Лабораторная работа №7.

Тема: "Авторизация работа с формами и Dango Admin"

Студент: Щипицин Р.А.,
группа ИУ5-51.

Описание задания лабораторной работы

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 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

Реализация.

1. Создание view

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

2. Создайте view, которая возвращает форму для авторизации.

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

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")
```

```html
<!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.

```python
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, перенаправление на страницу авторизации.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{{ auth }}
<p>
{% if user.is_authenticated %}

    <span class='a1' >HI, HELLO, HOLA , ПРИВЕТ {{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("/")
```

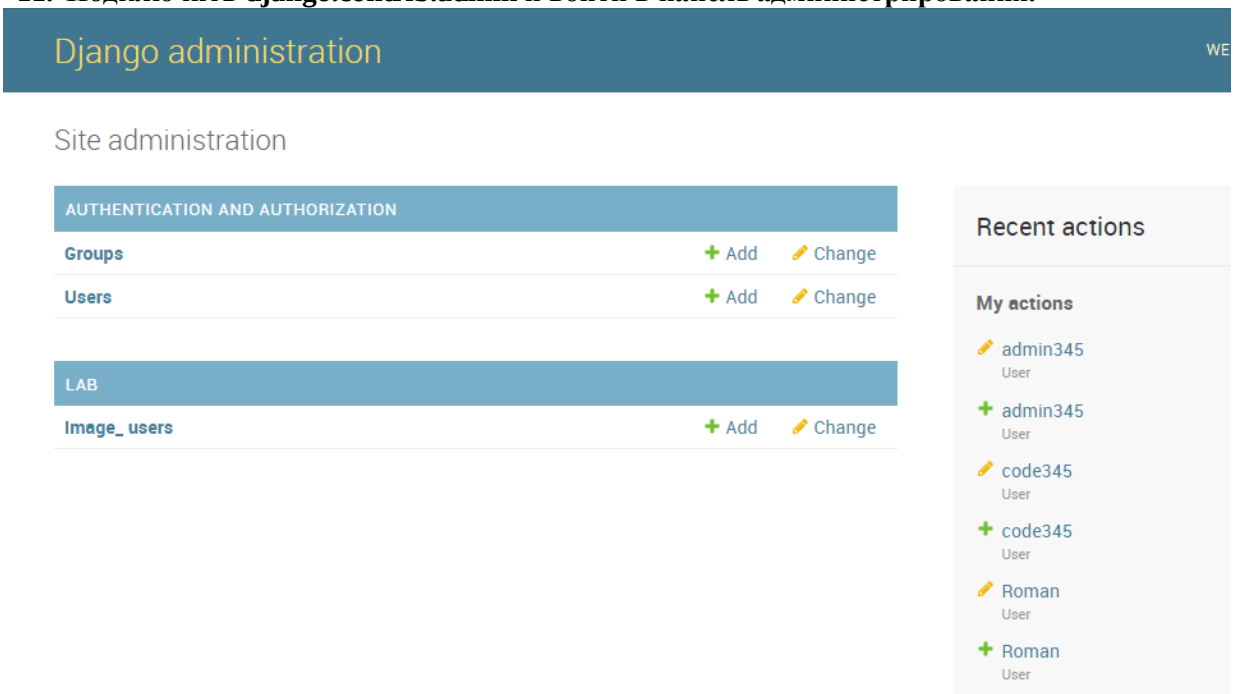
9. Заменить проверку на авторизацию на декоратор login_required

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

10. Добавить superuser'а через команду manage.py

```
C:\Link>python manage.py createsuperuser
Username: aud
Error: That username is already taken.
Username: front
Email address: pgo@bk.ru
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

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

```python
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)
```

Листинг

Register.html

```
<!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>
```

Register2.html

```
<!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>
```

Login.html

```
<!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>
```

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{{ auth }}
<p>
{% if user.is_authenticated %}

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

Admin.py

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

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

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

apps.py

```
from django.apps import AppConfig


class LabConfig(AppConfig):
    name = 'lab'
```

models.py

```
from django.db import models
from django import forms
from django.contrib.auth.models import User
# Create your models here.




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")


class LoginForm(forms.Form):
    login = forms.CharField(label='Login', min_length=5)
    password = forms.CharField(label='Password', min_length=8,
widget=forms.PasswordInput)

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

Urls.py

```
"""lab7 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please
see:
    https://docs.djangoproject.com/en/1.10/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url,
include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from django.conf.urls import include, url
```

```
from . import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^register/$', views.Register.as_view()),
    url(r'^register$', views.Register.as_view()),
    url(r'^login/$', views.Login.as_view()),
    url(r'^login$', views.Login.as_view()),
    url(r'^$', views.home),
    url(r'^succ/$', views.home),
    url(r'^logout$', views.Logout.as_view())
]
```

Vews.py

```
from django.shortcuts import render,redirect, render_to_response
from django.http import HttpResponseRedirect
from django.views import View
from django.utils.safestring import mark_safe
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
# Create your views here.
from lab.models import RegisterForm
class Register(View):
    def get(self, request):
        form = RegisterForm()
        return render(request, 'lab/register2.html', {'errors': '', 'form':
form.as_p()})

    def post(self, request):
        form = RegisterForm(request.POST)

        if not form.is_valid():
            return render(request, 'lab/register2.html', {'errors': '',
'form': form.as_p()})

        u = User(username=form.cleaned_data['login'],
email=form.cleaned_data['email'],
                 last_name=form.cleaned_data['surname'],
first_name=form.cleaned_data['name'])
        u.set_password(form.cleaned_data['password'])
        u.save()
        return redirect('/succ/')
    #def get(self,request):
       # return
render(request,'lab/register.html',{'errors':'','login':'','email':'','surna
me':'','name':''})

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

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

        user = authenticate(username=log, 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})

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

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