**Введение в Django**

**Что такое Django**

Django - это фреймворк для создания веб-приложений с помощью языка программирования Python.

Django был создан в 2005 году, когда веб-разработчики из газеты Lawrence Journal-World стали использовать Python в качестве языка для создания веб-сайтов. А в 2008 году вышел публичный первый релиз фреймворка. На сегодняшний день он продолжает развиваться. Так, текущей версией фреймворка на момент написания этой статьи является версия 4.0, которая вышла в декабре 2021 года. Ну и также постоянно выходят подверсии.

Django довольно популярен. Он используется на многих сайтах, в том числе таких, как Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla и многих других.

Фреймворк является бесплатным. Он развивается как open source, его исходный код открыт, его можно найти репозитории на [githube](https://github.com/django/django).

На Django можно создавать широкий диапазон веб-приложений: от небольших персональных сайтов до высоконагруженных сложных веб-сервисов.

Django по умолчанию предлагает готовую функциональность для ряда распространенных задач, например, систему аутентификации, генерацию карт сайта и т.д., благодаря чему нам можно не изобретать велосипед и достаточно взять уже готовые компоненты.

В Django большое внимание уделяется безопасности, благодаря чему фреймворк помогает разработчикам избежать многих распространенных проблем в системе безопасности, например, sql-инъекций.

Фреймворк Django реализует архитектурный паттерн **Model-View-Template** или сокращенно **MVT**, который по факту является модификацией распростаненного в веб-программировании паттерна MVC (Model=-View-Controller).

Схематично мы можем представить архитектуру MVT в Django следующим образом:

Основные элементы паттерна:

* **URL dispatcher**: при получение запроса на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.
* **View**: получает запрос, обрабатывает его и отправляет в ответ пользователю некоторый ответ. Если для обработки запроса необходимо обращение к модели и базе данных, то View взаимодействует с ними. Для создания ответа может применять Template или шаблоны. В архитектуре MVC этому компоненту соответствуют контроллеры (но не представления).
* **Model**: описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.
* **Template**: представляет логику представления в виде сгенерированной разметки html. В MVC этому компоненту соответствует View, то есть представления.

Когда к приложению приходит запрос, то URL dispatcher определяет, с каким ресурсом сопоставляется данный запрос и передает этот запрос выбранному ресурсу. Ресурс фактически представляет функцию или View, который получает запрос и определенным образом обрабатывает его. В процессе обработки View может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять в нее данные. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код, для генерации которого применяются шаблоны (Template).