



[Главная](#) → [Django](#) → Классы представлений, регистрация, оптимизация

## Делаем авторизацию пользователей на сайте



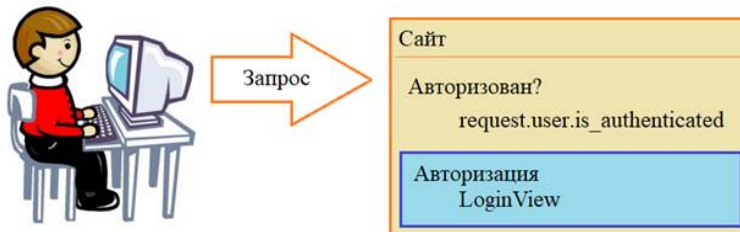
[Смотреть материал на видео](#)



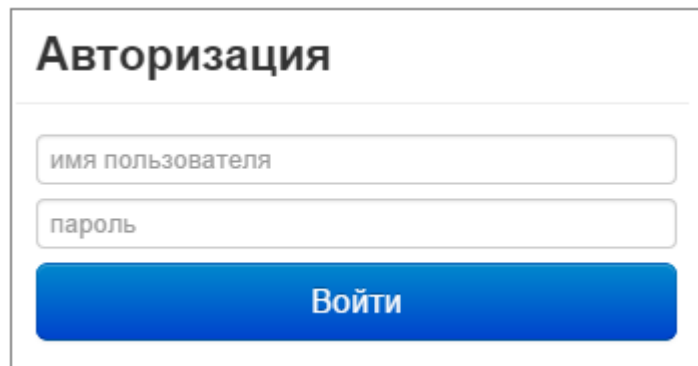
Архив проекта: [lesson-20-coolsite.zip](#)

На предыдущем занятии мы рассмотрели механизм регистрации пользователей. Продолжим эту тему и поговорим об авторизации пользователей на сайте.

Когда на сервер приходит запрос от пользователя, то на стороне сайта важно знать авторизован этот пользователь уже или нет. То есть, он мог ранее уже регистрироваться и авторизоваться на сайте, фреймворк Django сохранил эту информацию в сессии и при поступлении очередного запроса от этого же пользователя мы его должны воспринимать как авторизованного:



Если же пользователь не авторизован, то на сайте должен быть реализован механизм авторизации. Обычно, это форма, где пользователь вводит логин и пароль:



Авторизация

имя пользователя

пароль

Войти

Именно этот функционал мы и реализуем на данном занятии. А подробную информацию об авторизации можно почитать на странице русскоязычной документации:

<https://djbook.ru/rel3.0/topics/auth/default.html>

Итак, первым делом добавим в файле `views.py` класс представления, отвечающий за отображение формы авторизации:

```
class LoginUser(DataMixin, LoginV
    form_class = AuthenticationFo
    template_name = 'women/login.

    def get_context_data(self, *,
        context = super().get_con
        c_def = self.get_user_con
        return dict(list(context.
```

В качестве базового класса используется LoginView, в котором реализована вся необходимая логика работы, а также стандартный класс формы AuthenticationForm. Затем, мы указываем наш шаблон login.html для отображения формы со стандартным содержимым:

```
{% extends 'women/base.html' %}

{% block content %}
<h1>{{title}}</h1>

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Войти</button>
</form>

{% endblock %}
```

Осталось в файле urls.py связать маршрут login с нашим классом представления:

```
path('login/', LoginUser.as_view(
```

а функцию представления login поставим в комментарии. Все, если теперь перейти в браузер и щелкнуть по ссылке «Войти», то увидим стандартную форму авторизации. Причем, она уже работает. Давайте введем «user1», пароль и при нажатии на кнопку «Войти» происходит авторизация пользователя и делается автоматическое перенаправление в профайл. Нам это не нужно. Изменим адрес перенаправления, переопределив метод get\_success\_url() в классе LoginUser:

```
class LoginUser(DataMixin, LoginV
...
    def get_success_url(self):
        return reverse_lazy('home
```

Теперь, при авторизации будем попадать на главную страницу. Этот же эффект можно получить, определив константу:

```
LOGIN_REDIRECT_URL = '/'
```

в файле settings.py пакета конфигурации coolsite.

Следующим шагом улучшим отображение формы авторизации. Для этого в файле forms.py пропишем класс LoginUserForm, унаследовав его от базового AuthenticationForm:

```
class LoginUserForm(Authenticatio
    username = forms.CharField(la
    password = forms.CharField(la
```

И, затем, укажем его в классе представления LoginUser:

```
class LoginUser(DataMixin, LoginV
    form_class = LoginUserForm
...
```

Также в шаблоне login.html сделаем вывод полей через цикл:

```
<div class="form-error">{{ form.n  
  
{% for f in form %}  
<p><label class="form-label" for=  
<div class="form-error">{{ f.erro  
{% endfor %}
```

Обратите внимание, вначале не забываем делать отображение общих ошибок коллекции form.non\_field\_errors. Теперь форма выглядит гораздо лучше.

Следующим шагом, авторизованным пользователям вместо ссылок «Регистрация» и «Войти» будем показывать ссылку «Выйти». Для этого, в шаблоне base.html, добавим следующую проверку:

```
{% if request.user.is_authenticat  
<li class="last"> {{user.username  
{% else %}  
<li class="last"><a href="{% url  
{% endif %}
```

Здесь используется объект request, через него обращаемся к объекту user и уже у этого объекта проверяем свойство is\_authenticated. Если оно принимает значение True, значит, пользователь авторизован, иначе – не авторизован. Для авторизованных пользователей отображается ссылка «Выйти» с именем маршрута logout. Пропишем его в файле urls.py:

```
path('logout/', LoginUser.as_view
```

Если теперь обновить страницу сайта, то увидим эту ссылку «Выйти». Создадим для нее функцию представления, так как непосредственного отображения страницы она не предполагает. Эта функция будет иметь вид:

```
def logout_user(request):  
    logout(request)  
    return redirect('login')
```

Мы здесь используем стандартную функцию `logout()` фреймворка Django для выхода пользователя. А, затем, перенаправляем его на форму авторизации.

В принципе, в базовом варианте авторизация готова. Но мы сделаем еще одно улучшение. При успешной регистрации пользователя будем автоматически его авторизовывать, что, мне кажется логичным действием. Для этого, в классе `RegisterUser` переопределим специальный метод `form_valid()`:

```
def form_valid(self, form):  
    user = form.save()  
    login(self.request, user)  
    return redirect('home')
```

Он отрабатывает при успешной проверки формы регистрации, а значит, при успешной регистрации. Здесь мы самостоятельно сохраняем форму (добавляем пользователя в БД), а затем,

вызываем функцию фреймворка Django login для авторизации пользователя. После этого, делаем перенаправление на главную страницу.

Давайте посмотрим как это будет работать. Откроем форму регистрации, заполним поля и, нажимая на кнопку «Регистрация», попадаем на главную страницу. В главном меню рядом со ссылкой «Выйти» видим имя только что зарегистрированного пользователя, следовательно, мы вошли именно под ним.

Вот так, довольно просто в Django можно выполнять авторизацию пользователей на сайте.

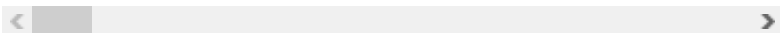
## Видео по теме



#1. Django - что это такое,  
порядок установки



#2. Модель MTV.  
Маршрутизация. Функции  
представления



← [Предыдущая](#)

[Следующая](#) →

© 2021 Частичное или полное копирование информации с данного сайта для распространения на других ресурсах, в том числе и бумажных, строго запрещено. Все тексты и изображения являются собственностью сайта

[Политика конфиденциальности](#) | [Пользовательское соглашение](#)