Relaciones N:M

DigitalHouse >
Coding School

Índice

“

Las relaciones en Sequelize existen para optimizar la obtención de datos en una consulta a la base de datos.

”

1 .belongsToMany()

.belongsToMany()

Hay veces donde una tabla posee una relación de muchos a muchos. Es decir, muchos registros de nuestra tabla pueden estar relacionados con varios de otra. Para poder aclararle esta relación a Sequelize debemos utilizar el método `.belongsToMany()`.

Para definir la relación, `.belongsToMany()` recibe dos parámetros: el primero es **el modelo con el que queremos relacionarlo** (llamándolo a través del parámetro que contiene nuestros modelos) y el segundo es un **objeto donde debemos detallar la relación.**

.belongsToMany()

{}

```
Pelicula.associate = function(modelos){
   Pelicula.belongsToMany(modelos.Actores, {

     as: "actores",

     through: "actor_movie",
     foreignKey: "genre_id",
     otherKey: "actor_id",
     timestamps: false
     });
}
```

Asignamos un alias con el que llamaremos luego a la relación.

.belongsToMany()

{}

```
Pelicula.associate = function(modelos){
   Pelicula.belongsToMany(modelos.Actores, {
     as: "actores",

     through: "actor_movie",
     foreignKey: "genre_id",
     otherKey: "actor_id",
     timestamps: false
     });
}
```

Nombre de la tabla intermedia que utilizamos para la relación.

.belongsToMany()

{}

```
Pelicula.associate = function(modelos){
   Pelicula.belongsToMany(modelos.Actores, {
     as: "actores",
     through: "actor_movie",
     foreignKey: "genre_id",
     otherKey: "actor_id",
     timestamps: false
     });
}
```

Referencia a la tabla del modelo que estamos referenciando.

.belongsToMany()

{}

```
Pelicula.associate = function(modelos){
   Pelicula.belongsToMany(modelos.Actores, {
     as: "actores",
     through: "actor_movie",
     foreignKey: "genre_id",
     otherKey: "actor_id",
     timestamps: false
     });
}
```

Referencia a la otra tabla que queremos referenciar.

.belongsToMany()

{}

```
Pelicula.associate = function(modelos){
   Pelicula.belongsToMany(modelos.Actores, {
     as: "actores",
     through: "actor_movie",
     foreignKey: "genre_id",
     otherKey: "actor_id",

     timestamps: false

     });
}
```

Aclaramos si la tabla pivot posee o no timestamps.

2 | Implementación

Implementación

Las consultas a la base de datos suelen recibir un objeto como parámetro.

Dentro de ese objeto deberíamos agregar un atributo **include** que recibe un array de objetos.

Cada objeto de ese array representa una relación. Debemos aclarar cuál queremos usar.

Para definir la relación creamos un atributo **association** que contenga el mismo alias que usamos en el modelo para llamar a la relación.

Implementación

{}

```
const db = require('../database/models');
db.Peliculas.findAll({
 include: [
 {association: "genero"},
 {association: "actores"}
 ]
}).then(resultados=>{
        console.log(resultados);
   })
```

Incluimos las asociaciones (relaciones) que creamos en los modelos.

En la vista

Para poder utilizar esta información en la vista, debemos llamar a las asociaciones a través de los atributos del modelo. Estos poseen el mismo nombre del alias que establecimos cuando creamos la relación.

EJS

```
// Muestro el género al que pertenece la película
<%= pelicula.genero %>
   // Muestro los actores de la película
   <% for (let i = 0; i < pelicula.actores.length; i++) { %>
       <%= pelicula.actores[i] %>
   <% } %>
```

¡ATENCIÓN!

Siempre que creamos una relación desde un modelo, debemos generar la misma desde el otro con quien está relacionado. Si no lo hacemos, Sequelize no reconoce la asociación.

Documentación

Para saber más podemos acceder a la documentación oficial de Sequelize haciendo clic en el siguiente: link.


DigitalHouse>
Coding School