Django\_api\_views

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!

Here is the API views code for CRUD operations using Django Rest Framework:  
```  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Product  
from.serializers import ProductSerializer  
  
# Create  
class ProductCreate(APIView):  
 serializer\_class = ProductSerializer  
  
 def post(self, request):  
 serializer = self.serializer\_class(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=status.HTTP\_201\_CREATED)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Read  
class ProductList(APIView):  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def get(self, request):  
 products = self.queryset.all()  
 serializer = self.serializer\_class(products, many=True)  
 return Response(serializer.data)  
  
# Update  
class ProductUpdate(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
 serializer\_class = ProductSerializer  
  
 def put(self, request, id):  
 product = self.queryset.get(id=id)  
 serializer = self.serializer\_class(product, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)  
  
# Delete  
class ProductDelete(APIView):  
 lookup\_field = 'id'  
 queryset = Product.objects.all()  
  
 def delete(self, request, id):  
 product = self.queryset.get(id=id)  
 product.delete()  
 return Response(status=status.HTTP\_204\_NO\_CONTENT)  
```  
Let me know if there is anything else I can assist you with!