The Anatomy of Web Map Services

Jonathan W. Lowe

This column covers the role of emerging technologies in the exchange of spatial information.

Jack Dangermond, president of ESRI (www.esri.com), predicts that Web map services will become "a kind of nervous system for the Earth," enabling GIS users to leverage geographic information to provide a framework for managing our planet's limited resources (see "One Last Thing," *Geospatial Solutions*, July, 2002). Once you understand how these services really work, you may very well agree with him. And widespread agreement about Web map services is exactly what's required to fulfill Dangermond's prophecy. A true planetary nervous system needs millions of data feeds, small and large, all compliant with the same communication standard.

Eager to participate, but unnerved by the details? This article dissects exemplary Web mapping server URLs in an attempt to reveal their inner workings. Scrub up before we operate!

Services and standards

Generally speaking, a Web map service is any network-accessible interface built with Internet technologies that produces maps of georeferenced data. In practice, users type URLs in Web browsers and receive graphic maps in return. It's that simple. The ever-popular MapQuest (www.mapquest.com) site is a familiar example. Input a street address on the Web page's form, click Map-It, and, voila! The MapQuest server returns a new Web page with your map (see Figure 1). This architecture qualifies MapQuest as a Web map service, though not as a standard one.

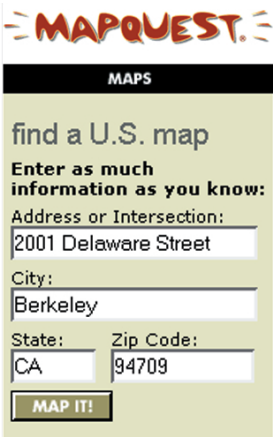


FIGURE 1 The MapQuest Web page formats your address text (entered on the site's introductory form) into a CGI URL such as http://www.mapquest.com/maps/map.adp?country=US&address=2001+Delaware+Street&city=Berkeley&state=CA&zipcode=94709.

Why not? At the click of the Map-It button, code within the Web page generates and sends an appropriate URL for digestion by the MapQuest server. Examining that URL reveals a long, complex string of text that only MapQuest's server understands. In other words, the service is proprietary. If MapQuest instead followed the Web map services standard, anybody could write a MapQuest URL from scratch, independent of any Web page forms or client side code.

MapQuest's developers shouldn't worry though; the standard is more an addition than an overhaul. In theory, the MapQuest server could continue to accept its proprietary URLs, but would also recognize and respond to a handful of standard URL requests for maps. As should become apparent, there are several good reasons for adopting such a standard.

CGI internship

Before explaining the standards that expose a Web map server to public query, a brief review of CGI is helpful. Back at the MapQuest site, the introductory Web form (see Figure 1) generates the following URL for the address "2001 Delaware Street, Berkeley, CA 94709": http://www.mapquest.com/maps/map.adp?country=US&address=2001+Delaware+Street&city=Berkeley&state=CA&zipode=94709&homesubmit.x=38&homesubmit.y=9.

Maybe I was too hasty in saying MapQuest's forms generate URLs that only MapQuest's server understands. Though proprietary, the components of this URL conform to a CGI template that makes deciphering possible. Chopping up our URL, "http://" is the protocol, "www.mapquest.com/" is the domain name, and "maps/map.adp?" is the location and name of a CGI program on MapQuest's Web server that listens for incoming Internet requests. Fol-

Glossary

CGI: Common gateway interface

GUI: Graphical user interface

OGC: Open GIS Consortium

WMS: Web Map Services (OGC Implementation Specification)

XML: Extensible markup language

Net Results columnist Jonathan W. Lowe is the owner of Local Knowledge Consulting (Berkeley, California), where he designs and implements spatial Web sites. Lowe can be contacted at info@giswebsite.com.

lowing the question mark are a series of parameter pairs linked by "=" characters and separated by "&" characters. For instance, "country=US" is the first pair, "address=2001+Delaware+Street" is the second, and so on.

Digesting it all. If Web map services are Earth's nervous system, CGI programs are the digestive tract. In this MapQuest example, the program (map.adp) swallows the list of parameter pairs and uses them as variables when geocoding addresses, generating JPEG maps, and finally regurgitating new Web page responses. To prove this concept, just fabricate a similar MapQuest URL using the same parameter names (such as country, address, state, and so forth) but with different values (for instance, replace "2001+Delaware+Street" with "2010+Virginia+St"). Then feed the homespun URL to a Web browser. For any valid address, MapQuest should still return a Web page with your requested map. No matter what the service, this pattern of "protocol://domain/directory/program?name=value" indicates a CGI exchange similar to our example, and is the basis of the Web map services standard recommended by the OpenGIS Consortium (OGC, www.opengis.org).

Nurse, pass the scalpel

OGC published version 1.1.1 of its WMS Implementation Specification in mid-January 2002, and it continues to advance the standard today. At its throbbing heart, the specification requires that conformant Web map servers respond to two URL-based requests, namely, *GetMap* and *GetCapabilities*.

When encountering a URL containing the parameter pair "REQUEST=GetMap," a standard Web map service will return its default map image (see Figure 2). An imaginary URL structure for this most basic of standard requests is as simple as "http://my.domain.com/directory/program.cgi?REQUEST=GetMap." To satisfy this request, the server would have to predefine the extent of the map view, the map projection, the image format (JPEG, PNG, or GIF), and the list of spatial data layers to include in the reply.

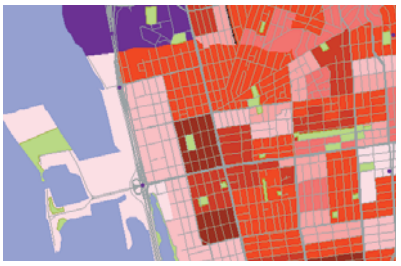
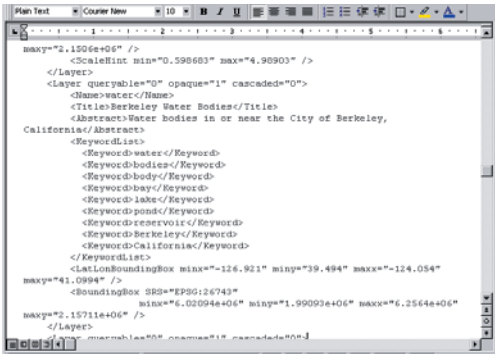


FIGURE 2 A default census and reference map is returned by an imaginary Web map service in response to http://my.domain.com/directory/program.cgi?REQUEST=GetMap.

Suppose a server has several data layers, but only one or two are important to a particular user's need? The

FIGURE 3 A custom imaginary map service response including only the data listed in the LAYERS parameter: http://my.domain.com/directory/program.cgi?REQUEST=GetMap&LAYERS=water+streets+parks

FIGURE 4 A custom imaginary map service response with projection specified: http://my.domain.com/directory/program.cgi?REQUEST=GetMap&LAYERS=water+streets&SRS=epsg:4326



```
maxy="2.1506e+06" />
      <ScaleHint min="0.598683" max="4.98903" />
    </Layer>
    <Layer queryable="0" opaque="1" cascaded="0">
      <Name>water</Name>
      <Title>Berkeley Water Bodies</Title>
      <Abstract>Water bodies in or near the City of Berkeley,
California</Abstract>
      <KeywordList>
        <Keyword>water</Keyword>
        <Keyword>bodies</Keyword>
        <Keyword>body</Keyword>
        <Keyword>bay</Keyword>
        <Keyword>lake</Keyword>
        <Keyword>pond</Keyword>
        <Keyword>reservoir</Keyword>
        <Keyword>Berkeley</Keyword>
        <Keyword>California</Keyword>
      </KeywordList>
      <LatLonBoundingBox minx="-126.921" miny="39.494" maxx="-124.054"
maxy="41.0994" />
      <BoundingBox SRS="EPSG:26743"
                   minx="6.02094e+06" miny="1.99093e+06" maxx="6.2564e+06"
maxy="2.15711e+06" />
    </Layer>
```

FIGURE 5 Text editor view of partial XML output from a GetCapabilities request, showing the water layer's details.

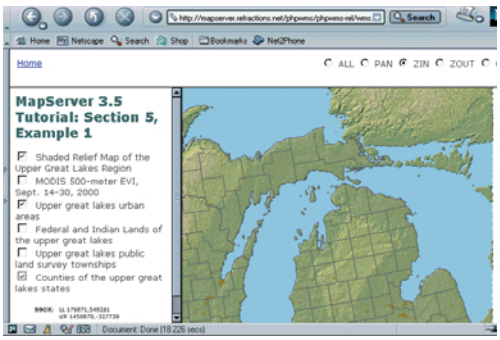


FIGURE 6 Referencing any compliant Web map server from within Refractions Research's WMS Client is similar to using a simple online GIS.

OGC specification defines additional optional parameter pairs which, when added to the basic URL, tell the server what to send, or what image size and map extent to draw, for instance (see Figures 3 and 4). Though textual rather than graphic, this approach sounds strangely similar to the traditional desktop GIS's table of contents GUI. On the GIS desktop, to see a layer, one just activates its checkbox. A Web map server recognizes URL-checkboxes when digesting a parameter pair like "LAYERS=streets+parcels+hydrants." Adding other parameters that control the drawing style of each layer is part of the ongoing evolution of the specification.

Turning a user-friendly graphic table of contents into a long and complicated URL may seem like a step backward, but it gives access to custom map images to anyone capable of creating a simple HTML document. Web site designers not interested in building interactive online GIS interfaces, but who just need current, static map images, can incorporate the output of Web map services seamlessly into their sites and always be assured of the most current content.

What's up, doc?

But wait a minute! When sending a URL to somebody else's Web map server, how is the URL's author supposed to know what layers exist in the first place, their names, or their maximum extents? Picking specific layers depends on knowing the list of layer choices. OGC anticipated the need for server metadata by requiring that conformant Web map servers also be able to answer the GetCapabilities request. Sending any conforming server a URL such as "http://my.domain.com/directory/program.cgi?REQUEST=GetCapabilities" should make the server spill its guts, fully disclosing its dataset names, extents, supported map projections, and other such metadata.

The response to a GetCapabilities request is an XML document (see Figure 5). If looking for hurricane positions during storm season, for example, an automatic program could poll each of a collection of Web services, drawing only those that listed "hurricanes" in their XML layer listings.

What's in a name? While automatic access to service data listings is wonderful, the content itself remains problematically arbitrary. First of all, a unified worldwide nervous system that pivots on text names requires a standard language. At the moment, that seems to be English. But even assuming the use of the English language, what's to stop one service from calling its street centerlines the "roads" layer or the "pavement" layer rather than the "streets" layer? Efforts to standardize thematic terminology by industries such as surveying, planning, hydrology, and others may eventually reach the Web mapping services community as well.

In the meantime, XML is providing structured data, and the OGC Web Map Services specification standardizes that structure. In combination, this data stream and its metadata enable plug-and-play Internet GIS interfaces that load all the layers from any compliant Web map service into the familiar GUI of a traditional GIS table of contents and map view. These envelopes of usability surrounding Web map servers' data streams are known as WMS clients. A simple WMS client example by Refractions Research appears at http://mapserver.refractions.net/phpwms/phpwms-rel/. For a complex example, visit the Cubewerx site at www.cubewerx.com/demo/cubeview/cubeview.cgi. To test either one as a client to a real Web map server, try entering the URL http://arbutus.gis.umn.edu/cgi-bin/wmsserv.cgi? into the text box at the bottom of either site's opening Web page (see Figure 6). Simple or complex, WMS clients usually offer basic pan and zoom functionality, but most efforts are recent and experimentation still follows the evolving OGC specification. (A third part of the OGC specification recommends but does not require a *GetFeatureInfo* request that

returns information about a particular map feature rather than the service's overall capabilities. If and when this third request becomes a requirement, then "identify" tools will likely become common functions of WMS Clients).

Transparent brain tissue

Was our operation a success? If this anatomical exploration of Web map services still hasn't convinced you to join the effort, consider one last capability — the one that ties our planetary nervous system together as a collective organism. When requesting maps of equal bounding box dimensions, map projection, and output size from two or more Web map servers, the result can be a single composite map! This capability is surprising considering that most Web map services return images rather than vectors, and images (or rasters or bitmaps) are opaque grids of pixels. How can several opaque images be unified into a composite map? Today's PNG, JPEG, and GIF formats support transparent backgrounds, so the servers can overlay many transparent images and retain partial views of the bottom layers, just as with vector data stacks. (Cascading the output of different Web map servers happens not as a URL parameter, but within the construction of a server, so some layers listed by one server may in fact originate in another.)

As an aside, Web map servers are not required to broadcast maps as images; vector streaming or XML data output is also considered in the specification; though, it's less developed than the image option thus far. The specification also describes the way to add on-the-fly features to map server output by putting the coordinates of a point, line, or polygon into the URL itself. The textually described feature then appears on top of the map image like a remora riding along on a shark.

Combining Web servers. Taking the composite mapping idea to a community level, combining the output of many separate distributed Web map servers by capturing and stacking their images is also possible. Today's online spatial data holdings are often, to quote the OGC spec, "vertically-integrated" sites, each with referential basemap data and overlying domain-specific layers. In contrast, the Web map server concept encourages data stewards to focus on their own particular data collections, and cascade the collections of others into their composite maps as needed. Because all data in this vision are supplied the moment they are requested, the most current views result. Of course, the more services in the list to be combined, the longer it takes to generate a composite image. Consequently, the issue of detailed local data generalization when requested at global scales is one of the hot discussion topics among the Web map services community.

The ability to combine disparate but standard Web map server output has sparked various distributed data projects. A noteworthy Canadian project clearly understands the nervous system concept. As documented by Robin Quenet, Rick Morrison, Brian Low, and Jim Wood, for example,

> The Canadian Forest Service (www.nrcan.gc.ca/cfs-scf/index_e.html) is building a system to address key science and policy issues in support of Canada's commitments in such areas as climate change, biodiversity, criteria and indicators of sustainable development, trade and its national and international reporting obligations. The system is designed to provide and integrate timely, accurate and spatially explicit forest resources information by identifying, analyzing and portraying the available data.

The project relies on a combination of one central heavy-duty Cubewerx server and many remote University of Minnesota MapServer servers.

Sharing data from Web map servers also raises discussion about credibility. Knowing about a map server's layer list is one thing. Knowing when the layers' data were last updated is another. The OGC specification devotes a detailed appendix to the subject of dates and times as metadata elements. Commercial image service providers like GlobeXplorer (www.globexplorer.com) and Pixxures (www.pixxures.com) have already recognized the importance of providing temporal metadata to their customers. For instance, when mosaicking multiple images into a single composite image, Pixxures retains the flight time for each pixel in the patchwork so that hovering the mouse over any area can return metadata about when the plane or satellite captured that individual pixel.

Equally important to temporal data is the reliability of the provider. If our industry continues its trend toward ever more open data sharing, the importance of branding and establishing a good reputation as a reliable data provider can only increase in parallel.

A case of nerves

So ends our dissection of the framework that Dangermond envisions as a planetary spatial nervous system. Is he on the right track? As one of many standards, Web mapping services could become just another waste product of our industry's gastrointestinal system if not adopted by the majority of spatial professionals. My opinion, for what it's worth, is that Web mapping services, whether a source of profit or a gift to the community, are well worth the implementation investment in Earth's long-term health.

Consider conducting a Web map services operation of your own. Two free products, the Apache (www.apache.org) Web server in conjunction with University of Minnesota's MapServer, can become a WMS server or client (or both at once!). Online documentation appears at the MapServer site (http://mapserver.gis.umn.edu).