Guía levantar un formulario desde Django con Forms y Models.

ABPro Proyecto Grupal 3 - Equipo 3

- **Al ser la continuación de la guía anterior, comenzaremos desde el punto 27 asumiendo que los otros pasos ya fueron realizados.**

27. Define un modelo:

➢ En el archivo “models.py” de tu aplicación, define un modelo utilizando la clase “models.Mode”l. Define los campos necesarios para tu formulario como atributos de la clase, utilizando los diferentes tipos de campos que ofrece Django.

models.py

```python
from django.db import models

# Create your models here.

class Usuario(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    correo = models.EmailField()
    telefono = models.CharField(max_length=20)
    dirección = models.CharField(max_length=100)
    Ciudad = models.CharField(max_length=100)

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

28. Crea un formulario:

forms.py

```python
from django import forms

class RegistroProveedorForm (forms.Form):
    razon_social = forms.CharField(label='Razón Social', required=True,
                                   max_length=50,
                                   error_messages={
                                       'required': 'La razón social es obligatoria',
                                       'max_length': 'El nombre no puede superar los 50 caracteres'
                                   },
                                   widget=forms.TextInput(attrs={
                                       'placeholder': 'Ingrese la razón social',
                                       'class': 'form-control'
                                   }),
                                   help_text='Queremos colaborar, ingrese la razón social'
                                   )
    nombre = forms.CharField(label='Nombre', required=True,…
    rep_legal = forms.CharField(label='Representante Legal', required=True,…
    rut = forms.CharField(label='RUT', required=True,…
    dirección = forms.CharField(label='Dirección', required=True,…
    correo = forms.EmailField(label='Email', required=True,…
    telefono = forms.CharField(label='Teléfono', required=True,…
```

En el archivo "forms.py” de tu aplicación, importa la clase forms de Django y crea una clase que herede de “forms.ModelForm”.
Especifica el modelo asociado al formulario utilizando la metaclase Meta.

29. Configura las vistas:

➢ En el archivo “views.py” de tu aplicación, importa el formulario y define una función de vista que renderice el formulario. Puedes utilizar la función render para cargar una plantilla HTML que contenga el formulario.

views.py

```python
from django.shortcuts import redirect, render
from .models import Usuario, proveedor
from .forms import RegistroProveedorForm
from django.views.generic import TemplateView

def mensaje(request):
    return render(request, 'index.html')

def lista_usuarios(request)
    usuarios = Usuario.obje
    return render(request,
```

Puede ser una estructura sencilla o puede ser más compleja, todo dependerá de lo que tu aplicación requiera.

views.py

```python
class ProveedorView(TemplateView):
  template_name = 'proveedores.html'

  def get(self, request, *args, **kwargs):
    formulario = RegistroProveedorForm()
    return render(request, self.template_name, { "formulario": formulario})

  def post(self, request, *args, **kwargs):
      formulario = RegistroProveedorForm(request.POST)
      mensajes = {
        "enviado": False,
        "resultado": None
      }
      if formulario.is_valid():
        razon_social = formulario.cleaned_data["razon_social"]
        nombre = formulario.cleaned_data["nombre"]
        rep_legal = formulario.cleaned_data["rep_legal"]
        rut =formulario.cleaned_data["rut"]
        dirección = formulario.cleaned_data["dirección"]
        correo = formulario.cleaned_data["correo"]
        telefono = formulario.cleaned_data["telefono"]
        registro = proveedor (
          razon_social = razon_social,
          nombre = nombre,
          rep_legal = rep_legal,
          rut = rut,
          dirección = dirección,
          correo = correo,
          telefono = telefono,
        )
        registro.save()
        mensajes = {"enviado": True, "resultado": "Mensaje enviado correctamente"}
      else:
        mensajes = {"enviado": False, "resultado": formulario.errors}
      return render(request, self.template_name, { "formulario": formulario, "mensajes": mensajes })
```

30. Define una URL:

➢ En el archivo “urls.py” de tu aplicación, configura una URL para la vista que creaste. Puedes utilizar la función path para mapear la URL a la vista correspondiente.

```python
path('proveedor/',ProveedorView.as_view(),name='registro_proveedores
'),
```

En este caso estaremos trabajando con la ruta de “Proveedor”

31. Crea una plantilla HTML:

➢ En el directorio de tu aplicación, crea un archivo HTML que contenga el código HTML necesario para mostrar el formulario. Utiliza las etiquetas y atributos de Django para renderizar los campos del formulario.

Este es un ejemplo el cual puede ser modificado a la preferencia de tu aplicación.

```
{% block content %}
  <form method="post">
    {% csrf_token %}
    {% for campo in formulario %}
      <div class="form-group">
            <label  for="{{  campo.id_for_label  }}">{{  campo.label
}}</label>
        {{ campo }}
        {% if campo.help_text %}
            <small  id="{{  campo.id_for_label  }}"  class="form-text
text-muted">{{ campo.help_text }}</small>
        {% endif %}
      </div>
    {% endfor %}
    {% if mensajes and mensajes.enviado %}
    <div class="alert alert-success class="mt-3" role="alert">
        {{ mensajes.resultado }}
    </div>
    {% endif %}
    {% if mensajes and not mensajes.enviado %}
    <div class="alert alert-danger class="mt-3" role="alert">
        {{ mensajes.resultado }}
    </div>
    {% endif %}
    <button type="submit" class="btn btn-dark mb-4">Enviar</button>
  </form>
{% endblock %}
```

32. Ejecuta las migraciones para crear las tablas correspondientes en la base de datos.:

➢ Utiliza los comandos:

```
>>> python manage.py makemigrations
```

Seguido a makemigrations utilizamos

```
>>> python manage.py migrate
```

Esto es para generar y aplicar las migraciones.

Recuerda que estos comandos deben aplicarse a nivel de proyecto, ya que se manejan a través del archivo "*manage.py*", estos deben ser realizados desde nuestro terminal.

33. Persistencia de datos::

➢ En la función de vista que renderiza el formulario, verifica si el método de la solicitud HTTP es POST (`method="post"`). Si es así, instancia el formulario con los datos de la solicitud y valida el formulario. Si el formulario es válido, guarda los datos en la base de datos utilizando el método `save()`. Si el formulario no es válido, muestra los errores en la plantilla para que estos puedan ser resueltos.

34. Levanta el servidor:

➢ Utilizando el comando

```
>>> python manage.py runserver
```

para levantar el servidor de desarrollo de Django.

35. Accede al formulario:

➢ Finalmente abre tu navegador web de preferencia y accede a la URL correspondiente al formulario que configuraste. Deberías poder ver y completar el formulario.

TE LO VENDO

Nosotros inventamos la triple "B"

Formulario para proveedores

Se parte de nuetro equipo

Razón Social

Ingrese la razón social

Queremos colaborar, ingrese la razón social

Nombre

Ingrese el nombre de la empresa

Queremos colaborar, ingrese el nombre de la empresa

Representante Legal

Ingrese al representante legal de la empresa

Queremos colaborar, ingrese el representante legal de la empresa

RUT

Ingrese el RUT de la empresa

Queremos colaborar, ingrese el RUT de la empresa

Dirección

Ingrese la dirección de la empresa

Queremos colaborar, ingrese la dirección de la empresa

Email

correo@ejemplo.com

Teléfono

No olvide ingresar el número de teléfono

Enviar