数据库与身份认证

1. 数据库的基本概念
   1. 什么是数据库

数据库是用来组织、存储和管理数据的仓库。数据库管理系统（数据库）。

* 1. 传统型数据库的数据组织结构

数据组织结构分为：数据库、数据表、数据行、字段。

1. MySQL的基本使用
   1. 使用SQL管理数据库
2. 什么是SQL

SQL是结构化查询语言，专门用来访问和处理数据库的编程语言。SQL语言只能在关系型数据库中使用。

* 1. SQL的SELECT语句

1. SELECT \* FROM 表名称/ SELECT 列名称 FROM 表名称

例：select \* from user/select username from user

* 1. SQL的INSERT INTO语句

1. INSERT INTO 表名称 (列1, 列2, ……) VALUES(‘值1’, ‘值2’, ……)

insert into user (username, password) values (‘zs’, 123456) //新增一条数据，username值为zs，password值为123456

* 1. SQL的UPDATE语句

1. UPDATE 表名称 SET 列名称 = ‘新值’ , 列名称 = ‘新值’ WHERE 列名称 = ‘某值’

update user set username = ‘zs’, password = ‘1’ where id = 1 //将user表中id=1的username的值更新为zs，password的值更新为1

* 1. SQL的DELETE语句

1. DELETE FROM 表名称 WHERE 列名称 = ‘值’

delete from user where id = 1 //删除user表中id = 1的这行数据

* 1. SQL的WHERE子句

WHERE子句可限定选择标准。

1. SELECT 列名称 FROM 表名称 WHERE 列名称 运算符 值

WHERE子句中可使用的运算符：= <> > < >= <= BETWEEN LIKE, <>可写为!=

* 1. SQL的AND和OR运算符

1. SELECT 列名称 FROM 表名称 WHERE 列 运算符 值 AND 列 运算符 值

例：select \* from user where status = 1 and id < 15

1. SELECT 列名称 FROM 表名称 WHERE 列 运算符 值 OR列 运算符 值

例：select \* from user where status = 1 or id < 15

* 1. SQL中的ORDER BY子句

1. SELECT \* FROM 表名称 ORDER BY 字段名 ASC/DESC //默认升序排列
2. SELECT \* FROM 表名称 ORDER BY 字段名1 ASC/DESC, 字段名2 ASC/DESC //多重排序，先按照字段名1进行排序，再按照字段名2进行排序

例：select \* from user oeder by username asc, password desc

* 1. SQL是我COUNT(\*)函数

1. SELECT COUNT(\*) FROM 表名 WHERE 列名 运算符 值

例：select count(\*) from user where status = 1 //获得user表中status = 1的数据总数

1. SELECT COUNT(\*) AS 别名FROM 表名 WHERE 列名 运算符 值

例：select username as uname, password as upwd form user //给查询结果起别名

1. 在项目中操作MySQL
   1. 在项目中操作数据库的步骤
2. 安装操作MySQL数据库的第三方模块；
3. 通过mysql模块链接到MySQL数据库；
4. 通过mysql模块执行SQL语句。
   1. 安装与配置mysql模块

mysql模块是托管于npm上的第三方模块，它提供了在Node.js项目中连接和操作MySQL数据库的能力。

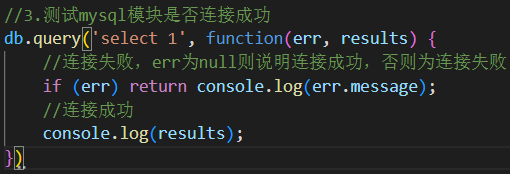
1. 安装mysql模块

npm I mysql

1. 配置mysql模块

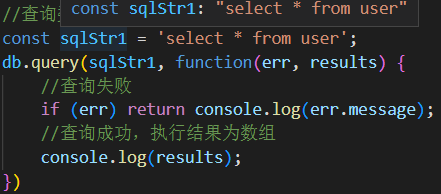


1. 测试mysql模块能否正常工作



* 1. 使用mysql模块操作MySQL数据库

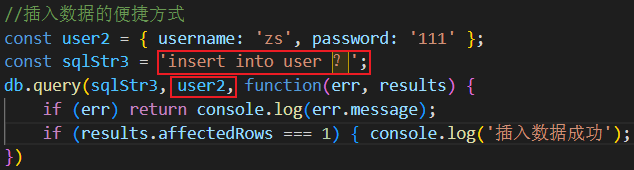
1. 查询数据



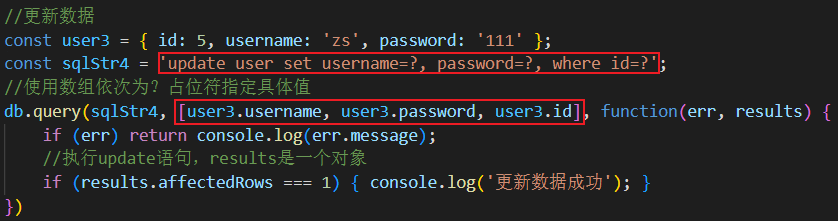
1. 插入数据



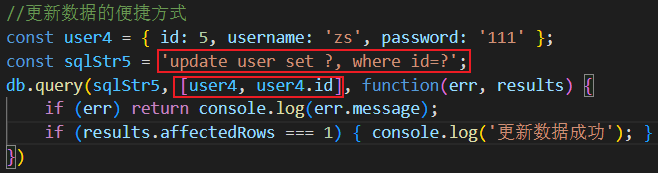
1. 插入数据的便捷方式



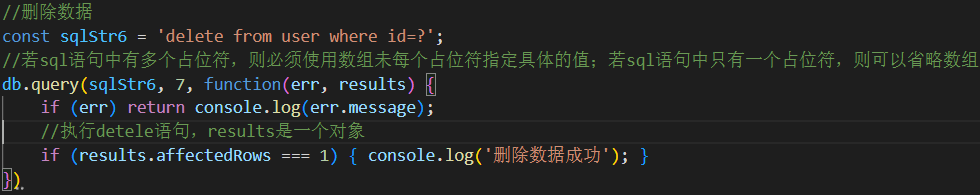
1. 更新数据



1. 更新数据的便捷方式



1. 删除数据



1. 标记删除
2. 前后端的身份认证
   1. Web开发模式
3. 服务端渲染的Web开发模式

服务器渲染：服务器发送给客户端的HTML页面，是在服务器通过字符串拼接动态生成的。

1. 服务器渲染的优缺点

优点：前端耗时少；有利于SEO

缺点：占用服务器端资源；不利于前后端分离，开发效率低。

1. 前后端分离的Web开发模式

前后端分离的Web开发模式依赖于Ajax技术的广泛应用。后端只负责提供API接口，前端使用Ajax调用接口。

1. 前后端分离的优缺点

优点：开发体验好；用户体验好；减轻了服务器端的渲染压力；

缺点：不利于SEO(搜索引擎优化)。

* 1. 身份认证

1. 什么是身份认证

通过一定的手段完成对用户身份的确认。例：手机验证码登录，二维码登录。

1. 不同开发模式下的身份认证

服务器渲染推荐使用Session认证机制；

前后端分离推荐使用JWT认证机制。

* 1. Session认证机制

1. HTTP协议的无状态性

HTTP协议的无状态性指每次HTTP请求都是独立的，连续多个请求之间没有直接的关系，服务器不会主动保留每次HTTP请求的状态。

1. 如何突破HTTP无状态的限制

Cookie

1. 什么是Cookie

Cookie是存储在用户浏览器中的一段不超过4KB的字符串。它由一个Name，一个Value和其它几个用于控制Cookie有效期、安全期、适用范围的可选属性组成。

不同域名下的Cookie各自独立，当客户端发起请求时，会自动把当前域名下多有未过期的Cookie一同发送到服务器。

Cookie的特性：自动发送；域名独立；过期时限；4KB限制。

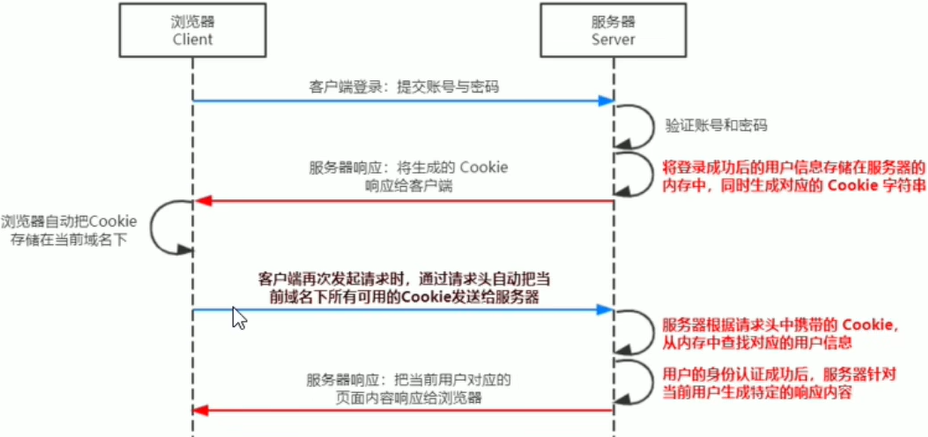
1. Cookie在身份认证中的作用

客户端第一次请求服务器时，服务器通过响应头的形式向客户端发送一个身份认证的Cookie，客户端收到后会自动保存在浏览器中。当客户端浏览器再次请求服务器时，浏览器会自动将身份认证相关的Cookie通过请求头的形式发送给服务器，服务器即可验明客户端的身份。

1. Cookie不具有安全性

Cookie存储在浏览器中，可读可写，容易伪造，所以不具有安全性。不可使用Cookie存储重要且隐私的数据。

1. Session的工作原理

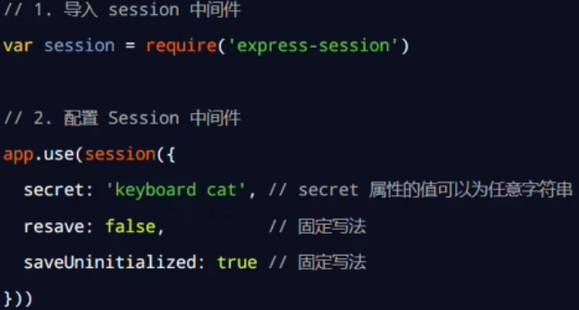


* 1. 在Express中使用Session中间件

1. 安装express-session中间件

执行npm install express-session命令

1. 配置express-session中间件



1. 向session中存数据