Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Разработка интернет приложений»

Отчет по лабораторной работе №6

«Авторизация, работа с формами и Django Admin.»

Выполнил:

студент группы ИУ5

Фомин В. Ю.

Подпись и дата:

Проверил:

доцент каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2017 г.

Оглавление

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’а через комманду 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

Views.py

```
from django.shortcuts import render
from django.http import HttpResponseRedirect, HttpResponse
from django.views.generic import ListView
from django import forms
from django.contrib.auth import authenticate,login, logout
from django.contrib.auth.decorators import login_required
from .models import  *

# Create your views here.
class TravelerList(ListView):
    model = Traveler
    template_name = 'traveler_list.html'


class HotelList(ListView):
    model = Hotel
    template_name = 'hotel_list.html'


class BookingList(ListView):
    model = Booking
    template_name = 'booking_list.html'


def registration_dumb(request):
    errors = {}
    request.encoding = 'utf-8'
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors['uname']='Введите логин'
        elif len(username) < 5:
            errors['uname']='Длина логина должна быть не меньше 5 символов'

        if User.objects.filter(username=username).exists():
            errors['uname']='Такой логин уже занят'

        password = request.POST.get('password')
        if not password:
            errors['psw']='Введите пароль'
        elif len(password) < 8:
            errors['psw']='Длина пароля должна быть не меньше 8 символов'

        password_repeat = request.POST.get('password2')
        if password != password_repeat:
            errors['psw2']='Пароли должны совпадать'

        email = request.POST.get('email')
        if not email:
            errors['email']='Введите email'

        last_name = request.POST.get('last_name')
        if not last_name:
            errors['lname']='Введите фамилию'
```

```
        first_name = request.POST.get('first_name')
        if not first_name:
            errors['fname']='Введите имя'

        if not errors:
            user = User.objects.create_user(username, email, password)
            trav = Traveler()
            trav.user = user
            trav.first_name = first_name
            trav.last_name = last_name
            trav.save()
            return HttpResponseRedirect('/labs/travelers')
        else:
            context = {'errors':errors, 'username':username, 'email': email, 'last_name': last_name, 'first_name':
first_name}
            return render(request, 'registration_dumb.html',context)

    return render(request, 'registration_dumb.html',{'errors':errors})


class RegistrationForm(forms.Form):
    username = forms.CharField(min_length=5,label='Логин')
    password = forms.CharField(min_length=8,widget=forms.PasswordInput, label='Пароль')
    password2 = forms.CharField(min_length=8, widget=forms.PasswordInput, label='Повторите ввод')
    email = forms.EmailField(label='Email')
    last_name = forms.CharField(label='Фамилия')
    first_name = forms.CharField(label='Имя')


def registration_traveler(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        is_val = form.is_valid()
        data = form.cleaned_data
        if data['password']!=data['password2']:
            is_val = False
            form.add_error('password2',['Пароли должны совпадать'])
        if User.objects.filter(username=data['username']).exists():
            form.add_error('username',['Такой логин уже занят'])
            is_val = False

        if is_val:
            data = form.cleaned_data
            user = User.objects.create_user(data['username'], data['email'], data['password'])
            trav = Traveler()
            trav.user = user
            trav.first_name = data['first_name']
            trav.last_name = data['last_name']
            trav.save()
            return HttpResponseRedirect('/labs/authorization')
    else:
        form = RegistrationForm()

    return render(request, 'registration_traveler.html',{'form':form})


@login_required(login_url='/labs/authorization')
def success_authorization(request):
    return HttpResponseRedirect('/labs')
```

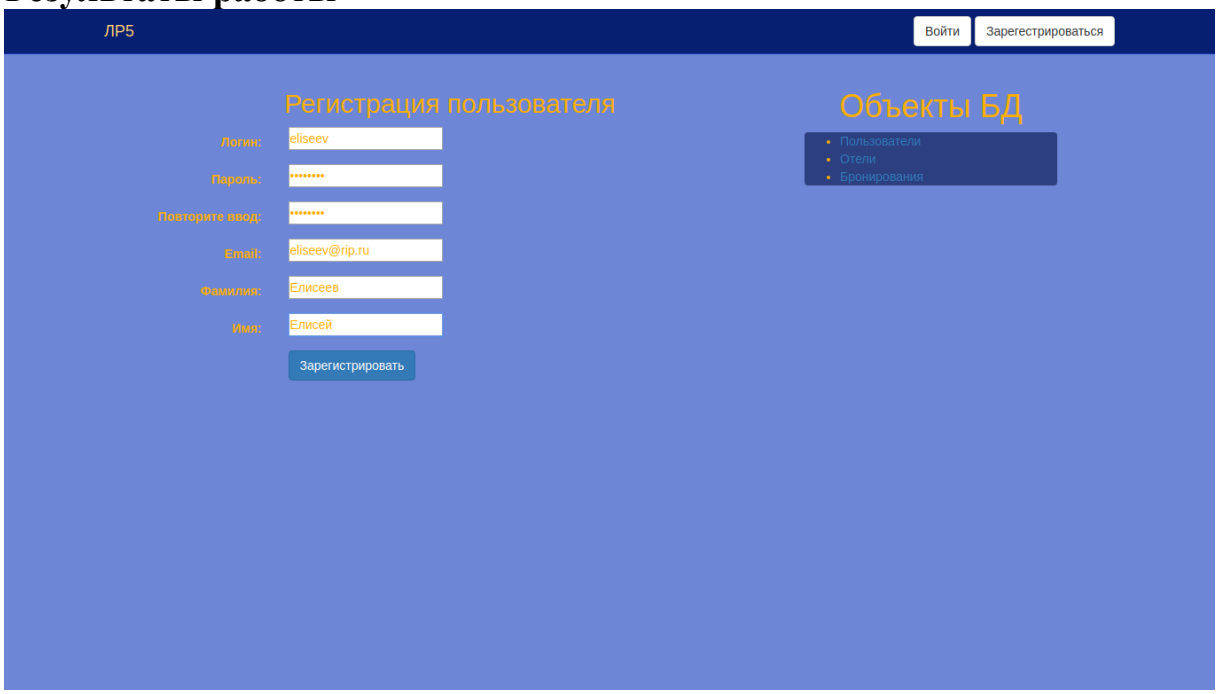
```
def success_authorization_dumb(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('/labs/')
    else:
        return HttpResponseRedirect('/labs/authorization')


def authorization(request):
    errors = {}
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors['uname']='Введите логин'
        elif len(username) < 5:
            errors['uname']='Длина логина должна быть не меньше 5 символов'

        password = request.POST.get('password')
        if not password:
            errors['psw']='Введите пароль'
        elif len(password) < 8:
            errors['psw']='Длина пароля должна быть не меньше 8 символов'

        user = authenticate(request, username=username, password=password)
        if user is None and 'uname' not in errors.keys() and 'psw' not in errors.keys():
            errors['login'] = 'Логин или пароль введены неправильно'

        if not errors:
            login(request,user)
            #return HttpResponseRedirect('/labs/success_authorization_dumb')
            return HttpResponseRedirect('/labs/success_authorization')
        else:
            context = {'errors':errors}
            return render(request, 'authorization.html',context)

    return render(request, 'authorization.html',{'errors':errors})

def logout_view(request):
    logout(request)
    return HttpResponseRedirect('/labs/')

class AutorizationForm(forms.Form):
    pass
```

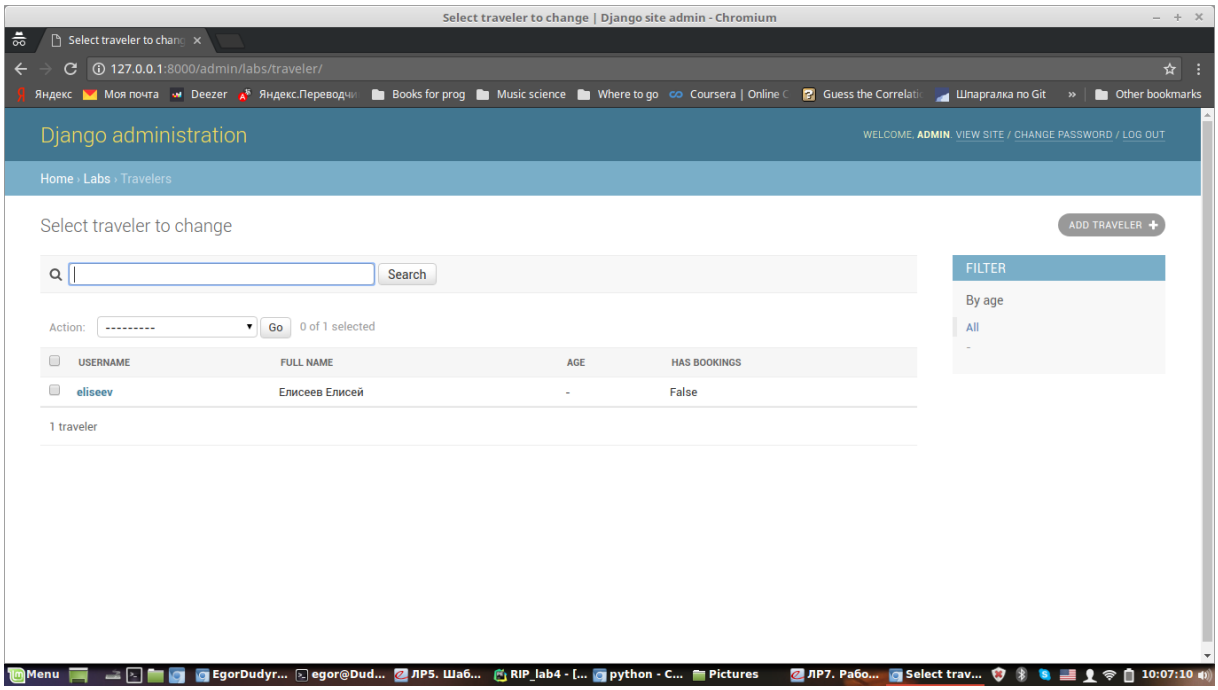
Admin.py

```
from django.contrib import admin
from .models import *

# Register your models here.
@admin.register(Traveler)
class TravelerAdmin(admin.ModelAdmin):
    #fields = ('first_name', 'last_name')
    list_display = ('username','full_name','age','has_bookings',)
    list_filter = ('age',)
    search_fields = ['last_name', 'first_name']

    def full_name(self, obj):
        return "{} {}".format(obj.last_name, obj.first_name)

    def username(self, obj):
        return "{}".format(obj.user.username)

    def has_bookings(self, obj):
```

```
        hs = Booking.objects.filter(user=obj)
        return len(hs)>0


@admin.register(Hotel)
class HotelAdmin(admin.ModelAdmin):
    empty_value_display = '-empty-'

@admin.register(Booking)
class BookingAdmin(admin.ModelAdmin):
    empty_value_display = '-empty-'
```

Результаты работы




Select traveler to change | Django site admin - Chromium
127.0.0.1:8000/admin/labs/traveler/
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Labs › Travelers
Select traveler to change
ADD TRAVELER
Search
FILTER
By age
All
Action: --------- Go 0 of 1 selected
USERNAME
FULL NAME
AGE
HAS BOOKINGS
eliseev
Елисеев Елисей
-
False
1 traveler