PHPMyWMS - an Open Source based, SVG oriented Framework for extended Web Map Services

Behr, F.-J.[*a], LI Hui.[**b], CHEN Hang[†c], Weldeslasie, F.[‡d], CHEN Zhuoer[§e]

[a]University of Applied Sciences Stuttgart, Department of Geomatics, Computer Science and Mathematics, Schellingstraße 24, D-70174 Stuttgart
[b] M-Way Solutions GmbH, Gehenbühlstr. 14a, D-70499 Stuttgart
[c] Spatial Information Research Center of Fujian Province, Fuzhou University , Key Lab of Data Mining and Information Sharing, Ministry of Education, Fuzhou University , Fuzhou 350002, China
[d]CCSF-Bureau of Engineering, MIS (Management Information System) Section, 1680 Mission St., San Francisco, CA 94103
[e]Room 301, No.14, Long 20, Xinzhuang Mingdu Road, 201100, Shanghai, China

Updated version of original paper, published in: Geospatial Information Technology, edited by Huayi Wu, Qing Zhu, Proceedings of SPIE Vol. 6421 (SPIE, Bellingham, WA, 2006) Article 64210V

ABSTRACT

The Web Map Server Implementation Specification (WMS), originally developed and published by the Open Geospatial Consortium, was finally adopted by ISO as an international standard. According to this standard maps are presented either in "picture" formats or "graphic element" formats. While the pictorial (visual) representation uses raster formats, graphic element formats include Scalable Vector Graphics (SVG) or Web Computer Graphics Metafile (WebCGM) format.

In this paper a framework for supporting Web Map Services using SVG will be presented. Geo-data, converted to SVG, is stored in a database management system, retrieved by the WMS server upon user request and transferred – optionally compressed – through the Internet. Finally it is visualized in Web browsers on desktop computers or mobile devices, natively or using plug-ins extending the browser's functionalities. Besides supporting the WMS request GetCapabilities, GetMap and GetFeatureInfo requests, additional formats and extensions have been included in this framework. The development proves that open, XML-based standards in combination with modern programming languages and integrated development environments allow rapid implementation of recommendations and standards in geo-informatics.

Keywords: Web Map Service (WMS), Scalable Vector Graphics (SVG), Open Geospatial Consortium (OGC), Web services, Open Source, AJAX, LBS

1 INTRODUCTION

With large number of spatial data resources available, the Web Map Server Implementation Specification, originally developed and published by the Open Geospatial Consortium (OGC 2002a, OGC 2006a) was finally adopted by ISO as an international standard titled "IS 19128:2005 Geographic information - Web map server interface". It is a basic specification enabling users to transparently access data of interest from one or several map servers.

This standard is indeed a remarkable technical and commercial breakthrough and is implemented by nearly 150 software products (OGC 2006b). Software conforming to this specification "is able to automatically overlay map images obtained from multiple dissimilar map servers, regardless of map scale, projection, earth coordinate system or digital format" (OGC 2004). This success is based on basic principles: using standard web protocols (HTTP) and ordinary Web browsers.

The service itself consists of three different operations with a corresponding set of parameters. As often defined for such services the first operation – the GetCapabilities request – provides general information to the user about available data,

[*] franz-josef.behr@hft-stuttgart.de; phone (+49) 711/8926-2693; fax: (+49) 711/8926-2556
[**] leeglanz@hotmail.com; phone: (+49) 17621934010
[†] unitony1980@hotmail.com
[‡] filmon44@yahoo.com
[§] jewelchen32@hotmail.com

projection, and format, so called service metadata. By the next operation – the GetMap request – the map itself is obtained and can be visualized by the client software, often combined with maps obtained from other servers. These two operations belong to the basic Web Map Service (basic WMS) which can be extended to a queryable WMS by supporting a third operation, the GetFeatureInfo request which provides additional attributive information about geographical features in the map.

While textual output for service metadata, error messages (exception reports), or responses to GetFeatureInfo requests is usually formatted as Extensible Markup Language (XML), maps are presented either in *picture formats* or *graphic element formats*.

The pictorial (visual) representation using raster formats is most often implemented due to two reasons: The specification originally denotes a map as a visual representation of geo-data, not the data itself (OGC 2002a). Secondly, standardized raster formats such as Graphics Interchange Format (GIF), Portable Network Graphics (PNG), or Joint Photographics Expert Group (JPEG) can be displayed by common Web browsers, additional formats such as Tagged Image File Format (TIFF) can be visualized by additional software invoked by the Web browser.

For the representation in graphic element formats the specification mentions Scalable Vector Graphics (SVG, http://www.w3.org/Graphics/SVG/) or Web Computer Graphics Metafile (WebCGM, http://www.w3.org/TR/REC-WebCGM/). These formats constitute a scale-independent description of graphical elements, so that scale and size of the display may be modified while preserving the relative arrangement of the graphic elements, as ISO 19128 describes.

In this paper a framework for supporting Web Map Services using SVG will be described. Geo-data is converted to SVG, stored in a ordinary relational database management system, is retrieved and transferred – optionally compressed – through the Internet, and is finally visualized in Web browsers on desktop computes or mobile devices, natively or using plug-ins extending the browser's functionality.

1.1 Related work

A few approaches for using SVG for open geospatial services are descrinbed in the literature.

A server based SVG generator service written in Java using the Batik toolkit is described by Zaslavsky et al. (2004). The services are partially compliant to OGC's WMS specification. Additionally, the authors describe several geometric manipulations for reducing the size of SVG documents and optimization of the Document Object Model (DOM, http://www.w3.org/DOM/).

Williams (2005a) and Neumann (2006) use PostGIS (http://www.postgis.org/), the spatially enabled version of PostgreSQL, to store geometries as simple features in WKT format. A server-side scripting language (PHP) extracts data from the database and creates SVG geometry fragments on the fly supporting a service similar to WMS's GetMap request.

1.2 Web Map Server Implementation Specification

There exist several versions of WMS implementations on the market. While WMS 1.1.1 is implemented by more than 140 products (OGC 2006b), the recent version, WMS 1.3, is supported only by 15 products,. This demonstrates the use, popularity and stability of the version 1.1.1 specification. Also, the framework described here is currently based on WMS 1.1.1 specification.

As mentioned above an OGC compliant WMS applies to three different requests: GetCapabilities, GetMap and GetFeatureInfo. Each server that follows the Web Map Service specification has to provide the interface to the users to input the standard parameters, and if these parameters are valid the server must serve the data corresponding to the request.

There are two methods to transfer the request from a client to a server: HTTP GET and HTTP POST. If the client is a browser, the parameters are usually sent inside the Uniform Resource Locator (URL) using HTTP GET; if the client is a mobile device the request is sent also in the URL's query string through wireless network. Such encoding can be used directly and handily by standard user agents (i. e. browsers), and hence may be bookmarked, pasted into HTML documents, and so forth (OGC 2002b), a benefit for the dissemination of WMS. This approach is also supported by the PHPMyWMS server described here. However some disadvantages have to be mentioned. The syntax for URIs (Berners-Lee 1998) is restricted to attribute-value pairs being not so well adopted for lists of layers or styling properties occurring in WMS requests (see section 1.4). Additionally the length of an URI is limited. To overcome these restrictions additionally a XML based encoding using HTPP POST is proposed (OGC 2002c).

1.3 GetCapabilities request

Sending a GetCapabilities request is the first step in the communication between client and server. The parameters are shown in Table 1:

Table 1: The parameters are required by the GetCapabilities request

Request Parameter	Mandatory/Optional	Description
VERSION=version	O	Request version
SERVICE=WMS	M	Service type
REQUEST=GetCapabilities	M	Request name
FORMAT=MIME_type	O	Output format of service metadata
UPDATESEQUENCE=string	O	Sequence number or string for cache control

A GetCapabilities request based on HTTP GET looks like this:

```
http://www.gis-
news.de/wms/getmapcap.php?VERSION=1.1.1&SERVICE=WMS&REQUEST=GetCapabilities
```

In this URL, `http` denotes the requested Internet protocol. `www.gis-news.de` is the server name; `/wms/getmapcap.php` shows the path of the document on the server. The `?` indicates the starting of the query string. By `SERVICE=WMS` we specify the service we want to obtain, a prerequisite to offer different services on the same platform. `VERSION=1.1.1` indicates the server should support OGC WMS version 1.1.1 and `REQUEST=GetCapabilities` tells the server that the user asks for the GetCapabilities service.

In response to a GetCapabilities request, the OGC web map server produces an Extensible Markup Language (XML) document containing the web map server's service metadata, describing all the operations it supports, and providing information about the available map layers, their extents, and the spatial reference system they belong to.. The client application has to parse the XML capabilities document to retrieve the necessary information used to request a map; that's why this response normally is not regarded as very user friendly without additional software providing data aggregation and presentation.

1.4 GetMap request

The mandatory and optional parameters of a GetMap request are shown in Table 2:

Table 2: Parameters of the GetMap request.

Request Parameter	Mandatory Optional	Description
VERSION=1.3.0	M	Request version
REQUEST=GetMap	M	Request name
LAYERS=layer_list	M	Comma-separated list of one or more map layers
STYLES=style_list	M	Comma-separated list of one rendering style per requested layer
CRS=namespace:identifier	M	Coordinate reference system
BBOX=minx,miny,maxx,maxy	M	Bounding box corners (lower left, upper right) in CRS units
WIDTH=output_width	M	Width in pixels of the map picture
HEIGHT=output_height	M	Height in pixels of the map picture
FORMAT=output_format	M	Output format of map
TRANSPARENT=TRUE|FALSE	O	Background transparency of map (default=FALSE)
BGCOLOR=color_value	O	red-green-blue color value for the background color (default=0xFFFFFF)
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=application/vnd.ogc.se_xml).
TIME=time	O	Time value of layer desired
ELEVATION=elevation	O	Elevation of layer desired

Additional parameters are used with Web Map Services that support the Styled Layer (OGC 2002a). The format to send GetMap request in URL is like this:

```
http://www.gis-
news.de/wms/getmapcap.php?VERSION=1.1.1&BBOX=189775.33,4816305.37,761662.27,5472414.18&LA
YERS=airports,ctybdpy2&STYLES=,,&REQUEST=GetMap&style=&SRS=EPSG:26715&4340&WIDTH=800&HEIG
HT=600&FORMAT=image/svg+xml&EXCEPTIONS=application/vnd.ogc.se_xml
```

Following the same principle as above, a valid request has to include all the mandatory parameters. The new parameters in the URL shall be explained briefly. `REQUEST=GetCapabilities` indicates the request name. `BBOX=189775.33,4816305.37,761662.27,5472414.18` and `SRS=EPSG:26715` define the bounding box in the given coordinates system. `HEIGHT=800` and `WIDTH=600` define the size of the image, a measure which can be used for raster formats as well as for SVG. `FORMAT=image/svg+xml` defines the MIME type of the image, in this case SVG format. `LAYERS= airports,ctybdpy2,AUSSTATE1` is the list of layers which are requested by the user. Style information is omitted here. All these parameters should be picked up from the GetCapabilities XML Document.

The example demonstrates that request can be restricted to specific layers, size, extent and desired spatial reference system. The latter is based on a code assigned according to the EPSG Geodetic Parameters dataset of the OGP Surveying & Positioning Committee, formerly European Petroleum Survey Group (EPSG, see http://www.epsg.org/).

1.5 GetFeatureInfo request

This request enables the client to query attribute data of features visible in the map. Its parameters are shown in Table 3:

Table 3: Parameters of the GetFeatureInfo request (adopted from OGC 2002a)

Request Parameter	Mandatory Optional	Description
VERSION=1.3.0	M	Request version
REQUEST= GetFeatureInfo	M	Request name
<map_request_copy>	M	Partial copy of the Map request parameters that generated the map for which information is desired.
QUERY_LAYERS=layer_list	M	Comma-separated list of one or more map layers to be queried
INFO_FORMAT=output_format	O	Return format (MIME type) of feature information.
FEATURE_COUNT=number	O	Number of features about which to return information (default=1).
X=pixel_column	M	X coordinate in pixels of feature (measured from upper left corner=0)
X=pixel_column	M	Y coordinate in pixels of feature (measured from upper left corner=0)
EXCEPTIONS=exception_format	O	The format (MIME type) in which exceptions are to be reported by the WMS (default=application/vnd.ogc.se_xml)
Vendor-specific parameters	O	Optional experimental parameters.

The parameters X and Y clearly depict the close relationship to raster formats – these usually provide no knowledge about coordinates except pixel locations. That's why a copy of the Map request parameters is mandatory part of the parameter list. Using SVG as map output format however, real-world coordinates can be used in the map displayed on client-side. By evaluation of JavaScript events like *onclick* these coordinates can be determined and integrated in a extended GetFeatureInfo request. In the same way a unique *id* can be derived through navigating in the SVG Document Object Model (DOM, http://www.w3.org/TR/SVG11/svgdom.html and Williams 2005a) and also used in the extended GetFeatureInfo request for the unique identification of the feature in the database.

2 SCALABLE VECTOR GRAPHICS

2.1 History

SVG is a XML-based language for describing two-dimensional graphics and supports mainly three types of graphic objects: *vector graphic shapes* (e.g. paths like circles, rectangles, straight lines, curves etc.), *raster images* and *text*. Graphical objects can be grouped (similar to layers in GIS), styled, transformed, animated and composed based on predefined objects. SVG documents are text based, which enhances searchability and accessibility of the graphics. Additionally the SVG recommendation includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility (Watt 2001).

SVG 1.1 is a W3C Recommendation since 2003 and forms the core of current SVG developments. SVG 1.2 is the specification currently being developed and exists as W3C Working Draft since April 2005.

Because of industry demand, two mobile profiles were introduced with SVG 1.1: SVG Tiny (SVGT) and SVG Basic (SVGB, see http://www.w3.org/TR/SVGMobile/). These are subsets of the full SVG standard, mainly intended for user agents with limited capabilities. In particular, SVG Tiny was defined for highly restricted mobile devices such as cellular phones, and SVG Basic was defined for higher level mobile devices, such as PDAs. Therefore SVG Tiny has been used for the mobile client developed in this framework (cf. section 4.2).

Leading print hardware companies are currently developing the SVG Print specification, a version of SVG specifically suited to hard-copy output, a possible replacement for traditional printer languages.

2.2 Browser support

Display of SVG documents is supported by native browser implementations, browser plug-ins, stand-alone viewer applications and mobile devices:

? *Browser implementations:* The Mozilla SVG project started 2001 and now supports a useful subset of the SVG 1.1 specification in the Mozilla core (Rowley et al. 2005). Therefore SVG support is provided both by the Mozilla Application Suite and Mozilla Firefox browser. The online mapping tools Google Maps and Windows Live Local use inline SVG in Mozilla Firefox to render the driving direction paths which also indicates the importance of SVG.
Opera supports SVG since version 8. With version 9, Opera increased its support to 1.1 basic and to keyboard navigation in SVG. The precise level of SVG support is well documented on Opera's website (http://www.opera.com/docs/specs/opera9/svg/).

WebKit (http://webkit.opendarwin.org/) is an open source web browser engine which is used within Apple's Safari browser and other browsers (http://wiki.opendarwin.org/index.php/WebKit:Applications_using_WebKit). It provides an experimental SVG implementation (http://webkit.opendarwin.org/projects/svg/status.xml).

Amaya, W3C's web browsing and authoring environment, supports a subset of the Scalable Vector Graphics (SVG) format, namely basic shapes, text, images, and foreignObject (http://www.w3.org/Amaya/).

? *Plug-ins:* The Adobe SVG Viewer, widely known as the ASV viewer, is still the most popular browser plug-in for SVG**. There have been 4 public releases of major version of the Adobe SVG viewer: ASV 3 is available for Windows®, Mac, Red Hat and Solaris operating system. The latest, ASV 6.0 preview release 1, being provided for developer evaluation on Windows platform, implements several experimental features from the the SVG 1.2 specification, including text wrapping to arbitrary shapes.

? *Stand-alone viewer applications:* Batik (http://xmlgraphics.apache.org/batik/) is a Java™ based toolkit allowing Java applications to parse, view, and manipulate SVG. Due to portability of Java programs, Batik can be used on different platforms.
KSVG2 is a Linux plug-in that provides support for a lot of SVG features in Linux's K Desktop Enviroment (KDE) and the Konqueror web browser (http://svg.kde.org/).

? *Mobile viewers:* TinyLine SVG Toolkit is a pure J2ME CLDC 1.0 implementation for visualization of SVG Tiny documents on mobile devices (http://www.tinyline.com/svgt/index.html), available for Windows CE, Embedded Linux, PalmOS, and Symbian. This viewer is used in the mobile viewer of the WMS framework described below. It's free of charge for personal and commercial use, and it can be run on a phone without a built-in SVG implementation.
BitFlash SVGT Player, a commercial product, available for different operating systems including Symbian and WindowsCE claims to be fully compliant with the W3C's SVG Tiny 1.1 and 1.2 specifications. (http://www.bitflash.com/ prod_playerSVGT.html).

** even if Adobe announced that customer support for Adobe SVG Viewer will be discontinued on January 1, 2008 (http://www.adobe.com/svg/eol.html [Nov 11, 2006]),

2.3 Benefits

SVG, more and more supported by modern browsers, has the following advantages:

? *Graphical quality*: SVG provides excellent graphics. Users can locally magnify their view of an image without sacrificing sharpness, detail, or clarity or additional server requests. The graphic is no more limited by fixed pixel sizes or degradation due to compression.

? *Use on mobile devices*: Because of the many advantages of SVGT, more and more mobile phones begin to support this format. For example, more than 130 SVG-enabled phones are currently listed at http://svg.org/special/svg_phones.

? *Accessibility*: Authors of well-structured SVG files can provide title and description elements as a meaningful extension to graphical object for supporting accessibility for people with disabilities (Campin 2003, Bulatov 2004). Screen readers can speak the title when objects are selected. For visually impaired users, a tactile copy of the SVG display can be created by printing to a tactile device which provides haptic effects. Blind users can locate features and listen to the titles.

? *Performance*: SVG is suitable for applications that require continuous map updates and/or high level of interactivity at the client. Map updates can be achieved using Ajax technology, interactivity is supported locally using ECMAScript. Furthermore some authors mention the reduced file size compared to raster files, an important factor for restricted network bandwidth.

? *True geo-data*: Supporting XML format, the server's output is not only a "portrayal of geographic information", but can be used additionally to deliver the geo-data itself, comparable to an operation of the Web Features Service (Kettemann and Zhu 2005).

? *XML technology*: As an XML based grammar SVG enables combination with other technologies like XLink and SMIL and communication between applications on different platforms. In particular this format can be used besides JSON in context of Asynchronous JavaScript and XML (Ajax) which is relevant for nowadays Internet mapping solutions. As well known, Google® map (amongst others) uses such techniques.

3 THE WMS SERVER IMPLEMENTATION

3.1 Database structure

The database – in this framework MySQL was used – consists mainly of two tables as described in table 4. The `featureclass` table stores meta-information about single layers. A layer, in this context, is a collection of features belonging to the same feature class. The attribute `layertype` describes where and in what format the features of this layers can be found. Currently only `localSVG` is supported, e. g. all features are stored in the `featuregeometry` table. Further developments will support getting data from other WMS servers (while acting as a cascading WMS server) or creating SVG on the fly out of commonly used GIS data formats like shape format. The attribute `description` contains textual information about the layer retrieved for the GetCapabilities request. The bounding box for the features is stored in the attributes `xmin`, `ymin`, `xmax`, `ymax` defining the lower left resp. the upper right corner. The spatial reference system is contained in the `srs` attribute usually specified according to the EPSG Geodetic Parameters dataset mentioned in section 1.4. Styling properties like `style="fill: rgb(0,0,256); fill-opacity: 0.8; stroke: rgb(55,0,0); stroke-width:3.0"` are stored in the style attribute.

Table 4: The create table statements show the structure of the PHPMyWMS database.

Featureclass	**featuregeometry**
CREATE TABLE `featureclass` (	CREATE TABLE `featuregeometry` (
`layertype` text NOT NULL,	`layer` varchar(60) NOT NULL default '',
`layer` text NOT NULL,	`recid` int(11) NOT NULL default '0',
`description` text,	`geomtype` varchar(20) NOT NULL default '',
`geomtype` varchar(20) NOT NULL default '',	`xmin` double default NULL ,
`xmin` double default NULL,	`ymin` double default NULL ,
`ymin` double default NULL,	`xmax` double default NULL ,
`xmax` double default NULL,	`ymax` double default NULL ,
`ymax` double default NULL,	`svggeom` text,
`srs` text,	`svgxlink` text,
`style` text	`srs` text,
);	`attributes` text NOT NULL ,

```
                                                     `style` text NOT NULL ,
                                                     PRIMARY KEY ( `recid` ));
```

The `featuregeometry` table partially consists of similar attributes, but is related to *single* features. The type of geometry (region, polyline, …) is stored in `geomtype`, the geometry itself in `svggeom`. `svgxlink` can be used to embed the SVG geometry into a hyperlink according to W3C's XLink recommendation (http://www.w3.org/XML/Linking).

The `attributes` field contains the feature's attribute / value pairs in XML format. Style properties for a feature can be supplied optionally in the `style` field.

3.2 Service meta data

During installation the relevant administrative metadata according to OGC's WMS Capabilities DTD (OGC 2002a, Appedix A.1) have to be specified and are stored as global variables for the PHP application. Additional metadata for the GetCapabilities response are derived from the `featureclass` table. For a GetCapabilities request the server generates a XML metadata document, which includes information about image formats supported, layers, coordinate systems, bounding box values for each layer, contact person, company etc. Additionally the formats supported are described in the `Format` elements in the metadata. The extended capabilities and operations are also defined by instances derived of `_ExtendedCapabilities` or `_ExtendedOperations` elements. After having validated user's request, required data are retrieved from the database. All extension names have been selected with care to avoid conflicting with other names mentioned in the standard.

3.3 Deploying geo-data

The WMS framework uses two methods to input spatial data into the spatial database. The first input method uses CSV formatted files (comma separated values) with information about layer, extents, geometry, reference system and attributes of spatial features. An example:

```
Badeorte,1,Point,6.1,50.7667,6.1,50.7667,<circle  cx="6.1"  cy="-50.7667"  r  =  "0.166667"
/>,,SRS_not_defined,<attributes><Name>Aachen</Name><Long>6.1</Long><Lat>50.7667</Lat></at
tributes>
```

Such CSV files have a very simple but comprehensive structure, so they can generated quite easily by GIS oriented SVG Conversion tools; one of them, Map2SVG*lite* is one of them (http://www.gis-news.de/svg/map2svg.htm).

The second method uses SVG formatted files directly. Such a file contains all relevant information within SVG elements. Because it is in XML format, it can be parsed, and all geographic information such as element ID, layer name, shape type and shape style, can be picked up and stored into database. All the geometries are ordered with corresponding layer names in the same order as in the original file, where geometry elements belong to corresponding group elements.

3.4 Parsing and transforming SVG geo-data

After loading, the geo-data is stored in the database. There SVG data can be retrieved and transferred to the client directly upon request. However in order to provide other rendering formats the SVG geometries have to be retrieved and parsed. For this task the Expat Library (http://expat.sourceforge.net/), written in C, is used. Expat is the underlying XML parser for the Mozilla project, Perl's XML::Parser, and other open-source XML parsers (Copper 1999a). It also performs very well in parsing XML documents (Copper 1999b).

The following code shows how to parse the SVG `path` element. The first two lines build the parser, and then Expat finds `path` elements. For the attributes `ID` and `D`, the parser will set the values to variants `path_D` and `path_ID`.

```
xml_set_element_handler($this->parser,array($this,'tag_open'),array($this,'tag_close'));
xml_set_character_data_handler($this->parser, array($this, 'cdata'));
...
function tag_open($parser, $tag, $attributes) {
   switch ($tag) {
     case 'PATH': {
       $this->attr_tag = $tag;
       if (count($attributes)) {
          foreach ($attributes as $k => $v) {
             switch($k){
                case 'ID':{$this->path_ID=$v; } break;
                case 'D':{$this->path_D=$v; } break;
             }
```

```
            }
         }
      }break; ...
```

Because the SVG recommendation describes several possibilities for encoding paths, additional steps have to be performed like eliminating unnecessary white space or consideration of relative coordinates. Because coordinates can be separated by whitespace and/or a comma, coordinates should be formatted with one standard.

3.5 Server Side Rendering

The server side rendering component supports PNG, GIF, JPEG and WBMP raster image format, but also extends the WMS specification by providing SWF and PDF output.

? *Rendering raster output*
After having parsed the geometrical information PNG, GIF, JPEG and WBMP raster images are generated using the built-in library GD of PHP5. GD itself is written in C, and "wrappers" are available for Perl, PHP and other languages (http://www.boutell.com/gd/). GD creates PNG, JPEG and GIF images, among other formats. GD is commonly used in many web applications to generate charts, graphics, or thumbnails, on the fly.

? *Generating SWF*
For SWF format, an additional built-in library named Ming was used in the *GetMap_SWF class*. Ming is open-source and allows to create Shockwave Format (SWF, "Flash") movies, including almost all of Flash's features, like shapes, gradients, bitmaps etc (http://de.php.net/ming). Despite PHP5 includes Ming as "standard equipment" and ming.dll is included in a standard PHP installation for Windows, sometimes it has to be activated by simply uncommenting the line `extension=ming.dll` in php.ini and restarting the PHP extension.

? *Generating Portable Document Format*
Because PHP5 does not have a built-in module for PDF generating, currently a library named PDFlib is used in the WMS server (http://www.pdflib.com/purchase/license-lite.html). Because the whole framework should rely upon open source technology it is planned to replace it by PDFlib Lite which can be used for free for open source projects (http://www.pdflib.com/purchase/license-lite.html), FPDF (http://php.scripsi.de/browse/package/421.html) or PDF-PHP (http://www.ros.co.nz/pdf/).

For these additional output formats some restrictions exist. For PNG, PDF, SWF, GIF, JPEG and WBMP format, Bezier curves and elliptical arc are not supported. For SVG and SVGT all types of geometry can be used.

3.6 Generating Exception XML Document

During the request processing, exceptions including errors might be thrown. A `SendException` class is used to send back an error XML document to tell the users what the error is and how to operate correctly.

4 THE SVG BASED WMS CLIENT IMPLEMENTATIONS

There are several WMS clients available in this field, but most of them are proprietary and work only for raster images. One goal of this project is to implement WMS clients especially supporting SVG images.

4.1 A browser based WMS client

The WMS client programmed in JavaScript and SVG allows the user to communicate with WMS servers for requesting maps. For the implementation of a dynamic WMS client, as the essential technology *Asynchronous JavaScript and XML* (AJAX) is used to complete the asynchronous progress of data exchanging with servers in the background without the user experiencing any visual interruptions, an approach based on the work of Williams, Neumann and Winter (Williams 2005a, Neumann et al. 2006).

The XMLHttpRequest object, the core technology of AJAX, is an optimal solution to execute the GetCapability request and fetch the responded XML file on client side (Garrett 2005). The most difficult thing of implementing a dynamic WMS client is how to parse the XML file retrieved from the server. By using XML DOM parser, the XML file which has a node-tree structure can be accessed and the needed elements can be found, extracted and optionally modified. All of these elements are used for updating the HTML page by the client side (see left part in Fig. 1), dynamically creating internal data structures for preparing the GetMap request.

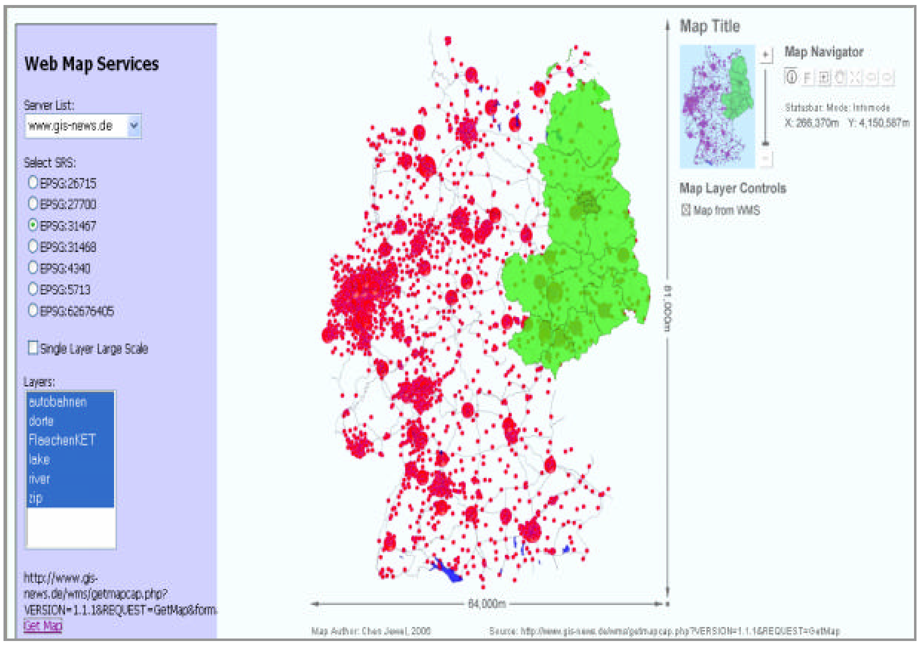


Figure 1: Browser based WMS viewer displaying meta-data (left side) and the result of the GetMap request, including overview map (Chen Zhouer 2006).

Finally the map data received from the server is displayed in a SVG based application where map display is controlled by several navigation tools also created using SVG and JavaScript (see Fig. 1).

4.2 Mobile client

In this project a WMS client for mobile devices was designed and implemented (Li Hui 2006) based on SVGT (see section 2.2) and Java 2 Micro Edition (J2ME), a lean Java platform targeted specifically to clients like mobile phones, PDAs, and embedded devices. J2ME's Mobile Information Device Profile (MIDP) defines an environment for graphical, networked applications and has been used together with the Connected Limited Device Configuration (CLDC) to implement the mobile WMS client.

The client developed has a connection management and provides a user interface derived from the service metadata (cf. Fig. 2). It can not only display SVGT data, but also map images in PNG format and has the capability to parse most of the GetCapabilities document responded from WMS, which follow the OGC WMS 1.1.1 implementation specification.

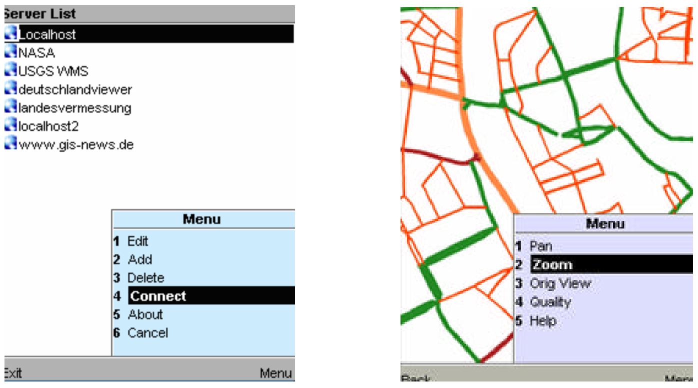


Figure 2: The mobile WMS clients provides a graphical user interface; typical functions like zoom and pan are available for exploration of the map obtained from the WMS server.

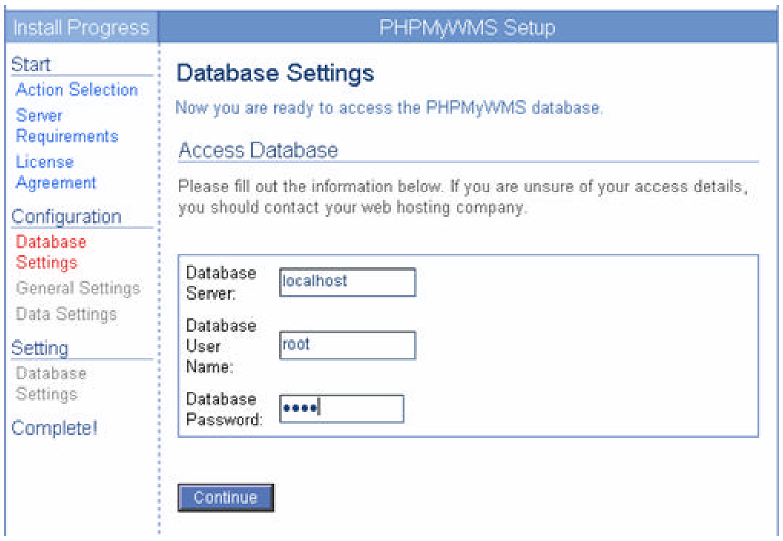
4.3 Server installation

PHPMyWMS is available for download from *http://www.easywms.com*. In addition the project is hosted by *SourceForge.net* and mirrored by *http://www.gis-news.de/wms*. It offers flexible and friendly user interfaces for the ease of installation, configuration and maintenance; users do not need special knowledge to install the server and operate the complex database setting directly. The download provides an automatic installation procedure programmed in PHP

(Figure 3) and checks the availability of all installation requirements (see Table 5). Therefore the existence of the pre-requisites is checked.

Table 5: Installation requirements for PHPMyWMS

Apache HTTP server, version 2.0.x or 1.3.x
MySQL 4.02 or newer as database management system
PHP 5.05
GD library: for raster image, PNG, JPEG, GIF and WBMP generation.
MING library: for SWF image generation
PDF library: for PDF image generating
Expat library: for XML parsing using PHP
phpMyAdmin: to manage the database.



Database Settings

Now you are ready to access the PHPMyWMS database.

Access Database

Please fill out the information below. If you are unsure of your access details, you should contact your web hosting company.

Database Server:	localhost
Database User Name:	root
Database Password:	••••

Figure 3: User guidance during installation process of PHPMyWMS.

5 CONCLUSIONS

An SVG-oriented framework for a Web Map Service environment has been implemented consisting of software to convert geo-data into SVG format and to upload it into a database and a WMS server application supporting the WMS operations GetCapabilities, GetMap, and GetFeatureInfo by providing service metadata, map data and attribute information for selected features. Besides SVG format the server also supports raster formats as well as SWF and PDF format. Map output can be displayed in standard browsers, SVG enabled browsers, and in two developed WMS viewer prototypes, one for ordinary browsers, another for mobile devices based on J2ME. This client, implemented as a MIDlet, can be ported to any smartphone (or even legacy phone) as long as the phone is Java-enabled.

During development and testing process, it has been proven that SVG is well suited for storing, transforming, displaying and transferring geo-data. However there are still some improvements possible:

? Styled Layer Descriptor (SLD) specification or even user-defined symbolization of feature data could be supported by the WMS server. Additionally it could work as a cascading server.

? The ability of parsing the XML files on client-side plays an important role and should be further extended in order to be able to handle service meta-data with different structure.

? In the mobile client, currently the user has to enter locational information manually. User support using Location Based Services (LBS), like GSM-based positioning techniques, or GPS support would be helpful.

? Using the browser based client, a server connection management should be provided. Also, the client can only display the full area of SVG map depending on the extents of the selected layers. The user however should be able to define his area of interest. Additionally the user interface should be extended, and GetFeatureInfo should be supported.

REFERENCES

1. Berners-Lee, T., Fielding, N., and Masinter, L. (1998): Uniform Resource Identifiers (URI): Generic Syntax, IETF RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt . [accessed 15.07.2006]

2. Bulatov, V., Gardner, J. A., (2004): Making Graphics Accessible. *Proceedings SVGopen 2004*, Tokyo, Japan, http://www.svgconference.com/2004/papers/SVGOpen2004MakingGraphicsAccessible/ [accessed 15.07.2006]

3. Campin, B. (2004): SVG Maps for People with Visual Impairment. *Proceedings SVGopen 2003*, Vancouver, Canada, http://www.svgopen.org/2003/papers/svgmappingforpeoplewithvisualimpairments/index.html [accessed 15.07.2006]

4. Copper, C. (1999a): *Using Expat*. http://www.xml.com/pub/a/1999/09/expat/index.html [accessed 15.07.2006]

5. Copper, C. (1999b): Benchmarking XML Parsers. http://www.xml.com/pub/a/Benchmark/article.html?page=1 [accessed 15.07.2006]

6. Garrett, J. J. (2005). *AJAX: A New Approach to Web Applications*. Technical report, Adaptive Path. http://www.adaptivepath.com/publications/essays/archives/000385.php. Accessed on 9.9.2005.

7. Li Hui., Behr, F.-J., Schröder, D. (2006): Design and Implement a Cartographic Client Application For Mobile Devices using SVG Tiny and J2ME. in: Geoinformatics 2006: GNSS and Integrated Geospatial Applications, edited by Deren Li, Linyuan Xia, Proceedings of SPIE Vol. 6418 (SPIE, Bellingham, WA, 2006) Article 641810

8. Kettemann, R., Zhu, W. (2005): Vektordaten der Liegenschaftskarte „on demand“ im eigenen GIS, *Ingenieurblatt Baden-Württemberg*, Heft 2 / 2005

9. Neumann, A., Winter, A. W., Williams, J (2006): *Dynamic Loading of Vector Geodata for SVG Mapping Applications Using Postgis, PHP and getURL()/XMLHttpRequest()*. http://www.carto.net/papers/svg/postgis_-geturl_xmlhttprequest/index.shtml

10. Open Geospatial Consortium Inc (2000). *OpenGIS ® Web Map Server Interface Implementation Specification - Revision 1.0.0. Technical report*,.http://www.opengeospatial.org/docs/00-028.pdf. [accessed on 15.12.2005.]

11. Open Geospatial Consortium Inc. (2002a): *OpenGIS® Web Map Server Implementation Specification - Version: 1.1.1*. http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf. [accessed 15.07.2006]

12. Open Geospatial Consortium Inc. (2002b): *Web Map Service Implementation Specification Part 2: XML for Requests using HTTP POST – version 0.0.3*. http://portal.opengeospatial.org/files/?artifact_id=1118. [accessed 15.07.2006]

13. Open Geospatial Consortium Inc. (2004): *The OpenGIS Web Map Server Cookbook*. http://portal.opengeospatial.org/files/?artifact_id=7769 [accessed 15.07.2006]

14. Open Geospatial Consortium Inc. (2006a): *OpenGIS® Web Map Server Implementation Specification - Version: 1.3.0*. http://portal.opengeospatial.org/files/?artifact_id=14416. [accessed 15.07.2006]

15. Open Geospatial Consortium Inc. (2006b): *Number of Specification or Interface Implementations*. http://www.opengeospatial.org/resources/?page=products&view=stats [accessed 15.07.2006]

16. Watt, A., Lilley C. et al. (2001): *SVG Unleashed*. SAMS, 1117 S., ISBN 0-672-32429-6

17. W3C (2006): *Mobile SVG Profiles: SVG Tiny and SVG Basic*. http://www.w3.org/TR/SVGMobile/ [accessed 20 Feb, 2006]

18. Rowley, T., Morris, J., Watt, J., Fritze, A. (2005): Implementing SVG in a Web Browser: Past, Present, and Future of Mozilla SVG. *Proceedings SVGopen 2005*, Enschede, Netherlands, http://www.svgopen.org/2005/papers/-MozillaSVG/index.html, [accessed 15.07.2006]

19. Williams J. 2005a: Interactive Hiking Map of Yosemite National Park. *Proceedings SVGopen 2005*, Enschede, Netherlands, http://www.svgopen.org/2005/papers/abstract_williams_yosemite_national_park/index.html and http://www.carto.net/williams/yosemite/ [accessed 15.07.2006]

20. Williams, J., Neumann, A. (2005b): *Manipulating SVG Documents Using ECMAScript (Javascript) and DOM*. http://www.carto.net/papers/svg/manipulating_svg_with_dom_ecmascript/index.shtml [accessed 07.02.2005]

21. Zhouer, Chen (2006): *Design and Implement a Cartographic WMS Client Application using SVG and JavaScript*. Unpublished Master’s Thesis, University of Applied Sciences Stuttgart

22. Zaslavsky, I., Memon, A. (2004): Web Services for Generating SVG Tiny Maps on Mobile Phones. *Proceedings SVGopen 2004*, Tokyo, Japan, http://www.svgopen.org/2004/papers/WebServicesForSVGTinyMapsOnMobile/ [accessed 15.07.2006]