Learning Django Web Development

Django, written in Python, is a web application framework designed to build complex web applications quickly without any hassle. It loosely follows the MVC pattern and adheres to the Don’t Repeat Yourself principle, which makes a database-driven application efficient and highly scalable, and is by far the most popular and mature Python web framework.

This book is a manual that will help you build a simple yet effective Django web application. It starts by introducing Django, setting it up, and shows you how to code simple programs. You will then learn to build your first Twitter-like app. Later on, you will be introduced to Hashtags, AJAX to enhance the user interface, and tweets. You will then move on to create an administration interface, learn database connectivity, and use third-party libraries. Then you will learn to debug and deploy Django projects, and also get a glimpse of Django with AngularJS and Elasticsearch. By the end of the book, you will be able to leverage the Django framework to develop a fully functional web application with minimal effort.

Who this book is written for
This book is for web developers who want to get started with Django for web development. Basic knowledge of Python programming is required but no knowledge of Django is expected.

What you will learn from this book
- Build Django web application right from scratch
- Use version control to manage the development project
- Learn to use Django with both SQL and NoSQL databases
- Build faster and more efficient webpages using a frontend framework Twitter Bootstrap
- Improve web application performance with caching
- Enhance your user interface with AJAX and add flavors to your website
- Deploy the Django web application to clouds such as AWS, Heroku, and OpenShift
- Get familiar with AngularJS and Elasticsearch for Django

$ 44.99
£ 29.99

Prices do not include local sales tax or VAT

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introduction to Django'
- A synopsis of the book’s content
- More information on Learning Django Web Development
About the Authors

Sanjeev Jaiswal is a computer graduate with 5 years of industrial experience. He basically uses Perl and GNU/Linux for his day-to-day work. He also teaches Drupal and WordPress CMS to bloggers. He first developed an interest in web application penetration testing in 2013; he is currently working on projects involving penetration testing, source code review, and log analysis, where he provides the analysis and defense of various kinds of web-based attacks.

Sanjeev loves teaching technical concepts to engineering students and IT professionals and has been teaching for the last 6 years in his leisure time. He founded Alien Coders (http://www.aliencoders.org), based on the learning through sharing principle for computer science students and IT professionals in 2010, which became a huge hit in India among engineering students.

He usually uploads technical videos on YouTube under the Alien Coders tag. He has got a huge fan base at his site because of his simple but effective way of teaching and his philanthropic nature toward students. You can follow him on Facebook at http://www.facebook.com/aliencoders and on Twitter at @aliencoders.

He wrote Instant PageSpeed Optimization, Packt Publishing, and looks forward to authoring or reviewing more books for Packt Publishing and other publishers.

Ratan Kumar is a computer science and engineering graduate with more than a year of start-up experience. He received the Technical Excellence Memento from the Association of Computer Engineering Students (ACES), Cochin University of Science and Technology.

When he was a product engineer at Profoundis, he worked on an international project based on services using Django. He was also part of the Microsoft accelerator program with Profoundis that was responsible for building the product iTestify, which was built using Django.
He then moved to Tracxn!, an organization that works on building platforms that can help venture capitalists, investment banks, and corporate developers find new and interesting start-ups in their investment sector. As a software developer and engineer, he majorly contributed to the development of the core product platform of Tracxn using Python for the initial scraping work, such as building cron scrappers to crawl millions of pages daily, cleaning them up, and analyzing them.

He also built the company's first product, which is called Tracxn Extension—a Chrome extension using AngularJS. He contributed to the Tracxn product platform using Grails as the framework. He also worked on Bootstrap—a frontend framework—to design the home page of tracxn.com.
Django, written in Python, is a web application framework designed to build complex web applications quickly without any hassle. It loosely follows the MVC pattern and adheres to the Don't Repeat ourself principle, which makes a database-driven application efficient and highly scalable, and is by far the most popular and mature Python web framework.

This book is a manual that will help you build a simple yet an effective Django web application. It starts by introducing Django to you and teaches you how to set it up and code simple programs. You will then learn to build your first Twitter-like application. Later on, you will be introduced to hashtags, Ajax (to enhance the user interface), and tweets. You will then move on to create an administration interface, learn database connectivity, and use third-party libraries. Then, you will learn to debug and deploy Django projects and will also get a glimpse of Django with AngularJS and Elasticsearch. By the end of this book, you will be able to leverage the Django framework to develop a fully functional web application with minimal effort.

**What this book covers**

*Chapter 1, Introduction to Django,* gives you an introduction to MVC web development frameworks and the history of Django and explains why Python and Django are the best tools to use to achieve the aim of this book.

*Chapter 2, Getting Started,* shows you how to set up our development environment on Unix/Linux, Windows, and Mac OS X. We will also see how to create our first project and connect it to a database.

*Chapter 3, Code Style in Django,* covers all the basic topics that you need to follow for building a website, such as coding practices for better Django web development, which IDE you should use, and version control.
Preface

Chapter 4, *Building an Application Like Twitter*, takes you through a tour of the main Django components and develops a working prototype for your Twitter application.

Chapter 5, *Introducing Hashtags*, teaches you how to design the algorithm to build a hashtag model and the mechanism to use a hashtag in your post.

Chapter 6, *Enhancing the User Interface with AJAX*, will help you enhance the UI experience using Ajax with Django.

Chapter 7, *Following and Commenting*, shows you how to create login, logout, and registration page templates. It will also show you how to allow another user to follow you and how to display the most followed user.

Chapter 8, *Creating an Administration Interface*, shows you the features of administrator interface using Django's inbuilt features and how we can show tweets in a customized way with a sidebar or pagination enabled.

Chapter 9, *Extending and Deploying*, prepares your application for deployment into a production environment by utilizing various features of the Django framework. It also shows you how to add support for multiple languages, improve performance by caching, automate testing, and configure the project for a production environment.

Chapter 10, *Extending Django*, speaks about how to improve the various aspects of your application, mainly performance and localization. It also teaches you how to deploy your project on a production server.

Chapter 11, *Database Connectivity*, covers the various forms of database connectivity, such as MySQL, NoSQL, PostgreSQL, and so on, which is required for any database-based application.

Chapter 12, *Using Third-party Packages*, talks about open source and how to use and implement open source third-party packages in your project.

Chapter 13, *The Art of Debugging*, shows you how to log and debug your code for better and efficient coding practice.

Chapter 14, *Deploying Django Projects*, shows you how to move a Django project from development to a production environment and the things that need to be taken care of before you go live.

Chapter 15, *What's Next?*, will take you to the next level where you will be introduced to the two most important and preferred components, AngularJS and Elasticsearch, used in the Django project.
Introduction to Django

Welcome to version 2.0 of Development with Django!

Django is a web development framework, and web development is a skill. To master any skill one can follow the famous "10,000 hours" rule, which says that if you practice anything for that amount of time you will certainly become an expert at it. But that's a lot of time, and without a proper plan, this can go wrong. Terribly wrong.

So, is there any better way to achieve your goal? Yes! Break the skill you want to learn into smaller subskills and then master them one at a time. (Programmers call this the "divide and conquer" rule.) You will need to identify the most important subskills by researching them. The more the frequent mentions of the subskill, the more important it becomes to master.

As you have decided to learn a new skill, as the author of this book, I request that you make a commitment, that you will stick with this book in the early frustrating hours. Frustration happens when you are learning a new thing, and trust me on this: when you feel it's too simple, you are doing it right.

In this chapter, we will cover the following topics:

- Why web development in the first place?
- What has changed in web development
- The MVC pattern in web development
- Why Django
- Inside Django
Why web development in the first place?
A website makes the first impression about the company or product directly on a global audience. Every startup now has a website, which helps to pitch their idea to their potential clients or investors.

Everything is online now, so instead of just sitting and watching the change, why not participate and learn to code it? Learning web development is one of the most valuable investments you can make with your time. It will not only benefit you by getting you a better job, but you will also be able to code your idea into a prototype in a very simple and straightforward manner.

Must-have ingredients for web development include user interface and user experience, but they are unfortunately out of the scope of this book.

What has changed in web development
Web development has made great progress during the last few years. Some of the improvements are listed as follows:

- **JavaScript**: Evolved from writing complex selectors to manipulating Document Object Model (DOM). Libraries such as jQuery and AngularJs have made frontend dynamics much simpler. JavaScript has even evolved to build a production-ready server-side framework called node.js.

- **Browsers**: Evolved from being as simplistic as breaking the page across browsers to now intelligently restoring the connection, telling you which tab is playing music, or flawlessly rendering a real-time game.

- **Open source**: Using code written by someone else has finally become preferable than writing your own code. This helped a lot of projects to stop reinventing the wheel, Django being one of the best examples.

- **API as spinal cord**: Web technologies today might not be the same tomorrow, or data might not be represented in the same way or in the same place tomorrow. In other words, more devices will come with different screen sizes. Therefore, it’s always best to have text separated from visuals.

- **User Interface**: In the past, the precious time of the development team was consumed by User Interface design. But frameworks such as Bootstrap and Foundation have made web development a lot easier.
• **Agile development**: Moving fast in the development cycle is acceptable to most startup companies. The complete requirement is never asked for at the beginning of the software development cycle. Therefore, continuous customer or stakeholder involvement is very important. The Django framework is the most suitable framework for this kind of development. As Django's slogan says, "the web framework for perfectionists with deadlines".

• **Evolution of cloud computing**: This has played a significant role at the hosting end of web applications and enables faster, more reliable, and cheaper solutions for getting online.

• **Birth of NoSQL**: Cutting costs much further, NoSQL gave freedoms such as *Store it Now, Find The Value Later* and *Store Anything Together* to developers being cloud friendly and more fault tolerant.

**The MVC pattern in web development**

In this book, you will learn about employing a Model-View-Controller (MVC) web framework called Django, which is written in Python, a powerful and popular programming language.

MVC works on the idea of separate presentation. The idea behind separated presentation is to make a clear division between domain objects that model our perception of the real world and presentation objects that are the user interface (UI) elements we see on the screen. Domain objects should be completely self-contained and should work without reference to the presentation or data-handling logic (controller). They should also be able to support multiple presentations, possibly simultaneously.

The benefits of this pattern are obvious. With it, designers can work on the interface without worrying about data storage or management. And developers are able to program the logic of data handling without getting into the details of presentation. As a result, the MVC pattern quickly found its way into web languages, and serious web developers started to embrace it over previous techniques.

This book emphasizes on utilizing Django and Python to create a Web 2.0 microblogging web application with many common features found in today's Web 2.0 sites. The book follows a tutorial style to introduce concepts and explain solutions to problems. It is not meant to be a reference manual for Python or Django, for both have plenty of resources already. The book only assumes working knowledge of standard web technologies (HTML and CSS) and the Python programming language. Django, on the other hand, will be explained as we build features throughout the chapters, until we realize our goal of having a working Web 2.0 application.
Multilingual support
Django supports multilingual websites through its built-in internationalization system. This can be very valuable for those working on websites with more than one language. The system makes translating the interface a very simple task.

So, to conclude, Django provides a set of integrated and mature components, with excellent documentation, at http://www.djangoproject.com/documentation/.

Thanks to its large community of developers and users, there has never been a better time to start learning a web development framework!

Why Django?
Since the spread of the MVC pattern into web development, and unlike most of the other languages, Python has enjoyed quite a few choices when it comes to web frameworks. Although choosing one from many can be confusing at first, having several competing frameworks can only be a good thing for the Python community.

Django is one of the available frameworks for Python, so the question is: what sets it apart to become the topic of this book?

First of all, Django provides a set of tightly integrated components. All of these components are developed by the Django team itself. Django was originally developed as an in-house framework to manage a series of news-oriented websites. Later, its code was released on the Internet and the Django team continued its development using the open source model. Because of its roots, Django's components were designed for integration, reusability, and speed from the start.

Django's database component, the Object-relational Mapper (ORM), provides a bridge between the data model and the database engine. It supports a large set of database systems, and switching from one engine to another is a matter of changing a configuration file. This gives the developer great flexibility if a decision is made to change from one database engine to another. If you are in trouble, you can find the driver (binary Python package) here: http://www.lfd.uci.edu/~gohlke/pythonlibs/.

In addition, Django provides a neat development environment. It comes with a lightweight web server for development and testing. When debugging mode is enabled, Django provides very thorough and detailed error messages with a lot of debugging information. All of this makes isolating and fixing bugs very easy.

Django supports multilingual websites through its built-in internationalization system. This can be very valuable for those working on websites with more than one language. The system makes translating the interface a very simple task.
The standard features expected of a web framework are all available in Django. These include the following:

- A template and text-filtering engine with simple but extensible syntax
- A form generation and validation API
- An extensible authentication system
- A caching system for speeding up the performance of applications
- A feed framework for generating RSS feeds

Even though Django does not provide a JavaScript library to simplify working with Ajax, choosing one and integrating it with Django is a straightforward matter, as we will see in later chapters.

So, to conclude, Django provides a set of integrated and mature components with excellent documentation, thanks to its large community of developers and users. With Django available, there has never been a better time to start learning a web development framework!

**Inside Django**

We will mention some important reasons why we use Django for better web development. Some of the most important features are explained in the following subsections.

**Django is mature**

Many corporations are directly using Django in their production and with constant contributions from developers around the world. Some famous sites include Pinterest and Quora. It has established itself as the perfect web development framework.

**Batteries included**

Django follows Python's batteries included philosophy, which means Django comes with many extra features and options that are important in solving common problems faced during web development.

**Tight integration between the component and modular framework**

Django is very flexible in terms of its integration with their party module. The chances of there existing a popular project (for example, MongoDB in database domain or SocialAuth in OpenID main) that does have an Application Program Interface (API) or complete plugin for Django integration are very few.
Object-relational mapper
This is one of the most important parts of the Django project. Django's database component, the ORM, provides a bridge between the data model and the database engine. The ORM layer provides features such as encapsulation, portability, safety, and expressiveness to Django's Modal Class, which are mapped to the configured database of choice.

Clean URL design
The URL system in Django is very flexible and powerful. It lets you define patterns for the URLs in your application and to define Python functions to handle each pattern.

This enables developers to create URLs that are both human-friendly (avoiding URL ending patterns such as .php, .aspx, and so on) and search engine-friendly.

Automatic administration interface
Django comes with an administration interface that is ready to be used. This interface makes the management of your application's data a breeze. It is also highly flexible and customizable.

Advanced development environment
In addition, Django provides a neat development environment. It comes with a lightweight web server for development and testing. When the debugging mode is enabled, Django provides very thorough and detailed error messages with a lot of debugging information. All of this makes isolating and fixing bugs very easy.

What's new in Django 1.6 and 1.7
With the latest release, version 1.6, Django has brought some major changes and a few of them are as follows:

- Python 3 is officially supported with this release, which means it is stable and can be used in production.
- The layout is simple. New defaults have been added, the Django Admin template has been added by default, and the Sites package has been removed.
- Clickjacking prevention has been added.
- The default Database is SQLite3.
- As old APIs are deprecated, the biggest change is that the transactions have been improved. The DB layer auto-commit is enabled by default.
• The DB connection in this release is persistant. Until Django 1.5, a new connection was made for every HTTP request, but from 1.6, the same connection will be reused between requests.

• Time zone defaults to UTC.

• Simple application integration.

• Scalable.

• Powerful configuration mechanism.

• There is no need to have a `models.py` file if you don't have a model.

• A new method has been added for its subclasses.

• It allows a cursor to be used as a context manager.

• Many features have been added for internationalization, form, and file upload.

• It has a better feature to avoid CSRF.

• Apart from these, a binary field has been introduced, as have HTML 5 input fields (e-mail, URL, and number).

You can read the newly added features in detail here: https://docs.djangoproject.com/en/1.7/releases/1.7/.

### Supported databases

Django has a great and powerful respect for data. Model the data correctly, and the rest of the site will just fall into place. With the convention that Django was designed for relational database, unofficial NoSQL implementation exists for Django as well. Here is the list of relational databases that Django supports:

- **SQL**: SQLite, MySQL, and PostgreSQL.
- **SQLite**: This is the default database for Django applications and is mainly used for testing purposes.
- **PostgreSQL**: This is an open source, widely used RDBS. We will build our microblogging example based on this.

MySQL and PostgreSQL are the two most common databases used in the Django community, and PostgreSQL is the most popular in the Django community.
Introduction to Django

• **NoSQL**: How about having a single table for your data, whether it contains user's information or their comments, and so on? In other words, how about having no rules for the structure of inserted data or nesting data, like Articles with subdocument array with comments? Sound strange? Yes, it is. In the early days, people were using the one and only relational database concept, but since the dawn of the cloud computing era, programmers love to implement NoSQL architecture for every possible single project. It doesn't store and doesn't follow any normal forms. You can't use joins, but there are many other advantages of using it.

App Engine, MongoDB, Elasticsearch, Cassandra, and Redis are some famous NoSQL DBs that Django supports. MongoDB is getting popular among the Django community these days.

• **MongoDB**: This is an open source, widely used NoSQL document-based database. We will be using it for creating our second small application for URL shortener.

In this book, we will mainly deal with three databases from the preceding list, but implementation of others can be almost identical with minimal configuration changes.

There are many famous websites powered by Django. Some of them are as follows:

• **Pinterest**: A content sharing service, especially for images and videos
• **Disqus**: A blog comment hosting service
• **Quora**: A question-and-answer based website
• **Bitbucket**: A free code hosting site for Git and mercurial
• **Mozilla Firefox**: The Mozilla support page

What you will learn using this book

This book focuses on building a microblogging web application and adding common Web 2.0 features to it. Some of these features are as follows:

• **Creating Django view, model, and controller**: This ideally deals with learning the Django framework, that is, how requests are handled on controllers to render the view after making the required manipulations with models that are stored on the database.

• **Tags and tag clouds**: In the microblogging site project, every message will have a hashtag in it (a tag starting with #). The mapping of these tags will be dealt with in this section.
• **Content customization and searching**: Searching for messages based on keywords or hashtags.

• **Ajax enhancements**: Using Ajax for autocomplete during search or tagging, and making edits in place for saved messages or tags.

• **Friend networks**: Listing all the friends of the profile and calculating other vital statistics.

Instead of concentrating on teaching various Django features, this book uses a tutorial style to teach how to implement these features using Django. Thus, it works as a complementary resource to the official Django documentation, which is freely available online.

Interested? Great! Prepare for the ride, as I guarantee that it will be both fun and interesting.

**Summary**

In this chapter, we have learned why web development is getting an edge and what has changed in the web technologies domain; how to leverage new web technologies using the Python and Django frameworks; what Django actually is and what we can achieve using it; and finally, the different kind of databases that support Django.

In the next chapter, we will cover the installation of Python and Django on various operating systems, such as Windows, Linux, and Mac, and setting up our first project using the Django platform.
Where to buy this book

You can buy Learning Django Web Development from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.