Integrating Spatial Information into JSF Java EE Web Applications with *GeoJSF*

Thorsten Kisner, Helge Hemmer and Klaus Jacobi
AHT GROUP AG
Management & Engineering
Essen, Germany
Email: {*t.kisner,h.hemmer,jacobi*}*@aht-group.com*

***Abstract*—In recent years an increasing acceptance of geographical information systems (GIS) to be used by clients in their web browsers and therefore a high demand for such WebGIS systems is observable. The processing and visualization of spatially referenced data became more and more important, even in fields of applications that are not traditional GIS domains like earth observation or remote sensing. Management information systems (MIS) or social networks among others have also evolved requirements for GIS features. For implementing enterprise architectures, many powerful frameworks and (Open Source) products are available on the market; considering the Java programming language in its enterprise edition (JavaEE) for example, several compliant application servers can be used, along different solutions to share geospatial data and many ways to work with these data in the clients web browser. Being in this concrete scope of developing a Java EE software with a presentation layer implemented in Java Server Faces (JSF), which is part of the Java EE standard specification, one quickly realizes the lack of a comprehensive integration of GIS functionality into JSF. In this paper, we propose a framework to develop web applications with geospatial elements by an own implementation of a component library for JSF technology, enabling the developer to enrich his application with powerful GIS features at ease.**

***Keywords*-WebGIS; WMS; WFS; JSF; JavaEE; integration; component library; spatial data**

I. Introduction

Compared to traditional desktop geographical information systems (GIS), a web based GIS based Open Source tools has several advantages. WebGIS solutions have a much broader accessing scope, the system can be accessed without the need of software installations, maintenance and licensing on the client side. Client software is (mostly) independent from platform and architecture, only a modern[1] web browser is required to access the information. Commercial GIS products with licensing costs and annual maintenance fees like *ArcIMS* offer state of the art technology with a rich variety of features, even if only a limited number of functionalities is used. If existing applications should be extended only with standard GIS functionalities or just a simple map, commercial GIS suites might be an oversized solution.

Either way – out-of-the-box solutions always require individual customizing. In both cases, using commercial or open source based software with free available libraries and products, detailed programming skills are required.

In the developing domain of JavaEE applications with JSF for it's presentation layer, developers are used to include existing (component) libraries into their applications to add the required features. For WebGIS applications, a JSF component library offering a seamless integration with features of user interaction would be highly appreciated. Such a framework is presented in this paper.

The remainder of this paper discusses the related work and positions the proposed approach (Section II). Section III introduces the Java Server Faces framework in which domain our proposal is implemented followed by a discussion of required GIS standards in Section IV. The implementation itself is discussed from a technical point of view in Section V. Section VI presents results and ends with an overview on future work.

II. Related Work

In many software domains, classic desktop application are replaced by rich internet applications (RIA). These products are running on a server and are accessed by clients through their Internet browser, giving them nearly the same user experience than in a desktop system. The same applies to geographical information systems (GIS). When common users of todays information systems come across the keyword "geographical information", most associate this with *Google Maps*, a web based GIS launched by Google in 2006. It is an obvious thought, because simple GIS features in web sites are often realized through *Google Maps*.

Even more complex information systems are build on top *Google Maps* as a viewer for geo-spatial data. In [1], it is used, besides other technologies, to show water resources on a map. Data is processed inside the application and visualized using a KML (Keyhole Markup Language) file through the *Google Maps API*. In other scenarios, information systems need to use their own maps and advanced features which *Google Maps* does not offer. Especially when it comes to infrastructure provision, there are often higher requirements to the GIS functionality.

Therefore, a lot of work was done to implement custom

[1]With activated JavaScript or ActiveX.

WMS/WFS[2] clients. For example, a system to monitor and plan pipelines of a urban drainage network is described in [2]. The GIS component of their project is realized through a Java Applet that is included on a web page.

Meeting the requirements of nowadays software projects, recent developments are more and more integrating AJAX (Asynchronous JavaScript and XML) features to the GIS functionality. Whereas [3] is combining *Google Maps* with extra information from Geo Web Services, a “Geo Stack” of several tools with a front-end utilizing the *OpenLayers* library was built up in [4]. The capabilities of *OpenLayers* are inspected in [5], pointing out that it can replace a classical desktop GIS client.

Geographical data processing is likely being only one component of a comprehensive information system, therefore it is needed to be integrated in larger infrastructures. One approach is a Service Oriented Architecture (SOA), like it is done in [6]. The implementation presented there is accessing WMS/WFS servers through XML Web Services. Respectively in [7], where JavaEE patterns are used to develop a system with an Enterprise Service Bus. A SOA environment in conjunction with a service to offer Styled Layer Definition files for *OpenLayers* is presented in [8]. A SOA style service bus is presented in [9] for sensor data in an application of Tsunami warning.

Besides these approaches to integrate applications as parts in a wide infrastructure, [10] concentrates on the joining of spatial data of different sources in one map using open source software like *OpenLayers* and *Mapserver*.

Another kind of integration is proposed by [11] using the commercial JSF Application Development Framework (ADF) by ESRI to create WebMap applications inside web front-ends. The same client is also used in many other web GIS like NASAs *Apollo Analysts Notebook* [12]. Besides *OpenLayers* and the ESRI product, there are other options like *MapXTreme*, which is inspected in [13].

The approach presented in this paper is extending the ideas of *geo-faces*[3] and *ol4jsf*[4], both projects providing a JSF component to integrate the *OpenLayers* client into a JavaEE web application. Like *geo-faces*, our component library *GeoJSF* uses the RichFaces Component Development Kit (CDK) – a toolkit to simplify the implementation of custom JSF components.

Whereas *geo-faces* and *ol4jsf* are basically offering a JSF tag to initialize an OpenLayers instance, *GeoJSF* is extending this approach with an integration framework and offers AJAX based interactions between the JavaScript client running on the client side and the JavaEE system on the server side with the additional integration of Enterprise Java Beans (EJB).

[2]WMS and WFS is discussed in Section IV.

[3]http://code.google.com/p/geo-faces

[4]https://ol4jsf.dev.java.net

Other proposals like [14], which is trying to find a universal information representation for geographical data, are a good option when having a heterogenous application environment. A novel approach to query spatial data is presented in [15], but concluding that object-relational mapping is the method of choice in typical JavaEE systems.

For other technologies not residing in the JavaEE framework, [16] presents novel ideas for implementing a web based application with techniques like Scalable Vector Graphics (SVG), whereas [17] presents an overview to build up a geo portal using Open Source technology.

III. Java Server Faces

Java Server Faces (JSF) is a server-side framework to develop the web enabled presentation layer of an enterprise application. Developers can use predefined and reusable component libraries (like the one we describe in our paper), link component elements to data sources managed by the server and connect client-side events to a server-side event handler.

The reference implementation for JSF is included in the Java Specification Request 127 [18] and includes simple GUI elements like links, buttons or input fields which have direct equivalents to tags of the HTML specification.

The components included in the reference implementation are sufficient to create web applications, but the web interface will only appear in an old fashioned style like web sites in the 90’s. Modern web sites or rich internet applications require high interactivity and have to satisfy graphical layouts (“Web 2.0”). There are many sophisticated components libraries available like *ICEfaces* from ICEsoft, *RichFaces* from Red Hat or *Trinidad* from the Apache Software Foundation which fulfill these s requirements.

A. JSF Architecture

The component framework is a Model-View-Controller (MVC) architecture shown in Figure 1.

The *Controller* is responsible for the navigation and user interaction and is implemented in the `FacesServlet`. Developers can define navigation rules and assign event handlers to different components.

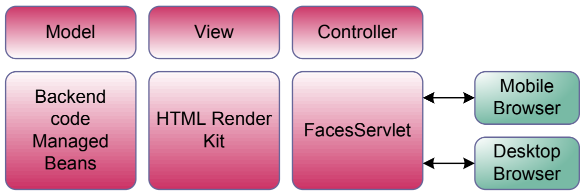

Figure 1. MVC architecture with JSF

The *Model* is represented by “Managed Beans” and is the Java part of JSF. Managed Beans are invoked by *View* components and are implemented in a POJO[5] concept. The

[5]Plain Old Java Object

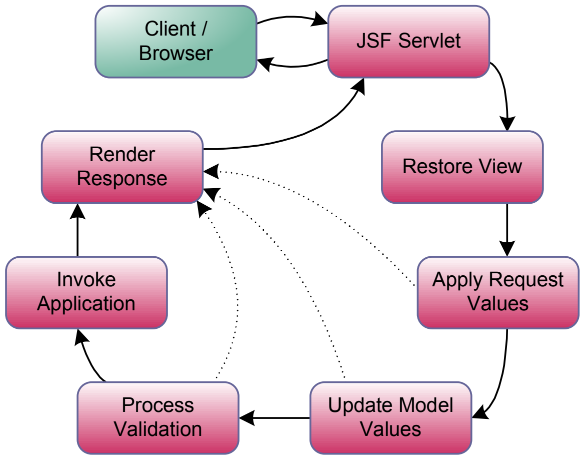

Figure 2. Life Cycle of a JSF application (short-circuited by errors)

View components itself are provided the by components libraries.

B. JSF Life Cycle

The different phases of a JSF request/response turn are outlined in Figure 2.

The state of the component tree is restored from the previous request in the *Restore View* phase, in the next phase *Apply Request Values* all new values are applied to the corresponding components. Before the model values are updated, validations and converters are invoked in phase three. After updating model values, the phase *Invoke Application* will be entered and after processing of all events, the response is rendered and sent back to the client.

There are shortcuts available from the first phases to *Render Response*. They are called if no query data are available or errors in the validation or conversion phase occur.

C. Implementation of components

A JSF component is a set of Java classes and XML configuration files. To develop a component, the followings steps are required and undertaken in our component library:

1) Implementation of a Java class which extends the basic component classes of JSF.
2) Implementation of a renderer class for the default render kit of the reference implementation which creates the output of the specific class.
3) Implementation of a class which describe the tags for the JSP[6] class.
4) Definition of the *Tag Library Definition* (TLD) file. This file describes all available tags with their required or optional attributes and allowed types.
5) The renderer of each component is defined in the *JSF Configuration* file. For each component the family (defined in the JSF standard), a Uniform Resource Identifier (URI) for the component and the actual renderer class defined above must be specified.

All elements, the three individual Java classes per element and both configuration files, will be provided by our JSF library as a JAR archive which directly can be used by web application developers. The archive must be available in the class path of the servlet container and the library must be declared in the head of the JSF file like shown in Listing 1.

Listing 1. Usage of *GeoJSF*

```
<?xml version=”1.0” encoding=”utf-8” ?>
<jsp:root version=”2.1”
   xmlns:html=”http://www.w3.org/1999/xhtml”
   xmlns:jsp=”http://java.sun.com/JSP/Page”
   xmlns:f=”http://java.sun.com/jsf/core”
   xmlns:h=”http://java.sun.com/jsf/html”
   xmlns:a4j=”http://richfaces.org/a4j”
   xmlns:rich=”http://richfaces.org/rich”
   xmlns:geojsf=”http://geojsf”>
<jsp:output doctype-root-element=”html”
   doctype-public=”-//W3C//DTD XHTML 1.1//EN”
   doctype-system=
       ”http://www.w3c.org/TR/xhtml11/DTD/xhtml11.dtd” />
<jsp:directive.page contentType=”text/html; charset=utf-8”
                        language=”java” />
<html>
  <jsp:directive.include file=”../include/head.jspx”/>
  <body><f:view><div><rich:layout>
  <rich:layoutPanel position=”top”><h:form><rich:panel>
<geojsf:map id=”olMap” serviceName=”LCBC”
   overviewMapDiv=”dOverview”
   width=”500” height=”400”
   serviceURL=”#{MapsBean.ol.wmsLayer.value.url}”
   layerOptions=”#{MapsBean.ol.wmsLayer.value.params}”
   centerX=”#{MapsBean.ol.view.x}”
   centerY=”#{MapsBean.ol.view.y}”
   zoomLevel=”#{MapsBean.ol.view.zoom}”>
</geojsf:map>
   </rich:panel></h:form></rich:layoutPanel>
   </rich:layout></div></f:view></body>
</html>
</jsp:root>
```

The map is included with the `<geojsf:map/>` tag and most of the attributes are bound to the managed bean `MapsBean` with the JSF Expression Language.

D. JSF Expression Language

JSF-EL is similar (but not compatible) with JSP-EL and is processed in the *Update Model Values* and *Render Response* phases (see Figure 2). The navigation to object properties is analogue to the XML Path Language (XPath) and is embedded in the syntax `#{...}`. The expression itself can be object-value bindings, arithmetic or logical expressions and method bindings.

IV. Geographical Information Systems

A geographic information system (GIS) (aka geospatial information system) is a system that captures, stores, analyzes, manages, and presents data that are linked to location.

The most important standards in this area are the ISO series 191xx[7] and publications from the Open Geospatial Consortium (OGC) [19].

[6]Java Server Pages

[7]ISO 19107, 19109, 19111, 19115, 19136

The OGC has developed different specifications for data exchange, including the Web Map Service (WMS) to encode maps as images, the Web Feature Service (WFS) for working with data referenced by geographic objects or vector data, the Web Coverage Service (WCS) for continuous data, live access to observations from sensors with the Sensor Collection Service (SCS) and the Geographic Markup Language (GML) to encode geographic objects and linked data for transport in XML data.

Due to the relevance of WMS and WFS to *GeoJSF* these standards are discussed in detail in the following sections.

A. Web Map Service

Being a special case of a Webservice, a Web Map Service (WMS) is an interface specification published by the OGC [20] to request and serve map images in the internet. An OGC compliant WMS uses Hypertext Transfer Protocol (HTTP) for transport and has to implement three functionalities.

A client can ask with `GetCapabilities` for general properties of the server like available output formats[8] or available layers for a specific map. The response is send to the client in a XML document.

The concrete map images of a spatial referenced map are requested with a `GetMap` query. The query can include optional parameters like the geospatial reference system, image size, map section or output format. A map image can directly requested by any web browser (wiht plain HTML), but intermediate clients (e.g. JavaScript) offer possibilities to change layers, zoom or move the map.

The optional `GetFeatureInfo` answers queries to specific coordinates or selected areas (with an optional search radius) in the actual map section. The response contains thematic information of the map (layer). The response can be offered in XML or formatted in HTML.

B. Web Feature Service

The Web Feature Service (WFS) [21] defined by OGC specifies an interface to access spatial data on distributed GIS systems. Other than the WMS, only access to vector data is possible.

OGC compliant WFS server support six functionalities, use HTTP for transport and encode the response with the XML based Geography Markup Language (GML).

The `GetCapabilities` response contains information on available feature types and supported operations. `DescribeFeatureType` provides detailed information on the structure of a feature type.

The spatial referenced data itself are requested by `GetFeature`, the response object can be filtered with given feature types or spatial dependencies. The GML encoded response object can be used to create a XLink[9] query to access further elements with `GetGmlObject`.

Write access to the provided data is provided with `Transaction`, this supports a revertable transaction to create, modify and delete objects. The `LockFeature` operation during a transaction prevents a concurrent write access to the specified object by other instances.

V. THE JSF COMPONENT LIBRARY GEOJSF

A. Architecture

In our scenario we are dealing with a distributed service architecture and different domains of information processing like shown in figure 3.

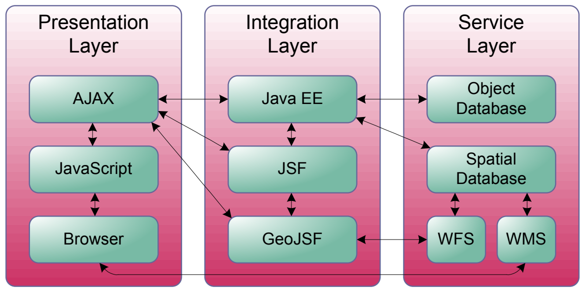

Figure 3. Layered architecture and components

- *Browser (Java Script):* The client side logic is located in JavaScript code running in a web browser. JavaScript is responsible for the client side event handling and the execution of server side commands. Data is transferred between client and server by JavaScript functions utilizing the AJAX framework. This allows partial page rendering and a high response time of the application.
- *Java EE Application Server:* The Enterprise Java Bean (EJB) specification is extended by spatial data types using HibernateSpatial. With this extension objects can be directly modelled on the EJB side with spatial characteristics like points, polygons or multi-polygons.

Listing 2. EJB annotation for spatial data types

```
@Column(name="the_geom")
@Type(type="org.hibernatespatial.GeometryUserType")
private Point geometry;
```

The presentation layer (graphical user interface) is implemented with JSF technology.

- *WMF/WFS Server:* We use GeoServer[10], an Open Source server written in Java, which is the reference implementation of the Open Geospatial Consortium WFS, WMS and WCS specification. Since the required protocols WFS and WMS are standardized, every OGC

[8]Raster images or vector formats like Scalable Vector Graphics (SVG) and Web Computer Graphics Metafile (WebCGM).

[9]The XML Linking Language (XLink) is used to create internal and external links within XML documents.

[10]http://www.geoserver.org

compliant server like the UMN MapServer[11] or Degree[12] can be used.

- *Spatial Database:* As a spatial database PostgreSQL in conjunction with *PostGIS*, an open source addition to PostgreSQL to make the database capable of using geographic objects, is used. In 2006, PostGIS was certified as a compliant “Simple Features for SQL” database by the Open Geospatial Consortium.
  The geographic objects represented by *HibernateSpatial* are stored in the *Well-Known Text* (WKT) format. Compared to the standard PostgreSQL database engine additional geometric functions (like spatial joins, intersections etc.) are added to the database as well as spatial reference systems describing the geodetic datum, geoid, coordinate system and map projection of the spatial objects.

Beside the client side components (JavaScript) and the JSF component library itself, *GeoJSF* consists of several classes supporting the managed beans on the application server side. This includes event handlers for client-side events as well as data structures representing all map related objects. Client events with spatial information like point queries (a click on the map with a individual search radius) or the selection of an area with a traverse are forwarded to customizeable factory classes building a WFS query which is send to the WFS server. The result is interpreted and the corresponding entity objects (Entity EJB) are instantiated. While these objects are under control of an `EntityManager`[13] the complete object tree is available with all (sub) child elements without any additional effort for programmers, the elements are loaded on demand from the database.

B. User Interaction

The user interaction and integration of WebGIS components into a JSF application can be easily explained in the following example in Figure 4. The screenshot is taken from a geographic information system (with integrated water resource management) for the Lake Chad Basin Commission (LCBC) in West Central Africa funded by the the German Society for Technical Cooperation (GTZ).

The page contains different elements, the menu bar on the top and both panels on the right side are standard JSF components of the component library *RichFaces*, the chart at the bottom is generated by *JFreeChart*. The map on the left side as well as the overview map are generated by *GeoJSF* allowing interactions to and from other JSF components.

1) The user can select/deselect different map layers using the *GeoJSF* Layer-Control, the available layers are defined for each thematic map and legend symbols are

[11]http://mapserver.org

[12]http://deegree.org

[13]The `EntityManager` is part of the Java Persistence API (JPA) and located in the application server.

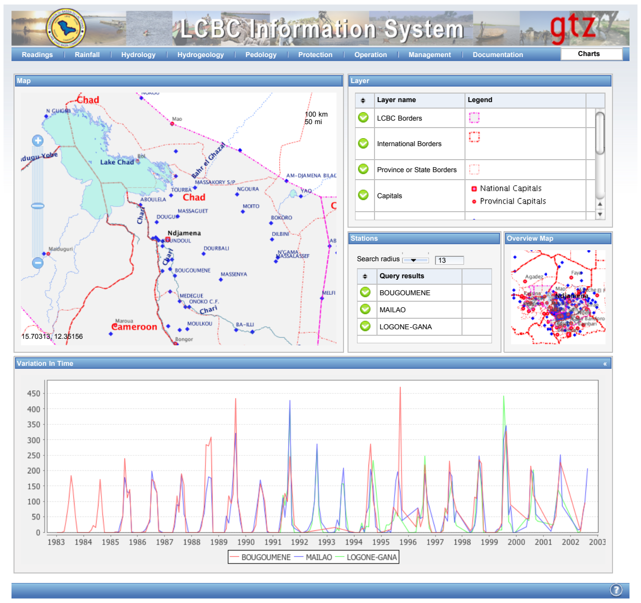

Figure 4. Example of a JSF application with *GeoJSF* maps

generated on the fly by the Styled Layer Descriptor (SLD) of the actual layer.

2) Changes in the layer selector will immediately re-render the interactive map on the left side. The map is based on *OpenLayers* to allow scrolling and zooming and is enhanced with controls to interact with the JSF backend at the server.
3) Movements in the map component or selections in the overview map are directly reflected on the corresponding map.
4) A click on the map invokes the JSF backing bean and a WFS query is created with the actual coordinates and search radius. In this example a list of all rainfall stations is retrieved, the corresponding EJB entity beans are instantiated and the table is populated with the station name.
5) Optionally the stations to be included in the chart generation can be selected and deselected.
6) The chart for the measured rainfall of all selected stations in the specified time interval is shown in the chart panel at the bottom and immediatly updated by new map or station selections.

VI. Conclusion and Future Work

The JavaEE platform is a feature rich framework fulfilling all requirements to build up robust and scalable enterprise applications. During our developments with this technology on various information systems, ranging from water information systems, management information systems or systems for enterprise resource planning, we were faced with the client’s demand on processing and visualization of spatial data. As a matter of course we did not want to change the technology or switch to commercial solutions. Realizing that at this time no free available product was available satisfying our needs, we decided to develop *GeoJSF* in the beginning of 2010.

Our experiences show that *GeoJSF* easily enables Java EE programmers to include interactive maps in their JSF applications. The separated development domains for Java programmers dealing with Java EE on one side and GIS experts on the other side dealing with spatial information can easily consolidated with a XML based definition for thematic maps. These maps usually contain of different layer (groups) and can be customized by SLD totally independent of the Java development.

Currently we are preparing to publish the library under an Open Source license in the SourceForge software repository. This includes a considerable effort of documentation and usage examples. Other objectives are a more comprehensive interaction between map elements and JSF backing beans, e.g. markers or thematic overlay elements.

References

[1] C. Granell, L. Diaz, and M. Gould, "Geospatial web service integration and mashups for water resource applications," in *Proceedings of ISPRS Congress Beijing 2008*, vol. XXXVI-I/B4/IV. International Society of Photogrammetry and Remote Sensing, 2008.

[2] J. Y. C.T. Yang, L. Ye, "A framework of web-gis for urban drainage network system," in *International Conference on Semantic Web & Web Services*, 2006.

[3] A. Sayar, M. Pierce, and G. Fox, "Integrating ajax approach into gis visualization web services," in *Proceedings of IEEE International Conference on Internet and Web Applications and Services ICIW'06 February 23-25, 2006*, 2006.

[4] S. A. Romalewski, "Rich interactive mapping experience through open source frameworks frameworks and ajax data visualization techniques," in *Proceedings of the ISPRS working group III/4, IV/8, IV/5*, T. H. Kolbe, H. Zhang, and S. Zlatanova, Eds. International Society of Photogrammetry and Remote Sensing, 2008.

[5] J. She, Q. Chen, S. Pan, and X. Feng, "Monitoring land use of construction based on webgis with ria technology," *Information Science and Engineering, International Conference on*, pp. 2104–2108, 2009.

[6] P. S. Talegaonkar, "Service oriented architecture for gis applications," in *The 12 International Conference of International Association for Computer Methods and Advances in Geomechanics (IACMAG)*, 2008.

[7] A. M. S. Fabiana Soares Santana, "Soc & soa in ecological niche modelling and agribusiness: Discussion and case studies," in *7th World Congress on Computers in Agriculture Conference Proceedings*, 2009.

[8] P. Townend, J. Xu, M. Birkin, A. Turner, and B. Wu, "Modelling and simulation for e-social science through the use of service-orientation and web 2.0 technologies," in *Proceedings of the 4th International Conference on e-Social Science*, 2008.

[9] J. Fleischer, R. Häner, S. Herrnkind, A. Kloth, U. Kriegel, H. Schwarting, and J. Wächter, "An integration platform for heterogeneous sensor systems in gitews: Tsunami service bus," *Natural Hazards and Earth System Sciences*, vol. 10, pp. 1239–1252, Jun. 2010.

[10] J. She, S. Pan, Q. Chen, X. Feng, H. Jiang, and K. Xiao, "Web-based integrative presentation of distributed spatial data," *Information Science and Engineering, International Conference on*, pp. 2233–2237, 2009.

[11] H. Lu, W. Nihong, W. Chang, and C. Yujia, "The research on the webgis application based on the j2ee framework and arcgis server," *Intelligent Computation Technology and Automation, International Conference on*, vol. 3, pp. 942–945, 2010.

[12] J. Wang, T. C. Stein, T. Heet, D. M. Scholes, R. E. Arvidson, and V. Heil-Chapdelaine, "A webgis for apollo analyst's notebook," *International Conference on Advanced Geographic Information Systems, Applications, and Services*, pp. 88–92, 2010.

[13] W. Jing-zhong and L. Hui-dan, "Research on the web gis technology based on mapxtreme," in *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 123–126.

[14] C. Granell, C. Abargues, L. Diaz, and J. Huerta, "Interlinking geoprocessing services," *International Conference on Advanced Geographic Information Systems, Applications, and Services*, pp. 99–104, 2010.

[15] M. Hasegawa, S. Bhalla, and T. Izumita, "A user oriented query interface for web-based geographic information systems," *Frontier of Computer Science and Technology, Japan-China Joint Workshop on*, vol. 0, pp. 106–113, 2007.

[16] "Building web-based spatial information solutions around open specifications and open source software," vol. 7, no. 4, pp. 447–466, 2003. [Online]. Available: http://doi.wiley.com/10.1111/1467-9671.00158

[17] G. H. Robert Berry, Richard Fry and S. Orford, "Bulding a geo-portal for enhancing collaborative socio-economic research in wales using open-source technonolgy," *Journal of Applied Research in Higher Education*, vol. 2, no. 1, pp. 77–92, January 2010.

[18] JSR-127 JavaServer Faces. [Online]. Available: http://jcp.org/en/jsr/detail?id=127

[19] Open Geospatial Consortium, Inc. (2010) Welcome to the ogc website. [Online]. Available: http://www.opengeospatial.org

[20] J. de la Beaujardiere, Ed., *OpenGIS Web Map Server Implementation Specification (1.3.0)*. Open Geospatial Consortium Inc., 2006.

[21] P. A. Vretanos, Ed., *Web Feature Service Implementation Specification (1.1.0)*. Open Geospatial Consortium Inc., 2005.