CM0133 Internet Computing

State and Content Management

Objectives

- State Management & Privacy Issues
  - Urls
  - Hidden Variables
  - Cookies
  - Sessions
- PHP open topics
  - Email
  - Date and Time
  - Code Reuse
- PHP Objects
- Content Management

Authentication vs. Authorization

- Authentication is the process of finding out WHO a user is. It is a mechanism whereby the web site may securely identify their users. Authentication systems provide an answers to the questions:
  - Who is the user?
  - Is the user really who he/she represents himself to be?
- Authorization is the process of finding out WHAT RIGHTS a user has. What level of access a particular authenticated user should have to secured resources controlled by the system.
  - Is user X authorized to access resource R?
  - Is user X authorized to perform operation P?
  - Is user X authorized to perform operation P on resource R?

State Management

- Browsers and web servers interact stateless with each other.
- HTTP is a stateless protocol
- Each HTTP request sent to a web server is independent of any other request. This allows users to browse the Web by following hypertext links and visiting pages in any order.
- HTTP allows applications to distribute or even replicate content across multiple servers to balance the load generated by a high number of requests.
- But ...

State Management

- Applications that require complex user interaction can't be implemented as a series of unrelated, stateless web pages. Think of ...
  - Shopping cart
  - Bank account
  - Library account
- There are two ways to build an application that keeps state:
  - The state can be stored in the browser and included within each request (hidden variables)
  - The state can be stored in variables stored on the server.

Passing State through URL

- Scenario: User logs in on login page. How do we maintain the user's identity on other pages?
- Solution: Pass username as URL string or form parameter

```
 <a href="link.html?userID=<?php echo
$_REQUEST['userID']; ?>">Link text</a>
```

Note that this needs to be added to all links

Using a Form

- Values can be passed as a hidden input parameter

```
<form action="link.php" method="post">
<input type="hidden" name="userID"
value="<?php echo $_REQUEST["userID"];?>"/>
…
<input type="submit" value="Next page"/>
</form>
```

What value to use ?

- What should we persist to be sure that a user is logged in?
  - Username: but can be tampered with, copied
  - Password: shouldn't be passed around
- Any value passed in the URL string can be tampered with by the user
- Login can create a token that is difficult to fake
  - Random ID generated by server
  - Store in database and verify each page
  - Use a timestamp, so we can disallow sessions after a set period of time

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Using URL parameters

- Advantages
  - Works everywhere
  - Technically simple
  - Can be accessed client-side [Javascript] and server-side [PHP]
- Disadvantages
  - Can be viewed or altered by user
  - Must rewrite all links or use forms
  - Lost if user closes browser window

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Cookies

- A cookie is a piece of text that is stored by the web browser between sessions
  - Cookies are set in the browser by a specific web server
  - Browser passes information back to that server only
- Commonly used to store user information
  - Or to tag individual browsers and identify them later
- Consists of four components
  - Source site
  - Attribute name
  - Attribute value
  - Expiration date (by default, when browser closes)

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

What cookies can do

- Store session data
  - Login info, pages visited, shopping cart
- Track users who visit your website
  - Assign them an ID, read it later
  - Even without logging in
- Allow users to personalize their experience
  - Set site settings, store in browser

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Cookies

- A cookie is a text string stored on the client machine by your script (to track users and manage transactions)
- Cookies are automatically returned (by the client), and can be accessed using a variable of the same name
- The following script sets a cookie with the variable name **`uname`** and the value **`$name`**. The cookie will expire after 20 minutes

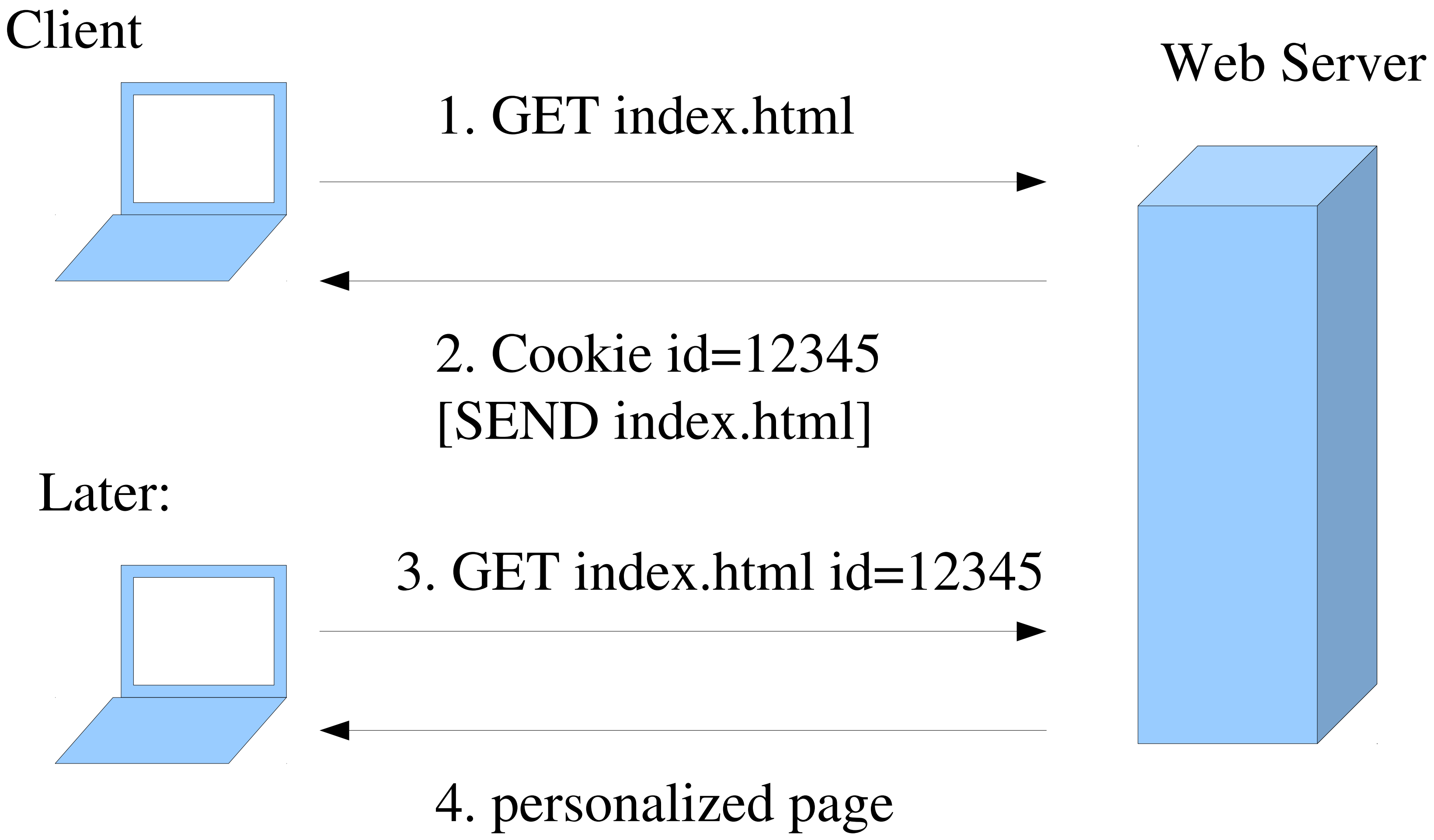
```
<?php setcookie("uname", $name, time()+1200);?>
<html>
      <body>
            <p>A cookie was set on this page!</p>
      </body>
</html>
```

Cookies

- The Cookie can now be used to monitor users on following pages.
- For example, a proceeding page may only be accessible if a cookie has been sent.
- You could set a cookie after e.g. a successful log-in to a secure site

```
<html>
<body>
<?php
 if (isset($_COOKIE["uname"]))
   print "Welcome " . $_COOKIE["uname"] . "!<br/>";
 else
   print "You are not logged in!<br />";
?>
</body>
</html>
```

How cookies work



http://faculty.washington.edu/loter/info344/slides/lect9.pdf

The messy details

- Server at domain.com sends cookie in HTTP response

```
HTTP/1.0 200
Content-Length: 1276
Content-Type: text/html
Date: Tue, 06 Nov 2001 04:12:49 GMT
Expires: Tue, 06 Nov 2001 04:12:59 GMT
Set-Cookie: id=12345
<html>...</html>
```

- Browser stores cookie on local system
- When browser revisits domain, passes cookie back

```
GET /index.php HTTP/1.0
Connection: Keep-Alive
Cookie: id=12345
Host: www.test.com
Referer: http://www.test.com/
```

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Using Cookies

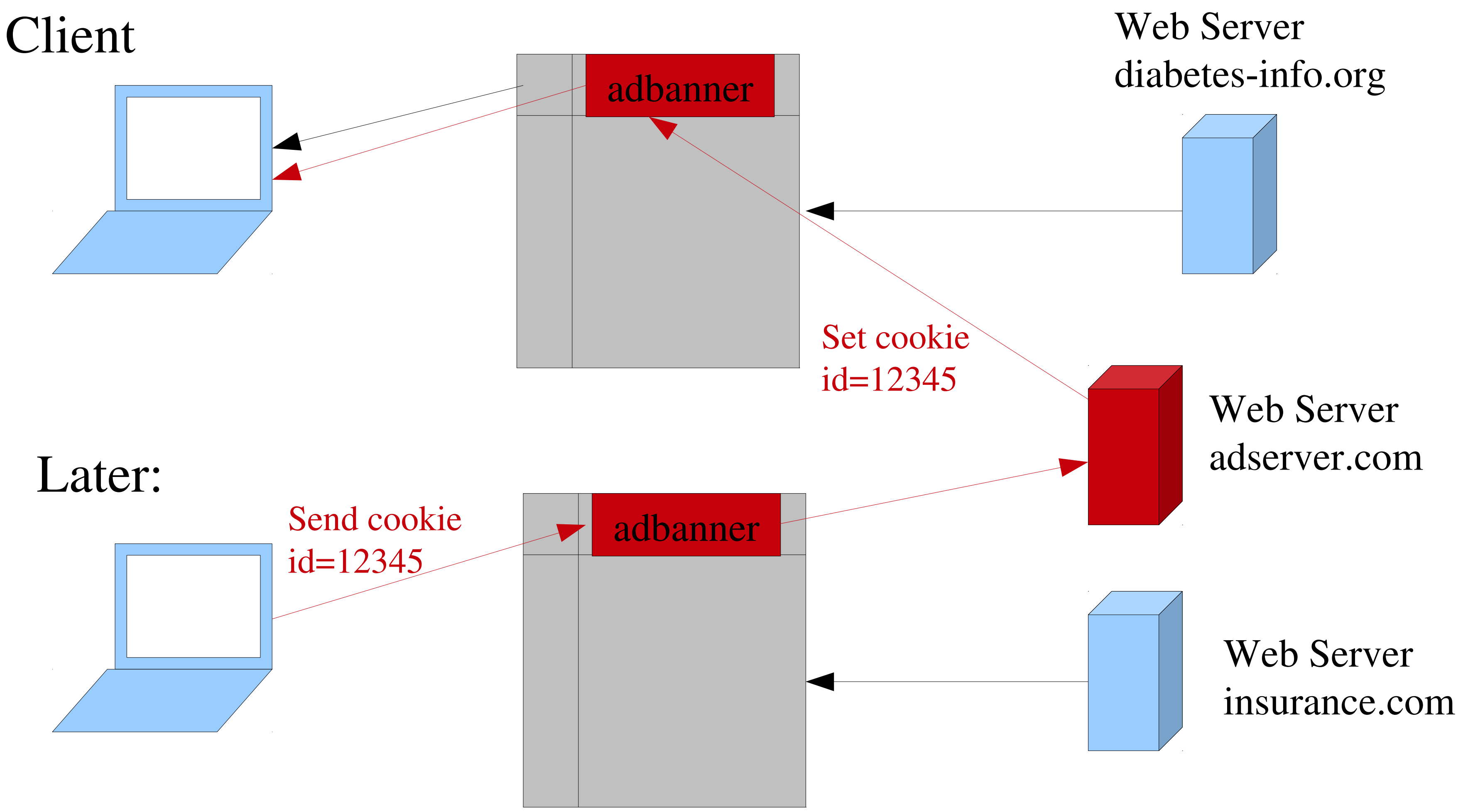
- Advantages
  - Easy
  - Can be used client- and server- side
  - A knowledgeable user has control over what is stored
  - Persistent across browser sessions
- Disadvantages
  - Can only store text
  - Cookies may only be up to a certain size [~4k]
  - May only be read by the site that set them
  - Some users turn cookies off in their browsers
  - Private browsing, e.g. Firefox 3.5
  - Users may not realize cookies are being set

Third Party Cookies

- In general, cookies may only be read by the website that sets them
- However, sometimes one web server provides content for multiple sites
  - Ad banners from advertising agencies
- These third-party (or "tracker") cookies can trace user through several sites

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Third Party Cookie Example



adserver has seen the user at diabetes-info.org and insurance.com!

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Session Management with PHP

- PHP provides an abstraction layer for managing browser sessions
- Can attach PHP variables (including simple objects) to a user's session
  - Can retrieve variables on any page
  - Variables last until browser closed
- Underneath the hood
  - PHP creates a random ID
  - Sends ID to user as cookie
  - PHP stores ID and variables in server-side text

http://faculty.washington.edu/loter/info344/slides/lect9.pdf

Sessions

- Sessions are similar to Cookies in many respects
- Used to track user activity and personalise interactions
- However:
  - Cookies are becoming unreliable
  - They store information on client without permission
  - Modern browser privacy/security settings can block cookies

- PHP Session variables store data on the server
- Connected to clients browser via the server and a Session ID
- Almost flawless in operation and invisible to the user

Sessions

- Setting Session variables is simple
- Imagine we have received `$name` via a HTML form
- We can store this information for use in other pages
- It is essential that if we are using sessions, the first thing we do is call:

  `session_start()`

```
<?php
     session_start();
     extract($_POST);
     $_SESSION[‘name’] = $name;
?>
```

Sessions

- Now, on another page we can see if a session exists
- If one does then we can welcome the visitor

```
<?php
     session_start();
     if($_SESSION['name']){
     print "Hi".$_SESSION['name'].". A session is registered";
     }else{
       print "There is no session registered…";
     }
?>
```

- The condition for the 'if' statement is true if the session variable **`name`** exists.
- If it isn't then we can take another course of action.

Sessions

- Sessions end when a user closes a browser.
- We can also terminate sessions to facilitate a logout by a user.
- Note that even though we are destroying this session, we still have to call `session_start()` first.

```
<?php
      session_start();
      session_destroy();
?>
```

Redirecting a Browser

- We can use the `header()` function to redirect a browser to a different page.
- For example, to redirect a browser to a page called `login.php` we would use

```
header('Location: login.php');
```

- This function is useful for returning a user to a login page if e.g. they have entered an incorrect password, or an appropriate session or cookie is not set

Email

- The **`mail`** command allows your scripts to send email

```
<?php
  $to = "tom@cs.cardiff.ac.uk";
  $subject = "PHP is Great";
  $body = "Hello how are you goodbye";
  $headers = "From: bob@cs.cardiff.ac.uk\n";

  $sent = mail($to, $subject, $body, $headers);
  if($sent)
    print "Mail sent to $to";
  else
    print "Error sending mail";
?>
```

Time in PHP

- Several library functions are available.
- Most of them generate a Unix timestamp or format a Unix timestamp in a human readable form
- A Unix timestamp consists of the number of seconds since the arbitrarily chosen time 1 January 1970 00:00:00.
- Most systems represent a timestamp as a 32-bit integer, allowing a range of dates from December 13, 1901 through January 19, 2038.
- Care must be taken when manipulating timestamps to avoid integer overflow errors.

Current Time and Date

integer time() returns the timestamp for the current date and time:

```
print time(); // e.g. 1064699133
```

To create a timestamp for a specific date between 13th Dec 1901 and 19th Jan 2038:

```
print mktime(11,34,0,2,26,2010);
// 11:34 A.M., 26th Feb 2010
```

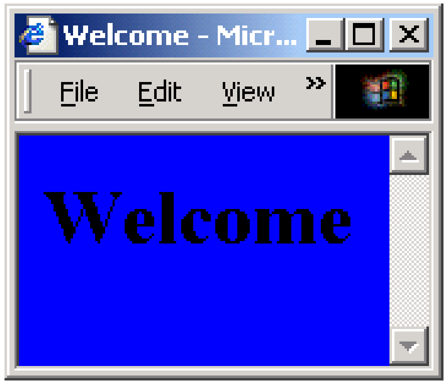
Various manipulations possible, e.g.:

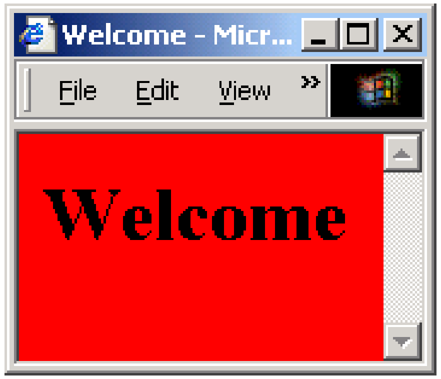
```
$coursework_timespan = 21; // days
$deadline = mktime(11,34,0,2,26+
$coursework_timespan,2010);
```

Dates and times

- Many functions available – see http://php.net
- The following sets the background colour to blue on Tuesdays

```
<?php
  if(date("D") == "Tue") $colour = "blue";
  else $colour = "red";
?>
<html>
 <head>
  <title>Welcome</title>
 </head>
 <body bgcolor = <?php print($colour) ?>>
   <h1>Welcome</h1>
 </body>
</html>
```





Code Reuse - Include() function

```
<html>
<body>


<div class="leftmenu">
<?php include("menu.php"); ?>
</div>


<h1>Welcome to my home page.</h1>
<p>Some text.</p>


</body>
</html>
```

menu.php

```
<a href="/default.php">Home</a>
<a href="/tut.php">Tutorials</a>
<a href="/ref.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/con.php">Contact Us</a>
```

Alternative is to use require() that utilizes a different error handling.

PHP and Object Orientation

- At first PHP wasn't an object oriented language (OO).
- Slowly features have been added
  - Classes
  - Constructors
  - Destructors
- PHP is still a procedural language and not comparable to OO languages like Java !

PHP Classes

- The class keyword defines a class, just as function defines a function.
- Properties are declared using the var keyword.
- Method declaration is identical to how functions are defined.

```
class guest_book {
        var $comments;
        var $last_visitor;

        function update($comment,
                        $visitor)
        {
                …
        }

}

$gb = new guest_book ;
```

Declaring property values

- You can only assign constant values to a property !
- If you try to assign something else PHP will die with an parse error.

```
var $last_visitor = 'Don'; // OK

var $last_visitor = 9; // OK

var $last_visitor = array('test'); // OK

var $last_visitor = pick_visitor(); // BAD !

var $last_visitor = 'Chris' . '9'; // BAD !
```

Assigning non constant values

- We need a method to assign a non constant value to a property.
- The function update adds a visitor comment to the top of the array of comments and sets that person as the latest visitor to the guest book.
- **$this** refers to the current object, to access e.g. the size property of an object use **$this->size**

```
var $last_visitor;

function update($comment, $visitor) {

   if(!empty($comment)) {

      array_unshift($this->comments,$comment);

      $this->last_visitor = $visitor;

   }

}
```

Class guest_book

Assigning non constant values at instantiation

```
class guest_book {
   var $comments;
   var $last_visitor;

   function guest_book($user) {
         $dbh = mysql_connect('localhost','username','password');
         $db= mysql_select_db('sites');
         $user = mysql_real_escape_string($user);
         $sql = “SELECT comments,last_visitor FROM guest_books WHERE user='$user';”;
         $r = mysql_query($sql);

         if($obj = mysql_fetch_object($r)) {
                    $this->comments = $obj->comments;
                   $this->last_visitor = $obj->last_visitor;
         }
    }
}
$gb = new guest_book('Stuart');   // constructor call
```

Explanations to class guest_book

- Note the use of the constructor guest_book().
- Watch out it is $this->comments rather than $this->$comments.
- Besides using -> to access a member variable you can also use :: This syntax can access static methods in a class.
- These methods are identical for every instance of a class, because they can't rely on instance specific data.

```
class convert {

// Celsius to Fahrenheit
   function c2f($degrees) {
          return (1.8 * $degrees) + 32;
   }
}
$f = convert::c2f(100/32); // 212
```

Instantiating Objects

- Use the new keyword:

```
class user {
   function load_info($username) {
       // load profile from DB
   }

   $user = new user;
   $user->load_info($_REQUEST['username']);
```

- You can instantiate multiple instances of the same object:

```
$adam = new user;
$adam->load_info('adam');
$dave = new user;
$dave->load_info('dave');
```

Object Constructors

- Define a method that is called when an object is instantiated E.g. automatically load information from a database into an object when it is created.

```
class user {
    function user($name,$passwd) {
        ...
    }
}
```

- A constructor is a function that has the same name as its class. This was not always supported in PHP. You might come across pseudo-constructors:

```
class user {
        init($name,$passwd) {
        ...
    }
}
$user = new user();
$user->init($username,$passwd);
```

Destroying and Cloning

- **Destroy an object**

```
$car = new car ;
…
unset($car) ;
```

- **Clone an object**

```
$rabbit = new rabbit;
$rabbit->eat();
$rabbit->hop();
$baby= $rabbit;    //  all you need for a clone is assigning
                   //  it to a new variable
```

Calling Methods on an Object Returned by Another Method

You need to call a method on an object returned by another method:

```
$orange = $fruit->get('citrus');

$orange->peel();
```

This is necessary because $fruit->get('citrus')->peel() is not properly dereferenced by the PHP interpreter. Only if you use Zend Engine 2, which supports direct dereferencing this problem is resolved.

Inheritance

To implement inheritance use the extends keyword in the class definition:

```
class DB {
   var $result;

   function getResult() {
      Return $this->result;
   }

   function query($sql) {
     error_log(“query() must be overriden by a DB-specific child”);
     return false;
   }
}

class MySQL extends DB {
   function query($sql) {
      $this->result = mysql_query($sql);
   }
}
```

Inheritance cont.

Parent:: allows you to explicitly call the parent method:

```
function escape($sql) {
  // escape special characters
  $safe_sql = mysql_real_escape_string($sql)
  // parent method adds '' around $sql
  $safe_sql = parent::escape($safe_sql);
  return $safe_sql;
}
```

Overriding Methods – Example II

```
class shape {
        function draw() {
                // write to screen
        }
}

class circle extends shape {

        function draw($origin,$radius) {
                // validate data
                if ($radius > 0) {
                        parent::draw();
                        return true;
                }

                return false;
        }
}
```

When you override a parent method by defining one in the child it isn't called unless you explicitly reference it.

Overriding Methods – Example II

- Only code inside the class can use parent:: , calling parent::draw from outside the class results in a parse error.
- THIS DOES NOT WORK:

```
$circle = new circle;
if($circle->draw($origin,$radius)) {
      $circle->parent::draw();
}
```

:-(

Further OO features

- Property overloading is supported as an experimental extension.
  - __get and __set methods are available to intercept property requests.
- Property overloading allows you to seamlessly obscure from the user the actual location of your object's properties and the data structure you use to store them (Encapsulation !).
- Method Polymorphism is NOT supported as a built-in feature. You could emulate it using various type checking functions (for details see Sklar and Trachtenberg (2002): PHP Cookbook)

Summary PHP

- Data Types
- Control Constructs
- Functions
- Objects and Classes
- Regular Expressions & String Manipulation
- File and Database Access
- State Management

These are powerful means to build dynamic web applications, but we don't have to build everything from scratch readymade frameworks are available → CMS.

Content Management Systems

A content management system (CMS) is a collection of procedures used to manage work flow in a collaborative environment. These procedures can be manual or computer-based. The procedures are designed to:

- Allow for a large number of people to contribute to and share stored data
- Control access to data, based on user roles. User roles define what information each user can view or edit
- Aid in easy storage and retrieval of data
- Reduce repetitive duplicate input
- Improve the ease of report writing
- Improve communication between users

http://en.wikipedia.org/wiki/Content_management_system

Examples – a random list

- Drupal - http://drupal.org/
- Joomla! Content Management - http://www.joomla.org/
- OpenCMS - http://www.opencms.org/en/
- CMSimple Content Management - http://www.cmsimple.org/
- TYPO3 Open Source Content Management System - http://typo3.com/
- and many more – search for “CMS PHP”