```
package cn.g4b.fm.common.util;
import cn.g4b.fm.common.model.CertInfo;
import org.apache.commons.lang.ArrayUtils;
import org.bouncycastle.asn1.ASN1ObjectIdentifier;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.x509.GeneralName;
import org.bouncycastle.asn1.x509.GeneralNames;
import org.bouncycastle.asn1.x509.KeyPurposeId;
import org.bouncycastle.cert.X509v3CertificateBuilder;
import org.bouncycastle.cert.jcajce.JcaX509CertificateConverter;
import org.bouncycastle.cert.jcajce.JcaX509v3CertificateBuilder;
import org.bouncycastle.jce.X509KeyUsage;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.operator.ContentSigner;
import org.bouncycastle.operator.jcajce.JcaContentSignerBuilder;
import org.bouncycastle.util.encoders.Base64;
import sun.misc.BASE64Decoder;
import javax.crypto.Cipher;
import javax.crypto.NullCipher;
import javax.security.auth.x500.X500Principal;
import java.io.*;
import java.math.BigInteger;
import java.security.*;
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.ECPrivateKey;
import java.security.interfaces.ECPublicKey;
import java.security.spec.ECPrivateKeySpec;
import java.security.spec.ECPublicKeySpec;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.*;
/**
 * 产生证书工具类
 */
public class EncrypUtil {
    static {
        Security.addProvider(new BouncyCastleProvider());
    }
    /**
     * 使用 CA 根证书和 CA 密钥签发用户证书
     * @throws Exception
     */
    public static CertInfo genCertWithCaSign(String dn,String pwd) throws Exception {
//          Security.addProvider(new BouncyCastleProvider());
        System.out.println("=============RSACA 根证书签发 RSA 证书=============");
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        InputStream                                    inputStream                                    =
EncrypUtil.class.getClassLoader().getResourceAsStream("CAPrikey");
        byte[] bytes = new byte[inputStream.available()];
        inputStream.read(bytes);
        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(bytes);
```

```
        PrivateKey caPrivateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);
        CertificateFactory certificateFactory = CertificateFactory.getInstance("X509", "BC");
        inputStream =
EncrypUtil.class.getClassLoader().getResourceAsStream("CARootCert.cer");
        Certificate caRootCert = certificateFactory.generateCertificate(inputStream);
        KeyPairGenerator g = KeyPairGenerator.getInstance("RSA", "BC");
        g.initialize(2048);
        KeyPair p = g.generateKeyPair();
        PrivateKey privKey = p.getPrivate();
        PublicKey pubKey = p.getPublic();
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.YEAR,20);
        ContentSigner sigGen = new
JcaContentSignerBuilder("SHA256WITHRSA").setProvider("BC").build(caPrivateKey);
        X509v3CertificateBuilder certGen = new JcaX509v3CertificateBuilder(
                (X509Certificate) caRootCert,
                BigInteger.valueOf(new Random().nextInt()),
                new Date(System.currentTimeMillis() - 50000),
                cal.getTime(),
                new X500Principal(dn),
                pubKey).addExtension(new ASN1ObjectIdentifier("2.5.29.15"), true,
                new X509KeyUsage(X509KeyUsage.digitalSignature |
X509KeyUsage.nonRepudiation))
                .addExtension(new ASN1ObjectIdentifier("2.5.29.37"), true,
                        new DERSequence(KeyPurposeId.anyExtendedKeyUsage))
                .addExtension(new ASN1ObjectIdentifier("2.5.29.17"), true,
                        new GeneralNames(new GeneralName[]
                                {
                                        new GeneralName(GeneralName.rfc822Name,
"gmca@g4b.cn"),
                                        new GeneralName(GeneralName.dNSName,
"ca.g4b.cn")
                                }));
        X509Certificate cert = new
JcaX509CertificateConverter().setProvider("BC").getCertificate(certGen.build(sigGen));
        cert.checkValidity(new Date());
        cert.verify(caRootCert.getPublicKey());
        ByteArrayInputStream bIn = new ByteArrayInputStream(cert.getEncoded());
        CertificateFactory fact = CertificateFactory.getInstance("X.509", "BC");
        cert = (X509Certificate) fact.generateCertificate(bIn);
        System.out.println("custCert:" + Base64.toBase64String(cert.getEncoded()));
        System.out.println("custPrivateKey:" + Base64.toBase64String(privKey.getEncoded()));
        System.out.println("custPublicKey:" + Base64.toBase64String(pubKey.getEncoded()));
        System.out.println("==============RSACA 根证书签发 RSA 证书==============");
        CertInfo certInfo = new CertInfo();
        certInfo.setPubKey(Base64.toBase64String(pubKey.getEncoded()));
        certInfo.setPriKey(Base64.toBase64String(privKey.getEncoded()));
        certInfo.setCert(Base64.toBase64String(cert.getEncoded()));
        certInfo.setCertSn(cert.getSerialNumber().toString(16));
        Certificate[] chain = {cert};
        byte[] keyStore = generatePfx(g4b_alias,privKey,pwd,chain);
        certInfo.setKeyStore(keyStore);
```

```
        return certInfo;
    }
    public static byte[] readFile(String path) throws Exception {
        FileInputStream fileInputStream = new FileInputStream(path);
        byte[] bytes = new byte[fileInputStream.available()];
        fileInputStream.read(bytes);
        return bytes;
    }
    /**
     * 利用 java 原生的类实现 SHA256 加密
     * @param str 加密后的报文
     * @return
     */
    public static String getSHA256(String str){
        MessageDigest messageDigest;
        String encodestr = "";
        try {
            messageDigest = MessageDigest.getInstance("SHA-256");
            messageDigest.update(str.getBytes("UTF-8"));
            encodestr = byte2Hex(messageDigest.digest());
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return encodestr;
    }
    /**
     * 将 byte 转为 16 进制
     * @param bytes
     * @return
     */
    private static String byte2Hex(byte[] bytes){
        StringBuffer stringBuffer = new StringBuffer();
        String temp = null;
        for (int i=0;i<bytes.length;i++){
            temp = Integer.toHexString(bytes[i] & 0xFF);
            if (temp.length()==1){
                //1 得到一位的进行补 0 操作
                stringBuffer.append("0");
            }
            stringBuffer.append(temp);
        }
        return stringBuffer.toString();
    }
    public static byte[] generatePfx(String alias,PrivateKey priKey,String pwd,Certificate[]
certChain) throws Exception{
        KeyStore outputKeyStore = KeyStore.getInstance("pkcs12");
        outputKeyStore.load(null,pwd.toCharArray());
        outputKeyStore.setKeyEntry(alias,priKey,pwd.toCharArray(),certChain);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        outputKeyStore.store(out,pwd.toCharArray());
```

```
        return out.toByteArray();
    }
    public static void readPfx(String path,String alias,String pwd)throws Exception{
        InputStream in = new FileInputStream(new File(path));
        KeyStore keyStore = KeyStore.getInstance("PKCS12");
        keyStore.load(in, pwd.toCharArray());
        java.security.cert.Certificate cert =    keyStore.getCertificate(alias);
        System.out.println(Base64.toBase64String(cert.getEncoded()));
        PrivateKey privateKey = (PrivateKey) keyStore.getKey(alias, pwd.toCharArray());
        System.out.println(privateKey);
    }
    /**
     * 解析获取私钥
     * @param key
     * @return
     * @throws Exception
     */
    public static PrivateKey getPrivateKey(String key,String algorithm) throws Exception {
        byte[] bytes = new BASE64Decoder().decodeBuffer(key);
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(bytes);
        KeyFactory keyFactory = KeyFactory.getInstance(algorithm);
        return keyFactory.generatePrivate(keySpec);
    }
    /**
     * 使用公钥证书进行加密
     * @param cert
     * @param data
     * @return
     * @throws Exception
     */
    public static byte[] encryByEcPublicKey(String cert,String data) throws Exception {
        if(StringUtils.isEmpty(cert) || StringUtils.isEmpty(data))
            return null;
        ECPublicKey publicKey = (ECPublicKey) getPublicKey(cert);
        ECPublicKeySpec ecPublicKeySpec = new ECPublicKeySpec(publicKey.getW(),
                publicKey.getParams());
        Cipher cipher = new NullCipher();
        cipher.init(Cipher.ENCRYPT_MODE, publicKey, ecPublicKeySpec.getParams());
        return cipher.doFinal(data.getBytes());
    }
    /**
     * 使用私钥进行解密
     * @param priKeyStr
     * @param data
     * @return
     * @throws Exception
     */
    public static byte[] dencryByEcPublicKey(String priKeyStr,String data) throws Exception {
        if(StringUtils.isEmpty(priKeyStr) || StringUtils.isEmpty(data))
            return null;
        ECPrivateKey priKey = (ECPrivateKey) getPrivateKey(priKeyStr);
        ECPrivateKeySpec ecPrivateKeySpec = new ECPrivateKeySpec(priKey.getS(),
```

```
                priKey.getParams());
        // 对数据解密
        Cipher cipher1 = new NullCipher();
        cipher1.init(Cipher.DECRYPT_MODE, priKey, ecPrivateKeySpec.getParams());
        byte[] bytes = cipher1.doFinal(Base64.decode(data));
//          System.out.println("解密后的结果=====>>>"+new String(bytes));
        return bytes;
    }
    /**
     * 获取私钥
     * @param key
     * @return
     * @throws Exception
     */
    public static PrivateKey getPrivateKey(String key) throws Exception {
        if(key == null || "".equals(key.trim()))
            return null;
        if(key.startsWith("-----BEGIN PRIVATE KEY-----")){
            key = key.replace("-----BEGIN PRIVATE KEY-----","").replace("-----END PRIVATE
KEY-----","");
        }
        byte[] bytes = new BASE64Decoder().decodeBuffer(key);
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(bytes);
        KeyFactory keyFactory = KeyFactory.getInstance("EC");
        return keyFactory.generatePrivate(keySpec);
    }
    /**
     * 从证书中获取公钥
     * @param cert
     * @return
     * @throws Exception
     */
    public static PublicKey getPublicKey(String cert) throws Exception{
        if(cert.startsWith("-----BEGIN CERTIFICATE-----")){
            cert = cert.replace("-----BEGIN CERTIFICATE-----","").replace("-----END
CERTIFICATE-----","");
        }
        InputStream stream = new ByteArrayInputStream(Base64.decode(cert));
        CertificateFactory certificateFactory = CertificateFactory.getInstance("X.509");
        X509Certificate x509Certificate = (X509Certificate)
certificateFactory.generateCertificate(stream);
        return x509Certificate.getPublicKey();
    }
    /**
     *
     * @param priKeyBase64
     * @return
     * @throws Exception
     */
    public static String signedByEc(String priKeyBase64,String signText) throws Exception{
        //签名
        PrivateKey privateKey = getPrivateKey(priKeyBase64,"EC");
```

```
        Signature signature = Signature.getInstance("SHA256withECDSA");
        signature.initSign(privateKey);
        signature.update(signText.getBytes());
        byte []arr = signature.sign();
        return Base64.toBase64String(arr);
    }
    public static boolean verifySignByEc(String cert,String text,String signValue)throws Exception{
        PublicKey publicKey = getPublicKey(cert);
        Signature signature = Signature.getInstance("SHA256withECDSA");
        signature.initVerify(publicKey);
        signature.update(text.getBytes());
        byte[] arr = Base64.decode(signValue);
        boolean bool = signature.verify(arr);
        return bool;
    }
    /**
     * 用于验证电子营业执照签名值
     * @param oldCertBase64  证书
     * @param oldDigest          签名原文
     * @param oldSignData      签名值
     * @return
     * @throws Exception
     */
    public static boolean verifySignData(String oldCertBase64, String oldDigest, String oldSignData)throws Exception{
        byte[] certByte = Base64.decode(oldCertBase64);
//          byte[] digestByte = Base64.decode(oldDigest);
        byte[] digestByte = oldDigest.getBytes();
        byte[] signDataByte = Base64.decode(oldSignData);
        CertificateFactory cf = CertificateFactory.getInstance("X.509", "BC");
        X509Certificate cert = (X509Certificate) cf.generateCertificate(new ByteArrayInputStream(certByte));
        PublicKey publicKey = cert.getPublicKey();
        String signAlgorithm = publicKey.getAlgorithm();
//          String mergeAlgorithm = "";
        List<String> mergeAlgorithmList = new ArrayList<>();
        boolean isSM2 = false;
        if ("RSA".equalsIgnoreCase(signAlgorithm)) {
//              mergeAlgorithm = "SHA1withRSA";
            mergeAlgorithmList.add("SHA1withRSA");
            mergeAlgorithmList.add("SHA256withRSA");
            isSM2 = false;
        } else if ("EC".equalsIgnoreCase(signAlgorithm)) {
//              mergeAlgorithm = "SM3withSM2";
            mergeAlgorithmList.add("SM3withSM2");
            isSM2 = true;
        }
        Signature signature = null;
        byte[] signDataByteClone = ArrayUtils.clone(signDataByte);
        ArrayUtils.reverse(signDataByteClone);
        byte[] errorDigestByte = oldDigest.getBytes("UTF-8");
```

```
        Map<String, byte[]> signDataMap = new HashMap<String, byte[]>();
        //改写签名值 ，simple(无操作的签名值)        reverse(经过反转的签名值)
reWrite(经过重新构造的签名值)
        signDataMap.put("simple", signDataByte);
        signDataMap.put("reverse", signDataByteClone);
        if (isSM2) {
            byte[] signDataByteReWrite = VerifySM2Util.reWriteSignDataValue(signDataByte);
            signDataMap.put("reWrite", signDataByteReWrite);
        }
        boolean verifyResult = false;
        for(String mergeAlgorithm:mergeAlgorithmList) {
            if (verifyResult) {
                break;
            }
                                try {
                    verifyResult = signature.verify(entryValue);
                } catch (SignatureException e) {
//                    e.printStackTrace();
                }
                if (verifyResult) {
                    break;
                }
            }
        }
        //如果经过上面三种情况后，依然验证不通过，则说明签名值或者证书,摘要有异常,
返回异常码和异常信息
        return verifyResult;
    }
    public static boolean verifySignByRSA(String cert,String text,String signValue)throws
Exception{
        System.out.println("cert====>>>"+cert);
        System.out.println("text====>>>"+text);
        System.out.println("signValue====>>>"+signValue);
        PublicKey publicKey = getPublicKey(cert);
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initVerify(publicKey);
        byte[] textBytes = text.getBytes("ISO8859-1");
        System.out.println("len==>>"+textBytes.length);
        signature.update(textBytes);
        byte[] arr = Base64.decode(signValue);
        boolean bool = signature.verify(arr);
        System.out.println("verify result ====>>>"+bool);
        return bool;
    }
    public static String signedByRSA(byte[] keystore,String signText,String pwd) throws
Exception{
        System.out.println("signText===>>"+signText);
        System.out.println("len==>>>"+signText.getBytes().length);
        ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(keystore);
        KeyStore keyStore = KeyStore.getInstance("PKCS12");
        keyStore.load(byteArrayInputStream, pwd.toCharArray());
        java.security.cert.Certificate cert =  keyStore.getCertificate(g4b_alias);
```

```
            System.out.println(Base64.toBase64String(cert.getEncoded()));
            PrivateKey privateKey = (PrivateKey) keyStore.getKey(g4b_alias, pwd.toCharArray());
            //签名
            Signature signature = Signature.getInstance("SHA256withRSA");
            signature.initSign(privateKey);
            signature.update(signText.getBytes("ISO8859-1"));
            byte []arr = signature.sign();
            return Base64.toBase64String(arr);
        }

}
package cn.g4b.fm.common.util;
import org.apache.poi.hssf.usermodel.*;
/**
 * Created by Asus on 2021/4/13.
 */
public class ExcelUtil {
    //写入 Excel
    public static HSSFWorkbook getHHSWorkbook(String sheetName, String[] title, String[][]
values, HSSFWorkbook wb){
            if(wb==null){
                wb = new HSSFWorkbook();
            }
            HSSFSheet sheet = wb.createSheet(sheetName);
            for(int i=0;i<title.length;i++){
                sheet.setColumnWidth(i,4000);
            }
            HSSFRow row = sheet.createRow(0);
            HSSFCellStyle style = wb.createCellStyle();
            style.setAlignment(HSSFCellStyle.ALIGN_CENTER);
            style.setBorderTop(HSSFCellStyle.BORDER_THIN);
            style.setBorderBottom(HSSFCellStyle.BORDER_THIN);
            style.setBorderLeft(HSSFCellStyle.BORDER_THIN);
            style.setBorderRight(HSSFCellStyle.BORDER_THIN);
            HSSFCell cell = null;
            for (int i=0;i<title.length;i++){
                cell = row.createCell(i);
                cell.setCellValue(title[i]);
                cell.setCellStyle(style);
            }
            for (int i=0;i<values.length;i++){
                HSSFRow row1 = sheet.createRow(i+1);
                for (int j=0;j<values[i].length;j++){
                    HSSFCell cell1 = row1.createCell(j);
                    cell1.setCellValue(values[i][j]);
                    cell1.setCellStyle(style);
                }
            }
            return wb;
    }
}
package cn.g4b.fm.common.util;
```

```
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
public class FontImageUtil {
    /**
     * 创建图片
     *250, 80
     * @param content 内容
     * @param width     宽 250
     * @param height    高 80
     * @return
     */
    public static byte[] createImage(String content,String time, Integer width, Integer height) {
        BufferedImage bi = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Font font = new Font("Serif", Font.BOLD, 11);
        Graphics2D g2 = (Graphics2D) bi.getGraphics();
        g2.setBackground(Color.WHITE);
        g2.clearRect(0, 0, width, height);
        g2.setPaint(Color.BLACK);
        Font font1 = new Font("宋体", Font.BOLD, 20);
        g2.setFont(font1);
        g2.drawString(content, 5, 30);
        g2.drawString(time, 5, 60);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        try {
            ImageIO.write(bi, "PNG", out);
            return out.toByteArray();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
package cn.g4b.fm.common.util;
import cn.g4b.fm.common.model.ConstantDefinition;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
```

```
import org.apache.http.entity.StringEntity;
import org.apache.http.entity.mime.HttpMultipartMode;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.entity.mime.content.StringBody;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import java.io.*;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.*;
public class HttpUtil {
    /**
     * http post 请求
     * @param params
     * @param file
     * @param url
     */
    public static String doPostWithFile(Map<String,String> params, File file,String url) throws ClientProtocolException, IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        CloseableHttpResponse httpResponse = null;
        RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(20000).setSocketTimeout(200000000).build();
        HttpPost httpPost = new HttpPost(url);
        httpPost.setConfig(requestConfig);
        MultipartEntityBuilder multipartEntityBuilder = MultipartEntityBuilder.create();
        if(file!=null)
            multipartEntityBuilder.addBinaryBody("file",file);
        if(params!=null){
            Iterator<String> its = params.keySet().iterator();
            while (its.hasNext()){
                String key = its.next();
                multipartEntityBuilder.addTextBody(key,params.get(key));
            }
        }
        HttpEntity httpEntity = multipartEntityBuilder.build();
        httpPost.setEntity(httpEntity);
        httpResponse = httpClient.execute(httpPost);
        HttpEntity responseEntity = httpResponse.getEntity();
        int statusCode = httpResponse.getStatusLine().getStatusCode();
        if(statusCode==200){
            BufferedReader reader = new BufferedReader(new InputStreamReader(responseEntity.getContent()));
            StringBuffer buffer = new StringBuffer();
            String str = "";
            while((str = reader.readLine())!=null) {
                buffer.append(str);
            }
```

```
                System.out.println(buffer.toString());
                return buffer.toString();
            }
            return null;
        }
        public static String doPostWithFile(Map<String,String> params, File file,String url,String charset) throws IOException {
            ContentType contentType = ContentType.create(HTTP.PLAIN_TEXT_TYPE, HTTP.UTF_8);
            HttpClient client=new DefaultHttpClient();// 开启一个客户端 HTTP 请求
            RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(20000).setSocketTimeout(200000000).build();
            HttpPost post = new HttpPost(url);//创建 HTTP POST 请求
            post.setConfig(requestConfig);
            MultipartEntityBuilder builder = MultipartEntityBuilder.create();
            builder.setCharset(Charset.forName(HTTP.UTF_8));//设置请求的编码格式
            builder.setMode(HttpMultipartMode.BROWSER_COMPATIBLE);//设置浏览器兼容模式
            builder.addBinaryBody(file.getName(), file);
//              builder.addTextBody("method", params.get("method"));//设置请求参数
//              builder.addTextBody("fileTypes", params.get("fileTypes"));//设置请求参数
            if(params!=null){
                Iterator<String> its = params.keySet().iterator();
                while (its.hasNext()){
                    String key = its.next();
                    StringBody stringBody=new StringBody(params.get(key),contentType);
                    builder.addPart(key, stringBody);
                }
            }
            HttpEntity entity = builder.build();// 生成 HTTP POST 实体
            post.setEntity(entity);//设置请求参数
            HttpResponse response = client.execute(post);// 发起请求 并返回请求的响应
            if (response.getStatusLine().getStatusCode()==200) {
                HttpEntity resEntity = response.getEntity();
                if(resEntity!=null){
                    String result = EntityUtils.toString(resEntity,charset);
                    return result;
                }
            }
            return null;
        }
        public static String doPost(Map<String,String> params,String url) throws IOException {
            CloseableHttpClient httpClient = null;
            HttpPost httpPost = null;
            String result = null;
            httpClient = HttpClients.createDefault();
            httpPost = new HttpPost(url);
            //设置参数
            List<NameValuePair> list = new ArrayList<NameValuePair>();
            if(params!=null){
                Iterator<String> iterator = params.keySet().iterator();
                while(iterator.hasNext()){
                    String key = iterator.next();
                    list.add(new NameValuePair() {
```

```
                    @Override
                    public String getName() {
                        return key;
                    }
                    @Override
                    public String getValue() {
                        return params.get(key);
                    }
                });
            }
        }
        if(list.size()>0){
            UrlEncodedFormEntity entity = new UrlEncodedFormEntity(list,"UTF-8");
            httpPost.setEntity(entity);
        }
        HttpResponse response = httpClient.execute(httpPost);
        if(response!=null){
            HttpEntity resEntity = response.getEntity();
            if(resEntity!=null){
                result = EntityUtils.toString(resEntity,"UTF-8");
                return result;
            }
        }
        return null;
    }
    public static String doGet(String url,String charset){
        CloseableHttpClient httpCilent2 = HttpClients.createDefault();
        RequestConfig requestConfig = RequestConfig.custom()
                .setConnectTimeout(5000)     //设置连接超时时间
                .setConnectionRequestTimeout(5000) //  设置请求超时时间
                .setSocketTimeout(50000)
                .setRedirectsEnabled(true)//默认允许自动重定向
                .build();
        HttpGet httpGet2 = new HttpGet(url);
        httpGet2.setConfig(requestConfig);
        String srtResult = "";
        try {
            HttpResponse httpResponse = httpCilent2.execute(httpGet2);
            if(httpResponse.getStatusLine().getStatusCode() == 200){
                srtResult  =  EntityUtils.toString(httpResponse.getEntity(),charset);//获得返回
的结果
                return srtResult;
            }else if(httpResponse.getStatusLine().getStatusCode() == 400){
                //.....
            }else if(httpResponse.getStatusLine().getStatusCode() == 500){
                //.....
            }
        } catch (IOException e) {
            e.printStackTrace();
        }finally {
            try {
                httpCilent2.close();
```

```
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            return null;
        }

        public static List parseNodeToList(JsonNode jsonNode){
            if(jsonNode!=null && jsonNode.size()!=0){
                List list = new ArrayList();
                for(int i=0;i<jsonNode.size();i++){
                    JsonNode czxxNode = jsonNode.get(i);
                    if(czxxNode!=null){
                        Iterator<String> its = czxxNode.fieldNames();
                        Map<String,String> map = new HashMap<String, String>();
                        while (its.hasNext()){
                            String name = its.next();
                            String value = czxxNode.get(name).asText();
                            map.put(name,value);
                        }
                        System.out.println("gsMap==>>>"+map);
                        list.add(map);
                    }
                }
                return list;
            }
            return null;
        }
        /**
         * http post 请求
         * @param params
         * @param bytes
         * @param url
         */
        public static String doPostWithFile(Map<String,String> params, byte[] bytes,String url,String
fileName) throws ClientProtocolException, IOException {
            CloseableHttpClient httpClient = HttpClientBuilder.create().build();
            CloseableHttpResponse httpResponse = null;
            RequestConfig                    requestConfig                    =
RequestConfig.custom().setConnectTimeout(20000).setSocketTimeout(200000000).build();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setConfig(requestConfig);
            MultipartEntityBuilder multipartEntityBuilder = MultipartEntityBuilder.create();
            if(bytes!=null)

multipartEntityBuilder.addBinaryBody("file",bytes,ContentType.DEFAULT_BINARY,fileName);
            if(params!=null){
                Iterator<String> its = params.keySet().iterator();
                while (its.hasNext()){
                    String key = its.next();
                    multipartEntityBuilder.addTextBody(key,params.get(key));
                }
```

```
            }
            HttpEntity httpEntity = multipartEntityBuilder.build();
            httpPost.setEntity(httpEntity);
            httpResponse = httpClient.execute(httpPost);
            HttpEntity responseEntity = httpResponse.getEntity();
            int statusCode = httpResponse.getStatusLine().getStatusCode();
            if(statusCode==200){
                BufferedReader reader = new BufferedReader(new
InputStreamReader(responseEntity.getContent()));
                StringBuffer buffer = new StringBuffer();
                String str = "";
                while((str = reader.readLine())!=null) {
                    buffer.append(str);
                }
                System.out.println(buffer.toString());
                return buffer.toString();
            }
            return null;
        }
        public static byte[] doGetBytes(String url){
            CloseableHttpClient httpCilent2 = HttpClients.createDefault();
            RequestConfig requestConfig = RequestConfig.custom()
                    .setConnectTimeout(5000)     //设置连接超时时间
                    .setConnectionRequestTimeout(5000) //  设置请求超时时间
                    .setSocketTimeout(50000)
                    .setRedirectsEnabled(true)//默认允许自动重定向
                    .build();
            HttpGet httpGet2 = new HttpGet(url);
            httpGet2.setConfig(requestConfig);
            String srtResult = "";
            try {
                HttpResponse httpResponse = httpCilent2.execute(httpGet2);
                if(httpResponse.getStatusLine().getStatusCode() == 200){
                    return EntityUtils.toByteArray(httpResponse.getEntity());
                }else if(httpResponse.getStatusLine().getStatusCode() == 400){
                    //.....
                }else if(httpResponse.getStatusLine().getStatusCode() == 500){
                    //.....
                }
            } catch (IOException e) {
                e.printStackTrace();
            }finally {
                try {
                    httpCilent2.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            return null;
        }
        public static String sendHttpPost(String url, String body,Map<String,String> header) throws
Exception {
```

```
            CloseableHttpClient httpClient = HttpClients.createDefault();
            HttpPost httpPost = new HttpPost(url);
            //设置超时时间
            RequestConfig requestConfig =
RequestConfig.custom().setSocketTimeout(200000000).setConnectTimeout(200000000).setConn
ectionRequestTimeout(200000000).build();
            httpPost.setConfig(requestConfig);
            httpPost.addHeader("Content-Type", "application/json");
            if(body!=null) {
                StringEntity s = new StringEntity(body, ContentType.APPLICATION_JSON);
                s.setContentEncoding("UTF-8");
                s.setContentType("application/json");
                httpPost.setEntity(s);
            }
            if(header!=null){
                Iterator<String> headerItor = header.keySet().iterator();
                while (headerItor.hasNext()){
                    String key = headerItor.next();
                    String value = header.get(key);
                    httpPost.addHeader(key,value);
                }
            }
            CloseableHttpResponse response = httpClient.execute(httpPost);
//              System.out.println(response.getStatusLine().getStatusCode() + "\n");
            HttpEntity entity = response.getEntity();
            String responseContent = EntityUtils.toString(entity, "UTF-8");
//              System.out.println(responseContent);
            response.close();
            httpClient.close();
            return responseContent;
        }
}
package cn.g4b.fm.common.util;
import cn.g4b.fm.common.model.SignatureInfo;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;
import com.itextpdf.text.pdf.security.BouncyCastleDigest;
import com.itextpdf.text.pdf.security.DigestAlgorithms;
import com.itextpdf.text.pdf.security.ExternalDigest;
import com.itextpdf.text.pdf.security.ExternalSignature;
import com.itextpdf.text.pdf.security.MakeSignature;
import com.itextpdf.text.pdf.security.PrivateKeySignature;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.util.encoders.Base64;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.*;
```

```
import java.security.cert.Certificate;
import java.security.cert.CertificateFactory;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Date;
public class ItextUtil {
    public static final char[] PASSWORD = "123456".toCharArray();//keystory 密码
    /**
     * 单多次签章通用
     * @param src
     * @throws GeneralSecurityException
     * @throws IOException
     * @throws DocumentException
     */
    public void sign(String src, SignatureInfo signatureInfo){
        InputStream inputStream = null;
        FileOutputStream outputStream = null;
        ByteArrayOutputStream result = new ByteArrayOutputStream();
        try {
            inputStream = new FileInputStream(src);
            ByteArrayOutputStream tempArrayOutputStream = new
ByteArrayOutputStream();
            PdfReader reader = new PdfReader(inputStream);
            //创建签章工具 PdfStamper ，最后一个 boolean 参数是否允许被追加签名
//                    PdfStamper stamper = PdfStamper.createSignature(reader,
tempArrayOutputStream, '\0', null, true);
            String target = src.substring(0,src.toLowerCase().indexOf(".pdf"))+"_sign.pdf";
            System.out.println("签名文件===>>>>"+src);
            System.out.println("生成签名域文件===>>>>"+target);
            PdfStamper stamper = new PdfStamper(reader,new FileOutputStream(target));
            int pageNo = reader.getNumberOfPages()+1;
            stamper.insertPage(pageNo,PageSize.A4);
            int pageNum = reader.getNumberOfPages();
            String[] txts = {""};
            BaseFont baseFont = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H",
BaseFont.NOT_EMBEDDED);
            int wordCt = 41;
            int ct = 0;
            int lineheight = 20;
            for(int i=0;i<txts.length;i++){
                String tmp = txts[i];
                while (tmp.length()>wordCt){
                    String str = tmp.substring(0,wordCt);
                    tmp = tmp.substring(wordCt);
                    PdfContentByte overContent = stamper.getOverContent(pageNum);
                    overContent.beginText();
                    overContent.setFontAndSize(baseFont,12);
                    overContent.setTextMatrix(0,0);

overContent.showTextAligned(Element.ALIGN_TOP,str,50,750-(ct*lineheight),0);
//                  overContent.showText(txts[i]);
                    overContent.endText();
                    ct ++;
```

```
                    }
                    if(tmp.length()!=0){
                        PdfContentByte overContent = stamper.getOverContent(pageNum);
                        overContent.beginText();
                        overContent.setFontAndSize(baseFont,12);
                        overContent.setTextMatrix(0,0);

overContent.showTextAligned(Element.ALIGN_TOP,tmp,50,750-(ct*lineheight),0);
//                  overContent.showText(txts[i]);
                        overContent.endText();
                        ct ++;
                    }
                }
                int x = 180, y = 750-((ct)*lineheight)-20, width = 80, height = 50; // 坐标系远点位于页面左下角，左下角到右下角为 x 轴，左下角到左上角为 y 轴
                Rectangle areaSignatureRect = new Rectangle(// 签名域区域，由两个对角点构成的矩形区域
                        x, // 点 1 x 坐标
                        y, // 点 1 y 坐标
                        x + width, // 点 2 x 坐标
                        y + height // 点 2 y 坐标
                );
                PdfFormField pdfFormField = PdfFormField.createSignature(stamper.getWriter());
                pdfFormField.setFieldName(signatureInfo.getFieldName()); // 签名域标识
                pdfFormField.setPage(pageNum);
                pdfFormField.setWidget(areaSignatureRect, PdfAnnotation.HIGHLIGHT_OUTLINE);
// 高亮显示
                // 设置区域宽高和边框厚度，以及边框颜色，填充颜色
                PdfAppearance pdfAppearance = PdfAppearance.createAppearance(
                        stamper.getWriter(),
                        width,
                        height
                );
                pdfAppearance.setColorStroke(BaseColor.LIGHT_GRAY); // 边框颜色
                pdfAppearance.setColorFill(BaseColor.WHITE); // 填充颜色
                // 填充矩形区域-开始
                pdfAppearance.rectangle(
                        0, // x 轴偏移
                        0, // y 轴偏移
                        width, // 宽
                        height // 高
                );
                pdfAppearance.fillStroke();
                // 填充矩形区域-结束
                // 将外观应用到签名域对象之上
                pdfFormField.setAppearance(PdfAnnotation.APPEARANCE_NORMAL,
pdfAppearance);
                stamper.addAnnotation(pdfFormField, pageNum);
                stamper.close();
//                          PdfStamper stamper = PdfStamper.createSignature(reader,
tempArrayOutputStream, '\0', null, true);
                inputStream = new FileInputStream(target);
```

```
                reader = new PdfReader(inputStream);
                stamper = PdfStamper.createSignature(reader, tempArrayOutputStream, '\0', null,
true);
                // 获取数字签章属性对象
                PdfSignatureAppearance appearance = stamper.getSignatureAppearance();
                appearance.setReason(signatureInfo.getReason());
                appearance.setLocation(signatureInfo.getLocation());
                //设置签名的签名域名称，多次追加签名的时候，签名预名称不能一样，图片
大小受表单域大小影响（过小导致压缩）
                appearance.setVisibleSignature(signatureInfo.getFieldName());
                //读取图章图片
                Image image = Image.getInstance(signatureInfo.getImage());
                appearance.setSignatureGraphic(image);
                appearance.setCertificationLevel(signatureInfo.getCertificationLevel());
                //设置图章的显示方式，如下选择的是只显示图章（还有其他的模式，可以图
章和签名描述一同显示）
                appearance.setRenderingMode(signatureInfo.getRenderingMode());
                // 摘要算法
                ExternalDigest digest = new BouncyCastleDigest();
                // 签名算法
                ExternalSignature signature = new PrivateKeySignature(signatureInfo.getPk(),
signatureInfo.getDigestAlgorithm(), null);
                // 调用 itext 签名方法完成 pdf 签章
                MakeSignature.signDetached(appearance, digest, signature,
signatureInfo.getChain(), null, null, null, 0, signatureInfo.getSubfilter());
                //定义输入流为生成的输出流内容，以完成多次签章的过程
                inputStream = new
ByteArrayInputStream(tempArrayOutputStream.toByteArray());
                result = tempArrayOutputStream;
                outputStream = new FileOutputStream(new File(src));
                outputStream.write(result.toByteArray());
                outputStream.flush();
                File targetFile = new File(target);
                if(targetFile.exists()){
                    System.out.println("删除文件====》》》》"+target);
                    targetFile.delete();
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    if(null!=outputStream){
                        outputStream.close();
                    }
                    if(null!=inputStream){
                        inputStream.close();
                    }
                    if(null!=result){
                        result.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
```

```
                }
            }
        }

        static public java.security.cert.X509Certificate fromString(String cert)
        {
            try
            {
                CertificateFactory certificateFactory = CertificateFactory.getInstance("X509", "BC");
                if (null == certificateFactory)
                    certificateFactory = java.security.cert.CertificateFactory.getInstance
                            ("X.509");
                final String strCertificate = "-----BEGIN CERTIFICATE-----\n"
                        + cert
                        + "\n-----END CERTIFICATE-----\n";
                final java.io.ByteArrayInputStream streamCertificate = new
java.io.ByteArrayInputStream
                        (strCertificate.getBytes("UTF-8"));
                return (java.security.cert.X509Certificate)certificateFactory.generateCertificate
                        (streamCertificate);
            }
            catch (Exception ex)
            {
                System.out.println( ex.getMessage());
            }
            return null;
        }
}
package cn.g4b.fm.common.util;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.security.MessageDigest;
/**
 * md5 加密工具
 */
public class MD5Utils {
        private static final String hexDigIts[] =
{"0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"};
        /**
         * MD5 加密
         * @param origin  字符
         * @param charsetname  编码
         * @return
         */
        public static String MD5Encode(String origin, String charsetname){
            String resultString = null;
            try{
                resultString = new String(origin);
                MessageDigest md = MessageDigest.getInstance("MD5");
                if(null == charsetname || "".equals(charsetname)){
```

```
                resultString = byteArrayToHexString(md.digest(resultString.getBytes()));
            }else{
                resultString = byteArrayToHexString(md.digest(resultString.getBytes(charsetname)));
            }
        }catch (Exception e){
        }
        return resultString;
    }
    public static String MD5Encode(byte[] data){
        String resultString = null;
        try{
            MessageDigest md = MessageDigest.getInstance("MD5");
            resultString = byteArrayToHexString(md.digest(data));
        }catch (Exception e){
        }
        return resultString;
    }
    public static String byteArrayToHexString(byte b[]){
        StringBuffer resultSb = new StringBuffer();
        for(int i = 0; i < b.length; i++){
            resultSb.append(byteToHexString(b[i]));
        }
        return resultSb.toString();
    }
    public static String byteToHexString(byte b){
        int n = b;
        if(n < 0){
            n += 256;
        }
        int d1 = n / 16;
        int d2 = n % 16;
        return hexDigIts[d1] + hexDigIts[d2];
    }
    public static void main(String[] args) throws Exception {
        InputStream in = new FileInputStream(new File("C:\\Users\\lbbmn\\Pictures\\Camera Roll\\6d0e6fa3160995dec65bcc69b02e23d7.png"));
        byte[] fileBytes = new byte[in.available()];
        in.read(fileBytes);
        in.close();
        System.out.println(MD5Utils.MD5Encode(fileBytes));
    }
}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="cn.g4b.fm.dao.sys.CbAppMapper">
  <resultMap id="BaseResultMap" type="cn.g4b.fm.model.sys.CbApp">
    <id column="app_id" jdbcType="VARCHAR" property="appId" />
    <result column="user_id" jdbcType="VARCHAR" property="userId" />
    <result column="hash_value" jdbcType="VARCHAR" property="hashValue" />
    <result column="app_name" jdbcType="VARCHAR" property="appName" />
```

```
    <result column="app_desc" jdbcType="VARCHAR" property="appDesc" />
    <result column="app_code" jdbcType="VARCHAR" property="appCode" />
    <result column="app_type" jdbcType="TINYINT" property="appType" />
    <result column="callback_url" jdbcType="VARCHAR" property="callbackUrl" />
    <result column="status" jdbcType="INTEGER" property="status" />
    <result column="is_deliver" jdbcType="INTEGER" property="isDeliver" />
  </resultMap>
  <sql id="Base_Column_List">
    app_id, user_id, hash_value, app_name, app_desc, app_code, app_type,
callback_url,status,is_deliver
  </sql>
  <select id="selectByPrimaryKey" parameterType="java.lang.String"
resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List" />
    from cb_app
    where app_id = #{appId,jdbcType=VARCHAR}
  </select>
  <select id="findList" resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List" />
    from cb_app
  </select>
  <select id="getAPPList" resultType="java.util.Map">
    select app_id as "appId",app_name as "appName" from cb_app
    where 1=1
    <if test="appId!=null and appId!=''">
      and app_id = #{appId,jdbcType=VARCHAR}
    </if>
  </select>
  <delete id="deleteByPrimaryKey" parameterType="java.lang.String">
    delete from cb_app
    where app_id = #{appId,jdbcType=VARCHAR}
  </delete>
  <insert id="insert" parameterType="cn.g4b.fm.model.sys.CbApp">
    insert into cb_app (app_id, user_id, hash_value,
      app_name, app_desc, app_code,
      app_type, callback_url,status)
    values (#{appId,jdbcType=VARCHAR}, #{userId,jdbcType=VARCHAR},
#{hashValue,jdbcType=VARCHAR},
      #{appName,jdbcType=VARCHAR}, #{appDesc,jdbcType=VARCHAR},
#{appCode,jdbcType=VARCHAR},
      #{appType,jdbcType=TINYINT},
#{callbackUrl,jdbcType=VARCHAR},#{status,jdbcType=INTEGER})
  </insert>
  <insert id="insertSelective" parameterType="cn.g4b.fm.model.sys.CbApp">
    insert into cb_app
    <trim prefix="(" suffix=")" suffixOverrides=",">
      <if test="appId != null">
        app_id,
      </if>
      <if test="userId != null">
```

```
            user_id,
        </if>
        <if test="hashValue != null">
            hash_value,
        </if>
        <if test="appName != null">
            app_name,
        </if>
        <if test="appDesc != null">
            app_desc,
        </if>
        <if test="appCode != null">
            app_code,
        </if>
        <if test="appType != null">
            app_type,
        </if>
        <if test="callbackUrl != null">
            callback_url,
        </if>
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides=",">
        <if test="appId != null">
            #{appId,jdbcType=VARCHAR},
        </if>
        <if test="userId != null">
            #{userId,jdbcType=VARCHAR},
        </if>
        <if test="hashValue != null">
            #{hashValue,jdbcType=VARCHAR},
        </if>
        <if test="appName != null">
            #{appName,jdbcType=VARCHAR},
        </if>
        <if test="appDesc != null">
            #{appDesc,jdbcType=VARCHAR},
        </if>
        <if test="appCode != null">
            #{appCode,jdbcType=VARCHAR},
        </if>
        <if test="appType != null">
            #{appType,jdbcType=TINYINT},
        </if>
        <if test="callbackUrl != null">
            #{callbackUrl,jdbcType=VARCHAR},
        </if>
    </trim>
</insert>
<update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.sys.CbApp">
    update cb_app
    <set>
        <if test="userId != null">
```

```
            user_id = #{userId,jdbcType=VARCHAR},
          </if>
          <if test="hashValue != null">
            hash_value = #{hashValue,jdbcType=VARCHAR},
          </if>
          <if test="appName != null">
            app_name = #{appName,jdbcType=VARCHAR},
          </if>
          <if test="appDesc != null">
            app_desc = #{appDesc,jdbcType=VARCHAR},
          </if>
          <if test="appCode != null">
            app_code = #{appCode,jdbcType=VARCHAR},
          </if>
          <if test="appType != null">
            app_type = #{appType,jdbcType=TINYINT},
          </if>
          <if test="callbackUrl != null">
            callback_url = #{callbackUrl,jdbcType=VARCHAR},
          </if>
        </set>
        where app_id = #{appId,jdbcType=VARCHAR}
    </update>
    <update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.sys.CbApp">
        update cb_app
        set user_id = #{userId,jdbcType=VARCHAR},
          hash_value = #{hashValue,jdbcType=VARCHAR},
          app_name = #{appName,jdbcType=VARCHAR},
          app_desc = #{appDesc,jdbcType=VARCHAR},
          app_code = #{appCode,jdbcType=VARCHAR},
          app_type = #{appType,jdbcType=TINYINT},
          callback_url = #{callbackUrl,jdbcType=VARCHAR},
          status = #{status,jdbcType=INTEGER}
        where app_id = #{appId,jdbcType=VARCHAR}
    </update>
    <update id="updateAppStatus">
        update cb_app
        set
          status = #{status,jdbcType=INTEGER}
        where app_id = #{appId,jdbcType=VARCHAR}
    </update>
    <update id="updateAppIsDeliver">
        update cb_app
        set
        is_deliver = #{isDeliver,jdbcType=INTEGER}
        where app_id = #{appId,jdbcType=VARCHAR}
    </update>
</mapper>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="cn.g4b.fm.dao.sys.SysCompanyMapper">
```

```
<resultMap id="BaseResultMap" type="cn.g4b.fm.model.sys.SysCompany">
  <id column="id" jdbcType="INTEGER" property="id" />
  <result column="comp_name" jdbcType="VARCHAR" property="compName" />
  <result column="address" jdbcType="VARCHAR" property="address" />
  <result column="legal_person" jdbcType="VARCHAR" property="legalPerson" />
  <result column="org_no" jdbcType="VARCHAR" property="orgNo" />
  <result column="remark" jdbcType="VARCHAR" property="remark" />
  <result column="status" jdbcType="INTEGER" property="status" />
  <result column="create_time" jdbcType="TIMESTAMP" property="createTime" />
  <result column="check" jdbcType="INTEGER" property="check" />
  <result column="pub_key" jdbcType="VARCHAR" property="pubKey" />
  <result column="pri_key" jdbcType="VARCHAR" property="priKey" />
  <result column="cert_info" jdbcType="VARCHAR" property="certInfo" />
</resultMap>
<sql id="Base_Column_List">
  id, comp_name, address, legal_person, org_no, remark, status,
create_time,pub_key,pri_key,cert_info
</sql>
<select id="selectByPrimaryKey" parameterType="java.lang.Integer"
resultMap="BaseResultMap">
  select
  <include refid="Base_Column_List" />
  from sys_company
  where id = #{id,jdbcType=INTEGER}
</select>
<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
  delete from sys_company
  where id = #{id,jdbcType=INTEGER}
</delete>
<insert id="insert" parameterType="cn.g4b.fm.model.sys.SysCompany">
  insert into sys_company (id, comp_name, address,
    legal_person, org_no, remark,
    status, create_time,pub_key,pri_key,cert_info)
  values (#{id,jdbcType=INTEGER}, #{compName,jdbcType=VARCHAR},
#{address,jdbcType=VARCHAR},
    #{legalPerson,jdbcType=VARCHAR}, #{orgNo,jdbcType=VARCHAR},
#{remark,jdbcType=VARCHAR},
    #{status,jdbcType=INTEGER},
#{createTime,jdbcType=TIMESTAMP},#{pubKey,jdbcType=VARCHAR},#{priKey,jdbcType=VARCHAR
},#{certInfo,jdbcType=VARCHAR})
</insert>
<insert id="insertSelective" parameterType="cn.g4b.fm.model.sys.SysCompany">
  insert into sys_company
  <trim prefix="(" suffix=")" suffixOverrides=",">
    <if test="id != null">
      id,
    </if>
    <if test="compName != null">
      comp_name,
    </if>
    <if test="address != null">
      address,
```

```
        </if>
        <if test="legalPerson != null">
          legal_person,
        </if>
        <if test="orgNo != null">
          org_no,
        </if>
        <if test="remark != null">
          remark,
        </if>
        <if test="status != null">
          status,
        </if>
        <if test="createTime != null">
          create_time,
        </if>
      </trim>
      <trim prefix="values (" suffix=")" suffixOverrides=",">
        <if test="id != null">
          #{id,jdbcType=INTEGER},
        </if>
        <if test="compName != null">
          #{compName,jdbcType=VARCHAR},
        </if>
        <if test="address != null">
          #{address,jdbcType=VARCHAR},
        </if>
        <if test="legalPerson != null">
          #{legalPerson,jdbcType=VARCHAR},
        </if>
        <if test="orgNo != null">
          #{orgNo,jdbcType=VARCHAR},
        </if>
        <if test="remark != null">
          #{remark,jdbcType=VARCHAR},
        </if>
        <if test="status != null">
          #{status,jdbcType=INTEGER},
        </if>
        <if test="createTime != null">
          #{createTime,jdbcType=TIMESTAMP},
        </if>
      </trim>
    </insert>
    <update                                                    id="updateByPrimaryKeySelective"
parameterType="cn.g4b.fm.model.sys.SysCompany">
      update sys_company
      <set>
        <if test="compName != null">
          comp_name = #{compName,jdbcType=VARCHAR},
        </if>
        <if test="address != null">
```

```
            address = #{address,jdbcType=VARCHAR},
        </if>
        <if test="legalPerson != null">
            legal_person = #{legalPerson,jdbcType=VARCHAR},
        </if>
        <if test="orgNo != null">
            org_no = #{orgNo,jdbcType=VARCHAR},
        </if>
        <if test="remark != null">
            remark = #{remark,jdbcType=VARCHAR},
        </if>
        <if test="status != null">
            status = #{status,jdbcType=INTEGER},
        </if>
        <if test="createTime != null">
            create_time = #{createTime,jdbcType=TIMESTAMP},
        </if>
    </set>
    where id = #{id,jdbcType=INTEGER}
</update>
<update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.sys.SysCompany">
    update sys_company
    set comp_name = #{compName,jdbcType=VARCHAR},
        address = #{address,jdbcType=VARCHAR},
        legal_person = #{legalPerson,jdbcType=VARCHAR},
        org_no = #{orgNo,jdbcType=VARCHAR},
        remark = #{remark,jdbcType=VARCHAR},
        status = #{status,jdbcType=INTEGER}
    where id = #{id,jdbcType=INTEGER}
</update>
<select id="findList" resultMap="BaseResultMap"
parameterType="cn.g4b.fm.model.sys.SysCompany">
        select
        <include refid="Base_Column_List" />
        from sys_company
</select>
<select id="findCompList" resultMap="BaseResultMap">
    SELECT comp.*
    <if test="userId!=null">
    ,uc.comp_id as 'check'
    </if>
    from sys_company comp
    <if test="userId!=null">
    LEFT JOIN sys_user_comp uc
    on (comp.id=uc.comp_id and uc.user_id=${userId})
    </if>
    where
    status = 0
    <if test="userType!=0">
    and EXISTS (select 1 from sys_user_comp where comp_id=comp.id and user_id=${loginId})
    </if>
</select>
```

```
  <select id="findCompListByUserId" resultMap="BaseResultMap"
parameterType="java.lang.Integer">
    select
    <include refid="Base_Column_List" />
    from sys_company
    where
    status = 0
    and exists (select 1 from sys_user_comp where user_id=${userId} and
comp_id=sys_company.id)
  </select>
</mapper>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="cn.g4b.fm.dao.sys.SysMenuMapper">
  <resultMap id="BaseResultMap" type="cn.g4b.fm.model.sys.SysMenu">
    <id column="id" jdbcType="INTEGER" property="id" />
    <result column="parent_id" jdbcType="INTEGER" property="parentId" />
    <result column="menu_name" jdbcType="VARCHAR" property="menuName" />
    <result column="menu_path" jdbcType="VARCHAR" property="menuPath" />
    <result column="permission" jdbcType="VARCHAR" property="permission" />
    <result column="orderby" jdbcType="INTEGER" property="orderby" />
    <result column="status" jdbcType="INTEGER" property="status" />
    <result column="check" jdbcType="INTEGER" property="check"/>
  </resultMap>
  <sql id="Base_Column_List">
    id, parent_id, menu_name, menu_path, permission, orderby, status
  </sql>
  <select id="selectByPrimaryKey" parameterType="java.lang.Integer"
resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List" />
    from sys_menu
    where id = #{id,jdbcType=INTEGER}
  </select>
  <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
    delete from sys_menu
    where id = #{id,jdbcType=INTEGER}
  </delete>
  <insert id="insert" parameterType="cn.g4b.fm.model.sys.SysMenu">
    insert into sys_menu (id, parent_id, menu_name,
      menu_path, permission, orderby,
      status)
    values (#{id,jdbcType=INTEGER}, #{parentId,jdbcType=INTEGER},
#{menuName,jdbcType=VARCHAR},
      #{menuPath,jdbcType=VARCHAR}, #{permission,jdbcType=VARCHAR},
#{orderby,jdbcType=INTEGER},
      #{status,jdbcType=INTEGER})
  </insert>
  <insert id="insertSelective" parameterType="cn.g4b.fm.model.sys.SysMenu">
    insert into sys_menu
    <trim prefix="(" suffix=")" suffixOverrides=",">
```

```
        <if test="id != null">
          id,
        </if>
        <if test="parentId != null">
          parent_id,
        </if>
        <if test="menuName != null">
          menu_name,
        </if>
        <if test="menuPath != null">
          menu_path,
        </if>
        <if test="permission != null">
          permission,
        </if>
        <if test="orderby != null">
          orderby,
        </if>
        <if test="status != null">
          status,
        </if>
      </trim>
      <trim prefix="values (" suffix=")" suffixOverrides=",">
        <if test="id != null">
          #{id,jdbcType=INTEGER},
        </if>
        <if test="parentId != null">
          #{parentId,jdbcType=INTEGER},
        </if>
        <if test="menuName != null">
          #{menuName,jdbcType=VARCHAR},
        </if>
        <if test="menuPath != null">
          #{menuPath,jdbcType=VARCHAR},
        </if>
        <if test="permission != null">
          #{permission,jdbcType=VARCHAR},
        </if>
        <if test="orderby != null">
          #{orderby,jdbcType=INTEGER},
        </if>
        <if test="status != null">
          #{status,jdbcType=INTEGER},
        </if>
      </trim>
    </insert>
    <update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.sys.SysMenu">
      update sys_menu
      <set>
        <if test="parentId != null">
          parent_id = #{parentId,jdbcType=INTEGER},
        </if>
```

```
        <if test="menuName != null">
          menu_name = #{menuName,jdbcType=VARCHAR},
        </if>
        <if test="menuPath != null">
          menu_path = #{menuPath,jdbcType=VARCHAR},
        </if>
        <if test="permission != null">
          permission = #{permission,jdbcType=VARCHAR},
        </if>
        <if test="orderby != null">
          orderby = #{orderby,jdbcType=INTEGER},
        </if>
        <if test="status != null">
          status = #{status,jdbcType=INTEGER},
        </if>
      </set>
      where id = #{id,jdbcType=INTEGER}
    </update>
    <update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.sys.SysMenu">
      update sys_menu
      set parent_id = #{parentId,jdbcType=INTEGER},
        menu_name = #{menuName,jdbcType=VARCHAR},
        menu_path = #{menuPath,jdbcType=VARCHAR},
        permission = #{permission,jdbcType=VARCHAR},
        orderby = #{orderby,jdbcType=INTEGER},
        status = #{status,jdbcType=INTEGER}
      where id = #{id,jdbcType=INTEGER}
    </update>
    <select id="findMenuList" resultMap="BaseResultMap">
      select
      <include refid="Base_Column_List" />
      from sys_menu order by orderby
    </select>

    <delete id="deleteMenus" parameterType="java.lang.String">
      delete from sys_menu where id in (${ids})
    </delete>

    <select id="findMenuListByParentId" resultType="cn.g4b.fm.common.model.TreeBean"
parameterType="java.lang.Integer">
      select id,menu_name as name from sys_menu where parent_id=${parentId} order by
orderby
    </select>
    <select id="findListByParentId" resultMap="BaseResultMap">
      select
      <include refid="Base_Column_List" />
      from sys_menu
      where parent_id=${parentId}
      <if test="userId!=null">
        and EXISTS (select 1 from sys_user_menu WHERE menu_id=sys_menu.id and
user_id=${userId})
      </if>
```

```
    and status=0
    order by orderby
  </select>
  <select id="findLoginMenus" resultMap="BaseResultMap" parameterType="java.lang.Integer">
    select
    <include refid="Base_Column_List" />
    from sys_menu where EXISTS (select 1 from sys_user_menu WHERE menu_id=sys_menu.id and user_id=${userId}) and status=0 order by orderby
  </select>

  <select id="findListAndCheckByParentId" resultMap="BaseResultMap">
    select m.*,um.user_id as 'check' from sys_menu m LEFT JOIN sys_user_menu um on (m.id=um.menu_id and um.user_id=${userId}) where
    parent_id=${parentId}
    <if test="loginId!=null">
      and EXISTS (select 1 from sys_user_menu WHERE menu_id=m.id and user_id=${loginId})
    </if>
    and status=0 order by orderby
  </select>
  <delete id="deleteByParentId" parameterType="java.lang.String">
    delete from sys_menu where parent_id=${parentId}
  </delete>
  <select id="findGeneralMenuList" resultMap="BaseResultMap" parameterType="java.lang.Integer">
    select
    <include refid="Base_Column_List" />
    from sys_menu
    where EXISTS (SELECT 1 from sys_role_menu WHERE sys_menu.id=menu_id and role_id in (select role_id from sys_role_user where user_id = #{userId,jdbcType=VARCHAR}))
    and status=0
    order by parent_id,orderby
  </select>
</mapper>
package cn.g4b.fm.service.sys;
import cn.g4b.fm.common.model.*;
import cn.g4b.fm.common.util.JsonNodeUtil;
import cn.g4b.fm.common.util.LogUtils;
import cn.g4b.fm.common.util.SM3Util;
import cn.g4b.fm.dao.sys.CbAppMapper;
import cn.g4b.fm.dao.sys.SysUserMapper;
import cn.g4b.fm.dao.szg.*;
import cn.g4b.fm.model.sys.CbApp;
import cn.g4b.fm.model.sys.SysCompany;
import cn.g4b.fm.model.sys.SysUser;
import cn.g4b.fm.model.sys.SysUserMenu;
import cn.g4b.fm.model.szg.*;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.github.pagehelper.Page;
import com.github.pagehelper.PageHelper;
```

```
                //用户认证
                CbApp appinfo = appMapper.selectByPrimaryKey(appId);
                if(appinfo == null || !userId.equals(appinfo.getUserId())
|| !hashValue.equals(appinfo.getHashValue())){
                    throw new BizException(CommonEnum.APPINFO_ERR);
                }
                if(appinfo.getStatus().intValue() != 1){
                    throw new BizException(CommonEnum.APPINFO_ERR,"该应用未上线");
                }
                String hv = SM3Util.encrypt(appinfo.getAppId()+appinfo.getAppCode());
                if(!hv.equals(hashValue)){
                    throw new BizException(CommonEnum.APPINFO_ERR);
                }
                String tokenId = UUID.randomUUID().toString();
                Map m = new HashMap();
                m.put("tokenId",tokenId);
                //设置登陆时间
                appinfo.setLoginTime(System.currentTimeMillis());
                //查询 appId 相关的 api
                List<String> apiList = apiMapper.getAppApiList(appId);
                appinfo.setApiList(apiList);
                //将 tokenId 放到 redis 缓存中，并设置他的有效时间
                boolean flag = redisUtil.set(tokenId,appinfo,LOGIN_TIMEOUT);
                if(!flag){
                    throw new BizException(CommonEnum.INTERNAL_SERVER_ERROR);
                }
                return ResultBody.success(m);
            } catch (JsonProcessingException e) {
                e.printStackTrace();
                throw new BizException(CommonEnum.PARAMS_CHECK_ERR);
            } catch (NullParamException e) {
                e.printStackTrace();
                return ResultBody.error(e.getMessage());
            } catch (IOException e) {
                e.printStackTrace();
                return ResultBody.error(e.getMessage());
            }
        }
        public PageBean<CbApp> findListByPage(PageBean pageBean, CbApp appInfo) {
            if(pageBean==null)
                pageBean = new PageBean();
            Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());
            List<CbApp> list = appMapper.findList(appInfo);
            pageBean.setData(list);
            pageBean.setCount(page.getTotal());
            return pageBean;
        }
        public PageBean<CbAppExtendInfo> findExtendListByPage(PageBean pageBean,
CbAppExtendInfo appInfo) {
            if(pageBean==null)
                pageBean = new PageBean();
            Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());
```

```
        List<CbAppExtendInfo> list = appExtendInfoMapper.findList(appInfo);
        pageBean.setData(list);
        pageBean.setCount(page.getTotal());
        return pageBean;
    }
    @Transactional(rollbackFor = Exception.class)
    public void saveAppInfo(CbApp appinfo) {
        if(appinfo == null){
            throw new BizException(CommonEnum.ERROR_NULL_PARAMS);
        }
        appinfo.setAppCode(UUID.randomUUID().toString());
        String hv = SM3Util.encrypt(appinfo.getAppId()+appinfo.getAppCode());
        appinfo.setHashValue(hv);
        appMapper.insert(appinfo);
    }
    @Transactional(rollbackFor = Exception.class)
    public void saveDevelopmentInfo(SysUser user, CbAppExtendInfo extendInfo) {
        //检查用户类型
        if(user.getUserType().intValue() != 1)
            throw new BizException(CommonEnum.PARAMS_CHECK_ERR,"用户类型错误");
        //检查经营范围
        String busScope = user.getBusScope();
        if(StringUtils.isEmpty(busScope))
            throw new BizException(CommonEnum.PARAMS_CHECK_ERR,"经营范围错误");
        String[] bss = busScope.split(",");
        boolean flag = true;
        for(String bs : bss){
            if("1".equals(bs)){
                flag = false;
                break;
            }
        }
        if(flag){
            throw new BizException(CommonEnum.PARAMS_CHECK_ERR,"经营范围错误");
        }
        CbApp appInfo = new CbApp();
        appInfo.setAppName(extendInfo.getAppName());
        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
        String appId = user.getUserName().toUpperCase()+"_"+sdf.format(new Date());
        appInfo.setAppId(appId);
        appInfo.setUserId(appId);
        appInfo.setCallbackUrl(extendInfo.getCallbackUrl());
        appInfo.setStatus(0);
        //保存 app 信息
        saveAppInfo(appInfo);
        //保存扩展信息
        extendInfo.setAppId(appId);
        appExtendInfoMapper.insert(extendInfo);
        //关联用户表信息
        user.setAppLinkAccount(appId);
        userMapper.updateByPrimaryKey(user);
    }
```

```
    public CbAppExtendInfo getExtentInfoByAppId(String appId) {
        return appExtendInfoMapper.getExtendInfoByAppId(appId);
    }
    @Transactional
    public void updateAppStatus(String appId, Integer status) {
        if(StringUtils.isEmpty(appId) || status == null){
            throw new BizException(CommonEnum.PARAMS_CHECK_ERR);
        }
        appMapper.updateAppStatus(appId,status);
    }
    @Transactional
    public void updateAppIsDeliver(String appId, Integer isDeliver) {
        if(StringUtils.isEmpty(appId) || isDeliver == null){
            throw new BizException(CommonEnum.PARAMS_CHECK_ERR);
        }
        appMapper.updateAppIsDeliver(appId,isDeliver);
    }
    public PageBean findPageList(PageBean pageBean, SzgSealInfoVo
szgSealInfoVo,SzgSealOrder sealOrder) {
        if(pageBean==null)
            pageBean = new PageBean();
        Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());
        if(szgSealInfoVo.getOrderType()!=null && (szgSealInfoVo.getOrderType() == "2" ||
szgSealInfoVo.getOrderType().equals("2"))){
            SzgStampOrderVo szgStampOrderVo = new SzgStampOrderVo();
            szgStampOrderVo.setStartTime(szgSealInfoVo.getStartedTime());
            szgStampOrderVo.setEndTime(szgSealInfoVo.getEndedTime());
            if(sealOrder.getAppId()!=null){
                szgStampOrderVo.setAppId(sealOrder.getAppId());
            }else if(sealOrder.getSpNo()!=null){
                szgStampOrderVo.setSpNo(sealOrder.getSpNo());
            }
            List<SzgStampOrderVo>                                    list
=
szgStampOrderMapper.findPageList(szgStampOrderVo);
            //前端同一个 table，重新赋值 orderstatus 值
            List<SzgStampOrder> stampList = new ArrayList<>();
            for (SzgStampOrder stampOrder:list){
                if (stampOrder.getOrderStatus()==1){
                    stampOrder.setOrderStatus(7);
                }else if(stampOrder.getOrderStatus()==2){
                    stampOrder.setOrderStatus(8);
                }else if(stampOrder.getOrderStatus()==3){
                    stampOrder.setOrderStatus(9);
                }else if(stampOrder.getOrderStatus()==4){
                    stampOrder.setOrderStatus(10);
                }else if(stampOrder.getOrderStatus()==5){
                    stampOrder.setOrderStatus(11);
                }else if(stampOrder.getOrderStatus()==6){
                    stampOrder.setOrderStatus(12);
                }
                stampList.add(stampOrder);
            }
```

```
                pageBean.setData(list);
                pageBean.setCount(page.getTotal());
                return pageBean;
            }else{
                SzgSealOrderVo szgSealOrderVo = new SzgSealOrderVo();
                szgSealOrderVo.setStartTime(szgSealInfoVo.getStartedTime());
                szgSealOrderVo.setEndTime(szgSealInfoVo.getEndedTime());
                if(sealOrder.getAppId()!=null){
                    szgSealOrderVo.setAppId(sealOrder.getAppId());
                }else if(sealOrder.getSpNo()!=null){
                    szgSealOrderVo.setSpNo(sealOrder.getSpNo());
                }
                List<SzgSealOrder> list = szgSealOrderMapper.findPageList(szgSealOrderVo);
                pageBean.setData(list);
                pageBean.setCount(page.getTotal());
                return pageBean;
            }
        }
        public List<Map> getApiListByAppId(String appId) {
            if(StringUtils.isEmpty(appId))
                return null;
            return apiMapper.getApiListByAppId(appId);
        }
        @Transactional(rollbackFor = Exception.class)
        public void saveAppApi(String apiIds, String appId) {
            if(StringUtils.isEmpty(appId))
                throw new BizException(CommonEnum.ERROR_NULL_PARAMS,"参数不能为空");
            String[] apiArr = apiIds.split(",");
            appApiMapper.deleteByAppId(appId);
            for (String apiId:apiArr){
                SzgAppApi appApi = new SzgAppApi(Integer.parseInt(apiId),appId);
                appApiMapper.insert(appApi);
            }
        }
        public List<Map> getAPPList() {
            return appMapper.getAPPList(null);
        }
}
package cn.g4b.fm.service.sys;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Component;
import org.springframework.util.CollectionUtils;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.TimeUnit;
/**
 * redisTemplate 封装
 *
 */
@Component
```

```
public class RedisService {
    @Autowired
    private RedisTemplate<String, Object> redisTemplate;
    public RedisService(RedisTemplate<String, Object> redisTemplate) {
        this.redisTemplate = redisTemplate;
    }
    /**
     * 指定缓存失效时间
     * @param key 键
     * @param time 时间(秒)
     * @return
     */
    public boolean expire(String key,long time){
        try {
            if(time>0){
                redisTemplate.expire(key, time, TimeUnit.SECONDS);
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 根据 key 获取过期时间
     * @param key 键 不能为 null
     * @return 时间(秒) 返回 0 代表为永久有效
     */
    public long getExpire(String key){
        return redisTemplate.getExpire(key,TimeUnit.SECONDS);
    }
    /**
     * 判断 key 是否存在
     * @param key 键
     * @return true 存在 false 不存在
     */
    public boolean hasKey(String key){
        try {
            return redisTemplate.hasKey(key);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 删除缓存
     * @param key 可以传一个值 或多个
     */
    @SuppressWarnings("unchecked")
    public void del(String ... key){
        if(key!=null&&key.length>0){
            if(key.length==1){
```

```
                redisTemplate.delete(key[0]);
            }else{
                redisTemplate.delete(CollectionUtils.arrayToList(key));
            }
        }
    }
    //============================String=============================
    /**
     * 普通缓存获取
     * @param key 键
     * @return 值
     */
    public Object get(String key){
        return key==null?null:redisTemplate.opsForValue().get(key);
    }
    /**
     * 普通缓存放入
     * @param key 键
     * @param value 值
     * @return true 成功 false 失败
     */
    public boolean set(String key,Object value) {
        try {
            redisTemplate.opsForValue().set(key, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 普通缓存放入并设置时间
     * @param key 键
     * @param value 值
     * @param time 时间(秒) time 要大于 0 如果 time 小于等于 0 将设置无限期
     * @return true 成功 false 失败
     */
    public boolean set(String key,Object value,long time){
        try {
            if(time>0){
                redisTemplate.opsForValue().set(key, value, time, TimeUnit.SECONDS);
            }else{
                set(key, value);
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 递增
```

```
     * @param key 键
     * @param delta 要增加几(大于 0)
     * @return
     */
    public long incr(String key, long delta){
        if(delta<0){
            throw new RuntimeException("递增因子必须大于 0");
        }
        return redisTemplate.opsForValue().increment(key, delta);
    }
    /**
     * 递减
     * @param key 键
     * @param delta 要减少几(小于 0)
     * @return
     */
    public long decr(String key, long delta){
        if(delta<0){
            throw new RuntimeException("递减因子必须大于 0");
        }
        return redisTemplate.opsForValue().increment(key, -delta);
    }
    //================================Map=================================
    /**
     * HashGet
     * @param key 键 不能为 null
     * @param item 项 不能为 null
     * @return 值
     */
    public Object hget(String key,String item){
        return redisTemplate.opsForHash().get(key, item);
    }
    /**
     * 获取 hashKey 对应的所有键值
     * @param key 键
     * @return 对应的多个键值
     */
    public Map<Object,Object> hmget(String key){
        return redisTemplate.opsForHash().entries(key);
    }
    /**
     * HashSet
     * @param key 键
     * @param map 对应多个键值
     * @return true 成功 false 失败
     */
    public boolean hmset(String key, Map<String,Object> map){
        try {
            redisTemplate.opsForHash().putAll(key, map);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
```

```
            return false;
        }
    }
    /**
     * HashSet 并设置时间
     * @param key 键
     * @param map 对应多个键值
     * @param time 时间(秒)
     * @return true 成功 false 失败
     */
    public boolean hmset(String key, Map<String,Object> map, long time){
        try {
            redisTemplate.opsForHash().putAll(key, map);
            if(time>0){
                expire(key, time);
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 向一张 hash 表中放入数据,如果不存在将创建
     * @param key 键
     * @param item 项
     * @param value 值
     * @return true 成功 false 失败
     */
    public boolean hset(String key,String item,Object value) {
        try {
            redisTemplate.opsForHash().put(key, item, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 向一张 hash 表中放入数据,如果不存在将创建
     * @param key 键
     * @param item 项
     * @param value 值
     * @param time 时间(秒) 注意:如果已存在的 hash 表有时间,这里将会替换原有的时
间
     * @return true 成功 false 失败
     */
    public boolean hset(String key,String item,Object value,long time) {
        try {
            redisTemplate.opsForHash().put(key, item, value);
            if(time>0){
                expire(key, time);
```

```
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 删除 hash 表中的值
     * @param key 键 不能为 null
     * @param item 项 可以使多个 不能为 null
     */
    public void hdel(String key, Object... item){
        redisTemplate.opsForHash().delete(key,item);
    }
    /**
     * 判断 hash 表中是否有该项的值
     * @param key 键 不能为 null
     * @param item 项 不能为 null
     * @return true 存在 false 不存在
     */
    public boolean hHasKey(String key, String item){
        return redisTemplate.opsForHash().hasKey(key, item);
    }
    /**
     * hash 递增 如果不存在,就会创建一个 并把新增后的值返回
     * @param key 键
     * @param item 项
     * @param by 要增加几(大于 0)
     * @return
     */
    public double hincr(String key, String item,double by){
        return redisTemplate.opsForHash().increment(key, item, by);
    }
    /**
     * hash 递减
     * @param key 键
     * @param item 项
     * @param by 要减少记(小于 0)
     * @return
     */
    public double hdecr(String key, String item,double by){
        return redisTemplate.opsForHash().increment(key, item,-by);
    }
    //============================set============================
    /**
     * 根据 key 获取 Set 中的所有值
     * @param key 键
     * @return
     */
    public Set<Object> sGet(String key){
        try {
```

```
            return redisTemplate.opsForSet().members(key);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
    /**
     * 根据 value 从一个 set 中查询,是否存在
     * @param key 键
     * @param value 值
     * @return true 存在 false 不存在
     */
    public boolean sHasKey(String key,Object value){
        try {
            return redisTemplate.opsForSet().isMember(key, value);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 将数据放入 set 缓存
     * @param key 键
     * @param values 值 可以是多个
     * @return 成功个数
     */
    public long sSet(String key, Object...values) {
        try {
            return redisTemplate.opsForSet().add(key, values);
        } catch (Exception e) {
            e.printStackTrace();
            return 0;
        }
    }
    /**
     * 将 set 数据放入缓存
     * @param key 键
     * @param time 时间(秒)
     * @param values 值 可以是多个
     * @return 成功个数
     */
    public long sSetAndTime(String key,long time,Object...values) {
        try {
            Long count = redisTemplate.opsForSet().add(key, values);
            if(time>0) {
                expire(key, time);
            }
            return count;
        } catch (Exception e) {
            e.printStackTrace();
            return 0;
        }
```

```
}
/**
 * 获取 set 缓存的长度
 * @param key 键
 * @return
 */
public long sGetSetSize(String key){
    try {
        return redisTemplate.opsForSet().size(key);
    } catch (Exception e) {
        e.printStackTrace();
        return 0;
    }
}
/**
 * 移除值为 value 的
 * @param key 键
 * @param values 值 可以是多个
 * @return 移除的个数
 */
public long setRemove(String key, Object ...values) {
    try {
        Long count = redisTemplate.opsForSet().remove(key, values);
        return count;
    } catch (Exception e) {
        e.printStackTrace();
        return 0;
    }
}
//================================list=================================
/**
 * 获取 list 缓存的内容
 * @param key 键
 * @param start 开始
 * @param end 结束  0 到 -1 代表所有值
 * @return
 */
public List<Object> lGet(String key, long start, long end){
    try {
        return redisTemplate.opsForList().range(key, start, end);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
/**
 * 获取 list 缓存的长度
 * @param key 键
 * @return
 */
public long lGetListSize(String key){
    try {
```

```
            return redisTemplate.opsForList().size(key);
        } catch (Exception e) {
            e.printStackTrace();
            return 0;
        }
    }
    /**
     * 通过索引 获取 list 中的值
     * @param key 键
     * @param index 索引  index>=0 时， 0 表头，1 第二个元素，依次类推；index<0 时，-1，表尾，-2 倒数第二个元素，依次类推
     * @return
     */
    public Object lGetIndex(String key,long index){
        try {
            return redisTemplate.opsForList().index(key, index);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
    /**
     * 将 list 放入缓存
     * @param key 键
     * @param value 值
     * @return
     */
    public boolean lSet(String key, Object value) {
        try {
            redisTemplate.opsForList().rightPush(key, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 将 list 放入缓存
     * @param key 键
     * @param value 值
     * @param time 时间(秒)
     * @return
     */
    public boolean lSet(String key, Object value, long time) {
        try {
            redisTemplate.opsForList().rightPush(key, value);
            if (time > 0) {
                expire(key, time);
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
```

```
            return false;
        }
    }
    /**
     * 将 list 放入缓存
     * @param key 键
     * @param value 值
     * @return
     */
    public boolean lSet(String key, List<Object> value) {
        try {
            redisTemplate.opsForList().rightPushAll(key, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 将 list 放入缓存
     * @param key 键
     * @param value 值
     * @param time 时间(秒)
     * @return
     */
    public boolean lSet(String key, List<Object> value, long time) {
        try {
            redisTemplate.opsForList().rightPushAll(key, value);
            if (time > 0) {
                expire(key, time);
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
    /**
     * 根据索引修改 list 中的某条数据
     * @param key 键
     * @param index 索引
     * @param value 值
     * @return
     */
    public boolean lUpdateIndex(String key, long index,Object value) {
        try {
            redisTemplate.opsForList().set(key, index, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
```

```
    }
    /**
     * 移除 N 个值为 value
     * @param key 键
     * @param count 移除多少个
     * @param value 值
     * @return 移除的个数
     */
    public long lRemove(String key,long count,Object value) {
        try {
            Long remove = redisTemplate.opsForList().remove(key, count, value);
            return remove;
        } catch (Exception e) {
            e.printStackTrace();
            return 0;
        }
    }
    public boolean tryLock(String key,Object value,long timeout,TimeUnit unit){
        return redisTemplate.opsForValue().setIfAbsent(key,value,timeout,unit);
    }
}
@Service
public class SysRoleService {
    private Logger logger = LogUtils.getPlatformLogger();
    @Autowired
    private SysRoleMapper roleMapper;
    @Autowired
    private SysMenuMapper menuMapper;
    @Autowired
    private SysRoleMenuMapper roleMenuMapper;
    @Autowired
    private SysRoleUserMapper roleUserMapper;
    public SysRole getRoleById(Integer roleId) {
        return roleMapper.selectByPrimaryKey(roleId);
    }
    public void saveUser(SysRole role) {
        if(role == null){
            throw new BizException(CommonEnum.ERROR_NULL_PARAMS);
        }
        if(role.getRoleId() == null){
            roleMapper.insert(role);
        }else{
            roleMapper.updateByPrimaryKey(role);
        }
    }
    public PageBean findListByPage(PageBean pageBean, SysRole role) {
        if(pageBean==null)
            pageBean = new PageBean();
        Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());
        List<SysCompany> list = roleMapper.findList(role);
        pageBean.setData(list);
        pageBean.setCount(page.getTotal());
```

```
        return pageBean;
    }
    public List<Map> findRoleMenuList(Integer roleId) {
//          if(roleId == null)
//              return null;
        List<TreeBean> memuList = menuMapper.findMenuListByParentId(0);
        List<Map> restList = new ArrayList<>();
        List<SysRoleMenu> roleMenuList = null;
        if(roleId != null){
            roleMenuList = roleMenuMapper.findListByRoleId(roleId);
        }
        if(memuList!=null && !memuList.isEmpty()){
            for(TreeBean b:memuList){
                Map m = new HashMap();
                m.put("id",b.getId());
                m.put("title",b.getName());
                m.put("spread",true);
                List<TreeBean> chList = menuMapper.findMenuListByParentId(b.getId());
                if(chList!=null && !chList.isEmpty()){
                    List<Map> chRestList = new ArrayList<>();
                    for(TreeBean c:chList){
                        Map cm = new HashMap();
                        cm.put("id",c.getId());
                        cm.put("title",c.getName());
                        if(roleMenuList!=null && !roleMenuList.isEmpty()){
                            for(SysRoleMenu roleMenu : roleMenuList){
                                if(roleMenu.getMenuId().intValue()                    ==
c.getId().intValue()){
                                    cm.put("checked",true);
                                    break;
                                }
                            }
                        }
                        chRestList.add(cm);
                    }
                    m.put("children",chRestList);
                }
                restList.add(m);
            }
        }
        return restList;
    }
    @Transactional(rollbackFor = Exception.class)
    public void saveRoleMenu(Integer roleId, String menuIds) throws FmException {
        if(roleId == null)
            throw new FmException(500,"roleId 不能为空");
        roleMenuMapper.deleteListByRoleId(roleId);
        if(!StringUtils.isEmpty(menuIds)){
            String[] menuList = menuIds.split(",");
            if(menuList.length != 0){
                for (String menuId : menuList){
                    SysRoleMenu roleMenu = new SysRoleMenu();
```

```
                        roleMenu.setRoleId(roleId);
                        roleMenu.setMenuId(Integer.parseInt(menuId));
                        roleMenuMapper.insert(roleMenu);
                    }
                }
            }
        }
        public List<SysRole> findRoleList(Integer userId) {
            List<SysRole> roleVoList = roleMapper.findRoleList();
            if(userId != null) {
                List<SysRoleUser> roleUserList = roleUserMapper.findListByUserId(userId);
                if(roleUserList != null && !roleUserList.isEmpty()){
                    for (SysRoleUser roleUser : roleUserList){
                        for(SysRole role:roleVoList){
                            if(roleUser.getRoleId().intValue() == role.getRoleId().intValue()){
                                role.setCheck(true);
                                break;
                            }
                        }
                    }
                }
            }
            return roleVoList;
        }
}
package cn.g4b.fm.configuration;
import com.alibaba.druid.pool.DruidDataSource;
import com.alibaba.druid.support.http.StatViewServlet;
import com.alibaba.druid.support.http.WebStatFilter;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import javax.servlet.ServletRegistration;
import javax.sql.DataSource;
import java.sql.SQLException;
@Configuration
public class DruidConfig {
    @Value("${druid.login.enabled}")
    private boolean druidLoginEnabled;
    @Value("${druid.login.username}")
    private String druidLoginUsername;
    @Value("${druid.login.password}")
    private String druidLoginPassword;
    @Value("${spring.datasource.url}")
    private String dbUrl;
    @Value("${spring.datasource.username}")
    private String username;
    @Value("${spring.datasource.password}")
    private String password;
```

```
    @Value("${spring.datasource.driver-class-name}")
    private String driverClassName;
    @Value("${spring.datasource.initialSize}")
    private int initialSize;
    @Value("${spring.datasource.minIdle}")
    private int minIdle;
    @Value("${spring.datasource.maxActive}")
    private int maxActive;
    @Value("${spring.datasource.maxWait}")
    private int maxWait;
    @Value("${spring.datasource.timeBetweenEvictionRunsMillis}")
    private int timeBetweenEvictionRunsMillis;
    @Value("${spring.datasource.minEvictableIdleTimeMillis}")
    private int minEvictableIdleTimeMillis;
    @Value("${spring.datasource.validationQuery}")
    private String validationQuery;
    @Value("${spring.datasource.testWhileIdle}")
    private boolean testWhileIdle;
    @Value("${spring.datasource.testOnBorrow}")
    private boolean testOnBorrow;
    @Value("${spring.datasource.testOnReturn}")
    private boolean testOnReturn;
    @Value("${spring.datasource.poolPreparedStatements}")
    private boolean poolPreparedStatements;
    @Value("${spring.datasource.filters}")
    private String filters;
    @Bean
    public ServletRegistrationBean    druidServlet(){
        ServletRegistrationBean reg = new ServletRegistrationBean();
        reg.setServlet(new StatViewServlet());
        reg.addUrlMappings("/druid/*");
        if(druidLoginEnabled){
            reg.addInitParameter("loginUsername",druidLoginUsername);
            reg.addInitParameter("loginPassword",druidLoginPassword);
        }
        return reg;
    }
    @Bean
    public FilterRegistrationBean filterRegistrationBean(){
        FilterRegistrationBean filterRegistrationBean = new FilterRegistrationBean();
        filterRegistrationBean.setFilter(new WebStatFilter());
        filterRegistrationBean.addUrlPatterns("/*");

filterRegistrationBean.addInitParameter("exclusions","*.js,*.gif,*.jpg,*.png,*.css,*.ico,/druid/*");
        filterRegistrationBean.addInitParameter("profileEnable","true");
        filterRegistrationBean.addInitParameter("principalCookieName","USER_COOKIE");
        filterRegistrationBean.addInitParameter("principalSessionName","USER_SESSION");
        return    filterRegistrationBean;
    }
    @Bean
    @Primary
    public DataSource druidDataSource(){
```

```
            DruidDataSource dataSource = new DruidDataSource();
            dataSource.setUrl(this.dbUrl);
            dataSource.setUsername(username);
            dataSource.setPassword(password);
            dataSource.setDriverClassName(driverClassName);
            dataSource.setInitialSize(initialSize);
            dataSource.setMinIdle(minIdle);
            dataSource.setMaxActive(maxActive);
            dataSource.setMaxWait(maxWait);
            dataSource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);
            dataSource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);
            dataSource.setValidationQuery(validationQuery);
            dataSource.setTestWhileIdle(testWhileIdle);
            dataSource.setTestOnBorrow(testOnBorrow);
            dataSource.setTestOnReturn(testOnReturn);
            dataSource.setPoolPreparedStatements(poolPreparedStatements);
            try {
                dataSource.setFilters(filters);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return dataSource;
        }
}
package cn.g4b.fm.configuration;
import com.github.pagehelper.PageHelper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import java.util.Properties;
/**
 * mybatis 分页配置
 */
@Configuration
public class PageHelperConfig {
    //配置 mybatis 的分页插件 pageHelper
    @Bean
    public PageHelper pageHelper(){
        PageHelper pageHelper = new PageHelper();
        Properties properties = new Properties();
        properties.setProperty("offsetAsPageNum","true");
        properties.setProperty("rowBoundsWithCount","true");
        properties.setProperty("reasonable","true");
        properties.setProperty("dialect","mysql");       //配置 mysql 数据库的方言
        pageHelper.setProperties(properties);
        return pageHelper;
    }
}
package cn.g4b.fm.configuration;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.springframework.core.convert.converter.Converter;
import org.springframework.util.StringUtils;
```

```
public class StringToDateConverter implements Converter<String, Date> {
    private static final String dateFormat = "yyyy-MM-dd HH:mm:ss";
    private static final String shortDateFormat = "yyyy-MM-dd";
    @Override
    public Date convert(String value) {
        if(StringUtils.isEmpty(value)) {
            return null;
        }
        value = value.trim();
        try {
            if(value.contains("-")) {
                SimpleDateFormat formatter;
                if(value.contains(":")) {
                    formatter = new SimpleDateFormat(dateFormat);
                }else {
                    formatter = new SimpleDateFormat(shortDateFormat);
                }
                Date dtDate = formatter.parse(value);
                return dtDate;
            }else if(value.matches("^\\d+$")) {
                Long lDate = new Long(value);
                return new Date(lDate);
            }
        } catch (Exception e) {
            throw new RuntimeException(String.format("parser %s to Date fail", value));
        }
        throw new RuntimeException(String.format("parser %s to Date fail", value));
    }
}
package cn.g4b.fm.configuration;
import cn.g4b.fm.common.util.LogUtils;
import org.apache.shiro.session.SessionException;
import org.apache.shiro.subject.Subject;
import org.apache.shiro.web.filter.authc.LogoutFilter;
import org.apache.shiro.web.util.WebUtils;
import org.slf4j.Logger;
import org.springframework.stereotype.Service;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import java.util.Locale;
public class SystemLogout extends LogoutFilter {
    private Logger log = LogUtils.getPlatformLogger();
    @Override
    protected boolean preHandle(ServletRequest request, ServletResponse response) throws
Exception {
        Subject subject = getSubject(request, response);
        // Check if POST only logout is enabled
        if (isPostOnlyLogout()) {
            // check if the current request's method is a POST, if not redirect
            if
(!WebUtils.toHttp(request).getMethod().toUpperCase(Locale.ENGLISH).equals("POST")) {
                return onLogoutRequestNotAPost(request, response);
```

```
                }
            }
            String redirectUrl = getRedirectUrl(request, response, subject);
            //try/catch added for SHIRO-298:
            try {
                subject.logout();
            } catch (SessionException ise) {
                log.debug("Encountered session exception during logout.  This can generally
safely be ignored.", ise);
            }
            issueRedirect(request, response, redirectUrl);
            return false;
        }
}
package cn.g4b.fm.configuration;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.convert.support.GenericConversionService;
import org.springframework.web.bind.support.ConfigurableWebBindingInitializer;
import
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter;
import javax.annotation.PostConstruct;
@Configuration
public class WebConfigBeans {
        @Autowired
        private RequestMappingHandlerAdapter handlerAdapter;
        @PostConstruct
        public void initEditableAvlidation() {
            ConfigurableWebBindingInitializer                initializer                =
(ConfigurableWebBindingInitializer)handlerAdapter.getWebBindingInitializer();
            if(initializer.getConversionService()!=null) {
                GenericConversionService                genericConversionService                =
(GenericConversionService)initializer.getConversionService();
                genericConversionService.addConverter(new StringToDateConverter());
            }
        }
}
package cn.g4b.fm.configuration;
import cn.g4b.fm.common.model.ConstantDefinition;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import org.springframework.util.ClassUtils;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import java.io.File;
@Component
public class WebConfigurer extends WebMvcConfigurerAdapter {
        public static String uploadPath;
        @Override
        public void addResourceHandlers(ResourceHandlerRegistry registry) {
            String path = ClassUtils.getDefaultClassLoader().getResource("").getPath();
            uploadPath    =    System.getProperty("user.dir")+    File.separator+"uploaded_files"+
```

```
File.separator;
            File file = new File(uploadPath);
            if(!file.exists()){
                file.mkdir();
            }
            ConstantDefinition.UPLOAD_FILE_PATH = uploadPath;
            ConstantDefinition.ZIP_FILE_PATH = uploadPath + File.separator +
"temp"+File.separator;
            file = new File(ConstantDefinition.ZIP_FILE_PATH);
            if(!file.exists()){
                file.mkdir();
            }
            registry.addResourceHandler("/files/**").addResourceLocations("file:///"+uploadPath);
        }
}
package cn.g4b.fm.configuration;
import cn.g4b.fm.annotation.LoginAppResolver;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.method.support.HandlerMethodArgumentResolver;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurationSupport;
import java.util.List;
/**
 * @Auther: lzk
 * @Date: 2020/4/8 13:06
 * @Description:
 */
@Configuration
public class WebMvcConfig extends WebMvcConfigurationSupport {
    @Autowired
    private LoginAppResolver loginAppResolver;
    @Override
    public void addArgumentResolvers(List<HandlerMethodArgumentResolver>
argumentResolvers) {
        super.addArgumentResolvers(argumentResolvers);
        argumentResolvers.add(loginAppResolver);
    }
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        //配置静态资源处理
        registry.addResourceHandler("/**")
                .addResourceLocations("resources/", "static/", "public/",
                        "META-INF/resources/")
                .addResourceLocations("classpath:resources/", "classpath:static/",
                        "classpath:public/", "classpath:META-INF/resources/")
                .addResourceLocations("file:///tmp/webapps/");
    }
}
package cn.g4b.fm.controller.common;
import cn.g4b.fm.model.sys.SysCompany;
import cn.g4b.fm.model.sys.SysUser;
```

```
import javax.servlet.http.HttpSession;
import java.util.List;
/**
 * 定义 Controller 常用的方法
 */
public class BaseController {
	private final String currentCompany = "currentCompany";	//默认企业
	private final String companyList = "companyList";	//企业列表
	private final String LoginUser = "sysUser";	//默认用户
	/**
	 * 获取当前登陆的默认企业
	 * @param session
	 * @return
	 */
	public SysCompany getcurrentCompany(HttpSession session){
		Object obj = session.getAttribute("currentCompany");
		if(obj!=null)
			return (SysCompany) obj;
		return null;
	}
	/**
	 * 获取当前登陆用户的所属企业列表
	 * @param session
	 * @return
	 */
	public List<SysCompany> getCompanyList(HttpSession session){
		Object obj = session.getAttribute("companyList");
		if(obj!=null)
			return (List<SysCompany>) obj;
		return null;
	}
	/**
	 * 获取当前登陆的用户信息
	 * @param session
	 * @return
	 */
	public SysUser getLoginUser(HttpSession session){
		Object obj = session.getAttribute("sysUser");
		if(obj!=null)
			return (SysUser) obj;
		return null;
	}
}
package cn.g4b.fm.controller.common;
import cn.g4b.fm.common.model.ResultBean;
import cn.g4b.fm.configuration.WebConfigurer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
```

```
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;
/**
 * 文件上传 Controller
 */
@Controller
@RequestMapping(value = "/fileUpload")
public class FileUploadController {
    /**
     * 文件上传
     * @return
     */
    @RequestMapping(value = "upload")
    @ResponseBody
    public ResultBean uploadFile(@RequestParam("file") MultipartFile file){
        ResultBean resultBean = new ResultBean();
        if (file.isEmpty()) {
            resultBean.setCode(-1);
            resultBean.setMessage("上传失败，请选择文件");
            return resultBean;
        }
        String fileName = file.getOriginalFilename();
        fileName                                                                    =
UUID.randomUUID().toString()+fileName.substring(fileName.lastIndexOf("."));
        String filePath = WebConfigurer.uploadPath;
        File dest = new File(filePath + fileName);
        try {
            file.transferTo(dest);
            resultBean.setCode(0);
            resultBean.setMessage("上传成功");
            Map restMap = new HashMap();
            restMap.put("fileName",file.getOriginalFilename());
            restMap.put("filePath","files/"+fileName);
            resultBean.setData(restMap);
        } catch (IOException e) {
            e.printStackTrace();
            resultBean.setCode(-1);
            resultBean.setMessage(e.getMessage());
        }
        return resultBean;
    }
}
package cn.g4b.fm.controller.common;
import cn.g4b.fm.common.model.BizException;
import cn.g4b.fm.common.model.CommonEnum;
import cn.g4b.fm.common.model.ResultBody;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.ControllerAdvice;
```

```
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import javax.servlet.http.HttpServletRequest;
/**
 * @Description: 自定义全局异常处理类
 * 捕捉 RuntimeException 时参数中不能带 HttpServletRequest，否则会出错问题，目前也不知道是什么原因，Exception 是可以有这个参数的
 * @Author: imzk
 * @Date: 2019/11/4 17:34
 */
@ControllerAdvice
public class GlobalExceptionHandler {
    private static final Logger logger = LoggerFactory.getLogger(GlobalExceptionHandler.class);
    /**
     * 处理空指针的异常
     * @param req
     * @param e
     * @return
     */
    @ExceptionHandler(value = NullPointerException.class)
    @ResponseBody
    public ResultBody exceptionHandler(HttpServletRequest req, NullPointerException e){
        e.printStackTrace();
        logger.error("发生空指针异常，原因：",e);
        return ResultBody.error(CommonEnum.BODY_NOT_MATCH);
    }
    @ExceptionHandler(value = BizException.class)
    @ResponseBody
    public ResultBody exceptionHandler(BizException e){
        e.printStackTrace();
        return ResultBody.error(e.getErrorCode(),e.getMessage());
    }
    /**
     * 处理运行时异常
     * @return
     */
    @ResponseBody
    @ExceptionHandler(value = RuntimeException.class)
    public ResultBody exceptionHandler(RuntimeException e){
        e.printStackTrace();
        return ResultBody.error(CommonEnum.INTERNAL_SERVER_ERROR,e.getMessage());
    }
    /**
     * 处理其他异常
     * @param e
     * @return
     */
    @ExceptionHandler(value = Exception.class)
    @ResponseBody
    public ResultBody exceptionHandler(Exception e){
        logger.error("未知异常！原因是:",e);
        return ResultBody.error(CommonEnum.INTERNAL_SERVER_ERROR);
```

```
    }
}
package cn.g4b.fm.model.sys;
import java.io.Serializable;
import java.util.List;
/**
 * @author
 * cb_appinfo 第三方应用系统表
提供给用户 app_id 跟 hash_value
                                        -&#
 */
public class CbApp implements Serializable {
    private String appId;
    private String userId;
    /**
     * hash(app_id+app_code)
            用户登陆时进行校验
     */
    private String hashValue;
    private String appName;
    private String appDesc;
    /**
     * 随机的字符串，不提供给调用方
     */
    private String appCode;
    /**
     * 应用系统类型 0：第三方应用系统  1：签署箱应用
     */
    private Byte appType;
    /**
     * 回调地址
     */
    private String callbackUrl;
    /**
     * 登陆时间——用于计算 go-fastdfs auth_token
     */
    private Long loginTime;
    /**
     * 0：待审核
     * 1：上线
     * 2：下线
     */
    private Integer status;
    /** 是否自动派发
     * 0：否
     * 1：是
     *
     */
    private Integer isDeliver;
    public Integer getIsDeliver() {
        return isDeliver;
    }
```

```
public void setIsDeliver(Integer isDeliver) {
    this.isDeliver = isDeliver;
}
private String tokenId;
private List<String> apiList;
private static final long serialVersionUID = 1L;
public String getAppId() {
    return appId;
}
public void setAppId(String appId) {
    this.appId = appId;
}
public String getUserId() {
    return userId;
}
public void setUserId(String userId) {
    this.userId = userId;
}
public String getHashValue() {
    return hashValue;
}
public void setHashValue(String hashValue) {
    this.hashValue = hashValue;
}
public String getAppName() {
    return appName;
}
public void setAppName(String appName) {
    this.appName = appName;
}
public String getAppDesc() {
    return appDesc;
}
public void setAppDesc(String appDesc) {
    this.appDesc = appDesc;
}
public String getAppCode() {
    return appCode;
}
public void setAppCode(String appCode) {
    this.appCode = appCode;
}
public Byte getAppType() {
    return appType;
}
public void setAppType(Byte appType) {
    this.appType = appType;
}
public List<String> getApiList() {
    return apiList;
}
public void setApiList(List<String> apiList) {
```

```
        this.apiList = apiList;
    }
    public String getTokenId() {
        return tokenId;
    }
    public void setTokenId(String tokenId) {
        this.tokenId = tokenId;
    }
    public Long getLoginTime() {
        return loginTime;
    }
    public void setLoginTime(Long loginTime) {
        this.loginTime = loginTime;
    }
    public String getCallbackUrl() {
        return callbackUrl;
    }
    public void setCallbackUrl(String callbackUrl) {
        this.callbackUrl = callbackUrl;
    }
    public Integer getStatus() {
        return status;
    }
    public void setStatus(Integer status) {
        this.status = status;
    }
    @Override
    public boolean equals(Object that) {
        if (this == that) {
            return true;
        }
        if (that == null) {
            return false;
        }
        if (getClass() != that.getClass()) {
            return false;
        }
        CbApp other = (CbApp) that;
        return (this.getAppId() == null ? other.getAppId() == null :
this.getAppId().equals(other.getAppId()))
                && (this.getUserId() == null ? other.getUserId() == null :
this.getUserId().equals(other.getUserId()))
                && (this.getHashValue() == null ? other.getHashValue() == null :
this.getHashValue().equals(other.getHashValue()))
                && (this.getAppName() == null ? other.getAppName() == null :
this.getAppName().equals(other.getAppName()))
                && (this.getAppDesc() == null ? other.getAppDesc() == null :
this.getAppDesc().equals(other.getAppDesc()))
                && (this.getAppCode() == null ? other.getAppCode() == null :
this.getAppCode().equals(other.getAppCode()))
                && (this.getAppType() == null ? other.getAppType() == null :
this.getAppType().equals(other.getAppType()));
```

```
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getAppId() == null) ? 0 : getAppId().hashCode());
        result = prime * result + ((getUserId() == null) ? 0 : getUserId().hashCode());
        result = prime * result + ((getHashValue() == null) ? 0 : getHashValue().hashCode());
        result = prime * result + ((getAppName() == null) ? 0 : getAppName().hashCode());
        result = prime * result + ((getAppDesc() == null) ? 0 : getAppDesc().hashCode());
        result = prime * result + ((getAppCode() == null) ? 0 : getAppCode().hashCode());
        result = prime * result + ((getAppType() == null) ? 0 : getAppType().hashCode());
        result = prime * result + ((getCallbackUrl() == null) ? 0 : getCallbackUrl().hashCode());
        return result;
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(getClass().getSimpleName());
        sb.append(" [");
        sb.append("Hash = ").append(hashCode());
        sb.append(", appId=").append(appId);
        sb.append(", userId=").append(userId);
        sb.append(", hashValue=").append(hashValue);
        sb.append(", appName=").append(appName);
        sb.append(", appDesc=").append(appDesc);
        sb.append(", appCode=").append(appCode);
        sb.append(", appType=").append(appType);
        sb.append(", callbackUrl=").append(callbackUrl);
        sb.append(", serialVersionUID=").append(serialVersionUID);
        sb.append("]");
        return sb.toString();
    }
}
package cn.g4b.fm.model.sys;
public class SysArea {
    private Integer id;
    private Integer parentId;
    private String code;
    private String name;
    private Integer sort;
    public SysArea() {
    }
    public SysArea(Integer parentId, String code, String name, Integer sort) {
        this.parentId = parentId;
        this.code = code;
        this.name = name;
        this.sort = sort;
    }
    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }
    public Integer getParentId() {
        return parentId;
    }
    public void setParentId(Integer parentId) {
        this.parentId = parentId;
    }
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code == null ? null : code.trim();
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name == null ? null : name.trim();
    }
    public Integer getSort() {
        return sort;
    }
    public void setSort(Integer sort) {
        this.sort = sort;
    }
}
package cn.g4b.fm.model.sys;
import com.fasterxml.jackson.annotation.JsonFormat;
import java.util.Date;
public class SysCompany {
    private Integer id;
    private String compName;
    private String address;
    private String legalPerson;
    private String orgNo;
    private String remark;
    private Integer status;
    @JsonFormat(pattern = "yyyy-MM-dd",timezone = "GMT+08")
    private Date createTime;
    private Integer check;
    private String pubKey;
    private String priKey;
    private String certInfo;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getCompName() {
```

```
        return compName;
    }
    public void setCompName(String compName) {
        this.compName = compName == null ? null : compName.trim();
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address == null ? null : address.trim();
    }
    public String getLegalPerson() {
        return legalPerson;
    }
    public void setLegalPerson(String legalPerson) {
        this.legalPerson = legalPerson == null ? null : legalPerson.trim();
    }
    public String getOrgNo() {
        return orgNo;
    }
    public void setOrgNo(String orgNo) {
        this.orgNo = orgNo == null ? null : orgNo.trim();
    }
    public String getRemark() {
        return remark;
    }
    public void setRemark(String remark) {
        this.remark = remark == null ? null : remark.trim();
    }
    public Integer getStatus() {
        return status;
    }
    public void setStatus(Integer status) {
        this.status = status;
    }
    public Date getCreateTime() {
        return createTime;
    }
    public void setCreateTime(Date createTime) {
        this.createTime = createTime;
    }
    public Integer getCheck() {
        return check;
    }
    public void setCheck(Integer check) {
        this.check = check;
    }
    public String getPubKey() {
        return pubKey;
    }
    public void setPubKey(String pubKey) {
        this.pubKey = pubKey;
```

```
	}
	public String getPriKey() {
		return priKey;
	}
	public void setPriKey(String priKey) {
		this.priKey = priKey;
	}
	public String getCertInfo() {
		return certInfo;
	}
	public void setCertInfo(String certInfo) {
		this.certInfo = certInfo;
	}
}
package cn.g4b.fm.model.sys;
import java.util.List;
public class SysMenu {
	private Integer id;
	private Integer parentId;
	private String menuName;
	private String menuPath;
	private String permission;
	private Integer orderby;
	private Integer status;
	private Integer check;
	private List<SysMenu> children;
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public Integer getParentId() {
		return parentId;
	}
	public void setParentId(Integer parentId) {
		this.parentId = parentId;
	}
	public String getMenuName() {
		return menuName;
	}
	public void setMenuName(String menuName) {
		this.menuName = menuName == null ? null : menuName.trim();
	}
	public String getMenuPath() {
		return menuPath;
	}
	public void setMenuPath(String menuPath) {
		this.menuPath = menuPath == null ? null : menuPath.trim();
	}
	public String getPermission() {
		return permission;
```

```
	}
	public void setPermission(String permission) {
		this.permission = permission == null ? null : permission.trim();
	}
	public Integer getOrderby() {
		return orderby;
	}
	public void setOrderby(Integer orderby) {
		this.orderby = orderby;
	}
	public Integer getStatus() {
		return status;
	}
	public void setStatus(Integer status) {
		this.status = status;
	}
	public List<SysMenu> getChildren() {
		return children;
	}
	public void setChildren(List<SysMenu> children) {
		this.children = children;
	}
	public Integer getCheck() {
		return check;
	}
	public void setCheck(Integer check) {
		this.check = check;
	}
}
package cn.g4b.fm.model.sys;
import java.io.Serializable;
/**
 * @author
 *
 */
public class SysRole implements Serializable {
	private Integer roleId;
	private String roleName;
	/**
	 * 角色状态 0：启动 2：禁用
	 */
	private Integer status;
	private boolean check;
	private static final long serialVersionUID = 1L;
	public Integer getRoleId() {
		return roleId;
	}
	public void setRoleId(Integer roleId) {
		this.roleId = roleId;
	}
	public String getRoleName() {
		return roleName;
```

```
    }
    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }
    public Integer getStatus() {
        return status;
    }
    public void setStatus(Integer status) {
        this.status = status;
    }
    @Override
    public boolean equals(Object that) {
        if (this == that) {
            return true;
        }
        if (that == null) {
            return false;
        }
        if (getClass() != that.getClass()) {
            return false;
        }
        SysRole other = (SysRole) that;
        return (this.getRoleId() == null ? other.getRoleId() == null :
this.getRoleId().equals(other.getRoleId()))
            && (this.getRoleName() == null ? other.getRoleName() == null :
this.getRoleName().equals(other.getRoleName()))
            && (this.getStatus() == null ? other.getStatus() == null :
this.getStatus().equals(other.getStatus()));
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getRoleId() == null) ? 0 : getRoleId().hashCode());
        result = prime * result + ((getRoleName() == null) ? 0 : getRoleName().hashCode());
        result = prime * result + ((getStatus() == null) ? 0 : getStatus().hashCode());
        return result;
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(getClass().getSimpleName());
        sb.append(" [");
        sb.append("Hash = ").append(hashCode());
        sb.append(", roleId=").append(roleId);
        sb.append(", roleName=").append(roleName);
        sb.append(", status=").append(status);
        sb.append(", serialVersionUID=").append(serialVersionUID);
        sb.append("]");
        return sb.toString();
    }
    public boolean isCheck() {
```

```
        return check;
    }
    public void setCheck(boolean check) {
        this.check = check;
    }
}
package cn.g4b.fm.model.sys;
import java.io.Serializable;
/**
 * @author
 *
 */
public class SysRoleMenu implements Serializable {
    private Integer roleId;
    private Integer menuId;
    private static final long serialVersionUID = 1L;
    public Integer getRoleId() {
        return roleId;
    }
    public void setRoleId(Integer roleId) {
        this.roleId = roleId;
    }
    public Integer getMenuId() {
        return menuId;
    }
    public void setMenuId(Integer menuId) {
        this.menuId = menuId;
    }
    @Override
    public boolean equals(Object that) {
        if (this == that) {
            return true;
        }
        if (that == null) {
            return false;
        }
        if (getClass() != that.getClass()) {
            return false;
        }
        SysRoleMenu other = (SysRoleMenu) that;
        return (this.getRoleId() == null ? other.getRoleId() == null :
this.getRoleId().equals(other.getRoleId()))
                && (this.getMenuId() == null ? other.getMenuId() == null :
this.getMenuId().equals(other.getMenuId()));
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getRoleId() == null) ? 0 : getRoleId().hashCode());
        result = prime * result + ((getMenuId() == null) ? 0 : getMenuId().hashCode());
        return result;
```

```
    }
    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append(getClass().getSimpleName());
        sb.append(" [");
        sb.append("Hash = ").append(hashCode());
        sb.append(", roleId=").append(roleId);
        sb.append(", menuId=").append(menuId);
        sb.append(", serialVersionUID=").append(serialVersionUID);
        sb.append("]");
        return sb.toString();
    }
}
```