Django\_api\_views

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).

Here is your requested views.py code for the model 'Customer':  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
Here's your requested views.py code for the model 'Customer':  
  
  
```python  
# Views for CRUD operations of Customers  
  
from rest\_framework.response import Response  
from rest\_framework.views import APIView  
from.models import Customer  
from.serializers import CustomerSerializer  
  
class ListCustomers(APIView):  
 # Get all customers  
 def get(self, request):  
 customers = Customer.objects.all()  
 serializer = CustomerSerializer(customers, many=True)  
 return Response(serializer.data)  
  
 # Create new customer  
 def post(self, request):  
 serializer = CustomerSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
  
  
class RetrieveUpdateDeleteCustomer(APIView):  
 # Get single customer by id  
 def get\_object(self, pk):  
 try:  
 return Customer.objects.get(pk=pk)  
 except Customer.DoesNotExist:  
 return Response(status=404)  
  
 # Retrieve single customer  
 def get(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer)  
 return Response(serializer.data)  
  
 # Update existing customer  
 def put(self, request, pk):  
 customer = self.get\_object(pk)  
 serializer = CustomerSerializer(customer, data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data)  
 return Response(serializer.errors, status=400)  
  
 # Delete existing customer  
 def delete(self, request, pk):  
 customer = self.get\_object(pk)  
 customer.delete()  
 return Response(status=204)  
```  
  
This provides the necessary CRUD functionality for the `Customer` model using RESTful APIs.  
  
The following endpoints are available:  
  
\* `/customers/`: Retrieves a list of all customers.  
\* `/customers/`: Creates a new customer.  
\* `/customers/{pk}/`: Retrieves a specific customer by ID (`pk`).  
\* `/customers/{pk}/`: Updates an existing customer.  
\* `/customers/{pk}/`: Deletes a specific customer by ID (`pk`).