Version: v6 - stable

Association Scopes

This section concerns association scopes, which are similar but not the same as model scopes.

Association scopes can be placed both on the associated model (the target of the association) and on the through table for Many-to-Many relationships.

Concept

Similarly to how a model scope is automatically applied on the model static calls, such as `Model.scope('foo').findAll()`, an association scope is a rule (more precisely, a set of default attributes and options) that is automatically applied on instance calls from the model. Here, *instance calls* mean method calls that are called from an instance (rather than from the Model itself). Mixins are the main example of instance methods (`instance.getSomething`, `instance.setSomething`, `instance.addSomething` and `instance.createSomething`).

Association scopes behave just like model scopes, in the sense that both cause an automatic application of things like `where` clauses to finder calls; the difference being that instead of applying to static finder calls (which is the case for model scopes), the association scopes automatically apply to instance finder calls (such as mixins).

Example

A basic example of an association scope for the One-to-Many association between models `Foo` and `Bar` is shown below.

- Setup:

```
const Foo = sequelize.define('foo', { name: DataTypes.STRING });
const Bar = sequelize.define('bar', { status: DataTypes.STRING });
Foo.hasMany(Bar, {
    scope: {
        status: 'open'
    },
    as: 'openBars'
});
await sequelize.sync();
const myFoo = await Foo.create({ name: "My Foo" });
```

- After this setup, calling `myFoo.getOpenBars()` generates the following SQL:

```
SELECT
    `id`, `status`, `createdAt`, `updatedAt`, `fooId`
FROM `bars` AS `bar`
WHERE `bar`.`status` = 'open' AND `bar`.`fooId` = 1;
```

With this we can see that upon calling the `.getOpenBars()` mixin, the association scope `{ status: 'open' }` was automatically applied into the `WHERE` clause of the generated SQL.

Achieving the same behavior with standard scopes

We could have achieved the same behavior with standard scopes:

```
// Foo.hasMany(Bar, {
//     scope: {
//         status: 'open'
//     },
//     as: 'openBars'
// });

Bar.addScope('open', {
    where: {
        status: 'open'
    }
});
Foo.hasMany(Bar);
Foo.hasMany(Bar.scope('open'), { as: 'openBars' });
```

With the above code, `myFoo.getOpenBars()` yields the same SQL shown above.

Edit this page

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