django

Views

Class-based views:

from django.views.generic **import** <view-name>

◆ **DetailView:** displays a certain object
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-display/#detailview

```
from django.views.generic import DetailView

from articles.models import Article

class ArticleDetailView(DetailView):

    model = Article
```

◆ **ListView:** displays a list of objects
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-display/#listview

```
from django.views.generic import ListView

from articles.models import Article

class ArticleListView(ListView):

    model = Article
```

◆ **CreateView:** displays a form and create an objects by its values
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-editing/#createview

```
from django.views.generic.edit import CreateView

from myapp.models import Author

class AuthorCreate(CreateView):

    model = Author

    fields = ['name']
```

◆ **FormView:** displays a form and redisplays if has any errors
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-editing/#formview

```
from myapp.forms import ContactForm
from django.views.generic import FormView
class ContactView(FormView):
    template_name = 'contact.html'
    form_class = ContactForm
    success_url = '/thanks/'

    def form_valid(self, form):
        # This method is called when valid form data has been POSTed.
        # It should return an HttpResponse.
        form.send_email()
        return super().form_valid(form)
```

◆ **UpdateView:** displays a form for editing an existing object
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-editing/#updateview

```
from django.views.generic.edit import UpdateView
from myapp.models import Author
class AuthorUpdate(UpdateView):
    model = Author
    fields = ['name']
    template_name_suffix = '_update_form'
```

◆ **DeleteView:** displays a confirmation page and deletes an existing object
https://docs.djangoproject.com/en/3.0/ref/class-based-views/generic-editing/#deleteview

```
from django.views.generic.edit import DeleteView
from myapp.models import Author
class AuthorDelete(DeleteView):
    model = Author
    success_url = 'authors'
```

Class Methods

- **def** get_context_data(self, **kwargs)
  Adds extra context data to the view

```
    def get_context_data(self, **kwargs):
        # Call the base implementation first to get a context
        context = super().get_context_data(**kwargs)
        # Add in a QuerySet of all the books
        context['book_list'] = Book.objects.all()
        return context
```

- **def** get_queryset(self)
  In a ListView, changes objets will be displayed

```
def get_queryset(self):
     return Subject.objects.filter(grade=self.request.user.grade)
```

GET and POST request handling

Class-Based Views:

def post(self, request, *args, **kwargs)
Handle a post request made on the view

```
def post(self, request, *args, **kwargs):
form = CreateSubjectForm(request.POST, request.FILES)
if form.is_valid():
      # form data handling
return redirect('subjects')
```

def get(self, request, *args, **kwargs)
Handle a get request made on the view

```
def get(self, request, *args, **kwargs):
      # data handling
      return render(request, self.template_name, {'form': form})
```

Function-Based Views:

GET and POST:

```
def get_name(request):
    if request.method == 'POST':
        form = NameForm(request.POST)
        if form.is_valid():
            # process the data in form.cleaned_data as required
            return HttpResponseRedirect('/thanks/')

    # if a GET (or any other method) we'll create a blank form
    else: # request.method == 'GET'
        form = NameForm()

    return render(request, 'name.html', {'form': form})
```