us. As a result, we can see the nearby shops on a map. WFS basically supports read-only operations, WFS that allows us to insert, update and delete geographic features called *WFS-T (Transactional Web Feature Service)*. Once again, refer to the example earlier, when we get the filtered data by location (shops around our house) from the server, if our application supports WFS-T, we may be able to edit geographic features, such as add a new shop, via the application interface without the infrastructure requirements. Furthermore, at the present of time, there is a technology available called *geolocation API* which makes Web browsers can determine user's location based on the used devices. It can be noticed that this kind of technology is helpful in GIS application development since the browsers can detect user's location and then the GIS application can make use of the positioning of users. For example, in this case, if we use the web-based GIS application to find the nearby shops, once we open the program on Web browsers, we can see the requested places including our location pinning on the map.

Currently, there are many GIS applications available; while some are commercial, some are

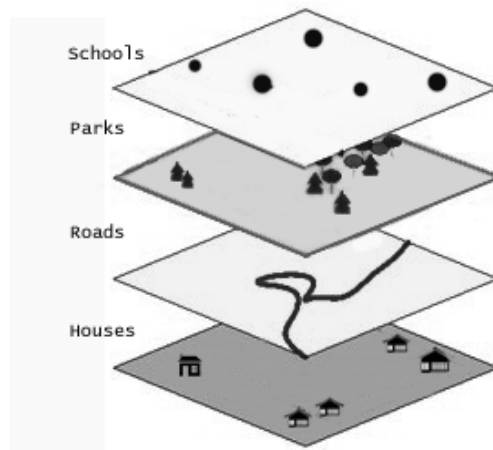


Figure 2: GIS consists of map layers

provided for free. Moreover, not only we are able to use those tools on powerful computers, but we are also able to use some tools on thin clients such as personal computers and mobile devices. As mobile devices have been involved in human communication, we can see that many people have their own mobile phones and use their devices not only to make or receive a phone call but also do somethings else. The modern mobile phones support many services such as MMS, MP3, camera, GPS, Internet, and so forth. At the present of time, many GIS applications developed for mobile phones have used GPS receivers of mobile phones to locate locations of users and displayed nearby attractions and facilities. Anyhow, not only GPS that can be used to provide user's position, but also location inferred from network signals such as IP address, MAC address, WiFi connection, and so on. And, as mentioned above about using Google Maps, if our mobile phones have GPS function and/or can be used to access to the Internet, no matter where we are, we will probably never get lost, imagine how easy our lives will be from making use of this kind of technology.

1.1 Topic covered by the Thesis

The purpose of the master's thesis is to review and try out the capabilities and features of Transactional Web Feature Service (WFS-T) in combination with geolocation in editing geographic features through mobile browsers based on one of those OGC standards called Web Feature Service (WFS). With WFS-T, users can do transactional operations such as create, update and delete geographic features. As in the project, the following issues are performed:

- Review of Web Feature Service Implementation Specification (OGC 04-094) version 1.1.0 mainly focusing on WFS-T section
- Investigate previous work about how WFS and WFS-T are used, what are advantages and disadvantages of implementing them
- Develop a prototype using OpenLayers library in combination with geolocation API to build a web-based geographic application which GeoServer is installed as a GIS server and Post-

greSQL with PostGIS is used as a datastore connecting to GeoServer

- Test editing of geographic features through mobile browsers and analyze the results

1.2 Keywords

Geographic Information Systems (GIS), GeoServer, OGC web services, WMS, WFS, WFS-T, Mobile browser, Geolocation

1.3 Problem description

Recently, there are numbers of open source GIS applications that offer for free. Comparing to proprietary software, the most advantage of using free software may be that it can help organizations to save their cost such as in licensing fees and upgrading fees. However, it may also provide some disadvantages if we choose the wrong one that is not fit to our projects.

And nowadays, *Web Map Server (WMS)* is one of the most important services that has been widely used in GIS systems. It allows users to request for geographic data which will return results as map images. Many GIS tools support WMS, while many of them do not support WFS-T. However, one good obvious issue of WFS-T is that it supports not only all operations of a basic WFS which allows users to get raw geographic features over the Internet, but also a transaction operation which allows users to modify geographic features. This benefit makes some providers and developers concern to implement WFS-T in their systems. Recently, there are some applications that already support WFS-T, by the way, many of them are not be able to run on mobile devices. So, in this project, we are going to research about how WFS-T in combination with a new technology like geolocation can be used together to perform editing of geographic features via mobile browsers on mobile phones and investigate advantages and disadvantages from implementing it.

1.4 Justification, Motivation and Benefits

From the introduction and problem description above, since there are various GIS applications provided for free as Open Source software, it is more comfortable for us to think about how we can develop the prototype because we do not need to concern much about the implementation cost in case that we choose to use Open Source tools.

At the present of time, many research papers have involved in WMS area. While some of them have been related to WFS, few of them have mentioned about WFS-T especially how WFS-T can work on mobile phones. Additionally, geolocation is a new technology released for awhile and becomes helpful in GIS this days including in developing applications for mobile devices.

We can see that mobile phones have played important roles in our society which most people have their own mobile phones. Not only we can use our mobile phones to talk or to send text messages to our friends, but many modern mobile phones provide a service to access to the Internet. So, if we can make use of this kind of technology in combination with WFS-T, it will help to provide some useful information for those who are interested in using and/or implementing WFS-T in combination with geolocation on mobile devices and will probably be able to enhance functions and value of developing the applications in the near future.

1.5 Research Questions

Based upon the background and motivation and the problem definition, the following research questions will be examined through the project:

«RQ-1» Is it be possible to implement WFS-T in combination with geolocation as a web application used on mobile browsers?

«RQ-2» What are the main challenges to make use of WFS-T in combination with geolocation on mobile phones?

«RQ-3» To what extent are map and geographic features can be displayed on mobile browsers?

1.6 Methodology

The idea of WFS-T is not new but somehow it is still not widely used while geolocation API is new and seems quite useful in the development of GIS applications especially for mobile phones. In this project, we are interested to implement something related to WFS-T and can work on mobile browsers. Not only the literature study is made, but also the prototype is developed. The literature study provides us some useful information about WFS-T and relevant issues in which we can adapt it with our system such as in planning and developing steps.

To make it more understandable, the methods we use to implement this project can be presented based on research questions as described below:

«RQ-1» *Is it be possible to implement WFS-T in combination with geolocation as a web application used on mobile browsers?*

To answer this question, both literature study and the prototype are made. Researching available technology and related papers give us the clear picture of how WFS-T and geolocation works including the requirements we need and also the issues we should concern in implementing them. In this case, for example, Web Feature Service Implementation Specification is reviewed and as a result, we get some requirements and also supported operations of WFS (which are discussed later in Related Work). This steps provide us significant background and lead to develop an implementation plan. And then, we can create a prototype based on the plan and once we perform a test, the answer to this question is revealed.

«RQ-2» *What are the main challenges to make use of WFS-T in combination with geolocation on mobile phones?*

Creating and testing a prototype definitely help us to answer this question because we can experience unexpected problems and difficulties during the implementation.

«RQ-3» *To what extent are map and geographic features can be displayed on mobile browsers?*

We are able to see how good geographic data can be shown on the screens when we perform a test. So, to answer this question, implementing a prototype is a must.

After we set goal of the project, to build our own GIS system, we need to investigate the current environment with available GIS tools in the market, for instance, we may want to have a GIS server, a spatial database with geographic data, and some GIS tools (in this project, we mainly

concentrate on using open source software). The investigations of platforms are made. After that, the test environment is setup with the provided geographic data from different (free and well-known) sources. The whole point of a mashup, a combination of features and/or data from more than one source to develop a new useful service, is to make use of available GIS technology to provide a web-based application allowing users to do online transactional editing of geographic data through their mobile browsers. After the testing, the results are analyzed and the conclusion is made.

2 Background

2.1 OGC Web Services

In recent years, there has been an explosion of web mapping applications which allows users to access to huge amounts of geographic data via their web browser. The operations to perform the processes online are called Web Service. Since the area of this kind of implementation seems big and there are many providers who have developed and provided web mapping applications in the market, to make one product/datum to be used with the others, the implementation has been developed by following the OpenGIS®Standards - OpenGIS® is a Registered Trademark of the Open Geospatial consortium, Inc (OGC). OGC has defined interfaces and protocols for supporting online interoperable solutions which its goal is to develop standards that enable interoperability of spatial information and services. Recently, the OGC standards baseline comprises 30 standards such as *Web Map Service (WMS)*, *Web Feature service (WFS)*, *Geography Markup Language Language (GML)*, *Keyhole Markup Language (KML)* and so on. Figure 3, SEWilco describes how OGC standards help GIS tools to communicate by using geoservices server with interfaces and applications sketch [4]; green represents read and write paths while dotted arrowed line indicates mostly read-only data flow.

Currently, there are more than 250 organizations from all around the world including government, private sector, and academic organizations, that have joined the OGC; the followings are some examples of the OGC's members which are well-known: *Google, ESRI, Microsoft, ORACLE, NASA, IBM, MIT and so on*. In the GIS world, especially in the area of interoperable web mapping, OGC web services are very meaningful because they provide the standardization and opportunities for users to share geospatial information even though data is from different sources and/or different applications. For instance, refer to the earlier scenario in the previous chapter, *we move to a new house and want to search for nearby shops*, we are not required to understand the background processes of how we can get the returned data, anyway, what is going on can also be described based on figure 3 as followings:

1. *We launch a GIS application, which created by using OpenLayers, via a web browser*
2. *We submit a request asking for nearby shops*
3. *The request is sent to different servers*
 - 1) *a server that is responsible for providing map images (as WMS), is asked for a map image*
 - 2) *a server that is responsible for providing geographic features (as WFS), is asked for information of nearby shops*
4. *Servers find relevant data and send responds back to the client application*
 - 1) *a server that is responsible for providing map images (as WMS), sends a map layer*
 - 2) *a server that is responsible for providing geographic features (as WFS), sends a GML file*
5. *The application processes the returned results*

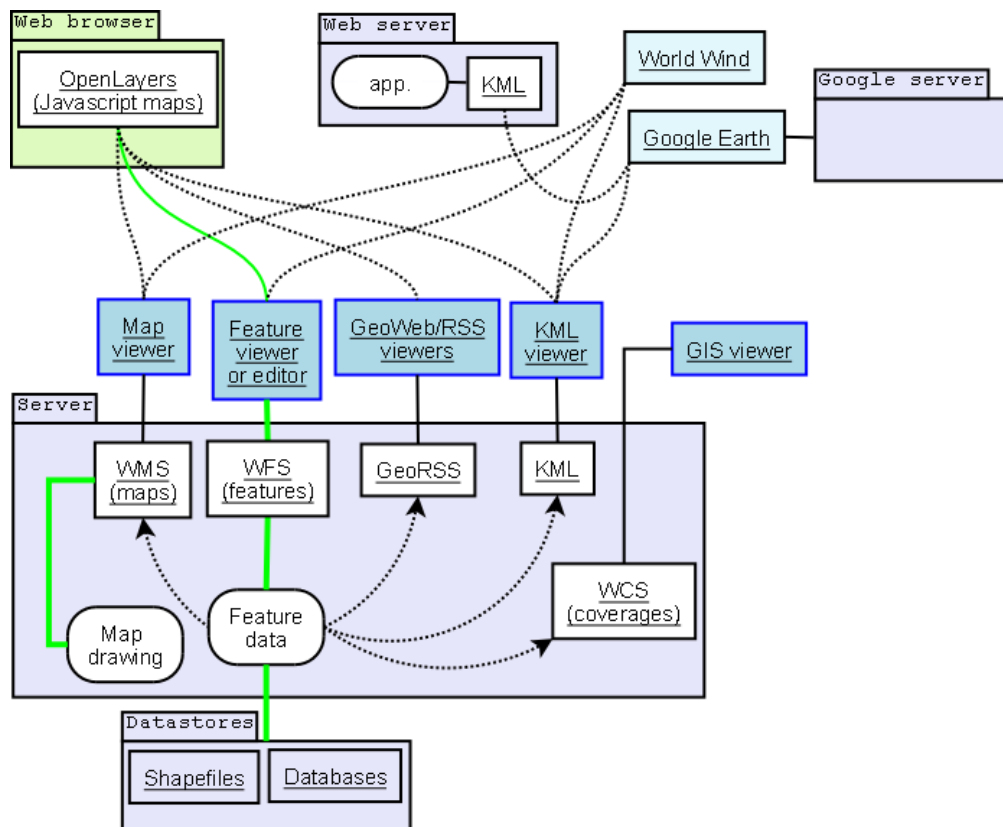


Figure 3: An example of how OGC standards help GIS tools communicate by SEWilco. [4]

1) displays a map image from WMS

2) validates GML and draws vectors, in this case, nearby shops are presented as points

6. The map and the nearby shops are displayed

We can see that the OGC web services, in this example: WMS and WFS, help to solve the problem of the lack of interoperability. With the use of open standards, we are able to manipulate spatial information no matter how we get it or where it is from.

2.1.1 WMS

OpenGIS Web Map Service (WMS) Implementation Specification is one of the most well-known Implementation Specifications by OGC. It offers a simple HTTP interface for map images from distributed geospatial databases. Once a client send a request by specifying the area of interest to the server, the service produces a content containing geographic information within the area that the client requests, and then send the response back to the client as map images (in the format of JPEG, GIF, PNG, or SVG). The client can easily view such images of the map through a web browser without needing to have any knowledge about how the data is stored or produced.

2.1.2 WFS

OpenGIS Web Feature Service (WFS) Implementation Specification is the main geospatial web service that provides an interface allowing requests for geographic features across the Internet no matter which platforms the clients are. Similar to WMS, once a client send a request to a web feature server asking for geographic data, the server will provide geospatial feature data and send a response back in GML format. More details about WFS will be described in the next chapter.

2.1.3 GML

Geography Markup Language (GML) is an XML grammar for describing geographic information. It is well developed by the OpenGIS Consortium to express geometries and geographical relations serving transaction operations over the Web. GML is also know as ISO19136 [5]. GML can be divided into 2 parts including a *GML document* and a *GML Schema*. This results to users and developers to be able to specific geographic features including points, lines and polygons.

2.2 Mobile Browsers

As same as web browsers on personal computers, web browsers designed to be used on mobile devices called `mobile browsers`. Currently, mobile devices have been involved in our daily lives, many applications have been developed to be supported on mobile phones. By the way, each application has its own limitation about supported platforms. Figure 4 illustrates smartphone operating systems and their market shares [6]. In 2006, users were mainly interested in using 5 operating systems including Symbian, Windows Mobile, RIM, Linux and Palm OS respectively. Symbian accounted more than 50 percent of the market and has been acting as a leader of mobile operating systems until the year of lately survey (2009). By the way, we can obviously see that its market share has decreased while the market shares of RIM, iPhone and Android have increased

significantly year by year.

In addition, each mobile operating system supports different mobile browsers. Table 1 shows default browsers used by major phone and PDA vendors. We can see that many of them are proprietary software; only *Android Browser* and *Firefox for mobile* are FOSS (Free and Open Source Software).

	2006	2007	2008	2009	2010
Symbian	67	63.5	52.4	46.9	?
RIM	7	9.6	16.6	19.9	?
Windows Mobile	14	12.0	11.8	8.7	?
iPhone	0	2.7	8.2	14.4	?
Linux	6	9.6	7.6	4.7	?
Palm OS	5	1.4	1.8		
Android			0.5	3.9	?
WebOS				0.7	?
Windows Phone 7					?
Bada OS					?
MeeGo					?
Other OSs	1	1.1	2.9	0.6	?

Sources: Canalys, 2006. Gartner: 2007, 2008, 2009.

Figure 4: Smartphone Operating System Market Share Percentage [6]

Browser	Creator	Current layout engine	Software license	Notes
jB5 Browser	Comviva	jB5 Browser Engine	proprietary	Linux, Symbian, Windows Mobile, and Brew platforms. jB5 Profiles addresses all segments of phones - Feature phone and Smartphone.
Polaris Browser	Infraware Inc.	Lumi (Ver. 6.X) WebKit (Ver. 7.X)	proprietary	Nokia, Samsung, LG Electronics, KYOCERA and other Smartphone and cellular phone in USA, China, Korea, etc
Kindle Basic Web	Amazon.com	NetFront	Proprietary	-
Android Browser	Google	WebKit	Apache 2.0 and GPL v2	-
WebOS Browser	Palm	WebKit	proprietary	-
BlackBerryBrowser	Research in Motion	Mango	proprietary	-
Firefox for mobile	Mozilla	Gecko	MPL 1.1 or later, GNU GPL 2.0 or later, GNU LGPL 2.1 or later	Currently released for Nokia Maemo and in development for Android
Internet Explorer Mobile	Microsoft	-	proprietary	-
Iris Browser	Torch Mobile Inc.	WebKit	proprietary	Acquired by Research in Motion - No longer supports Windows Mobile or Linux
Myriad Browser (Previously Openwave Mobile Browser)	Myriad Group	Fugu (Next version to use Webkit)	proprietary	Acquired from Openwave in 2008
NetFront	ACCESS Co., Ltd.	NetFront	proprietary	-
Nokia Series 40 Browser	Nokia	WebKit	proprietary	-
Obigo Browser	Obigo AB	-	proprietary	100% owned by Teleca AB
Opera Mobile	Opera Software	Presto	proprietary	Capable of reading HTML and reformat for small screens, installed on many phones on iPhone and iPod touch
Safari	Apple Inc	WebKit	Proprietary	Renders Flash 10, Ajax and Silverlight content.
Skyfire Mobile Browser	Skyfire	Gecko	proprietary	Currently supports Windows Mobile 5/6.x, Symbian S60 3rd & 5th Edition platforms
Netsailor Browser	Fantalog Interactive		proprietary	Convergence Web Browser for the expression of Multi-media in Korea.
uZard Web	Logicplant Co., Ltd.	MoRDAC (Mobile oriented Remote Display and Control)	proprietary	on Samsung, LG Electronics and other

Table 1: Default browsers used by major mobile phone and PDA vendors [7]

2.3 Geolocation

Geolocation is a combination of the words "geographic" and "location". It is the labeling of the geographic location of an object in the real world. Geolocation is a process that helps to determine where a user is located by making use of a user's device. For example, when a user uses her computer to connect to the Internet, geolocation can tell the position of the user by a user's IP address. Additionally, if a user connects to the Internet through her mobile phone, geolocation can figure her location by using a *network provider*, and/or a *GPS provider* [8]. Name of a network provider can be used to determine location based on availability of cell tower and WiFi access points which results are retrieved by means of a network lookup. On the other hand, name of a GPS location provider can be used to identify position using satellites.

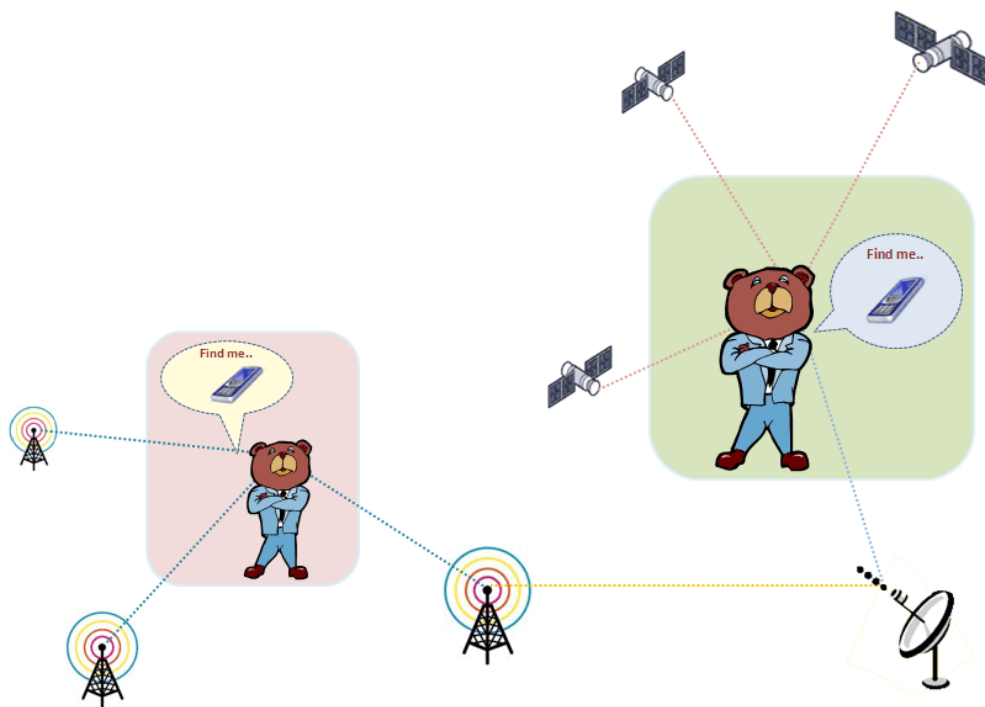


Figure 5: When you connect to the internet through your mobile browser, geolocation can figure your location by using a network provider or a gps provider

3 Related Work

At present, we can see that there is a high competition in the software development market. Many new tools have been released, while some tools have been out of date. Not only a proprietary or commercial software that always have new features/patches for users to upgrade versions, but also a (free) open source software that has been developed and released regarding users' demands. What is a (free) open source software? and why it is interesting to be used?

According to the Free software Foundation (www.fsf.org), software can be labelled as *free software* if the associated license conditions fulfil the "free software definition", which grants four freedoms:

1. The freedom to run the program, for any purpose
2. The freedom to study how the program works, and adapt it to your needs
3. The freedom to redistribute copies so you can help your neighbor
4. The freedom to improve the program, and to release your improvements to the public, so that the whole community benefits

From the definitions above, we come up to the opposite of *free software* which is *proprietary software*. While *free software* respects the users' freedoms, the *proprietary software* identifies ownership. The main benefit of using free software is that it can help users to save their budgets.

In the GIS world, there has been rapid growth in the number of open source GIS applications which they can be classified as software libraries, web servers, desktop GIS, spatial databases and GIS viewer [9]. While a lot of research groups have established their stand in the open world, users have to consider whether they can use data from different sources.

To make standard solution in GIS software developing, the Open GIS Consortium (OGC)

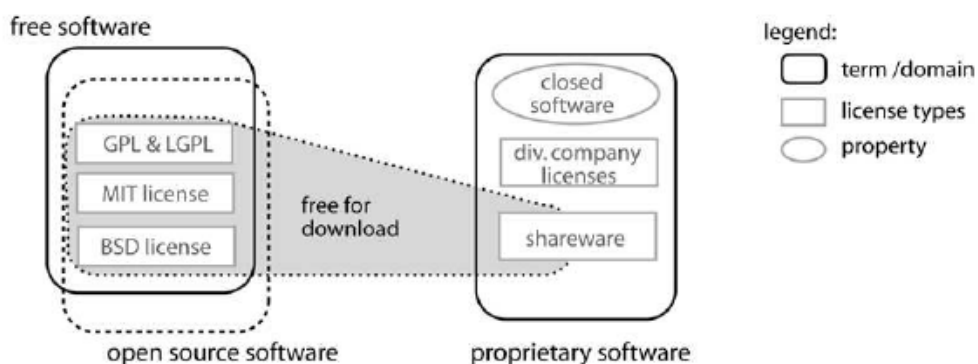


Figure 6: Terms used with respect to software licenses [9]

provides two programs (OGC Specification Program and OGC Interoperability Program) [10] for the planning, developing, reviewing, and last of all, officially adopting the specifications. In this project, Web Feature Service Implementation Specification will be involved [11].

The Open Geospatial Consortium, Inc (OGC) is an international industry consortium of 388 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards. – <http://www.opengeospatial.org/ogc>

For the project implementation, we will mainly focus on performing transactional operations since we are curious that why WFS is not used through a wide area as WMS. WFS-T is a part of WFS, the full WFS operations support INSERT, UPDATE, DELETE, LOCK, QUERY and DISCOVERY operations on geographic features using HTTP as the distributed computing platform [11] [12]. The protocol to be followed in order to process web feature service requests, the following steps will proceed [11] [13]:

1. A client application would request a capabilities document from the WFS.
2. A client application (optionally) makes a request to a web feature service for the definition of one or more of the feature or element types that the WFS can service.
3. The client application generates a request based on the definition of the feature types.
4. The request is posted to a web server.
5. The WFS is invoked to read and service the request.
6. When the WFS has completed processing the request, it will generate a status report and hand it back to the client. In the event that an error has occurred, the status report will indicate that fact.

Moreover, to support transaction and query processing, the following operations are defined:

1. **GetCapabilities:** When a WFS gets this request, it answers with a description of its capabilities.
2. **DescribeFeatureType:** A WFS can be asked to describe the structure of any of the feature types it serves.
3. **GetFeature:** A WFS can be asked to deliver features, however, the client should be able to constraint the query both spatially and non-spatially, and also specify what feature properties he/she wants to receive from the WFS.
4. **GetGmlObject:** A WFS can be asked to retrieve element instances by traversing XLinks that refer to their XML IDs. In this case, the client should be able to specify whether nested XLinks embedded in returned element data should also be retrieved.
5. **Transaction:** This request is composed of operations that modify features including create, update, and delete operations on geographic features.
6. **LockFeature:** This feature ensures that serializable transactions are supported by processing a lock request on a feature while a transaction on that feature is being process.

Figure 7 illustrates a communication between a WFS server and a client which HTTP is used to handle requests (as already mentioned above in details).

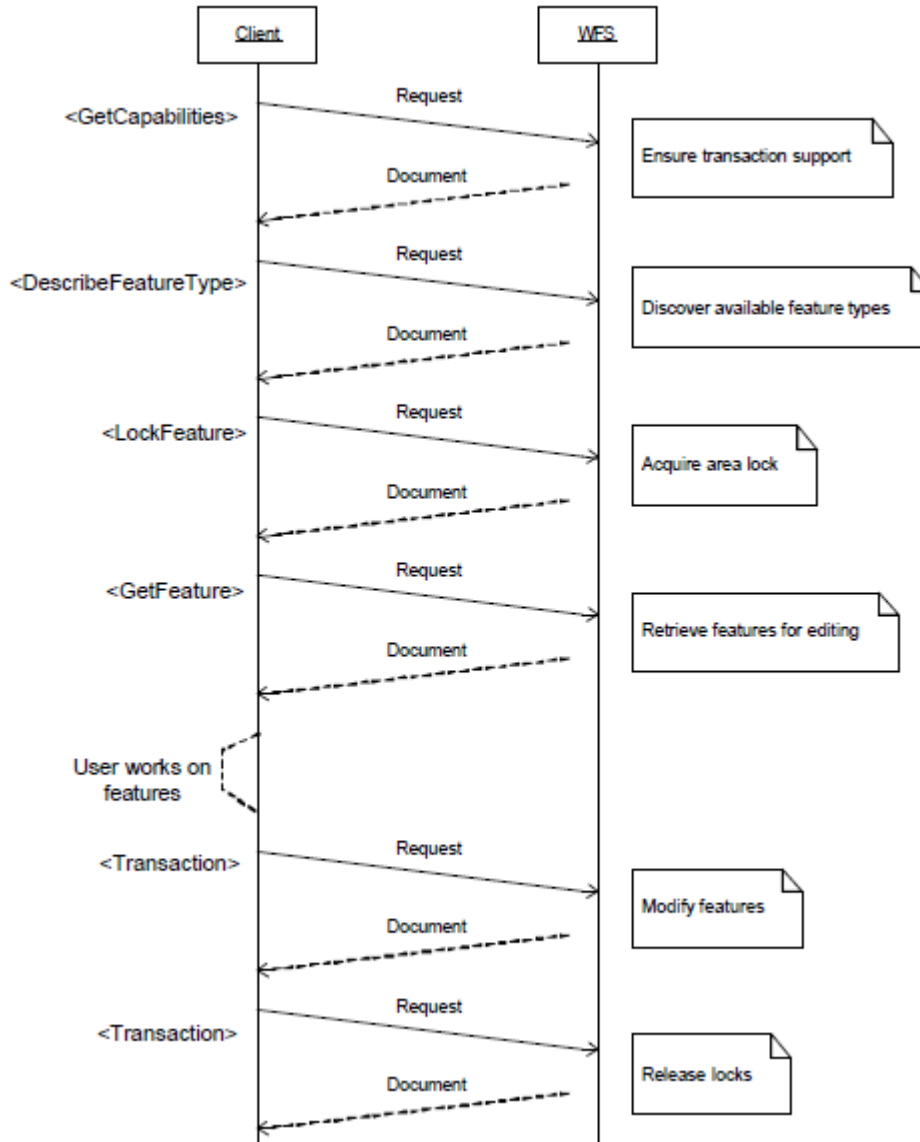


Figure 7: Web Feature Server Communication Diagram [14]

As of May 2010, there are two versions of Web Feature Service (WFS). The first version, 1.0.0, was adopted as an implementation specification by OGC on May 17, 2002. And then the next version is 1.1.0 which was published on May 3, 2005. As we already mentioned about the capabilities of WFS in the previous pages, table 2 shows the comparison of the operations between WFS 1.0.0 and WFS 1.1.0. Operations are the functions that can be initiated through the WFS

call and they all support the internal functions of transaction and query processing. We can see that the only difference between two versions is that WFS 1.1.0 has the ability to perform an operation called GetGMLObject. However, this function was implemented in WFS 1.1.0 which is related to XLink and Transaction classes. Table 3 shows the comparison of WFS classes between two version 1.0.0 and version 1.1.0. Each class can be described as below:

- **Basic WFS** supports the read-only operations including GetCapabilities, DescribeFeatureType and GetFeature.
- **Transaction WFS** supports all operations of a basic WFS and in addition it includes a Transaction operation. Moreover, this class can also optionally implement GetGmlObject and LockFeature operations.
- **XLink WFS**, as same as a Transaction WFS class, supports all the read-only operations and in addition it includes a GetGmlObject operation for local or remote XLinks, and provide the option for this operation to be activated during GetFeature operations.

Operations	WFS 1.0	WFS 1.1.0
GetCapabilities	Yes	Yes
DescribeFeatureType	Yes	Yes
GetFeature	Yes	Yes
GetGmlObject	No	Yes
Transaction	Yes	Yes
LockFeature	Yes	Yes

Table 2: WFS Operations Comparison

Classes	WFS 1.0	WFS 1.1.0
Basic WFS	Yes	Yes
Transaction WFS	Yes	Yes
XLink WFS	No	Yes

Table 3: WFS Classes Comparison

Furthermore, between WFS 1.0.0 and WFS 1.1.0, there is also a difference in the base GML schema used. The OGC WFS is one of the keys to GML deployment; *GML is used in both the request and response messages of the WFS*. The OGC interface offers standardized access to a feature data store and allows users to perform operations such as create, update and retrieve GML feature data. When a client sends a message requesting for geographic feature data to a WFS server, a response message will be sent back to client in GML format. However, WFS and GML have been developed independently. GML, as a format file, can be understandable by GIS applications and not completely served through OGC WFS. Therefore, the versions of GML identified in OGC WFS specifications are not the most recent versions of GML available [3] [15]. WFS 1.0.0 shall conform to GML version 2.1.1 as a minimum version for the identified WFS version [16] while

WFS 1.1.0 shall conform to GML version 3.1.1 as a minimum version for the identified WFS version [17].

Sandra Fällman mentioned in her master thesis, *Using WFS to build GIS support* [10], that during the period she implemented her project (in 2004), WFS was quite new and many negative aspects due to the functionality of WFS were hidden, only general information about the specification and scenarios were presented. She also raised some important issues about how access control handled in the OGC WFS Implementation Specification and how appropriate security levels could be implemented which at that time they were not mentioned in the specification. As the goal of her project is to investigate the technique of using WFS, she developed some demo systems and ended up with a result from implementing demo system level 2, as shown below in figure 8. The system comprises of a client application created by using *Java*, a WMS server (*MapServer*), a WFS server (*Deegree*) and a spatial database (*PostGIS*). Users can use the client application to request for geographic information which the tool is able to access to both the WMS server and the WFS server.

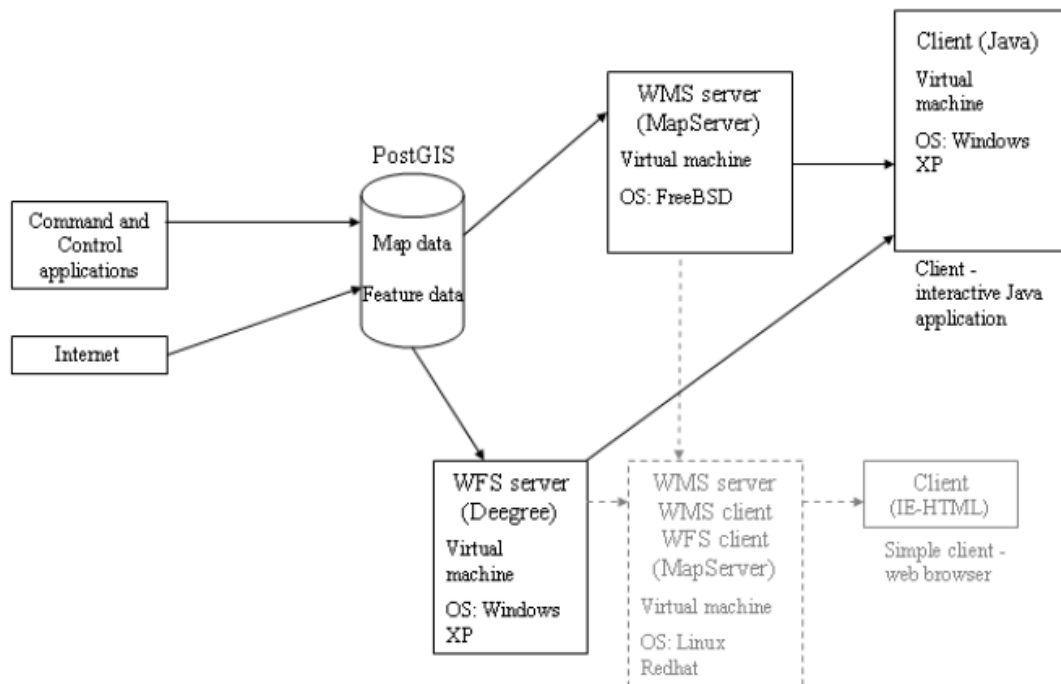


Figure 8: Demo System level 2; Using WFS to build GIS support by Sandra Fällman [10]

Knut Sælid researched the techniques of using WFS-T on mobile phones in his master's thesis, *WFS-T and mobile clients* [18]. His work proves that WFS can be used in a system to generate map images and geographic features, moreover, the results can be displayed successfully on mobile phones. He developed a prototype system called SMILEX including *server-side (SMILEX services)* and *client-side (SMILEX client)*. For the SMILEX services, he implemented both WMS and WFS

on the same platform using *GeoServer* while *MySQL* was used as a data storage. For the SMILEX client, it was developed on Java 2 Micro Edition (J2ME). Moreover, the JSR 179 location API was implemented to achieve a user position from making use of a GPS receiver on a mobile phone. The figure 9 below presents an overview of a communication in the SMILEX system. It allows users not only to ask for spatial information but also to perform transactions (insert, update, delete) on geographic features through mobile phones.

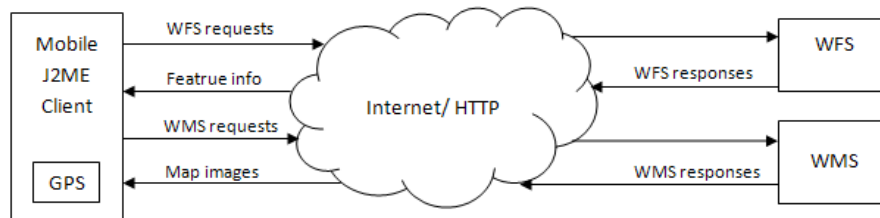


Figure 9: Prototype Communication Overview [18]

4 Research and Implementation

4.1 Investigated platforms

Since the main idea of this project is to find out whether WFS-T can be used in combination with geolocation to edit geographic features online via mobile browsers. To setup a test environment, we decided to create based on Sandra's demo system level 2 as mentioned in chapter 2, *Related Work*, since it was proved in her project, *Using WFS to build GIS support*, that the implemented system worked fine and no problems occurred. Therefore, we need GIS servers serving as a map server and a WFS server, a spatial database providing geographic features, a client application and, in addition, mobile phones that can use to access to the Internet. However, we decided *not* to use exactly the same platforms as the demo system of Sandra since the number of available GIS products in the market has been increasing each day. Therefore, to make a decision which services/platforms will be used, the investigations are made as discussed in greater details in the following sections.

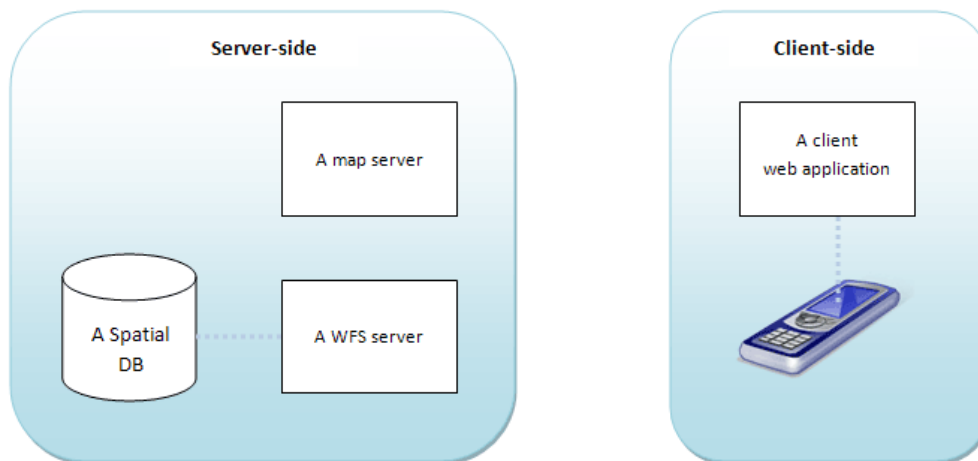


Figure 10: Requirements for a test system

4.1.1 Server-side

A MAP SERVER

Since in this project, we mainly focus on implementing WFS-T, so we consider to use available map images which there are such a variety of them in the market today. And, candidates belong to *Google Maps*, *Bing Maps*, and *OpenStreetMap*; all of them are well-known and providing interfaces to extensive mapping.

- Google Maps [19] is one of the most famous web mapping service applications owned by

Google. Similar to other Google services, JavaScript is greatly used in implementing Google Maps. Moreover, Google offers Google Maps APIs to allow users to manipulate Google maps and embed the maps on their own pages. Google Maps was launched by Google on February 8, 2005 and has always been released new features since then, for instance, *Google Bike Maps* - allows users to search for biking directions on Google Maps, *Google My Maps* - allows users to put their information on Google Maps by using geometric shapes such as points, lines and polygons, *Google Street View* - allows users to see maps based on panoramic street-level views, and so forth. Google Maps is proprietary software and provided for free for non-commercial use.

- Bing Maps [20] is a web-based service developed by Microsoft. To implement Bing Maps, Microsoft has used their own technology called MapPoint Web Service. And similar to Google Maps, Microsoft provides a set of APIs that enable users to manipulate Bing Maps, for instance, users can integrate Bing Maps and location functionality into their own Web site. Moreover, users are able to work with the Bing Maps Platform for free but the use of the maps must be on public-facing without password protected Web sites and the limitation to access in a 12 month period is no greater than 125,000 sessions or 500,000 transactions.
- OpenStreetMap(OSM) [21] is created by Steve Coast in 2004 under the inspiration of the famous site named Wikipedia [22] which is a collaborative project with the slogan "*The free encyclopedia that anyone can edit*". In the same way, OpenStreetMap is a collaborative project with the slogan "*The Free Wiki World Map*". It allows users to manipulate geographic data on the maps in a collaborative way no matter where they are. *The project was started because most maps we think of as free actually have legal or technical restrictions on their use, holding back people from using them in creative, productive or unexpected ways*[21]. Since everyone can take part and join the project, some parts of the map in OpenStreetMap may contain more detail than other services because they may be updated by local people who live in and/or are familiar with the areas. Additionally, not only we are able to manipulate geographic information on the map, but we are also able to download and use all data from OpenStreetMap for any purpose under an open source license.

The web map services above basically provide almost the same features which, in this case, we mainly concern on a map layer. Figure 11 illustrates the map images of the city of Gjøvik, Norway from different sources: Google Maps, Bing Maps and OpenStreetMap respectively. We can see that all of them provide maps with something in common such as the layouts of the maps, road names, and railway signs. However, OpenStreetMap is the chosen service as it is a collaborative work which people all around the world can participate, some areas of the based map (especially in small cities or rural areas) may contain more information such as bus routes, walking tracks, and flights of steps (stairs) on footways; for instance, in figure11, the blue dotted lines in OpenStreetMap represent cycling routes that are separate from the road. More examples of map features in OpenStreetMap are shown in figure12.

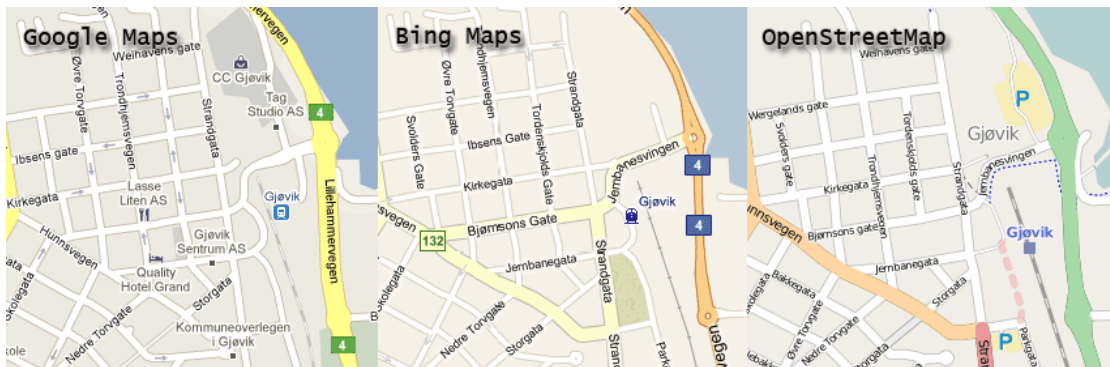


Figure 11: Map images from Google Maps, Bing Maps and OpenStreetMap

A WFS SERVER

There are many well-known platforms supporting the OGC Specifications available in the market today. However, the following products are concerned to be implemented: *MapServer*, *Deegree*, *GeoServer* and *ArcGIS server*.

- *MapServer* [23] is one of well-known Open Source GIS server used for publishing data and creating web mapping applications. It can be run on many major platforms such as Windows, Linux and Mac OS X. However, MapServer supports only basic WFS (read-only) that means we cannot use it to perform transaction operation. From MapServer official website, WFS page, it mentioned in To-do Items and Known Limitations that GeoServer is recommended for those needing WFS-T support [24].
- *Deegree* [25] is an Java API using for creating a GIS. It is also an Open Source provided for free and conforms to many OGC Specifications including a transactional Web Feature Service (WFS-T). For those new users who are interested in using Deegree, there is plenty of tutorial available online. However, the limitation of performing WFS-T by using Deegree is that it does not support transactions on shapefiles.
- *GeoServer* [26] is designed for interoperability. As same as Deegree, GeoServer is an Open-Source server written in Java and also offers full support for OGC standards including WFS-T. The main goal of GeoServer is to implement as a node within a free and open Spatial Data Infrastructure. It intends to offer a free and open web server to publish geospatial data just as the Apache HTTP Server has offered a free and open web server to publish HTML. With GeoServer (WFS-T enabled), users can update, delete, and insert geographic data. In short, GeoServer is one of those tools users need to display maps on web pages and it is used in conjunction with clients such as uDig, GVSig, MapBuilder, and so forth.
- *ArcGIS Server* [27] is commercial software developed by ESRI. It supports interoperability

Key	Value	Comment	Rendering	Photo
highway	trunk	Important roads that are not motorways.		
highway	steps	For flights of steps (stairs) on footways.		
highway	living_street	A street where pedestrians have priority over cars, children can play on the street, maximum speed is low.		
highway	pedestrian	For roads used mainly/ exclusively for pedestrians/ shopping areas.		
cycleway	track	A track is a route that is separate from the road.		
amenity	parking	Car Park. Nodes and areas will get a parking symbol. Areas will be colored.		
leisure	park	Open, green area for recreation, usually municipal		
power	tower	For towers or pylons carrying high voltage electricity cables. Normally constructed from steel latticework.		
man_made	lighthouse	Sends out a light beam to guide ships		

Figure 12: Example features in OpenStreetMap [21]

standards such as OGC and W3C. Moreover, it also supports integrated development environments such as Microsoft Visual Studio, Eclipse and NetBeans. However, ArcGIS Server is commonly used by organizations that want to distribute their own geographic data.

The comparison of each product is made as shown in table 4 , *Comparison of WFS Servers*. We can see that MapServer, Deegree and GeoServer are Open Source software while only ArcGIS Server is commercial software. Though ArcGIS Server is now using by the Geomatics group at faculty of technology, economy and management at Gjøvik University College, where we study, using it seems limit for those new users who interested in learning GIS due to it costs money to use. So, we would prefer to use free software. By the way, even though MapServer is one of Open Source applications which has been developed for quite some years, it does not support WFS-T. So, now, the choice stands between Deegree and GeoServer. Both of them have transactional capabilities and seems considerably widely used among GIS developers. However, we decided to use GeoServer as a WFS server. The main reason is because the author has some experiences in implementing it so under the time limit in developing the prototype, it would be better to use something that we are familiar with. Moreover, GeoServer is a fully functional open source WFS-T server that follows the OGC standards for geographic information sharing with good tutorial and demos available in its official website. So, as a result, GeoServer is selected to implement as a WFS server in this project.

Software	Language	Open Source	WFS	WFS-T
MapServer	C	Yes	Yes	No
Deegree	Java	Yes	Yes	Yes
				(but transactions on shapefiles do not work)
GeoServer	Java	Yes	Yes	Yes
ArcGIS Server	.Net/Java	No	Yes	Yes

Table 4: Comparison of GIS products to be used as a WFS server

DATASTORES

GeoServer reads a variety of data formats such as PostGIS, MySQL, Oracle Spatial, ArcSDE, Shapefiles and so forth. To make a decision about which datastore we should use in this project, the choice stands between *PostgreSQL with PostGIS*, *MySQL* and *Oracle Spatial*.

- PostgreSQL [28] with PostGIS [29]; PostgreSQL has been developed for more than 15 years as open source object-relational database system. It can be used on all major OS such as UNIX, Mac OS X and Windows. Moreover, there are many interfaces supporting Java, ODBC, Perl, Python, Ruby, C, C++, PHP, and so on. However, only PostgreSQL alone is unable to work with spatial data, we need to install PostGIS which is an extension adding support for geographic objects to the PostgreSQL object-relational database.
- MySQL [30] has been released since 1995 as an open source relational database management

system. Many well-known software projects such as WordPress, Wikipedia, Google, and so forth, use MySQL regarding it offers nice features and is flexible to be used as it provides source code available under the terms of the GNU General Public License. MySQL is written in C, C++ and can be used on cross-platform.

- Oracle Spatial [31] is commercial software offering data management for geographic data. It is an option to Oracle Enterprise Edition serving as a foundation for spatial information systems. It allows users to be able to manage geographic data directly in their applications and services.

As same as the reasons we used in making decisions in the previous steps, due primarily to time constraints, we would consider to use something that we know and it can work fine with our project. In this case, the author has worked with Oracle and PostgreSQL with PostGIS before, though MySQL database can be useful and worked well in the project, it is excluded as a candidate. And, between PostgreSQL with PostGIS and Oracle, the author would prefer 'PostgreSQL with PostGIS' due to it is Open Source software which means it is free of use.

4.1.2 Client-side

A CLIENT WEB APPLICATION

Since we want to see whether WFS-T can be used in combination with geolocation on mobile browsers, we need to develop a web application that can be used through mobile browsers. Web browser are responsible for reading HTML documents, then interpreting and showing them as web pages. Therefore, we plan to use HTML (Hyper Text Markup Language), *a language for describing web pages*, to create interactive interface. With HTML, we can embed many scripts (*such as JavaScript, JScript and ActionScript*) to make our pages more dynamic and more interactive. By the way, how can we make geographic information available on the Web? or Which GIS tools we can use to put a map including its detail on a web page? To answer the questions, an investigation of the following products are made:

- ArcGIS Web Mapping[32] by ESRI is a collection of APIs providing a solution for us to create interactive web applications. The term Web APIs primarily refers to the JavaScript, Flex, or Silverlight APIs which all of them allow us to add content to the map via online services, build stand-alone applications, connect to GIS servers, and so forth. For implementing cost, development and deployment with the APIs are free but the latter can be done under certain terms and conditions. By the way, to use ArcGIS Web Mapping, we need to have an ArcGIS Server available whose maps and tools we can use in our application.
- OpenLayers[33] is a free open source framework with the slogan - *Free Maps for the Web*. It is a JavaScript library which not only allows us to display geographic information in web browsers, but also provides an API for creating rich web-based geographic applications. In

OpenLayers, map tools are separated from map data so we can use all the tools to implement on all the data sources, this helps to fix the lack of interoperability. Moreover, it supports open standards and protocols for geographic data access such as WMS and WFS. The initially release of OpenLayers was launched in June 2006. After that, many new versions have been released accordingly with useful documents and examples available on its website. Furthermore, OpenLayers has a powerful community behind it with users from many countries all over the world and its functionalities have been keeping on developing and growing.

- OpenScales[34] is a community-driven mapping project implemented by a diverse group of people all around the world. It is a user-friendly interface designed for interoperability supporting a variety of OGC formats and protocols such as WMS and WFS. Moreover, it allows us to build rich Internet mapping applications and manipulate geographic information. OpenScales was originally created from FlexLayers which was a Flex port of OpenLayers; users who are familiar with OpenLayers can easily and quickly learn to use OpenScales because it offers many similarities to OpenLayers. By the way, the main difference between OpenScales and OpenLayers is that OpenScales uses ActionScript 3 and Flex while OpenLayers uses JavaScript.

The first framework, ArcGIS Web Mapping, is proprietary while the others, OpenLayers and OpenScales, are open source; all of them are provided for free. Unfortunately, ArcGIS Web Mapping can be used only if we implement an ArcGIS Server in our system. So, it is excluded from the list of candidates. Now, the choice stands between OpenLayers and OpenScales which both of them should be able to work fine in the project since they support a WFS protocol and allow us to develop web a mapping application. Anyway, OpenLayers is selected over OpenScales to be used for the implementation of the web application since OpenLayers seems to be far developed with good documentation and examples available on its website.

Since we want to test whether WFS-T can be used in combination with geolocation through mobile browsers, to use OpenLayers alone seems not enough; in this case, we need a solution that can help to provide a user location and show it on a browser. Actually, the concept of locating users by making use of the geographical position of their mobile devices is not new as there are many location-based mobile applications available in the market today such as *Google Latitude* - a location-aware mobile application created by Google; *Gowalla* - a mobile web application created by Alamofire allowing users to check-in to locations that they visit using their mobile devices; and *Foursquare* - a web and mobile application offering users to connect with friends and update their location. However, at the time of doing this project, there are still few solutions that can be used to implement a mobile web application in order to detect users' locations when they launch an application through their mobile browsers; the following APIs are concerned as candidates to be implemented in the project:

- Google AJAX APIs[35] developed by Google includes a simple object property called

`google.loader.ClientLocation` which allows our application to acquire a user's location based on a user's IP address. In general, this method can expose an approximate location of the user, but sometimes it is unable to locate users (especially for those who are behind an anonymous proxy), or may return us inaccurate location (based on IP address geolocation providers). Additionally, to use this solution, we need to sign up for an API key which is provided for free.

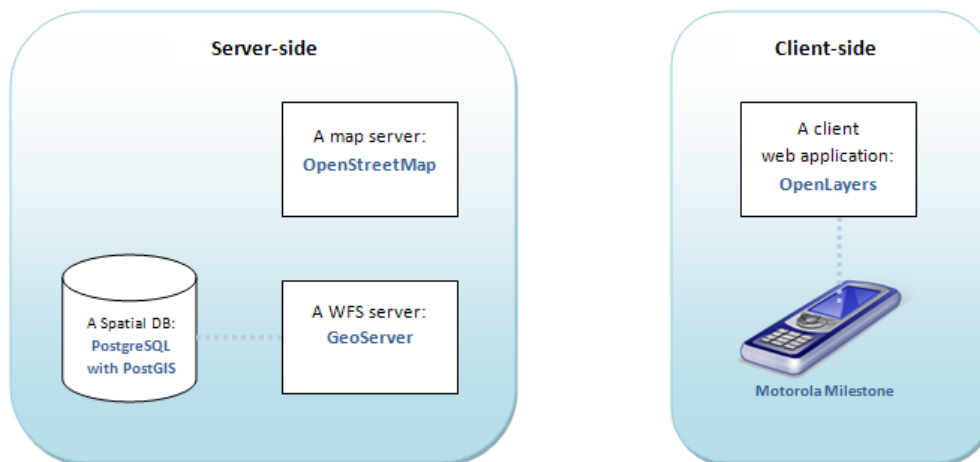
- **Gears API**[36] by Google provides a Geolocation API allowing our web application to gather a user's geographical position by using the `getCurrentPosition` method which it is able to determine the user's location by using a set of data such as the user's IP address and WiFi nodes. Moreover, the Gears Geolocation API is now capable to use WiFi antenna data to provide more accurate position data. Google Gears is an open source project that enables us to create powerful web application by providing many new features for us to add to web browsers. However, to be able to use Gears API, an installation of Gears on a computer may be required.
- **W3C Geolocation API**[37] by W3C, the well-known organization for the World Wide Web, offers a way to gather a user's location by associating a geographic location with a hosting device like Global Positioning System (GPS), IP address, RFID, WiFi and Bluetooth MAC addresses, and so forth. W3C Geolocation API has been developed based on earlier work such as *Geolocation in Firefox and Beyond*, *Gears Geolocation API*, and *LocationAware.org*. To use this API, neither an API key nor an installation are required. However, regarding security and privacy considerations, users will be asked whether they want to share their locations when they load the application via web browsers, this is to ensure that no location information is done though the API before users allow it to do so.

All of the above APIs are provided for free and seem work fine in this project. However, we decide to use **W3C Geolocation API** in this project since there is no work required before using it compare to the first two APIs, *Google AJAX and Gears*, which we either need to register for an API key or install a tool on our computer first.

Futhermore, for a client device, we plan to make a test on *Motorola Milestone* since it is sponsored by our school. Its operating system is Android 2.0 with some nice features support including *web browser with location enabled, aGPS, 3G and so on*.

4.2 Implementation

After the investigation of platforms, figure10 is filled up with selected products which the figure below illustrates the results showing an overview of what we are going to implement in our system. We categorize the system into 2 parts including 1) **SERVER-SIDE** and 2) **CLIENT-SIDE**. The first part contains a map server which **OpenStreetMap** is selected to be used to provide a map image, a WFS server which **GeoServer** is chosen for providing geographic features, and lastly, a spatial database which **PostgreSQL with PostGIS** is picked to be used to store geographic information. On the other hand, for client-side, **OpenLayers** is selected to be used to implement a web-based application and **Motorola Milestone** is chosen to be used as a test device.



An overview of system components

4.2.1 Preparation

SERVER-SIDE

1. Installation

- **GeoServer** was started in 2001 and has been developed and released continually. As of May 2010, the latest version of GeoServer is 2.0.2 released on May 25, 2010. By the way, GeoServer 2.0.1 is selected to be used in this project as it was the latest stable version released during the time we started doing the project.

GeoServer offers many choices as installation methods including using *OS-independent binary*, using *Web archive* and using *Installer*. In this case, we develop our project based on Windows OS so the easiest way seems to be using Windows Installer. After we download and run the installation file, the operations took only few minutes (more information about the installation steps can be found in **GeoServer User Manual** provided in `GeoServer.org`). Additionally, some required settings and problems during the setup are mentioned in appendix A

- **PostgreSQL** has the current stable version as 8.4.4 released on May 17, 2010 (for Windows OS). By the way, in this project, PostgreSQL 8.3.4 is installed on the same machine as GIS server. The installation process took few minutes like when we installed GeoServer. Nevertheless, only PostgreSQL alone cannot complete the requirement of being spatial database so we need to install PostGIS to add support for geographic objects to the PostgreSQL and allows PostgreSQL to be used as a backend spatial database for geographic information system. As a result, PostGIS 1.5 is installed.

2. Prepare data

After we have GeoServer and PostgreSQL with PostGIS installed on the machine, it is time to fill up the database. In this project, we decided to use geographic data from *OpenStreetMap (OSM)* project. OSM allows us to feel free to download and use the full information for any purpose we like under an open source license [21]. The following items are choices of geographic data formats that we are able to export from OSM:

1. **OpenStreetMap XML Data** contains all the current raw geographic data in an XML format. As a result, we will get a file called `map.osm` including Nodes, Ways, Relations, and Tags in the file.
2. **Mapnik Image** offers map images in the default of OpenStreetMap style called Mapnik. A set of image formats includes PNG, JPEG, SVG, PDF and PostScript.
3. **Osmarender Image**, similar to Mapnik Image, allows users to export a map image but in a different style which is called Osmarender. We can decide to export map images in PNG or JPEG.
4. **Embeddable HTML** allows us not only to embed the map in our website, but also to mark a specific location on the map. As a result, we will get an output text containing a link which looks similar to the following: `http://www.openstreetmap.org/?lat=60.79425&lon=10.686485&zoom=14&layers=B000FTFT`

In this case, we need a raw geodata, so we select to export OSM data as 'OpenStreetMap XML data'. Once we export data from OSM, we will get an `.osm` file in XML format with related geographic information, such as vector features, inside the area we specify in the map. The output file contains similar syntax like the following:

```

<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
  <bounds minlat="60.78605" minlon="10.67069" maxlat="60.80245" maxlon="10.70228"/>
  <node id="490165001" lat="60.7948358" lon="10.6893501" user="Kagee" uid="46430"
  visible="true" version="1" changeset="2418950" timestamp="2009-09-08T21:13:06Z">
    <tag k="amenity" v="post_box"/>
  </node>
  <way id="33652993" user="Kagee" uid="46430" visible="true" version="1" changeset=
  "961477" timestamp="2009-04-25T22:57:03Z">
    <nd ref="384977612"/>
    <nd ref="384977619"/>
    <tag k="highway" v="unclassified"/>
  </way>
  <relation id="952454" user="Kagee" uid="46430" visible="true" version="1"
  changeset="4923930" timestamp="2010-06-06T23:20:49Z">
    <member type="way" ref="35177205" role="road"/>
    <member type="way" ref="61348664" role="road"/>
    <tag k="type" v="associatedStreet"/>
  </relation>
</osm>

```

Then, we need to convert data into a format that can be loaded into PostGIS database. In this step, FME Universal Translator [38] can do the trick. However, in the paper *'Interface design for mobile devices'* [39], the author mentioned some concerned issues when designing mobile devices user interface which one of them is *the size of a mobile device is usually small so the developers of mobile applications have to cope with limited screen size*. Therefore, as we plan to show geographic information on mobile devices which they may be not suitable for displaying large amounts of data, we may also need to optimize our data too. Furthermore, a data store in GeoServer must be created in order to use it to connect to geographic data in PostGIS database. To sum it up, the processes of data preparation are illustrated as shown in figure 13 including:

- 1) *Exporting data from OpenStreetMap*
- 2) *Converting and Importing data using FME Universal Translator*
- 3) *Manipulating and Optimizing data with pgAdmin III*
- 4) *Creating a data store in GeoServer*

For more details about the preparation steps, see appendix C - Data Preparation.

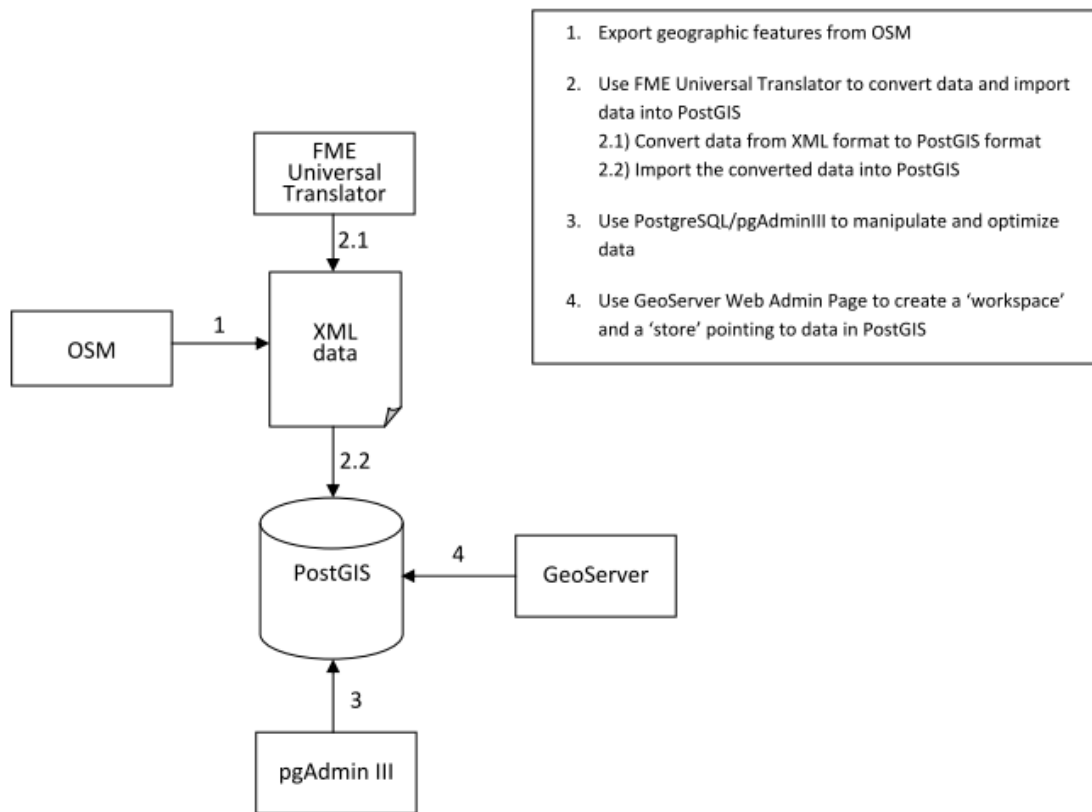


Figure 13: Data preparation steps

CLIENT-SIDE

Since the goal of this project is to try out the capability of WFS-T in combination with geolocation in order to edit geographic features through mobile browsers. Therefore, we must create a web-based application that is able to (1) *use to edit geographic features*, and (2) *determine a user's location*. As mentioned in the investigation part, to reach item (1), we will use *OpenLayers*, while for item (2), *W3C Geolocation APIs* can do the trick.

1. Using OpenLayers library to show geographic data and edit geographic features

OpenLayers is a pure JavaScript library providing for free as an open source. It helps us to easily put a dynamic map in our web page with no server-side dependencies. To use OpenLayers Javascript API, we need to include "OpenLayers.js" in our page. The file can be stored on our machine or if we wish to always use the latest release of the script, we should point a location of the file to `http://www.openlayers.org/api/OpenLayers.js`, as shown below in figure 14.

```

1 <html>
2   <head>
3     <title>WFS-T: OpenLayers, GeoServer, PostGIS, OSM</title>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5     <link href="default.css" rel="stylesheet" type="text/css" />
6
7     <script src="jquery 1.3.2.min.js"></script>
8     <script src="http://openlayers.org/api/OpenLayers.js"></script>
9     <script src="index.js"></script>
10    <script>
11

```

Figure 14: Include OpenLayers.js in a web page

Moreover, some essential codes of OpenLayers that we need to use in this project are mentioned below:

function	description
OpenLayers.Layer.WMS	Create a new WMS layer object
OpenLayers.Layer.Vector	Create a new vector layer
OpenLayers.Protocol.WFS (to be used with <i>OpenLayers.Layer.Vector</i>)	Used to create a versioned WFS protocol

2. Using W3C GeoLocation API to gather a user's location

We can locate a user's position by creating a location-aware web page to make a user share her location from using her mobile device, which in this case, we use Geolocation API. Generally, the sources of location information can be Global Positioning System (GPS) and location collected from network signals containing IP address, WiFi, MAC addresses, and so forth. The following code is an example of how to get a basic location information [37]:

```
//Example of a one-shot position request
navigator.geolocation.getCurrentPosition(showMap);
```

4.2.2 Testing

After a prototype is created, we have tested it both on *mobile browsers* and *web browsers on personal computers*, which some screenshots of the application are captured and revealed. For instance, figure 15 shows that when a user runs the application through a mobile browser, she is asked whether she allows the application to gather her location information. While figure 16 and 17 happen when a user launches the application through Firefox, which similar to the previous case that the user runs the application on the mobile browser, she is asked whether she wants to share her location information, the application will do so in case that she agrees. And once she clicks a *Share location* button to allow the application to determine where she is, as a result, her location including nearby places are displayed on a map as shown in figure 17; a yellow smile icon represents her location while green circles are nearby shops. Additionally, figure 18 describes a sequence of operations that the application provides for online transactional editing of geographic features through mobile browsers, which the use case contains all possible states since the application is launched until a user quits the application. Furthermore, refer to the earlier example in the Introduction chapter, *we move to a new house and want to search for nearby shops*, in this case, this application can be helpful, as we are going to discuss in greater details in the next paragraph.

A scenario for the system

Jasmin is an international student from China coming to Norway to study a master's degree in Media Technology at Gjøvik University College (GUC) in Gjøvik. Today is the first day that she reaches Gjøvik. She meets Li Xiatao, who is a second year student studying the same major as her; he comes to pick her up at the train station and drives her to Kallerud where she is going to stay. Kallerud is a student village located 500 meters from GUC and 2 km from Gjøvik city center with laundry and internet access are provided.

Once Jasmin arrives at her room in Kallerud, she finds that she needs to buy many stuff including food and some convenience products. She asks Xiatao whether there are any supermarkets around the campus. Instead of answering her question, Xiatao uses his mobile phone launching a web application via his mobile browser, the application asks whether he allows it to gather his location information, which he accept the request. and few minutes after that, he shows what are displayed on his mobile screen to her. As a result, Jasmin sees a map with some symbols marking on, which she can understand that a yellow icon represents her location while some green dots are nearby stores. Xiatao tells her that he often uses the application to search for nearby stores and once he discovers that the shop he visits is new and not on the map, he will add it through the application. On the other hand, if he knows that a shop is no longer open or moved to a new place, he will either delete or update its location on the map. Jasmin says thank you to Xiatao and plans to use her mobile phone to connect to the application to discover Gjøvik. Additionally, she tries to use the application on her personal computer and sees that it can work fine through Firefox.



Figure 15: Xiatao is asked to share a location when running the application through a mobile browser on his Android phone.



Figure 16: Jasmin is asked to share a location when running the application via Firefox on a PC.

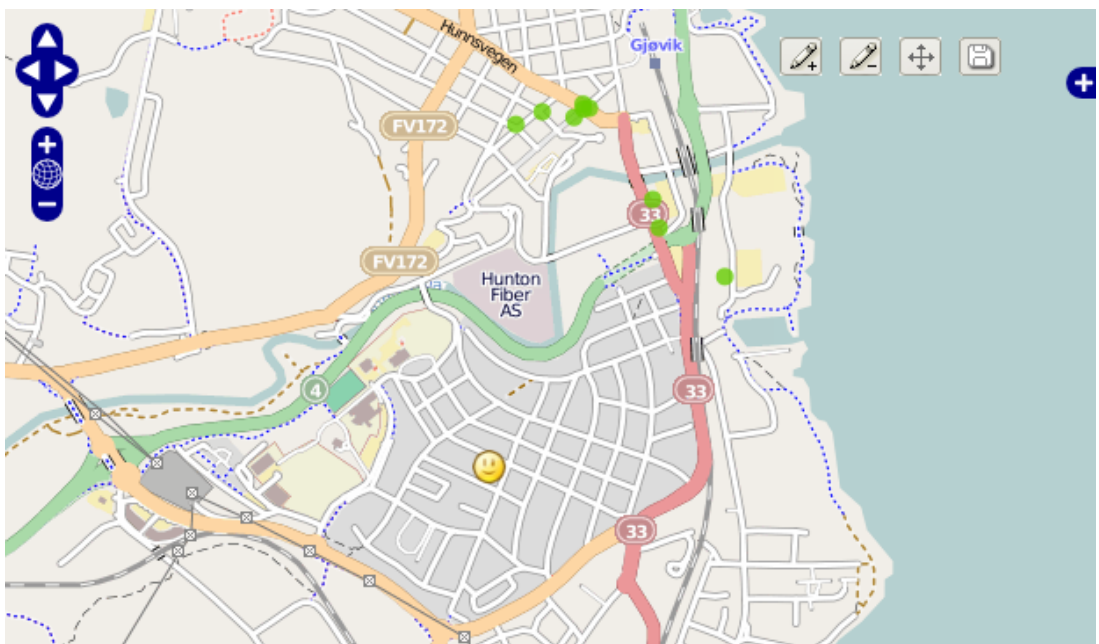


Figure 17: A yellow smiley icon locates Jasmin's location, green points represent nearby shops.



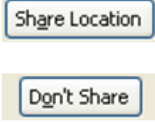







Step	Subject	Operation		Connection to/from
1	Client	launches the web application through mobile browser		Web server
2.1	Application	shows Map and geographic features		WMS, WFS servers
2.2	Client	is asked to share a location 1) If answer = Yes, continue the process 2) If answer = No, stop the process		geolocation
3	Application	identifies client's location based on availability of cell tower and/or WiFi access points		geolocation
4	Application	shows a client's location on the map		geolocation
5.1	Client	zooms map		WMS server
5.2	Client	Pans map		WMS server
5.3	Client	inserts new features		WFS server
5.4	Client	updates features		WFS server
5.5	Client	deletes features		WFS server
5.6	Client	saves features		WFS server
6	Client	exits application		

Figure 18: An application Use Case

5 Results and conclusion

5.1 Results

In this project, we have developed a prototype to see whether we can make use of WFS-T in combination with geolocation to perform online transactional editing of geographic features through mobile browsers. As a result, we can describe as answers to the research questions as followings:

«RQ-1» *Is it possible to implement WFS-T in combination with geolocation as a web-based application used on mobile browsers?*

From developing and performing a test, we found that WFS-T and geolocation can work well together through a mobile browser on Android phone. However, because W3C Geolocation API that we selected to use instead of using other geolocation APIs such as Google Loader and Gears, is still new, not all browsers support it. As a result, our applicant can work completely only on those browsers that support W3C Geolocation API. We have tried on various browsers both on PCs and mobile phones, results are listed below:

NORMAL BROWSERS:

- *Mozilla Firefox* (version 3.5 and later)
Once we open the application, we will be asked for permission to share our location (similar to figure 16, as shown earlier in the previous chapter). In case that we agree, Firefox will gather information about nearby wireless access points and our IP address. Then it will send the information to the default geolocation service provider which is Google Location Services to get an estimate of our location. Finally, the estimated location is shown on the application as a yellow smile marker 17.
- *Google Chrome* (version 5.0.371.0 dev and later)
Once we open the application, we will be asked whether we allow to be tracked our physical location. If we consent, then, the next steps will be like what we already mentioned in Firefox.
- *Safari* (version 5 and later)
The application works fine, which our location can be detected and shown on the map.
- *Internet Explorer*
As of now, there is no support for geolocation in the current version (IE8). Therefore, once we launch the application, our location will not be determined.

MOBILE BROWSERS:

- *Android Browser*
Android browser supports the W3C Geolocation API. As a result, the application can work

well on this platform. Similar to Mozilla Firefox, we will be asked whether we allow the application to gather our location information (as shown earlier in figure 15), and once we agree with it to do so, our location is presented on the map as a yellow smiley icon.

- *Safari on iPhone (OS 3.0 and later)*

Similar to Android browser, the application can work well on Safari.

«RQ-2» *What are the main challenges to make use of WFS-T in combination with geolocation on mobile phones?*

To develop a prototype, using OpenLayers library to deal with WFS-T is not that hard but somehow some small problems could always happen. And once the problems happen, it takes quite some times until we can find the solutions. In this project, when we have a problem, the OpenLayers community including the mailing lists are very helpful especially for those new users who want someone to help fixing their problems. However, Dean and Jim mentioned about the AAA slogan in their book, *Semantic Web for the Working Ontologist: Effective modeling in RDFS and OWL*, that on the World Wide Web, Anyone can say Anything about Any topic, which means people are free to write a page saying whatever they want, and then publish it through the Internet. No one can guarantee that what we have seen on the public web board is completely correct. Though the open source software like OpenLayers and GeoServer have been used by many people, when we find an error and post the problem on the Internet forums, we are not able to force anyone in the community to help answer your question. It is not like using commercial software in which when you have a problem, you can contact the technical support regarding the software product and may get a response back in short period. Moreover, in some cases, we can see that documentations of some open source products are unclear or not well structured with grammar errors. This kind of problems can lead new users to feel that it is hard to learn to use software. Additionally, since geolocation is a new technology, not many users realize about it or understand how it works. Moreover, not all browsers support geolocation. So, to implement the application (with geolocation-enabled) to work on every framework seems not possible at this time.

«RQ-3» *To what extent are map and geographic features can be displayed on mobile browsers?*

Since mobile phones have small screens, the main challenge of the implementation based on showing geographic information is belong to how to compose a suitable map image. From the test, we found that either a base map layer (OpenStreetMap) or overlay layers (Shops and Current Positions) can be shown successfully on mobile browsers as in this case we mainly performed a test on Motorola Milestone (Android), which its screen size (diagonal) is 3.7". Moreover, some limitations while running the application on mobile phones have been revealed such as it is not easy to select the geographic features (green dots) especially in case that two dots are close to each other. Additionally, it is also not easy in order to zoom or pan the map due to the same reason as earlier, buttons are close to each other.

5.2 Conclusion

While using WMS returns geographic data as maps, WFS can return results in various formats such as GML/KML and shape files. Therefore, we can say that WFS is more flexible due to it offers users to play with the output and can write their own queries to get the geographic information they need. WFS-T is an extension of WFS which one obvious feature of WFS-T is that it allows users to insert, update and delete geographic data.

From the project implementation, we found that it is possible to implement WFS-T in combination with geolocation as a web-based application to be used on mobile browsers. However, there are some difficulties and limitations in using the application including *an access to the Internet is required during the usage of the application, an error from selecting wrong geographic features can easily occur especially when the features are close to each other, and zooming and panning the map are not easy to perform due to the similar reason as selecting geographic features which is buttons are close to each other.*

Futhermore, the application is unable to work on those browsers that do not support geolocation due to W3C Geolocation API is still new. Geolocation is a process of tracking and mapping which can be performed by connecting a geographic location with IP address or device GPS coordinates. To bundle W3C Geolocation API in the application will make the web browser can determine where we are located. In this case, we may curious about our privacy. For those web browsers that support W3C Geolocation API such as Firefox and Google Chrome have a policy not to share our location without our permission. We can see that when we visit a page that geolocation code is bundled, we will be asked before any information is shared.

By the way, since Internet scams and frauds have been increasing each day, we may need to concern about what information we are going to share and how much we can trust the websites that ask for our information.

6 Future Work

WFS-T is not new but it still not widely used. While geolocation is a new technology that becomes well-known and useful this days. The implementation of WFS-T in combination with the geolocation technique for mobile browsers seems not much published. Therefore, to develop a prototype in this project seems to be interesting considering that there are now no similar web applications for mobile browsers available in the market. However, due primarily to time constraints, we cannot include all features of WFS-T in our project. Currently, with a rapid growth of mobile technology, new modern mobile phones have provided many functions, for example, we can use our mobile phones to listen to MP3, record a voice, take a photo, buy a movie ticket, book a hotel, and so on. Moreover, many smartphones now include built-in GPS receivers that means we are able to use map no matter where we are. And as a result, many applications can make use of the new technology to estimate our location and provide some useful information that we may be interested such as when we travel to one city and want to get some information about nearby restaurants, we can just open a web browser (with geolocation-enabled) and then we can see our location including the points of the nearby restaurants on the map. In the future, based on the created prototype, some modules can be implemented to enhance the functionalities such as to make users log on before using application, to make users upload their files (pictures, voices, comments) related to the places, and so forth. Additionally, few solutions of determining location of users from using web browsers have been published at present. Each solution has its own limitations, if we can find the way to make them work together especially to be able to perform in every platform, that will be nice.

Bibliography

- [1] Krygier, J. & Wood, D. *Making maps: a visual guide to map design for GIS*, p. 3. The Guilford Press, 2005.
- [2] ISO.org. ISO TC211. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=54904. (Visited Jun 2010).
- [3] Opegeospatial.org. OGC standards. <http://www.opegeospatial.org/standards/>. (visited May 2010).
- [4] SEWilco. 2007. Ogc standards help gis tools communicate. http://commons.wikimedia.org/wiki/File:Geoservices_server_with_apps.png. (Visited May 2010).
- [5] Portele, Clemens (ed.). August 2007. OpenGIS Implementation Specification #07-036: OpenGIS Geography Markup Language (GML) Encoding Standard, version 3.2.1.
- [6] Grigsby, J. 2010. Mobile operating systems and browsers are headed in opposite directions. <http://radar.oreilly.com/2010/05/mobile-operating-systems-and-b.html>. (Visited Jun 2010).
- [7] Wikipedia.org. Mobile Browser. http://en.wikipedia.org/wiki/Mobile_browser. (visited May 2010).
- [8] Developer.Android.com. Location Manager. <http://developer.android.com/reference/android/location/LocationManager.html>. (visited May 2010).
- [9] Steiniger, S. & Bocher, E. 5 SEPTEMBER 2008. An overview on current free and open source desktop GIS developments. *Int. J. of Geographical Information Science*.
- [10] Fällman, S. Using WFS to build GIS support. Master's thesis, Department of computing science, Umeå University, Sweden, 2004.
- [11] Vretanos, P. A. (ed). May 2005. OpenGIS Implementation Specification #04-094: Web Feature Service Implementation Specification, version 1.1.0.
- [12] Nanchang, P. R. 22-24 MAY 2009. A WebGIS framework for vector geospatial data sharing based on open source projects. *The 2009 International Symposium on Web Information Systems and Applications*, 124–127.
- [13] Jun, J., Chong-jun, Y., Ying-chao, R., & Miao, J. 2008. The research and application of WFS based GML. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. vol. XXXVII Part B4.

- [14] Jody Garnett and Brent Owens. 2003. Validating Web Feature server Design Document.
- [15] Chris deJager, AMEC Earth and Environmental Technology Business Unit. 24 May 2006. Web Feature Services, Considerations for CGDI Government Partners. version 1.0.
- [16] Cox, S., Cuthbert, A., Lake, R., and Matell, R. (eds.). April 2002. OpenGIS Implementation Specification #02-009: OpenGISGeography Markup Language (GML) Implementation Specification, version 2.1.1.
- [17] Cox, S., Daisey P., Lake, R., Portele C., Whiteside A. (eds.). January 2005. OpenGIS Implementation Specification #02-023r4: OpenGISGeography Markup Language (GML) Implementation Specification, version 3.1.1.
- [18] Sælid, Knut. WFS-T and Mobile Clients. Master's thesis, Department of informatics, University of Oslo, Norway, 2006.
- [19] Google. Google Maps. <http://maps.google.com/>. (visited June 2010).
- [20] Microsoft. Bing Maps. <http://www.bing.com/maps>. (visited June 2010).
- [21] OpenStreetMap.org. OpenStreetMap (OSM). <http://www.openstreetmap.org/>. (visited May 2010).
- [22] Wikipedia.com. Wikipedia. <http://en.wikipedia.org/wiki/Wikipedia>. (visited June 2010).
- [23] Mapserver.org. MapServer. <http://www.mapserver.org/>. (Visited May 2010).
- [24] Mapserver.org. MapServer - WFS Server. http://www.mapserver.org/ogc/wfs_server.html. (Visited May 2010).
- [25] Deegree.org. Deegree. <http://www.deegree.org/>. (Visited May 2010).
- [26] GeoServer.org. GeoServer. <http://www.geoserver.org/>. (Visited May 2010).
- [27] ESRI.com. Arcgis Server. <http://www.esri.com/software/arcgis/arcgisserver/index.html>. (Visited May 2010).
- [28] Postgresql.org. PostgreSQL. <http://www.postgresql.org/>. (visited May 2010).
- [29] refractions.net. PostGIS. <http://postgis.refractions.net/>. (visited May 2010).
- [30] MySQL.com. MySQL. <http://www.mysql.com/>. (visited May 2010).
- [31] Oracle.com. Oracle. <http://www.oracle.com/>. (visited May 2010).
- [32] ESRI. ArcGIS Web Mapping. <http://www.esri.com/software/arcgis/web-mapping/>. (visited June 2010).
- [33] OpenLayers.org. OpenLayers. <http://openlayers.org>. (visited May 2010).

- [34] Openscales.org. OpenScales. <http://openscales.org>. (visited June 2010).
- [35] Google. Google Loader - google.loader.ClientLocation. <http://code.google.com/apis/ajax/documentation/>. (Visited June 2010).
- [36] Google. Google Gears for Mobile - Geolocation API. http://code.google.com/apis/gears/api_geolocation.html. (Visited June 2010).
- [37] W3C. Geolocation API. <http://dev.w3.org/geo/api/spec-source.html>. (visited May 2010).
- [38] FME Desktop ESRI Edition. FME Universal Translator. <http://www.safe.com/>. (visited June 2010).
- [39] Vávra, M. January 2009. Interface design for mobile devices. *Cork Institute of technology*.
- [40] Cook, J. Web Mapping Course Notes. http://www.maths.lancs.ac.uk/~stantonm/Web_Mapping_Course_Notes.pdf. (Visited June 2010).

A GeoServer Issues

After we install GeoServer, the below process may be required to implement:

- We will NEED to set the JAVA_HOME environment variable if it is not already set. This is the path to your JDK/JRE such that %JAVA_HOME%\bin\java.exe exists.

- We may also need to set the GEOSERVER_HOME variable, which is the directory where GeoServer is installed, and the GEOSERVER_DATA_DIR variable, which is the location of the GeoServer data directory (usually %GEOSERVER_HOME%\data_dir). The latter is mandatory if you wish to use a data directory other than the one built in to GeoServer. The procedure for setting these variables is identical as shown in table 5 and table 20.

1. Navigate to Control Panel > System > Advanced > Environment Variables.
2. Under System variables click New.
3. For Variable name enter JAVA_HOME. For Variable value enter the path to your JDK/JRE.
4. Click OK three times.

Table 5: Setting environment variables

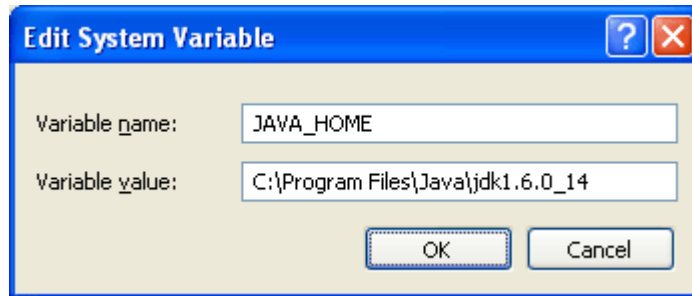


Figure 19: Setting JAVA_HOME

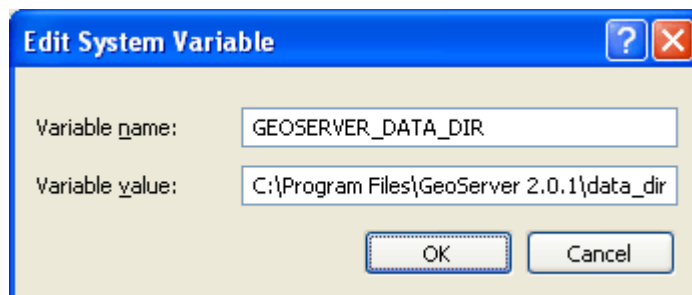


Figure 20: Setting GEOSERVER_HOME and GEOSERVER_DATADIR

B PostgreSQL with PostGIS Issues

PostgreSQL supports capital letters in table and column names, but it requires them to be quoted in queries. Table 6 shows some examples of how the results return when the table names are "norway" and "Norway".

Table Name	SQL command	Result
norway	select * from norway;	Success
Norway	select * from norway;	ERROR: relation "norway" does not exist SQL state: 42P01
Norway	select * from Norway;	ERROR: relation "norway" does not exist SQL state: 42P01
Norway	select * from "Norway";	Success

Table 6: Query problem in PostgreSQL

C Data preparation

C.1 Export data from OpenStreetMap (OSM)

To Export data from OpenStreetMap (as shown in figure 21), visit <http://www.openstreetmap.org/>. Click 'Export' tab, select 'area of interest' and then 'Export' it as 'OpenStreetMap XML Data'. We will get one XML file with '.osm' extension. The structure of the file will probably look like in figure 22.

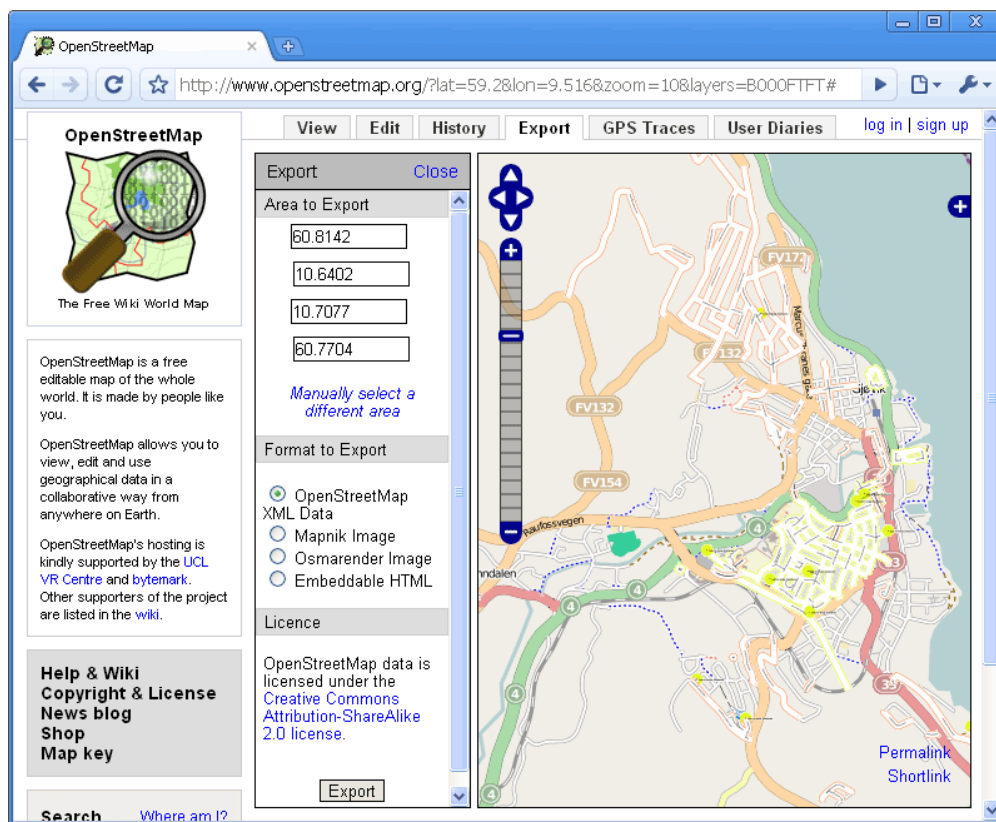


Figure 21: Export data from OSM

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <osm version="0.6" generator="CGImap 0.0.2">
3 <bounds minlat="60.7704" minlon="10.6402" maxlat="60.8142" maxlon="10.7077"/>
4 <node id="31264090" lat="60.7987" lon="10.6937617" user="Helge Hafting" uid="19565"
visible="true" version="3" changeset="504536" timestamp="2008-01-01T19:12:00Z">
5 <tag k="created_by" v="Potlatch 0.4a"/>
6 <tag k="is_in" v="Oppland, Norway"/>
7 <tag k="name" v="Gjøvik"/>
8 <tag k="place" v="town"/>
9 <tag k="source" v="earth-info.nga.mil"/>
10 </node>
11 <node id="35135172" lat="60.8885449" lon="10.6765891" user="kao" uid="68830" visible="true"
version="6" changeset="3423173" timestamp="2009-12-21T21:22:17Z"/>
12 <node id="35135174" lat="60.8892949" lon="10.6773391" user="kao" uid="68830" visible="true"
version="6" changeset="3423173" timestamp="2009-12-21T21:22:17Z"/>
13 <node id="35135176" lat="60.8905449" lon="10.6770891" user="kao" uid="68830" visible="true"
version="6" changeset="3423173" timestamp="2009-12-21T21:22:17Z"/>
14 <node id="35135178" lat="60.8902949" lon="10.6758391" user="kao" uid="68830" visible="true"
version="6" changeset="3423173" timestamp="2009-12-21T21:22:17Z"/>
```

Figure 22: An example of XML data from OSM

C.2 Convert and Import data using FME Universal Translator

Converting OSM XML data and importing the converted data into PostGIS database through FME Universal Translator can easily be done by setting Translation parameters as shown in figure 23.

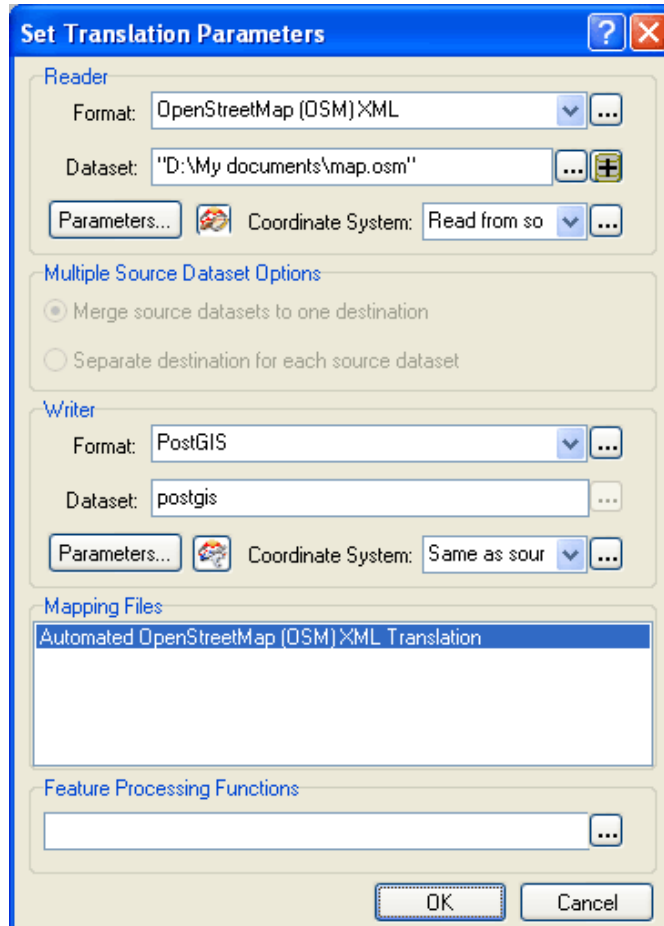


Figure 23: Using FME to convert data and write data into PostGIS

C.3 Manipulate and optimize data with PostgreSQL 8.4 / pgAdmin III

After importing data into PostGIS, we need to add a primary key into tables that we want to use with GeoServer otherwise we will get an error when we try to insert a new point, as shown in figure 24. GeoServer needs a primary key defined on each table in order to make transactions work, if our tables in the database do not have any primary keys, we will probably need to add one, for instance, we can add a serial (autoincrement) primary key to our tabl. After that, we are required to reload the configuration, and then we are supposed to be able to modify our data.

For example, the auto-increment primary key can be added as using the script below:

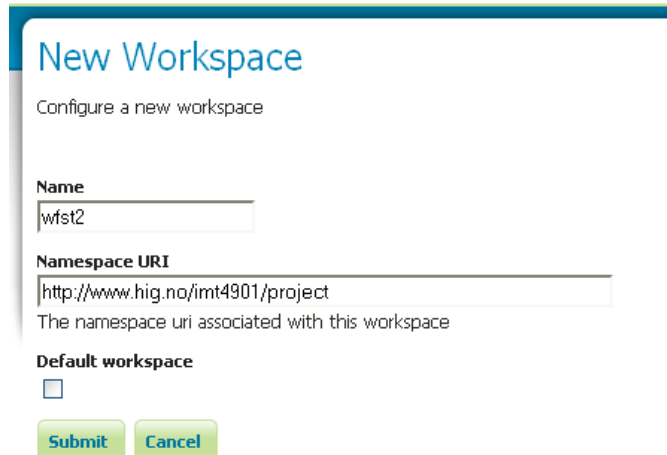
```
» alter table tourism_point add column id2 SERIAL;  
» alter table tourism_point add primary key (id2);
```

```
<?xml version="1.0" ?>  
<ServiceExceptionReport  
  version="1.2.0"  
  xmlns="http://www.opengis.net/ogc"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/ogc http://schemas.opengis.net/wfs/1.0.0/OGC-exception  
.xsd">  
  <ServiceException>  
    {http://www.hig.no/imt4901/project}tourism_point is read-only  
</ServiceException></ServiceExceptionReport>
```

Figure 24: An error occurs when adding a new point in a table with no PK available in the table

C.4 Create 'Workspace' and 'Store' in GeoServer

To use PostGIS with GeoServer, first, we will probably want to create 'Workspace' which is a container used to group similar layers together. And then, a new data source can be added by choosing 'PostGIS NG - PostGIS database' as a Vector Data Source. Figure 25 and 27 shows screenshots of adding new workspace and data store. Additionally, the document of how to create a new workspace and a store in details can be found in GeoServer.org.



New Workspace

Configure a new workspace

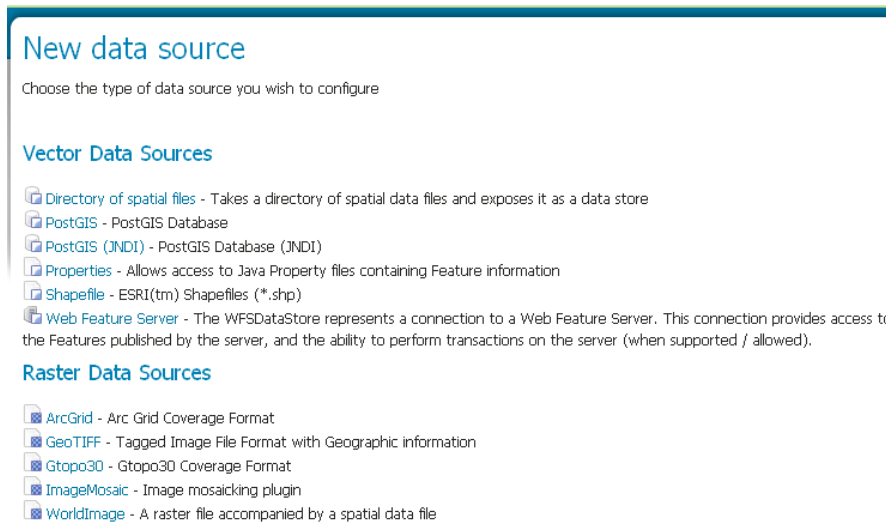
Name
wfst2

Namespace URI
http://www.hig.no/imt4901/project
The namespace uri associated with this workspace

Default workspace

Submit **Cancel**

Figure 25: Configure a workspace



New data source

Choose the type of data source you wish to configure

Vector Data Sources

- Directory of spatial files - Takes a directory of spatial data files and exposes it as a data store
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server - The WFSDataStore represents a connection to a Web Feature Server. This connection provides access to the Features published by the server, and the ability to perform transactions on the server (when supported / allowed).

Raster Data Sources

- ArcGrid - Arc Grid Coverage Format
- GeoTIFF - Tagged Image File Format with Geographic information
- Gtopo30 - Gtopo30 Coverage Format
- ImageMosaic - Image mosaicking plugin
- WorldImage - A raster file accompanied by a spatial data file

Figure 26: Available Data Sources

New Vector Data Source

PostGIS
PostGIS Database

Basic Store Info

Workspace *
wfst2

Data Source Name *
Gjovik

Description
Geographic features in Gjovik

Enabled

Connection Parameters

dbtype *
postgis

host *
localhost

port *
5432

database
postgis

schema
public

user *
postgres

passwd
●●●●●●●●

Namespace *
<http://www.hig.no/imt4901/project>

Figure 27: Data Info and Parameters of new vector data source

D OpenLayers Library

OpenLayers is a javascript-based web application which allows us to show geographic data on our web pages. To use it, we are not required to have much knowledge in javascript, but we need to respect the code and the order in which the components are declared and used.

The following code is an example of how to use OpenLayers to show map data and geographic features in this project:

```

1: <html>
2:   <head>
3:     <title>OpenLayers</title>
4:     <script src="http://openlayers.org/api/OpenLayers.js"></script>
5:     <script type="text/javascript">
6:       var lon = 5;
7:       var lat = 40;
8:       var zoom = 5;
9:       var map, select;
10:
11:       function init(){
12:         var options = {
13:           projection: new OpenLayers.Projection("EPSG:900913"),
14:           displayProjection: new OpenLayers.Projection("EPSG:4326"),
15:           units: "m",
16:           maxResolution: 156543.0339,
17:           maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
18:                                           20037508.34, 20037508.34)
19:         };
20:
21:         map = new OpenLayers.Map('map', options);
22:         var osm = new OpenLayers.Layer.TMS(
23:           "OpenStreetMap (Mapnik)",
24:           "http://tile.openstreetmap.org/", {
25:             type: 'png', getURL: osm_getTileURL,
26:             displayOutsideMaxExtent: true,
27:             attribution: '<a href="http://www.openstreetmap.org/">OpenStreetMap</a>'
28:           }
29:         );
30:         var saveStrategyPoint = new OpenLayers.Strategy.Save();
31:         var shops = new OpenLayers.Layer.Vector("Shops", {
32:           strategies: [new OpenLayers.Strategy.BBOX(), saveStrategyPoint],
33:           projection: new OpenLayers.Projection("EPSG:4326"),
34:           protocol: new OpenLayers.Protocol.WFS({
35:             version: "1.1.0",
36:             url: "/geoserver/wfs",
37:             featureType: "shop_point",
38:             srsName: "EPSG:4326",
39:             featureNS : "http://www.hig.no/imt4901/osm",
40:             geometryName: "geom"
41:           })
42:         });
43:
44:         map.addLayers([osm, shops]);
45:         map.addControl(new OpenLayers.Control.LayerSwitcher());
46:         map.zoomToExtent(
47:           new OpenLayers.Bounds(
48:             -12941319.173424, -4110122.686702, 18367287.606576, 11544180.703298
49:           ).transform(map.displayProjection, map.projection)
50:         );
51:
52:       }

```

```
53:
54:     function osm_getTileURL(bounds) {
55:         var res = this.map.getResolution();
56:         var x = Math.round((bounds.left - this.maxExtent.left) / (res * this.tileSize.w));
57:         var y = Math.round((this.maxExtent.top - bounds.top) / (res * this.tileSize.h));
58:         var z = this.map.getZoom();
59:         var limit = Math.pow(2, z);
60:
61:         if (y < 0 || y >= limit) {
62:             return OpenLayers.Util.getImageLocation() + "404.png";
63:         } else {
64:             x = ((x % limit) + limit) % limit;
65:             return this.url + z + "/" + x + "/" + y + "." + this.type;
66:         }
67:     }
68: </script>
69: </head>
70: <body onload="init()">
71:   <div id="map" ></div>
72: </body>
73: </html>
```

The steps of how the above code works can be explained as in the following terms [40]:

1. Load the OpenLayers library

```
4:     <script src="http://openlayers.org/api/OpenLayers.js"></script>
```

The `http://openlayers.org/api/OpenLayers.js` URL points to the location of a JavaScript file that loads OpenLayers. This file helps us to put a dynamic map in our web page with no server-side dependencies. The file, `OpenLayers.js`, can also be stored on our machine. Anyway, to always use the latest of the script, we may want to point a location of the file to the link as we mentioned in the example above. We can use debugging tools such as Firebug in Firefox to explore the OpenLayers library from our browser.

2. Define a container for a map

```
71:   <div id="map" ></div>
```

This is a container for the map that we create in our page markup. Later, when we initialize the "map" object, we will give the id of this div element to the map's constructor.

3. Set a display projection and create a map

```
12:     var options = {
13:         projection: new OpenLayers.Projection("EPSG:900913"),
14:         displayProjection: new OpenLayers.Projection("EPSG:4326"),
15:         units: "m",
16:         maxResolution: 156543.0339,
17:         maxExtent: new OpenLayers.Bounds(-20037508.34, -20037508.34,
18:                                           20037508.34, 20037508.34)
19:     };
20:
21:     map = new OpenLayers.Map('map', options);
```

OpenLayers allows us to set a display projection to prevent us confusion regarding map coordinates. In this case, transformation is made from the map projection *EPSG:900913* to the display projection *EPSG:4326*. Moreover, the coordinates that we use in `setCenter` are not longitude and latitude, but they are in projected units - meters. The `maxResolution` of the map defaults to fitting this extent into 256 pixels, resulting in a `maxResolution` of 156543.0339. And, for the `maxExtent` parameter, the coordinates stretch from -20037508.34 to 20037508.34 in each direction.

At line 21, we can see that there are a lot of "maps" in the statement. The first declares the variable, to give it a short name that can be used elsewhere in the Javascript code. Since the variable is declared outside the function that initializes it, it can be used globally, which helps

for debugging. The second- `OpenLayers.Map` called an object, tells the OpenLayers Javascript that we are using the `Map` class, which means that all of the built-in parameters and options can be accessed. The third is the name or identifier of the div that we are putting the map in, so OpenLayers knows where to draw the map on the page. So this section basically creates a new `OpenLayers.Map` object and gives it the variable name `map`.

4. Create Layers

```

22:     var osm = new OpenLayers.Layer.TMS(
23:         "OpenStreetMap (Mapnik)",
24:         "http://tile.openstreetmap.org/", {
25:             type: 'png', getURL: osm_getTileURL,
26:             displayOutsideMaxExtent: true,
27:             attribution: '<a href="http://www.openstreetmap.org/">OpenStreetMap</a>'
28:         }
29:     );
30:     var saveStrategyPoint = new OpenLayers.Strategy.Save();
31:     var shops = new OpenLayers.Layer.Vector("Shops", {
32:         strategies: [new OpenLayers.Strategy.BBOX(), saveStrategyPoint],
33:         projection: new OpenLayers.Projection("EPSG:4326"),
34:         protocol: new OpenLayers.Protocol.WFS({
35:             version: "1.1.0",
36:             url: "/geoserver/wfs",
37:             featureType: "shop_point",
38:             srsName: "EPSG:4326",
39:             featureNS : "http://www.hig.no/imt4901/osm",
40:             geometryName: "geom"
41:         })
42:     });

```

OpenLayers organizes a single map into several layers. This part of code tells OpenLayers that we want to create two new layers, in this case, including a *map layer* (`osm`) and a *geographic feature layer* (`shops`), which they can be referred to elsewhere as `osm` and `shops`.

5. Add layers to the map

```

44:     map.addLayers([osm, shops]);

```

This step is to add the created layers in no.4, `osm` and `shops`, to the map.

6. Create a layer switcher

```

45:     map.addControl(new OpenLayers.Control.LayerSwitcher());

```

This step is to create a layer switcher control to the map. Since earlier in the code, two layers are added to the map, we will see that "OpenStreetMap (Mapnik)" and "Shops" are listed in the control.

7. Position the map

```

46:     map.zoomToExtent(
47:         new OpenLayers.Bounds(
48:             -12941319.173424, -4110122.686702, 18367287.606576, 11544180.703298
49:         ).transform(map.displayProjection, map.projection)
50:     );

```

This code tells the map to zoom out to its maximum extent, which by default is the entire world. OpenLayers maps need to be told what part of the world they should display before anything will appear on the page. Calling `map.zoomToExtent()` is one way to do this. Alternative ways that offer more precision include using `zoomToExtent()` and `setCenter()`.

8. Load the map

```
70: <body onload="init()">
```

In order to ensure that our document is ready, it is important to make sure that the entire page has been loaded before the script creating the map is executed. We ensure this by putting the map-creating code in a function that is only called when the page's <body> element receives the onload event.

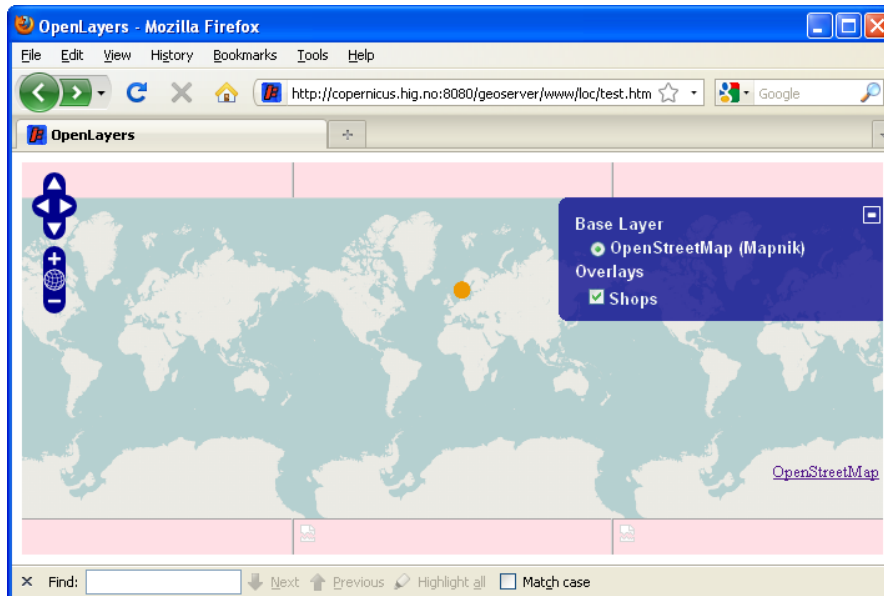


Figure 28: A page created from the given code

E W3C Geolocation API

From the W3C Geolocation API Specification [37], it defines the Geolocation API as *an API that provides scripted access to geographical location information associated with the hosting device*. The following code is an example of how we can use Geolocation API to detect a geographic location.

```
if(navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    alert('Your geographic coordinates: latitude= ' + position.coords.latitude + ',
    longitude= ' + position.coords.longitude);
  });
}
else {
  alert('Your browser does not support Geolocation API!!!');
}
```

To find out whether our browser support Geolocation or not, an object called `navigator.geolocation` is used. Once we launch the page, in case that our browsers support Geolocation, we will be asked if we will allow the application to get our location information (figure 29). If we accept the request, the browser will gather the information and provide a position object, in this code, there will be a pop-up window informing us our geographic coordinates (figure 30). On the other hand, if our browsers do not support Geolocation, we will get a pop-up window telling that our browsers do not support this feature (figure 31).

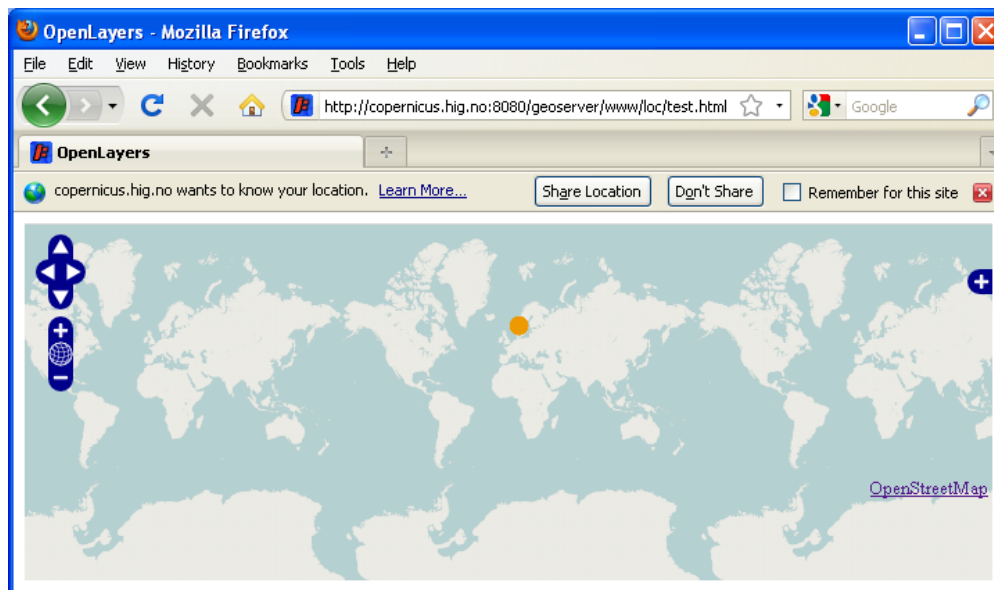


Figure 29: If a browser supports Geolocation, we will be asked whether we want to share our location information

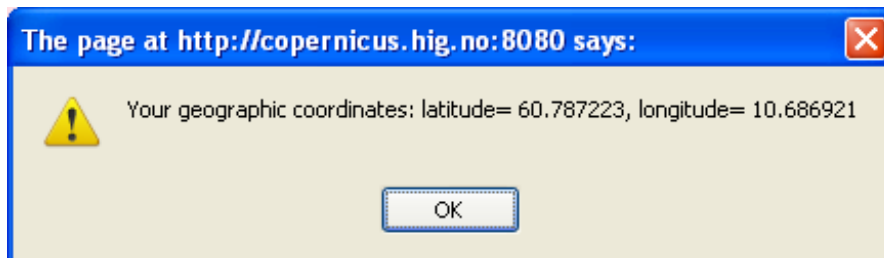


Figure 30: If a browser supports Geolocation and we allow it to gather our location information, we will get a pop-up window showing our position

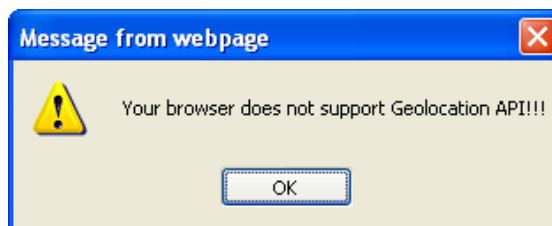


Figure 31: A pop-up window shows that our browser does not support geolocation

F A client application

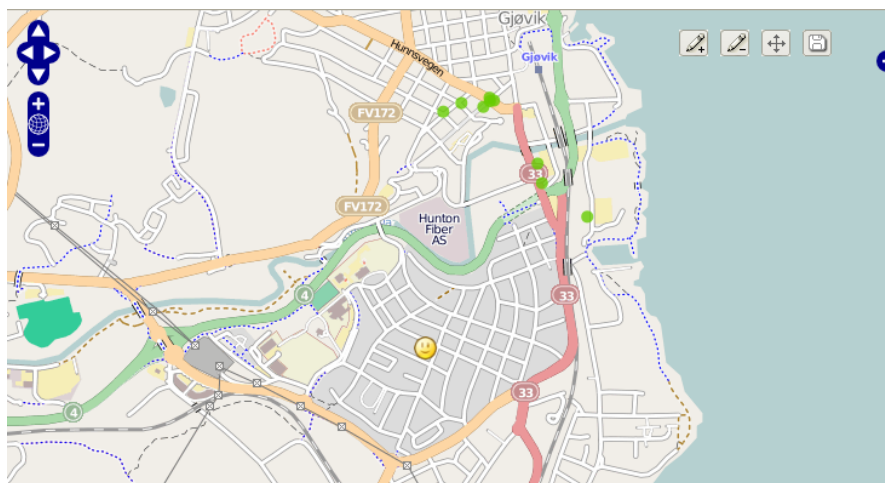
Some screenshots of a client application are revealed as shown below:

(The application can be accessed at <http://copernicus.hig.no:8080/geoserver/www/loc/imt4901.html>)

1. When a user launches an application, if a browser supports Geolocation, she will be asked whether she wants to share a location.



2. If the user accept the request, there will be a pop-up message informing her position as shown earlier in figure 30. After that, she will see a map with some symbols. A yellow smiley icon represents her location while green dots represent nearby shops.



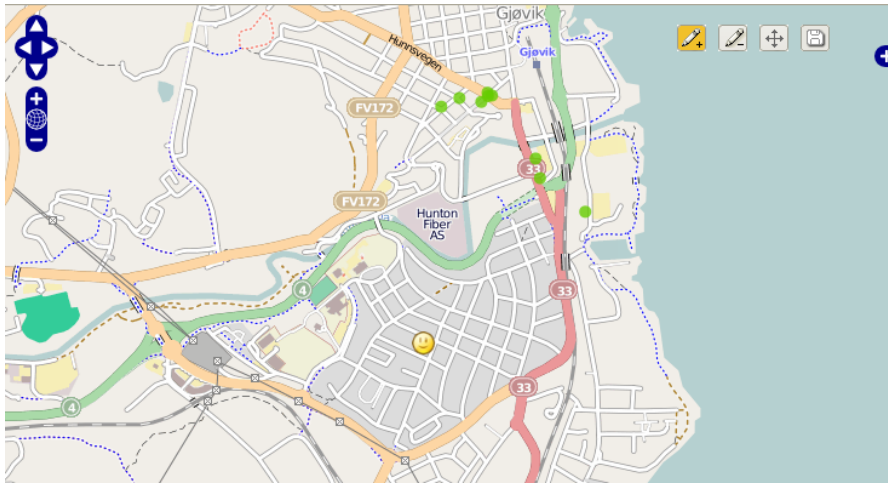
3. The below code shows a background process for requesting geographic features from a WFS server.

```
<wfs:getfeature xmlns:wfs="http://www.opengis.net/wfs" service="WFS" version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wfs:query typeName="feature:shop_point" srsName="EPSG:4326" xmlns:feature="http://www.hig.no/imt4901/osm">
    <ogc:filter xmlns:ogc="http://www.opengis.net/ogc">
      <ogc:bbox>
        <ogc:propertyname>geom</ogc:propertyname>
        <gml:envelope xmlns:gml="http://www.opengis.net/gml" srsName="EPSG:4326">
          <gml:lowercorner>-332.11322310912 -104.88306911346</gml:lowercorner>
          <gml:uppercorner>380.85552686241 135.26592890733</gml:uppercorner>
        </gml:envelope>
      </ogc:bbox>
    </ogc:filter>
  </wfs:query>
</wfs:getfeature>
```

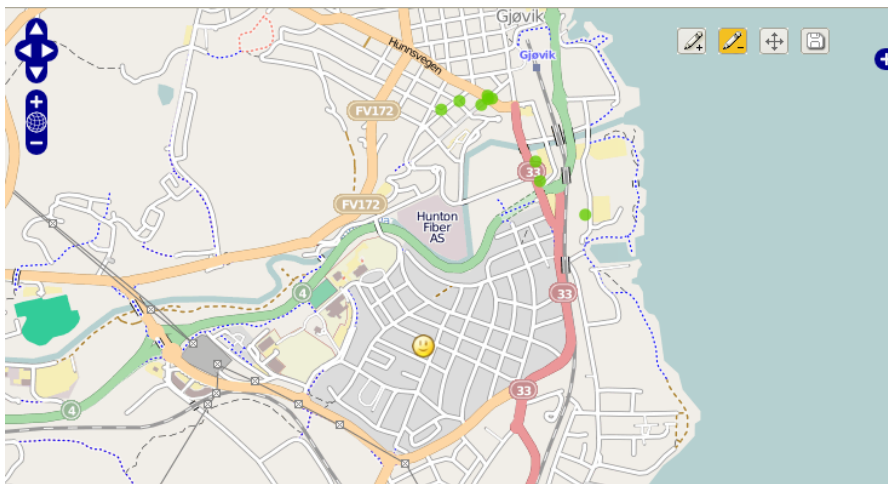
4. The below code shows a background response from requesting geographic features from a WFS server.

```
<wfs:featurecollection numberofFeatures="9" timeStamp="2010-07-01T20:32:12.290+02:00" xsi:schemaLocation="http://copernicus.hig.no:8080/geoserver/wfs?service=WFS&version=1.1.0&request=DescribeFeatureType&typeName=osm%3Ashop_point" http://www.opengis.net/wfs http://copernicus.hig.no:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http://www.opengis.net/ogc" xmlns:osm="http://www.hig.no/imt4901/osm" xmlns:tiger="http://www.census.gov" xmlns:wfs="http://www.opengis.net/wfs" xmlns:topp="http://www.openplans.org/topp" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="http://www.openplans.org/spearfish" xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <gml:featuremembers>
    <osm:shop_point gml:id="shop_point.1">
      <gml:name>Kiwi Strandgata </gml:name>
      <osm:id>4.88515702E8</osm:id>
      <osm:opening_hours>Mo-Fr 07:00-23:00; Sa 07:00-21:00 </osm:opening_hours>
      <osm:user>StianEllingsen </osm:user>
      <osm:shop>supermarket </osm:shop>
      <osm:timestamp>2009-09-12T21:12:37Z </osm:timestamp>
      <osm:visible>true </osm:visible>
      <osm:phone>+47 61183360 </osm:phone>
      <osm:name>Kiwi Strandgata </osm:name>
      <osm:geom>
        <gml:multiptoint srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
          <gml:pointmember>
            <gml:point>
              <gml:pos>10.6948648 60.7935921</gml:pos>
            </gml:point>
          </gml:pointmember>
        </gml:multiptoint>
      </osm:geom>
    </osm:shop_point>
    <osm:shop_point gml:id="shop_point.2">
      <gml:name>Rema 1000 Gjøvik Sentrum </gml:name>
      <osm:id>4.94458627E8</osm:id>
      <osm:opening_hours>Mo-Fr 07:00-22:00; Sa 08:00-21:00 </osm:opening_hours>
      <osm:addrhousenumber>1 A </osm:addrhousenumber>
      ...
      ...
    </osm:shop_point>
```

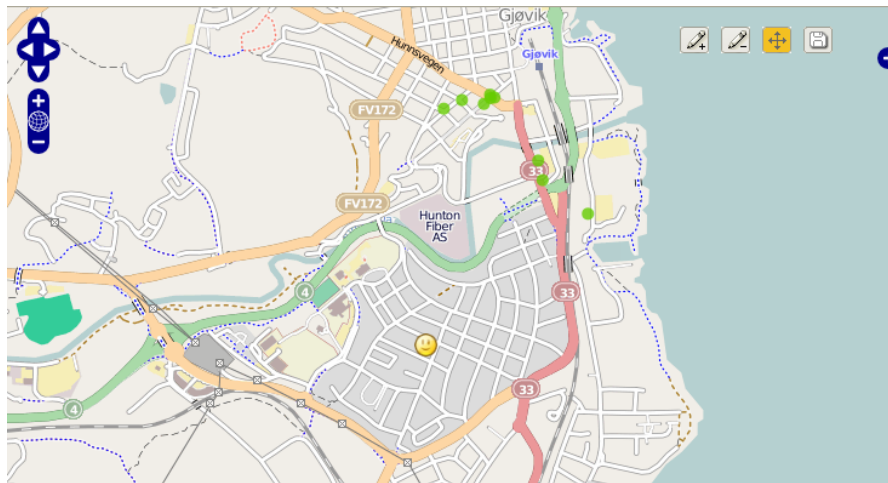
5. The user is able to add a new shop by selecting a *pencil with a plus sign* button, and then locate the new shop on the map.



6. The user is able to delete green dots by selecting a *pencil with a minus sign* button, and then select the point to delete.



7. The user is able to move green dots by selecting a *move* button, and then select the point to move.



8. The user can save the transaction by selecting a *save* button. The below code shows a background response when the user *adds a new point* on the map and selects the save button.

```
<wfs:transactionresponse version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs
http://copernicus.hig.no:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http:
//www.opengis.net/ogc" xmlns:osm="http://www.hig.no/imt4901/osm" xmlns:tiger="http:
//www.census.gov" xmlns:wfs="http://www.opengis.net/wfs" xmlns:topp="http://www.openplans.org
/topp" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="http://www.openplans.org
/spearfish" xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http:
//www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <wfs:transactionsummary>
    <wfs:totalinserted>1</wfs:totalinserted>
    <wfs:totalupdated>0</wfs:totalupdated>
    <wfs:totaldeleted>0</wfs:totaldeleted>
  </wfs:transactionsummary>
  <wfs:transactionresult></wfs:transactionresult>
  <wfs:insertresults>
    <wfs:feature>
      <ogc:featureid fid="shop_point.11"></ogc:featureid>
    </wfs:feature>
  </wfs:insertresults>
</wfs:transactionresponse>
```

9. The user can save the transaction by selecting a *save* button. The below code shows a background response when the user *deletes a point* on the map and selects the save button.

```
<wfs:transactionresponse version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs
http://copernicus.hig.no:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http:
//www.opengis.net/ogc" xmlns:osm="http://www.hig.no/imt4901/osm" xmlns:tiger="http:
//www.census.gov" xmlns:wfs="http://www.opengis.net/wfs" xmlns:topp="http://www.openplans.org
/topp" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="http://www.openplans.org
/spearfish" xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http:
//www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <wfs:transactionsummary>
    <wfs:totalinserted>0</wfs:totalinserted>
    <wfs:totalupdated>0</wfs:totalupdated>
    <wfs:totaldeleted>1</wfs:totaldeleted>
  </wfs:transactionsummary>
  <wfs:transactionresults></wfs:transactionresults>
  <wfs:insertresults>
    <wfs:feature>
      <ogc:featureid fid="none"></ogc:featureid>
    </wfs:feature>
  </wfs:insertresults>
</wfs:transactionresponse>
```

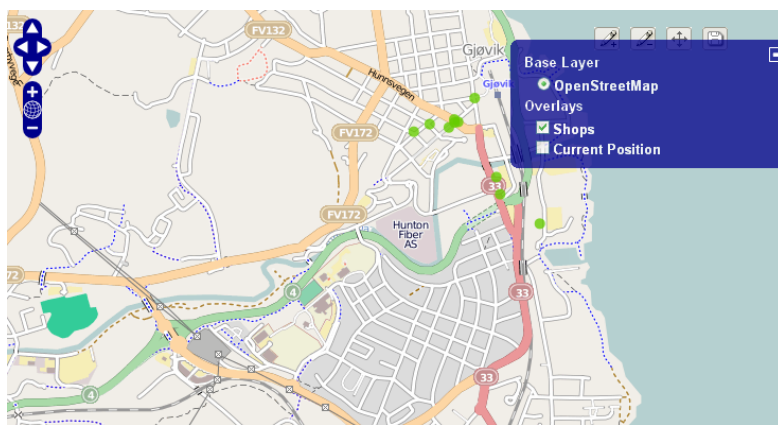
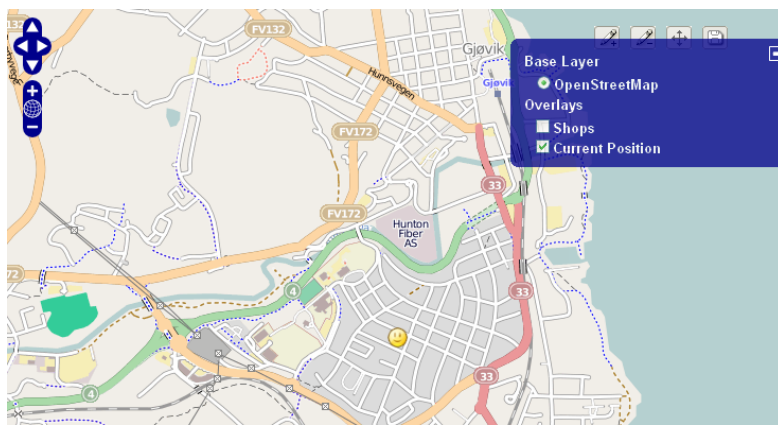
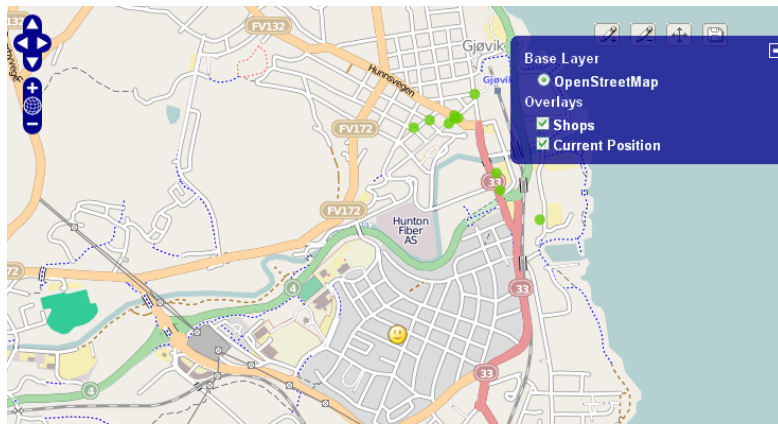
10. The user can save the transaction by selecting a *save* button. The below code shows a background response when the user *moves a point* on the map and selects the save button.

```
<wfs:transactionresponse version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs
http://copernicus.hig.no:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http:
//www.opengis.net/ogc" xmlns:osm="http://www.hig.no/imt4901/osm" xmlns:tiger="http:
//www.census.gov" xmlns:wfs="http://www.opengis.net/wfs" xmlns:topp="http://www.openplans.org
/topp" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="http://www.openplans.org
/spearfish" xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http:
//www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <wfs:transactionsummary>
    <wfs:totalinserted>0</wfs:totalinserted>
    <wfs:totalupdated>1</wfs:totalupdated>
    <wfs:totaldeleted>0</wfs:totaldeleted>
  </wfs:transactionsummary>
  <wfs:transactionresults></wfs:transactionresults>
  <wfs:insertresults>
    <wfs:feature>
      <ogc:featureid fid="none"></ogc:featureid>
    </wfs:feature>
  </wfs:insertresults>
</wfs:transactionresponse>
```

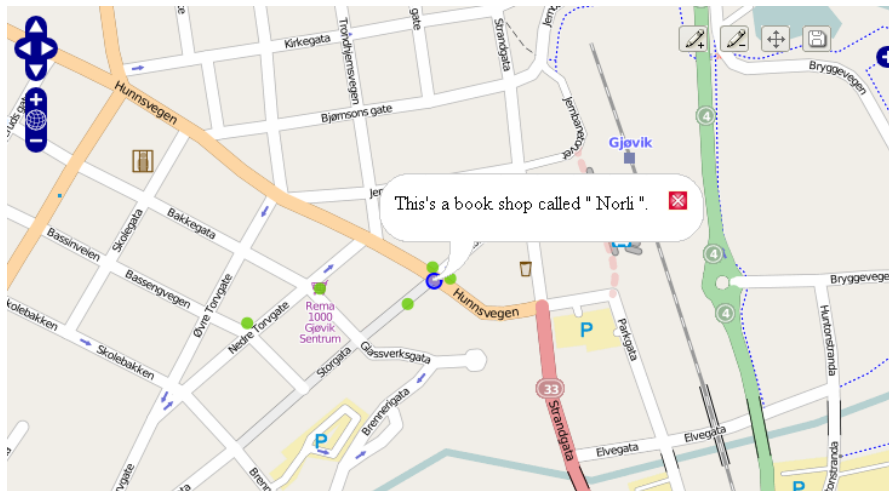
11. The user can save the transactions by selecting a *save* button. The below code shows a background response when the user *adds 3 new points, deletes 1 point, and moves 1 point* on the map, and then selects the *save* button.

```
<wfs:transactionresponse version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs
http://copernicus.hig.no:8080/geoserver/schemas/wfs/1.1.0/wfs.xsd" xmlns:ogc="http:
//www.opengis.net/ogc" xmlns:osm="http://www.hig.no/imt4901/osm" xmlns:tiger="http:
//www.census.gov" xmlns:wfs="http://www.opengis.net/wfs" xmlns:topp="http://www.openplans.org
/topp" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="http://www.openplans.org
/spearfish" xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http:
//www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink">
  <wfs:transactionsummary>
    <wfs:totalinserted>3</wfs:totalinserted>
    <wfs:totalupdated>1</wfs:totalupdated>
    <wfs:totaldeleted>1</wfs:totaldeleted>
  </wfs:transactionsummary>
  <wfs:transactionresults></wfs:transactionresults>
  <wfs:insertresults>
    <wfs:feature>
      <ogc:featureid fid="shop_point.18"></ogc:featureid>
    </wfs:feature>
    <wfs:feature>
      <ogc:featureid fid="shop_point.19"></ogc:featureid>
    </wfs:feature>
    <wfs:feature>
      <ogc:featureid fid="shop_point.20"></ogc:featureid>
    </wfs:feature>
  </wfs:insertresults>
</wfs:transactionresponse>
```

12. The user is able to choose whether she wants to see her location (Current Position) and/or nearby shops (Shops) on the map.



13. The user can select on any of geographic features (green dots) to see its description. Once the user does so, a pop-up message will appear with the geographic description.



14. The user can pan and/or zoom the map by selecting the buttons on the top left. For the below case, the user zooms in the map, as a result, she can see the map in more details.

