TECHNICAL BASICS

Web GIS is an exciting new area of application development. You probably have used various types of Web GIS applications: finding pizza restaurants near your home, finding business locations near your hotel, or tracking delivery fleets at work.

This chapter covers the basic concepts and technology behind Web GIS applications. Section 2.1 presents the basic technologies of the Web, including its three principal technologies—HTTP, URL, and HTML—as well as related technologies on the server side, the browser side, and the data transmission between them. Section 2.2 introduces the major components of Web GIS, which include the Web GIS server, the GIS database, and the client. Geobrowsers and the online virtual globe are defined in this section as popular and ready-to-use Web GIS clients. Section 2.3 talks about the thin client and thick client architectures, and then introduces a design pattern that is applicable to most Web GIS applications. Section 2.4 highlights design based on the user experience. While not intended to be a step-by-step guide or a "how to," the contents of this chapter can familiarize you with the basic technologies and principles of Web GIS applications design.

2.1 WEB FUNDAMENTALS

Web application architecture can be complex, because one application can be interconnected with other applications. This type of architecture will be explained later in the book. This section introduces the basic architecture of a Web application.

2.1.1 WEB PRINCIPLES

A basic Web application is essentially a client/server (C/S) architecture, in which the client is specifically called a Web client, and the server is called a Web server. Basic Web applications usually have a three-tier architecture consisting of the data tier, logical tier, and presentation tier (figure 2.1). The Web client is typically a Web browser—thus, the basic Web architecture is also referred to as the browser/server (B/S) architecture. The basic workflow of a Web application is as follows:

- A user uses a Web client, usually a browser, and initiates a request to the Web server by typing a URL in the address bar or clicking a URL link on a Web page.
- The Web server receives the request, parses the URL, locates the corresponding document or script, and returns the document or else executes scripts and returns the result of the script to the client as a response. The response is commonly in HTML format.
- The Web client (the browser) receives the response, renders it, and presents it to the user.

The three foundational linchpins of the World Wide Web (WWW) are HTTP, the URL, and HTML. These protocols were invented by Tim Berners-Lee in 1990. These technologies are now international standards, and their specifications are managed and maintained by W3C (World Wide Web Consortium).

HTTP

Because it involves multiple parties, the Web needs a protocol that defines the rules and procedures for all parties to follow. As a protocol, **HTTP (Hypertext Transfer Protocol) defines a set of rules**

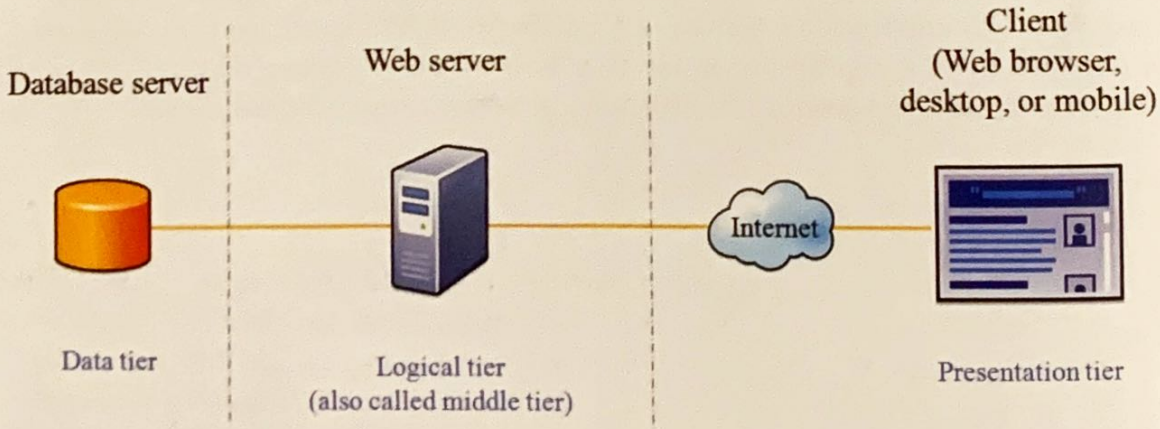

Figure 2.1 The basic architecture of a Web application includes three tiers.

and procedures that both Web clients and Web servers use to communicate with each other. For example, with HTTP, a Web server knows what information to put in the message header and what to put in the message body, and the Web client knows what to expect from both the response header and body.

HTTP defines eight methods of request: GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, and CONNECT. GET and POST are the two most commonly used requests. Both HTTP request and response messages have headers and bodies. The following information is typical of what can be found in an HTTP response header:

- **Cache control:** specifies directives that the client must obey (e.g., no cache)
- **Content type:** the MIME (Multipurpose Internet Mail Extensions) type indicates whether the message body is HTML (text/.html), a JPEG image (image/.jpeg), an MP3 audio file (audio/.mpeg3), or other file type
- **Status code:** denotes the status of the response (e.g., 200 means success, 403 means forbidden, and 500 means server error)

HTTP is characterized as simple, stateless, and flexible:

- **Simple:** In general, an HTTP communication scenario has three steps: (1) an HTTP client makes a connection to a Web server and sends a request method to the HTTP server; (2) the server processes the client's request while the client waits for the server's response; and (3) the server responds with the status code in the response header and data (if any) to the client and closes the connection.
- **Stateless:** After the server has responded to a client's request, the connection between the client and the server is typically dropped and forgotten. Servers do not retain information about the client between requests, which can greatly reduce the load on the server.
- **Flexible:** HTTP allows the transmission of any type of data object, and the content type can be specified in the HTTP header.

HTTPS (Secure Hypertext Transfer Protocol) refers to the use of ordinary HTTP over an encrypted Secure Sockets Layer (SSL) connection. With a plain HTTP connection, the data transferred between the client and the server can be intercepted. HTTPS avoids eavesdropping by encrypting the data. This ensures reasonable protection over the Internet, which is largely an insecure network. HTTPS is often used to transfer sensitive data such as personal user profiles, login passwords, and credit card information. Supporting HTTPS on a Web site requires installation of a certificate on the Web server that is issued by certification authorities.

URL

A URL (Uniform Resource Locator) is a subset of the URI (Uniform Resource Identifier) that specifies where an identified resource is available on the Internet and the mechanism for retrieving it. A URL is often called a Web address. Every Web page has a globally unique URL. Just as a street address is used to locate a household in the real world, a Web address is used to locate a Web page on the World Wide Web, which holds billions of Web pages. Without a URL, Web clients

would not be able to reach a Web server or the desired resources on the Web server. Without URLs, the Web resources could not be interlinked to form the Web.

The basic syntax of a URL is

```
Protocol: //hostname[: port]/filepathname?query_string#anchor
```

- Protocol specifies the transport protocol between the client and the server. Typical examples include HTTP, HTTPS, FTP, and MMS.
- The host name or IP address specifies the Web server to be reached. Sometimes, before the host name, there is a user name and a password, which can be used to connect to the server. The format is "username@password."
- The port number is optional. When the option is omitted, the default port for the protocol is used. For example, the default port for HTTP is 80, and for HTTPS is 443.
- The file path indicates the directory and file name of a resource on the Web server. Sometimes, the file path is an alias instead of a real directory or a real file name.
- The query string is optional and is one way to pass parameters to a script on the Web server. The query string normally contains one or more name-value pairs separated by ampersands, with names and values in each pair separated by an equal sign.
- The anchor in the URL specifies a location or a section on a Web page.

HTML

HTML (Hypertext Markup Language) is the main language for creating Web pages and is the source code for most Web pages. Similar to a Word document, HTML contains contents, a layout, and formatting information. When a Web page is loaded into a Web browser, the Web browser interprets the HTML code and displays the Web page the way the Web designer intended it to look. As a markup language, HTML is essentially plain text marked by a set of tags, such as head, body, table, center, and fonts. The appearance and layout information for HTML can be defined in Cascading Style Sheets (CSS), a scripting language used to style Web pages. CSS can be directly embedded in HTML files, or be in a separate file that is referenced in HTML files.

Web pages built with pure HTML usually have a plain user interface. HTML allows images, scripts, and objects to be embedded. Scripts such as JavaScript and objects such as browser plug-ins can implement logic and workflows to enhance the user interface.

2.1.2 RELATED TECHNOLOGY

This section covers Web application development technologies for the Web server and the Web client and the communications between them (figure 2.2).

SERVER-SIDE TECHNOLOGY

Server-side technology includes Web application servers and a range of programming languages that run inside them.

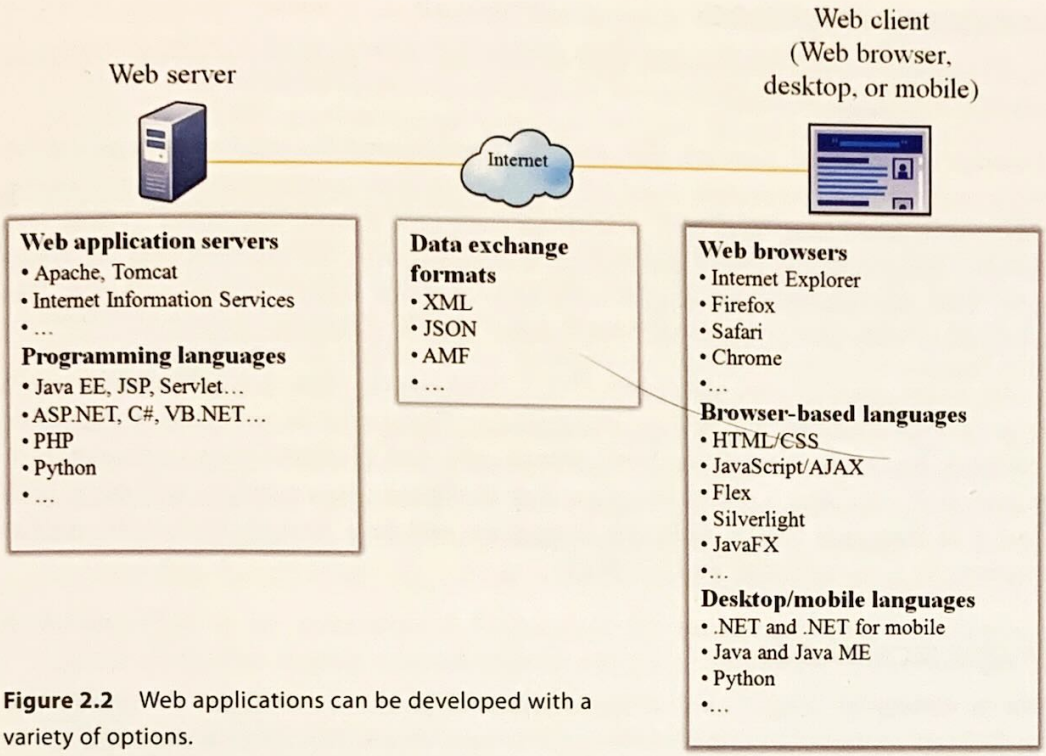

Figure 2.2 Web applications can be developed with a variety of options.

WEB APPLICATION SERVER

Web application servers host the Web sites that you read and serve the data for other Web applications to use. They are responsible for accepting HTTP requests from clients and serving clients with HTTP responses. Since they comply with HTTP specifications, Web application servers know how to communicate with Web clients. Web application servers are also containers that allow certain scripts such as JSP (JavaServer Pages), Java Servlet, and ASP.NET to run inside. Such scripts allow Web developers to implement certain business logic.

Examples of Web application servers include the following:

- **Apache Web Server and Tomcat** from the Apache Software Foundation: Apache is an open-source Web server characterized by simple, fast, and stable performance. It can run on almost all UNIX, Windows, and Linux system platforms. It is the most widely used Web server software (Netcraft 2009) and can support the Java suite of programming languages, and other languages when proper extensions are installed.
- **IIS** (Internet Information Services, formerly Internet Information Server), from Microsoft: IIS is a Web server application for use with Microsoft Windows. It is the world's second most popular Web server (Netcraft 2009) and can support the .NET suite of programming languages, and other languages when proper extensions are installed.
- Other examples of Web application server products include Nginx, Oracle/Sun Java System Web Server, and IBM WebSphere Web Application Server.

SERVER-SIDE PROGRAMMING TECHNOLOGY

Server-side programs execute on the Web server and within the Web application server container. The main programming languages include

- **JavaEE (Java Enterprise Edition), Servlet, JSP, and JSF (JavaServer Faces),** technologies of the Java family: These technologies are widely used for developing Web applications. James Gosling, coinventor of Java, once joked that "Java is C++ without the guns, knives, and clubs" (1996). Java is similar to the C++ programming language, though it drops some C++ features that are tricky if not used right. Web applications developed with Java can run inside Apache, Tomcat, the Oracle/Sun Java System Web Server, and IBM WebSphere Web Application Server.
- **ASP.NET,** an important part of the Microsoft .NET framework: The .NET framework is a common environment for building, deploying, and running Web services and Web applications. The .NET framework has technologies for both server-side and client/browser-side programming development. ASP.NET has a set of libraries that facilitate users writing ASP.NET Web pages (.aspx) in C# (a language that is different from C++ and Java though built with similar syntax and semantics to C++ and Java) and VB.NET.

CLIENT/BROWSER-SIDE TECHNOLOGY

Web clients include a variety of programs and applications that can run inside or outside the browser, including desktop applications and mobile applications, which run outside the browser. In this section, we focus on the Web browser, the most popular type of Web client.

WEB BROWSER

A Web browser is a software application for retrieving and presenting information resources on the Web. It provides a means for looking at and interacting with the Web. For most users, Web browsers represent the "face" of the Web. Technically, a Web browser is a client that implements HTTP specifications, HTML specifications, and JavaScript specifications, meaning that it knows how to communicate with Web servers, how to display an HTML, and how to interpret and execute JavaScript.

The most popular Web browsers are Internet Explorer (IE) and Mozilla Firefox (Net Applications 2009). Other Web browsers include Apple Safari and Google Chrome. There are slight differences in how these browsers support HTML and JavaScript specifications, meaning that the same Web page may appear and behave differently in different Web browsers.

BROWSER-SIDE PROGRAMMING TECHNOLOGY

- **JavaScript,** invented by Netscape in 1995, is a scripting language that primarily runs inside Web browsers to make Web pages dynamic and interactive. It is the most popular scripting language, used in millions and millions of Web pages. JavaScript is not Java, although its basic syntax is intentionally similar to both Java and C++ to reduce the number of new concepts required to learn the programming language. JavaScript is simple so that nonprofessional programmers can work with it, and it is safe to use because it has limited access to the local hard drive of the

client computer where it is running. JavaScript is usually platform independent, but there are circumstances when a JavaScript behaves differently in different browsers. Developers who want to build cross-browser applications need to be careful about possible incompatibility. The combination of HTML and JavaScript is called DHTML (dynamic HTML) and can be used to create Web sites that are interactive as opposed to static.

- **AJAX (Asynchronous JavaScript and XML)** has become popular since 2005. It is not a new programming language but a group of existing Web development techniques that are used to create better, faster, and more interactive Web applications on the browser. AJAX uses asynchronous communication between the browser and the Web server so that a Web page can retrieve data from the server in the background (asynchronously) without interfering with the display and behavior of the existing page. Without AJAX, you generally have to move from page to page and wait for a new page to be loaded before you can interact with it. With AJAX, you can remain on a page while the data needed from the server is retrieved, in XML or other formats, and then parsed and updated on the same page. The main contribution of AJAX is improvement of the user experience. From a software engineering point of view, AJAX allows developers to separate data from formatting, which is the preferred software design pattern.
- **Adobe Flex** is an open-source framework for building highly interactive, expressive Web applications that deploy consistently in all major browsers. It can present sophisticated animation and transmission effects, which makes it a powerful language for implementing rich and engaging Web applications, or rich Internet applications (RIAs, see section 2.4.2). Flex uses MXML, an XML-based language, to describe user interface layouts and behaviors, and ActionScript, a strong-type, object-oriented programming language, to create client logic. Programs developed with Flex can run inside Web browsers with the Adobe Flash Player plug-in or outside Web browsers on Adobe AIR, the cross-operating system runtime environment. This enables Flex applications to run consistently across all major browsers and platforms.
- **Microsoft Silverlight** is a cross-browser and cross-platform plug-in for delivering RIA that uses graphics, animation, or video within the .NET framework. It uses XAML (Extensible Application Markup Language) to develop the user interface, and .NET programming languages such as C# or VB.NET to develop business logic. Microsoft Silverlight programs can run inside Web browsers with a Microsoft Silverlight plug-in or on the desktop with a WPF (Windows Presentation Foundation) runtime environment. This makes Microsoft Silverlight a common method for developing browser-based and desktop applications.
- **Oracle/Sun JavaFX** is the Oracle/Sun software platform intended for creating and delivering RIAs that can run across a wide variety of devices. JavaFX is fully integrated with the Java Runtime Environment (JRE), which ensures that JavaFX applications can run in any browser, desktop, or mobile phone with the proper JRE installed.

DATA EXCHANGE FORMAT

Contemporary Web technology prefers to separate the data from the formatting and the exchanging of data from the presentation. The data format used for exchanging messages between the server and the client can greatly influence the load to bandwidth and therefore the efficiency of the application. There are three main formats for exchanging data over the Web:

- **XML (Extensible Markup Language)** is a markup language that allows you to define your own tags and attributes. XML has many strengths in that it is platform independent because it is a plain text file; it is self-descriptive since its tags and attributes usually have easy-to-understand names; it can be processed automatically, because it is well structured; and it can be validated against its schema. Because of these advantages, XML is probably the most commonly used data exchange format on the Web. However, it also has some disadvantages. It is bulky because it uses tags to delimit the data. And parsing XML (i.e., extracting values from certain nodes in an XML) is not efficient, especially in languages such as JavaScript, the most commonly used scripting language on the Web.
- **JSON (JavaScript Object Notation)** is a lightweight computer data interchange format. It is derived from the object literals of JavaScript, as defined in the ECMA (European Computer Manufacturers Association) Script Programming Language Standard. JSON is smaller in size than XML, and it is more efficient to parse in JavaScript where JSON is a native data type. The main application of JSON is in Web application programming, where it serves as an alternative to the XML format. The following example uses XML and JSON to describe the same information concerning the names and hobbies of two students.

An XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
      <student>
              <name>John</name>
              <hobby>Basket Ball</hobby>
      </student>
      <student>
              <name>Lisa</name>
              <hobby>Movie</hobby>
      </student>
</  students >
```

A JSON example

```
{"students": [
              {"name":"John", "hobby":"Basket Ball"},
              {"name":"Lisa", "hobby":"Movie"}
        ]
}
```

- **AMF (Action Message Format),** developed by Adobe, is a binary format used primarily for exchanging data between Adobe Flex/Flash Web client applications and Web servers. AMF is a native data type in Flex, making it more efficient to process than JSON. Compared with XML, which is a native data type in Flex as well, AMF is much smaller than XML, making AMF more efficient to transfer and process. Although AMF is mostly used in Flex applications, there are implementations of AMF in PHP, Java, and .NET as well.

2.2 WEB GIS BASIC ARCHITECTURE AND COMPONENTS

Web GIS extends a basic Web application by giving it GIS capabilities. The basic architecture of Web GIS is similar to Web applications but with the addition of GIS components (figure 2.3). In a basic workflow, a user uses a Web GIS application through a client, which can be a Web browser, a desktop application, or a mobile application. The client sends a request to the Web server over the Internet via HTTP. The Web server forwards GIS-related requests to the GIS server. The GIS server retrieves the needed data from the GIS database and processes the request, which can be to generate a map, conduct a query, or perform an analysis. The data, map, or other result is sent by the Web server to the client in a response via HTTP. The client then displays the result to the user, which completes the request and response cycle.

2.2.1 WEB GIS SERVER

The Web GIS server is the most important component in a Web GIS. Its functionality, ability to be customized, scalability, and performance are critical to the success of the Web GIS application. The capability and quality of a Web GIS application is largely determined by the Web GIS server it uses.

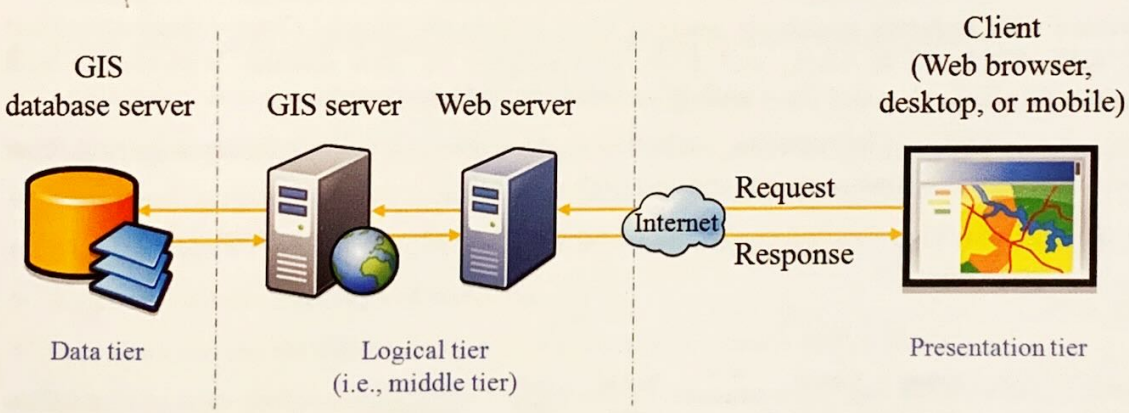

Figure 2.3 In the logical architecture and workflow of a basic Web GIS, the components can all be deployed on one computer, but logically, they are still three separate tiers.

GIS server technology has been evolving over the last two decades. Using ESRI products as an example, the first-generation commercial GIS server products ArcView IMS and MapObjects IMS were released around 1996. They represented an important addition to the GIS software family and revealed a new direction in GIS software products moving toward the Web platform. The second-generation commercial GIS Server released by ESRI was ArcIMS, around 1998. It offers enhanced performance and extended functionality, but it lacks the extensive functionality of the third-generation product, ArcGIS Server. ArcGIS Server offers a more comprehensive set of functionality (figure 2.4), including the following:

- Dissemination of 2D maps, 3D globes, query, search, feature editing, data extraction, tracking, imagery, address and place finding, routing, geometry processing (e.g., transferring coordinate system), and metadata capabilities to the Internet or intranet.
- Support of advanced and customizable geoprocessing capabilities.
- Delivery of capabilities via Web services, which are Web-based programming components that can be reused and remixed to create new applications. Web services technology can shrink data and application redundancy, optimize system configurations, and consolidate enterprise systems. The recent popular REST (Representational State Transfer)-style Web services have introduced many new benefits in addition to the traditional SOAP (Simple Object Access Protocol) Web services (see chapter 3).
- Compliance with industry standards, such as OGC (Open Geospatial Consortium)-compliant WMS, WFS, WCS, CSW, GML, and KML (see chapter 3). This enables Web-service-level interoperability (e.g., plug and play) among products by different vendors.
- Ways to achieve high-quality services, including cached services, optimized services, and other methods to accomplish fast performance and high scalability.
- Token services and other methods to secure geospatial Web services (see chapter 3).

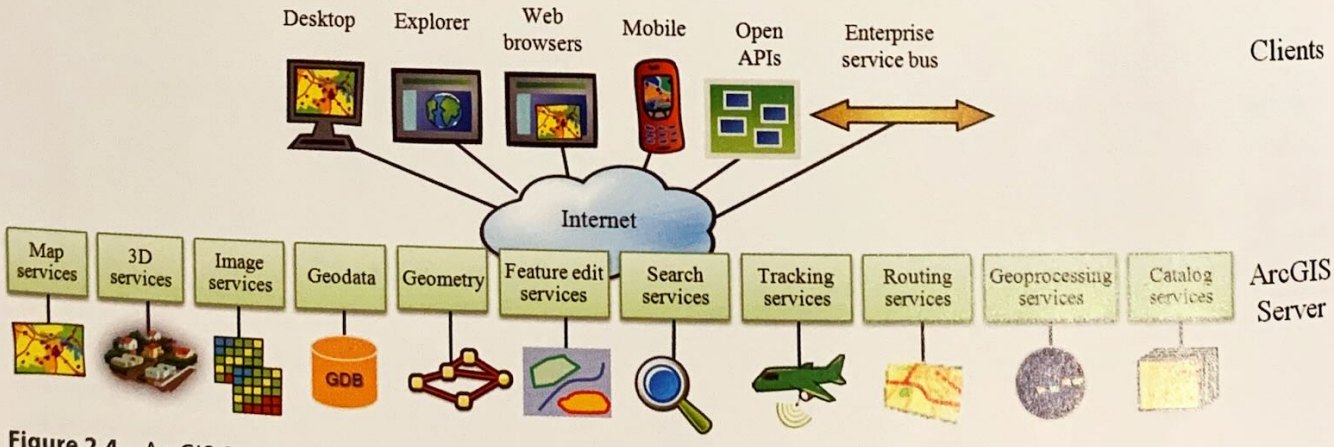

Figure 2.4 ArcGIS Server provides an array of functions to a variety of clients via Web services.

- Ability to be highly customized. Supports a variety of modern client-side and server-side APIs that can build highly interactive and engaging client applications for Web browsers, desktops, and mobile devices (see "Web GIS clients" below).

2.2.2 GIS DATABASE

The GIS database is the data storage and management framework for a GIS. It can hold a collection of geographic datasets of various types, such as basic vector data (points, lines, and polygons) and raster data (satellite and aerial images, for instance). Some GIS databases support more data types such as CAD data, 3D data, utility and transportation system data, GPS coordinates, and survey measurements. GIS databases range from a small, single-user database up to a larger, enterprise-level database, which can be edited and accessed by many users simultaneously. While some GIS databases store only collections of individual features, others manage the data models that define the spatial relationships and behaviors that are critical to many GIS tasks and analytical operations.

The GIS database is the underlying support for Web GIS applications. The answers delivered from a Web GIS application can only be as good as the quality of the information contained in the GIS database. While casual applications can rely on casual data sources, professional applications tend to need high-quality, authoritative, and up-to-date geographic information.

A strong GIS database product generally includes the following capabilities:

- Stores a rich collection of spatial data in centralized or distributed systems
- Applies sophisticated rules and relationships to the data
- Defines geospatial relational models (e.g., topologies, road networks)
- Maintains integrity of spatial data with a consistent and accurate database
- Works within a multiuser access and editing environment and supports versioning
- Supports custom features and behavior
- Provides a means for robust data security, backup, recovery, and rollback
- Maintains high performance when the volume of data increases and the number of simultaneous users increases.

2.2.3 WEB GIS CLIENTS

The client in a Web GIS application can play two roles. First, it represents the end-user interface for the entire system. It interacts with the user, collects user inputs, sends requests to the server, and presents the results to the user. Most end users don't know, and don't need to know, the particulars of the back-end server(s). All that they know about the system is how fast, stable, and user-friendly the client is. Second, the client, especially a thick client (see section 2.3), can also perform some geospatial processing tasks, such as dynamic classification for thematic mapping, cluster, and heat-map analysis. Web GIS clients are typically Web browsers but can also be desktop applications, mobile applications, or even server applications (when a server acts as a client of another server).

WEB BROWSER CLIENT

Web browsers are the main type of Web GIS client. Early browser clients were static, plain, and dull by today's standards. More recently, Web browsers have surpassed the static HTML and simple JavaScript stage. AJAX, Flex, Silverlight, and JavaFX technologies can create a rich, dynamic, and user-friendly interface known as an RIA (see section 2.4.2). With additional APIs such as ESRI's ArcGIS API for JavaScript, Flex, or Silverlight, Web browsers are now able to easily perform many types of GIS functions (see chapter 4). Examples of Web browser clients include ArcGIS Explorer Online.

DESKTOP APPLICATION CLIENT

Desktop applications such as those developed with Java, .NET, or other programming languages can act as Web GIS clients. These full-featured applications do not run inside a Web browser, so they are not limited to the "sandbox" environment of a browser (i.e., a tightly controlled set of resources for guest programs to run in). Desktop clients can access local resources, such as local files, the local database, and peripheral devices. They are rigorous and especially fit for resource-intensive applications.

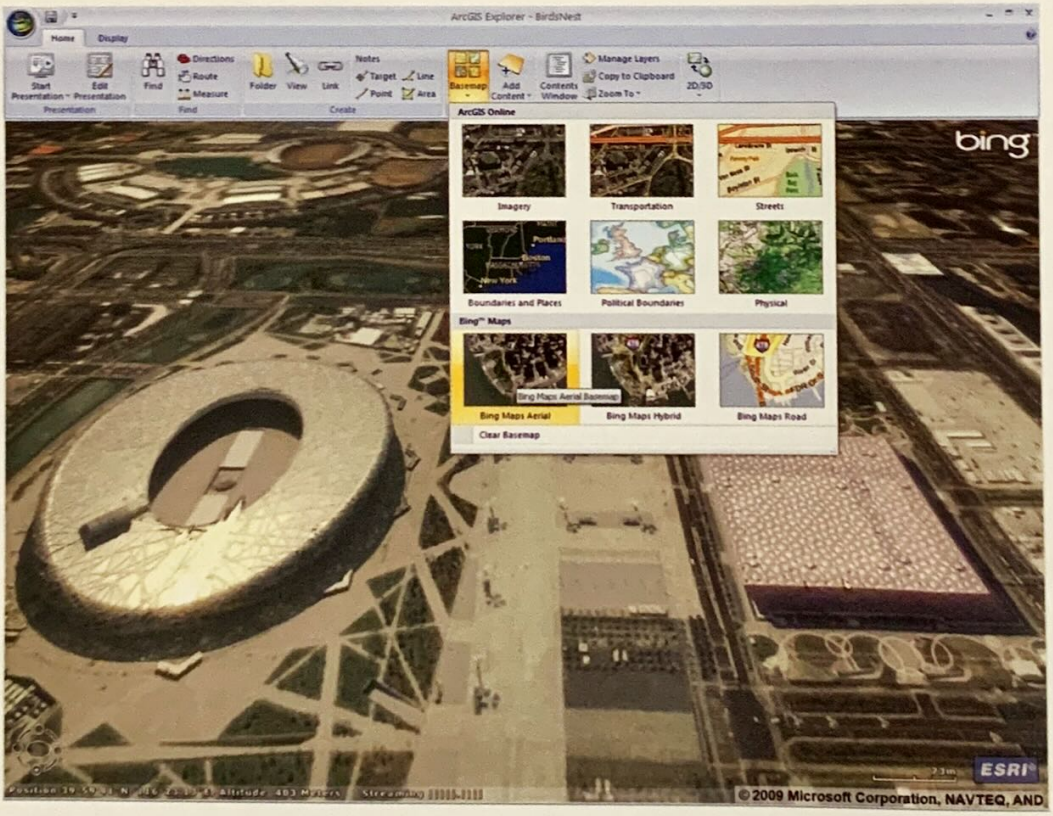

Figure 2.5 ArcGIS Explorer Desktop is a desktop Web GIS client that can view map services from ArcGIS Online, Microsoft Bing Maps, OGC standard map services, and users' own ArcGIS Server Web services.

Courtesy of Microsoft, NAVTEQ, and AND Automotive Navigation Data.

Examples of desktop clients include ArcGIS Explorer Desktop (figure 2.5), Google Earth, and ArcGIS Desktop. The role of many traditional desktop GIS products has been changing—in addition to running alone or being the client of a local server, they can also be used as the client of Web GIS applications. They can be a good solution when browser-based clients are not powerful enough, especially for professional users. For example, ArcGIS Desktop can use the maps and many other types of services available over the Web, or in the cloud, and perform many advanced operations that are not available in browser clients.

MOBILE CLIENT

Mobile devices, which allow the use of GIS on the go, are becoming a popular type of Web GIS client. Mobile clients basically can be classified into two categories: browser based and native application based. The capabilities of mobile browsers vary greatly. While some mobile browser-based clients can only deliver simple GIS functions with a plain user interface, others are getting close to being like the fully fledged Web browsers we use on desktops that can deliver a greater variety of GIS functions with a richer user interface. Native mobile applications, developed with .NET Mobile, Java Mobile Edition (Java ME), or other languages, have the added advantage of being able to access local peripheral devices such as GPS to acquire an accurate location of the mobile user and enable a range of applications such as survey, rescue, real-time traffic, navigation, and locating such things as businesses or friends (see chapter 5).

Examples of native mobile application clients (figure 2.6) include ArcGIS Mobile, ArcGIS for iPhone, Microsoft Bing Maps Mobile, Google Earth Mobile, and MapQuest Mobile (see chapter 5).

While most Web GIS clients are designed for specific applications, some clients can work with a variety of Web content types. Such clients are called geobrowsers. **Geobrowsers refer to map viewers that provide the ability to visualize standard Web map services and data available over the Web.** The standards for Web services, data, or maps available over the Web, which aim

Figure 2.6 ArcGIS Mobile and ArcGIS for iPhone are clients of ArcGIS Server. They can display map services and query data hosted remotely on ArcGIS Server, delivering GIS capabilities to the field. Left: the ArcGIS Mobile out-of-the-box application shows the default user interface. Right: ArcGIS for iPhone shows a basemap overlaid with utility networks.

Courtesy of Tele Atlas North America, Inc., and Trimble.

to achieve interoperability and allow the products of different vendors to work in a "plug and play" environment, are explained in chapter 3. Here are some examples of geobrowsers:

- The Carbon Project Gaia is a 2D geobrowser. It can access an array of geospatial resources such as the OGC WMS, WMTS, WCS, WFS, KML, GML, Microsoft Bing Maps, Yahoo Maps, and OpenStreetMap, as well as file formats such as the ESRI shapefile format.
- ArcGIS Explorer Desktop can be used as either a 2D or 3D geobrowser. It can display data and service standards such as OGC WMS, KML/KMZ, GeoRSS, ArcGIS Server services, and ArcIMS services.
- ArcGIS Desktop is also a geobrowser in that it can view and query OGC WMS, WFS, and WCS, as well as ArcGIS Server services.

A virtual globe is another type of popular Web GIS client. **A virtual globe is a 3D software model or representation of Earth or another world** (Butler 2006; Foresman 2008). It allows you to freely

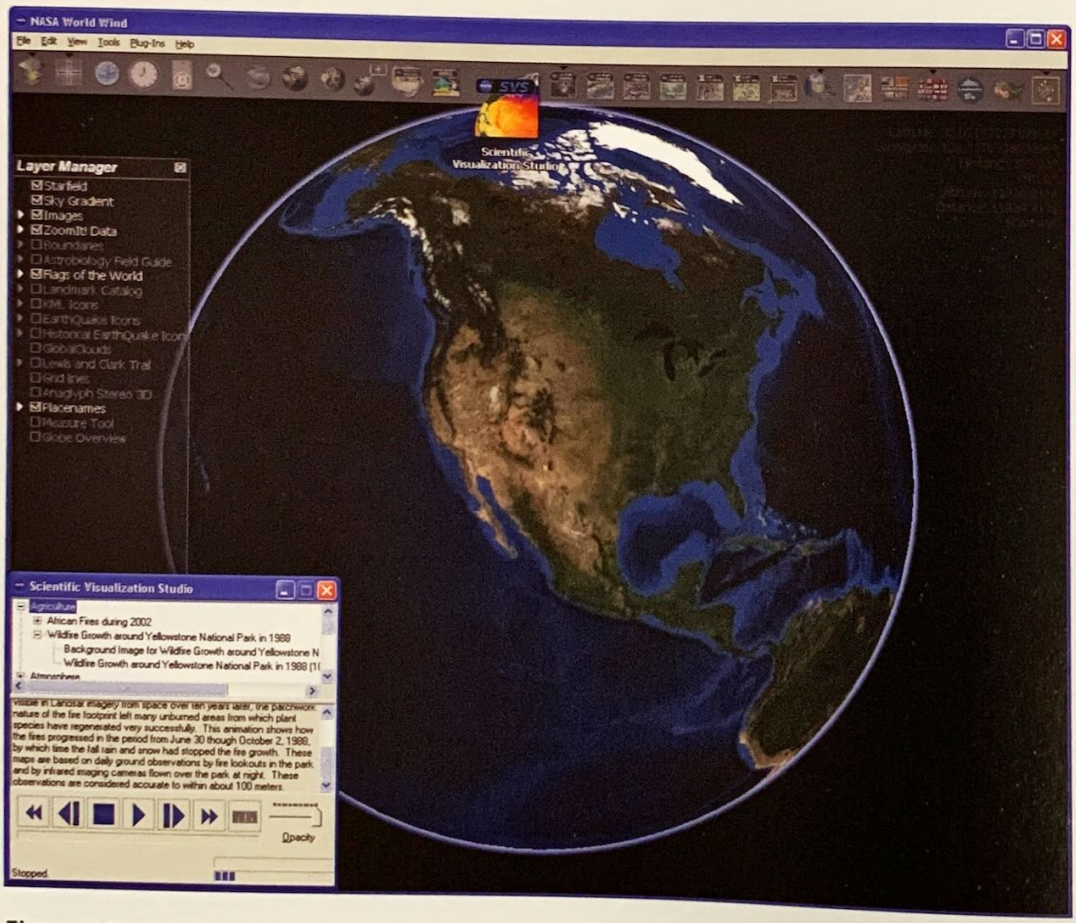

Figure 2.7 NASA World Wind is the first widely publicized online virtual globe (released in mid-2004). It lays NASA and USGS satellite imagery, aerial photography, topographic maps, and publicly available GIS data on top of 3D models of Earth or other planets.

Courtesy of NASA.

move around in the virtual environment by changing the viewing angle and position. Many virtual globes fetch and display data from the Web and are thus named online virtual globes. **An online virtual globe is a virtual globe that relies on the Web for its data and maps.**

Online virtual globes are mostly free to use. Some popular ones include NASA World Wind (figure 2.7), Google Earth, ArcGIS Explorer Desktop, Microsoft Bing Maps, and SkylineGlobe. Most online virtual globes are also geobrowsers, because they can drape and visualize a variety of standard geospatial contents across their surface.

2.2.4 CHALLENGES

A number of bottlenecks and other factors can challenge the quality of a Web GIS application (figure 2.8). They include the following:

- Pressure on the GIS server to support many users
- Pressure on the GIS database to support frequent data read/write
- Pressure on the Internet to frequently transfer large volumes of data
- Limited GIS capabilities of the client, especially Web browsers
- Inexperienced nonprofessional end users

Various solutions are introduced throughout the book, including Web services optimization, such as caching in advance, algorithm and system tuning, failover and load balance, and efficient use of Internet bandwidth (see section 3.5), proper workload partition between the client and the server (section 2.3), and effective user experience design (section 2.4).

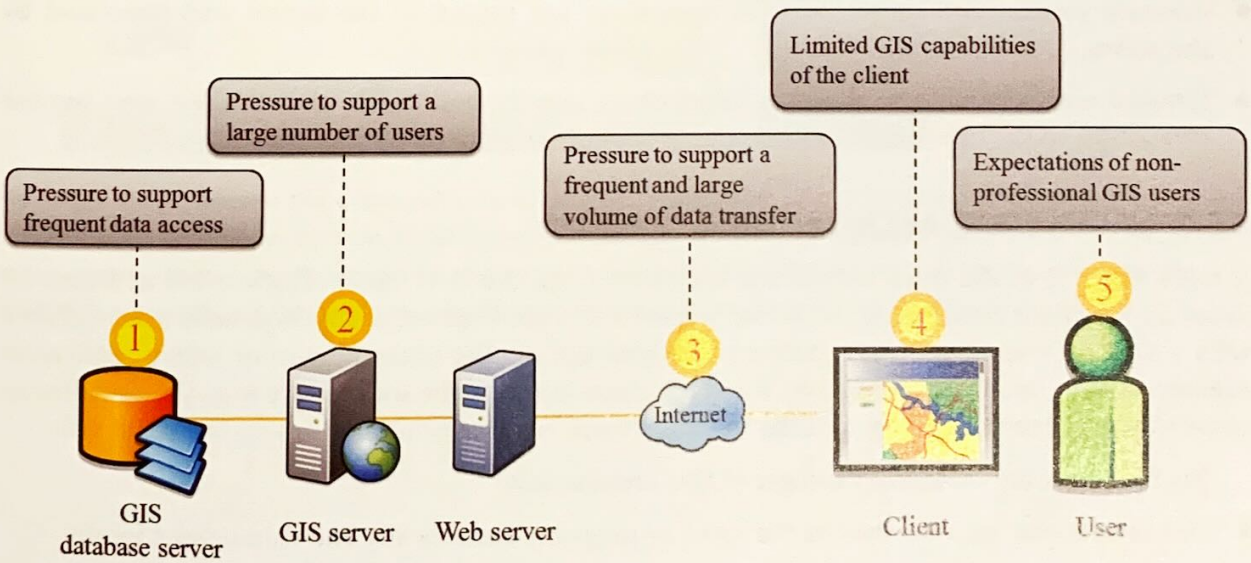

Figure 2.8 A variety of bottlenecks can challenge the quality of a Web GIS application.

2.3 THIN VERSUS THICK CLIENT ARCHITECTURE

A basic design consideration for any client-server system is how to partition the workload between the client and the server. Depending on how the workload is distributed, Web GIS applications can be categorized as either thin client architecture or thick client architecture (Gong 1999).

Both Web GIS servers and clients are becoming more powerful as technology advances, with the trend moving toward thick client architecture. But how thick should the client be? What is the best strategy to split the work between the server and the client? This section presents the best practice for addressing these questions.

2.3.1 THIN CLIENT ARCHITECTURE

Thin client architecture relies on the server to perform most of the work, leaving the client to do the least amount. The client simply sends the user's requests to the server. The server does the processing, such as generating a map and performing analysis. The results, typically in HTML format embedded with GIF, PNG, or JPEG images, are then returned to the client and displayed for the user.

An example of thin client architecture is the PARC Map Viewer (figure 1.6), the first Web GIS application in the world. The advantages of this approach are the following:

- The user doesn't need to install any software other than a Web browser, not even a plug-in.
- Since most of the heavy-lifting processes are done by the server, the client doesn't need a powerful computer, so these types of applications can work well even on low-end computers.

The main disadvantages are the following:

- Pressure on the GIS server: all GIS operations are routed to the server and processed by the server.
- Limited user interaction: the user interface is usually built with plain HTML and limited JavaScript, so the user interaction is not particularly smooth or compelling.

2.3.2 THICK CLIENT ARCHITECTURE

A thick client is at the other end of the spectrum from the thin client. Thick client architecture relies on the client rather than the server to perform most functions. This is usually accomplished with a Web browser plug-in or a native client application. The client plug-in or native application executes locally on client computers. The thick client requests the source data (e.g., the coordinates of vector data) from the server, and then renders maps and performs analysis on the client side.

The following are typical advantages of this architecture:

- Fast interaction with the user as the logic, or program, runs locally and data resides locally
- Less pressure on the server since there are fewer round trips to the server

The following are the disadvantages:

- Inconvenience associated with the installation of browser plug-ins or installation of native applications. There are cases when the user is not allowed to install plug-ins or native applications —on some government computers or company computers, for example.
- Limitations posed by the Internet bandwidth and client computing power. Typically, it is not feasible to transfer gigabytes of data over the Internet and have the client perform sophisticated GIS operations.

2.3.3 BEST PATTERN FOR WORKLOAD PARTITIONING

Web technology advances rapidly. Client-side technology such as plug-ins and JavaScript is becoming more and more powerful and can take on greater and greater workloads. A contemporary popular design strategy involves breaking down the workload into several categories and properly distributing it between the server and the client (figure 2.9).

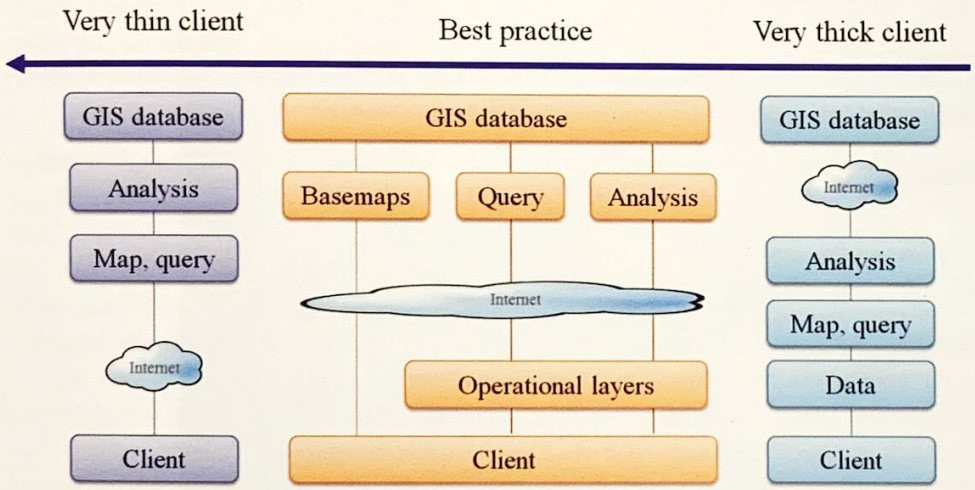

Figure 2.9 In the extreme thin client architecture, most GIS functions are performed by the server, while the opposite is true for extreme thick client architecture. Current best practice recommends basemaps done by the server and operational layers typically rendered by the browser, unless the data size is too large for the browser to handle. Essentially, easier functions are performed by the browser and complex functions are handled by the server.

This best-practice strategy (equation 2.1) recommends breaking the typical Web GIS application functions into basemaps, operational layers, and tools (Quinn 2008; Brown et al. 2008).

EQUATION 2.1

Web GIS application = basemaps + operational layers + tools

Where

- basemaps are usually precreated or created dynamically by the server;
- operational layers are typically dynamically rendered by the client, but should be rendered by the server if the data size is too large; and
- easier tasks are done by the client and complex ones by the server.

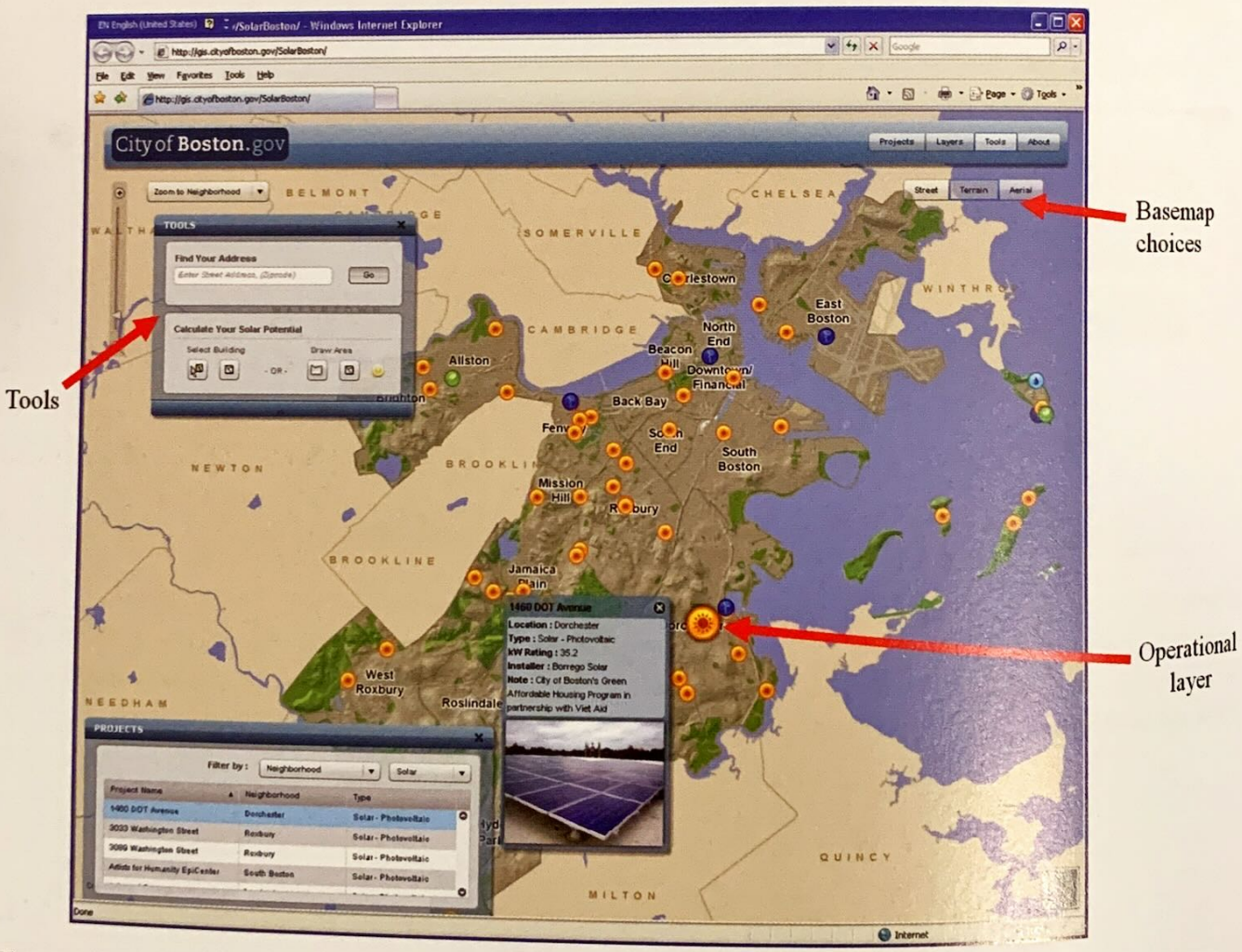

Figure 2.10 The Solar Boston Web application represents a best-practice design pattern that consists of basemaps, operational layers, and tools.

Courtesy of Boston Redevelopment Authority.

The Solar Boston Web GIS application (figure 2.10) is an example of best-practice design. Solar Boston is part of the Solar America Initiative, a campaign launched by the U.S. Department of Energy (DOE) to make solar electricity cost-competitive with traditional electricity production by 2015. To help meet the city's greenhouse gas reduction targets and support the goals of DOE, the City of Boston set a target of 25 megawatts of solar power to be installed by residents and organizations by 2015. To support the Solar Boston program, Boston is using Web GIS technology to map current solar installations, track progress toward the goal, and allow Bostonians to analyze their rooftop solar energy potential (Knight 2009). The Solar Boston Web GIS application was developed by the Boston Redevelopment Authority (BRA) in 2008 using ArcGIS Server and an early version of ArcGIS API for Flex.

BASEMAPS

Basemaps provide the locational reference or context for an application. They serve as the foundation for all subsequent operations. Best practice recommends that basemaps be generated by the server. Since a basemap tends to be relatively static, and in a typical setting is updated on an infrequent basis, basemaps should be cached, meaning the server should generate a set of map tile images in advance at predetermined scale levels for rapid display (see section 3.5.1). A basemap should be multiresolution and designed to be portrayed appropriately at a range of supported map scales.

The Solar Boston application provides three types of basemaps: a street map, terrain map, and aerial imagery. These basemaps were cached in advance by ArcGIS Server at multiple scales ranging from 1:80,000 to 1:1,250. The maps are based on the City of Boston's authoritative and detailed data, including the rooftops of each building. The street maps and terrain maps allow users to see the streets and building footprints as they zoom in from city level to building level. The terrain maps use the hillshade effect to portray the topography of the city, while the high-resolution aerial imagery gives users a sense of the city's geography. These basemaps provide context for the application and let users get oriented on what Boston looks like, where the green energy projects are located, and how these projects relate to other features such as surrounding buildings.

OPERATIONAL LAYERS

Operational layers are drawn on top of the basemap and contain the themes that end users will view and work with. Operational layers consist of, but are not limited to, the following types:

- **Observations or sensor feeds.** This material can be information that reflects status or situational awareness—for example, crime locations, traffic sensor feeds, real-time weather, readings from meters (such as stream gauges), observations from equipment or made by workers in the field, inspection results, addresses of customers, disease locations, or air quality and pollution monitors.
- **Editing layers.** These are the map layers that users work with—for example, to edit features. Common examples are the facilities layers edited by a utility worker.

- **Query results.** In many cases, applications will make a query request to the server, which will return a set of records as results. These can include a set of individual features or attribute records. Users often display and work with these results as map graphics in their Web GIS map applications.
- **Result layers that are derived from analytical models.** GIS analysis can be performed to derive new information that can be added as new map layers, and then explored, visualized, interpreted, and compared by end users.

Typically, operational layer data is small in size or is only displayed when users zoom to a certain level. The data is streamed to the client, and then rendered and managed by the client. There are several reasons behind this pattern. First, operational layers are often dynamic, and thus generally should not be cached in advance. Second, users need to interact with these layers to conduct their daily work. They expect operational layers to be responsive, reporting details on a mouse click and showing links to certain functions. The dynamic rendering and user interaction fall on the client side and can be implemented using browser APIs such as ArcGIS APIs for JavaScript, Flex, or Silverlight (see chapter 4). Occasionally, operational layer data can be too large to be effectively rendered by the browser. In this case, operational layer data should be rendered by the server.

In the Solar Boston application, the green energy projects layer is the main operational layer. The coordinates and attributes of the projects are extracted from the server to the browser. The browser uses the ArcGIS API for Flex to dynamically classify each project, displaying different icons for different types of projects (solar, biomass, water, or wind). When the mouse pointer hovers over a project, the project will display its attributes along with a photo in the message window.

TOOLS

Web GIS applications usually provide tools that perform processes beyond mapping. These tools range from the common types, such as finding an address or place-name, routing, and searching points of interest that match certain criteria, to more specialized tools that implement specific business logic for an enterprise. The solution varies as to whether the processes should be done by the server or the browser.

- **Have the client do it:** This method fits processes that are relatively easy and when the data needed is all on the client side. Typical examples include charting analysis results and generating heat maps based on a set of points (see figure 4.8 B and D).
- **Have the server do it:** This fits processes that are complex and when the data needed is not housed on the client side. Typical examples include finding and routing to the closest facility, calculating stream flows, and finding the best habitat by overlaying a number of data layers.

The Solar Boston application allows you to select a rooftop or draw a polygon on the map, and then calculate your monthly solar energy potential (figure 2.11). The solar potential calculation is computational intensive (Fu and Rich 2002). The calculation needs to be handled by the server because the algorithm is too complex for the browser and because the digital elevation model and building height needed by the algorithm are too large to be streamed to the browser. The calculations are done by the server in advance for existing rooftops (see section 3.5) and are done dynamically for the areas that users draw. The resultant monthly solar energy potentials are sent to the browser and displayed in charts.

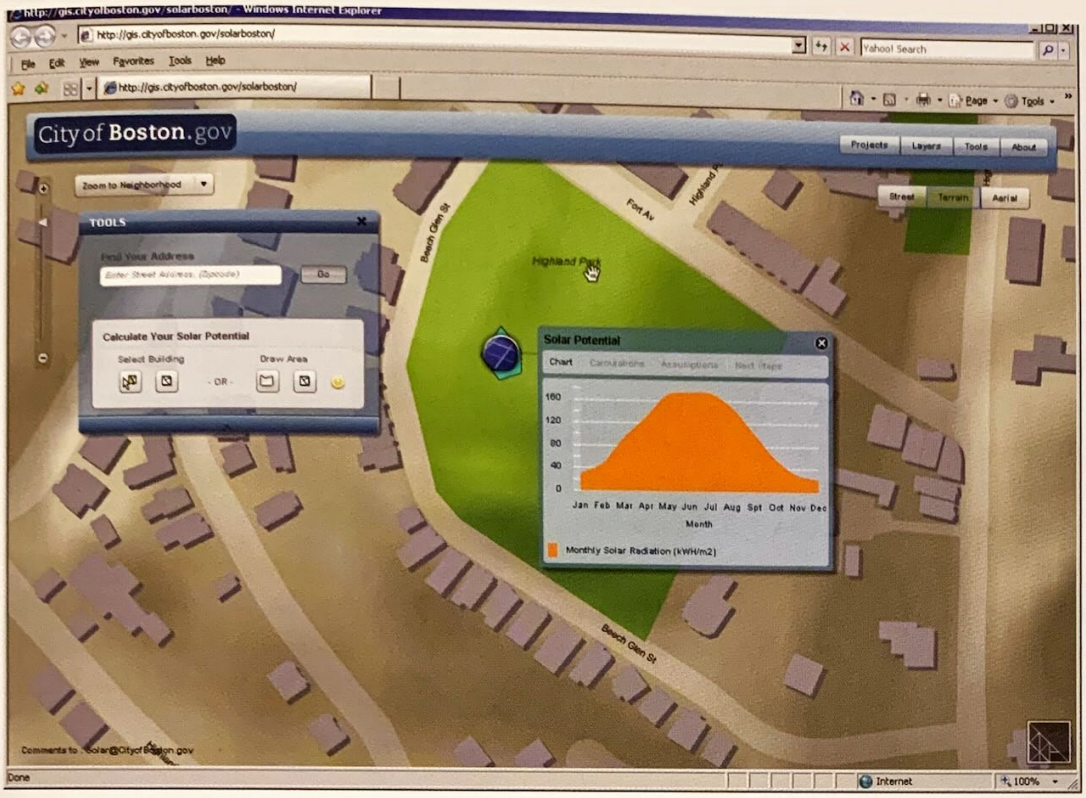

Figure 2.11 With the Solar Boston Web GIS application, users can select a rooftop or draw a polygon on the map, and then have the monthly solar energy potential calculated. The calculation is done by the server because of its complexity and the large volume of data needed.

Courtesy of Boston Redevelopment Authority.

2.4 USER EXPERIENCE DESIGN

The user experience has become an important consideration for Web application design, including Web GIS design. **User experience (UE or UX) refers to the level of satisfaction a person gets from using a product or a service.** Traditionally, GIS is most often used by professionals in the workplace, who have years of GIS training or experience and who know how to use GIS. Thus, professional GIS design has mostly focused on functionality and neglected improving the user experience. By unlocking GIS technology, the Web has delivered GIS to a large number of users who have no concept of GIS except that they expect it to be as easy to use as a regular Web page and as fun to use as a multimedia center. To meet user expectations, the Web should wrap necessary functionality with a pleasant user experience conveying the three main principles of UE design—making Web applications fast, easy, and fun to use. The first principle, speed, is mostly determined by the server and is closely related to Web service optimization (see section 3.5). The latter two principles are introduced in the following sections.

2.4.1 EASE OF USE

The most important factor in making a Web application easy to use is the maxim "Don't make me think!" (Krug 2000). This is in contrast to other mistaken criteria such as "no more than three clicks to content." What really counts instead of the number of clicks is how hard it is to click each time—that is, how long it takes to find the right place to click. Each click should be painless, unambiguous, and give you continued confidence that you are on the right track. Otherwise, a Web application can have a big learning curve that can turn users off.

Brian Noyle (2009), Clint Brown et al. (2008), and many others have outlined several points about how to make a Web GIS easy to use, including the following:

- **Hide complexity:** Web GIS should create intuitive and focused applications that serve a particular purpose, that provide as few tools and few layers as needed, and that don't flood users with unnecessary functions, features, and data layers that won't be used and that distract you by making you have to think about which tool to use. The terms should be self-explanatory, not GIS jargon.
- **Provide feedback:** Many GIS functions take multiple steps to complete. A good application should lead users through a well-defined workflow with visual cues, provide consistent and meaningful feedback in the user interface, and assure users that they are on the right track.

2.4.2 MAKING IT FUN

In the early 1990s, the world was impressed to see even a picture image appear on the Web. That kept users excited for a few years until they began to expect something even more impressive—with things like animation effects, compelling multimedia, and an intuitive interface. To keep pace with users' rising expectations, **rich Internet application (RIA)** technologies came into play. The term RIA was coined by Jeremy Allaire (2002) to describe the benefits of a new version of the Adobe Flash Player. Later, **RIA came to generally refer to Web applications that provide a rich and engaging user experience comparable to desktop applications.** RIAs are typically developed with AJAX, Flex, or Silverlight.

Higher user expectations and advances in RIA technology have resulted in many RIA-style Web GIS applications that use browser APIs such as ArcGIS APIs for JavaScript, Flex, and Silverlight. These applications can fetch map tiles for the current map extent and nearby areas and use asynchronous communications behind the scenes to provide users with smooth map pan and zoom. They can display operational layers vividly by using animation icons to simulate flying helicopters or flag terrorist attacks. Animation icons show situational details when the pointer hovers over a feature on the map. They can use pie or column charts to present demographic statistics for areas surrounding a new retail store. They can link locations on the map with photos of highway traffic and real-time videos of critical infrastructure. They can overlay maps on a video to simulate the path of an earthquake. They can play multiple temporal aerial photos in sequence to produce a movie effect showing historic land-use change. They can employ animation effects such as zoom, fade, flip, rotate, bounce, fly, ripple, sparkle, and many more—all demonstrating that GIS can be fun to use. These rich and engaging user experiences, when employed appropriately, can greatly increase user productivity and improve user satisfaction.

This chapter summarized the basic architectures, components, and programming technologies of Web applications and Web GIS applications. While not intended to be a "how to," the contents provide guidelines on design questions such as which basemap to use, whether to use one available on the Web or create it in-house, what the operational layers should be, how users should interact with them, what tools are needed, whether the processes should be done by the client or the server, and how to achieve a good user experience. Properly following these guidelines can lead to Web GIS applications that ensure good usability and fast performance.

Study questions

1. What are the three foundational technologies of WWW?
2. What is the three-tier architecture of a Web application?
3. List the main Web server technologies and the main browser technologies.
4. What are the typical data formats for exchanging data between the Web server and Web client? Compare them.
5. What are the components of a basic Web GIS application?
6. What is a geobrowser? What is an online virtual globe? Give some examples.
7. What is a thin client and what is a thick client in the context of Web GIS?
8. What is the best strategy to partition workloads between a server and a client in a Web GIS application? Give a good example of such a Web GIS application.
9. What is user experience, and what are the main principles of user experience design?
10. What is RIA? Describe a Web GIS application that you think is RIA. What technology does it use to achieve RIA?

References

Allaire, Jeremy. 2002. Macromedia Flash MX—A next-generation rich client. Macromedia white paper. `http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf` (accessed August 20, 2009).

Brown, Clint. 2009. Bringing your geographic information to life. ESRI International User Conference, San Diego, Calif., July 13–17.

Brown, Clint, Jeremy Bartley, Ismael Chivite, Bernie Szukalski, and Rob Shanks. 2008. ArcGIS in a Web 2.0 World. ESRI International User Conference, San Diego, Calif., August 4–8.

Butler, Declan. 2006. Virtual globe: The Web-wide world. *Nature* 439: 776–78.

Foresman, Timothy W. 2008. Evolution and implementation of the digital Earth vision, technology, and society. *International Journal of Digital Earth* 1 (1): 4–16.

Fu, Pinde, and Paul M. Rich. 2002. A geometric solar radiation model with applications in agriculture and forestry. *Computers and Electronics in Agriculture* 37 (1): 25–35.

Gong, Jianya. 1999. *Contemporary GIS theory and technology.* Wuhan, China: Wuhan University of Surveying and Mapping Science and Technology Press.

Gosling, James. 1996. Java is fusion of several kinds of programming. Fifth International World Wide Web Conference, Paris, May 6–10. http://iw3c2.cs.ust.hk/WWW5/www5conf.inria.fr-webcast/gosling.htm (accessed November 16, 2009).

Knight, Greg. 2009. Boston showcases solar power potential with ESRI's Web GIS. ESRI speaker series podcasts. http://www.esri.com/news/podcasts/audio/speaker/user_knight.mp3 (accessed August 22, 2009).

Krug, Steve. 2000. *Don't make me think! A common-sense approach to Web usability.* Berkeley, Calif.: Pearson Technology Group.

Net Applications. 2009. Browser market share. http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0 (accessed September 17, 2009).

Netcraft. 2009. August Web server survey. http://news.netcraft.com/archives/web_server_survey.html (accessed September 17, 2009).

Noyle, Brian. 2009. Usability and the GeoWeb, Parts 1, 2, and 3. GIS and .NET development blog. http://briannoyle.wordpress.com (accessed November 9, 2009).

Quinn, Sterling. 2008. Design patterns for Web maps. *ArcGIS Server Blog,* September 22. http://blogs.esri.com/Dev/blogs/arcgisserver/archive/2008/08/05/Design-patterns-for-Web-maps.aspx (accessed August 21, 2009).