

Topic: 10. CRUD using Generics

Speaker: Personal / Notebook: API Development using Django Framework



Generics according to [Django Documentation](#):

Django's generic views... were developed as a shortcut for common usage patterns... They take certain common idioms and patterns found in view development and abstract them so that you can quickly write common views of data without having to repeat yourself.

GenericAPIView

This class extends the REST framework's `APIView` class, adding commonly required behavior for standard list and detail views.

Each of the concrete generic views provided is built by combining `GenericAPIView`, with one or more mixin classes.

To read more about Generics, use this link to [Medium Digest](#)

The API views are:

ListAPIView	RetrieveAPIView
CreateAPIView	DestroyAPIView
UpdateAPIView	ListAPICreateView
RetrieveUpdateAPIView	RetrieveUpdateDestroyAPIView

1. To apply CRUD on Employee model using Generics. Comment out the code using Mixins and add:

The image shows a code editor with a dark theme. On the left, the Explorer pane shows a project structure with folders 'api' and 'django_rest_main', and various Python files. The file 'views.py' in the 'api' folder is selected and highlighted with a red box. The main editor area shows the code for 'views.py', with a red box highlighting the following code block:

```
136
137     # updates the specific record
138     def put(self, request, pk):
139         return self.update(request, pk)
140
141     # deletes/destroys a specific record
142     def delete(self, request, pk):
143         return self.destroy(request, pk)
144     """
145
146     # uses Generics for CRUD
147     # -----"
148
149     # lists all employees using Generics
150     class Employees(generics.ListAPIView):
151         queryset = Employee.objects.all()
152         serializer_class = EmployeeSerializer
153
154
155     class EmployeeDetail(generics.RetrieveAPIView):
156         pass
157
```

Running the URL path for listing the employees:

Employees

Employees

OPTIONS GET

GET /api/v1/employees/

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 1,
    "emp_id": "EMP001",
    "emp_name": "Rosilie Lim",
    "designation": "Software Developer"
  },
  {
    "id": 3,
    "emp_id": "EMP003",
    "emp_name": "Russell Lim",
    "designation": "Security"
  },
  {
    "id": 6,
    "emp_id": "EMP004",
    "emp_name": "Arnel Zethus",
    "designation": "AI Engineer"
  }
]
```

To allow a FORM for creating an employee, we add a `generics.CreateAPIView` to inherit.

```
EXPLORER
  DJANGOREST_APIPROJECT
    api
      __pycache__
      migrations
      __init__.py
      admin.py
      apps.py
      models.py
      serializers.py
      tests.py
      urls.py
      views.py
    django_rest_main
      __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py

api > views.py > ...
138 # updates the specific record
139 def put(self, request, pk):
140     return self.update(request, pk)
141
142 # deletes/destroys a specific record
143 def delete(self, request, pk):
144     return self.destroy(request, pk)
145     """
146
147 # uses Generics for CRUD
148 # -----"
149
150 # lists all employees using Generics
151 class Employees(generics.ListAPIView, generics.CreateAPIView):
152     queryset = Employee.objects.all()
153     serializer_class = EmployeeSerializer
154
155 # lists specific employee using Generics
156 class EmployeeDetail(generics.RetrieveAPIView):
157     pass
158
```

Employees

Employees

OPTIONS

GET

POST /api/v1/employees/

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "id": 7,
  "emp_id": "EMP002",
  "emp_name": "Ziggy DartVader",
  "designation": "Web Designer"
}
```

Raw data

HTML form

Emp id

EMP002

Emp name

Ziggy DartVader

Designation

Web Designer

POST

2. To list the Employee Detail, we use the generics.RetrieveAPIView:

```
138 # updates the specific record
139 def put(self, request, pk):
140     return self.update(request, pk)
141
142 # deletes/destroys a specific record
143 def delete(self, request, pk):
144     return self.destroy(request, pk)
145
146 """
147 # uses Generics for CRUD
148 # -----"
149
150 # lists all employees and create a new record using Generics
151 class Employees(generics.ListAPIView, generics.CreateAPIView):
152     queryset = Employee.objects.all()
153     serializer_class = EmployeeSerializer
154
155 # lists specific employee using Generics
156 class EmployeeDetail(generics.RetrieveAPIView):
157     queryset = Employee.objects.all()
158     serializer_class = EmployeeSerializer
159     lookup_field = 'pk'
160
```

To view the specific record:

→ 127.0.0.1:8000/api/v1/employees/3/

Django REST framework

Employees / Employee Detail

Employee Detail

GET /api/v1/employees/3/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id": 3,
  "emp_id": "EMP003",
  "emp_name": "Russell Lim",
  "designation": "Security"
}
```

3. To update the record, we use the generics.UpdateAPIView:

```
138 # updates the specific record
139 def put(self, request, pk):
140     return self.update(request, pk)
141
142 # deletes/destroys a specific record
143 def delete(self, request, pk):
144     return self.destroy(request, pk)
145     """
146
147 # uses Generics for CRUD
148 # -----"
149
150 # lists all employees and create a new record using Generics
151 class Employees(generics.ListAPIView, generics.CreateAPIView):
152     queryset = Employee.objects.all()
153     serializer_class = EmployeeSerializer
154
155 # lists specific employee using Generics
156 class EmployeeDetail(generics.RetrieveAPIView, generics.UpdateAPIView):
157     queryset = Employee.objects.all()
158     serializer_class = EmployeeSerializer
159     lookup_field = 'pk'
160
```

Updating the form:

Employees / Employee Detail

Employee Detail

OPTIONS

GET ▾

GET /api/v1/employees/3/

HTTP 200 OK
Allow: GET, PUT, PATCH, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "id": 3,  
  "emp_id": "EMP003",  
  "emp_name": "Russell Lim",  
  "designation": "Security"  
}
```

Raw data

HTML form

Emp id

EMP003

Emp name

Russell D'Highness

Designation

Security

PUT

After the PUT submission:



127.0.0.1:8000/api/v...



New Chrome available

Django REST framework

api_djangoadmin

Employees / Employee Detail

Employee Detail

OPTIONS

GET



PUT /api/v1/employees/3/

HTTP 200 OK
Allow: GET, PUT, PATCH, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id": 3,
  "emp_id": "EMP003",
  "emp_name": "Russell D'Highness",
  "designation": "Security"
}
```



Raw data

HTML form

Emp id

EMP003

Emp name

Russell D'Highness

Designation

Security

PUT

4. To delete a record, we add the `generics.DestroyAPIView`:

```
# lists/updates/deletes specific employee using Generics
class EmployeeDetail(generics.RetrieveAPIView, generics.UpdateAPIView, generics.DestroyAPIView):
    queryset = Employee.objects.all()
    serializer_class = EmployeeSerializer
    lookup_field = 'pk'
```

Before DELETE action:



127.0.0.1:8000/api/v...



New Chrome availat

Django REST framework

api_djangoadmin

Employees / Employee Detail

Employee Detail

DELETE

OPTIONS

GET



GET /api/v1/employees/3/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "id": 3,  
  "emp_id": "EMP003",  
  "emp_name": "Russell D'Highness",  
  "designation": "Security"  
}
```

Raw data

HTML form

Emp id

EMP003

Emp name

Russell D'Highness

Designation

Security

PUT

Django REST framework api_djangoadmin

Employee

Emp

GET /api/v1/employees/3/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "id": 3,  
  "emp_id": "EMP003",  
  "emp_name": "Russell D'Highness",  
  "designation": "Security"  
}
```

Raw data HTML form

Emp id EMP003

Emp name Russell D'Highness

Designation Security

PUT

Are you sure you want to delete this Employee Detail?

Cancel Delete

AFTER deleting record # 3:

Employees / Employee Detail

Employee Detail

DELETE

OPTIONS

GET

GET /api/v1/employees/3/

HTTP 404 Not Found
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "detail": "No Employee matches the given query."  
}
```

Raw data

HTML form

Emp id

Emp name

Designation

PUT

5. To delete/update/list a specific record using only `generics.RetrieveUpdateDestroyAPIView`.

```
# lists/updates/deletes specific employee using Generics using the long way  
# class EmployeeDetail(generics.RetrieveAPIView, generics.UpdateAPIView, generics.DestroyAPIView):  
#     queryset = Employee.objects.all()  
#     serializer_class = EmployeeSerializer  
#     lookup_field = 'pk'  
  
# lists/updates/deletes specific employee using Generics the faster way  
class EmployeeDetail(generics.RetrieveUpdateDestroyAPIView):  
    queryset = Employee.objects.all()  
    serializer_class = EmployeeSerializer  
    lookup_field = 'pk'
```

Employees / Employee Detail

Employee Detail

DELETE OPTIONS GET

GET /api/v1/employees/8/

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 8,
  "emp_id": "test",
  "emp_name": "test",
  "designation": "test"
}
```

Raw data HTML form

Emp id	<input type="text" value="test"/>
Emp name	<input type="text" value="test"/>
Designation	<input type="text" value="test"/>

PUT

In summary, Generics views are easier and more efficient than using other ways like function-based views or mixins.