基于Java Servlet 构建的用户信息管理系统

核心功能

1. 登录、注册
2. 添加用户信息
3. 删除某一个用户信息
4. 删除选中的用户信息
5. 分页查询所有用户信息
6. 模糊查询用户信息
7. 更新用户信息

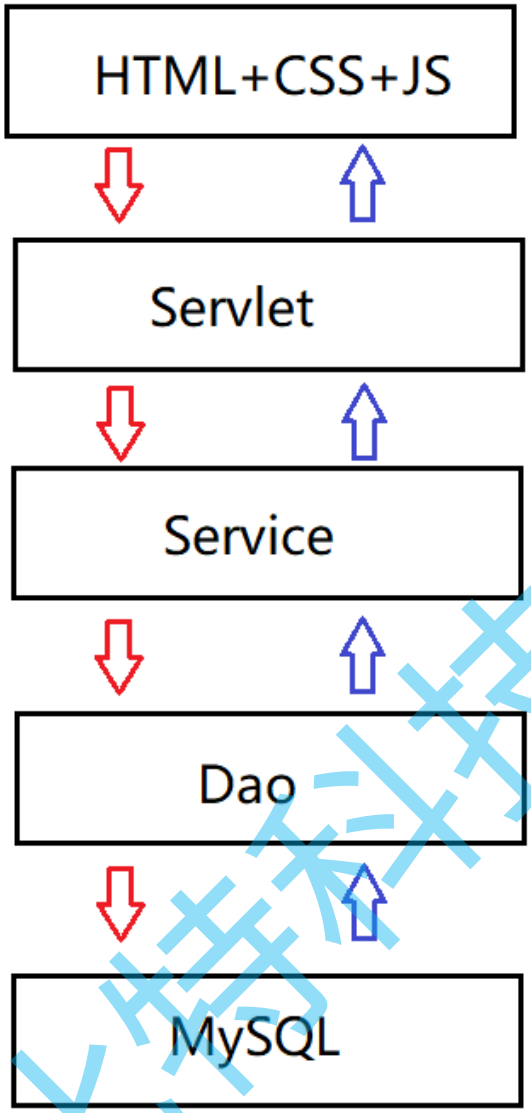
重要知识点

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。

比特科技



比特科技

数据库设计

只需要一张用户表，表示用户的信息

```
drop database if exists usermanger;

create DATABASE if not exists usermanger  character set utf8;

use usermanger;

drop table if exists usermessage;
create table `usermessage` (
    `id` INT PRIMARY KEY auto_increment,
    `name` varchar (60),
    `username` varchar (60) default 'bit',
    `password` varchar (60) default '123456',
    `gender` varchar (4),
    `age` int,
```

```
    `address` varchar (90),
    `qq` varchar (20),
    `email` varchar (30)
);

INSERT INTO usermessage VALUES(1,'张飞','zhangfei','123','男',18,'成
都','1262913815','126@qq.com');
INSERT INTO usermessage VALUES(2,'关羽','guanyu','1234','男',18,'陕
西','1262913816','1262@qq.com');
INSERT INTO usermessage VALUES(3,'张三','zhangsan','1235','女',19,'陕
西','1262913817','1263@qq.com');
INSERT INTO usermessage VALUES(4,'李四','lisi','1236','男',20,'北京','1262913818','1264@qq.com');
INSERT INTO usermessage VALUES(5,'王五','wangwu','1237','女',21,'陕
西','1262913819','1265@qq.com');
INSERT INTO usermessage VALUES(6,'孙权','sunquan','1238','男',22,'上
海','1262913814','1266@qq.com');
INSERT INTO usermessage VALUES(7,'孙悟空','sunwukong','1239','男',23,'陕
西','1262913813','1267@qq.com');
```

用户+页面管理模块设计

创建entity包

1. 创建User类。

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-20
 * Time: 10:47
 */
public class User {
    private int id;
    private String name;
    private String username;
    private String password;
    private String gender;
    private int age;
    private String address;
    private String qq;
    private String email;
    //get set toString 自己实现
}
```

2. 创建分页对象

```
/**
 * Created with IntelliJ IDEA.

 * Description:
```

比特科技

```
 * User: GAOBO
 * Date: 2020-05-20
 * Time: 10:55
 */
//泛型类的原因：为了以后好拓展
public class PageBean<T> {
    private int totalCount;   //总记录数
    private int totalPage; //总页码
    private List<T> list;  //每页中的数据
    private int currentPage;  //当前页码
    private int rows;   //每页的记录数
     //get set toString 自己实现
}
```

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：

```
<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();
```

比特科技

```
    }

    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);
```

比特科技

```
    }
}
```

输出结果为：

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

2 登录

```
请求：
POST  /loginServlet

响应：
{msg: true}
```

3 添加用户

```
请求：
POST  /addServlet

响应：
{msg: true}
```

4 删除某一个用户信息

```
请求：
GET  /deleteServlet?id=1
```

5 删除选中的用户信息

```
请求：
POST  /deleteSelectedServlet

响应：
{msg: true}
```

6 分页查询所有用户信息

```
请求：
POST      /findByPageServlet
data:{currentPage,rows,name,address,email}

响应：
响应体内容为，每一页的用户信息
```

比特科技

7 模糊查询用户信息

```
请求:
POST      /findByPageServlet
data:{currentPage,rows,name,address,email}

响应:
响应体内容为，每一页的用户信息
```

8 更新用户信息

```
--------------------更新用户信息之前，先请求得到要修改的用户的信息--------------------------
请求:
POST     /returnServlet

响应:
当前需要更新的用户的信息
--------------------------更新完成后，提交更新信息请求-------------------------------------
请求:
POST      /updateServlet

响应:
{msg: true}
```

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。

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-09
 * Time: 20:38
 */
public class JDBCUtils {
    private static String url = "jdbc:mysql://127.0.0.1:3306/usermanger?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
```

比特科技

```
        if(DATASOURCE == null){
            synchronized (JDBCUtils.class){
                if(DATASOURCE == null){
                    DATASOURCE = new MysqlDataSource();
                    ((MysqlDataSource) DATASOURCE).setUrl(url);
                    ((MysqlDataSource) DATASOURCE).setUser(username);
                    ((MysqlDataSource) DATASOURCE).setPassword(password);
                }
            }
        }
        return DATASOURCE;
    }

    public static Connection getConnection(){
        System.out.println("getConnection1");
        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 close(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

比特科技

```java
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-12
 * Time: 23:40
 */
public class UserDao {

    public User login(User loginUser) {
        return null;
    }

    public int add(User addUser) {
        return 0;
    }

    public int delete(int id) {
        return 0;
    }

    public User find(int id) {
        return null;
    }

    public int update(User updateUser) {
        return 0;
    }
    /**
     *    分页查询
     *    start：开始查询的起始位置
     *    rows：共查询的记录
     *    map：包含：currentPage、rows、name、address、email
     */
    public List<User> findByPage(int start, int rows,Map<String, String[]> map) {
        return null;
    }

    /**
     * 查询共有多少条记录
     * @param map 包含 name address email
     * @return
     */
    public int findAllRecord(Map<String, String[]> map) {
        return 1;
    }

}
```

比特科技

3 实现UserDao.login

```java
public User login(User loginUser) {
```

```
    Connection connection = null;
    PreparedStatement ps  = null;
    ResultSet rs = null;
    User user = null;
    try {
        String sql = "select * from usermessage where username=? and password=?";
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(sql);
        ps.setString(1, loginUser.getUsername());
        ps.setString(2,loginUser.getPassword());
        rs = ps.executeQuery();

        if(rs.next()){
            user = new User();
            user.setId(rs.getInt("id"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            user.setName(rs.getString("name"));
            user.setAddress(rs.getString("address"));
            user.setAge(rs.getInt("age"));
            user.setGender(rs.getString("gender"));
            user.setQq(rs.getString("qq"));
            user.setEmail(rs.getString("email"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        JDBCUtils.close(connection,ps,rs);
    }
    return user;
}
```

4 实现UserDao.add

```
public int add(User addUser) {
    Connection connection = null;
    PreparedStatement ps  = null;
    try {
        String sql="insert into usermessage(name,gender,age,address,qq,email)
values(?,?,?,?,?,?)";
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(sql);
        ps.setString(1, addUser.getName());
        ps.setString(2, addUser.getGender());
        ps.setInt(3,addUser.getAge());
        ps.setString(4, addUser.getAddress());
        ps.setString(5, addUser.getQq());
        ps.setString(6, addUser.getEmail());

        int ret = ps.executeUpdate();
        return ret;

    } catch (SQLException e) {
```

比特科技

```
            e.printStackTrace();
        }finally {
            JDBCUtils.close(connection,ps,null);
        }
        return 0;
    }
```

5 实现UserDao.delete

```
public int delete(int id) {
    System.out.println("Delete: "+id);
    //删除成功返回1
    Connection connection = null;
    PreparedStatement ps  = null;

    try {
        String sql="delete from usermessage where id=?";
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(sql);
        ps.setInt(1, id);
        int ret = ps.executeUpdate();
        return ret;
    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        JDBCUtils.close(connection,ps,null);
    }
    return 0;
}
```

6 实现UserDao.find

```
public User find(int id) {
    Connection connection = null;
    PreparedStatement ps  = null;
    ResultSet rs = null;

    User user = null;
    String sql = "select * from usermessage where id=?";
    try {
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(sql);
        ps.setInt(1,id);
        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.setName(rs.getString("name"));

            user.setAddress(rs.getString("address"));
```

比特科技

```
            user.setAge(rs.getInt("age"));
            user.setGender(rs.getString("gender"));
            user.setQq(rs.getString("qq"));
            user.setEmail(rs.getString("email"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        JDBCUtils.close(connection,ps,rs);
    }
    return user;
}
```

7 实现UserDao.update

```
public int update(User updateUser) {
    Connection connection = null;
    PreparedStatement ps  = null;

    String sql = "update usermessage set name=?,age=?,gender=?,address=?,qq=?,email=? where
id=?";
    try {
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(sql);
        ps.setString(1, updateUser.getName());
        ps.setInt(2,updateUser.getAge());
        ps.setString(3, updateUser.getGender());

        ps.setString(4, updateUser.getAddress());
        ps.setString(5, updateUser.getQq());
        ps.setString(6, updateUser.getEmail());
        ps.setInt(7,updateUser.getId());

        int ret = ps.executeUpdate();
        return ret;
    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        JDBCUtils.close(connection,ps,null);
    }
    return 0;
}
```

8 实现UserDao.findByPage

```
public static void setValues(PreparedStatement preparedStatement, Object... values) throws
SQLException {
    System.out.println("====setValues==" + values.length);
    for (int i = 0; i < values.length; i++) {
        preparedStatement.setObject(i + 1, values[i]);

    }
```

比特科技

```
}

/**
*    分页查询
*    start：开始查询的起始位置
*    rows：共查询的记录
*    map：包含：currentPage、rows、name、address、email
*/
public List<User> findByPage(int start, int rows,Map<String, String[]> map) {
    String sql="select * from usermessage where 1=1";
    StringBuilder s=new StringBuilder(sql);
    Set<String> set = map.keySet();
    List<Object> list =new ArrayList<>();
    for(String key  : set){
        String value=map.get(key)[0];
        if(value!=null && !"".equals(value)){
            //有值
            s.append(" and ").append(key).append(" like ? ");
            list.add("%"+value+"%");
        }
    }
    s.append(" limit ?,? ");

    list.add(start);
    list.add(rows);

    //s: select * from usermessage where 1=1 and name like ?  and address like ?  and email like
?  limit ?,?
    //select * from usermessage where 1=1 and name like '%gaobo%'  and address like '%陕西%'
    System.out.println("s: "+ s);

    //list: [%gaobo%, %陕西%, %gaobo@bitedu.tech%, 0, 5] -->这是第一页
    System.out.println("list: "+ list);

    Connection connection = null;
    PreparedStatement ps  = null;
    ResultSet rs = null;
    List<User> users = new ArrayList<>();

    try {

        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(s.toString());
        setValues(ps,list.toArray());
        rs = ps.executeQuery();

        while(rs.next()){
            User user = new User();
            user.setId(rs.getInt("id"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            user.setName(rs.getString("name"));

            user.setAddress(rs.getString("address"));
```

比特科技

```
            user.setAge(rs.getInt("age"));
            user.setGender(rs.getString("gender"));
            user.setQq(rs.getString("qq"));
            user.setEmail(rs.getString("email"));
            users.add(user);
        }
    }catch (SQLException e) {
        e.printStackTrace();
    }finally {
        JDBCUtils.close(connection,ps,rs);
    }
    return users;
}
```

9 实现UserDao.findAllRecord

```
/**
* 查询共有多少条记录
 * @param map 包含 name address email
* @return
*/
public int findAllRecord(Map<String, String[]> map) {
    String sql="select count(*) from usermessage where 1=1";
    StringBuilder s=new StringBuilder();
    s.append(sql);
    Set<String> keySet = map.keySet();
    List<Object> list=new ArrayList<>();
    for(String key:keySet){
        String value=map.get(key)[0];
        if(value!=null && !"".equals(value)){
            //有值
            s.append(" and ").append(key).append(" like ?");
            list.add("%"+value+"%");
        }
    }
    System.out.println("findAllRecord::sql:" + s);
    System.out.println("findAllRecord::list:"+list);

    Connection connection = null;
    PreparedStatement ps  = null;
    ResultSet rs = null;
    int count = 0;
    try {
        connection = JDBCUtils.getConnection();
        ps = connection.prepareStatement(s.toString());
        setValues(ps,list.toArray());
        rs = ps.executeQuery();
        if(rs.next()){
            count = rs.getInt(1);  //对总记录数赋值  等价于rs.getInt("id");
        }
    } catch (SQLException e) {

        e.printStackTrace();
```

比特科技

```
    } finally {
        JDBCUtils.close(connection,ps,rs);
    }
    return count;
}
```

Service层设计实现

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-14
 * 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;
    }

    //添加方法
    public int add(User addUser) {
        UserDao userDao=new UserDao();
        int i = userDao.add(addUser);
        return i;
    }
    //删除方法
    public int delete(int id) {
        UserDao userDao=new UserDao();
        int i = userDao.delete(id);
        return i;
    }
    //根据id查询
    public User find(int id) {
        UserDao userDao=new UserDao();
        User user = userDao.find(id);
        return user;
    }
    //更新方法
    public int update(User updateUser) {
        UserDao userDao=new UserDao();
        int i = userDao.update(updateUser);
        return i;
    }

    /**
     * @param currentPage 当前页
     * @param rows 每页的行数
```

比特科技

```
     * @param map  包含 name address email
     * @return
     *
     */
    public  PageBean<User> findAllByPage(int currentPage,int rows,Map<String, String[]> map) {

        PageBean<User> pageBean=new PageBean<>();

        UserDao userDao=new UserDao();

        int totalPage;
        int record = userDao.findAllRecord(map);//查询共有多少条记录

        //总共的页数 totalPage
        if(record%rows==0){
            totalPage=record/rows;
        }else{
            totalPage=record/rows+1;
        }

        /**
         *  每一页的开始位置 = (当前页数-1) * 行数
         *  第一页开始位置:  (1-1) *5 = 0
         *  第一页开始位置:  (2-1) *5 = 5
         */
        int start=(currentPage-1)*rows;

        List<User> users = userDao.findByPage(start, rows, map);

        pageBean.setCurrentPage(currentPage);
        pageBean.setList(users);
        pageBean.setRows(rows);
        pageBean.setTotalCount(record);
        pageBean.setTotalPage(totalPage);
        return pageBean;
    }

}
```

比特科技

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

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-16
 * Time: 23:30

 */
```

```
@WebServlet("/loginServlet")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println("comeheer");
        response.setContentType("application/json;charset=utf-8");   //设置响应数据的数据格式和编
码格式

        String username = request.getParameter("username");
        String password = request.getParameter("password");

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

        UserService userService=new UserService();   //创建service层的对象

        Map<String ,Object> return_map = new HashMap<>();   //创建一个map集合，存放返回到客户端的数
据

        User user = userService.login(loginUser);   //调用service层的登方法，判断是否登录成功
        if (user!=null){
            //说明数据库中有，显示登录成功，把登录信息存入session
            request.getSession().setAttribute("user",user);
            //返回给登录页面json数据
            return_map.put("msg",true);
        }else{
            System.out.println("账号或密码错误");
            //账号或密码错误
            return_map.put("msg",false);
        }

        ObjectMapper mapper = new ObjectMapper();   //利用Jackson将map转化为json对象

        mapper.writeValue(response.getWriter(),return_map);

    }
}
```

2 FindByPageServlet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。

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-16
 * Time: 20:38
 */
@WebServlet("/findByPageServlet")

public class FindByPageServlet extends HttpServlet {
```

比特科技

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("application/json;charset=utf-8");
        request.setCharacterEncoding("utf-8");
        String currentPage = request.getParameter("currentPage");//当前页：1
        String rows = request.getParameter("rows");//几行

        //获取到所有的前端参数：load(rows,currentPage,name,address,email)
        Map<String, String[]> parMap = request.getParameterMap();
        //这里注意一下，java的原生对象是不允许修改的,重新创建map，修改map就行了
        Map<String, String[]> map = new HashMap<>(parMap);
        map.remove("currentPage");
        map.remove("rows");

        //强制类型转换
        int current = Integer.parseInt(currentPage);//当前页 :如第1页
        int row = Integer.parseInt(rows);//行数   5
        //创建service层的对象
        UserService userService=new UserService();
        //查询  一页的5条记录
        PageBean<User> pageBean = userService.findAllByPage(current, row, map);

        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(response.getWriter(),pageBean);
    }
}
```

3 修改/更新用户信息实现

1. 先根据id，查询当前要修改的用户是否存在。把当前查询到的用户写入session。然后跳转的更新页面（`FindUserServlet`）。
2. 获取到当前用户的session信息，并且返回（ReturnServlet）。
3. 更新完成，提交信息（）。

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * 找到需要更新的用户，记录到session
 * User: GAOBO
 * Date: 2020-05-18
 * Time: 22:00
 */
@WebServlet("/findUserServlet")
public class FindUserServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setCharacterEncoding("utf-8");   //设置编码格式
        String id = request.getParameter("id");  //获取参数
        int i = Integer.parseInt(id);    //将String类型的id转化为INT类型
        //调用Service层的方法，查询id为i的数据
        UserService userService=new UserService();

        User user = userService.find(i);
```

比特科技

```
        //将user存到session域
        request.getSession().setAttribute("user",user);
        //跳转到update.html
        response.sendRedirect("/update.html");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        this.doPost(request, response);
    }
}
```

```
/**
 * Created with IntelliJ IDEA.
 * Description:把刚刚记录到session里面的数据，取出来转化为json.方便在更新页面上预显示信息
 * User: GAOBO
 * Date: 2020-05-18
 * Time: 22:30
 */
@WebServlet("/returnServlet")
public class ReturnServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("application/json;charset=utf-8");
        Object user = request.getSession().getAttribute("user");
        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(response.getWriter(),user);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        this.doPost(request, response);
    }
}
```

```
/**
 * Created with IntelliJ IDEA.
 * Description:当修改页面完成修改后，点击提交将新的user信息进行更新
 * User: GAOBO
 * Date: 2020-05-18
 * Time: 23:08
 */
@WebServlet("/updateServlet")
public class UpdateServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        request.setCharacterEncoding("utf-8");
```

比特科技

```
        response.setContentType("application/json;charset=utf-8");
        Object us = request.getSession().getAttribute("user");


        String name = request.getParameter("name");
        String gender = request.getParameter("gender");
        String ageString = request.getParameter("age");
        int age = Integer.parseInt(ageString);
        String address = request.getParameter("address");
        String qq = request.getParameter("qq");
        String email = request.getParameter("email");

        User user= (User) us;
        User updateUser =new User();
        //将id赋给updateUser对象
        updateUser.setId(user.getId());
        updateUser.setName(name);
        updateUser.setGender(gender);
        updateUser.setAge(age);
        updateUser.setAddress(address);
        updateUser.setQq(qq);
        updateUser.setEmail(email);

        //调用Service层的方法，更新
        UserService userService=new UserService();
        int i = userService.update(updateUser);
        Map<String,Object> return_map=new HashMap<>();
        if (i==1){
            return_map.put("msg",true);
        }else{
            return_map.put("msg",false);
        }
        ObjectMapper mapper=new ObjectMapper();
        mapper.writeValue(response.getWriter(),return_map);
    }
}
```

比特科技

4 删除用户信息实现

4.1 删除单个用户

获取前端参数id.

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-23
 * Time: 23:08
 */
@WebServlet("/deleteServlet")
public class DeleteServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
```

```
ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        String id = request.getParameter("id");
        System.out.println("DeleteServlet: "+id);
        UserService userService=new UserService();
        int i= 0;
        try {
            i = Integer.parseInt(id);
        } catch (NumberFormatException e) {
            e.printStackTrace();
        }
        int result = userService.delete(i);
        if(result==1){
            response.sendRedirect("/list.html");
        }else{
            response.sendRedirect("/list.html");
        }

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        this.doPost(request, response);
    }
}
```

4.2 删除选中用户

获取前端选中的id数组。

```
/**
 * Created with IntelliJ IDEA.
 * Description:
 * User: GAOBO
 * Date: 2020-05-24
 * Time: 00:08
 */
@WebServlet("/deleteSelectedServlet")
public class DeleteSelectedServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("application/json;charset=utf-8");
        String[] values = request.getParameterValues("id[]");
        System.out.println("deleteSelectedServlet: "+Arrays.toString(values));
        //删除
        int sum=0;
        UserService userService=new UserService();
        Map<String,Object> map=new HashMap<>();
        for(int i=0;i<values.length;i++){
            int j = Integer.parseInt(values[i]);
            //调用Service层方法删除
            int delete = userService.delete(j);
```

比特科技

```
            sum=sum+delete;
        }
        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(response.getWriter(),map);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        this.doPost(request, response);
    }
}
```

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";
```

比特科技

```
                }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. 对用户的字段进行添加如：用户等级，VIP，关注人数，粉丝数，头像，手机号等。
2. 功能上添加：根据等级进行查询，根据粉丝人数进行查询等。

比特科技