

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



Отчет по лабораторной работе № 7

« Работа с формами, авторизация, django admin »

по курсу

“Разработка Интернет-приложений”

ИСПОЛНИТЕЛЬ:

Березин И.С.

Группа ИУ5-53

"__" _____ 2016 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

"__" _____ 2016 г.

Москва 2016

Лабораторная работа №7

Авторизация, работа с формами и Django Admin.

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`'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

Файл `forms.py`:

```
from django import forms
import django.forms.widgets as widgets
from django.utils.translation import ugettext as _
from django.contrib.auth.models import User
def validate_password(password):
    if len(password) < 8:
        raise forms.ValidationError(
            _("Password length should be at least 8 characters"),
            code='invalid'
        )
    def validate_username(username):
        if len(username) < 5:
            raise forms.ValidationError(
                _("Username length should be at least 5 characters"),
                code='invalid'
            )
        if User.objects.filter(username=username).exists():
            raise forms.ValidationError(
                _("This username is already taken"),
                code="invalid"
            )
    class RegistrationForm(forms.Form):
        username = forms.CharField(
            label='Username',
            widget=widgets.TextInput(
                attrs={
                    'placeholder': 'Username',
                    'class': 'form-control input-sm',
                }
            ),
            max_length=150,
            validators=[validate_username]
```

```

)
password1 = forms.CharField(
    label='Password',
    widget=widgets.PasswordInput(
        attrs={
            'placeholder': 'Password',
            'class': 'form-control input-sm',
        }
    ),
    max_length=100,
    validators=[validate_password]
)
password2 = forms.CharField(
    label='Password confirm',
    widget=widgets.PasswordInput(
        attrs={
            'placeholder': 'Password confirmation',
            'class': 'form-control input-sm',
        }
    ),
    max_length=100
)
email = forms.EmailField(
    label='E-mail',
    widget=widgets.EmailInput(
        attrs={
            'placeholder': 'E-mail',
            'class': 'form-control input-sm',
        }
    ),
    max_length=500
)
family_name = forms.CharField(
    label='Family name',
    widget=widgets.TextInput(
        attrs={
            'placeholder': 'Family name',
            'class': 'form-control input-sm',
        }
    ),
    max_length=30
)
first_name = forms.CharField(
    label='First name',
    widget=widgets.TextInput(
        attrs={
            'placeholder': 'First name',
            'class': 'form-control input-sm',
        }
    ),
    max_length=30
)
def clean(self):
    cleaned_data = super(RegistrationForm, self).clean()

```

```

username = cleaned_data.get('username')
password1 = cleaned_data.get('password1')
password2 = cleaned_data.get('password2')
if password1 != password2:
    raise forms.ValidationError(
        _("Passwords should match"),
        code='invalid'
    )
return cleaned_data
def save(self):
    return User.objects.create(
        username=self.cleaned_data['username'],
        password=self.cleaned_data['password1'],
        email=self.cleaned_data['email'],
        first_name=self.cleaned_data['first_name'],
        last_name=self.cleaned_data['family_name']
    )
class AuthorizationForm(forms.Form):
    username = forms.CharField(
        label='Username',
        widget=widgets.TextInput(
            attrs={
                'placeholder': 'Username',
                'class': 'form-control input-sm',
            }
        ),
        max_length=100
    )
    password = forms.CharField(
        label='Password',
        widget=widgets.PasswordInput(
            attrs={
                'placeholder': 'Password',
                'class': 'form-control input-sm',
            }
        ),
        max_length=100
    )

```

Файл views.py:

```

from django.shortcuts import render, redirect
from django.views import View
from django.http import JsonResponse
from .forms import RegistrationForm, AuthorizationForm
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required
# view to display registration form
class RegistrationView(View):
    def get(self, request):
        return render(
            request,
            'registration.html',
            context={

```

```

'page': {'title': 'Registration page'},
'form': RegistrationForm(),
}
)
def post(self, request):
form = RegistrationForm(request.POST)
if form.is_valid():
form.save()
return redirect('auth')
# return JsonResponse(form.errors)
return render(
request,
'registration.html',
context={
'page': {'title': 'Registration page'},
'form': form,
}
)
class AuthorizationView(View):
def get(self, request):
return render(
request,
'authorization.html',
context={
'page': {'title': 'Authorization page'},
'form': AuthorizationForm(),
}
)
def post(self, request):
form = AuthorizationForm(request.POST)
if form.is_valid():
username = form.cleaned_data['username']
password = form.cleaned_data['password']
user = authenticate(username=username, password=password)
if user:
login(request, user)
return redirect('index')
return render(
request,
'authorization.html',
context={
'page': {'title': 'Authorization page'},
'form': form,
}
)
@login_required(login_url='auth')
def index(request):
return render(
request,
'index.html',
context={
'page': {'title': 'Success'},
}
)

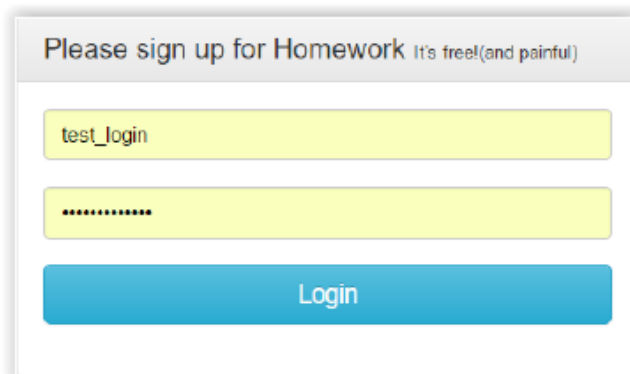
```

Файл admin.py:

```
from django.contrib import admin
from .models import Event
# Register your models here.
class AuthorEvent(admin.ModelAdmin):
    # fields = ('name',)
    list_filter = ('name',)
    list_display = ('name', 'address', 'time')
    actions = ['null_desc']
    search_fields = ('address', )
    def null_desc(self, request, queryset):
        queryset.update(desc=None)
admin.site.register(Event, AuthorEvent)
```

Результаты работы:

Представление для авторизации:

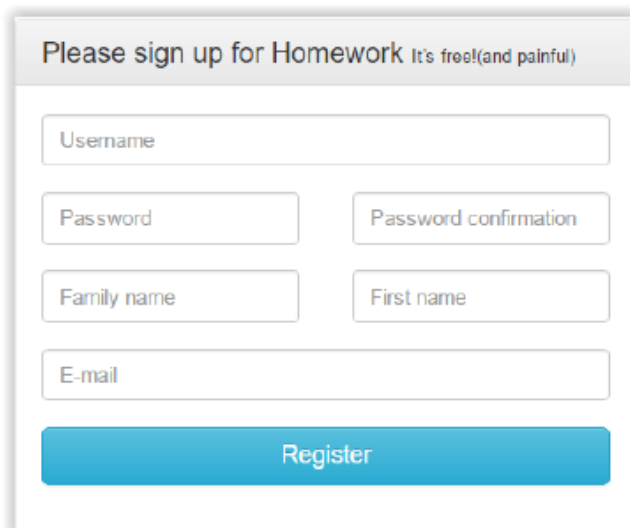


Please sign up for Homework It's free!(and painful)

test_login

Login

Представление для регистрации:



Please sign up for Homework It's free!(and painful)

Username

Password Password confirmation

Family name First name

E-mail

Register

Представление для index.html:

You passed the pain of registration
Don't worry, it could have been much worse You haven't tried to input captcha