# **CBVs (Class Based Views)**

CBV offers great functionality and for most experienced user of Django it’s their default choice for creating views.

# 

# **Hello world with CBVs**

Use the simplest available django view class:

*from django.views.generic import View*

We have to slightly change the way we call a class based view in the **urls.py** file of the **project**.

We need to add in a *.as\_view()* call off the class, this is an inherited method from the View.

Create a new project and app (it’s example) and do usual basic stuff in settings. Create templates like **base.html**, **index.html**.

\*In this example we just show how to convert a simple hello world function view into a class based view to get the idea of how they relate to each other.

In **base.html** add:

*<body>*

*{% block body\_block %}*

*{% endblock %}*

*</body>*

*</More footer html>*

In **index.html** add:

*{% extends “app/base.html” %}*

*{% block body\_block %}*

*<h1>Welcome</h1>*

*{% endblock %}*

In **views.py** add:

*from django.views.generic import View*

*from django.http import HttpResponse*

*class* ***CBView****(View):*

*def get(self, request)*

*return HttpResponse(“CLASS BASED VIEWS ARE COOL!”)*

In **urls.py** in urlpatterns add:

*path(‘index/’, views.****CBView****.as\_view())*

# 

# **Template View with CBVs**

<https://docs.djangoproject.com/en/2.2/topics/class-based-views/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Let’s firstly compare Function Based View and Class Based Template View:

**Function Based View**:

*def index(request):*

*return render(request, ‘index.html’)*

**Class Based Template View**:

*class IndexView(TemplateView):*

*template\_name = ‘index.html’*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In **views.py** add:

*from django.views.generic import View, TemplateView*

*class IndexView(TemplateView):*

*template\_name = ‘app/index.html’*

In **urls.py** of app add:

*path(‘index/’, views.IndexView.as\_view())*

*runserver*

Example of injecting the content into the template:

In **index.html** add:

*{% block block\_body %}*

*<h2>Test injection: {{* ***injectme*** *}}</h2>*

*{% endblock %}*

In **views.py** add:

*class IndexView(TemplateView):*

*template\_name = ‘app/index.html*

*def get\_context\_data(self, \*\*kwargs):*

*context = super().get\_context\_data(\*\*kwargs)*

*context[****‘injectme’****] = “BASIC INJECTION’*

*return context*

*runserver*

# **Detail View and List View**

<https://docs.djangoproject.com/en/2.2/ref/class-based-views/generic-display/>

When we have models, we want to either list the records from the model, or show details of a single record. We already know one way to do it: call *MyModel.objects.all()*

However these sorts of operations are very common!

So common that Django has some generic view classes we can inherit to very quickly display information from the model. CBV helps with it.

Here we gonna have a **template** folder inside of **app** folder (not project folder as before).

In **models.py** add:

*class School(models.Model):*

*name = models.CharField(max\_length=256)*

*principal = models.CharField(max\_length=256)*

*location= models.CharField(max\_length=256)*

*def \_\_str\_\_(self):*

*return self.name*

*class Students(models.Model):*

*name = models.CharField(max\_length=256)*

*age = models.PositiveIntegerField()*

*school = models.ForeignKey(School, related\_name=’students’)*

*def \_\_str\_\_(self):*

*return self.name*

In **admin.py** add:

*from app.models import School, Student*

*admin.site.register(School)*

*admin.site.register(Student)*

Make migrations, create a superuser.

In admin page create two school object (create a name, principal, location for it), create four students and define them in schools.

In **app** folder create a folder **templates** and inside it create a folder with app name.

In **app/templates/app** add **app\_base.html**, **school\_detail.html** and **school\_list.html**.

In **views.py** add:

*from django.views.generic import View, TemplateView, ListView, DetailView*

*from . import models*

*class SchoolListView(ListView):*

*model = models.School*

*# school\_list*

*class SchoolDetailView(DetailView):*

*model = models.School*

*template\_name = ‘app/school\_detail.html’*

In **project/templates/app** (in project and app folders):

In **base.html** add:

*<nav>*

*<ul>*

*<li><a href=”{% url ‘app:list’ %}”>Schools<a/></li>*

*<li><a href=”{% url ‘admin:index’ %}”>Admin<a/></li>*

*<li><a href=”{% url ‘app:list’ %}”><a/></li>*

*</ul>*

*</nav>*

Copy code from **project/templates/app/base.html** into **project/app/templates/app/app\_base.html**.

In **index.html** add:

*<h1>Home Page!</h1>*

In **project/app/templates/app** in **school\_list.html** add:

*{% extends ‘app/app\_base.html’ %}*

*{% block body\_block %}*

*<ol>*

*{% for school in* ***school\_list*** *%}*

*<h2><li>{{ school.name }}</li></h2>*

*{% endfor %}*

*</ol>*

*{% endblock %}*

*<!-- ListView in* ***views.py*** *takes the name of model (School in our case), lower case the name and adds \_list:*

*School.lower()+\_list =* ***school\_list*** *-->*

*<!--DetailView in* ***views.py*** *take the name of model (School as well in our case) and lower case the name so we get ‘school’ -->*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

! IF WE WANT TO HAVE ***schools*** INSTEAD OF ***school\_list*** do:

In **views.py** edit:

*class IndexView(TemplateView):*

*template\_name = ‘index.html’*

*class SchoolListView(ListView):*

***context\_object\_name = ‘schools’***

*model = models.School*

*# school\_list*

*class SchoolDetailView(DetailView):*

*context\_object\_name =* ***‘school\_detail’***

*model = models.School*

*template\_name = ‘app/school\_detail.html’*

In **school\_detail.html** add:

*{% extends ‘app/app\_base.html’ %}*

*{% block body\_block %}*

*<h1>Welcome to the School Detail Page!</h1>*

*<h2>School details:</h2>*

*<p>Name: {{ school\_detail.name }}</p>*

*<p>Principal:{ {* ***school\_detail****.principal }}</p>*

*<p>Location: {{* ***school\_detail****.location }}</p>*

*<h3>Students:</h3>*

*{% for student in* ***school\_detail****.students.all %} <!--students - related name in*

*model-->*

*<p>{{student.name}} who is {{student.age}} years old.</p>*

*{% endfor %}*

*<!-- In school\_detail.****students****.all students comes from the* ***related\_name*** *defined in a ForeignKey in a model Student -->*

*{% endblock %}*

In **urls.py** of **project** add:

*# in urlpatterns*

*path(‘app/’, include(‘app.urls’, namespace=’app’))*

In In **urls.py** of **app** add:

*from django.urls import re\_path #to use regular expressions*

*app\_name = ‘app’*

*# in urlpatterns*

*path(‘’, views.SchoolListView.as\_view(), name=’list’),*

*re\_path(r‘^(?P<****pk****>[-\w]+)/$’, views.SchoolDetailView.as\_view(), name=’detail’)*

In the example for no student a primary key was set. Instead of primary key django is using **id** that increments with every new student, and therefore every id is unique and corresponds to one and only appropriate student.

If we want to get to the detail of the school after clicking on the name of the school, we should edit **school\_list.html**:

*<h2><li>{{ school.name }}</li></h2> →*

*<h2><li><a href=”{{****school.id****}}”>{{ school.name }}</a></li></h2>*

When someone clicks on *{{ school.name }}* ref returns a single number - primary key (**pk**),

*r‘^(?P<****pk****>[-\w]+)/$’* means:

Grab the app extension of the domain name slash and whatever this number happens to be for the primary key, and take that in as the school details view.

In this example **pk** (primary key) is an identifier of the object with the name given in views.py in SchoolDetailView where context\_object\_name = 'school\_detail'.

Since object of a class School has no defined primary key, pk is a number of the record in the database.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

!!!

If we want to use a special index page for the app, name of this **index page** in a folder **project/app/templates/app/** should be different from the **index page** in the folder **project/templates/**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **CRUD Views**

<https://rayed.com/posts/2018/05/django-crud-create-retrieve-update-delete/>

CRUD (Create Retrieve Update Delete) is inherent to almost every website.

Pretty much any web site users has the ability for a user or an administrator to create content, retrieve prior content, update content or delete that content.

We gonna learn how to create, delete, or update information from the web app.

Start by exploring how to use the *CreateView* class.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

!!! While we are using *CreateView*class we will purposefully induce a few errors to clarify where certain variable names are coming from.

Mistakes are gonna be red colored.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Let’s create a new school using *CreateView*.

In **views.py** add:

*from django.views.generic import … CreateView, UpdateView, DeleteView*

*class SchoolCreateView(CreateView):*

*model = models.School*

*fields = (‘name’, ‘principal’, ‘location’)*

In **urls.py** edit:

*path(‘’, views.SchoolListView.as\_view(), name=’list’),*

*re\_path(r‘^(?P<****pk****>\d+)/$’, views.SchoolDetailView.as\_view(), name=’detail’),*

*re\_path(r‘^create/$’, views.SchoolCreateView.as\_view(), name=’create’)*

In **templates** in app folder create **school\_form.html** and add there:

*{% extends ‘app/ app\_base.html’ %}*

*{% block body\_block %}*

*<h1>*

*{% if not form.instance.pk %}*

*Create School*

*{% else %}*

*Update School*

*{% endif %}*

*</h1>*

*<form method=”POST”>*

*{% csrf\_token %}*

*{{ form.as\_p }}*

*<input type=”submit” value=”Submit”>*

*</form>*

*{% endblock %}*

<https://stackoverflow.com/questions/18265023/self-instance-in-django-modelform>

Without **school\_form.py** we will get an error. Django looks for this page if we want to create a new record (django looks for [name of model]+\_form). This html form is used in executing as\_view() of *UpdateView* and *CreateView* classes.

In **models.py** edit:

*from django.urls import reverse*

*class School(models.Model):*

*# see the previous part where this class is defined*

*…*

*def get\_absolute\_url(self):*

*return reverse(‘app:detail’, kwargs={‘pk’: self.pk})*

In **views.py** add:

*class SchoolUpdateView(UpdateView):*

*fields = (‘name’, ‘principal’)*

*model = models.School*

In **urls.py** add:

*re\_path(r‘^update/(?P<****pk****>\d+)/$’, views.SchoolUpdateView.as\_view(), name=’update’),*

In **school\_details.html** add:

*<p><a href=”{% url ‘app:update’ pk=school\_detail.pk %}”>Update</a></p>*

Deleting views.

In **views.py** add:

*from django import reverse\_lazy*

*class SchoolDeleteView(DeleteView):*

*model = models.School*

*success\_url = reverse\_lazy(“app:list”)*

<https://docs.djangoproject.com/en/2.2/ref/urlresolvers/>

In **urls.py** add:

*re\_path(r‘^delete/(?P<****pk****>\d+)/$’, views.SchoolDeleteView.as\_view(), name=’delete’),*

In **app/templates/app** add **school\_confirm\_delete.html**:

*{% extends ‘app/app\_base.html’ %}*

*{% block body\_block %}*

*<h1> Delete {{school.name}}?</h1>*

*<form method=”POST”>*

*{% csrf\_token %}*

*<input type=”submit” value=”Submit”>*

*<a href=”{% url ‘app:detail’ pk=school.pk %}”>Cancel</a>*

*</form>*

*{% endblock %}*

Why **pk**: <https://stackoverflow.com/questions/2165865/django-queries-id-vs-pk>