

# Content Management Systems

## Chapter 18

# Objectives

**1** Managing websites

**2** Content Management Systems (**CMS**)

**3** CMS **Components**

**4** **WordPress** Technical Overview

**5** Modifying **Themes**

**6** Customizing WordPress **Templates**

**7** Creating a **Custom Post Type**

**8** Writing a **Plugin**

---

Section 1 of 8

# MANAGING WEBSITES

# Managing Websites

The most significant drawback to the sites you have created so far is that these sites require a software developer to edit the code in order to make changes

For a small company, this can be a significant problem, since they cannot afford a full-time programmer on staff.

These companies want a system that is

- Easy for a nontechnical person to make changes to
- Consistent and professional looking across the site
- Cost effective

# Managing Websites

## Components of a Managed Website

A typical website requires :

**Management** provides a mechanism for uploading and managing images, documents, videos, and other assets.

**Menu control** manages the menus on a site and links menu items to particular pages.

**Search functionality** can be built into systems so that users can search the entire website.

**Template management** allows the structure of the site to be edited and then applied to all pages.

# Managing Websites

## Components of a Managed Website

**User management** permits multiple authors to work simultaneously and attribute changes to the appropriate individual. It can also restrict permissions.

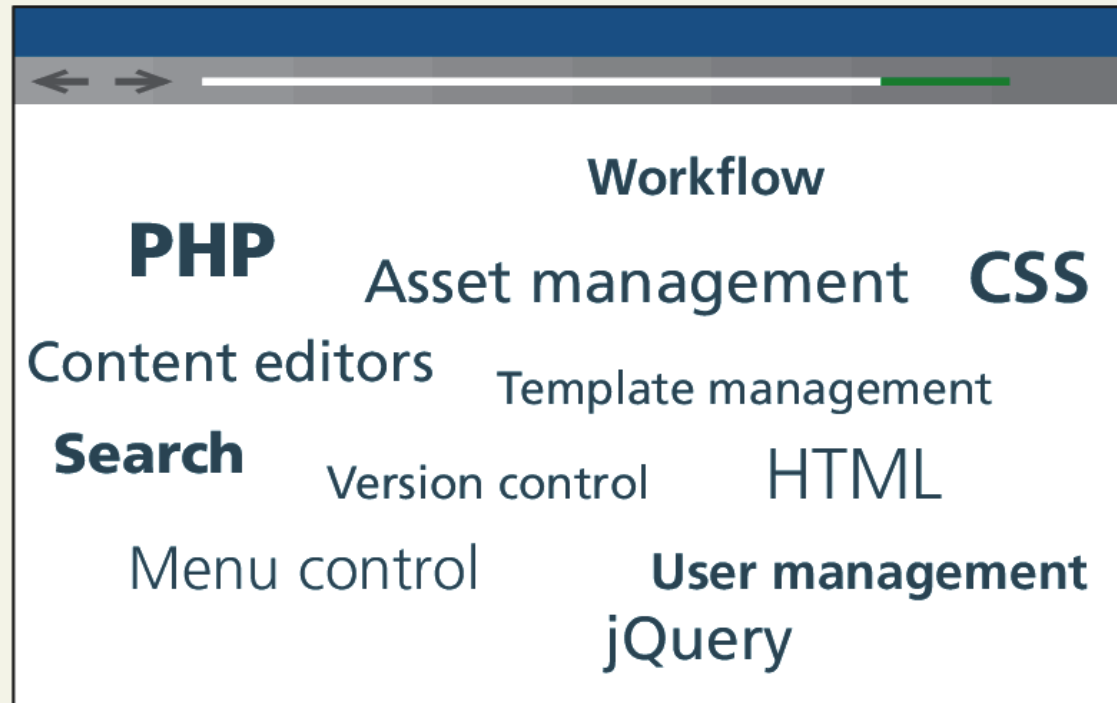
**Version control** tracks the changes in the site over time.

**Workflow** defines the process of approval for publishing content.

**WYSIWYG editor** allows nontechnical users to create and edit HTML content and CSS styles without manipulating code.

# Content Management Systems

Try to make it manageable



**CMS**

Section 2 of 8

# **CONTENT MANAGEMENT SYSTEMS (CMS)**



# Content Management Systems

Try to make it manageable

**Content management system (CMS)** is the name given to the category of software that easily manages websites with support for multiple users.

**CMS** is a **Document management systems (DMSs)** which has many features for documents including: file storage, multiuser workflows, versioning, searching, user management, publication, and others.

# Types of CMS

There are a lot

Around 109 open-source systems & 39 proprietary systems. We (Edinboro) used dotCMS for our old web site.

These systems are implemented using a wide range of development technologies including PHP, ASP.NET, Java, Ruby, Python, and others.

Some of these systems are free, while others can cost hundreds of thousands of dollars.

# Types of CMS

There are a lot

Other CMSs include:

- **DotNetNuke**
- **Drupal**
- **ExpressionEngine**
- **IBM Enterprise Content Management (ECM)**
- **Joomla!**
- **Moodle** ← I think of this as a LMS
- **Sharepoint**

# Types of CMS

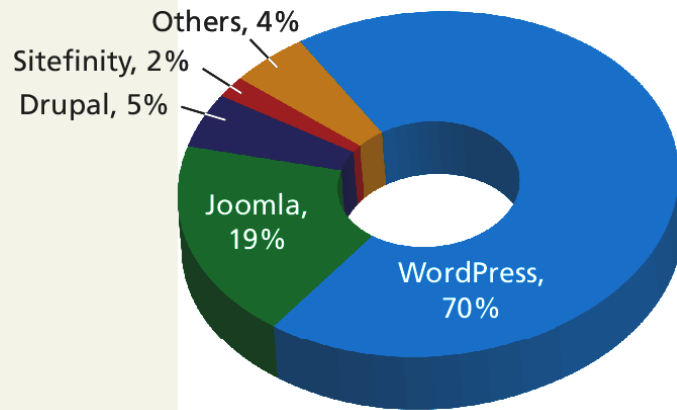
How to pick

When selecting a CMS there are several factors to consider including:

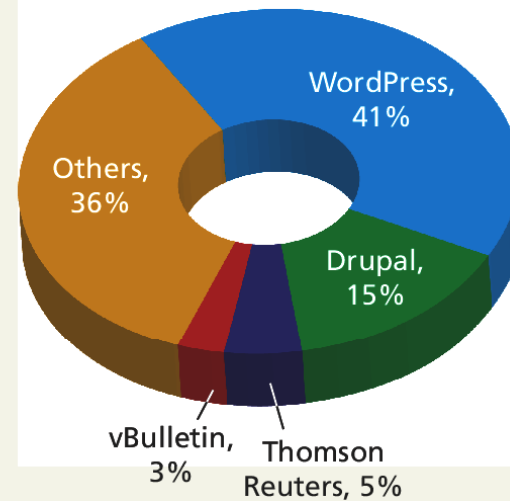
- **Technical requirements:** the functionality it offers as well as the server software and database compatibility. Your client may have additional requirements to consider.
- **System support:** Since you are going to rely on the CMS to patch bugs and add new features, it's important that the CMS community be active in supporting these types of updates or you will be at risk of attack.
- **Ease of use:** Probably the most important consideration is that the system itself must be easy to use by nontechnical staff.

# Popularity of CMS

Top 9,000,000 Sites



Top 10,000 Sites



Start

Section 3 of 8

# CMS COMPONENTS

# CMS Components

- Post and Page Management
- WYSIWYG Editors
- Template Management
- Menu Control
- User Management and Roles
- User Roles
- Workflow and Version Control
- Asset Management
- Search
- Upgrades and Updates ← modules you can download

# CMS Components

## Post and Page Management

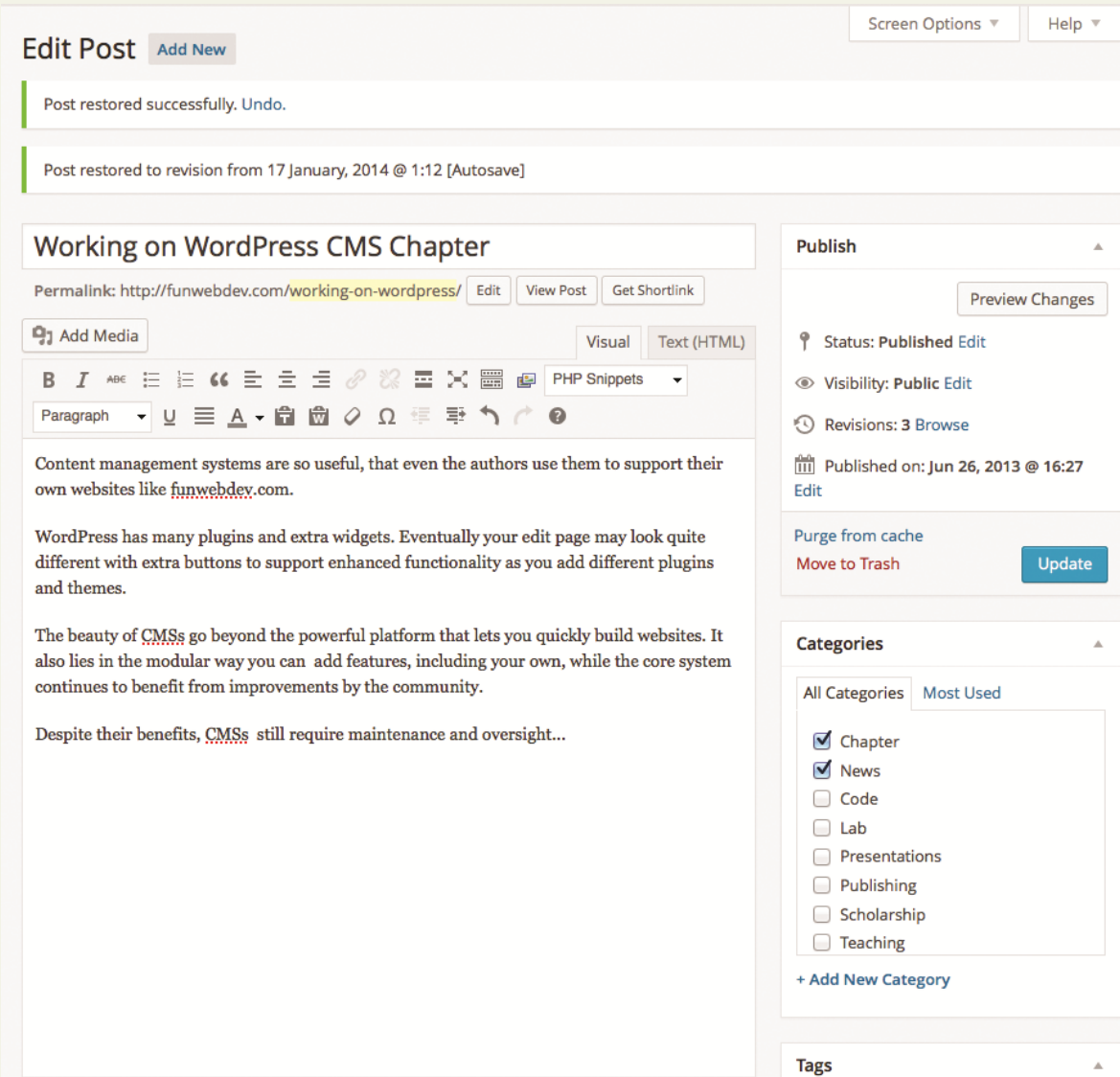
Blogging environments such as WordPress use **posts** as one important way of adding content to the site.

Posts are usually displayed in reverse chronological order (i.e., most recent first) and are typically assigned to categories or tagged with keywords as a way of organizing them.

Many sites allow users to comment on posts as well.



## Screenshot of a Post Editor in WordPress



My Workbench | BoroOnline x

https://boroonline.cs.edinboro.edu/users/dtucker#overlay=admin/workbench

Do you want Google Chrome to save your password? Save password Never for this site

My Workbench Dashboard Content Structure Appearance People Modules Configuration Reports Help Hello dtucker Log out

Add content Find content Blocks Views Performance Edit shortcuts


My Workbench MY CONTENT CREATE CONTENT MY SECTIONS MY DRAFTS NEEDS REVIEW

Home » Administration My Edits All Recent Content

There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

There are security updates available for one or more of your modules or themes. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

**My Profile**



dtucker  
edit my profile

**My Edits**

Five of the most recently updated pieces of content.

TITLE	SECTION	TYPE	PUBLISHED	LAST UPDATED
<a href="#">LinksHome</a>		LinksHome	No	9 months 2 days ago

[view all](#)

**All Recent Content**

TITLE	SECTION	TYPE	AUTHOR	LAST UPDATED	ACTIONS
<a href="#">Women's soccer announces 2015 recruits</a>	ArticleCL publisher	CLArticleSports	clantinen	11 hours 25 min ago	<a href="#">edit</a>
<a href="#">Men, women split weekend matches in West Virginia</a>	ArticleCL publisher	CLArticleSports	clantinen	11 hours 33 min ago	<a href="#">edit</a>
<a href="#">'Murderer's Row' gave new life to Edinboro wrestling</a>	ArticleCL publisher	CLArticleSports	clantinen	11 hours 37 min ago	<a href="#">edit</a>
<a href="#">Third-place team finish at nationals caps best season in Boro wrestling history</a>	ArticleCL publisher	CLArticleSports	clantinen	11 hours 43 min ago	<a href="#">edit</a>
<a href="#">...</a>	ArticleCL	CLArticleEntertainment	clantinen	12 hours 24 min ago	<a href="#">edit</a>

https://boroonline.cs.edinboro.edu/admin/workbench?render=overlay#

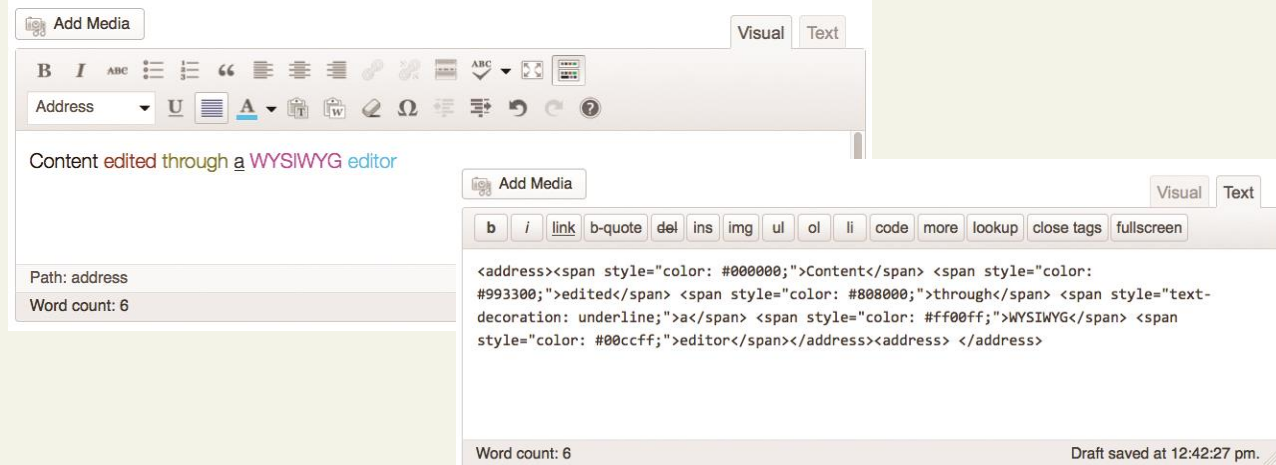
Check out the boroonline drupal site  
<https://boroonline.cs.edinboro.edu/user>

# CMS Components

What You See is What You Get

**What You See Is What You Get (WYSIWYG)** presents the users with an exact (or close) view of what the final product will look like.

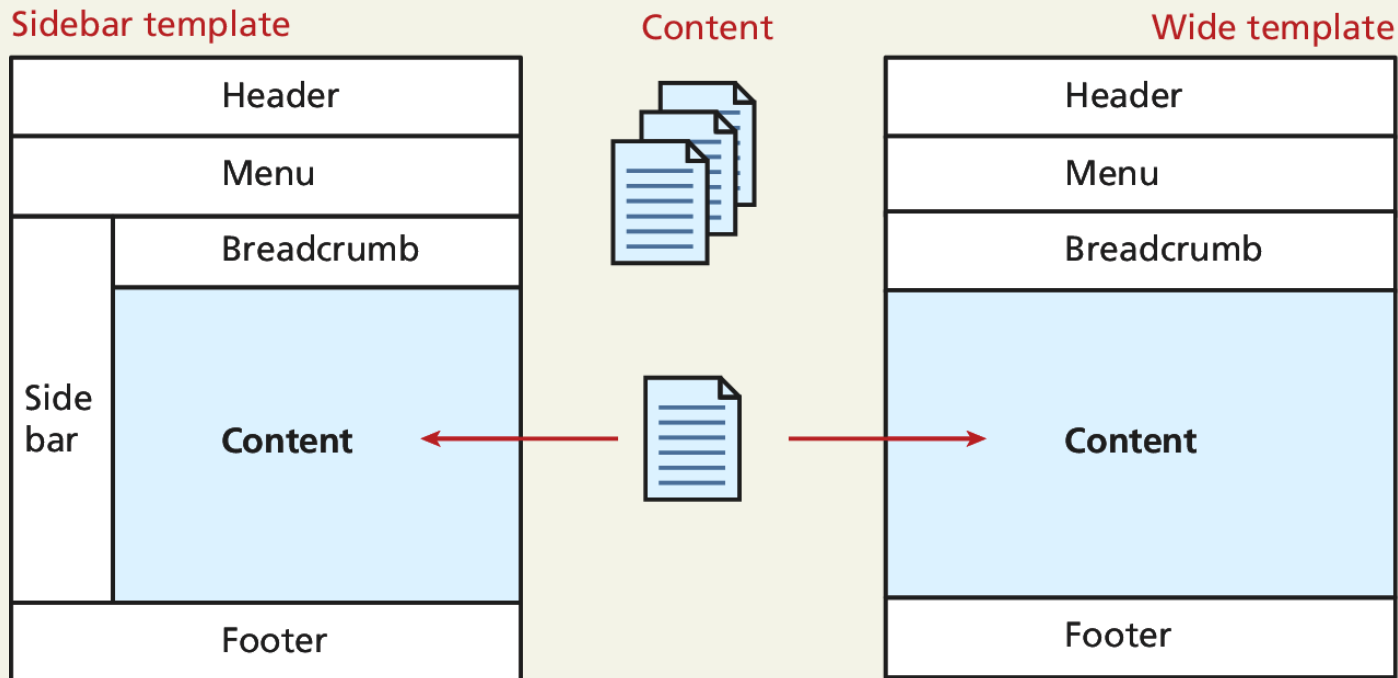
These tools generate HTML and CSS automatically through intuitive user interfaces



# CMS Components

## Template Management

**Template management** refers to the systems that manage the structure of a website, independently of the content of each particular page



# CMS Components

## Menu Control

Some key pieces of functionality that should be supported in the **menu control** include:

- Rearrange menu items and their hierarchy.
- Change the destination page or URL for any menu item.
- Add, edit, or remove menu items.
- Change the style and look/feel of the menu in one place.
- Manage short URLs associated with each menu item.

# CMS Components

## User Management

**User management** refers to a system's ability to have many users all working together on the same website simultaneously.

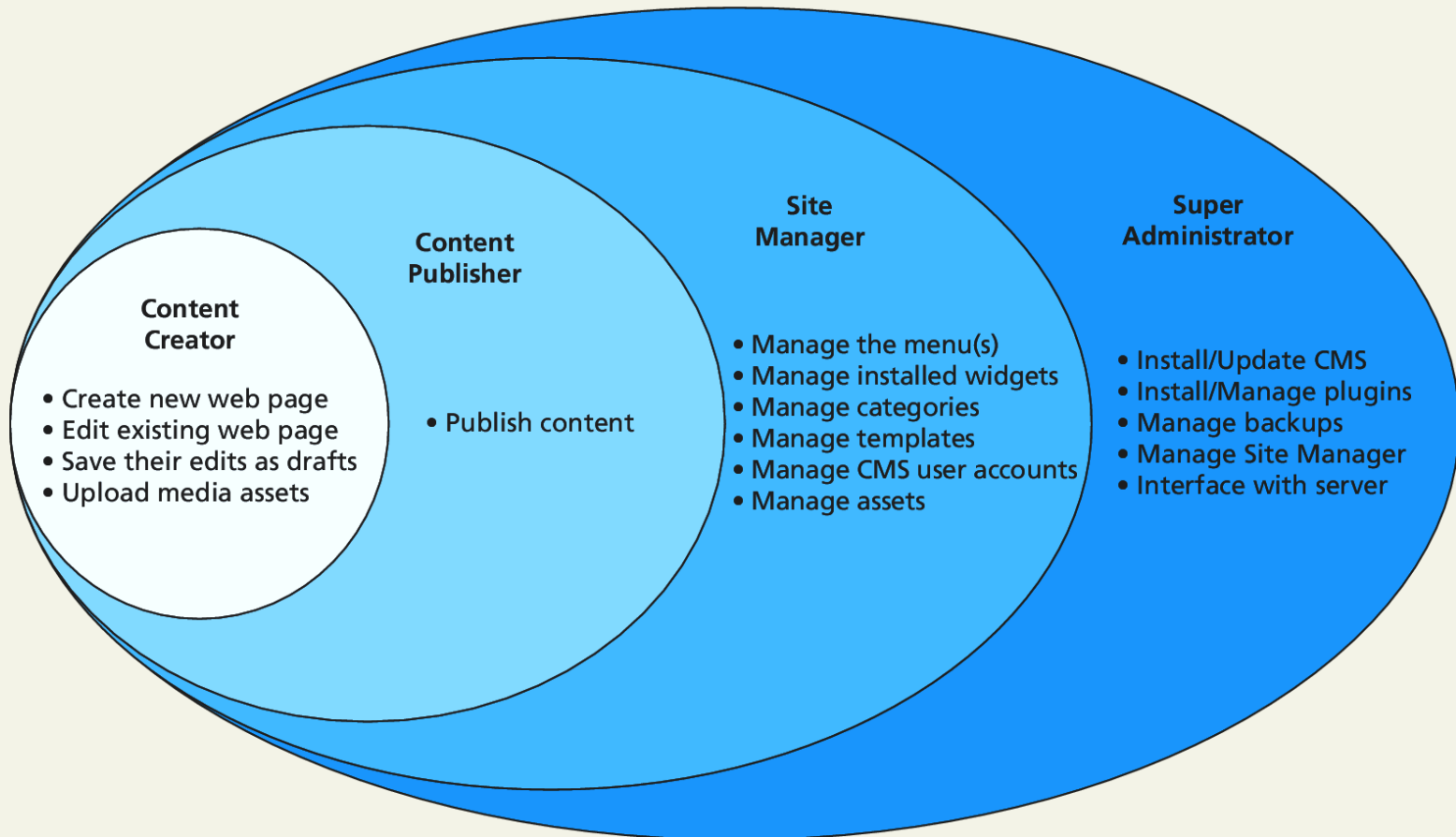
A CMS that includes user management must support:

- Adding a new user
- Resetting a user password
- Allowing users to recover their own passwords
- Allowing users to manage their own profiles, including name, avatars, and email addresses
- Tracking logins

# CMS Components

## User Roles

Users in a CMS are given a **user role**, which specifies which rights and privileges that user has.

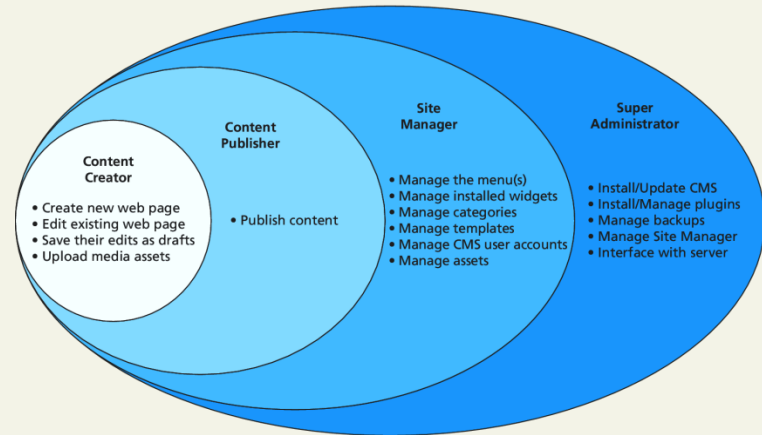


# User Roles

An important CMS Component

## Content Creator

- Create new web pages
- Edit existing web pages
- Save their edits in a draft form
- Upload media assets such as images and videos



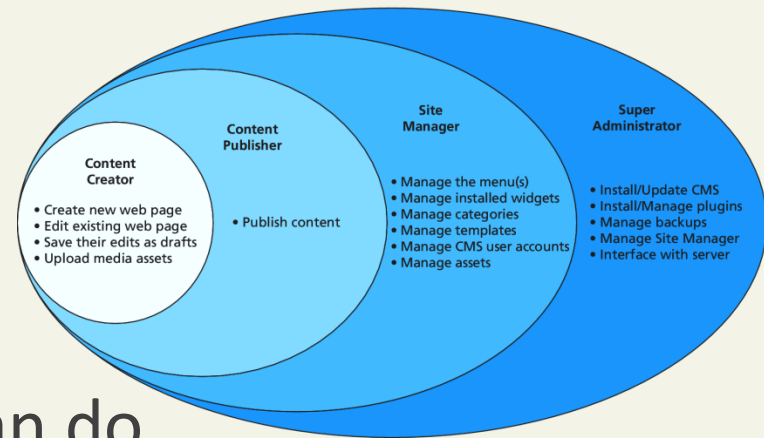


# User Roles

An important CMS Component

## Content Publisher

- Everything a creator can do
- determine if a submitted piece of content should be published
- they can publish immediately

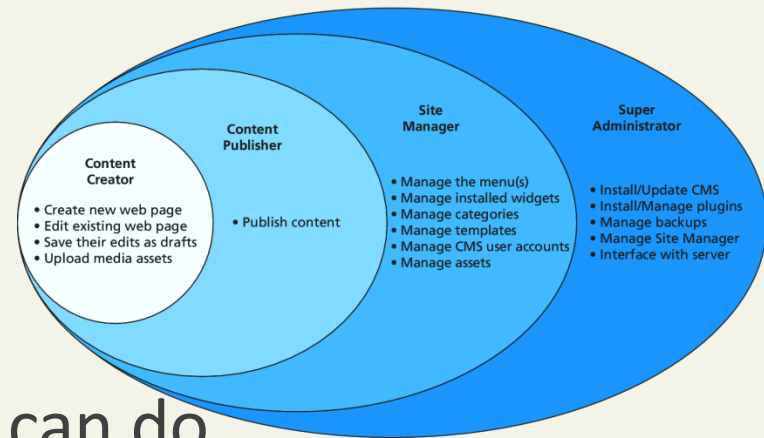


# User Roles

An important CMS Component

## Site Manager

- Everything a publisher can do
- Menu management
- Management of installed plugins and widgets
- Category and template management
- CMS user account management
- Asset management

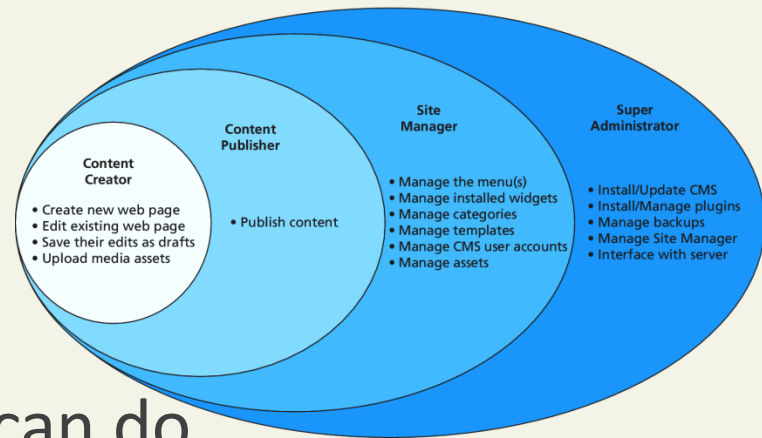


# User Roles

An important CMS Component

## Super Administrator

- Everything a manager can do
- Managing the backup strategy for the site
- Creating/deleting CMS site manager accounts
- Keeping the CMS up to date
- Managing plugin and template installation



# WordPress Roles

An implementation of Roles

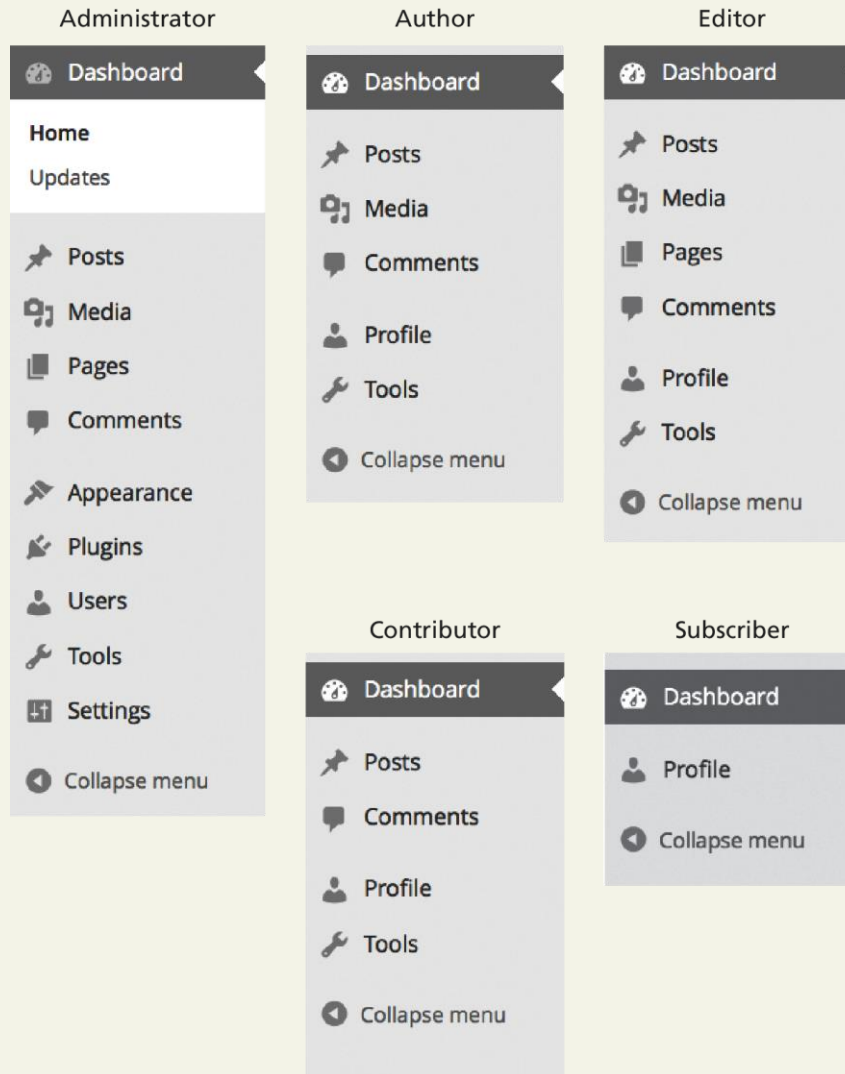
In WordPress the default roles are

- Administrator
- Author
- Editor
- Contributor
- Subscriber

Similar to Drupal

# WordPress Roles

The change in dashboard options with Roles



# CMS Components

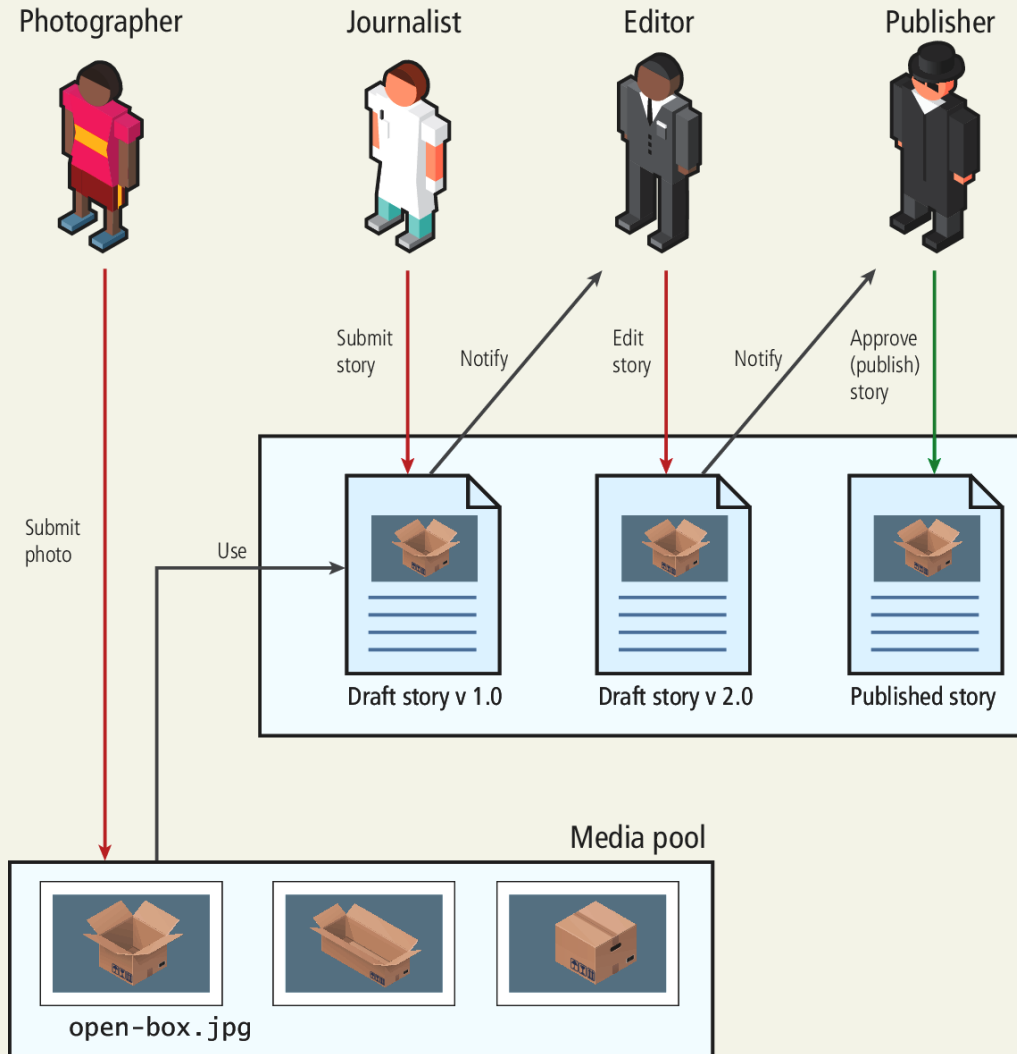
## Workflow and Version Control

**Workflow** refers to the process of approval for publishing content.

It is best understood by considering the way that journalists and editors work together at a newspaper.

# CMS Components

## Workflow and Version Control



# CMS Components

## Asset Management


**Asset management** software enables the user to:

- Import new assets
- Edit the metadata associated with assets
- Delete assets
- Browse assets for inclusion in content
- Perform searches or apply filters to find assets



# CMS Components










## Asset Management Interfaces in WordPress

 **Media Library** [Add New](#)

**All** (93) | **Images** (93) | **Unattached** (92)

**Bulk Actions** ▼ [Apply](#) **Show all dates** ▼ [Filter](#)







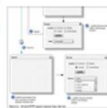





93 items « 1 of 5 »


<input type="checkbox"/>	File	Author	Uploaded to		Date
<input type="checkbox"/>	 <b>Chapter-6-banner</b> JPG	randy	(Unattached) <a href="#">Attach</a>		2013/03/03
<input type="checkbox"/>	 <b>Chapter-1-banner</b> JPG	randy	(Unattached) <a href="#">Attach</a>		2013/03/03
<input type="checkbox"/>	 <b>Chapter-06-41</b> JPG	randy	(Unattached) <a href="#">Attach</a>		2013/03/01
<input type="checkbox"/>	 <b>Chapter-06-25</b> JPG	randy	(Unattached) <a href="#">Attach</a>		2013/03/01

**Insert Media** ✕


[Upload Files](#) | [Media Library](#)

**Images** ▼



**1 selected** [Clear](#) 

**ATTACHMENT DETAILS**

 **Chapter-06-18.jpg**  
March 1, 2013  
755 x 1000  
[Edit Image](#)  
[Delete Permanently](#)

Title

Caption

[Insert into page](#)

# CMS Components

## Search

Searching has become a core way that users navigate websites.

There are three strategies to do website search:

- SQL queries using LIKE
- third-party search engines
- search indexes.

# CMS Components

## Upgrades and Updates

The screenshot shows the WordPress 3.5.2 dashboard. A yellow notification bar at the top states "WordPress 3.7 is available! Please update now." The left sidebar contains a menu with "Updates" and "Plugins" circled in red. The main content area includes a "Right Now" summary, a "QuickPress" widget, "Recent Drafts", "WordPress Blog" news, and "Other WordPress News".

**Right Now Summary:**

Content	Discussion
12 Posts	5 Comments
32 Pages	0 Approved
11 Categories	5 Pending
0 Tags	0 Spam

**Theme:** Superior Osaka WordPress Theme with 10 Widgets  
You are using WordPress 3.5.2. [Update to 3.7](#)

**QuickPress:** Enter title here, Add Media, Tags (separate with commas), Save Draft, Reset, Publish

**Recent Drafts:** There are no drafts at the moment

**WordPress Blog:**  
WordPress 3.7 "Basie" October 24, 2013  
Version 3.7 of WordPress, named "Basie" in honor of Count Basie, is available for download or update in your WordPress dashboard. This release features some of the most important architectural updates we've made to date. Here are the big ones: Updates while you sleep. With WordPress 3.7, you don't have to lift a finger to [...] [...]  
WordPress 3.7 Release Candidate #2 October 23, 2013  
The second release candidate of WordPress 3.7 is now available for testing! Those of you already testing WordPress 3.7 will be updated automatically to RC2. (Nice.) If you'd like to start testing, there's no time like the present! Try the WordPress Beta Tester plugin (you'll want "bleeding edge nightlies") or download the release candidate here (zip). Please [...]

**Other WordPress News:**  
WPTavern: How to Install WordPress Plugins Directly From Github  
WPTavern: WordPress Automatic Updates – No Options For You!  
WPTavern: Changing The WordPress Admin Username During Installation  
WPTavern: How To Find Live Examples Of Sites Using A WordPress Theme  
WPTavern: WordPress Core Developers Release Plugin To Test Automatic Background Updates

Thank you for creating with WordPress. [Get Version 3.7](#)



Section 4 of 8

# **WORDPRESS TECHNICAL OVERVIEW**

# WordPress Installation

5 minute install

WordPress proudly boasts that it can be installed in five minutes. The five-minute installation has only four steps:

1. Download and unzip WordPress.
2. Create a database on your server and a MySQL user with permissions to that database.
3. Move the unzipped files to the location on your server you want to host from
4. Run the install script by visiting the URL associated with that folder in a browser and answering several questions (about the site generally and connecting to the database).

# WordPress Installation

<5 minute install

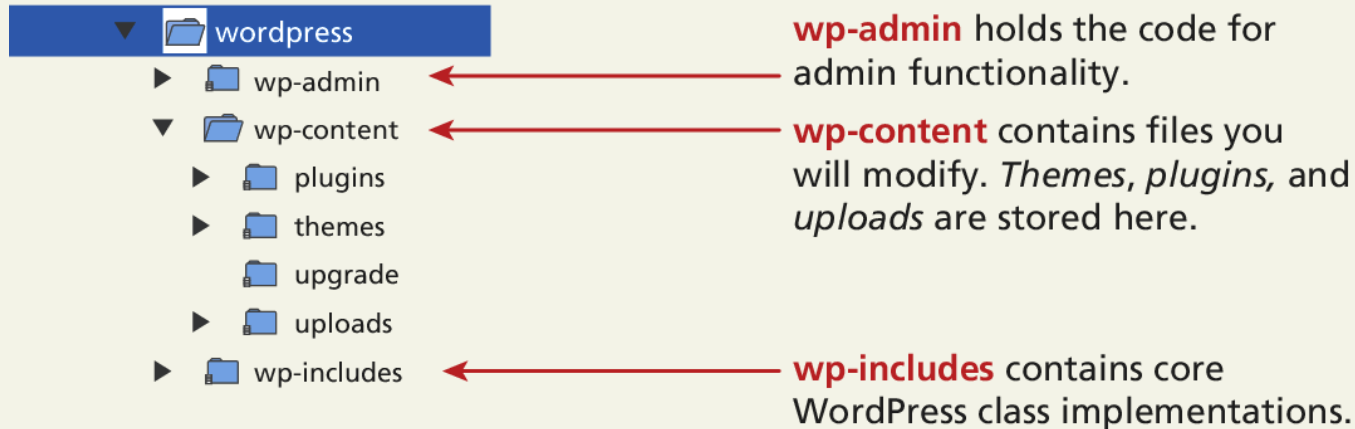
The file **wp-config.php** allows you to set all the values asked about in the interactive installation

```
/** The name of the database for WordPress */  
define('DB_NAME', 'ArtDatabase');  
/** MySQL database username */  
define('DB_USER', 'WordPressUser');  
/** MySQL database password */  
define('DB_PASSWORD', 'password');  
/** MySQL hostname */  
define('DB_HOST', 'localhost');
```

**LISTING 18.1** wp-config.php file excerpt illustrating how to configure WordPress to connect to a database

# File Structure

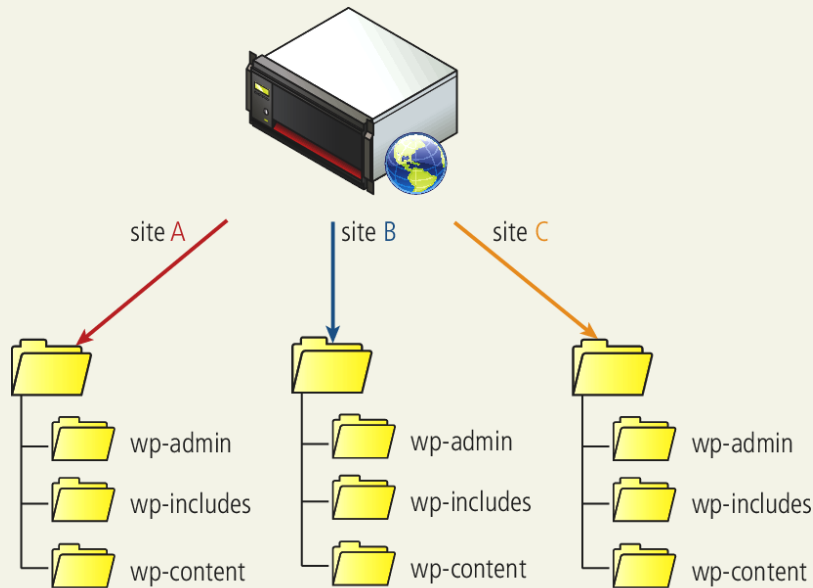
In the end it's all PHP code



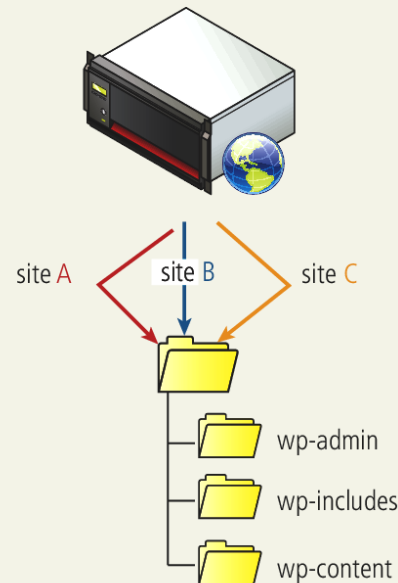


# Multiple Sites on 1 Installation

Advantages for the host (sometimes)



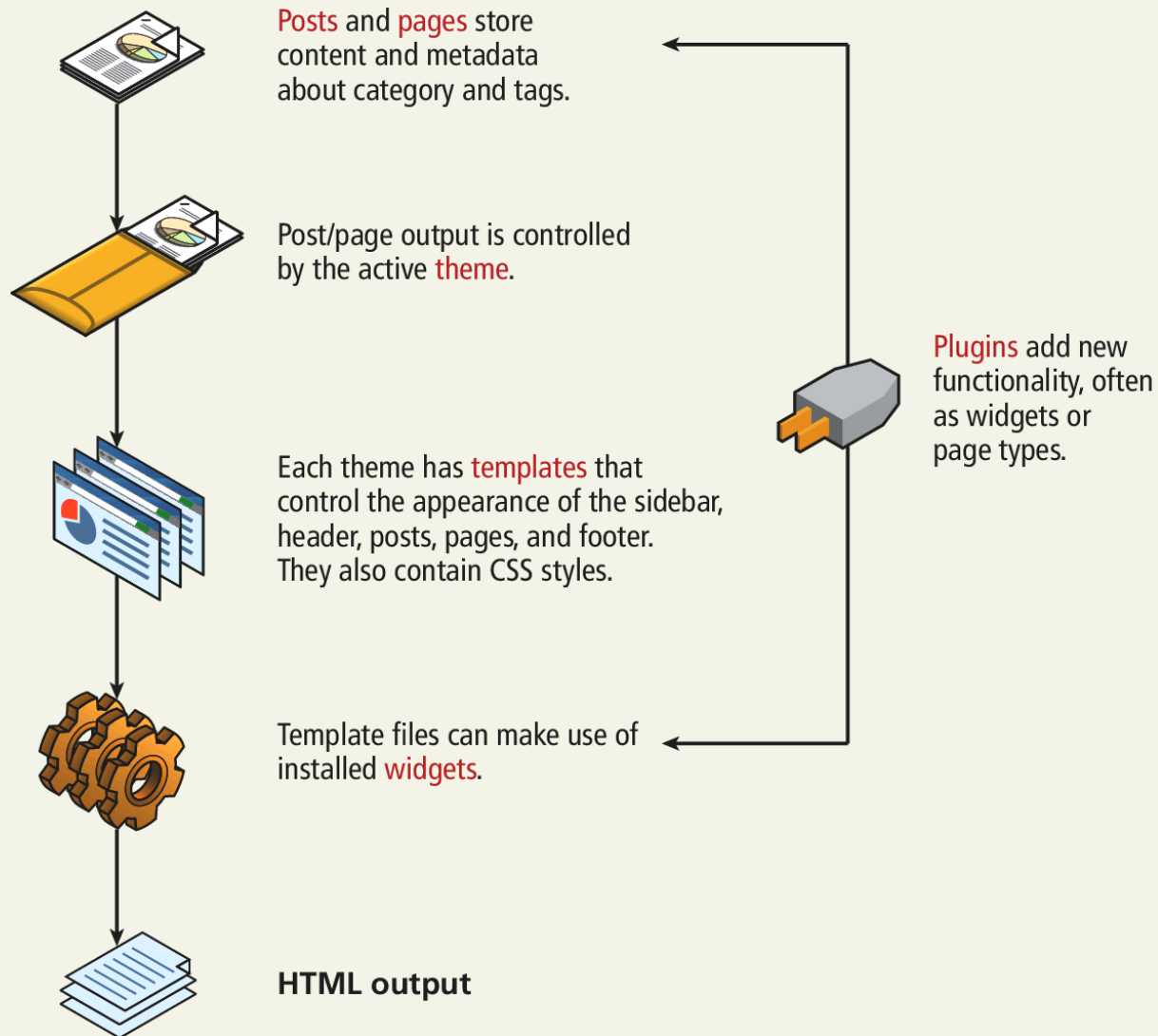
**Server with multiple WordPress installations**



**Multisite WordPress installation**

# WordPress Nomenclature

A visual summary



# WordPress Nomenclature

Posts and Pages ; Templates and Themes

- **Posts** are designed to capture a blog post, or a new update, or something else where you don't require a menu item.
- **Pages** in WordPress are blocks of content, which are normally associated with menu items.
- **WordPress templates** are the PHP files that control how content is pulled from the database and presented to the user.
- **WordPress themes** are a collection of templates, images, styles, and other code snippets that together define the look and feel of your entire site.

# WordPress Nomenclature

## Widgets

**WordPress widgets** allow dynamic content to be arranged in sidebars by nontechnical users through the dashboard. Examples include:

- **Calendar** displays a clickable calendar with links if any posts occurred this month.
- **Categories** displays lists of links to all existing categories.
- **Links** is a widget that allows users to manage internal or external links.
- **Pages** displays links to all pages.
- **Recent Comments** displays the most recent comments.
- **RSS** displays an RSS feed.
- **Tag Cloud** displays a clickable cloud of the top 45 words used as tag keywords.

# WordPress Nomenclature

## Plugins

**Plugins** refer to the third-party add-ons that extend the functionality of WordPress, many of which you can download for free.

- Plugins are modularized pieces of PHP code that interact with the WordPress core to add new features.
- Plugins are managed through the **Plugins** link on the dashboard.

# WordPress Nomenclature

## Permalinks

**Permalinks** is the term given to the links generated by WordPress when rendering the navigation (and other links) for the site.

Consider an unsightly URL such as the following:

**Example.com/?post\_type=textbook&p=396**

Permalink mappings allow URLs to be rewritten in order to make them easier for the user to understand.

**Example.com/textbook/fundamentals-of-web-development/**

# WordPress Nomenclature

Permalinks – Easy to SEO

## Common Settings

- |   |  |
|---|--|
| <input type="radio"/> Default                     | <code>http://funwebdev.com/?p=123</code>   |
| <input type="radio"/> Day and name                | <code>http://funwebdev.com/2013/10/20/sample-post/</code>                              |
| <input type="radio"/> Month and name              | <code>http://funwebdev.com/2013/10/sample-post/</code>                                 |
| <input type="radio"/> Numeric                     | <code>http://funwebdev.com/archives/123</code>   |
| <input type="radio"/> Post name                   | <code>http://funwebdev.com/sample-post/</code>   |
| <input checked="" type="radio"/> Custom Structure | <code>http://funwebdev.com</code> <input type="text" value="/%category%/%postname%/"/> |

# WordPress Nomenclature

## Taxonomies

WordPress supports classification of your posts with metadata related to

- **Authors**
- **Categories**
- **Tags**
- **Link Categories**
- **Custom Taxonomies**



# WordPress Nomenclature

## Taxonomies - Categories

Categories are the most intuitive method of classifying your posts in WordPress

**Categories** ▼

Title:

☐ Display as dropdown  
☒ Show post counts  
☐ Show hierarchy

[Delete](#) | [Close](#)

### Categories

- Chapter (6)
- Lab (1)
- News (10)
- Presentations (2)
- Publishing (1)
- Teaching (1)

# WordPress Nomenclature

## Taxonomies – Other classifications

- **Tags** are almost identical to categories except they are more open-ended, in that content creators can add them on the fly, and are not limited to the predefined terms like they are with categories.
- **Link categories** are used internally by WordPress by those who want to categorize external links.
- WordPress lets you define your own types of taxonomy

# WordPress Template Hierarchy

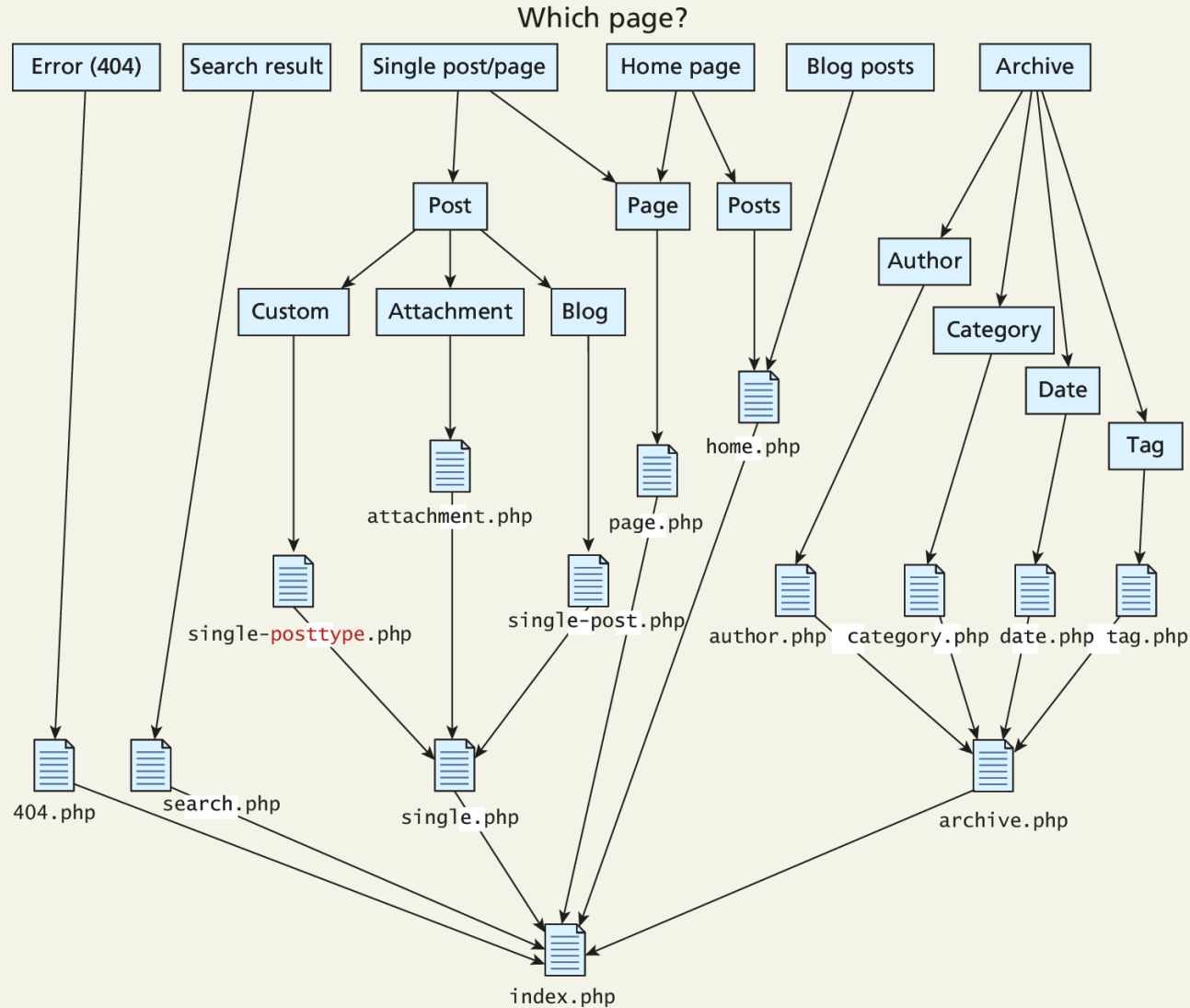
Which file executes?

The default WordPress installation comes with a default theme containing many templates to support the most common types of wireframes you will need. There are templates to display:

- a single page or post
- the home page
- 404 not found page
- categories of posts
- an archive of posts

# WordPress Template Hierarchy

Which file executes?



# WordPress Template Hierarchy

Which file executes?

When a user makes a request, the WordPress CMS determines which template to use to format and deliver the content based on the attributes of the requested page.

If a particular template cannot be found, WordPress continues going down the hierarchy until it finds one, ultimately ending with **index.php**.

Section 5 of 8

# MODIFYING THEMES

# Modifying Themes

Easy in WordPress


The easiest customization you can make to a WordPress installation is to change the **theme** through the dashboard

All the files you need to edit themes are found in the folder **/wp-content/themes/** with a subfolder containing every theme you have installed.


Each theme contains many files found in the previously shown hierarchy

# Changing themes


## In Dashboard



Manage Themes



Install Themes



Current Theme


### Superior Osaka WordPress Theme

By [Digital Cavalry](#) | Version 1.1

Superior and Powerfull Premium WordPress Theme

[Customize](#)    OPTIONS: [Widgets](#) | [Menus](#)

#### Available Themes

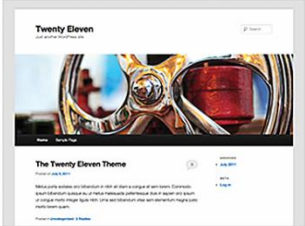


**MetroStyle**  
WordPress Theme

Reach of features.  
Customizable.  
Responsive.  
1440 to 320px.  
Flexible.

By [DlevMedia](#)


[Activate](#) | [Live Preview](#) | [Details](#)    [Delete](#)



**Twenty Eleven**  
By [the WordPress team](#)

[Activate](#) | [Live Preview](#) | [Details](#)    [Delete](#)

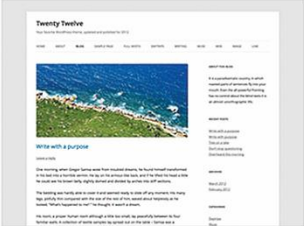
There is a new version of Twenty Eleven available. [View version 1.7 details](#) or [update now](#).



**Twenty Ten**  
By [the WordPress team](#)

[Activate](#) | [Live Preview](#) | [Details](#)    [Delete](#)

There is a new version of Twenty Ten available. [View version 1.6 details](#) or [update now](#).



**Twenty Twelve**  
By [the WordPress team](#)

[Activate](#) | [Live Preview](#) | [Details](#)    [Delete](#)

There is a new version of Twenty Twelve available. [View version 1.3 details](#) or [update now](#).



# Creating a Child Theme

Once you finish you can activate through dashboard

To start a child theme from an existing one, create a new folder on the server in the theme folder.

In that folder create a **style.css** file with the comment from Listing 18.2, which defines the theme name and the template to use with it.

```
/*
Theme Name:      Twenty Twelve Example Child
Theme URI:       http://funwebdev.com/
Description:     Theme to demonstrate child themes
Author:          Randy Connolly and Ricardo Hoar
Author URI:      http://funwebdev.com
Template:        twentytwelve
Version:         1.0.0
*/

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

**LISTING 18.2** Comment to define a child theme and import its style sheet

# Creating Theme Files

You may want to change what HTML is output

1. determine which template file you want to change.
2. Copy the newly particular template file to the child theme, leaving existing page templates alone

# Creating Theme Files

Tinker with the footer

Many sites want to modify the footer for the site, to modify the default link to WordPress if nothing else, all of which is stored in **footer.php**

```
</div><!-- #main .wrapper -->
<footer id="colophon" role="contentinfo">
  <div class="site-info">
    <a href='http://funwebdev.com'>Supported by Fun Web Dev</a>
  </div><!-- .site-info -->
</footer><!-- #colophon -->
</div><!-- #page -->

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

**LISTING 18.3** A sample footer.php template file with the change from the original in red

Section 6 of 8

# **CUSTOMIZING WORDPRESS TEMPLATES**

# Customizing Templates

Writing your own WordPress template is the easiest way to integrate your own custom functionality into WordPress

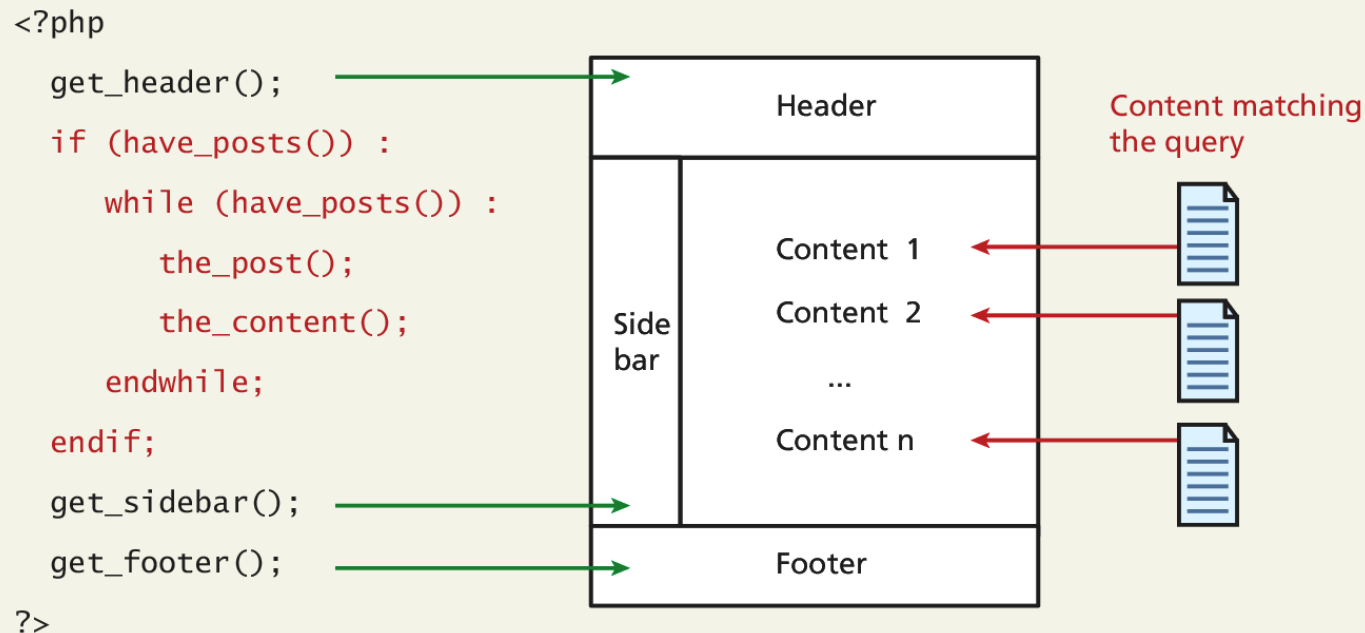
You've already seen how we can tinker with a template file.

Next you must understand the way WordPress works, which means learning about its core classes, the WordPress loop, template tags, and conditional tags

# The WordPress Loop

The main loop

The **WordPress loop** is the term given to the portion of the WordPress framework that pulls content from the database and displays it, which might include looping through multiple posts that need to be displayed.



# The WordPress Loop

The main loop

the template taken from the Twenty Twelve theme's **page.php** template that illustrates use of the loop and common WordPress tags.

```
<?php get_header(); ?>
<div id="primary" class="site-content">
  <div id="content" role="main">
    <?php while ( have_posts() ) : the_post(); ?>
      <?php get_template_part( 'content', 'page' ); ?>
      <?php comments_template( '', true ); ?>
    <?php endwhile; // end of the loop. ?>
  </div> <!-- #content -->
</div> <!-- #primary -->
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

**LISTING 18.4** A simple template file that uses the WordPress loop to print all posts matching the query

It creates a header, loops through all posts, and displays the content of each one (no title, no author, no date) and then outputs a sidebar and a footer.

# The WordPress Loop

The main loop

Because WordPress was written in a functional way to ensure efficient operation, the loop code can be somewhat tricky to understand.

We must first understand the core WordPress classes (written in PHP) that are used in the loop.



# Core WordPress Classes

## WP\_Query

WP\_Query takes the URL (or post data) and parses it to build the appropriate object.

You never have to explicitly create an instance of the WP\_Query object.

When you want WordPress to deviate from the default query, you can use the method `query_posts()` to change it and replace it entirely for use in the WordPress loop.

# Core WordPress Classes

## WP\_Query

The valid keys available to be passed to the WP\_Query object and query\_posts() are numerous.

As you develop you should be aware that

```
print_r($query->query_vars);
```

will output all the query values that are currently present so you can easily find out if you are setting the variables properly.

# Core WordPress Classes

WP\_Query example

To select posts by author with ID 7 in category number 1,2, or 5, except those with tag 17, you would write:

```
$queryArray= array("author" => 7,  
                  "cat" => array(1,2,5),  
                  "tag_not_in" => array(17));
```

# Core WordPress Classes

## WP\_User and the current user

In WordPress you are either serving to a logged-in user or a nonauthenticated user.

To get access to the currently logged-in user, you call

```
$current_user = wp_get_current_user();
```

This `$current_user` is an instance of the `WP_User` class. That class has many properties including `ID`, `first_name`, and `last_name`.

These functions include the ability to determine what capabilities a logged-in user has.

# Core WordPress Classes

WP\_User and the current user

To ask, for example, if the current user is allowed to publish a post you would write:

```
$cu = wp_get_current_user();  
  
if ($cu->has_cap('publish_post',123)) {  
    //the current user is allowed to publish post 123.  
}
```

# Template Tags

For referencing elements within the loop

**Template tags** are really functions that can be called from **inside** the WordPress loop. Categories of tags are:

- General Tags
- Author Tags
- Comment Tags
- Link Tags
- Page Tags

# Template Tags

## General Tags

General tags exist to give you access to global or general things about your site. Some key tags include:

- **get\_header()** includes the header.php file into your page.
- **get\_footer()**, like get\_header(), includes footer.php into your site.
- **get\_sidebar()** works like the methods above, including sidebar.php

# Template Tags

## Author Tags

Since authors are WordPress users, you will be able to access all the fields that can be associated with an author, including their email, full name, visible name, and links to their detail pages on the site.

**the\_author\_meta()** can be called with two parameters, the first being the field you want to retrieve, and the second being the userID. If no second parameter is passed, the userID for the author of the current post is used.

Some commonly used fields include: **display\_name**, **user\_firstname**, **user\_lastname**, **user\_email**, and **user\_url**. Less commonly used ones include: **user\_pass**, **ID**, and **description**.



# Template Tags

## Comment Tags

WordPress manages comments for you and provides the following functions to allow you to programmatically access comments related to the current post.

- **comments\_template()** allows you to import a comment template into this template much like `get_header()`. This way all customization for how comments are displayed can be managed there.
- **get\_comments()** outputs the list of comments matching a range of options passed in.
- **comment\_form()** embeds the form to add comments into the page.

# Template Tags

## Link Tags

Although pages are just a particular type of post, they are also associated with a site hierarchy and the menu. So while they have many essential elements of posts (described later) such as title, author, and date, they also have:

- **`get_ancestors()`** returns an array of the ancestor pages to the current one. They can be used to build a breadcrumb structure.
- **`wp_page_menu()`** can be used to create submenus of pages.

# Template Tags

## Page Tags

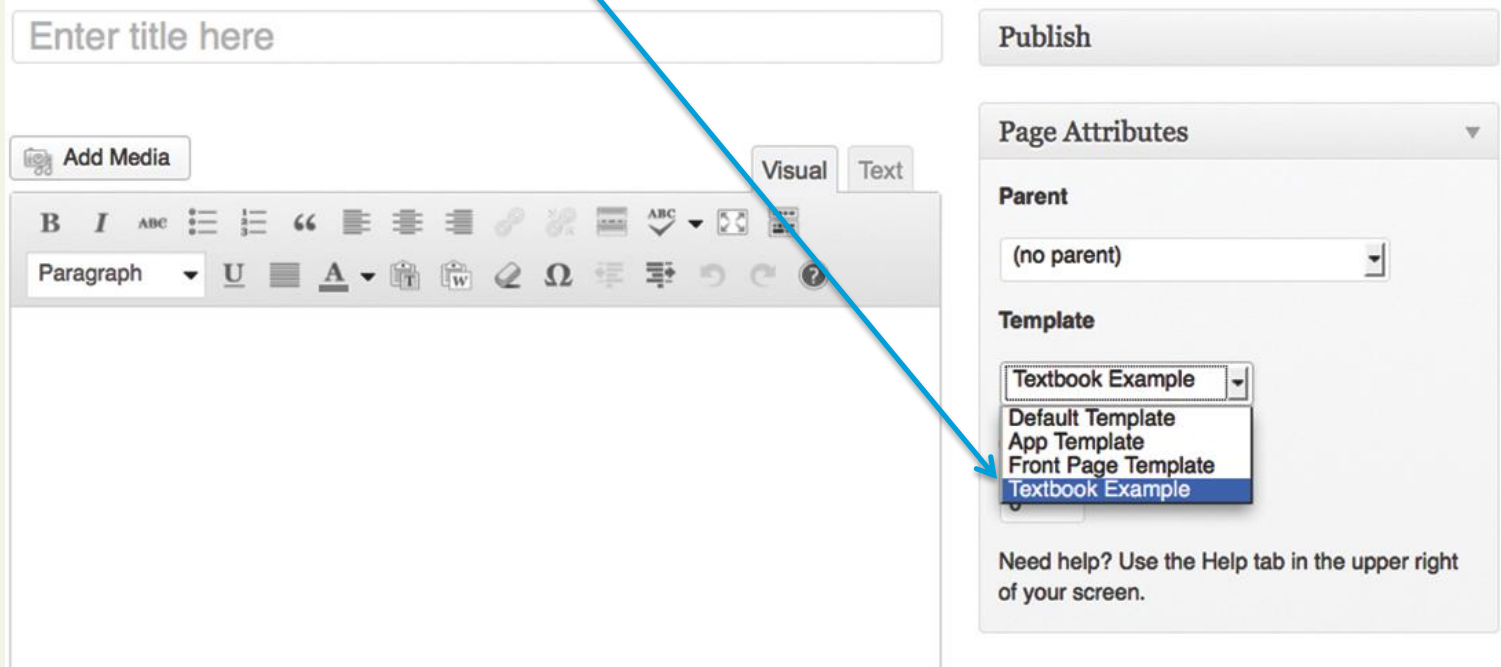
Link tags are especially important for a website, since links are the basis for the WWW. Some important ones include:

- **the\_permalink()** contains the permanent URL assigned to this post. It should be wrapped inside a <A> tag if it is to be clickable.
- **edit\_post\_link()** can be included if you want editors to easily be able to browse the site and click the link to edit a page. This is normally used in conjunction with conditional tags that tell us if the user is currently logged in.
- **get\_home\_url()** returns the URL of the site's home page.

# Creating a Page Template

A new Page template

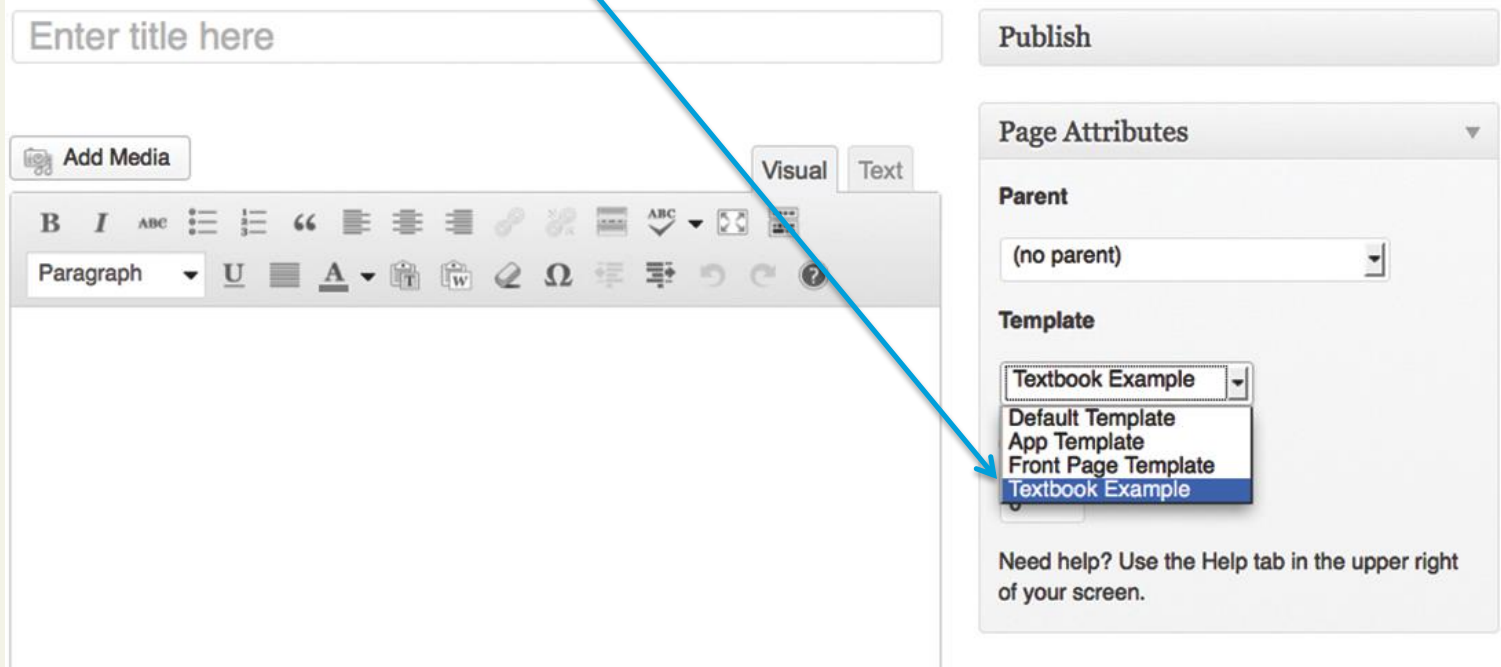
The end goal is that users in WordPress can choose to apply your new template when editing or creating a page using the dropdown interface



# Creating a Page Template

A new Page template

The end goal is that users in WordPress can choose to apply your new template when editing or creating a page using the dropdown interface



# Creating a Page Template

## Get Started

To get started you should create a folder named **page\_templates** in the child theme to hold your custom page types.

Create a PHP file (ours will be **textbook.php**) and add a comment block to define the template name and a description as shown

```
<?php
/**
 * Template Name: Textbook Template
 * Description: Demonstration of a custom page template
 */
?>
```

# Creating a Page Template

## Post Tags

Both pages and posts have the following data available:

- **the\_content()** displays the content of the post; it can optionally display a summary with a “More” link to all content.
- **the\_ID()** returns the underlying database ID, useful elsewhere.
- **the\_title()** returns the title of the current post and optionally prints it out.
- **the\_date()** returns the time and date of the post.

# Creating a Page Template

## Category Tags

Some template tags available to you, which draw their context from the current post:

- **the\_category()** will output a list of clickable links to each category page which this post belongs to.
- **category\_description()** outputs the text description associated with the category of the post.
- **the\_tags()** outputs clickable links to tag pages for every tag used in the post.
- **wp\_tag\_cloud()** outputs a word cloud using all the tags present in the site, *not the post*.



# Creating a Page Template

## Pagination Tags

Navigation tags in general are useful for building a well-interconnected website.

**`previous_post_link()/next_post_link()`** provide the links to the previous and next chronological posts if you wanted to have navigation forward and backward for single posts.

**`previous_posts_link()/next_posts_link()`** are pluralized forms of the above functions and allow you to get links to the previous or next set of items (say 10 per page).

# Creating a Page Template

Illustration using many tags

The diagram illustrates the use of various WordPress template tags in a page template. It shows a sample page layout with annotations linking specific tags to their output:

- the\_title()** points to the main title **Testing Themes**.
- the\_author\_meta('display\_name')** points to the author name **This page by: Ricardo**.
- the\_date()** points to the date **Last edited: June 29, 2013**.
- the\_ID()** points to the **PageID: 397** in the meta box.
- the\_content()** points to the main body text: **Testing themes is a fun thing to do, you tweak styles and use tags to access WordPress elements.**
- wp\_tag\_cloud()** points to the tag cloud at the bottom, which includes links like **apps**, **cryptoquip**, **Dictionary**, **games**, **holiday**, **hosting**, **iOS6**, **iOS7**, **iPhone5**, **jewelSlide**, **websites**, **wordpress**, **wordSlide**, and **wordTheme**.

On the right side, a yellow box displays user information, with arrows indicating the source tags:

- wp\_get\_current\_user()->user\_login** points to **Username: admin**.
- wp\_get\_current\_user()->firstname** points to **User first name: Ricardo**.
- wp\_get\_current\_user()->lastname** points to **User last name: Hoar**.
- wp\_get\_current\_user()->ID** points to **User ID: 1**.

Section 7 of 8

# CREATING A CUSTOM POST TYPE

# Custom Post Type

What types are there already?

You can access the type of a post by using `get_post_type()` anywhere in the loop.

- **post** is the default kind of content post, used for blog entries.
- **page** is a WordPress page, that is, a page associated with a menu item hierarchy.
- **attachment** defines a post that is an image or file attachment.
- **revision** versioning is also stored, so you can have posts that store versioning information
- **nav\_menu\_item** is reserved for menu items (which are still posts).

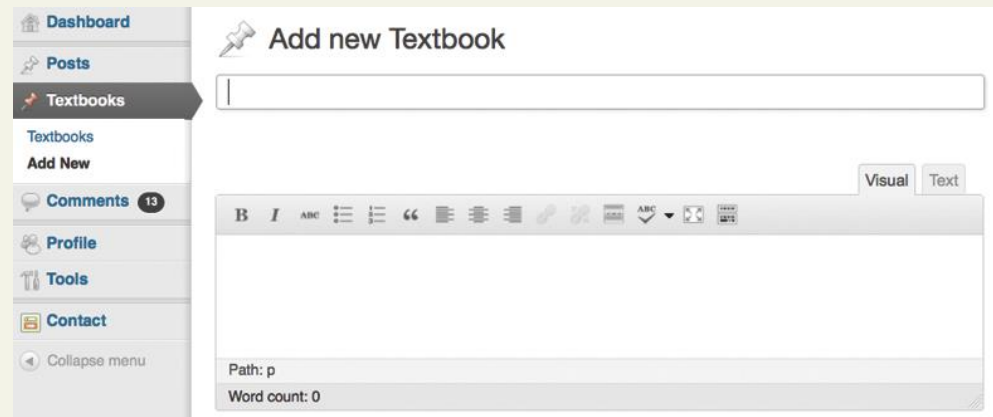
# Organization

What types are there already?

If you were to create a new type called *textbook*, you would be able to surf all posts of that type by going to **<http://example.com/textbook/>**.

You could then create a file named **single-*textbook.php*** to handle a single post, and **archive-*textbook.php*** to handle displaying all the *textbook* posts.

A new tab will appear allowing users to easily add and manage *textbook* posts.



# Registering your post type

Register with what? Where?

There is a file in WordPress named **functions.php**, which allows you to integrate your own post types into the framework.

To define the mere existence of the post type *textbook*, you would create a **functions.php** file in your child theme.

Unlike **style.css**, the **functions.php** of a child theme does not override its counterpart from the parent. Instead, it is loaded *in addition* to the parent's **functions**.

# Registering your post type

Register with what? Where?

```
<?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' );
```

**LISTING 18.6** Registering a new post type in a theme's functions.php

# Adding post specific fields

Now we're making sense.

The reason you normally create a specific type of post is that you can systematically define a new category of “item” in such a way that users can easily enter them.

In our textbook example, you might want to say that all *textbook* posts require details such as *publisher*, *date of publishing*, and *authors*.

To add those fields to the form, you must use the **add\_meta\_box()** function



# Adding post specific fields

Now we're making sense.

```
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
    }
}

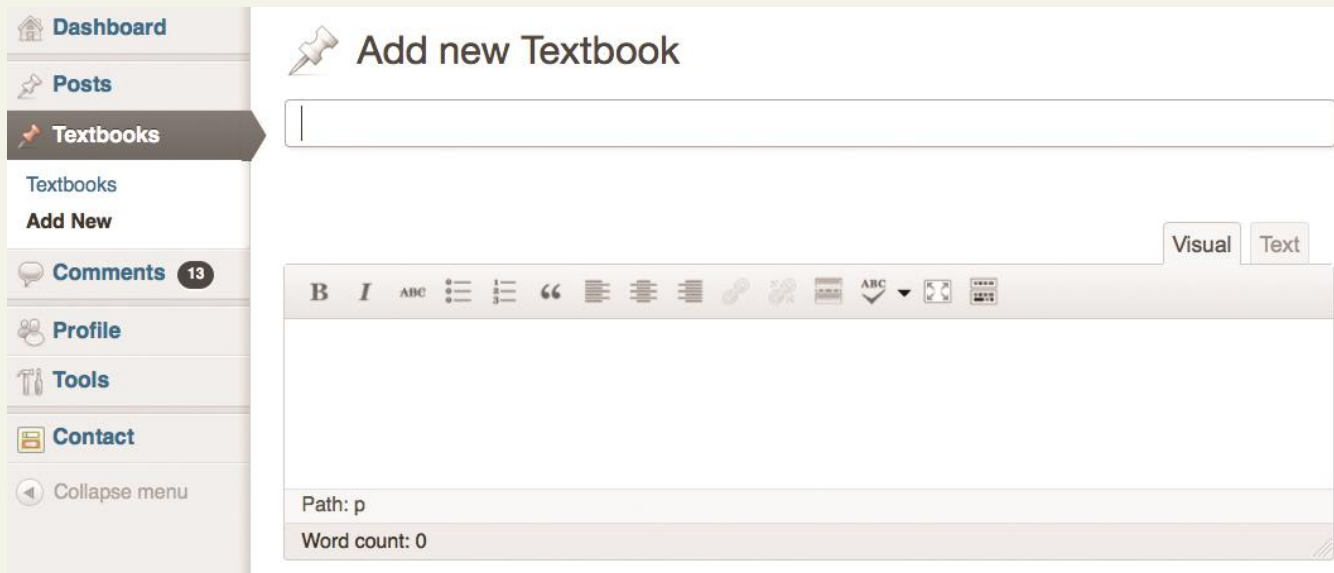
?>
Please enter the required details for a textbook here.
<div class="wrap">
<p><label>Publisher:</label><br />
<input name="textbook_pub" value="<?php echo $publisher; ?>" /></p>
<p><label>Author(s):</label><br />
<input name="textbook_author" value="<?php echo $author; ?>" /></p>
<p><label>Date:</label><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' );
```

**LISTING 18.7** Code to attach fields to the editing interface

# Adding post specific fields

Now we're making sense.



The screenshot displays a web application interface with a sidebar on the left and a main content area on the right.

**Sidebar:**

- Dashboard (house icon)
- Posts (pencil icon)
- Textbooks** (pencil icon, highlighted with a dark background)
- Textbooks (pencil icon)
- Add New (pencil icon)
- Comments (speech bubble icon, 13)
- Profile (person icon)
- Tools (wrench icon)
- Contact (envelope icon)
- Collapse menu (left arrow icon)

**Main Content Area:**

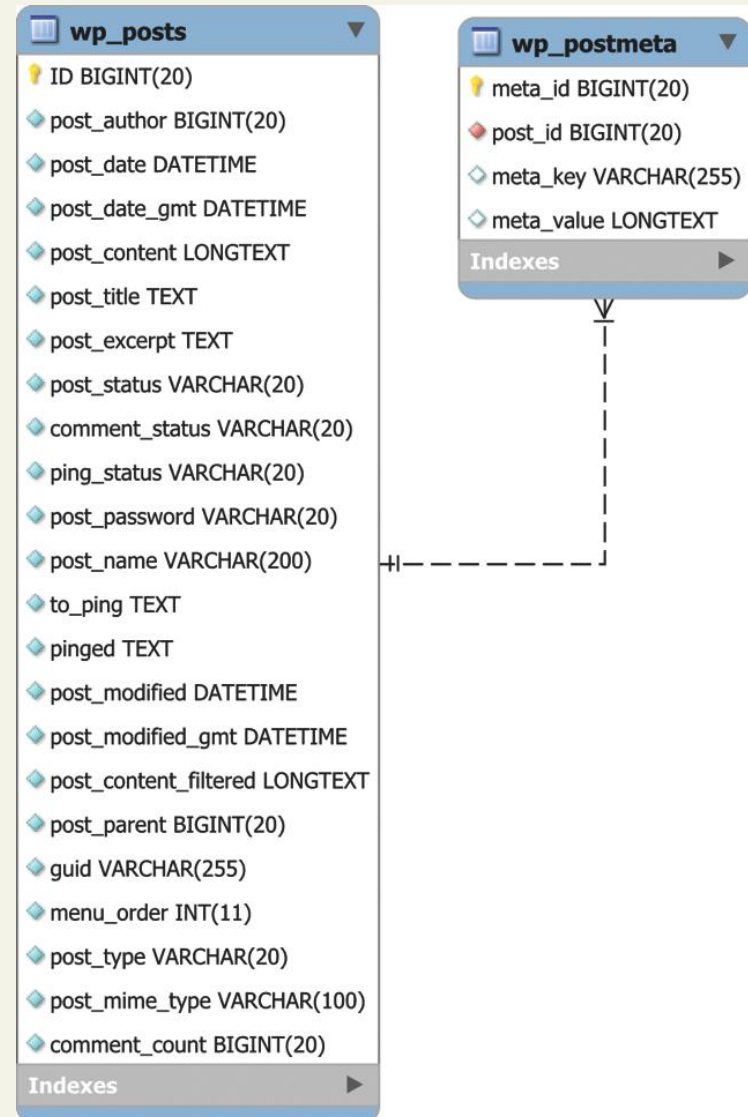
- Header: "Add new Textbook" with a pushpin icon.
- Form: A large text input field.
- Visual/Text tabs: Located below the input field.
- Rich Text Editor: A toolbar with icons for bold (B), italic (I), text color (ABC), bulleted list, numbered list, quote, link, unlink, insert link, unlink, undo, redo, and a dropdown menu (ABC). Below the toolbar is a large text area.
- Footer: A section with "Path: p" and "Word count: 0".

# Under the Hood

Where do our posts go?

Two tables are used in creating custom posts. The first is the **wp\_posts** table where things like the post date, author, title, status, and type are located.

Related directly to that is a **wp\_postmeta** table, which is where our custom fields are stored



# Under the Hood

Where do our posts go?

```
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');
```

**LISTING 18.8** Code to save input values from custom fields when the user saves/creates a textbook post

# Displaying our post type

Display that data we just stored

You will be need to define at least two templates.

- **single-textbook.php** and displays a single textbook, and
- archive of all the posts matching the type served from the file **archive-textbook.php**.

# Single Post Template

Display that data we just stored

```
<?php while ( have_posts() ) : the_post(); ?>
  <div class="title"> <?php the_title(); ?> </div>
  <?php
    global $post;
    $custom = get_post_custom($post->ID);
    $author = $custom['textbook_author'][0]; //authors
    $pubdate = $custom['textbook_date'][0]; //date
  ?>
  <div class='author'>
    By: <?php echo $author." (". $pubdate.")" ; ?></div>
  <div class='content'> <?php the_content(); ?> </div>
<?php endwhile; // end of the loop. ?>
<a href='<?php echo get_post_type_archive_link("textbook");?>'>Browse all Textbooks</a>
<?php
//navigation to newer/older posts
echo "Older Post: "; next_post_link();
echo "Newer Post: "; previous_post_link();
?>
```

**LISTING 18.9** The Single-textbook.php template excerpt used to format and output a single textbook post

# Archive Page Template

A List of posts

```
<?php while ( have_posts() ) : the_post(); ?>
  <div class="title">
    <a href=' <?php the_permalink();?> '> <?php the_title(); ?> </a>
  </div>
  <?php
    global $post;    //access the custom meta fields
    $custom = get_post_custom($post->ID);
    $author = $custom['textbook_author'][0]; //authors
    $pubdate = $custom['textbook_date'][0]; //date
  ?>
  <div class='author'>
    By: <?php echo $author." (".$pubdate.)"; ?>
  </div>
  <?php endwhile; // end of the loop. ?>
  <div class="nav-previous"><?php next_posts_link("Older Books"); ?>
  </div>
  <div class="nav-next"><?php previous_posts_link("Newer Books") ?></div>
```

**LISTING 18.10** The archive-textbook.php file, which is called upon to to display a list of textbooks

# Tweaking result sets

## Changing Pages Per Archive Page

One of the customizations you may want to make is to change how many posts are shown in an archive page. To accomplish that you have to add a **filter** to `functions.php` so that for our *textbook* post type the value is say 20 books, rather than the default.

```
function custom_posts_per_page($query)
{
    if ( $query->query_vars['post_type'] == 'textbook')
        $query->query_vars['posts_per_page'] = 20;
    return $query;
}

add_filter( 'pre_get_posts', 'custom_posts_per_page' );
```

**LISTING 18.11** Filter added to change the number of textbooks to display per page.



Section 8 of 8

# WRITING A PLUGIN

# Writing a Plugin

## Getting Started

Plugins allow you to write code independent of the main WordPress framework and then use hooks, filters, and actions to link to the main code.

Much like themes, WordPress plugins reside in their own folder **/wp-content/plugins/**

Like themes, you should begin by creating a folder to contain all the files for your theme. Name the plugin folder something unique, which has not yet been used.

# Writing a Plugin

## Getting Started

Our first act is to create the main file for the plugin, **index.php**, inside our folder.

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

**LISTING 18.12** Comment that defines a plugin inside `/wp-content/plugins/funwebdev-textbook/index.php`

# Hooks, Actions and Filters

## Getting Started

- **Hooks** are events that occur during the regular operation of WordPress.
- **Actions** are PHP functions executed at specific times in the WordPress core.
- **Filters** in WordPress allow you to choose a subset of data before doing something with it

# Reuse some code from before

## Convert Your Page Type Template to a Plugin

Take all the code we added to **functions.php** in the child theme and add it to our plugin's **index.php**.

Recall this code defined the textbook page, and added code to properly display textbooks in the admin interface.

# Activate Your Plugin

Finally something super easy.



**TextBook Plugin (funwebdev)**

Deactivate

Allows for management of textbooks

Version 1.0 | By Ricardo Hoar | [Visit plugin site](#)

# Output of the Plugin

To finish this plugin, you must move the code from the templates into the plugin file (**index.php**).

```
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');
```

**LISTING 18.13** Replacing the\_content() with a filter for our Textbook plugin

# Make it a widget

Widget, Plugin, Template, oh my

To the user, widgets are easy-to-manage and customizable components they can add to the sidebar.

To create a widget that displays a random book, we therefore only have to define one function for displaying the content of the widget.



# Make it a widget

Widget, Plugin, Template, oh my

```
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'
    )
);
```

**LISTING 18.14** Registering a sidebar widget that displays a random textbook

# What You've Learned

**1** Managing websites

**2** Content Management Systems (**CMS**)

**3** CMS **Components**

**4** **WordPress** Technical Overview

**5** Modifying **Themes**

**6** Customizing WordPress **Templates**

**7** Creating a **Custom Post Type**

**8** Writing a **Plugin**

---