Django and Flask

**Django** is a free and open-source Python web development framework that follows the Model–Template–View (MTV) pattern. It was created in the fall of 2003 by Adrian Holovaty and Simon Willison. Django was built to simplify the website development process. It focuses on reusable components, less code, and rapid development. It was influenced by several earlier frameworks including Zope and Plone and, in turn, has inspired lots of other frameworks that have come after it including Pyramid, CherryPy, Bottle, and Web2py.

**Flask** is a microframework that has a surprising beginning. It actually started as an April Fool’s joke. Before Flask was developed, Armin Ronacher ― the creator of Flask ― had written two other solutions called Werkzeug (a server framework) and Jinja2 (a template library). Armin thought it would be fun to take these two solutions and put them together in a zip file, so he wrote the Denied Framework (as he called it before named it Flask) . When a developer install this Denied Framework, installation program automatically unzip the file and run those two solutions at the same time. Believe it or not, Flask received a positive reaction.

## Flask vs Django: comparison table

|  |  |  |
| --- | --- | --- |
| **Features** | **Flask** | **Django** |
|  |  |  |
| **Type** | Lightweight microframework. | Full-stack framework. |
| **Template Engine** | Flask is based on the Jinja2 format motor. | Built-in template engine. |
| **Admin** | Does not have a feature to manage administration tasks, but it does have an extension: Flask-Admin. | Django already comes with an admin panel |
| **Database** | Has many libraries (e.g., SQLAlchemy, PyMongo, and PonyORM) and extensions (e.g., Flask-Peewee, Flask-Pony, and Flask-Alembic) available. | Built-in ORM, that enables developers to work with several relational database systems. |
|  |  |  |
| **Development** | Provides more control over the components the developer wishes to implement. | Since it integrates all the "batteries", developers have the required tools at their disposal for quick implementation. |
| **Flexibility** | High flexibility - Flask enables developers to build and add functionalities to simple applications in a flexible way. | Low flexibility - Developers are not as free to use other plugins and libraries as they are with Flask. |
|  |  |  |
| **Learn** | Easier to learn. | High learning curve. |

Why should we choose Django over Flask?

Admin

Flask does not have a feature to manage administration tasks, but it does have an extension - Flask-Admin - that can support various database backends, such as MongoEngine and SQLAlchemy.

Contrarily, Django already comes with an admin panel. This web application offers a user interface for developers to handle data based on their particular models. This interface can be customized according to specific needs.

### Database

Django has a **built-in ORM** (Object Relational Mapping) that enables developers to work with several relational database systems (e.g., **PostgreSQL**, **Oracle**, and **MySQL**). The ORM can also support database migrations and other database tasks. It stands out for its practicality and ease of use since it does not have to write long queries and is able to create templates, views and forms according to the specific data models.

Contrarily, Flask, once again, does not provide a built-in solution but has many libraries (e.g., SQLAlchemy, PyMongo, and PonyORM) and extensions (e.g., Flask-Peewee, Flask-Pony, and Flask-Alembic) available for both relational and non-relational databases.

### Development

Django stands out for being fast at developing complex web apps. Since it integrates all the "batteries", developers have the required tools at their disposal for quick implementation. Plus, this philosophy makes it easier to scale and maintain web apps.

Django also lets developers split a project into **multiple small page applications**. The same does not happen with [Flask](https://www.imaginarycloud.com/blog/flask-python/), where each project corresponds to a **single application**. Still, developers can add several views and models to a single application.

### Security

**Django has built-in protection against the most common attack vectors: cross-site request forgery (CSRF), cross-site scripting (XSS), and SQL injection.**

CSRF is an attack that coerces end-users into performing unwanted actions on an authenticated website. XSS is a similar attack that injects malicious code into trusted websites. SQL injection does the same, but with databases.

Flask relies on third-party extensions, giving developers the freedom to set up their own security protocols. Flask-Security library provides the same protections as Django, but it isn’t standard to the framework, meaning more work for the developers who set it up.

### Built-in template engine

Not at all like Django, Flask doesn’t have a built-in layout motor. Flask is based on the Jinja2 format motor. Jinja2 is itself impacted by the Django format motor. Its employments coordinate a sandboxed execution environment, permitting engineers to speed up the advancement handle for energetic web applications. While Django incorporates a built-in format motor that permits engineers to make user-facing layers for web applications consistently and quickly.

### **Project Layout**

Flask requires developers to each project as a single application. But the developers have option to add multiple models and views to the same application. On the other hand, Django allows developers to divide a project into multiple applications. Hence, it becomes easier for developers to write individual applications, and add functionality to the web application by integrating the applications into the project. The small applications further help developers to extend and maintain the web applications written in Python.

### Packages

Flask is minimalistic and has no restrictions, meaning developers can implement exactly what they want using external libraries. This makes Flask flexible and extensible.

Django, on the other hand, has a huge number of built-in packages. To be precise, there were 4,046 Django packages as of September 2019. This means you’re likely will find a package to build and run your application with less effort.

# **Which projects are built with Django?**

Many sites use Django. Since this framework is easy to scale and provides the ability to process a huge amount of data in real time, lots of high-traffic sites use it. Example:-



# **What types of projects are built with Flask?**

Flask is one of the leading Python web development frameworks. According to surveys by JetBrains, Flask usage among developers increased from 41% in 2017 to 47% in 2018. The reasons why world-famous companies such as Airbnb and Reddit use Flask are numerous. Flask gives you more control over your project, since you can choose which components to use and how you interact with them. Also, you can plug in any extension you need.

Websites of these world-famous companies are built with Flask:

