

## 4.Inserción many to many

La idea ahora sería ver cómo podemos realizar una inserción cuando existe una relación manytomany. Tal es el caso de nuestra nuestro modelo/tabla Book. Vamos a implementar la posibilidad de registrar un nuevo libro.

Creemos el enlace para poder tener la opción en libro.html

```
{% extends 'index.html' %}
<!--<h1>Listado de libros</h1>-->
{%block title%} Listado de Libros {%endblock title%}
{%block content%}
<table border="2">
  <thead>
    <th>Titulo</th>
    <th>Codigo</th>
    <th>Autor</th>
  </thead>
  <tbody>

    {% for bo in books %}
    <tr>
      <td>{{ bo.title }} </td>
      <td>{{ bo.cod }}</td>
      <td>{% for au in bo.author.all %}
        {{au}}
      {% endfor %}
      </td>
    </tr>
    {% endfor %}
  </tbody>
  <a href="/new-libro/">Registrar nuevo libro</a>
</table>
{%endblock content%}
```

Creamos así mismo la función en views.py y la correspondiente url.

Inicialmente creamos la función book\_create de esta forma

```
def book_create(request):

    return render(request, 'create_libro.html') #Si lo tenéis dentro de una subcarpeta dentro de
template, acordaros de poner el nombre de la carpeta delante de la página.
```

Y añadimos la url:

```
urlpatterns = [
    #path('hola/', views.index), #indicamos que la vista la queremos mostrar en esa ruta.
    path('', home), #quí no habría que meter la carpeta

    #path('otramas/', views.home),
    path('admin/', admin.site.urls),
    path('autor/', autor_list),
    path('libro/', libro_list),
    path ('new-autor/', autor_create),
```

```

    path ('update-autor/<int:pk>', autor_update), #Hay que pasarle el pk por parámetro. Para realizarlo
se hace entre símbolos de <> indicando en primer lugar el tipo del campo, en este caso int y luego el
nombre del campo pk.
    path ('delete-autor/<int:pk>', autor_delete),
    path ('new-book/', book_create)
]

```

Hay que añadir también en forms.py, el BookForm, al igual que hicimos con Autor, para tener asociado un formulario a la tabla Book, y no tener que meter campo a campo en la página html, cada uno de los input que deseemos incluir.

```

class BookForm (forms.ModelForm):
    class Meta:
        model=Book
        fields='__all__'

```

Ahora en nuestra views.py añadimos el contexto como parámetro, al igual que hicimos con los autores. Este parámetro contexto será el formulario, para que la página create\_libro.html lo pueda pintar. Hay que acordarse de importar BookForm :

```

def book_create(request):

    return render(request, 'create_libro.html', {'book_form': BookForm})

```

Igualmente, vamos a incorporar el formulario a nuestra página create\_book.html, que tiene código similar al de create\_autor.html.

```

{% extends 'index.html' %}
{%block title%} Registrar nuevo libro {%endblock title%}

{%block content%}

    {{book_form}}

{%endblock content%}

```

Nos saldría al pulsar registrar nuevo libro el siguiente formulario.

Vamos ir añadiendo a nuestra página create\_libro.html lo necesario para que pueda guardar correctamente los datos que introduzcamos. Añadimos el botón de guardar con el form, así como el csrf\_token para que no nos salga el error del csrf no encontrado.

```
{% extends 'index.html' %}
{%block title%} Registrar nuevo libro {%endblock title%}

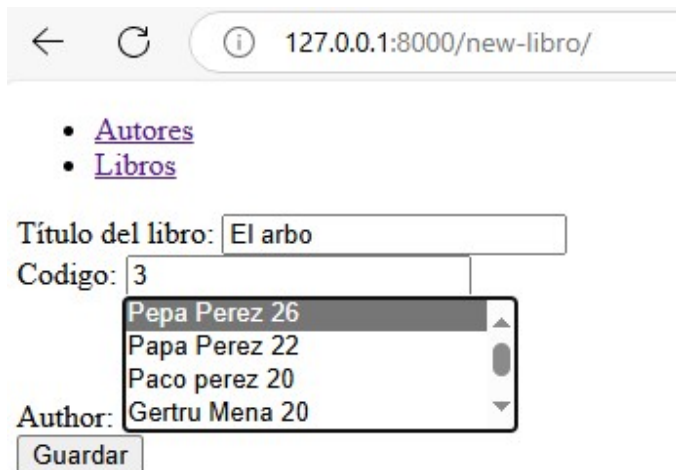
{%block content%}
    {%csrf_token%}
    <form action="" method="POST">
    {%csrf_token%}
    {{book_form}}
    <button type="submit">Guardar</button>
    </form>
{%endblock content%}
```

Cuando presionamos el botón de guardar debe entrar en nuestra views.py, en concreto en la función book\_create, pero el método es POST, no GET, como hasta ahora lo usábamos en esa función. Hay que especificar esa diferenciación en el código

Vamos además a realizar la siguiente prueba, para pintar en el servidor la información que nos gustaría guardar:

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':
        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
la información está en book_form.
            #Si realiza un print de esa información vamos a ver qué sale en el servidor.
            print(book_form.cleaned_data)
```

Intento insertar un libro de esta manera:



Al pulsar el botón guardar, me imprime en el servidor lo siguiente:

```
[08/Feb/2025 13:50:32] "GET /new-libro/ HTTP/1.1" 200 1521
{'title': 'El arbo', 'cod': '3', 'author': <QuerySet [<Author: Pepa Perez 26>]>}}
[08/Feb/2025 13:51:42] "POST /new-libro/ HTTP/1.1" 200 723
```

Si nos fijamos, en el campo author, está recibiendo un campo de tipo QuerySet, por el motivo de que la relación es manytomany y podría recibir más de uno.

Ahora realizamos la siguiente prueba, imprimiendo la información que se recibe antes de pasar la información a Form, y volvemos a guardar para ver qué sale en el servidor:

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':
        print (request.POST)
        print ("-----")
        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.
            #Si realiza un print de esa información vamos a ver qué sale en el servidor.
            print(book_form.cleaned_data) #En cleaned_data se guarda la información valida.
```



Cuando le damos a guardar, en el servidor nos sale

```
08/Feb/2025 14:01:54] "POST /new-libro/ HTTP/1.1" 500 65388
QueryDict: {'csrfmiddlewaretoken': ['mUjZaZnmVvFqzppBNEHIIzAkzuiGVNnEN1SE2o0r5RxigJM6mVithAuJ56EIC'], 'title': ['qerq'], 'cod': ['88'], 'author': ['2']}>
{'title': 'qerq', 'cod': '88', 'author': <QuerySet [(<Author: Pepa Perez 26>)]>}
```

Lo que hemos hecho ha sido imprimir el request.post tal cual nos está llegando e imprimir la información que nos llega cuando ya ha sido enviada una instancia de Form de Django y fue pasado el método `is_valid()`. En la variable `cleaned_data` se guarda la información validada.

Si comparamos las dos visualizaciones en el servidor, en ambos me imprime un diccionario pero la diferencia está en el campo `author`, en el primero me muestra el id y en el segundo me muestra un query-set. El motivo es porque lo que se va a hacer por detrás es llamar a la función del ORM de Django, `create`, que registrará en la BD, y cuando lo que se pasa son registros de relaciones de los modelos, el método `create` necesita que le enviemos la instancia completa, no el id solamente. Cuando hacemos el `is_valid`, digamos que convierte los datos recibidos al formato que necesita el `create` y cuando hay relaciones `manytomany` necesita un `QuerySet`. (Esto es simplemente más información)

Vamos a continuar con el guardado del registro (quitamos los print), pero antes mencionar que el método `create`, que necesita como parámetro lo siguiente:

```
...á en book_form.
#Si realiza un print de esa información vamos a ver qué sale en el servidor.
new_book=Book.objects.create(**kwargs) #kwargs es un diccionario que contiene la información del formulario.
```

Luego entonces nuestro código para poder guardar un libro quedaría:

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':
        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.
            #Si realiza un print de esa información vamos a ver qué sale en el servidor.
            new_book=Book.objects.create(**book_form) #No hace falta especificar los campos de mi
            formulario. Se realizaría de esta forma. La variable book_form es un diccionario. Con esto registramos
            el nuevo libro
```

Tal cual lo tenemos, nos sale el siguiente error:

## TypeError at /new-libro/

django.db.models.query.QuerySet.create() argument after \*\* must be a mapping, not BookForm

```
Request Method: POST
Request URL: http://127.0.0.1:8000/new-libro/
Django Version: 5.0.1
Exception Type: TypeError
Exception Value: django.db.models.query.QuerySet.create() argument after ** must be a mapping, not BookForm
Exception Location: C:\Users\gertr\workspace\UnoDjango\core\views.py, line 83, in book_create
Raised during: core.views.book_create
Python Executable: C:\Users\gertr\AppData\Local\Programs\Python\Python312\python.exe
Python Version: 3.12.6
Python Path: ['C:\\Users\\gertr\\workspace\\UnoDjango',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\python312.zip',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\DLLs',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages']
Server time: Sat, 08 Feb 2025 13:35:46 +0000
```

Este error nos está indicando, en la línea que estamos haciendo el create, que lo que se está recibiendo como parámetro debe ser un map y no una instancia de BookForm. Hay que pasarle el BookForm.cleaned\_data, que tiene la información que queremos.

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':

        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.
            #Si realiza un print de esa información vamos a ver qué sale en el servidor.
            new_book=Book.objects.create(**book_form.cleaned_data) #No hace falta especificar los
            campos de mi formulario. Se realizaría de esta forma. La variable book_form es un diccionario. Con esto
            registramos el nuevo libro
```

Vamos a tener otro error:

← ↻ ⓘ 127.0.0.1:8000/new-libro/

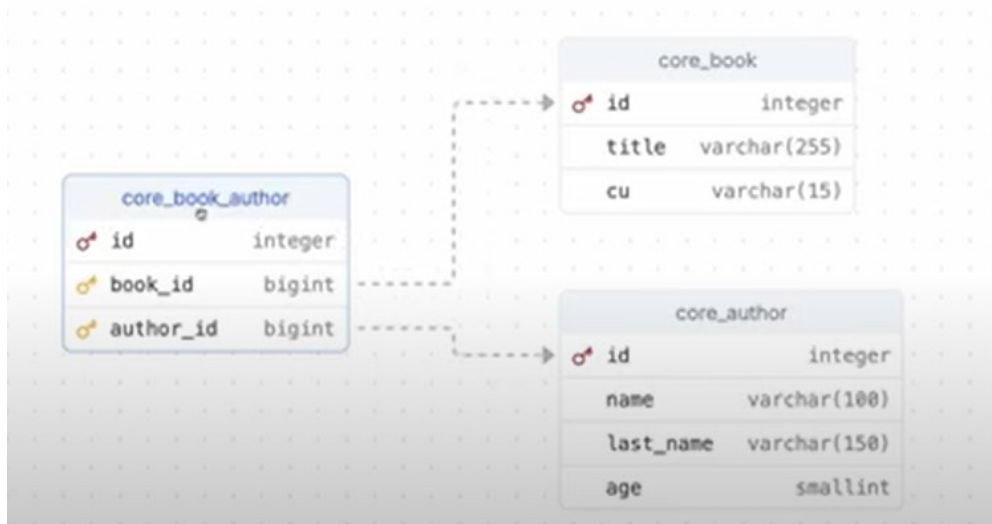
## TypeError at /new-libro/

Direct assignment to the forward side of a many-to-many set is prohibited. Use author.set() instead.

```
Request Method: POST
Request URL: http://127.0.0.1:8000/new-libro/
Django Version: 5.0.1
Exception Type: TypeError
Exception Value: Direct assignment to the forward side of a many-to-many set is prohibited. Use author.set() instead.
Exception Location: C:\Users\gertr\AppData\Local\Programs\Python\Python312\Lib\site-packages\django\db\models\fields\related_descriptors.py, line 658, in __set__
Raised during: core.views.book_create
Python Executable: C:\Users\gertr\AppData\Local\Programs\Python\Python312\python.exe
Python Version: 3.12.6
Python Path: ['C:\\Users\\gertr\\workspace\\UnoDjango',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\python312.zip',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\DLLs',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib',
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages']
Server time: Sat, 08 Feb 2025 13:43:30 +0000
```

Este error nos está indicando que no podemos usar directamente una asignación a un campo manytomany que hay que user el set. Si fuera onetoone or foreing key, no sale este error.

La cuestión es que cuando hacemos manytomany, Django crea una tabla adicional para poder normalizar la BD, a la que se conectan las otras dos tablas implicadas en la relación. En la intermedia es dónde se guarda la información de la relación. Existe pues además de la tabla Book y Author esta tabla intermedia. El problema es que no podemos asignar al modelo libro la relación, pues ésta está en la tabla core\_book\_author.



Entonces lo que habría que hacer es registrar la información por separado, por un lado los campos de la tabla Book y por otro el campo relacional.

```

def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':

        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.

            new_book=Book.objects.create(title=book_form.cleaned_data.get('title'),cod=book_form.cleaned_data.get('cod'))
            #Hasta aquí habrá registrado mi libro. Ahora pasamos al campo relaciona.
            #Ahora guardo la instancia que acabo de registrar, enviándole el id que quiero guardar
            new_book.set(book_form.cleaned_data.get('author'))
    
```

Nos da el siguiente error, porque el new book no tiene atributo set, lo tiene el campo implicado en la relación manytomany, vamos a modificarlo y añadimos una redirección hacia la página de consulta de libros, para que veamos el nuevo libro insertado:



← ↻ ⓘ 127.0.0.1:8000/new-libro/

## AttributeError at /new-libro/

'Book' object has no attribute 'set'

Request Method: POST  
Request URL: http://127.0.0.1:8000/new-libro/  
Django Version: 5.0.1  
Exception Type: AttributeError  
Exception Value: 'Book' object has no attribute 'set'  
Exception Location: C:\Users\gertr\workspace\UnoDjango\core\views.py, line 86, in book\_create  
Raised during: core.views.book\_create  
Python Executable: C:\Users\gertr\AppData\Local\Programs\Python\Python312\python.exe  
Python Version: 3.12.6  
Python Path: ['C:\\Users\\gertr\\workspace\\UnoDjango',  
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\python312.zip',  
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\DLLs',  
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib',  
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312',  
'C:\\Users\\gertr\\AppData\\Local\\Programs\\Python\\Python312\\Lib\\site-packages']  
Server time: Sat, 08 Feb 2025 14:14:10 +0000

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':

        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.

            new_book=Book.objects.create(title=book_form.cleaned_data.get('title'),cod=book_form.cleaned_data.get('cod'))
            #Hasta aquí habrá registrado mi libro. Ahora pasamos al campo relaciona.
            #Ahora guardo la instancia que acabo de registrar, enviándole el id que quiero guardar
            #el atributo set, lo tiene el campo implicado en la relación manytomany
            new_book.author.set(book_form.cleaned_data.get('author'))
            return redirect('/libro/')
```

Tratamos de insertar ahora el nuevo libro:



← ↻ ⓘ 127.0.0.1:8000/new-libro/

- [Autores](#)
- [Libros](#)

Título del libro:

Codigo:

Author: 

Pepa Perez 26

Papa Perez 22

Paco perez 20

Gertru Mena 20

Y me lo inserta:

← ↻ ⓘ 127.0.0.1:8000/libro/

- [Autores](#)
- [Libros](#)

[Registrar nuevo libro](#)

Titulo	Codigo	Autor
La gran princesa	222	
El principito	636	Gertru Mena 20
El gran misterio	98652	
La botella de vino	968	
El pastor alemán	3444	Pepa Perez 26
el arbol	23	
El matorral	2	Paco perez 20

Si insertamos dos autores para un libro:

← ↻ ⓘ 127.0.0.1:8000/new-libro/

- [Autores](#)
- [Libros](#)

Título del libro:

Codigo:

Author: 

Pepa Perez 26

Papa Perez 22

Paco perez 20

Gertru Mena 20

Se guarda así:

← ↻ ⓘ 127.0.0.1:8000/libro/

- [Autores](#)
- [Libros](#)

[Registrar nuevo libro](#)

Titulo	Codigo	Autor
La gran princesa	222	
El principito	636	Gertru Mena 20
El gran misterio	98652	
La botella de vino	968	
El pastor alemán	3444	Pepa Perez 26
el arbol	23	
El matorral	2	Paco perez 20
El arbusto	111	Pepa Perez 26 Papa Perez 22

Faltaría incluir la opción de qué ocurre cuándo el libro que estamos insertando no es válido:

```
def book_create(request):
    if request.method == 'GET':
        return render(request, 'create_libro.html', {'book_form': BookForm})
    if request.method == 'POST':

        book_form=BookForm(data=request.POST) #data es lo que le enviamos para que inserte
        if book_form.is_valid(): #Valido la información que me envían para guardar el libro. Realmente
            la información está en book_form.
```

```

        new_book=Book.objects.create(title=book_form.cleaned_data.get('title'),cod=book_form.cleaned_data.get('cod'))
        #Hasta aquí habrá registrado mi libro. Ahora pasamos al campo relaciona.
        #Ahora guardo la instancia que acabo de registrar, enviándole el id que quiero guardar
        #el atributo set, lo tiene el campo implicado en la relación manytomany
        new_book.author.set(book_form.cleaned_data.get('author'))
        return redirect('/libro/')
    else:
        return render(request, 'create_libro.html',{'book_form':book_form}) #Le enviamos a la
página de creación con los datos incluidos para que pueda comprobar qué dato está mal.

```

Despues de incorporar este else, si intento introducir un libro con un mismo código me avisaría de que el código está repetido:

← ↻ ⓘ 127.0.0.1:8000/new-libro/

- [Autores](#)
- [Libros](#)

Título del libro:

Codigo:

- Book with this Codigo already exists.

Author:

Realizar como tarea el update, será similar.