Московский государственный технический
университет им. Н.Э. Баумана
Факультет «Информатика и системы управления» Кафедра
«Системы обработки информации и управления»

“Разработка интернет- приложений”

«Python-классы»

Лабораторная работа № 6

Студент группы ИУ5-53
_____Атаманов В. В.
Преподаватель
_____ Гапанюк Е. Ю.

Москва 2017

Задание и порядок выполнения

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 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

Листинг:

models.py:

```
from django.db import models


class Author(models.Model):
    first_name = models.CharField(max_length=32)
    last_name = models.CharField(max_length=32)

    def __str__(self):
        return '{} {}'.format(self.first_name, self.last_name)

    class Meta:
        ordering = ('first_name', 'last_name')


class Book(models.Model):
    title = models.CharField(max_length=64)
    author = models.ForeignKey(Author)
    image = models.ImageField(upload_to='static/books')
    year = models.IntegerField(default= None)
    count = models.IntegerField(default= 0)

    def __str__(self):
        return self.title

    class Meta:
        ordering = ('title', 'image', 'year', 'count')
```

views.py:

```
from django.views.generic.list import ListView
from django.utils import timezone
from django.shortcuts import render, redirect
from django.contrib.auth import login, logout, authenticate
from django.views.generic.edit import *
from django.contrib.auth.decorators import login_required
from .forms import *
from django.views.generic.base import TemplateView
from django.utils.decorators import method_decorator
import json
from django.http import HttpResponseBadRequest, HttpResponse
from braces.views import JSONResponseMixin, AjaxResponseMixin


class FrontPage(TemplateView):

    template_name = "front.html"


class SignUp(FormView):
    template_name = 'sign_up.html'
    form_class = UserCreateForm
    success_url = '/store/'

    # def form_valid(self, form):
    #     return super(SignUp, self).form_valid(form)
```

```
class LogIn(FormView):

    template_name = 'log_in.html'
    form_class = AuthenticateForm
    success_url = '/store/'

    def post(self, request, *args, **kwargs):
        form = AuthenticateForm(data=request.POST)
        if form.is_valid():
            user = authenticate(username=form.cleaned_data['username'],
password=form.cleaned_data['password'])
            if user is not None and user.is_active:
                login(request, user)
                return redirect(self.success_url)
        return render(request, self.template_name, {'form': form})


@method_decorator(login_required, name='dispatch')
class Store(ListView):

    model = Book
    template_name = 'store.html'
    paginate_by = 6

    def get_context_data(self, **kwargs):
        context = super(Store, self).get_context_data(**kwargs)
        context['now'] = timezone.now()
        return context

    def get_queryset(self):
        if self.request.GET.get('stock') is not None:
            return Book.objects.filter(count__gt=0)
        else:
            return Book.objects.all()


class BookView(TemplateView):

    template_name = "book.html"

    def get(self, request, *args, **kwargs):
        return render(request, self.template_name, {'book':
Book.objects.get(id=id)})


class AddBook(CreateView):
    # JSONResponseMixin, AjaxResponseMixin,

    template_name = "add_book.html"
    form_class = BookForm

def logout_view(request):
    logout(request)
    request.session.clear()
    return redirect('front')
```

forms.py

```
from django.contrib.auth.models import User
from django.contrib.auth.forms import AuthenticationForm, UserCreationForm
from django.forms import *
from .models import *


class UserCreateForm(UserCreationForm):
    email = EmailField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Email', 'type': 'email'}))
    first_name = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Имя'}))
    last_name = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Фамилия'}))
    username = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Логин'}))
    password1 = CharField(required=True,
widget=widgets.PasswordInput(attrs={'placeholder': 'Пароль'}))
    password2 = CharField(required=True,
widget=widgets.PasswordInput(attrs={'placeholder': 'Подтверждение пароля'}))

    class Meta:
        fields = [
            'email',
            'first_name',
            'last_name',
            'username',
            'password1',
            'password2'
        ]
        model = User


class AuthenticateForm(AuthenticationForm):

    username = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Логин'}))
    password = CharField(required=True,
widget=widgets.PasswordInput(attrs={'placeholder': 'Пароль'}))


class BookForm(ModelForm):
    title = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Название'}))
    image = ImageField(required=True)
    year = IntegerField(required=True,
widget=widgets.NumberInput(attrs={'placeholder': 'Год издания'}))
    count = IntegerField(required=True,
widget=widgets.NumberInput(attrs={'placeholder': 'Количество'}))

    class Meta:
        model = Book
        fields = '__all__'


class AuthorForm(ModelForm):
    first_name = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Имя'}))
    last_name = CharField(required=True,
widget=widgets.TextInput(attrs={'placeholder': 'Фамилия'}))

    class Meta:
        model = Author
        fields = '__all__'
```