# Sequelize Relations

## Objectives

- To understand the concept of relations using Sequelize.
- To understand the "include" option for performing joins with our queries.
- To read the fantastic Sequelize manual
- To create a blog app with a content management system that can be used to update it's data

## Demo: Update & Delete

- $ node 10-Sequelize-Update-Delete/server.js
- Note: we can delete todo items by clicking the x button on the todo. Refresh!
- Note: when clicking a todo item, you can update the todos text. After editing, hit "Enter" to finish editing, or click anywhere else on the page to cancel editing.
- Also note: how clicking the check mark toggles a todo's complete property.

## Activity: Update & Delete (20 min)

See 10-Sequelize-Update-Delete

## Review: Update & Delete

Note: .destroy() and .update() in api-routes.js

Small flaw

It’s possible to save a todo with empty text or even one with a null text value.

## Activity: There Must Be a Better Way!

With a partner, discuss:

- What steps might take on the back end to test what we're receiving?
- What options might we have to further restrict what kinds of data can be saved into our Todos table?

validate

RTFM http://docs.sequelizejs.com/manual/tutorial/models-definition.html#validations

## Activity: Validations (15 min)

See 11-Sequelize-Validations

## Review: Validations

- { force: true }
  - This means that whenever we sync our database (whenever we start our app), we want to drop our tables and recreate them with any updated schemas.
  - This is useful during the development process when we're experimenting with our database structure. We'd want to remove this before deploying to a production environment. (otherwise we'd lose all of our data whenever our app starts)

## Reverse Engineering

http://lmgtfy.com/?q=reverse+engineering

## Stepping backwards through someone else’s thought process

## RTFM

- .blur && .on(‘blur’...
- .on(‘keyup’...
-

## Demo: POST Model

You will be setting up much of the back-end for this blogging Content Management System app

We can create, delete, and edit posts.

Additionally we can select which categories of posts we want to see by using the drop down on the blog page

## Activity: POST Model (15 min)

See 13-POST-Model

## Review: POST Model

- post.js
    - Note  the validations and flags in the models

## What is the difference between DataTypes.STRING and DataTypes.TEXT?

STRING is the equivalent of varchar (255) in MySQL. Useful for storing relatively small values.

TEXT is a virtually unlimited amount of storage for characters. We might use this if we needed to store something larger or of unknown size.

## Activity: CRUD

See 14-Blog-CRUD

## Review: CRUD

where?

## Demo: Post Author Relationships

Notice our app functionality has changed some

- Removed categories; added a second model, Authors.
- When we first start the app and try and create a new post, we're directed to the author-manager.html page, where we must first create an Author.
    a. After creating an Author, we now have the option to create a new Post for that Author
    b. After creating a new Post, we are redirected to the blog page where we see the Author's name by that Post.
- localhost:8080/api/posts

## Activity: Discuss Relations (5 min)

See Slack for instructions

| id | body | title |
|---|---|---|
| 1 | Lorem ipsum... | My Lorem Ipsum |

| id | name |
|---|---|
| 1 | John Doe |

How can we restructure our database tables if we wanted the ability to form a relationship between a Post and an Author?

## foreign key

Allows us to have multiple Posts all pointing to the same Author

Which table would have a foreign key and why?

What would that look like?

| id | body | title | AuthorId |
|---|---|---|---|
| 1 | Lorem ipsum... | My Lorem Ipsum | 1 |

RTFM https://www.w3schools.com/sql/sql_foreignkey.asp

## Demo: Associations

What we went over in the last activity is considered a **belongsTo** association, as well as a **hasMany** association in Sequelize.

These are some of the most common types of associations.

- A Post **belongsTo** an Author
- An Author **hasMany** posts

RTFM http://docs.sequelizejs.com/manual/tutorial/associations.html

## Activity: Associations

See 14-POST-Author-Association

## Review: Associations

- Inside Post.associate() we run the Post.belongsTo method and pass it our Author model
- We're adding a flag to our foreign key (AuthorId) saying this cannot be null. In other words, it won't let us create a Post without an Author.
- We associate our Author model with Posts and specify that it hasMany

RTFM http://docs.sequelizejs.com/en/latest/docs/associations/

## Activity: Joins (15 min)

See 15-Post-Author-Joins

## Review: Joins

- By just adding include: [<models>] as an option in our query we can easily get the associated data
- When we navigate to localhost:8080/api/authors we get all of the author data with their Posts attached.
    a. The same for localhost:808/api/posts and note how the same is true in the reverse.
- In MySQL, this is what's known as a "left outer join". We can do others with Sequelize, but this gives us very convenient access to both pieces of associated data.

TrilogyTV Sequelize + Handlebars Review Video