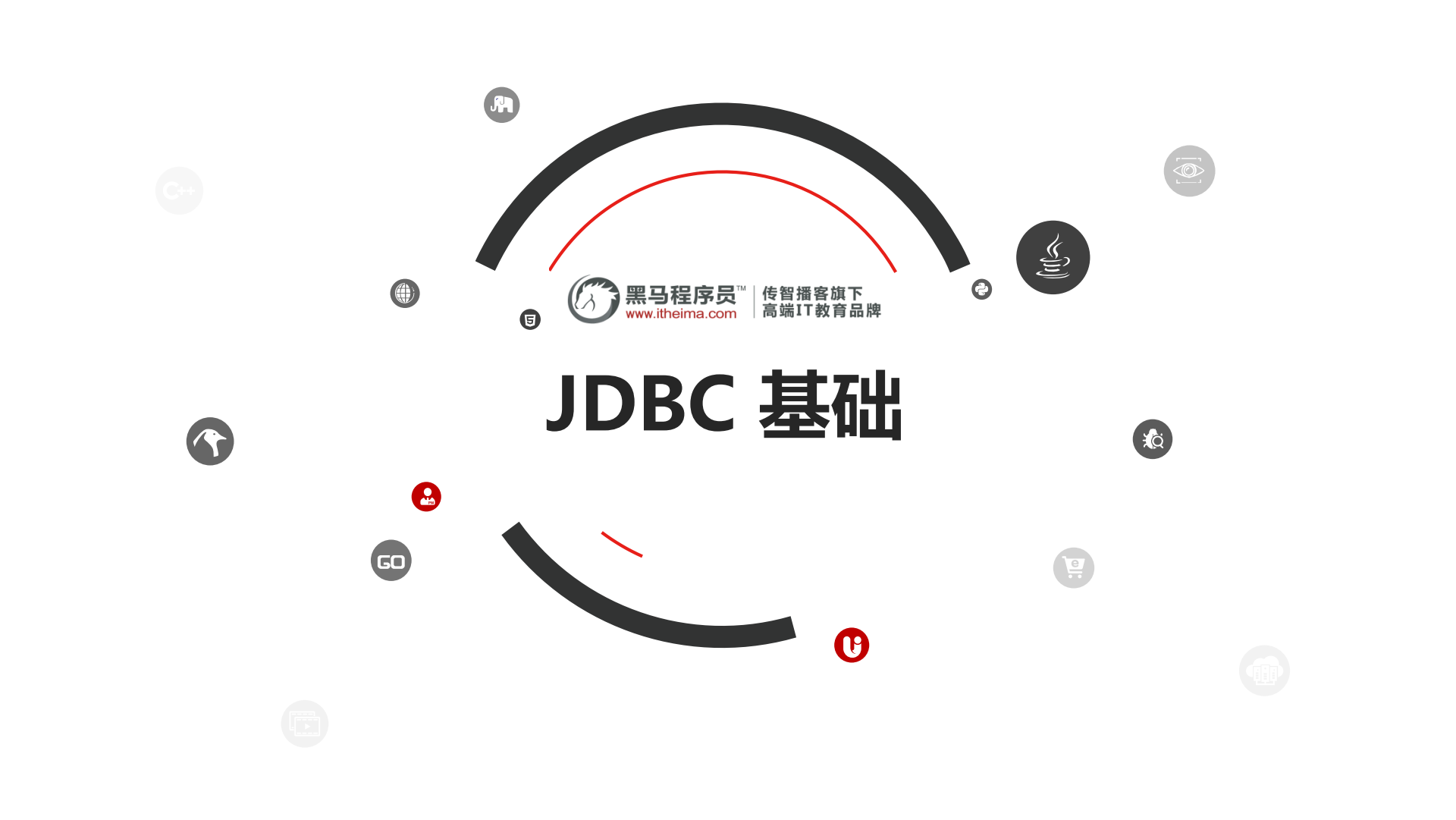



# JDBC 基础

# 目录 Contents

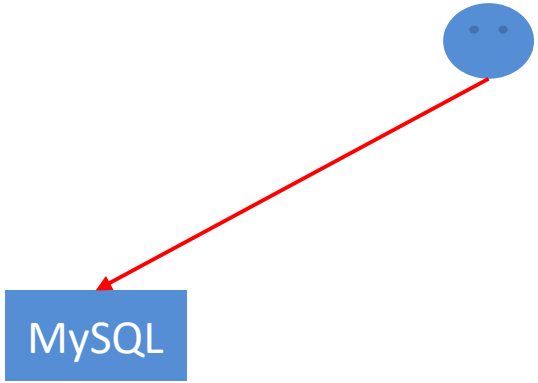
# JDBC 快速入门



## JDBC的概念

1. jdbc的概念
   - JDBC（Java DataBase Connectivity java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系型数据库提供统一访问，它是由一组用Java语言编写的类和接口组成的。

2. jdbc的本质
   - 其实就是java官方提供的一套规范(接口)。用于帮助开发人员快速实现不同关系型数据库的连接！



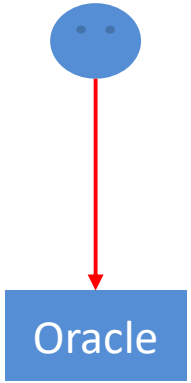
# JDBC 快速入门



## JDBC的概念

1. jdbc的概念
   - JDBC（Java DataBase Connectivity java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系型数据库提供统一访问，它是由一组用Java语言编写的类和接口组成的。

2. jdbc的本质
   - 其实就是java官方提供的一套规范(接口)。用于帮助开发人员快速实现不同关系型数据库的连接！



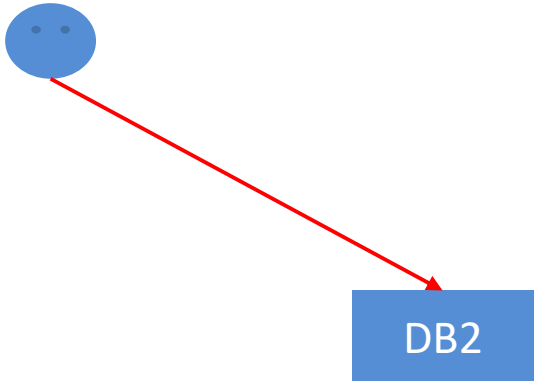
# JDBC 快速入门



## JDBC的概念

1. jdbc的概念
   - JDBC（Java DataBase Connectivity java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系型数据库提供统一访问，它是由一组用Java语言编写的类和接口组成的。

2. jdbc的本质
   - 其实就是java官方提供的一套规范(接口)。用于帮助开发人员快速实现不同关系型数据库的连接！



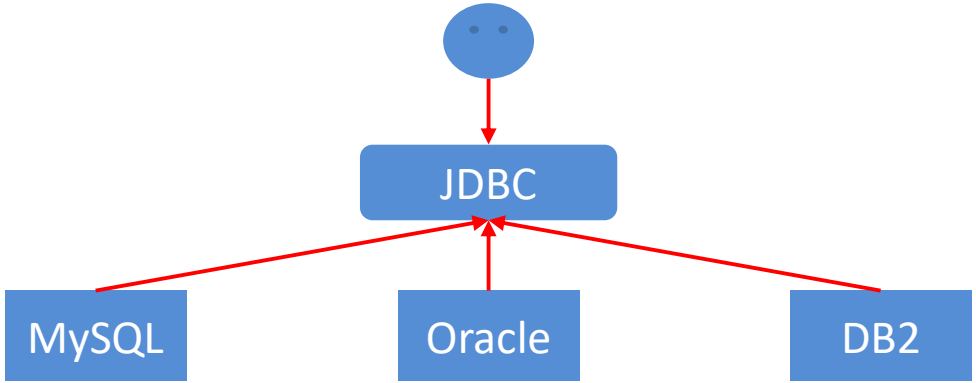
# JDBC 快速入门



## JDBC的概念

1. jdbc的概念
   - JDBC（Java DataBase Connectivity java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系型数据库提供统一访问，它是由一组用Java语言编写的类和接口组成的。

2. jdbc的本质
   - 其实就是java官方提供的一套规范(接口)。用于帮助开发人员快速实现不同关系型数据库的连接！



## JDBC的入门程序

3. jdbc的快速入门程序

① 导入jar包
② 注册驱动
③ 获取数据库连接
④ 获取执行者对象
⑤ 执行sql语句并返回结果
⑥ 处理结果
⑦ 释放资源

# 目录 Contents

# JDBC 功能类详解



## DriverManager

1. DriverManager驱动管理对象

① 注册驱动

- 注册给定的驱动程序：static void registerDriver(Driver driver);
- 写代码使用：Class.forName("com.mysql.jdbc.Driver");
- 在com.mysql.jdbc.Driver类中存在静态代码块

```
static {
    try {
        java.sql.DriverManager.registerDriver(new Driver());
    } catch (SQLException E) {
        throw new RuntimeException("Can't register driver!");
    }
}
```

注意：

- 我们不需要通过DriverManager调用静态方法registerDriver()，因为只要Driver类被使用，则会执行其静态代码块完成注册驱动
- mysql5之后可以省略注册驱动的步骤。在jar包中，存在一个java.sql.Driver配置文件，文件中指定了com.mysql.jdbc.Driver

# JDBC 功能类详解



## DriverManager

1. DriverManager驱动管理对象

② 获取数据库连接

获取数据库连接对象：static Connection getConnection(String url, String user, String password);

返回值：Connection 数据库连接对象

参数

url：指定连接的路径。语法：jdbc:mysql://ip地址(域名):端口号/数据库名称

user：用户名

password：密码

# JDBC 功能类详解



## Connection

1. Connection数据库连接对象

① 获取执行者对象

获取普通执行者对象：Statement createStatement();

获取预编译执行者对象：PreparedStatement prepareStatement(String sql);

② 管理事务

开启事务：setAutoCommit(boolean autoCommit);　参数为false，则开启事务。

提交事务：commit();

回滚事务：rollback();

③ 释放资源

立即将数据库连接对象释放：void close();

# JDBC 功能类详解



## Statement

1. Statement执行sql语句的对象
    ① 执行DML语句：int executeUpdate(String sql);
        返回值int：返回影响的行数。
        参数sql：可以执行insert、update、delete语句。
    ② 执行DQL语句：ResultSet executeQuery(String sql);
        返回值ResultSet：封装查询的结果。
        参数sql：可以执行select语句。
    ③ 释放资源
        立即将执行者对象释放：void close();

# JDBC 功能类详解



## ResultSet

1. ResultSet结果集对象

① 判断结果集中是否还有数据：boolean next();

有数据返回true，并将索引向下移动一行。

没有数据返回false。

② 获取结果集中的数据：XXX getXxx("列名");

XXX代表数据类型(要获取某列数据，这一列的数据类型)。

例如：String getString("name");　　int getInt("age");

③ 释放资源

立即将结果集对象释放：void close();

# 目录 Contents

# JDBC 案例



## 案例需求

使用JDBC完成对student表的CRUD操作

# JDBC 案例



## 数据准备

1. 创建数据库和数据表

```
-- 创建db14数据库
CREATE DATABASE db14;

-- 使用db14数据库
USE db14;

-- 创建student表
CREATE TABLE student(
    sid INT PRIMARY KEY AUTO_INCREMENT, -- 学生id
    NAME VARCHAR(20),                   -- 学生姓名
    age INT,                            -- 学生年龄
    birthday DATE,                      -- 学生生日
);

-- 添加数据
INSERT INTO student VALUES (NULL,'张三',23,'1999-09-23'),(NULL,'李四',24,'1998-08-
10'),(NULL,'王五',25,'1996-06-06'),(NULL,'赵六',26,'1994-10-20');
```

# JDBC 案例



## 数据准备

2. 创建Student类

```
public class Student {
    private Integer sid;
    private String name;
    private Integer age;
    private Date birthday;
```

注意：

自定义类的功能是为了封装表中每列数据，成员变量和列保持一致
所有基本数据类型需要使用对应包装类，以免表中null值无法赋值

# JDBC 案例



## 需求实现

1. 需求一：查询所有学生信息
2. 需求二：根据id查询学生信息
3. 需求三：新增学生信息
4. 需求四：修改学生信息
5. 需求五：删除学生信息

# 目录 Contents

# JDBC 工具类



## 抽取工具类

1. 编写配置文件

   在src目录下创建config.properties配置文件

```
driverClass=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/db14
username=root
password=root
```

2. 编写jdbc工具类

3. 使用jdbc工具类优化student表的CRUD操作

## 抽取工具类

4. Student表的CRUD操作整合页面

① 用户表的数据准备

```
-- 创建用户表
CREATE TABLE USER(
    uid VARCHAR(50) PRIMARY KEY,    -- 用户id
    ucode VARCHAR(50),              -- 用户标识
    loginname VARCHAR(100),         -- 登录用户名
    PASSWORD VARCHAR(100),          -- 登录密码
    username VARCHAR(100),          -- 用户名
    gender VARCHAR(10),             -- 用户性别
    birthday DATE,                  -- 出生日期
    dutydate DATE                   -- 入职日期
);

-- 添加一条测试数据
INSERT INTO `user` VALUES ('11111111', 'zhangsan001', 'zhangsan', '1234', '张三',
'男', '2008-10-28', '2018-10-28');
```

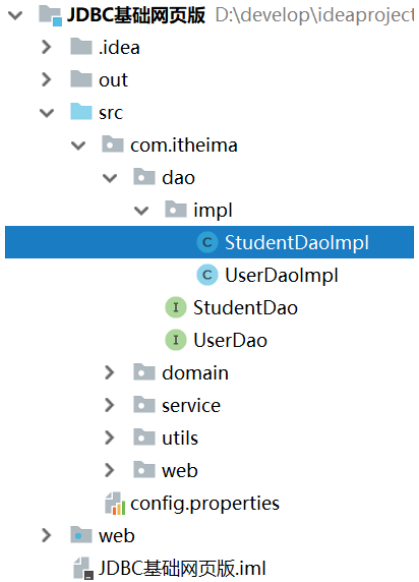
# JDBC 工具类



## 抽取工具类

4. Student表的CRUD操作整合页面

② 将dao层实现类复制到新项目的dao层即可



# 目录 Contents

# SQL 注入攻击



## SQL注入攻击

1. 什么是SQL注入攻击
    - 就是利用sql语句的漏洞来对系统进行攻击

2. SQL注入攻击的演示

3. SQL注入攻击的原理
    - 按照正常道理来说，我们在密码处输入的所有内容，都应该认为是密码的组成
    - 但是现在Statement对象在执行sql语句时，将密码的一部分内容当做查询条件来执行了

## SQL注入攻击

4. SQL注入攻击的解决

- PreparedStatement 预编译执行者对象
  - 在执行sql语句之前，将sql语句进行提前编译。明确sql语句的格式后，就不会改变了。剩余的内容都会认为是参数！
  - SQL语句中的参数使用?作为占位符
- 为?占位符赋值的方法：setXxx(参数1,参数2);
  - Xxx代表：数据类型
  - 参数1：?的位置编号(编号从1开始)
  - 参数2：?的实际参数

```
String sql = "DELETE FROM user WHERE name=?";
pstm = conn.prepareStatement(sql);
pstm.setString(1,"张三");
```

- 执行SQL语句
  - 执行insert、update、delete语句：int executeUpdate();
  - 执行select语句：ResultSet executeQuery();

# 目录 Contents

## JDBC管理事务

1. JDBC如何管理事务
    - 管理事务的功能类：Connection
        - 开启事务：setAutoCommit(boolean autoCommit)；参数为false，则开启事务。
        - 提交事务：commit();
        - 回滚事务：rollback();

2. 演示批量添加数据并在业务层管理事务

注意：事务的管理需要在业务层实现，因为dao层的功能要给很多模块提供功能的支撑，而有些模块是不需要事务的。


传智播客旗下高端IT教育品牌