CHAPTER SIX

Publishing maps on the Web

Barend Köbben

6.1 INTRODUCTION

The World Wide Web is a relatively new means of publishing maps and although nowadays millions of people use it every day, not many of those users know how it actually works. Many of them, quite sensibly, do not really care. But for the readers of this book some basic knowledge about the technical background of the Web can be helpful in understanding the different mapping possibilities available.

In this chapter, a short overview is given of how the World Wide Web sprouted from the Internet and of the file formats and communication protocols that are its building blocks. It will become clear how the basic set-up allows the publication only of static maps with very limited interactivity and how the possibilities can be broadened by adding server-side applications or client-side plugins and applications such as Java applets or scripts. ESRI's ArcView Internet Map Server will be used as an example of how all these techniques can be combined to offer mapping and GIS functionality on the Web.

6.2 INTERNET AND THE WORLD WIDE WEB

To understand the World Wide Web, one needs to understand Internet, because the Web is a subset of the broader concept of the Internet. The Internet has its roots in the very modest first start of computer networking called ARPAnet. This was a network created for the Advanced Research Projects Agency and funded by the U.S. Department of Defense. Its purpose was twofold: first to create the possibility of scientists residing in different locations to work together and share information and secondly, to spread vital information over a range of computers in different locations in order to diminish its vulnerability to nuclear attacks. The network was built as a distributed, decentralised system with each node of equal importance and the computers in the network communicated with each other using a so-called "communication protocol" (called NCP – Network Control Protocol), which defined how the systems should request, send and retrieve information from each other. By the end of 1969, this grandmother of the Internet connected four host computers at the University of California Los Angeles, the University of California Santa Barbara, the University of Utah and Stanford Research Institute (Salus, 1995). As a nice gesture to the history of web cartography, this configuration was mapped at the time, as can be seen in Figure 6.1.

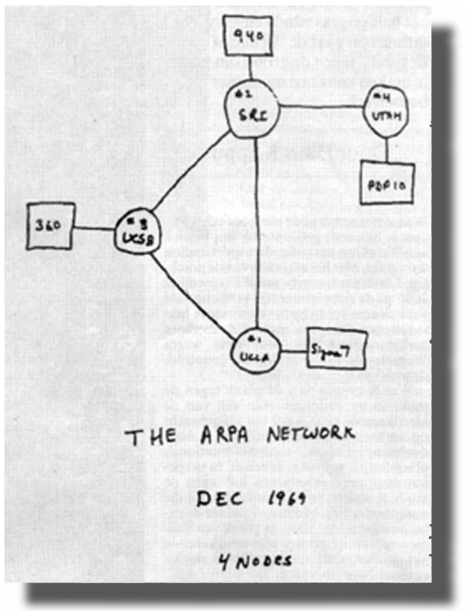

Figure 6.1 Sketch of the initial ARPAnet (from Salus, 1995).

Gradually, the ARPAnet grew to encompass many North American computers and in 1983 it switched to a new communication protocol called TCP/IP (for Transmission Control Protocol/Internet Protocol). Around the same time, USENET, the Unix users' network that carried most of the e-mail and news of that time and the European side (EUnet) were also converted to use TCP/IP. All kinds of other communication protocols, such as TelNet (for terminal services), FTP (for file transfers) and SMPT (for e-mail communication) were redirected to run "on top of" TCP/IP. This is considered to be the birth of the Internet as we know it (*URL 6.1*).

The reason TCP/IP is so important today is that it allows stand-alone networks to be connected to the Internet or linked together to create private intranets. Devices called routers or IP routers physically connect the networks that comprise an intranet. A router is a computer that transfers packets of data from one network to another. On a TCP/IP intranet, information travels in discrete units called IP packets or IP datagrams. TCP/IP software makes each computer attached to the network a sibling to all the others; in essence, it hides the routers and underlying network architectures and makes everything seem like one big network. Connections to the Internet are identified by 32-bit IP addresses, organised as dotted decimal numbers, for example 192.92.92.70. Given a remote computer's IP address, a computer on the Internet can send data to the remote computer as if the two were part of the same physical network (for more details, see *URL 6.2*). To make the addressing system more versatile, a system of Domain Name Servers (DNS) has been established. These map the numerical IP addresses to a hierarchical system of logical names. The IP address 192.92.92.70 thus is mapped to the IP name “kartoweb.itc.nl”, indicating a computer called “kartoweb” within the network of the ITC, which in turn is part of the Dutch (nl) section of the Internet. The DNS system is in fact a hierarchical distributed database, with each level administering its own subsection. As shown in Figure 6.2, higher level administrators can use *DNS aliasing* to re-route requests to lower level domains, so that the computer with the IP number 192.92.92.70 answers to the name “www.kartografie.nl” (a website maintained by the Netherlands Cartographic Society) as well as to “kartoweb.itc.nl”.

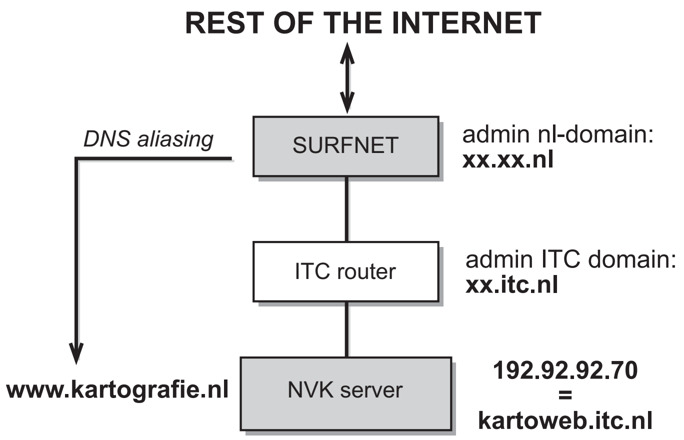

Figure 6.2 The Domain Name Serving system.

Somewhat confusingly, besides Internet there is this thing called the World Wide Web (WWW or simply “the Web”). It was originally started by Tim Berners-Lee at CERN (the European Laboratory for Particle Physics) to enable the dissemination through the Internet of information among the various research facilities of that institute. In order for this to be possible the communication protocols of the Internet were not sufficient, what was also needed were standardised ways to store the information. For this purpose, the HyperText Mark-up Language (HTML – see Appendix A.2) was devised. A prototype of HTML and a new communication protocol HTTP (HyperText Transfer Protocol), specifically geared towards finding and retrieving HTML and other documents from various servers (to be run “on top of” TCP/IP), was agreed upon in 1994 (*URL 6.3*).

In practice, the Web is now a vast collection of interconnected documents, spanning the world, which merges the techniques of networked information and hypertext to make a powerful global information system. It makes information accessible over the network as part of a seamless hypertext information space (Peterson, 1996).

6.3 BASIC PUBLISHING ON THE WEB

One of the strong points of the Web is the fact that it is virtually platform-independent. The data can be served by computers running AppleOS, Unix, Linux, BeOS, VMS or Windows, to name but a few. For retrieving and viewing the information, there are so-called *web browsers* available at little or no cost for every type of computer in existence, including even game computers, organisers and mobile phones. The underlying TCP/IP *communication protocol* makes sure all these computers can interact. But in order for all the browsers to be able to interpret and show the content correctly, there also are a limited number of standardised *data formats* that have to be used to store the information.

6.3.1 Basic web data formats

The main Web format is the HTML document mentioned earlier. In any hypertext document, if you want more information about a particular subject mentioned, you can usually just "click" on it to read further details. In fact, documents can be and often are linked to other documents by completely different authors, stored on computers in totally different locations. The HTML format takes this a step further because it supports hyper*media*, a superset of hypertext − it is any medium with pointers to other media (*URL 6.4*). This means that browsers not only display and link textual information, but also might be used to view images and animations and to listen to sounds.

HTML is a non-proprietary format and can be created and processed by a wide range of tools, from simple plain text editors – you type it in from scratch – to sophisticated WYSIWYG authoring tools. Converting information to HTML format is possible nowadays in most "office" environments (this term indicates much-used software for common office tasks, such as word processing, presentation and spreadsheets).

HTML is essentially plain text interspersed with so-called "tags" to structure text into headings, paragraphs, lists, hypertext links, etc. These tags are used by the browser that is used to interpret the HTML files to structure and layout the text. For example the HTML code:

```
<P>Hello <B>World</B><HR>How are <I>you</I>?<P>
<A HREF= "otherfile.html">Click here</A> to go.
```
would be shown as:

Hello **World**

How are *you?*
Click here to go.

Clicking on the words "Click here" would result in another file (stored using the name "otherfile.html") being loaded by the browser, interpreted and displayed on the screen.

To show graphics, so-called inline images can be embedded on-screen by referring to their files in the HTML code. For this, two raster formats are generally used: GIF (Graphics Interchange Format) and JPEG (Joint Photographics Expert Group). Both are raster formats, the former better suited for graphics having areas of solid colours, such as graphs and maps, and the latter more appropriate for half-tone graphics, such as photographs (for a more detailed description, see Appendix A.3).

6.3.2 Basic web publishing set-up

To publish maps and information on the Web, one needs only very few resources. The actual requirements depend largely on the flexibility and sophistication needed, but the minimum set-up includes the following elements:

1. A so-called "backbone" connection to the Internet. This is a fixed and high capacity connection. The data transfer capacity of the more usual modem connections is insufficient. Such backbones can be found at major companies, educational and government institutes, etc. When no such server is at hand, one can hire facilities from commercial Internet Service Providers.
2. One or more computers using the Internet connection mentioned in 1.
3. An application that understands the HTTP protocol and listens for requests coming in for documents stored on the servers connected to it. It basically waits for HTTP "GET" commands and will respond by transferring the files asked for to the requesting network node by using the HTTP "PUT"-command. This application is called a *web server*.
4. Basic text editing tools or a WYSIWYG editor to author the HTML files.
5. Basic graphic tools to make the graphics. These graphics should be specially designed for web use (see Chapter 7).

The last point is especially important, because of what currently is the major disadvantage of using the Web: the limited transfer capacity of the Internet. The hardware infrastructure is at present not sufficient for the large number of users, resulting in poor data transfer rates, especially when using a modem connection as the majority of users do. This is quickly becoming a genuine problem, such that the abbreviation WWW is sometimes said to mean "World Wide Wait". In order to offer users at least bearable response times, the files transferred over the Internet need to be as small as possible, making the use of large format, high-resolution graphic files undesirable.

The combination of the server and browser software enables the retrieval of static information only, that is the user can review text and graphics as and how they are stored on the server. If for example one wants to publish thematic maps derived from certain statistical data, one would have to prepare these maps in advance and include them as embedded images from which the user can choose. Thus, elements such as the layout, the visualisation techniques and classification would be fixed and the user cannot change these. This basic set-up is shown in Figure 6.3.

What happens in this basic set-up can be described as a simple series of requests and responses. It starts by the user providing the browser with the Internet address of the information he or she wants to view. This address will be in the form of a URL, a Universal Resource Locator, for example "http://kartoweb.itc.nl/ index.html". This URL could be typed in by the user or could be the result of a hyperlink from some other document. It will result in the browser sending a request (using the http "GET" command) to the server located at the IP address kartoweb.itc.nl for the file called "index.html". The server at this location will respond to this request by sending the file (using the http "PUT"command) to the IP address of the browser sending the request. This browser will read the HTML file and will interpret the tags and commands inside that document to show the user the information in the file. When, as in the example given in Figure 6.3, the code for an embedded image (the <IMG>–tag) is encountered, an additional request is sent for the image file stated in the "SRC=" parameter of the tag. The image is received, displayed on the screen and the browser resumes interpreting the HTML code.

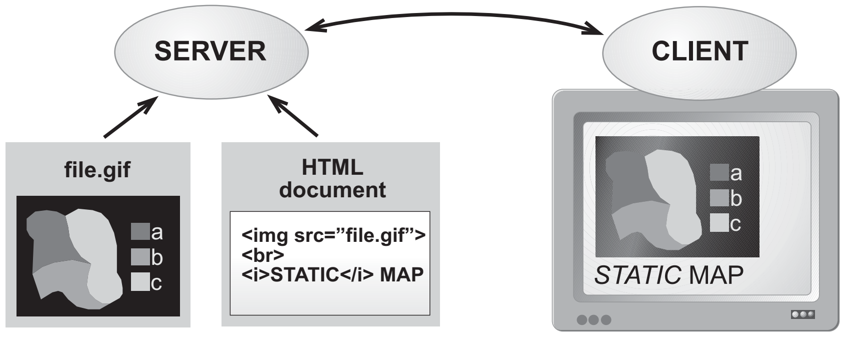

Figure 6.3 Basic web publishing.

Any image included in a HTML-file can be made to have certain "sensitive areas", defined by the coordinates of their bounding polygons. By clicking the mouse within such parts of the image the browser will link with another Internet location. Thus, a single image can provide different hyperlinks, each assigned to defined regions of the image. These images with "invisible buttons" are called *clickable images* or *image maps* (even though they are not always maps, but maybe logos or any other graphic). For an annotated example with more explanation, see the interactive version of Figure 6.4 on this book s website.

There are two kinds of clickable maps: "server-side" clickable maps and "client-side" clickable maps. With server-side maps, the coordinates of the bounding polygons are not stored in the presentation itself, but in a separate map file on the web server (hence the name). The disadvantages of this are that it creates extra data traffic between server and client and that one cannot test image maps locally, because a "live" connection to the supplier's server is needed. That is why recently the client-side clickable maps have evolved. Client-side image maps store the hyperlink information in the HTML document in which the image is embedded, not in a separate map file. When the user clicks a hotspot in the image, the associated hyperlink location is determined by the browser software and the user is transferred directly to that location. This makes client-side image maps faster than server-side image maps. Client-side image maps are nowadays supported by all major browsers.

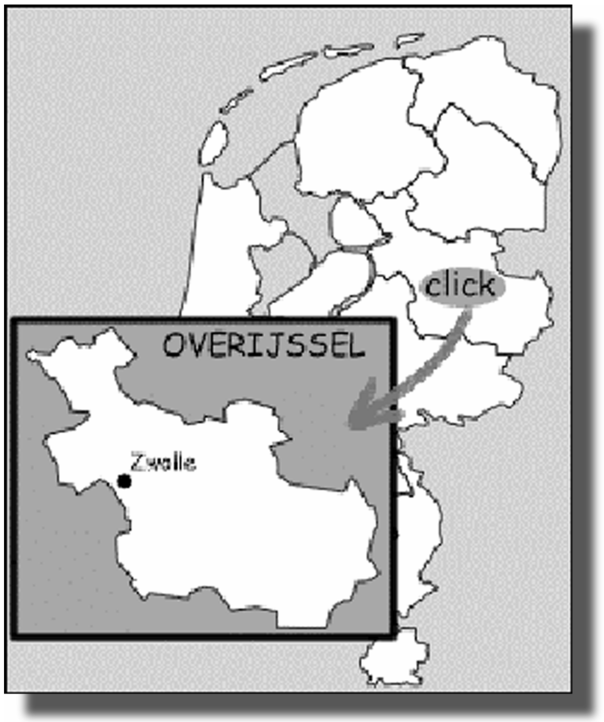

Figure 6.4 Screen dump of a clickable map (annotated interactive version on this book's website).

6.4 EXTENDING WEB MAP FUNCTIONALITY

The standard set-up described in the previous paragraph makes use of the configuration complying with the official standards of the World Wide Web, as maintained by its governing body, the World Wide Web Consortium (see *URL 6.5*). This set-up is quite limited, for example only the two raster formats mentioned (GIF and JPEG) could be used to show graphics. For mapping applications this severely limits the possibilities. Only static maps are possible, with no interactivity other than the clickable maps mentioned earlier. To overcome the limitations, many solutions are available. In the following paragraphs, additions to the basic system are described which could be realised at the client-side (i.e. the web browser) or at the server-side, or that could be a combination of both. One has to realise however, that many of the solutions discussed below require extensions to the standardised client-server functionality, thus undermining the platform-independence and making the information less generally useable.

6.4.1 Client-side functionality

Plugins

In the basic set-up described above, only three file formats are used: HTML for the text and layout and GIF or JPEG for the images. Using exactly the same client-server communication, other formats could theoretically be used, among them vector file formats. However, this requires that the web browser can interpret and display these file formats. For this purpose, most browser applications offer a mechanism that allows third-party programs to work together with the browser as a so-called *plugin.* This plugin technology is the simplest and probably the most used extension of the web functionality. Many thousands of plugins are available for almost as many file formats. The example in Figure 6.5 shows the use of the Acrobat plugin that Adobe devised for displaying their Portable Document Format (PDF), a popular document description format based on PostScript.

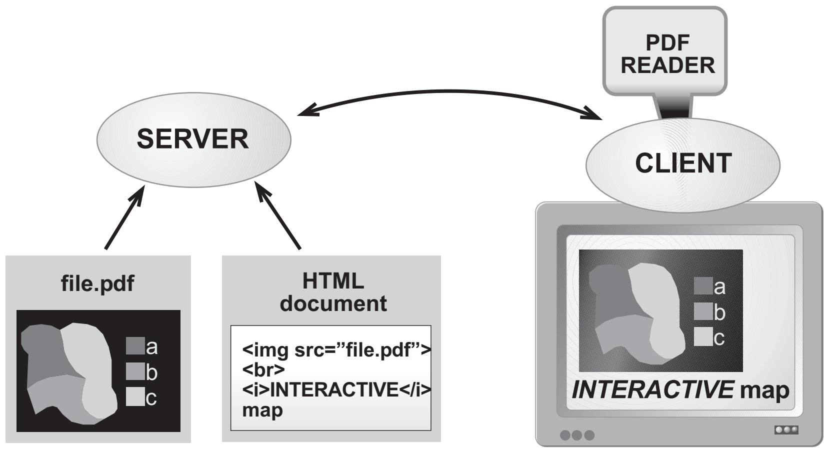

Figure 6.5 Plugin set-up (example of Adobe Acrobat PDF–reader).

As can be seen when comparing Figures 6.5 and 6.3, the only changes to the basic set-up are the addition of the plugin to the browser and the different extension of the filename in the "SRC=" parameter of the <IMG> tag. This extension allows the browser to recognise the file as not being a "native" format and to transfer the interpretation of the file to the appropriate plugin. The Acrobat PDF reader takes over control and allows the extended functionality of the PDF file: zooming and panning, printing at high resolution, selecting and querying text and graphics, etc.

The advantage of this system is the ready availability of plugins for very many file formats, thus making possible the distribution of almost every graphics file through the Web. Mostly these plugins are distributed free of charge by the proprietors of the file format.

The main disadvantage lies in downloading and installing these plugins and their all too frequent updates. While surfing the Web, the necessity to install a new plugin occurs almost as often as the need for a coffee. Furthermore, the plugins are programmed for specific operating systems and sometimes even for specific browsers only.

Java and JavaScript

The use of Java on the Web became widespread in a very short period. The main reasons for the rapid acceptance is the fact that it does not have the platform-dependence of the plugin solution and that the major players in the market are now including Java interpreters (so-called *virtual machines*) into their browser programs. Thus Web functionality can easily be extended by supplying small stand-alone Java programs or including Java scripts in the actual HTML code.

Java is an object-oriented programming language designed for fast execution and type safety (*URL 6.6*). Type safety means, for instance, that you cannot accidentally access memory (other than that reserved for the application).

Table 6.1: JavaScript compared to Java (from *URL 6.6*).

JavaScript	Java
Interpreted (not compiled) by client.	Compiled bytecodes downloaded from server, executed on client.
Object-based. No distinction between types of objects. Inheritance is through the prototype mechanism and properties and methods can be added to any object dynamically.	Object-oriented. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.
Code integrated with, and embedded in, HTML.	Applets distinct from HTML (accessed from HTML pages).
Variable data types not declared (loose typing).	Variable data types must be declared (strong typing).
Dynamic binding. Object references checked at runtime.	Static binding. Object references must exist at compile-time.
Cannot directly write to hard disk.	Can write to hard disk.

JavaScript and Java are similar in some ways but fundamentally different in others (see Table 6.1). Java's object-oriented model means that programs consist exclusively of classes and their methods. This makes Java programming more complex than JavaScript authoring. The JavaScript language resembles Java but does not have Java's static typing and strong type checking.

JavaScript supports most Java functionality plus extras such as predefined objects only relevant to running JavaScript in a browser. It is embedded directly in HTML pages and is interpreted by the browser completely at runtime. When the browser (or client) requests such a page, the server sends the full content of the document, including HTML and JavaScript statements, over the network to the client. The client reads the page from top to bottom, displaying the results of the HTML and executing JavaScript statements as it goes. Client-side JavaScript statements embedded in a HTML page can respond to user events such as mouse-clicks, form input, and page navigation. For example, you can write a JavaScript function to verify that users enter valid information into a form requesting a telephone number or zip code. Without any network transmission, the HTML page with embedded JavaScript can check the data entered and alert the user with a dialogue box if the input is invalid.

Using Java applets, the same code would be not incorporated in the HTML page, but would be pre-compiled into a so-called "Java class" and stored on the server. The HTML document would only hold a reference to the file holding the Java class, which would be loaded and run by the Java Virtual Machine, incorporated in or linked to the browser software. The main advantage is that execution is safer and much faster compared to interpretation of JavaScript, because the code is already compiled. This set-up is shown in Figure 6.6.

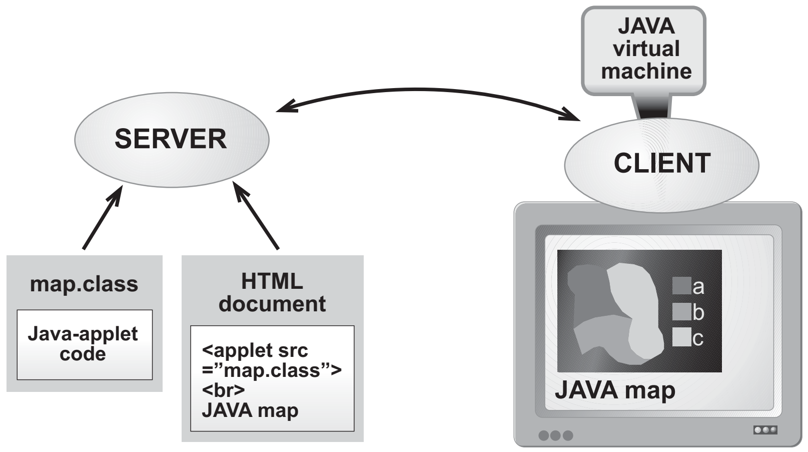

Figure 6.6 Set-up using Java applet.

The functionality offered by Java applets or scripts can vary from offering very simple button interactivity (using "roll-over" scripts to have a button change appearance when the mouse moves over it) to a sophisticated interactive mapping environment. An excellent example of the latter is the Descartes system, designed to help with exploratory analysis of spatially referenced data, for example statistical data of the municipalities of Overijssel province, as in *URL 6.7*.

Using Java to extend Web functionality has become very popular because it combines the advantages of both client-side and server-side solutions. The application is stored and maintained on the server, thus bypassing the need for the user to download updates. The platform-independent nature of Java ensures the application will run on virtually any computer and because the actual processing takes place at the client-side, the server load is kept low.

6.4.2 Server-side functionality

Besides the addition of functionality on the client side as described in Section 6.4.1, there is also the possibility for utility programs at the server side to be linked to the server software. This linkage is in most cases achieved using the Common Gateway Interface (CGI). This protocol defines the communication between the server software and the application, but also the way that the web browser can transfer information to the CGI application through the web server. The latter is achieved by adding commands and parameters to the URL, using a question mark to separate the CGI part from the rest and using ampersands to delimit parameters (eg. http://kartoweb.itc.nl?doThis&parameter1&parameter2).

CGI-compliant applications can be used for any number of tasks, for instance to provide database access or to customise information depending on user preferences. Thus, the user could get a map depicting the latest figures from a database, which can come from a remote server, visualised with the colours and classification the user has requested (see the example in Figure 5.3). The set-up of such a system can be seen in Figure 6.7.

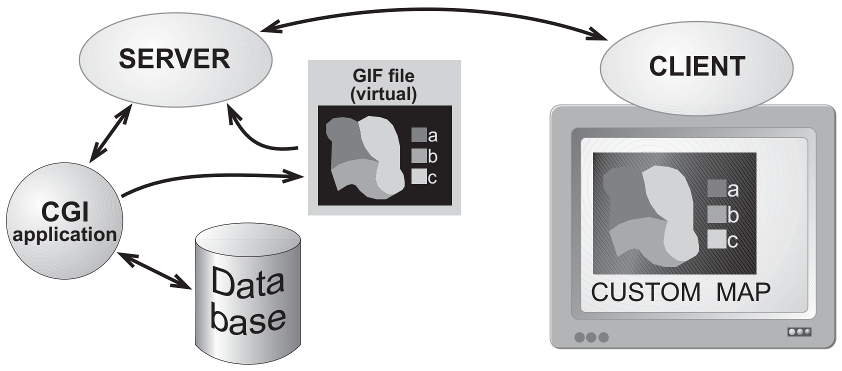

Figure 6.7 Server-side CGI set-up.

The nature of CGI-compliant software is varied: from simple self-programmed applications to support a page counter on a website"s home page to proprietary extensions to existing applications. Such an extension is the popular "Web Companion" plugin to Claris' FilemakerPro database software, which lets one publish an existing database on the Web. The much-acclaimed Oddens' Bookmarks site *(URL 6.8*), that offers an extensive overview of links to cartographic websites, uses this software.

The cartographic functionality offered by such a system can vary from offering very simple static maps to a full-blown electronic atlas system, such as the

National Atlas of Canada Online (*URL 6.9*). This has evolved from using CGI software developed in-house, as in 1993 when it was started there was no "off-the-shelf" software available for interactive mapping on the Internet. The software for the most recent 6th edition was programmed using ESRI technology (Frappier & Williams, 1999).

The advantage of server-side solutions is that the system can be used by any Web browser on any operating system without the need to install plugins, thus maintaining platform-independence. The main problem is the server load. On popular sites, the number of requests (usually called "hits") can run up to hundreds every minute. This puts tremendous demands on the applications running on the server side. The web server software itself has the comparatively easy task of receiving requests and sending files in response, but imagine a database system having to keep up by processing a hundred SQL-queries per minute!

6.4.3 Mixed solutions

Although the previous paragraphs separated client-side from server-side solutions, in practice often a combination of the two is used, thus combining the advantages of both. An example of this is the set-up shown in Figure 6.8, as used by ESRI's ArcView Internet Map Server. ESRI has developed solutions to enable its GIS and mapping applications to publish on the new medium. Almost all major GIS vendors have done this and although their approaches differ in detail, most use the combination mentioned above.

As can be seen in Figure 6.8, the ArcView program takes on the role of a CGI application. For this purpose an ArcView extension is installed, called Internet Map Server (IMS). This enables CGI-compliant commands to be received from the browser, through the web server. The command, say a map query, is processed by ArcView and the result (a map view) is converted to a GIF or JPEG file and sent to the browser. A Java applet called MapCafé is used to implement in the browser an interface similar to the standard ArcView interface. Clicking the zoom button and dragging a rectangle in the map would result in the Java applet building a CGI command to implement the required zoom action. The last item in this set-up is a plugin (called esrimap.dll) to the web server software, which enables the server to find the appropriate ArcView application to handle the request. This application can run on another computer and to decrease server load the plugin can distribute requests among a multitude of computers running the same ArcView application. Because the standard ArcView application is used, existing projects can be made to run on the Web with little effort, as has been done in the example of *URL 6.10*, which shows the countries where ITC Cartography consultancy has taken place in the period 1982 1998. The Java applet can be customised and the IMS can handle calls to all functionality within ArcView, including its built-in scripting language (Avenue). This makes the system very flexible.

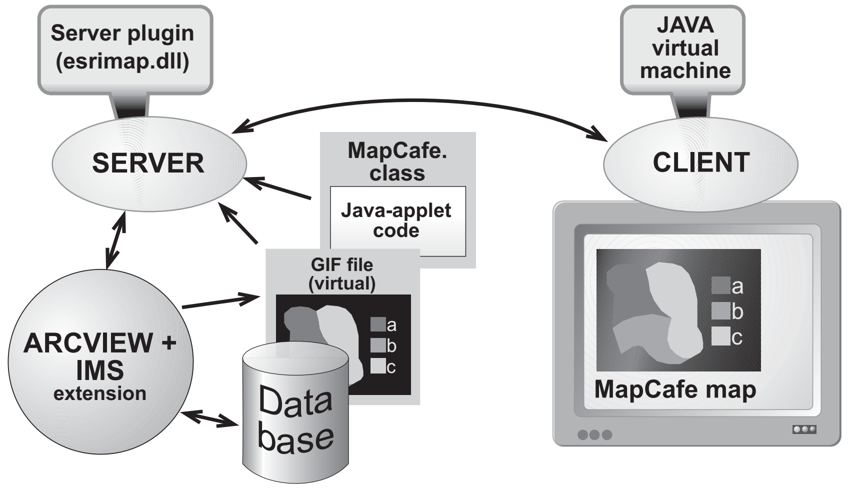

Figure 6.8 ESRI's ArcView Internet Map Server set-up.

In conclusion, it can be said that the Web is a powerful new way of publishing cartographic information. It offers a very simple mechanism to provide many users quickly with simple static maps at low costs. When the mapping environment needs to be more versatile, supplying interactivity or dynamic maps, functionality can be added on the client side or the server side. This could range from the simple addition of plugins to enable additional file formats, to complicated set-ups involving programming of additional Java or CGI applications.

URLs

URL 6.1	A Brief History of the Internet <http://www.isoc.org/internet-history/brief.html>
URL 6.2	PC Magazine's Beginner"s Guide to TCP-IP <http://www.zdnet.com/pcmag/issues/1520/pcmg0030.htm>
URL 6.3	A Short History of Internet Protocols at CERN <http://wwwinfo.cern.ch/pdp/ns/ben/TCPHIST.html>
URL 6.4	WWW Frequently Asked Questions <http://www.boutell.com/faq>
URL 6.5	W3C–The World Wide Web Consortium <http://www.w3.org/>
URL 6.6	Client-Side JavaScript Guide <http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm>
URL 6.7	Explorative maps of Overijssel using the Descartes system <http://allanon.gmd.de/and/java/iris/app/itc/indexm.html>
URL 6.8	Oddens' Bookmarks <http://oddens.geog.uu.nl>
URL 6.9	National Atlas of Canada Online <http://atlas.gc.ca/english/>
URL 6.10	ITC Cartography consultancy 1982-1998 <http://kartoweb.itc.nl/avmaps/consultancy98/consult98.html>

REFERENCES

Frappier, J. and Williams, D., 1999, An overview of the National Atlas of Canada. In: *Proceedings of the 19th International Cartographic Conference ICC99, Ottawa*, edited by Keller, C.P. (Ottawa: Canadian Institute of Geomatics), pp. 261-267. (See also: <http://atlas.gc.ca/english/about_us/ index.html #overview>)

Peterson, M.P., 1996, Cartographic animation on the Internet.. *Proceedings of the joint ICA Commissions Seminar on Teaching Animated Cartography*, Edited by Ormeling, F.J., Köbben, B and Perez Gomez, R (Utrecht: International Cartographic Association), pp. 11-14.

Salus, P.H., 1995, *Casting the Net: From ARPAnet to Internet and beyond*, (Reading, MA: Addison-Wesley Publishing Company).