#Views and Templates

#view

- “type” of web page in Django application
- generally serves a specific function and has a specific template
- in Django, web pages and other content are delivered by views
- each view is represented by a Python function (or method, in the case of class-based views)
- it choose a view by examining the URL that’s requested

view is responsible for doing one of two things:

- returning an HttpResponse object containing the content for the requested page
- or raising an exception such as Http404 when not able to send response for received request

view can read records from a database, or not
can use a template system such as Django’s – or a third-party Python template system – or not.
can generate a PDF file, output XML, create a ZIP file on the fly, anything you want, using whatever Python libraries you want

Note: All Django wants is HttpResponse or an exception.

#template
page’s design is hard-coded in the view
to change the way page look - need to edit python code
in this case, Django’s template system is used - to separate the design from Python by creating a template that the view can use

create templates directory in apps directory - usually django will look in this position

in settings file, “TEMPLATES” section describes how Django will load and render templates
by default, APP_DIRS option is True.
so, DjangoTemplates looks for a “templates” subdirectory in each of the INSTALLED_APPS.

in view funcion, code loads template and passes context

context - dictionary mapping template variable names to Python objects

render() function

- takes the request object as its first argument
- template name as second argument
- dictionary as its optional third argument (context), returns an HttpResponse object of the given template rendered with the given context.

#Raising a 404 error
view raises the Http404 exception if requeste doesn't exist(mostly some object in model)

#get_object_or_404()
- common idiom to use get() and raise Http404 if the object doesn't exist.
- this function takes a Django model as its first argument and an arbitrary number of keyword arguments, which it passes to the get() function of the model's manager. It raises Http404 if the object doesn't exist

#get_list_or_404() function
- works just as get_object_or_404()
- except using filter() instead of get()
- raises Http404 if the list is empty.

#django - template system

django - template system uses dot-lookup syntax to access variable attributes.

Example html file,

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
   <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

{{ question.question_text }}
→ django 1st look for dictionary lookup on object, in this case it fails
→ so it next it tries for attribute lookup which works
→ if attribute lookup had failed, it wil try to check list-index lookup

{% for %} tag
→ using python code in html file to iterate through object

More info - https://docs.djangoproject.com/en/4.1/topics/templates/

#Removing hardcoded URLs in templates

While referring url as below,

```
<a href="/polls/{{ question.id }}/">{{ question.question_text }}</a>
```

it becomes challenging to change urls on project with lot of templates.

In this case, url can be changed as below to solve,

```
<a href="{% url 'detail' question.id %}">{{ question.question_text }}</a>
```

and in urls.py file,

```
# the 'name' value as called by the {% url %} template tag
path('<int:question_id>/', views.detail, name='detail')
```

#Namespacing URL names (add namespaces to URLconf)

- in most case, django project have more than one app
- used to differentiate which apps view should refer while calling url in template tag {% url %}

In urls.py file, add “app_name” to set application namespace

#Example

#urls.py

```
app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
   ]
```

#html file - change url as follows(to point at the namespaced detail view)

```
<a href="{% url 'detail' question.id %}">{{ question.question_text }}</a>
```

change to,

```
<a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a>
```

#ref
https://docs.djangoproject.com/en/4.1/intro/tutorial03/