1. 中间件 app.use(...) ※中间件的使用是在服务器基础上

为主要的业务逻辑服务。接受到请求，以及做出响应

应用级中间件，路由级中间件，内置中间件，第三方中间件（插件），错误处理中间件。

1. 路由级中间件

路由器的使用。app.use('/user',userRouter)

1. 应用级中间件

也成为自定义中间件，是一个函数形式（回调函数）

app.use（'/拦截的url'，function（req,res,next）{

拦截指定路由

next( ) 表示执行下一个中间件或者路由

}）

1. 内置中间件

（express4中只保留了一个，其他都保留到第三方去了）

app.use(express.static('托管的目录'))

把所有静态资源（文件端HTML，CSS，js等）托管到public目录下。当浏览器请求静态资源，会自动到public目录下寻找，不需要再创建路由响应文件

可同时用多个托管目录，但会按照程序顺序托管目录调用

1. 第三方中间件

body-parser 中间件使用

|  |
| --- |
| 1. 下载安装   npm install body-parser   1. 引入中间件模块   const bodyParser = require('body-paeser');   1. 使用中间件 将post请求的数据解析为对象   app.use( bodyParser.urlencoded({  extended:false //表示不适用第三方的qs模块而是使用核心模块querystring来解析查询字符串为对象  }) )   1. 在post请求的路由中   req.body 返回对象 |

1. 获取数据的方式 ※

|  |  |  |
| --- | --- | --- |
| 名称 | 形式 | 获取（路由）**返回对象** |
| get | lid=1&pname=dell | req.query |
| 路由传递 | /shopping/4999 | req.params |
| post | 无法通过url查看（上边俩都可以） | req.body  前提使用中间件body-parser |

1. MySQL模块 （用web操作数据库）

建立web与数据库连接

//引入MySQL模块

const mysql = require('mysql');

//创建连接对象

var connection = mysql.createConnection({ //传递一个对象

host:'127.0.0.1',

port:'3306',

user:'root',

password:'',

database:'tedu' //连接后使用的数据库

});

//建立连接

connection.connect();

//执行SQL语句

connection.query(SQL语句 , callback)

err 可能产生的错误

result SQL语句执行的结果

执行删除、修改、插入返回值为对象，如果返回对象属性中affectedRows大于0，则说明执行成功。

执行查询返回数组

用占位符 可以防止SQL注入（影响你的数据）

connection.query('INSERT INTO emp VALUES(?,?,?,?,?,?)' , [null,'ZSY',1,'1998-5-22',8000,10] , function(err,result){

if(err) throw err;

console.log(result);

})

例：

//引入MySQL模块

const mysql = require('mysql');

//创建连接对象

var connection = mysql.createConnection({ //传一个对象

host:'127.0.0.1',

port:'3306',

user:'root',

password:'',

database:'tedu' //连接后使用的数据库

});

//建立连接

connection.connect();

//执行SQL语句

/\*

connection.query('UPDATE emp SET sex =1,salary = 5000 WHERE eid=10',function(err,result){

//err 可能产生的错误 resultSQL语句的结果

if(err) throw err;

console.log(result);

})

connection.query( `INSERT INTO emp VALUES(NULL,'SYR',0,'1996-10-14',10000,20)`,function(err,result){

if(err) throw err;

console.log(result);

} )

connection.query('SELECT \* FROM emp',function(err,result){

if(err) throw err;

console.log(result);

})

connection.query('INSERT INTO empVALUES(?,?,?,?,?,?)',

[null,'ZSY',1,'1998-5-22',8000,10],function(err,result){

if(err) throw err;

console.log(result);

})

\*/

//直接将对象插入数据库，插入要改成 SET 对象

var emp ={

ename:'zhang',

sex:0,

salary:5000,

birthday:'1982-07-09'

}

connection.query('INSERT INTO emp SET ?',[emp],function(err,result){

if(err) throw err,

console.log(result);

})

//关闭连接

connection.end()

1. 连接池

|  |
| --- |
| 使用连接池  //connectionLimit 用于设置连接池的大小，默认15个  var pool =mysql.createPool({ });  //执行SQL语句  pool.query(sql语句，callback) |

例

//使用连接池连接

const mysql = require('mysql');

//创建一个连接池对象

var pool =mysql.createPool({

host:'127.0.0.1',

port:3306,

user:'root',

password:'',

database:'tedu',

connectionLimit:30 //设置连接池的大小，默认15

});

//执行SQL语句

/\*

pool.query('SELECT \* FROM emp',function(err,result){

if (err)

{ throw err

}console.log(result);

}) \*/

pool.query('DELETE FROM emp WHERE eid=?',[5],function(err,result){

if(err) throw err;

console.log(result);

})