Noka Laboratory

Noka of China Co., Ltd.

BS 研究中心

BS Research Center

Noka Pro 开发手册

Noka Pro Development Manual

文档类别	技术手册
版　　本	V6.0
密　　级	不涉密
起 草 人	谢方建
审核部门	
确认部门	

二〇一八年三月

使用申明

本软件知识产权归诺亚(中国)开源科技所有，Noka Pro 为商业软件仅限于以学习或研究为目的使用，不能以任何形式使用或嵌入到其它商业软件内部使用。

本软件作为免费软件发布，不对其软件的使用承担任何责任，如有需要以商业软件发布或技术支持欢迎咨询具体发布事宜

Noka Laboratory
Rebin

网址：http://www.97521.com
电话：(028)85145708，85143878，85147698
电邮：xiefangjian@163.com

目录

第1章 Noka Pro 部署

1.1 Noka Pro 6.X 新特性

Noka Pro 6.X是基于Noka Tag 6.0研发的企业信息系统快速开发平台，集成了系统管理基础功能（包括基础字典、树形字典、权限管理、个人信息）可以快速组建任何企业信息系统所需要的基础架构。

Noka Pro 6.X第三方支撑平台

名称	版本	备注
JDK	1.8.0(以上)	
Tomcat	8.5(以上)	需要支持jsp2.0标准
WebLogic	9.2(以上)	
resin	4.0(以上)	

Noka Pro 6.X支持的数据列表。

数据库	版本	备注
MySQL	5.0(以上)	
由于采用了MyBatis框架，其它数据库可能需要对Mappear中的SQL语句进行微调		

Noka Pro 6.X 浏览器兼容列表

浏览器名称	版本	备注
Internet Explorer	8.0(以上)	
Mozilla Firefox	2.0(以上)	
Netscape Navigator		
Opera		
Apple Safari		
Google Chrome		

Noka Pro 6.X 技术架构清单

技术架构名称	版本	备注
Spring MVC	5.0	MVC框架
MyBatis	3.4.2	持久化框架
druid	1.0.9	数据库连接池
quartz	2.2.3	定时任务调度
Noka tag	6.0.20	标签库

1.2 Noka Pro 配置

Noka Pro 6.X 的配置非常简单，具体步骤如下：

1、在db目录下选择你所熟悉的数据库所对应的目录，执行里面的创建表sql语句，然后在执行inisql目录下的系统初始化sql语句。如果没有你需要的sql脚本请用PowerDesigner打开pord目录下的文件生成你需要的数据库sql脚本。

2、用你所熟悉的开发工具新建一个Java Web项目，导入NokaPro6.0下的src目录，该项目为Maven项目。

3、配置完成以后，就可以发布你的项目到相应的应用服务器上进行测试，系统初始登录用户名和密码都为admin。

4、在src目录下的system_config.xml文件里面，是noka pro的系统配置文件，内容看起来可能会是下面这样的：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE noka PUBLIC
    "-//noka//DTD noka-pro Configuration 3.2//EN" "system_config.dtd">
<xmlconst>
<!-- 数据列表每面显示记录总数  -->
<option key="dbtable-page-size"><![CDATA[10]]></option>
<option key="default-language"><![CDATA[zh-cn]]></option><!-- 默认语言 -->
<option key="path-language"><![CDATA[language]]></option><!--语文件路径 -->
<option
key="no-sessionCheck"><![CDATA[userlogin.xhtml,setnowurl.xhtml,login.xhtml,index.jsp,codeimg_nsk.codeimg,noka.ico,webchecnk.chk,dataserver.ser]]></option><!--排除Session 验证的路径,以逗号分隔 -->
<option
key="session-publick"><![CDATA[newmain.xhtml,mypasswordupdate.xhtml,mypassword.xhtml,ntreeselect_*.tre]]></option><!--需要 session 才能访问的共公路径 -->
<option key="no-sessionUrl"><![CDATA[${rooturl}/login.xhtml]]></option><!--session 验证失败重定向路径 -->
<option key="no-sqlUrl"><![CDATA[${rooturl}/login.xhtml]]></option><!--sql 验证失败重定向路径 -->
<option key="cache-file"><![CDATA[true]]></option><!--缓存文件 -->
<logs filename="nokapro.log" path="D://log//sys//" day="5"/><!--日志切割目录
linx :/home/user/xxx/xx/ -->
<redis port="6379" url="192.168.254.29"></redis><!-- redis 配置参数预留配置 -->
<dbtable-filters><!-- 数据过滤  key 指过滤 sql 的 key 一般可指表名,user-field 用户 id 字段,dept-field 用户所在部门字段 lan 语言文件的 key select 对应于数据过滤设置界面上的 5 个选项，为 false 表示不显示该选项 -->
    <table key="NK_SYS_WORDBOOK" user-field="WUSER" dept-field="WCWORKDEPT"
lang="org.noka.sys.worktitle"></table>
    <table key="NK_SYS_USERINFO" user-field="USID" dept-field="USWORK"
lang="org.noka.sys.usertitle" select="true,true,true,false,true"></table>
</dbtable-filters>
<additional-permissions>
    <!-- 附加权限配置 -->
    <permissions code="showmobile" title="org.noka.sys.showmobile"></permissions>
</additional-permissions>
</xmlconst>
```

第2章 Noka Pro 开发入门

2.1 NokaPro 目录规范说明

1、src（源码）目录约定规范说明

目录名称约定	作用
org.noka.aoplog	日志处理类所在目录，对应配置文件spring-aoplog.xml
org.noka.constvar	系统全局常量/变量类所在目录
org.noka.filter	Noka Pro系统拦截器所在目录
org.noka.function	系统级公共方法类所在目录
org.noka.vo	数据库实体类及其实体类配置文件所在目录
org.noka.controller	Spring MVC控制层目录
org.noka.idwork	ID主键生成器所在目录
org.noka.mapper	MyBatis实体类所在目录
org.noka.service	业务逻辑层所在目录
org.noka.session	Redis方式共享session所在目录，默认不开启
org.noka.tlg	系统级自用标签库实现类所在目录

2、配置文件说明

文件名称	作用
noka-config.xml	Noka tag标签库配置文件
nokapro.properties	系统属性配置文件，默认只有数据库连接信息配置
spring-aoplog.xml	Spring自定义日志注解配置文件
spring-mvc.xml	Spring MVC主配置文件
spring-mybatis.xml	Spring mybatis配置文件
spring-quartz.xml	Spring auartz定时任务调度配置文件，默认日志切割任务
spring-quartxjobs.xml	功能同spring-quartz.xml,在分布式环境下使用该配置
spring-elasticse.xml	搜索引擎配置文件，默认未开启，开启后在service层可直接通过注解来使用elasticsearchTemplate操作搜索引擎
spring-mongodb.xml	Mongodb配置文件，默认未开启
system_config.xml	系统配置文件

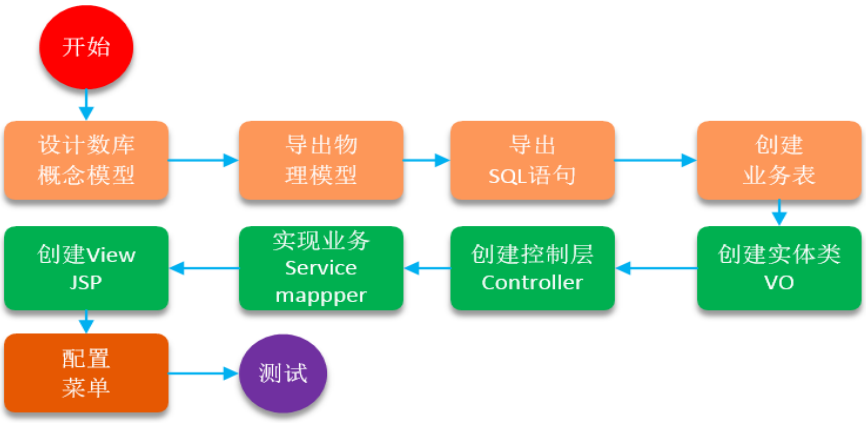
2、WEB（展现层）目录约定规范说明

目录名称约定	作用
inc	系统公共引用JSP所在目录
language	系统语言文件所地目录
main	系统登录界面及主界面所在目录
peopselect	系统级公用用户选择界面所在目录
popedom	用户及角色权限管理界面所在目录
script	系统javascript脚本所在目录
skins	系统皮肤文件所在目录
treebook	树形字典管理界面所在目录
user	系统当前用户修改自己密码及其基础信息界面所在目录
wordbook	基础字典管理界面所在目录

2.2 典型数据库操作功能开发实例

数据库典型的开发实例包括对数据表的增、删、改、查四大功能，Noka Pro是基于Spring MVC研发的基础二次开发平台，因此所有功能的研发应当遵循Spring MVC的各项编码规则，本平台采用三层架构以减少代码的复杂程度，并在一定程度提高系统响应性能。

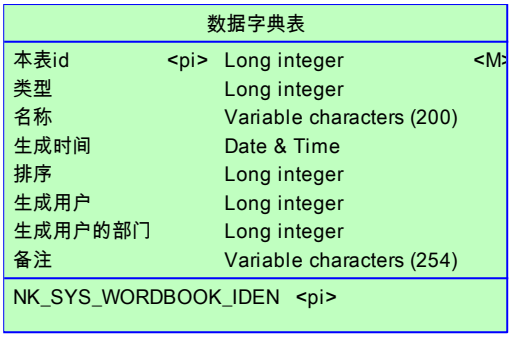
Noka Pro开发一般遵循以下开发步骤，其中部分步骤可以根据个人习惯前后调换，但在开发过程中必须遵循后面的编码规则，否则不能确保整个系统的高性能及其稳定性。



（图2.2-01 一般开发流程图）

Step 1

根据流程图我们第一步选设计业务所需要的数据表并建好数据库表，在本实例中我们假定需要一公共字典这样一个功能，既对字典信息进行增加、删除、修改、查询等功能，我们所涉及的字典信息表如下：



（图2.2-02 字典息表概念模型）

在设计数据库表时预留几个备用字段是一个比较好的习惯，便于业务扩展，另外在Noka Pro里数据表有三个字段在设计时建议作为必须字段，这三个字段为操作人ID，操作人所在部门ID，操作时间，上图所示的实例中对应的为登记人ID，

登记人所属部门，登记时间，关于更多Noka Pro数据表设计规范详见本手册附件。

Step 2

建立数据表及其实体类和对应的配置文件，这里的我们以MySQL为例，对应的实体类建议用第三方小工具自动生成，本例的人员信息表实体类及其配置文件

```
//1、实体类，注意：所有实体类都应该是可被序列化的
/**
 * <p>@Title 基础字典表</p>
 * <p>@Description </p>
 * <p>@Version 1.0.0</p>
 * <p>@author rebin</p>
 * <p>@date 2016 年 7 月 15 日</p>
 * <p>xiefangjian@163.com</p>
 * <p>Copyright © Noka Laboratory.All Rights Reserved.</p>
 */
public class WordBookVo implements Serializable {
    private static final long serialVersionUID = 5995819613372503528L;
    private Long wid = null;// bigint identity ;
    private Long wtype = null;// bigint ;
    private String wname = null;// varchar ;
    private String wtext = null;// varchar ;
    private Date wcdate = null;// datetime ;
    private Integer wcompositor = null;// int ;
    private Long wuser = null;// bigint
    private Long wcworkdept = null;// bigint ;
    public Long getWid() {
        return wid;
    }
    public void setWid(Long wid) {
        this.wid = wid;
    }
    public Long getWtype() {
        return wtype;
    }
    public void setWtype(Long wtype) {
        this.wtype = wtype;
    }
    public String getWname() {
```

```
        return wname;
    }
    public void setWname(String wname) {
        this.wname = wname;
    }
    public String getWtext() {
        return wtext;
    }
    public void setWtext(String wtext) {
        this.wtext = wtext;
    }
    public Date getWcdate() {
        return wcdate;
    }
    public void setWcdate(Date wcdate) {
        this.wcdate = wcdate;
    }
    public Integer getWcompositor() {
        return wcompositor;
    }
    public void setWcompositor(Integer wcompositor) {
        this.wcompositor = wcompositor;
    }
    public Long getWuser() {
        return wuser;
    }
    public void setWuser(Long wuser) {
        this.wuser = wuser;
    }
    public Long getWcworkdept() {
        return wcworkdept;
    }
    public void setWcworkdept(Long wcworkdept) {
        this.wcworkdept = wcworkdept;
    }
}
//2、对应增、删、改 mapper
/**
 * ********************************************************
 * <p>@Description 基础字典操作</p>
 * <p>@Version 1.0.0</p>
```


网址：http://www.97521.com
电话：(028)85145708，85143878，85147698
电邮：xiefangjian@163.com

```
 * <p>@author rebin</p>
 * <p>@time 2017 年 4 月 1 日 上午 8:45:10</p>
 * <p>xiefangjian@163.com</p>
 * <p>Copyright © Noka Laboratory.All Rights Reserved.</p>
 * ******************************************************
 */
public interface WordbookMapper {
    //---------------------新增字典-----------------------------------------
    @Insert("INSERT INTO
NK_SYS_WORDBOOK(WID,WTYPE,WNAME,WCDATE,WCOMPOSITOR,WUSER,WCWORKDEPT,WTEXT) VALUES
(#{wid},#{wtype},#{wname},#{wcdate},#{wcompositor},#{wuser},#{wcworkdept},#{wtext})
")
    Integer  insertWordBook(WordBookVo wordbook);
    //---------------------修改字典-----------------------------------------
    @Update("UPDATE NK_SYS_WORDBOOK SET
WTYPE=#{wtype},WNAME=#{wname},WCDATE=#{wcdate},WCOMPOSITOR=#{wcompositor},WUSER=#{w
user},WCWORKDEPT=#{wcworkdept},WTEXT=#{wtext} WHERE WID =#{wid}")
    Integer  UpdateWordBook(WordBookVo wordbook);
    //---------------------批量删除字典-------------------------------------
    @Delete("<script>DELETE FROM NK_SYS_WORDBOOK WHERE WID IN <foreach  item='id'
collection='ids' open='(' separator=',' close=')'> #{id} </foreach> </script>")
    Integer deleteWordBooks(@Param("ids") String[] ids);
}
```

Step 3

创建业务Controller，我们的业务Controller主要涉及数据展现、增加、删除、修改、查询五个功能，其中查询交由noka-tag标签自动完成，Controller里面只需要实现前四个功能即可，下面来看我们的Controller类实例：

```
/**
 * <p>@Title </p>
 * <p>@Description 字典管理</p>
 * <p>@Version 1.0.0</p>
 * <p>@author rebin</p>
 * <p>@date 2017 年 3 月 29 日</p>
 * <p>xiefangjian@163.com</p>
 * <p>Copyright © Noka Laboratory.All Rights Reserved.</p>
 */
@Controller
```

```
@RequestMapping("/wordbook")
public class WordBookController extends BaseController{
    @Autowired
    private WordBookService wordBookService;
    /**
     * <p>@Description 字典管理主界面</p>
     * <p>author rebin </p>
     * <p>@Time 上午 9:04:15</p>
     * <p>xiefangjian@163.com</p>
     * @param muid
     * @param request
     * @return
     */
    @RequestMapping(value = "/index")
    public String index(String muid,HttpServletRequest request) {
        Long wordt=Long.parseLong(muid);
        String SQL="SELECT WID AS "+$L("org.word.jsp.id")+",WNAME AS
"+$L("org.word.jsp.name_"+wordt)+", WTEXT AS
"+$L("org.word.jsp.notes")+",WCOMPOSITOR as
"+$L("org.word.jsp.pxun")+",WCWORKDEPT,WCDATE,WUSER FROM NK_SYS_WORDBOOK WHERE
WTYPE='"+wordt+"'";
        request.setAttribute("sql",  sqlFilter(SQL, "NK_SYS_WORDBOOK"));
        return "wordbook/input";
    }
    /**
     * <p>@Description 添加字典</p>
     * <p>author rebin </p>
     * <p>@Time 上午 9:04:24</p>
     * <p>xiefangjian@163.com</p>
     * @param wordbook
     * @param request
     * @param response
     * @return
     */
    @NLog("添加字典信息")
    @ResponseBody
    @RequestMapping(value = "/wordbookadd",method = RequestMethod.POST)
    public ControllerMsgVo wordbookadd(@ModelAttribute("wordbook") WordBookVo
wordbook,HttpServletRequest request,HttpServletResponse response){
        return wordBookService.insertWordBook(wordbook);
    }
```


网址：http://www.97521.com
电话：(028)85145708，85143878，85147698
电邮：xiefangjian@163.com

```
    /**
     * <p>@Description 修改字典</p>
     * <p>author rebin </p>
     * <p>@Time 上午 9:04:35</p>
     * <p>xiefangjian@163.com</p>
     * @param wordbook
     * @param request
     * @param response
     * @return
     */
    @NLog("修改字典信息")
    @ResponseBody
    @RequestMapping(value = "/wordBookupdate",method = RequestMethod.POST)
    public ControllerMsgVo wordBookupdate(@ModelAttribute("wordbook") WordBookVo
wordbook,HttpServletRequest request,HttpServletResponse response){
        return wordBookService.UpdateWordBook(wordbook);
    }
    /**
     * <p>@Description 删除字典</p>
     * <p>author rebin </p>
     * <p>@Time 上午 9:04:51</p>
     * <p>xiefangjian@163.com</p>
     * @param ids
     * @param request
     * @param response
     * @return
     */
    @NLog("删除字典")
    @ResponseBody
    @RequestMapping(value = "/wordBookdel",method = RequestMethod.POST)
    public ControllerMsgVo wordBookdel(@RequestParam("wordrows") String
ids,HttpServletRequest request,HttpServletResponse response){
        return wordBookService.deleteWordBooks(ids);
    }
}
```

Step 4

Controller建完后，需要编写业务服务层，服务层继承BaseService超类，该类实现了一些基础工具方法，服务层代码注解采用Spring的@Service，服务层代码如下：

```
@Service
public class WordBookService extends BaseService{
    @Autowired
    private WordbookMapper wordbookMapper;
    /**
     * <p>@Description 添加字典</p>
     * <p>author rebin </p>
     * <p>@Time 下午 4:45:43</p>
     * <p>xiefangjian@163.com</p>
     * @param wordbook
     * @return
     */
    public ControllerMsgVo  insertWordBook(WordBookVo wordbook){
        try{
            if(null!=wordbook){
                wordbook.setWid(nextID());
                if(wordbookMapper.insertWordBook(wordbook)>0){
                    return new ControllerMsgVo(1);
                }
            }
        }catch(Exception se){
            se.printStackTrace();
        }
        return new ControllerMsgVo(2);
    }
    /**
     * <p>@Description 修改字典</p>
     * <p>author rebin </p>
     * <p>@Time 下午 4:46:36</p>
     * <p>xiefangjian@163.com</p>
     * @param wordbook
     */
```

```
    public ControllerMsgVo  UpdateWordBook(WordBookVo wordbook){
        try{
            if(null!=wordbook){
                if(wordbookMapper.UpdateWordBook(wordbook)>0){
                    return new ControllerMsgVo(3);
                }
            }
        }catch(Exception se){
            se.printStackTrace();
        }
        return new ControllerMsgVo(4);
    }
    /**
     * <p>@Description 删除字典</p>
     * <p>author rebin </p>
     * <p>@Time 下午 4:48:05</p>
     * <p>xiefangjian@163.com</p>
     * @param ids
     */
    public ControllerMsgVo deleteWordBooks(String ids){
        try{
            if(null!=ids && ids.trim().length()>0){

    if(wordbookMapper.deleteWordBooks(ids.split(","))==ids.split(",").length){
                    return new ControllerMsgVo(5);
                }
            }
        }catch(Exception se){
            se.printStackTrace();
        }
        return new ControllerMsgVo(6);
    }
}
```

Step 5

Service完成以后，我们需要编写展现层的jsp页面，在nokapro里jsp页面控件由noka-tag代替struts的标签，每个jsp页面必须加载

<%@ include file="/inc/inc.jsp"%>（需要登录验证的）

或者是

<%@ include file="/inc/nosession_inc.jsp"%>（不需要登录验证的）

本实例的jsp页面如下所示：

```
<%@ page language="java"  pageEncoding="UTF-8"%>
<html>
  <head>
    <title></title>
    <%@ include file="/inc/inc.jsp"%>
    <n:script src="script/wordbook/input.js"></n:script>
    <n:link href="skins/css/body.css" rel="stylesheet"></n:link>
  </head>
<body>
<b:lan key="org.word.jsp.name_${muid}" value="nk_word_input"/>
<!-- 表单 start-->
<div>
<n:form action="" method="post" id="wordbookform" name="wordbookform"
onsuccess="onsuccess" onfailure="onfailure" classstyle="form">
<n:Hiddens
json="[{name:'wid',id:'wid_id'},{name:'wtype',value:'${muid}'},{name:'wcworkdept',v
alue:'${user.uswork}'},{name:'wuser',value:'${user.usid}'}]"
id="hidden_ids"></n:Hiddens>
<n:frow classstyle="form_row" >
    <n:fcell label="${nk_word_input}：" labelwidth="80px" classstyle="form_label"
inputclassstyle="form_input">
        <n:InputText disabled="${b:ButIf(request,'add||update','no:disabled')}"
width="240" name="wname" id="wname_id" allownull="false" maxlength="20"
mesg="${org.noka.sys.nonull}" classstyle="text"
ondblclick="DoubleClickClear(this)"/>
    </n:fcell>
    <n:fcell label="${org.word.jsp.pxun}：" labelwidth="80px" classstyle="form_label"
```

```
inputclassstyle="form_input">
        <n:InputText disabled="${b:ButIf(request,'add||update','no:disabled')}"
width="240" name="wcompositor" id="wcompositor_id" maxlength="20"
mesg="${org.word.jsp.intnonull}" classstyle="text"  allownull="false" chtype="regex"
veri="[1-9]{1,3}" ondblclick="DoubleClickClear(this)"/>
    </n:fcell>
</n:frow>
<n:frow classstyle="form_row_text">
    <n:fcell label="${org.word.jsp.notes}： " labelwidth="80px"
classstyle="form_label_text" inputclassstyle="form_input_text">
        <n:Textarea  name="wtext" id="wtext_id" width="700"
disabled="${b:ButIf(request,'add||update','no:disabled')}"  cols="50" rows="5"
classstyle="text"></n:Textarea>
    </n:fcell>
</n:frow>
<n:frow classstyle="form_row_button">
  <b:button id="add_id" name="add" classstyle="but" onclick="addss('${rooturl}')"
value="${org.noka.sys.add}" style="margin-left:-40px;"/>
  <b:button id="update_id"  name="update" style="display: none;margin-left:30px;"
classstyle="but"  onclick="updates('${rooturl}')" value="${org.noka.sys.update}"/>
</n:frow>
<n:forminfo/>
</n:form>
</div>
<div style="top:260px;left:10px;right:10px;bottom:10px;  position: absolute;">
<n:DBGrid

inputData="[{field:'wid_id',value:1},{field:'wname_id',value:2},{field:'wcompositor
_id',value:4},{field:'wtext_id',value:3}]"
    formConfirmation="${org.noka.sys.isdel}"
    formNull="${org.noka.sys.dellselect}"
    width="700"
    height="300"
    id="wordDb"
    checkrecord="wordrows"
    onfailure="onfailure"
    onsuccess="onsuccess"
    sql="${sql}"
    selectfiled="${nk_word_input} like %$[wdname,${nk_word_input}]%  |
#[ano,and^${org.noka.sys.tand}!or^${org.noka.sys.tor}]# ${org.word.jsp.notes} like
'%$[d_c,${org.word.jsp.notes}]%'"
```


网址：http://www.97521.com
电话：(028)85145708，85143878，85147698
电邮：xiefangjian@163.com

```
    selectinput="true"
    title="${org.word.jsp.wordlist}${b:ButIf(request,'add||update',org.noka.sys.dbcortitle)}"
    outtitle="打印标题"
    tableformname="dataform"
    checkname="wordids"
    cells="width:100,show:4|width:200,show:1|width:200,show:1"
    checkd="checkbox"
    onrowdblclick="detail"
    pagesize="50"
    tableformid="id_dataform"
    tableformAction="${rooturl}/wordbook/wordBookdel.xhtml"
    compositor="WCDATE"
    descorasc="desc"
    outpdfall="true"
    outpdfnopage="true"
    outexcelall="true"
    hideselect="${b:ButIf(request,'delete','false:true')}"
    custombutton="${b:but(request,'delete',org.noka.sys.dell,'wordDb_ajaxSubmitDataForm()')}"
    outfilename="基础字典表$[yyyy-MM-dd]"
    configid="${user.usid}"
/>
</div>
</body>
</html>
```

Step 6

为业务配置菜单，该步骤是将以上开发的功能配置到系统的菜单管理里面，在数据表NK_SYS_MENU中进行配置，该表是一个树形的表，系统只支持三级菜单，最后一级没有菜单图标，具体说明可参照NokaPro数据表设计文件。

配置完成菜单以后即可登录系统，先要权限管理里面将新配置的菜单赋予当前用户，刷新界面以后就可以看到新增加的功能。

第3章 Noka Pro 开发说明

3.1 Noka Pro 菜单配置说明

Noka Pro菜单配置以系统菜单表为主要配置对象，菜单在系统中分共分为三级，其中一、二级显示在左边以折叠式菜单展现出来，第三级菜单在右边以选项卡的方式展现出来。菜单配置表在NK_SYS_MENU表中，具体配置如下：

字段名称	说明
MENUID	菜单表ID，全表唯一，不可重复
MENUPID	菜单表的父ID，它与MENUID一起形成组成树状结构
MENUNAME	菜单的名称，这里写入的是lanuage中的key
MENUURL	菜单的URL地址，${rooturl}代表URL全路径
MENUOPERATE	该菜单下的按钮组
MENUIMAGE	菜单图标，只有二级菜单有用，需要用${skin}指代皮肤文件
MENUTAXIS	菜单在同级菜单中的显示先后顺序
MENUTARGET	菜单的连接目标，只有二级菜单有用，固定值mainFrame
MENUNBURL	该菜单下需要访问的公共路径，表示拥有该菜单权限即可访问
MENULEVE	该菜单在树形菜单中的等级关系

（1）MENUURL一般需建议写全路径，

三级菜单写法：如一个菜单的访问地址为abc.nk（注意Noka Pro的地址后缀为xhtml），菜单ID为12，则该菜单的URL可写为${rooturl}/abc.xhtml。

二级菜单写法：二级菜单为url地址，可以不写。

一级菜单写法：一级菜单的URL可为空，只需要ID和名字即可。

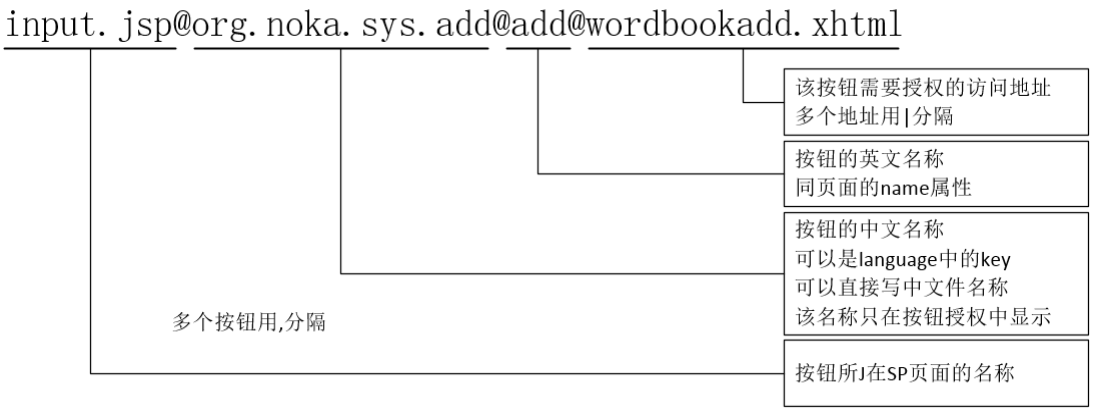
（2）MENULEVE由菜单的ID和父ID组成树形菜单的等级关系，其值格式为：

父ID的MENULENE_本级ID，如果是一级菜单为固定值-1_0_本级ID。

注：建议用系统设置中的菜单管理功能去操作，各项值参考基础字典的配置。

3.2 按钮配置说明

按钮可以控制操作权限，在页面中用<b:button>标签进行控制，使用方法和普通按钮相同，按钮权限配置如下：



3.3 Noka Pro 页面工具说明

Noka Pro页面工具主要为b标签系列工具，其它工具如noka-tag标签、c标签详见相对应的官方文档。

（1）B标签，该标签为按钮标签，自身具有权限控制功能，除以下几个特殊属性以外，其它属性同普通按钮标签属性等同。

属性	是否必须	说明
classstyle	否	等同于普通按钮的class属性
disabled	否	等同于普通按钮的disabled属性，其值为disabled时生效

（2）lan标签，语言文件读取标签，根据浏览器语言自动选择，传入值为语言文件的KEY。

属性	是否必须	说明
key	是	语言文件中所配置的key
value	否	该属性为变量名称，如设置该值，lan标签将根据key所取得的内容以value所示的变量名称设置在request中，通过${变量名}可取得

（3）but方法，该方法为JSTL方法，写法如下：

${b:but(request,'delete',org.noka.sys.dell,'dels()')}

第一个参数为按钮的自身的ID。

第二个参数为按钮的显示名称，这里传入的是语言文件的KEY。

第三个参数是按钮点击时调用的javascript方法。

（4）ButIf方法，该方法为JSTL方法，写法如下：

${b:ButIf(request, 'delete','false:true')}

第一个参数为按钮所在jsp页面的文件名称加检测值，其中检测值是指按钮在数据库里配置的标识，根据按钮标识检测是否有该按钮权限，支持多个条件，条件判断可以用&&表示并且，用||表示或者。如delte||add表示如果有删除或是添加按返回对应的值

第二个参数为返回值，当第二个参数验证成功以后返回:号前面的，失败返回:后面的。

（5）在任何JSP页面上均可以通过${user}获取当前登录用户的详细信息。

3.4 Noka Pro 多语言配置说明

默认状态下，语言文件存放在language目录下，语言文件命名规则遵循国际化支持的命名规则，其名称为language_具体语言，如简体中文则为language_zh-CN.xml。

```
<page id="org.noka.sys" description="系统全局信息">
<option key="dbcortitle" description="双击修改"><![CDATA[[双击修改]]]></option>
</page>
```

Page节点是对language的分组，一般来说不同的分级具有不同的功能，通常建议一个功能分一个组。

Id是指该组的ID，全文件唯一，不可重复。

Description仅作说明使用无实际意义，可选。

Option为language的具体配置项。

Key是指待该分组下的语言配置key，包内唯一。

![CDATA[[实际的内容]]]

在任何地方获取多语言信息时，应当是包的id加上包内的key共同组成获取语言内容的key，如上例所示，应为org.noka.sys.dbcortitle。

第4章 附件

4.1 JS 编码规范

Noka pro内置jquery-1.12.4.js和prototype-1.6.0.3.js两个js库，并做了兼容性处理，在js选择器中可分别独立使用，也可以一起使用，具体使用方法如下：

```
/* ----------------------------------------------------------------------
jquery 库的$方法被重命名为 j 方法,如 j()与原 jquery 的$()等同使用,即 j()获取的为 jquery 扩展对象
$()方法特指 prototype 库的特有方法，即$()获取的为 prototype 扩展的对象
在 jquery 扩展对象中的 context 内有 prototype 的扩展对象,所有的 noka 标签扩展方法都内置在 context 对象中，如:
j('#wordbookform').context.ajaxSubmit()与$('wordbookform').ajaxSubmit()等同
----------------------------------------------------------------------*/
jQuery(document).ready(function($){//jquery 闭包方式依然有效
    //这里内部的$()为 jquery 的方法
    $('#sfffs').context.hello();//context 为 prototype 扩展对象
});
j(document).ready(function($){//jquery 闭包方式依然有效
    //这里内部的$()为 jquery 的方法
    $('#sfffs').context.hello();//context 为 prototype 扩展对象
});
$('sfffs').hello();//直接使用$()获取的为 prototype 对象
```


网址：http://www.97521.com
电话：(028)85145708，85143878，85147698
电邮：xiefangjian@163.com

4.2 Noka Pro 编码规范

A、在Controller、Service内部或是其它通过http调用的java方法内部，必须用Spring注解的方式进行操作，直接操作数据库的，其数据库连接对象必须使用noka-tag的ServletNokaContext对象获取，使用完以后，不能关闭。

B、切记不能在嵌套、递归、循环内操作数据库。

C、格式化字段、数据码表等尽量根据业务需要，通过创建SQL方法完成。

4.3 Noka Pro 数据库设计规范

（1）数据库、数据表、字段命名一率采用全大写命名。

（2）数据库、数据表名称必须带NK_前缀。

（3）数据表如果是业务表必须使用NK_OPE_前缀，系统表必须使用NK_SYS_前缀。

（4）数据表字段必须带表名前缀，前缀取表名前两个字母。

（5）所有数据表必须预留5个以上的备用字段，字段类型为VARCHAR(254)。

（6）所有数据表必须设置三个系统字段，分别为数据录入人的ID，数据录入人所在部门ID，录入时间。

（7）所有表必须有一个以上的主键。

（8）数据库涉及码表的，一率使用数据字典或树形字典进行配置，不得在新建码表。