

3.24

2020年3月24日 0:02

- ctrl+shift+f12: 放大缩小
- shift+enter: 另起一行
- ctrl+x, c, v: 对一整行进行操作, ctrl+z: 回退一行

- 数据库

- 增加 (返回int) : insert into 数据库其中的一张表名 (字段, 字段) value(值, 值);
- 删除 (返回int) : delete from 数据库其中的一张表名 where 变量名 = ' ' ;
- 查询 (返回int) : select * from 数据库其中的一张表名
- 修改 (返回ResultSet) : update 数据库其中的一张表名 set 变量名=' (要改成的内容)',set 变量名=' (要改成的内容)' where 另一个没有要修改的变量名 = ' ' ;
- 查看当前选择的数据库: select database();
- 选择要进行操作的数据库: use 数据库名;
- 显示所有的数据库: show database();
- 只针对某些列做插入

```
mysql> insert into user(username,password) values('Jack','654646');
Query OK, 1 row affected (0.06 sec)
```

id	username	password
4	siki	123
5	Micheal	123
6	Jack	654646

3.25

2020年3月24日 0:02

- Git

- 创建版本库

- 初始化git仓库: 'git init'
 - 显示git目录: 'ls -ah'
 - 配置个人信息 (在git中设置当前使用的用户) : 名字: 'git config --global user.name "xxx" ' 邮箱: 'git config --global user.email "1371539005@qq.com" ' 每次备份都会把备份者信息存储

- 把文件添加到版本库: (2步: commit一次可以add多个文件)

- 把文件放在门口: 打开文件所在目录, 执行命令: 'git add ./xxx.扩展名' (./表示当前目录, 按tab可以自动补完文件名) 'git add .'添加文件夹下的所有文件
 - 把文件放进仓库: 执行命令: 'git commit -m "对本次提交的说明" '

- 版本回退

- 查看从最近到最远的提交日志: 'git log'
 - 只查看日志提交说明: 'git log --pretty=oneline'
 - 把文档回退到上一个版本: 'git reset --hard HEAD^'
 - HEAD指向的版本就是当前的版本 HEAD^: 上一个版本 HEAD^^: 上上一个版本 HEAD~100: 上100个版本
 - 把文档回到未来的某个版本: 'git reset --hard commit_id'
 - 所有文件版本的Commit_id 的查找(查看提交历史): 'git reflog'

- 修改

- 查看仓库当前状态: 'git status'
查看文件修改的具体内容: 'git diff xxx.扩展名'
 - 查看工作区和版本库里面最新版本的区别: 'git diff HEAD -- xxx.扩展名'
 - 撤销文件在工作区的全部修改'git checkout -- xxx.扩展名', 这里有两种情况:
 - 一种是文件自修改后还没有被放到暂存区, 现在, 撤销修改就回到和版本库一模一样的状态
 - 一种是文件已经添加到暂存区后, 又做了修改, 现在, 撤销操作就回到添加暂存区后的状态
 - 撤销文件在暂存区的修改, 重新放回工作区 (该修改会放到工作区中去) :
'git reset HEAD xxx.扩展名'

- 删除文件

- 从版本库中删除该文件: 使用命令'git rm xxx.扩展名', 并且'git commit -m "remove xxx.扩展名" '
 - 删错了从版本库中恢复到最新版本: 'git checkout -- xxx.扩展名' (该命令是用版本库中的版本替换工作区的版本, 无论工作区是修改还是删除, 都可以一键还原)

- 添加远程库: <https://www.liaoxuefeng.com/wiki/896043488029600/898732864121440>

3.26

2020年3月26日 8:26

- JDBC编程

- 一：注册驱动（告诉Java程序即将要连接的数据库）

- 方式一

- `Driver driver = new com.mysql.jdbc.Driver();`
`DriverManager.registerDriver(driver);`

- 要导包

- ◆ `import java.sql.Driver;`
`import java.sql.DriverManager;`

- 方式二：类加载方式注册驱动（常用）

- `try{Class.forName ("com.mysql.jdbc.Driver")`
 `}catch (ClassNotFoundException e) {`
 `e.printStackTrace () ;`
 `}`

- 二：获取链接（表示JVM的进程和数据库之间的通道打开了，使用完后一定要关闭）

- `/*`

- `url:统一资源定位符`

- `url包括以下几个部分：`

- `协议 IP PORT（服务器上软件的端口） 资源名`

- `jdbc:mysql://localhost:3306/text`

- `jdbc:mysql:// 协议`

- `localhost IP地址`

- `3306 MySQL数据库端口号`

- `text 具体数据库实例名`

- `说明：localhost和127.0.0.1都是本机地址`

- `*/`

- `String url = "jdbc:mysql://localhost:3306/text";`

- `String user = "root";`

- `String password = "myasdw902";`

- `Connection con = DriverManager.getConnection(url,user,password);`

- 要导包

- `import java.sql.Connection;`

- 三：获取数据库操作对象（专门执行sql语句的对象）

- `Statement stmt = con.createStatement();`//创建一个Statement对象来将SQL语句发送到数据库

- 四：执行SQL语句（DQL,DML...）

- `String sql = "insert into text_detail(text_name,text_password) value('henhao','945')";`

- `//sql语句，不需要分号`

- `//executeUpdate(String a)专门执行DML语句，返回int（影响数据库中的记录条数）`

- `int count = stmt.executeUpdate(sql);`//此处的sql语句为 增 删 改 语句

```
System.out.println(count == 1 ? "保存成功":"保存失败");
```

- 五：处理查询结果集（只有第四步执行的是select语句才有第五步）
- 六：释放资源（使用完后一定要关闭资源）
 - //为保证资源一定释放。在finally语句块中关闭资源
//并且要遵循先打开后关闭，要分别对其try...catch (SQLException e)
{e.printStackTrace()}..., 避免第一个close();出错后面的close没有被执行

3.27

2020年3月27日 19:39

- JDBC：实际开发中不建议把数据库的信息写死在程序中
 - 使用资源绑定器绑定属性配置文件
 - Java程序
 - 要导包： `import java.util.*;`//导入该包的全部类
 - /*仅仅需要把myresource.properties文件放在src下，如果是放在package下，则程序的filename应该 package名/文件名(不要后缀)*/
`ResourceBundle bundle = ResourceBundle.getBundle("resources/db");`
`String driver = bundle.getString("driver");`
`String url = bundle.getString("url");`
`String user = bundle.getString("user");`
`String password = bundle.getString("password");`
 - 配置文件：（xxx.properties）
//没有分号
`driver = com.mysql.jdbc.Driver`
`url = jdbc:mysql://localhost:3306/text`
`user = root`
`password = myasdw902`
 - 处理查询结果集
 - `ResultSet rs = null;` //关闭时先关闭它
 - `Re.next()`：使光标在结果集向下移一行，有效返回true，无效返回false