**Django**

Django (Джанго) — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией 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). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров.

Для работы с базой данных Django использует собственный ORM(Object-Relational Mapping, объектно-реляционное отображение), в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Миграция - особенность Active Record, позволяющая наращивать схему вашей базы данных со временем.

**Общие команды:**

***django-admin startproject <projectname>***

Создает структуру каталога *проекта* Django с указанным названием в текущем каталоге, или в котором вы укажите. По умолчанию новый каталог содержит manage.py и пакет проекта (содержащий settings.py и другие файлы).

***django-admin startapp <app\_label>***

Создает структуру каталога *приложения* Django с указанным названием в текущем каталоге, или в котором вы укажите. По умолчанию созданный каталог содержит файл models.py и другие файлы шаблона приложения.

*Какая разница между приложением и проектом? Приложение – это Web-приложение, которое предоставляет определенный функционал – например, Web-блог, хранилище каких-то записей или простое приложение для голосования. Проект – это совокупность приложений и конфигурации сайта. Проект может содержать несколько приложений. Приложение может использоваться несколькими проектами.*

***python manage.py runserver***

запускает сервер для разработки на локальном IP используя порт 8000

Если вы хотите изменить порт, укажите его как аргумент. Например, эта команда запускает сервер используя порт 8080:

***python manage.py runserver 8080***

Если вы хотите изменить IP сервера, передайте его вместе со значением порта. Чтобы слушать все публичные IP (полезно, если вы хотите показать свою работу на других компьютерах), используйте *python manage.py runserver 0.0.0.0:8000*

***python manage.py migrate***

 выполняет миграции и автоматически обновляет базу данных.

Команда [migrate](http://djbook.ru/rel1.9/ref/django-admin.html" \l "django-admin-migrate) проверяет настройку [INSTALLED\_APPS](http://djbook.ru/rel1.9/ref/settings.html#std:setting-INSTALLED_APPS) и создает все необходимые таблицы в базе данных, указанной в mysite/settings.py, применяя миграции, которые находятся в приложении

***python manage.py makemigrations <app\_label>***

Выполняя makemigrations, вы говорите Django, что внесли некоторые изменения в ваши модели (в нашем случае мы создали несколько новых) и хотели бы сохранить их в миграции.

***[python manage.py check](http://djbook.ru/rel1.9/ref/django-admin.html" \l "django-admin-check).***

ищет проблемы в вашем проекте, не применяя миграции и не изменяя базу данных.

## Язык шаблонов Django

Шаблон Django – это просто текстовый файл, или строка Python, которые следуют языку шаблонов Django. Определенные конструкции распознаются и интерпретируются шаблонизатором. Основные – это переменные и теги.

Шаблон рендерится с контекстом. Рендеринг заменяет переменные на их значения, которые ищутся в контексте, и выполняет теги. Все остальное выводится как есть.

Синтаксис языка шаблонов Django использует четыре конструкции.

#### Переменные

Переменные выводят значения из контекста, который является словарем.

Переменные выделяются {{ и }}, например:

*My first name is {{ first\_name }}. My last name is {{ last\_name }}.*

*Для контекста {'first\_name': 'John', 'last\_name': 'Doe'} шаблон отрендерит:*

*My first name is John. My last name is Doe.*

Обращение к ключам словаря, атрибутам объектов и элементам списка выполняется через точку:

*{{ my\_dict.key }}*

*{{ my\_list.0 }}*

Если значение переменной является вызываемый объект, шаблонизатор вызовет его без аргументов и подставит результат.

#### Теги

Теги позволяют добавлять произвольную логику в шаблон.

Например, теги могут выводить текст, добавлять логические операторы, такие как “if” или “for”, получать содержимое из базы данных, или предоставлять доступ к другим тегам.

Теги выделяются {% и %}, например:

*{%* [*csrf\_token*](http://djbook.ru/rel1.9/ref/templates/builtins.html#csrf-token) *%}*

Большинство тегов принимают аргументы:

*{%* [*cycle*](http://djbook.ru/rel1.9/ref/templates/builtins.html#cycle) *'odd' 'even' %}*

Некоторые теги требуют закрывающий тег:

*{%* [*if*](http://djbook.ru/rel1.9/ref/templates/builtins.html#if) *user.is\_authenticated %}Hello, {{ user.username }}.{% endif %}*

#### Фильтры

Фильтры преобразуют переменные и аргументы тегов.

Могут выглядеть таким образом:

*{{ django|*[*title*](http://djbook.ru/rel1.9/ref/templates/builtins.html#title) *}}*

Для контекста *{'django': 'the web framework for perfectionists with deadlines'}* этот шаблон выведет:

*The Web Framework For Perfectionists With Deadlines*

Некоторые фильтры принимают аргументы:

*{{ my\_date|*[*date*](http://djbook.ru/rel1.9/ref/templates/builtins.html#date)*:"Y-m-d" }}*

#### Комментарии

Комментарии могут выглядеть таким образом:

*{# this won't be rendered #}*

Тег [{% comment %}](http://djbook.ru/rel1.9/ref/templates/builtins.html#std:templatetag-comment) позволяет добавлять многострочные комментарии.