Class Based Views:

Create subclass of view (django.views.generic)

CRUD Operations using class based views:

django.views.generic

View

ListView

DetailView

CreateView

UpdateView

DeleteView

Reading data from database:

- Create a model Teacher

models.py(admissions app)

```
from django.db import models
class Teacher(models.Model):
    name = models.CharField(max_length=100)
    exp = models.IntegerField()
    subject =models.CharField(max_length=100)
    contact = models.CharField(max_length=100)
```

Go to python cmd prompt

Go to project location

python manage.py makemigrations

python manage.py migrate

Go to mysql cmd prompt

Set the path for mysql

path c:\xampp\mysql\bin

mysql –u root –p

show databases;

use schoolapp;

show tables;

desc admissions_teacher;

insert into admissions_teacher values(1, 'abc',10,'cse','8967564534');

insert into admissions_teacher values(2, 'def',12,'ece','9087654343');

Django provide a ListView to read data from database

To perform read operation we have to use ListView steps are as follows:

1. Create a class based view that is inherited to ListView.

2. Provide value to the model attribute (mandatory).

3. Default template name is modelname_list.html.You may pass your own template name by passing value to the template name (template name attributes).

4. Default Context object name (Object that receives all the objects from ORM) is modelname_list.So; you can set your own context_object_name by passing the value to this attribute.

5. Create a template with name modelname_list.html and print objects attribute from the list received (modelname_list).

6. Configure the url(classname.as_view())

views.py

```
from admissions.models import Teacher

from django.views.generic import ListView

class Teacherread(ListView):

    model=Teacher  #note:here data stored in teacher_list

                   #teacher_list=teacher.objects.all()

                   #context_object_name= 'result'

                   #default html page is teacher_list.html

                   #templatename='list.html'
```

- Go to templates folder and select admissions folder and create new file teacher_list.html

teacher_list.html:

```
<html>
<head>
<title>Teacher Page</title>
</head>
<body>
<table border="1">
<tr>
<th>Id</th>
<th>Name</th>
<th>Experience</th>
<th>Subject</th>
<th>Contact</th>
</tr>
{% for s in teacher_list %}
<tr>
<td>{{s.id}}</td>
<td>{{s.name}}</td>
<td>{{s.exp}}</td>
<td>{{s.subject}}</td>
<td>{{s.contact}}</td>
</tr>
```

{% endfor %}

</body>

</html>

urls.py(admissions app)

```
from admissions.views import Teacherread
urlpatterns = [
   path ('teacherlist/',Teacherread.as_view()),
]
```

To run the server application

python manage.py runserver

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

http://127.0.0.1:8000/ad/teacherlist

Retrieving a Single row using DetailView:

To perform this we need to use DetailView steps are as follows:

1. Create a class based view that inherits from DetailedView.
2. Provide value to model attribute(mandatory).
3. Default Template name is modelname_detail.html.You may pass your own template name by passing value to the template name attribute.
4. Default context object name(object that received all the object from ORM) is model name.You can set your own context_object_name by passing the values to this attributes.
5. Create a template with name modelname_detail.html and print the object attributes from the list received(modelname_list).

6. Configure the url
   path('teacherdetail/<int:pk>/',classname.as_view())

views.py

```
from django.views.generic import DetailView

class Teacherdetail(DetailView):

    model=Teacher  #Teacherdetail data stores in that name

                   #default html page is teacher_detail.html

                   #templatename='detail.html'
```

- Go to templates folder and select admissions folder and create new file teacher_detail.html

teacher_detail.html:

```
<html>

<head>

<title>Teacher Detail Page</title>

</head>

<body>

<h1>Name:{{teacher.name}}</h1>

<h1>Experience:{{teacher.exp}}</h1>

<h1>Subject:{{teacher.subject}}</h1>

<h1>Contact:{{teacher.contact}}</h1>
```

```
</body>

</html>
```

urls.py(admissions app)

```
from admissions.views import Teacherdetail
urlpatterns = [
   path ('teacherdetail/<int:pk>/',Teacherdetail.as_view()),
]
```

To run the server application

```
python manage.py runserver
```

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

```
http://127.0.0.1:8000/ad/teacherdetail
```

Create Operation Using Class Based Views:

To perform this operation we need to use CreateView steps are as follows:

1. Create a class based view that inherits from CreateView.
2. Provide value to the model attribute(mandatory).
3. Provide values to fields(fields=(fieldslist separated by comma))
4. Default template name is modelname_form.html.You may pass your own template name by passing value to your template name attributes.
5. Create a method get_absolute_url() in model class.

```
from django.urls import reverse
def get_absolute_url(self):
    return reverse('urlname',kwargs='pk':self.pk})
```

6. Create a template with name modelname_from.html
7. Configure the url classname.as_view().

views.py

```
from django.views.generic import CreateView

class Insertteacher(CreateView):

    model=Teacher #form=TeacherModelForm

    fields=( 'name','exp','subject','contact') #teacher_form.html
```

- Go to templates folder and select admissions folder and create new file teacher_form.html

teacher_form.html:

```
<html>

<head>

<title>Insert Teacher</title>

</head>

<body>

<form method= "POST">

<table>

{{form.as_table}}
```

```
</table>
<input type= "submit" value="Add Teacher">
{% csrf_token %}
</form>
</body>
</html>
```

urls.py(admissions app)

```
from admissions.views import Insertteacher

urlpatterns = [
    path ('insertteacher/',Insertteacher.as_view()),

]
```

To run the server application

python manage.py runserver

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

http://127.0.0.1:8000/ad/insertteacher

models.py(admissions app)

```
from django.urls  import reverse
from django.db import models
```

```
class Teacher(models.Model):
        name = models.CharField(max_length=100)
        exp = models.IntegerField()
        subject models.CharField(max_length=100)
        contact = models.CharField(max_length=100)

        def get_absolute_url(self):
                return reverse('listteacher')
```

urls.py(admissions app)

```
from admissions.views import Teacherread
from admissions.views import Insertteacher

urlpatterns = [
   path ('teacherlist/',Teacherread.as_view(),name='listteacher'),
   path ('insertteacher/',InsertTeacher.as_view()),

]
```

To run the server application

```
python manage.py runserver
```

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

```
http://127.0.0.1:8000/ad/teacherlist
```

Note:

1. We can return to a page (based on url by using reverse())

2. We can send the arguments to the url by using args/kwargs in reverse function.

3. Generally listteacher url doesn’t require any arguments so it returns error after adding the record to the database.

4. So, we have to send the arguments to this urls only when it is required.Here args is optional.

models.py(admissions):

```
def get_absolute_url(self):
        return reverse('listteacher',kwargs={'pk':self.pk})
```

urls.py(admissions):

```
from admissions.views import Teacherdetail
urlpatterns = [
    path
('teacherdetail.<int:pk>/',Teacherdetail.as_view(),name='listteacher'),
]
```

Update Operation Using Class Based Views:

To perform this operation we need to use UpdateView steps are as follows:

1. Create a class based view that inherits from UpdateView.
2. Provide value to the model attribute (mandatory).
3. Provide value to fields(fields=(fields list separated by comma))
4. Default template name modelname_form.html.You may pass your own template name by passing value to the template_name attribute.
5. Create a method get_absolute_url() in model class.

```
from django.urls import reverse
def get_absolute_url(self):
    return reverse('urlname',kwargs='pk':self.pk})
```

6. Create a template with name modelname_from.html and print the objects attributes from the list received(modelname_list)
7. Configure the url classname.as_view().

views.py

```
from django.views.generic import UpdateView
class Updateteacher(UpdateView):
    model=Teacher
    fields=( 'name','contact')
```

- Go to templates folder and select admissions folder and create new file teacher_form.html

teacher_form.html:

```
<html>
```

```
<head>
<title>UpdateTeacher</title>
</head>
<body>
<form method= "POST">
<table>
{{form.as_table}}
</table>
<input type= "submit" value="Update Teacher">
{% csrf_token %}
</form>
</body>
</html>
```

urls.py(admissions app)

```
from admissions.views import Updateacher
urlpatterns = [
   path ('updateteacher/<int:pk>/',Updateteacher.as_view()),
]
```

- Go to templates folder and select admissions folder and create new file teacher_list.html

teacher_list.html:

```
<html>
<head>
<title>Teacher Page</title>
</head>
<body>
<table border="1">
<tr>
<th>Id</th>
<th>Name</th>
<th>Experience</th>
<th>Subject</th>
<th>Contact</th>
<th>Update Action</th>
</tr>
{% for s in teacher_list %}
<tr>
<td>{{s.id}}</td>
<td>{{s.name}}</td>
<td>{{s.exp}}</td>
<td>{{s.subject}}</td>
<td>{{s.contact}}</td>
```

<td>Update</td>

</tr>

{% endfor %}

</body>

</html>

To run the server application

python manage.py runserver

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

http://127.0.0.1:8000/ad/updateteacher/2

Delete Operation Using Class Based Views:

To perform this operation we need to use DeleteView steps are as follows:

1. Create a class based view that inherits from DeleteView.
2. Provide value to the model attribute (mandatory).
3. Provide values to success_url=reverse_lazy('urlname')
4. Create modelname_confirm_delete.html to which django forward as and waits for our confirmation.
5. Configure the url(classname.as_view())

views.py

```
from django.views.generic import DeleteView

class Deleteteacher(DeleteView):

        model=Teacher
```

urls.py(admissions app)

```
from admissions.views import Deleteteacher
urlpatterns = [
    path ('deleteteacher/<int:pk>/',Deleteteacher.as_view()),
]
```

- Go to templates folder and select admissions folder and create new file teacher_list.html

teacher_list.html:

```
<html>

<head>

<title>Teacher Page</title>

</head>

<body>

<table border="1">

<tr>

<th>Id</th>

<th>Name</th>

<th>Experience</th>
```

```
<th>Subject</th>
<th>Contact</th>
<th>Update Action</th>
<th>Delete Action</th>
</tr>
{% for s in teacher_list %}
<tr>
<td>{{s.id}}</td>
<td>{{s.name}}</td>
<td>{{s.exp}}</td>
<td>{{s.subject}}</td>
<td>{{s.contact}}</td>
<td><a href= "/ad/updateteacher/{{s.id}}">Update</a></td>
<td><a href= "/ad/deleteteacher/{{s.id}}">Delete</a></td>
</tr>
{% endfor %}
</body>
</html>
```

- Go to templates folder and select admissions folder and create new file teacher_confirm_delete.html

teacher_confirm_delete.html:

```
<html>
<head>
<title>Delete Teacher Page</title>
</head>
<body>
<form method= "POST">
<button type="submit" value="confirm">Confirm</button>

{% csrf_token %}

</form>

<a href="/ad/teacherlist"><button type="button"
value="cancel">Cancel</button></a>

</body>

</html>
```

views.py

```
from django.views.generic import DeleteView

from django.urls import reverse_lazy

class Deleteteacher(DeleteView):

        model=Teacher

        success_url=reverse_lazy('listteacher')
```

To run the server application

python manage.py runserver

- Go to browser window and type this url http://127.0.0.1:8000
- It will shows the view action and choose the view particular action

http://127.0.0.1:8000/ad/deleteteacher/2