**Node JS with ORM:**

(Object Relational Mapping).

**Installations:**

Create new folder and run **npm init** for packages.json file.

**Install Express:**

Npm I express.

**Install Sequelize:**

npm i sequelize

**Install mysql2:**

Npm I mysql2.

**Connections to the Database:**

1: Create **dbConfig.js** file in b **config** folder.

module.exports = {

    HOST: 'localhost',

    USER: 'root',

    PASSWORD :  '',

    DB: 'node-orm',

    dialect: 'mysql',

    pool:{

        max:5,

        min:0,

        acquire: 30000,

        idle: 10000

    }

    //this pool tell the maximum and minimum connections and idel is times for move

    //from one connection to other and this pool part is not mush important.

}

2: create **index.js** file in **models** folder and use **dbConfig** file in this file.

const dbConfig = require("../config/dbConfig.js");

const {Sequelize , DataTypes} = require("sequelize");

const sequelize = new Sequelize(

    dbConfig.DB,

    dbConfig.USER,

    dbConfig.PASSWORD,

    {

        host:dbConfig.HOST,

        dialect : dbConfig.dialect,

        operatorsAliases:false,

        pool:{

            max:dbConfig.pool.max,

            min:dbConfig.pool.min,

            acquire:dbConfig.pool.acquire,

            idle:dbConfig.pool.idle

        }

    }

)

**Authenticate the Database Connection in Models.**

//for authenticate

sequelize.authenticate().then(()=>{

    console.log("connected...")

})

.catch(err =>{

    console.log("db error .. " + err);

})

**DB configurations in model index.js file after authentications:**

const db = {};

db.Sequelize = Sequelize;

db.sequelize = sequelize;

//db.users = require("./users")(sequelize , DataTypes);

db.products = require("./productModel.js")(sequelize , DataTypes);

db.reviews = require("./reviewsModel.js")(sequelize , DataTypes);

db.sequelize.sync({force: false}).then(()=>{

    console.log("yes re-sync is done");

})

module.exports = db;

**How can we define our own model.**

1: create new file in **Model** folder.

module.exports = (sequelize, DataTypes) =>{

    const Product  =sequelize.define("product", {

        title:{

            type: DataTypes.STRING,

            allowNull: false,

            unique:true,

        },

        price:{

            type:DataTypes.INTEGER

        },

        description:{

            type:DataTypes.TEXT

        },

        published:{

            type:DataTypes.BOOLEAN

        }

    })

    return Product

}

**Create Controllers:**

1: Create new folder with name controllers and then create new file in

This folder.

2: Import db from relevant models **index.js** in controller file.

const db = require('../models');

3: Import model that we declare in our models **index.js file**.

const User = db.users;

4: create functions like add, update, delete whatever you want and use

Upper **User** variable that store our **user model**.

const addUser = async (req , res) =>{

    let data = await User.create({name:'Usman', email:'usman@gmail.com', address:'Okara', status:1})

    let response = {

        data:data

    };

    res.status(200).send(response);

}

5: Export that created functions:

module.exports = {

    addUser

}

Complete code of controller.

const db = require('../models');

const User = db.users;

const addUser = async (req , res) =>{

    let data = await User.create({name:'Usman', email:'usman@gmail.com', address:'Okara', status:1})

    let response = {

        data:data

    };

    res.status(200).send(response);

}

module.exports = {

    addUser

}

**Create Routing file:**

1: Create new folder with name routes and new file with name routes.

2: Import relevant controller in routing files for make routes of that

Controller.

const userCtl = require('../controllers/userController.js');

3: Import express route for routing.

const router = require('express').Router();

4: Create route with specific type(get, post, put, delete etc).

router.get('/add', userCtl.addUser);

5: Export routes:

module.exports = router;

**Sequelize Queries:**

const db = require ("../models");

//create our main model

const Product = db.products;

here we import db from models index.js file.

**Add Record:**

const addProduct = async(req , res)=>{

    //information taken from body

    let info = {

        title: req.body.title,

        price: req.body.price,

        desciption: req.body.desciption,

        published : req.body.published ? req.body.published : false

    }

    //query for create

    const product = await Product.create(info);

    //set resonse status

    res.status(200).send(product);

    console.log(product);

}

**Add record Using build():**

let data = await Product.build({title:"Product2" , price:15 , desciption : "Product2 desciption" , published:true});

await data.save();

**Get All Records:**

const getAllProducts = async (req , res) =>{

    let products = await Product.findAll({});

    res.send(products);

    res.status(200).send(products);

}

**Get One Record:**

const getOneProduct = async (req , res)=>{

    let id = req.params.id;

    let product = await Product.findOne({where : { id: id } })

    res.status(200).send(product);

}

**Update Record:**

const updateProduct = async (req , res)=>{

    let id = req.params.id;

    const product = await Product.update( req.body , { where : { id: id } } );

    res.status(200).send(product);

}

**Delete Record:**

const deleteProduct = async (req, res) => {

    let id = req.params.id;

    const product = await Product.destroy( { where : { id: id } } );

    res.status(200).send("product is deleted");

}

**Where Conditions:**

const publishedProduct = async (req , res) => {

    const products = await Product.findAll( { where : {published : true } } );

    res.send(200).send(products);

}

**Specific Columns of Table:**

const getAllProducts = async (req , res) =>{

    //query if we want some specific information from the api

    let products = await Product.findAll({

        attributes: [

            'title',

            'price'

        ]

    });

    //send products data

    res.send(products);

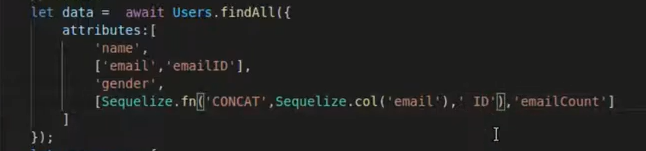
    res.status(200).send(products);

}

Here we use attributes array for fetch only required data from table.

**Sequelize Functions:**

Concat two columns or etc.



**Model Queries:**

**Fields:**

If you want to insert some selected fields into databse.

let data = await Product.create(

        {title:'product5' , price:20 , desciption:"Product Descriptios" , published:1},

        { fields:['title']}

        )

This query only allow to insert title in databe.

**Attributes:**

Use for **select** query for show selected fields in response.

let data = await Product.findAll({

        attributes:['title']

    })

**Include – Exclude:**

Exclude use if we don’t want to show specific filed it’s alternative of

**Attributes.**

let data = await Product.findAll({

        attributes:{exclude:['createdAt' , 'updatedAt']}

    })

This will show all table record but not show createdAt anf updatedAt.

**Where Conditions With Sequelize Operators.**

For use Sequelize operators we need to imort it.

const { Op } = require('sequelize');

Method of Use.

let data = await Product.findAll({

        where:{

            //id:1

            id:{ [Op.eq]:3 }

        }

    })

Other operators is.

[Op.or] - [Op.like] - [Op.not] - [Op.in]

let data = await Product.findAll({

        where:{

            //id:1

            //id:{ [Op.eq]:3 },

            title: { [Op.like]: `product3%` }

        }

    })

**Link:**

https://sequelize.org/v7/manual/model-querying-basics.html



**Group By – Order By:**

**Order By:**

let data = await Product.findAll({

        order:[

            ['id', 'DESC'],

            ['title', 'DESC'],

        ]

    })

**Order By Limit:**

let data = await Product.findAll({

        order:[

            ['id', 'DESC'],

            ['title', 'DESC'],

        ],

        limit:2

    })

**Offset:**

Mean how much records that you don’t want at start.

let data = await Product.findAll({

        order:[

            ['id', 'DESC'],

            ['title', 'DESC'],

        ],

        offset:2

    })

**Count:**

Counts numbers of records in database.

let data = await Product.count({});

**Finder:**

**FindAll():**

Use for find all record of table.

const finder = async (req , res) =>{

    let data = await Product.findAll({});

    let response = {

        data: data

    }

    res.status(200).json(response);

}

**findOne():**

    let data = await Product.findOne({});

**findByPk():**

find by primary key just add any number then it will return result matching with that primary key.

    let data = await Product.findByPk(3);

**findAndCountAll();**

it’s also tell the count number.

let data = await Product.findAndCountAll({});

**findOrCreate():**

First find we data not found then create.

let data = await Product.findOrCreate({

        where:{title:'Product6'},

        defaults:{

            price:15,

            description:"Product6 Descriptions",

        }

    });

It return two things one is **data** and second is **created** mean it create data or not or find data from table.

let [data , created] = await Product.findOrCreate({

        where:{title:'Product6'},

        defaults:{

            price:15,

            description:"Product6 Descriptions",

        }

    });

    let response = {

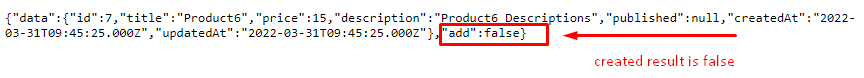
        data: data,

        add:created

    }

    res.status(200).json(response);

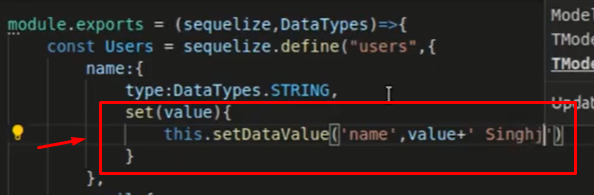
Result is if data exist in table.



**Setter and Getter.**

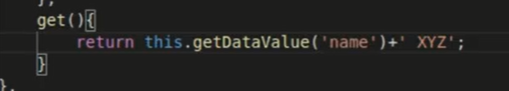
**Setter:**

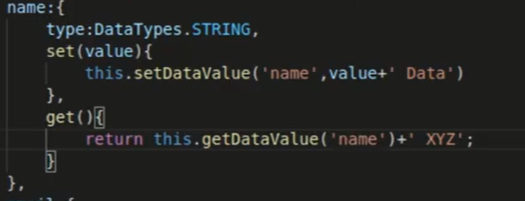
If we want to set something with every new record then we use **setter** and we used it in **models.** For example if we want to set ‘text’ with every name of new record then we use it.



**Getter:**

Used for set some value when we fetch data from database tables.





**Validations and Constraints:**

1: Sql level :- **Constraints:**

2: Sequelize level :- **Validations:**

**Constraints:**

Conditions that we add on model like **notNull** etc.

title:{

            type: DataTypes.STRING,

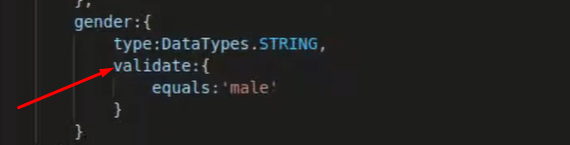
            allowNull: false,

            unique:true,

        },

**Validations:**

Also use in model for validate records.



Here we add validations for equal mean only save record whose gender is male.

**Relationships:**

**One to One:**

**First** we need to define our relationship in **index.js** file in **models** folder where we export db.

db.users.hasOne(db.posts , {foreignKey:'user\_id'});

db.posts.belongsTo(db.users, {foreignKey:'user\_id'});

Here **user\_id** is the foreign key of user table that we used inside posts table.

**Second** we need to use relationship models in our query for fetch data.

const oneToOne = async (req , res)=>{

    let data = await User.findAll({

        include:posts,

        where:{id:1}

    });

    let response = {

        daat:data

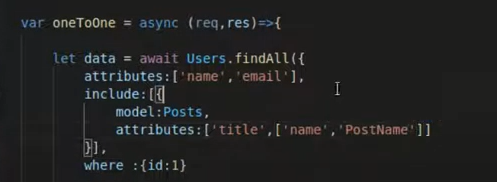
    };

    res.status(200).send(response);

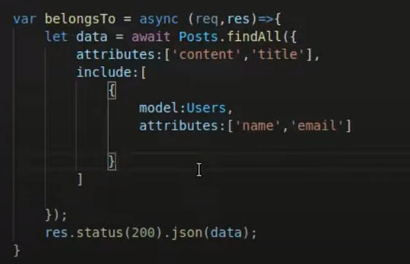
}

Here we use **include:posts** model mean also fetch posts of user 8.

**If want to fetch specific data mean specific fields of table:**



**Belongs To:**



**One to Many:**

We just need to change the relation name in **index.js** file in models folder.

db.users.hasMany(db.posts , {foreignKey:'user\_id'});

db.posts.belongsTo(db.users, {foreignKey:'user\_id'});

**Many to Many:**

**Index.js in models folder:**

db.posts.belongsToMany(db.tags, {through:'post\_tag'});

db.tags.belongsToMany(db.posts , {through:'post\_tag'})

**Function:**

const manayToMany = async (req, res) =>{

    // post to tag

    let data = await Post.findAll({

        attributes:['title', 'content'],

        include:[{

            model:Tag,

            attributes:['name']

        }]

    });

    res.status(200).send(data);

}