WORKFLOW-沃客福勒

企业软件研发平台

（简称：沃客福勒 v1.1）

源代码

```
package com.windowdb.wms;
import java.io.PrintStream;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.ServletComponentScan;
import org.springframework.cache.annotation.EnableCaching;
@EnableCaching
@ServletComponentScan
@SpringBootApplication
public class EnterpriseWindowDataBaseApplication {
    public static void main(String[] args) {
        PrintStream out = System.out;
        if (null != args)
            for (String a : args)
                out.println(a);
        SpringApplication.run(EnterpriseWindowDataBaseApplication.class, args);
        out.println("启动完毕");
    }
}
package window.database;
import java.io.File;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.commons.io.FileUtils;
public class 生成授权文件 {
    public static void main(String[] args) throws Exception {
        new 生成授权文件().init();
    }
    public void init() throws Exception {
        String basepath = "/Volumes/springboot.projects/window.database/src/";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        resource = java.util.ResourceBundle.getBundle("license", new java.util.Locale("zh",
"CN"));
        String m = resource.getString("M");
        Date date1 = sdf.parse(resource.getString("D"));
        Date date2 = new Date();
        Long day = (date1.getTime() - date2.getTime()) / (24 * 60 * 60 * 1000);
        day = Math.abs(day) + 1;
        /**
         * banner 文件中的注册信息
         */
```

```
        banner: {
            File bannerF = new File(basepath, "main/resources/banner.txt");
            List<String> lines = FileUtils.readLines(bannerF, "utf-8");
            StringBuffer data = new StringBuffer();
            for (String line : lines) {
                if (line.startsWith("授予用户：")) {
                    data.append("授予用户：").append(resource.getString("C")).append("  (已
绑定)\n");
                } else if (line.startsWith("访问地址：")) {
                    data.append("访问地址：").append(resource.getString("W")).append("
(已绑定)\n");
                } else if (line.startsWith("物理地址：")) {
                    data.append("物理地址：").append(resource.getString("A")).append("  (已
绑定)\n");
                } else if (line.trim().startsWith("base: WF Platform v")) {
                    data.append("      base: WF Platform v 1.1.").append(sdf.format(new
Date()).replace("-", "")).append("\n");
                } else if (line.trim().startsWith("core: WorkFlow v")) {
                    data.append("                              core:          WorkFlow          v
1.1.").append(sdf.format(date1).replace("-", "")).append(".").append(day).append("\n");
                } else {
                    data.append(line).append("\n");
                }
            }
            // 写入文件
            FileUtils.writeStringToFile(bannerF, data.toString(), "utf-8");
        }
        /**
         * 计算有效天数 给 ant 使用
         */
        validate: {
            try {
                File li = new File(basepath, "test/java/license.properties");
                File tr = new File(basepath, "main/resources/license.properties");
                if (!li.exists() || !tr.exists()) {
                    System.out.println("license 文件不存在! ");
                    System.exit(3);
                }
                List<String> lines = FileUtils.readLines(li, "utf-8");
                StringBuffer data = new StringBuffer();
                for (String line : lines) {
                    if (null == line || line.trim().startsWith("#") || line.trim().isEmpty()) {
                        continue;
                    }
                    if (line.trim().startsWith("V=")) {
                        data.append("V=").append(String.valueOf(day)).append("\n");
```

```
                    } else {
                        data.append(line).append("\n");
                    }
                }
                // 写入文件
                FileUtils.writeStringToFile(tr, data.toString(), "utf-8");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        // 刷新对象数据
        reflash: {
            com.windowdb.javascript.c.init();
            com.windowdb.wms.c.init();
            com.windowdb.wms.exception.c.init();
            com.windowdb.utils.c.init();
            com.windowdb.wms.service.c.init();
            com.windowdb.wms.dao.c.init();
            com.windowdb.wms.controller.c.init();
        }
        resource = java.util.ResourceBundle.getBundle("license", new java.util.Locale("zh",
"CN"));
        sb = new StringBuffer();
        sb.append(com.windowdb.javascript.c.getInstance().toString());
        sb.append(com.windowdb.utils.c.getInstance().toString());
        sb.append(com.windowdb.wms.c.getInstance().toString());
        sb.append(com.windowdb.wms.service.c.getInstance().toString());
        sb.append(com.windowdb.wms.dao.c.getInstance().toString());
        sb.append(com.windowdb.wms.controller.c.getInstance().toString());
        sb.append(com.windowdb.wms.exception.c.getInstance().toString());
        sb.append(m);
        // 生产软件 license
        license: {
            root = com.windowdb.utils.a.toMD5(sb.toString());
            String license = com.windowdb.utils.e.encode(sb.toString(), root);
            System.setProperty("L", license);
            info1 = com.windowdb.utils.a.toMD5(root.concat(license));
            info2                                                                    =
com.windowdb.utils.a.toMD5(root.concat(license).concat(info1).concat(sb.toString()));
            // 将 license (L) 写回 license.properties 文件
            File licenseFile = new File(basepath, "main/resources/license.properties");
            if (!licenseFile.exists()) {
                System.out.println("license 文件不存在! ");
                System.exit(3);
            }
            List<String> lines = FileUtils.readLines(licenseFile, "utf-8");
```

```
            sb = new StringBuffer();
            for (String line : lines) {
                if (null == line || line.trim().isEmpty())
                    continue;
                if (line.trim().replaceAll(" ", "").startsWith("L="))
                    sb.append("L=").append(license).append("\n");
                else
                    sb.append(line).append("\n");
            }
            FileUtils.writeStringToFile(licenseFile, sb.toString(), "utf-8");
            // 将 license (L) 写到 javascript 包的 messages_zh_CN.properties 文件
            licenseFile = new File(basepath,
"test/java/com/windowdb/javascript/messages_zh_CN.properties");
            if (!licenseFile.exists()) {
                System.out.println("messages 文件不存在! ");
                System.exit(3);
            }
            lines = FileUtils.readLines(licenseFile, "utf-8");
            sb = new StringBuffer();
            for (String line : lines) {
                if (null == line)
                    continue;
                else if (line.trim().replaceAll(" ", "").startsWith("package-info.0="))
                    sb.append("package-info.0=").append(info1).append("\n");
                else if (line.trim().replaceAll(" ", "").startsWith("package-info.1="))
                    sb.append("package-info.1=").append(license).append("\n");
                else
                    sb.append(line).append("\n");
            }
            FileUtils.writeStringToFile(licenseFile, sb.toString(), "utf-8");
        }
        // 刷新对象数据
        reflash: {
            com.windowdb.javascript.c.init();
            com.windowdb.wms.c.init();
            com.windowdb.wms.exception.c.init();
            com.windowdb.utils.c.init();
            com.windowdb.wms.service.c.init();
            com.windowdb.wms.dao.c.init();
            com.windowdb.wms.controller.c.init();
        }
        System.out.print(info1);
        deonecode(basepath, com.windowdb.javascript.c.class,
com.windowdb.javascript.c.getInstance());
        decode(basepath, com.windowdb.wms.c.class, com.windowdb.wms.c.getInstance());
        decode(basepath, com.windowdb.wms.exception.c.class,
```

```
com.windowdb.wms.exception.c.getInstance());
        decode(basepath, com.windowdb.utils.c.class, com.windowdb.utils.c.getInstance());
        decode(basepath, com.windowdb.wms.service.c.class,
com.windowdb.wms.service.c.getInstance());
        decode(basepath, com.windowdb.wms.dao.c.class,
com.windowdb.wms.dao.c.getInstance());
        decode(basepath, com.windowdb.wms.controller.c.class,
com.windowdb.wms.controller.c.getInstance());
    }
    private void deonecode(String basepath, Class<?> clz, com.windowdb.wms.b u) throws
Exception {
        File msg = getMessageFile(basepath, "test/java/" + clz.getPackage().getName());
        if (!msg.exists()) {
            System.out.println("SpringBootDemoApplicationTests.deonecode()1");
            System.out.println("文件不存在! ".concat(msg.getAbsolutePath()));
            System.exit(3);
        }
        String filepath = msg.getParent();
        if (null != basepath) {
            File f = new File(basepath, "test/java/" +
clz.getPackage().getName().replaceAll("\\.", "/"));
            filepath = f.getAbsolutePath();
            if (!f.exists()) {
                System.out.println("SpringBootDemoApplicationTests.deonecode()2");
                System.out.println("文件不存在! ".concat(f.getAbsolutePath()));
                System.exit(3);
            }
        }
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        resource = java.util.ResourceBundle.getBundle("license", new java.util.Locale("zh",
"CN"));
        String time = String.valueOf(sdf.parse(resource.getString("D")).getTime());
        String cm =
com.windowdb.utils.a.toMD5(com.windowdb.utils.a.toMD5(com.windowdb.utils.a.toMD5(com.
windowdb.utils.a.toMD5(clz.getPackage().getName()))));
        final List<String> lines = FileUtils.readLines(msg, "utf-8");
        Map<Long, String> map = new HashMap<Long, String>();
        String r0 = null;
        for (String line : lines) {
            if (line.indexOf("=") == -1)
                continue;
            String[] sp = line.split("=");
            Long mapkey = Long.valueOf(sp[0].replace("package-info.", ""));
            String plainText = line.replaceFirst(sp[0].concat("="), "");
            String cipher = null;
            String filek = "";
```

```
            filek = sp[0].concat("=");
            if (sp[0].endsWith(".0")) {
                cipher = time + "P=" + plainText;
                r0 = cipher;
            } else {
                if (null == r0)
                    throw new Exception("顺序不对");
                cipher = time + "P=" + com.windowdb.utils.e.encode(plainText,
com.windowdb.utils.a.toMD5(r0.trim()));
            }
            if (null == cipher || filek.equals(""))
                throw new Exception("加密失败");
            map.put(mapkey, filek.concat(cipher));
        }
        Long[] mapkeys = map.keySet().toArray(new Long[] {});
        Arrays.sort(mapkeys);
        StringBuffer newfc = new StringBuffer();
        for (Long k : mapkeys) {
            String line = map.get(k);
            newfc.append(line).append("\n");
            // System.out.println(line);
        }
        String[] ns = msg.getName().split("\\.");
        File newf = new File(filepath.replace("/test/java/", "/main/resources/"),
ns[0].replaceAll("_zh_CN", "").concat(".").concat(ns[1]));
        System.out.println("创建新文件：" + newf.getAbsolutePath());
        if (newf.exists()) {
            Boolean b = FileUtils.deleteQuietly(newf);
            if (b) {
                System.out.println("2,成功清理历史文件" + newf.getAbsolutePath());
            }
        }
        FileUtils.writeStringToFile(newf, newfc.toString(), "utf-8");
        Thread.sleep(3000);
    }
    private void decode(String basepath, Class<?> clz, com.windowdb.wms.b u) throws
Exception {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        resource = java.util.ResourceBundle.getBundle("license", new java.util.Locale("zh",
"CN"));
        String time = String.valueOf(sdf.parse(resource.getString("D")).getTime());
        File msg = getMessageFile(basepath, "test/java/" + clz.getPackage().getName());
        if (!msg.exists()) {
            System.out.println("SpringBootDemoApplicationTests.decode()1");
            System.out.println("文件不存在！ ".concat(msg.getAbsolutePath()));
            System.exit(3);
```

```
        }

        String filepath = msg.getParent();
        if (null != basepath) {
            filepath = new File(basepath, "test/java/" +
clz.getPackage().getName().replaceAll("\\.", "/")).getAbsolutePath();
        }
        final List<String> lines = FileUtils.readLines(msg, "utf-8");
        Map<Long, String> map = new HashMap<Long, String>();
        for (String line : lines) {
            if (line.indexOf("=") == -1)
                continue;
            String[] sp = line.split("=");
            Long mapkey = Long.valueOf(sp[0].replace("package-info.", ""));
            String plainText = line.replaceFirst(sp[0].concat("="), "");
            String cipher = null;
            String filek = "";
            if (plainText.getBytes().length > 110) {
                filek = sp[0].concat("=").concat(time).concat("A");
                String key = com.windowdb.utils.a.toMD5(u.toString()).trim();
                cipher = com.windowdb.utils.e.encode(plainText, key);
            } else {
                filek = sp[0].concat("=").concat(time).concat("P");
                String key =
com.windowdb.utils.a.toMD5(com.windowdb.utils.a.toMD5(u.toString().toUpperCase())).trim
();
                cipher = com.windowdb.utils.e.encode(plainText, key);
            }
            if (null == cipher || filek.equals(""))
                throw new Exception("加密失败");
            if (3306 == mapkey) {
                if (clz.getName().indexOf("javascript") > -1) {
                    cipher = sp[1];
                }
            }
            map.put(mapkey, filek.concat("=").concat(cipher));
        }
        Long[] mapkeys = map.keySet().toArray(new Long[] {});
        Arrays.sort(mapkeys);
        StringBuffer newfc = new StringBuffer();
        for (Long k : mapkeys) {
            String line = map.get(k);
            newfc.append(line).append("\n");
            // System.out.println(line);
        }
        String[] ns = msg.getName().split("\\.");
```

```
        File newf = new File(filepath.replace("/test/java/", "/main/resources/"),
ns[0].replaceAll("_zh_CN", "").concat(".").concat(ns[1]));
        System.out.println("创建新文件：" + newf.getAbsolutePath());
        if (newf.exists()) {
            Boolean b = FileUtils.deleteQuietly(newf);
            if (b) {
                System.out.println("2,成功清理历史文件" + newf.getAbsolutePath());
            }
        }
        if (!newf.getParentFile().exists()) {
            Boolean b = newf.getParentFile().mkdirs();
            if (b) {
                System.out.println("目录创建失败。" + newf.getParent());
            }
        }
        FileUtils.writeStringToFile(newf, newfc.toString(), "utf-8");
        Thread.sleep(3000);
    }
    private File getMessageFile(String basepath, String u) {
        String filepath = new File(basepath, u.replaceAll("\\.", "/")).getAbsolutePath();
        File f = new File(filepath, "messages_zh_CN.properties");
        if (!f.getParentFile().exists()) {
            f.getParentFile().mkdirs();
        }
        return f;
    }

    /**
     * 资源配置文件
     */
    java.util.ResourceBundle resource;
    /**
     * 需要频繁拼接的字符串
     */
    StringBuffer sb;
    /*-
     * 根种子
     *
     * des
     *
     * key：4 次 md5 (c+n+p+v+d+w+"企业数据库....")    作为 license 的 key
     *
     * 用于计算得到 license key
     *
     */
    String root;
```

```
    /*-
     * 1 级加密种子
     * des
     *
     * 4 次 md5(license+根种子)
     *
     */
    String info1;
    /*-
     * 2 级加密种子 4 次 md5(license+1 级种子)    变量文件名
     */
    String info2;
}
package window.database;
import java.io.File;
import java.io.IOException;
import java.util.Map;
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.io.FileUtils;
public class 水印 {
    private static String p = null;
    private static Map<String, String> m = new java.util.HashMap<String, String>();
    public static void main(String[] args) throws Exception {
        String sy =
"/Volumes/springboot.projects/WindowDB-master/src/main/resources/com/windowdb/wms/
service/sup/messages.properties";
        String s = "/Volumes/springboot.projects/WindowDB-master/src/main";
        java.util.ResourceBundle resource = java.util.ResourceBundle.getBundle("license",
                new java.util.Locale("zh", "CN"));
        File root = new File(s);
        cn(root, resource.getString("L"));
        if (m.isEmpty()) {
            System.exit(2);
        }
        StringBuffer sb = new StringBuffer();
        String[] ks = m.keySet().toArray(new String[] {});
        for (String k : ks) {
            sb.append(k);
            sb.append("=");
            sb.append(m.get(k).split("_")[0]);
            sb.append("\n");
        }
        sb.append(com.windowdb.utils.a.toMD5("l"));
        sb.append("=");
        sb.append(com.windowdb.utils.e.encode("" + m.size(),
com.windowdb.utils.a.toMD5("l")));
```

```
        File file = new File(sy);
        if (!file.getParentFile().exists()) {
            file.getParentFile().mkdirs();
        }
        try {
            FileUtils.writeStringToFile(file, sb.toString(), "utf-8");
        } catch (IOException e) {
            e.printStackTrace();
        }
        if (!file.exists()) {
                System.exit(2);
        }
    }
    private static void cn(File file, String key) {
        if (!file.exists()) {
            return;
        }
        if (null == p)
            p = file.getAbsolutePath();
        if (file.isDirectory()) {
            for (File f : file.listFiles()) {
                cn(f, key);
            }
        } else {
            try {
                String v = DigestUtils.md5Hex(FileUtils.readFileToByteArray(file));
                v = v.concat("_").concat(file.getAbsolutePath());
    m.put(com.windowdb.utils.a.toMD5(com.windowdb.utils.e.encode(file.getAbsolutePath().
replace(p, ""), key)), v);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
package window.database;
import java.io.PrintStream;
import io.xjar.boot.XBoot;
public class Mi {
    public static void main(String[] args) throws Exception {
        PrintStream out = System.out;
        out.println("开始加密...");
        String password = "workf";
        String                                    src                                    =
"/Volumes/springboot.projects/window.database/target/window-database-2.0.jar";
        String tar;
```

```
        tar = "/Users/colin/Desktop/workf.jar";
        for (String s : args) {
            if (s.startsWith("src=")) {
                src = s.replaceFirst("src=", "");
            }
            if (s.startsWith("tar=")) {
                tar = s.replaceFirst("tar=", "");
            }
        }
        XBoot.encrypt(src, tar, password, (entry) -> {
            String name = entry.getName();
            return (name.startsWith("com/windowdb") && name.endsWith(".class"));
        });
        out.print("软件打包完成，jar 包路径：");
        out.println(tar);
        out.print("软件包已经加密，启动密码是：");
        out.print(password);
        out.println("，请牢记启动密码！");
    }
}
package com.windowdb.wms.dao;
import java.io.PrintStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.ResourceBundle;
import java.util.WeakHashMap;
import javax.annotation.Resource;
import org.apache.commons.lang.StringUtils;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcDaoSupport;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
import org.springframework.stereotype.Repository;
import com.windowdb.javascript.b;
import net.sf.ehcache.Cache;
import net.sf.ehcache.Element;
import net.sf.json.JSONObject;
/**
 * 基本实体类 *
 * @author Midas
 *
 */
```

```
@SuppressWarnings({ "deprecation", "unchecked" })
@Repository
public class AbstractedDao {
    private final static String LICENSE = "license";
    @Resource
    private JdbcTemplate jdbcTemplate;
    private ResourceBundle resource = null;
    private String[] vn = null;
    public String[] getVn() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        if (null == vn) {
            vn = this.resource.getString(Const.M).split(Const.DOUHAOSTR.trim());
        }
        return vn;
    }
    public String getC() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.C.trim());
    }
    public String getW() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.W.trim());
    }
    public String getD() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.D.trim());
    }
    public String getL() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.L.trim());
    }
    public String getV() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.V.trim());
    }
    public String getP() {
        if (null == resource)
            resource = java.util.ResourceBundle.getBundle(LICENSE);
        return this.resource.getString(Const.P.trim());
    }
```

```
public String getN() {
    if (null == resource)
        resource = java.util.ResourceBundle.getBundle(LICENSE);
    return this.resource.getString(Const.N.trim());
}
static {
    System.setProperty("net.sf.ehcache.enableShutdownHook", "true");
}
protected SimpleDateFormat dateFormat = null;
protected SimpleDateFormat dateTimeFormat = null;
public boolean isreg() {
    return b.ready();
}
public SimpleDateFormat getDateFormat() {
    if (null == dateFormat)
        dateFormat = new SimpleDateFormat(Const.YYYYMMDD);
    return dateFormat;
}
public void setDateFormat(SimpleDateFormat dateFormat) {
    this.dateFormat = dateFormat;
}
public SimpleDateFormat getDateTimeFormat() {
    if (null == dateTimeFormat)
        dateTimeFormat = new SimpleDateFormat(Const.YYYYMMDDHHMMSS);
    return dateTimeFormat;
}
public void setDateTimeFormat(SimpleDateFormat dateTimeFormat) {
    this.dateTimeFormat = dateTimeFormat;
}
public JdbcTemplate getJdbcTemplate() {
    return jdbcTemplate;
}
public void setJdbcTemplate(JdbcTemplate p1) {
    this.jdbcTemplate = p1;
}
public Number insertLink(String t, String l, String r) throws Exception {
    return this.update(String.format(Const.INSERTLINK, new Object[] { t, l, r }));
}
/**
 * insert 数据并且返回 id
 *
 * @param sql
 * @param p
 * @return
 * @throws Exception
 */
```

```
    public Number insertBackKey(String p) throws Exception {
        return this.insertBackKey(p, null);
    }
    /**
     * insert 数据并且返回 id *
     * @param s
     * @param p
     * @return
     * @throws Exception
     */
    public Number insertBackKey(String s, Map<String, Object> p) throws Exception {
        if (s.indexOf(Const.PRINT_SQL) > -1) {
            PrintStream out = System.out;
            out.println();
            out.println();
            out.println(s);
            out.println();
            out.println();
        }
        Date now = new Date();
        if (null == p)
            p = new WeakHashMap<String, Object>();
        if (null != p.get(Const.ID))
            p.put(Const.ID, null);
        p.put(Const.INSERTTIME, now);
        p.put(Const.LASTWRITETIME, now);
        KeyHolder keyHolder = new GeneratedKeyHolder();
        NamedParameterJdbcDaoSupport np = new NamedParameterJdbcDaoSupport();
        np.setJdbcTemplate(jdbcTemplate);
        np.getNamedParameterJdbcTemplate().update(s,  new  MapSqlParameterSource(p),
keyHolder);
        return keyHolder.getKey();
    }
    /**
     * insert update delete  *
     * @param s
     * @return
     * @throws Exception
     */
    public Number update(String s) throws Exception {
        return update(s, null);
    }
    /**
     * insert update delete  *
     * @param p
     * @param p1
```

```
     * @return
     * @throws Exception
     */
    public Number update(String p, Map<String, Object> p1) throws Exception {
        if (null != p1)
            p1.put(Const.LASTWRITETIME, new Date());
        if (p.indexOf(Const.PRINT_SQL) > -1) {
            PrintStream out = System.out;
            out.println(Const.SPLIT_LINE);
            out.println();
            out.println(p);
            out.println();
            out.println();
            if (null != p1) {
                out.println();
                out.println();
                out.println(JSONObject.fromObject(p1));
                out.println();
                out.println();
            }
            out.println();
            out.println(Const.SPLIT_LINE);
        }
        NamedParameterJdbcDaoSupport np = new NamedParameterJdbcDaoSupport();
        np.setJdbcTemplate(jdbcTemplate);
        return np.getNamedParameterJdbcTemplate().update(p, null == p1 ? new
WeakHashMap<String, Object>() : p1);
    }
    /**
     * 从数据库中获取列表数据。 *
     * @param l
     * @return
     * @throws Exception
     */
    public List<Map<String, Object>> queryList(String l) throws Exception {
        return queryList(l, null);
    }
    /**
     * 从数据库中获取列表数据。 *
     * @param q
     * @param s
     * @return
     * @throws Exception
     */
    public List<Map<String, Object>> queryList(String q, Map<String, Object> s) throws
Exception {
```

```
        if (q.indexOf(Const.PRINT_SQL) > -1) {
            PrintStream out = System.out;
            out.println(Const.SPLIT_LINE);
            out.println();
            out.println(q);
            out.println();
            out.println();
            if (null != s) {
                out.println();
                out.println();
                out.println(JSONObject.fromObject(s));
                out.println();
                out.println();
            }
            out.println();
            out.println(Const.SPLIT_LINE);
        }
        NamedParameterJdbcDaoSupport np = new NamedParameterJdbcDaoSupport();
        np.setJdbcTemplate(jdbcTemplate);
        return np.getNamedParameterJdbcTemplate().queryForList(q, null == s ? new
WeakHashMap<String, Object>() : s);
    }
    /**
     * 从数据库中获取列表数据。 *
     * @param sql
     * @return
     * @throws Exception
     */
    public Map<String, Object> queryLimit(Map<String, Object> p) throws Exception {
        Map<String, Object> sqlParams = null;
        if (null == p || p.isEmpty())
            return null;
        if (null == p.get(Const.START))
            throw new RuntimeException(Const.PARAMSERROR);
        if (null == p.get(Const.LIMIT))
            throw new RuntimeException(Const.PARAMSERROR);
        if (null == p.get(Const.SQL))
            throw new RuntimeException(Const.PARAMSERROR);
        if (null != p.get(Const.PARAMS)) {
            if (!(p.get(Const.PARAMS) instanceof Map))
                throw new RuntimeException(Const.PARAMSERROR);
            sqlParams = (Map<String, Object>) p.get(Const.PARAMS);
        }
        String sql = p.get(Const.SQL).toString();
        Map<String, Object> map = new WeakHashMap<String, Object>();
        String totalSql = String.format((Const.TOTALSQL), new Object[] { sql });
```

```
        String listSql = String.format((Const.LIMITLISTSQL), new Object[] { sql,
p.get((Const.START)), p.get((Const.LIMIT)) });
        map.put(Const.TOTAL, this.queryLong(totalSql, sqlParams));
        map.put(Const.LIST, this.queryList(listSql, sqlParams));
        return map;
    }
    /**
     * 从数据库中获取单条数据。  *
     * @param s
     * @return
     * @throws Exception
     */
    public Map<String, Object> queryOne(String s) throws Exception {
        return queryOne(s, null);
    }
    /**
     * 从数据库中获取单条数据。  *
     * @param s
     * @param ps
     * @return
     * @throws Exception
     */
    public Map<String, Object> queryOne(String s, Map<String, Object> ps) throws
Exception {
        List<Map<String, Object>> list = queryList(s, ps);
        if (null == list || list.isEmpty())
            return null;
        else
            return list.get(0);
    }
    /**
     * 从数据库中获取单条数据的一个字符串。  *
     * @param s
     * @return
     * @throws Exception
     */
    public String queryString(String s) throws Exception {
        return queryString(s, null); }
    /**
     * 从数据库中获取单条数据的一个字符串。
     * @param s
     * @param p
     * @return
     * @throws Exception
     */
    public String queryString(String s, Map<String, Object> p) throws Exception {
```

```
        Object str = Const.EMPTY;
        Map<String, Object> map = this.queryOne(s, p);
        if (null != map && map.size() > 0)
            str = (null == (map.values().toArray()[0]) ? null :
(map.values().toArray()[0]).toString());
        return null == str ? Const.EMPTY : str.toString();
    }
    /**
     * 从数据库中获取单条数据的一个数字。
     * @param p
     * @return
     * @throws Exception
     */
    public Long queryLong(String p) throws Exception {
        return queryLong(p, null);
    }
    /**
     * 从数据库中获取单条数据的一个数字。
     * @param e
     * @param b
     * @return
     * @throws Exception
     */
    public Long queryLong(String e, Map<String, Object> b) throws Exception {
        List<Map<String, Object>> list = null;
        Map<String, Object> map = null;
        if (null == b || b.isEmpty()) {
            list = this.queryList(e);
        } else {
            list = this.queryList(e, b);
        }
        if (null == list || list.isEmpty())
            return null;
        map = list.get(0);
        if (null == map || map.isEmpty())
            return null;
        Long i = map.values().toArray(new Long[] {})[0];
        map = null;
        list = null;
        return i;
    }
    public Map<String, Object> queryPublicMenus() throws Exception {
        Map<String, Object> map = new HashMap<String, Object>();
        Cache cache = b.getCaManager().getCache(Const.DEFAULTMENUSSTR);
        Element e = cache.get(Const.MENUSSTR);
        if (null != e)
```

```
            return (Map<String, Object>) e.getValue();
        StringBuilder sql = new StringBuilder();
        sql.append((Const.MENUSPUBLIC));
        map.put(Const.MENUSSTR, this.queryList(sql.toString()));
        cache.put(new Element(Const.MENUSSTR, map));
        cache.flush();
        return map;
    }
    public Map<String, Object> queryMyMenus(Integer u) throws Exception {
        if (null == u)
            return null;
        Map<String, Object> map = new HashMap<String, Object>();
        Cache cache = b.getCaManager().getCache(Const.DEFAULTMENUSSTR);
        Element e = cache.get(u);
        if (null != e && null != e.getObjectKey()) {
            Map<String, Object> menus = ((Map<String, Object>) e.getObjectValue());
            List<Map<String, Object>> columns = (List<Map<String, Object>>)
menus.get(Const.COLUMNS);
            if (null != columns && !columns.isEmpty()) {
                return (Map<String, Object>) e.getObjectValue();
            }
        }
        List<Map<String, Object>> list;
        StringBuilder sql = new StringBuilder();
        sql.append(String.format((Const.MENUPREFIX), new Object[] { u, u }));
        String menuPrefix = this.queryString(sql.toString());
        map.put(Const.MENUPREFIXSTR, menuPrefix);
        sql.delete(0, sql.length());
        sql.append(String.format((Const.MENUS), new Object[] { u, u, menuPrefix, u }));
        list = this.queryList(sql.toString());
        map.put(Const.MENUSSTR, list);
        cache.put(new Element(u, map));
        cache.flush();
        return map;
    }
    public Map<String, Object> queryUserByToken(String t, Map<String, Object> u, Object l)
throws Exception {
        if (null == t)
            return null;
        Cache cache = b.getCaManager().getCache(Const.TOKENS);
        Element e;
        if (null != u) {
            e = new Element(t, u);
            if (null != l && Const.ONESTR.equals(l.toString()))
                e.setEternal(true);
            if (null != l && Const.ZSTR.equals(l.toString()))
```

```
                e.setEternal(false);
            cache.put(e);
            cache.flush();
            return u;
        }
        e = cache.get(t);
        if (null == e)
            return null;
        else if (null != l && Const.ONESTR.equals(l.toString())) {
            e.setEternal(true);
            cache.flush();
        } else if (null != l && Const.ZSTR.equals(l.toString())) {
            e.setEternal(false);
            cache.flush();
        }
        return (Map<String, Object>) e.getObjectValue();
    }
    public String queryList2String(String s) throws Exception {
        List<Map<String, Object>> list = this.queryList(s);
        if (list.isEmpty())
            return Const.EMPTY;
        StringBuffer sb = new StringBuffer();
        for (Map<String, Object> o : list) {
            if (o.isEmpty())
                continue;
            sb.append(StringUtils.join(o.values().toArray()));
        }
        return sb.toString();
    }
    public synchronized boolean granted(String v) {
        if (null == v)
            return false;
        String[] vn = this.getVn();
        if (null == vn)
            return false;
        for (String n : vn) {
            if (v.equalsIgnoreCase(n)) {
                return true;
            }
        }
        return false;
    }
}
package com.windowdb.wms.dao;
import java.util.List;
import java.util.Map;
```

```
import java.util.WeakHashMap;
import org.springframework.stereotype.Repository;
import com.windowdb.utils.a;
import com.windowdb.wms.exception.ExceptionNoId;
/**
 * @author 竹林春雨
 * @date 2017-01-04
 */
@Repository
public class WfInsTaskDao extends AbstractedDao {
    public List<Map<String, Object>> queryList(String field, Object value) throws Exception {
        String sql = String.format((Const.WF_INS_TASK_QUERYLIST), new Object[]
{ a.camelToUnderline(field), value });
        return super.queryList(sql);
    }
    public Map<String, Object> queryOne(int id) throws Exception {
        String sql = String.format((Const.WF_INS_TASK_QUERYONE), new Object[] { id });
        return super.queryOne(sql.toString());
    }
    public Number insertBackKey(Map<String, Object> map) throws Exception {
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.WF_DEF_ID.trim())) {
            columns.append(Const.WF_DEF_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_DEF_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.WF_NODE_ID.trim())) {
            columns.append(Const.WF_NODE_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_NODE_ID.trim()))).append(Const.DOUH
AOSTR);
        }
        if (null != map.get(Const.WF_INS_ID.trim())) {
            columns.append(Const.WF_INS_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_INS_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.APPLICANT_ID.trim())) {
            columns.append(Const.APPLICANT_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.APPLICANT_ID.trim()))).append(Const.DOUH
AOSTR);
        }
        if (null != map.get(Const.DONE.trim())) {
            columns.append(Const.DONE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.DONE.trim()))).append(Const.DOUHAOSTR);
        }
```

```
        if (null != map.get(Const.OFFICER_ID.trim())) {
            columns.append(Const.OFFICER_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.OFFICER_ID.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.NUM.trim())) {
            columns.append(Const.NUM_DOUHAO);
    values.append(a.escapeSql(map.get(Const.NUM.trim()))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.AGREE.trim())) {
            columns.append(Const.AGREE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.AGREE.trim()))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.ID.trim()))
            map.put(Const.ID, null);
        String title = (null == map.get(Const.TITLE.trim()) ? Const.EMPTY :
map.get(Const.TITLE.trim()).toString());
        columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE);
        values.append(Const._1_NOW_NOW).append(a.escapeSql(title));
        String sql = String.format((Const.WF_INS_TASK_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.insertBackKey(sql.toString(), map);
    }
    public Number insert(Map<String, Object> map) throws Exception {
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.WF_DEF_ID.trim())) {
            columns.append(Const.WF_DEF_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_DEF_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.WF_NODE_ID.trim())) {
            columns.append(Const.WF_NODE_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_NODE_ID.trim()))).append(Const.DOUH
AOSTR);
        }
        if (null != map.get(Const.WF_INS_ID.trim())) {
            columns.append(Const.WF_INS_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_INS_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.APPLICANT_ID.trim())) {
            columns.append(Const.APPLICANT_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.APPLICANT_ID.trim()))).append(Const.DOUH
AOSTR);
```

```
        }
        if (null != map.get(Const.DONE.trim())) {
            columns.append(Const.DONE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.DONE.trim()))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.OFFICER_ID.trim())) {
            columns.append(Const.OFFICER_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.OFFICER_ID.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.NUM.trim())) {
            columns.append(Const.NUM_DOUHAO);
    values.append(a.escapeSql(map.get(Const.NUM.trim()))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.AGREE.trim())) {
            columns.append(Const.AGREE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.AGREE.trim()))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.ID.trim()))
            map.put(Const.ID.trim(), null);
        String title = (null == map.get(Const.TITLE.trim()) ? Const.EMPTY :
map.get(Const.TITLE.trim()).toString());
    columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE).append(Cons
t.DOUHAOSTR);
    values.append(Const._1_NOW_NOW).append(a.escapeSql(title)).append(Const.DOUHA
OSTR);
        if (columns.toString().endsWith(Const.DOUHAOSTR))
            columns.delete(columns.length() - 1, columns.length());
        if (values.toString().endsWith(Const.DOUHAOSTR))
            values.delete(values.length() - 1, values.length());
        String sql = String.format((Const.WF_INS_TASK_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.update(sql.toString(), map);
    }
    public void update(Map<String, Object> map) throws Exception {
        if (null == map.get(Const.ID.trim()))
            throw new ExceptionNoId();
        StringBuffer set = new StringBuffer();
        if (null != map.get(Const.WF_DEF_ID.trim()))
    set.append(Const.WF_DEF_ID_DENGYU).append(a.escapeSql(map.get(Const.WF_DEF_I
D.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.WF_NODE_ID.trim()))
    set.append(Const.WF_NODE_ID_DENGYU).append(a.escapeSql(map.get(Const.WF_NOD
E_ID.trim())))
```

```
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.WF_INS_ID.trim()))
    set.append(Const.WF_INS_ID_DENGYU).append(a.escapeSql(map.get(Const.WF_INS_I
D.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.APPLICANT_ID.trim()))
    set.append(Const.APPLICANT_ID_DENGYU).append(a.escapeSql(map.get(Const.APPLIC
ANT_ID.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.DONE.trim()))
    set.append(Const.DONE_DENGYU).append(a.escapeSql(map.get(Const.DONE.trim()))).a
ppend(Const.DOUHAOSTR);
        if (null != map.get(Const.OFFICER_ID.trim()))
    set.append(Const.OFFICER_ID_DENGYU).append(a.escapeSql(map.get(Const.OFFICER_
ID.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.NUM.trim()))
    set.append(Const.NUM_DENGYU).append(a.escapeSql(map.get(Const.NUM.trim()))).app
end(Const.DOUHAOSTR);
        if (null != map.get(Const.AGREE.trim()))
    set.append(Const.AGREE_DENGYU).append(a.escapeSql(map.get(Const.AGREE.trim()))).
append(Const.DOUHAOSTR);
        if (null != map.get(Const.TITLE.trim()))
    set.append(Const.TITLE_DENGYU).append(a.escapeSql(map.get(Const.TITLE.trim()))).a
ppend(Const.DOUHAOSTR);
        set.append(Const.LAST_WRITE_TIME_NOW);
        if (set.toString().endsWith(Const.DOUHAOSTR))
            set.delete(set.length() - 1, set.length());
        String sql = String.format((Const.WF_INS_TASK_UPDATE),
                new Object[] { set.toString(), map.get(Const.ID.trim()) });
        super.update(sql.toString(), map);
    }
    public void delete(final Integer id) throws Exception {
        this.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.VISIBLE.trim(), Const.ZSTR);
                this.put(Const.ID.trim(), id);
            }
        });
    }
}
package com.windowdb.wms.dao;
import java.util.List;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.stereotype.Repository;
```

```
import com.windowdb.utils.a;
import com.windowdb.wms.exception.ExceptionNoId;
/**
 * @author 竹林春雨
 * @date 2017-01-04
 */
@Repository
public class WfInsDao extends AbstractedDao {
    public List<Map<String, Object>> queryList(String field, Object value) throws Exception {
        String sql = String.format((Const.WF_INS_QUERYLIST), new Object[]
{ a.camelToUnderline(field), value });
        return super.queryList(sql);
    }
    public Map<String, Object> queryOne(int id) throws Exception {
        String sql = String.format((Const.WF_INS_QUERYONE), new Object[] { id });
        return super.queryOne(sql.toString());
    }
    public Number insertBackKey(Map<String, Object> map) throws Exception {
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.DONE.trim())) {
            columns.append(Const.DONE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.DONE.trim()))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.WF_DEF_ID.trim())) {
            columns.append(Const.WF_DEF_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_DEF_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.CURRENT_NODE_ID.trim())) {
            columns.append(Const.CURRENT_NODE_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CURRENT_NODE_ID.trim()))).append(Const.
DOUHAOSTR);
        }
        if (null != map.get(Const.BILL_ID.trim())) {
            columns.append(Const.BILL_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.BILL_ID.trim()))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.INS_STATUS.trim())) {
            columns.append(Const.INS_STATUS_DOUHAO);
    values.append(a.escapeSql(map.get(Const.INS_STATUS.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.CREATOR_ID.trim())) {
            columns.append(Const.CREATOR_ID_DOUHAO);
```

```
    values.append(a.escapeSql(map.get(Const.CREATOR_ID.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.ID.trim()))
            map.put(Const.ID.trim(), null);
        String title = (null == map.get(Const.TITLE.trim()) ? Const.EMPTY :
map.get(Const.TITLE.trim()).toString());
        columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE);
        values.append(Const._1_NOW_NOW).append(a.escapeSql(title));
        String sql = String.format((Const.WF_INS_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.insertBackKey(sql.toString(), map);
    }
    public Number insert(Map<String, Object> map) throws Exception {
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.DONE.trim())) {
            columns.append(Const.DONE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.DONE.trim()))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.WF_DEF_ID.trim())) {
            columns.append(Const.WF_DEF_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_DEF_ID.trim()))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.CURRENT_NODE_ID.trim())) {
            columns.append(Const.CURRENT_NODE_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CURRENT_NODE_ID.trim()))).append(Const.
DOUHAOSTR);
        }
        if (null != map.get(Const.BILL_ID.trim())) {
            columns.append(Const.BILL_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.BILL_ID.trim()))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.INS_STATUS.trim())) {
            columns.append(Const.INS_STATUS_DOUHAO);
    values.append(a.escapeSql(map.get(Const.INS_STATUS.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.CREATOR_ID.trim())) {
            columns.append(Const.CREATOR_ID_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CREATOR_ID.trim()))).append(Const.DOUHA
OSTR);
        }
        if (null != map.get(Const.ID))
```

```
            map.put(Const.ID.trim(), null);
        String title = (null == map.get(Const.TITLE.trim()) ? Const.EMPTY :
map.get(Const.TITLE.trim()).toString());
    columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE).append(Cons
t.DOUHAOSTR);
    values.append(Const._1_NOW_NOW).append(a.escapeSql(title)).append(Const.DOUHA
OSTR);
        if (columns.toString().endsWith(Const.DOUHAOSTR))
            columns.delete(columns.length() - 1, columns.length());
        if (values.toString().endsWith(Const.DOUHAOSTR))
            values.delete(values.length() - 1, values.length());
        String sql = String.format((Const.WF_INS_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.update(sql.toString(), map);
    }
    public void update(Map<String, Object> map) throws Exception {
        if (null == map.get(Const.ID.trim()))
            throw new ExceptionNoId();
        StringBuffer set = new StringBuffer();
        if (null != map.get(Const.DONE.trim()))
    set.append(Const.DONE_DENGYU).append(a.escapeSql(map.get(Const.DONE.trim()))).a
ppend(Const.DOUHAOSTR);
        if (null != map.get(Const.WF_DEF_ID.trim()))
    set.append(Const.WF_DEF_ID_DENGYU).append(a.escapeSql(map.get(Const.WF_DEF_I
D.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.CURRENT_NODE_ID.trim()))
    set.append(Const.CURRENT_NODE_ID_DENGYU).append(a.escapeSql(map.get(Const.C
URRENT_NODE_ID.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.BILL_ID.trim()))
    set.append(Const.BILL_ID_DENGYU).append(a.escapeSql(map.get(Const.BILL_ID.trim())
)).append(Const.DOUHAOSTR);
        if (null != map.get(Const.INS_STATUS.trim()))
    set.append(Const.INS_STATUS_DENGYU).append(a.escapeSql(map.get(Const.INS_STAT
US.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.CREATOR_ID.trim()))
    set.append(Const.CREATOR_ID_DENGYU).append(a.escapeSql(map.get(Const.CREATOR
_ID.trim())))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.TITLE.trim()))
    set.append(Const.TITLE_DENGYU).append(a.escapeSql(map.get(Const.TITLE.trim()))).a
ppend(Const.DOUHAOSTR);
        set.append(Const.LAST_WRITE_TIME_NOW);
        if (set.toString().endsWith(Const.DOUHAOSTR))
```

```
            set.delete(set.length() - 1, set.length());
        String sql = String.format((Const.WF_INS_UPDATE), new Object[] { set.toString(),
map.get(Const.ID.trim()) });
        super.update(sql.toString(), map);
    }
    public void delete(final Integer id) throws Exception {
        this.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.VISIBLE, Const.ZSTR);
                this.put(Const.ID, id);
            }
        });
    }
}
package com.windowdb.wms.dao;
import java.util.List;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.stereotype.Repository;
import com.windowdb.utils.a;
import com.windowdb.wms.exception.ExceptionNoId;
/**
 * @author 竹林春雨
 * @date 2017-01-04
 */
@Repository
public class WfDefDao extends AbstractedDao {
    public List<Map<String, Object>> queryList(String field, Object value) throws Exception {
        String sql = String.format((Const.WF_DEF_QUERYLIST), new Object[]
{ a.camelToUnderline(field), value });
        return super.queryList(sql);
    }
    public Map<String, Object> queryOne(int id) throws Exception {
        String sql = String.format((Const.WF_DEF_QUERYONE), new Object[] { id });
        return super.queryOne(sql.toString());
    }
    public Number insertBackKey(Map<String, Object> map) throws Exception {
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.WF_STATUS)) {
            columns.append(Const.WF_STATUS_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_STATUS))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.WF_CREATOR)) {
            columns.append(Const.WF_CREATOR_DOUHAO);
```

```
    values.append(a.escapeSql(map.get(Const.WF_CREATOR))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.CODE)) {
            columns.append(Const.CODE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CODE))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.VERSION)) {
            columns.append(Const.VERSION_DOUHAO);
    values.append(a.escapeSql(map.get(Const.VERSION))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.TABLE_NAME)) {
            columns.append(Const.TABLE_NAME_DOUHAO);
    values.append(a.escapeSql(map.get(Const.TABLE_NAME))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.COLUMN_NAME)) {
            columns.append(Const.COLUMN_NAME_DOUHAO);
    values.append(a.escapeSql(map.get(Const.COLUMN_NAME))).append(Const.DOUHAOST
R);
        }
        if (null != map.get(Const.EFFECTIVE_DATE)) {
            columns.append(Const.EFFECTIVE_DATE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.EFFECTIVE_DATE))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.CLOSING_DATE)) {
            columns.append(Const.CLOSING_DATE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CLOSING_DATE))).append(Const.DOUHAOST
R);
        }
        if (null != map.get(Const.NOTE)) {
            columns.append(Const.NOTE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.NOTE))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.ID))
            map.put(Const.ID, null);
        String title = (null == map.get(Const.TITLE) ? Const.EMPTY :
map.get(Const.TITLE).toString());
        columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE);
        values.append(Const._1_NOW_NOW).append(a.escapeSql(title));
        String sql = String.format((Const.WF_DEF_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.insertBackKey(sql.toString(), map);
    }
    public Number insert(Map<String, Object> map) throws Exception {
```

```
        StringBuffer columns = new StringBuffer();
        StringBuffer values = new StringBuffer();
        if (null != map.get(Const.WF_STATUS)) {
            columns.append(Const.WF_STATUS_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_STATUS))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.WF_CREATOR)) {
            columns.append(Const.WF_CREATOR_DOUHAO);
    values.append(a.escapeSql(map.get(Const.WF_CREATOR))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.CODE)) {
            columns.append(Const.CODE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CODE))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.VERSION)) {
            columns.append(Const.VERSION_DOUHAO);
    values.append(a.escapeSql(map.get(Const.VERSION))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.TABLE_NAME)) {
            columns.append(Const.TABLE_NAME_DOUHAO);
    values.append(a.escapeSql(map.get(Const.TABLE_NAME))).append(Const.DOUHAOSTR)
;
        }
        if (null != map.get(Const.COLUMN_NAME)) {
            columns.append(Const.COLUMN_NAME_DOUHAO);
    values.append(a.escapeSql(map.get(Const.COLUMN_NAME))).append(Const.DOUHAOST
R);
        }
        if (null != map.get(Const.EFFECTIVE_DATE)) {
            columns.append(Const.EFFECTIVE_DATE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.EFFECTIVE_DATE))).append(Const.DOUHAO
STR);
        }
        if (null != map.get(Const.CLOSING_DATE)) {
            columns.append(Const.CLOSING_DATE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.CLOSING_DATE))).append(Const.DOUHAOST
R);
        }
        if (null != map.get(Const.NOTE)) {
            columns.append(Const.NOTE_DOUHAO);
    values.append(a.escapeSql(map.get(Const.NOTE))).append(Const.DOUHAOSTR);
        }
        if (null != map.get(Const.ID))
            map.put(Const.ID, null);
        String title = (null == map.get(Const.TITLE) ? Const.EMPTY :
```

```
map.get(Const.TITLE).toString());
    columns.append(Const.VISIBLE_INSERT_TIME_LAST_WRITE_TIME_TITLE).append(Cons
t.DOUHAOSTR);
    values.append(Const._1_NOW_NOW).append(a.escapeSql(title)).append(Const.DOUHA
OSTR);
        if (columns.toString().endsWith(Const.DOUHAOSTR))
            columns.delete(columns.length() - 1, columns.length());
        if (values.toString().endsWith(Const.DOUHAOSTR))
            values.delete(values.length() - 1, values.length());
        String sql = String.format((Const.WF_DEF_INSERT), new Object[]
{ columns.toString(), values.toString() });
        return super.update(sql.toString(), map);
    }
    public void update(Map<String, Object> map) throws Exception {
        if (null == map.get(Const.ID))
            throw new ExceptionNoId();
        StringBuffer set = new StringBuffer();
        if (null != map.get(Const.WF_STATUS))
    set.append(Const.WF_STATUS_DENGYU).append(a.escapeSql(map.get(Const.WF_STAT
US))).append(Const.DOUHAOSTR);
        if (null != map.get(Const.WF_CREATOR))
    set.append(Const.WF_CREATOR_DENGYU).append(a.escapeSql(map.get(Const.WF_CRE
ATOR))).append(Const.DOUHAOSTR);
        if (null != map.get(Const.CODE))
    set.append(Const.CODE_DENGYU).append(a.escapeSql(map.get(Const.CODE))).append
(Const.DOUHAOSTR);
        if (null != map.get(Const.VERSION))
    set.append(Const.VERSION_DENGYU).append(a.escapeSql(map.get(Const.VERSION))).
append(Const.DOUHAOSTR);
        if (null != map.get(Const.TABLE_NAME))
    set.append(Const.TABLE_NAME_DENGYU).append(a.escapeSql(map.get(Const.TABLE_N
AME))).append(Const.DOUHAOSTR);
        if (null != map.get(Const.COLUMN_NAME))
    set.append(Const.COLUMN_NAME_DENGYU).append(a.escapeSql(map.get(Const.COLUM
N_NAME)))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.EFFECTIVE_DATE))
    set.append(Const.EFFECTIVE_DATE_DENGYU).append(a.escapeSql(map.get(Const.EFFE
CTIVE_DATE)))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.CLOSING_DATE))
    set.append(Const.CLOSING_DATE_DENGYU).append(a.escapeSql(map.get(Const.CLOSI
NG_DATE)))
                    .append(Const.DOUHAOSTR);
        if (null != map.get(Const.NOTE))
```

```
    set.append(Const.NOTE_DENGYU).append(a.escapeSql(map.get(Const.NOTE))).append(
Const.DOUHAOSTR);
        if (null != map.get(Const.TITLE))
    set.append(Const.TITLE_DENGYU).append(a.escapeSql(map.get(Const.TITLE))).append(
Const.DOUHAOSTR);
        set.append(Const.LAST_WRITE_TIME_NOW);
        if (set.toString().endsWith(Const.DOUHAOSTR))
            set.delete(set.length() - 1, set.length());
        String sql = String.format((Const.WF_DEF_UPDATE), new Object[] { set.toString(),
map.get(Const.ID) });
        super.update(sql.toString(), map);
    }
    public void delete(final Integer id) throws Exception {
        this.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.VISIBLE, Const.ZSTR);
                this.put(Const.ID, id);
            }
        });
    }
}
package com.windowdb.wms.service;
import java.util.List;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.windowdb.wms.dao.ShortcutModeDao;
/**
 * @author 竹林春雨
 */
@Service
@Transactional
public class ShortcutModeService extends AbstractedService {
    @Autowired
    protected ShortcutModeDao shortcutModeDao;
    @Override
    public Map<String, Object> queryPageData(Integer userId) throws Exception {
        Map<String, Object> map = new WeakHashMap<String, Object>();
        List<Map<String, Object>> shortcutModeList =
this.shortcutModeDao.queryList(String.format(Const.SHORTCUTMODELISTSQL, new Object[]
{ userId }));
        List<Map<String, Object>> shortcutUrlList =
this.shortcutModeDao.queryList(String.format(Const.SHORTCUTURLLISTSQL, new Object[]
{ userId }));
```

```
        map.put(Const.SHORTCUTMODELIST, shortcutModeList);
        map.put(Const.SHORTCUTURLLIST, shortcutUrlList);
        map.putAll(super.queryPageData(userId));
        return map;
    }
}
package com.windowdb.wms.service;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;
import com.windowdb.wms.dao.WfDefDao;
import com.windowdb.wms.dao.WfDefNodeDao;
import com.windowdb.wms.dao.WfInsDao;
import com.windowdb.wms.dao.WfInsTaskDao;
import com.windowdb.wms.exception.ExceptionWorkFlowNoneBizData;
import com.windowdb.wms.exception.ExceptionWorkFlowNoneWfCode;
import com.windowdb.wms.exception.ExceptionWorkFlowParameterNoneBizid;
import com.windowdb.wms.exception.ExceptionWorkFlowParameterNoneStartUser;
import com.windowdb.wms.exception.ExceptionWorkFlowParameterNoneWfCode;
import com.windowdb.wms.exception.ExceptionWorkFlowStarted;
/**
 * @author 竹林春雨
 */
@Service
@Transactional
public class WfInsService extends AbstractedService {
    @Autowired
    protected WfInsDao wfInsDao;
    @Autowired
    protected WfInsTaskDao wfInsTaskDao;
    @Autowired
    protected WfDefDao wfDefDao;
    @Autowired
    protected WfDefNodeDao wfDefNodeDao;
    /**
     * 启动流程，并且将任务安排到第一个节点;
     * 要求调用本方法的方法必须有独立的事物，否则抛出异常。
     * @param wfCode
     * @param bizId
     * @throws Exception
     */
    @Transactional(propagation = Propagation.MANDATORY)
    public synchronized void startUp(final Map<String, Object> user, final String wfCode, final
```

```
Object billId,
            final String say) throws Exception {
        if (null == user || user.isEmpty() || null == user.get(Const.IDLOW)) {
            throw new ExceptionWorkFlowParameterNoneStartUser();
        }
        if (null == wfCode) {
            throw new ExceptionWorkFlowParameterNoneWfCode();
        }
        if (null == billId) {
            throw new ExceptionWorkFlowParameterNoneBizid();
        }
        final Map<String, Object> wfDef = this.wfDefDao
                .queryOne(String.format(Const.WFDEFSQL, new Object[] { wfCode }));
        // 检查业务数据是否存在
        Long count = this.wfDefDao.queryLong(String.format(Const.TRYBIZDATASQL, new
Object[] {
                wfDef.get(Const.TABLE_NAME).toString(),
wfDef.get(Const.COLUMN_NAME).toString(), billId.toString() }));
        if (count == 0) {
            throw new ExceptionWorkFlowNoneBizData();
        }
        // 尝试修改业务数据的状态
        try {
            String tableName = wfDef.get(Const.TABLE_NAME).toString();
            this.wfDefDao.update(String.format(Const.UPDATEBIZDATASTATUS, new Object[]
{ tableName, billId }));
        } catch (Exception e) {
            e.printStackTrace();
        }
        // 检查流程数据是否已经存在
        count = this.wfDefDao.queryLong(String.format(Const.TRYWFDATASQL, new Object[]
{ billId.toString() }));
        if (count > 0) {
            throw new ExceptionWorkFlowStarted();
        }
        if (null == wfDef || wfDef.isEmpty()) {
            throw new ExceptionWorkFlowNoneWfCode();
        }
        final Map<String, Object> fristNode = wfDefNodeDao
                .queryOne(String.format(Const.FRISTNODESQL,          new          Object[]
{ wfDef.get(Const.IDLOW) }));
        final Number insId = this.wfInsDao.insertBackKey(new WeakHashMap<String,
Object>() {
            {
                this.put(Const.DONE, 0);
                this.put(Const.WF_DEF_ID, wfDef.get(Const.IDLOW));
```

```
                this.put(Const.CURRENT_NODE_ID, fristNode.get(Const.IDLOW));
                this.put(Const.BILL_ID, billId);
                this.put(Const.INS_STATUS, 1);
                this.put(Const.CREATOR_ID, user.get(Const.IDLOW));
                this.put(Const.TITLE,
user.get(Const.TITLE_LOW).toString().concat(Const.FAQI)
                        .concat(wfDef.get(Const.TITLE_LOW).toString()));
            }
        });
        Map<String, Object> param = new WeakHashMap<String, Object>() {
            {
                this.put(Const.DONE, 0);
                this.put(Const.WF_DEF_ID, wfDef.get(Const.IDLOW));
                this.put(Const.WF_NODE_ID, fristNode.get(Const.IDLOW));
                this.put(Const.WF_INS_ID, insId);
                this.put(Const.APPLICANT_ID, user.get(Const.IDLOW));
                // officer
                this.put(Const.NUM, -1);
                // agree
                this.put(Const.TITLE,          null          ==          say          ?
user.get(Const.TITLE_LOW).toString().concat(Const.FAQI)
                        .concat(wfDef.get(Const.TITLE_LOW).toString()) : say);
            }
        };
        /*-
         *  杜绝重复请求
         *  如果 最新的一条 task
         *  wf_ins_id 相同
         *  done 相同
         *  wf_def_id 相同
         *  current_node_id 相同
         *  bill_id 相同
         *  ins_status 相同
         *  creator_id 相同
         *  TITLE 相同
         *  那么系统认为是重复提交；直接抛出异常
         */
        Long  counter  =  this.wfInsTaskDao.queryLong(String.format(Const.SQL$5,  new
Object[] { insId, user.get(Const.ID),
                wfDef.get(Const.IDLOW),                fristNode.get(Const.IDLOW),
param.get(Const.TITLE) })
        );
        if (counter > 0)
            throw new RuntimeException(Const.DATAISEXISTS);
        final Number taskId = this.wfInsTaskDao.insertBackKey(param);
        // 指定任务的候选受理人
```

```
        final Number num = this.wfInsTaskDao
                .update(String.format(Const.TASKUSER, new Object[] { taskId,
fristNode.get(Const.IDLOW) }));
        // 更新参与的候选人数量;
        wfInsTaskDao.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.ID, insId);
                this.put(Const.NUM, num);
            }
        });
    }
}
package com.windowdb.wms.service;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.windowdb.utils.a;
import com.windowdb.wms.dao.WfDefDao;
import com.windowdb.wms.dao.WfDefNodeBallotRulesDao;
import com.windowdb.wms.dao.WfDefNodeDao;
import com.windowdb.wms.dao.WfDefNodeRouteDao;
import com.windowdb.wms.dao.WfInsDao;
import com.windowdb.wms.dao.WfInsTaskAcceptanceDao;
import com.windowdb.wms.dao.WfInsTaskDao;
import com.windowdb.wms.exception.ExceptionWorkFlowNoAutoRoute;
import com.windowdb.wms.exception.ExceptionWorkFlowNoClickRoute;
import com.windowdb.wms.exception.ExceptionWorkFlowRouteBizClassConfig;
import com.windowdb.wms.exception.ExceptionWorkFlowRouteMayTarget;
import com.windowdb.wms.exception.ExceptionWorkFlowTaskDone;
import com.windowdb.wms.service.sup.RouteAuto;
import com.windowdb.wms.service.sup.RouteManual;
import com.windowdb.wms.service.sup.RouteManualDefault;
/**
 * @author 竹林春雨
 */
@Service
@Transactional
@SuppressWarnings({ "unused", "serial" })
```

```
public class WfInsTaskService extends AbstractedService {
    @Autowired
    protected WfDefDao wfDefDao;
    @Autowired
    protected WfInsDao wfInsDao;
    @Autowired
    protected WfInsTaskDao wfInsTaskDao;
    @Autowired
    protected WfDefNodeDao wfDefNodeDao;
    @Autowired
    protected WfDefNodeRouteDao wfDefNodeRouteDao;
    @Autowired
    private WfInsTaskAcceptanceDao wfInsTaskAcceptanceDao;
    @Autowired
    private WfDefNodeBallotRulesDao wfDefNodeBallotRulesDao;
    public Map<String, Object> queryLimit(Map<String, Object> map) throws Exception {
        return null;
    }
    @Override
    public Map<String, Object> queryPageData(Integer userId) throws Exception {
        Map<String, Object> map = new WeakHashMap<String, Object>();
        map.putAll(super.queryPageData(userId));
        map.put(Const.BILLLIST,
                wfInsTaskDao.queryList(String.format(Const.BILLLISTSQL, new Object[]
{ userId, userId, userId })));
        return map;
    }
    public Map<String, Object> myTodoList(Integer userId, int start, int limit) throws
Exception {
        Map<String, Object> map = new WeakHashMap<String, Object>();
        map.putAll(this.queryPageData(userId));
        String mytodolistsql = String.format(Const.MYTODOLIST, new Object[] { userId,
userId });
        Map<String, Object> datagrid = new WeakHashMap<String, Object>();
        datagrid.put(Const.TOTAL,
                wfInsTaskDao.queryLong(String.format(Const.TOTALSQL, new Object[]
{ mytodolistsql })));
        datagrid.put(Const.LIST,
                wfInsTaskDao.queryList(String.format(Const.LISTSQL, new Object[]
{ mytodolistsql, limit, start })));
        map.put(Const.DATAGRID, datagrid);
        return map;
    }
    public Map<String, Object> myDoneList(Integer userId, int start, int limit) throws
Exception {
        Map<String, Object> map = new WeakHashMap<String, Object>();
```

```
        map.putAll(this.queryPageData(userId));
        String mydonelist = String.format(Const.MYDONELIST, new Object[] { userId });
        Map<String, Object> datagrid = new WeakHashMap<String, Object>();
        datagrid.put(Const.TOTAL, wfInsTaskDao.queryLong(String.format(Const.TOTALSQL,
new Object[] { mydonelist })));
        datagrid.put(Const.LIST,
                wfInsTaskDao.queryList(String.format(Const.LISTSQL, new Object[]
{ mydonelist, limit, start })));
        map.put(Const.DATAGRID, datagrid);
        return map;
    }
    @Transactional(readOnly = true)
    public Map<String, Object> toDoFormData(Integer userId, Integer taskId) throws
Exception {
        Map<String, Object> map = new WeakHashMap<String, Object>();
        map.putAll(this.queryPageData(userId));
        String ok = wfInsTaskDao
                .queryString(String.format(Const.OKSQL, new Object[] { userId, userId,
taskId, userId, userId }));
        if (null == ok || !Const.OK_EN.equalsIgnoreCase(ok)) {
            throw new ExceptionWorkFlowTaskDone();
        }
        Map<String, Object> node =
this.wfDefNodeDao.queryOne(String.format(Const.NODESQL, new Object[] { taskId }));
        map.put(Const.NODE, node);
        List<Map<String, Object>> routeList = null;
        Object route = node.get(Const.ROUTE_TYPE);
        /**
         * 如果是自动路由
         */
        if (Const.VALUE0.equals(route.toString())) {
            routeList = new ArrayList<Map<String, Object>>() {
                {
                    this.add(new WeakHashMap<String, Object>() {
                        {
                            this.put(Const.NAME, Const.ROUTE);
                            this.put(Const.TITLE, Const.TITLE1);
                            this.put(Const.VALUE, Const.VALUE1);
                        }
                    });
                    this.add(new WeakHashMap<String, Object>() {
                        {
                            this.put(Const.NAME, Const.ROUTE);
                            this.put(Const.TITLE, Const.TITLE2);
                            this.put(Const.VALUE, Const.VALUE0);
                        }
```

```
                    });
                }
            };
        } else {
            routeList = wfDefNodeRouteDao
                    .queryList(String.format(Const.ROUTELISTSQL, new Object[]
{ node.get(Const.ID).toString() }));
            List<Map<String, Object>> nodeUserList = this.wfDefNodeDao
                    .queryList(String.format(Const.NODEUSERLISTSQL, new Object[]
{ node.get(Const.ID).toString() }));
            map.put(Const.NODEUSERLIST, nodeUserList);
        }
        map.put(Const.ROUTELIST, routeList);
        return map;
    }
    /**
     * @param user
     * @param routeId
     * @param officerId
     * @param say
     * @throws Exception
     */
    public void doTask(String token, Map<String, Object> user, Integer taskId, Integer
routeId, Integer nextOfficerId,
            String say) throws Exception {
        final java.util.Date now = new java.util.Date();
        Map<String, Object> taskAndFromNode = this.wfInsTaskDao
                .queryOne(String.format(Const.TASKANDFROMNODE, new Object[]
{ taskId.toString() }));
        Object routeType = taskAndFromNode.get(Const.FROM_NODE_ROUTE_TYPE);
        // 自动路由
        if (Const.VALUE0.equals(routeType.toString())) {
            doAuto(token, now, user, taskId, routeId, taskAndFromNode, nextOfficerId, say);
        }
        // 手动路由
        if (Const.VALUE1.equals(routeType.toString())) {
            doClick(token, now, user, taskId, routeId, taskAndFromNode, nextOfficerId, say);
        }
    }
    /**
     * 手动路由
     * @param user
     * @param taskId
     * @param routeInfo
     * @param nextOfficerId
     * @param say
```

```
     * @throws Exception
     */
    private synchronized void doClick(final String token, final Date now, final Map<String, Object> user,
            final Integer taskId, final Integer routeId, final Map<String, Object> taskAndFromNode,
            Integer nextOfficerId_, final String say) throws Exception {
        Map<String, Object> routeInfo = this.wfDefNodeRouteDao
                .queryOne(String.format(Const.ROUTEINFO, new Object[] { taskId, routeId }));
        final Object ins_id, task_insert_time, wf_def_id, bill_id, route_title, route_type, from_node_index,
                from_node_timeliness, from_node_type, from_node_id, to_node_index, to_node_type, to_node_id;
        bill_id = taskAndFromNode.get(Const.BILL_ID);
        ins_id = taskAndFromNode.get(Const.INS_ID);
        task_insert_time = taskAndFromNode.get(Const.TASK_INSERT_TIME);
        wf_def_id = taskAndFromNode.get(Const.WF_DEF_ID);
        from_node_index = taskAndFromNode.get(Const.FROM_NODE_INDEX);
        from_node_timeliness = taskAndFromNode.get(Const.FROM_NODE_TIMELINESS);
        from_node_type = taskAndFromNode.get(Const.FROM_NODE_TYPE);
        from_node_id = taskAndFromNode.get(Const.FROM_NODE_ID);
        route_type = taskAndFromNode.get(Const.FROM_NODE_ROUTE_TYPE_);
        to_node_index = routeInfo.get(Const.TO_NODE_INDEX);
        to_node_type = routeInfo.get(Const.TO_NODE_TYPE);
        to_node_id = routeInfo.get(Const.TO_NODE_ID);
        route_title = routeInfo.get(Const.ROUTE_TITLE);
        if (!Const.VALUE1.equals(route_type.toString())) {
            throw new ExceptionWorkFlowNoClickRoute();
        }
        execCheckScript(token, user, routeId, ins_id, task_insert_time, wf_def_id, bill_id, from_node_index,
                from_node_timeliness, from_node_type, from_node_id, to_node_index, to_node_type, to_node_id, 0);
        // 现有的待办变成已办
        this.wfInsTaskDao.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.ID.trim(), taskId);
                this.put(Const.DONE.trim(), 1);
                this.put(Const.OFFICER_ID.trim(), user.get(Const.ID.trim()));
                this.put(Const.NOW.trim(), now);
            }
        });
        // 0 开始节点,说明是第一步,获取流程创建人作为执行人
        if (null == nextOfficerId_ && Const.VALUE0.equals(to_node_index.toString())) {
            String creatorid = this.wfDefNodeDao.queryString(
```

```
                    new
StringBuffer(Const.SELECT_CREATOR_ID_FROM_WF_INS_WHERE_ID).append(ins_id).toStri
ng());
            nextOfficerId_ = Integer.valueOf(creatorid);
        }
        final Integer nextOfficerId = nextOfficerId_;
        RouteManual route = null;
        // 执行自定义动作
        try {
            route = (RouteManual) Class.forName(String.format(Const.ROUTEMANUAL, new
Object[] { routeId }))
                    .newInstance();
        } catch (ClassNotFoundException e) {
            route = new RouteManualDefault();
        } catch (Exception e) {
            e.printStackTrace();
        }
        route.go(this.wfInsTaskDao, bill_id, new WeakHashMap<String, Object>() {
            {
                this.put(Const.WORKFLOWID, wf_def_id);
                this.put(Const.TASKID, taskId);
                this.put(Const.ROUTEID, routeId);
                this.put(Const.ROUTETYPE, route_type);
                this.put(Const.FROMNODEINDEX, from_node_index);
                this.put(Const.FROMNODETYPE, from_node_type);
                this.put(Const.FROMNODEID, from_node_id);
                this.put(Const.TONODEINDEX, to_node_index);
                this.put(Const.TONODETYPE, to_node_type);
                this.put(Const.TONODEID, to_node_id);
                this.put(Const.ROUTETITLE, route_title);
                this.put(Const.NEXTOFFICERID, nextOfficerId);
                this.put(Const.TASK_INSERT_TIME, task_insert_time);
                this.put(Const.FROM_NODE_TIMELINESS, from_node_timeliness);
                this.put(Const.SAY, say);
            }
        });
        int wfStatus = 1;
        if (Const.MYQQNUM.equals(to_node_index.toString())) {
            wfStatus = 4;
        }
        String tableName = this.wfDefDao
    .queryString(Const.SELECT_TABLE_NAME_FROM_WF_DEF_WHERE_ID.concat(wf_def_id.
toString()));
        Number n = 0;
        try {
            n                                                                    =
```

```
this.wfDefDao.update(Const.UPDATE.concat(tableName).concat(Const.BLANK_SPACE)
                    .concat(Const.SET_WF_STATUS).concat(new
StringBuffer(Const.EMPTY).append(wfStatus).toString())
    .concat(Const.BLANK_SPACE).concat(Const.WHERE_ID).concat(bill_id.toString()));
        } catch (Exception e) {
            throw new RuntimeException(Const.NOBIZTABLE);
        }
        if (null == n || n.intValue() < 1) {
            throw new RuntimeException(Const.NOBIZTABLEDATA);
        }
        final int status = wfStatus;
        // 同步流程实例的状态，以及当前节点
        this.wfInsDao.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.ID, ins_id);
                this.put(Const.INS_STATUS, status);
                this.put(Const.CURRENT_NODE_ID, to_node_id);
                if (Const.MYQQNUM.equals(to_node_index.toString()))
                    this.put(Const.DONE, 1);
                else
                    this.put(Const.DONE, 0);
            }
        });
        Map<String, Object> param = new WeakHashMap<String, Object>() {
            {
                this.put(Const.NUM, 0);
                this.put(Const.WF_INS_ID, ins_id);
                // 6996899 结束节点,直接结束，否则作为新任务；
                if (Const.MYQQNUM.equals(to_node_index.toString()))
                    this.put(Const.DONE, 1);
                else
                    this.put(Const.DONE, 0);
                this.put(Const.APPLICANT_ID, user.get(Const.ID));
                this.put(Const.WF_DEF_ID, wf_def_id);
                this.put(Const.WF_NODE_ID, to_node_id);
                this.put(Const.TITLE,
user.get(Const.NAME).toString().concat(route_title.toString()));
            }
        };
        /*-
         *  杜绝重复请求
         *  如果 最新的一条 task
         *  wf_ins_id 相同
         *  done 相同
         *  applicant_id 相同
         *  wf_def_id 相同
```

```
         *  wf_node_id 相同
         *  title 相同
         *  那么系统认为是重复提交；直接抛出异常
         */
        Long count = this.wfInsTaskDao.queryLong(String.format(Const.SQL$4, new Object[]
{ ins_id,
                param.get(Const.DONE),  user.get(Const.ID),  wf_def_id,  to_node_id,
param.get(Const.TITLE) }));
        if (count > 0)
            throw new RuntimeException(Const.DATAISEXISTS);
        // 增加一条新的待办任务
        final Number newTaskId = wfInsTaskDao.insertBackKey(param);
        // 6996899 结束节点,说明是最后一步
        if (Const.MYQQNUM.equals(to_node_index.toString())) {
            return;
        }
        final Number num;
        String creatorid = null;
        // 判断目标节点受理模式 节点模式： 0,指派模式，1,单人竞取模式 2,全员通过模式 3,表决模式
        if (Const.VALUE0.equals(to_node_type.toString())) {
            if (null == nextOfficerId) {
                creatorid = this.wfDefNodeDao.queryString(
                        new
StringBuffer(Const.SELECT_CREATOR_ID_FROM_WF_INS_WHERE_ID).append(ins_id).toStri
ng());
                this.wfInsTaskDao.insertLink(Const._TASK_USER,  newTaskId.toString(),
creatorid);
            } else {
                this.wfInsTaskDao.insertLink(Const._TASK_USER,  newTaskId.toString(),
nextOfficerId.toString());
            }
            num = 1;
        } else {
            num  =  this.wfInsTaskDao.update(String.format(Const.TASKUSER,  new  Object[]
{ newTaskId, to_node_id }));
        }
        final String officerid = (null != nextOfficerId ? nextOfficerId.toString()
                : (null == nextOfficerId_ ? creatorid : nextOfficerId_.toString()));
        wfInsTaskDao.update(new WeakHashMap<String, Object>() {
            {
                this.put(Const.NUM, null == num ? 0 : num);
                this.put(Const.ID, newTaskId);
            }
        });
        // 受理结果
        this.wfInsTaskAcceptanceDao.insert(new WeakHashMap<String, Object>() {
```

```
            {
                this.put(Const.TASK_ID, taskId);
                this.put(Const.OFFICER_ID, user.get(Const.ID));
                this.put(Const.AGREE_MODE, 1);
                this.put(Const.SAY, say);
                this.put(Const.NOW, now);
                this.put(Const.TITLE, user.get(Const.NAME).toString().concat(Const.DOIT));
            }
        });
        // 最后执行动作脚本
        execActionScript(token, user, routeId, ins_id, task_insert_time, wf_def_id, bill_id,
from_node_index,
                from_node_timeliness, from_node_type, from_node_id, to_node_index,
to_node_type, to_node_id);
        // 最后检查约束是否完全符合
        execCheckScript(token, user, routeId, ins_id, task_insert_time, wf_def_id, bill_id,
from_node_index,
                from_node_timeliness, from_node_type, from_node_id, to_node_index,
to_node_type, to_node_id, 1);
    }
    private void execCheckScript(String token, Map<String, Object> user, Integer routeId,
Object ins_id,
            Object task_insert_time, Object wf_def_id, Object bill_id, Object
from_node_index,
            Object from_node_timeliness, Object from_node_type, Object from_node_id,
Object to_node_index,
            Object to_node_type, Object to_node_id, int ba) throws Exception {
        if (null == routeId) {
            return;
        }
        List<Map<String, Object>> sqls = this.wfDefDao.queryList(
    String.format(Const.SELECT_FROM_WF_DEF_NODE_ROUTE_CHECKSCRIPT_WHERE_RID
_S_AND_B_A_S_ORDER_BY_ORDER,
                        new Object[] { routeId, String.valueOf(ba) }));
        for (Map<String, Object> sqlInfo : sqls) {
            String sql = sqlInfo.get(Const.SQL.toLowerCase()).toString();
            if (null == sql || Const.EMPTY.equals(sql.trim()))
                continue;
            sql = replaceSqlKey(sql, Const.MY_DOT_ID.toLowerCase(),
user.get(Const.IDLOW));
            sql = replaceSqlKey(sql, Const.ROUTE_ID, routeId);
            sql = replaceSqlKey(sql, Const.INS_ID.toLowerCase(), ins_id);
            sql = replaceSqlKey(sql, Const.TASK_INSERT_TIME.toLowerCase(),
task_insert_time);
            sql = replaceSqlKey(sql, Const.WF_DEF_ID.toLowerCase(), wf_def_id);
            sql = replaceSqlKey(sql, Const.FROM_NODE_INDEX.toLowerCase(),
```

```
from_node_index);
            sql = replaceSqlKey(sql, Const.FROM_NODE_TIMELINESS.toLowerCase(),
from_node_timeliness);
            sql = replaceSqlKey(sql, Const.FROM_NODE_TYPE.toLowerCase(),
from_node_type);
            sql = replaceSqlKey(sql, Const.FROM_NODE_ID.toLowerCase(), from_node_id);
            sql = replaceSqlKey(sql, Const.FROM_NODE_INDEX.toLowerCase(),
to_node_index);
            sql = replaceSqlKey(sql, Const.TO_NODE_TYPE.toLowerCase(), to_node_type);
            sql = replaceSqlKey(sql, Const.TO_NODE_ID.toLowerCase(), to_node_id);
            sql = sql.concat(Const.LIMIT_1);
            String msg = this.wfDefDao.queryString(sql);
            if (null != msg && !Const.EMPTY.equals(msg.trim()))
                throw new RuntimeException(msg);
        }
    }
    private void execActionScript(final String token, final Map<String, Object> user, final
Integer routeId,
            final Object ins_id, final Object task_insert_time, final Object wf_def_id, final
Object bill_id,
            final Object from_node_index, final Object from_node_timeliness, final Object
from_node_type,
            final Object from_node_id, final Object to_node_index, final Object to_node_type,
final Object to_node_id)
            throws Exception {
        if (null == routeId) {
            return;
        }
        List<Map<String, Object>> sqls;
        Map<String, Object> keys = new LinkedHashMap<String, Object>();
        sqls = this.wfDefDao.queryList(String.format(
    Const.SELECT_FROM_WF_DEF_NODE_ROUTE_ACTIONSCRIPT_WHERE_RID_S_ORDER_
BY_ORDER, new Object[] { routeId }));
        for (Map<String, Object> sqlInfo : sqls) {
            String sqlid = sqlInfo.get(Const.IDLOW.toLowerCase()).toString();
            String sql = sqlInfo.get(Const.SQL.toLowerCase()).toString();
            sql = replaceSqlKey(sql, Const.MY_DOT_ID.toLowerCase(),
user.get(Const.IDLOW));
            sql = replaceSqlKey(sql, Const.ROUTE_ID, routeId);
            sql = replaceSqlKey(sql, Const.INS_ID.toLowerCase(), ins_id);
            sql = replaceSqlKey(sql, Const.TASK_INSERT_TIME.toLowerCase(),
task_insert_time);
            sql = replaceSqlKey(sql, Const.WF_DEF_ID.toLowerCase(), wf_def_id);
            sql = replaceSqlKey(sql, Const.FROM_NODE_INDEX.toLowerCase(),
from_node_index);
            sql = replaceSqlKey(sql, Const.FROM_NODE_TIMELINESS.toLowerCase(),
```

```
from_node_timeliness);
            sql = replaceSqlKey(sql, Const.FROM_NODE_TYPE.toLowerCase(),
from_node_type);
            sql = replaceSqlKey(sql, Const.FROM_NODE_ID.toLowerCase(), from_node_id);
            sql = replaceSqlKey(sql, Const.FROM_NODE_INDEX.toLowerCase(),
to_node_index);
            sql = replaceSqlKey(sql, Const.TO_NODE_TYPE.toLowerCase(), to_node_type);
            sql = replaceSqlKey(sql, Const.TO_NODE_ID.toLowerCase(), to_node_id);
            Iterator<String> ks = keys.keySet().iterator();
            while (ks.hasNext()) {
                String k = ks.next();
                if (null == k || Const.EMPTY.equals(k.trim()))
                    continue;
                sql = sql.replace(
                        new
StringBuffer(Const.$_RETURNKEY).append(k.trim()).append(Const.$RIGHT_HUAKUOHAO).t
oString(),
                        a.escapeSql(keys.get(k)));
                sql = replaceSqlKey(sql, new
StringBuffer(Const.RETURNKEY_).append(k.trim()).toString(),
                        a.escapeSql(keys.get(k)));
            }
            if (sql.indexOf(Const.$HUALEFT) > -1) {
                // System.out.println(sqlid.concat("# 路由 sql 参数错误 ,请检查。"));
            }
            Integer returnkey = Integer.valueOf(sqlInfo.get(Const.RETURNKEY).toString());
            if (1 == returnkey.intValue()) {
                Number key = this.wfDefDao.insertBackKey(sql);
                if (null == key) {
                    throw new
java.lang.RuntimeException(sqlid.concat(Const.ERRORMSG0001));
                }
                sql = String.format(

    Const.INSERT_INTO_S_DB_LOG_UPDATE_ROW_OPTIME_USER_ID_SESSIONID_VIEW_
NAME_TEMPLID_METHOD_NAME_SQL_VALUES_NOW_S_S_S_S_S_S,
                        new Object[] { Const.CONFIG_SCHEMA,
a.escapeSql(user.get(Const.IDLOW)), a.escapeSql(token),
    a.escapeSql(Const.WF_DEF_NODE_ROUTE_ACTIONSCRIPT.toLowerCase()),
a.escapeSql(routeId),
    a.escapeSql(Const.WORKFLOW_DUMP.concat(key.toString())),
a.escapeSql(sql.toString()) });
                Number log = this.wfDefDao.insertBackKey(sql);
                if (0 == log.intValue())
                    throw new RuntimeException(Const.OPFIAL);
                keys.put(sqlid, key);
```

```
            } else if (2 == returnkey.intValue()) {
                String key = this.wfDefDao.queryString(sql);
                keys.put(sqlid, key);
            } else {
                this.wfDefDao.update(sql);
                sql = String.format(
    Const.INSERT_INTO_S_DB_LOG_UPDATE_ROW_OPTIME_USER_ID_SESSIONID_VIEW_
NAME_TEMPLID_METHOD_NAME_SQL_VALUES_NOW_S_S_S_S_S_S,
                        new Object[] { Const.CONFIG_SCHEMA,
a.escapeSql(user.get(Const.IDLOW)), a.escapeSql(token),
    a.escapeSql(Const.WF_DEF_NODE_ROUTE_ACTIONSCRIPT.toLowerCase()),
a.escapeSql(routeId),a.escapeSql(Const.WORKFLOW_DUMP), a.escapeSql(sql.toString()) });
                Number log = this.wfDefDao.insertBackKey(sql);
                if (0 == log.intValue())
                    throw new RuntimeException(Const.OPFIAL);
            }
        }
    }
    private String replaceSqlKey(String sql, String k, Object v) {
        if (k != null && sql.indexOf(k) > -1) {
            sql = sql.replace(
                    new
StringBuffer(Const.$LEFTHUAKUOHAO).append(k.trim()).append(Const.$RIGHT_HUAKUOHA
O).toString(),a.escapeSql(v));
        }
        return sql;
    }
    /**
     * 自动路由
     * @param user
     * @param taskId
     * @param routeInfo
     * @param nextOfficerId
     * @param say
     * @throws Exception
     */
    @SuppressWarnings("null")
    private synchronized void doAuto(final String token, final Date now, final Map<String,
Object> user,
            final Integer taskId, final Integer agree, final Map<String, Object>
taskAndFromNode,
            final Integer nextOfficerId, final String say) throws Exception {
        final Object ins_id, task_insert_time, from_node_timeliness, wf_def_id, bill_id,
from_node_route_type,
                from_node_index, from_node_type, from_node_id;
        bill_id = taskAndFromNode.get(Const.BILL_ID);
```

```
        wf_def_id = taskAndFromNode.get(Const.WF_DEF_ID);
        from_node_index = taskAndFromNode.get(Const.FROM_NODE_INDEX);
        from_node_timeliness = taskAndFromNode.get(Const.FROM_NODE_TIMELINESS);
        from_node_type = taskAndFromNode.get(Const.FROM_NODE_TYPE);
        from_node_id = taskAndFromNode.get(Const.FROM_NODE_INDEX);
        from_node_route_type = taskAndFromNode.get(Const.FROM_NODE_ROUTE_TYPE_);
        task_insert_time = taskAndFromNode.get(Const.TASK_INSERT_TIME);
        ins_id = taskAndFromNode.get(Const.INS_ID);
        Integer rid = null;
        if (!Const.VALUE0.equals(from_node_route_type.toString())) {
            throw new ExceptionWorkFlowNoAutoRoute();
        }
        List<Map<String, Object>> routeList = this.wfDefNodeRouteDao
                .queryList(String.format(Const.ROUTELISTSQL, new Object[]
{ from_node_id }));
        Object to_node_index = null;
        Object to_node_id = null;
        Object route_title;
        // 是否结束掉该任务
        boolean taskdone = false;
        String cnOp = Const.EMPTY;
        // 1,单人竞取模式
        if (Const.VALUE1.equals(from_node_type.toString())) {
            cnOp = Const.DOIT;
            taskdone = true;
        }
        // 2,全员确定模式
        if (Const.VALUE2.equals(from_node_type.toString())) {
            cnOp = Const.OK;
            List<Map<String, Object>> acceptList = this.wfInsTaskAcceptanceDao
                    .queryList(String.format(Const.ACCEPTLISTSQL, new Object[]
{ from_node_id }));
            if (null == acceptList || acceptList.isEmpty()) {
                taskdone = true;
            }
        }
        // 3,表决模式
        if (Const.VALUE3.equals(from_node_type.toString())) {
            if (agree == 2)
                cnOp = Const.QIQUAN;
            if (agree == 1)
                cnOp = Const.ZANGCHENG;
            if (agree == 0)
                cnOp = Const.FANDUI;
            // 获取通过规则
            List<Map<String, Object>> rules =
```

```
this.wfDefNodeBallotRulesDao.queryList(Const.NODE_ID, from_node_id);
            Map<String, Object> map = this.wfInsTaskDao
                    .queryOne(String.format(Const.RULESSQL, new Object[] { taskId,
taskId }));
            // 是否同意：0，弃权；1，赞成；2，反对,
            Integer all = Integer.valueOf(null == map.get(Const.ALL) ? Const.VALUE0 :
map.get(Const.ALL).toString());
            Integer a1 = Integer
                    .valueOf(null == map.get(Const.VALUE1) ? Const.VALUE0 :
map.get(Const.VALUE1).toString());
            for (Map<String, Object> m : rules) {
                if (!Const.VALUE1.equals(map.get(Const.VISIBLE).toString())
                        && !Const.TRUE_.equals(map.get(Const.VISIBLE).toString()))
                    continue;
                Integer num = Integer
                        .valueOf(null == map.get(Const.NUM) ? Const.VALUE0 :
map.get(Const.NUM).toString());
                // 0,大于、1,大于等于、2,小于、3,小于等于、4,等于
                Object tan_mode = m.get(Const.TAN_MODE);
                // 0,同意的占多少票，1,同意的占总票数的百分比
                Object tan_unit_mode = m.get(Const.TAN_UNIT_MODE);
                if (Const.VALUE0.equalsIgnoreCase(tan_unit_mode.toString())) {
                    if (Const.VALUE0.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = (a1 > num);
                    }
                    if (Const.VALUE1.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = (a1 >= num);
                    }
                    if (Const.VALUE2.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = (a1 < num);
                    }
                    if (Const.VALUE3.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = (a1 <= num);
                    }
                    if (Const.VALUE4.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = (a1 == num);
                    }
                }
                if (Const.VALUE1.equalsIgnoreCase(tan_unit_mode.toString())) {
                    if (Const.VALUE0.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = ((a1 / all) > (num / 100));
                    }
                    if (Const.VALUE1.equalsIgnoreCase(tan_mode.toString())) {
                        taskdone = ((a1 / all) >= (num / 100));
                    }
                    if (Const.VALUE2.equalsIgnoreCase(tan_mode.toString())) {
```

```
                    taskdone = ((a1 / all) < (num / 100));
                }
                if (Const.VALUE3.equalsIgnoreCase(tan_mode.toString())) {
                    taskdone = ((a1 / all) <= (num / 100));
                }
                if (Const.VALUE4.equalsIgnoreCase(tan_mode.toString())) {
                    taskdone = ((a1 / all) == (num / 100));
                }
            }
        }
    }
    Object to_node_type = Const.EMPTY;
    for (final Map<String, Object> map : routeList) {
        if (!Const.VALUE1.equals(map.get(Const.VISIBLE).toString())
                && !Const.TRUE_.equals(map.get(Const.VISIBLE).toString()))
            continue;
        final Object routeId = map.get(Const.ID);
        RouteAuto route = null;
        // 执行自定义动作
        try {
            route = (RouteAuto) Class.forName(String.format(Const.ROUTEAUTO, new
Object[] { routeId }))
                    .newInstance();
        } catch (Exception e) {
            route = new RouteAuto() {
                @Override
                public boolean isCanGo(WfInsTaskDao dao, Object billId, Map<String,
Object> wfInfo)
                        throws Exception {
                    return true;
                }
                @Override
                public void go(WfInsTaskDao dao, Object billId, Map<String, Object>
wfInfo) throws Exception {
                }
            };
        }
        try {
            boolean iscango = route.isCanGo(this.wfInsTaskDao, bill_id, new
WeakHashMap<String, Object>() {
                {
                    this.put(Const.WORKFLOWID, wf_def_id);
                    this.put(Const.TASKID, taskId);
                    this.put(Const.ROUTEID, routeId);
                    this.put(Const.ROUTETYPE, from_node_route_type);
                    this.put(Const.FROMNODEINDEX, from_node_index);
```

```
                        this.put(Const.FROMNODETYPE, from_node_type);
                        this.put(Const.FROMNODEID, from_node_id);
                        this.put(Const.TONODEID, map.get(Const.TO_NODE_ID));
                        this.put(Const.TASK_INSERT_TIME, task_insert_time);
                        this.put(Const.NEXTOFFICERID, nextOfficerId);
                        this.put(Const.FROM_NODE_TIMELINESS, from_node_timeliness);
                        this.put(Const.SAY, say);
                    }
                });
                if (iscango) {
                    if (null != to_node_index) {
                        throw new ExceptionWorkFlowRouteMayTarget();
                    }
                    to_node_id = map.get(Const.TO_NODE_ID);
                    route_title = map.get(Const.TITLE);
                    if (taskdone) {
                        rid = Integer.valueOf(routeId.toString());
                        to_node_type = map.get(Const.TO_NODE_TYPE);
                        route.go(this.wfInsTaskDao, bill_id, new WeakHashMap<String,
Object>() {
                            {
                                this.put(Const.WORKFLOWID, wf_def_id);
                                this.put(Const.TASKID, taskId);
                                this.put(Const.ROUTEID, routeId);
                                this.put(Const.ROUTETYPE, from_node_route_type);
                                this.put(Const.FROMNODEINDEX, from_node_index);
                                this.put(Const.FROMNODETYPE, from_node_type);
                                this.put(Const.FROMNODEID, from_node_id);
                                this.put(Const.TONODEID, map.get(Const.TO_NODE_ID));
                                this.put(Const.TASK_INSERT_TIME, task_insert_time);
                                this.put(Const.NEXTOFFICERID, nextOfficerId);
                                this.put(Const.FROM_NODE_TIMELINESS,
from_node_timeliness);
                                this.put(Const.SAY, say);
                            }
                        });
                    }
                }
            } catch (Exception e) {
                throw new ExceptionWorkFlowRouteBizClassConfig();
            }
        }
        final Object tnid = to_node_id;
        // final Object routeTitle = route_title;
        // 受理结果
        this.wfInsTaskAcceptanceDao.insert(new WeakHashMap<String, Object>() {
```

```
            {
                this.put(Const.TASK_ID, taskId);
                this.put(Const.OFFICER_ID, user.get(Const.ID));
                this.put(Const.AGREE_MODE, agree);
                this.put(Const.SAY, say);
                this.put(Const.NOW, now);
                this.put(Const.TITLE, user.get(Const.NAME).toString().concat(Const.DOIT));
            }
        });
        int wfStatus = 1;
        if (Const.MYQQNUM.equals(to_node_index.toString())) {
            wfStatus = 4;
        }
        String tableName = this.wfDefDao
    .queryString(Const.SELECT_TABLE_NAME_FROM_WF_DEF_WHERE_ID.concat(wf_def_id.
toString()));
        Number n = 0;
        try {
            n                                                                    =
this.wfDefDao.update(Const.UPDATE.concat(tableName).concat(Const.SET_WF_STATUS)
                    .concat(new
StringBuffer(Const.EMPTY).append(wfStatus).toString()).concat(Const.WHERE_ID)
                    .concat(bill_id.toString()));
        } catch (Exception e) {
            throw new RuntimeException(Const.NOBIZTABLE);
        }
        if (null == n || n.intValue() < 1) {
            throw new RuntimeException(Const.NOBIZTABLEDATA);
        }
        // 执行动作脚本
        if (null != rid)
            execActionScript(token, user, rid, ins_id, task_insert_time, wf_def_id, bill_id,
from_node_index,
                    from_node_timeliness, from_node_type, from_node_id, to_node_index,
to_node_type, to_node_id);
        // 6996899 结束节点,说明是最后一步，直接结束；
        if (Const.MYQQNUM.equals(to_node_index.toString())) {
            return;
        }
        // 如果结束任务的话，需要插入新任务以及 done 掉本次任务。
        if (taskdone) {
            // 现有的代办变成已办
            this.wfInsTaskDao.update(new WeakHashMap<String, Object>() {
                {
                    this.put(Const.ID, taskId);
                    this.put(Const.DONE, 1);
```

```
                this.put(Const.OFFICER, user.get(Const.ID));
                this.put(Const.NOW, now);
            }
        });
        // 增加一条新的待办任务
        final String op = cnOp;
        Map<String, Object> param = new WeakHashMap<String, Object>() {
            {
                this.put(Const.NUM, 0);
                this.put(Const.WF_INS_ID, ins_id);
                this.put(Const.DONE, 0);
                this.put(Const.APPLICANT_ID, user.get(Const.ID));
                this.put(Const.WF_DEF_ID, wf_def_id);
                this.put(Const.WF_NODE_ID, tnid);
                this.put(Const.TITLE, user.get(Const.NAME).toString().concat(op));
            }
        };
        /*-
         *  杜绝重复请求
         *  如果 最新的一条 task
         *  wf_ins_id 相同
         *  done 相同
         *  applicant_id 相同
         *  wf_def_id 相同
         *  wf_node_id 相同
         *  title 相同
         *  那么系统认为是重复提交；直接抛出异常
         */
        Long count = this.wfInsTaskDao.queryLong(String.format(Const.SQL$4, new
Object[] { ins_id,
                param.get(Const.DONE), user.get(Const.ID), wf_def_id, to_node_id,
param.get(Const.TITLE) }));
        if (count > 0)
            throw new RuntimeException(Const.DATAISEXISTS);
        final Number newTaskId = wfInsTaskDao.insertBackKey(param);
        final Number num;
        // 判断目标节点受理模式 节点模式： 0,指派模式，1,单人竞取模式 2,全员通过模式 3,表决
模式
        if (Const.VALUE0.equals(to_node_type.toString())) {
            if (null == nextOfficerId) {
                throw new RuntimeException(Const.ERRORNONTARGETNODEUSER);
            }
            this.wfInsTaskDao.insertLink(Const._TASK_USER,      newTaskId.toString(),
user.get(Const.ID).toString());
            num = 1;
        } else {
```

```
                num = this.wfInsTaskDao.update(String.format(Const.TASKUSER, new
Object[] { newTaskId, to_node_id }));
            }
            wfInsTaskDao.update(new WeakHashMap<String, Object>() {
                {
                    this.put(Const.NUM, null == num ? 0 : num);
                    this.put(Const.ID, newTaskId);
                }
            });
        }
    }
    public Map<String, Object> myHist(Integer userId, Integer insId) throws Exception {
        Map<String, Object> map = this.queryPageData(userId);
        map.put(Const.HISTLIST,  this.wfInsTaskDao.queryList(String.format(Const.MYHIST,
new Object[] { insId })));
        return map;
    }
    /**
     * @param token
     * @param user
     * @param params
     * @param taskid
     * @return
     * @throws Exception
     */
    public Map<String, Object> getWfTodoForm(String token, Map<String, Object> user,
Map<String, Object> params,
            Integer taskid) throws Exception {
        Map<String, Object> returnMap = new HashMap<String, Object>();
        StringBuffer sql = new StringBuffer(
    Const.SELECT_N_ID_AS_NODE_ID_D_FUN_CODE_I_BILL_ID_CONCAT_CASE_WHEN_N_
EFFECTIVE_DATE_NOW_THEN_CONCAT_N_N_ID_T_WF_NODE_ID_INNER_JOIN_WF_DEF_D
_ON_N_WF_DEF_ID_D_ID_AND_T_WF_DEF_ID_D_ID_INNER_JOIN_WF_INS_I_ON_I_ID_T_
WF_INS_ID_WHERE_T_ID_S_LIMIT);
        Map<String, Object> map = this.wfInsTaskDao
                .queryOne(String.format(sql.toString(), new Object[] { taskid.toString() }));
        returnMap.put(Const.FUN_CODE, map.get(Const.FUN_CODE));
        returnMap.put(Const.BILL_ID.toLowerCase(),
map.get(Const.BILL_ID.toLowerCase()));
        returnMap.put(Const.NODE_ID.toLowerCase(),
map.get(Const.NODE_ID.toLowerCase()));
        if (null == map || map.isEmpty()) {
            returnMap.put(Const.ERROR_MSG, Const.ERRORNOAUTH);
            return returnMap;
        }
        Object o = map.get(Const.ERROR_MSG);
```

```
        if (null != o && Const.EMPTY.equals(o.toString().trim())) {
            if (o.toString().endsWith(Const.DOUHAO))
                o = o.toString().substring(0, o.toString().length() - 1);
            returnMap.put(Const.ERROR_MSG, o.toString());
        }
        if (null == map.get(Const.ROUTE_TYPE_LOW) || null == map.get(Const.NODE_TYPE))
{
            returnMap.put(Const.ERROR_MSG, Const.ERRORCONFERROR);
            return returnMap;
        }
        String routeType = map.get(Const.ROUTE_TYPE_LOW).toString();
        returnMap.put(Const.ROUTE_TYPE2, routeType);
        if (Const.TRUE_.equals(routeType) || Const.VALUE1.equals(routeType)) {
            sql.delete(0, sql.length());
sql.append(Const.SELECT_R_ID_R_TITLE_TN_NODE_TYPE_AS_TNTYPE_TN_FILTER_OP_FOR
MULA_R_DEFAULT_SELECTED_FROM_WF_DEF_NODE_ROUTE_AS_R_LEFT_JOIN_WF_DEF_N
ODE_AS_TN_ON_R_TO_NODE_ID_TN_ID_WHERE_R_VISIBLE_1_AND_R_NODE_ID_SELECT
_T_WF_NODE_ID_FROM_WF_INS_TASK_T_WHERE_T_VISIBLE_1_AND_T_ID_S_LIMIT_1);
            List<Map<String, Object>> routeList = this.wfInsTaskDao
                    .queryList(String.format(sql.toString(), new Object[]
{ taskid.toString() }));
            for (Map<String, Object> rinfo : routeList) {
                String nodeType = rinfo.get(Const.TNTYPE).toString();
                returnMap.put(Const.NODE$$TYPE, nodeType);
                if (Const.VALUE0.equals(nodeType)) {
                    String checked = Const.CHECKED2;
                    String andwhere = (null == rinfo.get(Const.FILTER_OP_FORMULA) ?
Const.EMPTY
                            : rinfo.get(Const.FILTER_OP_FORMULA).toString());
                    if (null != user && null != user.get(Const.IDLOW)) {
                        andwhere = andwhere.replace(Const.$_MY_ID,
a.escapeSql(user.get(Const.IDLOW)));
                        andwhere = andwhere.replace(Const.$$_MY_ID,
a.escapeSql(user.get(Const.IDLOW)));
                    }
                    if (null != user && null != user.get(Const.TRUENAME.toLowerCase()))
                        andwhere = andwhere.replace(Const.$_MY_NAME,
                                a.escapeSql(user.get(Const.TRUENAME.toLowerCase())));
                    if (null != System.getProperty(Const.BASE_PATH)
    && !Const.EMPTY.equals(System.getProperty(Const.BASE_PATH))) {
                        andwhere = andwhere.replace(Const.$_BASE_PATH,
System.getProperty(Const.BASE_PATH));
                        andwhere = andwhere.replace(Const.$$_BASE_PATH, new
StringBuffer(Const.DANYINHAO)
    .append(System.getProperty(Const.BASE_PATH)).append(Const.DANYINHAO).toString())
;
```

```
                    }
                    if (null != map.get(Const.BILL_ID.toLowerCase())) {
                        andwhere = andwhere.replace(Const.$_BILL_ID,
map.get(Const.BILL_ID.toLowerCase()).toString());
                        andwhere = andwhere.replace(Const.$$_BILL_ID,
map.get(Const.BILL_ID.toLowerCase()).toString());
                    }
                    if (null != map.get(Const.NODE_ID.toLowerCase())) {
                        andwhere = andwhere.replace(Const.$_NODE_ID,
map.get(Const.NODE_ID.toLowerCase()).toString());
                        andwhere = andwhere.replace(Const.$$_NODE_ID,
map.get(Const.NODE_ID.toLowerCase()).toString());
                    }
                    if (null != map.get(Const.FUN_CODE)) {
                        andwhere = andwhere.replace(Const.$_FUN_CODE, new
StringBuffer(Const.DANYINHAO)
    .append(map.get(Const.FUN_CODE).toString()).append(Const.DANYINHAO).toString());
                    }
                    andwhere = andwhere.replace(Const.$_TASKID, taskid.toString());
                    andwhere = andwhere.replace(Const.$$_TASKID, taskid.toString());
                    sql.delete(0, sql.length());
                    List<Map<String, Object>> userList = this.wfInsTaskDao
                            .queryList(String.format(Const.SQL$3, new Object[] { andwhere,
rinfo.get(Const.IDLOW) }));
                    if (null == userList)
                        userList = new ArrayList<Map<String, Object>>();
                    if (userList.isEmpty()) {
                        userList.add(new HashMap<String, Object>() {
                            {
                                this.put(Const.TEXT, Const.ERRORNOUSER);
                            }
                        });
                    }
                    rinfo.put(Const.USER_LIST, userList);
                    rinfo.put(Const.CHECKED2, checked);
                    checked = Const.EMPTY;
                    rinfo.remove(Const.FILTER_OP_FORMULA);
                }
            }
            returnMap.put(Const.ROUTE_LIST, routeList);
        }
        returnMap.put(Const.HIST, this.wfInsTaskDao.queryList(Const.SQL$1, new
WeakHashMap<String, Object>() {
            {
                this.put(Const.TASKID.toLowerCase(), taskid);
            }
```

```
        }));
        returnMap.put(Const.WFMAP,      this.wfInsTaskDao.queryList(Const.SQL$2,      new
WeakHashMap<String, Object>() {
            {
                this.put(Const.TASKID.toLowerCase(), taskid);
            }
        }));
        return returnMap;
    }
    public void doWfCommit(String token, Map<String, Object> user, Map<String, Object>
params, Integer taskid)
            throws Exception {
        if (null == params || null == params.get(Const.ROUTE_ID)) {
            throw new Exception(Const.ERRORPARAMNOROUTE);
        }
        Integer routeId = Integer.valueOf(params.get(Const.ROUTE_ID).toString());

        Integer nextOfficerId = (null == params.get(Const.NEXT_OFFICER_ID)
                ||
Const.EMPTY.equals(params.get(Const.NEXT_OFFICER_ID).toString().trim()) ? null
                        :
Integer.valueOf(params.get(Const.NEXT_OFFICER_ID).toString()));
        String      say      =      null      ==      params.get(Const.SAY2)      ?      Const.EMPTY      :
params.get(Const.SAY2).toString();
        this.doTask(token, user, taskid, routeId, nextOfficerId, say);
    }
}
package com.windowdb.wms.service;
import java.util.List;
import java.util.Map;
import java.util.WeakHashMap;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.windowdb.javascript.b;
import com.windowdb.utils.d;
import com.windowdb.utils.a;
import com.windowdb.wms.dao.BranchDictDao;
import com.windowdb.wms.dao.UserCustomerDao;
import com.windowdb.wms.dao.UserDao;
import net.sf.ehcache.Cache;
/**
 * @author 竹林春雨
 */
@Service
@Transactional
```

```
public class UserService extends AbstractedService {
    @Autowired
    protected UserCustomerDao userCustomerDao;
    @Autowired
    protected BranchDictDao branchDictDao;
    @Autowired
    protected UserDao userDao;
    public Map<String, Object> queryLimit(Map<String, Object> map) throws Exception {
        return null;
    }
    public Map<String, Object> dologin(Map<String, Object> map) throws Exception {
        String p = map.get(Const.X).toString();
        if (null == p || Const.EMPTY.equals(p.trim())) {
            throw new Exception(Const.PASSWDISNULL);
        }
        String l = map.get(Const.Z).toString();
        p = d.getInstance(l).decryptStringByJs(p);
        String md5p = a.toMD5(p);
        if (null == map.get(Const.Z) ||
map.get(Const.Z).toString().indexOf(Const.DANYINHAO) > -1) {
            map.put(Const.SUCCESS.toLowerCase(), false);
            map.put(Const.MSG.toLowerCase(), Const.LOGINFAILNODEPT);
            return map;
        }
        List<Map<String, Object>> list = this.userDao.queryList(Const.LOGIN_NAME,
map.get(Const.Z));
        if (null == list || list.isEmpty()) {
            map.putAll(this.queryPublicMenus());
            map.put(Const.SUCCESS.toLowerCase(), false);
            map.put(Const.MSG.toLowerCase(), Const.LOGINFAILNOUSER);
            return map;
        }
        String password = list.get(0).get(Const.PASSWORDLOW).toString();
        if (!password.equalsIgnoreCase(md5p)) {
            map.putAll(this.queryPublicMenus());
            map.put(Const.SUCCESS.toLowerCase(), false);
            map.put(Const.MSG.toLowerCase(), Const.LOGINFAILPASSWORDERROE);
            return map;
        }
        Map<String, Object> user = list.get(0);
        // 增加用户积分
    this.userDao.update(Const.UPDATE_PERSON_USER_SET_LOGIN_COUNT_LOGIN_COUNT
_1_INTEGRAL_INTEGRAL_COALESCE_SELECT_LOGIN_SCORE_FROM_BRANCH_COM_DEPT_
CONF_WHERE_ID_SELECT_CURRENT_DEPT_FROM_PERSON_USER_STAFF_WHERE_ID_ID_L
IMIT_1_1_WHERE_ID_ID, user);
        // 记录当日积分
```

```
        final Object uid = user.get(Const.ID);
    this.userDao.update(Const.INSERT_INTO_PERSON_USER_SCORE_LOG_UID_TODAY_IN
SERT_TIME_SCORE_TYPE_VISIBLE_DEPT_ID_NOTE_VALUES_ID_DATE_FORMAT_NOW_Y_M
_D_NOW_COALESCE_SELECT_LOGIN_SCORE_FROM_BRANCH_COM_DEPT_CONF_WHERE_I
D_SELECT_CURRENT_DEPT_FROM_PERSON_USER_STAFF_WHERE_ID_ID_LIMIT_1_1_0_1_
SELECT_CURRENT_DEPT_FROM_PERSON_USER_STAFF_WHERE_ID_ID_LIMIT_1,          new
WeakHashMap<String, Object>() {
            {
                this.put(Const.IDLOW, uid);
            }
        });
        user.put(Const.MY_SHORTCUT_MODE,
this.userDao.queryList(Const.SELECT_SM_FUNC_ID_SM_TITLE_B_HREF_FROM_SHORTCUT_
MODE_SM_INNER_JOIN_BRANCH_AUTH_B_ON_B_ID_SM_FUNC_ID_WHERE_SM_USER_ID_
ID_AND_SM_VISIBLE_1_ORDER_BY_SM_SORT_NUM, new WeakHashMap<String, Object>()
{
            {
                this.put(Const.IDLOW, uid);
            }
        }));
        user.put(Const.MY_SHORTCUT_URL,
this.userDao.queryList(Const.SELECT_SM_TITLE_SM_WEBSITE_URL_AS_HREF_SM_OPEN_M
ETHOD_FROM_SHORTCUT_URL_SM_WHERE_SM_USER_ID_ID_AND_SM_VISIBLE_1_ORDER
_BY_SM_SORT_NUM, new WeakHashMap<String, Object>() {
            {
                this.put(Const.IDLOW, uid);
            }
        }));
        user    =    this.queryUserByToken(map.get(Const.TOKEN.toLowerCase()).toString(),
list.get(0), map.get(Const.LAZY));
        map.put(Const.USER2, user);
    map.putAll(this.queryPageData(Integer.valueOf(user.get(Const.IDLOW).toString())));
        map.put(Const.SUCCESS.toLowerCase(), true);
        map.put(Const.MSG.toLowerCase(), Const.LOGINSUCCESSMSG);
        boolean synctime = super.sysTimeAndSqlIsSync(1);
        // System.out.println("synctime==========");
        // System.out.println(synctime);
        if (!synctime) {
            return new java.util.WeakHashMap<String, Object>() {
                {
                    map.put(Const.SUCCESS.toLowerCase(), false);
                    map.put(Const.MSG.toLowerCase(),
Const.ERRMSG.concat(Const.ERRCODE_003));
                }
            };
        }
```

```
        return map;
    }
    public String checkLoginName(final Object loginName, final Object id) throws Exception {
        String msg = Const.EMPTY;
        List<Map<String, Object>> list = userDao.queryList(Const.LOGIN_NAME,
loginName);
        if (null == id && list.size() > 0) {
            msg = Const.CHANGELOGINNAME;
        } else {
            for (Map<String, Object> map : list) {
                if (!id.toString().equals(map.get(Const.IDLOW).toString()))
                    msg = Const.CHANGELOGINNAME;
            }
        }
        return msg;
    }
    public String checkPrivateMobilePhone(final Object privateMobilePhone, final Object id)
throws Exception {
        String msg = Const.EMPTY;
        List<Map<String, Object>> list = userDao.queryList(Const.PRIVATE_MOBILE_PHONE,
privateMobilePhone);
        if (null == id && list.size() > 0) {
            msg = Const.CHANGEPHONE;
        } else {
            for (Map<String, Object> map : list) {
                if (!id.toString().equals(map.get(Const.IDLOW).toString()))
                    msg = Const.CHANGEPHONE;
            }
        }
        return msg;
    }
    public String checkEmail(final Object email, final Object id) throws Exception {
        String msg = Const.EMPTY;
        List<Map<String, Object>> list = userDao.queryList(Const.EMAIL, email);
        if (null == id && list.size() > 0) {
            msg = Const.CHANGEEMAIL;
        } else {
            for (Map<String, Object> map : list) {
                if (!id.toString().equals(map.get(Const.IDLOW).toString()))
                    msg = Const.CHANGEEMAIL;
            }
        }
        return msg;
    }
    public Map<String, Object> dojoin(Map<String, Object> params) throws Exception {
        Number id = null;
```

```
        if (null != params.get(Const.PASSWORDLOW))
            params.put(Const.PASSWORDLOW,
a.toMD5(params.get(Const.PASSWORDLOW).toString()));
        if (null == params.get(Const.IDLOW) ||
Const.EMPTY.equals(params.get(Const.IDLOW))) {
            params.put(Const.TITLE_LOW, params.get(Const.LOGIN_NAME));
            params.put(Const.NAME_LOW, params.get(Const.LOGIN_NAME));
            params.put(Const.WF$STATUS, 0);
            params.put(Const.TYPE.toLowerCase(), 1);
            params.put(Const.VISIBLE.toLowerCase(), 1);
            params.put(Const.INTEGRAL.toLowerCase(), 0);
            params.put(Const.ON$LINE$TIME, 0);
            params.put(Const.LOGIN$COUNT, 0);
            params.put(Const.CUST_LEVEL, branchDictDao.queryList(Const.CODE,
Const._00010005).get(0).get(Const.IDLOW));
            id = this.userDao.insertBackKey(params);
            params.put(Const.IDLOW, id);
            this.userCustomerDao.insert(params);
            this.userCustomerDao.insertLink(Const._ROLE_USER,
Const.SELECT_ID_FROM_ROLE_WHERE_CODE_00020000, id.toString());
        } else {
            this.userDao.update(params);
        }
        params.putAll(this.queryPublicMenus());
        return params;
    }
    public void dologout(String token) {
        Cache cache = b.getCaManager().getCache(Const.TOKENS);
        cache.remove(token);
    }
}
```