Web Programming with PHP

Web Programming with PHP

- We know that we can use HTML to build websites, but websites built using pure HTML suffer from a serious limitation.

- Imagine we want to create a website that displays the current time in Cambridge, MA, displaying it to the latest minute.

Web Programming with PHP

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:08
    </body>
</html>
```

Web Programming with PHP

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:09
    </body>
</html>
```

Web Programming with PHP

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:10
    </body>
</html>
```

Web Programming with PHP

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:11
    </body>
</html>
```

Web Programming with PHP

- Websites that are pure HTML are completely static. The only way we can update the content of our pages is to manually open up our source files, edit and save, and then the next time the user visits or refreshes the page they'll get the content.

- Incorporating PHP into our code can make our code quite a bit more flexible and introduce a way for our pages to update or be dynamic without requiring our intervention.

Web Programming with PHP

- To run an instance of Apache (a popular web server with the capacity to interpret PHP), simply create a directory where you would like your pages to live and then run the command

```
apache50 start /path/to/dir
```

Web Programming with PHP

- To test that your Apache server is configured correctly, here's a common equivalent of "hello, world" for PHP web programming:

```
<?php phpinfo(); ?>
```

Web Programming with PHP

```
<?php
    date_default_timezone_set('US/Eastern');
    $time = date('H:i:s', time());
?>

<html>
   <head>
      <title>Current Time in Cambridge</title>
   </head>
   <body>
      The current time in Cambridge is <?= $time ?>.
   </body>
</html>
```

Web Programming with PHP

```
<?php
    date_default_timezone_set('US/Eastern');
    $time = date('H:i:s', time());
?>

<html>
   <head>
      <title>Current Time in Cambridge</title>
   </head>
   <body>
      The current time in Cambridge is <?= $time ?>.
   </body>
</html>
```

Web Programming with PHP

- Because the PHP interpreter ignores everything that is not inside of the PHP delimiters, that means that we can intersperse HTML and PHP without any problems, only using PHP for those parts of our website that require dynamism, and letting the static information remain the same.

- Why is it advantageous for us to do this? Instead of putting everything inside of the PHP delimiters and printing out HTML?

Web Programming with PHP

- There's not too much to writing PHP code for the web that's different from writing it for the command line… so far.

- What if we want to pass information **between** files that we have on our PHP server? Or what if we want to allow the user to pass information **into** our web pages?

- PHP supports a number of special *superglobal variables* to accomplish this.

Web Programming with PHP

- **$_GET**

- $_GET is perhaps the simplest of these superglobal variables for passing data between PHP files or allowing the user to supply your PHP page with data.

- With this technique, we scrape data supplied at the URL which is stored in a **query string** consisting of key-value pairs, which are then stored in the $_GET associative array.

Web Programming with PHP

```
<?php

    foreach($_GET as $key => $value)
    {
        print(“<p>{$key}: {$value}</p>”);
    }
?>
```

Web Programming with PHP

- **$_GET**

- Can you envision one possible issue with the $_GET superglobal?

Web Programming with PHP

- **`$_GET`**

- Can you envision one possible issue with the `$_GET` superglobal?

`http://localhost/get3.php?name=Doug&pw=squadgoals`

Web Programming with PHP

- **`$_GET`**

- Can you envision one possible issue with the `$_GET` superglobal?

`http://localhost/get3.php?name=Doug&pw=squadgoals`

Web Programming with PHP

- **$_GET**

- Can you envision one possible issue with the $_GET superglobal?

- $_GET is great for getting information to the page quickly, but can be problematic for passing *sensitive* information, as it makes it very easy for an attacker (or someone less malicious) to easily see that sensitive information.

Web Programming with PHP

- **$_POST**

- $_POST is an alternative method for transmitting data between pages or allowing users to send information that most frequently comes up in the context of HTML forms.
  - Though information can be sent via HTML forms with the GET method, too!

- Unlike $_GET, using this technique submits user information inside of the HTTP headers, instead of via the URL.

Web Programming with PHP

```
<html>
   <head>
      <title>Posting Form</title>
   </head>
   <body>
      <form action="post.php" method="post">
         <input name="name" type="text" />
         <input name="pw" type="password" />
         <input type="submit" />
      </form>
   </body>
</html>
```

Web Programming with PHP

```
<html>
    <head>
        <title>Posting Form</title>
    </head>
    <body>
        <form action="post.php" method="post">
            <input name="name" type="text" />
            <input name="pw" type="password" />
            <input type="submit" />
        </form>
    </body>
</html>
```

Web Programming with PHP

```
<html>
   <head>
      <title>Posting Form</title>
   </head>
   <body>
      <form action="post.php" method="post">
         <input name="name" type="text" />
         <input name="pw" type="password" />
         <input type="submit" />
      </form>
   </body>
</html>
```

Web Programming with PHP

```
<html>
    <head>
        <title>Posting Form</title>
    </head>
    <body>
        <form action="post.php" method="post">
            <input name="name" type="text" />
            <input name="pw" type="password" />
            <input type="submit" />
        </form>
    </body>
</html>
```

Web Programming with PHP

```
<?php
    echo(“<p>” . $_POST[“name”] . “</p>”);
    echo(“<p>” . $_POST[“pw”] . “</p>”);
?>
```

Web Programming with PHP

```
<?php
    echo(“<p>” . $_POST[“name”] . “</p>”);
    echo(“<p>” . $_POST[“pw”] . “</p>”);
?>
```

Web Programming with PHP

- **At-home exercise**

- Create a pair of web pages:
  - The first, a simple form with a single field and a submit button.
  - The second, which is expecting input from the form via the POST method and based on the input creates an *n*-by-*n* multiplication, where *n* is a number the user submits through the form.

- No need to do any error checking. You can assume the user is not malicious and will always input a reasonably-small value.

Web Programming with PHP

- **`$_SESSION`**

- `$_SESSION` is one additional superglobal that PHP provides to store data that you'd like to be available across multiple pages so long as a user is logged in to your website, so that you don't have to repeatedly pass information via `$_POST`.

- After you create a session with the PHP function `session_start()`, you can use and modify `$_SESSION` across your site and its contents will not be lost until you `session_destroy()`.

Web Programming with PHP

- Writing fairly dynamic websites with PHP drastically improves your site's quality without requiring too much additional complexity.

- Practice, practice, practice! You already know the basics from your experience with C, now you're just applying those basics with a slightly different language and in a slightly different context.

- Don't be afraid to look things up on the internet! `php.net`, for example, is a fantastic resource for PHP's documentation!