基于Java Servlet 构建的在线音乐服务器

核心功能

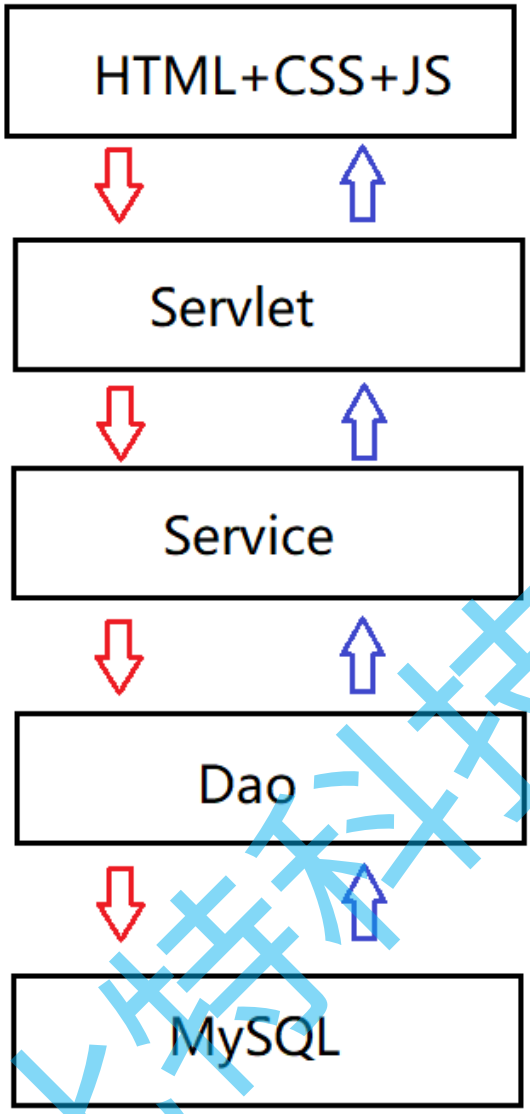
1. 登录、注册
2. 上传音乐
3. 删除某一个音乐信息
4. 删除选中的音乐信息
5. 查询音乐(包含查找指定/模糊匹配的音乐)
6. 添加音乐到“喜欢列表”。
7. 查询喜欢的音乐(包含查找指定/模糊匹配的音乐)

重要知识点

1. 简单的Web服务器设计能力
2. Java 操作 MySQL 数据库
3. 数据库设计
4. json 的使用
5. 强化 HTTP 协议的理解
6. Servlet 的使用
7. Java集合的使用
8. 前端知识的简单使用如：HTML+CSS+JS

整体架构

项目整体基于HTTP协议，前端使用HTML+CSS+JS构建页面整体布局，后端采用分层结构，分为Servlet层，Service层，Dao层的设计，以达到在设计上的高内聚低耦合。



数据库设计

music

id	title	singer	time	url	userid
1	南方姑娘	赵雷	2020-06-29	music/十九岁	1
2	后来	刘若英		music/...	2
3	红玫瑰	陈奕迅			1
4	优点	李荣浩			1
5	匆匆那年	王菲			2

一个用户可以喜欢多个音乐
一个音乐可以被多个用户喜欢

多对多的关系
设计中间表
lovemusic

id	user_id	music_id
1	1	3
2	1	4
3	2	4

user表

id	username	password	age	gender	email
1	gaobo	123	18	男	.....qq.com
2	bit	bit	12	男	....@bit.com

```sql
-- 数据库
drop database if exists `musicserver`;
create database if not exists `musicserver` character set utf8;


-- 使用数据库
use `musicserver`;


DROP TABLE IF EXISTS `music`;
CREATE TABLE `music` (
  `id` int PRIMARY KEY  AUTO_INCREMENT,
  `title` varchar(50) NOT NULL,
  `singer` varchar(30) NOT NULL,
  `time` varchar(13) NOT NULL,
  `url` varchar(100) NOT NULL,
  `userid` int(11) NOT NULL
);


DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `id` INT PRIMARY KEY AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `password` varchar(32) NOT NULL,
  `age` INT NOT NULL,
  `gender` varchar(2) NOT NULL,
  `email` varchar(50) NOT NULL
);


DROP TABLE IF EXISTS `lovemusic`;
CREATE TABLE `lovemusic` (
  `id` int PRIMARY KEY  AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `music_id` int(11) NOT NULL
);

INSERT INTO user(username,password,age,gender,email)
VALUES("bit","123","10","男","1262913815@qq.com");
```


用户+音乐模块设计

创建entity包

1. 创建User类。

```java
package entity;

public class User {
    private int id;
    private String username;
    private String password;
    private String gender;
    private int age;
    private String email;
}
```

2. 创建Music类

```java
package entity;

public class Music {
    private int id;
    private String title;
    private String singer;
    private Date time;
    private String url;
    private int userid;
}
```


服务器 API 设计

1 关于 Json

Json 是一种常见是数据格式组织方式. 源于 JavaScript, 是一种键值对风格的数据格式. 在Java中 我们可以采用Jackson库中的ObjectMapper类来完成 Json 的解析和构造。

以下是Maven中的依赖：如何去Maven中查找对应依赖：

```xml
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.5</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.5</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.9.5</version>
</dependency>
```

代码示例：

提供一个实体类Person

```
public  class Person {
    private int id;
    private String name;
    private String password;

    public Person() {
        super();
    }

    public Person(int id, String name, String password) {
        this.id = id;
        this.name = name;
        this.password = password;
    }

    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;
    }
}
```


进行json转换：

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
```

```
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-20
 * Time: 15:18
 */
public class Main {
    public static void main(String[] args) throws JsonProcessingException {
        ObjectMapper objectMapper = new ObjectMapper();
        Person person = new Person(1, "tom", "123");
        String jsonString = objectMapper.writeValueAsString(person);
        System.out.println("JsonString: " + jsonString);
    }
}
```

输出结果为：

```
JsonString: {"id":1,"name":"tom","password":"123"}
```

2 登录

```
请求：
POST  /loginServlet
data：{username,password}

响应：
{msg: true}
```

3 上传音乐

```
请求1：上传音乐到服务器目录
POST  /upload

请求2：将音乐信息同步插入到数据库当中
POST  /uploadsucess
```

4 删除某一个音乐信息

```
请求：
POST  /deleteServlet
data：{"id": id}

响应：
{msg: true}
```

5 删除选中的音乐信息

```
请求:
POST  /deleteSelMusicServlet
 data:{"id":id}//id为数组

响应:
{msg: true}
```

6 查询音乐(包含查找指定/模糊匹配的音乐)

```
请求:
POST      /findLoveMusic
data:{musicName:musicName}
```

7 添加音乐到“喜欢列表”

```
请求:
POST      /loveMusicServlet
data: {"id": obj}

响应:
{msg: true}
```

8 查询喜欢的音乐(包含查找指定/模糊匹配的音乐)

```
请求:
POST      /findLoveMusic
data:{musicName:musicName}
```

9 移除喜欢的某个音乐

```
请求:
POST      /removeLoveServlet
data: {"id": obj}
```


创建一个 JavaWeb 项目

参考《手把手教你创建一个WEB项目》

封装数据库操作

1 创建一个util包

创建JDBCUtils类。

```
package util;

import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;

import javax.sql.DataSource;
import java.sql.*;


public class DBUtils {
    private static String url = "jdbc:mysql://127.0.0.1:3306/terrymusic?useSSL=false";
    private static String password = "111111";
    private static String username = "root";

    private static volatile DataSource DATASOURCE;

    private static DataSource getDataSource(){
        // 双重校验锁
        if(DATASOURCE == null){
            synchronized (DBUtils.class){
                if(DATASOURCE == null){
                    DATASOURCE = new MysqlDataSource();
                    ((MysqlDataSource) DATASOURCE).setUrl(url);
                    ((MysqlDataSource) DATASOURCE).setUser(username);
                    ((MysqlDataSource) DATASOURCE).setPassword(password);
                }
            }
        }
        return DATASOURCE;
    }

    public static Connection getConn(){
        try {
            //从池子里获取连接
            Connection connection = getDataSource().getConnection();
            return connection;
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException("获取数据库连接失败");
        }
    }

    public static void getClose(Connection connection, PreparedStatement statement, ResultSet
resultSet) {
        if(resultSet!=null) {
            try {
                resultSet.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(statement!=null) {
            try {
                statement.close();
```

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

}
```

2 创建dao包和UserDao类

```
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import entity.User;
import util.DBUtils;

public class UserDao {

    /**
     *
     * 依据用户名查询，如果找不到，返回null，
     * 否则返回一个User对象（包含了用户的所有信息）
     *
     */
    public User login(String username) {

    }

    /**
     * 注册
     */
    public void insertUser(User user) {

    }
}
```

3 实现UserDao.login

```
public User login(User loginUser) {

    User user = null;
```

```
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            conn = DBUtils.getConn();
            ps =  conn.prepareStatement("select*from user where  username=? and password=?");
            ps.setString(1, loginUser.getUsername());
            ps.setString(2, loginUser.getPassword());
            rs = ps.executeQuery();
            while(rs.next()) {
                user = new User();
                user.setId(rs.getInt("id"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("password"));
                user.setAge(rs.getInt("age"));
                user.setGender(rs.getString("gender"));
                user.setEmail(rs.getString("email"));
            }
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }finally {
            DBUtils.getClose(conn, ps, rs);
        }
        return user;
}
```

4 实现UserDao.register

```
public void register(User user) {
    Connection conn = null;
    PreparedStatement ps = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("insert into user values(null,?,?,?,?,?)");
        ps.setString(1, user.getUsername());
        ps.setString(2, user.getPassword());
        ps.setString(3, user.getGender());
        ps.setInt(4, user.getAge());
        ps.setString(5, user.getEmail());
        ps.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }finally {
        DBUtils.getClose(conn, ps, null);
    }
}
```

5 创建MusicDao类

```java
package dao;

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

import entity.Music;
import util.DBUtils;

public class MusicDao {
    /**
     * 查询全部歌单
     */
    public List<Music> findMusic(){

    }

    /**
     * 根据id查找音乐
     * @param id
     * @return
     */
    public  Music findMusicById(int id){

    }

    /**
     * 根据关键字查询歌单
     */
    public List<Music> ifMusic(String str){

    }

    /**
     * 上传音乐
     * @param title
     * @param singer
     * @param time
     * @param url
     * @param userid
     * @return
     */
    public int insert(String title, String singer, String time, String url,
                      int userid) {

    }

    /**
     * 删除歌曲:
     */
```

```
public  int deleteMusicById(int id) {

}

 /**
 *删除歌曲，需要先看该歌曲之前是否被添加到了，喜欢的音乐列表当中
 */
public boolean findLoveMusicOnDel(int id) {

}
/**
 * 当删除服务器上的音乐时，同时在我喜欢的列表的数据库中进行删除。
 * @param musicId
 * @return
 */
public int removeLoveMusicOnDelete(int musicId) {

}



/**
 * 添加音乐到“喜欢”列表中
 * 用户-》音乐
 * 多对多
 * 需要中间表
 */
public boolean insertLoveMusic(int userId,int musicId) {

}

/**
 * 添加喜欢的音乐的时候，需要先判断该音乐是否存在，也就是说，该用户是否之前添加过这个音乐为喜欢。
 * @param musicID
 * @return
 */
public boolean findMusicByMusicId(int user_id,int musicID) {

}

/**
 * @param userId 用户id
 * @param musicId 歌曲id
 * @return 返回受影响的行数
 * 移除当前用户喜欢的这首音乐，因为同一首音乐可能多个用户喜欢，所以需要传入当前用户的id
 */
public int removeLoveMusic(int userId,int musicId) {

}



 /**
```

```
     * 查询该用户喜欢的全部歌单,只查询一张lovemusic是做不到的。需要联表查询。
     * @param user_id
     * @return
     */
    public List<Music> findLoveMusic(int user_id){

    }
    /**
     * 根据关键字查询该用户喜欢的某个音乐
     * @param str
     * @return
     */
    public List<Music> ifMusicLove(String str,int user_id){

    }
}
```

6 实现MusicDao.findMusic

```
/**
 * 查询全部歌单
 */
public List<Music> findMusic(){
    List<Music> musics = new ArrayList<>();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("select*from music");
        rs = ps.executeQuery();
        while(rs.next()) {
            Music music = new Music();
            music.setId(rs.getInt("id"));
            music.setTitle(rs.getString("title"));
            music.setSinger(rs.getString("singer"));
            music.setTime(rs.getDate("time"));
            music.setUrl(rs.getString("url"));
            music.setUserid(rs.getInt("userid"));
            musics.add(music);
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }finally {
        DBUtils.getClose(conn, ps, rs);
    }

    return musics;
}
```

7 实现MusicDao.findMusicById

```
/**
* 根据id查找音乐
 * @param id
* @return
*/
public  Music findMusicById(int id){
    Music music = null;
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("select * from music where id=?");
        ps.setInt(1,id);
        rs = ps.executeQuery();
        if(rs.next()) {
            music = new Music();
            music.setId(rs.getInt("id"));
            music.setTitle(rs.getString("title"));
            music.setSinger(rs.getString("singer"));
            music.setTime(rs.getDate("time"));
            music.setUrl(rs.getString("url"));
            music.setUserid(rs.getInt("userid"));
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }finally {
        DBUtils.getClose(conn, ps, rs);
    }
    return music;
}
```


8 实现MusicDao.ifMusic

```
/**
* 根据关键字查询歌单
*/
public List<Music> ifMusic(String str){
    List<Music> musics = new ArrayList<>();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("select*from music where title like '%"+str+"%'");
        rs = ps.executeQuery();

        while(rs.next()) {
```

```
            Music music = new Music();
            music.setId(rs.getInt("id"));
            music.setTitle(rs.getString("title"));
            music.setSinger(rs.getString("singer"));
            music.setTime(rs.getDate("time"));
            music.setUrl(rs.getString("url"));
            music.setUserid(rs.getInt("userid"));
            musics.add(music);
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    } finally {
        DBUtils.getClose(conn, ps, rs);
    }
    return musics;
}
```

9 实现MusicDao.Insert

```
/**
 * 上传音乐
*/
public int Insert(String title, String singer, String time, String url,
                  int userid) {
    Connection conn = DBUtils.getConn();
    PreparedStatement pst=null;
    int number = 0;
    try {
        pst=conn.prepareStatement("insert into music(title,singer,time,url,userid)
values(?,?,?,?,?)");
        pst.setString(1,title);
        pst.setString(2,singer);
        pst.setString(3,time);
        pst.setString(4,url);
        pst.setInt(5,userid);
        number=pst.executeUpdate();
        return number;
    } catch (SQLException e) {
        e.printStackTrace();
    }finally
    {
        DBUtils.getClose(conn, pst, null);
    }
    return 0;
}
```

10 实现MusicDao.deleteMusicById

```
/**
* 删除歌曲:
```

```
 */
public  int deleteMusicById(int id) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        String sql = "delete from music where id=?";
        connection = DBUtils.getConn();
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1,id);
        int ret = preparedStatement.executeUpdate();
        if(ret == 1) {
            //同时删除中间表中的数据
            //1、看中间表是否有数据，如果有删除
            if(findLoveMusicOnDel(id)) {
                int ret2 = removeLoveMusicOnDelete(id);
                if(ret2 == 1){
                    return 1;
                }
            } else {
                //如果没有找到，说明这首歌，没有被添加到喜欢的列表
                return 1;
            }
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        DBUtils.getClose(connection,preparedStatement,null);
    }
    return 0;
}

/**
* 看中间表是否有该id的音乐数据
*/
public boolean findLoveMusicOnDel(int id) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    try {
        String sql = "select * from lovemusic where music_id=?";
        connection = DBUtils.getConn();
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1,id);
        resultSet = preparedStatement.executeQuery();
        if(resultSet.next()) {
            return true;
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        DBUtils.getClose(connection,preparedStatement,null);

    }
```

```
        return false;
    }

     /**
     * 当删除服务器上的音乐时，同时在我喜欢的列表的数据库中进行删除。
     * @param musicId
     * @return
     */
    public int removeLoveMusicOnDelete(int musicId) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        try {
            String sql = "delete from lovemusic where music_id=?";
            connection = DBUtils.getConn();
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1,musicId);
            int ret = preparedStatement.executeUpdate();
            if(ret == 1) {
                return ret;
            }
        }catch (SQLException e) {
            e.printStackTrace();
        }finally {
            DBUtils.getClose(connection,preparedStatement,null);
        }
        return 0;
    }
```

11 实现MusicDao.insertLoveMusic

```
    /**
     * 添加音乐到“喜欢”列表中
     * 用户-》音乐
     * 多对多
     * 需要中间表
     */
    public boolean insertLoveMusic(int userId,int musicId) {
        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
            String sql = "insert into lovemusic(user_id, music_id) VALUES (?,?)";
            connection = DBUtils.getConn();
            preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1,userId);
            preparedStatement.setInt(2,musicId);
            int ret = preparedStatement.executeUpdate();

            if (ret == 1) {
```

```
            return true;
        }
        return false;
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        DBUtils.getClose(connection,preparedStatement,null);
    }
    return false;
}
```

12 实现MusicDao.removeLoveMusic

```
/**
     * @param userId 用户id
     * @param musicId 歌曲id
     * @return 返回受影响的行数
     * 移除当前用户喜欢的这首音乐，因为同一首音乐可能多个用户喜欢
     */
public int removeLoveMusic(int userId,int musicId) {
    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        String sql = "delete from lovemusic where user_id=? and music_id=?";
        connection = DBUtils.getConn();
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setInt(1,userId);
        preparedStatement.setInt(2,musicId);
        int ret = preparedStatement.executeUpdate();
        if(ret == 1) {
            return ret;
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        DBUtils.getClose(connection,preparedStatement,null);
    }
    return 0;
}
```

13 实现MusicDao.findMusicByMusicId

```
/**
* 添加喜欢的音乐的时候，需要先判断该音乐是否存在
* @param musicID
* @return
*/

public boolean findMusicByMusicId(int user_id,int musicID) {
```

```
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("select * from lovemusic where music_id=? and user_id=?");
        ps.setInt(1,musicID);
        ps.setInt(2,user_id);
        rs = ps.executeQuery();
        if(rs.next()) {
            return true;
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }finally {
        DBUtils.getClose(conn, ps, rs);
    }
    return false;
}
```

14 实现MusicDao.findLoveMusic

```
 /**
 * 查询用户喜欢的全部歌单
 * @param user_id
 * @return
 */
public List<Music> findLoveMusic(int user_id){
    List<Music> musics = new ArrayList<>();
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    try {
        conn = DBUtils.getConn();
        ps = conn.prepareStatement("select m.id as music_id,title,singer,time,url,userid from
lovemusic lm,music m where lm.music_id=m.id and user_id=?");
        ps.setInt(1,user_id);
        rs = ps.executeQuery();
        while(rs.next()) {
            Music music = new Music();
            music.setId(rs.getInt("music_id"));
            music.setTitle(rs.getString("title"));
            music.setSinger(rs.getString("singer"));
            music.setTime(rs.getDate("time"));
            music.setUrl(rs.getString("url"));
            music.setUserid(rs.getInt("userid"));
            musics.add(music);
        }
    } catch (Exception e) {

        e.printStackTrace();
```

```
            throw new RuntimeException(e);
        }finally {
            DBUtils.getClose(conn, ps, rs);
        }

        return musics;
    }
```

15 实现MusicDao.ifMusicLove

```
    /**
    * 根据关键字查询喜欢的歌单
    * @param str
    * @return
    */
    public List<Music> ifMusicLove(String str,int user_id){
        List<Music> musics = new ArrayList<>();
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            conn = DBUtils.getConn();
            //ps = conn.prepareStatement("select*from music where title like '%"+str+"%'");
            ps = conn.prepareStatement("select m.id as music_id,title,singer,time,url,userid from
    lovemusic lm,music m where lm.music_id=m.id and user_id=? and  title like '%"+str+"%'");
            ps.setInt(1,user_id);
            rs = ps.executeQuery();
            while(rs.next()) {
                Music music = new Music();
                music.setId(rs.getInt("music_id"));
                music.setTitle(rs.getString("title"));
                music.setSinger(rs.getString("singer"));
                music.setTime(rs.getDate("time"));
                music.setUrl(rs.getString("url"));
                music.setUserid(rs.getInt("userid"));
                musics.add(music);
            }
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        } finally {
            DBUtils.getClose(conn, ps, rs);
        }
        return musics;
    }
```

Service层设计实现（拓展后续可自行实现）

```
    /**
```

```
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-06-26
 * Time: 23:17
 */
public class UserService {
    //登录方法
    public User login(User loginUser) {
        UserDao userDao = new UserDao();
        User user = userDao.login(loginUser);
        //System.out.println("UserService "+ user);
        return user;
    }

}
```

Servlet实现与实现

首先在项目根目录下创建一个 servlet 包。包装实现如下servlet类。

1 LoginServlet实现

```
@WebServlet("/loginServlet")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        resp.setContentType("application/json;charset=utf-8");
        String username = req.getParameter("username");
        String password = req.getParameter("password");

        System.out.println("username: "+username);
        System.out.println("password: "+password);

        UserDao dao = new UserDao();

        Map<String ,Object> return_map = new HashMap<>();

        User loginUser =new User();  //创建一个数据库实体类对象
        loginUser.setUsername(username);
        loginUser.setPassword(password);

        try {
            User user = dao.login(loginUser);
            if(user != null) {
                req.getSession().setAttribute("user", user);//绑定数据
                return_map.put("msg",true);
                System.out.println("发发发发发 ");
            }else {

                System.out.println("密码错误! ");
```

```
                return_map.put("msg",false);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        ObjectMapper mapper = new ObjectMapper();   //利用Jackson将map转化为json对象
        mapper.writeValue(resp.getWriter(),return_map);
    }
}
```

2 FindMusicServlet实现

用户进行登录后，需要先进行查询，将查询结果显示到页面上。

```
@WebServlet("/findMusic")
public class FindMusicServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html; charset=utf-8");
        System.out.println("测试查找函数");
        String str = req.getParameter("musicName");
        MusicDao dao = new MusicDao();
        List<Music> musics = null;
        if(str!=null) {
            musics = dao.ifMusic(str);//关键字查询

        }else {
            musics = dao.findMusic();
        }
        for (Music music : musics) {
            System.out.println(music.getUrl());
        }

        ObjectMapper mapper = new ObjectMapper();
        mapper.writeValue(resp.getWriter(),musics);
    }
}
```

3 删除音乐信息实现

3.1 删除某个音乐（DeleteMusicServlet）

获取前端参数id.

```
@WebServlet("/deleteServlet")
public class DeleteMusicServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println("删除指定音乐！ ");
```

```
        req.setCharacterEncoding("utf-8");
        resp.setContentType("application/json;charset=utf-8");
        Map<String,Object> map=new HashMap<>();
        String strId = req.getParameter("id");
        int id = Integer.parseInt(strId);
        System.out.println("id:"+ id);
        try {
            MusicDao musicDao = new MusicDao();
            //1.查找有没有当前id
            Music music = musicDao.findMusicById(id);
            //没有这个id的音乐 直接返回
            if(music == null) return;
            //2、如果有就开始删除库中的音乐
            int delete = musicDao.deleteMusicById(id);
            System.out.println("delete:"+delete);

            if(delete == 1){
                //3、数据库删除完成后，检查还是否存在。如果不存在，那么删除掉磁盘上的文件
                File file = new
File("E:\\Javaproject\\GaoBoMusic\\web\\"+music.getUrl()+".mp3");
                System.out.println("文件是否存在: "+file.exists());
                System.out.println("file: "+file);
                if(file.delete()){
                    //证明删除成功
                    map.put("msg",true);
                }else {
                    map.put("msg",false);
                    System.out.println("文件名: "+file.getName());
                    System.out.println("删除文件失败! ");
                }

            }else {
                map.put("msg",false);
            }
        }catch (Exception e) {
            e.printStackTrace();
        }
        //将map转化为json
        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(resp.getWriter(),map);
    }
}
```

3.2 删除选中音乐

获取前端选中的id数组。

```
@WebServlet("/deleteSelMusicServlet")
public class DeleteSelMusicServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
```

```
        resp.setContentType("application/json;charset=utf-8");

        String[] values = req.getParameterValues("id[]");
        System.out.println("deleteSelectedServlet: "+Arrays.toString(values));
        //删除
        int sum=0;
        Map<String,Object> map=new HashMap<>();

        MusicDao musicDao = new MusicDao();
        for(int i=0;i<values.length;i++){
            int j = Integer.parseInt(values[i]);
            System.out.println("j :"+j);
            //调用Service层方法删除
            Music music = musicDao.findMusicById(j);
            int delete = musicDao.deleteMusicById(j);
            //sum=sum+delete;
            if(delete == 1) {
                //3、数据库删除完成后，检查还是否存在。如果不存在，那么删除掉磁盘上的文件
                File file = new File("E:\\Javaproject\\GaoBoMusic\\web\\" + music.getUrl() +
".mp3");
                System.out.println("文件是否存在: " + file.exists());
                System.out.println("file: " + file);
                if (file.delete()) {
                    //证明删除成功
                    //map.put("msg", true);
                    sum=sum+delete;
                } else {
                    //map.put("msg", false);
                    System.out.println("文件名: " + file.getName());
                    System.out.println("删除文件失败! ");
                }
            }
        }
        System.out.println("sum: "+sum);
        //sum==values.length 说明选中的所有元素已经全部删除了
        if(sum==values.length){
            //证明删除成功
            map.put("msg",true);
        }else {
            map.put("msg",false);
        }
        //将map转化为json
        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(resp.getWriter(),map);
    }
}
```


4 上传音乐

上传音乐分为2步：

第一步将音乐上传到服务器

第二步将音乐信息存放到数据库

```java
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-06-23
 * Time: 15:41
 */
@WebServlet("/upload")
//@MultipartConfig
public class UploadMusicServlet extends HttpServlet {

    private final String SAVEPATH="E://Javaproject//GaoBoMusic//web//music//";

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        User user = (User) request.getSession().getAttribute("user");
        if (user == null) {
            request.setAttribute("Msg", "请登录后再进行上传");
        } else {

            FileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);
            List<FileItem> items = null;

            try {
                items = upload.parseRequest(request);
            } catch (FileUploadException e) {
                e.printStackTrace();
                return;
            }


            System.out.println("items:"+items );
            FileItem item = items.get(0);
            System.out.println("item:  "+item);

            String fileName = item.getName();

            System.out.println("fileName"+fileName);
            request.getSession().setAttribute("fileName", fileName);

            try {
                item.write(new File(SAVEPATH, fileName));
            } catch (Exception e) {
                e.printStackTrace();
            }
            response.sendRedirect("uploadsucess.html");
        }
    }
}
```

第二步：

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-06-23
 * Time: 17:12
 */
@WebServlet("/uploadsucess")
public class UploadInsertServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html; charset=utf-8");

        String strings = (String)req.getSession().getAttribute("fileName");
        String[] titles = strings.split("\\.");
        String title = titles[0];

        System.out.println("title:" + title);
        String url = "music/"+title;
        System.out.println("url: "+url);
        String singer = req.getParameter("singer");
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
        String time=sdf.format(new Date());
        MusicDao dao = new MusicDao();
        User user = (User) req.getSession().getAttribute("user");
        int user_id = user.getId();
        int num = dao.Insert(title,singer,time,url,user_id);
        if(num!=0){
            resp.sendRedirect("list.html");
        }
    }
}
```

5 添加喜欢的音乐到喜欢列表

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-06-23
 * Time: 14:56
 */
@WebServlet("/loveMusicServlet")
public class LoveMusicServlet extends HttpServlet {
    @Override

    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
```

```
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        resp.setContentType("application/json;charset=utf-8");

        String strId = req.getParameter("id");
        int musicId = Integer.parseInt(strId);
        System.out.println("musicID: "+musicId);

        User user = (User) req.getSession().getAttribute("user");
        int user_id = user.getId();
        MusicDao musicDao = new MusicDao();

        Map<String,Object> map=new HashMap<>();

        //插入之前需要先查看是否该音乐已经被添加到喜欢列表
        boolean effect = musicDao.findMusicByMusicId(user_id,musicId);
        if(effect) {
            map.put("msg",false);
        }else {
            boolean flg = musicDao.insertLoveMusic(user_id,musicId);
            if(flg) {
                map.put("msg",true);
            }else {
                map.put("msg",false);
            }
        }
        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(resp.getWriter(),map);
    }
}
```

6 查找我喜欢的音乐列表

```
@WebServlet("/findLoveMusic")
public class FindLoveMusicServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        resp.setContentType("application/json;charset=utf-8");
        String str = req.getParameter("loveMusicName");
        System.out.println("loveMusicName:"+str);
        User user = (User) req.getSession().getAttribute("user");
        int user_id = user.getId();
        MusicDao musicDao = new MusicDao();
        List<Music> musics = null;
        if(str!=null) {
            musics = musicDao.ifMusicLove(str,user_id);//关键字查询
        }else {
            musics = musicDao.findLoveMusic(user_id);

        }
```

```
        for (Music music : musics) {
            System.out.println(music.getUrl());
        }

        ObjectMapper mapper = new ObjectMapper();
        mapper.writeValue(resp.getWriter(),musics);
    }
}
```

7 移除我喜欢的音乐

```
@WebServlet("/removeLoveServlet")
public class RemoveLoveServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("utf-8");
        resp.setContentType("application/json;charset=utf-8");

        User user = (User) req.getSession().getAttribute("user");
        int user_id = user.getId();

        Map<String,Object> map=new HashMap<>();
        String strId = req.getParameter("id");
        int music_id = Integer.parseInt(strId);

        MusicDao musicDao = new MusicDao();
        int delete = musicDao.removeLoveMusic(user_id,music_id);
        if(delete == 1) {
            map.put("msg",true);
        }else {
            map.put("msg",false);
        }

        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(resp.getWriter(),map);
    }
}
```


前端页面的设计

前端采用HTML+CSS+JS设计。

直接在百度上搜索 "免费网页模板", 能找到很多免费模板网站. 可以直接基于现成的漂亮的页面进行修改.

> tips: 做减法比做加法更容易.

将网页模板解压缩, 拷贝到项目的`webapp 或者 web`目录中.

网址分享:

http://tpl.amazeui.org/

https://ajz.fkw.com/pro11.html?_ta=150&kw=145

前后端服务器数据交互-以登录为例：

```
<script>
    //登录请求
    $(function () {
    $("#submit").click(function () {
        var username=$("#user").val();
        var password=$("#password").val();
        $.ajax({
            url:"/loginServlet",//发送请求的地址
            data:{"username":username,"password":password},//发送给服务器的数据
            type:"POST",//请求方式 ("POST" 或 "GET"), 默认为 "GET"
            dataType:"json",//预期服务器返回的数据类型
            success:function (data) {//请求成功后的回调函数
                console.log(data);
                if(data.msg===true){
                    window.location.href="list.html";
                }else{
                    /*window.location.reload(); 数据清空*/
                    $("#message").text("账号或密码错误，请重试!");
                    $("#user").val("");
                    $("#password").val("");
                    $("#verifycode").val("");
                }
            }
        });
    });
});
</script>
```

后续拓展

1. 在框架上，可以添加一个service层，从整体上达到高内聚低耦合
2. 从业务上可以添加MV列表，实现和音乐列表相同的操作