CBV — Class Based View

Django предоставляет возможность обрабатывать запросы пользователя с помощью классов представления.
Все классы представления являются дочерними класса от класса **View**
Преимущества классов представления над функциями представления:
1. Разделение логики обработки методов запроса
2. Наследование, что позволяет удобно реализовывать примеси (**Mixins**)
Классы представления основанные на классе **View**, используются в определенных случаях, но они узконаправленные и не позволяют до конца реализовывать глубокий функционал на одной странице

Создание класса представления на основании класса View:

```python
class MyView(View):
    template_name = 'app/template.html'

    def get(self, request: HttpRequest):
        return render(request, self.template_name)
```

Определяя метод внутри класса совпадающий с методом **HTTP** запроса, отвечает за обработку только данного типа запроса
Все остальные классы представления, основанные на **View**, уже имеют определенные некоторые атрибуты и методы, которые мы можем расширять или полностью переопределять

Основные атрибуты дочерних классов View:

model — Указания ссылки на класс модели при работе с такими классами представления как **ListView**, **DetailView**
template_name — указания пути к шаблону, который будет использоваться
context_object_name — имя, по которому будут доступны данные, после запроса в БД, на шаблоне
form_class - указание ссылки на класс формы
success_url — имя ссылки, на которую будет перенаправлен пользователь после успешной записи данных в БД
get_queryset — метод, в котором производится выборка и возврат данных
get_context_data — метод, для определения данных доступных на шаблоне (примеси)

Регистрация/Авторизация пользователя

Django предоставляет регистрацию и авторизацию пользователя по умолчанию

Для реализации регистрации необходимо определить некоторые классы представления и ссылки:

```python
urlpatterns = [
    path('signup/', SingUpView.as_view(), name='signup'),
    path('login/', LoginView.as_view(), name='login'),
    path('accounts/profile/', AccountView.as_view(), name='account'),
    path('logout', LogoutView.as_view(), name='logout'),
    path('', index, name='index'),
]
```

```python
class RegisterView(UserCreationForm):
    class Meta:
        model = User
        fields = ('username', 'password1', 'password2', 'email')


class SingUpView(CreateView):
    form_class = RegisterView
    success_url = reverse_lazy('login')
    template_name = 'registration/signup.html'
```

В классе **RegisterView** переопределяем предоставляемую **Django** форму для регистарции, так как по умолчанию, для регистрации пользователя, **Django** запрашивает только **username** и **пароль**, если для нашей регистрации этого не достаточно, необходимо переопределить данную форму чтобы расширить необходимые поля

Класс **SignUpView** является классом представления на основании класса **CreateView**

```python
class AccountView(LoginRequiredMixin, View):
    template_name = 'app/account.html'
    login_url = '/login/'

    def get(self, request):
        print(request.user)
        return render(request, self.template_name)
```

Для ограничения доступа к конкретной странице только зарегистрированным пользователям, определяется класс представления на основании любого класса представления, с примесью **LoginRequiredMixin,** данный класс, будет определять авторизован ли пользователь, и в случае если он не авторизован, перенаправлять на ссылку указанную в атрибуте **login_url**

Имя авторизованного пользователя доступно в атрибуте **user** аргумента **request**

Для более полноценной логики работы системы регистрации/авторизации а именно для

```
path('password-change/', PasswordChangeView.as_view(), name='password_change'),
path('password-change/done/', PasswordChangeDoneView.as_view(), name='password_change_done'),
path('password-reset/', PasswordResetView.as_view(), name='password_reset'),
path('password-reset/done/', PasswordResetDoneView.as_view(), name='password_reset_done'),
path('reset/<uidb64>/<token>/', PasswordResetConfirmView.as_view(), name='password_reset_confirm'),
path('reset/done/', PasswordResetCompleteView.as_view(), name='password_reset_complete'),
```

возможности сбросить/сменить пароль, необходимо определить следующие пути, рекомендуется переопределить классы представления по умолчанию, для изменения шаблонов по умолчанию.