A brief introduction to Django

Guido de Hooge, Jan Oude Vrielink

Introduction

This document gives you a brief introduction and overview of Django[1]. Django is a free open-source web framework for Python. It offers developers a framework to quickly create scalable web applications. The document starts with a general description of Django. After that, reusable apps, the Model View Template structure, Django Flow and user authentication are explained. In the end, you find some tips about getting started with Django.

Background

Django offers fast and scalable development of web applications. Large companies, including Instagram, Spotify and Dropbox use Django. 45% of the Python developers used Django according to the 2021 JetBrains Developer's Survey[2], which surveyed over 31.000 developers. Hence it is not likely that Django will become irrelevant soon as a web development framework [1].

The founders of Django are Adrian Holovaty and Simon Willison. They created Django to fulfil their need to develop database-driven applications rapidly. Django was released as an open-source project in July 2005 [2].

The Django project has an active community. The Django community consists of Django contributors and users. The Django Forum is a website where the community discusses various matters relating to Django. The community uses several mailing lists to communicate current Django affairs. Django uses extensive documentation to transfer Django knowledge. Members contribute to Django by writing code, writing documentation or localising Django by translations to other languages. The non-profit Django Software Foundation (DSF) supports the Django community, promotes Django and develops Django [3].

Django uses the BSD 3-clause license. This license allows the software to be used for commercial activities, modified, and distributed. The only constraint is that the software must retain the original license [4].

Python is an object-oriented high-level programming language. It uses dynamic typing and dynamic binding. Python is well known because it is easy to learn, and the language is easily readable. It is mainly used for rapid application development, scripting or as a language to connect existing applications

[1] https://www.djangoproject.com [last accessed: July 6, 2021].
[2] https://www.jetbrains.com/lp/devecosystem-2021/ [last accessed: March 20, 2022].

that use different programming languages. Django is built on Python for rapid application development [5].

Web framework

A web framework supports the development of web applications. A web framework is software for removing the overhead associated with the developers everyday activities in web development. Examples of typical activities are database access and authentication. Instead of developers having to program database access themselves, the developers use a framework that does the activity for them. This way, the developers save time during development. Django is an example of a web framework built on Python. Ruby on Rails[3] built on Ruby, NodeJs[4] and Express[5] built on JavaScript are also examples of well-known web frameworks [6]. Django is an open-source web framework. Django offers a wide variety of components that are useful for web development. Examples include authentication, RSS feeds, contact forms and chat services.

Besides rapid development, Django offers the developer scalability of web applications. Django does this by using a "shared nothing" architecture. This architecture separates components of an application. The architecture assures that it is easy to add hardware on each application level and thus scale up. Big companies such as Instagram and Mozilla use Django. These companies process up to 50.000 hits per second [3].

Django offers rapid web application development with a focus on security. The components developers use in the Django framework protect the application against well-known cyber-attacks such as cross-site scripting, request forgery and SQL injection[6] [7, Chapter 7].

Reusable apps

A Django project represents an entire website. One project can consist of multiple apps. Apps are sub-modules of the project. Apps are the building blocks of the project. An app can be used in different projects. This makes apps in Django reusable. The Django framework relies on reusable apps. Reusable apps ensure that developers do not have to 'reinvent the wheel' for everyday web development activities such as user authentication, database management, testing, admin and sessions. Reusable apps are wrapped in Django packages. A package facilitates developers to integrate the reusable app into their projects.

Packages are available through widely-used libraries. An example of a widely-used library is the Python Package Index[7] [8] (PyPI). Django developers easily

[3] https://rubyonrails.org [last accessed: November 3, 2021].
[4] https://nodejs.org [last accessed: November 3, 2021].
[5] https://expressjs.com [last accessed: November 4, 2021].
[6] https://docs.djangoproject.com/en/3.2/topics/security/ [last accessed: July 6, 2021].
[7] https://pypi.org [last accessed: November 6, 2021].

install packages for their web applications by using Pip. Pip stands for "Pip Installs Packages". Pip is a package management system for installing Python packages from PyPI. PyPI is a famous repository for software written in Python. It contains over 300.000 projects.

Model View Template (MVT)

Django uses the MVT structure. The MVT structure is strongly related to the Model View Controller (MVC) architectural pattern [9]. Both structures separate data into three components with their responsibility. The components of the two structures are almost identical but use different naming for specific responsibilities[8]. The model manages the data in both structures. The view describes what data you see in MVT. In MVC, what data you see is the responsibility of the controller. The view describes how you see the data in MVC. The template describes how you see the data in MVT. Figure 1 summarises MVT and MVC.

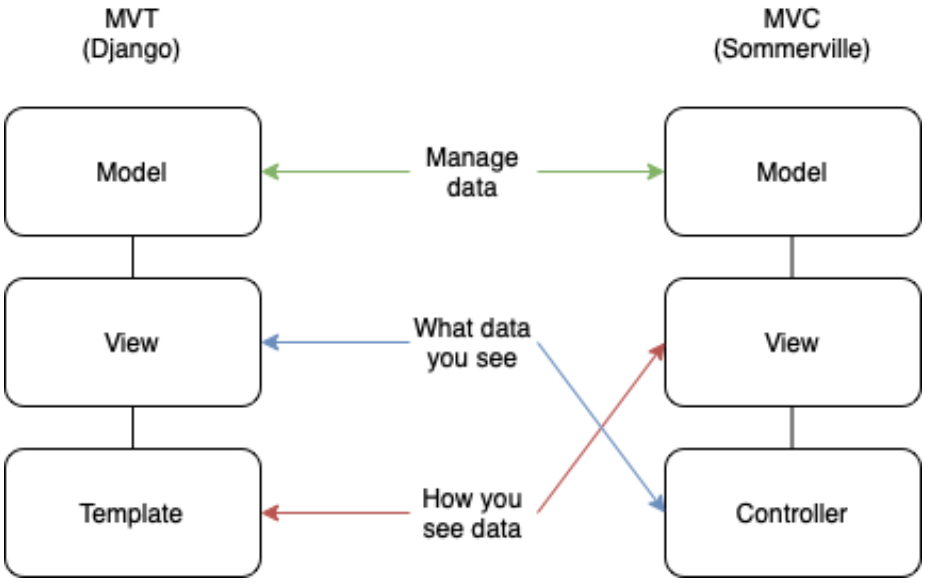

Figure 1: MVT and MVC

In Django, the model is responsible for maintaining the data. A model is a high-level abstraction of a database. The model enables developers to focus on data objects instead of the underlying database where the data is stored. The model consists of a Python class with attributes and methods. Each attribute

[8] https://docs.djangoproject.com/en/3.1/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names [last accessed: April 4, 2022].

represents a database column. An instance of the class corresponds with a record from the database. Methods enable developers to work with the model instances. Django provides methods to create, update or delete records. To create or update a record in the database, an instance of the model class calls the save() method. This technique is called Object Relational Mapping (ORM) [10]. ORM creates a layer between the database and the object-oriented programming languages. The layer maps how an object relates to a database table. Developers can store data by working with objects without knowledge of database queries.

The view is responsible for presenting the data in Django. Django creates an HttpRequest object with data about the request when a page is requested from the webserver. The view then takes the object as an argument and returns an HttpResponse object or raises an exception. There are two types of views, function-based and class-based views. A function-based view is a function that takes an HttpRequest as an argument and returns a HttpResponse. Class-based views are Python classes. A class-based view responds to an HttpRequest with methods from a class. Class-based views make object-oriented techniques available for view, such as inheritance. This makes class-based views reusable [11].

The template is responsible for the layout of the web pages. Templates are text files that generate HTML files dynamically by combining front-end code such as HTML/CSS with back-end Python code. Templates separate front-end and back-end code using Django's Template Language (DTL). DTL is the link between front-end scripts and the back-end programming language. DTL enables developers to include Python code in the HTML templates [12]. This separates the front-end from the back-end in the Django HTML templates.

Django flow

Django enables web developers to create dynamic websites. Website Techdifferences[9] explains the difference between static and dynamic web pages.

Static web pages are simple and written in the HTML language, and stored in the server. The server receives a request for a web page. The server sends a response with the requested web page to the client without additional processing. The server locates the page on the hard disk, adds an HTTP header and replies back to an HTTP response. The peculiar thing about a static web page is that the content in these web pages does not change depending on the request. The content is always the same unless the content is changed physically on the server's hard disk. That is the reason these web pages are known as static web pages.

Dynamic web pages provide a solution for static web pages. The dynamic web page content varies depending on several parameters. Dynamic web pages do not simply send an HTML page in response, unlike static web pages. The web server calls a program located on the hard disk to access a database and perform transaction procedures. The application program produces HTML output used

[9] https://techdifferences.com/difference-between-static-and-dynamic-web-pages.html [last accessed: February 10, 2022].

to construct an HTTP response by the web server. The web server sends the created HTTP response back to the web browser.

The Django website For Everyone[10] describes the typical flow of Django tasks. Figure 2 depicts the flow. The arrow from Click to Routing is an HTTP request. The HTTP request is sent to the Django Framework. The Routing function in Django forwards the request to the correct view. The view to be addressed depends on the URL in the HTTP request. The urls.py file specifies for all URLs the correct view. The View function receives the forward HTTP request, gathers relevant information and renders the new web page. An HTTP response sends the web page back to the browser. The arrow from Views to the browser represents this HTTP response. A JSON object can replace the HTTP request object.

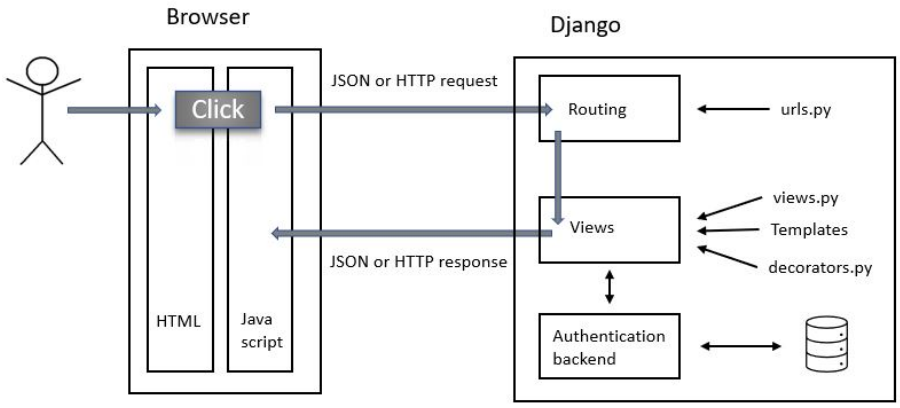

Figure 2: Django flow[11]

Authentication in Django

Django supports authentication with a default module for authentication. This module is called `django.contrib.auth` [3]. The authentication module acts as an interface for the authentication system. The following sections describe the building blocks of the authentication system. The User model is the core of the authentication system. The User model is a class that defines User objects. User objects have attributes such as username and email. The User objects have methods such as get_username() and check_password(raw_password). Developers can change the default User model to a custom User model[12].

[10] https://www.dj4e.com/ [last accessed: February 12, 2022].

[11] https://slidetodoc.com/the-structure-of-a-django-application-dr-charles/ [last accessed: March 23, 2022].

[12] https://docs.djangoproject.com/en/4.0/topics/auth/customizing/#substituting-a-custom-user-model [last accessed: February 9, 2022].

Password is also an attribute of the User object. Django uses hashing and random salting to securely save passwords in the User object database. Salted means that characters are added to the password. Salted passwords prevent dictionary attacks.

The authentication backend is the model for how User objects authenticate. The default authentication backend is the `django.contrib.auth.backends.ModelBackend` module. The default authentication backend uses a username and password to authenticate users. The authentication backend specifies the behaviour of the authenticate method. The authenticate method tries to authenticate a user across a list of authentication backends. The authenticate method returns a User object when valid credentials are found. An authenticated user is associated with a User object. Unauthenticated users do not have an associated User object. Django authentication can be customized by providing an own authentication backend[13].

What is next?

Django has many tutorials and documentation online. Corey Schafer has made a YouTube tutorial for Django[14]. This comprehensive YouTube tutorial is an excellent place to start if you have less experience with programming. The Django documentation website is a good place to start if you are an experienced programmer[15].

Working with Django often requires the installation of many Python packages. Using a virtual environment to manage all the packages is good practice. Corey Schafer made a tutorial[16] to get a basic understanding of working with virtual environments. Good luck!

References

[1] R. Poulton, "Djangosites shiny websites, powered by django," Accessed on 2021-06-17. [Online]. Available: https://djangosites.org/

[2] J. Forcier, P. Bissex, and W. J. Chun, *Python web development with Django*. Addison-Wesley Professional, 2008.

[3] Django Software Foundation, "The web framework for perfectionists with deadlines — django," 2021, Accessed on 2021-06-04. [Online]. Available: https://www.djangoproject.com/

[4] Opensource.org, "The 3-clause BSD license," Accessed on 2021-06-04. [Online]. Available: https://opensource.org/licenses/BSD-3-Clause

[13] `https://docs.djangoproject.com/en/4.0/topics/auth/customizing/` [last accessed: February 9, 2022].

[14] `https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p` [last accessed: May 8, 2022].

[15] `https://docs.djangoproject.com/en/4.0/` [last accessed: May 8, 2022].

[16] `https://www.youtube.com/watch?v=zDYL22QNiWk` [last accessed: May 8, 2022].

[5] Python Software Foundation, "What is python? executive summary," Accessed on 2021-06-17. [Online]. Available: https://www.python.org/doc/essays/blurb/

[6] GeeksforGeeks, "Top 10 frameworks for web applications," 2020, Accessed on 2021-07-06. [Online]. Available: https://www.geeksforgeeks.org/top-10-frameworks-for-web-applications/

[7] M. T. Goodrich and R. Tamassia, *Introduction to computer security*, 1st ed. Pearson Education, 2014.

[8] Django Software Foundation, "Documentation," Accessed on 2021-06-21. [Online]. Available: https://docs.djangoproject.com/en/3.2/intro/reusable-apps/

[9] I. Sommerville, *Software engineering*. Pearson, 2018.

[10] E. J. O'Neil, "Object/relational mapping 2008: hibernate and the entity data model (edm)," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1351–1356.

[11] GeeksforGeeks, "Django project MVT structure - geeksforgeeks," 2021, Accessed on 2021-06-04. [Online]. Available: https://www.geeksforgeeks.org/django-project-mvt-structure/

[12] AskPython, "Django tutorials," Accessed on 2021-06-17. [Online]. Available: https://www.askpython.com/django/