Java项目草稿

人际管理系统

- 帮助管理个人人际关系
- 涵盖周围人信息
- 人员分类
- 对人际关系进行分析

注册/登录页面

适当美华添加背景图片

主页/内容

个人信息板块

- 个人

  图片上传，信息填写等
- 收藏
- 关注
- 注销

  回到登陆页面

数据库

- user库 存放账号信息
- person库 存放个人信息

2020/7/18 第一天

使用**BootStrap框架**编写HTML代码,BS可以理解为编写好的CSS代码,通过自己写HTML框架,class设置为相应格式,就可以使用相应的样式,十分方便,但局限性也很明显,尽管如此还是很适合我这种后端来重新编写前端代码的.

1.完成了导航栏和注册页面

注意javascript的导入，需要在body的最后导入两个js文件，缺一不可.

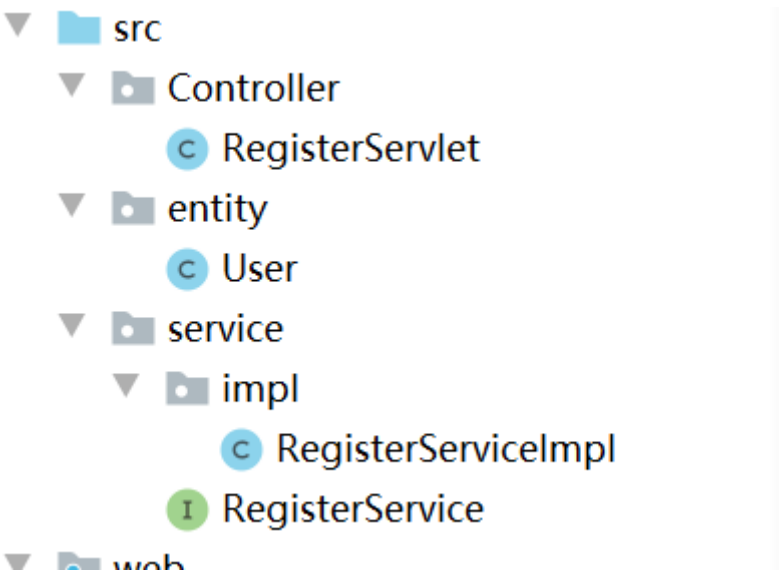
```
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
```

上个截图

2.使用表单的method 为 post,action为创建的Java程序

3.使用MVC框架搭建程序

流程可以分为 **Controller->Service->Repository->DB**,分别创建相关的文件夹和使用相应的组件,目的也是为了避免程序臃肿,要各个程序各司其职,像一条流水线.这是今天所创建的,创建了User对象,RegisterServlet以及Service



当前编写的RegisterServlet类

```
@WebServlet("/register")
public class RegisterServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
//        设置编码
        resp.setCharacterEncoding("utf-8");
        resp.setContentType("text/html; charset=utf-8");
        PrintWriter out = resp.getWriter();

//        获取注册信息
        String email = req.getParameter("email");
        String tel = req.getParameter("tel");
        String name = req.getParameter("name");
        String password = req.getParameter("password");
```

```
        entity.User user = new User(email,tel,name,password);

    }
}
```

4.User对象

```
package entity;

public class User {
    private String email;
    private String tel;
    private String name;
    private String password;

    public User() {
    }

    public User(String email, String tel, String name, String password) {
        this.email = email;
        this.tel = tel;
        this.name = name;
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

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

    public String getTel() {
        return tel;
    }

    public void setTel(String tel) {
        this.tel = tel;
    }

    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;
    }
```

```
}
```

5.使用接口的目的是方便调试,方便改进等

```
public interface RegisterService {
    public void Register(User user);
}
```

接口中编写了一个注册方法,传入User对象作为参数,明天准备将对象写入到数据库.

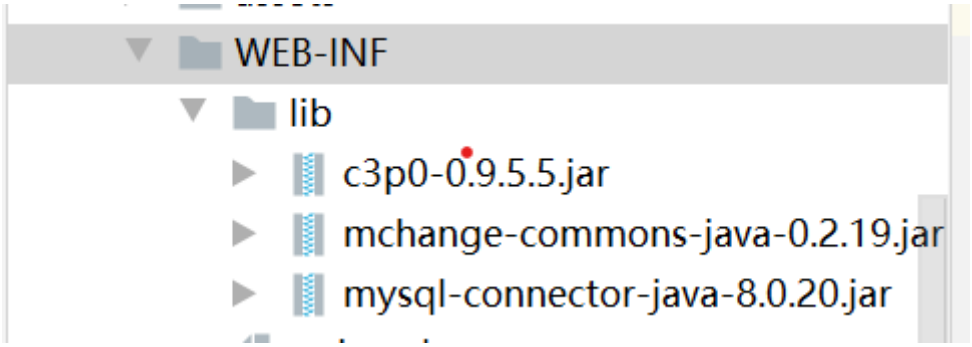
2020/7/19 第二天

1.配置文件

将c3p0-config.xml放入src根目录，配置信息如下

```
<?xml version="1.0" encoding="UTF-8"?>

<c3p0-config>

    <named-config name="testc3p0">

        <property name="driverClass">com.mysql.cj.jdbc.Driver</property>
<!--        3306/后跟数据库的名称，serverTimezone必须要加，否则会报错(版本问题)-->
        <property name="jdbcUrl">jdbc:mysql://localhost:3306/User?
serverTimezone=UTC</property>
<!--        用户名和密码-->
        <property name="user">root</property>
        <property name="password">123456</property>

        <property name="acquireIncrement">5</property>
        <property name="initialPoolSize">5</property>
        <property name="maxPoolSize">10</property>
        <property name="minPoolSize">5</property>

    </named-config>

</c3p0-config>
```

然后将c3p0的jar包，mchange-commons jar包和mysql-connector的jar包放入web-inf包下，用IDEA导入。



2.然后创建utils包，

创建JDBCTools类，导入数据库连接池

```
package utils;

import com.mchange.v2.c3p0.ComboPooledDataSource;

import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCTools {
    private static DataSource dataSource;

    static {
        dataSource = new ComboPooledDataSource("testc3p0");
    }

    public static Connection getConnection(){
        Connection connection = null;
        try {
            connection = dataSource.getConnection();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return connection;
    }

    public static void release (Connection connection, Statement statement,
ResultSet resultSet){

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

```
    }

    //测试是否连接成功
    public static void main(String[] args) {
        System.out.println(JDBCTools.getConnection());
    }
}
```

成功以后的输出

```
7月 19, 2020 9:29:45 上午 com.mchange.v2.log.MLog
信息: MLog clients using java 1.4+ standard logging.
7月 19, 2020 9:29:45 上午 com.mchange.v2.c3p0.C3P0Registry
信息: Initializing c3p0-0.9.5.5 [built 11-December-2019 22:07:46 -0800; debug?
true; trace: 10]
7月 19, 2020 9:29:45 上午 com.mchange.v2.c3p0.impl.AbstractPoolBackedDataSource
信息: Initializing c3p0 pool... com.mchange.v2.c3p0.ComboPooledDataSource [
acquireIncrement -> 5, acquireRetryAttempts -> 30, acquireRetryDelay -> 1000,
autoCommitOnClose -> false, automaticTestTable -> null, breakAfterAcquireFailure
-> false, checkoutTimeout -> 0, connectionCustomizerClassName -> null,
connectionTesterClassName -> com.mchange.v2.c3p0.impl.DefaultConnectionTester,
contextClassLoaderSource -> caller, dataSourceName -> testc3p0,
debugUnreturnedConnectionStackTraces -> false, description -> null, driverClass
-> com.mysql.cj.jdbc.Driver, extensions -> {}, factoryClassLocation -> null,
forceIgnoreUnresolvedTransactions -> false, forceSynchronousCheckins -> false,
forceUseNamedDriverClass -> false, identityToken ->
1hge160abrxryse1powhpb|51081592, idleConnectionTestPeriod -> 0, initialPoolSize
-> 5, jdbcUrl -> jdbc:mysql://localhost:3306/Relationship?serverTimezone=UTC,
maxAdministrativeTaskTime -> 0, maxConnectionAge -> 0, maxIdleTime -> 0,
maxIdleTimeExcessConnections -> 0, maxPoolSize -> 10, maxStatements -> 0,
maxStatementsPerConnection -> 0, minPoolSize -> 5, numHelperThreads -> 3,
preferredTestQuery -> null, privilegeSpawnedThreads -> false, properties ->
{password=******, user=******}, propertyCycle -> 0,
statementCacheNumDeferredCloseThreads -> 0, testConnectionOnCheckin -> false,
testConnectionOnCheckout -> false, unreturnedConnectionTimeout -> 0,
userOverrides -> {}, usesTraditionalReflectiveProxies -> false ]
com.mchange.v2.c3p0.impl.NewProxyConnection@3967e60c [wrapping:
com.mysql.cj.jdbc.ConnectionImpl@60d8c9b7]

Process finished with exit code 0
```

3.创建数据库

创建relationship数据库，创建user表

<Filter Criteria>

	email	tel	name	password
1	2568805557@qq.com	15925688557	jindian	123456
2	2376456392@qq.com	15908629246	zhiyuan	123456
3	1828524004@qq.com	18928524004	lakesi	123456

4. 在repository中写入注册方法

与数据库取得连接，然后执行sql语句，拿到执行的返回值(判断是否执行成功)

```
public class UserRepositoryImpl implements UserRepository {
    @Override
    public int Register(User user) {
        //使用num判断是否注册成功
        int num = 0;

        Connection connection = JDBCTools.getConnection();
        String sql = ("INSERT INTO user values (?,?,?,?)");
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        try {
            statement = connection.prepareStatement(sql);
            statement.setString(1,user.getEmail());
            statement.setString(2,user.getTel());
            statement.setString(3,user.getName());
            statement.setString(4,user.getPassword());
            num = statement.executeUpdate();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            JDBCTools.release(connection,statement,resultSet);
        }

        return num;

    }
```

然后参数逐层返回，注册成功则跳转首页。

5.在repository中写入登录方法

登录流程和注册类似，**LoginServlet -> LoginService ->Repository** ,注意一点，登陆成功的判断条件，以及sql语句，这里选择接受sql执行语句的返回值，如果成功则创建一个非null的User，否则，就返回null的User，然后在向上传递的过程中判断是否为null即可表示是否登录成功。

```
    //登录
    @Override
    public User Login(User user) {
        Connection connection = JDBCTools.getConnection();
        String sql = ("select * from user where tel = ? and password = ?");
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        User user1 = null;
        try {
            statement = connection.prepareStatement(sql);
            statement.setString(1,user.getTel());
            statement.setString(2,user.getPassword());
            resultSet = statement.executeQuery();
            if(resultSet.next()){
                //如果不是null，则登陆成功
                user1 = new User();
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
```

```
            JDBCTools.release(connection,statement,resultSet);
        }
        return user1;
    }
```

6.主页显示用户昵称

通过session共享机制，在jsp中将个人替换为java变量

```
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true"
                       aria-expanded="false"><%=uname%><span class="caret">
</span></a>
```

然后在登录时获取登陆对象的昵称信息，那么就需要底层的数据库中获取user的昵称

```
  public User Login(User user) {
        Connection connection = JDBCTools.getConnection();
        String sql = ("select * from user where tel = ? and password = ?");
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        User user1 = null;
        try {
            statement = connection.prepareStatement(sql);
            statement.setString(1, user.getTel());
            statement.setString(2, user.getPassword());
            resultSet = statement.executeQuery();
            while (resultSet.next()) {
                //一定要new一个user，否则会空指针
                user1 = new User();
                user1.setEmail(resultSet.getString("email"));
                user1.setTel(resultSet.getString("tel"));
                user1.setName(resultSet.getString("name"));
                user1.setPassword(resultSet.getString("password"));
            }
```

注意，当resultset.next为true的时候，一定要在赋值的时候new一个User（），这个bug找了很久，因为之前的user1是null。

然后再LoginService中创建一个Search方法，专门返回登录后完整的user对象

```
@Override
public User Search(User user) {
    UserRepositoryImpl userRepository = new UserRepositoryImpl();
    User user1 = userRepository.Login(user);
    return user1;
}
```

然后页面就可以显示用户的昵称了

7.未登录用户跳转登陆页面，禁止访问主页

```
<%
    //如果用户名为空，不允许访问首页，从而禁止跳过登录
    String uname = (String)session.getAttribute("uname");
    if (uname == null){
        response.sendRedirect("Login.jsp");
    }
%>
```

2020/7/21 第四天

1.向数据库写入图片

form一定要写入enctype="multipart/form-data",代表传入数据，input type使用表单

```
 <form class="form-inline" action="/homepage" method="post"
enctype="multipart/form-data">
                    <div class="form-group">
                        上传背景图片<input type="file"  name="UserCover">
                    </div>
                    <div class="form-group">
                        <input type="submit" value="提交">
                    </div>
                </form>
```

Servlet

```
 @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        try {
            //创建核心组件
            DiskFileItemFactory fileItemFactory = new DiskFileItemFactory();
            ServletFileUpload servletFileUpload = new
ServletFileUpload(fileItemFactory);
            //拿到一个FileItem集合
            List<FileItem> list = servletFileUpload.parseRequest(req);
            for(FileItem fileItem : list){
                //判断传递来的是否是文件
                if(fileItem.isFormField()){ //说明是表单数据,则说明出现错误
                    return;
                }else{ //说明是文件
                    String fileName = fileItem.getName();
                    //获取输入流
                    InputStream inputStream = fileItem.getInputStream();

                    ByteArrayOutputStream out = new ByteArrayOutputStream();
                    //输出流写图片数据
                    int temp = 0;
                    byte[] b = new byte[1024];
                    while ((temp = inputStream.read(b))!= -1){
                        out.write(b,0,temp);
                    }
//                    获取图片二进制数据
                    byte[] array = out.toByteArray();
```

```
                    HomePageServiceImpl homePageService = new
HomePageServiceImpl();
                    //获取session中的tel
                    HttpSession httpSession = req.getSession();
                    String tel = (String)httpSession.getAttribute("utel");
                    if(homePageService.UploadUserCover(array,tel)){
                        System.out.println("上传成功");
                    } else {
                        System.out.println("上传失败");
                    }

                    //关闭相关组件
                    out.close();
                    inputStream.close();
                }
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        }
    }
```

数据库

```
//更改用户背景
    @Override
    public int UploadUserCover(byte[] bytes,String tel) {
        //通过num判断是否更改成功
        int num = 0;
        Connection connection = JDBCTools.getConnection();
        String sql = ("update person set usercover=? where tel=?");
        PreparedStatement statement = null;
        ResultSet resultSet = null;

        try {
            //创建blob对象便于传入数据库
            Blob blob = connection.createBlob();
            blob.setBytes(1,bytes);
            //写入数据
            statement = connection.prepareStatement(sql);
            statement.setBlob(1,blob);
            statement.setString(2, tel);
            //num用于接收结果
            num = statement.executeUpdate();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            JDBCTools.release(connection, statement, resultSet);
        }
        return num;
    }
```

中间还附带着是否存入成功，及其他条件

2.读取数据库图片显示到页面

图片src填入servlet地址

```
<img class="UserCover" src="/test">
```

servlet使用输出流

```
@WebServlet("/test")
public class test extends HttpServlet {
    public static byte[] read() {
        Connection connection = JDBCTools.getConnection();
        String sql = ("select usercover from person where tel='15908629246'");
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        byte[] bytes = null;


        try {
            statement = connection.prepareStatement(sql);
            resultSet = statement.executeQuery();
            //创建blob接受resultset得到的blob数据
            while (resultSet.next()) {
                Blob blob = resultSet.getBlob("usercover");
                bytes = blob.getBytes(1, (int)blob.length());


            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            JDBCTools.release(connection, statement, resultSet);
        }

        return bytes;
    }


    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        test test = new test();
        byte[] b = test.read();
        OutputStream outputStream = resp.getOutputStream();
        outputStream.write(b);
        outputStream.flush();
    }
}
```

3.实现了更改头像和背景图片的功能

注意，调试了真的很久很久，两个图片src需要分开获取，创建两个路径，比较复杂，然后一直405/500的报错，很他妈的烦，有一次错误原因竟然是浏览器没重启，已重启图片就显示成功了，这个更操蛋，我真的想爆粗口，太烦了，本来昨天已经实现的功能，今天完善一下，结果问题频出。

然后整个的逻辑是这样，页面打开就从数据库获取图片，如果要更改，就更新数据库，将新的图片写入数据库，然后浏览器刷新，显示新的图片。

2020/7/22 第五天

1. 用户信息表单

通过<%=>表达式传递java变量，使用session的utel作为tel传入，然后构造一个获取person类的方法(不需要使用servlet)

```
<%
    String tel = (String)session.getAttribute("utel");
    //如果号码为空，一样重新登陆
    if(tel == null){
        response.sendRedirect("Login.jsp");
    }
    //定义一个类获取person，再用这个类去获取数据库中的信息
    Person person = new Person();
    GetPersonInfo getPersonInfo = new GetPersonInfo();
    person = getPersonInfo.getPerson(tel);
    String sex = person.getSex();
    String intro = person.getIntro();
    String work = person.getWork();
    String location = person.getLocation();
%>
```

2. person类

```
public class Person {
    private String tel;
    private String sex;
    private String intro;
    private String work;
    private String location;
```

3.getpersoninfo

```
public class GetPersonInfo {
    public static Person getPerson(String tel){
        Person person = new Person();
        HomePageServiceImpl homePageService = new HomePageServiceImpl();
        person = homePageService.GetHomePageInfo(tel);
        return person;
    }
}
```

传递的过程可以省略，直接看数据库

```
 @Override
    public Person GetPersonInfo(String tel) {
        Connection connection = JDBCTools.getConnection();
        String sql = ("SELECT * from person where tel = ?");
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        Person person = null;

        try {
            statement = connection.prepareStatement(sql);

            statement.setString(1,tel);
            resultSet = statement.executeQuery();
            while (resultSet.next()){
                person = new Person();
                person.setSex(resultSet.getString("sex"));
                person.setWork(resultSet.getString("work"));
                person.setLocation(resultSet.getString("location"));
                person.setIntro(resultSet.getString("intro"));
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        } finally {
            JDBCTools.release(connection, statement, resultSet);
        }
        return person;
    }
```

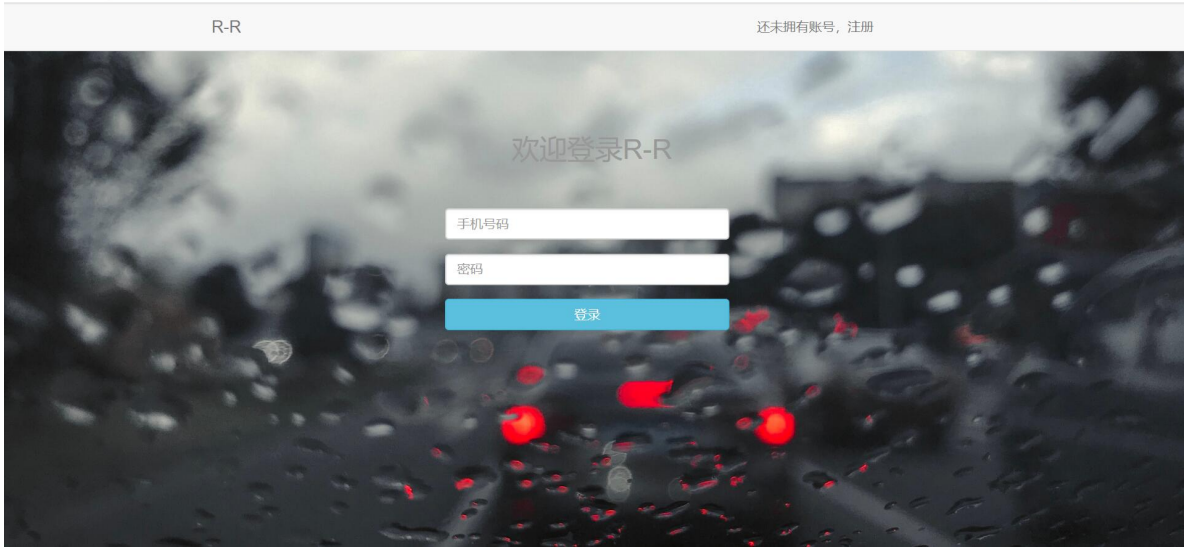
通过数据库的信息，构造一个person类，逐渐传递出去

4.数据库

 12313

	tel	usercover	userportrait	sex	int
1	15908629246	1000x240 JPEG image 95.83 kB	720x720 JPEG image 54.39 kB	男	快乐
2	15925688557	<null>	<null>	<null>	<null
3	18928524004	1000x240 JPEG image 95.83 kB	252.31 kB (204.8 kB loaded)⏎000000…	拉克丝	我的扌

5.登陆页面(给body添加背景)



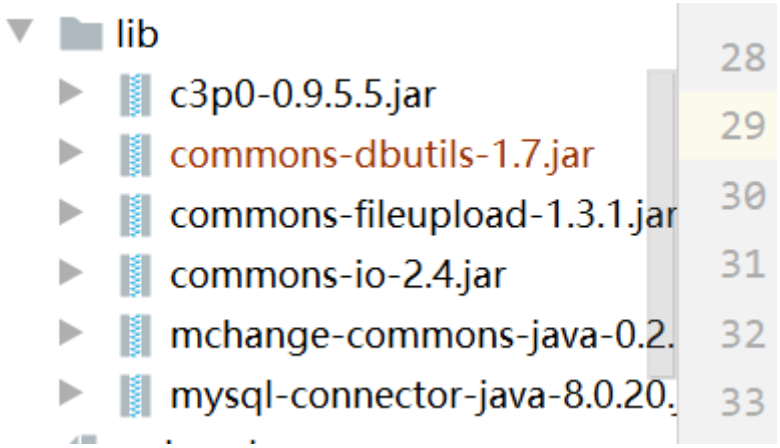
6. 个人信息界面

含有背景图、头像、个人信息，都是通过数据库存放，数据库访问直接显示的，如果提交更改，页面会刷新，然后显示新的更改过后的信息

2020/7/26以及7/27 重新完善项目

随着jsp学习的深入，对项目有了优化想法

1.使用apache的dbutils库简化数据库操作，有效的降低了数十行左右代码量，同时增加了多行注释，增加可阅读性



例如person的repository中

```
public class PersonRepositoryImpl implements PersonRepository {

    static DataSource dataSource = new ComboPooledDataSource("testc3p0");

    /**
     * 读取用户背景图片
     * 使用dbutils简化操作，用简短的语句实现对数据库的增删改查
     * 同时如果没有自主创建的话，dbutils会自动关闭conn等资源，无需手动释放
     * @return 返回数据库图片二进制数组
     */
    @Override
    public byte[] readCover(String tel) {
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("select usercover from person where tel=?");
        Object[] params = {tel};
        byte[] bytes = null;
        try {
            bytes = queryRunner.query(sql, new ScalarHandler<>(), params);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return bytes;
    }

    /**
     * 读取用户头像
     *
     * @return 返回用户头像二进制数组
     */
    @Override
    public byte[] readPor(String tel) {
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("select userportrait from person where tel=?");
        Object[] params = {tel};
        byte[] bytes = null;
        try {
            bytes = queryRunner.query(sql, new ScalarHandler<>(), params);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return bytes;
    }
```

```
    /**
     * 上传用户个人信息
     *
     * @param person 一个person类
     * @return The number of rows updated.
     */
    @Override
    public int UploadPersonInfo(Person person) {
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("UPDATE person set sex = ?,work = ?,location = ?,intro = ?
where tel=?");
        //初始化一个num接收返回值
        int num = 0;
        try {
            num = queryRunner.update(sql, person.getSex(), person.getWork(),
person.getLocation(), person.getIntro(), person.getTel());
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return num;
    }

    /**
     * 获取用户个人信息
     *
     * @param tel 电话号码
     * @return 返回一个Person对象，使用dbutils的BeanHandler
     */
    @Override
    public Person GetPersonInfo(String tel) {
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("SELECT * from person where tel = ?");
        Person person = null;
        try {
            person = queryRunner.query(sql, new BeanHandler<Person>
(Person.class), tel);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return person;
    }

    /**
     * 更改用户背景
     *
     * @param bytes 背景的byte数组
     * @param tel   用户电话
     * @return The number of rows updated.
     */
    @Override
    public int UploadUserCover(byte[] bytes, String tel) {
        //通过num判断是否更改成功
        int num = 0;
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("update person set usercover = ? where tel=?");
        try {
            num = queryRunner.update(sql, bytes, tel);
```

```
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return num;
    }


    /**
     * 更改用户头像
     *
     * @param bytes 头像的二进制数组
     * @param tel   电话
     * @return The number of rows updated.
     */
    @Override
    public int UploadUserPor(byte[] bytes, String tel) {
        //通过num判断是否更改成功
        int num = 0;
        QueryRunner queryRunner = new QueryRunner(dataSource);
        String sql = ("update person set userportrait = ? where tel = ?");

        try {
            num = queryRunner.update(sql, bytes, tel);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return num;
    }

}
```

并调整了两个数据库的结构

2. 编写过滤器

编写了一个身份过滤器，可以拦截未登录访问主页的用户，并有不错的**反馈信息**，注意，反馈信息也十分重要，让开发者更了解代码的运行状况

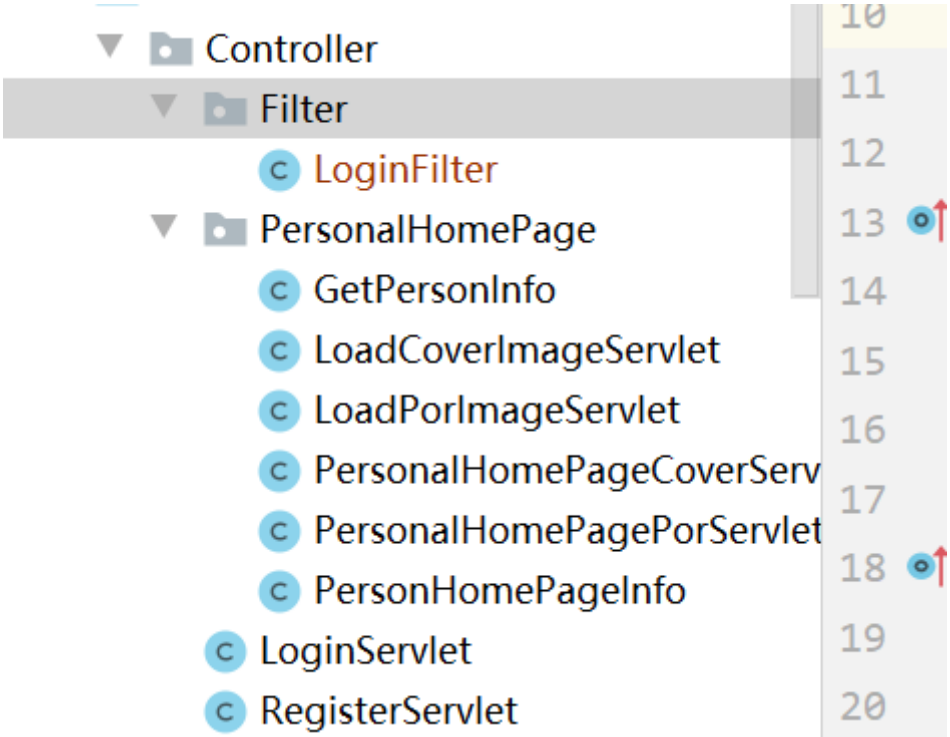
```
@WebFilter(filterName = "Authentication Filters",urlPatterns = "*.jsp")
public class LoginFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("Authentication Filters init...");
    }

    @Override
    public void destroy() {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException {
        //初始化三个对象
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;
        HttpSession session = request.getSession();
```

```
        String url = request.getRequestURI();
        String tel = (String)session.getAttribute("tel");

        System.out.print(url+"正在访问Authentication Filters");
        // 未登录允许访问注册页面和登陆页面
        if(url.equals("/Login.jsp") || url.equals("/Register.jsp")){
            System.out.print("||允许访问"+"\n");
            filterChain.doFilter(request,response);
        } else{
            if(tel == null){
                System.out.print("||已拦截"+"\n");
                response.sendRedirect("/Login.jsp");
            }else{
                System.out.print("||允许访问"+"\n");
                filterChain.doFilter(request,response);
            }
        }

    }
}
```

拦截信息

```
/正在访问Authentication Filters||已拦截
/Login.jsp正在访问Authentication Filters||允许访问
/正在访问Authentication Filters||已拦截
/Login.jsp正在访问Authentication Filters||允许访问
/正在访问Authentication Filters||已拦截
/Login.jsp正在访问Authentication Filters||允许访问
/Register.jsp正在访问Authentication Filters||允许访问
```

3.调整controller结构，增加package，管理更方便



4. 使用jquery优化代码，使之可以弹出提示窗口，并可以用ajax实现局部刷新

```
    <script type="text/javascript" src="assets/js/jquery-3.4.1.min.js"></script>
<%--    控制登录--%>
    <script type="text/javascript">
        function   login() {
            var $tel = $("#inputPhone").val();
            var $password = $("#inputPassword").val();
            $.ajax({
                url: "/login",
                type: "post",
                data: "tel=" + $tel + "&password=" + $password,
                dataType: "JSON",
                success: function (result) {
                    if (result === 1) {
                        alert("登陆成功~");
                        // 跳转到主页
                        $(location).attr("href","HomePage.jsp");
                    } else if (result === 0) {
                        alert("登陆失败！");
                    }
                }
            });
        }
    </script>
```

这种方式比表单提交更具灵活性

但页面跳转需要在jquery中书写，这点详细可以见另一个markdown或博客。

5.熟练的使用Gitee即码云，以及在IDEA中更新项目

先commit再push

2020/7/28 2020/7/29 图片的局部刷新，验证码

1.图片局部刷新

图片的局部刷新使用jquery和ajax

局部刷新的核心方法，使用jquery的更改属性方法，以及强制刷新，即访问servlet的时候添加时间戳，这里有一点，**src**是直接提供请求访问servlet的，不需要手动post。

```
$("#portrait-img").attr("src","/loadporimage?t="+(new Date().getTime()));
```

更换头像的代码

需要掌握jquery和ajax的一部分方法

```
function submitpor() {
        var formdata = new FormData($("#uploadpor")[0]); //创建一个fordata
        formdata.append("img_por", $("#UserPortrait")[0].files[1]); //把file添加
进去，name命名为img
        $.ajax({
            url: "/homepagepor",
            type: "post",
            data: formdata,
            //一些配置属性，具体用途不知，但有必要，并且result的类型变化了
            async: false,
            cache: false,
            contentType: false,
            processData: false,
            success: function (result) {
                if (result === "1") {
                    alert("修改个人头像成功~")
                    //局部刷新图片
                    $("#portrait-img").attr("src","/loadporimage?t="+(new
Date().getTime()));
                } else if (result === "0") {
                    alert("修改个人头像失败！");
                }
            }
        });
    }
```

我们再看看servlet的返回值

```
                    String tel = (String)httpSession.getAttribute("utel");
                    if(homePageService.UploadUserPortrait(array,tel)){
                        System.out.println("头像上传成功");
                        //上传成功返回1
                        printWriter.write("1");
                    } else {
                        System.out.println("头像上传失败");
                        //上传失败返回0
                        printWriter.write("0");
                    }
```

然后在ajax中用result接收这个值即可

2.验证码

验证码采用jsp的格式，传递给图片的src，然后并把验证码真实内容提交session，以便servlet能够将输入内容和验证码真实内容进行比较，验证码输入错误则提示，并刷新页面

```
<%@ page import="java.awt.*" %>
<%@ page import="java.util.Random" %>
<%@ page import="java.awt.image.BufferedImage" %>
<%@ page import="javax.imageio.ImageIO" %>
```

```
<%@ page contentType="image/jpeg;charset=UTF-8" language="java" %>
<%!
    /**
     * 产生随机颜色的方法
     * @return 返回一个颜色
     */
    public Color getColor() {
        Random ran = new Random();
        int r = ran.nextInt(256);
        int g = ran.nextInt(256);
        int b = ran.nextInt(256);
        return new Color(r, g, b);
    }

    /**
     * 产生四位随机数
     * @return 返回随机数的String
     */
    public String getNum() {
        int ran = (int) (Math.random() * 9000) + 1000;
        return String.valueOf(ran);
    }
%>

<%
    //禁止缓存，放置验证码过期
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Expires", "0");

    //参数是宽、高、类型
    BufferedImage image = new BufferedImage(80, 30, BufferedImage.TYPE_INT_RGB);

    //画笔
    Graphics graphics = image.getGraphics();
    graphics.fillRect(0, 0, 80, 30);

    //绘制干扰线条
    for (int i = 0; i < 60; i++) {
        Random ran = new Random();
        //线条位置
        int xBegin = ran.nextInt(80);
        int yBegin = ran.nextInt(30);
        int xEnd = ran.nextInt(xBegin + 10);
        int yEnd = ran.nextInt(yBegin + 10);

        graphics.setColor(getColor());
        //绘制线条
        graphics.drawLine(xBegin,yBegin,xEnd,yEnd);
    }

    graphics.setFont(new Font("seif",Font.BOLD,20));
    //绘制验证码,黑色
    graphics.setColor(Color.BLACK);
    String checkCode = getNum();
    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < 4; i++) {
        sb.append(checkCode.charAt(i)+" "); //验证码的每一位数字
```

```
    }

    //绘制验证码
    graphics.drawString(sb.toString(),15,20);

    //验证码真实值，供使用时比较
    session.setAttribute("CHECKCODE",checkCode);

    //真实的产生图片
    ImageIO.write(image,"jpeg",response.getOutputStream());

    //关闭
    out.clear();
    out = pageContext.pushBody(); //<inpput type="image" src="XXX" />
%>
```

同样的，验证码也采取局部刷新的方式，且自动提交吗，同时图片用超链接包裹，点击即可更改属性，超链接的href是javascript的一个function。

```
<a href="javascript:reloadCheckImg();"><img id="VertificationImage"
src="VertificationImage.jsp"/></a>
```

```
        function reloadCheckImg(){
            $("#VertificationImage").attr("src","VertificationImage.jsp?t="+(new
Date().getTime())); //把src给新的值
        }

        $(document).ready(function (){
            $("#checkcodeId").blur(function (){
                var $checkcode = $("#checkcodeId").val();
                //文本框输入的值发送到服务端
                //服务端获取输入的值和真实的验证码对比，并返回验证结果
                $.post(
                    "CheckCodeServlet",
                    "checkcode="+$checkcode,
                    function (result){
                        alert(result);
                        if (result === "验证码输入错误！"){
                            $(location).attr("href","Login.jsp");
                        }
                    }
                )
            })
        })
```