Tutorial: Crear un CRUD Completo en Django

---

1. Crear el Modelo

Vamos a crear un modelo en la aplicación `inventario` que represente, por ejemplo, un sistema de inventario de productos.

Abre el archivo `models.py` en la carpeta `inventario` y crea el siguiente modelo:

```python
from django.db import models

class Producto(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    cantidad = models.IntegerField()

    def __str__(self):
        return self.nombre
```

Explicación:

- **`nombre`**: Un campo de texto para el nombre del producto.
- **`descripcion`**: Un campo de texto para describir el producto.
- **`precio`**: Un campo numérico para el precio del producto.
- **`cantidad`**: Un campo numérico para la cantidad del producto en inventario.

---

2. Crear las Migraciones y Aplicarlas

Una vez que hemos definido nuestro modelo, debemos crear las migraciones y aplicarlas para crear la tabla correspondiente en la base de datos.

Paso 1: Crear Migraciones

Ejecuta el siguiente comando para crear las migraciones:

```
python manage.py makemigrations
```

Paso 2: Aplicar Migraciones

Aplica las migraciones para crear la tabla en la base de datos:

```
python manage.py migrate
```

---

3. Crear las Vistas y URL Patterns

Ahora, crearemos las vistas para manejar las operaciones del CRUD (Crear, Leer, Actualizar, Eliminar). Estas vistas estarán conectadas con URLs específicas para cada acción.

Paso 1: Crear las Vistas

Abre el archivo `views.py` de la aplicación `inventario` y define las vistas para las operaciones CRUD.

```python
from django.shortcuts import render, get_object_or_404, redirect
from .models import Producto
from .forms import ProductoForm

# Vista para listar los productos
def lista_productos(request):
    productos = Producto.objects.all()
    return render(request, 'inventario/lista_productos.html', {'productos': productos})

# Vista para ver un producto específico
def ver_producto(request, pk):
    producto = get_object_or_404(Producto, pk=pk)
    return render(request, 'inventario/ver_producto.html', {'producto': producto})

# Vista para crear un nuevo producto
def crear_producto(request):
    if request.method == 'POST':
        form = ProductoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('lista_productos')
    else:
        form = ProductoForm()
    return render(request, 'inventario/crear_producto.html', {'form': form})

# Vista para editar un producto existente
def editar_producto(request, pk):
    producto = get_object_or_404(Producto, pk=pk)
    if request.method == 'POST':
        form = ProductoForm(request.POST, instance=producto)
        if form.is_valid():
            form.save()
            return redirect('lista_productos')
    else:
        form = Product Form(instance=producto)
    return render(request, 'inventario/editar_producto.html', {'form': form})

# Vista para eliminar un producto
def eliminar_producto(request, pk):
    producto = get_object_or_404(Producto, pk=pk)
    if request.method == 'POST':
        producto.delete()
        return redirect('lista_productos')
    return render(request, 'inventario/eliminar_producto.html', {'producto': producto})
```

Explicación de las vistas:

- **`lista_productos`**: Muestra todos los productos del inventario.
- **`ver_producto`**: Muestra los detalles de un producto específico.
- **`crear_producto`**: Muestra un formulario para crear un nuevo producto.
- **`editar_producto`**: Muestra un formulario para editar un producto existente.
- **`eliminar_producto`**: Elimina un producto del inventario.

Paso 2: Crear el archivo `urls.py`

Ahora, necesitamos conectar estas vistas a URLs. Crea un archivo `urls.py` dentro de la aplicación `inventario` y define las rutas.

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.lista_productos, name='lista_productos'),
    path('producto/<int:pk>/', views.ver_producto, name='ver_producto'),
    path('producto/nuevo/', views.crear_producto, name='crear_producto'),
    path('producto/editar/<int:pk>/', views.editar_producto,name='editar_producto'),
    path('producto/eliminar/<int:pk>/',views.eliminar_producto,name='eliminar_producto'),
]
```

Paso 3: Incluir las URLs en el proyecto

Abre el archivo `urls.py` del proyecto principal (`mi_proyecto/urls.py`) y agrega la inclusión de las URLs de la aplicación `inventario`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inventario/', include('inventario.urls')),
]
```

---

4. Crear los Formularios

Para las operaciones de crear y editar productos, vamos a utilizar formularios. Crearemos un archivo `forms.py` en la aplicación `inventario` con el siguiente contenido:

```
from django import forms
from .models import Producto

class ProductoForm(forms.ModelForm):
    class Meta:
        model = Producto
        fields = ['nombre', 'descripcion', 'precio', 'cantidad']
```

---

5. Crear las Plantillas

Finalmente, vamos a crear las plantillas HTML correspondientes para cada vista.

`lista_productos.html`

```
{% extends 'base.html' %}

{% block content %}
    <h1>Productos</h1>
    <a href="{% url 'crear_producto' %}">Crear Producto</a>
    <table>
```

```
        <thead>
            <tr>
                <th>Nombre</th>
                <th>Precio</th>
                <th>Acciones</th>
            </tr>
        </thead>
        <tbody>
            {% for producto in productos %}
                <tr>
                    <td>{{ producto.nombre }}</td>
                    <td>{{ producto.precio }}</td>
                    <td>
                        <a href="{% url 'ver_producto' producto.pk %}">Ver</a>
                        <a href="{% url 'editar_producto' producto.pk %}">Editar</a>
                        <a href="{% url 'eliminar_producto' producto.pk %}">Eliminar</a>
                    </td>
                </tr>
            {% empty %}
                <tr>
                    <td colspan="3">No hay productos disponibles.</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

ver_producto.html

```
{% extends 'base.html' %}

{% block content %}
    <h1>{{ producto.nombre }}</h1>
    <p><strong>Descripción:</strong> {{ producto.descripcion }}</p>
    <p><strong>Precio:</strong> {{ producto.precio }}</p>
    <p><strong>Cantidad:</strong> {{ producto.cantidad }}</p>
    <a href="{% url 'editar_producto' producto.pk %}">Editar</a>
    <a href="{% url 'eliminar_producto' producto.pk %}">Eliminar</a>
{% endblock %}
```

`crear_producto.html` y `editar_producto.html`

Estas plantillas se parecen entre sí, ya que ambas muestran un formulario.

```
{% extends 'base.html' %}

{% block content %}
    <h1>{% if producto %}Editar{% else %}Crear{% endif %} Producto</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">
          {% if producto %}Actualizar{% else %}Crear{% endif %}
        </button>
    </form>
{% endblock %}
```

`eliminar_producto.html`

```
 {% extends 'base.html' %}

{% block content %}
    <h1>Eliminar Producto</h1>
    <p>¿Estás seguro de que quieres eliminar el producto
       "{{ producto.nombre }}"?</p>
    <form method="post">
        {% csrf_token %}
        <button type="submit">Sí, eliminar</button>
    </form>
    <a href="{% url 'lista_productos' %}">Cancelar</a>
{% endblock %}
```

---

6. Probar el CRUD

Con todo esto en su lugar, ahora puedes probar tu CRUD. Ejecuta el servidor de desarrollo:

```
python manage.py runserver
```

Visita las URL correspondientes (`/inventario/`, `/inventario/producto/nuevo/`, etc.) para asegurarte de que todas las operaciones de Crear, Leer, Actualizar y Eliminar funcionan correctamente.