复习

web服务器

请求 request

响应 response

1.中间件

为主要的业务逻辑服务；接收到请求，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、错误处理中间件

(1)路由级中间件

路由器的使用

(2)应用级中间件

也称为自定义中间件，是一个函数(回调函数)

app.use( '/reg', function(req,res,next){ } )

拦截url为/reg的路由；

next是一个函数，表示执行下一个中间件或者路由

(3)内置中间件

app.use( express.static('托管的目录') )

把所有的静态资源托管到某个目录，如果浏览器请求静态资源，自动到该目录下寻找，无需创建路由

练习: 把静态资源托管到 files 目录。

(4)第三方中间件

body-parser中间件使用

|  |
| --- |
| //1.下载安装  npm install body-parser;  //2.引入中间件模块  const bodyParser=require('body-parser');  //3.使用中间件  //将post请求数据解析为对象  app.use( bodyParser.urlencoded({  extended:false //不使用第三方的qs模块解析查询字符串为对象，而是使用querystring模块  }) )  //4.在post请求的路由中  req.body 返回对象 |

2.获取数据的方式

|  |  |  |
| --- | --- | --- |
| 名称 | 形式 | 获取（路由） |
| get | lid=1&pname=dell | req.query |
| 路由传递 | /shopping/4999 | req.params |
| post | 无法通过url查看 | req.body  前提需要使用中间件body-parser |

3.mysql模块

mysql.exe -h127.0.0.1 -P3306 -uroot -p

mysql -uroot

INSERT INTO emp VALUES(...)

DELETE FROM emp WHERE eid=5;

UPDATE emp SET upwd='1234',sex=0 WHERE eid=4;

SELECT \* FROM emp;

|  |
| --- |
| //创建连接对象  var connection=mysql.createConnection({})  //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语句的执行结果  //关闭连接  connection.end() |

执行删除、修改、插入返回对象，如果返回对象的属性中affectedRows大于0，说明执行成功。

执行查询返回数组

用户名和密码同时满足的结果，返回一条数据

SQL注入

SELECT \* FROM user WHERE uname='root' AND upwd='123456' **or 1=1**;

|  |
| --- |
| 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) |

练习：托管静态资源到当前目录下的public目录，创建注册网页user\_reg.html

练习：在user.js路由器文件中，创建路由器，添加路由(post /reg)，导出路由器；

在app.js引入路由器，并挂载到 /user下

require('./routes/user.js')

课后任务

(1)复习今天内容，整理思维导图

(2)练习

创建web服务器(服务器，路由)，托管静态资源到public，创建文件add.html，点击提交，将数据插入到tedu下的数据表dept中

