# 前面的orm项目虽然能够完成指定的功能，不过他的效率还是比较低，比较好的方式是async+await的方式，那么怎么把我们的函数变为async+await？

# 还是MySQL-demo2-self-orm项目，我们来学习async+await版本

## 1.新建一个04-mysql-app-orm-async-await.js，可以把03-mysql-app-orm.js的内容全部复制过来，然后改为async+await的代码，为了方便比较，我们还是保留了用来的代码把他们注释了

|  |
| --- |
| let path = require('path')  let express = require('express')  let bodyParser = require('body-parser') //导入bodyParser中间件  let db = require('./db/nodejs-orm/index') //导入orm模块  let orm\_config = require('./config/config') //导入数据库配置  let port = 3000  let app = express()  //注册bodyParser中间件  app.use(bodyParser.urlencoded({extended:false}))  //app.use(express.static(\_\_dirname+'/public'))//静态服务器，根据需要配置  //配置模板引擎和视图路径  app.engine('html',require('express-art-template'))  app.set('view options',{      debug: process.env.NODE\_ENV!='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  app.get('/',(req,res)=>{       res.render('index')  })  //连接数据库  db.connect(orm\_config)  //创建学生模型  let stuModel = db.model('students')  //获取学生列表的路由  app.get('/get\_data',(req,res)=>{      //查询所有,其实就是查询所有is\_deleted=0的学生is\_deleted=1就相当于删除了,不能做任何处理      // stuModel.find('is\_deleted=0',(err,data)=>{      //     res.render('student-list',{students:data})      // }) //这个代码效率不高，需要改为下面的代码      (async function get\_data(){            let result = await new Promise((resolve,reject)=>{              stuModel.find('is\_deleted=0',(err,data)=>{                      if(err){                          reject(err)                      }                      resolve(data)               })            })          res.render('student-list',{students:result})      })() //立即调用函数      // get\_data()  })  //显示学生详情信息的路由  app.get('/student\_detail/:id',(req,res)=>{      // let id = req.params.id      // let sql = 'select \* from students where id=' + id +";"      // //查询数据库，返回到浏览器        // stuModel.find('id='+id,(err,data)=>{      //     res.render('detail',{students:data})      // }) //这个代码效率不高，需要改为下面的代码        (async function(){          let id = req.params.id          let result =await new Promise((resolve,reject)=>{              stuModel.find('id='+id,(err,data)=>{                  if(err){                      reject(err)                  }                  resolve(data)              })          })          res.render('detail',{students:result})      })()    })  //渲染新增学生页面  app.get('/new',(req,res)=>{      res.render('new')  })  //实现新增学生功能的路由  app.post('/add\_stu',(req,res)=>{      // console.log(req.body);      //获取表单数据      // let stu = req.body      // stuModel.insert(stu,(err,data)=>{      //    if(err){      //     console.log(err);      //     return      //    }      //    res.redirect('/get\_data')      // })//这个效率不高，使用下面的写法较好      (async function(){          let student = req.body          await new Promise((resolve,reject)=>{           stuModel.insert(student,(err,data)=>{                   if(err){                      reject(err)                   }                   resolve(data)               })          })          res.redirect('/get\_data')      })()  })  //根据动态路由:id获取到学生id然后查询数据渲染edit页面  app.get('/edit\_student/:id',(req,res)=>{      let id = req.params.id //获取id      //根据id查询学生学习      stuModel.find("id="+id,(err,data)=>{          if(err){              console.log(err);              return;          }          res.render('edit.html',{student:data[0]}) //虽然数据是一个数组但是它只有一个元素,可以用下标0来获取      })    })  //将修改后的学生信息更新到数据库  app.post('/edit\_student',(req,res)=>{      // let student = req.body //获取表单数据      // //注意学生的id必须使用一个hidden字段传递过来      // //更新学生信息      // stuModel.update('id='+student.id,student,(err,data)=>{      //     if(err){      //         console.log(err);      //         return;      //     }      //     res.redirect('/get\_data')      // }) //这个效率不高，我们使用下面的写法     (async function(){         let student = req.body         await new Promise((resolve,reject)=>{          stuModel.update('id='+student.id,student,(err,data)=>{                  if(err){                     reject(err)                  }                  resolve(data)              })         })         res.redirect('/get\_data')     })()    })  //删除学生信息,其实就是给这个学生信息的is\_delete设置为1  app.get('/delete\_student/:id',(req,res)=>{      // let id = req.params.id //获取id      // //根据id查询学生学习      // stuModel.update('id='+id,{is\_deleted:1},(err,data)=>{      //     if(err){      //         console.log(err);      //         return;      //     }      //     res.redirect('/get\_data')      // })      (async function(){          let id = req.params.id          await new Promise((resolve,reject)=>{           stuModel.update('id='+id,{is\_deleted:1},(err,data)=>{                   if(err){                      reject(err)                   }                   resolve(data)               })          })          res.redirect('/get\_data')      })()  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

## 2.其他文件都不需要改动，项目结构如下

|  |
| --- |
|  |

### config/config.js

|  |
| --- |
| let orm\_config = {      host: 'localhost',//数据库地址      port:'3306',      user: 'root',//用户名，没有可不填      password: 'root',//密码，没有可不填      database: 'qd22'//数据库名称  }  module.exports = orm\_config |

### db/nodejs-orm/index.js

|  |
| --- |
| const mysql = require('mysql');  //数据库配置  let options = {};  let tableSQL = '';  let isConnect = false;  function Model(name, option) {      this.name = name;      this.option = option;  };  /\*\*  \* @description: 查询数据  \* @param {} options：可选参数  \* @param {Function} callback :（err,results）=>{}  \*/  Model.prototype.find = function (options, callback) {      if (!isConnect) {          this.connect(err => {              isConnect = true;              var str = '';              if (!callback) {                  str = `select \* from ${this.name}`;                  callback = options;              } else if (options.constructor == Array) {                  str = `select ${options.join()} from ${this.name}`;              } else {                  str = `select \* from ${this.name} where ${options}`;              };              //console.log(str);              connection.query(str, (error, results, fields) => {                  // console.log(error.sqlState);                  if (error && error.sqlState == '42S02') {                      callback('表格不存在', []);                  } else {                      callback(error, results, fields);                  };              });              return this;          })      } else {          var str = '';          if (!callback) {              str = `select \* from ${this.name}`;              callback = options;          } else if (options.constructor == Array) {              str = `select ${options.join()} from ${this.name}`;          } else {              str = `select \* from ${this.name} where ${options}`;          };          //console.log(str);          connection.query(str, (error, results, fields) => {              // console.log(error.sqlState);              if (error && error.sqlState == '42S02') {                  callback('表格不存在', []);              } else {                  callback(error, results, fields);              };          });          return this;      }  };  /\*\*  \* @description: 分页查询  \* @param {Object} options :   { where:查询条件, number: 当前页数 , count : 每页数量 }  \* @return:  \*/  Model.prototype.limit = function (options, callback) {      var str = '';      if (!options.where) {          str = `select \* from ${this.name} limit ${(options.number - 1) \* options.count},${options.count}`;      } else {          str = str = `select \* from ${this.name} where ${options.where} limit ${(options.number - 1) \* options.count},${options.count}`;      };      console.log(str);      connection.query(str, (error, results, fields) => {          callback(error, results, fields);      });      return this;  };  /\*\*  \* @description: 插入数据  \* @param {Object} obj:对象或者数组  \* @param {Function} callback :（err,results）=>{}  \*/  Model.prototype.insert = function (obj, callback) {      if (!isConnect) {          this.connect(err => {              if (err) {                  throw err;              } else {                  connection.query(tableSQL, (error, results, fields) => {                      if (Array.isArray(obj)) {                          for (var i = 0; i < obj.length; i++) {                              this.insertObj(obj[i], callback)                          }                      } else {                          this.insertObj(obj, callback)                      }                  });              }          });      } else {          if (Array.isArray(obj)) {              for (var i = 0; i < obj.length; i++) {                  this.insertObj(obj[i], callback)              }          } else {              this.insertObj(obj, callback)          }      }  };  Model.prototype.insertObj = function (obj, callback) {      let keys = [];      let values = '';      for (var key in obj) {          keys.push(key);          values += `"${obj[key]}",`;      };      values = values.replace(/,$/, '');      let str = `INSERT INTO ${this.name} (${keys.join()}) VALUES (${values})`;      connection.query(str, (error, results, fields) => {          callback(error, results);      });  }  /\*\*  \* @description: 更新数据  \* @param {Object} option：可选参数 更新条件  \* @param {Object} obj： 修改后的数据  \* @param {Function} callback :（err,results）=>{}  \*/  Model.prototype.update = function (option, obj, callback) {      let str = '';      if (arguments.length == 2) {          callback = obj;          obj = option;          str = `UPDATE ${this.name} SET `;          for (var key in obj) {              str += `${key}='${obj[key]}', `;          };          str = str.replace(/(, )$/, '');      } else {          str = `UPDATE ${this.name} SET `;          for (var key in obj) {              str += `${key}='${obj[key]}', `;          };          str = str.replace(/(, )$/, '');          str += ` where ${option}`;      };      console.log(str);      connection.query(str, (error, results, fields) => {          callback(error, results, fields);      });      return this;  };  /\*\*  \* @description: 删除数据  \* @param {Object} option：可选参数 删除条件  \* @param {Function} callback :（err,results）=>{}  \*/  Model.prototype.delete = function (option, callback) {      var str = '';      if (!callback) {          str = `delete from ${this.name}`;          callback = option;      } else {          str = `delete from ${this.name} where ${option}`;      };      console.log(str);      connection.query(str, (error, results, fields) => {          callback(error, results, fields);      });      return this;  };  /\*\*  \* @description: 执行sql语句  \* @param {String} str : sql语句  \* @param {Function} callback :（err,results）=>{}  \*/  Model.prototype.sql = function (str, callback) {      connection.query(str, (error, results, fields) => {          callback(error, results, fields);      });      return this;  };  /\*\*  \* @description: 删除model表格 （慎用！）  \* @param {type}  \* @return:  \*/  Model.prototype.drop = function (callback) {      connection.query(`DROP TABLE ${this.name}`, (error, results, fields) => {          callback(error, results, fields);      });      return this;  };  //连接检测  Model.prototype.connect = function (callback) {      let p1 = new Promise((resolve, reject) => {          connection.connect((err) => {              if (err) {                  //console.log(err.stack);                  //console.log(err);//42000 数据库不存在  28000账号错误                  //console.log(err.sqlState);//42000 数据库不存在  28000账号错误                  reject(err);              } else {                  resolve();              }          });      });      p1.then(() => {          callback(null);      }, err => {          if (err.sqlState == 42000) {              createDatabase(callback);          } else if (err.sqlState == 28000) {              callback('数据库账号或密码错误');          } else {              callback(err);          }      });  };  //创建数据库  let createDatabase = function (callback) {      let p2 = new Promise((resolve, reject) => {          connection = mysql.createConnection({              host: options.host,//数据库地址              port: options.port,//端口号              user: options.user,//用户名，没有可不填              password: options.password,//密码，没有可不填          });          connection.connect((err) => {              //if (err) throw error;              if (err) {                  reject(err);              } else {                  resolve();              }          });      });      let p3 = new Promise((resolve, reject) => {          connection.query(`CREATE DATABASE ${options.database}`, (err, results, fields) => {              //if (error) throw error;              if (err) {                  reject(err);              } else {                  resolve();              }          });      });      let p4 = new Promise((resolve, reject) => {          connection.query(`use ${options.database}`, (err, results, fields) => {              if (err) {                  reject(err);              } else {                  resolve();              }          });      });      let pAll = Promise.all([p2, p3, p4]);      pAll.then(() => {          callback(null);      }).catch((err) => {          callback(err);      });  }    let ormTool = {      /\*\*      \* @description:连接数据库      \* @param {String} host: 主机名 默认localhost      \* @param {Number} port: 端口号 默认3306      \* @param {String} user: 用户名      \* @param {String} password: 密码      \* @param {String} database: 数据库名称 默认hm      \* @return:      \*/      connect: function ({ host = 'localhost', port = 3306, user = '', password = '', database = 'hm' }) {          databaseName = database;//全局存储当前数据库名称          options = {              host,//数据库地址              port,//端口号              user,//用户名，没有可不填              password,//密码，没有可不填              database//数据库名称          };          connection = mysql.createConnection(options);      },      /\*\*      \* @description:创建model (表格模型对象)      \* @param {String} name:表格名称      \* @param {Object} options:表格数据结构      \* @return: Model对象：负责数据库增删改查      \*/      model: function (name, options) {          let str = 'id int primary key auto\_increment, ';          for (var key in options) {              if (options[key] == Number) {                  str += `${key} numeric,`;              } else if (options[key] == Date) {                  str += `${key} timestamp,`;              } else {                  str += `${key} varchar(255),`;              }          };          str = str.replace(/,$/, '');          //console.log(`CREATE TABLE ${name} (${str})`);          //console.log(str);          tableSQL = `CREATE TABLE ${name} (${str})`;          return new Model(name, options);      }  };  module.exports = ormTool; |

### views/detail.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>学生详情</title>  </head>  <body>      <h2>学生信息</h2>      <hr>      <h3>          <ul style="list-style: none;">              {{each students}}              <li>姓名：{{$value.name}}</li>              <li>性别：{{$value.gender}}</li>              <li>年龄：{{$value.age}}</li>              <li>身高：{{$value.height}}</li>              {{/each}}          </ul>      </h3>  </body>  </html> |

### views/edit.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>编辑学生信息</title>  </head>  <body>      <form action="/edit\_student" method="post">          <h1添加学生< /h1>              <input type="hidden" name="id" value={{student.id}}> <!-- 这个很重要否则post路由无法更新数据 -->              姓名:              <input type="text" name="name" value={{student.name}}>              <br><br>              年龄:              <input type="text" name="age" value={{student.age}}>              <br><br>              性别:              <input type="text" name="gender" value={{student.gender}}>              <br><br>              身高:              <input type="text" name="height" value={{student.height}}>              <br><br>              班级id:              <input type="text" name="cls\_id" value={{student.cls\_id}}>              <br><br>              <input type="submit" value="提交">              <input type="reset">      </form>  </body>  </html> |

### views/index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>  </head>  <body>      <h1>首页</h1>      <hr>      <h3><a href="/get\_data">查看学生列表</a></h3>  </body>  </html> |

### views/new.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加学生</title>  </head>  <body>      <form action="/add\_stu" method="post">          <h1添加学生< /h1>              姓名:              <input type="text" name="name" value="">              <br><br>              年龄:              <input type="text" name="age" value="">              <br><br>              性别:              <input type="radio" name="gender" value="1" checked>男              <input type="radio" name="gender" value="2">女              <br><br>              身高:              <input type="text" name="height" value="">              <br><br>              班级id:              <input type="text" name="cls\_id">              <br><br>              <input type="submit" value="提交">              <input type="reset">      </form>  </body>  </html> |

### views/student-list.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>学生列表</title>      <style>          a {              text-decoration: none;              font-weight: 540;          }          a:link {              color: rgb(24, 22, 22);              font-size: 18px;          }          a:visited {              color: deepskyblue;          }          a:hover {              color: lime;          }          a:active {              color: skyblue;          }      </style>  </head>  <body>      <h2>学生列表</h2>      <hr>      <table>          {{each students}}          <tr>              <td>学生姓名：</td>              <td>{{$value.name}}</td>              <td>详情：</td>              <td><a href="/student\_detail/{{$value.id}}">学生详情</a></td>              <td><a href="/edit\_student/{{$value.id}}">修改</a></td>              <td><a href="/delete\_student/{{$value.id}}">删除</a></td>          </tr>          {{/each}}      </table>      <a href="/new">添加学生</a>  </body>  </html> |

## 3.测试一下，能够正常运行

|  |
| --- |
|  |
|  |
|  |
|  |
|  |