

CT216 Software Engineering Tutorial

Eoin O Fiachain

4th/11th November 2004

1 Architecture for CT216 Project

In *Tutorial 1* we looked at the HTTP protocol, HTML and URLs. In *Tutorial 2* we looked at using PHP for server-side scripting. In *Tutorial 3* we examined the MySQL relational database management system.

We will now consider an architecture for the CT216 Software Engineering Project to enable a dynamic database-driven web application.

The chosen architecture combines elements from the three previous tutorials.

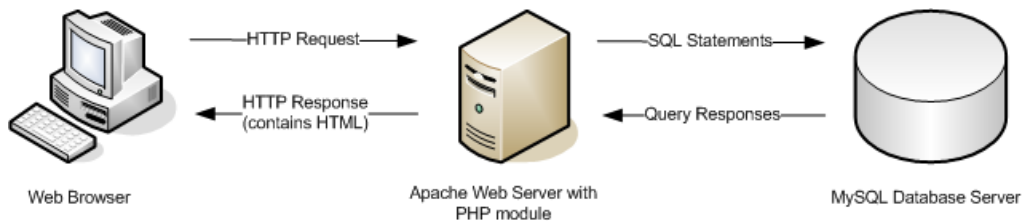


Figure 1: An Architecture for a Dynamic Database-driven Web Application

When a user views a web-page the following steps would typically occur:

1. The web browser uses the HTTP protocol to *request* a particular web-page from the Apache web server. The web-page is identified by a URL.
2. The Apache web server calls the PHP module to dynamically generate HTML content for the web-page.

3. The PHP web-server module processes the PHP script (“source code”) and generates HTML output. The module queries the MySQL database server to retrieve necessary data.
4. The output of the PHP script (containing HTML) is then sent back to the web browser using the HTTP protocol as part of the *response*.
5. The web browser parses the HTML and constructs a display of the web-page from the tags and data within the HTML file.

The *client-server* model occurs in two different contexts in this architecture:

- In HTTP communications between the web browser and the Apache web server, the web browser acts as a *client* and connects to the Apache web *server*.
- When the PHP scripts are executed, the PHP web-server module acts as a *client* and connects to the MySQL database *server* using SQL statements and a native MySQL networking protocol.

1.1 Project Teams

For the CT216 project the class will be divided into different teams who will each work on different aspects of the project e.g. a database group, design group etc.

Each team can concentrate on particular elements of the overall system. However, an understanding of how the system works as a whole is absolutely essential for all.

Groups will need to work closely with each other. Scripts and functions created by one group will need to successfully integrate with the work of others. This will require good project management skills and both effective inter-group and intra-group communication.

2 Accessing MySQL from PHP scripts

An add-on library for MySQL enables a variety of extra functions for communicating with a MySQL database server.

This library is built-in with the PHP package available on *blackboard.nuigalway.ie*. In other cases, it may need to be installed and configured separately.

2.1 Connecting and Disconnecting to the MySQL Server

Before executing any commands on a MySQL server, the user must firstly *connect* to the server. After all commands have been executed the user should then *disconnect* from the server in order to free up resources.

The functions used for this include:

- **mysql_connect** - establishes a connection
- **mysql_error** - returns a description of the most recent error
- **mysql_select_db** - selects a database (similar to the USE command)
- **mysql_close** - closes a connection

The *mysql_connect* function returns a unique identifier representing the new connection, or FALSE if a connection cannot be established.

This unique identifier can be used to associate queries with a particular connection. Most MySQL-related functions have an optional *link_identifier* argument which provides for this. However, if this *link_identifier* argument is omitted then the functions will automatically choose the most recently opened connection.

The following script connects to a database server located at host *localhost* with the username *user* and the password *password*. It selects a database called *films*

The script terminates with an error message if a problem occurs establishing the connection.

```

<?php
$link = mysql_connect('localhost', 'user', 'password');

if (!$link) {
    die('Could not connect: ' . mysql_error());
}

echo 'Connected successfully';

// make films the current db
$db_selected = mysql_select_db('films', $link);

mysql_close($link);
?>

```

2.2 Querying database server

The **mysql_query** function is used to query the database server. It takes a SQL command as a parameter and executes it on the server.

2.2.1 Row Returning Queries

These are queries to the database server which could result in rows returning. These involve SQL commands SELECT, SHOW, EXPLAIN and DESCRIBE.

In this case, the **mysql_query** function normally returns a unique identifier for the query response. The unique identifier can be used to retrieve the contents of the query response using the **mysql_fetch_row** function.

The **mysql_fetch_row** returns an array of values for each row. The first element in the array corresponds to the first column in the row. The second element in the array corresponds to the second column in the row and so on. When there are no more rows to return the function will return FALSE.

The following example retrieves the name and release year of each film in the films database than used in *Tutorial 3*. It assumes that a connection has already been established to a database.

```

# Queries the database server
$result = mysql_query('SELECT name,year FROM film');
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}

# Prints out each row in the query
while ($row = mysql_fetch_row($result))
{
    echo $row[0] . " (" . $row[1] . ")<br/>\n";
}

```

The output of the script is:

```

Big Lebowski, The (1998)<br/>
Reservoir Dogs (1992)<br/>
Pulp Fiction (1994)<br/>
Kill Bill, Volume 1 (2003)<br/>

```

If an error occurs then **mysql_query** returns the value FALSE. As with connection errors, the **mysql_error** function will return a string describing the error that occurred.

2.2.2 Non-Row Returning Queries

These queries involve commands that never return rows, such as UPDATE, DELETE, CREATE TABLE, DROP TABLE etc.

The **mysql_query** function is used similarly for non-row returning queries except that it will never return an identifier. It returns TRUE if no errors are encountered and FALSE otherwise.

```

$sql = "UPDATE film SET name='Fahrenheit 9/11' WHERE id=2";
$result = mysql_query($sql);
if (!$result) {
    echo 'Query Error: ' . mysql_error();
} else {
    echo 'Successful Query';
}

```

2.3 Escaping Strings

When embedding SQL string expressions within PHP string expressions care must be taken to ensure that the correct type of quotes are employed.

PHP string expressions can begin with either double-quotes or single-quotes. SQL string expressions normally employ single-quotes, although in some cases double-quotes can also be used.

If we use double-quotes for a PHP string expression when we must use single-quotes for any SQL string expressions contained within, and vice versa.

Examining the `$sql` variable in the previous example:

```
# Correct
$sql = "UPDATE film SET name='Farenheit 9/11' WHERE id=2";

# Incorrect
$sql = 'UPDATE film SET name='Farenheit 9/11' WHERE id=2';
```

Special characters such as `'` `"` `?` `%` etc. cannot be written into SQL statements directly. They must be “escaped” beforehand, by pre-pending them with a backslash.

The `mysql_escape_string` function will transform a normal string into an “escaped string” that can be used in a SQL query.

2.3.1 Magic Quotes GPC

PHP has a special feature called *Magic Quotes GPC*. This feature automatically escapes user-inputted strings that are submitted via GET, POST and COOKIE operations.

This is so as to avoid certain security problems that can occur when users deliberately insert unwanted characters into an input string in order to gain unauthorised access to a server. These are known as *SQL Injection Attacks* and it is absolutely essential to write web applications that are not vulnerable to them.

The *Magic Quotes GPC* feature can be switched on or off in the `php.ini` file but it is turned on by default.

3 Development Tools for CT216 Project

3.1 Packages available on blackboard.nuigalway.ie

The following packages are available on `blackboard.nuigalway.ie` and are necessary for development in the CT216 project:

- Apache Web Server Version 1.3
- PHP 4.3
- MySQL 4.0 Server and Clients Installer

The installation of these items are detailed in the document “Installing Apache and PHP for CT216 Software Engineering Project”, also available on `blackboard.nuigalway.ie`.

3.2 Eclipse

Although any text-editor (such as Notepad) is sufficient for PHP development there are a number of development environments available that offer extra features such as syntax-highlighting of code.

One common option is to use the *Eclipse* IDE together with the *PHPEclipse* plugin.

These are available at:

- <http://www.eclipse.org/>
- <http://www.phpeclipse.de/>

Eclipse is a freely-available open-source IDE that is typically used for Java development. With the *PHPEclipse* plug-in it can be effectively used for PHP development.

3.3 Source Code Management with CVS

3.4 Source Code Management

In a large programming project, many developers will be working on the same source code files. In order to manage changes to files and to prevent one programmer overwriting another's work, there needs a way of coordinating and integrating the changes each programmer makes.

Source Code Management software provides a means of managing source code and recording a history of all changes made to source code. Typically SCM software provides for the following:

- Locking and concurrency management
- Versioning and revision history
- Synchronization
- Project forking

Commonly used SCM software includes Microsoft Visual Sourcesafe, CVS and Subversion.

3.5 CVS

CVS is a freely-available open-source tool that we will use in the CT216 Project.

It works on the *client-server* model where a CVS server acts as a repository for all source code. Developers use client applications to access the repository.

Each CVS repository is divided into different partitions known as *modules*.

The basic operation of CVS is as follows:

1. A developer *checks-out* a module source-code from the server
2. The developer makes changes to the source code.
3. The developer *commits* these changes back into the repository. Any changes he made to the source files are merged with the current source files in the repository.

As many developers can simultaneously be working on the same source-code files, the locally checked-out version of source code can become out-of-sync with the repository version. Developers can *update* their current checked-out code with the latest code available on the server. This merges any changes any repository-changes with local-changes.

Other features of CVS include:

- Developers can view a history of source files, examining the changes made over time.
- Source code can be *branched* off into different sub-projects, which can be later merged back together again.
- Developers can revert back to earlier versions of source code to trace potential problems.
- Automatic version numbering of source code and applications can be employed.

3.5.1 Code Conflicts

CVS provides for effective source code management for large projects involving many developers. Good communication is necessary between users nonetheless.

For example, problems may occur if two developers each see a problem in the source code and if they both try to simultaneously remedy it. Each then *commits* their changes to the server.

The CVS server will merge both changes into the same source file. If the changes conflict the program may no longer work as intended.

However, these problems are thankfully rare and can be mitigated against with effective *branching* of sub-projects. In any case, such conflicts can generally easily be resolved by examining the source history.

3.5.2 CVS Client Software for CT216 Project

It is desirable for all students to be using the same CVS client software to avoid minor incompatibility issues. A CVS repository has been set up for the CT216 project and further details of this along with appropriate CVS client software will be made available in due course.