CSY2028
Web Programming
Topic 14

Tom Butler
thomas.butler@northampton.ac.uk

Topic 14

- Quick recap of last week
- Single point of entry
- .htaccess files and mod_rewrite

Last Week

- Separating out HTML from database logic
- Creating a PHP file for storing the layout html (layout.php)
- Filling layout.php with content

Layout.php

```
<!DOCTYPE html>
<html>
        <head>
                <title><?php echo $title; ?></title>
                <link rel="stylesheet" href="styles.css" />
        </head>

        <body>

                <header>
                        <h1>My Website</h1>
                </header>

                <nav>
                        <ul>
                                <li><a href="index.php">Home</a>
                                <li><a href="about.php">About</a>
                                <li><a href="contact.php">Contact</a>
                        </ul>
                </nav>

                <main>
                        <?php echo $content; ?>
                </main>

                <footer>
                        &copy; 2016
                </footer>
        </body>

</html>
```

Variables in place of content that will be replaced

Layout.php

- If this was placed in layout.php it could be used like this:

```php
<?php
$title = 'My Website - Home';

$content = '<p>Welcome to my website</p>';

require 'layout.php';
```

```php
<?php
$title = 'My Website - Contact';

$content = '<p>You can contact me by telephone and by email:...</p>';

require 'layout.php';
```

Layout.php

- Because the template contains the echo command and prints to the screen, the content of layout.php is included
- Then the contents of book-list-template.php is included
- What's needed is to load the contents of the file book-list-template.php into a variable e.g.

```
<?php
$booksTable = new DatabaseTable($pdo, 'book');

$books = $booksTable->find('publisher', 'Penguin Books');

$content = require 'book-list-template.php';
require 'layout.php';
```

Layout.php

```php
<?php
$booksTable = new DatabaseTable($pdo, 'book');

$books = $booksTable->find('publisher', 'Penguin Books');

$content = require 'book-list-template.php';
require 'layout.php';
```

- Unfortunately the require function in PHP doesn't quite work this way
- However, it is possible to do this

Output buffering

- The output buffer can be started with the code:

```
ob_start();
```

- Anything that would be printed to the screen after this has been called will be written to the buffer instead

```
<?php
ob_start();
echo '<p>Welcome to my website</p>';
```

Output buffering

- You can then read from the buffer using

```php
$contents = ob_get_clean();
```

```php
<?php
ob_start();
echo '<p>Welcome to my website</p>';
$contents = ob_get_clean();
```

The **$contents** variable will now store the string
Welcome to my website
And it won't have been printed to the screen

Output buffering

- This also works with require statements:

```php
<?php
ob_start();
require 'page.php';
$contents = ob_get_clean();
```

```php
<?php
$booksTable = new DatabaseTable($pdo, 'book');

$books = $booksTable->find('publisher', 'Penguin Books');

ob_start();
require 'book-list-template.php';
$content = ob_get_clean();
```

The **$content** variable will now store the HTML generated in the book-list-template.php file
And it won't have been printed to the screen

Output buffering

- By combining output buffering and layout.php from earlier it's possible to set the content of the layout for the dynamic pages:

```php
<?php
$booksTable = new DatabaseTable($pdo, 'book');

$books = $booksTable->find('publisher', 'Penguin Books');

ob_start();
require 'book-list-template.php';
$content = ob_get_clean();


$title = 'Books published by Penguin Books';
require 'layout.php';
```

Output buffering

- Because this is something that you'll probably want to do a lot, it's better to move it to a function:

```
function loadTemplate($fileName) {
        ob_start();
        require $fileName;
        $contents = ob_get_clean();
        return $contents;
}
```

-

```
<?php
$booksTable = new DatabaseTable($pdo, 'book');

$books = $booksTable->find('publisher', 'Penguin Books');

$content = loadTemplate('book-list-template.php');


$title = 'Books published by Penguin Books';
require 'layout.php';
```

Template function

- This now allows the loadTemplate function to load any template file and set any variables:

```php
function loadTemplate($fileName, $templateVars) {
        ob_start();
        require $fileName;
        $contents = ob_get_clean();
        return $contents;
}
```

```php
$books = $booksTable->find('publisher', 'Penguin Books');

$templateVars = ['title' =>'My Website',
'content' =>'<p>Welcome to my website</p>'
];
$content = loadTemplate('layout.php',$templateVars);
```

Template function

- This works, however it does require using an array for all the variables in the template

```
<!DOCTYPE html>
<html>
        <head>
                <title><?php echo $templateVars['title']; ?></title>
                <link rel="stylesheet" href="styles.css" />
        </head>

        <body>
…..
```

- For simplicity it would be better to use just the variable name:

```
<!DOCTYPE html>
<html>
        <head>
                <title><?php echo $title; ?></title>
                <link rel="stylesheet" href="styles.css" />
        </head>

        <body>
…..
```

Template function

- PHP contains a function called *extract()*
- This function takes an array
  - For each element in the array, it creates a variable with the name of the key

```php
$myArray = ['title' => 'My page title'];

extract($myArray);

echo $title;
```

```
Output:
My page title
```

Template function

- This function can then be used to load any template or load one template into another:

```php
$books = $booksTable->find('publisher', 'Penguin Books');

$templateVars = ['books' => $books];

$content = loadTemplate('book-list-template.php', $templateVars);




$templateVars = ['title' => 'My Website',
'content' => $content
];
$content = loadTemplate('layout.php', $templateVars);

echo $content;
```

Load the list template and store it
In $content
the $books variable is set
in $templateVars

Load the layout and set the content variable to the loaded template


$penguinBooks = $booksTable->find('publisher', 'Penguin Books');
$templateVars = ['books' => $penguinBooks];
$content = loadTemplate('book-list-template.php', $templateVars);
<?php
echo '<ul>';
foreach ($books as $book) {
echo '
' . $book['title'] . '
';
}
echo '
';
The variable $content will now store:

Moby Dick
Treasure Island


```php
$templateVars = ['title' => 'My Website',
'content' => $content
];

$layoutHTML = loadTemplate('layout.php', $templateVars);

echo $layoutHTML;
```

```
Contents of $content variable (from last slide)
<ul>
      <li>Moby Dick</li>
      <li>Treasure Island</li>
</ul>
```

```php
<!DOCTYPE html>
<html>
    <head>
         <title><?php echo $title; ?></title>
        <link rel="stylesheet" href="styles.css" />
    </head>

    <body>
       <header>
              <h1>My Website</h1>
       </header>
       <nav>
          <ul>
            <li><a href="index.php">Home</a>
            <li><a href="about.php">About</a>
            <li><a href="contact.php">Contact</a>
          </ul>
       </nav>

       <main>
         <?php echo $content; ?>
       </main>
       <footer>&copy; 2016</footer>
   </body>
</html>
```

```php
<!DOCTYPE html>
<html>
    <head>
         <title>My title</title>
        <link rel="stylesheet" href="styles.css" />
    </head>

    <body>
       <header>
              <h1>My Website</h1>
       </header>
       <nav>
          <ul>
            <li><a href="index.php">Home</a>
            <li><a href="about.php">About</a>
            <li><a href="contact.php">Contact</a>
          </ul>
       </nav>

       <main>
         <ul>
            <li>Moby Dick</li>
            <li>Treasure Island</li>
         </ul>
       </main>
       <footer>&copy; 2016</footer>
   </body>
</html>
```

loadTemplate function

- The loadTemplate function allows loading any PHP file and setting the variables it uses

```php
$templateVars = [
'title' => 'My Website',
'content' => $content
];

$layoutHTML = loadTemplate('layout.php', $templateVars);

echo $layoutHTML;
```

Array of variables/values to set

Name of PHP File

Single Entry Point

- It's usual that most pages will do a similar job:
  - Load a layout with the outer HTML that is common to every page (Head tag, body tag, navigation, footer, etc)
  - Load the middle content, unique for that page
  - Put the middle content into the layout

Single Entry Point

- This results in repeated code. A lot of pages will look like this:

```php
<?php

//Include the file that contains the loadTemplate function
require 'loadTemplate.php';

$content = loadTemplate('about.php', []);

$templateVars = [
  'title' => 'About our company',
  'content' => $content
];

echo loadTemplate('layout.php', $templateVars);
```

Single Entry Point

- Pages with forms will include some additional logic e.g.

```php
<?php

//Include the file that contains the loadTemplate function
require 'loadTemplate.php';


//Include the file that contains the DatabaseTable class and the $pdo object
require 'database.php';


$users = new DatabaseTable($pdo, 'users');

if (isset($_POST['submit'])) {
        $users->save($_POST['user']);
        $user = $_POST['user'];
}
else {
        if (isset($_GET['id'])) {
                $user = $users->find('id', $_GET['id']);
        }
        else $user = false;
}

$content = loadTemplate('account.php', [
        'user '= $user;
]);

$templateVars = [
  'title' => 'About our company',
  'content' => $content
];

echo loadTemplate('layout.php', $templateVars);
```

Account.php

- This would then use a HTML form such as this:

```
<form action="" method="post">

        <input type="hidden" value="<?php if (isset($user['id'])) echo $user['id'] ?>" />

        <label>First name</label> <input name="user[firstname]"
                        value="<?php if (isset($user['firstname'])) echo $user['firstname'] ?>" />

        <label>Surname</label> <input name="user[surname]"
                        value="<?php if (isset($user['surname'])) echo $user['surname'] ?>" />

        <input type="submit" name="submit" value="Save" />
</form>
```

Similar pages

- Most pages will look similar. They will contain this code:

```php
<?php

//Include the file that contains the loadTemplate function
require 'loadTemplate.php';

$content = loadTemplate('….');

$templateVars = [
  'title' => 'About our company',
  'content' => $content
];

echo loadTemplate('layout.php', $templateVars);
```

Load a template into the $content variable

Set the page title and content
For the layout

Similar pages

- It's possible to move this logic into its own file and reference it using a GET variable

index.php

```php
<?php


require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

about.php

```php
<?php

$title =  'About our company',
$content = loadTemplate('about-template.php');
```

Similar pages

- Then by visiting the url
- index.php?page=about
- It will automatically load the about template into the layout and set the content

index.php

```php
<?php


require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

about.php

```php
<?php

$title =  'About our company',
$content = loadTemplate('about-template.php');
```

Adding pages

- Adding new pages to the website is now very easy. For example, to add a Contact page the following files could be added

contact.php

```php
<?php

$title =  'Contact Us',
$content = loadTemplate('contact-template.php');
```

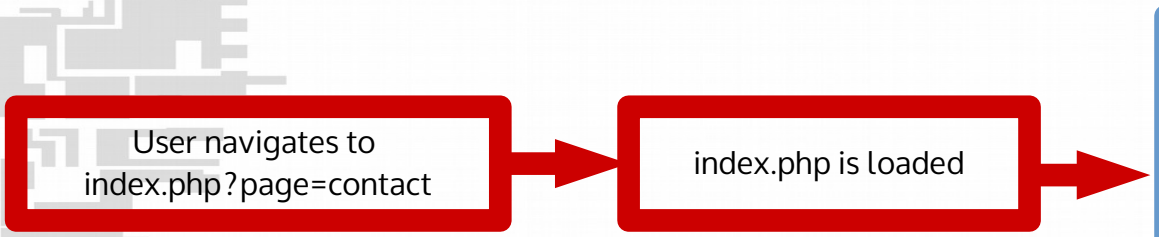
contact-template.php

```php
<h2>Contact us</h2>

<p>Please email us at
<a href="mailto:support@company.com">
     support@company.com
</a></p>
```

Adding pages

- When visiting `index.php?page=contact`
- The contact template and logic will be automatically loaded and displayed in the layout

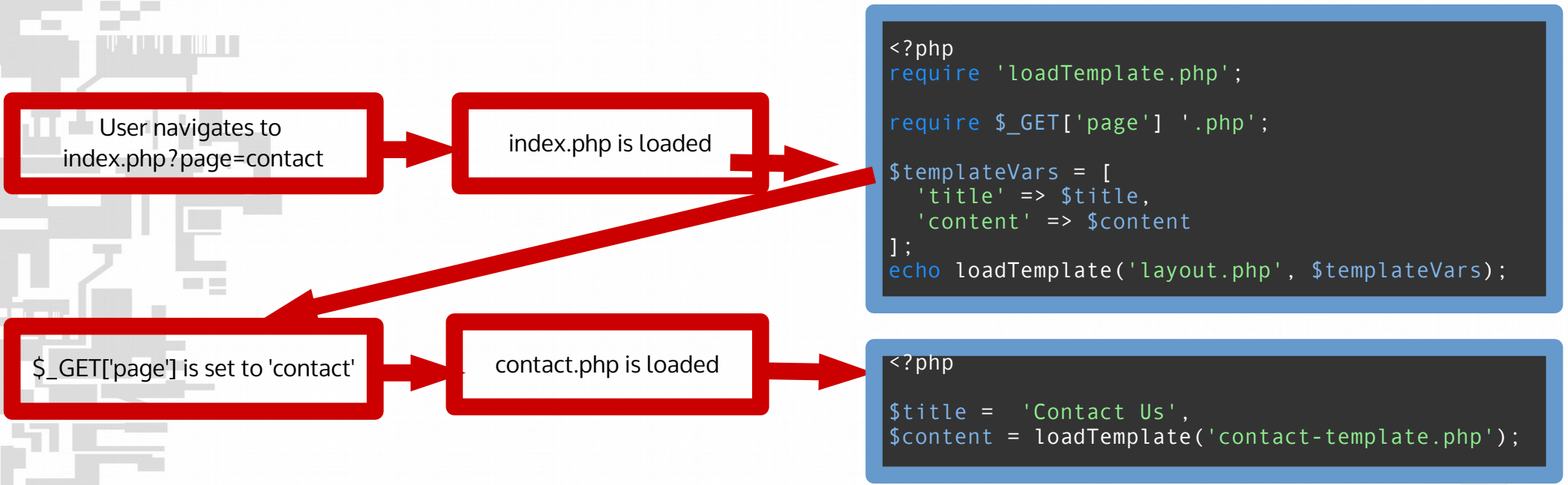



```php
<?php


require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```


User navigates to index.php?page=contact
index.php is loaded
<?php
require 'loadTemplate.php';
require $_GET['page'] '.php';
$templateVars = [
'title' => $title,
'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
$_GET['page'] is set to 'contact'
contact.php is loaded
<?php
$title = 'Contact Us',
$content = loadTemplate('contact-template.php');

User navigates to index.php?page=contact

index.php is loaded

```php
<?php
require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

$_GET['page'] is set to 'contact'

contact.php is loaded

```php
<?php
$title =  'Contact Us',
$content = loadTemplate('contact-template.php');
```

contact.php sets the $title And $content variables for index.php

```php
<?php
require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

User navigates to index.php?page=contact

index.php is loaded

```php
<?php
require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

$_GET['page'] is set to 'contact'

contact.php is loaded

```php
<?php
$title =  'Contact Us',
$content = loadTemplate('contact-template.php');
```

contact.php sets the $title
And $content variables
for index.php

```php
<?php
require 'loadTemplate.php';

require $_GET['page'] '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('layout.php', $templateVars);
```

The layout template is loaded
And the $title and $content
variables are set:

contact-template.php

```php
<?php
$title =  'Contact Us',
$content = loadTemplate('contact-template.php');
```

```php
<h2>Contact us</h2>
<p>Please email us at
<a href="mailto:support@company.com">
      support@company.com
</a></p>
```

```php
<!DOCTYPE html>
<html>
    <head>
        <title><?php echo $title; ?></title>
        <link rel="stylesheet" href="styles.css" />
    </head>

    <body>
       <header>
              <h1>My Website</h1>
       </header>
       <nav>
          <ul>
            <li><a href="index.php">Home</a>
            <li><a href="about.php">About</a>
            <li><a href="contact.php">Contact</a>
          </ul>
       </nav>

       <main>
         <?php echo $content; ?>
       </main>
       <footer>&copy; 2016</footer>
   </body>
</html>
```

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Contact Us</title>
        <link rel="stylesheet" href="styles.css" />
    </head>

    <body>
       <header>
              <h1>My Website</h1>
       </header>
       <nav>
          <ul>
            <li><a href="index.php">Home</a>
            <li><a href="about.php">About</a>
            <li><a href="contact.php">Contact</a>
          </ul>
       </nav>
       <main>
         <h2>Contact us</h2>

            <p>Please email us at
            <a href="mailto:support@company.com">
                  support@company.com
           </a></p>
       </main>
       <footer>&copy; 2016</footer>
   </body>
</html>
```

Forms

- This can also be used with pages that are more dynamic, e.g. using forms
- The following form could be used to register a user or update their details:

account.php

```php
<?php
//Include the file that contains the DatabaseTable
//class and the $pdo object
require 'database.php';


$users = new DatabaseTable($pdo, 'users');

if (isset($_POST['submit'])) {
        $users->save($_POST['user']);
        $user = $_POST['user'];
        $content = 'Record saved';
}
else {
        if (isset($_GET['id'])) {
                $user = $users->find('id', $_GET['id']);
        }
        else $user = false;

        $content = loadTemplate('account-template.php', [
          'user' => $user;
        ]);
}

$title = 'Manage Account';
```

account-template.php

```php
<form action="" method="post">

 <input type="hidden"
        value="<?php if ($user) echo $user['id'] ?>" />

 <label>First name</label>

  <input name="user[firstname]"
         value="<?php if ($user)
                echo $user['firstname'] ?>" />

 <label>Surname</label>

 <input name="user[surname]"
       value="<?php if ($user)
              echo $user['surname'] ?>" />

  <input type="submit" name="submit" value="Save" />
</form>
```

Quickly adding pages

- This allows you to quickly develop a website
- New pages can easily be added by:
  - Creating a PHP code that contains all the logic for that page and loads a template e.g. register.php
  - Creating a template file that contains all the HTML code specific for that page e.g. the form
  - Visit index.php?page=register and the HTML for that page will be loaded into the HTML layout

Exercise 1

- Amend the pages you've been working on over the last few weeks to use this new format using `index.php` as a single point of entry.
- Users should visit index.php?page=addmessage or index.php?page=list instead of list.php and addmessage.php respectively

File structure

- Using this approach with separated files, it's possible for users to visit the individual files without going through index.php
- This can be a problem because users might see PHP errors giving away how the site is built.
- It also causes issues with search engines, search engines might find these pages and list them in search results. If someone comes to your site and sees a broken page it doesn't look good for the company

Storing files outside public_html

- Any file inside the public_html directory is accessible on the website by visiting http://v.je/$fileName
- You can store files on the server *outside* of the public_html directory
- The public_html directory resides in a directory called *website*
- Anything in this folder exists on the *virtual server*

Storing files outside public_html

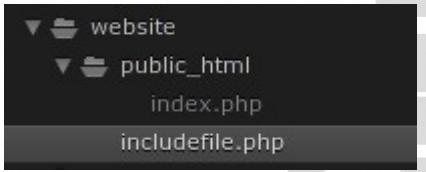
- You can store files outside of the public_html directory by placing them up a level in the *website* folder and then reference them in your *require* statements using `../`
- In a path, `../` means "the level above"

Storing files outside public_html

- You can write a require statement like this

```php
<?php
require '../includefile.php';
```

- This will look for the file `includefile.php` in the level above public_html

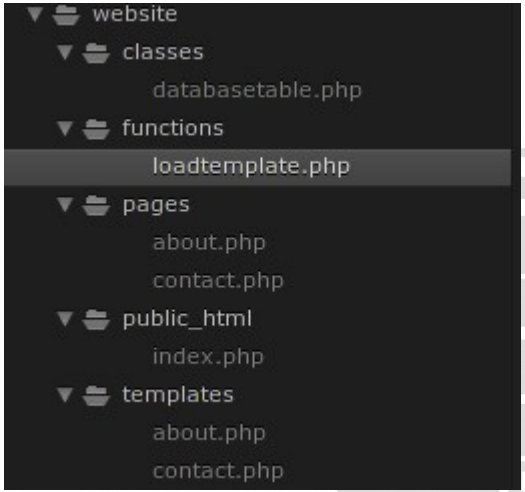


Website file structure

- To keep things organised it's a good idea to group files together based on the tasks they perform
- e.g.
  - A directory for templates
  - A directory for individual pages
  - A directory for classes
  - A directory for functions

Website file structure

- The only PHP file inside the public_html directory is now index.php
- All requests will go through that e.g.
- v.je/index.php?page=about
- v.je/index.php?page=contact
- Which will then load all the relevant files



Require statements

- A require statement loads files from something called the *current working directory*
- On a website this is almost always the public_html directory

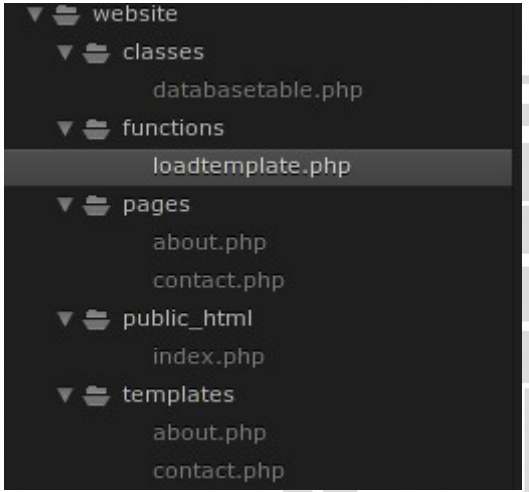
index.php

```php
<?php
require '../pages/contact.php';
```

contact.php

```php
<?php
require 'about.php';
```

This will look for 'about.php' in public_html, not the pages directory!



Updated index.php

- The updated index.php with this file structure looks like this:

```php
<?php
require '../functions/loadtemplate.php';

require '../pages/' . $_GET['page'] . '.php';

$templateVars = [
  'title' => $title,
  'content' => $content
];
echo loadTemplate('../templates/layout.php', $templateVars);
```

Exercise 2

- Separate all your files into the directory structure from the previous slides

URL formats

- The URL format looks like this:
  - /index.php?page=contact
  - /index.php?page=about
  - /index.php?page=register
- This then loads the relevant file into the layout
- Although this works, the URL is not very user friendly

URL Formats

- It's possible to change the URL that a page uses
- This is done via the web server
- You can configure the web server with a file named .htaccess
- .htaccess is placed in the public_html directory and contains instructions for the web server
- There are a lot of options you can set, however a very useful one is an apache extension called mod_rewrite

.htaccess files

- .htaccess files provide instructions for the webserver:
- The file name is .htaccess, that's a file starting with a dot
- NOTE: Some text editors (e.g. notepad) cannot create files starting with a dot, you will need to use a modern editor to create the file

mod_rewrite

- mod_rewrite is enabled using the command
- ```
  RewriteEngine On
  ```
- Inside the .htaccess file
- Once the RewriteEngine is turned on you can create custom URLs
  ```

mod_rewrite

- When you visit a page on the website, mod_rewrite checks any configured *rules* that have been applied
- Rules can be as simple as "if the page requested is contact.php show the user home.php instead"
- Which is done like this:

```
RewriteEngine On
RewriteRule contact.php home.php
```

mod_rewrite

- When someone navigates to the page contact.php they will see the contents of home.php as if they'd gone directly to it
- contact.php will also be displayed in the address bar!

```
RewriteEngine On
RewriteRule contact.php home.php
```

mod_rewrite

- This allows us to make the URLs more user friendly
- We could use this to show index.php?page=contact on /contact
- http://v.je/contact
- Will now show the page that is actually stored on http://v.je/index.php?page=contact

```
RewriteEngine On
RewriteRule contact index.php?page=contact
```

mod_rewrite

- You can add more than one rule

```
RewriteEngine On
RewriteRule contact index.php?page=contact
RewriteRule about index.php?page=about
RewriteRule account index.php?page=account
```

- However, this requires a lot of repetition
- Each page has to be configured in .htaccess
- A better method is using a wildcard

mod_rewrite

- A wilcard can be set up like this

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule (.*) index.php?page=$1 [QSA,L]
```

This says “Don't redirect if it's a file”
Otherwise
/style.css would be redirected to
/index.php?page=style.css

Which probably isn't what you want to do!

Match any URL

$1 will be whatever the URL that was typed in was

mod_rewrite

- Now, if you visit v.je/contact it will load v.je/index.php?page=contact, /register will show index.php?page=register and so on!

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule (.*) index.php?page=$1 [QSA,L]
```

Exercise 3

- Use mod_rewrite to make the URLs nicer so you can visit /list to list messages, /edit to show the add user page, etc.