Class 07: Building a CMS with PHP and MySQL, and Website Security

Template-driven websites
How can PHP pass values from one page to another?
URL parameters
Multiple parameters
$GET
Data validation
Sanitizing data
How websites are attacked
SQL injection
Hacking is easy
What hackers can do
Precautions (how to protect yourself)
What to do if you are hacked

References

PHP and MySQL for Dynamic Web Sites 5th Ed. by Larry Ullman
Head First PHP & MySQL by Lynn Beighley and Michael Morrison
PHP & MySQL by Jon Duckett
PHP Solutions: Dynamic Web Design Made Easy 2nd Ed. by David Powers

Homework

Read: Chapter 10 of PHP and MySQL for Dynamic Web Sites
OR Chapter 5 of Head First PHP & MySQL
Prisca's excellent overview of Content Management Systems
9 security tips to protect your website from hackers

The "takeaway" from this week's session is that you should be aware of website security issues, take precautions to avoid problems and always filter and/or sanitize any data that originates from outside a PHP script before using it. A simple test will do the job:

```
if (isset($_GET['id']) && filter_var($_GET['id'],
FILTER_VALIDATE_INT)) {
	$article=$_GET['id'];
}else{
	header('HTTP/1.0 404 Not Found');
	exit("<h1>Not Found</h1>\n<p>The submitted data is not
valid.</p>");
}
```

The code above tests to see if a value exists and if that value is an integer.

Take a look at the Articles website files and make sure you understand how it works.

Course materials: Content Management

Prepare for the Major Project firmness crit – think carefully about using a CMS.

A note on evaluating PHP variables within HTML code.

Once you have the data you need from a database field, you will often want to use the value of the variable in your HTML code. There are several ways you can do this. Here are some examples.

Say we want to print a heading for an article. The heading level is h2 and the string is stored in a variable called $headline. Any one of the following will achieve the outcome we want.

```
<?php
echo "<h2>$headline</h2>";
?>
```

This option is useful if you have several lines of HTML with embedded variables – you can echo all your HTML using a single echo statement. For example:

```
<?php
echo "<h2>$headline</h2>
<p>An introductory paragraph for the article</p>
$markup
<footer>Copyright $now</footer>";
?>
```

```
<h2><?php echo $headline; ?></h2>
```

This option is useful if you just want to use the value of a variable within your normal HTML code, however...

```
<h2><?=$headline?></h2>
```

This option is a shorthand version of the one above. The equals symbol implies that this is a PHP echo statement. It is referred to as the *short echo tag*.

See PHP tags in the PHP Manual and/or the short YouTube video by Dave Hollingworth for more information.