Der QueryBuilder in TypeORM ist ein leistungsstarkes Werkzeug, um komplexe Datenbankabfragen zu erstellen und auszuführen. Mit dem QueryBuilder können Sie Abfragen schrittweise und programmatisch aufbauen, was ihn flexibler und oft leserlicher als reine SQL-Strings macht.

Hier ist ein einfacher Leitfaden, wie Sie den QueryBuilder in einem NestJS-Projekt mit TypeORM verwenden können:

1. **Einbindung des EntityManager oder Repository**: Normalerweise verwenden Sie den QueryBuilder über ein Repository oder den EntityManager. Wenn Sie ein Repository verwenden, sieht der Zugriff auf den QueryBuilder etwa so aus:

typescript

this.repository.createQueryBuilder('entityAlias');

Wenn Sie den EntityManager verwenden:

typescript

 this.entityManager.createQueryBuilder(EntityClass, 'entityAlias');

 **Grundlegende Verwendung**:

Angenommen, Sie haben eine User-Entität und möchten alle Benutzer abrufen, deren Name "John" ist:

typescript

 const users = await this.repository

.createQueryBuilder('user')

.where('user.name = :name', { name: 'John' })

.getMany();

 **Join-Operationen**:

Nehmen wir an, Sie haben eine Post-Entität, die mit der User-Entität verknüpft ist und Sie möchten alle Beiträge eines bestimmten Benutzers abrufen:

typescript

 const posts = await this.repository

.createQueryBuilder('post')

.innerJoinAndSelect('post.user', 'user', 'user.name = :name', { name: 'John' })

.getMany();

 **Aggregatfunktionen**:

Wenn Sie die Anzahl der Benutzer in Ihrer Datenbank abrufen möchten:

typescript

 const count = await this.repository

.createQueryBuilder('user')

.getCount();

 **Sortieren und Paginieren**:

Wenn Sie Benutzer in absteigender Reihenfolge nach ihrem Erstellungsdatum sortieren und die ersten 10 abrufen möchten:

typescript

 const users = await this.repository

.createQueryBuilder('user')

.orderBy('user.creationDate', 'DESC')

.take(10)

.getMany();

 **Update und Delete**:

Um den Namen eines Benutzers zu aktualisieren:

typescript

await this.repository

.createQueryBuilder()

.update(User)

.set({ name: 'UpdatedName' })

.where('id = :id', { id: 1 })

.execute();

Um einen Benutzer zu löschen:

typescript

1. await this.repository
2. .createQueryBuilder()
3. .delete()
4. .from(User)
5. .where('id = :id', { id: 1 })
6. .execute();

Der QueryBuilder bietet viele weitere Funktionen und Möglichkeiten, z. B. für komplexe Bedingungen, Unterabfragen und vieles mehr. Der Schlüssel liegt darin, zu verstehen, wie Sie den Builder schrittweise aufbauen können, um die gewünschte Abfrage zu erhalten. Es ist auch empfehlenswert, die offizielle [TypeORM-Dokumentation](https://typeorm.io/#/) zu konsultieren, um eine vollständige Übersicht und Beispiele für die Verwendung des QueryBuilders zu erhalten