Author：相忠良
Email: ugoood@163.com

起始于：May 29, 2018
最后更新日期：June 6, 2018

声明：本笔记依据传智播客方立勋老师 **Java Web** 的授课视频内容记录而成，中间加入了自己的理解。本笔记目的是强化自己学习所用。若有疏漏或不当之处，请在评论区指出。谢谢。
涉及的图片，文档写完后，一次性更新。

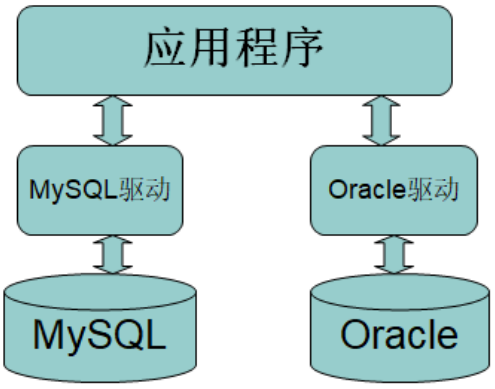
day14 JDBC

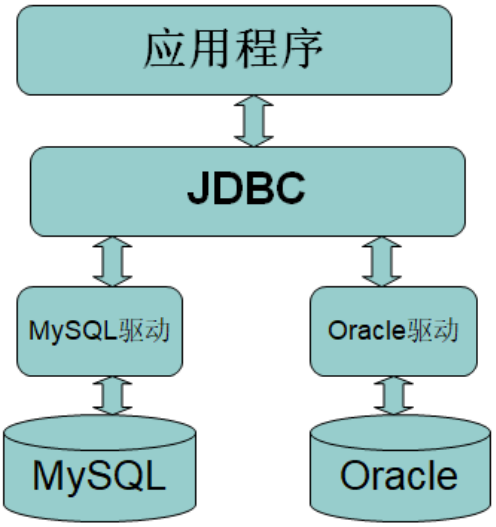
1. JDBC 入门

JDBC(Java Data Base Connectivity) 是 java 数据库连接，主要有接口组成。
JDBC的作用：可以通过java程序操作数据库。

Tip：JDBC简介

- 数据库驱动
- SUN公司为了简化、统一对数据库的操作，定义了一套Java操作数据库的规范，称之为JDBC。





组成JDBC的2个包：

- java.sql
- javax.sql
  这2个包就是JDBC，就是接口。

除导入JDBC外，还需导入相应数据库JDBC接口的实现，即数据库驱动。我们用的是`mysql-connector-java-5.0.8-bin.jar`针对mysql的数据库驱动。

做JDBC试验前的准备工作：
user.sql文件，代码如下：

```
create database day14 character set utf8 collate utf8_general_ci;

use day14;

create table users(
    id int primary key,
    name varchar(40),
    password varchar(40),
    email varchar(60),
    birthday date
);

insert into users(id,name,password,email,birthday) values(1,'zs','12345
6','zs@sina.com','1980-12-04');
insert into users(id,name,password,email,birthday) values(2,'lisi','12345
6','lisi@sina.com','1981-12-04');
insert into users(id,name,password,email,birthday) values(3,'wangwu','123
456','wangwu@sina.com','1979-12-04');
```

然后建立普通java工程day14，并建立lib文件夹，将`mysql-connector-java-5.0.8-bin.jar`复制到lib，并把该包变为奶瓶。

建立Demo1.java，操作 day14 数据库的 users 表，代码如下：

```java
package cn.wk.demo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Demo1 {
    public static void main(String[] args) throws SQLException {

      /*
      jdbc:mysql 表示协议
      //localchost:3306 本机3306端口，是mysql服务器端口
      day14 要连接的数据库
      */
        String url = "jdbc:mysql://localhost:3306/day14";
        String username = "root";
        String password = "root";

        // 1. 加载驱动
        DriverManager.registerDriver(new com.mysql.jdbc.Driver());

        // 2. 获取连接
        Connection conn = DriverManager.getConnection(url, username, pass
word);

        // 3. 获取向数据库发sql语句的statement对象
        Statement st = conn.createStatement();

        // 4. 向数据库发sql,获取数据库返回的结果集
        ResultSet rs = st.executeQuery("select * from users");

        // 5. 从结果集中获取数据
        while (rs.next()) {
            System.out.print("id = " + rs.getObject("id") + "  ");
            System.out.print("name = " + rs.getObject("name") + "  ");
            System.out.print("password = " + rs.getObject("passwor
d") + "  ");
            System.out.print("email = " + rs.getObject("email") + "  ");
            System.out.print("birthday = " + rs.getObject("birthda
y") + "  ");
            System.out.println();
        }

        // 6. 释放资源（释放链接）
        rs.close();
        st.close();
        conn.close();
    }
```

```
}
```

2. jdbc程序步骤详解1

1. 加载驱动的细节；
2. 获取连接的细节。

   如下：

Tip：程序详解—DriverManager

- Jdbc程序中的DriverManager用于加载驱动，并创建与数据库的链接，这个API的常用方法：
  - DriverManager.registerDriver(new Driver())
  - DriverManager.getConnection(url, user, password)，
- 注意：在实际开发中并不推荐采用registerDriver方法注册驱动。原因有二：

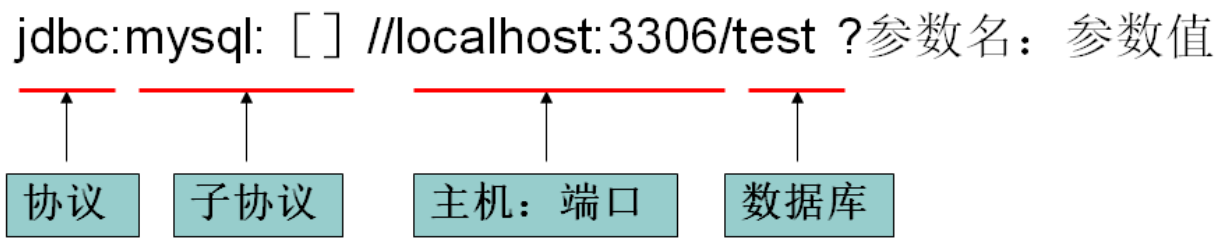
  一、查看Driver的源代码可以看到，如果采用此种方式，会导致驱动程序注册两次，也就是在内存中会有两个Driver对象。

  二、程序依赖mysql的api，脱离mysql的jar包，程序将无法编译，将来程序切换底层数据库将会非常麻烦。

  - 推荐方式：Class.forName(“com.mysql.jdbc.Driver”);
    - 采用此种方式不会导致驱动对象在内存中重复出现，并且采用此种方式，程序仅仅只需要一个字符串，不需要依赖具体的驱动，使程序的灵活性更高。
- 同样，在开发中也不建议采用具体的驱动类型指向getConnection方法返回的connection对象。

Tip：数据库URL

- URL用于标识数据库的位置，程序员通过URL地址告诉JDBC程序连接哪个数据库，URL的写法为：



- 常用数据库URL地址的写法：
  - Oracle写法：jdbc:oracle:thin:@localhost:1521:sid
  - SqlServer—jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=sid
  - MySql—jdbc:mysql://localhost:3306/sid
- Mysql的url地址的简写形式： jdbc:mysql:///sid
- 常用属性：useUnicode=true&characterEncoding=UTF-8

依据上面所说，修改了上节的代码，代码片段如下：

```
// localhost简写，?后接各种参数，无论mysql用何种字符集，均推荐字符显式设定
String url = "jdbc:mysql:///day14"
    + "?user=root&password=root"
    + "&useUnicode=true&characterEncoding=UTF-8";

Class.forName("com.mysql.jdbc.Driver"); // <---- 加载类，同时注册
Connection conn = DriverManager.getConnection(url);
```

3. jdbc程序步骤详解2

Connection对象：
Jdbc程序中的Connection，它用于代表数据库的链接，Collection是数据库编程中最重要的一个对象，客户端与数据库所有交互都是通过connection对象完成的，这个对象的常用方法：

- createStatement()：创建向数据库发送sql的statement对象。
- prepareStatement(sql) ：创建向数据库发送预编译sql的PrepareSatement对象。
- prepareCall(sql)：创建执行存储过程的callableStatement对象。
- setAutoCommit(boolean autoCommit)：设置事务是否自动提交。
- commit() ：在链接上提交事务。

- rollback() ： 在此链接上回滚事务。

Statement对象：
Statement对象用于向数据库发送SQL语句， Statement对象常用方法：

- executeQuery(String sql) ： 用于向数据发送查询语句。
- executeUpdate(String sql)：用于向数据库发送insert、update或delete语句
- execute(String sql)：用于向数据库发送任意sql语句，一般不用。
- addBatch(String sql) ： 把多条sql语句放到一个批处理中。
- executeBatch()：向数据库发送一批sql语句执行。

ResultSet对象：
ResultSet用于代表Sql语句的执行结果。
下图表达了，mysql 表中字段类型与 ResultSet 方法对照表：

Tip： *常用数据类型转换表*

SQL类型	Jdbc对应方法	返回类型
BIT(1) bit(10)	getBoolean getBytes()	Boolean byte[]
TINYINT	getByte()	Byte
SMALLINT	getShort()	Short
Int	getInt()	Int
BIGINT	getLong()	Long
CHAR, VARCHAR, LONGVARCHAR	getString()	String
Text(clob) Blob	getClob getBlob()	Clob Blob
DATE	getDate()	java.sql.Date
TIME	getTime()	java.sql.Time
TIMESTAMP	getTimestamp()	java.sql.Timestamp

最后一定要释放 conn，因为mysql的链接资源很少很宝贵，释放技巧如下：
资源释放代码放入finally里。

模板代码 如下：

```java
package cn.wk.demo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import cn.wk.domain.User;

public class Demo2 {
    public static void main(String[] args) throws SQLException,
            ClassNotFoundException {

        String url = "jdbc:mysql:///day14";
        String username = "root";
        String password = "root";

        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;

        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(url, username, password);
            st = conn.createStatement(); // throw
            rs = st.executeQuery("select * from users");
            while (rs.next()) {
                User user = new User();
                user.setId(rs.getInt("id"));
                user.setName(rs.getString("name"));
                user.setPassword(rs.getString("password"));
                user.setEmail(rs.getString("email"));
                user.setBirthday(rs.getDate("birthday"));
            }
        } finally {
            if (rs != null) {
                try {
                    rs.close(); // throw new
                } catch (Exception e) {
                    e.printStackTrace();
                }
                rs = null;
            }
            if (st != null) {
                try {
                    st.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
```

```
                st = null;
            }
            if (conn != null) {
                try {
                    conn.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

顺便建了个User类，做为javabean对象，代码如下：

```java
package cn.wk.domain;

import java.util.Date;

public class User {
    private int id;
    private String name;
    private String password;
    private String email;
    private Date birthday;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }
```

```
}
```

4. 使用 jdbc 完成 crud

案例：

1. `Demo3.java` 实现crud方法；
2. `db.properties` 资源文件，达到灵活切换数据库驱动、url、username 和 password；
3. `cn.wk.utils.JdbcUtils` 工具类，实现驱动加载、建立链接和释放链接这种公共代码。

Demo3.java，如下：

```java
package cn.wk.demo;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import org.junit.Test;

import cn.wk.domain.User;
import cn.wk.utils.JdbcUtils;

public class Demo3 {

    @Test
    public void insert() throws SQLException {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "insert into users(id,name,password,email,birthd
ay)"
                    + "values (4,'eee','123','ee@sina.com','1980-11-1
1')";
            int num = st.executeUpdate(sql);
            if (num > 0) {
                System.out.println("插入成功! ! ! ");
            }
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Test
    public void update() throws SQLException {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "update users set name = 'fff' where id = '4'";
            int num = st.executeUpdate(sql);
            if (num > 0) {
                System.out.println("更新成功! ! ! ");
            }
```

```
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Test
    public void delete() throws SQLException {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "delete from users where id='4'";
            int num = st.executeUpdate(sql);
            if (num > 0) {
                System.out.println("删除成功！！！");
            }
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Test
    public void find() throws SQLException {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "select * from users where id='2'";
            rs = st.executeQuery(sql);
            if (rs != null) {
                System.out.println("查询成功！！！");
            }
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Test
    public void getAll() throws SQLException {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "select id,name,password,email,birthday from use
```

```
rs";
            rs = st.executeQuery(sql);
            List list = new ArrayList();

            while (rs.next()) {
                User user = new User();
                user.setId(rs.getInt("id"));
                user.setName(rs.getString("name"));
                user.setPassword(rs.getString("password"));
                user.setEmail(rs.getString("email"));
                user.setBirthday(rs.getDate("birthday"));
                list.add(user);
            }

            System.out.println(list);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }
}
```

`db.properties` 资源文件：

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/day14
username=root
password=root
```

`cn.wk.utils.JdbcUtils` 工具类：

```java
package cn.wk.utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class JdbcUtils {

    private static Properties config = new Properties();

    static {
        try {
            // 读配置文件 db.properties
            config.load(JdbcUtils.class.getClassLoader().getResourceAsStr
eam(
                    "db.properties"));
            Class.forName(config.getProperty("driver"));
        } catch (Exception e) {
            throw new ExceptionInInitializerError(e); // 异常转换成错误
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(config.getProperty("url"),
                config.getProperty("username"), config.getProperty("passw
ord"));
    }

    public static void release(Connection conn, Statement st, ResultSet r
s) {
        // 模板代码
        if (rs != null) {
            try {
                rs.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            rs = null;
        }
        if (st != null) {
            try {
                st.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            st = null;
        }
```

```
        if (conn != null) {
            try {
                conn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

5. 切换到 oracle 完成 crud

更换`db.properties`文件信息为 oracle 的，几乎不用改程序，就可以用oracle做crud。
为保证程序有移植性，即想做到不依赖具体的数据库管理系统，那么，**Connection** 等对象就用 **sun** 公司的接口，而不能用具体数据库的相应对象！
如：我们导入的是`java.sql.Connection`。

6. 用 jdbc 改造用户模块

准备工作：
复制 day09_user 工程，命名为 day14_user，并点击该工程属性，选 web 选项，将 `Web Context-root` 改为`/day09_user`。

`UserDaoJdbcImpl`代码如下：

```java
package cn.wk.dao.impl;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

import cn.wk.dao.UserDao;
import cn.wk.domain.User;
import cn.wk.utils.JdbcUtils;

public class UserDaoJdbcImpl implements UserDao {

    @Override
    public void add(User user) {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement(); // conn.preparedment() 用占位符替换位置
            String sql = "insert into users(id,username,password,email,birthday,nickname)"
                    + "values('"
                    + user.getId()
                    + "','"
                    + user.getUsername()
                    + "','"
                    + user.getPassword()
                    + "','"
                    + user.getEmail()
                    + "','"
                    + user.getBirthday().toLocaleString()
                    + "','"
                    + user.getNickname() + "')";
            int num = st.executeUpdate(sql);
            if (num < 1) {
                throw new RuntimeException("注册用户失败！！！");
            }

        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Override
    public User find(String username, String password) {
```

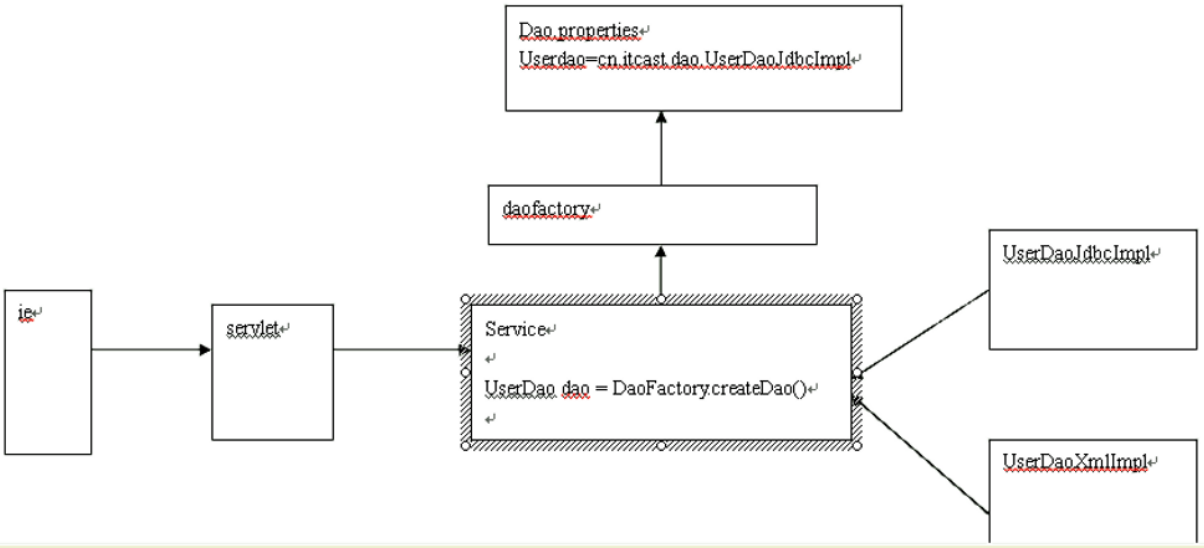
```
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "select * from users where username='" + usernam
e
                    + "' and password='" + password + "'";
            rs = st.executeQuery(sql);
            if (rs.next()) {
                User user = new User();
                user.setBirthday(rs.getDate("birthday"));
                user.setEmail(rs.getString("email"));
                user.setId(rs.getString("id"));
                user.setNickname(rs.getString("nickname"));
                user.setPassword(rs.getString("password"));
                user.setUsername(rs.getString("username"));
                return user;
            }
            return null;
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Override
    public boolean find(String username) {
        Connection conn = null;
        Statement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            st = conn.createStatement();
            String sql = "select * from users where username='" + usernam
e
                    + "'";
            rs = st.executeQuery(sql);
            if (rs.next()) {
                return true;
            }
            return false;
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
```

```
    }
}
```

其他的不写了，好累。(2018年5月30日 23:00 于 潍坊科技学院软件学院 315 办公室)

7. dao 工厂和预防 sql 注入

为使 service 层的 UserDao 与 实际的实现解耦，需专门建立 dao工厂类，由该工厂读取`dao.properties`这个配置文件，实现调用具体的 UserDao 实现，如调用`UserDaoJdbcImpl`或`UserDaoXmlImpl`等其他实现，如下图所示：



由于有可能有很多工厂，故建立`cn.wk.factory`包来存放工厂类。此时，我们在该包内建立`DaoFactory`类，如下：

```java
package cn.wk.factory;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class DaoFactory {

    private Properties daoConfig = new Properties();

    // 工厂是单例的，即所有的dao只由一个工厂来生产
    // new 对象同时，加载配置文件
    private DaoFactory() {
        InputStream in = DaoFactory.class.getClassLoader().getResourceAsS
tream(
                "dao.properties");
        try {
            daoConfig.load(in);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    private static DaoFactory instance = new DaoFactory();
    public static DaoFactory getInstance(){
        return instance;
    }

    // 将来可能产生多个 不同类型的 dao，故此处用泛型

    //                  若给   UserDao.class，本方法就产生UserDao的dao
    //                     DepartmentDao.class 同上
    // 你给我一个接口类型， 我给你返回一个接口实现
    // 为避免调用者强转，此处应该用泛型

    // 泛型的学习，重点：
    // Class<T> : 人家传进来什么类型， T就代表什么类型
    // <T> T : <T> 是类型声明， T 是返回的类型
    public <T> T createDao(Class<T> clazz){

        // 1. 得到传进来的接口名称
        // clazz.getName()  //cn.wk.dao.UserDao
        String name = clazz.getSimpleName();
        String className = daoConfig.getProperty(name);
        try {
            // 加载类，并产生实例
            T dao = (T)Class.forName(className).newInstance();
            return dao;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
```

```
    }
}
```

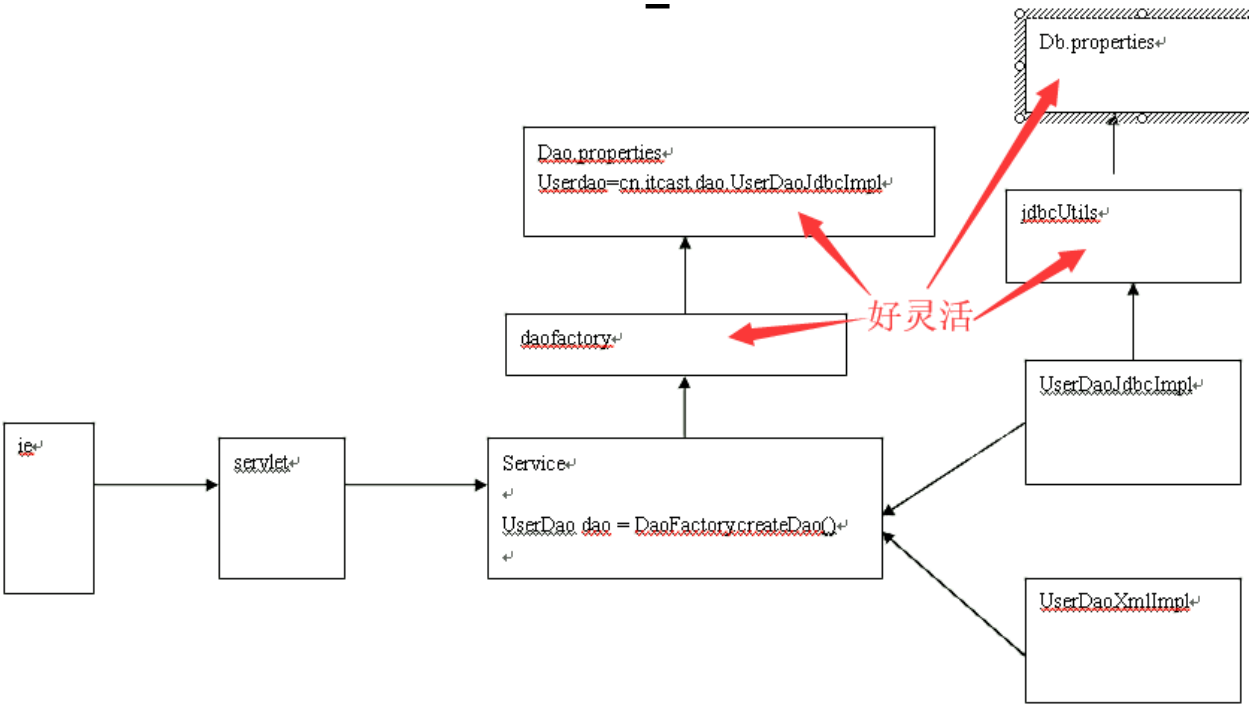
还需建`dao.properties`配置文件，如下：

```
UserDao=cn.wk.dao.impl.UserDaoJdbcImpl
```

修改`BusinessServiceImpl`如下：

```
// private UserDao dao = new UserDaoJdbcImpl(); // 可用工厂模式 或 sprin
g 解耦，以后解耦

// 利用工厂，使得业务类，无代码依赖于具体实现，已完全解耦
private UserDao dao = DaoFactory.getInstance().createDao(UserDao.class);
```

经过改造，系统结构非常好，好灵活，如下图：



关于异常：
应该每一层都自定义异常，以便抛出异常时，能快速定位是哪层出现异常。
对待异常的态度，方立勋遵循了 Spring 作者的理念，即：
你是否希望上层处理本异常，若希望，则抛出 **checked** 异常，否则抛出 **unchecked** 异常。有时，我们就想对上层造麻烦，担心上层忘记处理本层某异常，那就不用犹豫，抛 **checked** 异常即可！

7.1 preparedStatement 对象

statement 和 **preparedStatement** 的区别：

1. preparedStatement 是 statement 的崽；
2. preparedStatement 可以防止sql注入问题；
3. preparedStatement 会对sql语句进行预编译，以减轻数据库服务器的压力。

preparedStatement 使用的代码片段，一般就用 preparedStatement 而不用 statement，如下：

```
String url = "jdbc:mysql:///day14";
String username = "root";
String password = "root";

Connection conn = null;
preparedStatement st = null;
ResultSet rs = null;

Class.forName("com.mysql.jdbc.Driver");
conn = DriverManager.getConnection(url, username, password);
String sql = "select * from users where username=?"; // ?占位符
st = conn.preparedStatement(sql);
st.setString(1, username); // 填补?号

rs = st.executeQuery();
if(rs.next()){
  rs.close();
  st.close();
  conn.close();
  return true;
}
rs.close();
st.close();
conn.close();
return false;
```

8. jdbc 实现客户关系管理案例

要实现一个客户关系管理系统，客户信息 customer 表如下：

字段名	类型
id	varchar(40)
name	varchar(20)
gender	varchar(4)
birthday	date
cellphone	varchar(20)
email	varchar(40)
preference	varchar(100)
type	varchar(40)
description	varchar(255)

```
1.搭建环境
    1.1 导开发包
        mysql驱动
        beanUtils
        log4j开发
        jstl开发包

    1.2 创建组织程序的包
        cn.wk.domain
        cn.wk.dao
        cn.wk.dao.impl
        cn.wk.service
        cn.wk.service.impl
        cn.wk.web.controller
        cn.wk.web.UI
        cn.wk.utils
        cn.wk.exception
        junit.test

        WEB-INF/jsp

    1.3 为应用创建相应库和表
        create database day14_customer character set utf8 collate utf8_ge
neral_ci;
        use day14_customer;
        create table customer
        (
            id varchar(40) primary key,
            name varchar(40) not null,
            gender varchar(4) not null,
            birthday date,
            cellphone varchar(20),
            email varchar(40),
            preference varchar(255),
            type varchar(100) not null,
            description varchar(255)
        );

2.建实体

3.写dao

4.写service

5.写web
```

根据应用创建数据库及表，根据表创建 javabean，如下：

```java
package cn.wk.domain;

import java.util.Date;

public class Customer {
    private String id;
    private String name;
    private String gender;
    private Date birthday;
    private String cellphone;
    private String email;
    private String preference;
    private String type;
    private String description;

    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }
    public String getCellphone() {
        return cellphone;
    }
    public void setCellphone(String cellphone) {
        this.cellphone = cellphone;
```

```
    }
    public String getPreference() {
        return preference;
    }
    public void setPreference(String preference) {
        this.preference = preference;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```

要连接数据库，则创建连数据库的工具类，这算是一种模板代码，`cn.wk.utils.JdbcUtils`，如下：

```java
package cn.wk.utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class JdbcUtils {

    private static Properties config = new Properties();

    static {
        try {
            // 读配置文件 db.properties
            config.load(JdbcUtils.class.getClassLoader().getResourceAsStr
eam(
                    "db.properties"));
            Class.forName(config.getProperty("driver"));
        } catch (Exception e) {
            throw new ExceptionInInitializerError(e); // 异常转换成错误
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(config.getProperty("url"),
                config.getProperty("username"), config.getProperty("passw
ord"));
    }

    public static void release(Connection conn, Statement st, ResultSet r
s) {
        // 模板代码
        if (rs != null) {
            try {
                rs.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            rs = null;
        }
        if (st != null) {
            try {
                st.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            st = null;
        }
```

```
        if (conn != null) {
            try {
                conn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

连数据库时，需用到配置文件`db.properties`，它在 src 目录下，代码如下：

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/day14_customer
username=root
password=root
```

再创建操作数据库进行 crud 的实现类`cn.wk.dao.impl.CustomerDaoImpl`，如下：

```java
package cn.wk.dao.impl;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import cn.wk.dao.CustomerDao;
import cn.wk.domain.Customer;
import cn.wk.exception.DaoException;
import cn.wk.utils.JdbcUtils;

public class CustomerDaoImpl implements CustomerDao {

    @Override
    public void add(Customer c) {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "insert into customer (id,name,gender,birthday,c
ellphone,email,preference,type,description) values (?,?,?,?,?,?,?,?,?)";
            st = conn.prepareStatement(sql);
            st.setString(1, c.getId());
            st.setString(2, c.getName());
            st.setString(3, c.getGender());
            st.setDate(4, new java.sql.Date(c.getBirthday().getTime()));
            st.setString(5, c.getCellphone());
            st.setString(6, c.getEmail());
            st.setString(7, c.getPreference());
            st.setString(8, c.getType());
            st.setString(9, c.getDescription());

            st.executeUpdate();

        } catch (Exception e) {
            throw new DaoException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Override
    public void update(Customer c) {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;
```

```
        try {
            conn = JdbcUtils.getConnection();
            String sql = "update customer set name=?,gender=?,birthday=?,
cellphone=?,email=?,preference=?,type=?,description=? where id=?";
            st = conn.prepareStatement(sql);
            st.setString(1, c.getName());
            st.setString(2, c.getGender());
            st.setDate(3, new java.sql.Date(c.getBirthday().getTime()));
            st.setString(4, c.getCellphone());
            st.setString(5, c.getEmail());
            st.setString(6, c.getPreference());
            st.setString(7, c.getType());
            st.setString(8, c.getDescription());
            st.setString(9, c.getId());
            st.executeUpdate();
        } catch (Exception e) {
            throw new DaoException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Override
    public void delete(String id) {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "delete from customer where id=?";
            st = conn.prepareStatement(sql);
            st.setString(1, id);
            st.executeUpdate();
        } catch (Exception e) {
            throw new DaoException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }

    @Override
    public Customer find(String id) {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "select * from customer where id=?";
```

```
            st = conn.prepareStatement(sql);
            st.setString(1, id);
            rs = st.executeQuery();
            if (rs.next()) {
                Customer c = new Customer();
                c.setBirthday(rs.getDate("birthday"));
                c.setCellphone(rs.getString("cellphone"));
                c.setDescription(rs.getString("description"));
                c.setEmail(rs.getString("email"));
                c.setGender(rs.getString("gender"));
                c.setId(rs.getString("id"));
                c.setName(rs.getString("name"));
                c.setPreference(rs.getString("preference"));
                c.setType(rs.getString("type"));
                return c;
            }
        } catch (Exception e) {
            throw new DaoException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
        return null;
    }

    @Override
    public List<Customer> getAll() {
        Connection conn = null;
        PreparedStatement st = null;
        ResultSet rs = null;

        try {
            conn = JdbcUtils.getConnection();
            String sql = "select * from customer";
            st = conn.prepareStatement(sql);
            rs = st.executeQuery();

            List<Customer> list = new ArrayList<Customer>();
            while (rs.next()) {
                Customer c = new Customer();
                c.setBirthday(rs.getDate("birthday"));
                c.setCellphone(rs.getString("cellphone"));
                c.setDescription(rs.getString("description"));
                c.setEmail(rs.getString("email"));
                c.setGender(rs.getString("gender"));
                c.setId(rs.getString("id"));
                c.setName(rs.getString("name"));
                c.setPreference(rs.getString("preference"));
                c.setType(rs.getString("type"));
                list.add(c);
            }
            return list;
```

```
        } catch (Exception e) {
            throw new DaoException(e);
        } finally {
            JdbcUtils.release(conn, st, rs);
        }
    }
}
```

再对操作数据库的实现类`cn.wk.dao.impl.CustomerDaoImpl`抽取成接口`cn.wk.dao.CustomerDao`，如下：

```
package cn.wk.dao;
import java.util.List;
import cn.wk.domain.Customer;

public interface CustomerDao {
    public abstract void add(Customer c);
    public abstract void update(Customer c);
    public abstract void delete(String id);
    public abstract Customer find(String id);
    public abstract List<Customer> getAll();
}
```

这时，注意到每层都应建立自己的异常类，以方便将来发生异常时，方便程序员了解到底是哪层发生异常。故建立`cn.wk.exception.DaoException`，并声明该异常类为`RuntimeException`，同时继承父类的所有方法，如下：

```
package cn.wk.exception;
public class DaoException extends RuntimeException {
    public DaoException() {}
    public DaoException(String message) {super(message);}
    public DaoException(Throwable cause) {super(cause);}
    public DaoException(String message, Throwable cause) {super(messag
e, cause);}
    public DaoException(String message, Throwable cause,
            boolean enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

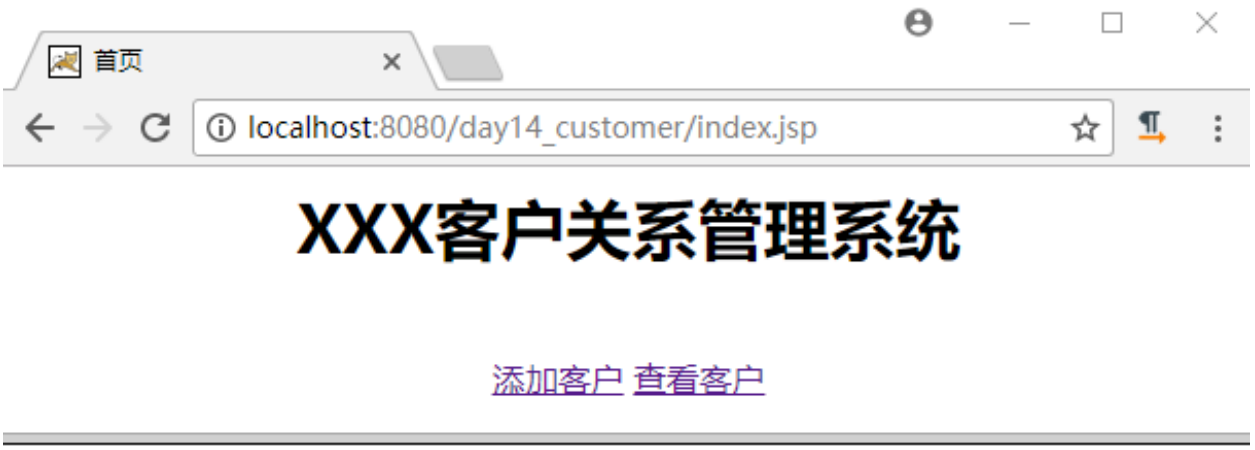
接着，想到本工程的业务仅仅是对客户信息进行 crud，业务逻辑超级简单。所以，我们可以立即建立业务层实现`cn.wk.service.impl.BusinessServiceImpl`，如下(注意：若业务逻辑很复杂，本层代码也会很复杂)：

```java
package cn.wk.service.impl;

import java.util.List;

import cn.wk.dao.CustomerDao;
import cn.wk.dao.impl.CustomerDaoImpl;
import cn.wk.domain.Customer;
import cn.wk.service.BusinessService;

// 薄薄的业务层
public class BusinessServiceImpl implements BusinessService {

    private CustomerDao dao = new CustomerDaoImpl();

    @Override
    public void addCustomer(Customer c){
        dao.add(c);
    }

    @Override
    public void updateCustomer(Customer c){
        dao.update(c);
    }

    @Override
    public void deleteCustomer(String id){
        dao.delete(id);
    }

    @Override
    public Customer findCustomer(String id){
        return dao.find(id);
    }

    @Override
    public List<Customer> getAllCustormer(){
        return dao.getAll();
    }
}
```

然后立即进行业务层实现的抽取，形成接口`cn.wk.service.BusinessService`

```java
package cn.wk.service;
import java.util.List;
import cn.wk.domain.Customer;
public interface BusinessService {
    public abstract void addCustomer(Customer c);
    public abstract void updateCustomer(Customer c);
    public abstract void deleteCustomer(String id);
    public abstract Customer findCustomer(String id);
    public abstract List<Customer> getAllCustormer();
}
```

接下来，和 web 相关，也就是和用户相关的层的代码会较复杂，要小心了。
首先考虑首页 index.jsp，本例用了分帧技术，涉及了 index.jsp 和 head.jsp，想达到如下效果：



index.jsp 如下：

```jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>首页</title>
  </head>

  <frameset rows="20%,*">
    <frame name="head" src="${pageContext.request.contextPath}/head.jsp">
    <frame name="main">
  </frameset>
</html>
```

head.jsp 如下：

```jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>head</title>
  </head>

  <body style="text-align:center;">
    <h1>XXX客户关系管理系统</h1>
    <br>

    <a href="${pageContext.request.contextPath}/servlet/AddCustomerServle
t" target="main">添加客户</a>
    <a href="${pageContext.request.contextPath}/servlet/ListCustomerServl
et" target="main">查看客户</a>
  </body>
</html>
```

注意到 head.jsp 中的2个超链接，结果显示在 main 那个帧里。
根据上述的2个超链接，分别编写那2个控制器servlet程序。
先看添加客户按钮的超链接`cn.wk.web.controller.AddCustomerServlet`，当客户点击“添加客户”按钮时，发出的是 get 请求， 由`AddCustomerServlet`的 doGet 方法处理，该方法又把请求转发给`/WEB-INF/jsp/addcustomer.jsp`去显示表单，用户填写完表单后，再
用`action="${pageContext.request.contextPath}/servlet/AddCustomerServlet" method="post"`使用 post 请求又连接
回`cn.wk.web.controller.AddCustomerServlet`这个servlet控制器，该控制器用 doPost 方法接受请求并处理(就是把客户填写的表单封装成一个 **Customer** 对象，

并转交给业务层处理，业务层将该对象转交给**dao**，由**dao**负责将对 **Customer** 对象存入数据库)，其中涉及到 message 的信息传送，以表达“添加成功或失败”信息，并回显给客户。

`cn.wk.web.controller.AddCustomerServlet` 如下：

```java
package cn.wk.web.controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import cn.wk.domain.Customer;
import cn.wk.service.BusinessService;
import cn.wk.service.impl.BusinessServiceImpl;
import cn.wk.utils.Globals;
import cn.wk.utils.WebUtils;

public class AddCustomerServlet extends HttpServlet {

    // 给用户提供一个添加界面
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        // 信息带给jsp
        req.setAttribute("genders", Globals.genders);
        req.setAttribute("preferences", Globals.preferences);
        req.setAttribute("types", Globals.types);

        // 视图
        req.getRequestDispatcher("/WEB-INF/jsp/addcustomer.jsp").forward
(req,
                resp);
    }

    // 处理用户的添加请求
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        try {
            req.setCharacterEncoding("UTF-8");
            // 表单校验

            // 把表单数据封装到 customer 对象中 , 用工具类 WebUtils
            Customer c = WebUtils.request2Bean(req, Customer.class);
            c.setId(WebUtils.generateID());

            BusinessService service = new BusinessServiceImpl();
            service.addCustomer(c);
```

```
            req.setAttribute("message", "添加成功！！");

        } catch (Exception e) {
            e.printStackTrace();
            req.setAttribute("message", "添加失败！！");
        }
        req.getRequestDispatcher("/message.jsp").forward(req, resp);
    }
}
```

控制层的`cn.wk.web.controller.AddCustomerServlet`还引入了`cn.wk.utils.Globals`这个工具类，该类目的是不想把`genders, preferences, types`写死，通过修改该类，达到直接控制那3个属性的目的，代码如下：

```
package cn.wk.utils;

public class Globals {
    public static String genders[] = { "男", "女", "人妖" };
    public static String preferences[] = { "唱歌", "跳舞", "桑拿", "打麻
将", "看凤姐",
            "夜生活", "xxx" };
    public static String types[] = { "vip客户", "重点客户", "普通客户" };
}
```

`/WEB-INF/jsp/addcustomer.jsp`如下(涉及一个显示日期的js控件`/WEB-INF/js/ShowCalendar.js`)：
该代码涉及了一个`makepre()`的 js 函数，功能是把客户的多个preference组合成一个字符串，并弄出一个 隐式输入项 送入要提交的表单中。`makepre()`函数涉及了 javascript 的 DOM 编程，挺有意思的。

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>添加用户的视图</title>
    <script type="text/javascript" src="${pageContext.request.contextPat
h }/js/ShowCalendar.js"></script>
    <script type="text/javascript">
        function makepre(){
            var pres = document.getElementsByName("pre");
            var preference = "";
            for(var i = 0; i < pres.length; i++){
                var input = pres[i];
                if(input.checked==true){
                    preference = preference + input.value + ",";
                }
            }
            // 组装字符串    跳舞,打麻将,看凤姐
            preference = preference.substr(0, preference.length - 1);

            var form = document.getElementById("form");
            var input = document.createElement("input");
            input.type = "hidden";
            input.name = "preference";
            input.value = preference;

            form.appendChild(input);
            return true;
        }
    </script>
</head>

<body style="text-align:center;">
    <br />
    <form id="form"
        action="${pageContext.request.contextPath}/servlet/AddCustomerSer
vlet"
        method="post" onsubmit="return makepre()">
        <!-- js代码，当按submit按钮时，就调用makepre()方法 -->

        <table border="1" width="30%">
            <tr>
                <td>客户姓名</td>
                <td><input type="text" name="name"></td>
            </tr>

            <tr>
                <td>性别</td>
```

```
                <td><c:forEach var="gender" items="${genders}">
                        <input type="radio" name="gender" value="${gende
r}"> ${gender}
                </c:forEach></td>
            </tr>

            <tr>
                <td>生日</td>
                <td><input type="text" name="birthday"
                    onClick="showCalendar(this.id)" id="birthday"></td>
            </tr>

            <tr>
                <td>手机</td>
                <td><input type="text" name="cellphone"></td>
            </tr>

            <tr>
                <td>邮箱</td>
                <td><input type="text" name="email"></td>
            </tr>

            <tr>
                <td>爱好</td>
                <td><c:forEach var="p" items="${preferences}">
                        <input type="checkbox" name="pre" value="${p}">
${p}
                </c:forEach></td>
            </tr>

            <tr>
                <td>客户类型</td>
                <td><c:forEach var="t" items="${types}">
                        <input type="radio" name="type" value="${t}">${t}
                </c:forEach></td>
            </tr>

            <tr>
                <td>客户备注</td>
                <td><textarea rows="5" cols="100" name="description"></te
xtarea>
                </td>
            </tr>

            <tr>
                <td><input type="reset" value="重置"></td>
                <td><input type="submit" value="添加客户"></td>
            </tr>
        </table>
    </form>
</body>
```

```
</html>
```

`/message.jsp` 如下：

我也把那个显示日期的 js 控件 `/WEB-INF/js/ShowCalendar.js` 代码贴出来：

```
// 日期选择
// By Ziyue(http://www.web-v.com/)
// 使用方法:
// <script type="text/javascript" src="${pageContext.request.contextPath }/js/ShowCalendar.js"></script>
// <input name="birthday" type="text" id="birthday" title="点击选择" onClick="showCalendar(this.id)">

var today;
document
        .writeln("<div id='Calendar' style='position:absolute; z-index:1; visibility: hidden; filter:\"progid:DXImageTransform.Microsoft.Shadow(direction=135,color=#999999,strength=3)\"'></div>");

function getDays(month, year) {
    var daysInMonth = new Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    // 下面的这段代码是判断当前是否是闰年的
    if (1 == month)
        return ((0 == year % 4) && (0 != (year % 100))) || (0 == year % 400) ? 29
                : 28;
    else
        return daysInMonth[month];
}

function getToday() {
    // 得到今天的年,月,日
    this.now = new Date();
    this.year = this.now.getFullYear();
    this.month = this.now.getMonth();
    this.day = this.now.getDate();
}

function getStringDay(str) {
    // 得到输入框的年,月,日
    var str = str.split("-");

    this.now = new Date(parseFloat(str[0]), parseFloat(str[1]) - 1,
            parseFloat(str[2]));
    this.year = this.now.getFullYear();
    this.month = this.now.getMonth();
    this.day = this.now.getDate();
}

function newCalendar() {
    var parseYear = parseInt(document.all.Year.options[document.all.Year.selectedIndex].value);
    var newCal = new Date(parseYear, document.all.Month.selectedIndex, 1);
```

```
    var day = -1;
    var startDay = newCal.getDay();
    var daily = 0;

    if ((today.year == newCal.getFullYear())
            && (today.month == newCal.getMonth()))
        day = today.day;

    var tableCal = document.all.calendar;
    var intDaysInMonth = getDays(newCal.getMonth(), newCal.getFullYear
());

    for (var intWeek = 1; intWeek < tableCal.rows.length; intWeek++)
        for (var intDay = 0; intDay < tableCal.rows[intWeek].cells.lengt
h; intDay++) {
            var cell = tableCal.rows[intWeek].cells[intDay];
            if ((intDay == startDay) && (0 == daily))
                daily = 1;

            if (day == daily) // 今天，调用今天的Class
            {
                cell.style.background = '#6699CC';
                cell.style.color = '#FFFFFF';
                // cell.style.fontWeight='bold';
            } else if (intDay == 6) // 周六
                cell.style.color = 'green';
            else if (intDay == 0) // 周日
                cell.style.color = 'red';

            if ((daily > 0) && (daily <= intDaysInMonth)) {
                cell.innerText = daily;
                daily++;
            } else
                cell.innerText = "";
        }
}

function GetDate(InputBox) {
    var sDate;
    // 这段代码处理鼠标点击的情况
    if (event.srcElement.tagName == "TD")
        if (event.srcElement.innerText != "") {
            sDate = document.all.Year.value + "-" + document.all.Month.va
lue
                    + "-" + event.srcElement.innerText;
            eval("document.all." + InputBox).value = sDate;
            HiddenCalendar();
        }
}

function HiddenCalendar() {
```

```
    // 关闭选择窗口
    document.all.Calendar.style.visibility = 'hidden';
}

function showCalendar(InputBox) {
    var months = new Array("一月", "二月", "三月", "四月", "五月", "六
月", "七月", "八月",
            "九月", "十月", "十一月", "十二月");
    var days = new Array("日", "一", "二", "三", "四", "五", "六");

    var x, y, intLoop, intWeeks, intDays;
    var DivContent;
    var year, month, day;
    var o = eval("document.all." + InputBox);
    var thisyear; // 真正的今年年份

    thisyear = new getToday();
    thisyear = thisyear.year;

    today = o.value;
    if (isDate(today))
        today = new getStringDay(today);
    else
        today = new getToday();

    // 显示的位置
    x = o.offsetLeft;
    y = o.offsetTop;
    while (o = o.offsetParent) {
        x += o.offsetLeft;
        y += o.offsetTop;
    }
    document.all.Calendar.style.left = x + 2;
    document.all.Calendar.style.top = y + 20;
    document.all.Calendar.style.visibility = "visible";

    // 下面开始输出日历表格(border-color:#9DBAF7)
    DivContent = "<table border='0' cellspacing='0' style='border:1px sol
id #0066FF; background-color:#EDF2FC'>";
    DivContent += "<tr>";
    DivContent += "<td style='border-bottom:1px solid #0066FF; background
-color:#C7D8FA'>";

    // 年
    DivContent += "<select name='Year' id='Year' onChange='newCalendar
()' style='font-family:Verdana; font-size:12px'>";
    for (intLoop = thisyear - 35; intLoop < (thisyear + 2); intLoop++)
        DivContent += "<option value= " + intLoop + " "
                + (today.year == intLoop ? "Selected" : "") + ">" + intLo
op
                + "</option>";
```

```
    DivContent += "</select>";

    // 月
    DivContent += "<select name='Month' id='Month' onChange='newCalendar
()' style='font-family:Verdana; font-size:12px'>";
    for (intLoop = 0; intLoop < months.length; intLoop++)
        DivContent += "<option value= " + (intLoop + 1) + " "
                + (today.month == intLoop ? "Selected" : "") + ">"
                + months[intLoop] + "</option>";
    DivContent += "</select>";

    DivContent += "</td>";

    DivContent += "<td style='border-bottom:1px solid #0066FF; background
-color:#C7D8FA; font-weight:bold; font-family:Wingdings 2,Wingdings,Webdi
ngs; font-size:16px; padding-top:2px; color:#4477FF; cursor:hand' align
='center' title='关闭' onClick='javascript:HiddenCalendar()'>S</td>";
    DivContent += "</tr>";

    DivContent += "<tr><td align='center' colspan='2'>";
    DivContent += "<table id='calendar' border='0' width='100%'>";

    // 星期
    DivContent += "<tr>";
    for (intLoop = 0; intLoop < days.length; intLoop++)
        DivContent += "<td align='center' style='font-size:12px'>"
                + days[intLoop] + "</td>";
    DivContent += "</tr>";

    // 天
    for (intWeeks = 0; intWeeks < 6; intWeeks++) {
        DivContent += "<tr>";
        for (intDays = 0; intDays < days.length; intDays++)
            DivContent += "<td onClick='GetDate(\""
                    + InputBox
                    + "\")' style='cursor:hand; border-right:1px solid #B
BBBBB; border-bottom:1px solid #BBBBBB; color:#215DC6; font-family:Verdan
a; font-size:12px' align='center'></td>";
        DivContent += "</tr>";
    }
    DivContent += "</table></td></tr></table>";

    document.all.Calendar.innerHTML = DivContent;
    newCalendar();
}

function isDate(dateStr) {
    var datePat = /^(\d{4})(\-)(\d{1,2})(\-)(\d{1,2})$/;
    var matchArray = dateStr.match(datePat);
    if (matchArray == null)
        return false;
```

```
    var month = matchArray[3];
    var day = matchArray[5];
    var year = matchArray[1];
    if (month < 1 || month > 12)
        return false;
    if (day < 1 || day > 31)
        return false;
    if ((month == 4 || month == 6 || month == 9 || month == 11) && day =
= 31)
        return false;
    if (month == 2) {
        var isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 =
= 0));
        if (day > 29 || (day == 29 && !isleap))
            return false;
    }
    return true;
}
```

把 request 中表单信息封装成javabean需建立一个`cn.wk.utils.WebUtils`工具类。因需把request中的一个 String 类型的参数`birthday=1999-01-01`转换成`java.utils.Date`对象，需注册一个转换器，这玩意对我来说很陌生，代码如下：

```java
package cn.wk.utils;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.UUID;

import javax.servlet.http.HttpServletRequest;

import org.apache.commons.beanutils.BeanUtils;
import org.apache.commons.beanutils.ConvertUtils;
import org.apache.commons.beanutils.Converter;

public class WebUtils {
    public static <T> T request2Bean(HttpServletRequest request,
            Class<T> beanClass) {
        try {
            T bean = beanClass.newInstance();

            // 得到 request 里所有数据
            Map map = request.getParameterMap();

            // 填充
            // map{name=aa,password=bb,birthday=1999-01-01} 填充到bean
            // bean{name=aa,password=bb,birthday=Date}

            ConvertUtils.register(new Converter() {

                @Override
                // 字符串 转成 java.util.Date 返回
                public Object convert(Class type, Object value) {
                    if (value == null) {
                        return null;
                    }

                    String str = (String) value;
                    if (str.trim().equals("")) {
                        return null;
                    }

                    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-d
d");
                    try {
                        return df.parse(str);
                    } catch (ParseException e) {
                        throw new RuntimeException(e);
                    }
                }
```

```
            }, Date.class);

            BeanUtils.populate(bean, map);
            return bean;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    public static String generateID(){
        return UUID.randomUUID().toString();
    }
}
```

head.jsp 中的另一个按钮“查看客户”，跳转到一个控制层的servlet上，`cn.wk.web.controller.ListCustomerServlet`代码如下：

```
package cn.wk.web.controller;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import cn.wk.service.BusinessService;
import cn.wk.service.impl.BusinessServiceImpl;

// 得到所有客户显示
public class ListCustomerServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        try {
            BusinessService service = new BusinessServiceImpl();
            List list = service.getAllCustormer();
            req.setAttribute("list", list);
            req.getRequestDispatcher("/WEB-INF/jsp/listcustomer.jsp").for
ward(
                    req, resp);
        } catch (Exception e) {
            e.printStackTrace();
            req.setAttribute("message", "查看客户失败！！");
            req.getRequestDispatcher("/message.jsp").forward(req, resp);
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

通过上面的servlet，除了发生异常时发送“查看客户失败！！”的信息到 `/message.jsp`，无异常时将request转发给 `/WEB-INF/jsp/listcustomer.jsp`，其代码如下：

```jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>列出所有客户</title>
</head>

<body style="text-align: center;">
    <table frame="border" width="85%">
        <tr>
            <td>客户姓名</td>
            <td>性别</td>
            <td>生日</td>
            <td>手机</td>
            <td>邮箱</td>
            <td>爱好</td>
            <td>类型</td>
            <td>备注</td>
            <td>操作</td>
        </tr>

        <c:forEach var="c" items="#{requestScope.list}">
            <tr>
                <td>${c.name }</td>
                <td>${c.gender }</td>
                <td>${c.birthday }</td>
                <td>${c.cellphone }</td>
                <td>${c.email }</td>
                <td>${c.preference }</td>
                <td>${c.type }</td>
                <td>${c.description }</td>
                <td>
                    <a href="#">修改</a>
                    <a href="#">删除</a>
                </td>
            </tr>
        </c:forEach>
    </table>
</body>
</html>
```

上述 jsp 页面仍会显示在 index.jsp 中`name="main"`的分帧里，原因是在 head.jsp 中有这样的定义：

```
<a href="${pageContext.request.contextPath}/servlet/ListCustomerServle
t" target="main">查看客户</a>
```

即用户点击`查看客户`按钮，跳转到控制层的`ListCustomerServlet`这个servlet上，然后request又由该servlet转发到`/WEB-INF/jsp/listcustomer.jsp`这个数据展示页面上，而这个展示页面仍受 head.jsp 中的`target="main"`的控制，最终拼接在 index.jsp 页面指定的位置上。
真够复杂的了。