OpenLayers: Datenintegration in einem Open-Source-Map-Browser

(mit 7 Bildern, 1 Tabelle und 8 Textboxen)

Von Franz-Josef Behr, Stuttgart

ZUSAMMENFASSUNG: Geoinformation im Internet ist durch hohe Diversifizierung geprägt: viele unterschiedliche Formate und Systeme, viele Angebote, aus dem öffentlichen, kommerziellen und freien Umfeld. Die Interoperabilität der Dienste und der durch sie bereitgestellten Daten gewinnt besondere Bedeutung.

In diesem Beitrag wird mit OpenLayers ein frei verfügbarer Mapping-Client vorgestellt, dessen Schwerpunkt explizit auf der Integration unterschiedlicher Dienste und Formate liegt. Die organisatorischen Rahmenbedingungen dieses Open-Source-Projekts können als exemplarisch für erfolgreiche Projekte dieses Typs gelten. Verschiedene Dienste und Formate werden vorgestellt. Es wird gezeigt, in welcher Weise diese Daten in einem modernen Map Browser geladen und visualisiert werden können. Die Tendenz zeigt das (Zusammen-) Wachsen von freien Daten und freien Programmen.

Keywords: OpenLayers, Open Source, Web Mapping, Visualisierung, OpenStreetMap, WMS, WFS, GML, XML, GeoRSS, KML

ABSTRACT: Geoinformation deployed over the internet is characterized by a high degree of diversification: many different formats and systems, many offers from the public, commercial and free environment. Interoperability of the services and of the data provided by them is gaining increased importance.

In this contribution a freely available Mapping-Client is presented by means of OpenLayers, the emphasis being explicitly placed on the integration of different services and formats. The organisational framework conditions of this Open-Source Project may be considered to be exemplary with regard to other successful projects of this type. Further, various services and formats are introduced in this paper. It is demonstrated in which way these data can be loaded and visualized in a modern Map Browser. The overall tendency is towards an (increasing) growing together of free data and free programs.

RÉSUMÉ: La géoinformation figurant sur l'internet est caractérisée par un haut degré de diversification: beaucoup de formats et systèmes différents, beaucoup d'offres du côté de l'environnement publique, commercial et libre. L'interopérabilité des services et des données fournies par ces derniers gagne une importance particulière.

Le présent article introduit, au moyen d'OpenLayers, un Maping-Client librement disponible dont l'accent est placé explicitement sur l'intégration de services et formats différents. Les conditions cadres organisationnelles de ce projet Source Ouverte peuvent être considérées comme exemplaires pour d'autres projets de ce genre. De plus, de divers services et formats sont présentés. Il est démontré comment ces données peuvent être chargées et visualisées dans un Map Browser moderne. La tendance générale se développe vers une fusion grandissante entre données libres et programmes libres.

1 Einleitung

Die Suche nach dem im August 2007 in der Wüste von Nevada verschollenen Steve Fossett motivierte nach Angaben der Nachrichtenagentur Reuters den britischen Milliardär Richard Branson zu der Idee, Google Earth für die Suche nach dem Vermissten einzusetzen[1]. Die Analyse der Bilder (von Google) der relevanten Tage sollten detailliert nach dem abgestürzten Flugzeug durchsucht werden, das innerhalb eines 600 Quadratmeilen großen Gebietes vermutet wurde.

Diese Nachricht macht in Hinblick auf die Situation der Geoinformatik zwei Dinge deutlich: Zum einen den erstaunlichen Grad der Wahrnehmung von Geoinformationsprodukten seitens der Öffentlichkeit, zum anderen die eingeschränkte (in diesem Fall zu optimistische) Einschätzung der Leistungsfähigkeit bei Datengewinnung und -analyse. Obwohl internetbasierte Darstellungen von Geoinformationen sowohl durch kommerzielle als auch durch Open-Source-Produkte schon seit Jahren im World Wide Web verfügbar waren – ohne große Aufmerksamkeit zu erregen – , kam dieser Durchbruch erst durch die Verfügbarkeit aktueller kommerzieller Online-Mapping-Lösungen großer IT-Unternehmen:

Dieser Wandel steht in engem Zusammenhang mit dem, was *Tim O'Reilly* (2005) als eine evolutionäre Entwicklung des Internets, kurz als „Web 2.0" bezeichnete: der Übergang von reiner Informationsbereitstellung zur interaktiven, gemeinschaftlichen Schaffung von Inhalten, wie es auch im Bereich der Geodaten geschieht (New York Times 2007). Einige Zahlen zu diesem Wandel:

- Nutzer der Photo-Plattform Flickr haben mehr als 25 Millionen Bilder mit Geotags versehen, so dass diese auf Karten oder mittels einer 3D-Software wie Google Earth angezeigt werden können.
- Für Google Earth wurden Millionen von ergänzende Beschriftungselementen und Bildern von mehr als 850 000 Nutzern hinzugefügt[2].
- In den OpenStreetMaps-Datenbanken sind 225 Millionen Knoten und 18 Millionen Wegsegmente erfasst; von 25 000 registrierten Nutzern arbeiten etwa 10% aktiv an der Erfassung (*Ramm* 2008 sowie http://www.openstreetmap.org/stats/data_stats.html). Für Karlsruhe wurden in OpenStreetMaps mehr als 800 000 GPS-Punkte erfasst, um mehr als 1200 km Straßen und Wege zu digitalisieren (*Topf* 2007).

Für die Visualisierung dieser Daten stehen neben virtuellen Globen eine Reihe von kommerziellen oder freien browserbasierten Applikationen zur Verfügung; Bild 1 zeigt einige davon. OpenLayers, ein derartiges Programm, das seinen Schwerpunkt auf der Integration unterschiedlicher Datenquellen stellt, wird nachfolgend beschrieben.

2 OpenLayers: Geschichte, Eigenschaften, Komponenten

Bei OpenLayers (http://www.openlayers.org) handelt es sich um eine Programmierschnittstelle (application programming interface, API) zur Entwicklung von Web Mapping Applikationen. Das API ist vollständig in objektbasiertem JavaScript implementiert und verwendet Komponenten von prototype.js[3] und Rico[4]. Die Implementierung wurde im Jahr 2005 im

1 http://today.reuters.com/news/articlenews.aspx?type=technologyNews&storyid=2007-09-05T170415Z_01_N05213136_RTRUKOC_0_US-FOSSETT-TECH.xml

2 http://economist.com/science/tq/displaystory.cfm?story_id=9719045 [2007-12-09]

3 http://www.prototypejs.org/

4 http://openrico.org/

Anschluss an die Where 2.0-Konferenz begonnen. Zusätzlich getrieben von den Interessen der Firma MetaCarta Inc. (http://metacarta.com/) wurde das erste Release vor der Where 2.0-Konferenz 2006 (http://conferences.oreillynet.com/where2006/) freigegeben (vgl. Bild 1).

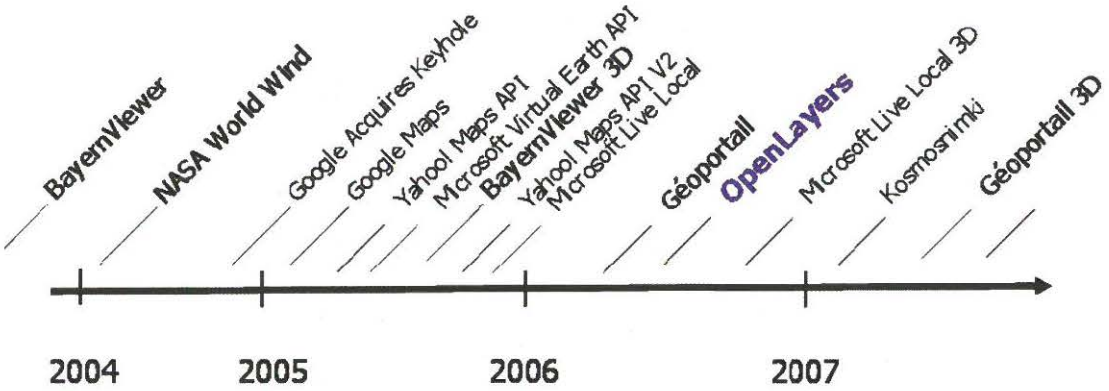


Bild 1 – Überblick über die Entwicklung von 2D- und 3D-Viewern.

Bei geschätzten mehreren Zehntausend Nutzern ist OpenLayers nach Durchlaufen eines Inkubationsprozesses (OSGeo 2006) seit wenigen Monaten als Produkt der Open Source Geospatial Foundation gelistet (http://www.osgeo.org/).

Wie andere Web Mapping-Ansätze beruhen Laden, Visualisieren und Abfragen von Daten auf der AJAX-Technologie (**A**synchronous **J**avaScript **a**nd **X**ML) und einer effizienten Kachelung (tiling) von Daten. OpenLayers steht quelloffen unter einer 3-Klausel-BSD-Lizenz zur Verfügung (http://svn.openlayers.org/trunk/openlayers/license.txt). Der Quellcode kann entsprechend dieser Lizenz unbegrenzt gelesen, genutzt, modifiziert und verteilt werden. Darüber hinaus kann er mit proprietärer Software kombiniert und unter anderer Lizenz weiter verteilt werden (*Brügge* et al. 2004), auch in Form von kommerziellen Produkten. Der ursprüngliche Copyright-Vermerk darf jedoch im Quellcode nicht entfernt werden.

2.1 Aufbau einer OpenLayers-Anwendung

Der prinzipielle Quelltext-Aufbau eines javascript-basierten Map-Browsers ist unabhängig vom zugrundeliegenden API ähnlich und wird anhand Bild 2 für eine einfache OpenLayers-Anwendung erklärt.

Eingebettet im `body`-Abschnitt einer HTML-Seite ist Zeile 23 die Karte als `div`-Element mit dem eindeutigen Schlüssel `map`. Dieses XML-Element dient als Platzhalter zur Aufnahme der Kartendarstellung. Position, Größe und Aussehen des Kartenrahmens werden im `style`-Abschnitt (Zeilen 7–12) festgelegt und über den eindeutigen Namen `map` zugeordnet.

Im `head`-Abschnitt dieser Seite (Zeilen 3–6) wird der JavaScript-Sourcecode des OpenLayers-API inkludiert (Zeile 4). Im gezeigten Beispiel wird die auf dem Server openlayers.org gehostete Version verwendet; der Sourcecode kann jedoch genauso vom lokalen System oder einem anderen Server bezogen werden.

```
1.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2.  <html>
3.    <head>
4.    <script src="http://openlayers.org/api/2/OpenLayers.js"></script>
5.      <title>Simple Map</title>
6.    </head>
7.    <style type="text/css">
8.        #map {
9.            width: 100%;height: 100%;
10.           border: 1px solid black; background-color: white;
11.       }
12.   </style>
13.   <script type="text/javascript">
14.         function init() {
15.       var map = new OpenLayers.Map('map');
16.       var wms = new OpenLayers.Layer.WMS( "OpenLayers WMS",
17.           "http://labs.metacarta.com/wms/vmap0", {layers: 'basic'} );
18.       map.addLayer(wms);
19.       map.zoomToMaxExtent();
20.         }
21.   </script>
22.   <body onload='init()'>
23.     <div id="map"></div>
24.   </body>
25. </html>
```

Bild 2 – Prinzipieller Sourcecode einer OpenLayers-Anwendung.

Das API wird im `script`-Abschnitt (Zeilen 13–21) verwendet. Eingebettet in die Funktion `init()` wird in Zeile 15 ein Kartenobjekt angelegt. Im Konstruktoraufruf wird dabei über den Namen `map` der Bezug zwischen Kartenobjekt und `div`-Element hergestellt. In den Zeilen 16 und 17 wird ein WMS-Layerobjekt als Basislayer geschaffen, der seine Daten vom Server `labs.metacarta.com` bezieht und in Zeile 18 dem Kartenobjekt zur Ansicht hinzugefügt wird. In Zeile 19 wird auf die Gesamtausdehnung der Daten gezoomt.

Angestoßen wird dieser Prozess durch ein JavaScript-Ereignis in Zeile 22: Ist die Seite geladen (`onload`) wird die oben erwähnte Funktion `init()` aufgerufen.

Die somit erzeugte Kartendarstellung kann sehr einfach mit weiteren Elementen wie Layerkontrolle, Übersichtskarte, Zoom- und Pan-Werkzeugen, Permalink, Koordinatenanzeige, Digitalisierfunktionen usw. erweitert werden.

2.2 *Architektur der Mapping-Anwendung*

Wird OpenLayers als Anzeigeclient verwendet, bietet sich als vollständige Internet-Mapping-Lösung eine mehrschichtige Architektur unter Nutzung von REST-Prinzipien[5] an (Bild 3).

5 Representational state transfer (*Fielding* 2000)

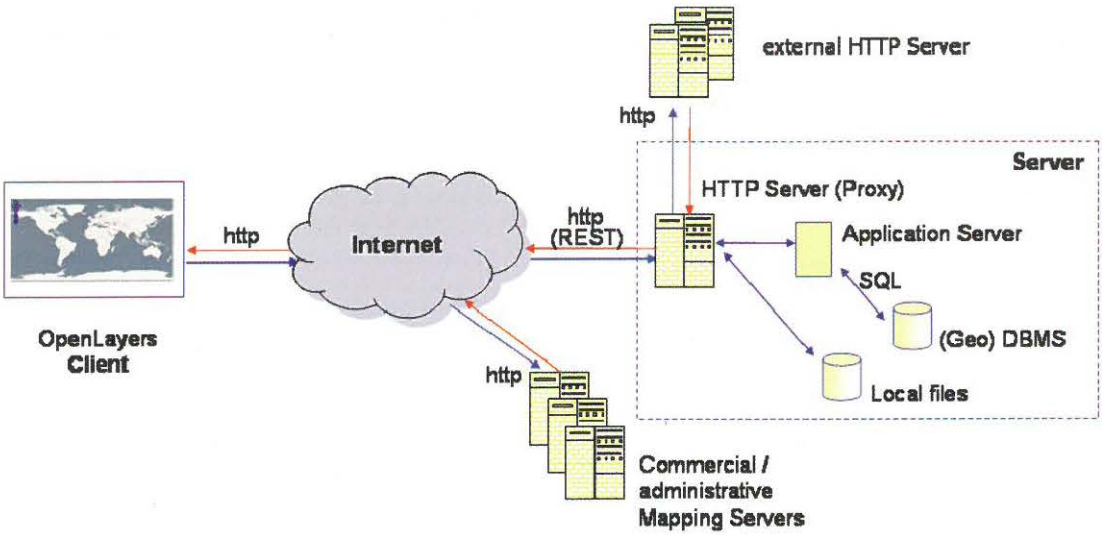


Bild 3 – Beispiel für eine OpenLayers-Client-Server-Architektur.

Der OpenLayers-Client wird von einen HTTP-Server (in der Zeichnung in der Mitte rechts) bereitgestellt. Der Client kommuniziert über das Internet mit seinem Server und bei Bedarf mit weiteren Servern; dabei kommt das Hypertext Transfer Protocol (HTTP) zum Einsatz.

Serverseitig können sowohl lokale Dateien (local files) mit Daten im KML-, GML-, CSV- oder WKT-Format als auch Informationen aus Geodatenbanken über einen Applikationsserver angesprochen werden. Weitere Datenquellen anderer, externer Server können dadurch verfügbar gemacht werden, dass der HTTP-Server als Proxy-Server fungiert.

Anders verhält es sich mit Servern, die rechts unten dargestellt sind; dabei handelt es sich um Server anerkannter, zumeist kommerzieller Mapping-Systeme wie Google Maps, Yahoo!, Windows Live oder NASA WorldWind, deren Daten direkt über die OpenLayers-API integriert werden.

3 Software-Engineering-Aspekte

3.1 Akteure und Struktur der Open-Source-Softwareentwicklung

Wie erfolgt nun der Entwicklungsprozess im Open-Source-Umfeld? Im Rahmen der Open-Source-Software-Entwicklung lassen sich allgemein verschiedene Akteure identifizieren (*Riehle* 2007):

- Benutzer (*user*): Anwender, welche die Software einsetzen;
- Beitragende (*contributers*): Entwickler, die aktiv Neuentwicklungen, Fehlerkorrekturen (patches) und Feedback beitragen;
- *Committers*: Entwickler mit Berechtigung, neuen Programmcode und Korrekturen in das Subversion Quelltext-Repository[6] einzuspielen.

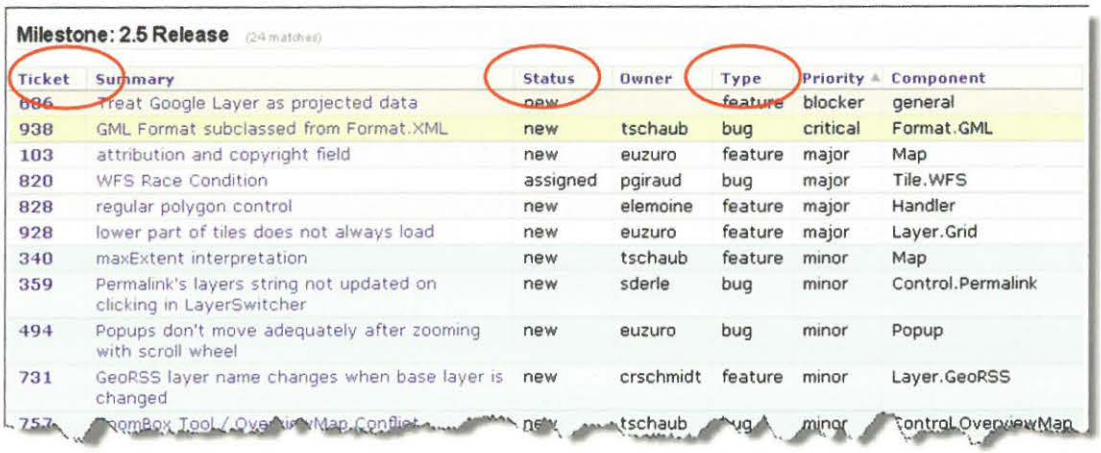
Einzelne Softwareentwickler spielen hierbei häufig mehrere Rollen: Ein Beitragender wird in der Regel auch ein Nutzer sein, ein Committer ist auch als Beitragender aktiv.

---

6 Subversion ist Open-Source-System zur Versionsverwaltung von Dateien und Verzeichnissen, http://subversion.tigris.org/

Bei dem hier betrachteten Produkt ist die Entwicklung durch den Business Case (OpenLayers 2007) sowie publizierte, festgelegte Richtlinien und Verfahrensregeln auf eine breite, anwenderorientierte und freie Lösung hin angelegt. Dies wird in mehrerlei Hinsicht deutlich:

Anwender oder Entwickler können Fehlermeldungen (bugs) oder Wünsche für neue Produkteigenschaften in das Ticketsystem des Projektes eintragen (siehe Bild 4)[7]. Korrekturen können den Tickets beigefügt werden. Der Stand der Bearbeitung ist jederzeit im Web einsehbar, Details werden zusätzlich in ihrem zeitlichen Zusammenhang dokumentiert.



Milestone: 2.5 Release (24 matches)

Ticket	Summary	Status	Owner	Type	Priority	Component
686	Treat Google Layer as projected data	new		feature	blocker	general
938	GML Format subclassed from Format.XML	new	tschaub	bug	critical	Format.GML
103	attribution and copyright field	new	euzuro	feature	major	Map
820	WFS Race Condition	assigned	pgiraud	bug	major	Tile.WFS
828	regular polygon control	new	elemoine	feature	major	Handler
928	lower part of tiles does not always load	new	euzuro	feature	major	Layer.Grid
340	maxExtent interpretation	new	tschaub	feature	minor	Map
359	Permalink's layers string not updated on clicking in LayerSwitcher	new	sderle	bug	minor	Control.Permalink
494	Popups don't move adequately after zooming with scroll wheel	new	euzuro	bug	minor	Popup
731	GeoRSS layer name changes when base layer is changed	new	crschmidt	feature	minor	Layer.GeoRSS
757	ZoomBox Tool / OverviewMap Conflict	new	tschaub	bug	minor	Control.OverviewMap

Bild 4 – Durch ein Ticketsystem können Anforderungen und Weiterentwicklungen angeregt, kontrolliert und freigegeben werden.

Mitglieder der Mailingliste können Vorschläge für weitere Produktentwicklungen einbringen, über die das *Project Steering Committee* abstimmt, das die Projektentwicklung überwacht und steuert. Seine sieben Mitglieder (allesamt Committer, Stand Dezember 2007[8]) stammen aus Nordamerika bzw. Australien und verfügen zum Teil über einen ausgeprägten Erfahrungshintergrund in Open-Source-Projekten im Geoinformatikumfeld (wie z. B. MapServer, ka-Map, MapGuide, MapBuilder). Enge Beziehungen bestehen zur Open Source Geospatial Foundation (http://osgeo.org).

Der berufliche Hintergrund von Contributers und Committers zeigt, dass die Entwicklung (wie auch in anderen Open-Source-Projekten) nicht von „Hobby-Entwicklern" getragen wird, sondern beruflich motiviert ist (*Brügge* et al. 2004): Beiträge werden auch bei OpenLayers von Unternehmen erbracht, die direkt von dem entstandenen Produkt profitieren.

Bezüglich eingereichten Vorschlägen gelten klare Abstimmungsregeln (*Schütze* 2007:61):

- +1 volle Unterstützung des Antrags; Bereitschaft zur aktiven Mitarbeit;
- +0 Unterstützung des Antrags; jedoch keine Bereitschaft zur Mitarbeit;
- 0 keine Meinung;
- –0 geringe Nichtübereinstimmung mit dem Antrag; jedoch keine Veto;
- –1 Veto; Blockierung des Antrags.

7 Erläuterungen und ein entsprechendes Regelwerk finden sich unter http://trac.openlayers.org/wiki/FilingTickets [2007-12-09].

8 http://trac.openlayers.org/wiki/SteeringCommitteeMembers [2007-12-09]

Ein Antrag wird angenommen, wenn Summe der Voten mindestens +2 beträgt und kein Mitglied ein Veto einlegt. Eine Abstimmung ist ebenfalls vorgesehen, wenn die Kompatibilität zu früheren Softwareversionen gefährdet ist, größere Mengen neuen Codes integriert und neue Versionen publiziert werden, oder wenn es gilt, allgemeine Verfahrensfragen zu klären.

Committee Chair ist derzeit ein Mitarbeiter des Unternehmens MetaCarta. Es liegt in seiner Verantwortung, den Diskussionsprozess zu fördern und gegebenenfalls Konflikte zu schlichten. Er überwacht die Zusammensetzung des Komitees, dessen Zusammensetzung und Leitung ebenfalls durch festgelegte demokratische Prozesse bestimmt werden.

An dieser Projektstruktur wird deutlich, dass erfolgreiche Softwareprojekte im Open-Source-Bereich über koordinierende Institutionen verfügen, die zwar Partizipation ermöglichen, den Entwicklern aber auch deutlich machen, was in welcher Form zu tun ist, um ein stabiles, hochwertiges Produkt zu erstellen.

3.2 Qualitätssicherung im Softwareentwicklungsprozess

Für effizientes und konsistentes Prüfen der entwickelten Software wird das Test.AnotherWay-Framework eingesetzt[9]. Es bietet die Möglichkeit, HTML- und JavaScript-Quellcode zu testen und die Ergebnisse anzuzeigen. Im Rahmen der Produktentwicklung ist so das automatisierte Ausführen einer großen Zahl von Tests über ein zentrales Webinterface möglich[10]. Mausbewegungen und Klicks können aufgezeichnet und später für Testläufe abgespielt werden.

4 Integration von Fremddaten

In den Projektzielen wird klar Datenintegration als Zielrichtung festgelegt (OpenLayers 2006): „OpenLayers makes it easy to put a dynamic map in any web page, using geographic data loaded from any source." Ergänzend dazu heißt es im Geschäftskonzept (OpenLayers 2007): „OpenLayers is an AJAX toolkit for freely combining geographic data from any source on the network. [...] It allows anyone to rapidly construct applications using layers from different providers of georeferenced data."

In Bezug auf Datenintegration lassen sich Vektor- und Raster-Layer unterscheiden (Tabelle 1). Einzelne dieser Datenarten werden in den nachfolgenden Abschnitten näher beschrieben und hinsichtlich ihrer Integration im Client erläutert.

9 http://straytree.com/TestAnotherWay/doc/index.html

10 http://openlayers.org/dev/tests/run-tests.html [2007-12-10]

Tabelle 1 – Datenarten in OpenLayers.

Vektorlayer (Punkte, Linien, Flächen)	Rasterlayer
OGC WFS	OGC WMS
GeoRSS	Google Maps
GeoJSON	MSN Live Local
GML	Yahoo! Maps
KML	Multimap
CSV	ka-Map
WKT	WorldWind
	ArcGIS Server 9.2[11]
	Map24[12]
Darstellung über SVG[13] bzw. VML	Darstellung auf Kacheln (tiles) basierend im Browser mittels div-Elementen

Vektordaten können nicht nur angezeigt werden, sondern es ist mit OpenLayers auch möglich, Daten in diesen Formaten zu erzeugen[14]. Mit Ausnahme von GeoJSON, CSV- und WKT-Daten liegt den vektororientierten Datenarten die eXtensible Markup Language (XML) zu Grunde. Unter diesem Oberbegriff verbergen sich eine große Reihe unterschiedlicher Sprachen und Konzepte, die weitgehend vom World Wide Web Consortium (W3C, http://w3.org/) koordiniert und verantwortet werden. Diese Auszeichnungssprache erlaubt es, als Metasprache weitere Sprachen zu definieren, ein Ansatz, der auch in der Geoinformatik in den letzten Jahren rasante Verbreitung gefunden hat. Für die Implementierung der jeweils nötigen Objektklassen bietet die gemeinsame Wurzel – XML – große Vorteile durch Nutzung von Vererbungsmechanismen und gemeinsamer Parserfunktionalität.

OpenLayers erlaubt nicht nur die Visualisierung und Überlagerung dieser Vektordaten, sondern unterstützt durch eine Komponente (widget) die Erfassung von Daten sowie durch spezielle Klassenmethoden die Serialisierung und somit die Speicherung der Vektordaten auf einem Server.

Bei all den positiven Aspekten sei jedoch darauf hingewiesen, dass die Unterstützung für verschiedene Formate noch nicht vollständig realisiert ist. Tim Schaub, einer der Entwickler, stellt Anfang November 2007 in einer Mail fest[15]: „OpenLayers doesn't currently deal correctly or completely with protocol/payload versions for WMS, WFS, KML, or GML."

11 derzeit noch in der Testphase, http://dev.openlayers.org/sandbox/stvn/arcgis-server/examples/arcgis-server.html

12 derzeit noch in der Testphase, siehe Kap. 4.5.

13 Scalable Vector Graphics, ein vom Word Wide Web Consortium entwickelter Standard für die Visualisierung von Vektordaten, siehe Ueberschär (2006) und W3C (2003).

14 Ein Beispiel dazu findet sich unter http://openlayers.org/dev/examples/vector-formats.html [2007-12-10]

15 http://www.nabble.com/openLayer-and-GML3.1---to13644751.html#a13644751 [2007-12-10]

4.1 WMS

Häufig werden Daten für OpenLayers über einen Web Map Service (WMS) bereitgestellt. Die Spezifikation, ursprünglich vom Geospatial Consortium (OGC 2004) entwickelt und dann als internationaler Standard von der ISO übernommen, stellt eine entscheidende Beschreibung dar, wie Daten in vergleichsweise einfacher Art und Weise verfügbar gemacht und überlagert werden können.

Der Erfolg dieses Standards (wie der nachfolgend beschriebenen Ansätze) beruht unter Anderem darauf, dass die Kommunikation zwischen den beteiligten Partnern (Clients und Server) über standardisierte Web-Protokolle wie HTTP abgewickelt wird und übliche Web-Browser verwendet werden.

Der Dienst sieht drei Operationen mit einem entsprechenden Satz an Parametern vor. Wie für viele dieser Dienste üblich bietet die erste Operation – die GetCapabilities-Anforderung – allgemeine Informationen über das Dienstangebot selbst, so genannte Service Metadaten, in diesem Fall über verfügbare Daten, Kartenprojektionen, Datenformate und organisatorische Angaben. Diese Metadaten können in OpenLayers durch eine Erweiterung (http://www.ominiverdi.org/) ausgewertet und in die Anwendungsoberfläche integriert werden.

Im nächsten Schritt stellt die GetMap-Operation Karteninformation bereit; das Ergebnis kann durch OpenLayers, auch in Kombination mit weiteren Datenebenen, visualisiert werden. Da Daten in unterschiedlichen Kartenprojektionen angeboten werden können, sind an die Client-Software prinzipiell hohe Anforderungen für das Überlagern von Daten in unterschiedlichen Projektionen zu erfüllen.

Ein Service kann über die GetFeatureInfo-Operation zu einem Queryable WMS erweitert werden. Dieser stellt Sachinformationen zu Geoobjekten (features) bereit.

Das Openlayers-API unterstützt GetMap- und GetFeatureInfo-Anfragen. Für einen GetMap-Request werden Parameter wie URL des Dienstes, Transparenz, gewünschte Layer und Ausgabeformat spezifiziert und der neue Layer dem Kartenobjekt hinzugefügt:

Textbox 1 – Integration von WMS-Daten in OpenLayers.

```
var transparentWms = new OpenLayers.Layer.WMS( "World Map",
      "http://world.freemap.in/cgi-bin/mapserv?",
      {map: '/www/freemap.in/world/map/factbooktrans.map',
       transparent:'true', layers: 'factbook', 'format':'png'} );
map.addLayer(transparentWms);
```

Für eine GetFeatureInfo-Anfrage werden aus einem Mausklick (click event) die Koordinaten der Mausposition gewonnen. Diese werden mit weiteren Parametern an den Server übertragen. Die Notation des Konstruktoraufrufs für den Request ist weitgehend selbsterklärend. Das Resultat wird über eine Callback-Funktion (hier setHTML) ausgewertet und zur Anzeige gebracht[16].

16 Quelle: http://www.openlayers.org/dev/examples/getfeatureinfo.html

Textbox 2 – Auslösen und Auswerten einer GetFeatureInfo-Anfrage.

```
map.events.register('click', map, function (e) {
      var url =  wms.getFullRequestString({
      REQUEST: "GetFeatureInfo",
      EXCEPTIONS: "application/vnd.ogc.se_xml",
      BBOX: wms.map.getExtent().toBBOX(),
      X: e.xy.x,
      Y: e.xy.y,
      INFO_FORMAT: 'text/html',
      QUERY_LAYERS: wms.params.LAYERS,
      WIDTH: wms.map.size.w,
      HEIGHT: wms.map.size.h});
      OpenLayers.loadURL(url, '', this, setHTML);
      OpenLayers.Event.stop(e);
   }
);
function setHTML(response) {
   OpenLayers.Util.getElement('nodeList').innerHTML = response.responseText;
}
```

4.2 Web Feature Service und Geography Markup Language

Während bei den meisten WMS-Server-Implementierungen der Schwerpunkt auf der Erzeugung von Kartenbildern im Rasterformat liegt, geht es beim Web Feature Service (WFS) um die direkte Bereitstellung von Objektinformation (Lage- und Sachinformationen).

Der Dienst sieht folgende Operationen vor (Open Geospatial Consortium 2002);

- GetCapabilities: liefert ein XML-Dokument mit Service Metadaten zurück;
- DescribeFeatureType: liefert Metainformationen zu den Geodaten;
- GetFeature: Geodaten werden abgerufen;
- Transaction: Es können Änderungen in der Datenbasis vorgenommen werden (Löschen, Hinzufügen, Aktualisieren);
- LockFeature: Sperrung der Features, so dass kein anderer Prozess diese ändern kann.

Das API unterstützt mit der entsprechenden Objektklasse sowohl die GetFeature-Operation mit Anzeige der erhaltenen GML-kodierten Daten als auch die WFS-Transaktionen Insert, Update und Delete.

4.3 *GeoRSS*

GeoRSS wurde 2006 vom World Wide Web Consortium in einer Arbeitsgruppe aufgegriffen; der Ortsbezug rückt damit spürbar mehr in den Blickpunkt der allgemeinen Informationstechnologie. Neben Yahoo! Maps, Google Maps, Live Local, WorldKit, MapInfo und FME wird es auch von Content Management Systemen wie Drupal unterstützt.

Das GeoRSS-Objektmodell erlaubt die Beschreibung von punkt-, linien- und flächenhaften Geometrien sowie Rechtecken als Eigenschaft geographischer Objekte (georss.org 2007). In seiner Einfachheit stellt es eine modifizierte Variante des Simple Features Models dar (*Herring* 2006, vgl. Textbox 3); Einflüsse von GML sind ebenfalls spürbar.

Textbox 3 – Georss-Beispiel für punkt-, linien- und flächenförmige Daten.

```
<georss:point>45.256 -71.92</georss:point>
<georss:line>
   45.256 -110.45 46.46 -109.48 43.84 -109.86
</georss:line>
<georss:polygon>
   45.256 -110.45 46.46 -109.48 43.84 -109.86 45.256 -110.45
</georss:polygon>
<georss:box>42.943 -71.032 43.039 -69.856</georss:box>
```

In OpenLayers ist ein GeoRSS Simple Parser implementiert, der auf Atom bzw. RSS basierendes GeoRSS vollständig verarbeiten kann (*Schmidt* 2007). Die Integration ist denkbar einfach:

Textbox 4 – Integration von GeoRSS-Daten in OpenLayers.

```
var newl = new OpenLayers.Layer.GeoRSS( 'GeoRSS', 'georss.xml');
map.addLayer(newl);
```

Für die Präsentation lassen sich gezielt Vorgaben machen; es ist unter Anderem möglich, Bilder ähnlich wie bei panoramio.com in die Karte integrieren zu lassen[17].

4.4 *Keyhole Markup Language*

Die Auszeichnungssprache Keyhole Markup Language (KML), ursprünglich von der Firma Keyhole entwickelt, liegt in Version 2.2 liegt als Betaversion seit Mitte 2007 dem Open Geospatial Consortium (OGC) zur Diskussion vor. KML ist im Rahmen dieses Standardisierungsprozesses als ergänzender oder konkurrierender Standard zu GML anzusehen, der bereits von einer Reihe von Applikationen wie Google Earth, Google Maps, NASA WorldWind, ESRI ArcGIS Explorer, Adobe PhotoShop und AutoCAD unterstützt wird.

17 http://openlayers.org/dev/examples/georss-flickr.html [2008-02-14]

Wesentliches Strukturierungselement in KML sind `placemark`-Elemente:

Textbox 5 – Beispiel für KML-Daten.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Placemark>
  <name>Zürich</name>
  <description>TopTec Lutz, Zürich</description>
  <Point id="khPoint736">
    <coordinates>8.530270063576751,47.4161609293732,0</coordinates>
  </Point>
</Placemark>
</kml>
```

Ein KML-Layer kann sehr einfach in OpenLayers integriert werden; dabei werden alle Geometrien gemäß KML 2.1-Spezifikation unterstützt. Im nachfolgenden Beispiel wird auf dem Server im Unterverzeichnis kml die Ressource `mc-search.kml` angesprochen und als Datenebene der Karte hinzugefügt[18].

Textbox 6 – Integration von KML-Daten in OpenLayers.

```
map.addLayer(
    new OpenLayers.Layer.GML(
            "KML",
            "kml/mc-search.kml",
           {format: OpenLayers.Format.KML}
    )
);
```

4.5 Integration kommerzieller Kartendienste

Wie standardisierte Kartendienste können auch kommerzielle Kartendienste leicht in OpenLayers eingebunden und angesprochen werden. Datenebenen des Mapping-Dienstes von Google werden über die Google-Klasse des OpenLayers-API verfügbar gemacht:

Textbox 7 – Integration von Google-Maps-Daten in OpenLayers.

```
var googleLayer = new OpenLayers.Layer.Google(
    "Google Hybrid",
    {type: G_HYBRID_MAP, 'sphericalMercator': true,
    'maxZoomLevel':18}
);
```

18 Christopher Schmidt, 19.09.2007, MessageID 20070919111302.GB18087@metacarta.com, http://www.nabble.com/XML---GeoRSS---GML---KML-and-combined-layers--28Markers---SVG-29-to12774098.html#a12775403

Neben G_HYBRID_MAP (Bild 5) stehen natürlich auch die Kartentypen G_SATELLITE_MAP, und G_ NORMAL_MAP zur Verfügung.

Zu beachten ist bei der Datenintegration, dass Daten dieser kommerziellen Dienste in der Regel in Mercatorprojektion geliefert werden (*Purvis* 2007); für die Kombination mit Daten anderer Projektionen ist deshalb explizit die Mercator-Projektion zu spezifizieren (Textbox 7).

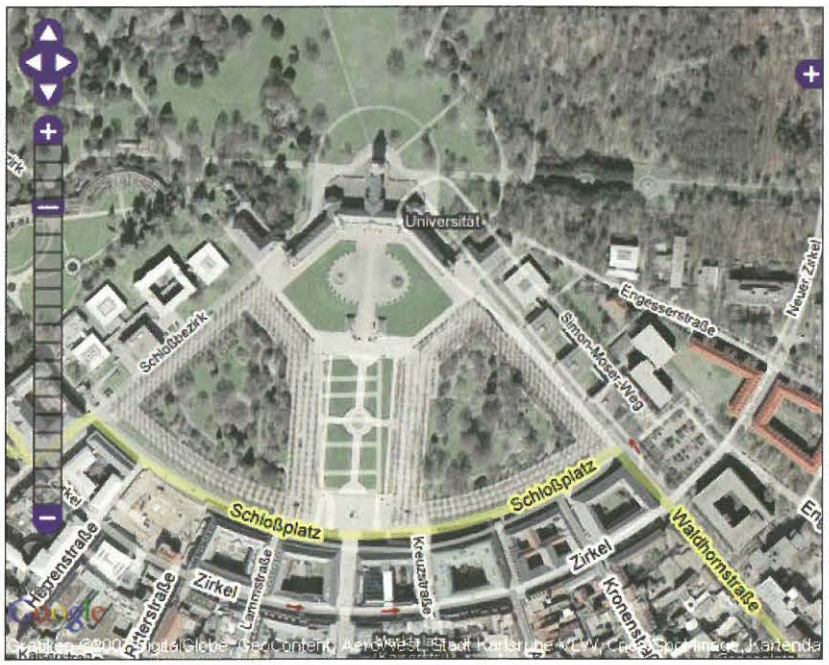


Bild 5 – Datenebene von Google Maps integriert im OpenLayers-Client.

In ähnlicher Weise lassen sich Daten von Microsoft Live Local und Yahoo! Maps integrieren.

Im Oktober 2007 stellte die deutsche componio GmbH ein weiteres Layerobjekt für OpenLayers vor. Diese Erweiterung ermöglicht es, das Kartenmaterial der deutschen Mapsolute GmbH über das API auf eigenen Webseiten zu integrieren. Benötigt wird, wie bei Google Maps, ein site-spezifischer Key, der über http://devnet.map24.com/ erhältlich ist.

Interessanterweise kann die Routing-Funktionalität von Map24 direkt über das API angesprochen werden:

Textbox 8 – Nutzung der Routing-Funktionalität von Map24 in OpenLayers.

```
router = new Map24.RoutingServiceStub();
router.calculateRoute({
   Start: new Map24.Coordinate(darmstadt.lon*60, darmstadt.lat*60),
   Destination: new Map24.Coordinate(flughafenffm.lon*60,
flughafenffm.lat*60),
   CallbackFunction: function(route) {
      routeID = route.RouteID;
   },
   ShowRoute: true
});
```

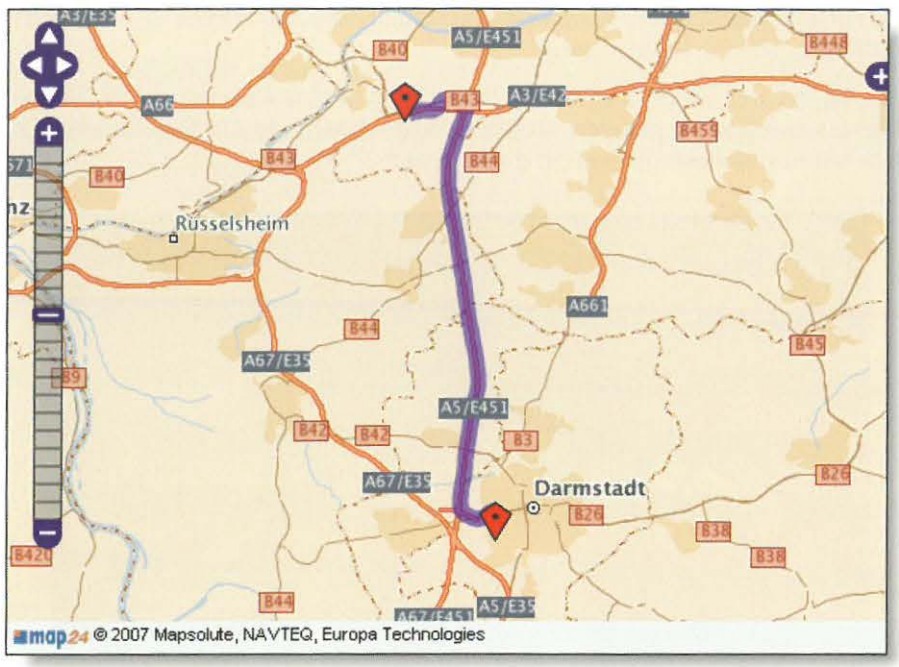


Bild 6 – Map24-Routingergebnis im OpenLayers-Client.

Die Erweiterung wird in naher Zukunft fester Bestandteil der OpenLayers-Bibliothek und ist bereits über die OpenLayers-Seiten in der Sandbox der Entwickler verfügbar (http://www.componio.net/system/galleries/externallinks/componio/OpenLayers-Projektseiten).

4.6 Zusammenspiel mit OpenStreetMaps

Wohin geht die Entwicklung? Bild 7 zeigt die Visualisierung von Daten von OpenStreetMap (vgl. *Ramm & Topf* 2008) und OpenAeriaImages (http://openaerialmap.hypercube.telascience.org/) mittels OpenLayers – ein freies Werkzeug für frei verfügbare Daten.

Deutlich zeigt Bild 7, dass die frei verfügbaren Rasterdaten noch nicht die für den Massenmarkt wünschenswerte Auflösung besitzen; das Straßennetz macht jedoch einen lagerichtigen, flächendeckenden Eindruck.

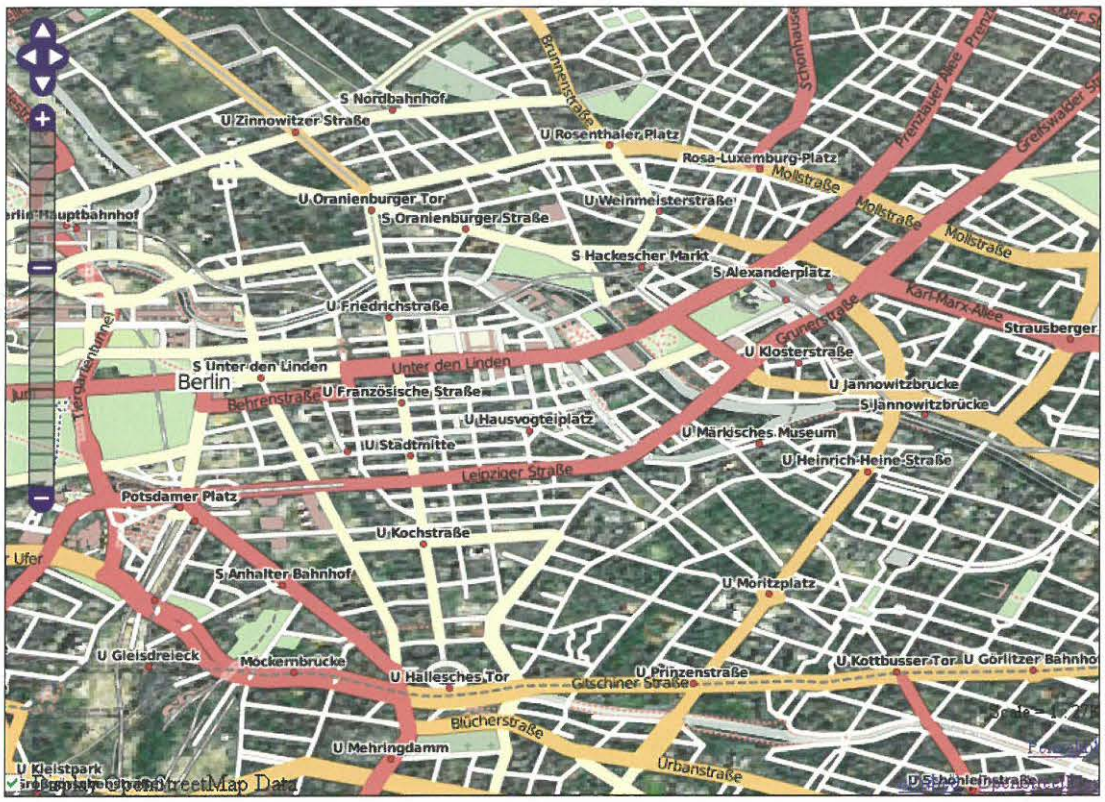


Bild 7 – OpenStreetMaps- und OpenAeriaImages -Daten, gemeinsam mit OpenLayers publiziert (http://openaerialmap.hypercube.telascience.org/map/?zoom=13&lat=52.51499&lon=13.39817&layers=BT).

5 Zusammenfassung

OpenLayers zeigt die Leistungsfähigkeit heutiger javaScript-basierter Mapping-Clients. Der klassenorientierte Ansatz begünstigt eine zügige Entwicklung und die Integration ähnlich strukturierter Daten aus unterschiedlichen Quellen. Die Komplexität für den Anwender wird signifikant reduziert. Freie Software und freie Geodaten konvergieren.

Dennoch sei auf einige Schwachpunkte hingewiesen.

- Die Implementierung einzelner Formate ist teilweise nur unvollständig.
- Wie alle browser-basierten Lösungen leidet auch OpenLayers an technologisch bedingten Einschränkungen seitens der Browser. Nach Schätzungen ist für Vektorgeometrien ab einer Menge von ca. 2500 Koordinatenpaaren bzw. ca. 100–200 Geoobjekten mit Performanceeinbußen zu rechnen.[19]

Insgesamt überzeugt das Produkt durch seine Stabilität, Handhabbarkeit und seine freie Verfügbarkeit.

19 Jedes geometrische Objekt wird als DOM-Objekt im Browser geführt. Diese Einschränkung gilt jedoch nicht für Rasterlayer, in denen beliebig viele Objekte visualisiert sein können.

Literaturverzeichnis

Brügge, Bernd, Harhoff, Dietmar, Picot, Oliver Creighton, Marina Fiedler, Joachim Henkel (2004): Open-Source -Software. Eine ökonomische und technische Analyse. Berlin:Springer, ISBN: 3-540-20366-4

Fielding, Roy Thomas (2000): Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm [2008-02-12]

Herring John R. (ed.) (2006): OpenGIS® Implementation Specification for Geographic information – Simple feature access – Part 2: SQL option. http://portal.opengeospatial.org/files/?artifact_id=18242 [2008-02-12]

New York Times (2007): With Tools on Web, Amateurs Reshape Mapmaking. http://www.nytimes.com/2007/07/27/technology/27maps.html [Tue, 11 Sep 2007 18:49:56 GMT]

Open Geospatial Consortium (2002): Web Feature Service Implementation Specification. https://portal.opengeospatial.org/files/?artifact_id=7176 [10.02.2005]

Open Geospatial Consortium (2004): Web Map Service. http://portal.opengis.org/files/?artifact_id=5316 [10.02.2005]

OpenLayers (2006): The OpenLayers Project Definition. http://trac.openlayers.org/wiki/ProjectDefinition [2007-12-10]

OpenLayers (2007): What is OpenLayers? http://trac.openlayers.org/wiki/BusinessCase [2007-12-10]

O'Reilly, Tim (2005): What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software. http://www.oreilly.de/artikel/web20.html [2008-02-13]; in deutscher Übersetzung: Holz, Patrick, http://www.distinguish.de/?page_id=37 [2008-02-13]

OSGeo (2006): General Principles of Incubation. http://www.osgeo.org/incubator/process/principles.html [2008-02-13

Purvis, Michael, Jeffrey Sambells, Cameron Turner (2007): Google Maps Anwendungen mit PHP und Ajax. MITP

Ramm, Frederik (2008): Message-ID: 20080214202619.GB5149@lochewe.mathy.remote.org. frederik@remote.org

Ramm, Frederik, Topf, Jochen (2008): OpenStreetMap: Die freie Weltkarte nutzen und mitgestalten. Lehmanns Media, 2008, ISBN 978-3-86541-262-1

Riehle, Dirk (2007): Geld verdienen mit Open-Source-Software. Vortrag am 23. Januar 2007, OOP Konferenz, München, available online: http://www.riehle.org/computer-science/industry/2006/os-2006-geld-verdienen.pdf [2007-12-10]

Schmidt, Christopher, 2007: XML / GeoRSS / GML / KML and combined layers. MessageID 20070919111302.GB18087@metacarta.com, http://www.nabble.com/XML---GeoRSS---GML---KML-and-combined-layers--28Markers---SVG-29-to12774098.html#a12775403

Schütze, Emanuel, 2007: Stand der Technik und Potenziale von Smart Map Browsing im Webbrowser – am Beispiel der Freien WebMapping-Anwendung OpenLayers. Diplomarbeit Studiengang Medieninformatik an der Hochschule Bremen, http://swm.wald.intevation.org/html/index_de.html [14 Sep 2007]

Topf, Jochen (2007): Das OpenStreetMap-Projekt. http://www.remote.org/jochen/work/pub/vortrag-fossgis-openstreetmap.pdf [Tue, 11 Sep 2007 18:39:16 GMT]

Ueberschär, Nicole, Winter, Andre M. (2006): Visualisieren von Geodaten mit SVG im Internet. Band 1, Wichmann, ISBN 3879074313

W3C (2003): Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/SVG11/ [2007-12-10]

Mitteilungen des Bundesamtes
für Kartographie und Geodäsie
Band 41

Arbeitsgruppe Automation in Kartographie, Photogrammetrie und GIS

Tagung 2007

Verlag des Bundesamtes für Kartographie und Geodäsie
Frankfurt am Main 2008