package com.yawa.util;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.jfinal.render.Render;

public class CaptchaRender extends Render{

private static Random random = new Random();

private static String randString = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";// 随机产生的字符串

private int width = 80;

private int height = 30;

private int lineSize = 40;

private int stringNum = 4;

public CaptchaRender(){}

public CaptchaRender(int width,int height,int lineSize,int stringNum){

this.width = width;

this.height = height;

this.lineSize = lineSize;

this.stringNum = stringNum;

}

/\*

\* 获得字体

\*/

private static Font getFont() {

return new Font("Fixedsys", Font.CENTER\_BASELINE, 18);

}

/\*

\* 获得颜色

\*/

private static Color getRandColor(int fc, int bc) {

if (fc > 255)

fc = 255;

if (bc > 255)

bc = 255;

int r = fc + random.nextInt(bc - fc - 16);

int g = fc + random.nextInt(bc - fc - 14);

int b = fc + random.nextInt(bc - fc - 18);

return new Color(r, g, b);

}

/\*\*

\* 生成随机图片

\*/

public void getRandcode(HttpServletRequest request,

HttpServletResponse response) {

BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE\_INT\_BGR);

Graphics g = image.getGraphics();// 产生Image对象的Graphics对象,改对象可以在图像上进行各种绘制操作

g.fillRect(0, 0, width, height);

g.setFont(new Font("Times New Roman", Font.ROMAN\_BASELINE, 18));

g.setColor(getRandColor(110, 133));

for (int i = 0; i <= lineSize; i++) drowLine(g);

String randomString = "";

for (int i = 1; i <= stringNum; i++) randomString = drowString(g, randomString, i);

SecurityContextUtil.setCaptcha(request, randomString);

g.dispose();

try {

ImageIO.write(image, "JPEG", response.getOutputStream());// 将内存中的图片通过流动形式输出到客户端

} catch (Exception e) {

e.printStackTrace();

}

}

/\*

\* 绘制字符串

\*/

private static String drowString(Graphics g, String randomString, int i) {

g.setFont(getFont());

g.setColor(new Color(random.nextInt(101), random.nextInt(111), random.nextInt(121)));

String rand = String.valueOf(getRandomChar(random.nextInt(randString.length())));

randomString += rand;

g.translate(random.nextInt(3), random.nextInt(3));

g.drawString(rand, 13 \* i, 16);

return randomString;

}

/\*

\* 绘制干扰线

\*/

private void drowLine(Graphics g) {

int x = random.nextInt(width);

int y = random.nextInt(height);

int xl = random.nextInt(13);

int yl = random.nextInt(15);

g.drawLine(x, y, x + xl, y + yl);

}

/\*\*

\* 获取length长度随机字符串

\* @param length

\* @return

\*/

public static String getRandomString(int length){

StringBuffer str = new StringBuffer();

for(int i = 0;i < 8;i++)

str.append(String.valueOf(getRandomChar(random.nextInt(randString.length()))));

return str.toString();

}

/\*

\* 获取随机的字符

\*/

public static String getRandomChar(int num) {

return String.valueOf(randString.charAt(num));

}

@Override

public void render() {

response.setContentType("image/jpeg");// 设置相应类型,告诉浏览器输出的内容为图片

response.setHeader("Pragma", "No-cache");// 设置响应头信息，告诉浏览器不要缓存此内容

response.setHeader("Cache-Control", "no-cache");

response.setDateHeader("Expire", 0);

try {

getRandcode(request, response);// 输出图片方法

} catch (Exception e) {

e.printStackTrace();

}

}

}

package com.yawa.util;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import com.yawa.core.model.Person;

public class SecurityContextUtil {

public static final String SESSION\_LOGIN\_PERSON = "SESSION\_LOGIN\_PERSON";

public static final String SESSION\_LOGIN\_CAPTCHA = "SESSION\_LOGIN\_CAPTCHA";

public static final String SESSION\_LOGIN\_CAPTCHA\_INIT = "SESSION\_LOGIN\_CAPTCHA\_INIT";

public static final int CAPTCHA\_VALID\_TIME = 10 \* 60 \* 1000; // 验证码超时时间

/\*\*

\* 获取当前登录用户

\* @return

\* @throws AJAX\_SessionValidateException

\*/

public static Person getLoginUser(HttpServletRequest request){

return (Person)request.getSession().getAttribute(SESSION\_LOGIN\_PERSON);

}

public static Person getLoginUser(HttpSession session){

return (Person)session.getAttribute(SESSION\_LOGIN\_PERSON);

}

/\*\*

\* 获取当前登录账号标识

\* @param request

\* @return

\*/

public static Integer getLoginUserId(HttpServletRequest request){

return getLoginUser(request).getInt("id");

}

public static Integer getLoginUserId(HttpSession session){

return getLoginUser(session).getInt("id");

}

/\*\*

\* 更新当前登录用户

\* @return

\*/

public static void setLoginUser(HttpServletRequest request,Person user) {

request.getSession().setAttribute(SESSION\_LOGIN\_PERSON, user);

}

/\*\*

\* Session 过期

\* @param req

\*/

public static void invalidateSession(HttpServletRequest request){

request.getSession().invalidate();

}

public static void setCaptcha(HttpServletRequest request,String captcha) {

request.getSession().setAttribute(SESSION\_LOGIN\_CAPTCHA, captcha);

request.getSession().setAttribute(SESSION\_LOGIN\_CAPTCHA\_INIT, System.currentTimeMillis());

}

/\*\*

\* 校验验证码是否正确

\* @param request

\* @param captcha

\* @return

\*/

public static boolean isCaptchaCorrect(HttpServletRequest request,String captcha) {

String session\_captcha = (String)request.getSession().getAttribute(SESSION\_LOGIN\_CAPTCHA);

return captcha.toUpperCase().equals(session\_captcha);

}

/\*\*

\* 验证码是否有效

\* @param request

\* @return

\*/

public static boolean isCaptchaValid(HttpServletRequest request) {

long start = 0;

Object startTime = request.getSession().getAttribute(SESSION\_LOGIN\_CAPTCHA\_INIT);

if(startTime != null){

start = Long.parseLong(startTime.toString());

}

return System.currentTimeMillis() - start < CAPTCHA\_VALID\_TIME;

}

}

package com.yawa.util;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

public class SocketUtil {

/\*\*

\* Socket 请求

\* @param ip

\* @param port

\* @param content

\* @return

\* @throws UnknownHostException

\* @throws IOException

\*/

public static String sent(String ip, int port, String content)

throws UnknownHostException, IOException {

// 向服务器端发送请求，服务器IP地址和服务器监听的端口号

Socket client = new Socket(ip, port);

// 通过printWriter 来向服务器发送消息

PrintWriter printWriter = new PrintWriter(client.getOutputStream());

System.out.println("连接已建立...发送内容：" + content);

// 发送消息

printWriter.println(content);

printWriter.flush();

// InputStreamReader是低层和高层串流之间的桥梁

// client.getInputStream()从Socket取得输入串流

InputStreamReader streamReader = new InputStreamReader(

client.getInputStream());

// 链接数据串流，建立BufferedReader来读取，将BufferReader链接到InputStreamReder

BufferedReader reader = new BufferedReader(streamReader);

String advice = reader.readLine();

reader.close();

return advice;

}

}

package com.yawa.util;

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Random;

import javax.servlet.http.HttpServletRequest;

import com.alipay.util.UtilDate;

public class ToolKit {

private static final char[] DIGITS = { '0', '1', '2', '3', '4', '5', '6',

'7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };

public static String getRequestURI(HttpServletRequest request, String uri) {

int port = request.getServerPort();

if (port == 80) {

return "http://" + request.getServerName()

+ request.getContextPath() + "/" + uri;

} else {

return "http://" + request.getServerName() + ":" + port

+ request.getContextPath() + "/" + uri;

}

}

/\*\*

\* 对字符串进行MD5加密

\*

\* @param text

\* 明文

\* @return 密文

\*/

public static String md5(String text) {

MessageDigest msgDigest = null;

try {

msgDigest = MessageDigest.getInstance("MD5");

} catch (NoSuchAlgorithmException e) {

throw new IllegalStateException(

"System doesn't support MD5 algorithm.");

}

try {

msgDigest.update(text.getBytes("utf-8"));

} catch (UnsupportedEncodingException e) {

throw new IllegalStateException(

"System doesn't support your EncodingException.");

}

byte[] bytes = msgDigest.digest();

String md5Str = new String(encodeHex(bytes));

return md5Str;

}

public static char[] encodeHex(byte[] data) {

int l = data.length;

char[] out = new char[l << 1];

for (int i = 0, j = 0; i < l; i++) {

out[j++] = DIGITS[(0xF0 & data[i]) >>> 4];

out[j++] = DIGITS[0x0F & data[i]];

}

return out;

}

public static String getRandomCode(){

Random rand = new Random();

String result = "";

for(int i=0;i<6;i++){

result += rand.nextInt(10);

}

return result;

}

/\*\*

\* 生成订单号的方法

\* @return

\*/

public static synchronized String getLocalTrmSeqNum() {

return UtilDate.getOrderNum() + getRandomCode();

}

}

package com.yawa.util.common;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import org.apache.commons.httpclient.HttpClient;

import org.apache.commons.httpclient.HttpStatus;

import org.apache.commons.httpclient.NameValuePair;

import org.apache.commons.httpclient.methods.GetMethod;

import org.apache.commons.httpclient.methods.PostMethod;

import org.apache.commons.httpclient.params.HttpMethodParams;

public class HttpClientUtil {

public static String post(String URLString,NameValuePair[] pairs) {

StringBuffer response = new StringBuffer();

try {

HttpClient client = new HttpClient();

PostMethod method = new PostMethod(URLString);

method.addParameters(pairs);

method.getParams().setParameter(HttpMethodParams.HTTP\_CONTENT\_CHARSET, "UTF-8");

// 请求参数

try {

client.executeMethod(method); // 这一步就把文件上传了

if (method.getStatusCode() == HttpStatus.SC\_OK) {

BufferedReader reader = new BufferedReader(new InputStreamReader(method.getResponseBodyAsStream(), "UTF-8"));

String line = "";

while ((line = reader.readLine()) != null) {

response.append(line);

}

reader.close();

}

} catch (IOException e) {

System.out.println("执行HTTP Post请求" + URLString + "时，发生异常！");

e.printStackTrace();

} finally {

method.releaseConnection();

}

} catch (Exception e) {

e.printStackTrace();

}

return response.toString();

}

public static String get(String URLString) {

StringBuffer response = new StringBuffer();

try {

HttpClient client = new HttpClient();

GetMethod getMethod = new GetMethod(URLString);

getMethod.getParams().setParameter(HttpMethodParams.HTTP\_CONTENT\_CHARSET, "UTF-8");

// 请求参数

try {

client.executeMethod(getMethod); // 这一步就把文件上传了

if (getMethod.getStatusCode() == HttpStatus.SC\_OK) {

BufferedReader reader = new BufferedReader(new InputStreamReader(getMethod.getResponseBodyAsStream(), "UTF-8"));

String line = "";

while ((line = reader.readLine()) != null)

response.append(line);

reader.close();

}

} catch (IOException e) {

System.out.println("执行HTTP get请求" + URLString + "时，发生异常！");

e.printStackTrace();

} finally {

getMethod.releaseConnection();

}

} catch (Exception e) {

e.printStackTrace();

}

return response.toString();

}

}

package com.yawa.util.common;

import java.net.URLEncoder;

import org.apache.commons.lang.StringUtils;

import com.alibaba.fastjson.JSONObject;

import com.yawa.util.model.ConstantItems;

public class RemoteUtils {

public static void sendMsg(String mobile, String message) throws Exception {

String appData = "{'appKey':'sendMsg','login':'','userID':'','inParam':{'mobile':'" + mobile + "','message':'" + message + "'}}";

String httpURL = ConstantItems.HTTP\_SEND\_MESSAGE + URLEncoder.encode(appData, "utf-8");;

String res = HttpClientUtil.get(httpURL);

if(StringUtils.isEmpty(res))

throw new Exception("未获取返回结果！");

JSONObject json = JSONObject.parseObject(res);

if(json.getInteger("RspCode") != 100)

throw new Exception(json.getString("RspDesc"));

}

}

package com.yawa.util.db;

import com.jfinal.config.Constants;

import com.jfinal.config.Handlers;

import com.jfinal.config.Interceptors;

import com.jfinal.config.JFinalConfig;

import com.jfinal.config.Plugins;

import com.jfinal.config.Routes;

import com.jfinal.kit.PropKit;

import com.jfinal.plugin.activerecord.ActiveRecordPlugin;

import com.jfinal.plugin.activerecord.CaseInsensitiveContainerFactory;

import com.jfinal.plugin.c3p0.C3p0Plugin;

import com.jfinal.render.ViewType;

import com.yawa.core.controller.HomeController;

import com.yawa.core.model.Account;

import com.yawa.core.model.AccountMapping;

import com.yawa.core.model.PayLog;

import com.yawa.core.model.Person;

import com.yawa.core.model.Unit;

import com.yawa.util.model.Attach;

/\*\*

\* API引导式配置

\*/

public class DBConfig extends JFinalConfig {

/\*\*

\* 配置常量

\*/

public void configConstant(Constants me) {

// 加载少量必要配置，随后可用PropKit.get(...)获取值

PropKit.use("jdbc\_config.txt");

me.setDevMode(PropKit.getBoolean("devMode", false));

// 设置视图类型为Jsp，否则默认为FreeMarker

me.setViewType(ViewType.JSP);

me.setError404View("/error/404.jsp");

me.setError500View("/error/500.jsp");

}

/\*\*

\* 配置路由

\*/

public void configRoute(Routes me) {

// 系统管理

me.add("/", HomeController.class);

}

/\*\*

\* 配置插件

\*/

public void configPlugin(Plugins me) {

// 配置C3p0数据库连接池插件

C3p0Plugin c3p0Plugin = new C3p0Plugin(PropKit.get("jdbcUrl"), PropKit.get("user"),PropKit.get("password").trim());

me.add(c3p0Plugin);

// 配置ActiveRecord插件

ActiveRecordPlugin arp = new ActiveRecordPlugin(c3p0Plugin);

// 忽略大小写

arp.setContainerFactory(new CaseInsensitiveContainerFactory(true));

me.add(arp);

// 表与模型映射

arp.addMapping("attaches", Attach.class);

arp.addMapping("ct\_persons", Person.class);

arp.addMapping("ct\_accounts", Account.class);

arp.addMapping("ct\_account\_mapping", AccountMapping.class);

arp.addMapping("ct\_payment\_logs", PayLog.class);

arp.addMapping("ct\_units", Unit.class);

}

/\*\*

\* 配置全局拦截器

\*/

public void configInterceptor(Interceptors me) {

}

/\*\*

\* 配置处理器

\*/

public void configHandler(Handlers me) {

}

}

package com.yawa.util.model;

import com.jfinal.plugin.activerecord.Model;

/\*\*

\* 附件表

\* @author wrz

\*

\*/

@SuppressWarnings("serial")

public class Attach extends Model<Attach> {

public static final Attach me = new Attach();

public Attach getFirst(String attach\_type, Integer attach\_id) {

return findFirst("select \* from attaches where attach\_type = ? and attach\_id = ? and state = 1", attach\_type,attach\_id);

}

}

package com.yawa.util.model;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.List;

/\*\*

\* 分页查询返回的实体类型

\* @author yuguitao

\* @param <E>

\* @create at 2010-04-16

\*/

@SuppressWarnings("serial")

public class Page<E> implements Serializable, Iterable<E> {

// Extjs　分页

protected List<E> data;

protected int pageSize;

protected int pageNumber;

protected long total = 0;

public Page(long total) {

this.total = total;

}

public Page(int pageNumber,int pageSize,long totalCount) {

this(pageNumber,pageSize,totalCount,new ArrayList<E>(0));

}

public Page(int pageNumber,int pageSize,long total,List<E> result) {

if(pageSize <= 0) throw new IllegalArgumentException("[pageSize] must great than zero");

this.pageSize = pageSize;

this.pageNumber = pageNumber;

this.total = total;

setResult(result);

}

public void setResult(List<E> elements) {

if (elements == null)

throw new IllegalArgumentException("'result' must be not null");

this.data = elements;

}

/\*\*

\* 每一页显示的条目数

\*

\* @return 每一页显示的条目数

\*/

public int getPageSize() {

return pageSize;

}

/\*\*

\* 当前页包含的数据

\*

\* @return 当前页数据源

\*/

public List<E> getData() {

return data;

}

/\*\*

\* 得到数据库的第一条记录号

\* @return

\*/

public int getFirstResult() {

return pageNumber \* pageSize;

}

public long getTotal() {

return total;

}

@SuppressWarnings("unchecked")

public Iterator<E> iterator() {

return (Iterator<E>) (data == null ? Collections.emptyList().iterator() : data.iterator());

}

}

package com.yawa.util.model;

/\*\*

\* JSON 数据请求返回内容

\* @author 虞贵涛

\* @create at 2012-04-12

\*/

public class ResponseData {

//~ Static fields ==================================================================================================

public static final ResponseData SUCCESS\_NO\_DATA = new ResponseData(true);

public static final ResponseData FAILED\_NO\_DATA = new ResponseData(false);

//~ Instance fields ================================================================================================

private boolean success;

private Object message;

private Object data;

//~ Constructors ===================================================================================================

public ResponseData(boolean success) {

this(success, null);

}

public ResponseData(boolean success,Object message) {

this.success = success;

this.message = message;

}

public ResponseData(boolean success,Object message,Object data) {

this(success,message);

this.data = data;

}

//~ Methods ========================================================================================================

public boolean isSuccess() {

return success;

}

public void setSuccess(boolean success) {

this.success = success;

}

public Object getMessage() {

return message;

}

public void setMessage(Object message) {

this.message = message;

}

public Object getData() {

return data;

}

public void setData(Object data) {

this.data = data;

}

}

package com.yawa.core.controller;

import java.util.Date;

import org.apache.commons.lang.StringUtils;

import com.alipay.util.AlipayCore;

import com.alipay.util.AlipaySubmit;

import com.jfinal.aop.Before;

import com.jfinal.core.Controller;

import com.yawa.core.interceptor.SessionInterceptor;

import com.yawa.core.model.Account;

import com.yawa.core.model.AccountMapping;

import com.yawa.core.model.PayLog;

import com.yawa.core.model.Person;

import com.yawa.util.SecurityContextUtil;

import com.yawa.util.ToolKit;

import com.yawa.util.common.RemoteUtils;

import com.yawa.util.model.ResponseData;

public class HomeController extends Controller {

/\*\*

\* 主页

\*/

public void index() {

Person person = SecurityContextUtil.getLoginUser(getRequest());

setAttr("person", person);

render("index.jsp");

}

/\*\*

\* 业务缴费

\*/

public void charge() {

Person person = SecurityContextUtil.getLoginUser(getRequest());

setAttr("person", person);

render("charge.jsp");

}

/\*\*

\* 用户充值

\*/

public void userCharge() {

AccountMapping mapping = AccountMapping.me.getMappdingByCode(getPara("meter\_code"));

if(mapping == null){

renderJson(new ResponseData(false, "未找到相应的天然气号码记录!"));

return;

}

Person person = Person.me.findById(mapping.getInt("person\_id"));

if(person == null){

renderJson(new ResponseData(false, "未找到相应的账户信息!"));

return;

}

if(person.getInt("state") == 0){

renderJson(new ResponseData(false, "此账户已被注销!"));

return;

}

Account account = Account.me.findById(mapping.get("acct\_id"));

if(account == null){

renderJson(new ResponseData(false, "账户信息不存在，请联系管理员!"));

return;

}

String money = getPara("money");

PayLog payLog = new PayLog();

String out\_trade\_no = ToolKit.getLocalTrmSeqNum();

payLog.set("oskey",out\_trade\_no);

payLog.set("acct\_id", mapping.get("id"));

payLog.set("money", money);

if (payLog != null) {

payLog.set("iscash", 0);

payLog.set("channel","zfb");

payLog.set("status", "d");

payLog.set("creator", person.get("id"));

payLog.set("create\_date", new Date());

payLog.save();

}

setAttr("formData", AlipaySubmit.buildRequest(

AlipayCore.buildApliyForm(out\_trade\_no,"燃气充值缴费", money),"get","ok"));

render("/alipay/alipayapi.jsp");

}

/\*\*

\* 支付宝支付结果通知

\*/

public void chargeNotice() {

String trade\_no = getPara("out\_trade\_no");

String trade\_status = getPara("trade\_status");

PayLog payLog = PayLog.me.findFirst("select \* from ct\_payment\_logs where oskey = ?", trade\_no);

if(payLog != null){

payLog.set("status", trade\_status.equals("TRADE\_SUCCESS")? "1":"0").update();

if(trade\_status.equals("TRADE\_SUCCESS")){

// 交易充值成功

try{

AccountMapping mapping = AccountMapping.me.findById(payLog.getStr("acct\_id"));

Account acct = Account.me.findById(mapping.getInt("acct\_id"));

Float balance = Float.parseFloat(acct.getStr("balance"))

+ Float.parseFloat(payLog.getStr("money"));

acct.set("balance", balance).update();

}catch(Exception e){

payLog.set("status","E").set("remarks", e.getMessage()).update();

}

}

}

redirect("/index");

}

/\*\*

\* 界面登录

\*/

public void login() {

render("login.jsp");

}

/\*\*

\* 用户登录验证

\*/

public void userLogin() {

try{

Person person = Person.me.findUser(getPara("meter\_code"));

if(person == null){

renderJson(new ResponseData(false,"燃气号码不正确！"));

return;

}

// if(!SecurityContextUtil.isCaptchaValid(getRequest())){

// renderJson(new ResponseData(false,"短信验证码已经超时！"));

// return;

// }

// if(!SecurityContextUtil.isCaptchaCorrect(getRequest(), getPara("captcha"))){

// renderJson(new ResponseData(false,"短信验证码错误！"));

// return;

// }

if(person.getInt("state") == 0){

renderJson(new ResponseData(false,"账户信息已被注销，请联系管理员！"));

return;

}

if(person.getInt("validate") == 0){

renderJson(new ResponseData(false,"此账户尚未经过认证，请先注册认证！"));

return;

}

if(!person.getStr("password").equals(ToolKit.md5(person.getStr("salt") + getPara("password")))){

renderJson(new ResponseData(false,"登录密码不正确！"));

return;

}

SecurityContextUtil.setLoginUser(getRequest(), person);

}catch(Exception e){

renderJson(new ResponseData(false,"系统处理错误，错误原因：" + e.getMessage()));

return;

}

renderJson(new ResponseData(true));

}

/\*\*

\* 注册

\*/

public void register() {

render("register.jsp");

}

/\*\*

\* 注册认证

\*/

public void userReg() {

try{

Person person = Person.me.findUser(getPara("meter\_code"));

if(person == null){

renderJson(new ResponseData(false,"燃气号码不正确！"));

return;

}

// if(!SecurityContextUtil.isCaptchaValid(getRequest())){

// renderJson(new ResponseData(false,"短信验证码已经超时！"));

// return;

// }

// if(!SecurityContextUtil.isCaptchaCorrect(getRequest(), getPara("captcha"))){

// renderJson(new ResponseData(false,"短信验证码错误！"));

// return;

// }

if(person.getInt("state") == 0){

renderJson(new ResponseData(false,"账户信息已被注销，请联系管理员！"));

return;

}

if(person.getInt("validate") == 1){

renderJson(new ResponseData(false,"此账户已经注册认证，请直接登录！"));

return;

}

if(!person.getStr("cert\_num").equals(getPara("cert\_num"))){

renderJson(new ResponseData(false,"预留证件号码不准确，请联系管理员！"));

return;

}

person

.set("password", ToolKit.md5(person.getStr("salt") + getPara("password")))

.set("validate", 1)

.set("validate\_date", new Date())

.update();

SecurityContextUtil.setLoginUser(getRequest(), person);

}catch(Exception e){

renderJson(new ResponseData(false,"系统处理错误，错误原因：" + e.getMessage()));

return;

}

renderJson(new ResponseData(true));

}

/\*\*

\* 历史缴费记录

\*/

@Before(SessionInterceptor.class)

public void payLog() {

render("payLog.jsp");

}

/\*\*

\* 燃气使用记录

\*/

@Before(SessionInterceptor.class)

public void useLog() {

render("useLog.jsp");

}

/\*\*

\* 注销

\*/

@Before(SessionInterceptor.class)

public void logOut() {

SecurityContextUtil.invalidateSession(getRequest());

redirect("/index");

}

/\*\*

\* 用户缴费记录

\*/

@Before(SessionInterceptor.class)

public void getPayLogPages() {

Person person = SecurityContextUtil.getLoginUser(getRequest());

renderJson(PayLog.me.getPages(person.getLong("id"),getParaToInt("page"), getParaToInt("limit")));

}

/\*\*

\* 获取短信验证码

\*/

public void sendPhoneMsg() {

String meter\_code = getPara("meter\_code");

if(StringUtils.isEmpty(meter\_code)){

renderJson(new ResponseData(false,"燃气号码不能为空!"));

return;

}

AccountMapping mapping = AccountMapping.me.findFirst("select \* from ct\_account\_mapping where state = 1 and meter\_code = ?",meter\_code);

if(mapping == null){

renderJson(new ResponseData(false,"未找到燃气账户!"));

return;

}

Person person = Person.me.findById(mapping.getInt("person\_id"));

if(person == null){

renderJson(new ResponseData(false,"未找到个人基本信息!"));

return;

}

if(StringUtils.isEmpty(person.getStr("telephone"))){

renderJson(new ResponseData(false,"账户绑定的手机号为空!"));

return;

}

String random = ToolKit.getRandomCode();

try{

String msg = "自助缴费平台提醒您,您当前操作的验证码为：" + random + ",如不是本人发送，请忽略此短信!";

RemoteUtils.sendMsg(person.getStr("telephone"),msg);

SecurityContextUtil.setCaptcha(getRequest(), random);

renderJson(new ResponseData(true,"短信验证码发送成功，验证码在十分钟内有效!"));

}catch(Exception e){

renderJson(new ResponseData(false,"系统处理错误，错误原因：" + e.getMessage()));

}

}

}

package com.yawa.core.interceptor;

import com.jfinal.aop.Interceptor;

import com.jfinal.aop.Invocation;

import com.jfinal.core.Controller;

import com.yawa.core.model.Person;

import com.yawa.util.SecurityContextUtil;

/\*\*

\* SessionInterceptor

\* 用户过期或无效拦截

\*/

public class SessionInterceptor implements Interceptor {

public void intercept(Invocation inv) {

Controller controller = inv.getController();

Person user = SecurityContextUtil.getLoginUser(controller.getRequest());

if (user != null)

inv.invoke();

else

controller.redirect("/login");

}

}

package com.yawa.core.model;

import com.jfinal.plugin.activerecord.Model;

@SuppressWarnings("serial")

public class Account extends Model<Account> {

public static final Account me = new Account();

}

package com.yawa.core.model;

import com.jfinal.plugin.activerecord.Model;

@SuppressWarnings("serial")

public class AccountMapping extends Model<AccountMapping> {

public static final AccountMapping me = new AccountMapping();

public AccountMapping getMappdingByCode(String meter\_code){

return findFirst("select \* from ct\_account\_mapping "

+ "where state = 1 and meter\_code = ?",meter\_code);

}

}

package com.yawa.core.model;

import com.jfinal.plugin.activerecord.Db;

import com.jfinal.plugin.activerecord.Model;

import com.yawa.util.model.Page;

@SuppressWarnings("serial")

public class PayLog extends Model<PayLog> {

public static final PayLog me = new PayLog();

/\*\*

\* 分页列表

\* @param mappingid

\* @param page

\* @param limit

\* @return

\*/

public Page<PayLog> getPages(Long person, Integer page, Integer limit) {

String sql = "select count(1) from ct\_payment\_logs cpl,ct\_account\_mapping cam where cpl.status='1' and cpl.acct\_id = cam.id and cam.person\_id = ?";

Long count = Db.queryLong(sql,person);

sql = "select cpl.\*,cam.meter\_code,"

+ "(select d.name from dict d where d.group\_code='RECHARGE.CHANNEL' and d.code = cpl.channel) channelname,"

+ "(case cpl.iscash when 1 then '是' else '否' end) iscashname "

+ "from ct\_payment\_logs cpl,ct\_account\_mapping cam "

+ "where cpl.status='1' and cpl.acct\_id = cam.id and cam.person\_id = ? "

+ "order by cpl.id desc limit ?,?";

return new Page<PayLog>(page, limit, count, find(sql,person,(page - 1) \* limit, limit));

}

}

package com.yawa.core.model;

import com.jfinal.plugin.activerecord.Model;

@SuppressWarnings("serial")

public class Person extends Model<Person> {

public static final Person me = new Person();

public Person findUser(String meter) {

return findFirst("select pn.\*,mp.meter\_code from ct\_account\_mapping mp,ct\_persons pn where mp.person\_id = pn.id and mp.state = 1"

+ " and mp.meter\_code = ? order by mp.id asc",meter);

}

}

package com.yawa.core.model;

import com.jfinal.plugin.activerecord.Model;

@SuppressWarnings("serial")

public class Unit extends Model<Unit> {

public static final Unit me = new Unit();

}

-------END-----