JDBC
(Java Data Base Connectivity)

王新阳
wxyyuppie@bjfu.edu.cn

主要内容

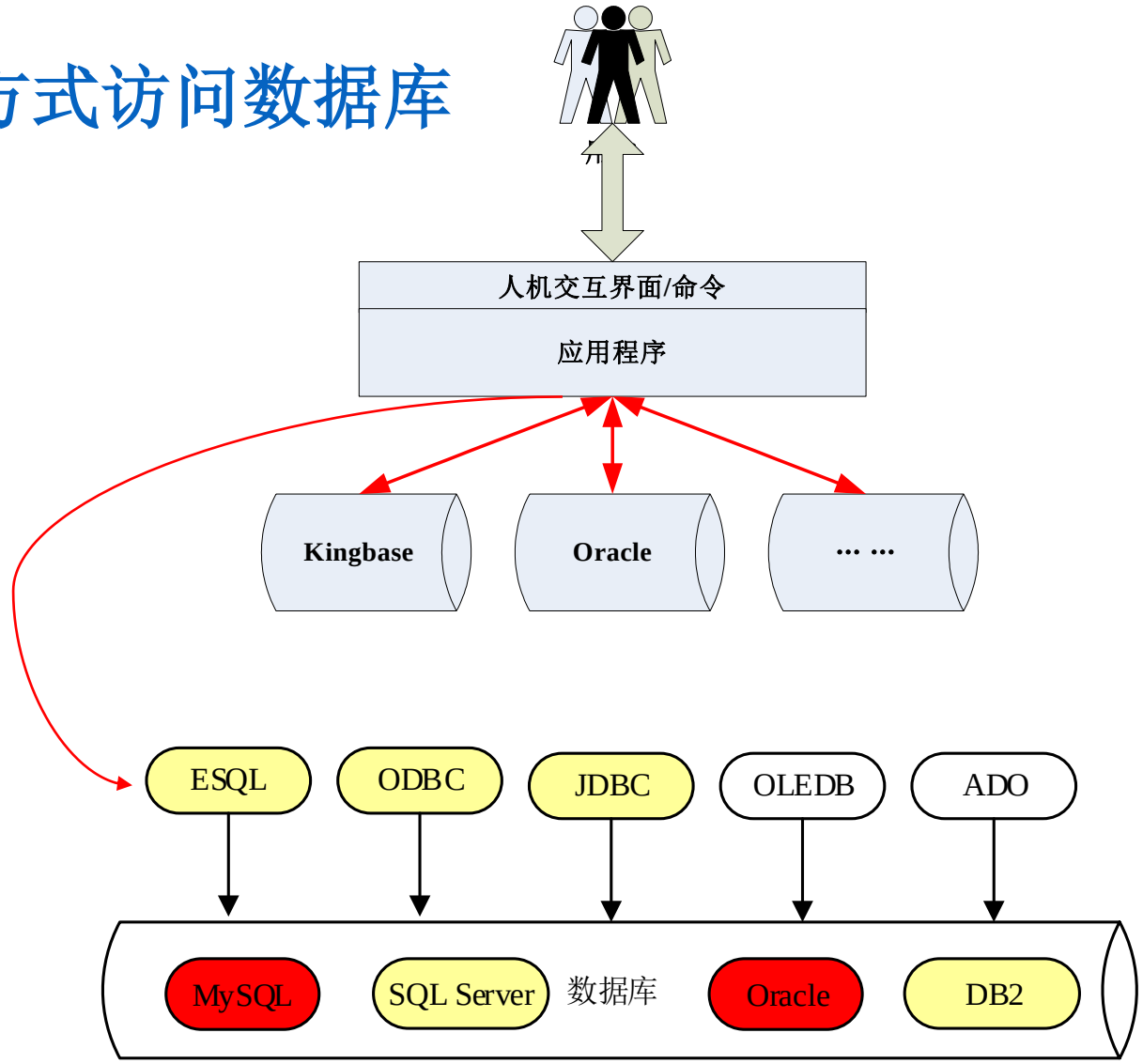
- JDBC介绍
- JDBC数据库基本编程
- JDBC高级编程

JDBC介绍

高级语言如何与数据库交互

- 高级语言的应用程序需要特定的方式访问数据库

- 特定的方式
  - JDBC（Java 语言）
  - ODBC（C、C++、Python等）
  - … …



JDBC的作用

- 作用
  - JDBC本质上是一系列的应用程序接口(API)
  - 通过JAVA语言访问任何结构化数据库
  - 通过JDBC API写出的程序,能够将SQL语句发送到相应的任何一种数据库
  - 由java.sql和javax.sql包组成
    - ✓import java.sql.*;
    - ✓import javax.sql.*;

JDBC的作用

- 作用

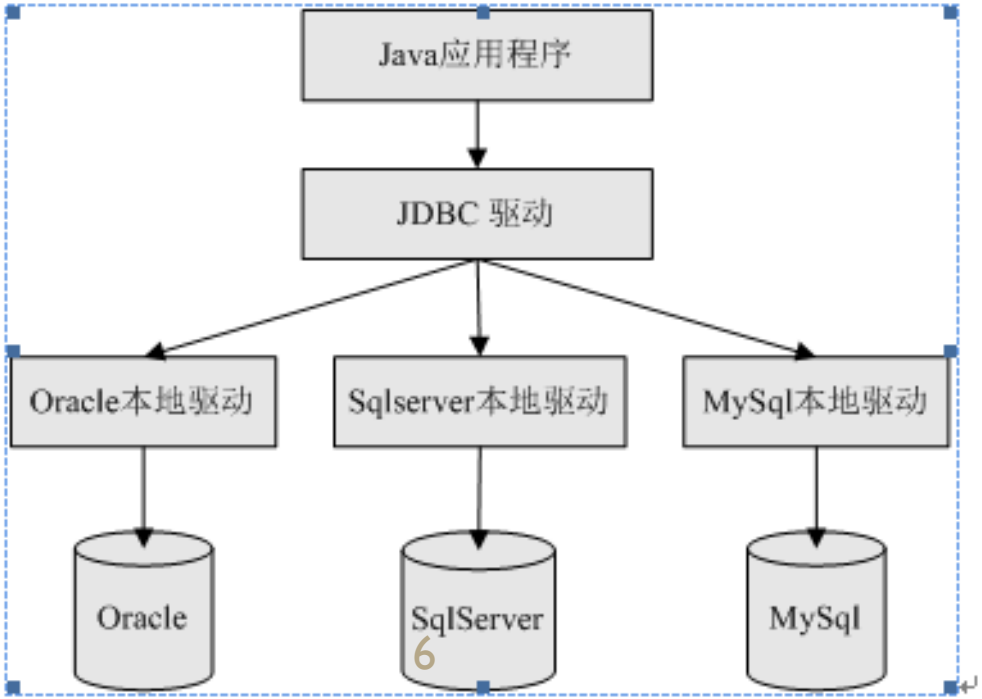
✓ 通过使用JDBC，开发人员可以很方便地将SQL语句传送给几乎任何一种数据库。也就是说，开发人员可以不必写一个程序访问SyBase，写另一个程序访问Oracle，再写一个程序访问Microsoft的SQLServer。用JDBC写的程序能够自动地将SQL语句传送给相应的数据库管理系统（DBMS）。不但如此，使用Java编写的应用程序可以在任何支持Java的平台上运行，不必在不同的平台上编写不同的应用。

简单地说，JDBC能完成下列三件事：

（1）同一个数据库建立连接；

（2）向数据库发送SQL语句；

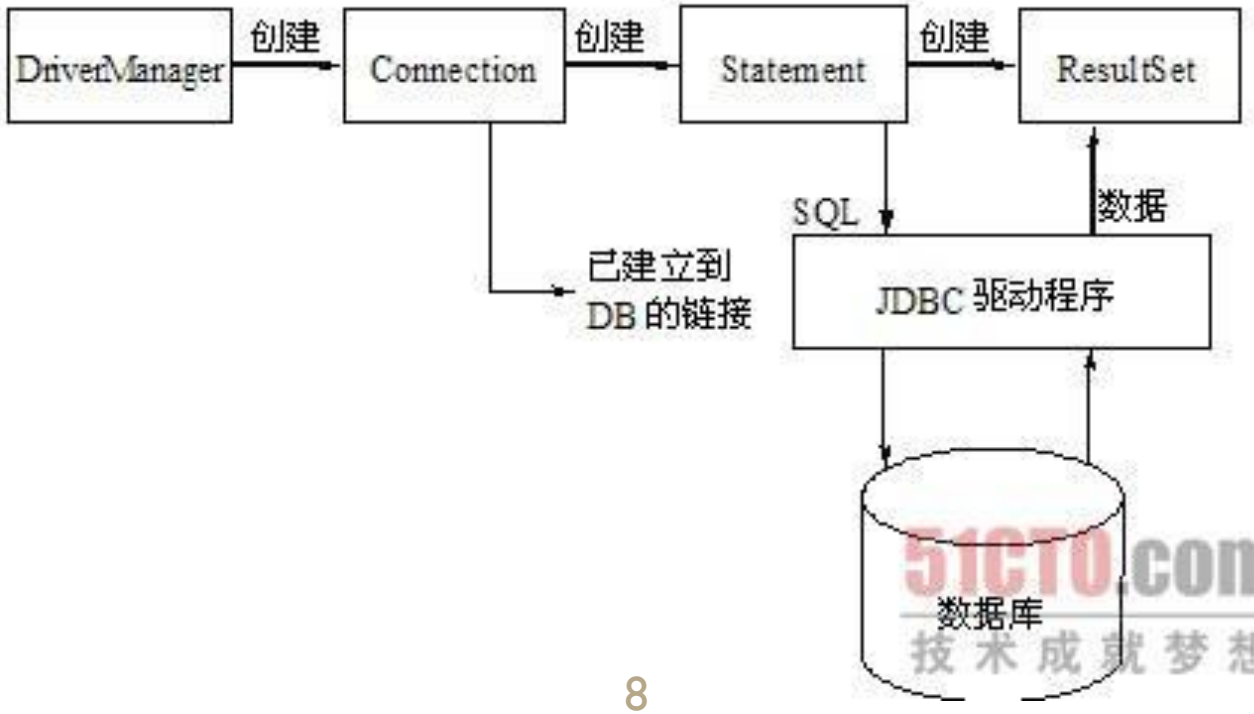
（3）处理数据库返回的结果。



JDBC基本编程

JDBC的设置步骤

1. 注册驱动程序（只做一次）
2. 建立连接(Connection对象)
3. 创建命令对象(Statement/PreparedStatement对象)并执行SQL语句
4. 处理执行结果(ResultSet对象)
5. 释放资源



实例1

- 目的：
  - 根据用户输入的客户id，到数据库中查找是否存在给定id的客户，如果存在，就查询出用户账户余额，根据用户余额确定不同的响应页面。

- MVCTest.war
  - 数据库连接
    - ✓beans.DBConnection.java
  - beans.BankBean.java
    - ✓到数据库中查找是否存在给定id的客户，如果存在，就创建一个Bean，否则返回Null

1、注册驱动

1. 放置JDBC包或配置ODBC

- 针对提供JDBC包的数据库系统，例如SQLServer2000/2005/2008,JRE6.0运行环境
  - ✓ 需要事先把SQLServer的JDBC包sqljdbc4.jar复制到JRE安装目录的\lib\ext下，或者复制到工程项目的lib目录下。
    - 其它环境变量的前提设置： 已设置好JAVA_HOME和classpath
- 针对没有提供JDBC包的数据库系统，例如Access
  - ✓ 需要事先配置ODBC数据源，使用sun提供的JDBC-ODBC桥
    - 由JRE安装目录下的\lib\rt.jar提供

2. 在程序中加载驱动器——Class.forName(“数据库驱动器名称”)

- 不同数据库，驱动程序名称不同, 例如SQLServer
  - ✓ Class.*forName*("com.microsoft.sqlserver.jdbc.SQLServerDriver");
- MySQL数据源
  - ✓ Class.forName("com.mysql.jdbc.Driver");
- 例如ODBC数据源
  - ✓ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

2、建立连接(Connection)

1. 声明Connection对象

- Connection conn=null;

2. 通过驱动程序建立连接

- 方式1：conn = DriverManager.getConnection(url, user, password);
  - ✓ url格式：
    - JDBC:子协议:子名称: //主机IP地址:端口;数据库名;属性名=属性值; …
    - 其他参数如：useUnicode=true&characterEncoding=GBK。

- 方式2： conn = DriverManager.getConnection(url);
  - 此时，user,password用“属性名=属性值”方式放置在url字符串中
  - 或者表示没有用户名和密码

2、建立连接(Connection)——连接示例

- 连接SQLServer
  - 方式1：
    - ✓ String dburl="jdbc:sqlserver://localhost:1433;databaseName=txdatabase";
    - ✓conn=DriverManager.*getConnection*(dburl, ”sa”， “123”);
  - 方式2：
    - ✓String dburl="jdbc:sqlserver://localhost:1433; databaseName=txdatabase;user=sa;password=123";
    - ✓conn=DriverManager.getConnection(dburl);

1433为SQLServer默认的 端口号

数据库的名字

- 连接MySQL方式：
  - ✓ String url="jdbc:mysql://localhost:3306/java web";
  - ✓ conn=DriverManager.*getConnection*(dburl, ”sa”, 123”);

3306为MySQL默认的端口号

数据库的名字

用户名

密码

3、创建命令对象(Statement)并执行SQL语

1. Statement对象，执行不需要参数的SQL语句

- Statement st = conn.createStatement();
- st.executeQuery(sql语句);
- 例如：
  - ✓st. executeQuery(“select * from bankaccount”);

3、创建执行SQL的语句(Statement)

2. PreparedStatement对象，执行需要传送参数的SQL语句

- String sql = “select * from table_name where col_name=? ” ;
- PreparedStatement ps = conn.prepareStatement(sql);
- ps.setString(1, “col_value”);
- ps.executeQuery();
- 例如：
  - ✓ String sql=“select * from bankaccount where bankid=? and balance>?”;
  - ✓ ps = conn.preparedStatement(sql);
  - ✓ ps.setString(1, userid);
  - ✓ ps.setInt(2, 100);
  - ✓ ps.executeQuery();

表示第*n*个参数

?代表要传入参数的位置，可以依据?的顺序依次绑定外部变量，向SQL语句中传入参数

数据库中不同的数据类型，在PreparedStatement 中均有相应的set方法设置对应的参数，可参见JDK API 帮助 文档

4、处理执行结果(ResultSet)

- 对查询的结果集进行读取

1. 获取Statement对象st执行查询的结果集
   - ResultSet rs = st.executeQuery(sql);
2. 获取PreparedStatement对象ps执行查询的结果集
   - ResultSet rs = ps.executeQuery();

- 依次处理结果集中的每条记录

```
While(rs.next()){
    //获得当前记录中指定列名的数据
    rs.getString("col_name1");
    rs.getInt("col_name2");
    //…
}
```

获得查询记录集时默认记录指针指向第一条记录之前的位置，rs.next（）函数会使记录指针下移一条记录

数据库中不同的数据类型，在ResultSet 中均有相应的get方法设置对应的列，可参见JDK API 帮助文档

5、释放资源

- 释放ResultSet, Statement,Connection.
  - 数据库连接（Connection）是非常稀有的资源，用完后必须马上释放，如果Connection不能及时正确的关闭将导致系统宕机。Connection的使用原则是尽量晚创建，尽量早的释放。
  - 释放资源的函数

```
public static void dbClose(Connection conn,
   PreparedStatement ps, ResultSet rs)    throws SQLException
{
            rs.close();
            ps.close();
            conn.close();
    }
```

插入/更新/删除

示例：JDBCTest.war
DetailedInfoBean.java
public boolean
insertBeantoDB()函数

- 功能
  - 执行INSERT语句，返回操作的记录数
  - 执行UPDATE，返回被修改的记录数
  - 执行DELETE，返回被删除的记录数

- 代码模板
  - 使用Statement对象执行不带参数的SQL语句

```
conn = getConnection();
Statement st = conn.createStatement();
String sql="delete或insert或update语句";
int i = st.executeUpdate(sql); //执行SQL语句
```

基本的CRUD(Create\Read\Upate\Delete)操作

插入/更新/删除

示例：JDBCTest.war
DetailedInfoBean.java
public boolean
insertBeantoDB()函数

- 使用PreparedStatement对象执行带参数的SQL语句

```
conn = getConnection();
String sql="delete或insert或update语句";//参数用?代替PreparedStatement ps =
conn.prepareStatement(sql);//准备语句ps.setString(1,username);//用变量username绑定
Sql语句中的第1个参数ps.setInt(2,age);//整型数据使用setInt
int i=ps.executeUpdate();//执行SQL语句
```

基本的CRUD(Create\Read\Upate\Delete)操作

查询

- 执行SELECT，返回查询结果集

示例：JDBCTest.war
DetailedInfoBean.java
Public DetailedInfoBean
getDetailedInfoBean(String un)函数

- 代码模板
  - 使用Statement对象执行不带参数的SQL语句

```
conn = getConnection();
Statement st = conn.createStatement();
String sql="select语句";
ResultSet rs= st.executeQuery(sql); //执行SQL语句
```

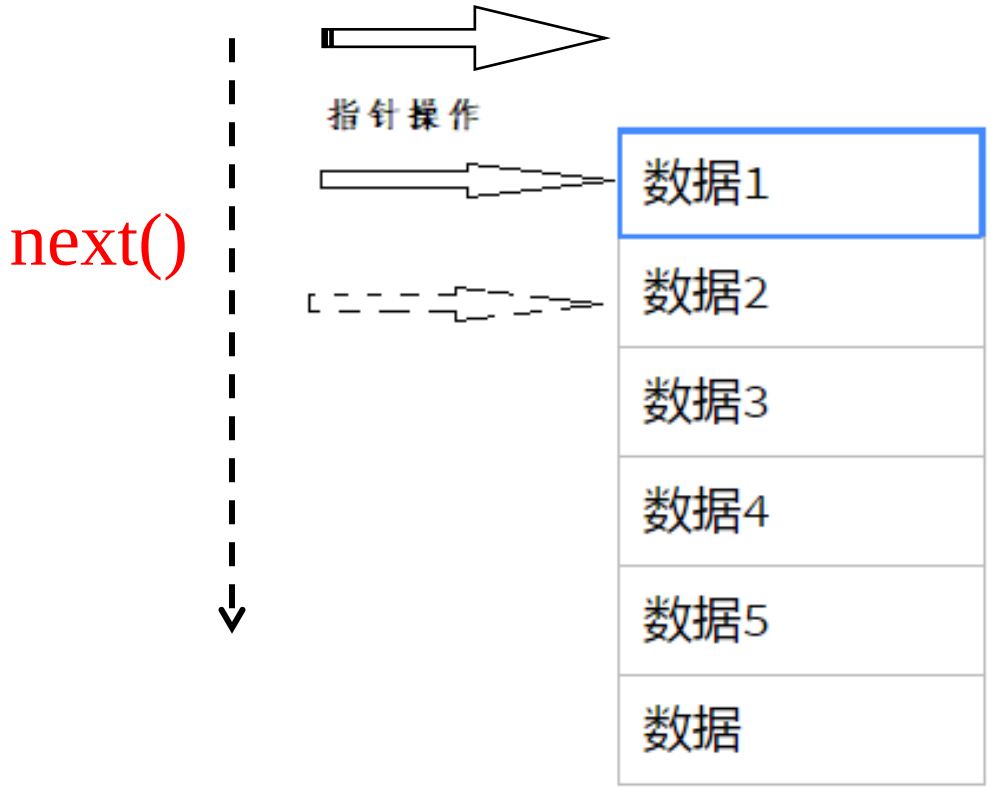
  - 使用PreparedStatement对象执行带参数的SQL语句

```
conn = getConnection();
String sql="select语句";//参数用?代替
PreparedStatement ps = conn.prepareStatement(sql);//准备语句
ps.setString(1,username);//用变量username绑定Sql语句中的第1个参数
ps.setInt(2,age);//整型数据使用setInt
ResultSet rs=ps.executeQuery();//执行SQL语句
```

基本的CRUD(Create\Read\Upate\Delete)操作

查询

- 查询结果集ResultSet：从数据库查询到的所有数据被封装到该对象中
  - ✓ 每次读取一行；
  - ✓ 调用getXXX(“column name”)将相应字段的值读取为指定类型
  - ✓ 调用next()移动到下一行，直到所有数据读取。



基本的CRUD(Create\Read\Upate\Delete)操作

查询示例

```
try {
    // 1. 加载JDBC驱动（MySQL的驱动类名为com.mysql.cj.jdbc.Driver）
    Class.forName("com.mysql.cj.jdbc.Driver");

    // 2. 建立数据库连接
    connection = DriverManager.getConnection(url, username, password);

    // 3. 创建SQL语句
    String sql = "SELECT id, name, age FROM users"; // 替换为你的SQL查询

    // 4. 创建Statement对象
    statement = connection.createStatement();

    // 5. 执行查询并获取结果集
    resultSet = statement.executeQuery(sql);

    // 6. 通过ResultSet遍历查询结果
    while (resultSet.next()) {
        // 通过列名或列索引获取数据
        int id = resultSet.getInt("id");
        String name = resultSet.getString("name");
        int age = resultSet.getInt("age");

        // 打印输出结果
        System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
    }
```

依次读取每个字段的值

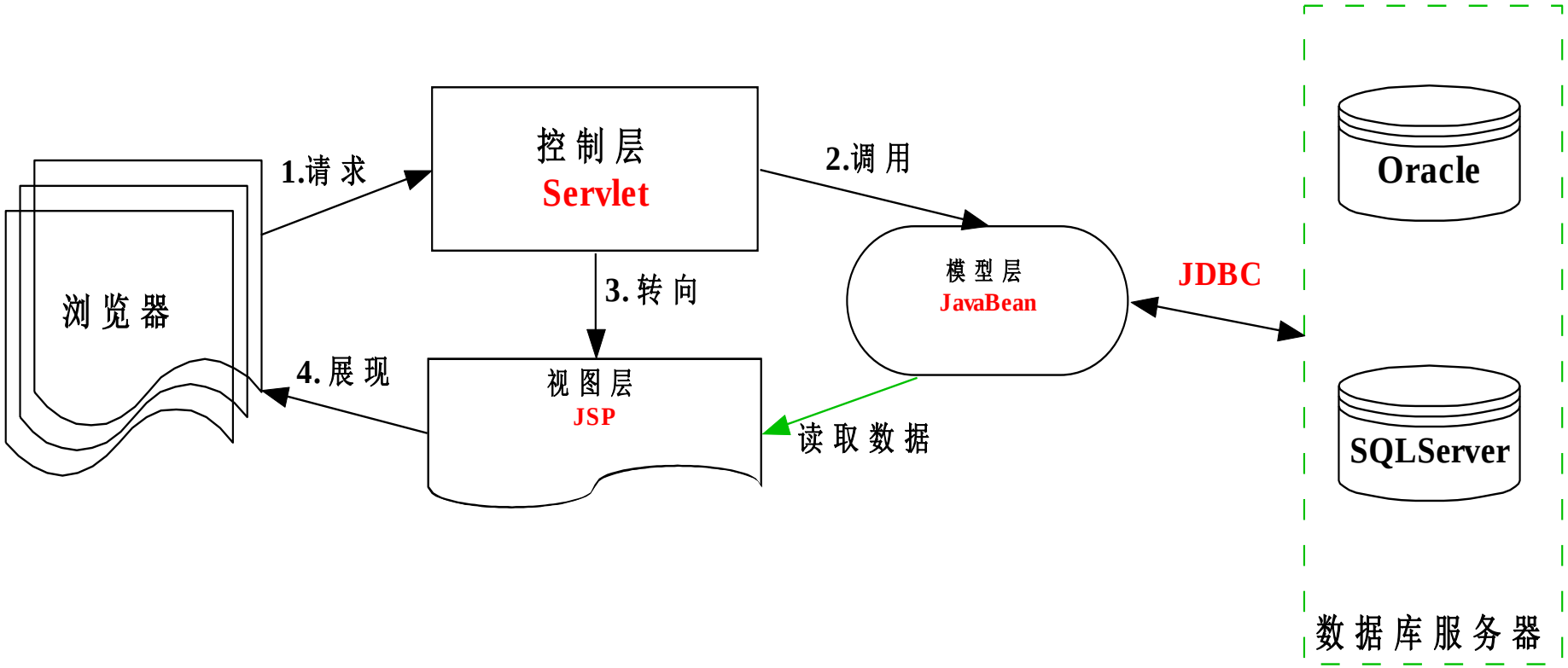
CRUD总结

- 增、删、改用Statement.executeUpdate(或PreparedStatement.executeUpdate)来完成，返回整数(匹配的记录数)，这类操作相对简单

- 查询用Statement.executeQuery(或PreparedStatement.executeQuery)来完成，返回的是ResultSet对象，ResultSet中包含了查询的结果；查询相对与增、删、改要复杂一些，因为有查询结果要处理。

基于Servlet+JSP+Bean的MVC架构

- M—模型层
- V —视图层
- C —控制层



实例2设计—JDBCTest.war

1. 数据库设计——loginuser数据库
   - user
     - (username, pwd)
   - detailedInfo
     - (username,*age,sexy,picturelocation*)

只有账户信息

username	password
www	abc
yyy	123

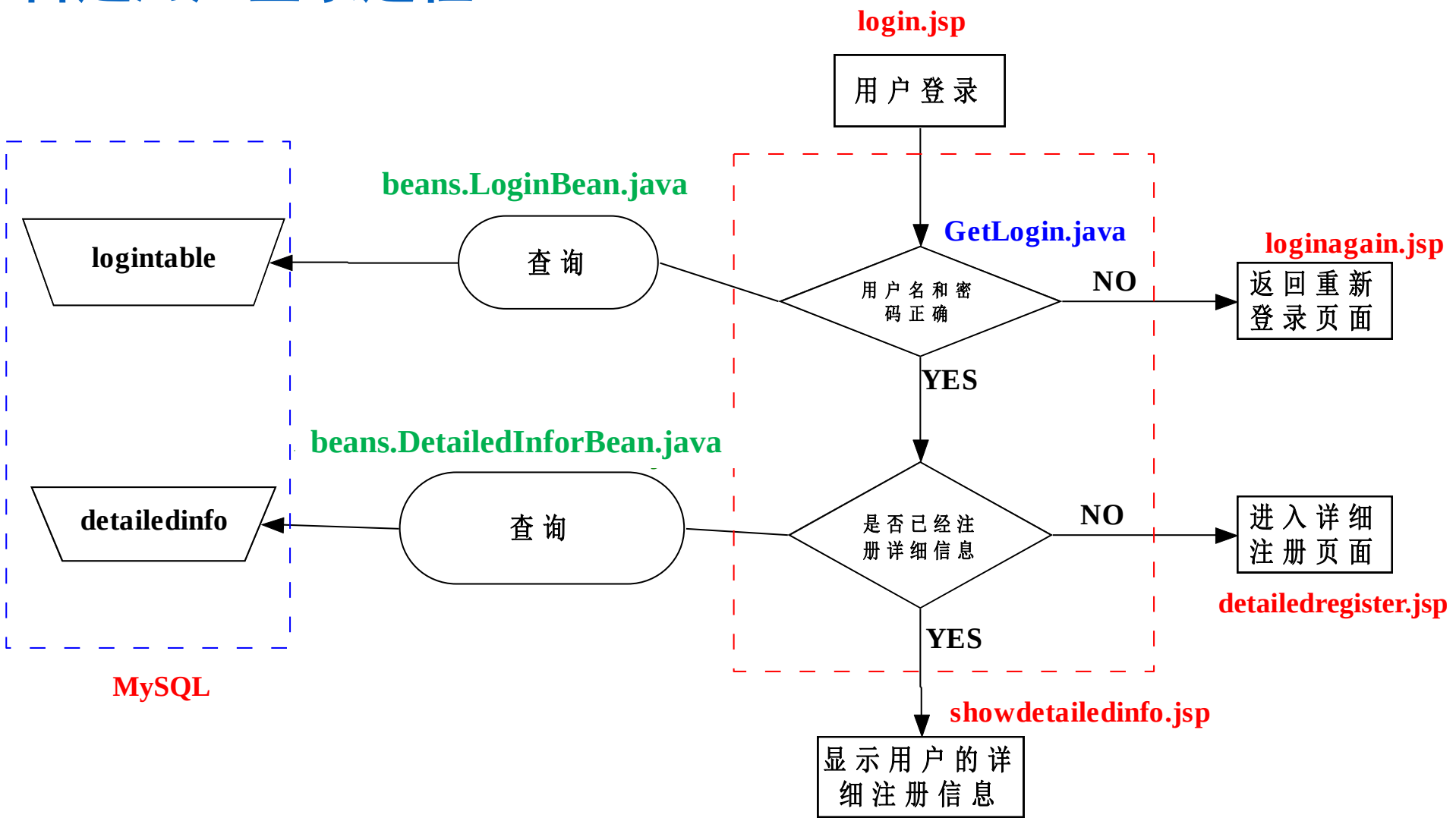
只有注册信息

username	age	sexy	picturelocation
ggg	20	m	E:\workspace\myec
www	18	f	E:\workspace

2. MVC架构
   - 模型层-----绿色文件名
   - 视图层-----红色文件名
   - 控制层-----蓝色文件名

实例2设计—JDBCTest.war

1. 普通用户登录过程



实例2设计—JDBCTest.war

思考：

1. 如何设计和实现用户修改个人信息的功能？

2. 普通用户注册过程应该如何设计和实现？

3. 如何设计和实现管理员的管理功能？

JDBC高级编程

JDBC高级编程(1)

1、PreparedStatement 概念

什么是带参数的Sql语句，在JDBC中如果Sql语句带有?（占位符）号，如：

String sql=

"select * from students where sno=?";这里并不是表示查询学号为?的学生，而是表示这里是一个参数，在执行该语句之前，必须到参数赋值。

在JDBC应用中，如果你已经是稍有水平开发者，你就应该始终以PreparedStatement代替Statement.也就是说,减少使用Statement。基于以下的原因：

JDBC高级编程(2)

(1) 代码的可读性和可维护性

虽然用PreparedStatement来代替Statement会使代码多出几行,但这样的代码无论从可读性还是可维护性上来说.都比直接用Statement的代码好多了。

```
cmd.executeUpdate("insert into tb_name (col1,col2,col2,col4) values ('"+var1+"','"+var2+"',"+var3+",'"+var4+"')");
cmd = con.prepareStatement("insert into tb_name (col1,col2,col2,col4) values (?,?,?,?)")
```

JDBC高级编程(3)

(2)　PreparedStatement提高性能

每一种数据库都会尽最大努力对预编译语句提供最大的性能优化.因为预编译语句有可能被重复调用。所以语句在被DB的编译器编译后的执行代码被缓存下来,那么下次调用时只要是相同的预编译语句就不需要编译,只要将参数直接传入编译过的语句执行代码中(相当于一个涵数)就会得到执行。

JDBC高级编程(4)

2、如何使用PreparedStatement

第1步：通过连接获得PreparedStatement对象，用带占位符(?)的构造sql语句。

```
PreparedStatement cmd = con.preparedStatement( “select * from Students where sname=?”);
```

第2步：设置输入参数值 。一定要在执行sql语句设置值。

```
cmd.setString(1, "张三");//1表示从左到右数的第一个参数，"张三"是为第一个参数设置的值，如果还有其它参数需要一个一个赋值。
```

JDBC高级编程(5)

第3步：执行sql语句

```
rs = cmd.excuteQuery();
```

Statement发送完整的Sql语句到数据库不是直接执行而是由数据库先编译，再运行。每次都需要编译。而PreparedStatement是先发送带参数的Sql语句，由数据库先编译，再发送一组组参数值。

JDBC高级编程(6)

```
public static void main(String[] args) throws Exception {
        String driver = "com.mysql.jdbc.Driver";
        Class.forName(driver);
        String url = "jdbc:mysql://127.0.0.1:3306/support";
        Connection con = DriverManager.getConnection(url, "root", "4846");
        String sql = "insert into student values(?,?,?,?)";
        PreparedStatement cmd = con.prepareStatement(sql);
        cmd.setInt(1,95007);
        cmd.setString(2,"吴兵");
        cmd.setString(3,"男");
        cmd.setString(4,"计算机科学系");
        cmd.executeUpdate();
        con.close();
}
```

如何获得元数据 MetaData

1、结果集元数据对象：ResultSetMetaData meta = rs.getMetaData();

字段个数：meta.getColomnCount();

字段名字：meta.getColumnName();

字段JDBC类型：meta.getColumnType();

字段数据库类型：meta.getColumnTypeName();

2、数据库元数据对象：DatabaseMetaData dbmd = con.getMetaData();

数据库名：dbmd.getDatabaseProductName();

数据库版本号：dbmd.getDatabaseProductVersion();

数据库驱动名：dbmd.getDriverName();

数据库驱动版本号：dbmd.getDriverVersion();

数据库Url：dbmd.getURL();

该连接的登陆名：dbmd.getUserName();

感谢聆听

Thanks For Your Listening!