(https://ctrlzblog.com)

Un Blog Sobre Django Y Desarrollo Web

Migraciones De Django: Cómo Agregar Campos No Anulables Sin Un Valor Predeterminado

Si desea crear una migración pero desea asignar un valor personalizado a cada una de las filas existentes, puede agregar Python personalizado a su migración.

bases de datos (https://ctrlzblog.com/tag/databases/) django (https://ctrlzblog.com/tag/django/) migraciones (https://ctrlzblog.com/tag/migrations/) modelos (https://ctrlzblog.com/tag/models/)

En esta publicación, veremos cómo agregar un campo no anulable y manejar las filas existentes. En lugar de proporcionar un valor predeterminado o forzar a la columna a aceptar valores nulos, agregaremos automáticamente valores a las filas existentes, donde cada valor es específico de esa fila.

Si todavía está ganando confianza en el manejo de migraciones, le recomiendo leer mi guía para principiantes sobre migraciones (https://ctrlzblog.com/django-migrations-how-to-avoid-messing-up-your-database/) , donde aprenderá cómo evitar errores en la base de datos.

El problema

Quería agregar un campo slug a un modelo existente.

Es importante que no permita valores en blanco, ya que el slug se usará para las URL.

Pero cuando intenté ejecutarlo `python manage.py makemigrations`, recibí este error:

```
It is impossible to add a non-nullable field 'slug' to movie without
specifying a default. This is because the database needs something to
populate existing rows.
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with
a null value for this column)
 2) Quit and manually define a default value in models.py.
```

Estas son mis opciones:

1. Asignar un valor predeterminado arbitrario

Preferiría no hacerlo, ya que estaría asignando deliberadamente valores no válidos a mis datos existentes. Podría ejecutar un script después para asignar el slug correcto a cada película

existente, pero prefiero no tener el paso adicional. Creo confusión si está intentando configurar el proyecto en otra máquina.

2. Permitir valores en blanco

He hecho esto varias veces en proyectos personales como una solución rápida, pero no es lo ideal. Si veo `blank=True` en un modelo, eso me dice que los valores en blanco son aceptables. Los slug se utilizan para construir URL, por lo que los valores en blanco definitivamente no son aceptables en este caso.

3. Suelta la base de datos

Esto es algo que se debe evitar en la medida de lo posible. Eliminar el `db.sqlite3` archivo, volver a ejecutar sus migraciones y recrear todos sus datos puede parecer tentador cuando está en un lío, pero se necesita mucho esfuerzo para que el contenido de su base de datos vuelva al mismo estado en que estaba antes. Una vez que su producto se haya implementado, descartar la base de datos nunca será una opción, por lo que le recomiendo que aprenda a evitar esto más temprano que tarde.

4. Personaliza tu migración

Aquí es donde agrega código a su migración para asignar el valor correcto para cada fila existente. Si está trabajando con otro desarrollador o en varias máquinas, este enfoque

obtendrá los resultados que necesita en un solo paso, sin ejecutar scripts adicionales.

Cómo hacemos esto?

Vamos a asignar temporalmente un valor predeterminado arbitrario.

Luego vamos a agregar una operación a nuestra migración que pasará por cada una de nuestras filas existentes y actualizará el nuevo campo slug con el valor correcto.

Empecemos.

Cómo personalizar una migración

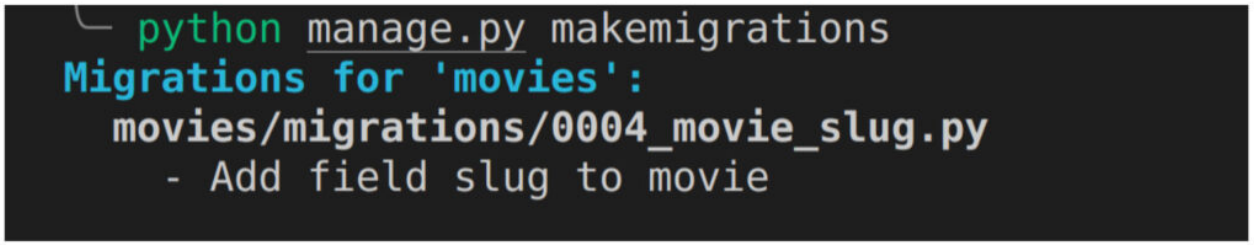
1. Crea tu migración

Empieza por correr `python manage.py makemigration`

Cuando reciba el siguiente mensaje, presione 1:

```
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with
a null value for this column)
 2) Quit and manually define a default value in models.py.
```

Proporcione cualquier valor. Como la columna que estoy agregando es una cadena, elegí “abc”. Esto le dará a Django lo que necesita para crear la migración. Vamos a personalizar la migración antes de ejecutarla. Nuestro código de Python sobrescribirá cualquier columna con el "abc" valor predeterminado.



migration created with default "abc"

2. Personaliza la migración

Por ahora, he establecido un valor predeterminado `"abc"` que no quiero que se aplique a la base de datos. Hice esto para apaciguar a Django para que cree la migración, pero nos aseguraremos de `"abc"` que no se aplique.

Las migraciones de Django contienen una lista llamada `operations`. La migración se crea con una única operación: `migrations.AlterField`. Lo que vamos a hacer es agregar una segunda operación que revisará cada una de nuestras películas, calculará el slug correcto del título de la película y actualizará la película. Las operaciones se ejecutan en orden, por lo que actualizaremos el slug de cada película una vez que se haya creado el campo.

Escribe el código personalizado

Necesitamos escribir dos funciones: una que actualizará la película con nuestro slug deseado y otra que restaurará el slug al valor predeterminado.

Hacemos esto porque las migraciones siempre deben ser reversibles. Cuando ejecutemos la migración, `forwards` se llamará. Si alguna vez necesitamos revertir la migración, `backwards` se llama.

Agregar la operación

Agregue lo siguiente a `operations`:

```
migrations.RunPython(forwards, backwards)
```

Consulte la esencia a continuación para ver el código completo de la migración:

```
# Generated by Django 4.0.5 on 2022-06-04 18:08

import autoslug.fields
from django.db import migrations
from django.utils.text import slugify

DEFAULT = "abc"


def forwards(apps, _):
    Movie = apps.get_model("movies", "Movie")
    movies = Movie.objects.all()
    for movie in movies:
        movie.slug = slugify(movie.title)
        movie.save()

```

```
    Movie = apps.get_model("movies", "Movie")
    movies = Movie.objects.all()
    for movie in movies:
        movie.slug = DEFAULT
        movie.save()


class Migration(migrations.Migration):

    dependencies = [
        ('movies', '0003_alter_movie_popularity'),
    ]

    operations = [
        migrations.AddField(
            model_name='movie',
            name='slug',
            field=autoslug.fields.AutoSlugField(
                always_update=True, default=DEFAULT, editable=True, populate_from='title'),
            preserve_default=False,
        ),
        migrations.RunPython(forwards, backwards)
    ]
```

0004_mo view raw (https://gist.github.com/aliceridgway/4af16fe3b45c498bcb964c6392159e8a/raw/b11e0dc209662b1ae7926dd60b9be9b0d86837ff/0004_movie_slug.py)
vie_slug.py (https://gist.github.com/aliceridgway/4af16fe3b45c498bcb964c6392159e8a#file-0004_movie_slug-py) hosted with ❤ by GitHub (https://github.com)

3. Ejecute la migración

La migración ya está lista para ejecutarse. Ejecute `python manage.py migrate` en su

Resultado

Vaya a admin para verificar que el valor correcto para el nuevo campo se haya aplicado a las filas existentes.

Conclusión

Hemos demostrado que no tiene que abandonar su base de datos ni comprometer los valores predeterminados cuando desea agregar un campo que no admite valores NULL a uno de sus modelos.

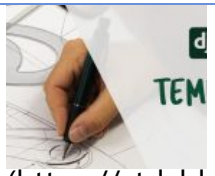
Al comprender cómo las migraciones de Django ejecutan las operaciones en orden, podemos usarlas `migrations.RunPython(forwards, backwards)` para agregar código personalizado. Esto es útil cuando queremos agregar un campo y darle a cada fila existente un valor personalizado.

Artículos Relacionados

(https://ctrlzblog.com/how-to-add-a-non-nullable-foreign-key/)

Cómo hacer que un campo de clave externa no sea anulable con un valor predeterminado (https://ctrlzblog.com/how-to-add-a-non-nullable-foreign-key/)

Las claves externas que no aceptan valores NULL requieren el ID del objeto predeterminado. Aprenda a crear una migración personalizada para crear una predeterminada.



(https://ctrlzblog.com/how-to-create-a-base-template-for-your-django-project/)

Cómo crear una plantilla base para tu proyecto Django (https://ctrlzblog.com/how-to-create-a-base-template-for-your-django-project/)

Las plantillas de Django almacenan el HTML para sus proyectos. Django usa su propio lenguaje de plantillas para permitirle agregar datos desde el back-end a su

(https://ctrlzblog.com/django-foreign-key-example-how-to-add-categories-to-a-blog/)

Ejemplo de clave externa de Django: cómo agregar categorías a un blog (https://ctrlzblog.com/django-foreign-key-example-how-to-add-categories-to-a-blog/)

Esta publicación brinda un ejemplo de cómo agregar claves externas a su proyecto. Usando publicaciones de blog y categorías como nuestro ejemplo, agregaremos una clave externa y permitiremos a los usuarios ver las publicaciones por categoría.

(https://ctrlzblog.com/how-to-do-crud-with-django-using-function-based-views/)

Cómo hacer CRUD con Django usando vistas basadas en funciones (https://ctrlzblog.com/how-to-do-crud-with-django-using-function-based-views/)

Aprender a procesar datos CRUD es una habilidad fundamental para cualquier desarrollador web. CRUD significa Crear, Leer, Actualizar, Eliminar. En este tutorial, estoy

(https://ctrlzblog.com/how-to-add-tags-to-your-blog-a-django-manytomanyfield-

Cómo agregar etiquetas a su blog (un ejemplo de Django ManyToManyField) (https://ctrlzblog.com/how-to-add-tags-to-your-blog-a-django-manytomanyfield-

(https://ctrlzblog.com/4-types-of-model-methods-django-beginners-should-know-

4 tipos de métodos modelo que los principiantes de Django deben conocer (https://ctrlzblog.com/4-types-of-model-methods-django-beginners-should-know-

manytomanyfield-example/)

En este tutorial, le mostraré cómo agregar etiquetas al blog básico de Django usando ManyToManyField. Los blogs son un proyecto común

django-beginners-should-know-about/)

Potencie sus clases modelo agregando métodos.

(https://ctrlzblog.com/django-migrations-how-to-avoid-messing-up-your-database/)

Migraciones de Django: cómo evitar estropear su base de datos (https://ctrlzblog.com/django-migrations-how-to-avoid-messing-up-your-database/)

Comprender cómo funcionan las migraciones en Django es clave para evitar errores difíciles de corregir.

(https://ctrlzblog.com/django-models-how-to-automatically-populate-slug-fields-for-urls/)

Modelos de Django: cómo completar automáticamente los campos de slug para las URL (https://ctrlzblog.com/django-models-how-to-automatically-populate-slug-fields-for-urls/)

Los campos de slug dependen de que el usuario escriba el slug ellos mismos. Aprende a llenar el slug automáticamente desde un título.

(https://ctrlzblog.com/how-to-test-django-models-with-examples/)

Cómo probar modelos de Django (con ejemplos) (https://ctrlzblog.com/how-to-test-django-models-with-examples/)

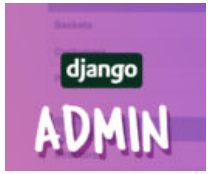
Aprenda a probar modelos de Django.

(https://ctrlzblog.com/a-beginners-guide-to-unit-testing-in-django/)

Una guía para principiantes sobre pruebas unitarias en Django (https://ctrlzblog.com/a-beginners-guide-to-unit-testing-in-django/)

Escribir pruebas requiere esfuerzo, pero le ahorrará tiempo a largo plazo.

Cómo aprender Django gratis en 2022 (https://ctrlzblog.com/how-to-learn-django-for-free-in-2022/)



Cómo comenzar con la administración de Django (https://ctrlzblog.com/how-to-set-up-django-admin/)

django-for-free-in-2022/) recomendados para todos los niveles.

django-admin/) personalización básica del panel de administración, aprenda cómo comenzar con Django Admin.

(https://twitter.com/aliceridgway404)