# 还是MySQL-demo2-self-orm项目，我们用async-await把它的代码改写了，但是还不完整，因为有错误时我们没有处理，现在来添加异常处理

|  |
| --- |
|  |

## 1.把04-mysql-app-orm-async-await.js复制一份改名05-mysql-app-orm-handle-exceptions.js

|  |
| --- |
|  |

## 2.给所有的路由函数的代码加上try..catch(error)语句

|  |
| --- |
| 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 = 3001  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      let result;      (async function get\_data(){          try {                result = await new Promise((resolve,reject)=>{                  stuModel.find('is\_deleted=0',(err,data)=>{                          if(err){                              reject(err)                          }                          resolve(data)                   })                })            } catch (error) {              console.log(error);              res.send({errMsg:'数据库查询错误'})              return;          }          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          try {               result =await new Promise((resolve,reject)=>{                stuModel.find('id='+id,(err,data)=>{                  if(err){                      reject(err)                  }                  resolve(data)                })              })          } catch (error) {              console.log(error);              res.send({errMsg:'数据库查询错误'})              return;          }          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          try {               await new Promise((resolve,reject)=>{                  stuModel.insert(student,(err,data)=>{                      if(err){                          reject(err)                      }                      resolve(data)                  })              })          } catch (error) {              console.log(error);              res.send({errMsg:'插入数据错误'})              return;          }            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      // })      let id = req.params.id //获取id      let result      (async function ed(){          try {              result = await new Promise((resovle,reject)=>{                  stuModel.find("id="+id,(err,data)=>{                  if(err){                     reject(err);                  }                  resovle(data)              })           })          } catch (error) {              console.log(error);              res.send({errMsg:'数据库查询错误'})              return;          }          res.render('edit.html',{student:result[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      //需要添加异常处理代码，使得程序更加健壮     (async function(){      try {          let student = req.body          await new Promise((resolve,reject)=>{           stuModel.update('id='+student.id,student,(err,data)=>{                   if(err){                      reject(err)                   }                   resolve(data)               })          })      } catch (error) {          console.log(error);          res.send({errMsg:'更新数据错误'})          return;      }        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(){          try {              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)                   })              })          } catch (error) {              console.log(error);              res.send({errMsg:'删除数据错误'})              return;          }            res.redirect('/get\_data')      })()  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

### 测试一下，正常运行：

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

### 我们把student-list.html修改一下

|  |
| --- |
| <!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><span>学生列表 </span> &nbsp;&nbsp;&nbsp;<h3><a href="/new">添加学生</a></h3>      </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>  </body>  </html> |

## 其他文件没有修改，可以查看：[上一节的笔记](node框架Express+mysql笔记06-使用ORM操作mysql使用async+await.docx)