

HUNAN UNIVERSITY

**编程新技术实务**

|  |  |  |
| --- | --- | --- |
| **实验四：** | | **系统登录/注册模块(Android app)的开发** |
|  | **小组成员：** | **谢正宇 张继伟** |
|  | **专业班级：** | **计科1802** |
|  | **完成日期：** | **2020年12月28日** |

# 1 引言

## 实验目的

通过使用Android API进行系统注册模块的开发，包括前台的Android原生app以后后台服务模块的开发，要求后台使用JavaEE框架实现。

## 实验对应知识点

Android的应用编程框架以及Android API的使用，JavaEE框架的应用开发。

## 实验环境（开发工具）

实验工具

IntelliJ Idea，腾讯云，JDK，Tomcat，Android SDK，Mysql

## 实验基础

Java编程、Android API的编程、JavaEE中JSP、Servlet的编程。

## 实验内容

一、Android app的开发

对于Android app，本组实现了如下界面：

（1）登录界面：包含用户名、密码的文字标识以及相应的输入栏，登录的按钮。当输入用户名以及密码后，点击登录按钮，则提交数据提交至后台进行验证，如通过验证则跳转至欢迎界面，否则跳转回登录界面，并提示用户的验证错误原因；当用户点击新用户注册文字则跳转至注册界面；当用户点击找回密码文字时则跳转至校验用户密码界面，否则停留在登录界面。

（2）注册界面：包含用户名、密码以及确认密码的文字标识以及相应的输入栏，注册以及返回按钮。当输入相关信息后，首先验证输入的信息是否符合要求（后台校验用户名是否存在；密码为6-12位，只允许包含英文字母、数字和\_，同时要求确认密码必须与密码一致），如不符合要求则在界面内提示错误，只有符合要求才提交给后台进行注册操作。如注册成功则跳转至欢迎界面，否则跳转回注册界面并提示用户的注册错误原因；当用户点击返回按钮则返回登录界面。

（3）校验用户界面：包含用户名文字标识以及相应的输入栏，提交按钮。当输入用户名后，点击提交按钮，则提交数据提交至后台进行验证（即验证需要改密码的用户是否存在），如通过验证则跳转至欢迎界面，否则跳转回登录界面，并提示用户不存在。

（4）忘记密码界面：包含手机号、验证码的文字标识以及相应的输入栏，获取验证码、提交及返回按钮。当输入手机号后，点击获取验证码，弹出相关提示，确认后手机号会收到短信验证码。当验证码正确并且输入的手机号与登录界面输入的用户名相匹配时，点击提交按钮才能跳转至修改密码界面，否则跳转回忘记密码界面并提示错误原因。当用户点击返回按钮则返回登录界面。

（5）修改密码界面：包含密码、确认密码的文字标识以及相应的输入栏，密码为6-12位，只允许包含英文字母、数字和\_，同时要求确认密码必须与密码一致。如不符合要求则在界面内提示错误，只有符合要求才提交给后台进行修改密码操作。如修改成功则跳转至登录界面，否则跳转回修改密码界面并提示出错原因。当用户点击返回按钮则返回登录界面。

（6）欢迎界面：显示对用户的欢迎信息，其中包括用户的用户名、姓名等。用户信息上面会根据用户登录的时间显示“上午好！+用户名”或者“下午好+用户名”或“晚上好”+用户名”

二、后台服务的开发

对于Android app调用的后台服务而言，要求使用JavaEE中的JSP或者Servlet进行编写，响应信息可以使用XML或JSON等方式进行封装，封装的信息由Android app再进行解析处理。

对于注册信息的存储，要求使用mysql数据库进行存放，其中用户名作为表的主键进行存储。

# 2 模块汇总

**2.1 模块汇总表**

**APP端** app

1、UI界面模块(layout)

作为视图层展现给用户

2、活动模块(main/java)

用户操作的执行代码，其中包括客户端网络通信编程

3、属性模块(drawable)

存储数据和属性以及需要调用的图片等文件

**服务端server**

模块名称

1、Lab1

封装数据，并实现增删改查功能

2、service

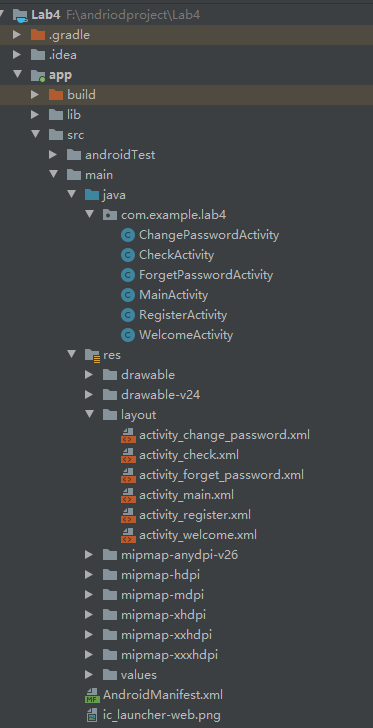
两个接口，一个用于操作users表，一个用于操作perosn表

3、servlet

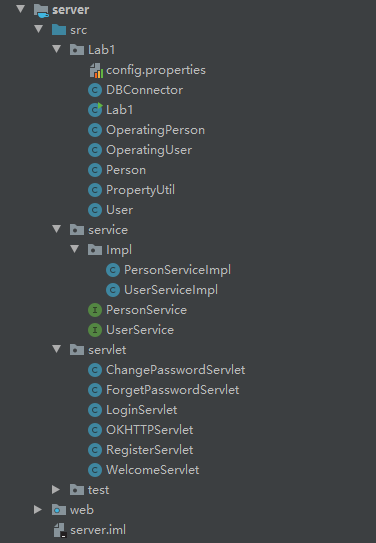
实现前后台的控制

**2.2 模块内容**

APP端：



服务端：



# 3 子系统模块设计

**3.1 客户端——UI界面模块**

|  |  |
| --- | --- |
| **模块名称** | **UI界面模块** |
| **功能描述** | 直接显示在屏幕上的视图,可以在其中放置多个控件并且设置其布局方式使其呈现在用户面前。总共六个界面 |
| **接口与属性** | View和ViewGroup对象创建的用户见面元素是可以和用户互动的，VirwGroup对象是为了定义布局的接口而保存其他的View Android提供一个View和ViewGroup子类的集合，这个集合能提供相同的输入控制（例如按钮和文本框）和各种各样的布局模式（例如一个线性或者相对布局）。 |
| **数据结构**  **与算法** | **在本设计中，包含六个XML文件：**  **一、登录界面（activity\_main.xml）**  包含了两个输入栏（EditText）,分别对应用户名和密码，两个输入栏的左侧都提供了两个文本框（TextView）显示用户名和密码，提示用户输入用户名和密码。输入栏的上方是“首页”的文字信息（TextView）。输入栏的下方是一个登录按钮，当用户输入用户名及密码时点击登录即可。登录按钮下面是找回密码和新用户注册字样，找回密码及新用户注册点击这两个按钮即可即可。点击按钮跳转页面是通过onclick属性绑定相关的函数。  **二、注册界面（activity\_register.xml）**  包含了六个输入栏，输入框左侧都有相应的提示，分别为用户名、姓名、年龄、电话、密码、确认密码。输入栏上方是一行“注册页”的文字。输入栏下方是提交按钮和返回按钮，同样的是利用onclick属性绑定相关的函数。   1. **校验用户界面（activity\_forget\_password.xml）**   一个输入框，对应用户名输入，输入框左侧提供了一个文本框显示“用户名”文字提示。输入框上面是“校验用户名”文字提示。输入框下面是提交按钮。   1. **忘记密码界面（activity\_forget\_password.xml）**   两个输入框，分别对应电话号码和短信验证码，输入框左侧提供了两个文本框显示电话号码和短信验证码，对应电话号码和短信验证码的输入。短信验证码框的右侧提供了一个按钮用来获取验证码，按钮显示“获取验证码”文字提示。输入栏的上方是“短信验证页”的文字提示。输入栏下方是提交和返回按钮。   1. **修改密码界面（activity\_change\_password.xml）**   两个输入框，分别对应新密码的输入和新密码二次输入，输入框左侧提供了两个文本框显示新密码和确认密码。输入栏的上方是“修改密码页”的文字提示。输入栏下方是提交和返回按钮。  **六、欢迎界面（activity\_welcome.xml）**  主要是四文本框，分别对应用户名、姓名、年龄和电话号码的显示（View）。上方给出“Welcome”+用户名的提示文字，welcome会根据当前时间显示“上午好””下午好”“晚上好”，下方是返回按钮。 |
| **补充说明** | 六个UI界面都采用线性布局。 |

**3.2 客户端——活动模块**

|  |  |
| --- | --- |
| 模块名称 | 活动模块（Activity） |
| 功能描述 | 实现Android的客户端与服务端的通信功能。包括异步线程处理，访问服务端，用http实现客户端和服务器端的通信等。 |
| 接口与属性 | Android目前提供两种http通信接口：  标准Java接口(java.net) ----HttpURLConnection，可以实现简单的基于URL请求、响应功能；  Apache接口(org.appache.http)----HttpClient，相对更大更全能，但是速度相对较慢。  几乎在任何项目的代码中我们都能看到这两个类的身影，使用率非常高。不过HttpURLConnection和HttpClient的用法还是稍微有些复杂的，如果不进行适当封装的话，很容易就会写出不少重复代码。于是乎，一些Android网络通信框架也就应运而生，比如AsyncHttpClient、Volley等。本次实验采用的是Volley框架。 |
| 数据结构  与算法 | HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。  HTTP协议的主要特点：  1.支持C/S（客户/服务器）模式。  2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST，每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。  3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。  4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。  5.无状态：HTTP协议是无状态协议，无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。  HTTP URL 的格式如下：  http://host[":"port][abs\_path]  http表示要通过HTTP协议来定位网络资源；host表示合法的Internet主机域名或者IP地址；port指定一个端口号，为空则使用默认端口80；abs\_path指定请求资源的URI（Web上任意的可用资源）。  HTTP有两种报文分别是请求报文和响应报文，让我们先来看看请求报文。  HTTP的请求报文：  先来看看请求报文的一般格式：    通常来说一个HTTP请求报文由请求行、请求报头、空行、和请求数据4个部分组成。  请求行：  请求行由请求方法，URL字段和HTTP协议的版本组成，格式如下：  Method Request-URI HTTP-Version CRLF  其中 Method表示请求方法；Request-URI是一个统一资源标识符；HTTP-Version表示请求的HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。  HTTP请求方法有8种：  分别是GET、POST、DELETE、PUT、HEAD、TRACE、CONNECT 、OPTIONS。其中PUT、DELETE、POST、GET分别对应着增删改查，对于移动开发最常用的就是POST和GET了。  GET：请求获取Request-URI所标识的资源  POST：在Request-URI所标识的资源后附加新的数据  HEAD：请求获取由Request-URI所标识的资源的响应消息报头  PUT： 请求服务器存储一个资源，并用Request-URI作为其标识  DELETE ：请求服务器删除Request-URI所标识的资源  TRACE ： 请求服务器回送收到的请求信息，主要用于测试或诊断  CONNECT： HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。  OPTIONS ：请求查询服务器的性能，或者查询与资源相关的选项和需求  例如我去访问百度请求行是：  GET http://www.baidu.com HTTP/1.1  请求报头：  在请求行之后会有0个或者多个请求报头，每个请求报头都包含一个名字和一个值，它们之间用“：”分割。请求头部会以一个空行，发送回车符和换行符，通知服务器以下不会有请求头。关于请求报头，会在后面的消息报头一节做统一的解释。  请求数据：  请求数据不在GET方法中使用，而是在POST方法中使用。POST方法适用于需要客户填写表单的场合，与请求数据相关的最常用的请求头是Content-Type和Content-Length。  具体方法：  主要使用到两种http请求方法：Get与Post方法。  Post请求可以向服务器传送数据，而且数据放在HTML HEADER内一起传送到服务端URL地址，数据对用户不可见。而Get是把参数数据队列加到提交的URL中，值和表单内各个字段一一对应。  Get方式：Get机制用的是在URL地址里面通过？号间隔，然后以name=value的形式给客户端传递参数。所以首先要在Android工程下的onCreate方法定义好其URL地址以及要传递的参数，然后通过URL打开一个HttpURLConnection链接，此链接可以获得InputStream字节流对象，也是往服务端输出和从服务端返回数据的重要过程，而若服务端往andorid返回信息时候，就可以通过InputStreamReader作转换，将返回来的数据用BufferReader显示出来。  Post方式：Post传输方式不在URL里传递，也解决了get传输量小、容易篡改及不安全等一系列不足。主要是通过对HttpURLConnection的设置，让其支持post传输方式，然后在通过相关属性传递参数（若需要传递中文字符，则可以通过URLEncoder编码，而在获取端采用URLDecoder解码即可）  以ChangePasswordActivity为例：  请求地址为：  String url =  "http://47.100.39.146:8080/server/changePasswordServlet";  其中47.100.39.146是我们购买的阿里云轻量级应用服务器的外网，8080是Tomcat的端口，server是JavaEE后台的项目名，  changePasswordServlet是后台处理的Servlet。  取得请求队列：  RequestQueue requestQueue =  Volley.newRequestQueue(getApplicationContext());  创建StringRequest，定义字符串请求的请求方式为POST(省略第一个参数会默认为GET方式)：  final StringRequest request =  new StringRequest(Request.Method.POST, url,  new Response.Listener<String>() {...}, new Response.ErrorListener(){...})  传输参数是通过重写getParams()函数实现的：  @Override  protected Map<String, String> getParams() throws AuthFailureError {  Map<String, String> params = new HashMap<>();  params.put("username", username);  params.put("password", password);  return params;  }  如上代码，传输了两个参数，一个为username，另一个为password。  将请求添加到队列中  requestQueue.add(request);  至此，客户端向服务端发送数据的介绍告一段落。  在Activity里还需实现的是实现布局文件（.xml）里onclick绑定的函数。  普通的页面跳转的一个函数如下：  public void BackToHome (View view){  Intent intent =  new Intent(ChangePasswordActivity.this, MainActivity.class);  startActivity(intent);  }  若想在页面之间实现信息的传递，可利用  intent.putExtra("username",username);和  username = getIntent().getStringExtra("username");来传递。  还需实现对用户输入数据的合法性检测，可利用正则表达式判断。 |

**3.3客户端——属性模块**

|  |  |
| --- | --- |
| 模块名称 | 属性模块 |
| 功能描述 | R.java文件自动生成，用来定义Android程序中所有各类型的资源的索引。（它是只读的，开发人员不对其修改）。 |
| 接口与属性 | android工程所有资源信息(组件、图片、字符等等)都是由HashMap<Integer,Object>来存储的key值就是R.java中的静态变量值，value就是相对应的各种对象信息(组件、图片、字符等等)。 |
| 数据结构  与算法 | R.java文件中默认有attr、drawable、layout、string等四个静态内部类，每个静态内部类分别对应着一种资源，如layout静态内部类对应layout中的界面文件，其中每个静态内部类中的静态常量分别定义一条资源标识符，如public static final int main=0x7f030000;对应的是layout目录下的main.xml文件。 |
| 补充说明 | R.java及本地资源文件，使用“@+”声明的资源，系统会自动在R.java中创建。 |

**3.4 服务端——Lab1**

|  |  |
| --- | --- |
| 模块名称 | Lab1 |
| 功能描述 | 在实验一的代码上补充一些功能或调整来适应实验四的需求，包含    config.properties配置文件，以及六个实现user表和person表增删改查功能的文件。 |
| 接口与属性 | 实现users表和perosn表的增删改查 |
| 数据结构  与算法 | //OperatingPerson  package Lab1; import java.sql.SQLException; public class OperatingUser{  public void addUser(User u,DBConnector db) throws Exception {  String sql = "insert into users values('"+u.getUsername()+"','" +u.getPassword() +"');";  db.executeUpdateInsert("users",sql,u.getUsername());  }  public void deleteUser(String username,DBConnector db) throws Exception{  //String sql = "insert into person values('"+p.getUsername()+"','"+p.getName()+"',"+p.getAge()+",'"+p.getTeleno()+"');";  db.executeUpdateDelete("users",username);  }  public void clearUsers(DBConnector db){  db.clear("users");  }  public boolean findUser(String username,DBConnector db) throws SQLException {  return db.find("users",username);  }  public void updateUser(User u,DBConnector db)throws SQLException{  db.update("users",u);  }  public int checkUserPassword(User u,DBConnector db) throws SQLException {  return db.checkPassword(u);  } }  //operatingUser  package Lab1;  import java.sql.SQLException;  public class OperatingPerson {  public void addPerson(Person p, DBConnector db) throws Exception{  String sql = "insert into person values('"+p.getUsername()+"','"+p.getName()+"',"+p.getAge()+",'"+p.getTeleno()+"');";  db.executeUpdateInsert("person",sql,p.getUsername());  }  public void deletePerson(String username, DBConnector db) throws Exception{  //String sql = "insert into person values('"+p.getUsername()+"','"+p.getName()+"',"+p.getAge()+",'"+p.getTeleno()+"');";  db.executeUpdateDelete("person",username);  }  public void clearPerson(DBConnector db){  db.clear("person");  }  public boolean findPerson(String username, DBConnector db) throws SQLException {  return db.find("person",username);  }  public void updatePerson(String[] s, DBConnector db)throws SQLException{  //db.update("person",s);  }  public int checkTelephone(Person p,DBConnector db)throws SQLException{  return db.checkTelephoneNumber(p);  }  public Person getPerson(String username,DBConnector db)throws SQLException{  return db.getPerson(username);  } } |

**3.6服务端——service**

|  |  |
| --- | --- |
| 模块名称 | service |
| 功能描述 | 为servlet模块根据需求操作users表和person表提供接口 |
| 接口与属性 | 两个接口，一个操作users表，一个操作persons表 |
| 数据结构  与算法 | package service.Impl; import Lab1.DBConnector; import Lab1.OperatingPerson; import Lab1.Person; import service.PersonService;  public class PersonServiceImpl implements PersonService {  OperatingPerson opP=new OperatingPerson();  @Override  public boolean addPerson(Person p, DBConnector dButil) {  boolean isok = true;  try{  opP.addPerson(p,dButil);  }catch (Exception e){  e.printStackTrace();  }  return isok;  }   @Override  public boolean isCorrectTelenum(String username, String telenum, DBConnector dButil) {  boolean isCorrect = false;  try{  if(opP.checkTelephone(new Person(username,"",0,telenum),dButil)==1){  isCorrect=true;  }  }catch (Exception e){  e.printStackTrace();  }  return isCorrect;  }   @Override  public Person getPerson(String username, DBConnector dButil) {  Person p=new Person(username,"");  try {  p=opP.getPerson(username, dButil);  }catch (Exception e){  e.printStackTrace();  }  return p;  }   @Override  public boolean isRegistered(String username, DBConnector dButil) {  boolean isRegister=false;  try{  isRegister=opP.findPerson(username,dButil);  }catch (Exception e){  e.printStackTrace();  }  return isRegister;  } }  package service.Impl;  import Lab1.DBConnector; import Lab1.OperatingUser; import Lab1.User; import service.UserService;  import java.sql.SQLException;  public class UserServiceImpl implements UserService {  OperatingUser opU=new OperatingUser();  @Override  public int verifyLogin(User u, DBConnector dButil)throws SQLException {  boolean zero=false;  boolean one=false;  try {  if(opU.checkUserPassword(u, dButil)==0){  zero=true;  }  else if(opU.checkUserPassword(u, dButil)==1){  one=true;  }  } catch (SQLException e) {  e.printStackTrace();  }  if(zero) return 0;  else if(one) return 1;  else return -1;  }   @Override  public boolean addUser(User u, DBConnector dButil) {  boolean isok = false;  try{  opU.addUser(u,dButil);  }catch (Exception e){  e.printStackTrace();  }  try{  if(opU.checkUserPassword(u, dButil)==1)  isok=true;  }catch (SQLException e){  e.printStackTrace();  }  return isok;  }   @Override  public boolean changePassword(User u, DBConnector dButil){  boolean isok = false;  try{  opU.updateUser(u,dButil);  }catch (Exception e){  e.printStackTrace();  }  try{  if(opU.checkUserPassword(u, dButil)==1)  isok=true;  }catch (SQLException e){  e.printStackTrace();  }  return isok;  }   @Override  public boolean isRegistered(String username, DBConnector dButil) {  boolean isRegister=false;  try{  isRegister=opU.findUser(username,dButil);  }catch (Exception e){  e.printStackTrace();  }  return isRegister;  } } |

**3.6服务端——servlet**

|  |  |
| --- | --- |
| 模块名称 | servlet |
| 功能描述 | 这里主要是服务端控制客户端的程序，具体体现为LoginServlet、RegisterServlet、ForgetPasswordServlet、ChangePasswordServlet和WelcomeServlet，OKHTTPServlet，请求内容通过servlet返回给客户端，实现客户端和服务器的通信，注意这里需要访问数据库中的数据。 |
| 接口与属性 | 所有Servlet应用必须直接或者间接实现Servlet接口, Servlet容器会将实现了Servlet接口的类加载至容器，以供访问。 |
| 数据结构  与算法 | 以LoginServlet为例：  package servlet;  import Lab1.DBConnector; import Lab1.Person; import Lab1.User; import net.sf.json.JSONObject; import service.Impl.UserServiceImpl; import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.io.PrintWriter; import java.sql.SQLException; import java.util.HashMap; import java.util.Map;  @WebServlet(name = "LoginServlet",urlPatterns = "/loginServlet") public class LoginServlet extends HttpServlet {  DBConnector dButil=new DBConnector();  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //设置相应内容类型  response.setContentType("text/html;charset=utf-8");  request.setCharacterEncoding("utf-8");  response.setCharacterEncoding("utf-8");  try(PrintWriter out = response.getWriter()){  String username = request.getParameter("username").trim();  String password = request.getParameter("password").trim();  //System.out.println(username);  UserServiceImpl service = new UserServiceImpl();  int verigyResult = service.verifyLogin(new User(username,password),dButil);   Map<String,String> params = new HashMap<>();  JSONObject jsonObject = new JSONObject();   if(verigyResult == -1){  params.put("Result","TheUserDoesNotExist");  }else if(verigyResult == 0){  params.put("Result","PasswordError");  }else if(verigyResult == 1){  params.put("Result","CorrectPassword");  }  jsonObject.put("params",params);  out.write(jsonObject.toString());   } catch (SQLException e) {  e.printStackTrace();  }  }   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doPost(request,response);  } } 1、设置相应内容类型 2、获取参数 3、调用service层函数，完成相关操作，得到返回结果 4、将结果封装到JSONObject中，返回 |

# 4 APP各界面展示

**登录界面：**



**注册界面：**



**校验用户页**



**忘记密码界面：**



**修改密码界面：**



**欢迎界面：**

