

**Московский государственный технический университет  
им. Н.Э. Баумана**

**Разработка интернет-приложений  
Лабораторная работа № 7  
“ Авторизация, работа с формами и  
Django Admin.”**

Выполнил:  
студент группы ИУ5-53  
Пирмамедов М. Э.  
Подпись:  
Дата:

Москва 2017г.

1.Создайте view, которая возвращает форму для регистрации.

Поля формы:

- Логин
- Пароль
- Повторный ввод пароля
- Email
- Фамилия
- Имя

```
def RegisterClass(request):
    names_dict = {}
    errors = []

    if request.method == "POST":
        names_list = ["username", "password", "password2", "email", "firstname", "secondname"]
        names_dict = {x: request.POST.get(x, "") for x in names_list}
        for k, v in names_dict.items():
            print("{}:{}".format(k, v))
        username = request.POST.get("username")
        if not username:
            errors.append('Введите логин')
        elif len(username) < 5:
            errors.append("Логин должен содержать не менее 5 символов")
        password = request.POST.get('password')
        if not password:
            errors.append("Введите пароль")
        elif len(password) < 8:
            errors.append("Длина пароля должна быть не менее 8 символов")
        password_repeat = request.POST.get("password2")
        if password != password_repeat:
            errors.append("Пароли должны совпадать")
        if not errors:
            return HttpResponseRedirect("/computers/")
    return render(request, 'my_app/registration.html', {"errors": errors, 'names_dict': names_dict})
```

Результат

Регистрация

|  |
|--|
| Логин  |
| Пароль   |
| Повторите пароль   |
| e-mail   |
| Фамилия  |
| Имя  |
| <input type="button" value="Отправить"/> <input type="button" value="Отмена"/> |

2. Создайте view, которая возвращает форму для авторизации.

Поля формы:

- Логин
- Пароль

```

def authorization(request):
    names_dict = {}
    errors = []

    if request.method == "POST":
        names_list = ["username", "password"]
        names_dict = {x: request.POST.get(x, "") for x in names_list}
        username = request.POST.get("username")
        if not username:
            errors.append('Введите логин')
        password = request.POST.get('password')
        if not password:
            errors.append("Введите пароль")
        if not errors:
            return HttpResponseRedirect("/computers/")
    print(list(errors))
    return render(request, 'my_app/authorization.html', {"errors": errors, 'names_dict': names_dict})

```

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

Валидация в HTML:

Валидация в view:

См п.2

4. При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать.

## Пример(Неправильно введен логин, появляется ошибка, введенный логин не исчезает)

127.0.0.1:8000/authorize/

Неправильное имя пользователя или пароль

### Авторизация

Логин:

Введите пароль:

[регистрация](#)

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

```
def RegisterDjango(request):
    names_dict = {}
    errors = []

    form = RegisterForm(request.POST or None)

    # TODO заменить на методы класса
    if request.method == "POST" and form.is_valid():
        data = form.cleaned_data
        names_list = ["firstname", "password", "email", "secondname", "login", "password2"]
        names_dict = {x: request.POST.get(x, "") for x in names_list}

        login = data["login"]

        if not login:
            errors.append('Введите логин')
        elif len(login) < 5:
            errors.append("Логин должен содержать не менее 5 символов")

        password = data['password']
        if not password:
            errors.append("Введите пароль")
        elif len(password) < 8:
            errors.append("Длина пароля должна быть не менее 8 символов")
        elif password != data["password2"]:
            errors.append("Пароли не совпадают")

        email = data['email']
        if not email:
            errors.append("Введите e-mail")

        firstname = data['firstname']
        if not firstname:
            errors.append("Введите Имя")
```

```

secondname = data['secondname']
if not secondname:
    errors.append("Введите фамилию")

if not errors:
    # updating data for registration
    new_form = form.save(commit=False)

    new_form.login = data.get('login')
    new_form.firstname = data.get('firstname')
    new_form.secondname = data.get('secondname')
    new_form.password = data.get('password')
    new_form.email = data.get('email')
    new_form.password = md5(new_form.password.encode('utf-8')).hexdigest()

    new_form.save()
    form.save_m2m()

    return HttpResponseRedirect("/computers/")
return render(request, 'my_app/register.html', {"errors": errors, 'names_dict': names_dict, "form": form})

```

## Регистрация

e-mail:

Имя:

Фамилия:

Логин:

Введите пароль:

Повторите пароль:

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

**Для логина используем метод `login_user`, `authenticate`**

```

def AuthorizeDjango(request):
    names_dict = {}
    errors = []
    form = AuthorizeForm(request.POST or None)
    if request.method == "POST" and form.is_valid():
        data = form.cleaned_data
        names_list = ["login", "password"]
        names_dict = {x: request.POST.get(x, "") for x in names_list}

        login = data["login"]
        if not login:
            errors.append('Введите логин')
        password = data['password']

        if not password:
            errors.append("Введите пароль")

        if not errors:
            try:
                userdata = CustomerModel.objects.get(login=data["login"])
                if userdata.password != md5(password.encode('utf-8')).hexdigest():
                    errors.append("Неправильное имя пользователя или пароль")

            except CustomerModel.DoesNotExist:
                errors.append("Неправильное имя пользователя или пароль")

            if not errors:
                user = authenticate(username=login, password=password)
                if user is not None:
                    if request.user.is_authenticated():
                        pass
                    login_user(request, user)
                    return HttpResponseRedirect("/computers/")

    print(list(errors))
    return render(request, 'my_app/authorize.html', {"errors": errors, 'names_dict': names_dict, "form": form})

```

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

См п.6

**8. Реализовать view для выхода из аккаунта.**

```

def LogoutClass(request):
    var = request.user.is_authenticated()
    if request.user.is_authenticated():
        logout(request)
    return redirect('AuthorizeDjango')

```

**9. Заменить проверку на авторизацию на декоратор login\_required  
Для ClassBasedView**

```

url(r'^computers/', login_required(ComputersClass.as_view()), name="computers"),
url(r'^logout/$', LogoutClass.as_view(), name="logout")

```

**10. Добавить superuser'а через команду manage.py**

См п. 12, 13

**11. Подключить django.contrib.admin и войти в панель администрирования.**

См п. 12, 13

**12. Зарегистрировать все свои модели в django.contrib.admin**

```

from django.contrib import admin
from my_app.models import OrderModel, ComputerModel, CustomerModel

class CustomerFilter(admin.ModelAdmin):
    pass

class CustomerAdmin(admin.ModelAdmin):
    exclude = ('password', )
    list_filter = ('firstname', 'secondname')
    search_fields = ['id']
    pass

admin.site.register(CustomerModel, CustomerAdmin)
admin.site.register(OrderModel)
admin.site.register(ComputerModel)

```

### 13. Для выбранной модели настроить страницу администрирования:

- Настроить вывод необходимых полей в списке
- Добавить фильтры
- Добавить поиск
- Добавить дополнительное поле в список

Django administration

WELCOME, ADMIN.

Site administration

AUTHENTICATION AND AUTHORIZATION

|        |       |        |
|--------|-------|--------|
| Groups | + Add | Change |
| Users  | + Add | Change |

MY\_APP

|                 |       |        |
|-----------------|-------|--------|
| Computer models | + Add | Change |
| Customer models | + Add | Change |
| Order models    | + Add | Change |

Recent actions

My actions

+ V.VASYA

User

✗ OrderModel object

Order model

'name': 'Asus n751jk,

'description': '17,3 1920x1080 ips display, intel core i7-4800 3,2GHz, 16Gb RAM, 256Gb SSD, wifi 2,4GHz bon+ac, bluetooth 4.0

### Добавленные пользователи

Home > Authentication and Authorization > Users

Select user to change

Q

Search

Action:

-----

Go

0 of 2 selected

| <input type="checkbox"/> | USERNAME | EMAIL ADDRESS         | FIRST NAME | LAST NAME | STAFF STATUS |
|--------------------------|----------|-----------------------|------------|-----------|--------------|
| <input type="checkbox"/> | admin    | vpupkin9208@gmail.com |            |           | ✓            |
| <input type="checkbox"/> | V.VASYA  |                       |            |           | ✗            |

Отображение модели Customer(Поиск, фильтры, и т.д.)

Select customer model to change

ADD CUSTOMER MODEL +

|                          |   |                     |
|--------------------------|---|---------------------|
| Q                        | <input type="text"/>  | Search              |
| Active                   | <input type="text"/>  | Go 0 of 11 selected |
| CUSTOMER MODEL           |   |                     |
| <input type="checkbox"/> | login', 'V VASYA', 'secondname' BACH/bEB, 'firstname' BACR, 'email' vasya.petrov@gmail.com, 'password' 97db1846570837fce6f652a400f1c26a |                     |
| <input type="checkbox"/> | login', 'Y VANOY', 'secondname' MRAHOB, 'firstname' MRAH, 'email' Yvan_Yvanov@gmail.com, 'password' 25d55ad283aa400af464c75d713c07ad    |                     |
| <input type="checkbox"/> | login', 'secondname', 'firstname', 'email' Yvan_Yvanov@gmail.com, 'password' 25d55ad283aa400af464c75d713c07ad                           |                     |
| <input type="checkbox"/> | login', 'secondname', 'firstname', 'email' Yvan_Yvanov@gmail.com, 'password' <md5 HASH object @ 0x03A004A0>                             |                     |
| <input type="checkbox"/> | login', 'secondname', 'firstname', 'email' Yvan_Yvanov@gmail.com, 'password' <md5 HASH object @ 0x036E9E18>                             |                     |
| <input type="checkbox"/> | login', 'secondname', 'firstname', 'email' Yvan_Yvanov@gmail.com, 'password':   |                     |
| <input type="checkbox"/> | login', 'secondname', 'firstname', 'email' Yvan_Yvanov@gmail.com, 'password':   |                     |

|               |  |
|---------------|--|
| FILTER        |  |
| By firstname  |  |
| All           |  |
| Ivan_Jandv    |  |
| qwertyname    |  |
| oogedgdgat    |  |
| BACR          |  |
| MRAH          |  |
| By secondname |  |
| All           |  |
| dugodfgdgy    |  |
| ie            |  |
| BACH/bEB      |  |
| unaware       |  |