

Rakhitha Nimesh Ratnayake

WordPress Web Application Development

Third Edition

Build rapid web applications with cutting-edge technologies using WordPress



Packt>

Contents

- [1: WordPress as a Web Application Framework](#)
 - [Chapter 1: WordPress as a Web Application Framework](#)
 - [WordPress as a CMS](#)
 - [WordPress as a web application framework](#)
 - [Simplifying development with built-in features](#)
 - [Identifying the components of WordPress](#)
 - [A development plan for the forum management application](#)
 - [Understanding limitations and sticking to guidelines](#)
 - [Building a question-answer interface](#)
 - [Enhancing features of the questions plugin](#)
 - [Categorizing questions](#)
 - [Summary](#)
- [2: Implementing Membership Roles, Permissions, and Features](#)
 - [Chapter 2: Implementing Membership Roles, Permissions, and Features](#)
 - [Introduction to user management](#)
 - [Getting started with user roles](#)
 - [Understanding user capabilities](#)
 - [Registering application users](#)
 - [Implementing frontend registration](#)
 - [Creating a login form in the frontend](#)
 - [Essential user management features for web applications](#)
 - [Implementing user management features with popular plugins](#)
 - [Time to practice](#)
 - [Summary](#)
- [3: Planning and Customizing the Core Database](#)
 - [Chapter 3: Planning and Customizing the Core Database](#)
 - [Understanding the WordPress database](#)
 - [Exploring the role of existing tables](#)
 - [Adapting existing tables in web applications](#)
 - [Extending the database with custom tables](#)
 - [Planning the forum application tables](#)
 - [Querying the database](#)
 - [Limitations and considerations](#)
 - [Summary](#)
- [4: Building Blocks of Web Applications](#)
 - [Chapter 4: Building Blocks of Web Applications](#)

- [Introduction to custom content types](#)
- [Planning custom post types for an application](#)
- [Implementing custom post types for a forum application](#)
- [What is a template engine?](#)
- [Persisting custom field data](#)
- [Introduction to post type templates](#)
- [Introducing custom post type relationships](#)
- [Pods framework for custom content types](#)
- [Implementing custom post type features with popular plugins](#)
- [Time to practice](#)
- [Summary](#)
- [5: Implementing Application Content Restrictions](#)
 - [Chapter 5: Implementing Application Content Restrictions](#)
 - [Introduction to content restrictions](#)
 - [Practical usage of content restrictions](#)
 - [Understanding restriction levels](#)
 - [Implementing content restrictions in posts/pages](#)
 - [Enabling restrictions on WordPress core features](#)
 - [Supplementary content restriction types and techniques](#)
 - [Useful plugins for content restrictions](#)
 - [Time to practice](#)
 - [Summary](#)
- [6: Developing Pluggable Modules](#)
 - [Chapter 6: Developing Pluggable Modules](#)
 - [A brief introduction to WordPress plugins](#)
 - [WordPress plugins for web development](#)
 - [Tips for developing extendable plugins](#)
 - [Time to practice](#)
 - [Summary](#)
- [7: Customizing the Dashboard for Powerful Backends](#)
 - [Chapter 7: Customizing the Dashboard for Powerful Backends](#)
 - [Understanding the admin dashboard](#)
 - [Customizing the admin toolbar](#)
 - [Customizing the main navigation menu](#)
 - [Adding features with custom pages](#)
 - [Building options pages](#)
 - [Using feature-packed admin list tables](#)

- [An awesome visual presentation for admin screens](#)
 - [The responsive nature of the admin dashboard](#)
 - [Supplementary admin dashboard features](#)
 - [Time for action](#)
 - [Summary](#)
- [8: Adjusting Theme for Amazing Frontends](#)
 - [Chapter 8: Adjusting Theme for Amazing Frontends](#)
 - [An introduction to the WordPress application frontend](#)
 - [Web application layout creation techniques](#)
 - [Building the forum application home page](#)
 - [Generating the application frontend menu](#)
 - [Managing options and widgets with customizer](#)
 - [Creating pluggable templates](#)
 - [Extending the home page template with action hooks](#)
 - [Planning action hooks for layouts](#)
 - [Managing custom CSS with live preview](#)
 - [Responsive previews in theme customizer](#)
 - [Time for action](#)
 - [Summary](#)
- [9: Enhancing the Power of Open Source Libraries and Plugins](#)
 - [Chapter 9: Enhancing the Power of Open Source Libraries and Plugins](#)
 - [Why choose open source libraries?](#)
 - [Open source libraries inside the WordPress core](#)
 - [Open source JavaScript libraries in the WordPress core](#)
 - [Using PHPMailer for custom e-mail sending](#)
 - [Implementing user authentication with OpenAuth](#)
 - [Using third-party libraries and plugins](#)
 - [Using open source plugins for web development](#)
 - [Using plugins for checking security of other plugins](#)
 - [Time for action](#)
 - [Summary](#)
- [10: Listening to Third-Party Applications](#)
 - [Chapter 10: Listening to Third-Party Applications](#)
 - [Introduction to APIs](#)
 - [The WordPress XML-RPC API for web applications](#)
 - [Building the API client](#)

- [Creating a custom API](#)
- [Integrating API user authentication](#)
- [Integrating API access tokens](#)
- [Providing the API documentation](#)
- [WordPress REST API for web applications](#)
- [Time for action](#)
- [Summary](#)
- [11: Integrating and Finalizing the Forum Management Application](#)
 - [Chapter 11: Integrating and Finalizing the Forum Management Application](#)
 - [Integrating and structuring the forum application](#)
 - [Integrating the template loader into a user manager](#)
 - [Working with the restructured application](#)
 - [Updating a user profile with additional fields](#)
 - [Scheduling subscriber notifications](#)
 - [Time for action](#)
 - [Final thoughts](#)
 - [Summary](#)
- [12: Supplementary Modules for Web Development](#)
 - [Chapter 12: Supplementary Modules for Web Development](#)
 - [Internationalization](#)
 - [Working with media grid and image editor](#)
 - [Introduction to the post editor](#)
 - [Lesser-known WordPress features](#)
 - [Managing application scripts and styles](#)
 - [Version control](#)
 - [E-commerce](#)
 - [Migrating WordPress applications](#)
 - [Importing and exporting application content](#)
 - [Introduction to multisite](#)
 - [Time for action](#)
 - [Summary](#)
- [13: Configurations, Tools, and Resources](#)
 - [Chapter 13: Configurations, Tools, and Resources](#)
 - [Configuring and setting up WordPress](#)
 - [Open source libraries and plugins](#)
 - [Online resources and tutorials](#)

Chapter 1. WordPress as a Web Application Framework

In recent years, WordPress has matured from the most popular blogging platform to the most popular content management system. Thousands of developers around the world are making a living from WordPress design and development. As more and more people are interested in using WordPress, the dream of using this amazing framework for web application development is becoming possible.

The future seems bright as WordPress has already got dozens of built-in features, which can be easily adapted to web application development using slight modifications. Since you are already reading this book, you have to be someone who is really excited to see how WordPress fits into web application development. Throughout this book, we will learn how we can inject the best practices of web development into the WordPress framework to build web applications using a rapid process.

Basically, this book will be important for developers from two different perspectives. On the one hand, beginner to intermediate level WordPress developers can get knowledge of cutting-edge web development technologies and techniques to build complex applications. On the other hand, web development experts who are already familiar with popular PHP frameworks can learn WordPress for rapid application development. So, let's get started!

In this chapter, we will cover the following topics:

- WordPress as a CMS
- WordPress as a web application framework
- Simplifying development with built-in features
- Identifying the components of WordPress
- Making a development plan for forum management application
- Understanding limitations and sticking with guidelines
- Building a question-answer interface
- Enhancing features of the questions plugin

In order to work with this book, you should be familiar with WordPress themes, plugins, and its overall process. Developers who are experienced in PHP frameworks can work with this book while using the reference sources to learn WordPress. By the end of this chapter, you will have the ability to make the decision to choose WordPress for web development.

WordPress as a CMS

Way back in 2003, WordPress released its first version as a simple blogging platform and it continued to improve until it became the most popular blogging tool. Later, it continued to improve as a **CMS (Content Management System)** and now has a reputation for being the most popular CMS for over five years. These days, everyone sees WordPress as a CMS rather than just a blogging tool.

Now the question is, where will it go next?

Recent versions of WordPress have included popular web development libraries such as `Backbone.js` and `Underscore.js` and developers are building different types of applications with WordPress. Also, the most recent introduction of the REST API is a major indication that WordPress is moving towards the direction of building web applications. The combination of the REST API and modern JavaScript frameworks will enable developers to build complex web applications with WordPress.

Before we consider the application development aspects of WordPress, it's ideal to figure out the reasons for it being such a popular CMS. The following are some of the reasons behind the success of WordPress as a CMS:

1. The plugin-based architecture for adding independent features and the existence of over 40,000 open source plugins
2. The ability to create unlimited free websites at www.wordpress.com and use the basic WordPress features
3. A super simple and easy-to-access administration interface
4. A fast learning curve and comprehensive documentation for beginners
5. A rapid development process involving themes and plugins
6. An active development community with awesome support
7. The flexibility in building websites with its themes, plugins, widgets, and hooks
8. The availability of large premium theme and plugin marketplaces for

developers to sell advanced plugin/themes and users to build advanced sites with those premium plugins/themes without needing a developer

These reasons prove why WordPress is the top CMS for website development. However, experienced developers who work with full stack web applications don't believe that WordPress has a future in web application development. While it's up for debate, we'll see what WordPress has to offer for web development.

Once you complete reading this book, you will be able to decide whether WordPress has a future in web applications. I have been working with full stack frameworks for several years, and I certainly believe in the future of WordPress for web development.

WordPress as a web application framework

In practice, the decision to choose a development framework depends on the complexity of your application. Developers will tend to go for frameworks in most scenarios. It's important to figure out why we go with frameworks for web development. Here's a list of possible reasons why frameworks become a priority in web application development:

- Frameworks provide stable foundations for building custom functionalities
- Usually, stable frameworks have a large development community with an active support
- They have built-in features to address the common aspects of application development, such as routing, language support, form validation, user management, and more
- They have a large amount of utility functions to address repetitive tasks

Full stack development frameworks such as **Zend**, **CodeIgniter**, and **CakePHP** adhere to the points mentioned in the preceding section, which in turn becomes the framework of choice for most developers. However, we have to keep in mind that WordPress is an application where we build applications on top of existing features. On the other hand, traditional frameworks are foundations used for building applications such as WordPress. Now, let's take a look at how WordPress fits into the boots of the web application framework.

The MVC versus event-driven architecture

A vast majority of web development frameworks are built to work with MVC architecture, where an application is separated into independent layers called models, views, and controllers. In MVC, we have a clear understanding of what goes where and when each of the layers will be integrated in the process.

So, the first thing most developers will look at is the availability of MVC in WordPress. Unfortunately, WordPress is not built on top of the MVC

architecture. This is one of the main reasons why developers refuse to choose it as a development framework. Even though it is not MVC, we can create custom execution processes to make it work like an MVC application. Also, we can find frameworks such as **WP MVC**, which can be used to take advantage of both WordPress's native functionality and its vast plugin library and all of the many advantages of an MVC framework. Unlike other frameworks, it won't have the full capabilities of MVC. However, the unavailability of MVC architecture doesn't mean that we cannot develop quality applications with WordPress. There are many other ways to separate concerns in WordPress applications.

On the other hand WordPress, relies on a procedural event-driven architecture with its action hooks and filters system. Once a user makes a request, these actions will get executed in a certain order to provide the response to the user. You can find the complete execution procedure at http://codex.wordpress.org/Plugin_API/Action_Reference.

In the event-driven architecture, both model and controller code gets scattered throughout the theme and plugin files. In the upcoming chapters, we will look at how we can separate these concerns with the event-driven architecture, in order to develop maintainable applications.

Simplifying development with built-in features

As we discussed in the previous section, the quality of a framework depends on its core features. The better the quality of the core, the better it will be for developing quality and maintainable applications. It's surprising to see the availability of a number of WordPress features directly related to web development, even though it is meant to create websites.

Let's get a brief introduction to the WordPress core features to see how they fit into web application development.

User management

Built-in user management features are quite advanced in order to cater to the most common requirements of any web application. Its user roles and capability handling make it much easier to control the access to specific areas of your application. We can separate users into multiple levels using roles and then use capabilities to define the permitted functionality for each user level. Most full stack frameworks don't have built-in user management features, and hence this can be considered as an advantage of using WordPress.

Media management

File uploading and managing is a common and time consuming task in web applications. Media uploader, which comes built-in with WordPress, can be effectively used to automate the file-related tasks without writing much source code. A super-simple interface makes it so easy for application users to handle file-related tasks.

Template management

WordPress offers a simple template management system for its themes. It is not as complex or fully featured as a typical template engine. However, it offers a wide range of capabilities from a CMS development perspective, which we can extend to suit web applications.

Database management

In most scenarios, we will be using the existing database table structure for our application development. WordPress database management functionalities offer a quick and easy way of working with existing tables with its own style of functions. Unlike other frameworks, WordPress provides a built-in database structure, and hence most of the functionalities can be used to directly work with these tables without writing custom SQL queries.

Routing

Comprehensive support for routing is provided through permalinks. WordPress makes it simple to change the default routing and choose your own routing, in order to build search engine-friendly URLs.

XML-RPC API

Building an API is essential for allowing third-party access to our application. WordPress provides a built-in API for accessing CMS-related functionality through its XML-RPC interface. Also, developers are allowed to create custom API functions through plugins, making it highly flexible for complex applications.

REST API

The REST API makes it possible to give third-party access to the application data, similar to XML-RPC API. This API uses easy to understand HTTP requests and JSON format, making it easier to communicate with WordPress applications. JavaScript is becoming the modern trend in developing applications. So, the availability of JSON in the REST API will allow external

users to access and manipulate WordPress data within their JavaScript-based applications.

Caching

Caching in WordPress can be categorized into two sections called **persistent** and **nonpersistent** caching. Nonpersistent caching is provided by the WordPress cache object while persistent caching is provided through its Transient API. Caching techniques in WordPress are simple compared to other frameworks, but it's powerful enough to cater for complex web applications.

Scheduling

As developers, you might have worked with cron jobs for executing certain tasks at specified intervals. WordPress offers the same scheduling functionality through built-in functions, similar to a cron job. However, WordPress cron execution is slightly different from normal cron jobs. In WordPress, cron won't be executed unless someone visits the site. Typically, it's used for scheduling future posts. However, it can be extended to cater complex scheduling functionality.

Plugins and widgets

The power of WordPress comes from its plugin mechanism, which allows us to dynamically add or remove functionality without interrupting other parts of the application. Widgets can be considered as a part of the plugin architecture and will be discussed in detail further in this chapter.

Themes

The design of a WordPress site comes through the theme. This site offers many built-in template files to cater to the default functionality. Themes can be easily extended for custom functionality. Also, the design of the site can be changed instantly by switching to a compatible theme.

Actions and filters

Actions and filters are part of the WordPress hook system. Actions are events that occur during a request. We can use WordPress actions to execute certain functionalities after a specific event is completed. On the other hand, filters are functions that are used to filter, modify, and return data. Flexibility is one of the key reasons for the higher popularity of WordPress compared to other CMSs. WordPress has its own way of extending functionality of custom features as well as core features through actions and filters. These actions and filters allow developers to build advanced applications and plugins, which can be easily extended with minor code changes. As a WordPress developer, it's a must to know the perfect use of these actions and filters in order to build highly-flexible systems.

The admin dashboard

WordPress offers a fully-featured backend for administrators as well as normal users. These interfaces can be easily customized to adapt to custom applications. All the application-related lists, settings, and data can be handled through the admin section.

The overall collection of features provided by WordPress can be effectively used to match the core functionalities provided by full stack PHP frameworks.

Identifying the components of WordPress

WordPress comes up with a set of prebuilt components, which are intended to provide different features and functionality for an application. A flexible theme and powerful admin features act as the core of WordPress websites, while plugins and widgets extend the core with application-specific features. As a CMS, we all have a pretty good understanding of how these components fit into a WordPress website.

Here our goal is to develop web applications with WordPress, and hence it is important to identify the functionality of these components from the perspective of web applications. So, we will look at each of the following components, how they fit into web applications, and how we can take advantage of them to create flexible applications through a rapid development process:

- The role of WordPress themes
- The role of admin dashboard
- The role of plugins
- The role of widgets

The role of WordPress themes

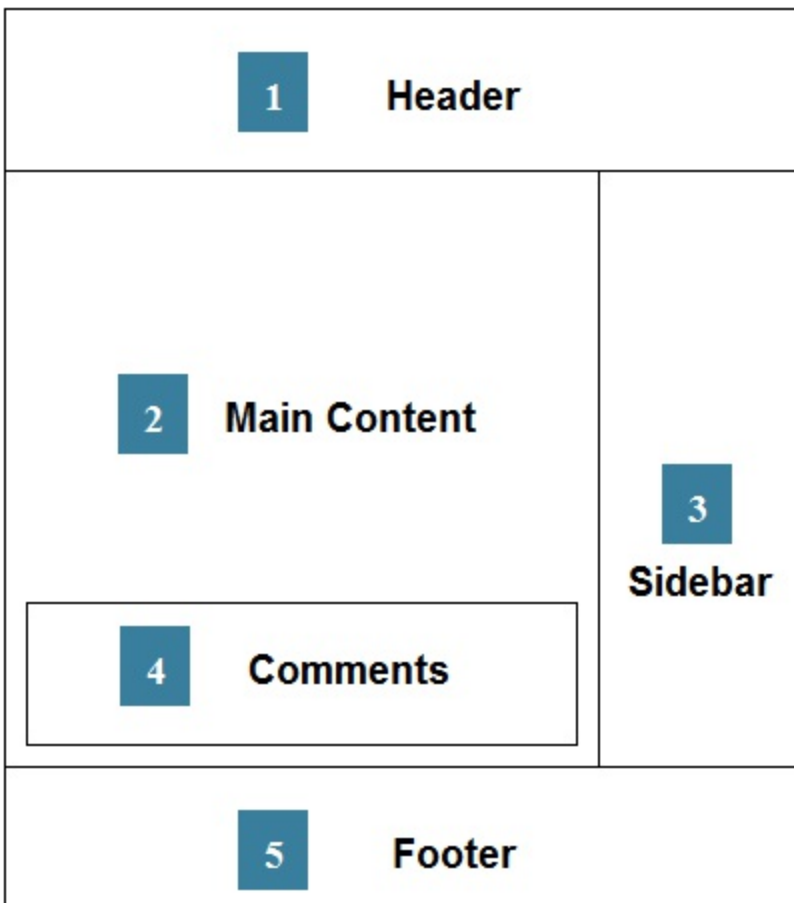
Most of us are used to seeing WordPress as a CMS. In its default view, a theme is a collection of files used to skin your web application layouts. In web applications, it's recommended to separate different components into layers such as models, views, and controllers. WordPress doesn't adhere to the MVC architecture. However, we can easily visualize themes or templates as the presentation layer of WordPress.

In simple terms, views should contain the HTML needed to generate the layout and all the data it needs should be passed to the views. WordPress is built to create content management systems, and hence it doesn't focus on separating views from its business logic. Themes contain views, also known as template files, as a mix of both HTML code and PHP logic. As web application

developers, we need to alter the behavior of existing themes in order to limit the logic inside templates and use plugins to parse the necessary model data to views.

Structure of a WordPress page layout

Typically, posts or pages created in WordPress consist of five common sections. Most of these components will be common across all the pages in the website. In web applications, we also separate the common layout content into separate views to be included inside other views. It's important for us to focus on how we can adapt the layout into web application-specific structure. Let's visualize the common layout of WordPress using the following image:



Having looked at the structure, it's obvious that the **Header**, **Footer**, and the **Main Content** areas are mandatory even for web applications. However, the

Footer and **Comments** section will play a less important role in web applications, compared to web pages. The **Sidebar** is important in web applications, even though it won't be used with the same meaning. It can be quite useful as a dynamic widget area.

Customizing the application layout

Web applications can be categorized as **projects** and **products**. A project is something we develop to target the specific requirements of a client. On the other hand, a product is an application created based on the common set of requirements for a wide range of users. Therefore, customizations will be required on layouts of your product based on different clients.

WordPress themes make it simple to customize the layout and features using child themes. We can make the necessary modifications in the child theme while keeping the core layout in the parent theme. This will prevent any code duplications in customizing layouts. Also, the ability to switch themes is a powerful feature that eases the layout customization.

The role of the admin dashboard

The administration interface of an application plays one of the most important roles behind the scenes. WordPress offers one of the most powerful and easy-to-access admin areas among other competitive frameworks. Most of you should be familiar with using the admin area for CMS functionalities.

However, we will have to understand how each component in the admin area suits the development of web applications.

The admin dashboard

The dashboard is the location where all the users get redirected, once logged into the admin area. Usually, it contains dynamic widget areas with the most important data of your application. The dashboard can play a major role in web applications, compared to blogging or CMS functionality. The dashboard contains a set of default widgets that are mainly focused on the main WordPress features such as posts, pages, and comments. In web applications,

we can remove the existing widgets related to CMS and add application-specific widgets to create a powerful dashboard. WordPress offers a well-defined API to create custom admin dashboard widgets and hence we can create a very powerful dashboard using custom widgets for custom requirements in web applications.

Posts and pages

Posts in WordPress are built for creating content such as articles and tutorials. In web applications, posts will be the most important section to create different types of data. Often, we will choose custom post types instead of normal posts for building advanced data creation sections. On the other hand, pages are typically used to provide the static content of the site. Usually, we have static pages such as About Us, Contact Us, Services, and so on.

Users

User management is a must-use section for any kind of web application. User roles, capabilities, and profiles will be managed in this section by the authorized users.

Appearance

Themes and application configurations will be managed in this section. Widgets and theme options will be the important sections related to web applications. Generally, widgets are used in the sidebars of WordPress sites to display information such as recent members, comments, posts, and so on. However, in web applications, widgets can play a much bigger role as we can use widgets to split the main template into multiple sections. Also, these types of widgetized areas become handy in applications where the majority of features are implemented with AJAX.

The theme options panel can be used as the general settings panel of web applications where we define the settings related to templates and generic site-specific configurations.

Settings

This section involves general application settings. Most of the prebuilt items in this section are suited for blogs and websites. We can customize this section to add new configuration areas related to our plugins, used in web application development.

There are some other sections, such as links, pages, and comments, which will not be used frequently in complex web application development. The ability to add new sections is one of the key reasons for its flexibility.

The role of plugins

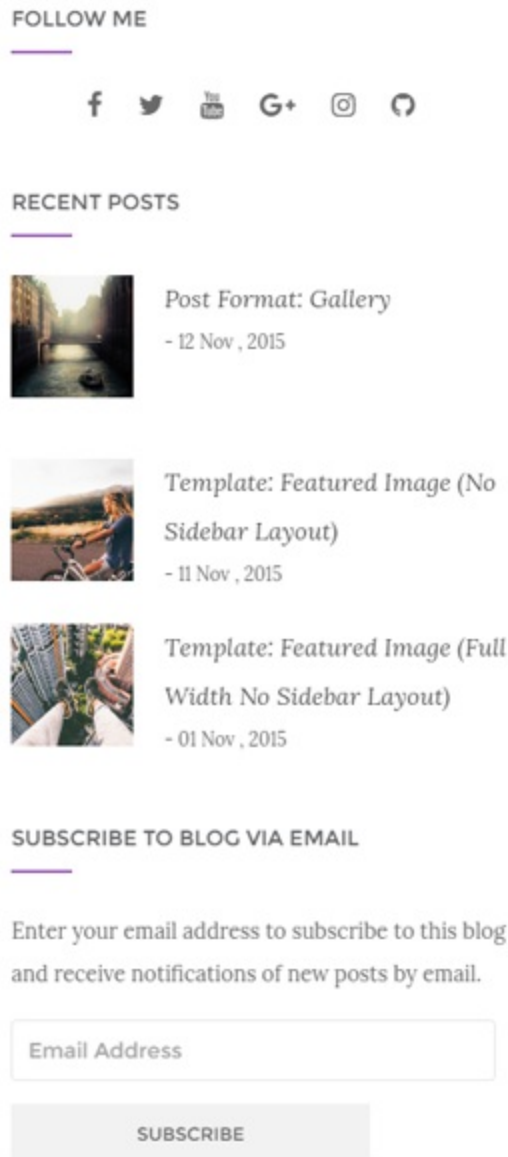
Under normal circumstances, WordPress developers use functions that involve application logic scattered across theme files and plugins. Some developers even change the core files of WordPress, which is considered a very bad practice. In web applications, we need to be much more organized.

In *The role of WordPress themes* section, we discussed the purpose of having a theme for web applications. Plugins will be and should be used to provide the main logic and content of your application. The plugins architecture is a powerful way to add or remove features without affecting the core. Also, we have the ability to separate independent modules into their own plugins, making it easier to maintain. On top of this, plugins have the ability to extend other plugins. Since there are over 40,000 free plugins and a large number of premium plugins, sometimes you don't have to develop anything for WordPress applications. You can just use a number of plugins and integrate them properly to build advanced applications.

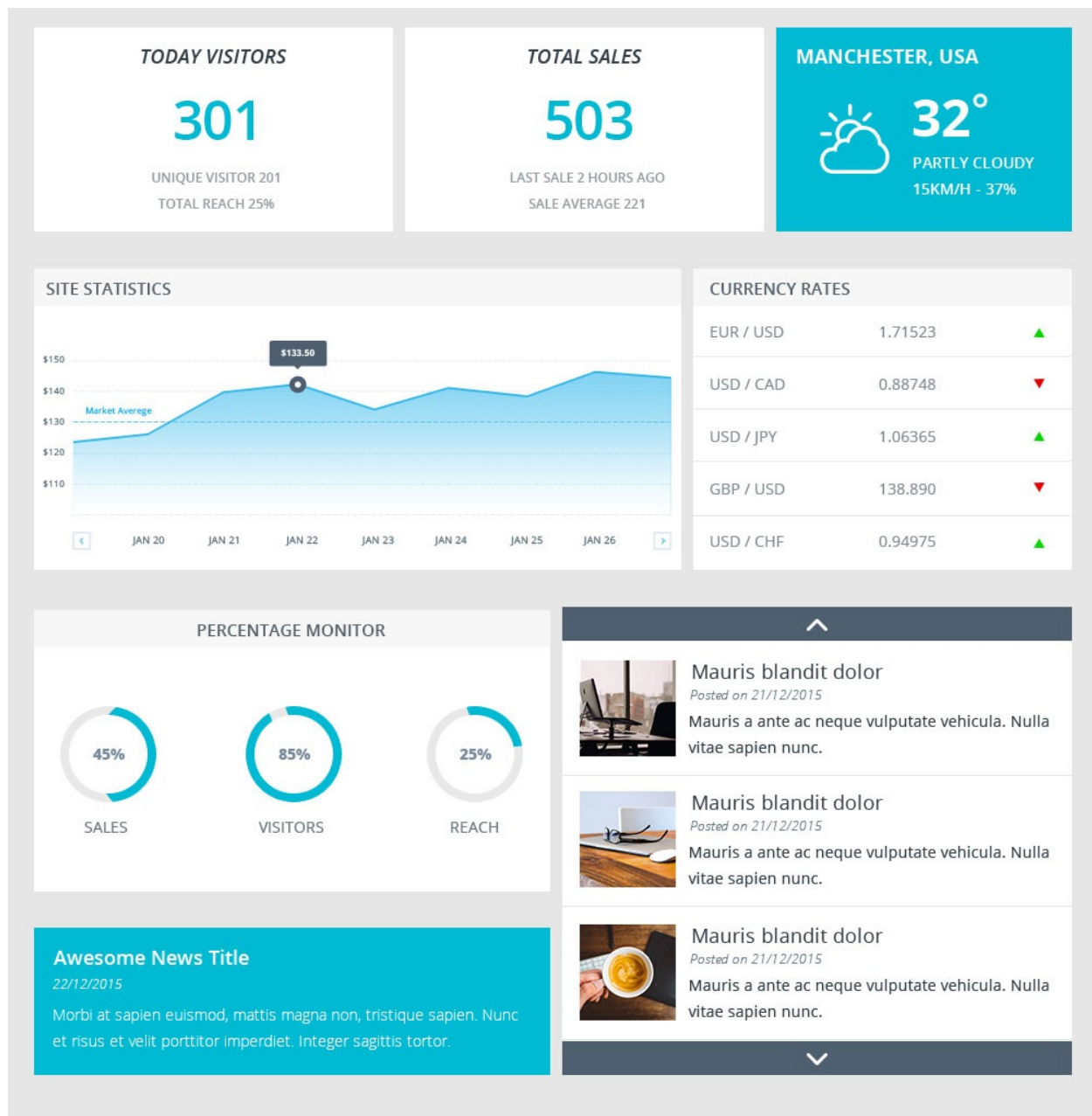
The role of widgets

The official documentation of WordPress refers to widgets as a component that adds content and features to your sidebar. From a typical blogging or CMS user's perspective, it's a completely valid statement. Actually, the widgets offer more in web applications by going beyond the content that populates sidebars. Modern WordPress themes provide a wide range of built-in widgets for

advanced functionality, making it much more easier to build applications. The following screenshot shows a typical widgetized sidebar of a website:



We can use dynamic widgetized areas to include complex components as widgets, making it easy to add or remove features without changing source code. The following screenshot shows a sample dynamic widgetized area. We can use the same technique for developing applications with WordPress:



Throughout these sections, we have covered the main components of WordPress and how they fit into the actual web application development. Now we have a good understanding of the components, we can plan our application developed throughout this book.

A development plan for the forum management application

Typically, a WordPress book consists of several chapters, each of them containing different practical examples to suit each section. In this book, our main goal is to learn how we can build full stack web applications using built-in WordPress features. Therefore, I thought of building a complete application, explaining each and every aspect of web development.

Throughout this book, we will develop an online forum management system for creating public forums or managing a support forum for a specific product or service. This application can be considered as a mini version of a powerful forum system like **bbPress**. We will be starting the development of this application from [Chapter 2, Implementing Membership Roles, Permissions, and Features](#).

Planning is a crucial task in web development, through which we will save a lot of time and avoid potential risks in the long run. First, we need to get a basic idea about the goal of the application, its features and functionalities, and the structure of components to see how it fits into WordPress.

Application goals and target audience

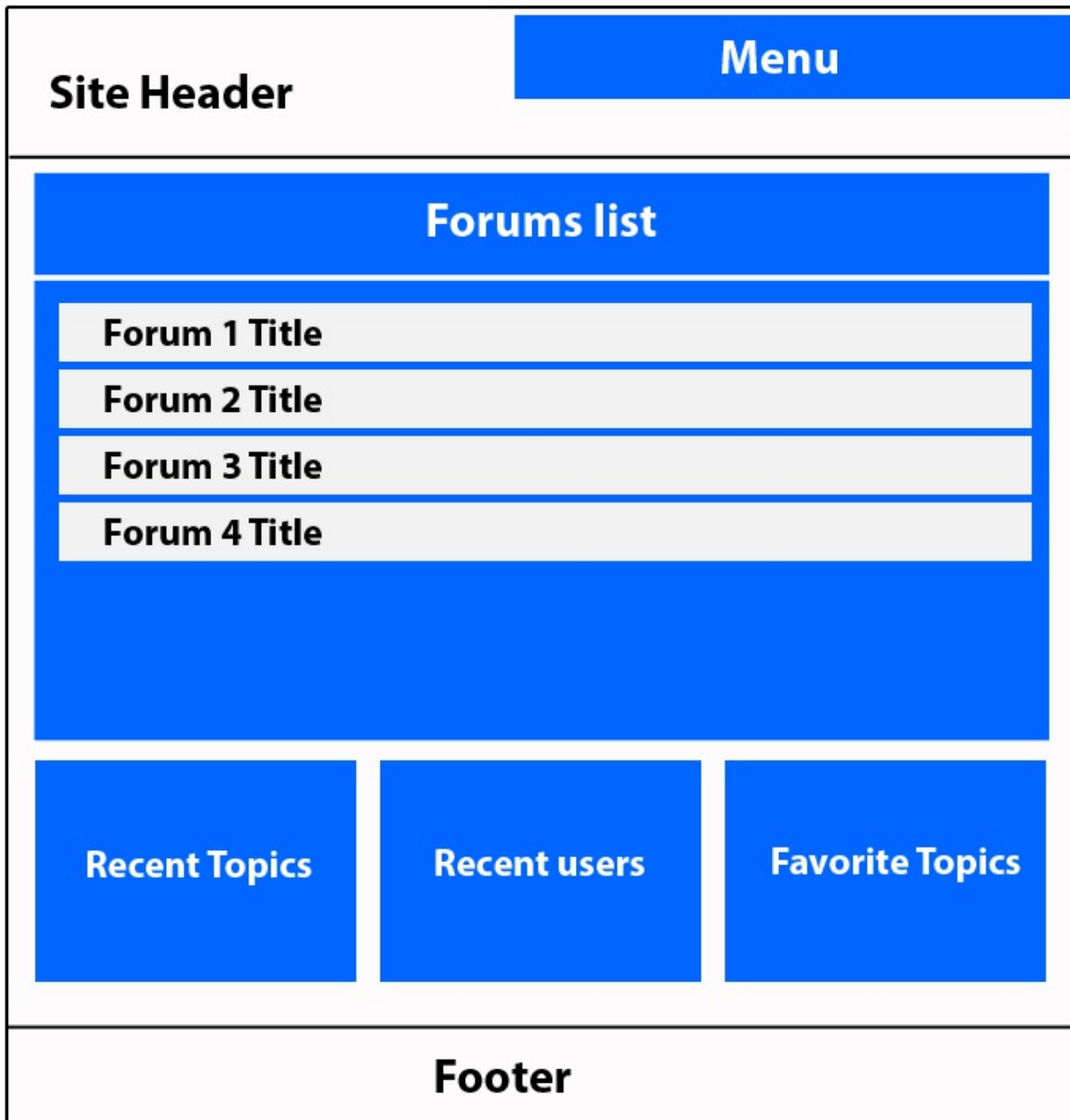
Anyone who is using the Internet on a day-to-day basis knows the importance of online discussion boards, also known as **forums**. These forums allow us to participate in a large community and discuss common matters, either related to a specific subject or a product. The application developed throughout this book is intended to provide a simple and flexible forum management application using a WordPress plugin with the goals of:

- Learning to develop a forum application
- Learning to use features of various online forums
- Learning to manage a forum for your product or service

This application will be targeted towards all the people who have participated in an online forum or used a support system of a product they purchased. I believe that both output of this application and the contents of the book will be ideal for PHP developers who want to jump into WordPress application development.

Planning the application

Basically, our application consists of both frontend and backend, which is common to most web applications. In the frontend, both registered and unregistered users will have different functionalities based on their user roles. The following image shows the structure of our application home page:



The backend will be developed by customizing the built-in admin features of WordPress. Existing and new functionalities of the admin area will be customized based on the user role permissions.

User roles of the application

The application consists of four user roles, including the built-in admin role.

User roles and their respective functionalities are explained in the following section:

- **Admin:** Users of this role manage the application configurations, settings, and capabilities of the users.
- **Moderator:** Users of this role manage the forums and topics. These users can create topics, manage topic statuses, and provide admin level feedback in case of a support forum.
- **Premium Members:** These users gain access to the forums by purchasing a package or subscription. These users will have premium forum features and access to all content without restrictions.
- **Free Members:** These are normal registered users who want to access the forums. These users will have limited access to content and forum features.

Note

Registration is required for all four user roles in the forum management application.

Planning application features and functions

Our main intention for building this application is to learn how WordPress can be adapted to advanced web application development. Therefore, we will be considering various small requirements, rather than covering all aspects of a similar system. Each of the functionalities will be focused on explaining various modules in web applications and the approach of WordPress in building similar functionality.

Let's consider the following list of functions, which we will be developing throughout this book:

- **Forum user profile management:** Users who register for forums will have a profile where they can edit personal details and view their forum topic details without going into the forum.
- **Forums and Topics management:** The application provides the ability to

create forums and topics under these forums. Admin staff will be responsible for creating and closing forum tickets while users will have the ability to create their own tickets.

- **Frontend login and registration:** Typically, web applications contain the login and registration in the frontend, whereas WordPress provides it in the admin area. Therefore, custom implementation of login and registration will be implemented in the application frontend with the support of social login.
- **Manage forum topic permissions:** Various features will be developed to provide restrictions to forum and topic features for different user levels.
- **Joining a Forum:** Users will have features to join a forum and admins will have features to approve/reject user requests.
- **Settings panel:** A comprehensive settings panel will be developed for administrators to configure general application settings from the backend.
- **REST API:** A large number of popular web applications come up with a fully functional API to allow access to third-party applications. In this application, we will be developing a simple API to access the developer details and activities from external sources.
- **Notification service:** A simple notification service will be developed to manage e-mail notifications on new topics in forums and responses to subscribed topics in forums.
- **Responsive design:** With the increase of mobile devices for Internet browsing, more and more applications are converting their apps to suit various devices. So, we will be targeting different devices for a fully responsive design from the beginning of the development process.
- **Third-party libraries:** Throughout the book, we will be creating functionalities such as OpenAuth login, RSS feed generation, and template management to understand the use of third-party libraries in WordPress.

While these are our main functionalities, we will also develop small features and components on top of them to explain the major aspects of web development.

If you are still not convinced, you can take a look at the various types of WordPress powered web applications at

http://www.innovativephp.com/demo/packt/wordpress_applications.

Understanding limitations and sticking to guidelines

As with every framework, WordPress has its limitations in developing web applications. Developers need to understand the limitations before deciding to choose the framework for application development.

In this section, we will learn the limitations while building simple guidelines for choosing WordPress for web development. Let's get started!

- **Lack of support for MVC:** We talked about the architecture of WordPress and its support for MVC in one of the earlier sections. As a developer, you need to figure out ways to work with WordPress in order to suit web applications. If you are someone who cannot work without MVC, WordPress may not be the best solution for your application.
- **Database migration:** If you are well experienced in web development, you will have a pretty good idea about the importance of choosing databases considering the possibilities of migrating to another one in later stages. This can be a limitation in WordPress as it's built to work with MySQL databases. Using it with another database will be quite difficult, if not impossible. So, if you need the database to be migrated to some other database, WordPress will not be the best solution.
- **Security:** This is one of the most important aspects of application development. WordPress is open source and many sites use free plugins without proper security and code checks. Therefore, when proper security measures are not implemented, potential hackers and programs can identify the vulnerabilities and generate security threats. Many people tend to think WordPress is insecure due to such reasons.
- **Performance:** The performance of your application is something we get to experience in the later stages of the project when we go into a live environment. It's important to plan ahead on performance considerations as they can come through internal and external reasons. WordPress has a built-in database structure and we will use it in most of the projects. It's

designed to suit CMS functionality, and sticking with the same tables for different types of projects will not provide the optimized table structure. Therefore, performance might be a limitation for critical applications interacting with millions of records each day, unless you optimize your caching, indexing, and other database optimization strategies.

- **Architecture:** WordPress runs on an event-driven architecture, packed with features. Often developers misuse the hooks without proper planning, affecting the performance of the application. So, you have to be responsible in planning the database and necessary hooks in order to avoid performance overheads.
- **Regular Updates:** WordPress has a very active community involving its development for new features and fixing the issues in existing features. Once a new version of the core is released, plugin developers will also update their plugins to be compatible with the latest version. Hence, you need to perform additional tasks to update the core, themes, and plugins, which can be a limitation when you don't have a proper maintenance team.
- **Object Oriented Development:** Experienced web developers will always look for object-oriented frameworks for development. WordPress started its coding with procedural architecture and is now moving rapidly toward object-oriented architecture. So, there will be a mix of both procedural and object-oriented code. WordPress also uses a hook-based architecture to provide functionality for both procedural and object-oriented codes. Developers who are familiar with other PHP frameworks might find it difficult to come to terms with the mix of procedural and object-oriented code as well as the hook-based architecture. So, you have to decide whether you are comfortable with its existing coding styles.

If you are a developer or designer who thinks these limitations could cause major concerns for your projects, WordPress may not be the right solution for you.

Building a question-answer interface

Throughout the previous sections, we learned the basics of web application frameworks while looking at how WordPress fits into web development. By now, you should be able to visualize the potential of WordPress for application development and how it can change your career as a developer. Being human, we always prefer a practical approach to learning new things over the more conventional theoretical approach.

So, I will complete this chapter by converting default WordPress functionality into a simple question-answer interface, such as Stack Overflow, to give you a glimpse of what we will develop throughout this book.

Prerequisites for building a question-answer interface

We will be using version 4.7.2 as the latest stable version; this is available at the time of writing this book. I suggest that you set up a fresh WordPress installation for this book, if you haven't already done so.

Note

Also, we will be using the Twenty Seventeen theme, which is available with default WordPress installation. Make sure that you activate the Twenty Seventeen theme in your WordPress installation.

First, we have to create an outline containing the list of tasks to be implemented for this scenario:

- Create questions using the admin section of WordPress
- Allow users to answer questions using comments
- Allow question creators to mark each answer as correct or incorrect
- Highlight the correct answers of each question

- Customize the question list to include a number of answers and number of correct answers

Now, it's time to get things started.

Creating questions

The goal of this application is to let people submit questions and get answers from various experts in the same field. First off, we need to create a method to add questions and answers. By default, WordPress allows us to create posts and submit comments to the posts. In this scenario, a post can be considered as the question and comments can be considered as the answers. Therefore, we have the capability of directly using normal post creation for building this interface.

However, I would like to choose a slightly different approach by using the custom post types plugin, which you can find at http://codex.wordpress.org/Post_Types#Custom_Post_Types, in order to keep the default functionality of posts and let the new functionality be implemented separately without affecting the existing ones. We will create a plugin to implement the necessary tasks for our application:

1. First off, create a folder called `wpwa-questions` inside the `/wp-content/plugins` folder and add a new file called `wpwa-questions.php`.
2. Next, we need to add the block comment to define our file as a plugin:

```
/*
Plugin Name: WPWA Questions
Plugin URI: -
Description: Question and Answer Interface using WordPress
Version: 1.0
Author: Rakhitha Nimesh
Author URI: http://www.wpexpertdeveloper.com/
License: GPLv2 or later
Text Domain: wpwa-questions
*/
```

3. Having created the main plugin file, we can now create the structure of

our plugin with the necessary settings as shown in the following code section:

```
if( !class_exists( 'WPWA_Questions' ) ) {
    class WPWA_Questions{
        private static $instance;
        public static function instance() {

            if ( ! isset( self::$instance ) && ! ( self::$instance ) ) {
                self::$instance = new WPWA_Questions();
                self::$instance->setup_constants();
                self::$instance->includes();

                add_action('admin_enqueue_scripts', array($this->load_admin_scripts));
                add_action('wp_enqueue_scripts', array($this->load_scripts));
            }
            return self::$instance;
        }
        public function setup_constants() {
            if ( ! defined( 'WPWA_VERSION' ) ) {
                define( 'WPWA_VERSION', '1.0' );
            }
            if ( ! defined( 'WPWA_PLUGIN_DIR' ) ) {
                define('WPWA_PLUGIN_DIR', plugin_dir_path( __FILE__ ));
            }
            if ( ! defined( 'WPWA_PLUGIN_URL' ) ) {
                define( 'WPWA_PLUGIN_URL', plugin_dir_url( __FILE__ ));
            }
        }
        public function load_scripts(){ }
        public function load_admin_scripts(){ }
        private function includes() { }
        public function load_textdomain() { }
    }
}
```

4. First, we create a class called `WPWA_Questions` as the main class of the question-answer plugin. Then we define a variable to hold the instance of the class and use the instance function to generate an object from this class. This static function and the private instance variables make sure that we only have one instance of our plugin class. We have also included the necessary function calls and filters in this function.
5. These functions are used to handle the most basic requirements of any WordPress plugin. Since they are common to all plugins, we keep these

functions inside the main file. Let's look at the functionality of these functions:

- `setup_constants`: This function is used to define the constants of the application such as version, plugin directory path, and plugin directory URL
 - `load_scripts`: This function is used to load all the plugin specific scripts and styles on the frontend of the application
 - `load_admin_scripts`: This function is used to load all the plugin specific scripts and styles on the backend of the application
 - `includes`: This function is used to load all the other files of the plugin
 - `load_text_domain`: This function is used to configure the language settings of the plugin
6. Next, we initialize the plugin by calling the following code after the class definition:

```
function WPWA_Questions() {  
    global $wpwa;  
    $wpwa = WPWA_Questions::instance();  
}  
WPWA_Questions();
```

7. Having created the main plugin file, we can move into creating a custom post type called `wpwa-question` using the following code snippet. Include this code snippet in your `WPWA_Questions` class file of the plugin:

```
public function register_wp_questions() {  
    $labels = array(  
        'name' => __( 'Questions', 'wpwa_questions' ),  
        'singular_name' => __( 'Question',  
        'wpwa_questions'),  
        'add_new' => __( 'Add New', 'wpwa_questions'),  
        'add_new_item' => __( 'Add New Question',  
        'wpwa_questions'),  
        'edit_item' => __( 'Edit Questions',  
        'wpwa_questions'),  
        'new_item' => __( 'New Question',  
        'wpwa_questions'),
```

```

        'view_item' => ____( 'View Question',
'wpwa_questions' ),
        'search_items' => ____( 'Search Questions',
'wpwa_questions' ),
        'not_found' => ____( 'No Questions found',
'wpwa_questions' ),
        'not_found_in_trash' => ____( 'No Questions found in
Trash', 'wpwa_questions' ),
        'parent_item_colon' => ____( 'Parent Question:',
'wpwa_questions' ),
        'menu_name' => ____( 'Questions', 'wpwa_questions' ),
    );
    $args = array(
        'labels' => $labels,
        'hierarchical' => true,
        'description' => ____( 'Questions and Answers',
'wpwa_questions' ),
        'supports' => array( 'title', 'editor',
'comments' ),
        'public' => true,
        'show_ui' => true,
        'show_in_menu' => true,
        'show_in_nav_menus' => true,
        'publicly_queryable' => true,
        'exclude_from_search' => false,
        'has_archive' => true,
        'query_var' => true,
        'can_export' => true,
        'rewrite' => true,
        'capability_type' => 'post'
    );
    register_post_type( 'wpwa_question', $args );
}

```

This is the most basic and default code for custom post type creation, and I assume that you are familiar with the syntax. We have enabled title, editor, and comments in the support section of the configuration. These fields will act as the roles of question title, question description, and answers. Other configurations contain the default values and hence explanations will be omitted. If you are not familiar, make sure to have a look at documentation on custom post creation at http://codex.wordpress.org/Function_Reference/register_post_type.

Note

Beginner to intermediate level developers and designers tend to include the logic inside the `functions.php` file in the theme. This is considered a bad practice as it becomes extremely difficult to maintain because your application becomes larger. So, we will be using plugins to add functionality throughout this book and the drawbacks of the `functions.php` technique will be discussed in later chapters.

8. Then, you have to add the following code inside the instance function of `WPWA_Questions` class to initialize the custom post type creation code:

```
add_action('init',  
array(self::$instance, 'register_wp_questions'));
```

9. Once the code is included, you will get a new section on the admin area for creating questions. This section will be similar to the posts section inside the WordPress admin. Add a few questions and insert some comments using different users before we move into the next stage.

Before we go into the development of questions and answers, we need to make some configurations so that our plugin works without any issues. Let's look at the configuration process:

1. First, we have to look at the comment-related settings inside **Discussion Settings** in the WordPress **Settings** section. Here, you can find a setting called **Before a comment appears**.
2. Disable both checkboxes so that users can answer and get their answers displayed without the approval process. Depending on the complexity of application, you can decide whether to enable these checkboxes and change the implementation.
3. The second setting we have to change is the **Permalinks**. Once we create a new custom post type and view it in a browser, it will redirect you to a 404 page not found page. Therefore, we have to go to the **Permalinks** section of WordPress **Settings** and update the **Permalinks** using the

Save Changes button. This won't change the **Permalinks**. However, this will flush the rewrite rules so that we can use the new custom post type without 404 errors.

Now, we can start working with the answer-related features.

Customizing the comments template

Usually, the comments section is designed to show the comments of a normal post.

While using comments for custom features such as answers, we need to customize the existing template and use our own designs by performing the following steps:

1. So, open the `comments.php` file inside the **Twenty Seventeen** theme.
2. Navigate through the code and you will find a code section similar to the following one:

```
wp_list_comments( array(
    'avatar_size' => 100,
    'style'       => 'ol',
    'short_ping'  => true,
    'reply_text'  => twentyseventeen_get_svg( array( 'icc
) );
```

3. We need to modify this section of code to suit the requirements of the answers list. The simplest method is to edit the `comments.php` file of the theme and add the custom code changes. However, modifying core theme or plugin files is considered a bad practice since you lose the custom changes on theme or plugin updates. So, we have to provide this template within our plugin to make sure that we can update the theme when needed.
4. Let's copy the `comments.php` file from the Twenty Seventeen theme to the root of our plugin folder. WordPress will use the `wp_list_comments` function inside the `comments.php` file to show the list of answers for each question. We need to modify the answers list in order to include the answer status button. So, we will change the previous implementation to the following in the `comments.php` file we added inside our plugin:

```

<?php
    global $wpwa;

    if (get_post_type( $post ) == "wpwa_question"){
        wp_list_comments:

    }else{
        wp_list_comments( array(
            'avatar_size' => 100,
            'style'        => 'ol',
            'short_ping'   => true,
            'reply_text'   => twentyseventeen_get_svg( array
        ) );
    }
?>

```

5. Here, we will include a conditional check for the post type in order to choose the correct answer list generation function. When the post type is `wpwa_question`, we call the `wp_list_comments` function with the callback parameter defined as `comment_list`, which will be the custom function for generating the answers list.

Note

Arguments of the `wp_list_comments` function can be either an array or string. Here, we have preferred array-based arguments over string-based arguments.

6. We have a custom `comments.php` file inside the plugin. However, WordPress is not aware of the existence of this file and hence will load the original `comments.php` file within the theme. So, we have to include our custom template by adding the following filter code to instance the function:

```

add_filter('comments_template',
    array(self::$instance, 'load_comments_template'));

```

7. Finally, we have to implement the `load_comments_template` function inside the main class of the plugin to use our comments template from the

plugins folder:

```
public function load_comments_template($template){
    return WPWA_PLUGIN_DIR.'comments.php';
}
```

In the next section, we will be completing the customization of the comments template by adding answer statuses and allowing users to change the statuses.

Changing the status of answers

Once the users provide their answers, the creator of the question should be able to mark them as correct or incorrect answers. So, we will implement a button for each answer to mark the status. Only the creator of the questions will be able to mark the answers. Once the button is clicked, an AJAX request will be made to store the status of the answer in the database.

Implementation of the `comment_list` function goes inside the main class of `wpwa-questions.php` file of our plugin. This function contains lengthy code, which is not necessary for our explanations. Hence, I'll be explaining the important sections of the code. It's ideal to work with the full code for the `comment_list` function from the source code folder:

```
function comment_list( $comment, $args, $depth ) {
    global $post;
    $GLOBALS['comment'] = $comment;

    $current_user = wp_get_current_user();
    $author_id = $post->post_author;
    $show_answer_status = false;

    if ( is_user_logged_in() && $current_user->ID == $author_id ) {
        $show_answer_status = true;
    }
    $comment_id = get_comment_ID();
    $answer_status = get_comment_meta( $comment_id,
    "_wpwa_answer_status", true );

    // Rest of the Code
}
```

The `comment_list` function is used as the callback function of the comments list, and hence it will contain three parameters by default. Remember that the button for marking the answer status should be only visible to the creator of the question.

In order to change the status of answers, follow these steps:

1. First, we will get the current logged-in user from the `wp_get_current_user` function. Also, we can get the creator of the question using the global `$post` object.
2. Next, we will check whether the logged-in user created the question. If so, we will set `show_answer_status` to true. Also, we have to retrieve the status of the current answer by passing the `comment_id` and `_wpwa_answer_status` keys to the `get_comment_meta` function.
3. Then, we will have to include the common code for generating a comments list with the necessary condition checks.
4. Open the `wpwa-questions.php` file of the plugin and go through the rest of the `comment_list` function to get an idea of how the comments loop works.
5. Next, we have to highlight the correct answers of each question and I'll be using an image as the highlighter. In the source code, we use the following code after the header tag to show the correct answer highlighter:

```
<?php
    // Display image of a tick for correct answers
    if ( $answer_status ) {
        echo "<div class='tick'><img src='".plugins_url(
        'img/tick.png', __FILE__ )."' alt='Answer Status'
        /></div>";
    }
?>
```

6. In the source code, you will see a `<div>` element with the class `reply` for creating the comment reply link. We will need to insert our answer button status code right after this, as shown in the following code:

```
<div>
    <?php
    // Display the button for authors to make the answer
    if ( $show_answer_status ) {
        $question_status = '';
```

```

        $question_status_text = '';
        if ( $answer_status ) {
            $question_status = 'invalid';
            question_status_text = __('Mark as
Incorrect', 'wpwa_questions');
        } else {
            $question_status = 'valid';
            $question_status_text = __('Mark as
Correct', 'wpwa_questions');
        }
    ?>
    <input type="button" value="<?php echo
$question_status_text; ?>" class="answer-status
answer_status-<?php echo $comment_id; ?>"
data-ques-status="<?php echo $question_status; ?>" />
    <input type="hidden" value="<?php echo $comment_id; ?>"
class="hcomment" />
    <?php
    }
    ?>
</div>

```

7. If the `show_answer_status` variable is set to true, we get the comment ID, which will be our answer ID, using the `get_comment_ID` function. Then, we will get the status of answer as true or false using the `_wpwa_answer_status` key from the `wp_commentmeta` table.
 8. Based on the returned value, we will define buttons for either **Mark as Incorrect** or **Mark as Correct**. Also, we will specify some CSS classes and HTML5 data attributes to be used later with jQuery.
 9. Finally, we keep the `comment_id` in a hidden variable called `hcomment`.
-
10. Once you include the code, the button will be displayed for the author of the question, as shown in the following screen:

What is WordPress REST API?

Recent WordPress versions mentions about the availability of a REST API. Can anyone let me know what is REST API and how we can use it in our websites?

Edit

One Reply to “What is WordPress REST API?”



Answered by admin

The WordPress REST API provides API endpoints for WordPress data types that allow developers to interact with sites remotely by sending and receiving JSON (JavaScript Object Notation) objects. When you send content to or make a request to the API, the response will be returned in JSON. This enables developers to create, read and update WordPress content from client-side JavaScript or from external applications, even those written in languages beyond PHP.

Reply

Mark as Correct

11. Next, we need to implement the AJAX request for marking the status of the answer as true or false.

Before this, we need to see how we can include our scripts and styles into WordPress plugins. We added empty functions to include the scripts and styles while preparing the structure of this plugin. Here is the code for including

custom scripts and styles for our plugin inside the `load_scripts` function we created earlier. Copy the following code into the `load_scripts` function of the `wpwa-questions.php` file of your plugin:

```
public function load_scripts() {
    wp_enqueue_script( 'jquery' );
    wp_register_script( 'wpwa-questions', plugins_url(
'js/questions.js', __FILE__ ), array('jquery'), '1.0', TRUE );
    wp_enqueue_script( 'wpwa-questions' );
    wp_register_style( 'wpwa-questions-css', plugins_url(
'css/questions.css', __FILE__ ) );
    wp_enqueue_style( 'wpwa-questions-css' );
    $config_array = array(
        'ajaxURL' => admin_url( 'admin-ajax.php' ),
        'ajaxNonce' => wp_create_nonce( 'ques-nonce' )
    );
    wp_localize_script( 'wpwa-questions', 'wpwaconf', $config_array );
}
```

WordPress comes with an action hook built-in called `wp_enqueue_scripts` for adding JavaScript and CSS files. The `wp_enqueue_script` action is used to include script files into the page while the `wp_register_script` action is used to add custom files. Since jQuery is built-in to WordPress, we can just use the `wp_enqueue_script` action to include jQuery into the page. We also have a custom JavaScript file called `questions.js`, which will contain the functions for our application.

Inside JavaScript files, we cannot access the PHP variables directly. WordPress provides a function called `wp_localize_script` to pass PHP variables into script files. The first parameter contains the handle of the script for binding data, which will be `wp_questions` in this scenario. The second parameter is the variable name to be used inside JavaScript files to access these values. The third and final parameters will be the configuration array with the values.

Then, we can include our `questions.css` file using the `wp_register_style` and `wp_enqueue_style` functions, which will be similar to JavaScript, file inclusion syntax, we discussed previously. Now, everything is set up properly to create the AJAX request.

Saving the status of answers

Once the author clicks the button, the status has to be saved to the database as true or false depending on the current status of the answer. Let's go through the jQuery code located inside the `questions.js` file for making the AJAX request to the server:

```
jQuery(document).ready(function($) {
    $(".answer-status").click( function() {
        $(body).on("click", ".answer-status" , function() {

            var answer_button = $(this);
            var answer_status  = $(this).attr("data-ques-status")
            // Get the ID of the clicked answer using hidden field
            var comment_id = $(this).parent().find(".hcomment").val();
            var data = {
                "comment_id":comment_id,
                "status": answer_status
            };

            $.post( wpwaconf.ajaxURL, {
                action:"mark_answer_status",
                nonce:wpwaconf.ajaxNonce,
                data : data,
            }, function( data ) {
                if("success" == data.status){
                    if("valid" == answer_status){
                        answer_button.val("Mark as Incorrect");
                        answer_button.attr("
data-ques-status","invalid");
                    }else{
                        answer_button.val("Mark as Correct");
                        answer_button.attr("
data-ques-status","valid");
                    }
                }
            }, "json");
        });
    });
});
```

Let's understand the implementation of saving the answer status. This code snippet executes every time a user clicks the button to change the status. We get the current status of the answer by using the `data-ques-status` attribute of the button. Next, we get the ID of the answer using the hidden field with

`hcomment` as the CSS class. Then, we send the data through an AJAX request to the `mark_answer_status` action.

Once the AJAX response is received, we display the new status of the answer and update the `data` attribute of the button with the new status.

Note

The important thing to note here is that we have used the configuration settings assigned in the previous section, using the `wpwacnf` variable. Once a server returns the response with success status, the button will be updated to contain the new status and display text.

The next step of this process is to implement the server-side code for handling the AJAX request. First, we need to define AJAX handler functions using the WordPress `add_action` function. Since only logged-in users are permitted to mark the status, we don't need to implement the `add_action` function for `wp_ajax_nopriv_{action}`. Let's add the AJAX action to the instance function of the main class of the plugin:

```
add_action('wp_ajax_mark_answer_status',  
array(self::$instance, 'mark_answer_status'));
```

Then, we need to add the `mark_answer_status` function inside the main class. Implementation of the `mark_answer_status` function is given in the following code:

```
function mark_answer_status() {  
    $data = isset( $_POST['data'] ) ? $_POST['data'] : array()  
    $comment_id = isset( $data["comment_id"] ) ?  
absint($data["comment_id"]) : 0;  
    $answer_status = isset( $data["status"] ) ? $data["status"]  
0;  
    // Mark answers in correct status to incorrect  
    // or incorrect status to correct  
    if ("valid" == $answer_status) {  
        update_comment_meta( $comment_id, "_wpwa_answer_status"  
    } else {  
        update_comment_meta( $comment_id, "_wpwa_answer_status",
```

```
}  
echo json_encode( array("status" => "success") );  
exit;  
}
```

We can get the necessary data from the `$_POST` array and use it to mark the status of the answer using the `update_comment_meta` function. This example contains the most basic implementation of the data saving process. In real applications, we need to implement necessary validations and error handling.

Now, the author who asked the question has the ability to mark answers as correct or incorrect. So, we have implemented a nice and simple interface for creating a question-answer site with WordPress. The final task of the process will be the implementation of the questions list.

Generating a question list

Usually, WordPress uses the `archive.php` file of the theme for generating post lists of any type. We can use a file called `archive-{post type}.php` for creating different layouts for different post types. Here, we will create a customized layout for our questions. The simplest solution is to copy the archive file and create a new file called `archive-{post type}.php` inside the theme. However, it's not the ideal way to create a custom template due to the reasons we discussed while creating the `comments.php` file. So, we have to use a plugin specific template and override the default behavior of the archive file. Refer to the following steps:

1. Make a copy of the existing `archive.php` file of the TwentySeventeen theme, copy it to the root folder of the plugin, and rename it `questions-list-template.php`. Here, you will find the following code section:

```
get_template_part( 'template-parts/post/content', get_
```

2. The TwentySeventeen theme uses a separate template for generating the content of each post type. We have to replace this with our own template inside the plugin. The `get_template_part` function is only used to load the theme specific template. So, we can't use it to load a template file from a plugin. The solution is to include the template file using the

following code:

```
require WPWA_PLUGIN_DIR . 'content.php';
```

3. Next, we have to create the content template by copying the `content.php` file of the theme from the `template-parts/post/content` folder to the root folder of our plugin.
4. Finally, we need to consider the implementation of the `content.php` file. In the questions list, only the question title will be displayed, and therefore, we don't need the content of the post. So, we have to either remove or comment the `the_excerpt` and `the_content` functions of the template. We can comment the following line within this template:

```
the_content( sprintf(
__ ( 'Continue reading<span class="screen-reader-text"> "%s
' ) );
```

5. Then, we will create our own metadata by adding the following code to the `<div>` element with the `entry-content` class:

```
<div class="answer_controls"><?php
comments_popup_link(__('No Answers &darr;', 'twentyseventeen'),
__('1 Answer &darr;', 'responsive'), __('% Answers &darr;',
'twentyseventeen')); ?>
</div>
<div class="answer_controls">
<?php wpwa_get_correct_answers(get_the_ID()); ?>
</div>
<div class="answer_controls">
<?php echo get_the_date(); ?>
</div>
<div style="clear: both"></div>
```

The first container will make use of the existing `comments_popup_link` function to get the number of answers given for the questions.

6. Then, we need to display the number of correct answers of each question. The custom function called `get_correct_answers` is created to get the correct answers. The following code contains the implementation of the `get_correct_answers` function inside the main class of the plugin:

```
function get_correct_answers( $post_id ) {
```

```

    $args = array(
        'post_id'    => $post_id,
        'status'     => 'approve',
        'meta_key'   => '_wpwa_answer_status',
        'meta_value' => 1,
    );
    // Get number of correct answers for given question
    $comments = get_comments( $args );
    printf(__( '<cite class="fn">%s</cite> correct answers: ' . count( $comments ) );
}

```

We can set the array of arguments to include the conditions to retrieve the approved answers of each post, which contains the correct answers. The number of results generated from the `get_comments` function will be returned as correct answers.

We have completed the code for displaying the questions list. However, you will still see a list similar to the default posts list. The reason for that is we created a custom template and WordPress is not aware of its existence. So, the default template file is loaded instead of our custom template.

7. We need to override the existing archive template of WordPress by adding the following action to the instance function of the main plugin class:

```

add_filter( 'archive_template',
    array(self::$instance, 'questions_list_template' ) );

```

8. Here, we have the implementation of the `questions_list_template` function inside the main class of the plugin:

```

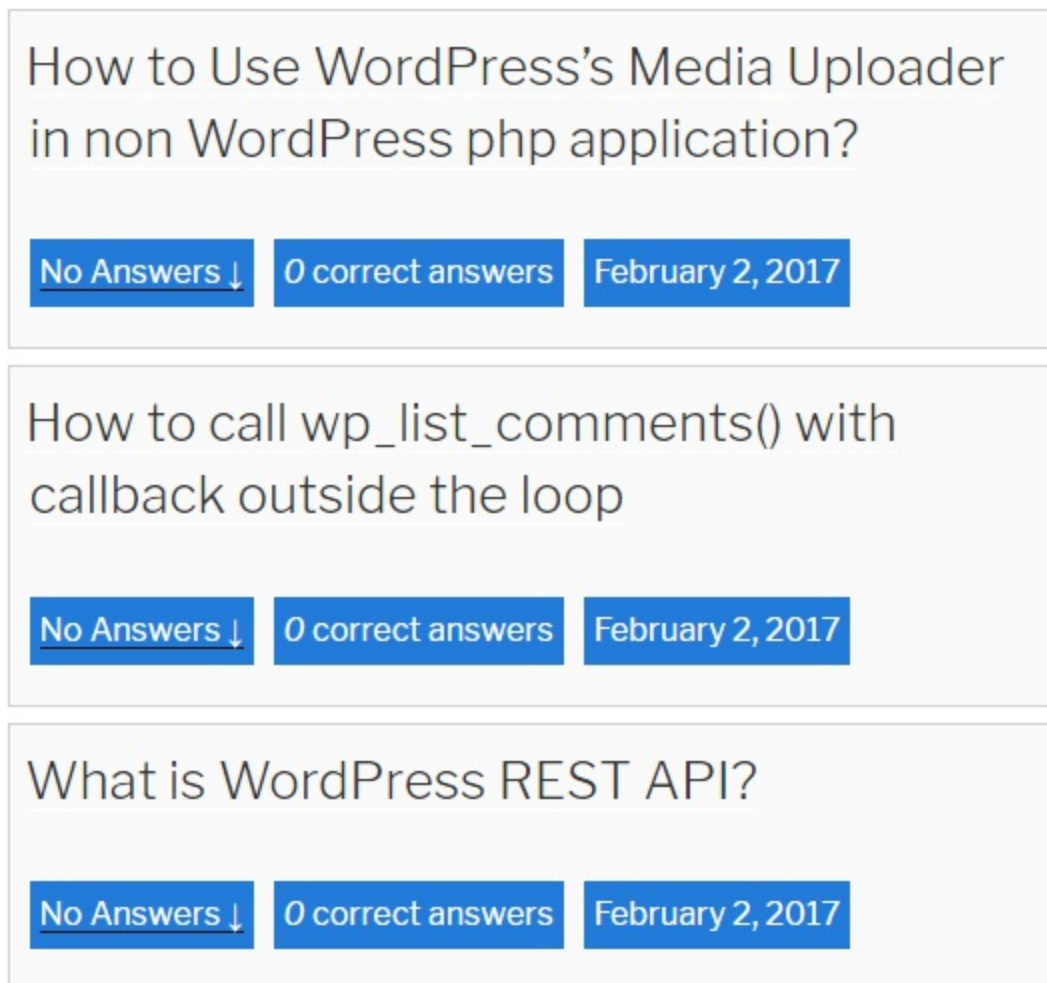
public function questions_list_template($template){
    global $post;

    if ( is_post_type_archive ( 'wpwa_question' ) ) {
        $template = WPWA_PLUGIN_DIR . '/questions-list-template.php';
    }
    return $template;
}

```

This function overrides the archive template when the questions list is displayed and keeps the default template for posts lists or any other custom post type lists.

Now, you should have a question list similar to the following screenshot:



Throughout this section, we looked at how we can convert the existing functionalities of WordPress for building a simple question-answer interface. We took the quick and dirty path for this implementation by mixing HTML and PHP code inside both, themes and plugins.

Note

I suggest that you go through the [Chapter 1](#), *WordPress as a Web Application Framework*, `source code` folder and try this implementation on your own test server. This demonstration was created to show the flexibility of WordPress. Some of you might

not understand the whole implementation. Don't worry, as we will develop a web application from scratch using a detailed explanation in the following chapters.

Enhancing features of the questions plugin

In the previous sections, we illustrated how to quickly adapt WordPress into different kinds of implementations by customizing its core features. However, I had to limit the functionality to the most basic level in order to keep this chapter short and interesting for beginners. If you are willing to improve it, you can try other new features of such an application. Here, we will be looking at some of the enhancements to this plugin and how we can use core WordPress features to implement them.

Customizing the design of questions

This is one of the major requirements in such an application. Here, we have a very basic layout and it's difficult to know whether this is a question or just a normal post. Consider the following screenshot for a well-designed question interface:

[Home](#) / [Questions](#) / [Fashion And Beauty](#) / [Fashion](#)



Remember that we customized the comments list for adding new options. Similarly, we can customize the design of questions to create an interface such as the previous screenshot by using a separate template file for the questions custom post type. We have to create a file called `single-wpwa_question.php` and change the design and functionality as we want.

Categorizing questions

Categories allow us to filter the results and limit them to a certain extent. It's an essential feature in the question-answer application so that users can directly view questions related to their topic instead of browsing all the questions. WordPress provides categories by default. However, these categories are mainly intended for posts. Therefore, we have to use custom taxonomies to create categories for custom post types such as questions. More details about taxonomies will be discussed in the following chapters.

Approving and rejecting questions

Currently, our application can post questions without any approval process. However, this is not the ideal implementation as it can create a lot of spam questions. In a well-built application, there should be a feature to approve/reject questions when needed. This feature can be easily built with WordPress admin lists. We have a list of questions in the backend, and we can use the **Bulk Actions** dropdown on top to add custom actions and implement this feature.

Adding star rating to answers

The author who asked the question can mark the answers as correct or incorrect. However, there can be scenarios where answers are not marked as correct are more suitable than answers marked as correct. Therefore, we can introduce star rating features to answers so that the public can rate the answers. The person who looks for the answer can consider the rating before choosing answers. Implementation of this requirement is similar to the functionality of the **Mark as Correct** button. We have to get a JS plugin for star rating and integrate to the interface through the `comment_list` function. Then we can use AJAX to mark the status of answers.

We have looked at some of the possible enhancements to such an application.

You can think of many more such functionalities and implement them in your own version.

In the following chapters, we'll see the limitations in this approach in complex web applications and how we can organize things better to write quality and maintainable code.

Summary

Our main goal was to find how WordPress fits into web application development. We started this chapter by identifying the CMS functionalities of WordPress. We explored the features and functionalities of popular full stack frameworks and compared them with the existing functionalities of WordPress.

Then, we looked at the existing components and features of WordPress and how each of those components fits into a real-world web application. We also planned the forum management application requirements and identified the limitations in using WordPress for web applications.

Finally, we converted the default interface into a question-answer interface in a rapid process using existing functionalities, without interrupting the default behavior of WordPress and themes.

By now, you should be able to decide whether to choose WordPress for your web application, visualize how your requirements fit into the components of WordPress, and identify and minimize the limitations.

In the next chapter, we will start developing the forum management application with the user management. Before we go there, I suggest that you research the user management features of other frameworks and look at your previous projects to identify the functionalities.

Chapter 2. Implementing Membership Roles, Permissions, and Features

The success of any web application or website depends heavily on its user base. There are plenty of great web applications that go unnoticed by many people due to the lack of user interaction. As developers, it's our responsibility to build a simple and interactive user management process, as visitors decide whether to stay on or leave a website by looking at the complexity of initial tasks such as registration and login.

In this chapter, we will be mainly concentrating on adapting existing user management functionalities into typical web applications. In order to accomplish our goal, we will execute some tasks outside the box to bring user management features from the WordPress core to WordPress themes.

While striving to build a better user experience, we will also take a look at some advanced aspects of web application development, such as routing, controlling, and custom templating.

In this chapter, we will cover the following topics:

- Introducing user management
- Understanding user roles and capabilities
- Creating a simple MVC-like process
- Implementing registration...

Introduction to user management

Usually, popular PHP development frameworks such as **Zend**, **CakePHP**, **CodeIgniter**, and **Laravel** don't provide a built-in user module. Developers tend to build their own user management modules and use them across many projects of the same framework. WordPress offers a built-in user management system to cater to common user management tasks found in web applications. Such things include the following:

- Managing user roles and capabilities
- A built-in user registration functionality
- A built-in user login functionality
- A built-in forgot password functionality

Developers are likely to encounter these tasks in almost all web applications. In most cases, these features and functions can be effectively used without significant changes in the code. However, web applications are much more advanced and hence we might need various customizations on these existing features. It's important to explore the possibility of extending these functions in order to be compatible with advanced application requirements. In the following sections, we will learn how to extend these common functionalities to suit various scenarios.

Getting started with user roles

In simple terms, user roles define the types of users in a system. WordPress offers built-in functions for working with every aspect of user roles. In this section, we will look at how we can manage these tasks by implementing the user roles for our application. We can create a new user role by calling the `add_role` function. The following code illustrates the basic form of user role creation:

```
$result = add_role( 'role_name', 'Display Name', array(
    'read' => true,
    'edit_posts' => true,
    'delete_posts' => false
) );
```

The first parameter takes the role name, which is a unique key to identify the role. The second parameter will be the display name, which will be shown in the admin area. The final parameter will take the necessary capabilities of the user role. You can find out more about existing user roles at http://codex.wordpress.org/Roles_and_Capabilities. In this scenario, `read`, `edit_posts`, and `delete_posts` will be the capabilities while `true` and `false` are used to enable and disable status.

Creating application user roles

As planned earlier, we will need...

Understanding user capabilities

Capabilities can be considered as tasks which users are permitted to perform inside the application. A single user role can perform many capabilities, while a single capability can be performed by many user roles. Typically, we use the term "access control" for handling capabilities in web applications. Let's see how capabilities work inside WordPress.

Creating your first capability

Capabilities are always associated with user roles and hence we cannot create new capabilities without providing a user role. Let's look at the following code for associating custom capabilities with our member user roles created earlier in the *Creating application user roles* section:

```
public function add_application_user_capabilities() {  
    $role = get_role( 'wpwaf_premium_member' );  
    $role->add_cap( 'follow_forum_activities' );  
  
    $role = get_role( 'wpwaf_free_member' );  
    $role->add_cap( 'follow_forum_activities' );  
}
```

First, we need to retrieve the user role as an object using the `get_role` function. Then, we can associate new or existing capability using the `add_cap...`

Registering application users

An administration panel is built into the WordPress framework, allowing us to log in through the admin screen. Therefore, we have a registration area which can be used to add new users by providing a username and e-mail. In web applications, registration can become complex, compared to the simple registration process in WordPress. Let's consider some typical requirements of the web application registration process in comparison with WordPress:

- **User-friendly interface:** An application can have different types of user roles. Until registration is completed, everyone is treated as a normal application user with the ability to view public content. Typically, users are used to seeing fancy registration forms inside the main site rather than a completely different login area such as with WordPress. Therefore, we need to explore the possibilities of adding WordPress registration to the frontend.
- **Requesting detailed information:** Most web applications will have at least four or five fields in the user registration form for grabbing detailed information about the user. Therefore, we need to look at the...

Implementing frontend registration

Fortunately, we can make use of the existing functionalities to implement registration from the frontend. We can use a regular HTTP request or AJAX-based technique to implement this feature. In this book, I will focus on using normal HTTP POST requests instead of using AJAX. Our first task is to create the registration form in the frontend.

There are various ways to implement such forms in the frontend. Let's look at some of the possibilities as described in the following section:

- Shortcode implementation
- Page template implementation
- Custom template implementation

Now, let's look at the implementation of each of these techniques.

Shortcode implementation

Shortcodes are the quickest way to add dynamic content to your pages. In this situation, we need to create a page for registration. Therefore, we need to create a shortcode that generates the registration form.

Note

We can add shortcodes to our application by including them inside `functions.php` file of the theme or within any custom plugin. I recommend adding these shortcodes in a custom plugin since you will loose the additional code in `functions.php`...

Creating a login form in the frontend

The frontend login can be found in many WordPress websites, including small blogs. Usually, we place the login form in the sidebar of the website. In web applications, user interfaces are more complex and different compared to normal websites. Hence, we will implement a full-page login screen as we did with registration. First, we need to update our controller with another case for login, as shown in the following code:

```
switch ( $control_action ) {  
    // Other cases  
    case 'login':  
        do_action( 'wpwaf_before_login_form' );  
        $wpwaf->login->display_login_form();  
        break;  
}
```

We have used a new class called `login` to call this function. So we need to create a new class called `WPWAF_Login` inside the `classes` folder and do the object creation and file inclusion inside the `WPWAF_Forum` class as we did in previous occurrences. Once it's completed, we can add the `display_login_form` function to the new class using the following code:

```
public function display_login_form(){  
    global $wpwaf_login_params;  
    ...  
}
```

Essential user management features for web applications

As discussed throughout this chapter, **user management** is one of the most important tasks in web application development. Only limited types of applications can be built without needing user management. So we discussed some of the core user management features available in WordPress and we will be exploring more built-in user features in upcoming chapters.

The built-in user management features of WordPress focus more on the administrative side. There are very limited user management features for the user in the frontend of the application. Therefore, often we need to develop additional user management features for the frontend or use custom plugins that provide such features. In this section, we will be discussing some of the most important and commonly used user management features, as follows:

- Frontend login and registration
- Custom profile fields
- Private data
- Search and member list
- Frontend profile

Let's get started.

Frontend login and registration

This is one of the most important aspects of user management as it's the gateway for building the user base of your application....

Implementing user management features with popular plugins

In the previous section, we discussed the need for custom user-related functionality in managing large web applications. We can use our own code to create all of the mentioned features to suit web applications. However, implementing these features from scratch is a highly time-consuming task requiring perfect planning. So we need to consider the type of application and the requirements before deciding to implement these features from scratch.

Usually, it's a wise decision to use an existing plugin-based solution in such scenarios, unless user management is the core feature of your application and existing solutions are not capable of handling your application requirements. We can find dozens of free and premium user management plugins by professional developers to manage the user-related tasks of your application. In the next section, we will be looking at three such plugins among the dozens available online.

BuddyPress

BuddyPress is one of the most popular plugins in the WordPress plugin directory. It is used to build applications with a user community and interactions...

Time to practice

In this chapter, we have implemented a simple registration and login functionality from the frontend. Before we have a complete user creation and authentication system, there are plenty of other tasks to be completed. So, I would recommend you try out the following tasks in order to be comfortable with implementing such functionalities for web applications:

- Create a frontend functionality for lost passwords
- Block the default WordPress login page and redirect it to our custom page
- Include extra fields in the registration form

Summary

In this chapter, we have explored the basics of user roles and capabilities related to web application development. We were able to choose the user roles for our application considering the various possibilities provided by WordPress.

Next, we learned how to create custom routes in order to achieve an MVC-like process using the frontend controller and custom template system.

Finally, we looked at how we can customize the built-in registration and login process in the frontend to cater to advanced requirements in web application development.

By now, you should be capable of defining user roles and capabilities to match your application, creating custom routers for common modules, implementing custom controllers with custom template systems, and customizing the existing user registration and authentication process.

In the next chapter, we will look at how we can adapt the existing database of WordPress into web applications, while planning the database for a forum management application. Stay tuned for another exciting chapter.

Chapter 3. Planning and Customizing the Core Database

In the previous chapter, we looked at user management and permission related features of WordPress, while implementing basic user creation features. We can now move into one of the most important aspects of an application.

Generally, a database acts as the primary location from which your web application data will be accessible from frontend interfaces or third-party systems. Planning and designing the database should be one of the highest priority tasks in the initial stages of a project. As developers, we have the opportunity to design the database from scratch in many web applications. WordPress comes with a pre-structured database, and therefore the task of planning the table structure and adapting to existing tables becomes much more complex than everyone thinks. Throughout this chapter, we will focus on the basics of planning and accessing databases for web applications. This chapter contains important content for the rest of the book and is theoretical compared to other chapters.

In this chapter, we will cover the following topics:

- Understanding the WordPress database

• Understanding the WordPress database

Typical full stack web development frameworks don't come with a predefined database structure. Instead, these frameworks focus on the core foundation of an application while allowing the developers to focus on the application-specific features. On the other hand, WordPress provides a preplanned database structure with a fixed set of tables. WordPress is built to function as a content management system, and therefore it can be classified as a product rather than a pure development framework. The WordPress core database is designed to power the generic functionalities of a CMS. Therefore, it's our responsibility to use our skills to make it work as an application development framework.

The WordPress database is intended to work with MySQL, and therefore we need to have a MySQL database set up before installing WordPress. On successful installation, WordPress will create 11 database tables to cater for the core functionality with the default MySQL table engine.

Note

MyISAM was used as the default MySQL table engine prior to version 5.5.5, and this has been changed to InnoDB from version 5.5...

• Exploring the role of existing tables

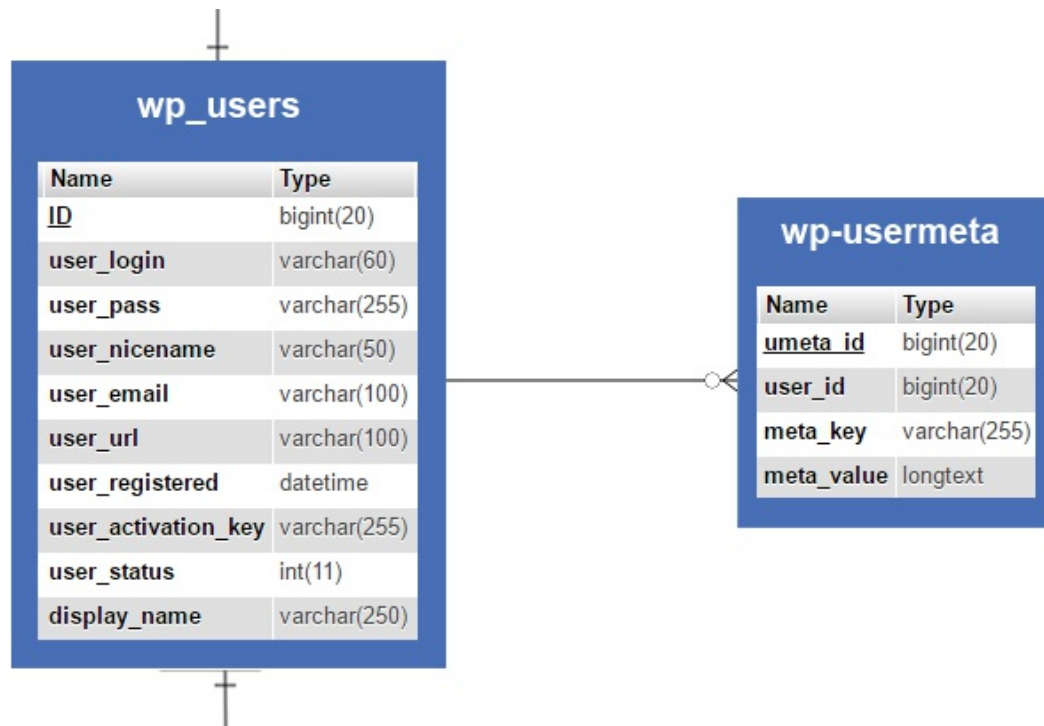
Assuming that most of you are existing WordPress developers, you will have a solid understanding of an existing database table structure. However, I suggest that you continue with this section, as web applications can present a different perspective on using these tables. Based on the functionality, we will categorize the existing tables into four sections, as follows:

- User-related tables
- Post-related tables
- Term-related tables
- Other tables

Let's look at how each table fits into these categories, and look at their roles in web applications.

User-related tables

This category consists of two tables that contain the user-related information of your application. Let's take a look at the relationship between user-related tables before moving onto the explanation:



The two tables shown in the preceding diagram are as follows:

- **wp_users**: All the registered users will be stored in this table with their basic details, such as name, e-mail, username, and password.
- **wp-usermeta**: This table is used to store additional information about the users as key-value pairs. User roles and capabilities can be considered as the...

• **Adapting existing tables in web applications**

Unlike CMSes, web applications have the possibility of scaling infinitely as they become popular and stable. Such systems can contain hundreds of database tables to cater to various requirements. Here, we are trying to build such applications using this popular CMS framework. Therefore, we need to figure out which features we can build using existing tables and which features need their own table structures.

We should be trying to maximize the use of existing tables in every possible scenario to get the most out of WordPress. Built-in database access functions are optimized to work directly with existing tables, allowing us to minimize the time for implementation. On the other hand, we need to write custom queries from scratch to work with newly created tables. Let's find out the possible ways of adapting the existing tables using the four categories we discussed in the previous section.

User-related tables

In web applications, the user table plays the same role as in a normal CMS. Therefore, we don't have to worry about changing the default functionality. Any other user-related...

• Extending the database with custom tables

A default WordPress database can be extended by any number of custom tables to suit our project's requirements. The only thing we have to consider is the creation of custom tables over existing ones. There are two major reasons for creating custom tables:

- **Difficulty of matching data to existing tables:** In the previous section, we considered real application requirements and matched the data to existing tables. Unfortunately, it's not practical in every scenario. Consider a system where the user purchases books from a shopping cart. We need to keep all the payment and order details for tracking purposes, and these records act as transactions in the system. There is no way that we can find a compatible table for this kind of requirement. Such requirements will be implemented using a collection of custom tables.
- **Increased data volume:** As I mentioned earlier, the posts table plays a major role in web applications. When it comes to large-scale applications with a sizeable amount of data, it's not recommended to keep all the data in a posts table. Let's assume that we are building a product...

• Planning the forum application tables

The forum management system developed throughout this book will make use of existing tables in every possible scenario. However, it's hard to imagine even an average web application without using custom tables, so here we will identify the possible custom tables for our system. You might need to revisit the planning section in the *Development plan for forum management application* section of [Chapter 1](#), *WordPress as a Web Application Framework*, in order to remind yourself of the system requirements. We planned to create a functionality to allow free and premium members to subscribe to follow topic activities in the system. Let's discuss the requirement in detail to identify the potential tables we'll need.

Note

Forum users will be able to create their own topics and respond to topics of other users. It's important to receive notifications on the topics, since users can't log into the forum and check topics regularly, so users will be allowed to subscribe to the topics they are interested in and get notifications when updates are available on those topics.

This is a very simple and practical...

• Querying the database

As with most frameworks, WordPress provides a built-in interface for interacting with the database. Most of the database operations will be handled by the `wpdb` class located inside the `wp-includes` directory. The `wpdb` class will be available inside your plugins and themes as a global variable and provides access to all the tables inside the WordPress database, including custom tables.

Note

Using the `wpdb` class for CRUD operations is straightforward with its built-in methods. A complete guide for using the `wpdb` class can be found at

http://codex.wordpress.org/Class_Reference/wpdb.

Querying the existing tables

WordPress provides well-optimized built-in methods for accessing the existing database tables. Therefore, accessing these tables becomes straightforward. Let's see how basic **Create, Read, Update, Delete (CRUD)** operations are executed on existing tables.

Inserting records

All the existing tables contain a prebuilt insert method for creating new records. The following list illustrates a few of the built-in insert functions:

- `wp_insert_post`: This creates a new post or page in the `wp_posts` table. If this is used on...

• Limitations and considerations

We have less flexibility with the WordPress built-in database than when we design a database from scratch. Limitations and features unique to WordPress need to be understood clearly to make full use of the framework and avoid potential bottlenecks. Let's find out some of the WordPress-specific features and their usage in web applications.

Transaction support

In advanced applications, we can have multiple database queries, which need to be executed inside a single process. We have to make sure that either all queries get executed successfully or none of them get executed, to maintain the consistency of the data. This process is known as **transaction management** in application development. In simple website development, we rarely get such requirements for handling transactions. As mentioned earlier, MySQL version 5.5 upwards uses **InnoDB** as the table engine, and therefore we are able to implement transaction support. However, WordPress doesn't offer any libraries or functions to handle transactions, and therefore all transaction handling should be implemented manually.

Post revisions

WordPress provides an...

• Summary

Understanding the WordPress database is the key to building successful web applications. Throughout this chapter, we looked at the role of existing tables and the need for custom database tables through practical scenarios.

Database querying techniques and limitations were introduced with the necessary examples. By now, you should have a clear understanding of choosing the right type of table for your next project. We used a theoretical approach with practical scenarios to learn the basics of database design and implementation inside WordPress.

The real excitement begins in the next chapter, where we start the development of our main modules in web applications using the building blocks of WordPress. Stay tuned!

• Chapter 4. Building Blocks of Web Applications

The majority of WordPress-powered systems are either simple websites or blogs. Adapting WordPress for building complex web applications can be a complex task for beginner developers who are used to working with simple websites every day. Understanding the process of handling web application-specific functions becomes vital in such scenarios.

Managing data is one of the most important tasks in web applications. WordPress offers a concept called custom post types for modeling application data and backend interfaces. I believe this is the foundation of most web applications and, hence, named this chapter *Building Blocks of Web Applications*.

While exploring the advanced use cases of custom post type implementations, we will get used to popular web development techniques such as modularizing, template management, data validations, and rapid application development in a practical process.

In this chapter, we will cover the following topics:

- Introduction to custom content types
- Planning custom post types for an application
- Implementing the custom post type settings
- Validating post creation

- ◦ **Introduction to custom content types**

In WordPress terms, custom content types are referred to as custom post types. The term custom post type misleads some people to think of them as different types of normal posts. In reality, these post types can model almost anything in real web applications. This is similar to collections in other web frameworks such as **Ruby on Rails** or **Meteor.js**. These post types are stored in the normal posts table and this could well be the reason behind its conflicting naming convention.

Prior to the introduction of custom post types, we only had the ability to use normal posts with custom fields to cater to advanced requirements. The process of handling multiple post types was a complex task. The inability to manage different post types in their own lists and the inability to add different fields to different posts are some of the limitations with the old process. With the introduction of custom post types, we now have the ability to separate each different type of the post type to act as a model to cater to complex requirements. The demand for using these custom post types to build complex applications...

• Planning custom post types for an application

Having got a brief introduction to custom post types and their roles in web applications, we will find the necessary custom post types for our forum application. The majority of our application and the data will be based on these custom post types. So, let's look at the detailed requirements of a forum application.

The main purpose of this application is to let users create topics and discuss them with other users or get answers from the administrative staff. Therefore, we have to look at two important components: forums and topics. Now, try to visualize the subsections for each of these components. It is obvious that we can sort these components into two custom post types. The following sections illustrate the detailed subcomponents of each of these models.

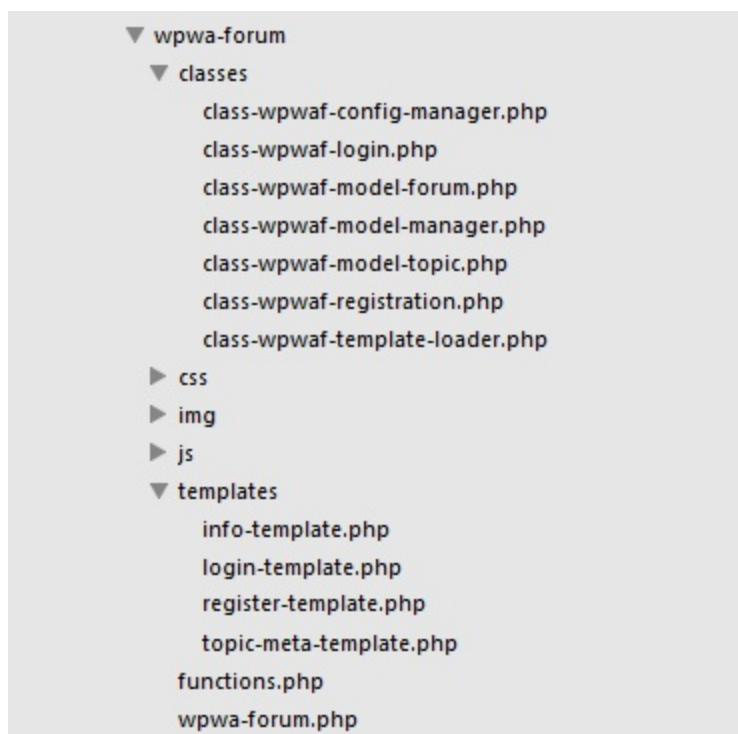
Forums

A forum is an online discussion where members can discuss important topics with other members. This is the model that holds the members of the forum and the discussed topics as well as the replies. We will limit the implementation of forums to some common fields in order to cover the different areas of custom post types. The...

• Implementing custom post types for a forum application

In this section, we will extend the plugin developed in [Chapter 3](#), *Planning and Customizing Core Database*, and implement the custom post type-related functionality. First, we have to create a new file called `class-wpwap-model-manager.php` inside the `classes` directory of the `wpwa-forum` plugin.

Most web applications will be larger in scale compared to normal websites or blogs. Implementing all custom post functionalities in one file is not the most ideal or practical thing to do. So, our plan here is to keep the initialization and generic configurations in the main file, while separating each of the custom post types into their own class files. Before we go any further, I would like you to have a look at the updated folder structure of the plugin using the following screenshot:



Now, let's go through each of the new files and folders to identify their roles:

- `class-wpwap-model-manager.php`: this includes the main initialization and configuration code for the models
- `class-wpwap-model-forum.php`: this includes the main initialization and configuration code for the forum model
-

• What is a template engine?

The template engine is a library or framework that separates logic from template files. These libraries provide their own syntaxes for passing the necessary values to the template from the controllers or models. Once successfully implemented, we shouldn't have complex code inside template files other than simple `if-else` statements and loops.

There are plenty of open source template engines available for PHP. **Smarty**, **Mustache**, and **Twig** are some of the popular ones among them. However, integrating this type of template engine in WordPress is a complex task, compared to using it in other PHP frameworks. The architecture of WordPress is different from any other PHP framework, as it drives on action hooks and filters. Therefore, the integration needs to be capable of handling WordPress-specific things such as actions, filters, widgets, template tags, and so on.

Note

The Twig templates engine created by SensioLabs is one of my favorites, as it offers many unique features compared to other template engines. If you are not familiar with template engines, I suggest that you look at the Twig documentation at

• Persisting custom field data

You already know that default fields and taxonomies are automatically saved to the database on post publish. In order to complete the topic creation process, we need to save the custom field data to the meta tables. As usual, we have to update the constructor of the `WPWAF_Model_Topic` class to add the necessary actions to match the following code:

```
public function __construct() {  
    // Instance variable initializations  
    // Other actions  
    add_action('save_post', array($this, 'save_topic_meta'  
}
```

WordPress doesn't offer an out-of-the-box solution for form validation, as most websites don't have complex forms. This becomes a considerable limitation in web applications. So, let's explore the possible workarounds to reduce these limitations. The action `save_post` inside the constructor will only be called once the post is saved to the database with the default field data. We can do the necessary validations and processing for custom fields inside the function defined for the `save_post` action. Unfortunately, we cannot prevent the post from saving when the form is not validated...

• Introduction to post type templates

A post type template is one of the most valuable features added in recent versions of WordPress. This template allows us to design and style different posts and custom post types with different templates. In [Chapter 2, *Implementing Membership Roles, Permissions, and Features*](#), we discussed page templates and their importance. This is a similar feature for posts instead of pages.

In our forum application, we have a custom post type called topics. In order to display topics on the frontend, we need a template different from the normal post template. So the earlier versions of WordPress allowed us to use the `single-{post_type}.php` file of the theme to design custom post types differently from normal posts. However, this technique only allows us to display all our topics with the same design. Assume that we want to display normal topics with one design and sticky topics with another design. This is not possible with earlier versions of WordPress without modifying the theme files. The introduction of post type templates has made it possible to have unlimited designs for different posts of the same...

• Introducing custom post type relationships

In general, we use relational databases in developing applications, where our models will be matched to separate database tables. Each model will be related to one or more other modules. However, in WordPress, we have all the custom post types stored in the posts table. Hence, it's not possible to create relationships between different post types with the existing functionality.

Note

WordPress developers around the world have been conducting discussions to get the post relationship capability built into the core. Even though the response from WordPress core development seems positive, we still don't have it on the core version as of 4.7. You can have a look at this interesting discussion at

<http://make.wordpress.org/core/2013/07/28/potential-roadmap-for-taxonomy-meta-and-post-relationships/>.

Since this is one of the most important aspects of web application development, we have no choice but to look for custom solutions. We have two simple solutions for adding post relationships in our applications:

- Assigning post relationships using custom meta fields
- Using third-party plugins for adding...

• Pods framework for custom content types

Up until this point, we have explored the advanced usage of WordPress custom post types by considering the perspective of web applications. Although custom post types provide immense power and flexibility for web applications, manual implementation is a time-consuming task with an awful amount of duplicate and complex code. One of the major reasons for using WordPress for web development is its ability to build things rapidly. Therefore, we need to look for quicker and more flexible solutions beyond manual custom post implementations.

Pods is a custom content type management framework, which has been becoming popular among developers in recent years. Most of the tedious functionalities are baked into the framework, while providing us with a simpler interface for managing custom content types. Apart from simplifying the process, Pods provides an extensive list of functionalities that are not available with default WordPress administrative screens.

Let's consider some of the key features of the Pods framework, compared to the manual implementation of custom post types. The Pods framework allows...

• Implementing custom post type features with popular plugins

Throughout this chapter, we developed common custom post features needed for web applications. Even the implementation of common features from scratch needed considerable effort. As applications grow, we need more custom post types, hence we need features such as managing settings through interfaces, advanced post field types, and managing field saving/validations without coding. In such situations, it's a wise decision to use plugins with these advanced features rather than building everything from scratch.

So, we discussed Pods in the previous section as it's one of the best custom post type management plugins available. Here, are we going to look at two popular alternatives, so that you can compare and use the right solution for your application.

Custom Post Type UI

This is a highly popular plugin by **WebDevStudios** with over 400,000 active installs in the WordPress plugin directory. This plugin can be used to create and manage custom post types and custom taxonomies with all the available settings. So, we can change the way custom post types work without needing to change...

• Time to practice

In this chapter, we discussed the advanced usages of custom post types to suit web application functionalities. Now, I would recommend that you try out the following tasks in order to evaluate the things you learned in this chapter:

- In the `post_relationships` section, we enabled relationships using the `join_topics_to_forums` function. With the use of this function, we can generate a new problem considering the possibility of extending. Find the issue and try to solve it by yourself.
- We enabled relationships between custom post types. Research the possibilities of including metadata for relationships.

• Summary

In this chapter, we tackled custom post types to learn the advanced usages within web applications. Generally, custom posts act as the core building blocks of any type of complex web application. We went through the basic code of custom post-related functionality, while developing the initial forums and topics post types of the forum management application.

We were able to explore advanced techniques, such as modularizing custom post types, and template engines to cater to common web application requirements. Also, we had to go through a tough process for validating form data due to the limited support offered by WordPress.

We learned the importance of relating custom post types using the `Posts 2 Posts` plugin. Finally, we explored the possibilities of improving the custom post type management process with the use of an amazing framework called Pods.

In the next chapter, we will see how to use the WordPress plugin beyond the conventional use by implementing pluggable and extendable plugins. Until then, get your hands dirty by playing with custom post type plugins.

• Chapter 5. Implementing Application Content Restrictions

In the previous chapter, we looked at the creation of web application content using custom post types and managing various custom post type related features. We can now move into conditionally restricting the content created in previous chapters. Content is the most important aspect of any application. Providing the same features to all users limits your application from gaining exposure as well as causing management nightmares. Therefore, we need at least two user types to manage the application features and use the application features.

WordPress doesn't provide built-in features for restricting content and hence we have to develop everything from scratch or use an existing content restriction plugin. In this chapter, we will be implementing some of the common restriction types while discussing the process for implementing advanced restrictions.

On the one hand, content restrictions provide great benefits to your application and on the other hand, they might cause loop holes in the application, creating major security threats. So, we have a great responsibility in...

• Introduction to content restrictions

Generally, we refer to information in textual, visual, or aural forms as content in websites. In web applications, we can consider content as anything that is used by the administrators of the application or an application's users. The following are some of the commonly used content types in WordPress applications:

- Posts and pages
- Custom post types
- Widgets
- Menus

In applications or websites, we can't provide every feature to everyone. So, we need to identify the various types of users in our application and ascertain what each of these user types can do within our application. Creating limitations of what each user can see within our application is referred to as content restrictions or content permissions.

In WordPress, we need to log in to the backend to create a post or change any settings. This is the most basic form of content restriction, where we limit the backend features to administrative staff with proper authentication. In web applications, we have to manage content restrictions on both the frontend as well as the backend.

In the upcoming sections, we will be looking at various types of...

• Practical usage of content restrictions

In normal websites, we provide content access to anyone who visits the website. Modern web applications are built on top of basic content restriction types as well as some innovative restriction types to attract more users to the application. Usage of these restrictions varies based on the type of application and the type of content.

We have to identify the type of restriction to be implemented in an application and the benefits expected from applying the restriction. So, let's look at some of the common uses of content restrictions and their benefits:

- **Restricting individual posts or the entire site:** We use this restriction to keep the privacy of your content from unintended users or to gain monetary benefits for some part of the content.
- **Restricting content behind a social Locker:** We use this type of restriction to increase the exposure of the site in social media to attract more customers and promote your application.
- **Restricting content behind e-mail subscriptions:** This technique is ideal for capturing the e-mails of your application users so that you can send notification about your...

• Understanding restriction levels

We have two important aspects in content restrictions. First, we need to identify different content types that need to be restricted and then identify which user types are restricted from accessing this content. So, we group the users and define specific restriction levels for these user groups. Let's identify the most commonly used restriction levels in WordPress web applications:

- User roles-based restrictions
- User groups-based restrictions
- Membership plans-based restrictions
- Unique password-based restrictions

Let's take a look at how these restriction levels are implemented.

User roles-based restrictions

User roles is a default WordPress feature and it's the only technique that we can use as a restriction level without needing custom coding or third-party plugins. As we discussed in [Chapter 2, *Implementing Membership Roles, Permissions, and Features*](#), WordPress gives you six built-in user roles. These user roles also have built-in restrictions to the backend of the application, making it easier to manage restrictions on the backend content.

The implementation of frontend restrictions requires minimum...

• Implementing content restrictions in posts/pages

Posts and pages are the core content types used in any WordPress website. As we discussed in [Chapter 4, *Building Blocks of Web Applications*](#), custom post types play a major role compared to posts and pages in websites. So, understanding how to implement restrictions on these three content types is key to developing a quality application with proper authorization. These content types can be restricted in many common and innovative ways. We will be looking at the most common restrictions on these content types to suit any web application:

- **Shortcode-based restrictions:** This technique is used to restrict a part of the post/page/custom post type from different user levels. We use a shortcode, capable of handling different restrictions, and place it on a post editor. Any content placed between the opening and closing shortcode tags will be restricted based on the restriction level.
- **Individual post/page restrictions:** This technique is used to restrict individual posts/pages/custom post types instead of restricting part of the post. We define the restrictions in a meta box inside the post and...

• Enabling restrictions on WordPress core features

Generally, posts and pages are the common content used for applying restrictions in applications. However, it's important to understand restrictions on other core WordPress features as we need to integrate all the features in order to build a quality application. So, we are going to look at some of the core WordPress features and techniques for applying restrictions.

Restrictions on posts

We have already discussed several content restriction techniques on posts. Apart from these restrictions, we can also implement global post restrictions and post category restrictions:

- **Global post restrictions:** We can restrict all posts or all posts of a single custom post type at once without applying restrictions on individual posts
- **Category restrictions:** We can restrict all posts from a given category at once

Both types of restrictions can be managed with the `template_redirect` action we used in previous sections.

Restrictions on searches

The built-in search features of WordPress allow us to search posts, pages, and custom post types. In some situations, we may want to restrict the content types that...

• **Supplementary content restriction types and techniques**

Up to this point, we discussed the most commonly used content restriction types and their usage. There are many other supplementary restriction types used in modern web applications. We are going to look at some of the important ones among the dozens of restrictions available.

Restrictions on custom generated content

This is one of the most important restriction types and there are limited plugins that support restrictions on custom content. Custom generated content is generally provided by plugins and hence applying restrictions might be difficult in a normal procedure. In [Chapter 2, Implementing Membership Roles, Permissions, and Features](#), we created a custom path called `/user/login` and `/user/registration`. So, the content generated from these custom URLs can't be restricted with normal content restriction types. The following are some of the popular plugins with custom generated content:

- `Woocommerce`
- `BuddyPress`

These popular plugins provide built-in actions and filters for the custom content and hence we can use them to apply restrictions on some of the custom content. In...

• Useful plugins for content restrictions

We implemented some of the common restriction types in web applications and discussed the usage of various other restriction types. We kept the implementation to a basic level in order to let you easily understand the core concept. In real web applications, we will need more flexibility and control over the restrictions. So, we are going to look at some of the plugins with advanced content restrictions and settings to cater to advanced application requirements.

Restrict Content

This is a popular free plugin by Pippin Williamson with over 10,000 active installs. Primary functionality of this plugin is the ability to add restrictions to registered users. We can use this plugin to add shortcode-based restrictions and individual post/page/custom post type restrictions. Apart from these two main features, this plugin also provides the ability to restrict content on your WordPress RSS feeds. This is a simple solution when you want to apply basic content restrictions on your site without manual coding.

You can download and check the features of the free version of this plugin at

• Time to practice

In this chapter, we discussed the basic and advanced techniques for implementing content restrictions. Now, I would recommend that you try out the following tasks in order to evaluate the things you learned in this chapter:

- We enabled user role-based restrictions using the shortcodes technique. Implement the same shortcode-based technique for user capabilities.
- We added individual post/page restrictions for individual posts. Try to implement the same technique for user roles, before checking the source codes for this chapter.

• Summary

Content restriction is one of the significant features in applications where you have different type of users. We began this chapter by learning the importance and practical usage of content restrictions. Also, we looked at various user levels used in restrictions.

We implemented the most common restriction types on our forum application while understanding how posts can be restricted in various lower and higher levels. The real power of content restrictions comes with the integration of all the features in an application and hence restrictions on built-in WordPress features were discussed.

Finally, we went through the supplementary content restriction features used in modern applications to increase the exposure while identifying plugins that can be used to automate these restrictions without manual development.

In the next chapter, we will see how to use the WordPress plugin beyond conventional use by implementing pluggable and extendable plugins. Until then, get your hands dirty by playing with custom post types plugins.

• Chapter 6. Developing Pluggable Modules

Plugins are the heart of WordPress, which makes web applications possible. WordPress plugins are used to extend its core features as independent modules. As a developer, it's important to understand the architecture of WordPress plugins and design patterns in order to be successful in developing large-scale applications.

Anyone who has basic programming knowledge can create plugins to meet application-specific requirements. However, it takes considerable effort to develop plugins that are reusable across a wide range of projects. In this chapter, we will build a few plugins to demonstrate the importance of reusability and extensibility. WordPress developers who don't have good experience in web application development shouldn't skip this chapter as plugins are the most important part of web application development.

I will assume that you have sound knowledge of basic plugin development using existing WordPress features in order to be comfortable understanding the concepts discussed in this chapter.

In this chapter, we will cover the following topics:

- A brief introduction to WordPress plugins

- ◦ **A brief introduction to WordPress plugins**

WordPress offers one of the most flexible plugin architectures, alongside other similar frameworks such as **Joomla** and **Drupal**. The existence of over 40,000 plugins in the WordPress plugin directory proves the vital role of plugins. In typical websites, we create simple plugins to tweak the theme's functionalities or application-specific tasks. The complexity of web applications forces us to modularize the functionalities to enhance their maintainability. Most application developers will be familiar with the concept of the open-closed principle.

Note

The open-closed principle states that the design and writing of code should be done in a way that new functionality should be added with minimum changes in the existing code. The design of the code should be done in a way to allow the addition of new functionalities as new classes, keeping as much of the existing code unchanged as possible. You can find more information at <http://www.oodeesign.com/open-close-principle.html>.

We can easily achieve the open-closed principle with WordPress plugins. Plugins can be developed to be open for new features...

• WordPress plugins for web development

By default, WordPress offers blogging and CMS functionalities to cater to simple applications. In real web applications, we need to develop most things using the existing features provided by WordPress. In short, all those web application-related features will be implemented using plugins. In this book, we have created a main plugin for the forum management application. This plugin was intended to provide forums and topics specific tasks in our application and is hence not reusable in different projects. We need more and more reusable plugins and libraries as developers who are willing to take long journeys in web application development with WordPress. In this section, we will discuss the various types of such plugins:

- Reusable libraries
- Extensible plugins
- Pluggable plugins

Note

Keep in mind that plugins are categorized into the previously mentioned types conceptually, for the sake of understanding the various features of plugin development. Don't try to search using the preceding categories as they are not defined anywhere.

Creating reusable libraries with plugins

These days, web developers will...

• Tips for developing extendable plugins

Plugins are used as the primary components in building web applications. In normal circumstances, we have to use existing plugins as well as develop our own plugins. So it's important to improve the extendibility of plugins to adapt to the custom requirements of different applications. So, let's look at some of the important tips for developing extendable plugins:

- Add custom actions with `do_action` before and after executing major actions in your plugin, such as registration, login, data saving, and so on.
- Add custom filters with `apply_filters` when you have an output for displaying or variables and objects that are worth modifying.
- Use two parameters for filters and keep the second parameter as an array or object, allowing you to add custom attributes without breaking the functionality of existing implementations. It's not wise to add many parameters.
- Use one or two parameters for actions and keep one parameter as an array or object, allowing you to add custom attributes without breaking the functionality of existing implementations. It's not wise to add many parameters.
- Place common functions...

• Time to practice

Developing high quality plugins is the key to success in web development using WordPress. In this chapter, we introduced various techniques for creating extensible plugins. Now, it's time for you to take one step further by exploring the various other ways of using plugins. Take some time to try out the following tasks to get the best out of this chapter:

- In this chapter, we integrated a media uploader to custom fields and restricted the file types using actions. However, restrictions will be global across all types of posts. Try to make the restrictions based on custom post types and custom fields. We should be able to customize the media uploader for each field.
- Use the `wp_handle_upload` function to implement a manual file uploading to cater to complex scenarios which cannot be developed using the existing media uploader.
- Create pluggable plugins using inheritance without considering pluggable functions.

• Summary

We began this chapter by exploring the importance and architecture of WordPress plugins. Here, we identified the importance of creating reusable plugins by categorizing such plugins into three types called reusable libraries, extensible plugins, and pluggable plugins.

While building these plugins, we learned the use of actions, filters, and pluggable functions within WordPress. The integration of a media uploader was very important for web applications, which works with file-related functionalities. Also, we looked at how we can extend our content restrictions module with custom actions and filters. Finally, we discussed some important tips for building extendable plugins.

In the next chapter, [Chapter 7](#), *Customizing the Dashboard for Powerful Backends*, we will master the use of the WordPress admin section to build highly customizable backends using existing features. Stay tuned, as this will be important for developers who are planning to use WordPress as a backend system without using its theme.

• **Chapter 7. Customizing the Dashboard for Powerful Backends**

Usually, full-stack PHP frameworks don't provide built-in admin sections. So, developers have to build an application's backend from scratch. WordPress is mainly built on an existing database, which makes it possible to provide a prebuilt admin section. Most of the admin functionality is developed to cater to the existing content management functionality. As developers, you won't be able to develop complex applications without having the knowledge of extending and customizing the capabilities of existing features.

The structure and content of this chapter is built in a way that enables the tackling of the extendable and customizable components of admin screens and features. We will be looking at the various aspects of an admin interface while building the forum management application.

In this chapter, we will cover the following topics:

- Understanding the admin dashboard
- Customizing the admin toolbar
- Customizing the main navigation menu
- Adding features with custom pages
- Building options pages
- Using feature-packed admin list tables
- Adding content restrictions to admin list tables

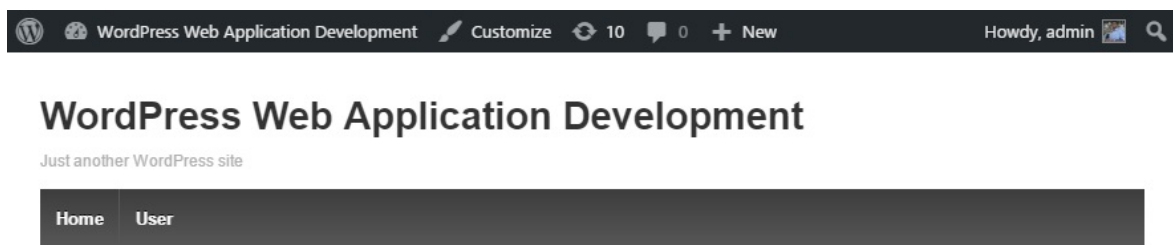
• ◦ **Understanding the admin dashboard**

WordPress offers one of the most convenient admin sections among similar frameworks, such as **Drupal** and **Joomla**, for building any kind of application. In the previous chapters, we looked at the administration screens related to various areas such as user management, custom post types, and posts. Here, we will look at some of the remaining components from the perspective of web application development. Let's identify the list of sections we will consider:

- The admin toolbar
- The main navigation menu
- Option and menu pages
- Admin list tables
- Responsive design capabilities

• Customizing the admin toolbar

The admin toolbar is located at the top of the admin screen to allow direct access to the most used parts of your website. Once you log in, the admin toolbar will be displayed on the admin dashboard as well as at the frontend. Typical web applications contain separate access menus for the frontend and backend. Hence, web developers might find it difficult to understand the availability of the admin toolbar at the frontend from the perspective of the functionality as well as the look and feel. In web applications, it's your choice whether to remove the admin toolbar from the frontend or customize it to provide a useful functionality. In this section, we will look at both methods to simplify your decision about the admin toolbar. First, let's preview the admin toolbar at the frontend with its default settings, as shown in the following screenshot:



Let's add a new class called `class-wpwapf-dashboard.php` to our main forum manager plugin for functionalities in the admin section:

```
class WPWAF_Dashboard {  
    public function __construct() { }  
}  
?>
```

As usual, we need to include the

• Customizing the main navigation menu

In WordPress, the main navigation menu is located on the left-hand side of the screen where we have access to all the sections of the application. In a similar way to the admin toolbar, we have the ability to extend the main navigation menu with customized versions.

Let's start by adding the admin menu invoking an action to the constructor:

```
add_action('admin_menu', array( $this, 'customize_main_na
```

Now, consider the initial implementation of the `customize_main_navigation` function:

```
public function customize_main_navigation() {
    global $menu, $submenu;
    echo "<pre>"; print_r($menu); echo "</pre>"; exit;
}
```

The preceding code uses the global variable `$menu` for accessing the available main navigation menu items. Before we begin the customizations, it's important to get used to the structure of the menu array using a `print_r` statement. A part of the output generated from the `print_r` statement is shown in the following section:

```
Array
(
    [2] => Array
        (
            [0] => Dashboard
            [1] => read
            [2] => index.php
            [3] =>
        )
    ...
)
```

• Adding features with custom pages

WordPress was originally created as a blogging platform and evolved into a content management system. Hence, most of the core functionality is implemented on the concept of posts and pages. In web applications, we need to go way beyond these basic posts and pages to build quality applications. Custom menu pages play a vital role in implementing custom functionalities within the WordPress admin dashboard. Let's consider the two main types of custom pages in the default context:

- **Custom menu pages:** Generally, these pages are blank by default. We need to implement the interface as well as implementation for catering for custom requirements that can't take advantage of the core features of WordPress.
- **Options pages:** These are used to manage the options of the application. Even though options pages are generally used for theme options, we can manage any type of applications-specific settings with these pages.

• Building options pages

The theme options page is implemented in each and every WordPress theme by default. However, the design and the available options may vary based on the quality and features of the theme. We selected a theme called **Responsive** (<https://wordpress.org/themes/responsive/>) for the purpose of this book. So, let's take a look at the default theme options panel of the Responsive theme using the following screenshot:

Responsive Theme Options

Instructions Help Translate Showcase More Themes

Get More Such Free Themes By CyberChimps

Theme Elements

Logo Upload

Home Page

Default Layouts

Social Icons

CSS Styles

Scripts

Disable breadcrumb list? ☐ Check to disable

Disable Call to Action Button? ☐ Check to disable

Enable minified css? ☐ Check to enable

Enable Blog Title ☐ Check to enable

Title Text

Copyright Text

Display Powered By WordPress Link ☐

Save Options Restore Defaults Upgrade

The Responsive theme uses its own layout structure for the options page. Generally, we have two ways of creating options pages for plugins:

- Using custom menu pages with our own template and processing
- Using WordPress options pages with the options and settings API

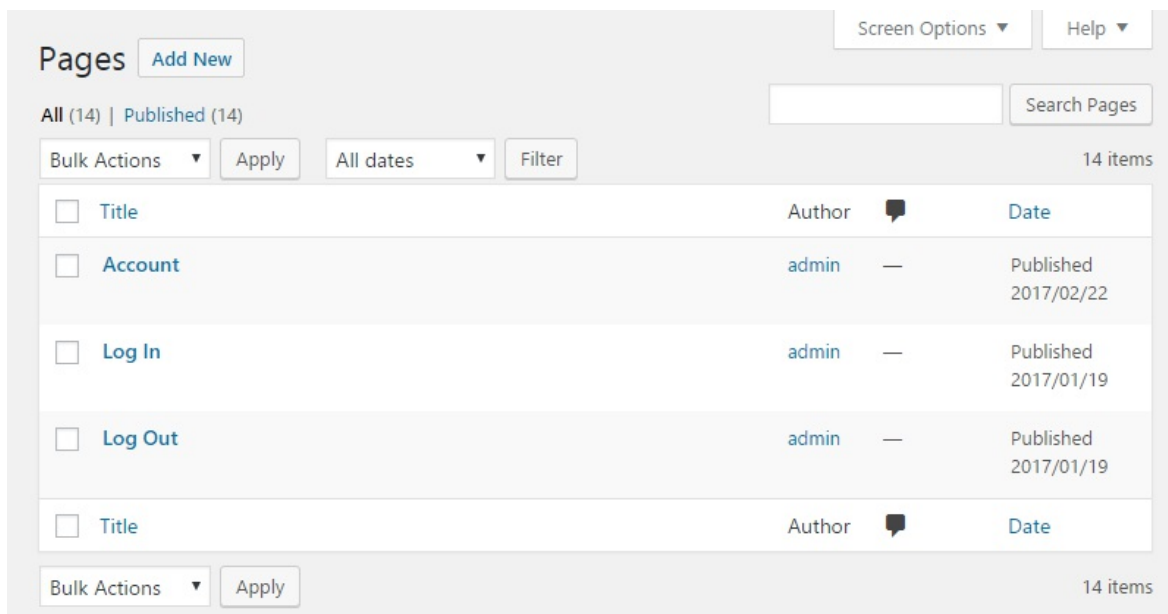
Both techniques can be effectively used for web applications. However,

most developers will pick custom menu pages for large scale applications. Let's identify the differences between the two techniques:

- Custom menu pages create a separate menu item on the left menu, while the options page adds a submenu to the **Settings** menu.
- Options created with the options and settings API will be stored as individual options in the `wp_options` table. When using...

• Using feature-packed admin list tables

In web applications, you will find a heavy usage of CRUD operations. Therefore, we need tables to display the list of records. These days, developers have the choice of implementing common lists using client-side JavaScript as well as PHP. These lists contain functionalities such as pagination, selections, sorting, and so on. Building these types of lists from scratch is not recommended unless you are planning to build a common library. WordPress offers a feature-packed list for its core features using the `WP_List_Table` class located in the `wp-admin/includes/wp-list-table.php` file. We have the ability to extend this class to create application-specific custom lists. First, we'll look at the default list used for core features, as shown in the following screenshot:



The screenshot displays the WordPress 'Pages' admin interface. At the top, there's a 'Pages' header with an 'Add New' button. Below it, a filter bar shows 'All (14)' and 'Published (14)'. A search bar labeled 'Search Pages' is on the right. The main table has columns for 'Title', 'Author', and 'Date'. It lists several pages, including 'Account', 'Log In', and 'Log Out', each with a checkbox for selection. At the bottom, there are bulk action controls and a '14 items' indicator.

<input type="checkbox"/>	Title	Author	Date
<input type="checkbox"/>	Account	admin	Published 2017/02/22
<input type="checkbox"/>	Log In	admin	Published 2017/01/19
<input type="checkbox"/>	Log Out	admin	Published 2017/01/19
<input type="checkbox"/>	Title	Author	Date

As you can see, most of the common tasks, such as filtering, sorting, custom actions, searching, and pagination, are built into this list, which is easily customized by creating child classes. In the next section, we will discuss the default admin lists and how we can customize them in our applications.

Working with default admin...

- **An awesome visual presentation for admin screens**

In general, users who visit websites or applications don't understand the technical aspects. Such users evaluate systems based on the user friendliness, simplicity, and richness of the interface. Hence, we need to think about the design of the admin pages. Most WordPress clients don't prefer the default interface as it is seen commonly by users. This is where admin themes become handy in providing application-specific designs. Even with admin themes, we cannot change the structure as it affects the core functionality. However, we can provide eye-catching interfaces by changing the default styles of the admin theme.

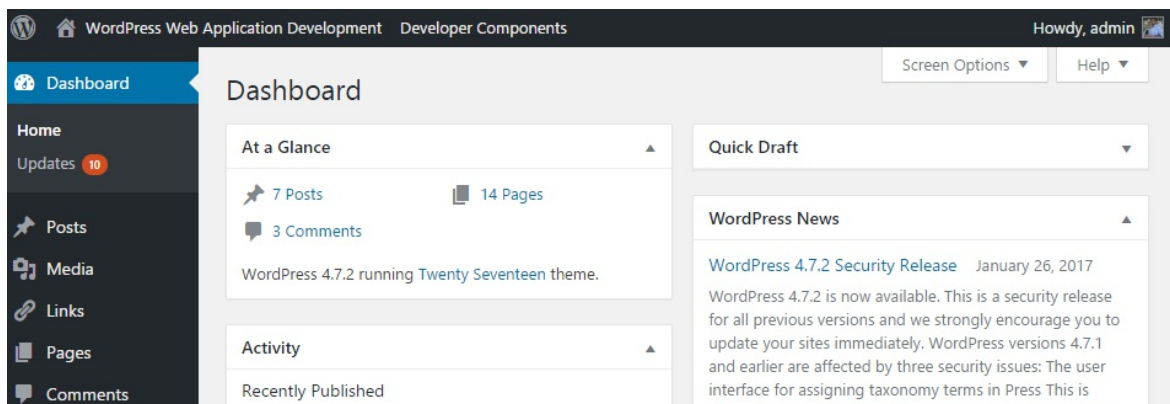
Using existing themes

WordPress Version 3.8 and higher provides the ability to change the color theme of the admin section using eight different color schemes. This is a great feature for changing the look and feel of admin screens in WordPress. Users are allowed to pick their own color scheme for the site, making it flexible for different users with different color preferences. You can change the color theme from the **Your Profile** section of the **Users** menu in the WordPress...

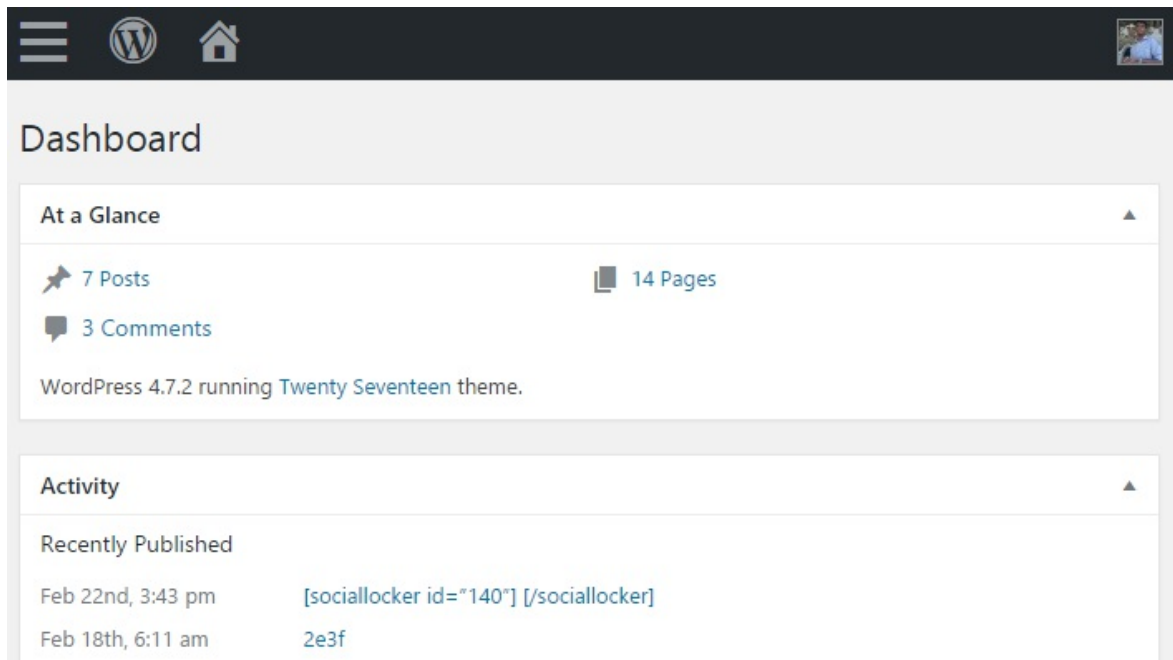
• The responsive nature of the admin dashboard

The responsive design has become one of the major trends in web application development with an increase in the usage of mobile-based devices. Responsive applications are built using style sheets that adapt to various screen resolutions with the help of media queries. Fortunately, the WordPress admin dashboard is responsive by default, and hence we can make responsive backends without major implementations.

Let's consider the following screenshot of an admin dashboard in its default resolution to understand its responsive nature:



Now, let's preview the mobile version of the same screen using the following screenshot:



With the low screen resolution, the theme has made adjustments to keep the responsiveness by minimizing the main navigation menu and increasing the size of the dashboard widgets. This is an example of the responsive nature of the WordPress admin section. Try other screens to get an idea of how elements are adjusted to keep the responsive nature.

Since the admin section is responsive by default, we don't have to do anything else to make it responsive. However, keep in mind that...

• **Supplementary admin dashboard features**

We have discussed some of the most important admin features in this chapter. There are many other supplementary features that add benefits to applications. In this section, we are going to briefly introduce some of the useful admin features available in WordPress.

Dashboard widgets

By default, the WordPress dashboard will be displayed on initial login and it contains default widgets such as At a glance, Quick Draft, WordPress News, Welcome, and so on. The WordPress admin dashboard is a great place to display important information in a summarized way. We have the ability to remove existing widgets as well as add new widgets from plugins and themes. So, it's the ideal place to keep the most commonly used functionality as well as the most commonly needed data displays.

Screen options menu

The **screen options** menu is displayed near the top right-hand section of your screen along with the **help** menu. This menu is displayed only on screens where additional options are available. This menu shows the features available in the current screen and the data available for displaying. You can use the checkboxes...

• Time for action

In this chapter, we covered the basics of an admin panel-related functionality to be compatible with web applications. Now, it's time for you to take these things beyond the basics by implementing the following actions:

- We created a default type of the admin list table to allow subscriptions. Now, try to include an AJAX-based star rating system to allow users to rate topics by implementing a custom column in the existing list.
- We started the implementation of the custom admin theme by setting up styles for the left navigation menu. Try to complete the theme by styling the remaining components.
- Add custom actions, filters, and custom columns for the user list similar to the features we used for post lists.

• Summary

Throughout this chapter, we looked at some of the exciting features of the WordPress admin dashboard, and how we can customize them to suit complex applications. We started by customizing the admin toolbar and the main navigation menu for different types of users. Most of the access permissions to the menu were provided through user capabilities, and hence we didn't need the manual permission checking in building the menu.

Typical web applications contain a large amount of applications settings to let users customize the application based on their preferences. So, we looked at creating our own options pages as well as the default options management features provided by the WordPress core. Also, we looked at the default lists in the admin panel and extended the existing WordPress admin list tables to cater to custom functionality beyond core implementation. We also looked at how we can apply restrictions on the items in admin list tables.

Next, we looked at the importance of responsive web design and how the WordPress admin dashboard adapts to responsive layouts while showing a glimpse into the WordPress admin theme design....

• Chapter 8. Adjusting Theme for Amazing Frontends

Generally, users who visit web applications don't have any clue about the functionality, accuracy, or quality of the code of the application. Instead, they decide the value of the application based on its user interfaces and the simplicity of using its features. Most expert-level web developers tend to give more focus on development tasks in complex applications. However, the application's design plays a vital role in building the initial user base. WordPress uses themes that allow you to create the frontend of web applications with highly extendable features that go beyond conventional layout designs. Developers and designers should have the capability to turn default WordPress themes into amazing frontends for web applications.

In this chapter, we will focus on the extendable capabilities of themes while exploring the roles of the main theme files for web applications. Widgetized layouts are essential for building flexible applications, and hence we will also look at the possibilities of integrating widgetized layouts with WordPress action hooks. It's important to have a very...

• An introduction to the WordPress application frontend

WordPress powers its frontend with a concept called themes, consisting of a set of predefined template files to match the structure of default website layouts. In contrast to web applications, a WordPress theme works in a unique way. In [Chapter 1](#), *WordPress as a Web Application Framework*, we had a brief introduction to the role of a WordPress theme and its most common layout. Preparing a theme for web applications can be one of the more complicated tasks, not discussed widely in the WordPress development community. Usually, web applications are associated with unique templates, which are entirely different from the default page-based nature of websites.

A basic file structure of the WordPress theme

As a WordPress developer, you should have a fairly good idea about the default file structure of WordPress themes. Let's briefly introduce the default files before identifying their use in web applications. Think about a typical web application layout where we have a common header, footer, and content area. In WordPress, the content area is mainly populated by pages or posts. The...

• Web application layout creation techniques

As we move into developing web applications, the logic and screens will become complex, resulting in the need for custom templates beyond the conventional ones. There are a wide range of techniques for putting such functionality into the WordPress code. Each of these techniques have their own pros and cons. Choosing the appropriate technique is vital in avoiding potential bottlenecks in large-scale applications. Here is a list of techniques for creating dynamic content within WordPress applications:

- Static pages with shortcodes
- Page templates
- Custom templates with custom routing

Shortcodes and page templates

We discussed static pages with shortcodes and page templates in [Chapter 2, *Implementing Membership Roles, Permissions, and Features*](#). The shortcode technique should be used carefully in web applications due to the lack of control it displays within the source code.

In the preceding two techniques, the site admin has the capability of changing the structure and core functionality of an application through the dashboard by changing the database content. Usually, the site admin is someone who...

• Building the forum application home page

So far, we have learned the theoretical aspects of creating templates inside WordPress themes. Now, it's time to put them into practice by creating the home page for the forum application. In this section, we will talk about the importance of widget-based layouts for web applications while building the home page. Our home page consists of two major parts:

- Forums list of the application
- Various widgets for forum application data

Let's implement these two parts starting with the forums list.

Building the forum list using shortcode

The `Forumslist` on the home page is the main entry point to our application where we list all the available forums with the link to the individual forum page. We can implement the forum list using a custom template, shortcode, or a widget. In this scenario, we are going to use a shortcode to develop the forums list since it can be reused in many other parts of the application. Let's start by updating the constructor of the `WPWAF_Model_Forum` class to include the new shortcode:

```
add_shortcode('wpwaf_forums_list', array( $this, 'disp
```

Next, we are...

• Generating the application frontend menu

Typically, a web application's frontend navigation menu varies from the backend menu. WordPress has a unique backend with the admin dashboard. Logged-in users will see the backend menu on the top of the frontend screens as well. In the previous chapter, we looked at various ways of customizing the backend navigation menu. Here, we will look at how the frontend menu works within WordPress.

Navigate to the `themes` folder and open the `header.php` file of the Responsive theme. You will find the implementation for the frontend menu using the `wp_nav_menu` function. This function is used to display the navigation menus generated from the **Appearance** section of the WordPress admin dashboard. As far as the forum application is concerned, we may need four different frontend menus for free members, premium members, moderators, and normal guest users. By default, WordPress uses the assigned menu or the default page list to display the menu. Here, we need to create four different navigation menus based on the user role.

Creating a navigation menu

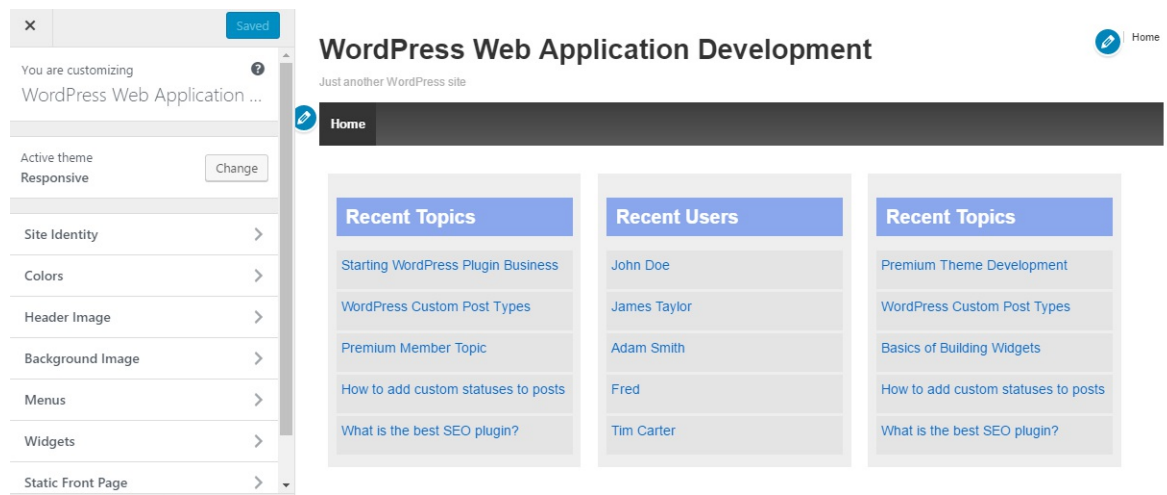
We have to log in to the forum application as the administrator...

• Managing options and widgets with customizer

The WordPress theme customizer is a great feature for customizing a theme's settings and components from the frontend. This feature lets you preview those changes in real time and hence is useful for administrators. Generally, customizer is capable of editing the following sections in real time:

- **Site title and Tagline**
- **Colors**
- **Header Image**
- **Background Image**
- **Navigation**
- **Widgets**

The sections of the customizer are theme-dependent, and hence, you will see more or fewer sections in various themes. You can access the theme customizer by navigating to the **Appearance | Customize** menu. The following screenshot previews the default customizer screen of the **Responsive** theme for our forum application:



All these sections play a part in designing a website with WordPress. However, theme options and widgets are the most important components from the web development perspective, and hence, we will discuss the usage of options and widgets.

Adding custom options to the theme customizer

Apart from widgets, all other sections mentioned in the previous section are related to theme options. These are built-in options of...

• Creating pluggable templates

Templates can be categorized into several types based on their functionality. Each of these types of template plays a different role within complex applications. The proper combination of these template types can result in highly maintainable and reusable applications. Let's explore the functionality of the various template types.

The simplest type of template contains the complete design for each and every screen in the application. These types of template are not reusable. We can also have template parts that get included into some other main templates. These types of template are highly reusable across multiple templates. Header and footer are the most common examples of such templates.

These types of template are highly adaptable and flexible for future enhancements or the modification of existing features. WordPress provides the capability to create similar types of templates with its pluggable architecture by using action hooks. There is a drastic difference between the way reusable templates work in WordPress and normal web applications. However, the final output is quite similar in nature. Let's...

• Extending the home page template with action hooks

Let's identify the practical usage of action hooks in pluggable templates for extending web application layouts. In earlier sections, we developed a home page with three widgets with a reusable template inside a dynamic widget area. Now, we have to figure out the extendable locations of those widgets. Consider the following scenario--Assume that we have been asked to add a button in front of each topic in the home page **Recent Topics** widget. Users who are logged in to the application can click on the button to instantly mark topics as favorites. The implementation of this requirement needs to be done without affecting or changing the other two widgets. Also, we have to plan for similar future requirements for other widgets.

The most simple and preferred way of many beginner-level developers is to create three separate templates for the widgets and directly assign the button to the widget by modifying the existing code. As a developer, you should be familiar with the open-closed principle. Let's see the Wikipedia definition of the open-closed principle:

"software entities (classes,...

• Planning action hooks for layouts

Usually, WordPress theme developers build template files using unique designs and place the action hooks later. These hooks are mainly placed before and after the main content of the templates. This technique works well for designing themes for websites. However, a web application requires flexible templates, and hence, we should be focusing on optimizing the flexibility as much as possible. So the planning of hook points needs to be done prior to designing. Consider the following sample template code of a typical structure of a hook-based template:

```
<?php do_action('before_menu'); ?>
<div class='menu'>
<div class='menu_header'>Header</div>
<ul><li>Item 1</li>
<li>Item 2</li></ul>
</div>
<?php do_action('after_menu'); ?>
```

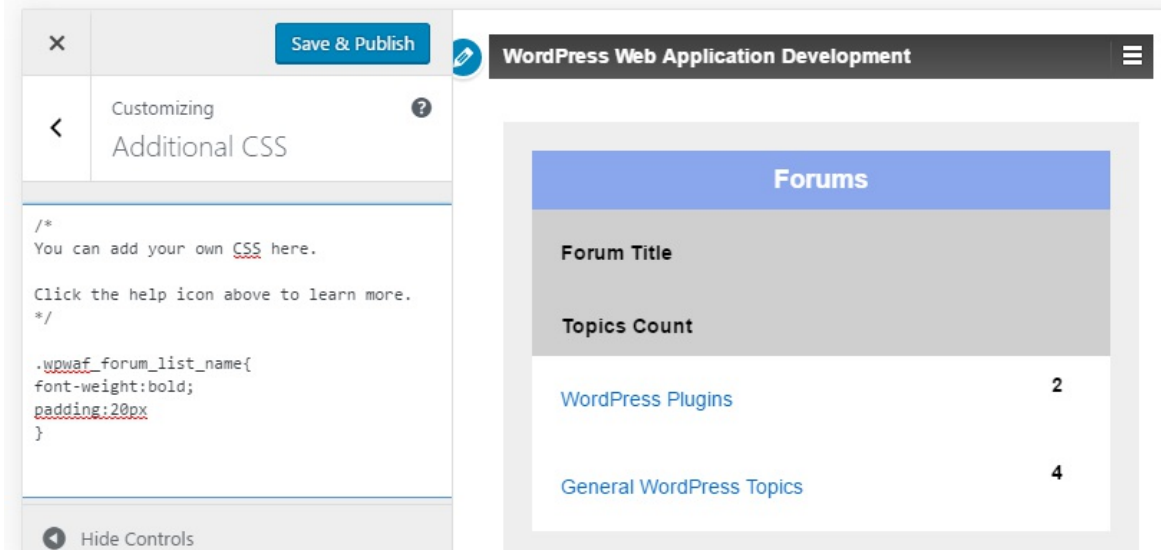
The preceding code is well structured for extending using action hooks. However, we can only add new content before and after the menu container. There is no way to change the content inside the menu container. Let's see how we can increase the flexibility using the following code:

```
<?php do_action('before_menu'); ?>
<?php do_action('menu'); ...
```


• Managing custom CSS with live preview

Generally, we use customized versions of themes and plugins to build applications without developing all features from scratch. Even in this book, we had to use a theme and some plugins while developing the major requirements from scratch. So, when using combinations of themes and other plugins, we tend to get styling issues as these plugins and themes are not fully compatible with one another. Therefore, it's almost a must that we customize features and designs of these plugins to create a uniform design.

Custom CSS plays a major role in customizing the styles of these themes and plugins. Until recently, we had to use a style sheet of child theme or external plugins to add custom CSS. WordPress 4.7 introduced a feature for custom CSS, where we can add CSS through theme customizer and preview the changes instantly. Let's look at how we can add and preview CSS using the following screenshot:

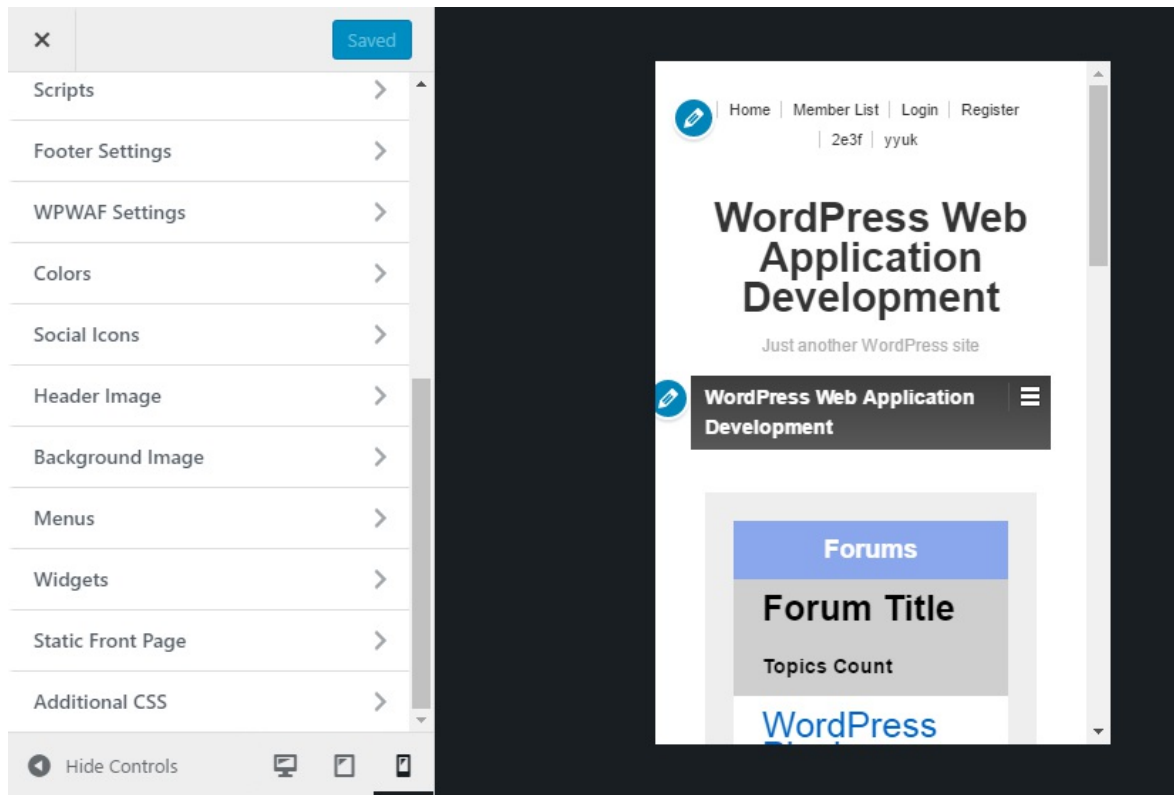


You can find the **Additional CSS** section in the bottom of the left menu in the theme customizer. We can add dynamic CSS in the left area and it will be instantly updated on the right side as shown in the...

• Responsive previews in theme customizer

In the past, we used desktop computers or laptops to access large web applications. But these days users tend to use various kinds of mobile-based devices to access large web applications. Even e-commerce based applications are available on mobile and, unlike in the past, users are not afraid of doing transactions through mobiles. So responsive design has become a must in website and application design. If we don't consider responsive design, users will have major limitations in accessing the application through mobiles and hence we could lose many potential users of the application.

Until recently, we had to use other plugins or third-party services to check the responsiveness of your application screens. WordPress 4.5 introduced responsive previews in the theme customizer where we can check different screen sizes instantly without needing any other service. We can go to **Theme Customizer** and you will see the various screen icons at the bottom of the left menu, as shown in the following screenshot:



As you can see, we have selected the smallest icon for mobiles and the screen on the right is...

• Time for action

In this chapter, we discussed some of the advanced techniques in WordPress themes. Developers who don't have any exposure to advanced application development with WordPress might find it a bit difficult to understand these techniques. Therefore, I suggest you to try the following tasks to get familiar with the advanced theme creation techniques:

- Automate the process of frontend menu item creation. The admin should be able to add menu items to multiple navigation menus in a single event or the complete menu should be generated based on permission levels.
- Complete the process of marking topics as favorites using AJAX and the necessary WordPress actions.
- We implemented user role-based restrictions for menu items. Try to use the same technique to add user role-based restrictions to widgets.

• Summary

The frontend of an application presents the backend data to the user in an interactive way. The possibility of requesting frontend changes in an application is relatively high as compared to the backend. Therefore, it's important to make the application's design as stylish and as flexible as possible. Advanced web applications will require complex layouts that can be extended by new features. Planning for the future is important, and hence, we prioritized the content of this chapter to talk about extending the capabilities of WordPress theme files using a widgetized architecture and custom action hooks.

We also had a look at the integration of custom hooks with widgetized areas while building the basic home page for the forum application. A navigation menu is vital for providing access to templates based on user roles and permissions. Here, we looked at how we can manage frontend menu items based on user roles and how to display them on the frontend. Finally, we looked to how to use the WordPress theme customizer with instant previews.

In the next chapter, we will look at the use of an open source plugin within WordPress....

• Chapter 9. Enhancing the Power of Open Source Libraries and Plugins

WordPress is one of the most popular open source frameworks, serving millions of people around the world. The WordPress core itself uses dozens of open source libraries to power existing features. Web application development is a time-consuming affair compared to generic websites. Hence, developers get very limited opportunities for building everything from the ground up, creating the need for using stable open source libraries.

With the latest versions, WordPress has given higher priority for using stable and trending open source libraries within its core. Backbone.js, Underscore.js, and jQuery masonry are recent popular additions among such open source library. Inclusion of these types of libraries gives a hint about the improvement of WordPress as a web development framework.

We will discuss the various use of these existing open source libraries within the core and how to adapt them into our applications. This chapter also includes some popular techniques such as social logins, to illustrate the integration of external libraries that don't come with the core...

• **Why choose open source libraries?**

Open source frameworks and libraries are taking control of web development. On the one hand, they are completely free and allow developers to customize and create their own versions, while on the other, large communities are building around open source frameworks. Hence, these frameworks are improving in leaps and bounds at an increasing speed, providing developers with more stable and bug-free versions. WordPress uses dozens of open source libraries and there are thousands of open source plugins in its plugins directory. Therefore, it's important to know how to use these open source libraries in order to make our lives easier as developers. Let's consider some of the advantages of using stable open source products:

- A large community support
- The ability to customize existing features by changing source code
- No fees are involved based on per-site or per-person licensing
- Usually reliable and stable

These reasons prove why WordPress uses these libraries to provide features, and why developers should be making use of them to build complex web applications.

• Open source libraries inside the WordPress core

As mentioned earlier, there are several libraries available within the core that have yet to be noticed by many WordPress developers. Most beginner-level developers tend to include such libraries in their plugins when it's already available inside the core framework. This happens purely due to the nature of WordPress development, where most development is done for generic websites with very limited dynamic content.

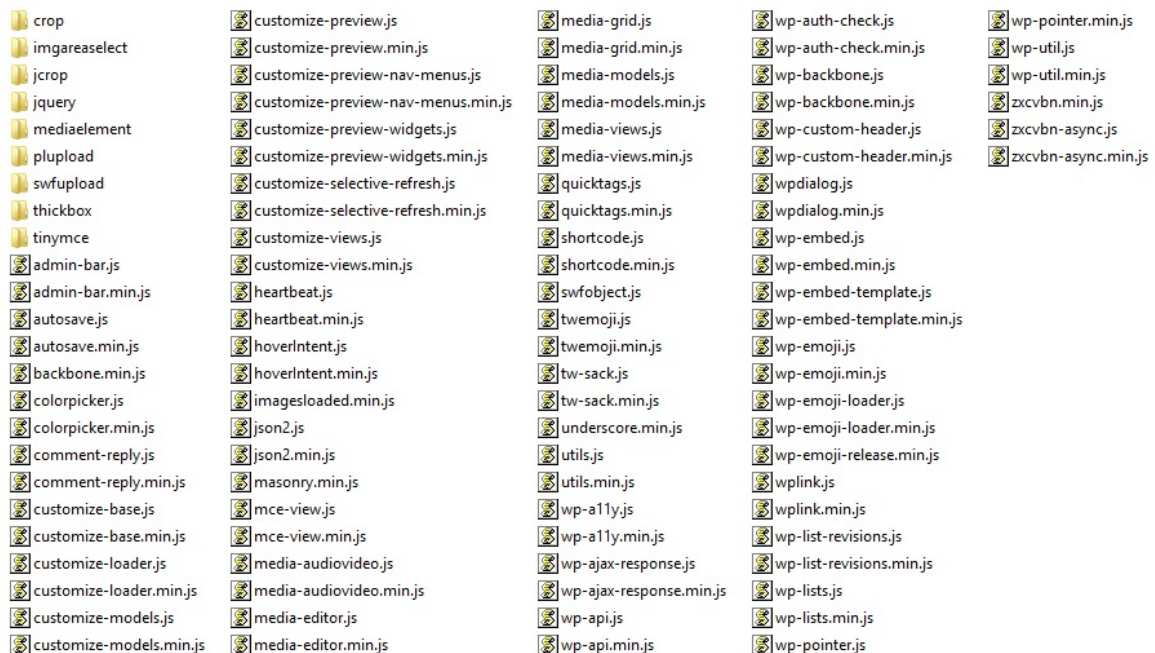
As we move into web application development, we should understand the need for using existing libraries whenever possible due to the following reasons:

- WordPress contains libraries that are more stable and compatible
- Using a WordPress built-in library prevents conflicts in using different version of the same library
- This also prevents the duplication of libraries and reduces the size of project files

I am sure you are familiar with libraries and plugins such as `jQuery` and `TinyMCE` within WordPress. Let's discover lesser-known and recently added libraries in order to make full use of them with web applications.

• Open source JavaScript libraries in the WordPress core

Most open source JavaScript libraries are located inside the `wp-includes/js` folder of your WordPress installation. Let's take a look at the following screenshot for JavaScript libraries available with WordPress 4.7:



As you can see, there are a large number of built-in libraries inside the `wp-includes/js` folder. WordPress uses jQuery for most of its core features, and hence, there is a separate folder for jQuery-related libraries, for example, jQuery UI, Masonry, jQuery Form plugin, and many more. Developers can use all of these libraries inside their own plugins and themes without duplicating the files. `Backbone.js` and `Underscore.js` are the latest additions to the WordPress core. These two libraries are becoming highly popular among web developers for building modularized client-side code for large-scale applications. WordPress integration with `Backbone.js` and `Underscore.js` has been rarely explained in online

resources. So we will look at the integration of these two libraries into the forum application and to identify the use of JavaScript libraries included in the WordPress...

• Using PHPMailer for custom e-mail sending

In the previous section, we looked at the use of `Backbone.js` and `Underscore.js` to identify the usage of open source JavaScript libraries within the WordPress core. Now, it's time to identify the use of open source PHP libraries within the WordPress core, so we will choose `PHPMailer` among a number of other open source libraries. `PHPMailer` is one of the most popular e-mail sending libraries used inside many frameworks as a plugin or library. This library eases the complex tasks of creating advanced e-mails with attachments and third-party account authentications.

Note

`PHPMailer` has been added to GitHub to improve its development process. You can get more information about this library at <https://github.com/PHPMailer/PHPMailer>.

WordPress has a copy of this library integrated into the core for all e-mail-related tasks. We can find the `PHPMailer` library inside the `wp-includes` folder within the file called `class.phpmailer.php`.

Usage of PHPMailer within the WordPress core

Basically, the functionality of this library is invoked using a common function called `wp_mail` located in the `pluggable.php` file...

• Implementing user authentication with OpenAuth

Log in with open authentication has become a highly popular method among application users as it provides quicker authentication compared to the conventional registration forms. So many users prefer the use of social logins to authenticate themselves and try the application before deciding to register. Let's take a look at the definition of **OAuth** by Wikipedia:

"OAuth is an open standard for authorization. OAuth provides client applications a 'secure delegated access' to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner, or end-user. The client then uses the access token to access the protected resources hosted by the resource server."

OpenAuth is becoming the standard third-party authentication system for providing such functionality. Most...

• Using third-party libraries and plugins

We discussed the importance of open source libraries in detail. Most WordPress developers prefer the creation of web applications by installing a bunch of third-party plugins. Ideally, developers should be focusing on limiting the number of plugins within an application to improve the structure of code and possible conflicts.

In web applications, we will need to use third-party resources, no matter how much we want to develop our own code. So it's important to identify stable plugins to be used for applications and ensure that these plugins are safe to use.

• Using open source plugins for web development

There are hundreds of quality free and premium plugins that can be used extensively in web development. Before using any plugin, you should check the popularity of the plugin, the quality of the code, and the support provided by the plugin developers. In this section, we are going to look at some useful and stable plugins for adding functionality to your web applications:

- **Advanced Custom Fields:** User data input, process, and output are the main functionality of most web applications and hence we need custom fields to capture the details. This plugin is the ideal solution to capturing custom data to your posts, pages, and custom post types. This plugin offers over 15 field types and a visually simple data adding process. You can download this plugin and check the available features at <https://wordpress.org/plugins/advanced-custom-fields/>.
- **Ninja Forms:** ACF plugins allowed us to capture custom data for posts, pages, and custom post types. These types of forms plugin are even more useful in web applications since we can capture data using dynamic forms without needing to use WordPress core...

• Using plugins for checking security of other plugins

WordPress is the most used framework for building websites in the world and hence it's the framework with the most security threats. Third-party libraries can contain malicious code that enables security holes in your applications allowing attackers to easily create exploits. Even though there are some tools for checking malicious code, none of them are 100% accurate, and we can't guarantee the results. The following are some plugins for checking malicious code of your plugins and themes:

- **Exploit Scanner:** You can find this plugin at <https://wordpress.org/plugins/exploit-scanner/>
- **Plugin Security Scanner:** You can find this plugin at <https://wordpress.org/plugins/plugin-security-scanner/>
- **Theme Check:** You can find this plugin at <https://wordpress.org/plugins/theme-check/>
- **Theme Authenticity Checker (TAC):** You can find this plugin at <https://wordpress.org/plugins/tac/>

As developers, we should be always looking for stable and consistent libraries for our projects. So it's preferable to work with existing WordPress libraries and stable third-party plugins as much as possible when developing...

• Time for action

As usual, we discussed plenty of practical usages for open source libraries within WordPress. We completed the implementation of a few scenarios and left out some tasks for future development. As developers, you should take this opportunity to get experience in integrating various third-party libraries:

- Track the latest published topics in a separate database table and schedule a cron job for sending notification periodically
- Create a member-specific RSS feed for topics and let users subscribe to get updates
- Implement edit and delete functionality for a topic list using `Backbone.js`
- Integrate OAuth login for Facebook and Twitter

• Summary

The open source nature of WordPress has improved developer engagement to customize and improve existing features by developing plugins, and contributing to the core framework. Inside the core framework, we can find dozens of popular open source libraries and plugins. We planned this chapter to understand the usage of trending open source libraries within the core.

First, we looked at the open source libraries inside the core.

`Backbone.js` and `Underscore.js` are trending as popular libraries for web development and hence have been included in the latest WordPress version. Throughout this chapter, we looked at the use of `Backbone.js` inside WordPress while building the forum member profile page of the forum application. We looked into `Backbone.js` concepts such as models, collections, validation, views, and events.

Later on, we looked at the usage of existing PHP libraries within WordPress by using `PHPMailer` to build a custom e-mail sending interface. In web applications, developers don't always get the opportunity to build everything from scratch. So it's important to make use of existing libraries as much as possible.

As...

• Chapter 10. Listening to Third-Party Applications

The complexity and size of web applications prompts developers to think about rapid development processes through third-party applications. Basically, we use third-party frameworks and libraries to automate the common tasks of web applications. Alternatively, we can use third-party services to provide functionalities that are not directly related to the core logic of the application. Using APIs is a popular way of working with third-party services. The creation of an API opens the gates for third-party applications to access the data of our applications.

WordPress provides the ability to create an API through its built-in APIs powered by **XML-RPC** and **REST**. Also, WordPress is moving completely toward the JSON REST API, and it's available in core with WordPress 4.7. The existing XML-RPC and REST APIs cater to the blogging and CMS functionalities, while allowing developers to extend the APIs with custom functionalities. This chapter covers the basics of an existing API, while building the foundation of a forum management system API. Here, you will learn the necessary techniques for...

• Introduction to APIs

API is the acronym for **Application Programming Interface**. According to the definition on Wikipedia, an API specifies a set of functions that accomplishes specific tasks or allows working with specific software components. As web applications grow larger, we might need to provide the application services or data to third-party applications. We cannot let third-party applications access our source code or database directly due to security reasons. So, APIs allow the access of data and services of the application through a restricted interface, where users can only access the data provided through the API. Typically, users are requested to authenticate themselves by providing usernames and necessary passwords or API keys. So, let's look at the advantages of having an application-specific API.

The advantages of having an API

Often, we see the involvement of APIs with popular web and mobile applications. As the owner of the application, you have many direct and indirect advantages by providing an API for third-party applications. Let's go through some of the distinct advantages of having an API:

- Access to the API can...

• The WordPress XML-RPC API for web applications

With the latest versions, the WordPress API has matured into a secure and flexible solution that easily extends to cater to complex features. This was considered to be an insecure feature that exposed the security vulnerabilities of WordPress; hence, was disabled by default in earlier versions. As of Version 3.5, XML-RPC is enabled by default and the enable/disable option from the admin dashboard has been completely removed.

The existing APIs mainly focus on addressing functionalities for blogging and CMS-related tasks. In web applications, we can make use of these API functions to build an API for third-party applications and users. The following list contains the existing components of the WordPress XML-RPC API:

- Posts
- Taxonomies
- Media
- Comments
- Options
- Users

The complete list of components and respective API functions can be found in the WordPress codex at http://codex.wordpress.org/XML-RPC_WordPress_API.

Let's see how we can use the existing API functions of WordPress.

• Building the API client

WordPress provides support for its API through the `xmlrpc.php` file located inside the root of the installation directory. Basically, we need two components to build and use an API:

- **The API server:** This is the application where the API function resides
- **The API client:** This is a third-party application or service that requests the functionality of an API

Since we will be using the existing API functions, we don't need to worry about the server, as it's built inside the core. So, we will build a third-party client to access the service. Later, we will improve the API server to implement custom functionalities that go beyond the existing API functions. The API client is responsible for providing the following features:

- Authenticating the user with the API
- Making XML-RPC requests to the server through the curl command
- Defining and populating the API functions with the necessary parameters

With the preceding features in mind, let's look at the implementation of an API client:

```
class WPWA_XMLRPC_Client {
    private $xml_rpc_url;
    private $username;
    private $password;
    public function...
```

• Creating a custom API

Custom APIs are essential for adding web application-specific behaviors, which go beyond the generic blogging functionality. We need the implementation for both the server and client to create a custom API. Here, we will build an API function that outputs the list of topics of a specified forum in the forum application. Here, we will use a separate plugin for API creation as an API is usually a separate component from the application. Let's get started by creating another plugin folder called `wpwa-xml-rpc-api` with the main file called `class-wpwa-xml-rpc-api.php`. Since this is a plugin with a basic set of functions, we don't use the plugin coding structure used in the forum plugin.

Let's look at the initial code to build the API server:

```
class WPWAF_XML_RPC_API {
    public function __construct() {
        add_filter('xmlrpc_methods', array($this, 'xml_rpc_
    }
    public function xml_rpc_api($methods) {
        $methods['wpwaf.getForumTopics'] = array($this,
'forum_topics_list');
        return $methods;
    }
}
new WPWAF_XML_RPC_API();
```

First, we use the plugin...

• Integrating API user authentication

Building a stable API is not one of the simplest tasks in web development. However, once you have an API, hundreds of third-party applications will be requesting to connect to the API, including potential hackers. So, it's important to protect your API from malicious requests and avoid an unnecessary overload of traffic. Therefore, we can request an API authentication before providing access to the user. Also, providing the API through SSL is almost a must to secure your API.

The existing API functions come built-in with user authentication; hence, we had to use user credentials in the section where we retrieved a list of forums and topics. Here, we need to manually implement the authentication process for custom API methods. Let's create another API method for subscribing to the topics of the forum application. This feature is already implemented in the admin dashboard using admin list tables. Now, we will provide the same functionality for the API users.

Let's get started by modifying the `xml_rpc_api` function as follows:

```
public function xml_rpc_api($methods) {  
    ...  
}
```

• Integrating API access tokens

In the preceding section, we introduced API authentication to prevent unnecessary access to the API. Even the authenticated users can overload the API by accessing it unnecessarily. Therefore, we need to implement user tokens for limiting the use of the API. There can be many reasons for limiting requests to an API. We can think of two main reasons for limiting the API access as listed here:

- To avoid the unnecessary overloading of server resources
- To bill the users based on API usage

If you are developing a premium API, it's important to track its usage for billing purposes. Various APIs use unique parameters to measure the API usage. Here are some of the unique ways of measuring API usage:

- The Twitter API uses the number of requests per hour to measure API usage
- Google Translate uses the number of words translated to measure API usage
- Google Cloud SQL uses input and output storage to measure API usage

Here, we won't measure the usage or limit the access to the forum API. Instead, we will be creating user tokens for measuring the usage in future. First, we have to create an admin menu page for generating...

• Providing the API documentation

Typically, most popular APIs provide complete documentation for accessing the API methods. Alternatively, we can use a new API method to provide details about all the other API methods and parameters. This allows third-party users to request an API method and get the details about all the other functions.

Note

WordPress uses the API method called `system.listMethods` for listing all the existing methods inside the API. Here, we will take one step further by providing the parameters of API methods with the complete list.

We can start the process by adding another API method to the `xml_rpc_api` function, as shown in the following code:

```
public function xml_rpc_api($methods) {
    $methods['wpwaf.subscribeToTopics'] = array( $this,
    $methods['wpwaf.getForumTopics']= array( $this, 'foru
    $methods['wpwaf.apiDoc'] = array( $this, 'api_doc' );
    return $methods;
}
```

Once updated, we can use the following code to provide details about the API methods:

```
public function api_doc() {
    $api_doc = array();

    ...
}
```

• WordPress REST API for web applications

REST APIs have become a popular way of providing application features as a service in all types of web application. It's already become the standard for providing third-party access and could well be the future of web application development. Most popular online applications such as Facebook, Twitter, Google, LinkedIn, and Amazon provide features to application developers through a well defined REST API. Let's take a look at the Wikipedia definition to identify what REST is and how it works:

"Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations."

Basically, REST API is a set of functions that can be accessed through HTTP protocol using HTTP methods such as GET, POST, PUT, DELETE. It has been a long time since the discussions started about the need for REST API in WordPress. So, the REST API was introduced in a external...

• Time for action

Throughout this chapter, we looked at the various aspects of the WordPress XML-RPC API and REST API while developing practical scenarios. In order to build stable and complex custom APIs, you should have practical experience of the preceding techniques. I recommend that you try the following tasks to extend the knowledge gathered in this chapter:

- You can measure the API usage through the number of requests.
- We created an API function to list all the forums and topics of the application. Try to introduce the filtering of results with additional parameters.
- The XML-RPC API created in this chapter returns an array as the result. Introduce different result formats such as JSON, XML, and array, so that developers can choose their preferred format.
- Implement OAuth based token authentication for REST API.

• Summary

We started this chapter with the intention of building XML-RPC-based and REST-based APIs for web applications. Then, we discussed the usage of the existing XML-RPC API functions while building an API client from scratch.

We looked at the modern way of implementing API features through WordPress REST API. We discussed various aspects of REST API such as custom post type support, creating custom endpoints and creating internal and external API clients.

Complex applications will always exceed the limits of an existing API; hence, we looked at the possibility of creating a custom API with both XML-RPC and REST. User authentication is one of the most important aspects of an API and hence we implemented token based authentication in XML-RPC for preventing unnecessary API access and measuring the API usage. Similarly, we looked at various authentication techniques for REST API and completed the implementation with token based authentication for REST API.

Finally, we looked at the possibility of creating the API documentation through another API function. Having completed the API creation techniques, you should now be able to...

• **Chapter 11. Integrating and Finalizing the Forum Management Application**

Building a large web application is a complex task that should be planned and managed with well-defined processes. Typically, we separate large applications into smaller submodules, where each submodule is tested independently from other modules. Finally, we integrated all the modules to complete the application. The integration of modules is one of the most difficult tasks in application development.

The forum management application created throughout this book intended to illustrate the advanced concepts of WordPress web application development. Therefore, we had to use different techniques in different modules to understand the issues and find feasible solutions. In real world, we have to limit the use of different techniques and keep the consistency across all the features of the application. So, we will be fixing some of the inconsistencies of the application while restructuring the necessary components. After the completion of this chapter, developers should be able to build similar or complex applications without any difficulty.

In this chapter, we...

• Integrating and structuring the forum application

Throughout the first 10 chapters, we implemented the functionality of a forum management system using one main plugin and few other independent plugins. In each chapter, we explained some of the advanced concepts while developing the features related to that concept. So, our application was structured based on modules in WordPress. In the real world, we plan all the features of the application while separating them into sections based on functionality. Here, we have separated them into sections based on WordPress core modules. In the upcoming sections, we are going to fix the inconsistent components in our application and complete the remaining functionality of a basic forum application.

• Integrating the template loader into a user manager

In [Chapter 2](#), *Implementing Membership Roles, Permissions, and Features*, we used direct file inclusions to load the necessary templates. A few chapters later, we improved the loading of templates by introducing a common template loader. Now, we can integrate the template loader into the user management functionality to keep the code consistent.

Let's apply the template loader instead of using direct template inclusions. Consider the template loading section of the `display_registration_form` function, as shown in the following code:

```
include dirname(__FILE__) . '/templates/register-templa
exit;
```

Now, we can take a look at the modified version with the support of the template loader object, as shown in the following code:

```
ob_start();
$wpwa_template_loader->get_template_part('register');
echo ob_get_clean();
```

With the implementation of the new process, we can pass the template data through a global variable called `$wpwa_template_data`. This is the process recommended by WordPress instead of including templates directly. We can replace all the occurrences of...

• Working with the restructured application

Having resolved the inconsistencies, we now have to understand the process of creating new functionalities from scratch and integrating with the features we created so far. So, in this section, we will build the forum details and topic details pages of our application. In previous chapters, we created the home page of the application to display the list of available forums with topics count using a shortcode. Once, users click on the forum link from the list, they will be navigated to the **Forum Details** page. This is the core functionality of the application where the user interacts with other users. Let's identify the requirements of the forum details page.

Building the forum page

This is the page that lists the details of a given forum and the available topics inside the forum. First, we have to identify the possible techniques of implementing such a page and choosing the ideal technique for your application. Let's identify some of the different techniques:

- **Using a shortcode:** Create a shortcode for the forum details page and use it on a WordPress page. This is not the best technique, unless...

• Updating a user profile with additional fields

The forum user profile page was created in [Chapter 9](#), *Enhancing the Power of Open Source Libraries and Plugins*, with the use of `Backbone.js` and `Underscore.js`. The **Profile** section of this page was limited to the name of the user as we had very limited information for the users. Here, we will capture more information by using additional fields on the profile screen of the WordPress dashboard. So, let's update the constructor function of the `WPWAF_User` class to add the necessary actions for editing the profile, as shown in the following code:

```
add_action('show_user_profile', array($this,
    "add_profile_fields"));
add_action('edit_user_profile', array($this,
    "add_profile_fields"));
```

We have defined two actions to be executed on the user profile screen. Both the `show_user_profile` and `edit_user_profile` actions are used to add new fields to the end of the user edit form. According to the preceding code, the addition of new fields will be implemented in the `add_profile_fields` function of the `WPWAF_User` class. Let's look at the implementation of the `add_profile_fields` function:

```
...
```

• Scheduling subscriber notifications

Sending notifications is a common task in any web application. In this scenario, we have subscribers who want to receive e-mail updates about forum activities. In [Chapter 9](#), *Enhancing the Power of Open Source Libraries and Plugins*, we created a simple e-mail notification system on post publish for sending e-mail notifications for topic creators and administrators. Using the same technique used for sending forum topic updates can become impossible due to the large number of subscribers. Therefore, we will take a look at the scheduling features of WordPress for automating the notification sending process.

As a developer, you might be familiar with cron, which executes certain tasks in a time-based manner. WordPress scheduling functions offer the same functionality with less flexibility. In WordPress, this action will be triggered only when someone visits the site after the scheduled time has passed. In a normal cron job, the action will be triggered without any interaction from users. Let's see how to schedule subscriber notifications for predefined time intervals using the `wp_schedule_event...`

• Time for action

Throughout this book, we have developed various practical scenarios to learn the art of web application development. Here, we have the final set of actions before we complete the forum application for this book. By now, you should have all the knowledge to get started with WordPress web development. After reading this chapter, you need to try the following set of actions for getting experienced with the process:

- Find out different ways of structuring WordPress for web applications.
- Add forum-user permission checks to backend topic creation.
- Link users, moderator, and administrators in topic and forum details pages to their respective profile pages.
- Display the files attached to the forum by administrators from the backend.
- Implement topic statuses to identify open and resolved topics.

• Final thoughts

WordPress is slowly but surely approaching as a framework for web application development. Developers are getting started on building larger applications by customizing existing modules and features. However, there are a lot of limitations and a lack of resources for web development-related tasks. So, the best practices and design patterns are yet to be defined for building applications with WordPress.

In this book, we developed an application structure, considering the best practices of a general web application development. The WordPress architecture is different from the typical PHP frameworks, and hence this structure might not be the best solution. As developers, we want to drive WordPress into a fully featured web application framework.

In this chapter, we completed the development of the demo forum application for this book. So, feel free to innovate your own application structures and the techniques.

• Summary

We began this chapter by looking at the issues of the application plugins developed throughout the first 10 chapters of the book. Once the inconsistencies were fixed, we implemented a few new requirements, such as subscriber notifications, forum details page, topic details page, topic creation, join to forums, and additional user profile fields in order to understand how to add new features into existing applications and integrate all parts.

Here, we are completing the demo application for this book with the basic foundation of the forum application. You can improve and complete the remaining functionality of forum application with the code provided in book website.

You can now take a look at the next chapter for supplementary modules for WordPress applications. This will be a theoretical chapter explaining the concepts to improve your application with utility features.

• Chapter 12. Supplementary Modules for Web Development

In web application development, we mainly focus and plan our business logic. Throughout the first 11 chapters of this book, we developed features that are directly related to the forum management application. However, there are supplementary features that are not related to the business requirements of the application and yet play a vital part in the success of a project.

Multilanguage support, caching, performance, and security are important features of any web application. WordPress provides built-in features to support these types of non-application-related tasks. Apart from this, WordPress also offers some features that can be used to improve the flexibility and user experience of applications. Throughout this chapter, we will give a brief introduction to these supplementary modules so that you can use them when the opportunity arises.

In this chapter, we will cover the following topics:

- Internationalization
- Working with the media grid and image editor
- Introduction to the post editor
- Lesser-known WordPress features
- Managing application scripts and styles
- Version control
-
- Importing...

• Internationalization

Internationalization is the process of making your application ready for translating to other languages. WordPress itself provides translation for its core by default. In most cases, we will be developing web applications using plugins or themes. So, it's essential to make the theme and plugins translatable for an improved flexibility. We can develop web applications as standalone projects for a specific client or as projects for any client that you wish to use. Internationalization is more important in the latter as a product is used by many clients compared to a project. Even with projects, it's an important aspect of development, when your client wants a non-English application.

In this section, we will look at internationalization support in WordPress and how to translate and manage plugins.

Introduction to WordPress translation support

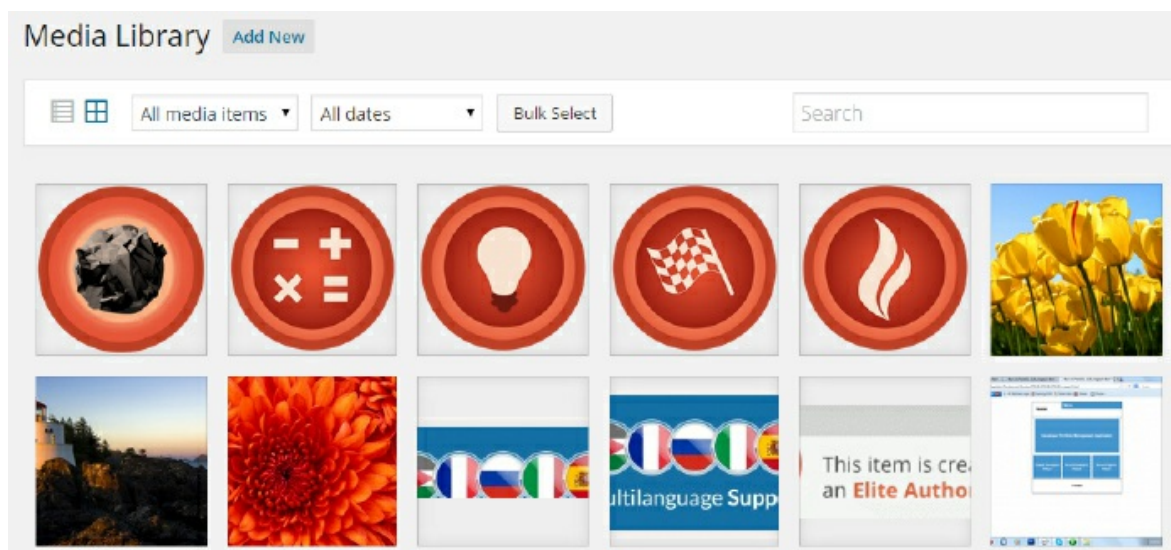
WordPress allows translation support for plugins using the **GetText Portable Object**. We need to have three things to enable translation support in WordPress:

- Defining the text domain
- Enabling translations on strings using WordPress functions
- Loading the text domain

WordPress uses...

• Working with media grid and image editor

In normal web applications, we let users upload images and files using the HTML file upload field. These files are uploaded to a specific server location. However, what if the admin wants to view the files or edit them? We have to manually download the files, make the modifications, and upload them again. WordPress offers a media management section by default. This section is improved in WordPress 4.0+ versions to include the media grid and image editor. This is a great feature for managing media as administrators. You can use this to let your users upload images and edit them before saving them in the WordPress backend. Since it's a built-in feature, you don't have to develop any functionality for image uploading in WordPress applications. Let's see the default display of the new media grid using the following screenshot:



Once you click on a single item, all the details will be displayed in a pop-up menu with the ability to navigate between the other media items. Also it lets us edit images instantly without needing any tools. The following screenshot previews the media item information...

• Introduction to the post editor

The WordPress post editor provides amazing features to edit the content of your site. It's improving with every version, and now we have come to a stage where we are considering frontend direct content editing. The default post editor provides lots of built-in items to format your content using HTML tags. Also, we use the post editor for adding shortcodes to content through buttons.

Using the WordPress editor

In web applications, we usually need to use text areas to get descriptive information from the user. In such scenarios, we just put the default HTML text area field or use a comprehensive editor such as *TinyMCE*. Adding and configuring these types of editors usually takes a lot of time. In WordPress, we have the ability to use a built-in editor anywhere you wish. Normally, we see it as the post/page editor. However, we can use it on both the frontend and backend of the application by just including the `wp_editor` function. This will create a nice content editor similar to the WordPress post editor. The following code shows the syntax of the `wp_editor` function:

```
wp_editor( $content, $editor_id, ...
```

• Lesser-known WordPress features

Throughout this book, we have looked at the major components related to web application development. WordPress also offers some additional features that are rarely noticed among the developer community. Let's get a brief introduction to the following lesser-known features of WordPress:

- Caching
- Transients
- Testing
- Security
- Performance

Caching

In complex web applications, performance becomes a critical task. There are various ways of improving the performance from the application level as well as the database level. Caching is one of the major features of the performance-improving process, where you keep the result of complex logic or larger files in the memory or database for quick retrievals.

WordPress offers a set of functions for managing caching within applications. Caching is provided through a class called `WP_Object_Cache`, which can be used effectively to manage a non-persistent cache.

We can cache the data using the built-in `wp_cache_add` function, as defined in the following code:

```
wp_cache_add( $key, $data, $group, $expire );
```

`$group` parameter defines the group name of the cached data. It's somewhat...

• Managing application scripts and styles

In application development, we use scripts and style files from third-party open source libraries, WordPress core, and the files created from scratch for our own application requirements. So, it very important to understand how to load and work with these files properly. When scripts and styles are not used properly, we have to face some issues as listed:

- Scripts and styles of one plugin could create conflicts with scripts and styles of another plugin.
- Possibility of duplicate file inclusion.
- Application might try to use the scripts before they are included.

We can use the WordPress recommended techniques for loading of scripts and styles to avoid such issues. WordPress recommends developers use `wp_enqueue_scripts` for the frontend, and the `admin_enqueue_scripts` function for the backend loading of scripts and styles. So, we have to use following syntax to add the files correctly into WordPress:

```
functionwpwa_enqueue_scripts(){
    wp_register_script( 'wpwa-questions', plugins_url(
        'js/questions.js', __FILE__ ), array('jquery'), '1.0',
        wp_enqueue_script( 'wpwa-questions'...
```

• Version control

Version control is the process of managing changes to your source code files. Generally, this is not critical in applications with a single developer. However, this is a *must use* aspect in application development with large teams. In such applications, many developers need to work on the same file, and it's impossible to wait till the other developer completes working on the file. In such scenarios, version control becomes a must use feature. Even in a single developer application, it's useful to track the changes. So, let's identify the important features we get with version control:

- **Managing versions:** We release applications or plugins several times with new features. We need the capability to track previous versions and use them instantly. Version controlling allows us to keep different versions using the concept, branches. So, it's very easy to access any version of your application without needing to keep backups manually.
- **Reduce code conflicts:** If multiple people are working on the same code file, we can't prevent creating conflicts and losing codes. Version control allows us to get an updated version of the...

• E-commerce

The transaction of buying and selling online using Internet is known as e-commerce. The main components of an e-commerce application consist of product management, order management, stock management, shopping cart, and payment handling. In general, building an e-commerce based site with other frameworks is a highly costly process and hence everyone can't afford it. However, building an e-commerce application with WordPress is a very easy task, even without any development knowledge. Since this is a zero-cost solution, anyone can build his/her own product stores within few hours and start selling.

WordPress offers advanced free and premium e-commerce plugins, and all we have to do is install and use them or customize them according to our needs. Before choosing an e-commerce plugin, we need to identify what we are selling and what the best solution is. We can categorize different types of products used in online selling as following:

- Physical products such as books, CDs, and shirts
- Digital products such as e-books, plugins, and presentations
- Services such as hiring a developer or designer
- Subscriptions or memberships that...

• Migrating WordPress applications

Generally, we develop applications on a local server or testing server before they are migrated to a live server, after proper testing. This is one of the tedious tasks as a developer or site owner, unless we use the proper tools. When developing web applications with other frameworks, we don't have modules called plugins. So, we have to manually back up the database, files, and uploaded media, and upload to another server via FTP. Instead, we can use WordPress existing tools to completely or partially migrate the application without much effort from our end. There are plugins that allow you to back up the database, files, and media separately, and then import them manually to the live server. We are going to look at a plugin that offers all these features within the same plugin, making migration a super simple task.

You can find a plugin called *All-in-One WP Migration*. This plugin is available at <https://wordpress.org/plugins/all-in-one-wp-migration/>. Once activated, you can click on the **Export** item from the **All-In-One WP Migration** menu item on the left menu. You will get a screen similar to...

• Importing and exporting application content

In the previous section, we looked at how to migrate an entire site with all its content. Importing and exporting content is similar, where we back up the database and import it to another site. As usual, we can find many existing plugins for the import/export process. However, most of these plugins import/export entire database tables and it's not the ideal solution in some scenarios.

Assume we want to export all the users with custom profile data in our forum application. Since most existing plugins export the entire database table, it's not possible to export only the necessary content. Export with a plugin will add all usermeta details, instead of only exporting the custom fields we created. In such scenarios, we need to build our own import and exporting features. So, in our application, we might need different sections to:

- Import/export users with custom fields
- Import/export topics of a given forum

Also, we will be using many plugins in web development instead of developing everything from scratch. So, we should check the import/export features of each plugin we use. Most of the...

• Introduction to multisite

WordPress provides a module called multisite where you can create multiple networks of WordPress sites using a single installation. All the sites in the network share the same files. These sites can be installed as subfolders of the main site or subdomains. We need to know how WordPress multisite is used and how it can support web application development.

Let's identify the common usages of WordPress multisite:

- It lets users create their own blog, website, or product-selling website within your site. This is a widely used technique where you let your users purchase a membership to manage their site within your network.
- It manages multiple products. WordPress theme and plugin developers use this technique to create demo sites for their plugins and themes using a single installation.
- It manages the branches of a large-scale organization. Many large organizations have branches in multiple countries. So, multisite allows them to manage a separate site for each country within the network. Since all branches have similar features, multisite makes it very easy to manage.

Having looked at the practical use cases of...

• Time for action

Throughout this book, we developed various practical scenarios to learn the art of web application development. Here, we have the final set of actions before we complete this book. By now, you should have all the knowledge of getting started with WordPress web development. After reading this chapter, you need to try the following set of actions for getting experienced with the process:

- Translate the Forum Manager plugin using the technique discussed in this chapter.
- Implement conditional scripts and styles loading in the forum plugin.
- Figure out practical scenarios for implementing Cache and Transients.

• Summary

Generally, other PHP frameworks don't provide advanced built-in features for supplementary modules we discussed in this chapter. So, we have to find or develop them from scratch. However, WordPress comes with a powerful plugin directory with close to 50,000 plugins. It keeps increasing every day at a speed beyond the imagination. So, we can find an existing plugin for almost all the features we need. All the supplementary modules discussed in this chapter are covered by a large number of plugins and hence we can quickly work on nonfunctional tasks of our application compared to other PHP frameworks.

WordPress is slowly but surely becoming a trend in web application development. Developers are getting started on building larger applications by customizing existing modules and features. However, there are a lot of limitations and a lack of resources for web development-related tasks. So, the best practices and design patterns are yet to be defined for building applications with WordPress.

We started this book by discussing how WordPress can be adapted to web applications and developed a simple question-answer interface using...

• Chapter 13. Configurations, Tools, and Resources

In this appendix, we will set up and configure WordPress and necessary tools to follow the demo application in this book. You will also find a list of resources and tutorials on libraries and plugins used in this book. Let's start by configuring and setting up WordPress.

• **Configuring and setting up WordPress**

WordPress is a CMS that can be installed in a few minutes with an easy setup guide. Throughout this book, we are implementing a forum management application with advanced users. This short guide is intended to help you set up your WordPress installation with necessary configurations to be compatible with the features of our application. Let's get started!

Step 1– downloading WordPress

We are using WordPress 4.7.2 as the latest version available at the time of writing this book, so we have to download version 4.7.2 from the official website at <http://wordpress.org/download/>.

Step 2 – creating the application folder

First, we need to create a folder for our application inside the web root directory. Then extract the contents of the downloaded zip file into the application folder. Finally, we have to provide the necessary permissions to create files inside the `application` folder. Make sure that you provide write permission for the `wp-config.php` file before starting the installation. Generally, we can use 755 permissions for directories and 644 permissions for files. You can learn more about...

• Open source libraries and plugins

We used a number of open source libraries and plugins throughout the book. The following list illustrates all the libraries and plugins used with the respective URLs to get more information:

- **The Responsive theme:** This theme is developed by CyberChimps. You can find this at <http://goo.gl/Uf9Mp1>
- **The Members plugin:** This plugin is developed by Justin Tadlock. This can be found at <http://goo.gl/HuhDax>
- **The Rewrite Rules Inspector plugin:** This plugin is developed by Daniel Bachhuber and Automattic. You can find this at <http://goo.gl/oBVJmL>
- **The Posts 2 Posts plugin:** This plugin is developed by Alex Ciobica and scribu. This is available at <http://goo.gl/8pQGmT>
- **Pods-Custom Content Types and Fields:** You can find this framework at <http://goo.gl/ixMspf>
- **The Custom List Table Example plugin:** This plugin is developed by Matt Van Andel. You can find this at <http://goo.gl/3tnfmf>
- **Backbone.js:** This library can be found at <http://goo.gl/VyhEDl>
- **Underscore.js:** This library is available at <http://goo.gl/aZ42YD>
- **PHPMailer:** This library can be found at <http://goo.gl/VX90ym>
- **Postman extension:** This extension for chrome is available...

• Online resources and tutorials

Web application development with WordPress has still not matured, so you will find various perspectives from various people about using WordPress as an application development framework. This section provides various tutorials and articles for understanding various perspectives on using WordPress for web application development:

- *Wordpress As An Application Platform*, Tom McFarlin (<http://goo.gl/gONP3i>)
- *WordPress For Application Development*, Tom McFarlin (<http://goo.gl/ubDasf>)
- *My Thoughts on Building Web Applications with WordPress*, Tom McFarlin (<http://goo.gl/fTUqQf>)
- *Why WordPress Isn't Viewed as an Application Framework*, Tom McFarlin (<http://goo.gl/Ophmak>)
- *Using WordPress as a Web Application Framework*, Harish Chouhan (<http://goo.gl/BFHqVB>)
- *8 Awesome WordPress Web Apps Paving the Way for WordPress as a SaaS Platform* (<https://goo.gl/uKdI6Z>)
- *Build an App With WordPress - The compulsory todo list*, Harley Alexander (<http://goo.gl/rwMB6c>)