**Django+Flask**

**Django, что это?  
Django** — [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [(написанный](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) на языке программирования Python) для [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), использующий шаблон проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller). Проект поддерживается организацией [Django Software Foundation](https://ru.wikipedia.org/wiki/Django_Software_Foundation" \o "Django Software Foundation).

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, [Ruby on Rails](https://ru.wikipedia.org/wiki/Ruby_on_Rails" \o "Ruby on Rails)). Один из основных принципов фреймворка — [DRY](https://ru.wikipedia.org/wiki/DRY) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Don't repeat yourself*). Благодаря этому сокращается время создания сайтов. То есть при использовании Django не нужно несколько раз переписывать один и тот же код. Фреймворк позволяет создавать сайт из компонентов. Это можно сравнить со строительством крепости с помощью Lego

Для работы с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) Django использует собственный [ORM](https://ru.wikipedia.org/wiki/ORM), в котором модель данных описывается классами Python, и по ней генерируется схема [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%94).

Django применяют для создания:

* CRM-систем.
* CMS.
* Коммуникационных платформ.
* Сервисов бронирования номеров.
* Платформ управления документооборотом.

Django используется в cайтах [Instagram](https://ru.wikipedia.org/wiki/Instagram), [Disqus](https://ru.wikipedia.org/wiki/Disqus" \o "Disqus), [Mozilla](https://ru.wikipedia.org/wiki/Mozilla), [The Washington Times](https://ru.wikipedia.org/wiki/The_Washington_Times), [Pinterest](https://ru.wikipedia.org/wiki/Pinterest), [YouTube](https://ru.wikipedia.org/wiki/YouTube), [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)) и др.

**\*Model-View-Controller** (**MVC**, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, [пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

**\*ORM** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Object-Relational Mapping, [рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *объектно-реляционное отображение*, или преобразование) — технология программирования, которая связывает [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с концепциями [объектно-ориентированных языков программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), создавая «виртуальную [объектную базу](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных».

**Как работает Django?**

Django подходит для разработки высоконагруженных веб-приложений. Это возможно благодаря архитектуре фреймворка: она похожа на MVC. Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представлениеа презентационная логика Представления реализуется в Django уровнем Шаблонов. Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление».

**На что способен Django?**

Django - ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями.

**Почему стоит использовать Django?**

* **Развитая экосистема**

Опытные разработчики рекомендуют воспринимать Django как систему. Это значит, что фреймворк обычно используется с большим количеством сторонних приложений. Их можно выбирать в зависимости от потребностей конкретного проекта.

* **Зрелость**

Django был представлен в 2005 году. За 15 лет существования он сильно изменился и усовершенствовался. В фреймворке постоянно появляются новые возможности, а старые совершенствуются.

* **Административная панель**

Административная панель Django автоматически генерируется при создании приложения. Это избавляет разработчика от необходимости создавать админку вручную.

* **Расширяемость**

Функциональность Django расширяется с помощью плагинов. Это программные модули, которые позволяют быстро добавить на сайт нужную функцию. В [официальном каталоге](https://djangopackages.org/) есть сотни плагинов, которые позволяют легко реализовать на сайте sitemap.xml, управлять доступами, подключить платёжную систему Stripe и так далее. При необходимости можно отключать или заменять плагины, чтобы адаптировать приложение к текущим нуждам проекта.

* **Библиотеки**

В популярных языках программирования есть библиотеки, с помощью которых удобно решать специальные задачи. В библиотеках можно найти готовые решения: функции, классы, конфигурации и так далее. Благодаря таким решениям расширяются возможности языка, а также упрощается создание приложений.

**Но есть и недостатки данного фреймворка:**

#### Не подходит для небольших проектов

Возможности Django могут быть избыточными для небольшого проекта. Однако в экосистеме Python есть другие фреймворки, которые можно использовать в таких случаях. Например, если вы хотите сделать простой чат, лучше использовать Flask.

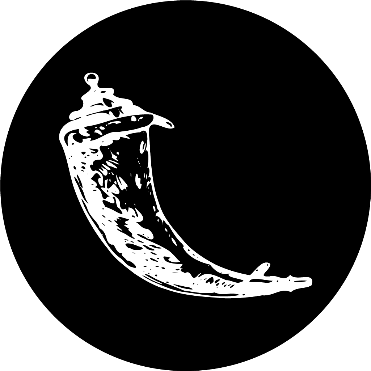
#### Нет поддержки WebSocket по умолчанию

Протокол WebSocket обеспечивает обмен данными между сервером и браузером в режиме реального времени. Django не поддерживает эту функциональность из коробки. Но проблему можно решить с помощью других инструментов, например, aiohttp.

* **Монолитность**Некоторые модули Django, например, ORM или формы, сложно заменить. Разработчикам приходится тратить много времени, чтобы изменить внутреннюю структуру фреймворка и использовать альтернативные инструменты.

#### Непредсказуемость поведения некоторых компонентов

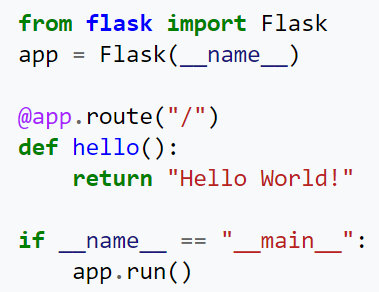
Поведением некоторых компонентов Django не всегда просто управлять. Например, к таким компонентам относится административная консоль. Если вы захотите добавить что-то, что не входит в экосистему Django, это может потребовать серьёзных усилий и много времени.



**А теперь о Flask:**

**Flask** это тоже [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA" \o "Фреймворк) для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке программирования [Python](https://ru.wikipedia.org/wiki/Python" \o "Python). Относится к категории так называемых [микрофреймворков](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA&action=edit&redlink=1" \o "Микрофреймворк (страница отсутствует)) — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности. Однако Flask поддерживает расширения, которые могут добавлять функциональности приложению, как если бы они были реализованы в самом Flask. Существуют расширения для ORM, проверки форм, обработки загрузки, различных технологий открытой аутентификации и нескольких общих инструментов, связанных с инфраструктурой. Расширения обновляются гораздо чаще, чем основная программа Flask.

Простеший пример веб-приложения, которое показывает «[Hello World](https://ru.wikipedia.org/wiki/Hello_World" \o "Hello World)!»:

**Как работает этот код?**

* **Импортируем Flask**
* Инициализируем переменную приложения, используя атрибут \_\_name\_\_
* **@app.route** – это [декоратор Python](https://python-scripts.com/decorators). Мы используем его для **маршрутизации трафика** из определенного URL в расположенной ниже функции. Используя различные вызовы **@app.route**, мы можем «спровоцировать» различные части кода, когда пользователь посещает разные части нашего приложения. В данном случае, у нас только один маршрутизатор «/», который является корнем по умолчанию в нашем приложении.
* Функция под названием hello не так уж важна. Вместо вызова этой функции из той или иной части нашего кода, она будет вызвана автоматически. Это хорошая практика для того, чтобы дать ей релевантное название.
* Функция hallo возвращает строку нашему пользователю. Обычно происходит **рендер шаблона** или **обработка HTML**, чтобы пользователь могу видеть аккуратно оформленную страницу, но и возврат строк Python также хорошо работает.
* Далее обычный шаблон Python, используемый для того, чтобы убедиться в том, что мы не запускаем ничего в автоматическом режиме, если наш код был импортирован из другого скрипта Python.
* В конце вызывается **метод run()** приложения **app**. Это запускает сервер разработки для Flask и дает возможность посетить наше веб приложение с локальной машины путем посещения localhost.

**Django vs Flask:**

**Ключевые отличия:**

* **Flask** предоставляет простоту, гибкость и аккуратность в работе, позволяя пользователю самому выбирать, как реализовать те или иные вещи.
* **Django** предоставляет пакет «*все включено*»: у вас есть панель админа, интерфейсы баз данных, ORM, и структура каталогов для ваших приложений и проектов.

**Как выбрать?**

* **Flask**, если вы хотите получить больше опыта и возможности для обучения, или же если вам нужно **больше контроля** над используемыми компонентами.
* **Django**, если вас интересует **конечный продукт**. Особенно если вы работаете с прямолинейным приложением, таким как новостной сайт, магазин, блог, и если вам нужно, чтобы всегда один прозрачный способ выполнить ту или иную задачу.

*Таким образом оба фреймворка обладают как достоинствами, так и недостатками. Выбор между ними зависит от конкретной задачи.*