# 数据库技术

张海宁

贵州大学

*hnzhang1@gzu.edu.cn*

June 1, 2018

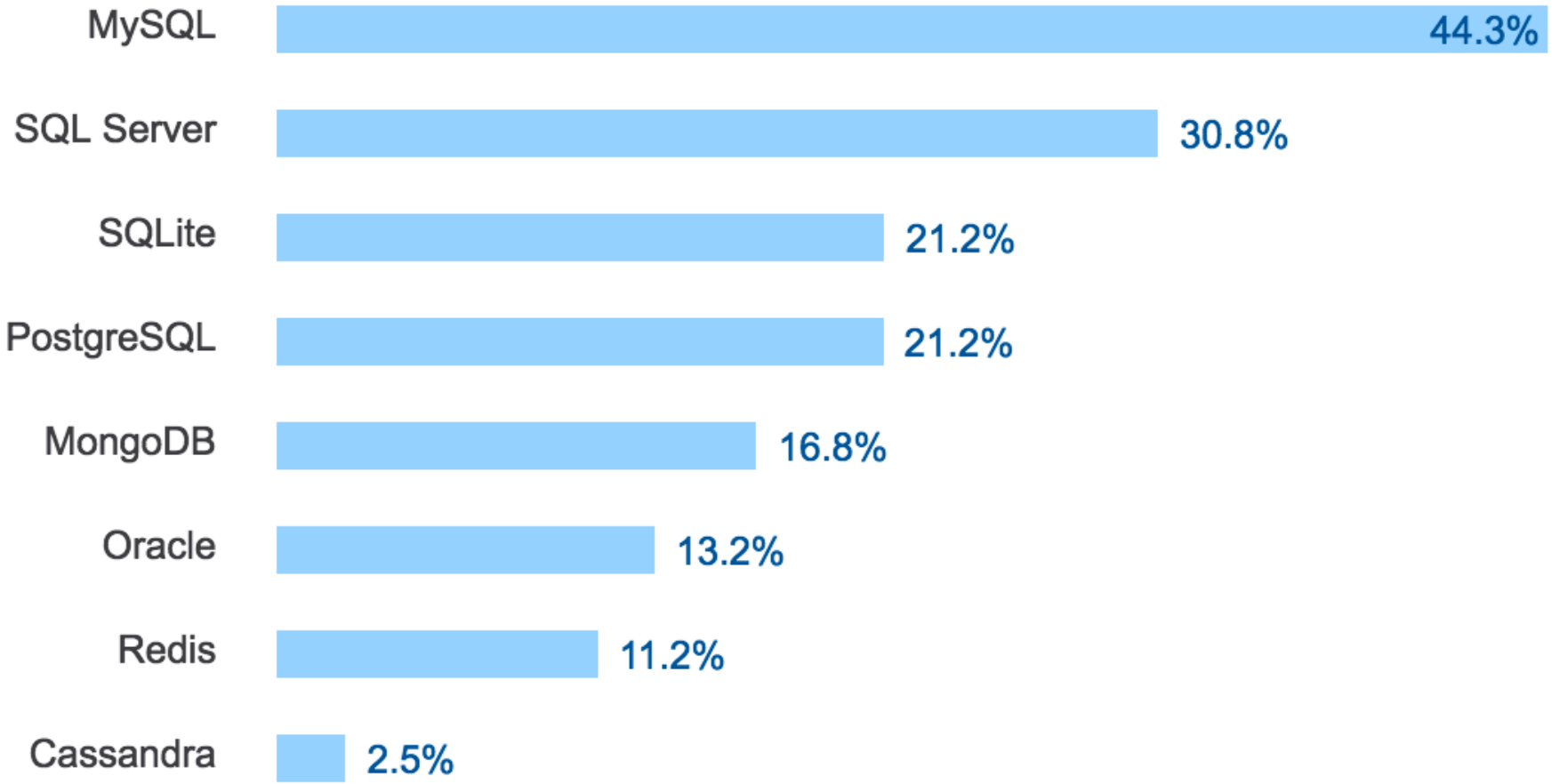
# Overview

# MySQL

Many of the world's largest and fastest-growing organizations including Facebook, Google, Adobe, Alcatel Lucent and Zappos rely on MySQL to save time and money powering their high-volume Web sites, business-critical systems and packaged software.

# database popularity

**Most popular databases in 2017 according to StackOverflow survey**




https://www.eversql.com/
most-popular-databases-in-2017-according-to-stackoverflow-sur

# 分析

1. The most popular database is MySQL, and not by far comes SQL Server. Almost half of the developers who answered the survey (44.3% out of 36,935 responders) are using MySQL. It seems RDBMS databases and specifically MySQL are not going anywhere anytime soon.
2. RDBMS databases are still significantly more common than NoSQL databases such as MongoDB.
3. Relatively new technologies are starting to gain market share in the databases world –Redis (first release at 2009) and Cassandra (first release at 2008).
4. Almost 1/4 of all programmers (23.3%) are using SQLite, which is a lite SQL database which is based on a single file. This small database software is gaining popularity among developers, probably mostly for simple and standalone applications.

# download and install MySQL

## Download MySQL and MySQL connectors

https://dev.mysql.com/downloads/workbench/(GUI 工具)
https://dev.mysql.com/downloads/mysql/
https://dev.mysql.com/downloads/connector/j/

## Install

一般地，对于 MySQL 和 MySQL workbench 默认下一步安装即可。MySQL Connector/J 解压，将 mysql-connector-java-xxx-bin.jar 文件加入到 tomcat 解压目录下的 lib 子目录中即可。

# JDBC

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database.
JDBC 是 oracle 公司提出的一个标准，其具体实现由各家数据库公司自己进行。

# JDBC 的功能

1. 同数据库建立连接
2. 向数据库发送 sql 语句
3. 处理从数据库返回的结果

# JDBC 中常用的接口

| 接口 | 作用 |
| --- | --- |
| Driver | 负责加载驱动 |
| DriverManager | 负责管理驱动和连接数据库 |
| Connection | 管理某个数据库连接 |
| Statement | 执行 sql 语句，并返回结果 |
| ResultSet | 获得检索结果 |

Table: JDBC 的常用接口

# 连接数据库的一般流程

jsp 连接数据库的一般流程如下：

1. 加载 jdbc 驱动程序
2. 创建数据库连接
3. 执行 sql 语句
4. 获取查询结果
5. 关闭连接

# 加载 jdbc 驱动

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ page import="java.sql.*" %>
<%
    try{

        Class.forName("com.mysql.jdbc.Driver");

    }catch(ClassNotFoundException e){
        System.out.println("驱动加载失败！");
        e.printStackTrace();
    }
%>
```

# 建立数据库连接

使用DriverManager类的 getConnection() 方法建立 sql 连接。

```
<%
    try{
        Class.forName("com.mysql.jdbc.Driver");

        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost??useUnicode=true&
            characterEncoding=UTF-8","root","password");
    }catch(Exception e){
        System.out.println("出现错误，具体内容如下：");
        e.printStackTrace();
    }
%>
```

# 执行 sql 语句

使用 Connection 类的对象conn 的 createStatement() 方法创建 Statement 类的对象st，再使用 st 对象来执行 sql 语句 (executeQuery() 或 executeUpdate())。

```
<%
    try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost/","root","password");
        Statement st = conn.createStatement();
        st.executeQuery("show databases;");
}catch(Exception e){
        System.out.println("出现错误，具体内容如下：");
        e.printStackTrace();
}
%>
```

# 获取查询结果并关闭数据库连接

通过 Statement 类的对象st来执行 sql 语句 (executeQuery() 或 executeUpdate())，会返回一个 ResultSet 类的对象rs，rs 对象拥有一些方法可以获取查询到的值。

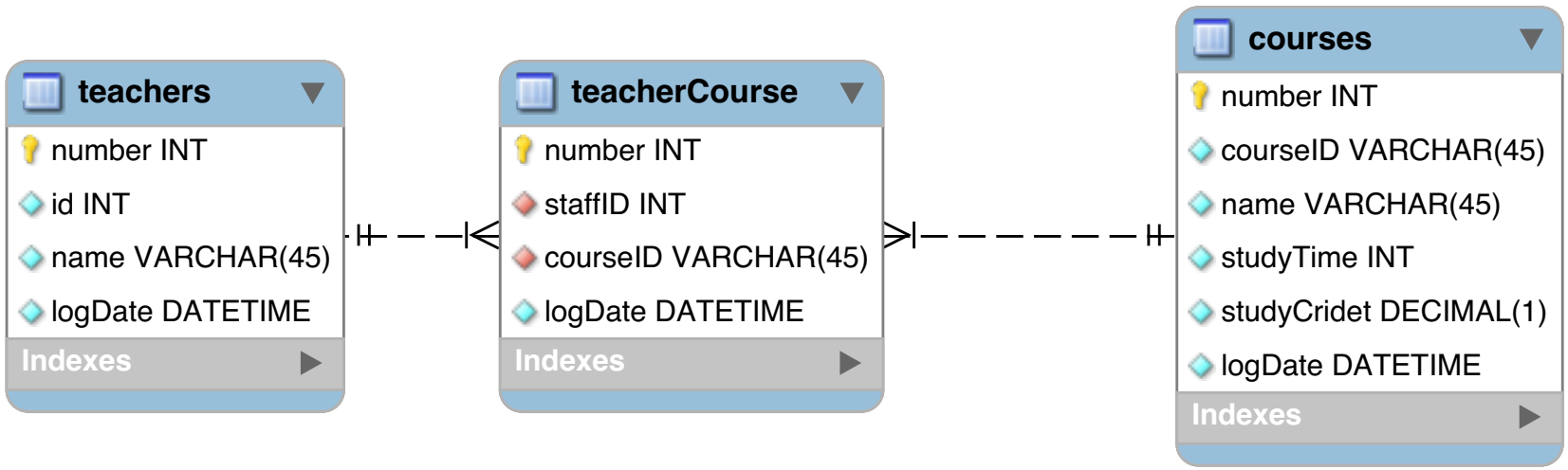
```
<%
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection(
     "jdbc:mysql://localhost/","root","password");
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery("show databases;");
    while(rs.next()){
    String str = rs.getString(1);
    System.out.println(str);
}
     conn.close();
}catch(Exception e){ e.printStackTrace(); }
%>
```

# 完整 java 代码

```
<%
 try{
     Class.forName("com.mysql.jdbc.Driver");
     Connection conn = DriverManager.getConnection(
      "jdbc:mysql://localhost/","root","password");
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery("show databases;");
     int column = rs.getMetaData().getColumnCount();
     System.out.println(column);
     while(rs.next()){
       System.out.println(rs.getString(1));
     }
      conn.close();
 }catch(Exception e){ e.printStackTrace(); }
%>
```

# 建表

接下来本 ppt 参照以下 ER 图建立数据库及表以进行演示。



# 信息录入页面


所开课程
本学期课表
仪器设备、元件
开放实验项目
联系方式
申请排课
测试
教师信息
姓名:
职工号:
Submit
Reset

# 信息录入界面关键代码

```
<form action="teacher_add.jsp" method="post">
<table>
<tr><td>姓名: <input type="text" name="nm"></td></tr>
<tr><td>
职工号: <input type="number" name="staffid" min=200600
</td></tr>
<tr><td><input type="submit"></td>
<td><input type="reset"></td></tr>
</table>
</form>
```

# 写入数据库代码 1/2

```
String teacher_name =
    request.getParameter("nm");
int teacher_id =
  Integer.parseInt(request.getParameter("staffid"));
Calendar c = Calendar.getInstance();
String d = c.get(Calendar.YEAR)+"-"+
    (c.get(Calendar.MONTH)+1)+"-"+
    c.get(Calendar.DATE)+" "+
    c.get(Calendar.HOUR_OF_DAY)+":"+
    c.get(Calendar.MINUTE)+":"+
    c.get(Calendar.SECOND);
```

# 写入数据库代码 2/2

```
String sql =
 "insert into teachers (id,name,logDate) values(?,?,?)
int rz=0;
try{
   Class.forName("com.mysql.jdbc.Driver");
   Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/lab","root","password");
   PreparedStatement prep = conn.prepareStatement(sql);
   prep.setInt(1, teacher_id);
   prep.setString(2,teacher_name);
   prep.setString(3, d);
   rz = prep.executeUpdate();
   System.out.println(rz);
   conn.close();
   if(rz==1){
      response.sendRedirect("teacher.jsp");
   }
```

# 查询数据结果

| 所开课程 | 本学期课表 | 仪器设备、元件 | 开放实验项目 | 联系方式 |
|---|---|---|---|---|

申请排课

测试

教师信息

开放实验项目

资料下载

当前系统中教师信息如下:

| id | name | 记载日期 |
|---|---|---|
| 20157517 | ??? | 0001-02-05 00:00:00.0 |
| 20060001 | 张海宁 | 2018-04-05 00:00:00.0 |
| 20060002 | Alex | 0001-02-05 00:00:00.0 |
| 20060003 | graft | 2018-03-06 18:07:23.0 |

姓名:

职工号:

Submit Reset

# 查询数据关键代码

```
<table>
<tr><td>id</td><td>name</td><td>记载日期</td></tr>
<%
  Class.forName("com.mysql.jdbc.Driver");
  Connection conn = DriverManager.getConnection(
   "jdbc:mysql://localhost/lab","root","password");
  Statement st = conn.createStatement();
  String sql = "select * from teachers;";
  ResultSet rs = st.executeQuery(sql);
  while(rs.next()){
  out.print("<tr><td>"); out.print(rs.getInt(2));
  out.print("</td><td>"); out.print(rs.getString(3));
  out.print("</td><td>"); out.print(rs.getString(4));
  out.print("</td></tr>");
%>
</table>
```

# session

与 cookie 对象类似，session 对象也是用来保存与用户请求有关的一些数据。服务器为每个用户生成一个 session 对象，用于保存用户的信息，跟踪用户的操作状态。session 使用 Map 这种数据结构来保存数据。其常用方法如下表所示：

| 方法 | 说明 |
| --- | --- |
| setAttribute(String name, Object obj) | 设置属性及对应的值 |
| invalidate() | 销毁 session 对象 |
| getAttribute(String name) | 获得指定属性的值 |
| getAttributeNames() | 获得 session 对象中所有的属性的 |

Table: session 对象常用方法

# 判断 session 中是否有需要的值

**nav.jsp**

```
<%
String user="";
Enumeration<String> enu = session.getAttributeNames();
while(enu.hasMoreElements()){
String attr = enu.nextElement();
if(attr.equals("user")){
user = (String)session.getAttribute("user");
out.print("你好, "+user+"。");
out.print("<form action=\"logout.jsp\" method=\"post\"
out.print("<input type=\"submit\" value=\"logout\"></f
 break; } }
if(user.equals("")){
out.print("<form action=\"login.jsp\" method=\"post\">
out.print("ID:<input type=\"number\" name=\"id\">");
out.print("<input type=\"submit\" value=\"login\"></fo
%>
```

# 查询数据库，创建 session

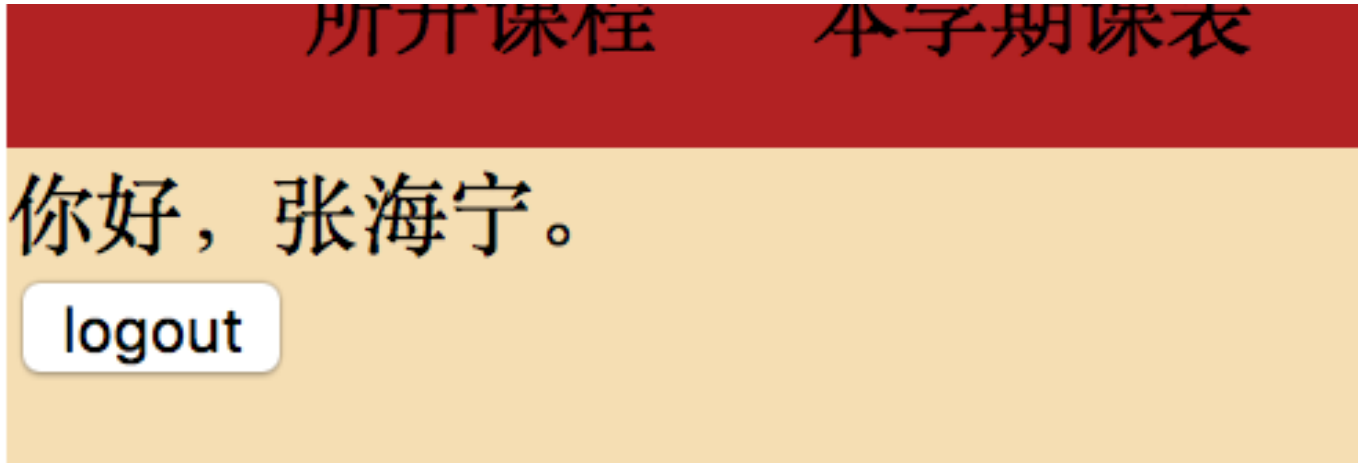
**login.jsp**

```
<%
int id = Integer.parseInt(request.getParameter("id"));
Db db = new Db();
Connection conn = db.getConnection();
String sql = "select name from teachers where id=?";
PreparedStatement pst = conn.prepareStatement(sql);
pst.setInt(1, id);
ResultSet rs = pst.executeQuery();
if(rs.first()){
   session.setAttribute("id", id);
   session.setAttribute("user", rs.getString("name"));
}
conn.close();
response.sendRedirect("index.jsp");
%>
```

# before and after login

before login:


after login:



# 页面显示和页面间传递中文乱码问题

```
<%@ page language="java"
    contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

- charset
  用于设定服务器响应的编码格式，即 tomcat 会根据此字符集对响应进行编码
- pageEncoding
  JSP 文件会以此编码进行保存

```
request.setCharacterEncoding("UTF-8");
String teacher_name = request.getParameter("nm");
```

- request.setCharacterEncoding("UTF-8")
  用于设置表单部分数据的编码格式，本条语句必须在 request.getParameter() 前调用

# JSP 写中文到 MySQL 的乱码问题

```
conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/lab?
    useUnicode=true&characterEncoding=UTF-8",
    "root","password");
```

**在连接数据库时，指定编码格式。**

# 作业

使用 session 对象，为本小组的项目增加登陆注销功能。

# The End