**Official sequelize link -----** [**https://sequelize.org/**](https://sequelize.org/)

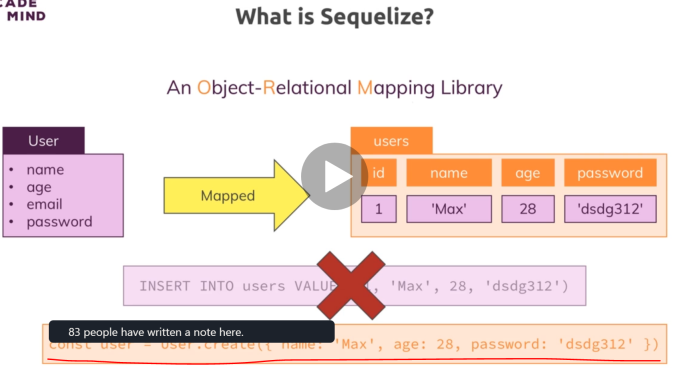
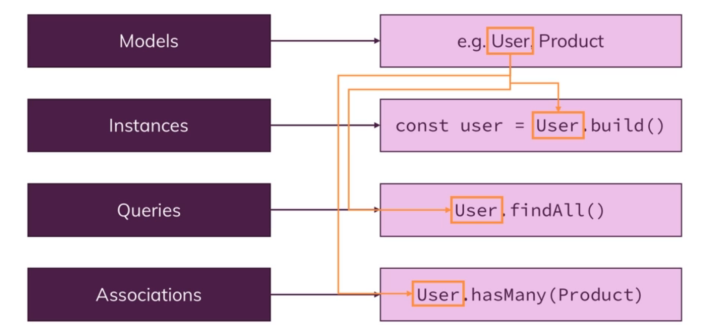
**To understands methods and how it works in sequelize**

* **Sequelize as mysql**

Sequelize is nothing but forming a crud (create, read, update, delete) with the third party package

So we do connection with database by sequelize.

So sequelize help us to ease our work **with seqelize we do not have to write sql query commands because it saves in third party package so we can write our code in javaScript** so our js code with sequelize intract with mysql without writing the sql code

So we can interact with our model like user above with our normal js code.

**Installing sequelize**

Before installing sequelize make shore that you had install Mysql in your dependencies in npm



After that instead of using mysql and export now we export our sequelize so do not have use sql query command instead use js commands for that.

// first we have to import the sequelize class

const Sequelize = require('sequelize');

// first three argument as (schema/dataBase name,user name of mysql, password of mysql)

const sequelize = new Sequelize('product-data','root','Abcd@1234',{

    dialect: 'mysql', // defines name of sql in your system

    host: 'localhost'

})

module.exports = sequelize;

**so after that we change our modules because that is where we want to interact with our data from Database so we use that above export to defines the path of our database and use to store values in it by modules**

// requires sequelize to use sequelize methods

const Sequelize = require('sequelize');

// getting our database path to store data in it

const sequelize= require('../util/dataBase');

// defining the data from dataBase so can be used by product

const product = sequelize.define('product',{

  id: {

    type: Sequelize.INTEGER,

    autoIncrement: true,

    allowNull: false,

    primaryKey: true

  },

  title: Sequelize.STRING,

  price: {

    type: Sequelize.DOUBLE,

    allowNull: false

  },

  imageUrl: {

    type: Sequelize.STRING,

    allowNull: false

  },

  description: {

    type: Sequelize.STRING,

    allowNull: false

  }

});

module.exports = product;

so we are creating the data in form of object in js and it will store in database by sequleize

because sequelize convert our js code into mysql formate.

But how do we create a table into our database we can simply do it by calling a function which can listen all the defines of sequelize and we re doing it on our app.js file where we are listening our app

sequelize.sync()

.then(result => {

    console.log(result);

    app.listen(3000);

})

.catch(err => console.log(err))

So it will listen only when we connected to the database

Now it is the time to use controller so we do have to use controller so it can saves data and fetch data as well so for that first just improvise our add - product controller in that we have to pass our details in object format which is getting by user side and save into mysql data base so that we can fetch it when nedded

So first we have to call our model file which handles the product data in our controller page and call **create** method in it which will save the data instantly when called

exports.postAddProduct = (req, res, next) => {

  const title = req.body.title;

  const imageUrl = req.body.imageUrl;

  const price = req.body.price;

  const description = req.body.description;

  Product.create({

    title: title,

    price: price,

    imageUrl: imageUrl,

    description: description

  })

  .then(result => console.log(result))

  .catch(err => console.log(err))

};

And in our shop controller we just have to replace our fetchAll method with findAll method which is called as findAll() in sequelize

// fetchAll is used as findAll in sequelize

exports.getProducts = (req, res, next) => {

  Product.findAll()

  .then(products => {

    res.render('shop/product-list', {

      prods: products,

      pageTitle: 'All Products',

      path: '/products'

    });

  })

  .catch(err => console.log(err))

};

So we just getting our product data from mysql database

And if we want to get detail of product we have to use the product id and we can pass that id in sequelize function which is **findByPk()** replacement of findBIid

We can also use where inside findAll() method to get data by particular object

 Product.findAll({ where: {id: prodId} })

  .then(product => {

    res.render('shop/product-detail', {

      product: product[0],

      pageTitle: product[0].title,

      path: '/products'

    });

  })

  .catch(err => console.log(err))

};

**Or we can also use in place**

  const prodId = req.params.productId;

  Product.findByPk(prodId)

  .then(product => {

    res.render('shop/product-detail', {

      product: product,

      pageTitle: product.title,

      path: '/products'

    });

  })

We now have to change our remaining functions and methods in admin and shop js

So when admin wants to edit product it will get the previous data when press edit and can update the data as well so we have to **use geteditproduct** and **post edit product** functions

We can delete the product by destroy method which will destroy our data from database

exports.postDeleteProduct = (req, res, next) => {

  const prodId = req.body.productId;

  Product.findByPk(prodId)

  .then(product => {

**return product.destroy()**// delete the data from database

  })

  .then (() => {

    console.log('product deleted')

    res.redirect('/admin/products')

  })

  .catch(err => console.log(err)) // it will run after return only

};

You can also use where operations in destroy instead of using findByPk()

* Creating user Module

There is no use of database if user is not connected with it for that we have to create a user module which will connected by database throw database path

const Sequelize = require('sequelize');

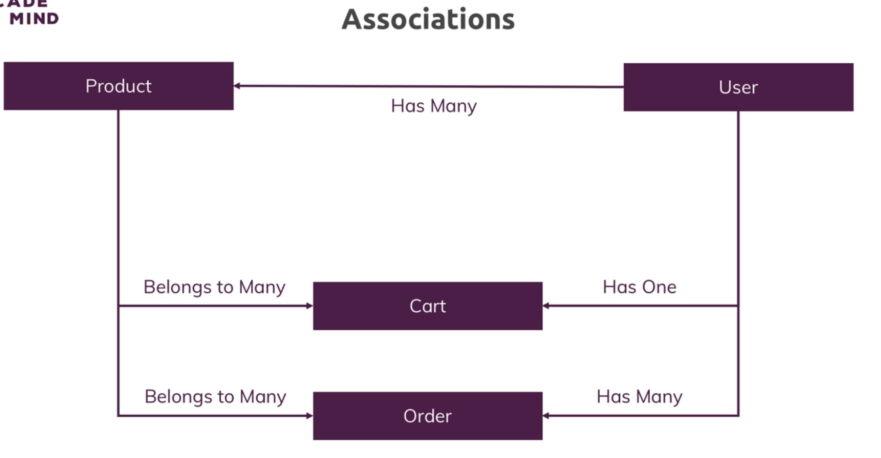
const sequelize = require('../util/dataBase') // database path

const user = sequelize.define('user', {

here we define id ,name,email and then export that

* **ASSOSIATIONS in sequelize**

Association is the relation between the modules in data base so their may me many relation



So the product belongs to many cart and many orders as well and the user have only one cart but can order as many orders

**There is lot more in the linked page about associations** <https://sequelize.org/master/manual/assocs.html>

So we have to call our product and user and defines the relation between them in our app we can do it by just give the relation before sync

product.belongsTo(user, {constraints: true, onDelete: 'CASCADE'})

user.hasMany(product);

the first one will set the relation of user as a foreign key so when user deleted the product has to be deleted it is done by on Delete ‘cascade command and user has many product

after that we have to force our squelizer to add the relations and we can do it by

sequelize.sync({ force: true })

but we also want the user authentication because user is the one who have relation with product but we will do the authentication later for now we just create a dummy user

sequelize.sync()

.then(() => {

    return user.findByPk(1) // check wether their is any user because first user have 1 id

})

.then(User => {

    if(!User) {

        return user.create({ name:'jay', Email:'jay12@gamil.com'})

    }

    return User

})

.then((user) => {

    console.log(user)

    app.listen(3000);

})

.catch(err => console.log(err))

So the program run and find the primary key of 1 to check is their any user an if not then it will create a user and then listen to the server

app.use((req,res,next)=> {

    user.findByPk(1) // we always het user in middleware because we created user down at the time of sync

    .then(User => {

        req.user = User;

        next();

    })

    .catch(err => console.log(err));

and we also called user .findByPk() in middleware because when we listen we created the user if it is not present so their fore middleware of user always run

* **Changing admin js with after the user relation**

exports.postAddProduct = (req, res, next) => {

  const title = req.body.title;

  const imageUrl = req.body.imageUrl;

  const price = req.body.price;

  const description = req.body.description;

  req.user.createProduct({ // we are using that relation of user with product

    title: title,

    price: price,

    imageUrl: imageUrl,

    description: description

  })

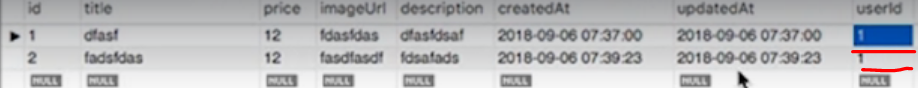
  .then(() => {

    console.log('product added')

    res.redirect('/admin/products')

  })

So as we are creating a user and setting the relation of it with user has many so now instead of creating product by **Product.create** we will now create it by **user.createProduct** which will create a createProduct function for us because our name of product is Product



Now we also change some of the function of admin where we want that user products such as get the edit product from user and get all product from user

* Now **we create cart and cartItem separately**

we have to create a cart and cart item separately because user have to interact with cart and so cart will interact with many products due to association

and the relation between product, cart, user, cart item is

// this will create user id as foreign key in cart table

user.hasOne(cart);

cart.belongsTo(user);

// here many to many relation so we have to store both the id specifically using through in table

cart.belongsToMany(product, { through: cartItem});

product.belongsToMany(cart, { through: cartItem});

so the cart item weill take the id’s of both cart as well as product

the use of these relation is that we can call cart as method from req.user in run time because cart contains the user id as foreign key so we can call it by that user

after seting realtion we have to set the cart page which using msql database

exports.postCart = (req, res, next) => {

  const prodId = req.body.productId;

  let fetchedCart;

  let newQuantity = 1;

  req.user.getCart() // first get the cart-items as relation of user and cart

  .then(cart => {

    fetchedCart = cart

    return cart.getProducts({ where: {id: prodId } })

    //then return the product contains particular id

  })

  .then(products => {

    let product;

    if(products.length > 0) {

      product = products[0]; // get the data part of that product

    }

    if(product){ // get old quantity if product exist

      const oldQuantity = product.cartItem.quantity;

      newQuantity = oldQuantity + 1;

      return product;

    }

    return Product.findByPk(prodId);

  })

  .then(product => { // calling add product which will add in database

    return fetchedCart.addProduct(product, { through: { quantity: newQuantity }})

  })

  .then(() => {

    res.redirect('/cart');

  })

  .catch(err => console.log(err));

};

Now we create our order js code in which we want to connect our order in data base throw order item which we can get our order item through that order id we can do it by set relation

// relation of order with all in database

order.belongsTo(user);

user.hasMany(order);

order.belongsToMany(product, { through: orderItems });

so it will create creates a relation between order and all the above in which order contains the user id in which it will act and order item contains order id and product id as foreign key so they can connect them too. Now we add a order now by adding product in our order by cart page

exports.postOrder = (req,res,next) => {

  let fetchedCart;

  req.user.getCart()

  .then(cart => {

    fetchedCart = cart;

    return cart.getProducts()

  })

  .then(products => {

    return req.user.createOrder()

           .then(order => {

             return order.addProducts(products.map(product => {

               // get the order item from database table add quantity one by one to all product

               product.orderItem = { quantity: product.cartItem.quantity }

               return product

             }))

           })

           .catch(err => console.log(err));

  })

  .then(() => {

    return fetchedCart.setProducts(null); // remove cart products

  })

  .then(() => {

      res.redirect('/orders');

  })

  .catch(err => console.log(err));

}

// creates an order details passing products by include due to relation

exports.getOrders = (req, res, next) => {

  // due to association we can call it in our ejs file

  req.user.getOrders({ include: ['products']})

  .then(orders => {

    res.render('shop/orders', {

      path: '/orders',

      pageTitle: 'Your Orders',

      orders: orders

    });

  })

};

so when ever we click a button order now in our cart page we want add items in our order page and redirect to order page for that we have to just call cart first and from that we take all the product and then we create a order in which we are going to save our products by adding quantity one by one in all the products

the we have to if we wand to delete our cart because there is no use of products in cart if we ordered our product so we can do it by setting our fetched cart to null

and then we can go to our orders page their we can create it in ejs by passing products as an array of containing all list of product in cart this can be done by include due to their relations