Practical 8 – Object Oriented PHP

For purposes of this example, you are going to create a *page* class. The main goal of this class is to limit the amount of HTML needed to create a new page. It should allow you to alter the parts that change from page to page, while automatically generating the elements that stay the same. The class should provide a flexible framework for creating new pages and should not compromise your freedom.

Because you are generating the page from a script rather than with static HTML, you can add any number of clever things including functionality to:

- Enable you to alter page elements in only one place. If you change the Copyright notice for example, or add an extra button, you should need to make the change in only a single place.
- Have default content for most parts of the page but be able to modify each element where required, setting custom values for elements such as the title and meta-tags.
- Recognize which page is being viewed and alter navigation elements to suit; there is no point in having a button that takes you to the home page located on the home page.
- Allow you to replace standard elements for particular pages. If, for instance, you want different navigation buttons in sections of the site, you should be able to replace the standard ones.

1. Getting Started

To get started, create a class called `Page`. Save it into a file called `page.inc`. The class needs some attributes. For this example, set elements that you might want changed from page to page as attributes of the class. The main content of the page, the title and meta-tags for example should be elements that may change from page to page.

The navigation buttons should probably be kept the same from page to page to avoid confusing people, but to change them easily you can make them an attribute too. Since the number of buttons can be variable, you may want to use an array and store both the text for the button and the URL it should point to, like this:

```
Public $buttons = array(“Home” => “home.php”, “Contact” =>
“contact.php”, “Services” => “services.php”);
```

2. Adding Operations/Methods to your class

To provide some functionality, the class also needs operations. You can start by providing **accessor** functions to set and get the values of the attributes you defined:

```
public function __set($name, $value){
     $this->$name = $value;
}
```

This __set() function does not contain error checking but it can easily be added later. Because you are unlikely to want to get these parameters later, you may not want to provide a __get() function.

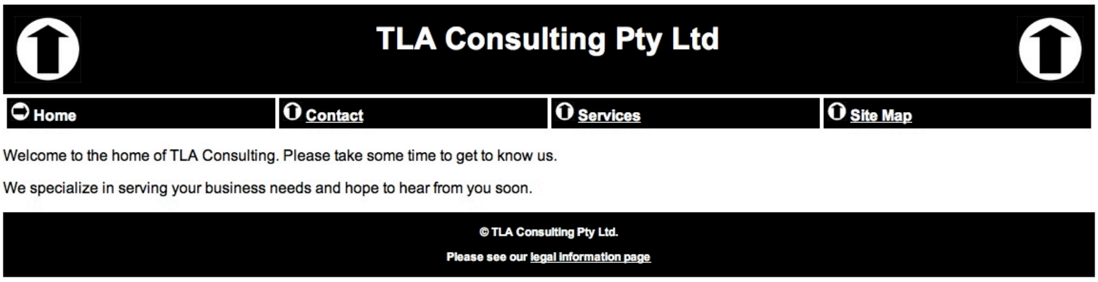
3. The main method of your class

The main purpose of this class is to display a page of HTML. You therefore need a function to do this. Call it Display() and add this code to it:

```
public function Display()
  {
    echo "<html>\n<head>\n";
    $this -> DisplayTitle();
    $this -> DisplayKeywords();
    $this -> DisplayStyles();
    echo "</head>\n<body>\n";
    $this -> DisplayHeader();
    $this -> DisplayMenu($this->buttons);
    echo $this->content;
    $this -> DisplayFooter();
    echo "</body>\n</html>\n";
  }
```

You now have a general structure for the rest of your class. You need to provide the code for the different methods that are to be used by the Display() method. At this point it shouldn’t matter much if the code is visible from outside your class, or if it’s inherited or not from child classes, so make them all public for the time being.

The following page is an example of what’s possible:



Welcome to the home of TLA Consulting. Please take some time to get to know us.

We specialize in serving your business needs and hope to hear from you soon.

© TLA Consulting Pty Ltd.

Please see our legal information page

4. Create a new instance of your class

Test your class by creating a new page called “`home.php`”, which includes the “`page.inc`” library. Then add a new instance of your class and call the Display() method:

```
<?php
   require(‘page.inc’);

   $homepage = new page();

   $homepage->content = ‘<p>Welcome to the homepage of
                        this website.  Please take some
                        time to get to know us.</p>
                        <p>We specialize in serving your
                        business needs and hope to hear
                        from you</p>’;

   $homepage->Display();
?>
```

Notice how little work you now need to generate new pages using this class.

5. Inheriting from the page class

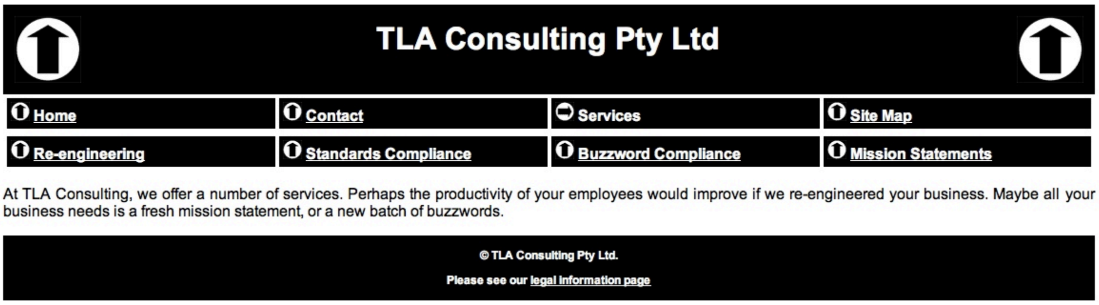
Using the page class means that all your pages need to be very similar to each other. If you want some sections of your website to have a variant of the standard page, you can create other classes by copying `page.inc` to `page2.inc` and making the necessary changes there. However if you make a change to `page.inc` you may need to remember to make similar changes to other pages. You can work around this problem by using inheritance:

```
<?php
    Require (‘page.inc’);
    Class servicePage extends page {
     Private $row2buttons = array{
          ‘re-engineering’ => ‘reengineering.php’,
          ‘Standards Compliance’ => ‘standards.php’,
          ‘Buzzword Compliance’ => ‘buzzword.php’,
          ‘Mission Statement’ => ‘mission.php’};
          Public function Display() {
      echo "<html>\n<head>\n";
      $this -> DisplayTitle();
      $this -> DisplayKeywords();
      $this -> DisplayStyles();
      echo "</head>\n<body>\n";
      $this -> DisplayHeader();
      $this -> DisplayMenu($this->buttons);
      $this -> DisplayMenu($this->row2buttons);
      echo $this->content;
      $this -> DisplayFooter();
      echo "</body>\n</html>\n";
     }
    }
?>
```

Then your services page may look like:

```
<?php
     Require (servicesPage.inc);
     $services = new servicePage();
     $services -> content = ‘<p>We offer a number of
services, such as...  blah blah</p>’;
     $services -> Display();
?>
```

Think of the advantages of using this to create your pages.



TLA Consulting Pty Ltd

At TLA Consulting, we offer a number of services. Perhaps the productivity of your employees would improve if we re-engineered your business. Maybe all your business needs is a fresh mission statement, or a new batch of buzzwords.

© TLA Consulting Pty Ltd.

Please see our legal information page