Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»

"Разработка интернет-приложений"

Лабораторная работа №9

отчет

Студент группы РТ5-51 Попков В.Е.

Москва 2016

Задание и порядок выполнения

1. Создайте view, которая возвращает форму для регистрации

```
def get(self,request):
    return render(request,'lab/register.html',{'errors':'','login':'','email':'','surname':'','name':''})
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{{ errors }}
<form action="/register" method="POST">
    <label for="login">Логин</label>
    <input type="text" name="login" id="login" value="{{ login }}"><br>

    <label for="password">Пароль</label>
    <input type="password" name="password" id="password"><br>

    <label for="password">Пароль</label>
    <input type="password" name="password2" id="password"><br>

    <label for="password">Email</label>
    <input type="text" name="email" id="password" value="{{ email }}"><br>

    <label for="password">Фамилия</label>
    <input type="text" name="surname" id="password" value="{{ surname }}"><br>

    <label for="password">Имя</label>
    <input type="text" name="name" id="password" value="{{ name }}"><br>

    {% csrf_token %}
    <button type="submit">Зарегистрироваться</button>
</form>
</body>
</html>
```

2. Создайте view, которая возвращает форму для авторизации.

```
class Login(View):
    def get(self, request):
        return render(request, 'lab/login.html', {'errors': '', 'login': ''})
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{{ errors }}


<form action="/login" method="POST">
   <label for="login">Логин</label>
   <input type="text" name="login" id="login" value="{{ login }}"><br>

   <label for="password">Пароль</label>
   <input type="password" name="password" id="password"><br>

    {% csrf_token %}

   <button type="save">Войти</button>
</form>
<span class='a1'><a href='register' title="Выход">  Регистрация </a></span>
</body>


</html>
```

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.

```
def post(self, request):
    login = request.POST['login']
    password = request.POST['password']
    password2 = request.POST['password2']
    email = request.POST['email']
    surname = request.POST['surname']
    name = request.POST['name']
    errors =[]
    if len(login) < 5:
        errors.append("Логин короткий")
    if len(password) < 8:
        errors.append("Пароль короткий")
    if password != password2:
        errors.append("Пароли не совпадают")
    if len(email) < 1 or len(surname) < 1 or len(name) < 1:
        errors.append("Все поля должны быть заполнены")
    if len(errors) == 0:
            usrs = User.objects.filter(username=login)
            if len(usrs) > 0:
                errors.append("Пользователь с данным логином уже существует")
            else:
                u = User(username=login, email=email, last_name=surname, first_name=name)
                u.set_password(password)
                u.save()
    if len(errors) > 0:
            return render(request, 'lab/register.html', {'errors': mark_safe('<br>'.join(errors)), 'login': login,
                                                         'email': email, 'surname': surname, 'name': name})
```

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

```
class RegisterForm(forms.Form):
    login = forms.CharField(label='Login', min_length=5)
    password = forms.CharField(label='Password', min_length=8, widget=forms.PasswordInput)
    password2 = forms.CharField(label='Password', min_length=8, widget=forms.PasswordInput)
    email = forms.CharField(label='Email', min_length=1)
    surname = forms.CharField(label='Surname', min_length=1)
    name = forms.CharField(label='Name', min_length=1)

    def clean(self):
        cleaned_data = super(RegisterForm, self).clean()
        password = cleaned_data.get('password')
        password2 = cleaned_data.get('password2')
        if password != password2:
            raise forms.ValidationError("Пароли не совпадают")
        usrs = User.objects.filter(username=cleaned_data.get('login'))
        if len(usrs) > 0:
            raise forms.ValidationError("Пользователь с данным логином уже существует")
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{{ errors }}

<form action="/register" method="POST">
  {{ form }}
    {% csrf_token %}
   <button type="submit">Зарегистрироваться</button>
</form>
</body>
</html>
```

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

```
class Login(View):
    def get(self, request):
        return render(request, 'lab/login.html', {'errors': '', 'login': ''})

    def post(self, request):
        username = request.POST['login']
        password = request.POST['password']
        errors = []

        user = authenticate(username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('/')
        errors.append('Логин или пароль неверны')
        return render(request, 'lab/login.html', {'errors': mark_safe('<br>'.join(errors)), 'login': login})
```

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

```
<p>
{% if user.is_authenticated %}

    <span class='a1' >Здравствуй, {{user.username}}</span>
    <span class='a1'><a href='logout' title="Выход">  Выход </a></span>
{% endif %}
</p>
</body>
</html>
```

8. Реализовать view для выхода из аккаунта

```
class Logout(View):
    success_url = "/"
    def get(self, request):
        logout(request)
        return HttpResponseRedirect("/")
```

9. Заменить проверку на авторизацию на декоратор login_required

```
@login_required(login_url='/login')
def home(request):
    a = 'You are authenticated'
    return render(request, 'lab/home.html', {'auth': a})
```

10. Добавить superuser’a через команду manage.py

```
C:\Users\Влад\Downloads\LABA7>python manage.py createsuperuser
Username: admin
Email address: admin@admin.com
Password:
Password (again):
Superuser created successfully.
```

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

```
class Book_User(models.Model):
    user=models.CharField(max_length=200)
    book=models.CharField(max_length=100)
    number=models.IntegerField()
```

```
from django.contrib import admin
from .models import Book_User
# Register your models here.

class PostAdmin(admin.ModelAdmin):
    list_display = ('user', 'book', 'number')
    list_filter = ('user',)
    search_fields = ['book']

admin.site.register(Book_User, PostAdmin)
```