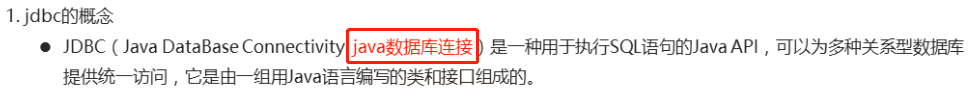
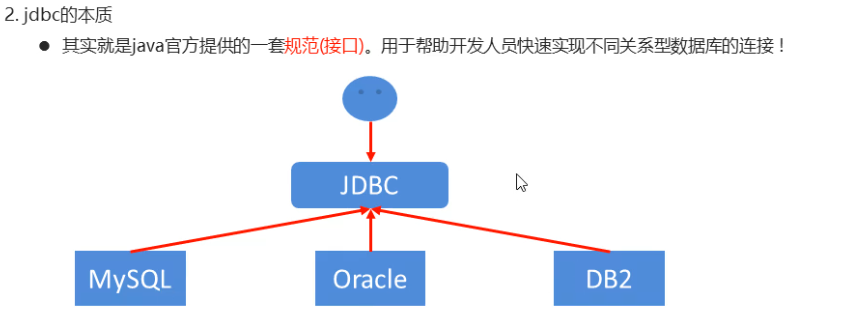
01 jdbc基础

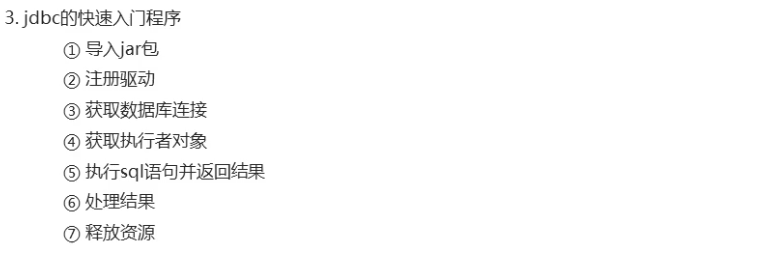
# 1（掌握）JDBC快速入门

## 1.1（掌握）jdbc概念





## 1.2（掌握）jdbc快速入门



### 源码

|  |
| --- |
| public class JDBCDemo01 {  public static void main(String[] args) throws Exception{  //1.导入jar包  //2.注册驱动  Class.forName("com.mysql.jdbc.Driver");  //3.获取连接  Connection con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //4.获取执行者对象  Statement stat = con.createStatement();  //5.执行sql语句，并且接收结果  String sql = "SELECT \* FROM user";  ResultSet rs = stat.executeQuery(sql);  //6.处理结果  while(rs.next()) {  System.out.println(rs.getInt("id") + "\t" + rs.getString("name"));  }  //7.释放资源  con.close();  stat.close();  con.close();  }  } |

# 2（掌握）JDBC功能类详解

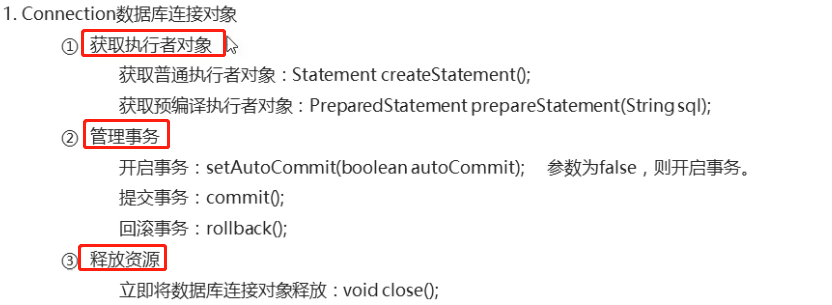
## 2.1（掌握）DriverManager



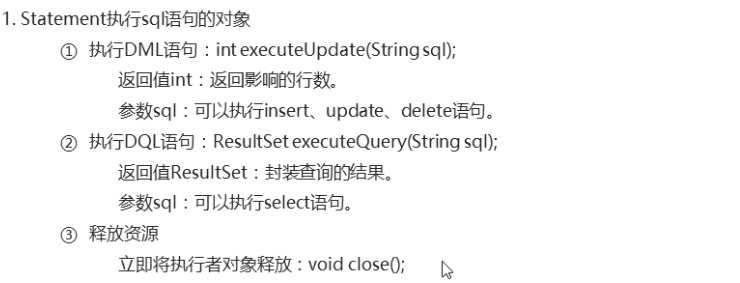




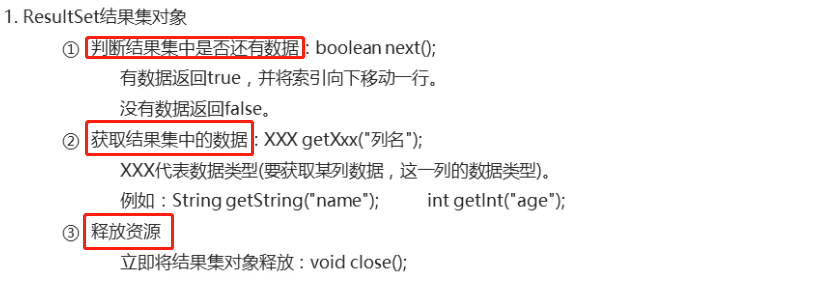
## 2.2（掌握）Connection



## 2.3（掌握）Statement



## 2.4（掌握）ResultSet



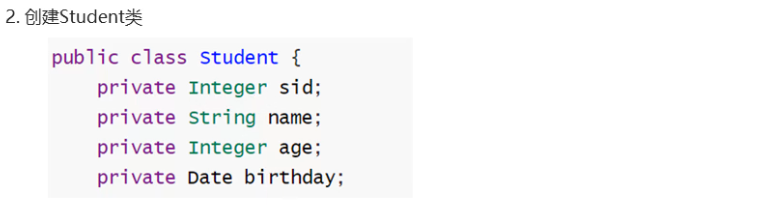
# 3（掌握）JDBC案例

## 3.1（掌握）需求介绍和数据准备



### 数据准备

|  |
| --- |
| -- 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'); |

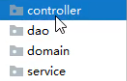




### 源码准备

|  |
| --- |
| public class Student {  private Integer sid;  private String name;  private Integer age;  private Date birthday;//使用java.util.Date类  public Student() {  }  public Student(Integer sid, String name, Integer age, Date birthday) {  this.sid = sid;  this.name = name;  this.age = age;  this.birthday = birthday;  }  public Integer getSid() {  return sid;  }  public void setSid(Integer sid) {  this.sid = sid;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public Integer getAge() {  return age;  }  public void setAge(Integer age) {  this.age = age;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  @Override  public String toString() {  return "Student{" +  "sid=" + sid +  ", name='" + name + '\'' +  ", age=" + age +  ", birthday=" + birthday +  '}';  }  } |

### 搭建三层架构



### 源码准备

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  /\*  查询所有学生信息  \*/  @Test  public void findAll() {  }  /\*  条件查询，根据id查询学生信息  \*/  @Test  public void findById() {  }  /\*  添加学生信息  \*/  @Test  public void insert() {  }  /\*  修改学生信息  \*/  @Test  public void update() {  }  /\*  删除学生信息  \*/  @Test  public void delete() {  }  } |

/dao/StudentDao.java

|  |
| --- |
| /\*  Dao层接口  \*/  public interface StudentDao {  //查询所有学生信息  public abstract ArrayList<Student> findAll();  //条件查询，根据id获取学生信息  public abstract Student findById(Integer id);  //新增学生信息  public abstract int insert(Student stu);  //修改学生信息  public abstract int update(Student stu);  //删除学生信息  public abstract int delete(Integer id);  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  查询所有学生信息  \*/  @Override  public ArrayList<Student> findAll() {    }  /\*  条件查询，根据id查询学生信息  \*/  @Override  public Student findById(Integer id) {    }  /\*  添加学生信息  \*/  @Override  public int insert(Student stu) {    }  /\*  修改学生信息  \*/  @Override  public int update(Student stu) {    }  /\*  删除学生信息  \*/  @Override  public int delete(Integer id) {    }  } |

/service/StudentService.java

|  |
| --- |
| /\*  Service层接口  \*/  public interface StudentService {  //查询所有学生信息  public abstract ArrayList<Student> findAll();  //条件查询，根据id获取学生信息  public abstract Student findById(Integer id);  //新增学生信息  public abstract int insert(Student stu);  //修改学生信息  public abstract int update(Student stu);  //删除学生信息  public abstract int delete(Integer id);  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  private StudentDao dao = new StudentDaoImpl();  /\*  查询所有学生信息  \*/  @Override  public ArrayList<Student> findAll() {  return dao.findAll();  }  /\*  条件查询，根据id查询学生信息  \*/  @Override  public Student findById(Integer id) {  return dao.findById(id);  }  /\*  添加学生信息  \*/  @Override  public int insert(Student stu) {  return dao.insert(stu);  }  /\*  修改学生信息  \*/  @Override  public int update(Student stu) {  return dao.update(stu);  }  /\*  删除学生信息  \*/  @Override  public int delete(Integer id) {  return dao.delete(id);  }  } |

## 3.2（掌握）需求一：查询所有学生信息

### 源码

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  private StudentService service = new StudentServiceImpl();  /\*  查询所有学生信息  \*/  @Test  public void findAll() {  ArrayList<Student> list = service.findAll();  for(Student stu : list) {  System.out.println(stu);  }  }  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  查询所有学生信息  \*/  @Override  public ArrayList<Student> findAll() {  ArrayList<Student> list = new ArrayList<>();  Connection con = null;  Statement stat = null;  ResultSet rs = null;  try{  con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "SELECT \* FROM student";  rs = stat.executeQuery(sql);  //5.处理结果集  while(rs.next()) {  Integer sid = rs.getInt("sid");  String name = rs.getString("name");  Integer age = rs.getInt("age");  Date birthday = rs.getDate("birthday");  //封装Student对象  Student stu = new Student(sid,name,age,birthday);  //将student对象保存到集合中  list.add(stu);  }  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(rs != null) {  try {  rs.close();  } catch (SQLException e) {  e.printStackTrace();  }  } }  //将集合对象返回  return list;  }  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  private StudentDao dao = new StudentDaoImpl();  /\*  查询所有学生信息  \*/  @Override  public ArrayList<Student> findAll() {  return dao.findAll();  }  } |

## 3.3（掌握）需求二：根据id查询学生信息

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  private StudentService service = new StudentServiceImpl();  /\*  条件查询，根据id查询学生信息  \*/  @Test  public void findById() {  Student stu = service.findById(3);  System.out.println(stu);  }  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  条件查询，根据id查询学生信息  \*/  @Override  public Student findById(Integer id) {  Student stu = new Student();  Connection con = null;  Statement stat = null;  ResultSet rs = null;  try{  con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "SELECT \* FROM student WHERE sid='"+id+"'";  rs = stat.executeQuery(sql);  //5.处理结果集  while(rs.next()) {  Integer sid = rs.getInt("sid");  String name = rs.getString("name");  Integer age = rs.getInt("age");  Date birthday = rs.getDate("birthday");  //封装Student对象  stu.setSid(sid);  stu.setName(name);  stu.setAge(age);  stu.setBirthday(birthday);  }  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(rs != null) {  try {  rs.close();  } catch (SQLException e) {  e.printStackTrace();  }  } }  //将对象返回  return stu;  }  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  /\*  条件查询，根据id查询学生信息  \*/  @Override  public Student findById(Integer id) {  return dao.findById(id);  }  } |

## 3.4（掌握）需求三：添加学生信息

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  /\*  添加学生信息  \*/  @Test  public void insert() {  Student stu = new Student(5,"周七",27,new Date());  int result = service.insert(stu);  if(result!=0) {  System.out.println("添加成功");  }else {  System.out.println("添加失败");  }  }  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  添加学生信息  \*/  @Override  public int insert(Student stu) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  Date d = stu.getBirthday();  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  String birthday = sdf.format(d);  String sql = "INSERT INTO student VALUES ('"+stu.getSid()+"','"+stu.getName()+"','"+stu.getAge()+"','"+birthday+"')";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  //将结果返回  return result;  }  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  /\*  添加学生信息  \*/  @Override  public int insert(Student stu) {  return dao.insert(stu);  }  } |

## 3.5（掌握）需求四：修改学生信息

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  /\*  修改学生信息  \*/  @Test  public void update() {  Student stu = service.findById(5);  stu.setName("周七七");  int result = service.update(stu);  if(result != 0) {  System.out.println("修改成功");  }else {  System.out.println("修改失败");  }  }  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  修改学生信息  \*/  @Override  public int update(Student stu) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  Date d = stu.getBirthday();  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  String birthday = sdf.format(d);  String sql = "UPDATE student SET sid='"+stu.getSid()+"',name='"+stu.getName()+"',age='"+stu.getAge()+"',birthday='"+birthday+"' WHERE sid='"+stu.getSid()+"'";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  //将结果返回  return result;  }  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  /\*  修改学生信息  \*/  @Override  public int update(Student stu) {  return dao.update(stu);  }  } |

## 3.6（掌握）需求五：删除学生信息

/controller/StudentController.java

|  |
| --- |
| public class StudentController {  /\*  删除学生信息  \*/  @Test  public void delete() {  int result = service.delete(5);  if(result != 0) {  System.out.println("删除成功");  }else {  System.out.println("删除失败");  }  }  } |

/dao/StudentDaoImpl.java

|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  删除学生信息  \*/  @Override  public int delete(Integer id) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = DriverManager.getConnection("jdbc:mysql://192.168.59.129:3306/db2","root","itheima");  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "DELETE FROM student WHERE sid='"+id+"'";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  }  //将结果返回  return result;  }  } |

/service/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements StudentService {  /\*  删除学生信息  \*/  @Override  public int delete(Integer id) {  return dao.delete(id);  }  } |

# 4（掌握）JDBC工具类

## 4.1（掌握）配置文件编写

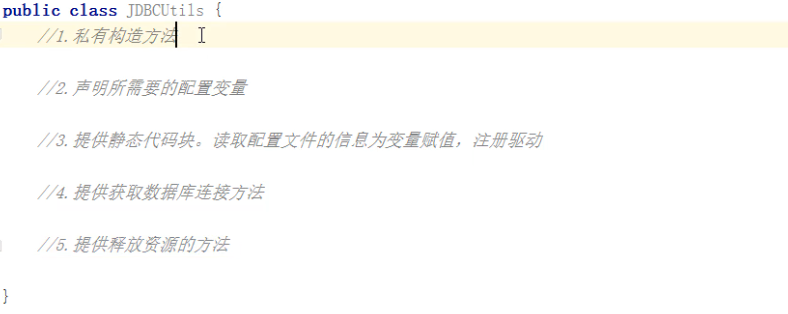


### 源码

src/config.properties

|  |
| --- |
| driverClass=com.mysql.jdbc.Driver  url=jdbc:mysql://192.168.59.129:3306/db14  username=root  password=itheima |

## 4.2（掌握）工具类的编写



### 源码

|  |
| --- |
| /\*  JDBC工具类  \*/  public class JDBCUtils {  //1.私有构造方法  private JDBCUtils(){}  //2.声明所需要的配置变量  private static String driverClass;  private static String url;  private static String username;  private static String password;  private static Connection con;  //3.提供静态代码块。读取配置文件的信息为变量赋值，注册驱动  static{  try {  //读取配置文件的信息为变量赋值  InputStream is = JDBCUtils.class.getClassLoader().getResourceAsStream("config.properties");  Properties prop = new Properties();  prop.load(is);  driverClass = prop.getProperty("driverClass");  url = prop.getProperty("url");  username = prop.getProperty("username");  password = prop.getProperty("password");  //注册驱动  Class.forName(driverClass);  } catch (Exception e) {  e.printStackTrace();  }  }  //4.提供获取数据库连接方法  public static Connection getConnection() {  try {  con = DriverManager.getConnection(url,username,password);  } catch (SQLException e) {  e.printStackTrace();  }  return con;  }  //5.提供释放资源的方法  public static void close(Connection con, Statement stat, ResultSet rs) {  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(rs != null) {  try {  rs.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  public static void close(Connection con, Statement stat) {  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  if(stat != null) {  try {  stat.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }  } |

## 4.3（掌握）优化学生案例



### 源码

/dao/StudentDaoImpl.java

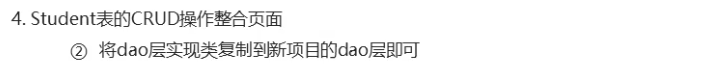
|  |
| --- |
| public class StudentDaoImpl implements StudentDao {  /\*  查询所有学生信息  \*/  @Override  public ArrayList<Student> findAll() {  ArrayList<Student> list = new ArrayList<>();  Connection con = null;  Statement stat = null;  ResultSet rs = null;  try{  con = JDBCUtils.getConnection();  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "SELECT \* FROM student";  rs = stat.executeQuery(sql);  //5.处理结果集  while(rs.next()) {  Integer sid = rs.getInt("sid");  String name = rs.getString("name");  Integer age = rs.getInt("age");  Date birthday = rs.getDate("birthday");  //封装Student对象  Student stu = new Student(sid,name,age,birthday);  //将student对象保存到集合中  list.add(stu);  }  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  JDBCUtils.close(con,stat,rs);  }  //将集合对象返回  return list;  }  /\*  条件查询，根据id查询学生信息  \*/  @Override  public Student findById(Integer id) {  Student stu = new Student();  Connection con = null;  Statement stat = null;  ResultSet rs = null;  try{  con = JDBCUtils.getConnection();  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "SELECT \* FROM student WHERE sid='"+id+"'";  rs = stat.executeQuery(sql);  //5.处理结果集  while(rs.next()) {  Integer sid = rs.getInt("sid");  String name = rs.getString("name");  Integer age = rs.getInt("age");  Date birthday = rs.getDate("birthday");  //封装Student对象  stu.setSid(sid);  stu.setName(name);  stu.setAge(age);  stu.setBirthday(birthday);  }  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  JDBCUtils.close(con,stat,rs);  }  //将对象返回  return stu;  }  /\*  添加学生信息  \*/  @Override  public int insert(Student stu) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = JDBCUtils.getConnection();  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  Date d = stu.getBirthday();  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  String birthday = sdf.format(d);  String sql = "INSERT INTO student VALUES ('"+stu.getSid()+"','"+stu.getName()+"','"+stu.getAge()+"','"+birthday+"')";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  JDBCUtils.close(con,stat);  }  //将结果返回  return result;  }  /\*  修改学生信息  \*/  @Override  public int update(Student stu) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = JDBCUtils.getConnection();  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  Date d = stu.getBirthday();  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  String birthday = sdf.format(d);  String sql = "UPDATE student SET sid='"+stu.getSid()+"',name='"+stu.getName()+"',age='"+stu.getAge()+"',birthday='"+birthday+"' WHERE sid='"+stu.getSid()+"'";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  JDBCUtils.close(con,stat);  }  //将结果返回  return result;  }  /\*  删除学生信息  \*/  @Override  public int delete(Integer id) {  Connection con = null;  Statement stat = null;  int result = 0;  try{  con = JDBCUtils.getConnection();  //3.获取执行者对象  stat = con.createStatement();  //4.执行sql语句，并且接收返回的结果集  String sql = "DELETE FROM student WHERE sid='"+id+"'";  result = stat.executeUpdate(sql);  } catch(Exception e) {  e.printStackTrace();  } finally {  //6.释放资源  JDBCUtils.close(con,stat);  }  //将结果返回  return result;  }  } |

## 4.4（掌握）学生表操作整合页面

### 数据准备

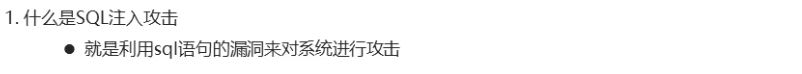
|  |
| --- |
| -- 创建用户表  CREATE TABLE USER(  uid VARCHAR(50) PRIMARY KEY, -- 用户id  ucode VARCHAR(50), -- 用户标识  loginname VARCHAR(100), -- 登录用户名  PASSWORD VARCHAR(100), -- 登录密码  username VARCHAR(100), -- 用户名  gender VARCHAR(10), -- 用户性别  birthday DATE, -- 出生日期  dutydate DATE -- 入职日期  );  -- 添加一条测试数据  INSERT INTO USER VALUES ('11111111', 'zhangsan001', 'zhangsan', '1234', '张三', '男', '2008-10-28', '2018-10-28'); |

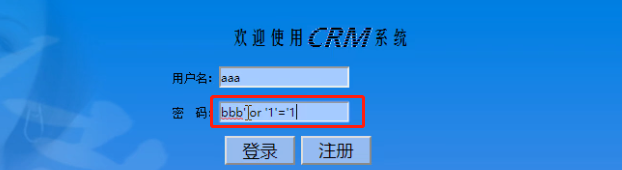
**导入JDBC基础网页版**

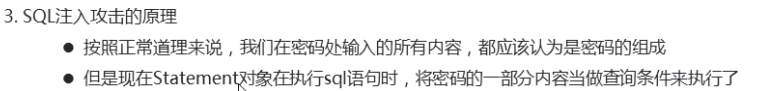


# 5（了解）SQL注入攻击

## 5.1（了解）sql注入攻击的演示







**注意：**

该项目需要使用 / 作为Application context。

## 5.2（掌握）sql注入攻击的解决



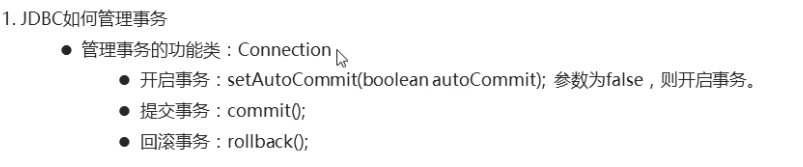
### 源码

JDBC基础网页版/src/com/itheima/dao/impl/UserDaoImpl.java

|  |
| --- |
| public class UserDaoImpl implements UserDao {  /\*  使用PreparedStatement登陆方法，解决注入攻击  \*/  @Override  public User findByLoginNameAndPassword(String loginName, String password) {  //定义必要信息  Connection conn = null;  PreparedStatement st = null;  ResultSet rs = null;  User user = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.定义SQL语句  String sql = "SELECT \* FROM user WHERE loginname=? AND password=?";  //3.获取操作对象，执行sql语句，获取结果集  st = conn.prepareStatement(sql);  st.setString(1,loginName);  st.setString(2,password);  rs = st.executeQuery();  //4.获取结果集  if (rs.next()) {  //5.封装  user = new User();  user.setUid(rs.getString("uid"));  user.setUcode(rs.getString("ucode"));  user.setUsername(rs.getString("username"));  user.setPassword(rs.getString("password"));  user.setGender(rs.getString("gender"));  user.setDutydate(rs.getDate("dutydate"));  user.setBirthday(rs.getDate("birthday"));  user.setLoginname(rs.getString("loginname"));  }  //6.返回  return user;  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,st,rs);  }  }  /\*  使用Statement的登录方法，有注入攻击  \*/  /\*@Override  public User findByLoginNameAndPassword(String loginName, String password) {  //定义必要信息  Connection conn = null;  Statement st = null;  ResultSet rs = null;  User user = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.定义SQL语句  String sql = "SELECT \* FROM user WHERE loginname='"+loginName+"' AND password='"+password+"'";  System.out.println(sql);  //3.获取操作对象，执行sql语句，获取结果集  st = conn.createStatement();  rs = st.executeQuery(sql);  //4.获取结果集  if (rs.next()) {  //5.封装  user = new User();  user.setUid(rs.getString("uid"));  user.setUcode(rs.getString("ucode"));  user.setUsername(rs.getString("username"));  user.setPassword(rs.getString("password"));  user.setGender(rs.getString("gender"));  user.setDutydate(rs.getDate("dutydate"));  user.setBirthday(rs.getDate("birthday"));  user.setLoginname(rs.getString("loginname"));  }  //6.返回  return user;  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,st,rs);  }  }\*/  @Override  public List<User> findAll() {  //定义必要信息  Connection conn = null;  PreparedStatement pstm = null;  ResultSet rs = null;  List<User> users = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.获取操作对象  pstm = conn.prepareStatement("select \* from user");  //3.执行sql语句，获取结果集  rs = pstm.executeQuery();  //4.遍历结果集  users = new ArrayList<User>();  while (rs.next()) {  //5.封装  User user = new User();  user.setUid(rs.getString("uid"));  user.setUcode(rs.getString("ucode"));  user.setUsername(rs.getString("username"));  user.setPassword(rs.getString("password"));  user.setGender(rs.getString("gender"));  user.setDutydate(rs.getDate("dutydate"));  user.setBirthday(rs.getDate("birthday"));  user.setLoginname(rs.getString("loginname"));  //加入到集合中  users.add(user);  }  //6.返回  return users;  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,pstm,rs);  }  }  @Override  public User findById(String uid) {  //定义必要信息  Connection conn = null;  PreparedStatement pstm = null;  ResultSet rs = null;  User user = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.获取操作对象  pstm = conn.prepareStatement("select \* from user where uid = ? ");  //3.设置参数  pstm.setString(1,uid);  //4.执行sql语句，获取结果集  rs = pstm.executeQuery();  //5.获取结果集  if (rs.next()) {  //6.封装  user = new User();  user.setUid(rs.getString("uid"));  user.setUcode(rs.getString("ucode"));  user.setUsername(rs.getString("username"));  user.setPassword(rs.getString("password"));  user.setGender(rs.getString("gender"));  user.setDutydate(rs.getDate("dutydate"));  user.setBirthday(rs.getDate("birthday"));  user.setLoginname(rs.getString("loginname"));  }  //7.返回  return user;  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,pstm,rs);  }  }  @Override  public void update(User user) {  //定义必要信息  Connection conn = null;  PreparedStatement pstm = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.获取操作对象  pstm = conn.prepareStatement("update user set ucode=?,loginname=?,password=?,username=?,gender=?,birthday=?,dutydate=? where uid=?");  //3.设置参数  pstm.setString(1,user.getUcode());  pstm.setString(2,user.getLoginname());  pstm.setString(3,user.getPassword());  pstm.setString(4,user.getUsername());  pstm.setString(5,user.getGender());  pstm.setDate(6,new Date(user.getBirthday().getTime()));  pstm.setDate(7,new Date(user.getDutydate().getTime()));  pstm.setString(8,user.getUid());  //4.执行sql语句，获取结果集  pstm.executeUpdate();  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,pstm);  }  }  @Override  public void delete(String uid) {  //定义必要信息  Connection conn = null;  PreparedStatement pstm = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.获取操作对象  pstm = conn.prepareStatement("delete from user where uid=?");  //3.设置参数  pstm.setString(1,uid);  //4.执行sql语句，获取结果集  pstm.executeUpdate();  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,pstm);  }  }  /\*  添加用户数据  \*/  @Override  public void save(User user) {  //定义必要信息  Connection conn = null;  PreparedStatement pstm = null;  try {  //1.获取连接  conn = JDBCUtils.getConnection();  //2.获取操作对象  pstm = conn.prepareStatement("insert into user(uid,ucode,loginname,password,username,gender,birthday,dutydate)values(?,?,?,?,?,?,?,?)");  //3.设置参数  pstm.setString(1,user.getUid());  pstm.setString(2,user.getUcode());  pstm.setString(3,user.getLoginname());  pstm.setString(4,user.getPassword());  pstm.setString(5,user.getUsername());  pstm.setString(6,user.getGender());  pstm.setDate(7,new Date(user.getBirthday().getTime()));  pstm.setDate(8,new Date(user.getDutydate().getTime()));  //4.执行sql语句，获取结果集  pstm.executeUpdate();  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(conn,pstm);  }  }  /\*  支持事务的添加  \*/  @Override  public void save(Connection connection, User user) {  //定义必要信息  PreparedStatement pstm = null;  try {  //2.获取操作对象  pstm = connection.prepareStatement("insert into user(uid,ucode,loginname,password,username,gender,birthday,dutydate)values(?,?,?,?,?,?,?,?)");  //3.设置参数  pstm.setString(1,user.getUid());  pstm.setString(2,user.getUcode());  pstm.setString(3,user.getLoginname());  pstm.setString(4,user.getPassword());  pstm.setString(5,user.getUsername());  pstm.setString(6,user.getGender());  pstm.setDate(7,new Date(user.getBirthday().getTime()));  pstm.setDate(8,new Date(user.getDutydate().getTime()));  //4.执行sql语句，获取结果集  pstm.executeUpdate();  }catch (Exception e){  throw new RuntimeException(e);  }finally {  JDBCUtils.close(null,pstm,null);  }  }  } |

# 6（掌握）JDBC事务

## 6.1（掌握）jdbc管理事务的介绍



## 6.2（掌握）jdbc管理事务的演示

操作项目：JDBC基础网页版

### 源码

|  |
| --- |
| public class UserServiceImpl implements UserService {  private UserDao userDao = new UserDaoImpl();  /\*  事务要控制在此处  \*/  @Override  public void batchAdd(List<User> users) {  //获取数据库连接对象  Connection con = JDBCUtils.getConnection();  try {  //开启事务  con.setAutoCommit(false);  for (User user : users) {  //1.创建ID,并把UUID中的-替换  String uid = UUID.randomUUID().toString().replace("-", "").toUpperCase();  //2.给user的uid赋值  user.setUid(uid);  //3.生成员工编号  user.setUcode(uid);  //出现异常  //int n = 1 / 0;  //4.保存  userDao.save(con,user);  }  //提交事务  con.commit();  }catch (Exception e){  //回滚事务  try {  con.rollback();  } catch (SQLException e1) {  e1.printStackTrace();  }  e.printStackTrace();  } finally {  //释放资源  JDBCUtils.close(con,null);  }  }  } |