

Django Views

1. Create an API for updating records, using PUT and PATCH.
2. Create an API for deleting records with custom error messages.
3. Create an API for bulk deletion.

PUT and Patch

```
# Update using PUT and PATCH

class UpdateApi(generics.UpdateAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataSerializer

    def put(self, request, *args, **kwargs):
        instance = self.get_object()
        serlaizer=self.get_serializer(instance,data=request.data)

        if serlaizer.is_valid():
            serlaizer.save()
            return Response({'Message': 'Updated successfully'},status=200)
        return Response({'Message':"Failed to update","erroe":serlaizer.errors},status=400)

    def patch(self, request, *args, **kwargs):
        instance = self.get_object()
        serlaizer=self.get_serializer(instance,data=request.data,partial=True)

        if serlaizer.is_valid():
            serlaizer.save()
            return Response({'Message': 'Updated successfully'},status=200)
        return Response({'Message':"Failed to update","erroe":serlaizer.errors},status=400)
```

PUT

The screenshot displays a REST client interface with a PUT request configured. The URL is `http://127.0.0.1:8000/update_user_data/1/`. The request body is a JSON object with the following structure:

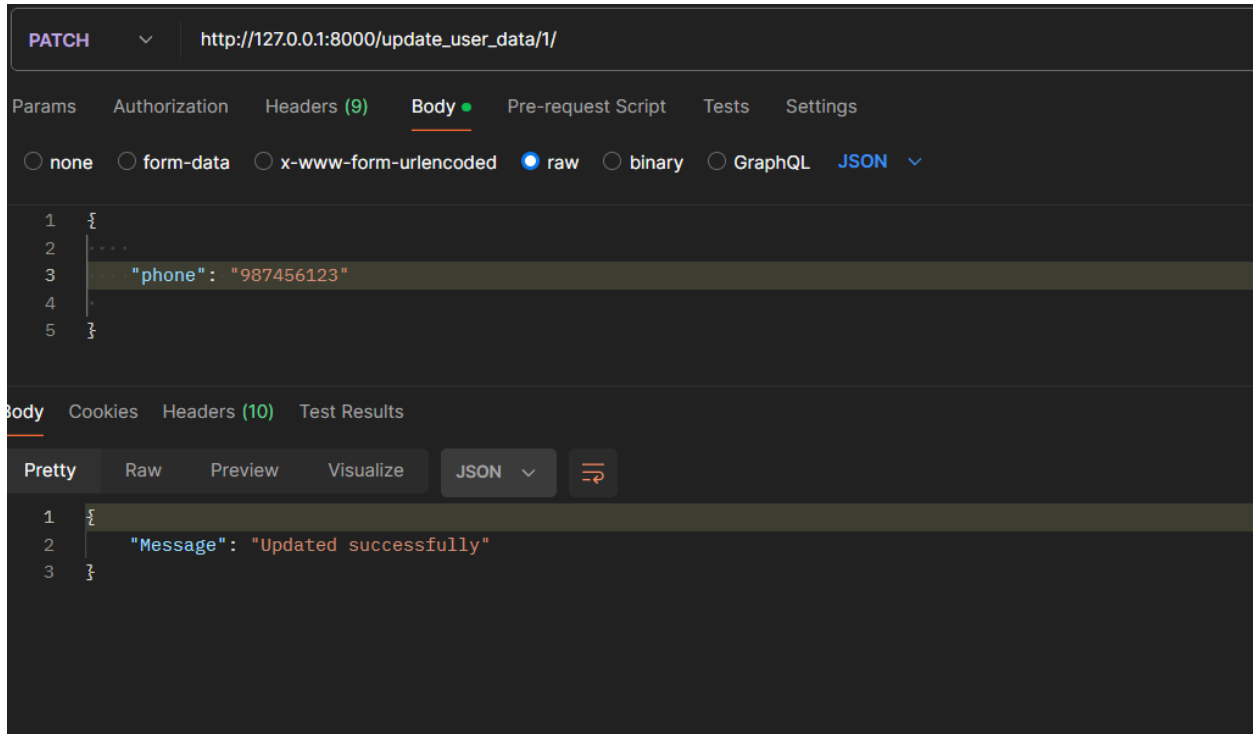
```
1 {  
2   "address": "Abcdf",  
3   "phone": "12345678",  
4   "user": 1  
5 }
```

The response body is also a JSON object, showing a successful update message:

```
1 {  
2   "Message": "Updated successfully"  
3 }
```

The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the raw JSON data. Below the request body, there are tabs for Cookies, Headers (10), and Test Results. The response body is displayed in a Pretty JSON format, with a dropdown menu set to JSON and a button to toggle between raw and pretty views.

Patch



Delete

```
# Api for delete
class DeleteApi(generics.DestroyAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataSerializer

    def destroy(self, request, *args, **kwargs):
        instance = self.get_object()
        try:
            self.perform_destroy(instance)
        except Exception as e:
            return Response({'error': 'Failed to delete the object', 'detail': str(e)}, status=400)

        return Response({'Message': 'Object deleted successfully'}, status=200)
```


DELETE ▼ http://127.0.0.1:8000/deleteapi/4/

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
1 {
2   "Message": "Object deleted successfully"
3 }
```

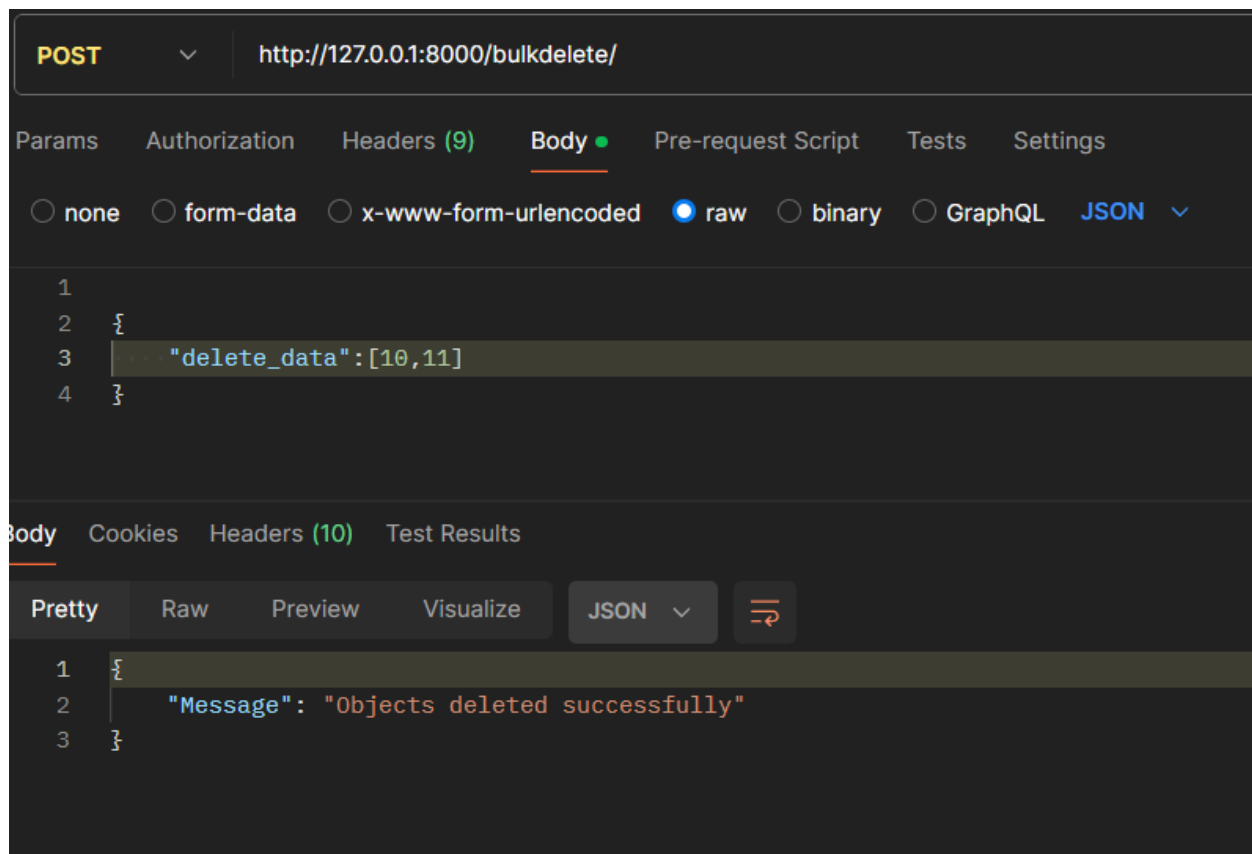
Bulk Delete

```
# Api for bulk delete

class BulkDelete(APIView):

    def post(self, request):
        data = request.data
        delete_data=data.get('delete_data')
        try:
            delete_data_object= UserData.objects.filter(pk__in=delete_data).delete()
        except Exception as e:
            return Response({'error': 'Failed to delete the object', 'detail': str(e)}, status=400)

        return Response({'Message': 'Objects deleted successfully'}, status=200)
```



Project Link

https://github.com/Albin-Kuriachan/python_beniex/tree/main/17.04.2024