Version: v6 - stable

Model Querying - Finders

Finder methods are the ones that generate `SELECT` queries.

By default, the results of all finder methods are instances of the model class (as opposed to being just plain JavaScript objects). This means that after the database returns the results, Sequelize automatically wraps everything in proper instance objects. In a few cases, when there are too many results, this wrapping can be inefficient. To disable this wrapping and receive a plain response instead, pass `{ raw: true }` as an option to the finder method.

findAll

The `findAll` method is already known from the previous tutorial. It generates a standard `SELECT` query which will retrieve all entries from the table (unless restricted by something like a `where` clause, for example).

findByPk

The `findByPk` method obtains only a single entry from the table, using the provided primary key.

```
const project = await Project.findByPk(123);
if (project === null) {
  console.log('Not found!');
} else {
  console.log(project instanceof Project); // true
  // Its primary key is 123
}
```

findOne

The `findOne` method obtains the first entry it finds (that fulfills the optional query options, if provided).

```
const project = await Project.findOne({ where: { title: 'My Title' } });
if (project === null) {
  console.log('Not found!');
} else {
  console.log(project instanceof Project); // true
  console.log(project.title); // 'My Title'
}
```

findOrCreate

The method `findOrCreate` will create an entry in the table unless it can find one fulfilling the query options. In both cases, it will return an instance (either the found instance or the created instance) and a boolean indicating whether that instance was created or already existed.

The `where` option is considered for finding the entry, and the `defaults` option is used to define what must be created in case nothing was found. If the `defaults` do not contain values for every column, Sequelize will take the values given to `where` (if present).

Let's assume we have an empty database with a `User` model which has a `username` and a `job`.

```
const [user, created] = await User.findOrCreate({
  where: { username: 'sdepold' },
  defaults: {
    job: 'Technical Lead JavaScript'
  }
});
console.log(user.username); // 'sdepold'
console.log(user.job); // This may or may not be 'Technical Lead JavaScript'
console.log(created); // The boolean indicating whether this instance was just created
if (created) {
  console.log(user.job); // This will certainly be 'Technical Lead JavaScript'
}
```

findAndCountAll

The `findAndCountAll` method is a convenience method that combines `findAll` and `count`. This is useful when dealing with queries related to pagination where you want to retrieve data with a `limit` and `offset` but also need to know the total number of records that match the query.

When `group` is not provided, the `findAndCountAll` method returns an object with two properties:

- `count` - an integer - the total number records matching the query
- `rows` - an array of objects - the obtained records

When `group` is provided, the `findAndCountAll` method returns an object with two properties:

- `count` - an array of objects - contains the count in each group and the projected attributes
- `rows` - an array of objects - the obtained records

```
const { count, rows } = await Project.findAndCountAll({
  where: {
    title: {
      [Op.like]: 'foo%'
    }
  },
  offset: 10,
  limit: 2
});
```

```
console.log(count);
console.log(rows);
```

Edit this page

Last updated on ***Oct 13, 2022*** *by* ***Rik Smale***

Previous
« Model Querying - Basics

Next
Getters, Setters & Virtuals »