3.1登录及个人信息管理模块

**3.1.1项目文件结构**

以登录模块为例

login

src

main

java

com.login

controller

LoginController.java

ViewController.java

pojo

User.java

utils

LoginCacheUtil

LoginApp.java

resources

templates

login.html

application.yml

application.yml 为配置文件

LoginApp.java 为启动类

User.java 定义用户实体

LoginCacheUtil.java 是登录缓存类

LoginController.java 进行登录功能逻辑的实现

ViewContrtoller.java 实现页面跳转逻辑

login.html 为页面设计文件

templates文件夹下保存页面文件——用户界面层

ViewController控制页面跳转和响应信息——控制层

LoginController用户登录的业务逻辑层的Controller——业务逻辑层

User.java 保存用户实体类及表单信息——数据持久层

**3.1.2模块设计类图**

**登录模块**

1.LoginApp

@SpringBootApplication 注解表明这是一个SpringBoot启动类

方法：

SpringApplication.run(LoginApp.class, args);

2.ViewController



控制页面跳转的控制器，toLogin表示跳转到login页面

方法：

public String toLogin(String target, Http session, @CookieValuie(“TOKEN”) Cookie cookie);

根据cookie信息判断是否已经登录，若已登录则不会跳转到登录页面，而会根据session传过来的重定向地址target跳转到重定向地址

3.Login.html



4.User



参数：

int user\_id

Varchar(32) user\_name

Varchar(32) user\_password

方法：

使用@Data 添加getter/setter 方法

使用@NoArgsConstructor 添加无参构造器

使用@AllArgsConstructor 添加全餐构造器

使用@Accessors(chain = true) 添加链式调用便于new

5.LoginController



参数：

HttpSession session服务端存储客户登录响应信息，如重定向地址，错误信息

User user 提供用户登录信息表单

Cookie cookie 将登录信息token保存到cookie中实现单点登录

interface userMapper 提供与数据库的操作接口

方法：

public String doLogin(User user, HttpSession session， HttpServletResponse response)

通过登录的用户名和密码去查找数据库中的用户，先判断用户是否已经登录，（通过cookie中的token判断是否登录）再判断用户密码是否正确，若登录失败则通过session返回错误信息，若登录成功则将用户登录信息保存到LoginCacheUtil中，返回redirect+target

6.LoginCacheUtil



用于保存用户登录信息，token保存于此

参数：

public static Map<String, User> loginUser = new HashMap<>();

方法：

public void put（token） 放入token

public String get（token） 取出token

**注册模块**

1.RegistApp

@SpringBootApplication 注解表明这是一个SpringBoot启动类

方法：

SpringApplication.run(RegistApp.class, args);

2.ViewController



控制页面跳转的控制器，toRegist表示跳转到注册页面

方法：

public String toRegist(String target, Httpsession session);

根据session传过来的注册结果信息，若注册成功则会重定向到target登录地址

3.regist.html

4.User



参数：

int user\_id

Varchar(32) user\_name

Varchar(32) user\_password

方法：

使用@Data 添加getter/setter 方法

使用@NoArgsConstructor 添加无参构造器

使用@AllArgsConstructor 添加全餐构造器

使用@Accessors(chain = true) 添加链式调用便于new

5.RegistController



参数：

HttpSession session服务端存储客户注册响应信息，如重定向地址，错误信息

User user 提供用户注册信息表单

interface userMapper 提供与数据库的操作接口

方法：

public String doRegist(User user, HttpSession session， HttpServletResponse response)

获取用户输入信息保存到user实体类，通过service层进行信息验证，方便后续操作。与数据库接口判断用户是否已经注册，若已注册session存入已注册msg错误信息并显示在页面上。若未注册则执行注册并将用户信息保存到数据库并返回target为login即返回登录页面

**找回密码模块**

1.FindPassApp



@SpringBootApplication 注解表明这是一个SpringBoot启动类

方法：

SpringApplication.run(FindPassApp.class, args);

2.ViewController



控制页面跳转的控制器，toFindPass表示跳转到找回密码页面

方法：

public String toFindPass(String target, Httpsession session);

根据session传过来的找回密码结果信息，若重置密码成功则会重定向到target登录地址

3.FindPass.html



4.User



参数：

int user\_id

Varchar(32) user\_name

Varchar(32) user\_password

方法：

使用@Data 添加getter/setter 方法

使用@NoArgsConstructor 添加无参构造器

使用@AllArgsConstructor 添加全餐构造器

使用@Accessors(chain = true) 添加链式调用便于new

5.FindPassController



参数：

HttpSession session服务端存储客户操作响应信息，如重定向地址，错误信息

User user 用户实体类，保存用户信息便于后续操作

interface userMapper 提供与数据库的操作接口

方法：

public String doFindPass(User user, HttpSession session， HttpServletResponse response)

获取用户输入信息保存到user实体类，通过service层进行信息验证。与数据库接口判断用户是否已经存在，若用户不存在则session存入用户不存在msg错误信息并显示在页面上。若用户存在则重置密码为用户输入的新密码并将用户信息保存到数据库并返回target为login即返回登录页面

**修改个人信息模块**



1.InfoApp



@SpringBootApplication 注解表明这是一个SpringBoot启动类

方法：

SpringApplication.run(InfoApp.class, args); 启动方法

public RestTemplate restTemplate(); 发HTTP请求用户数据判断token

2.ViewController



控制页面跳转，但由于个人信息模块是子系统，需要通过cookie判断是否已经登录，若未登录会跳转到登录页面

参数：

Httpsession session 服务端用户登录信息以及响应信息

Cookie cookie 用户客户端保存的登录信息，保存token

RestTemplate restTemplate 发送HTTP请求，请求用户数据判断token的值是否为空

String USER\_INFO\_ADDRESS = <http://login.xxx.com:/login/info?token=>

方法：

public String toInfo（@CookieValue（required = false, value = “token”）Cookie cookie. HttpSession session）

通过判断cookie的token值以判断用户是否已登录，并通过session来保存响应信息loginUser，并交给页面以决定是否展示登录页面

3.info.html

点击左上角箭头会返回，点击对应个人资料栏右边的箭头可以出现一个输入框输入修改的个人信息

4.UserInfo



参数：

int user\_id 用户id为一自增的主键，不可更改

Varchar(32) user\_name 用户名

Varchar(32) user\_password 用户密码

Varchar(32) user\_email 邮箱

int user\_phone 手机号

int user\_age 年龄

Tinyint user\_gender 性别

String user\_background 既往病史及过敏史

方法：

使用@Data 添加getter/setter 方法

使用@NoArgsConstructor 添加无参构造器

使用@AllArgsConstructor 添加全参构造器

使用@Accessors(chain = true) 添加链式调用便于new

5.InfoController



参数：

HttpSession session服务端存储客户操作响应信息，如跳转target

UserInfo user\_info 用户信息实体类，保存用户个人信息

interface userMapper 提供与数据库的操作接口

方法：

public String correctInfo(UserInfo user\_info, HttpSession session， HttpServletResponse response)

获取用户输入信息保存到UserInfo实体类，通过service层进行信息验证。与数据库接口userMapper修改信息，若修改成功则session存入修改成功提示并显示在页面上并返回target为info即返回个人信息页面

**3.1.3接口设计**



UserMapper即为登录系统功能模块和数据库的接口

public int GetMaxId(); 可以获取总共有多少个用户id

public bool Exists(user\_name); 根据用户名查询用户是否存在

public int Add(); 执行相应的插入操作

public void Update(); 执行更新操作

public void Delete(); 执行删除操作

public User GetList(); 获取完整用户表单

**3.1.4数据库设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 用户个人信息表 | | | |
| 字段名 | 类型 | 说明 | 属性 |
| user\_id | int | 用户id | PK, NOT NULL, IDENTITY |
| user\_name | Varchar(32) | 账号 | NOT NULL |
| user\_password | Varchar(32) | 密码 | NOT NULL |
| user\_email | Varchar(32) | 邮件地址 |  |
| user\_phone | int | 电话号码 | NOT NULL |
| user\_age | int | 年龄 | NOT NULL |
| user\_gender | Tinyint | 性别 | NOT NULL |
| user\_background | Varchar(500) | 既往病史 |  |