//DB库

var MongoClient = require('mongodb').MongoClient;

var ObjectId = require('mongodb').ObjectID;

var Config = require('./config.js');

//DB操作类

class Db {

//创建单例模式

static dbInstance(){

if (!Db.instance){

Db.instance = new Db();

}

return Db.instance;

}

//构造函数

constructor(){

var dbClient = '';

this.connect();

}

//连接数据库

connect(){

return new Promise((resolve,reject)=>{

if (!this.dbClient){

MongoClient.connect(Config.dburl,(err,client)=>{

if (err){

reject(err);

} else {

this.dbClient = client.db(Config.dbName);

resolve(this.dbClient);

}

})

} else {

resolve(this.dbClient);

}

})

}

getObjectId(id){

return new ObjectId(id);

}

//查找数据

find(collectionName,json){

return new Promise((resolve,reject)=>{

this.connect().then((db)=>{

var result = db.collection(collectionName).find(json);

result.toArray((err,docs)=>{

if (err){

reject(err);

return;

} else {

resolve(docs);

}

})

})

})

}

/\*插入数据\*/

insert(collectionName,json){

return new Promise((resolve,reject) => {

this.connect().then((db) => {

db.collection(collectionName).insertOne(json,(err,result) => {

if (err){

reject(err);

return;

} else {

resolve(result);

}

})

})

})

}

//更新数据库

update(collectionName,json1,json2){

return new Promise((resolve,reject) => {

this.connect().then((db)=>{

db.collection(collectionName).updateOne(json1,{$set:json2},(err,result)=>{

if (err){

reject(err);

} else {

resolve(result);

}

})

})

})

}

updateMany(collectionName,json1,json2){

return new Promise((resolve,reject) => {

this.connect().then((db)=>{

db.collection(collectionName).updateMany(json1,{$set:json2},(err,result)=>{

if (err){

reject(err);

} else {

resolve(result);

}

})

})

})

}

// 删除数据

del(collectionName,json){

return new Promise((resolve,reject) => {

this.connect().then((db)=>{

db.collection(collectionName).removeOne(json,(err,result)=>{

if (err){

reject(err);

} else {

resolve(result);

}

})

})

})

}

}

module.exports = Db.dbInstance();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//新增

router.get('/add',async (ctx) => {

await ctx.render('add');

})

//执行新增

router.post('/adddo', async(ctx)=>{

//接收表单数据

let data = ctx.request.body;

let res = await DB.insert('students',data); //因为是异步，必须等到其返回，所以要使用await，否则下面的try会被执行，从而报错

try{

if (res.result.ok){

ctx.body = '新增成功';

ctx.redirect('/list');

}

} catch(err){

ctx.body = '新增失败';

return;

}

})

//修改

router.get('/edit',async (ctx)=>{

//获取ID参数

let id = ctx.query.id;

let data = await DB.find('students',{"\_id":DB.getObjectId(id)});

console.log(data);

await ctx.render('edit',{'item':data[0]});

})

router.post('/editdo',async (ctx) => {

//获取表单数据

let newData = ctx.request.body;

let id = DB.getObjectId(newData.id);

let name= newData.name;

let age = newData.age;

let sex = newData.sex;

//更新

let res = await DB.update('students',{"\_id":id},{"name":name,'age':age,'sex':sex});

try{

if (res.result.ok){

ctx.redirect('/list');

}

} catch(err){

ctx.body = '修改失败';

}

})

//删除

router.get('/del',async (ctx)=>{

//获取ID

let id = DB.getObjectId(ctx.query.id);

let res = await DB.del('students',{"\_id":id});

try{

if (res.result.ok){

ctx.redirect('/list');

}

} catch(err){

ctx.body = '删除出错';

}

})