



Отчёт по лабораторной работе №7

«Авторизация, работа с формами и Django Admin»

По курсу «Разработка интернет приложений»

Выполнил:

Студент группы РТ5-51

Разуваев К. А.

Москва, 2017

---

## **Задание:**

Основная цель данной лабораторной работы – научиться обрабатывать веб-формы на стороне приложения, освоить инструменты, которые предоставляет Django, по работе с формами. Также в этой лабораторной работе вы освоите инструменты Django по работе с авторизацией и реализуете простейшую авторизацию. Напоследок, вы познакомитесь с инструментом администрирования Django – как в несколько строчек кода сделать панель администратора сайта.

1. Создайте view, которая возвращает форму для регистрации.

### **Поля формы:**

- Логин
- Пароль
- Повторный ввод пароля
- Email
- Фамилия
- Имя

2. Создайте view, которая возвращает форму для авторизации.

### **Поля формы:**

- Логин
- Пароль

3. При отправке формы регистрации во view проверять каждый параметр по правилам валидации, если валидация всех полей пройдена, то создавать пользователя и делать перенаправление на страницу логина, а ошибки, если они есть, выводить над формой.

### **Правила валидации:**

- Логин не меньше 5 символов
- Пароль не меньше 8 символов
- Пароли должны совпадать
- Все поля должны быть заполнены
- Логин – уникален для каждого пользователя

4. При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать.

5. Переписать view регистрации с использованием Django Form, правила валидации удалить из view, использовать встроенный механизм валидации полей.

6. Во view авторизации реализовать логин при POST запросе. При успешной авторизации должен происходить переход на страницу успешной авторизации.

7. Страница успешной авторизации должна проверять, что пользователь авторизован. Иначе делать перенаправление на страницу авторизации.

8. Реализовать view для выхода из аккаунта.

9. Заменить проверку на авторизацию на декоратор login\_required

10. Добавить superuser'a через команду manage.py

11. Подключить django.contrib.admin и войти в панель администрирования.

12. Зарегистрировать все свои модели в django.contrib.admin

13. Для выбранной модели настроить страницу администрирования:

- Настроить вывод необходимых полей в списке
- Добавить фильтры
- Добавить поиск
- Добавить дополнительное поле в список

## Исходный код:

### MyApp/urls.py:

```
from django.conf.urls import url, include
from django.contrib import admin
from app.views import *
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', main, name="main"),
    url(r'^(?P<id>\d+)', concerts, name='concerts'),
    url(r'^tickets/', include('app.urls')),
    url(r'^tickets/', TicketsView.as_view(), name='tickets_url'),
    url(r'^hello/', hello, name="hello"),
    url(r'^orders/', OrdersView.as_view(), name="orders"),
    url(r'^login/', loginn, name="login"),
    url(r'^registration/', registration, name='registration_url'),
    url(r'^registration2/', registration2, name='registration2_url'),
    url(r'^auth/', authorization, name='auth_url'),
    url(r'^logout/', logout, name='logout_url'),
]

if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_URL)
```

### views.py:

```
from datetime import datetime
from django.shortcuts import render, HttpResponseRedirect
from django.views.generic import View, ListView
from .models import *
from app.registration import *
from django.contrib.auth.models import User
from django.contrib import auth
from django.contrib.auth import authenticate
from django.contrib.auth.decorators import login_required

def main(request):
    return render(request, 'main.html')

def tickets(request):
    return render(request, 'tickets.html')

def hello(request):
    return render(request, 'hello.html')

def loginn(request):
    return render(request, 'user.html', locals())

def concerts(request, id):
    name = ['BMTH_info', 'CTE_info']
    BMTH_info = 'BMTH bla bla'
    CTE_info = 'Crown the empire bla bla'
    info = [BMTH_info, CTE_info]
    data1 = {'concert': {'id': id}}
    data2 = {'concerts': [{'id': '1', 'concert_name': 'Bring Me The Horizon',
'info': BMTH_info},
```

```

        {'id': '2', 'concert_name': 'Crown The Empire',
'info': CTE_info}}]
    return render(request, 'concerts.html', locals())

class TicketsView(View):
    def get(self, request):
        #variable = 'Django'
        today_date = datetime.now()
        data = {
            'tickets': [
                {'title': 'Первый билет', 'id': 1},
                {'title': 'Второй билет', 'id': 2}
            ]
        }
        return render(request, 'tickets.html', locals())

class TicketView(View):
    def get(self, request, id):
        #variable = 'Django'
        today_date = datetime.now()
        data = {
            'ticket': {
                'id': id
            }
        }
        return render(request, 'ticket.html', locals())

class OrdersView(ListView):
    model = Order
    template_name = 'orders.html'

def registration(request):
    errors = {'username': '', 'password': '', 'password2': '', 'email': '',
'firstname': '', 'surname': ''}
    error_flag = False
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors['username'] = 'Введите логин'
            error_flag = True
        elif len(username) < 5:
            errors['username'] = 'Логин должен превышать 5 символов'
            error_flag = True
        elif User.objects.filter(username=username).exists():
            errors['username'] = 'Такой логин уже существует'
            error_flag = True
        password = request.POST.get('password')
        if not password:
            errors['password'] = 'Введите пароль'
            error_flag = True
        elif len(password) < 8:
            errors['password'] = 'Длина пароля должна превышать 8 символов'
        password_repeat = request.POST.get('password2')
        if password != password_repeat:
            errors['password2'] = 'Пароли должны совпадать'
            error_flag = True
        email = request.POST.get('email')
        if not email:
            errors['email'] = 'Введите e-mail'
        firstname = request.POST.get('firstname')
        if not firstname:
            errors['firstname'] = 'Введите имя'
        surname = request.POST.get('surname')

```

```

        if not surname:
            errors['surname'] = 'Введите фамилию'
        if not error_flag:
            # ...
            user = User.objects.create_user(username=username,
password=password, email=email, first_name=firstname, last_name=surname)
            return HttpResponseRedirect('/login/')
        return render(request, 'registration.html', locals())

def registration2(request):
    form = RegistrationForm(request.POST or None)
    if request.method == 'POST':
        if form.is_valid():
            user =
User.objects.create_user(username=request.POST.get('username'),
                           email=request.POST.get('email'),
password=request.POST.get('password'),
first_name=request.POST.get('firstname'),
last_name=request.POST.get('surname'))
            # ...
            return HttpResponseRedirect('/login/')
        return render(request, 'registration2.html', {'form': form})

def authorization(request):
    error = ""
    username = None
    password = None
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username, password=password)
        if user:
            auth.login(request, user)
            return HttpResponseRedirect('/')
        else:
            error = "Пользователь не найден"
    return render(request, 'auth.html', locals())

def logout(request):
    auth.logout(request)
    return HttpResponseRedirect('/')

```

### models.py:

```

from django.db import models

# Create your models here.
class Order(models.Model):
    fio_customer = models.CharField(max_length=100)
    computer_model = models.CharField(max_length=100)

```

### admin.py:

```

from django.contrib import admin
from models import *

class OrdersAdmin(admin.ModelAdmin):

```

```

fields = ('fio_customer', 'computer_model')
list_filter = ('fio_customer', 'computer_model')
list_display = ('fio_customer', 'computer_model')
search_fields = ('fio_customer', 'computer_model')
list_per_page = 10

class SubjectsAdmin(admin.ModelAdmin):
    list_per_page = 10

admin.site.register(Order, OrdersAdmin)

```

### registration.py:

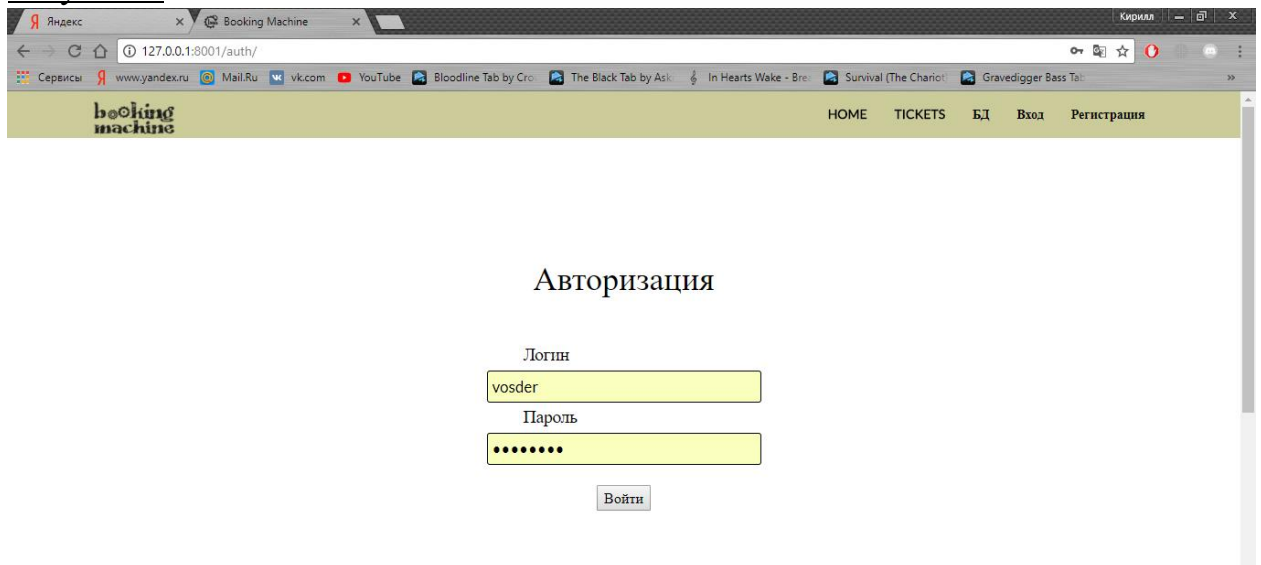
```

from django import forms

class RegistrationForm(forms.Form):
    username = forms.CharField(min_length=5, label='Логин')
    password = forms.CharField(min_length=6, widget=forms.PasswordInput,
label='Пароль')
    password2 = forms.CharField(min_length=6, widget=forms.PasswordInput,
label='Повторите пароль')
    email = forms.EmailField(widget=forms.EmailInput, label='E-mail')
    firstname = forms.CharField(label='Имя')
    surname = forms.CharField(label='Фамилия')

```

### Результат:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8001/auth/'. The browser's tab is titled 'Booking Machine'. The page header features the 'booking machine' logo and navigation links: HOME, TICKETS, БД, Вход, and Регистрация. The main content area is titled 'Авторизация' (Authorization). Below the title is a form with two input fields: 'Логин' (Username) and 'Пароль' (Password). The 'Логин' field contains the text 'vosder'. The 'Пароль' field is masked with dots. At the bottom of the form is a button labeled 'Войти' (Login).

Яндекс Booking Machine Кирилл

127.0.0.1:8001/registration/ Сервисы www.yandex.ru Mail.Ru vk.com YouTube Bloodline Tab by Cro The Black Tab by Ask In Hearts Wake - Bre Survival (The Chariot) Gravedigger Bass Tab

booking machine HOME TICKETS БД Вход Регистрация

## Регистрация

Логин

Пароль

Повторите пароль

E-mail

Имя

Фамилия

Зарегистрироваться

Форма

Яндекс Booking Machine Кирилл

127.0.0.1:8001/registration2/ Сервисы www.yandex.ru Mail.Ru vk.com YouTube Bloodline Tab by Cro The Black Tab by Ask In Hearts Wake - Bre Survival (The Chariot) Gravedigger Bass Tab

booking machine HOME TICKETS БД Вход Регистрация

## Регистрация 2

Логин:

Пароль:

Повторите пароль:

E-mail:

Имя:

Фамилия:

Зарегистрироваться

Select order to change

ADD ORDER +

Q  Search

Action: 

-----

 Go 0 of 3 selected

<input type="checkbox"/> FIO CUSTOMER	COMPUTER MODEL
<input type="checkbox"/> Третий покупатель	Acer
<input type="checkbox"/> Второй покупатель	Lenovo
<input type="checkbox"/> Первый покупатель	ASUS

3 orders

FILTER

By fio customer

All

Второй покупатель

Первый покупатель

Третий покупатель

By computer model

All

ASUS

Acer

Lenovo