Dynamic Web Content
Dynamic Web Development [DWDDCO701]

Thomas Devine
http://noucamp
thomas.devine@lyit.ie

September 15, 2007
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Client/Server communication (static)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Client/Server communication (dynamic)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Three-Tier Architecture Model of Web Database Applications</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>HTTP Request Response</td>
<td>7</td>
</tr>
</tbody>
</table>
1 Document Information

This document has been produced primarily from the books Learning PHP [1] and Web Database Applications with PHP & MySQL [2]. There is no requirement for you to purchase these or any other book. These notes will be sufficient for any assignments and examination assessment for this module.

2 Client/Server Communication

When you sit down at your computer and go to a web page using a browser such as Internet Explorer or Mozilla, you cause a little conversation to happen over the Internet between your computer (client) and another computer (server). This conversation and how it makes a web page appear on your screen may involve a static or a dynamic web page.

2.1 Static Web Pages

Figure 1 illustrates the journey of a static web page from the server to your screen.

Here’s what’s happening in the numbered steps of the Figure 1:

1. You type www.example.com/catalog.html into the location bar of Internet Explorer.

2. Internet Explorer sends a message over the Internet to the computer named www.example.com asking for the /catalog.html page.

3. Apache, a program running on the www.example.com computer, gets the message and reads the catalog.html file from the disk drive.

4. Apache sends the contents of the file back to your computer over the Internet as a response to Internet Explorer’s request.

5. Internet Explorer displays the page on the screen, following the instructions of the HTML tags in the page.
Every time a browser asks for http://www.example.com/catalog.html, the web server sends back the contents of the same catalog.html file. The catalog.html file contains HTML commands. The contents of this file only change if someone edits the file on the server. The catalog.html file is an example of a static web page.

2.2 Dynamic Web Pages

Figure 2 illustrates the journey of a dynamic web page from the server to your screen.

![Figure 2: Client/Server communication (dynamic)](image)

When a dynamic web page is involved, however, the server does more work for its half of the conversation. Figure 2 shows what happens when a web browser asks for a page that is generated by a dynamic scripting language like PHP.

Here’s what’s happening in the numbered steps of the PHP-enabled conversation:

1. You type www.example.com/catalog1/yak.php into the location bar of Internet Explorer.
2. Internet Explorer sends a message over the Internet to the computer named www.example.com asking for the /catalog/yak.php page.
3. Apache, a program running on the www.example.com computer, gets the message and asks the PHP interpreter, another program running on the www.example.com computer, "What does /catalog/yak.php look like?"
4. The PHP interpreter reads the file /catalog/yak.php from the disk drive.
5. The PHP interpreter runs the commands in yak.php, possibly exchanging data with a database program such as MySQL.
6. The PHP interpreter takes the yak.php program output and sends it back to Apache as an answer to "What does /catalog/yak.php look like?"
7. Apache sends the page contents it got from the PHP interpreter back to your computer over the Internet in response to Internet Explorer’s request.
8. Internet Explorer displays the page on the screen, following the instructions of the HTML tags in the page.
3 Three-Tier Architectures

This document describes web database applications built around a three-tier architecture model, shown in Figure 3. At the base of an application is the database tier, consisting of the database management system that manages the database containing the data users create, delete, modify, and query. Built on top of the database tier is the complex middle tier, which contains most of the application logic and communicates data between the other tiers. On top is the client tier, usually web browser software that interacts with the application.

![Three-Tier Architecture Model of Web Database Applications](image)

Figure 3: Three-Tier Architecture Model of Web Database Applications

The formality of describing most web database applications as three-tier architectures hides the reality that the applications must bring together different protocols and software. The majority of the material in this module discusses the middle tier and the application logic that brings together the fundamentally different client and database tiers.

When we use the term “the Web,” we mean three major, distinct standards and the tools based on these standards

- the Hypertext Markup Language (HTML),
- the Hypertext Transfer Protocol (HTTP), and
- the TCP/IP networking protocol suite.

HTML works well for structuring and presenting information using a web browser application. TCP/IP is an effective networking protocol that transfers data between applications over the Internet and has little impact on web database application developers. The problem in building web database applications is interfacing traditional database applications to the Web using HTTP. This is where the complex application logic is needed.
3.1 Hypertext Transfer Protocol

The three-tier architecture provides a conceptual framework for web database applications. The Web itself provides the protocols and network that connect the client and middle tiers of the application; that is, it provides the connection between the web browser and the web server. HTTP is one component that binds together the three-tier architecture. A detailed knowledge of HTTP isn’t necessary to understand the material in this module, but it’s important to understand the problems HTTP presents for web database applications. The HTTP protocol is used by web browsers to request resources from web servers, and for web servers to return responses.

HTTP allows resources to be communicated and shared over the Web. Most web servers and web browsers communicate using HTTP. HTTP communications dominate Internet network traffic.

3.1.1 HTTP example

HTTP is simple. A client web browser sends a request for a resource to a web server, and the web server sends back a response. The HTTP response carries the resource—the HTML document, image, or output of a program—back to the web browser. This simple request-response model is shown in Figure 4.

![Figure 4: HTTP Request Response](image)

An HTTP request is a textual description of a resource and additional header information. Consider the following example request:

```
GET /index.html HTTP/1.0
From: thomas@computer.org (Thomas Devine)
User-agent: mozilla-browser/version-1.0
Accept: text/plain, text/html
```

This example uses a GET method to request an HTML page index.html with HTTP/1.0. In this example, three additional header lines identify the user and the web browser and define what data types can be accepted by the browser. A request is normally made by a web browser.

An HTTP response has a response code and message, additional headers, and usually the resource that has been requested. An example response to the request for index.html is as follows:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1234

<html>
  <body>
    <h1>Welcome to the Web</h1>
  </body>
</html>
```
The first line of the response agrees to use HTTP/1.0 and confirms that the request succeeded by reporting the response code 200 and the message OK; another common response is 404 Not Found. In this example, five lines of additional headers identify the current date and time, the web server software, the data type, the length of the response, and when the resource was last modified. After a blank line, the resource itself follows. In this example the resource is the requested HTML document, index.html.

4 The Client Tier

The client tier in the three-tier architecture model is usually a web browser. Web browser software processes and displays HTML resources, issues HTTP requests for resources, and processes HTTP responses. Web browsers are simple clients that are easily deployed and supported on a wide range of platforms (Windows, MAC, Linux).

There are many browser products available, and each browser product has different features. The two most popular windowing-based browsers are Mozilla and Internet Explorer. Some features of web browsers are:-

- All web browsers are HTTP clients that send requests and display responses from web servers;
- All browsers interpret pages marked up with HTML when rendering a page with headings, images, hypertext links, and so on;
- Most browsers can display images, play movies and sounds;
- Many browsers can run JavaScript that is embedded in HTML pages. JavaScript is used, for example, to validate a form or change how a page is presented based on user actions.
Some web browsers can run components developed in the Java. Most browsers can apply Cascading Style Sheets (CSS) to HTML pages to control the presentation of HTML elements.

While this module isn’t a guide to writing HTML, we discuss HTML features as they are used throughout the module. We introduce JavaScript client-side scripting for validation of data entry and manipulating the web browser later.

5 The Middle Tier

In most three-tier web database systems, the majority of the application logic is in the middle tier. The client tier presents data to and collects data from the user; the database tier stores and retrieves the data. The middle tier serves most of the remaining roles that bring together the other tiers.

In the application framework used in this module, the components of the middle tier are:

- a web server,
- a web scripting language,
- and the scripting language engine.

A web server processes HTTP requests and formulates responses. In the case of web database applications, these requests are often for programs that interact with an underlying database management system. The web server we use throughout this module is the Apache HTTP server, an open source web server.

We use the PHP scripting language as our middle-tier scripting language. PHP is also an open source project and it is the most popular Apache HTTP server add-on module.

5.1 Web Servers

Web servers are often referred to as HTTP servers. The term "HTTP server" is a good summary of their function – their basic task is to listen for HTTP requests on a network, receive HTTP requests made by user agents (usually web browsers), serve the requests, and return HTTP responses that contain the requested resources.

There are essentially two types of request made to a web server – the first asks for a file, often a static HTML web page or an image to be returned, and the second asks for a program to be run and its output to be returned to the user agent.

Requests for web scripts that access a database are examples of HTTP requests that require a server to run a program. With the software used in this module, the HTTP requests are for PHP script resources, which require that the PHP engine be run, a script retrieved and processed, and the script output captured.
5.1.1 Apache HTTP server

Like most users of the Apache HTTP server, we call it Apache. Apache is an open-source web server.

Apache is fast and scalable. It can handle simultaneous requests from user agents and is designed to run under multitasking operating systems, such as Linux and Microsoft Windows. It’s also lightweight, has low process requirements, can effectively handle changes in request loads, and can run fast on even modest hardware.

How Apache listens on the network and serves requests is controlled by its configuration file – httpd.conf. The server administrator controls the behavior of Apache through more than 150 directives that affect resource requirements, response time, flexibility in dealing with request load variability, security, how HTTP requests are handled and logged, and most other aspects of its operation.

5.2 Web Scripting with PHP

PHP has emerged as a component of many medium- and large-scale web database applications. This isn’t to say that other scripting languages don’t have excellent features. However, there are many reasons that make PHP a good choice, including:

- PHP is open source, meaning it is entirely free.
- One or more PHP scripts can be embedded into static HTML files.
- PHP runs on many different platforms.

6 The Database Tier

The database tier is the base of a web database application. Understanding system requirements, choosing database-tier software, designing databases, and building the tier are the first steps in successful web database application development.

In a three-tier architecture application, the database tier manages the data. The data management typically includes storage and retrieval of data, as well as managing updates, allowing simultaneous, or concurrent, access by more than one middle-tier process, providing security, ensuring the integrity of data, and providing support services such as data backup.

Managing relational data in the third-tier requires DBMS software. For most DBMSs, the query language of choice is SQL. In this module, we use the MySQL RDBMS to manage data.

6.1 The MySQL DBMS

MySQL is a medium-scale DBMS, with most of the features of a large-scale system and the ability to manage very large quantities of data. Its design is ideally suited to managing the databases that are typical of many web database applications.
MySQL has a well-deserved reputation for speed, and it is particularly well designed for applications where retrieval of data is more common than updates and where small, simple updates are the general class of modifications. These are characteristics typical of most web database applications. Also, like PHP and Apache, MySQL is open source software.

6.1.1 SQL

SQL is the standard relational database interaction language. Almost all relational database systems, including MySQL, support SQL as the tool to create, manage, secure, and query databases.
References
