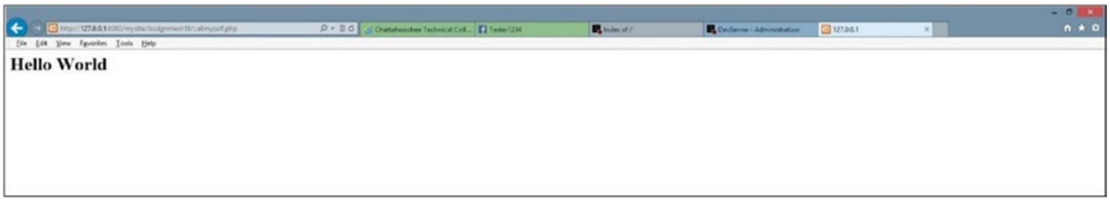

Hello World

Figure 2-11. *callmyself.php after the button is clicked*

The advantage of this technique is that all code can be contained in one file. Thus, all changes occur in one place. The disadvantage of this technique is that all code is in one file. The more complicate the code, the "messier" it can become. One way to clean up the code would be to move the code into functions contained within other PHP libraries (we will look at this later). The additional disadvantage is that the file name cannot be changed without affecting the users. If you changed the file name to `mynewprogram.php`, you would need to inform all your users of the new name (and maybe location). The previous examples using AJAX allowed you to change the file name within the HTML code page, but did not require you to change the actual name of the HTML page that the user would request.

Do It

1. Find the `callmyself.php` file on the book's web site. Download the file to your Apache `projects` folder. Change and add `print` statements to display your complete name, the term, and your major. Test your program. Did your program run successfully? If not, why not?

PHP Three-Tier Architecture

Most of this chapter has been discovering the different platforms (or containers) that can "host" PHP applications. We have found that PHP can display its output in almost any container (PC, Facebook, smart phone/mobile device, or browser). The ease at which PHP can interact with JavaScript, HTML, and CSS has provided this flexibility. Today, almost any platform has the ability to interact with the Internet (and those that don't will sometime in the future). Any platform that can interact with the Internet can also interact with a PHP application.

This independence (or flexibility) of the interface demonstrates a logical separation of the platform or host of the interface from other "tiers" (parts) of the application. This leads into a discussion of the three-tier architecture and the logical design of PHP applications. The larger the application, the more likely the application will need to be broken into modules. Also, it's more likely that these modules could reside on different servers (or web servers). Larger applications likely will require multiple programmers writing code at the same time. These programmers may even use different languages to create the program modules.

Building a large application is not much different than assembling a car. The individual components of the car (body, wheels, electronics, and engine) are assembled individually first. Each completed component is then placed inside the chassis of the car. The components are then connected (hoses, wires, and belts) to other components. When completed, all components of the car work together. If a component breaks, it can be replaced without causing replacement or changes to any other component in the car.

The idea of modular (or component) programming is based on the methodology of blocks of code that can be created individually to be assembled with other modules to produce a working application. The modules can be modified or replaced without requiring changes to other modules. This methodology has been around for a while. Even today, many programs are not modular because smaller programs can work efficiently without being broken into modules. However, as these applications expand into larger