Django: Class Based Views | Я.Шпора

Class Based Views (CBV) наследуются от встроенных классов-дженериков.

- Для отображения списка объектов пользовательский CBV надо унаследовать от встроенного класса `ListView`.
- Для отображения отдельного объекта пользовательский CBV надо унаследовать от встроенного класса `DetailView`.
- Для управления формой, предназначенной для создания объектов, пользовательский CBV надо унаследовать от встроенного класса `CreateView`.

Список встроенных CBV в документации.

Имена пользовательским CBV традиционно дают по схеме `Имя_моделиИмя_CBV`, например — `BirthdayListView` или `BirthdayCreateView`.

Чтобы назначить view-класс обработчиком для запроса, отправленного на определённый URL, view-класс нужно импортировать в файл *urls.py* и в функции `path()` вызвать метод `.as_view()` этого класса:

```
# views.py
# Описываем view-класс:
class BirthdayListView(ListView):
    ...
```

Связываем шаблон адреса запроса с классом-обработчиком:

```
# urls.py
# Импортируем view-классы в urls.py
from app_name import views

urlpatterns = [
        ...,
    # Связываем относительный путь list/ с классом - обработчиком запроса
```

```
    path('list/', views.BirthdayListView.as_view(), name='list'),
]
```

Дженерик ListView

Используется для представления списка объектов.

```
...
from django.views.generic import ListView
...

# Наследуем класс от встроенного ListView:
class BirthdayListView(ListView):
    model = Birthday  # Модель, с которой работает CBV.
    ordering = 'id'   # Сортировка объектов по умолчанию.
    paginate_by = 10  # Настройки пагинации.
```

Дженерик DetailView

Используется для представления отдельного объекта модели.

```
# birthday/views.py
...
from django.views.generic import DetailView
...


class BirthdayDetailView(DetailView):
    model = Birthday
```

Чтобы дополнить или переопределить словарь контекста — применяют метод `get_context_data()`.

```
# birthday/views.py
class BirthdayDetailView(DetailView):
    model = Birthday

    def get_context_data(self, **kwargs):
        # Получаем словарь контекста:
```

```
        context = super().get_context_data(**kwargs)
        # Добавляем в словарь новый ключ:
        context['birthday_countdown'] = calculate_birthday_countdown(
            # Дату рождения берём из объекта в словаре context:
            self.object.birthday
        )
        # Возвращаем словарь контекста.
        return context
```

Дженерик CreateView

Используется для управления формой, предназначенной для создания объектов заданной модели.

```
...
from django.views.generic import CreateView
from django.urls import reverse_lazy
...


class BirthdayCreateView(CreateView):
    model = Birthday  # Модель, с которой работает CBV.
    # Этот класс сам может создать форму на основе модели.
    # Указываем поля, которые должны быть в форме:
    fields = '__all__'
    # Явным образом указываем шаблон:
    template_name = 'birthday/birthday.html'
    # Указываем namespace:name страницы, куда будет перенаправлен пользов
    # после успешного создания объекта:
    success_url = reverse_lazy('birthday:list')
```

Базовое представление TemplateView

Используется для отображения статичных страниц.

```
# Импортируем класс TemplateView, чтобы унаследоваться от него.
from django.views.generic import TemplateView
```

```python
class HomePage(TemplateView):
    # В атрибуте template_name обязательно указывается имя шаблона,
    # на основе которого будет создана возвращаемая страница.
    template_name = 'pages/index.html'
```