

Web Programming

YJ – Aug 2015

Course Outline

There're 16 lectures (2.5 hrs each) in this course. We'll mainly focus on web programming using LAMP model.

During the 24 hours tutorials in each week, our tutors will help you go through all the contents in the lecture notes and help you with the assignments

All scheduled lectures are learner-paced which means they could be altered depends on the situation. We help you gain rather than cram!

Course Outline

- ❖ L1 ~ Web programming intro
- ❖ L2 ~ HTML5
- ❖ L3 ~ CSS3
- ❖ L4 ~ JavaScripts
- ❖ L5 ~ PHP 1 Basic
- ❖ L6 ~ PHP 2 OO && Functions
- ❖ L7 ~ PHP 3 OO && Sessions
- ❖ L8 ~ PHP 4 Advanced features
- ❖ L9 ~ Mysql
- ❖ L10 ~ Mysql 2
- ❖ L11 ~ XML/JSON
- ❖ L12 ~ Ajax
- ❖ L13 ~ Framework: Bootstrap && Laravel
- ❖ L14 ~ CMS: Wordpress, Joomla, etc.
- ❖ L15 ~ Final Projects
- ❖ L16 ~ Final Projects

Object-Oriented Cont.

Accessibility

Accessibility controls how and from where properties and methods can be accessed. There are three visibility keywords: public, protected, and private.

❖ Public

- > All the methods and properties you've used so far have been public. This means that they can be accessed anywhere, both within the class and externally.

❖ Protected

- > When a property or method is declared protected, **it can only be accessed within the class itself or in descendant classes** (classes that extend the class containing the protected method).

❖ Private

- > A property or method declared private is accessible **only from within the class that defines it**. This means that *even if a new class extends the class that defines a private property*, that property or method will not be available at all within the child class.

Accessibility

```
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj = new MyClass();
echo $obj->public; // Works
echo $obj->protected; // Fatal Error
echo $obj->private; // Fatal Error
$obj->printHello(); // Shows Public, Protected
                    and Private
```

```
class MyClass2 extends MyClass
{
    // We can redeclare the public and protected method, but not private
    protected $protected = 'Protected2';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj2 = new MyClass2();
echo $obj2->public; // Works
echo $obj2->protected; // Fatal Error
echo $obj2->private; // Undefined
$obj2->printHello(); // Shows Public, Protected2, Undefined
```

Accessibility

```
class MyClass
{
    // Declare a public constructor
    public function __construct() { }

    // Declare a public method
    public function MyPublic() { }

    // Declare a protected method
    protected function MyProtected() { }

    // Declare a private method
    private function MyPrivate() { }

    // This is public
    function Foo()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}

$class = new MyClass;
$class->MyPublic(); // Works
$class->MyProtected(); // Fatal Error
$class->MyPrivate(); // Fatal Error
$class->
>Foo(); // Public, Protected and Private work
```

```
class MyClass2 extends MyClass
{
    // This is public
    function Foo2()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate(); // Fatal Error
    }
}

$class2 = new MyClass2;
$class2->MyPublic(); // Works
$class2->
>Foo2(); // Public and Protected work,
not Private
```

Accessibility

final keyword, which prevents child classes from overriding a method by prefixing the definition with *final*. If the class itself is being defined final then it cannot be extended.

```
<?php
class BaseClass {
    public function test() {
        echo "BaseClass::test() called\n";
    }

    final public function moreTesting() {
        echo "BaseClass::moreTesting() called\n";
    }
}

class ChildClass extends BaseClass {
    public function moreTesting() {
        echo "ChildClass::moreTesting() called\n";
    }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

Static

Declaring class members or methods as static makes them accessible without needing an instantiation of the class. A member declared as static can not be accessed with an instantiated class object (though a static method can).

```
public static function plusOne()  
{  
    return "The count is " .  
    ++self::$count . "<br />";  
}  
  
echo MyClass::plusOne();
```

```
class Foo  
{  
    public static $my_static = 'foo';  
  
    public function staticValue() {  
        return self::$my_static;  
    }  
}  
  
print Foo::$my_static . "\n";
```


Interface

Interfaces are defined to provide a common function names to the implementors.

Different implementors can implement those interfaces according to their requirements. You can say, interfaces are skeletons which are implemented by developers.

```
interface Mail {  
    public function sendMail();  
}  
  
class Report implements Mail {  
    // sendMail() Definition goes here  
}
```

Abstract Class

An abstract class is one that cannot be instantiated, only inherited. You declare an abstract class with the keyword **abstract**, like this:

When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same accessibility.

Abstract vs Interface

- ❖ Abstract Classes focus on a kind of things similarity.

It is the notion to extend from something, and optionally add some new feature or override some existing feature (to do differently). But using inheritance, you share a big part of code with the parent. **You are** a parent + some other things.

People are considered of type mammal and as such would not be considered of type vehicle.

- ❖ Interfaces focus on collation of similar function.

It is representing some abilities (we says a class is *implementing* an interface to says that it has these abilities). An interface can be implemented by 2 classes which are completely different and do not share their code (except for methods they implements).

For example: You are a human being and are of type mammal. If you want to fly then you will need to implement a flying Interface. If you want to shoot while flying, then you also need to implement the gun Interface.

Maintaining State

- ❖ HTTP is a Stateless Protocol

HTTP has no built-in way of carrying state from page to page. This means that when a user clicks a link in page 1 and moves on to page 2, we have no way of knowing if it is the same user

- ❖ We would like to be able to track a user through a single session when using a web site. This can enable us to:

- > Log a user in and out
- > Track their behaviour
- > Build useful and secure applications more easily

- ❖ 3 different ways:

- > Passing Hidden POST Variables
- > Cookies
- > Sessions (built into PHP language)

Passing Hidden POST Variables

You may have used this approach previously. Replace each link with a form button with action pointing to the next page and use a hidden variable.

```
<input type="hidden" name="username" value="joe" />
```

```
<?php
```

```
    $number_of_visits++;
```

```
?>
```

```
<input type="hidden" name="number_of_visits"  
    value="<?php echo number_of_visits?>" />
```

Cookies

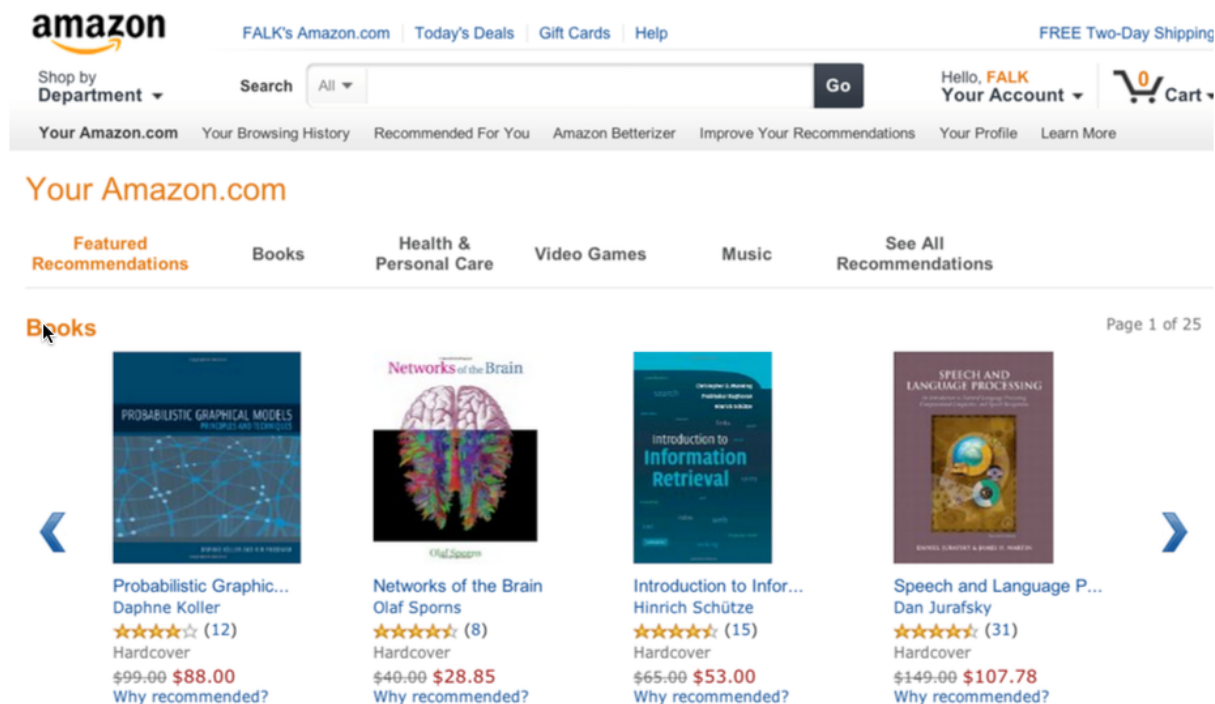
Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

- ❖ Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- ❖ Browser stores this information on local machine for future use.
- ❖ When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Cookies

- ❖ When you visit certain websites (e.g. amazon.com), they can look at a unique identifier that was previously stored in a cookie on your machine.
- ❖ The website can then recognize you without you needing to re-enter a username.
- ❖ Sometimes this is convenient.



The screenshot shows the Amazon homepage with the 'Your Amazon.com' section. The navigation bar includes the Amazon logo, links to 'FALK's Amazon.com', 'Today's Deals', 'Gift Cards', and 'Help', and a 'FREE Two-Day Shipping' badge. The search bar is set to 'All' with a 'Go' button. The user is logged in as 'FALK' with a 'Hello, FALK Your Account' dropdown and a shopping cart icon showing 0 items. Below the navigation bar, the 'Your Amazon.com' section features a 'Featured Recommendations' header and a row of category links: 'Books', 'Health & Personal Care', 'Video Games', 'Music', and 'See All Recommendations'. The 'Books' category is selected, displaying a list of four books. The first book is 'Probabilistic Graphical Models: Principles and Techniques' by Daphne Koller, priced at \$88.00. The second is 'Networks of the Brain' by Olaf Sporns, priced at \$28.85. The third is 'Introduction to Information Retrieval' by Hinrich Schütze, priced at \$53.00. The fourth is 'Speech and Language Processing' by Dan Jurafsky, priced at \$107.78. Each book listing includes a star rating, the number of reviews, the format (Hardcover), the original price, the current price, and a 'Why recommended?' link. Navigation arrows are visible on the left and right sides of the book list.

amazon

FALK's Amazon.com | Today's Deals | Gift Cards | Help

FREE Two-Day Shipping

Shop by Department ▼ Search All ▼ Go

Hello, FALK Your Account ▼ Cart 0

Your Amazon.com Your Browsing History Recommended For You Amazon Betterizer Improve Your Recommendations Your Profile Learn More

Your Amazon.com

Featured Recommendations

Books Health & Personal Care Video Games Music See All Recommendations

Books

Page 1 of 25

Probabilistic Graphical Models: Principles and Techniques

Daphne Koller

★★★★☆ (12)

Hardcover

\$99.00 **\$88.00**

Why recommended?

Networks of the Brain

Olaf Sporns

★★★★☆ (8)

Hardcover

\$40.00 **\$28.85**

Why recommended?

Introduction to Information Retrieval

Hinrich Schütze

★★★★☆ (15)

Hardcover

\$65.00 **\$53.00**

Why recommended?

Speech and Language Processing

Dan Jurafsky

★★★★☆ (31)

Hardcover

\$149.00 **\$107.78**

Why recommended?

Cookies

There are a number of fields in a cookie:

- ❖ **Name**: what the data is called (a variable name)
- ❖ **Expires**: the date when cookie is no longer relevant
- ❖ **Domain**: the server that generated the cookie
- ❖ **Path**: used to specify a subset of URLs on the site where the cookie is relevant
- ❖ **Secure**: if this is set, the cookie will only be transmitted over HTTPS (encrypted HTTP - when you see the little padlock in the corner of your browser)

Cookies

Setting Cookies

`setcookie(name, value, expire, path, domain, security);`

`setcookie("name", "", time()- 60, "/", "", 0);`

Access Cookies

`echo $_COOKIE["name"]. "
";`

Delete Cookies

`setcookie("name", "", time()- 60, "/", "", 0);`

Cookies

Uses of Cookies

- ❖ Cookies are a flexible way of maintaining state or other information for a user.
- ❖ We can use them to store data on a user's machine for a period of time.
- ❖ A common use of cookies is with banner ads - the ad will set a cookie so the ad server knows which ads you have already seen.

Limitations of Cookies

- ❖ Could be switched off.
- ❖ Cookie data is stored on the client.

Sessions

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

When a session is started following things happen:

- ❖ PHP first creates a unique identifier for that particular session
- ❖ A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- ❖ Alternatively PHP, can automatically add the session ID to the end of urls. (You will need to compile PHP with --enable-trans-sid for this to work.)
- ❖ A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier.

Sessions

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

- ❖ Session variables contain data for a particular user – perhaps the number of items in their shopping cart, the ids of those items, or the total amount.
- ❖ The variables are actually stored at the *server*, and are accessed via the *session id* from your cookie.
- ❖ This is more secure than storing the data directly in a cookie.

Sessions

You begin a session by calling the function

session_start();

This will either:

- ❖ Create a new session, **or**
- ❖ Resume the current session, if a session id can be found from a cookie or in the URL.
- ❖ Generally, in a site that uses sessions, you will call this function at the start of every page where you need access to session variables.
- ❖ This creates the unique session id and stores it in a cookie (or adds it to the URL).
- ❖ If you reload the page, **session_start()** will find the session id and load any associated session variables.

Sessions

Setting Sessions

```
<?php
session_start();
if(isset($_SESSION['views'])) $_SESSION['views']=$_SESSION['views']+1;
else $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```


Destory Sessions

```
session_destroy();
//OR
unset($_SESSION['views']);
```

MVC

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.



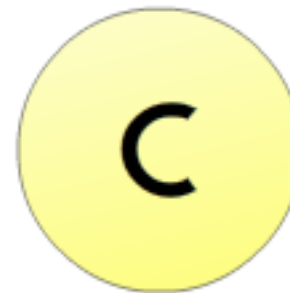
 model


database related,
not necessary
database. Data
can be XML or
even text files



 view

your website
design/ HTML
files. NO images,
css, etc.here.
only html layout



 controller

application logic,
process the user
request and get
appropriate data,
then output a design
from View

MVC

❖ **Model**

The Model is the name given to the permanent storage of the data used in the overall design. It must allow access for the data to be viewed, or collected and written to, and is the bridge between the View component and the Controller component in the overall pattern.

❖ **View**

The View is where data, requested from the Model, is viewed and its final output is determined. Traditionally in web apps built using MVC, the View is the part of the system where the HTML is generated and displayed.

❖ **Controller**

The final component of the triad is the Controller. Its job is to handle data that the user inputs or submits, and update the Model accordingly.

The Controller can be summed up simply as a collector of information, which then passes it on to the Model to be organized for storage, and does not contain any logic other than that needed to collect the input. The Controller is also only connected to a single View and to a single Model, making it a one way data flow system, with handshakes and signoffs at each point of data exchange.

MVC

