**Оглавление**

[1 Общие сведения. 1](#_Toc153985773)

[1.1 Что такое Django 1](#_Toc153985774)

[1.2 Установка и настройка. 1](#_Toc153985775)

[1.3 Основные команды Django 1](#_Toc153985776)

[2 Руководство программиста 2](#_Toc153985777)

[2.1 Модели 2](#_Toc153985778)

[2.2 Представления 2](#_Toc153985779)

[2.3 Шаблоны 3](#_Toc153985780)

[2.4 Работа с формами 4](#_Toc153985781)

[3 Руководство пользователя 4](#_Toc153985782)

[3.1 Навигация. 4](#_Toc153985783)

[3.2 Добавление позиции в базу данных. 5](#_Toc153985784)

[3.3 Изменение позиции меню: 5](#_Toc153985785)

[3.4 Удаление позиции меню: 6](#_Toc153985786)

# Общие сведения.

## Что такое Django

**Django** - это специальный инструмент, который помогает разработчикам создавать веб-сайты и приложения. Он называется фреймворком, потому что предоставляет готовые инструменты и правила для работы с веб-сайтами.

## Установка и настройка.

Чтобы установить Джанго, нужно ввести в терминал команду:

$ pip install Django

После ввести в терминал:

$ django-admin startproject

И создаться джанго-проект.

## Основные команды Django

Важные команды, которыми желательно использовать при разработке проекта.

Создание миграций модели:

$ python manage.py makemigrations learning\_logs

Проведение миграций:

$ python manage.py migrate

Создание учтённой записи администратора:

$ python manage.py createsuperuser

# Руководство программиста

## Модели

**Модель Django** — это встроенная функция, которую Django использует для создания таблиц, их полей и различных ограничений. Проще говоря, модели Django - это база данных SQL, которую используют с Django. Модели Django упрощают задачи и организуют таблицы в модели.

Positions – это одна из моделей для базы данных позиции товаров созданная мной.

class Position(models.Model):  
 name = models.CharField('Название', max\_length=30)  
 type = models.CharField('Тип', max\_length=30)  
 gramm = models.CharField('Литры', max\_length=10)  
 sostav = models.TextField('Описание')  
 price = models.IntegerField('Цена')  
  
 def \_\_str\_\_(self):  
 return self.name  
 class Meta:  
 verbose\_name="Позиция"  
 verbose\_name\_plural="Позиции товаров"

## Представления

В Django представления (views) — это код, который управляет тем, что отображается в браузере после запроса пользователя на сервер.

Для создания представления в Django, необходимо определить функцию, которая будет получать запрос и возвращать объект ответа HTTP или вызвать метод, которые Django предоставляет для создания ответа.

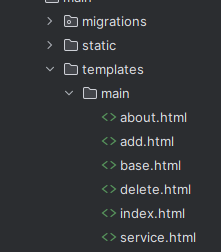
Код файла views.py:

def index(request):  
 positions\_all = Position.objects.all()  
 positions = PostFilter(request.GET, queryset=positions\_all)  
 return render(request, 'main/index.html', {'title': 'Главная страница сайта', 'positions': positions})  
  
  
def about(request):  
 return render(request, 'main/about.html')  
  
  
def service(request):  
 return render(request, 'main/service.html')  
  
  
def add(request):  
 error = ''  
 if request.method == 'POST':  
 form = PosForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('home')  
 else:  
 error = 'Форма заполнена неправильно или не до конца'  
 form = PosForm()  
 context = {  
 'form': form  
 }  
 return render(request, 'main/add.html', context)  
  
  
class Delete(DeleteView):  
 model = Position  
 success\_url = '/'  
 template\_name = 'main/delete.html'  
  
  
class Update(UpdateView):  
 model = Position  
 template\_name = 'main/add.html'  
 form\_class = PosForm  
 success\_url = '/'

## Шаблоны

**Шаблоны** **Django** — **это** файлы, которые определяют, как должен выглядеть HTML-код страниц веб-приложения.

Шаблоны хранятся в папке templates находящейся в директории main/templates/main.



## Работа с формами

В HTML форма представляет собой набор элементов внутри <form>...</form>, которые позволяют посетителю выполнять такие действия, как ввод текста, выбор параметров, манипулирование объектами или элементами управления и так далее, а затем отправлять эту информацию обратно на сервер.

GET и POST - единственные HTTP-методы, которые можно использовать при работе с формами.

Форма входа в систему Django возвращается с использованием POST метода, в котором браузер объединяет данные формы, кодирует их для передачи, отправляет на сервер, а затем получает обратно свой ответ.

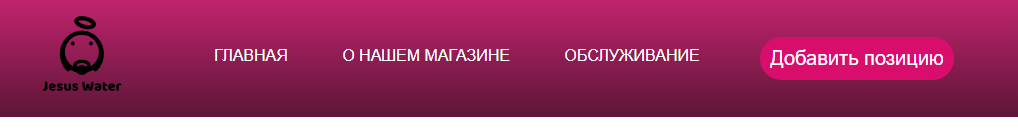
Код в файле forms.py:

class PosForm(ModelForm):  
 class Meta:  
 model = Position  
 fields = ["name", "type", "gramm", "sostav", "price"]  
 widgets = {  
 "name": TextInput(attrs={'placeholder': 'Название','class': 'form-input'}),  
 "type": TextInput(attrs={'placeholder': 'Тип','class': 'form-input'}),  
 "gramm": TextInput(attrs={'placeholder': 'Литры','class': 'form-input'}),  
 "sostav": Textarea(attrs={'placeholder': 'Описание','class': 'form-input'}),  
 "price": TextInput(attrs={'placeholder': 'Цена в руб.','class': 'form-input'}),

# Руководство пользователя

## **Навигация**.

Главное меню сверху сайта это навигация по сайту.



«Главная» - начальная и главная страница.

«О нашем магазине» - переносит на страницу с информацией о магазине.

«Обслуживание» - переносит на страницу с информацией об обслуживаня.

## Добавление позиции в базу данных.

|  |  |
| --- | --- |
| В главном меню выберите пункт «Добавить позицию» |  |
| Заполните поля формы. |  |
| Нажать «Добавить» |  |
| Вас переадресует на Главную страницу, где будет отображена новая позиция. |  |

## Изменение позиции меню:

|  |  |
| --- | --- |
| Перейдите на Главную страницу |  |
| В строке с позицией нажмите кнопку «Изменить» |  |
| Измените нужные поля. |  |
| Нажмите кнопку «Добавить» |  |

## Удаление позиции меню:

|  |  |
| --- | --- |
| Перейдите на Главную страницу |  |
| В строке с позицией нажмите кнопку «Удалить» |  |
| Подтвердите удаление |  |