8 Dynamic Web Applications

In the last few chapters, we covered the inner workings of both Web clients and Web servers. Now we examine the nature of *Web applications*: custom applications that operate within the context of a Web server environment, communicating with other Web applications, servers, and clients.

Our first step is to define the notion of a 'Web application'. We then take a look at the processing flow that occurs in a Web application, making a clear distinction between steps that are the responsibility of the application and steps that are 'taken care of' by the Web server and/or the application framework. We will examine typical application functions, along with the recommended best practices for implementing those functions. Finally, we shall go over design and development issues specifically related to applications that interact with database management systems.

8.1 HISTORICAL PERSPECTIVE

Before we discuss the nature of 'Web applications,' we need to put that term in its proper perspective. In the past, an 'application' was defined as a program (such as Microsoft Word or Adobe Photoshop), an instance of which executes on a single system. That definition changed as the technology evolved.

8.1.1 Client-server applications

Client-server applications are groups of distributed programs running on networked computers, and interacting over known communication protocols. Rather than performing all the processing on a single system and transmitting formatted results to 'dumb' terminals, client-server applications distribute processing between dedicated server and client machines. This architecture was facilitated by the proliferation

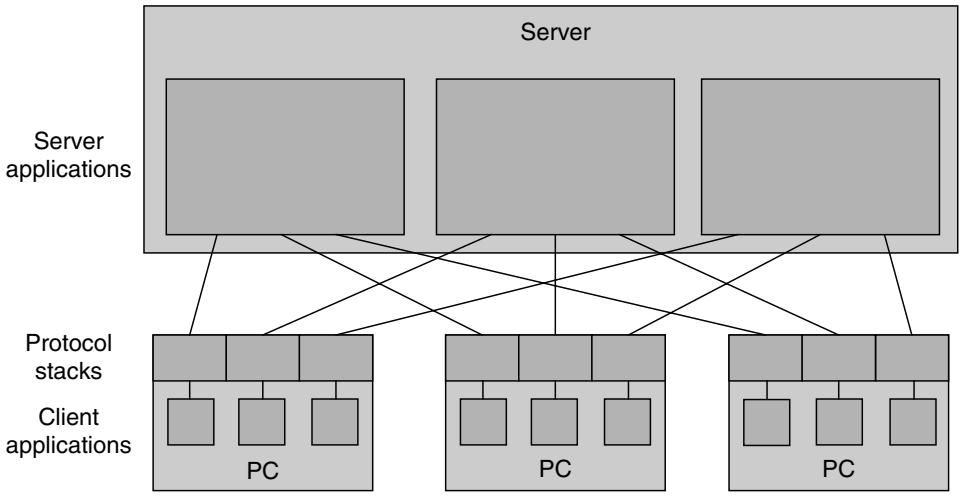

Figure 8.1 Client-server applications that use fat clients

of personal computers, whose additional processing power allowed some of the complex processing to be offloaded from servers down to the clients.

Over time, some proprietary client-server application platforms (e.g. PowerBuilder) grew to be very complex, and their configuration and maintenance became a nightmare. With each new version, the size and complexity of the client platform base seemed to increase by another order of magnitude, resulting in what were referred to as *fat clients*. This application bloat became a serious problem, especially as the number of fat clients installed on a single PC grew as well, as shown in Figure 8.1.

8.1.2 Web applications

A Web application is a client-server application that (generally) uses the Web browser as its client. Browsers send requests to servers, and the servers generate responses and return them to the browsers. They differ from older client-server applications because they make use of a common client program, namely the Web browser, as illustrated in Figure 8.2.

There are important advantages to using Web browsers as clients:

1. Web browsers are ubiquitous. They are present on virtually every desktop and can be used to interact with many different Web applications. There is no need to install several specialized client programs on the desktop, dramatically reducing maintenance headaches.
2. Browsers provide mechanisms to securely download and execute more complex clients (e.g. applets, ActiveX components, and Flash movie players) when additional functionality that browsers alone cannot provide is required

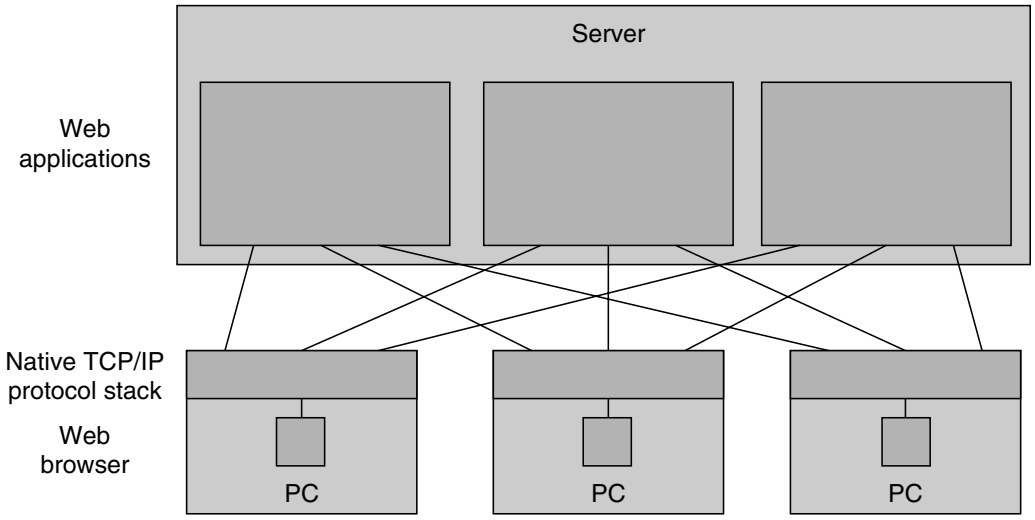

Figure 8.2 Web applications that use a single client application (browser) and a single protocol stack

8.1.3 Multi-tier Web applications

Although Web applications fit into the client-server paradigm, they go beyond it. The interaction between browsers and servers is not all there is to a Web application. The Web servers themselves operate as 'clients' when they interact with other back-end servers, including databases and legacy systems. Thus, the architecture of most Web applications is better described as the *multi-tier* architecture. While simpler client-server applications have only two tiers with distinct roles (the client and the server tiers), multi-tier applications have (as the name implies) multiple tiers, each of which can act as both client and server when communicating with its neighbors.

In multi-tier applications, each tier represents an application layer (browser, web server, application server, database/legacy system, etc.). The bottom layer is the Web client (usually a browser or an intelligent agent) that initiates processing by submitting a request to a Web server. Every set of adjacent tiers represents a pairing of a client and a server, and every intermediate tier may act as either client or server, depending on which of its neighbors it is interacting with. For example, just as the browser connects to the Web server to make a request, the portion of the application executing on the Web server may connect to a business logic or data model layer, acting as a client for that layer's services.

8.2 APPLICATION ARCHITECTURE

Let's take a look at the processing flow associated with a Web application, and the functionality required to perform that processing (Figure 8.3):