

Книга 1: Простая регистрация в Django (email + пароль)



Автор [Максат. К.]
Дата начала: 12 мая 2025

♦ Введение

В этой книге мы создадим простую, но полностью рабочую систему регистрации пользователя на Django. Такой подход часто используется в административных панелях, внутренних сервисах, MVP-прототипах или проектах, где важна скорость запуска, а не сложная авторизация.

Мы начнём с самого начала, без лишней теории, только практика. После прочтения ты сможешь:

- Создавать формы регистрации с Django
- Подключать и настраивать систему пользователей
- Обрабатывать ввод данных (email, пароль)
- Реализовать регистрацию с перенаправлением на вход или приветственную страницу

♦ Что тебе нужно до начала

- Python 3.8+
- Django 4+
- Установленная среда разработки (рекомендуется VS Code)
- Понимание основ Python (если пишешь в Django — это уже есть)

Структура книги и глав:

1. Структура проекта и маршруты

- Создание проекта `config` и приложений `accounts`, `blog`
- Настройка маршрутов `urls.py`

2. Пользовательская модель (опционально)

(в этой книге — пропускается, так как используется стандартная модель *User*)

3. Форма регистрации

- `RegisterForm` с валидацией пароля

4. Представление (views)

- `register_view`, сохранение пользователя, авто-вход

5. HTML-шаблоны

- `register.html`, `login.html`, `home.html`
- CSRF, рендеринг формы, логика показа

6. Тестирование

- Проверка регистрации, входа и выхода через браузер

7. Лучшие практики и что дальше

- Заключение, рекомендации, следующая книга

Глава 1. Настройка маршрутов

♦ Исходные данные

У нас есть Django-проект `config`, и два приложения:

- `blog` — для основного контента
- `accounts` — для регистрации и авторизации пользователей

Теперь настроим маршруты так, чтобы:

- Все URL проекта определялись централизованно в `config/urls.py`
 - `accounts` и `blog` имели свои собственные маршруты
 - При необходимости можно было подключить admin-панель
- ♦ **Файл: `config/urls.py`**

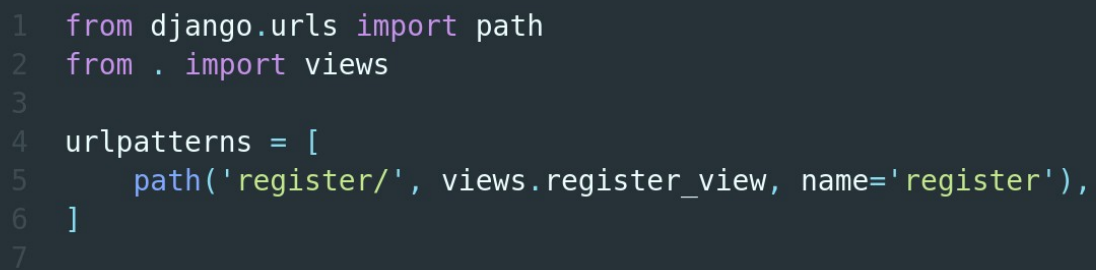
```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('', include('blog.urls')),      # Главная страница и статьи
7     path('accounts/', include('accounts.urls')), # Регистрация и вход
8 ]
9
```

💬 **Объяснение:**

- `admin/` — стандартная админка Django.
- `'accounts/'` — пространство для страниц регистрации, входа, выхода и т. д.

♦ Создаём маршруты в accounts

Создаём файл `accounts/urls.py`, если он ещё не создан:



```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('register/', views.register_view, name='register'),
6 ]
7
```

✓ Результат:

- При переходе на `/accounts/register/` откроется форма регистрации.
- Всё организовано модульно: каждый app — со своими маршрутами.

- ♦ Глава 2. Форма регистрации и обработка запроса
- ♦ 1. Создаём форму регистрации

Файл: `accounts/forms.py`

```
1 from django import forms
2 from django.contrib.auth.models import User
3 from django.core.exceptions import ValidationError
4
5 class RegisterForm(forms.ModelForm):
6     password = forms.CharField(widget=forms.PasswordInput)
7     password_confirm = forms.CharField(widget=forms.PasswordInput, label="Подтвердите пароль")
8
9     class Meta:
10         model = User
11         fields = ['username', 'email', 'password']
12
13     def clean(self):
14         cleaned_data = super().clean()
15         password = cleaned_data.get('password')
16         password_confirm = cleaned_data.get('password_confirm')
17
18         if password and password_confirm and password != password_confirm:
19             raise ValidationError("Пароли не совпадают")
20
21         return cleaned_data
22
```

🗨️ Объяснение:

- Используется `ModelForm` на основе встроенной модели `User`.
- Поля: `username`, `email`, `password`, `password_confirm`.
- В методе `clean()` сравниваются пароли.
- Пароли скрыты через `PasswordInput`.

♦ 2. Создаём представление (view)

Файл: accounts/views.py

```
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.models import User
3 from django.contrib.auth import login
4 from .forms import RegisterForm
5
6 def register_view(request):
7     if request.method == 'POST':
8         form = RegisterForm(request.POST)
9         if form.is_valid():
10             user = form.save(commit=False)
11             user.set_password(form.cleaned_data['password'])
12             user.save()
13             login(request, user) # автоматический вход после регистрации
14             return redirect('home') # перенаправление после регистрации
15     else:
16         form = RegisterForm()
17
18     return render(request, 'accounts/register.html', {'form': form})
```

💬 Объяснение:

- При GET-запросе показывается форма.
- При POST — валидируем данные, сохраняем пользователя, хэшируем пароль.
- Сразу авторизуем пользователя и редиректим на 'home'.

♦ Глава 3. Шаблон формы регистрации

Создаём папку `templates/accounts/` и в ней файл `register.html`:

Файл: `accounts/templates/accounts/register.html`

```
1  <!DOCTYPE html>
2  <html lang="ru">
3  <head>
4      <meta charset="UTF-8">
5      <title>Регистрация</title>
6  </head>
7  <body>
8      <h2>Регистрация</h2>
9      <form method="post">
10         {% csrf_token %}
11         {{ form.as_p }}
12         <button type="submit">Зарегистрироваться</button>
13     </form>
14 </body>
15 </html>
16
```

💬 Объяснение:

- `{{ form.as_p }}` — отрисовывает все поля формы в виде параграфов `<p>`.
- `{% csrf_token %}` — защита от CSRF-атак.
- Простая, минимальная вёрстка, которую легко кастомизировать позже.

◆ Подключение шаблонов

В файле `settings.py` нужно указать, где Django будет искать шаблоны.

Убедись, что у тебя в `DIRS` прописано:

```
1  TEMPLATES = [  
2      {  
3          'BACKEND': 'django.template.backends.django.DjangoTemplates',  
4          'DIRS': [BASE_DIR / 'templates'], # <-- вот здесь  
5      },  
6  ],  
7  ]
```

И структура проекта должна быть примерно такой:

```
1  config/  
2      settings.py  
3  templates/  
4      accounts/  
5          register.html  
6  accounts/  
7      views.py  
8      forms.py  
9      urls.py  
10
```

✓ Теперь, если ты откроешь в браузере путь `/accounts/register/`, отобразится рабочая форма регистрации.

Следующий шаг: **настроим переадресацию после регистрации** и создадим простую домашнюю страницу, чтобы не было ошибки при `redirect('home')`. Готов?

♦ Глава 4. Домашняя страница и редирект

♦ 1. Добавим маршрут и представление для главной страницы

Мы используем приложение `blog` как основное, поэтому создадим в нём:

Файл: `blog/views.py`

```
1 def home_view(request):
2     return render(request=request,
3                   template_name='blog/home.html')
```

Файл: `blog/urls.py`

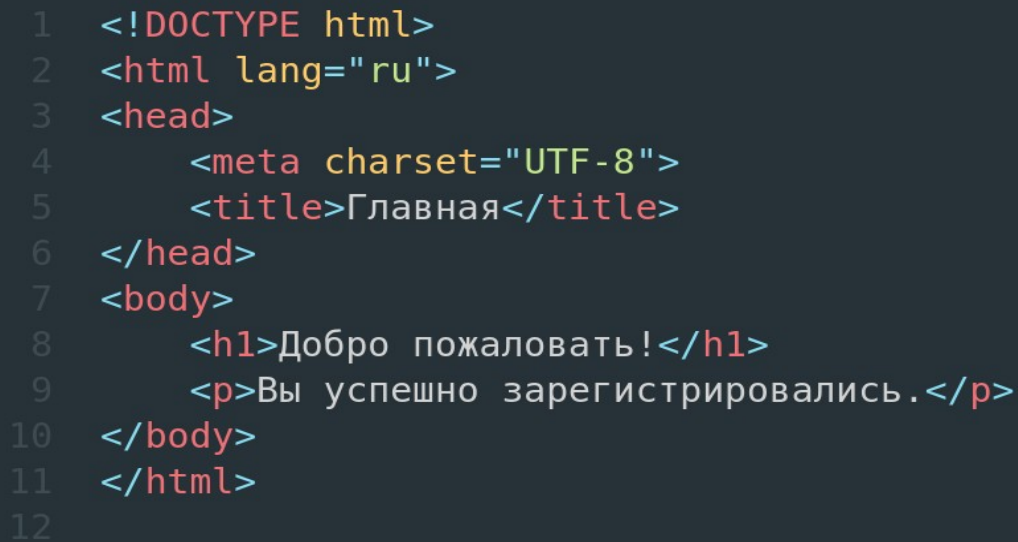
Если `blog/urls.py` ещё не существует — создаём:

```
1 from django.urls import path
2 from .views import home_view
3
4
5 urlpatterns = [
6     path('', home_view, name='home')
7 ]
```

♦ 2. Шаблон главной страницы

Создаём файл:

blog/templates/blog/home.html



```
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4     <meta charset="UTF-8">
5     <title>Главная</title>
6 </head>
7 <body>
8     <h1>Добро пожаловать!</h1>
9     <p>Вы успешно зарегистрировались.</p>
10 </body>
11 </html>
12
```

♦ 3. Проверка редиректа

В `register_view()` мы уже делаем:

```
return redirect('home')
```

Теперь это будет вести на главную страницу без ошибок, потому что маршрут 'home' зарегистрирован в `blog/urls.py`.

✓ Теперь ты можешь:

- Перейти по адресу `/accounts/register/`
- Зарегистрироваться
- Быть автоматически залогиненным
- Увидеть домашнюю страницу с сообщением о регистрации

♦ Глава 5. Вход и выход пользователя

♦ 1. Представление для входа

Django уже предоставляет готовую выюху `LoginView`, которую можно использовать без лишнего кода.

Файл: `accounts/urls.py` (добавим маршруты `login` и `logout`)

```
1 from django.urls import path
2 from django.contrib.auth.views import LoginView, LogoutView
3 from . import views
4
5 urlpatterns = [
6     path('register/', views.register_view, name='register'),
7     path('login/', LoginView.as_view(template_name='accounts/login.html'), name='login'),
8     path('logout/', LogoutView.as_view(next_page='login'), name='logout'),
9 ]
10
```

♦ 2. Шаблон для входа

Создаём файл:

`accounts/templates/accounts/login.html`

```
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4     <meta charset="UTF-8">
5     <title>Вход</title>
6 </head>
7 <body>
8     <h2>Вход</h2>
9     <form method="post">
10         {% csrf_token %}
11         {{ form.as_p }}
12         <button type="submit">Войти</button>
13     </form>
14     <p>Нет аккаунта? <a href="{% url 'register' %}">Зарегистрироваться</a></p>
15 </body>
16 </html>
17
```

♦ 3. Настройки аутентификации

В `settings.py` добавим:

```
1 LOGIN_URL = 'login'
2 LOGIN_REDIRECT_URL = 'home'
3 LOGOUT_REDIRECT_URL = 'login'
```

♦ 4. Добавим навигацию на главной

Обновим `blog/templates/blog/home.html`, чтобы пользователь мог выйти:

```
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4     <meta charset="UTF-8">
5     <title>Главная</title>
6 </head>
7 <body>
8     <h1>Добро пожаловать!</h1>
9
10     {% if user.is_authenticated %}
11         <p>Привет, {{ user.username }}!
12         <a href="{% url 'logout' %}">Выйти</a>
13     </p>
14     {% else %}
15         <p>
16             <a href="{% url 'login' %}">Войти</a> |
17             <a href="{% url 'register' %}">Регистрация</a>
18         </p>
19     {% endif %}
20
21 </body>
22 </html>
23
```

✓ Всё готово:

- `/accounts/register/` — регистрация
- `/accounts/login/` — вход
- `/accounts/logout/` — выход
- Главная страница динамически показывает ссылки в зависимости от входа

◆ Заключение

Поздравляю! Ты только что создал с нуля рабочую систему регистрации пользователей на Django. Без лишнего теоретизирования, без отвлечений — только реальный, минимальный, но боеспособный вариант:

- Регистрация через email и пароль
- Вход и выход
- Перенаправление и шаблоны
- Работа с формами и встроенными инструментами Django

Такой вариант отлично подойдёт для MVP-проектов, админок, панелей управления, или вообще любого проекта, где нужна простая аутентификация без подтверждения по email.

➡ Что дальше?

Если ты хочешь больше возможностей, то впереди — следующие книги этой серии. Мы рассмотрим:

- Регистрацию с подтверждением email
- Активацию аккаунта через ссылку
- Регистрацию через SMS
- Социальные входы (Google, GitHub)
- JWT-токены, SPA/React-входы
- И даже кастомную модель пользователя

Спасибо за чтение 🙌

Ты сделал важный шаг. А теперь — готов ко второму?

♦ Глава 7. Что дальше?

Поздравляю — ты завершил первую книгу из серии про все варианты регистрации в Django.

За короткое время ты:

- Подготовил структуру проекта с двумя приложениями
- Настроил маршруты и шаблоны
- Реализовал регистрацию пользователей через email и пароль
- Добавил вход, выход и автоматический вход после регистрации
- Освоил валидацию формы и работу с `ModelForm`

Это — прочный фундамент. Уже с этим можно строить реальные проекты или админ-панели.

♦ Лучшие практики:

- Используй `LoginRequiredMixin` для защиты страниц
- Не показывай email или пароль в ошибках формы
- Рано или поздно стоит перейти на кастомную модель пользователя

➡ Продолжение серии

Во второй книге мы добавим:

- Подтверждение регистрации по email
- Активацию аккаунта через ссылку
- И защиту от фейковых регистраций

Спасибо, что прочитал. До встречи в следующей книге!