

LAB 21

CONTENT MANAGEMENT SYSTEMS

What You Will Learn

- How to install and configure WordPress
- How to customize WordPress using freely available plugins and themes
- How to customize themes yourself
- How to develop custom features inside the WordPress CMS

Approximate Time

The exercises in this lab should take approximately 120 minutes to complete.

Fundamentals of Web Development, 2nd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
<http://www.funwebdev.com>

Date Last Revised: May 20, 2017

TITLE

PREPARING DIRECTORIES

- 1 If you haven't done so already, create a folder in your personal drive for all the labs for this book.
- 2 From the main labs folder (either downloaded from the textbook's web site using the code provided with the textbook or in a common location provided by your instructor), copy the folder titled lab0X to your course folder created in step one.

For this lab you will require a webserver with PHP and MySQL that you should have set up in previous chapters.

Since this chapter uses the Wordpress CMS, there are some sections that will defer you to Wordpress documentation, which might be updated as new versions are released. Learning how to use the online documentation is an important skill to develop.

GETTING STARTED WITH WORDPRESS

Exercise 21.1 Set Up WordPress

- 1 For this example you will need a folder to store all the WordPress files. In our case we named the folder lab21.

Note: If you want to use this WordPress CMS for another task, consider using the name of the domain you will be hosting as the name for the folder.
- 2 Once the folder is created, enter the folder on the command line and download the latest version of WordPress using the following command:

```
curl -O https://wordpress.org/latest.tar.gz
```

Note: There are other ways to download the Wordpress archive to your server, including downloading through a browser and transferring manually.

- 3 With the [.tar.gz](#) archive downloaded, it can be extracted with a single command on the command line:

```
tar -xzf latest.tar.gz
```

The above command extracts the archive into a folder named [wordpress](#)

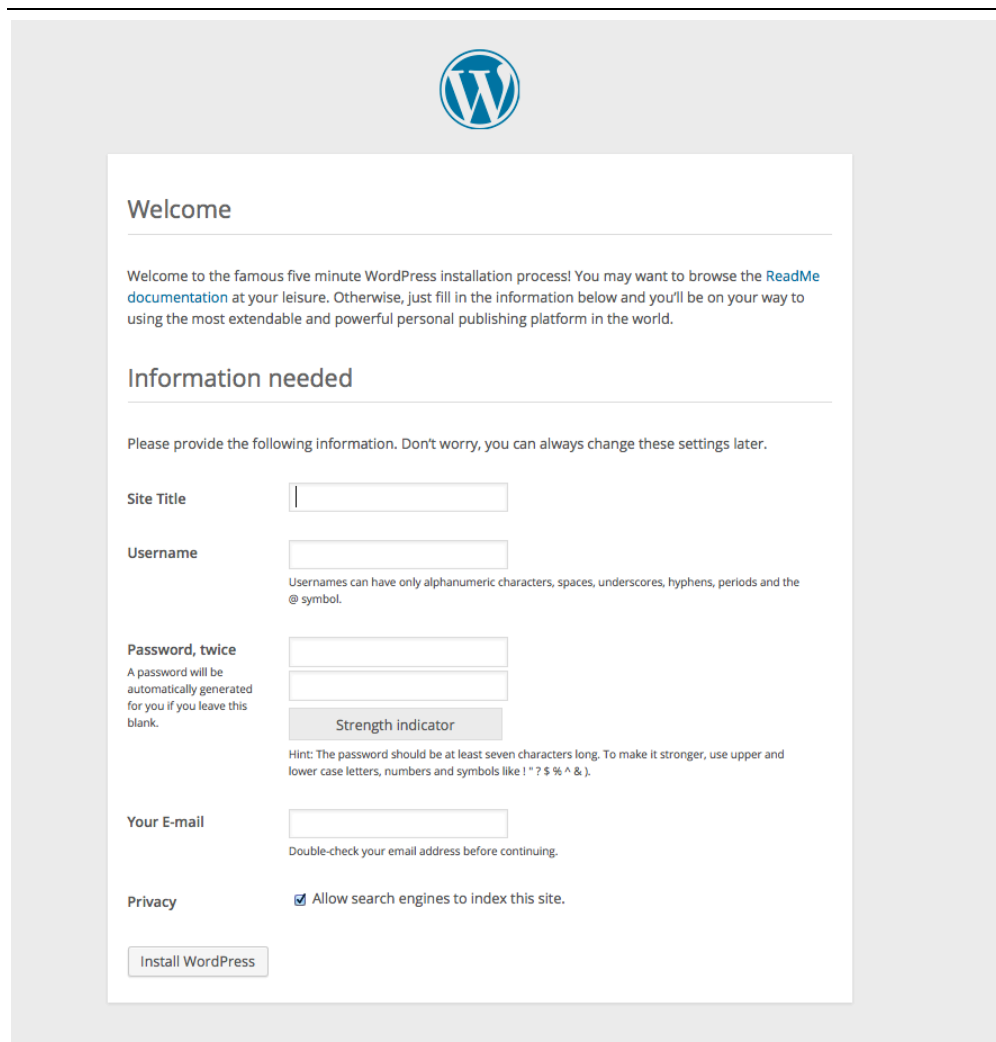
- 4 Now, you must create the [wp-config.php](#) file based on the [wp-config-sample.php](#)

file. Begin by making a copy of the provided file into the destination file name:

```
cp wp-config-sample.php wp-config.php
```

Now configure the known aspects of the WordPress site in that file. In particular you will need to set up the database host, username and password. We suggest using phpMyAdmin to create a new database and database user for your site (see Chapter 11 for more details on this).

With `wp-config.php` set up., when you visit <http://localhost/lab21/wordpress/> (or whatever URL is associated with your new WordPress installation) you can follow the directions on the browser to complete your setup as shown in Figure 21.1



The screenshot shows the WordPress installation questionnaire interface. At the top is the WordPress logo. Below it is a 'Welcome' section with a message about the five-minute installation process and a link to the documentation. The 'Information needed' section follows, with a note that settings can be changed later. It contains several input fields: 'Site Title', 'Username' (with a note about allowed characters), 'Password, twice' (with a note about automatic generation and a 'Strength indicator' button), 'Your E-mail' (with a note to double-check), and a 'Privacy' section with a checked checkbox for 'Allow search engines to index this site.' At the bottom is an 'Install WordPress' button.

WordPress logo

Welcome

Welcome to the famous five minute WordPress installation process! You may want to browse the [ReadMe documentation](#) at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods and the @ symbol.

Password, twice
A password will be automatically generated for you if you leave this blank.

Hint: The password should be at least seven characters long. To make it stronger, use upper and lower case letters, numbers and symbols like ! " ? \$ % ^ & .

Your E-mail
Double-check your email address before continuing.

Privacy ☒ Allow search engines to index this site.

Figure 21.1 Screenshot of the WordPress Installation Questionnaire once `wp-config.php`

is set up to connect to your database.

After following the directions in the browser you will have successfully set up WordPress with the default appearance of the latest WordPress theme and some sample content. You will have an administrator account for your new WordPress site and can now perform the remaining aspects of this Lab. In our case the new site appears similar to that in Figure 21.2

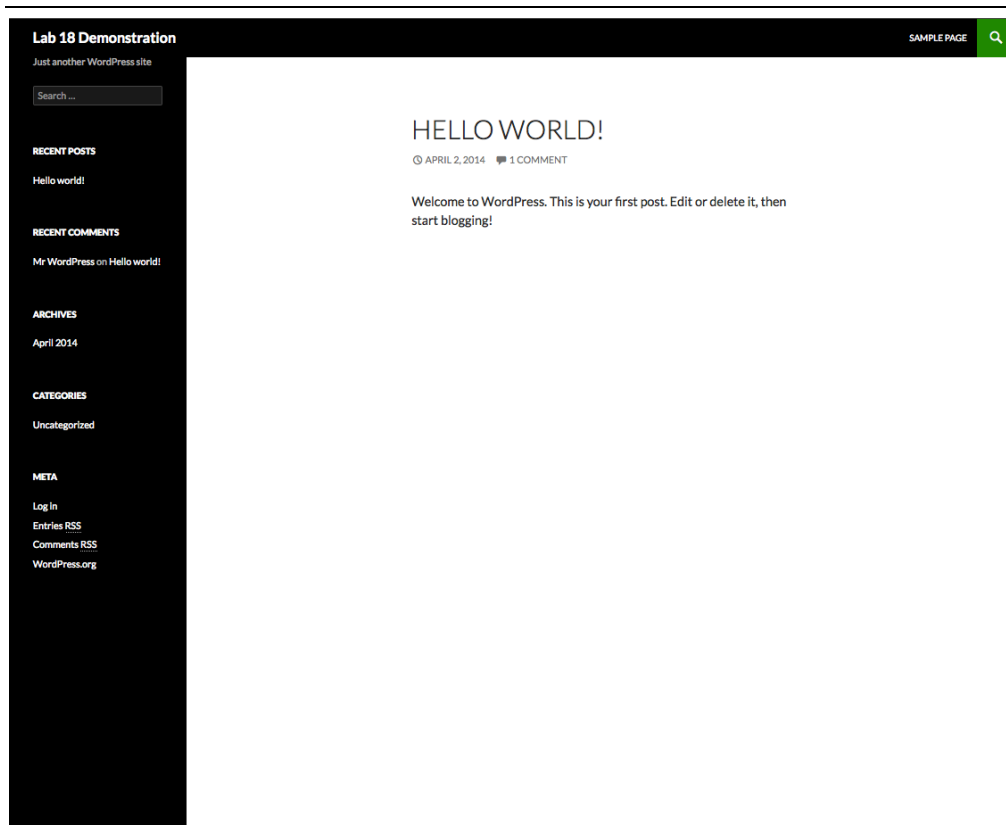


Figure 21.2 A screenshot of the newly installed WordPress site

Exercise 21.2 – Create Pages

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1.
- 2 Begin by logging in as the administrator (or another user you have with permission to create pages). The admin URL is [/wp-admin/](#) off the root of your site.

- 3 Login to WordPress as an administrator. Hover on "Pages" in the left menu and then choose New Page as shown in Figure 21.3.

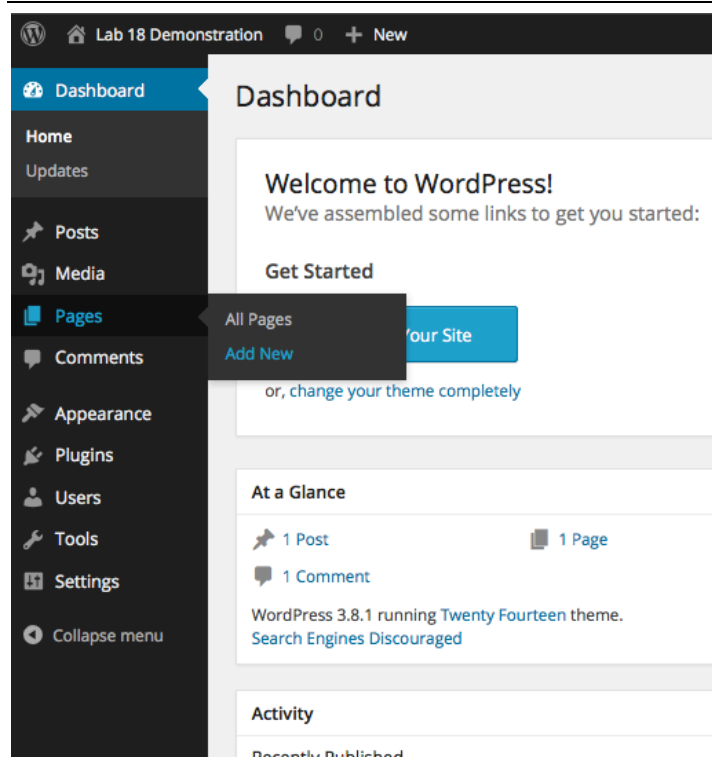


Figure 21.3 Adding a new Page in WordPress

- 4 Fill in some content for the title and content area as shown in Figure 21.4. When you are finished click publish on the right hand side.

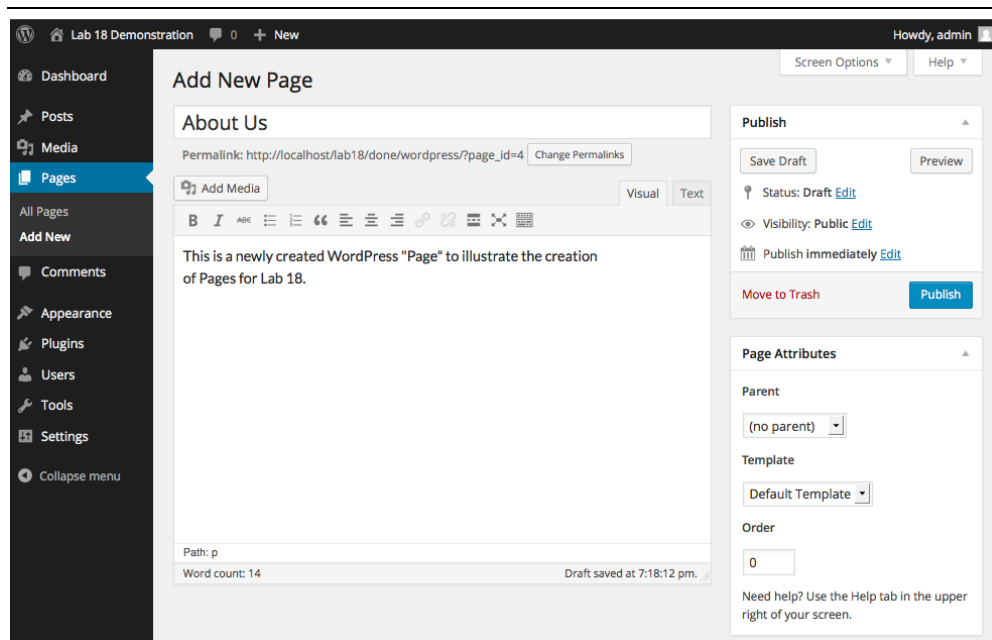


Figure 21.4 Editor to create a new page.

Note: There are some powerful publishing options aside from publishing immediately. You can save as draft or schedule publication for a later date. Excellent if you don't want to wait until midnight to post about a topical matter.

- 5 Go to your main site and see that the newly published page with the title "About Us" is now appearing in the main menu of the site, as shown in Figure 21.5

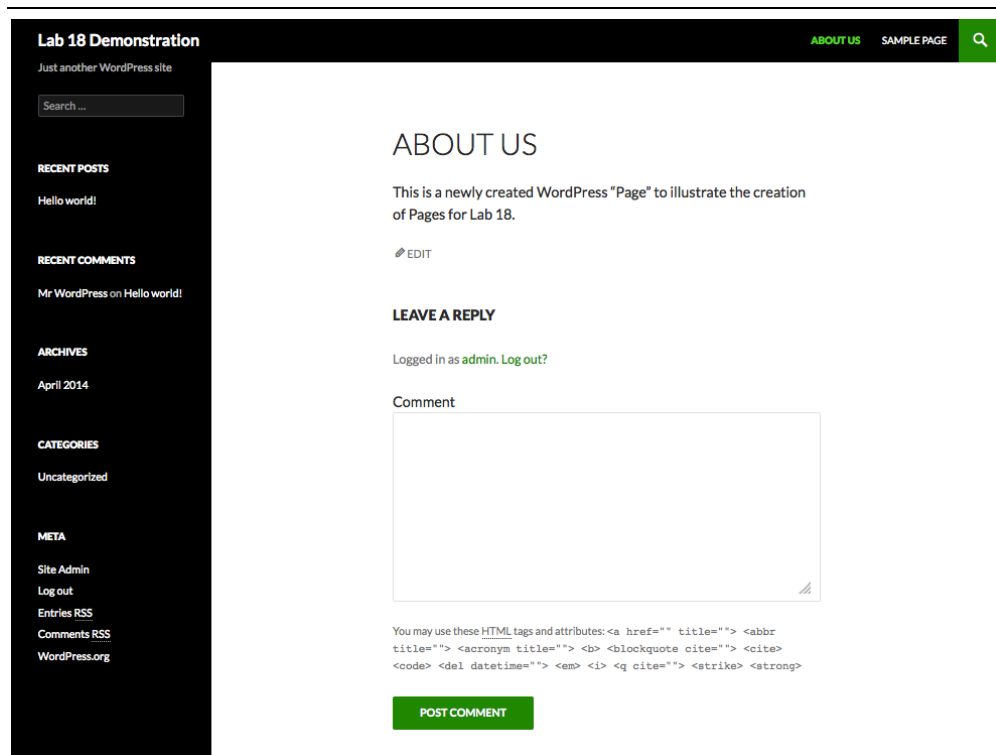


Figure 21.5 A Newly published page appearing in the site menu linked to the new content and lots of widgets in the sidebars.

- 6 Publish one more page, but this time make it a subpage of the "About Us" Page. This is accomplished by choosing the "About Us" page as the parent of this new page before publishing as shown in Figure 21.6,

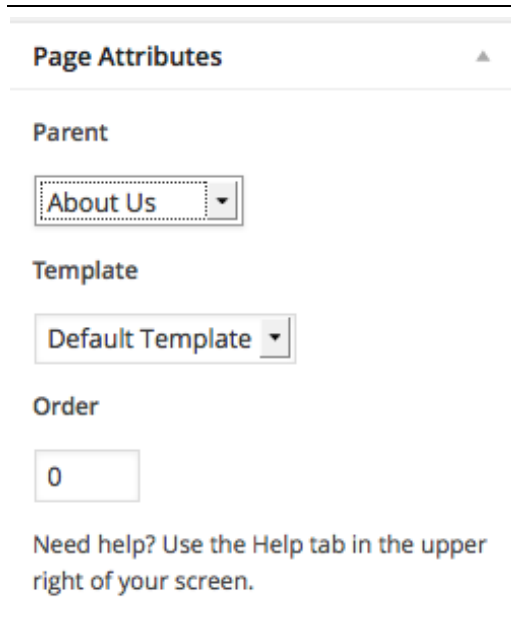


Figure 21.6 The Page Attributes tab when creating or editing a page.

The net result is that your navigational menu will contain the page as a subpage in the menu hierarchy as shown in Figure 21.7. As an exercise for the reader use the options shown in Figure 21.6 to reorder the menu items.

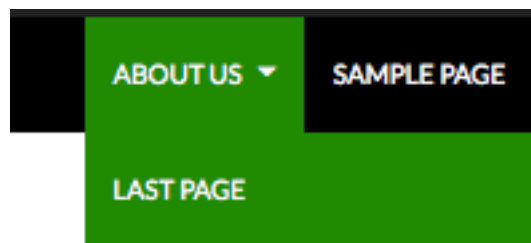


Figure 21.7 Hierarchy in Wordpress menu based on the "parent" attribute of the pages.

MORE WORDPRESS ADMINISTRATION

Exercise 21.3 – Navigation from Pages

- 1 This lab requires that you have a working WordPress installation as described in

Exercise 21.1. Ideally you have also added some pages as described in Exercise 21.2.

- 2 Login to WordPress as an administrator. Navigate to the Appearance section. In that section there is a section named menu as shown in Figure 21.8. *Note: If you are using a different theme this interface might not contain the exact same menu system described in this lab.*

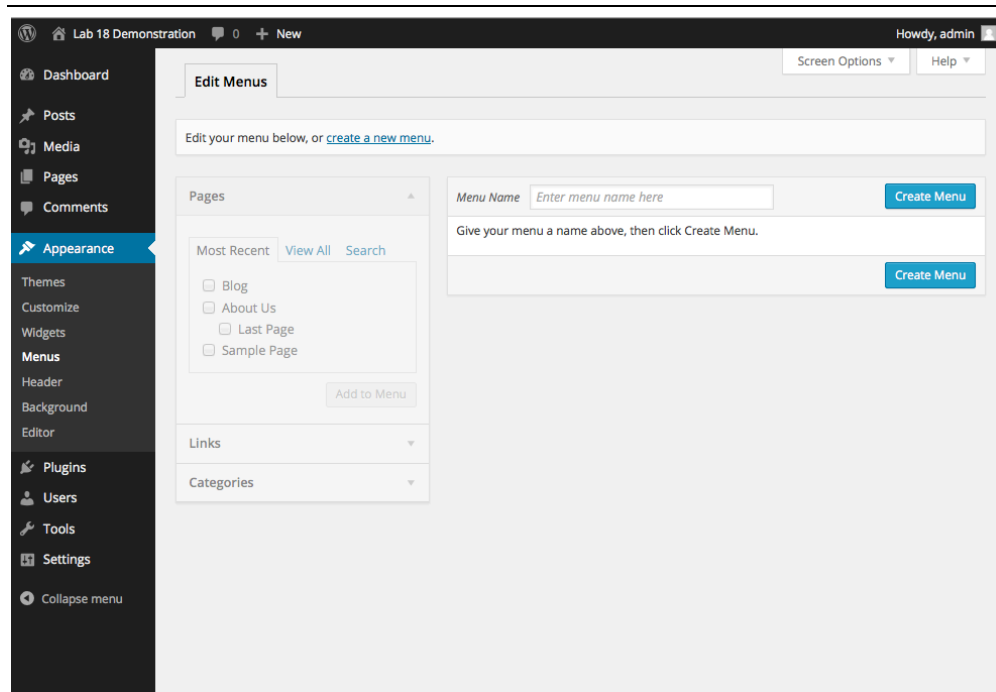
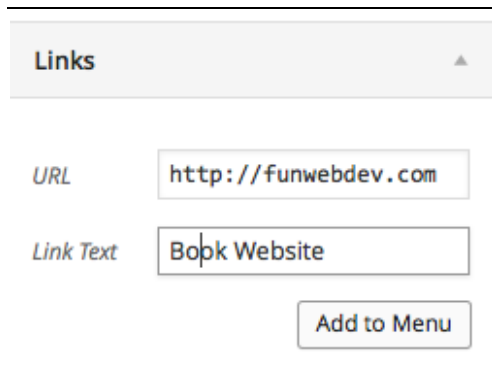


Figure 21.8 Menu item in the Appearance part section of the admin interface.

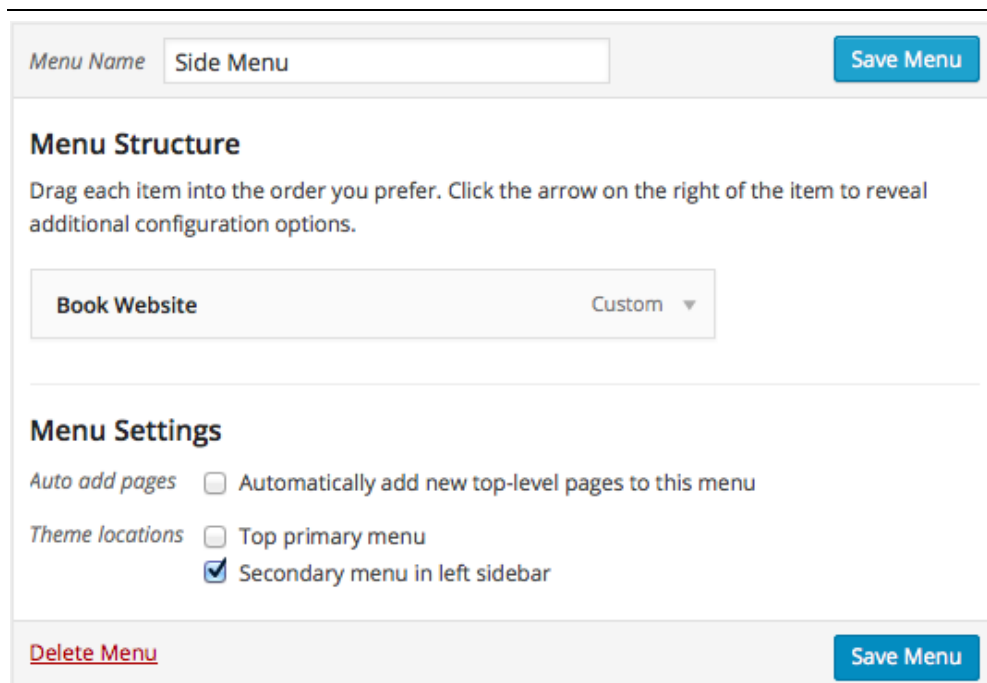
- 3 Click "Create a new menu" and name it something descriptive. We used "Side Menu".
- 4 Now create a new menu item which is just a link to another site as shown in Figure 21.9 and click add To Menu.



The screenshot shows a form titled "Links" with a small upward arrow icon. It contains two input fields: "URL" with the value "http://funwebdev.com" and "Link Text" with the value "Book Website". Below these fields is a button labeled "Add to Menu".

Figure 21.9 Adding a link to a menu

- 5 Now select the checkbox to put this menu item in the left of the theme (again if using a different theme you may have different options). Click save as shown in Figure 21.10 and you will now have the menu in the left of all pages.



The screenshot shows the "Side Menu" configuration page. At the top, the "Menu Name" is "Side Menu" and there is a "Save Menu" button. Below this is the "Menu Structure" section, which includes instructions: "Drag each item into the order you prefer. Click the arrow on the right of the item to reveal additional configuration options." There is one menu item, "Book Website", with a "Custom" dropdown arrow. Below the menu structure is the "Menu Settings" section, which includes three checkboxes: "Auto add pages" (unchecked), "Top primary menu" (unchecked), and "Secondary menu in left sidebar" (checked). At the bottom, there is a "Delete Menu" link and a "Save Menu" button.

Figure 21.10 The side menu being edited in the admin interface.

- 6 The other way that items are added to sidebars is through Widgets. We will now turn off all other widgets (for the time being) to show a more bare bones site.

Navigate to Appearance->Widgets and drag all of the items in the "Primary sidebar" out of the interface and drop them as shown in Figure 21.11 (they will disappear from the menu but still be there if you want to add them back later).

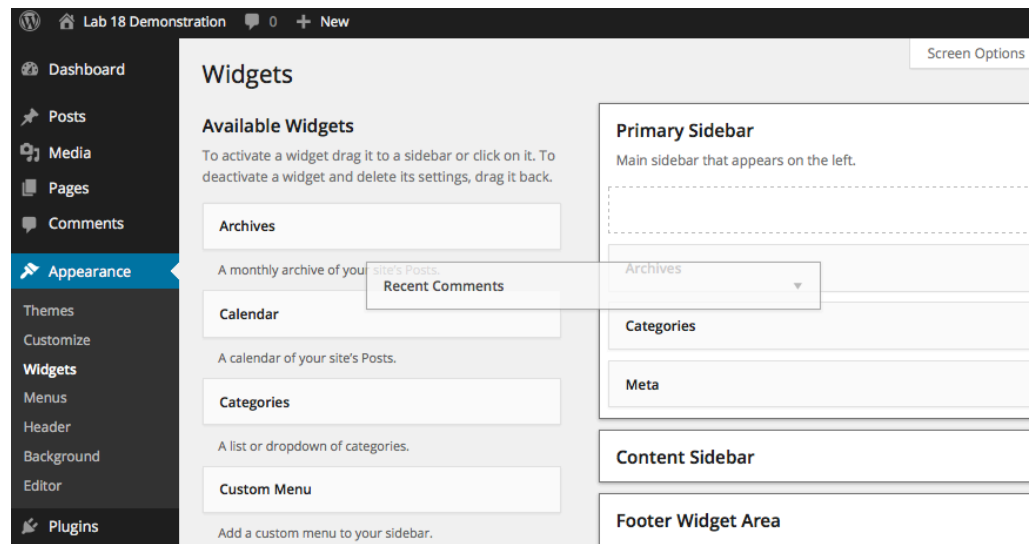


Figure 21.11 Removing widgets that are enabled in a new WordPress Installation.

Now your page should have only the page menu on top and the one item in the side menu as illustrated in Figure 21.12.

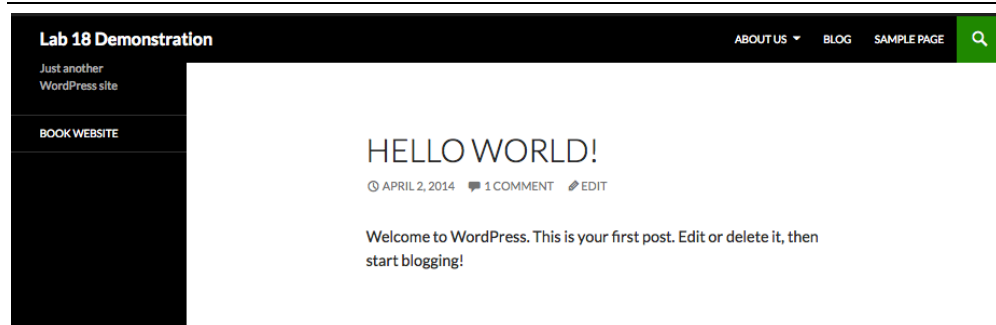


Figure 21.12 Screenshot of the cleaned up menu bars.

Exercise 21.4 — CREATE WORDPRESS USERS

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1.
- 2 Login to WordPress as an administrator and navigate to the users section of the portal. There you will see a list of all the users currently associated with this WordPress site. You likely only have 1 user so far as shown in Figure 21.13.

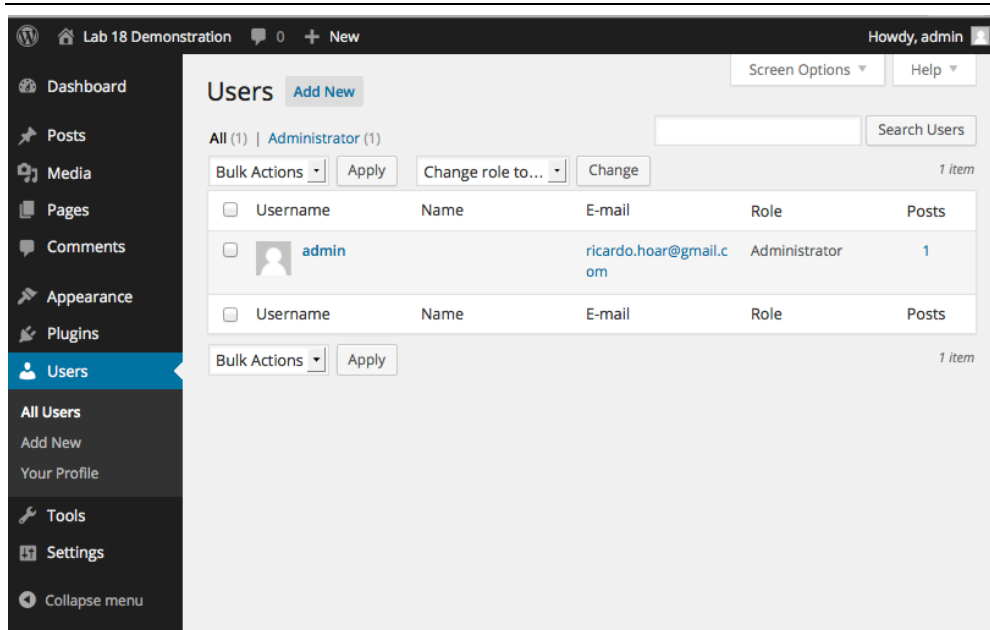


Figure 21.13 Users part of the admin interface.

- 3 Click on "add New" user and enter the new user's information. While some fields are required, others are optional (like a link to a website for each user). The components of this are later used in :user" pages by many themes,

That's it! You can choose different levels of privilege as required for your circumstance. And make as many users as you want!

- 4 Now click on "Your Profile" to set up specific options for your user. In addition to color themes, biographical data and an avatar in you can also choose whether to enable the toolbar when viewing the site (this slows down some installations) or the visual editor.

Exercise 21.5 — INSTALL A PLUGIN

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1. You will install a popular gallery plugin for showing galleries of images.
- 2 Login to WordPress as an administrator and click on Plugin-> Add New in the main left menu. You will be provided with a search facility as well as a word cloud of popular plugins.

Type "NextGEN Gallery" into the search bar and select "install" on the first item by Photocrati Media. You will be prompted with a popup whether you want to proceed.

- 3 You will now be presented with a page asking for your credentials to the server where WordPress is installed as shown in Figure 21.14.

Installing Plugin: NextGEN Gallery 2.0.61

Connection Information

To perform the requested action, WordPress needs to access your web server. Please enter your FTP credentials to proceed. If you do not remember your credentials, you should contact your web host.

Hostname

FTP Username

FTP Password

This password will not be stored on the server.

Connection Type ☒ FTP ☐ FTPS (SSL)

Figure 21.14 Interface to enter your login credentials and download the plugin.

If you are working on live domain with FTP access (increasingly rare) you can enter your credentials here, click proceed and the plugin will be downloaded to your WordPress installation location.

There are a great many reasons why this might not work including:

- Not working on a live domain
- No FTP access
- Uncomforted with where how credentials are being used

Thankfully, installing plugins is easily done without this interface. Visit the WordPress plugin page for the location of the zip file containing the entire plugin. In our case the URL for the plugin is

<http://downloads.wordpress.org/plugin/nextgen-gallery.zip>

That means that on your production or localhost machine you can download the plugin by going to the location of your WordPress installation and go to the plugins folder [/wp-content/plugins/](#)

From there you can download and extract the plugin with the commands

```
curl -O http://downloads.wordpress.org/plugin/nextgen-gallery.zip
unzip nextgen-gallery.zip
```

- 4 Now by navigating back to the list of plugins you will see the newly downloaded plugin listed as shown in Figure 21.15. Click Activate to make the plugin active in your site.

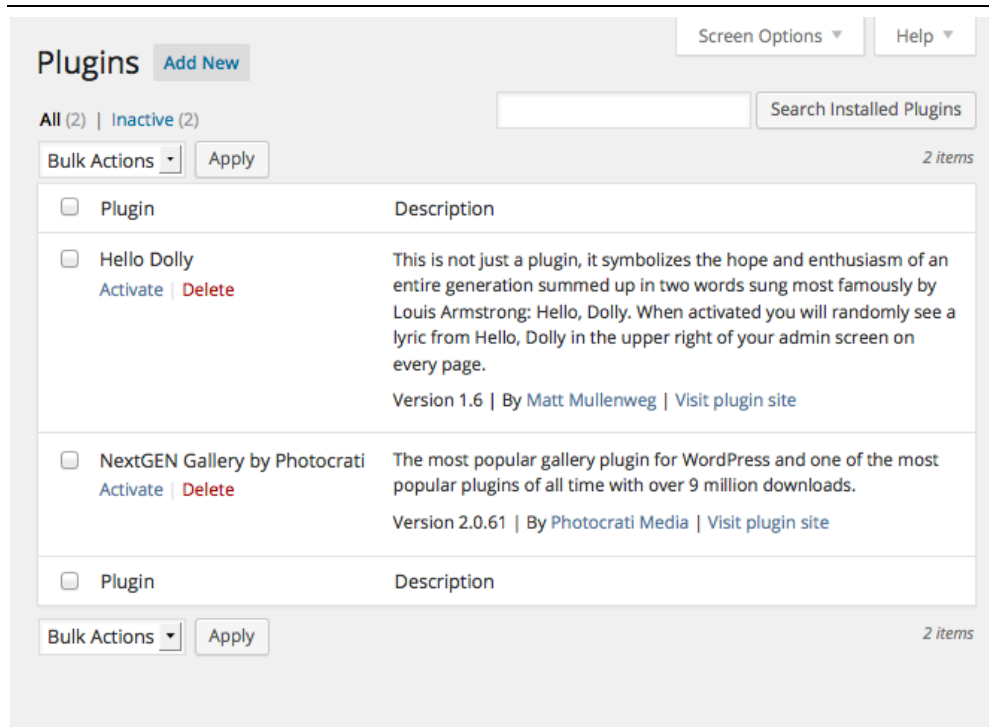


Figure 21.15 Screenshot of the interface showing the newly installed (but not active) plugin.

- 5 Upon clicking activate a new menu item will appear in your admin panel called "Gallery" (at the bottom). Click on that to start learning about how to create and manage galleries of images.
- 6 In the Appearance -> Widgets section of the interface (used to remove all the widgets in Exercise 21.3) drag the "NextGen Image Gallery" into the side bar to get a gallery as shown in Figure 21.16.

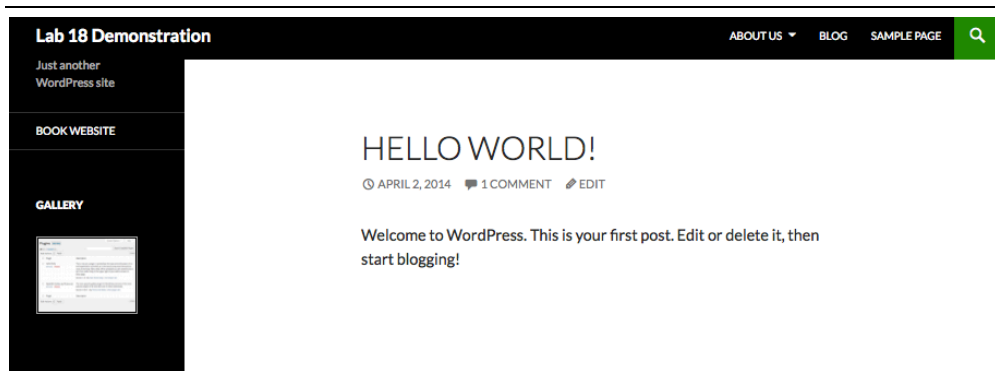


Figure 21.16 The NEXTGen plugin for image galleries included in the sidebar of our site.

Your site now has a professional looking gallery that can be used in multiple places. Read more from the site that created the plugin on how to customize it and use it elsewhere on your site.

Exercise 21.6 — DEFINE A CHILD THEME

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1. You will be modifying the default theme that comes with WordPress. By creating a child theme you maintain the original theme in a separate unmodified state so that it can easily be updated and your changes kept separate.
- 2 At the time of writing, the latest (and default) theme is **twentyseventeen**. In the [wp-content/themes/](#) folder you will see a folder named **twentyseventeen**, containing all the theme files. To define a child theme create a new folder named

twentyseventeen-child.

Inside of that folder you will define any files that override the files in the parent theme.

- 3 First copy over the screenshot (for now) so that it looks nice in the WordPress interface.

Now create a file named style.css and paste the following code into that file:

```
/*
Theme Name: Lab 21 Child Theme
Description: Proof of concept to illustrate child themes
Theme URI: http://funwebdev.com
Template: twentyseventeen
Version: 1.0.0
*/

@import url("../twentyseventeen/style.css");
```

Feel free to make changes to the name of the theme, url and description to suit your tastes.

- 4 Now when you return to the Appearance->Themes you will see your new child theme alongside the other themes (as shown in Figure 21.17) that came installed with WordPress. "Activate" the child theme and note that nothing looks any different.

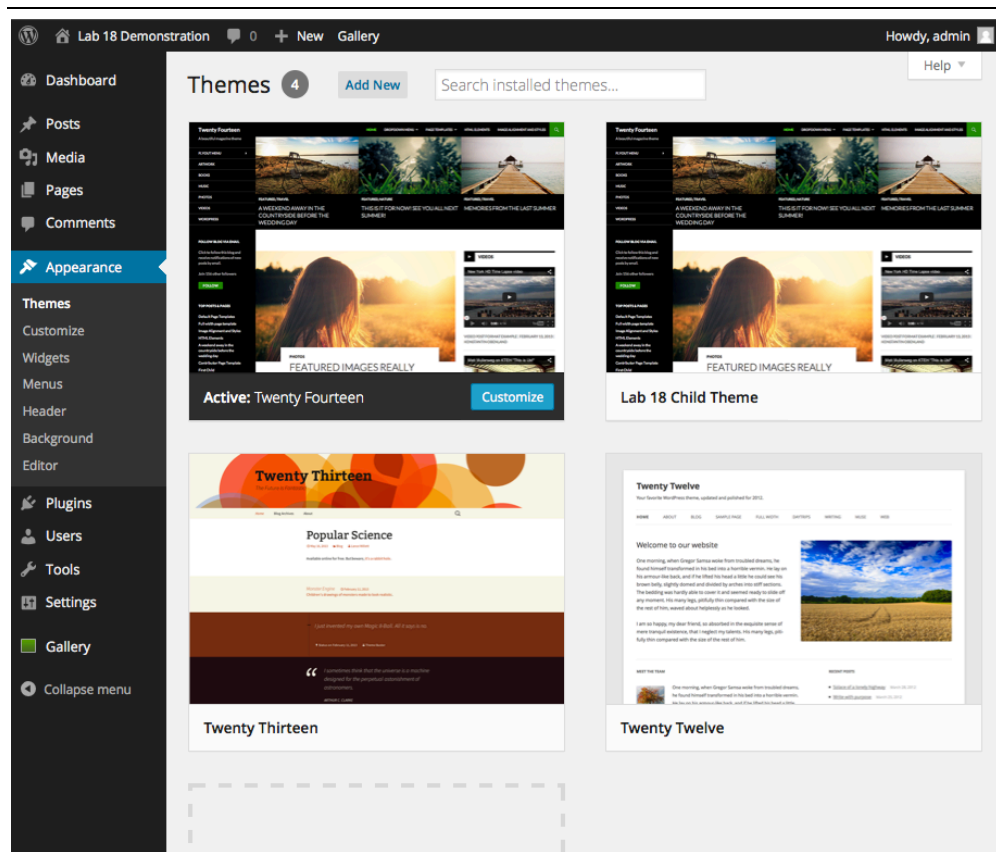


Figure 21.17 WordPress Theme selection interface showing newly defined child theme

- 5 Now, with the new theme selected, go into the style.css file you created and override several aspects of the CSS color scheme. In our case we added to the file:

```
.site-header, .site-footer, .site-sidebar {
    background-color: #777777;
}
#secondary {
    background-color: #777777;
}

.search-toggle, .search-toggle:hover, .search-toggle.active, .search-box {
    background-color: #A6412A;
}
```

- Now the site will have grey bars instead of black, and a red search area instead of a green one. To finalize this theme take a screenshot of it and replace the [screenshot.png](#) file with a more accurate screenshot so that your theme is distinct looking in the theme sector as shown in Figure 21.21.

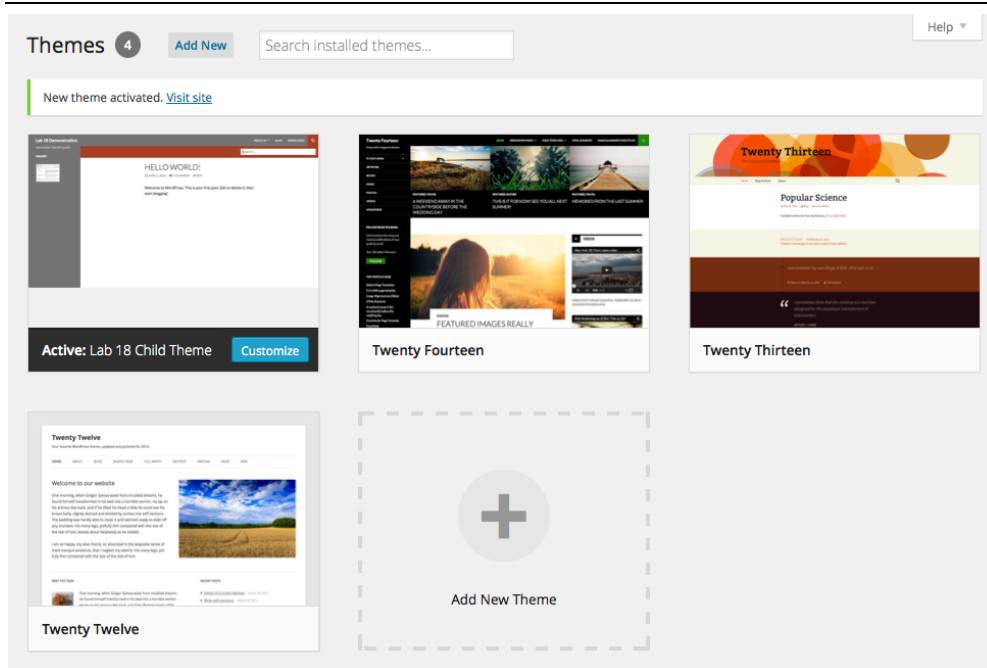


Figure 21.21 Theme with the updated screenshot to illustrate the different color scheme.

ADVANCED WORDPRESS CUSTOMIZATIONS

Exercise 21.7 — CUSTOM TEMPLATE PAGE

- This lab requires that you have a working WordPress installation as described in Exercise 21.1 and that you have created a child theme from Exercise 21.6. You will first modify a piece of a template and then create a custom template page.
- One of the most common changes from a default WordPress template is to change the content inside of the footer. In our case (and your too likely) the default footer says "Proudly powered by WordPress" with a link to wordpress.org.

The template used to make the footer in the parent theme is [footer.php](#). To make changes to that file in your child theme, first copy the existing [footer.php](#) from the parent theme's folder into your child theme folder.

- Open the file in a code editor and note the location of the Link to WordPress. Replace

that link with a link to a website of your choosing. In our case we will link to funwebdev.com so our modified `footer.php` looks like (with changes in red).

```
<?php
/**
 * The template for displaying the footer
 *
 * Contains footer content and the closing of the #main and #page div
 * elements.
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since Twenty Seventeen 1.0
 */

</div><!-- #main -->

<footer id="colophon" class="site-footer"
role="contentinfo">

    <?php get_sidebar( 'footer' ); ?>

    <div class="site-info">
        <?php do_action( 'twentyseventeen_credits'
    ); ?>
    WordPress Exercises part of
    <a href='http://funwebdev.com'>Fundamentals of Web Development</a>
    </div><!-- .site-info -->
</footer><!-- #colophon -->
</div><!-- #page -->

<?php wp_footer(); ?>
</body>
</html>
```

Note: most of the template file is left as is. This is purposeful since as a rule of thumb, when modifying the code of others, if you aren't sure what it does you should leave it alone. In this case it's pretty obvious that closing `<div>` tags are matching opening `<div>` tags in another template file. While one might be able to guess at the purpose and need for of the other function calls, for now we will leave them as is.

- 4 Another, slightly more advanced modification of a theme is to make your own kind of template file. In this case we will make a file that is almost like a regular Page, but includes an extra banner ad linking to our website. Feel free to enhance this exercise by making your template file somehow more different.

Create a subfolder called `page-templates` in your child theme folder.

Create a file named `sponsoredPage.php` and make it look similar to:

```
<?php
/**
 * Template Name: Sponsored Page
 * Description: Example Tempalte For Lab 21
```

```

*
* @package WordPress
* @subpackage Twenty_Seventeen
* @since Twenty Seventeen 1.0
*/

get_header(); ?>

<div id="main-content" class="main-content">

<?php
    if ( is_front_page() && twentyseventeen_has_featured_posts() ) {
        // Include the featured content template.
        get_template_part( 'featured-content' );
    }
?>
    <div id="primary" class="content-area">
        <div id="content" class="site-content" role="main">

            <?php
                // Start the Loop.
                while ( have_posts() ) : the_post();

                    // Include the page content
                    template.
                    get_template_part( 'content',
                    'page' );

                    //Add the sponsorship tag:
                    echo "<div class='site-content'><a href='http://funwebdev.com'><img
                    src='http://funwebdev.com/banner.png'></a></div>";

                    // If comments are open or we have at least one comment, load up the
                    comment template.
                    if ( comments_open() ||
get_comments_number() ) {
                        comments_template();
                    }
                    endwhile;
                ?>

                </div><!-- #content -->
            </div><!-- #primary -->
            <?php get_sidebar( 'content' ); ?>
        </div><!-- #main-content -->

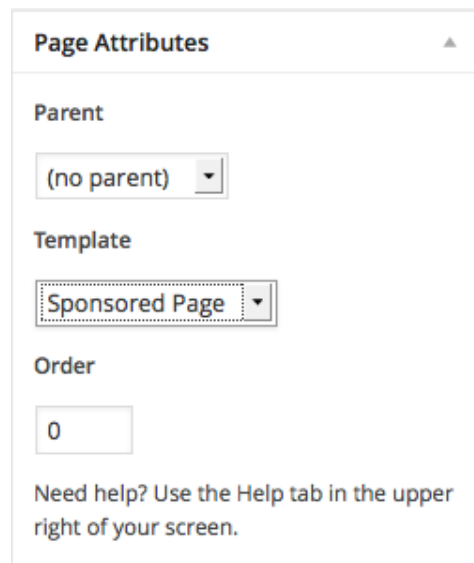
    <?php
    get_sidebar();
    get_footer();

```

Note: The initial code is all taken from the parent theme's page.php template file. Only minor modifications (in red) were made to get a new and different template type.

- 5 To see if this template is now installed and working correctly create a new Page (see Exercise 21.2), but this time select from the drop down menu the **Sponsored Page**

type we just defined as shown in Figure 21.19.



Page Attributes

Parent

(no parent) ▾

Template

Sponsored Page ▾

Order

0

Need help? Use the Help tab in the upper right of your screen.

Figure 21.19 Screenshot of the Page attributes with the new template being chosen

- 6 When the page is added and seen in a browser the content will appear in the menu just like any other but the page itself will include a banner ad below the content as shown in Figure 21.20.

Note: If you get a complete blank then you have an error and will have to investigate your apache logs for clues (WordPress does not echo out errors for security reasons).

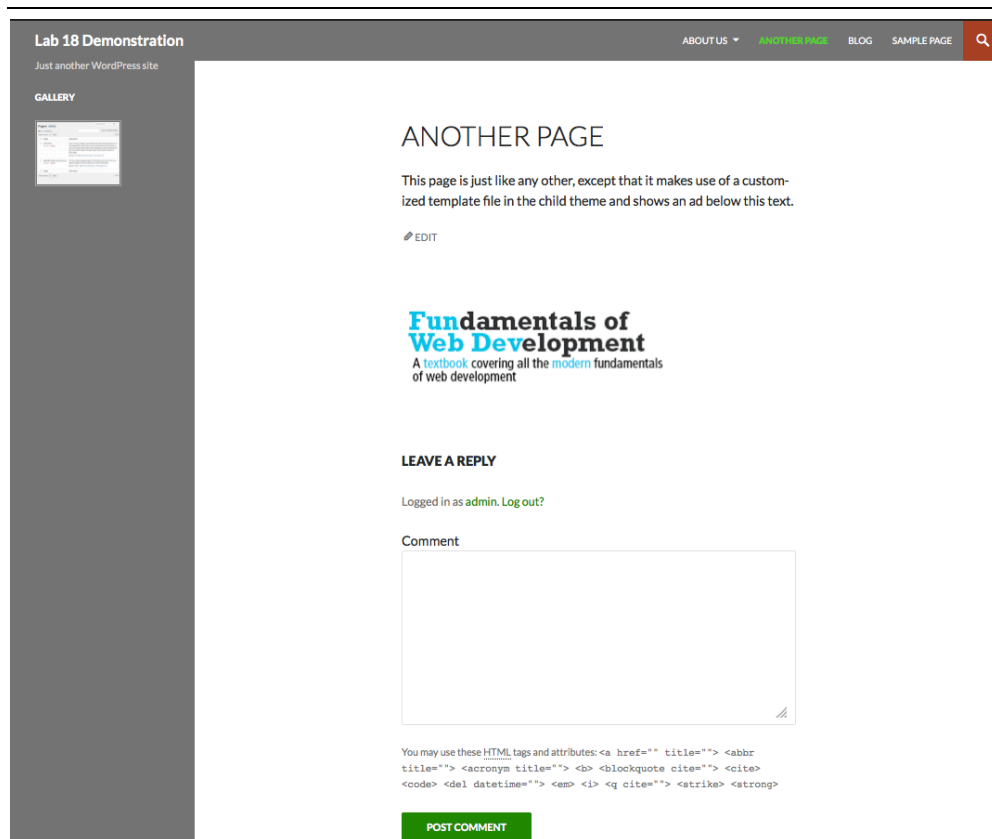


Figure 21.20 The new page in action with the Banner being shown below content.

You can image making custom template types for contact pages, maps, and web services. Many developers use templates to integrate their custom code with WordPress and never go beyond template files (into Widgets and Plugins). This is not recommended because sites often want to change themes to try a new look and feel, and all custom code in the theme would have to be ported to the new theme as well.

Exercise 21.8 — CUSTOM POST TYPE

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1. You will be creating a custom post type for a textbook as described in Chapter 21. Alternatively, consider creating a post type for something that you might have many of such as: Albums, Artworks, Bands, Books, Cities, etc...

- 2 To define the custom post type create a file in your child theme named `functions.php`. This file, unlike others, does not override the parent file, but supplements it. In `functions.php` write and register a function named `textbook_init()` as follows:

```
<?php
function textbook_init() {
$labels = array(
'name' => __('Textbooks'),
'singular_name' => __('Textbook'),
'add_new_item' => __("Add new Textbook"),
);
$args = array(
'labels' => $labels,
'description' => 'Holds textbooks',
'public' => true,
'supports' => array( 'title', 'editor', 'thumbnail',
'excerpt', 'comments' ),
'has_archive' => true,
);
register_post_type( 'textbook', $args );
}
add_action( 'init', 'textbook_init' );
```

Now in the admin interface (shown in Figure 21.21) you will see an extra section for "Textbooks" our new post type.

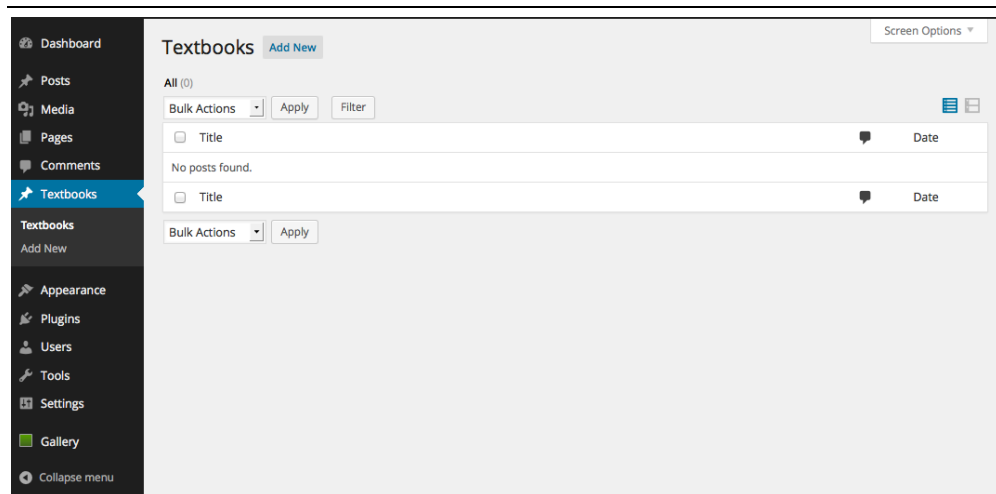


Figure 21.21 The Textbook section of the admin interface after writing `functions.php`

- 3 Now you will add some special fields that are associated with a textbook by adding more code to [functions.php](#). This time add to the file:

```
function textbook_admin_init() {
add_meta_box(
'textbook_details', // $id
'Textbook Details', // $title
'textbook_callback', // $callback
'textbook', // $post_type
'normal', // $context
'high' // $priority
);
function textbook_callback() {
global $post;
$custom = get_post_custom($post->ID);
$publisher = $custom['textbook_pub'][0]; // publisher
$author = $custom['textbook_author'][0]; // authors
$pub_date = $custom['textbook_date'][0]; //date
?>
Please enter the required details for a textbook here.
<div class="wrap">
<p><label>Publisher:</label><br />
<input name="textbook_pub" value="<?php echo $publisher; ?>" /></p>
<p><label>Author(s):</label><br />
<input name="textbook_author" value="<?php echo $author; ?>" /></p>
<p><label>Date:</label><br />
<input name="textbook_date" type="date"
value="<?php echo $pub_date; ?>" /></p>
</div>
<?php
}
}
// add function to put boxes on the 'edit textbook post' page
add_action( 'admin_init', 'textbook_admin_init' );
```

Now when you go to create a new Textbook you see the additional fields below the regular post section as shown in Figure 21.22.

Add new Textbook

Enter title here

Add Media Visual Text

B I [List Icons] [Quote Icon] [Link Icon] [Table Icon] [Image Icon]

Path: p

Word count: 0

Textbook Details

Please enter the required details for a textbook here.

Publisher:

Author(s):

Date:

Figure 21.22 The textbook editing interface with additional fields.

- 4 Finally you have to add code to actually save those extra fields to the database when the user saves a textbook post. Add the following code to [functions.php](#) to complete the Textbook Post type.

```
function textbook_save_data() {
    global $post;
    update_post_meta($post->ID, 'textbook_pub',
        $_POST['textbook_pub']);
    update_post_meta($post->ID, 'textbook_author',
        $_POST['textbook_author']);
    update_post_meta($post->ID, 'textbook_date',
        $_POST['textbook_date']);
}
// attach your function
```

```
add_action('save_post', 'textbook_save_data');
```

- 5 In the admin interface, now create and publish a post for your favorite textbook. You should be able to see the extra fields in the admin interface, but if you go to "see post" it will not display any of the extra fields. To make the post display properly we need to create template files specifically for a textbook that make use of the extra fields.

Exercise 21.9 — DISPLAY A POST

- 1 This exercise builds on the previous one. Although our custom post type is working it is not being displayed.
- 2 Copy the parent theme's [single.php](#) file into your child theme and rename it [single-textbook.php](#). Now make the following modification (in red) to make the textbook information appear in the post after the regular post (with title and content):

```
<?php
/**
 * The Template for displaying all single posts
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since Twenty Seventeen 1.0
 */

get_header(); ?>

<div id="primary" class="content-area">
    <div id="content" class="site-content" role="main">
        <?php

                // Start the Loop.
                while ( have_posts() ) : the_post();
                    get_template_part( 'content', get_post_format() );

global $post;

$custom = get_post_custom($post->ID);
$author = $custom['textbook_author'][0];           //authors
$pubdate = $custom['textbook_date'][0]; //date
$publisher = $custom['textbook_pub'][0]; //publisher
?>
    <div class='author entry-content'>
By: <?php echo $author."<br>";
    echo "Published by: ".$publisher." (".$pubdate.)"; ?></div>
```

```

<?php
                                // Previous/next post navigation.
                                twentyseventeen_post_nav();

                                // If comments are open or we have
                                // at least one comment, load up the comment template.
                                if ( comments_open() ||
get_comments_number() ) {
                                    comments_template();
                                }
                                endwhile;
                                ?>
                                </div><!-- #content -->
                                </div><!-- #primary -->

<?php
get_sidebar( 'content' );
get_sidebar();
get_footer();

```

After you save this file refreshing the page on your site should output the post with the extra fields as shown in Figure 21.23.

FUNDAMENTALS OF WEB DEVELOPMENT

LEAVE A COMMENT EDIT

By: Randy Connolly, Ricardo Hoar
Published by: Pearson (Feb 28, 2014)

LEAVE A REPLY

Logged in as [admin](#). [Log out?](#)

Comment

You may use these HTML tags and attributes: <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike>

POST COMMENT

Figure 21.23 Screenshot of the textbook with the template file completed.

- To complete this custom post type you must also create a file, based on the [archive.php](#) template to display an archive of textbooks. That file should be named [archive-textbook.php](#) in your child theme and be modified as follows (in red):

...

```
get_template_part( 'content', get_post_format() );
$custom = get_post_custom($post->ID);
$author = $custom['textbook_author'][0]; //authors
$pubdate = $custom['textbook_date'][0]; //date
$publisher = $custom['textbook_pub'][0]; //publisher

echo "<div class='author entry-content'>";
echo 'By:'. $author. " Published by ". $publisher. " (". $pubdate. ") </div>";
endwhile;
```

...

You can check if this file is working by surfing to `/textbooks/` off the main WordPress part of your site (turn permalinks on for this).

Exercise 21.10 — WRITE A PLUGIN

- 1 This lab requires that you have a working WordPress installation as described in Exercise 21.1. In theory you do not need to the textbook post type, and in fact should reset to use the parent theme where those custom post types are not defined.
- 2 In the WordPress `/wp-content/plugins/` directory create a new folder to hold our textbook plugin. Create a file named `index.php` and paste the following block of comment at the top:

```
/*
Plugin Name: TextBook Plugin (funwebdev)
Description: Allows for management of textbooks
Version: 1.0
Author: Ricardo Hoar
License: GPL2
*/
```

Now paste all the code written in Exercise 8.9's `functions.php` into the file and save. If you go into your plugins folder you will be able to activate and deactivate the new plugin, just like any other as shown in Figure 21.24.

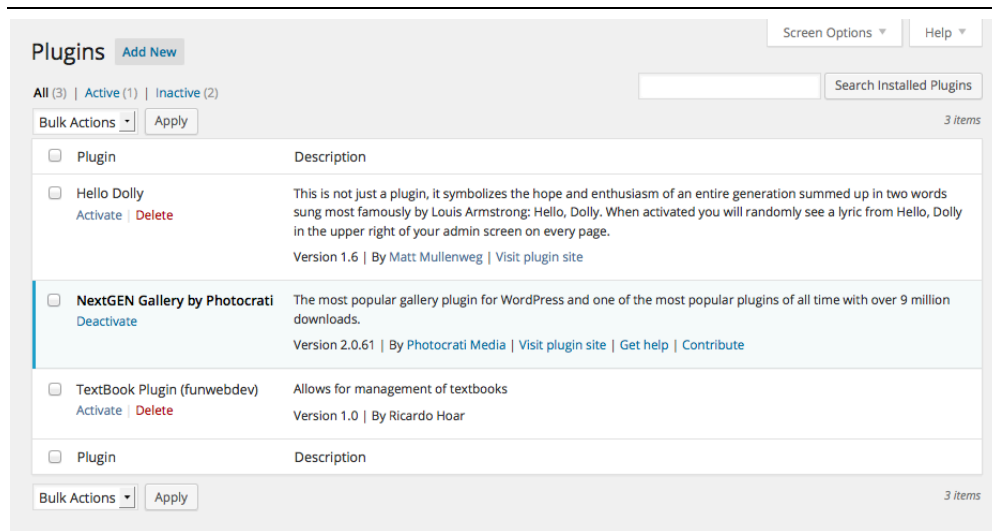


Figure 21.24 Seeing the plugin alongside other plugins in the admin interface

- 3 Now, since our rendering code was written in the child theme, we have to move the rendering code into our `index.php` file. In particular paste the following code which attaches some rendering code to a particular filter.

```
function textbook_content_display($content) {
    global $post;
    //check for the custom post type
    if (get_post_type() != "textbook") {
        return $content;
    }
    else {
        $custom = get_post_custom($post->ID);
        $newContent='<div class="title">'. get_the_title($post->ID).
        '</div>';
        $author = $custom['textbook_author'][0]; //authors
        $pubdate = $custom['textbook_date'][0]; //date
        $newContent .= '<div class="author"> By:' . $author;
        $newContent .= '(' . $pubdate . ')</div>';
        $newContent .= '<div class="content">' . $content . '</div>';
        return $newContent;
    }
}
add_filter('the_content','textbook_content_display');
```

- 4 Now, just as with the custom post, you have a custom type of post item, but it can be used with any theme. Just to confirm how much more powerful a plugin is , from a custom type in a theme, try changing themes, and note that the Textbooks still work (with a custom post, the custom post breaks when you change theme).

Exercise 21.11 — DEFINE A WIDGET

- 1 This ultimate lab in WordPress converts your newly defined plugin from Exercise 21.10 into a widget that can be manipulated just like other widgets, in the sidebar etc.
- 2 Add to the index.php for your textbook plugin code to define the widget as follows:


```
function textbook_widget_display($args) {
    echo $before_widget;
    echo $before_title . '<h2>Random Book</h2>' . $after_title;
    echo $after_widget;
    $args = array(
        'posts_per_page' => 1,
        'post_type' => array('textbook'),
        'orderby' => "rand"
    );
    $bookQuery = new WP_Query();
    $bookQuery->query($args);
    while ( $bookQuery->have_posts() ) : $bookQuery->the_post();
        the_content();
    endwhile;
}
// Register
wp_register_sidebar_widget(
    'funwebdev_textbook_widget',// unique widget id
    'Random Textbook', // widget name
    'textbook_widget_display', // callback function
    array( // options
        'description' => 'Displays a random Textbook'
    )
);
```
- 3 Now go to the Appearance->Widgets section of your panel and look for the Random Textbook widget. Drag it into the sidebar to make it active on your site as shown in Figure 21.25

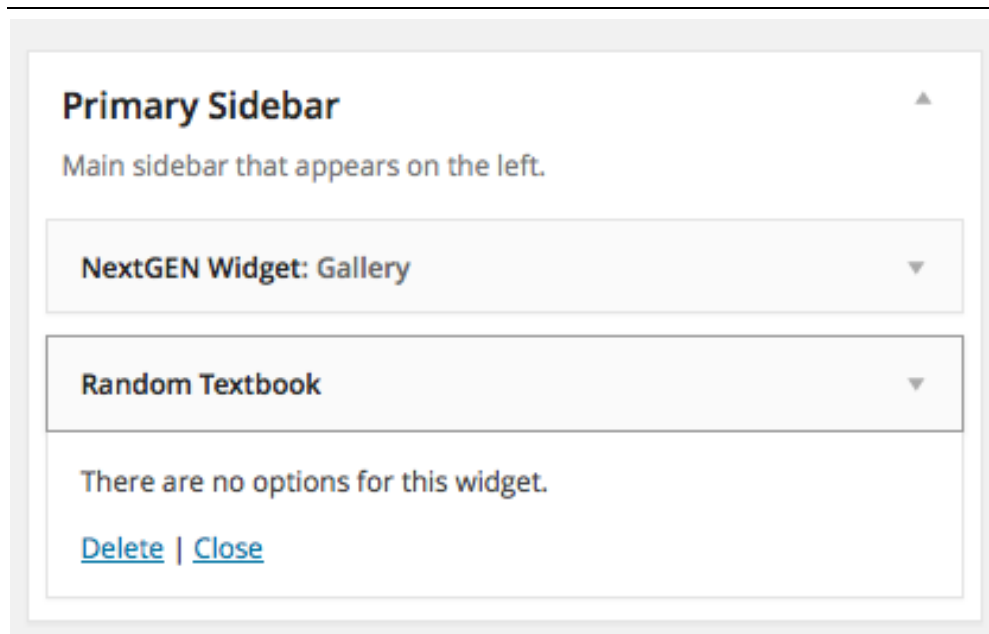


Figure 21,25 The Widget being added to the sidebar in the admin interface.

- 4 Surf to the site and you should see a "random textbook" being displayed in the sidebar. Your widget is complete, drawing from your plugin which adds a custom type to any theme. The site now has a 3rd party plugin (or many) and makes use of a custom plugin and widget to add a random book to the sidebar as shown in Figure 21,26



Figure 21,26 A screenshot of the Widget in action.
