|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 文件编号： |  | | | 密级： | 普通 | | |
| 文档类型： | 配置开发平台用户指南 | | | 文档用途： |  | | |
| 总 页 数： |  | 正 文： |  | | | 附 录： |  |
| 编 制： |  | | | | | | |

配置开发平台用户指南

中油龙慧自动化工程有限公司

2013年2月

修订历史版本

|  |  |  |
| --- | --- | --- |
| **日期** | **修订内容** | **修订人** |
| 2012-12-05 | 创建文档 | 陈春晖 |
| 2013-02-17 | 补充配置开发平台首页配置 | 陈文明 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目 录

第一章 文档概述 1

1.1 文档目的 1

1.2 预期读者 1

1.3 术语与缩略语解释 1

1.4 涉众 1

1.5 参考资料 1

第二章 数据表设计约定 2

2.1 表名 2

2.2 字段名 2

2.3 系统字段 2

2.4 字段类型约束 2

2.5 字段注释 3

2.6 外键 3

2.7 多对多对应关系 3

2.8 一对多对应关系 3

第三章 流程 4

3.1 流程按钮 4

第四章 常用配置 5

4.1 首页配置 5

4.1.1 默认打开首页，保留工作台 5

4.1.1.1 效果图 5

4.1.1.2 配置步骤 5

4.1.2 默认打开首页，不保留工作台 8

4.1.2.1 效果图 8

4.1.2.2 配置步骤 8

4.2 一体化组件配置 9

4.2.1 配置工作 9

4.3 消息提醒配置 10

4.3.1 消息提醒 10

4.3.2 菜单配置 10

4.3.3 桌面配置 11

4.4 顶部下拉面板配置 11

4.5 自定义登录配置 12

4.5.1 平台默认的登录逻辑 12

4.5.2 自定义登录逻辑 12

4.6 日期框配置 14

4.6.1 日期框的基本配置 14

4.6.2 对日期框的格式进行控制 14

4.6.3 自定义日期格式配置 15

4.6.4 例子 15

4.7 只读配置 15

4.7.1 列表只读而卡片非只读配置 15

4.7.2 主服务只读而子服务非只读配置 16

4.7.3 主服务非只读而子服务只读配置 17

4.8 系统提示信息使用 17

4.8.1 通用使用方式 17

4.8.2 在扩展list.js中使用 18

4.8.3 在扩展card.js中使用 18

4.9 主子表组件配置 18

4.10 任务调度的使用配置 19

4.10.1 任务配置 19

4.10.2 任务类的编写 20

4.10.3 调度执行 21

4.11 高级查询配置 21

4.11.1 设定高级查询字段 21

4.11.2 查询字段高级配置 22

4.12 工作流监听与RTX通知 24

4.12.1 工作流监听配置 24

4.12.2 RTX通知 30

4.13 cas单点登录配置 30

4.13.1 服务器端配置 30

4.13.1.1 准备工作 30

4.13.1.2 添加数据源配置 31

4.13.1.3 定义密码加密方式 31

4.13.1.4 获取用户信息 31

4.13.1.5 取消https验证 34

4.13.2 客户端配置 34

4.13.2.1 准备工作 34

4.13.2.2 添加单点登录过滤配置 34

4.13.2.3 平台登出配置 37

4.13.2.4 客户端过滤器 37

第五章 平台前端 38

5.1 视图 38

5.1.1 卡片视图 38

5.1.1.1 js文件 38

5.1.2 列表视图 38

5.1.2.1 js文件 38

5.1.3 字典选择弹出框视图 38

5.1.3.1 js文件 38

5.1.4 工作流卡片视图 38

5.1.4.1 js文件 38

5.1.5 首页视图 38

5.1.5.1 js文件 38

5.1.6 门户视图 38

5.1.6.1 js文件 38

5.1.7 下拉框视图 39

5.1.7.1 js文件 39

5.2 组件 39

5.2.1 卡片 39

5.2.2 表单 39

5.3 其他 39

第六章 平台后端 40

6.1 参数 40

6.1.1 ParamBean 40

6.1.2 OutBean 40

6.2 服务扩展 40

6.2.1 BaseServ 40

6.2.2 CommonServ 40

6.3 服务调用 40

6.4 数据库 40

6.4.1 SqlBean 40

6.4.2 数据库链接 40

第七章 系统约束 41

7.1 公司编码 41

7.2 分组框 41

1. 文档概述
   1. 文档目的
   2. 预期读者

* 设计开发人员
* 软件测试人员
  1. 术语与缩略语解释
  2. 涉众
  3. 参考资料

1. 数据表设计约定
   1. 表名

所有的表名必须带有项目前缀，如生产运营系统项目简称为“TRQ\_SC”,则名称为TABLE的完整表名为“TRQ\_SC\_TABLE”。

* 1. 字段名

所有的字段名，除系统字段，都需以除去项目前缀的表名为前缀，如表名为“TRQ\_SC\_TABLE”的ID字段，其全名为“TABLE\_ID”。

* 1. 系统字段

一般数据表默认需包含以下字段：

|  |  |  |
| --- | --- | --- |
| 字段编码 | 字段类型 | 字段注释 |
| S\_PUBLIC | NUMBER(4) | 公共标志，1：是；2：否 |
| S\_FLAG | NUMBER(4) | 启用标志，1：是；2：否 |
| S\_CMPY | VARCHAR2(40) | 所属公司 |
| S\_DEPT | VARCHAR2(40) | 所属部门 |
| S\_USER | VARCHAR2(40) | 创建人 |
| S\_MTIME | VARCHAR2(30) | 更新时间 |

当该表记录处于流程中时，需要包含以下字段：

|  |  |  |
| --- | --- | --- |
| 字段编码 | 字段类型 | 字段注释 |
| S\_WF\_INST | VARCHAR2(40) | 流程实例 |
| S\_WF\_STATE | NUMBER(4) | 流程状态，1：未结；2：已结 |
| S\_WF\_USER | VARCHAR2(2000) | 流程用户 |
| S\_WF\_NODE | VARCHAR2(2000) | 流程节点 |

* 1. 字段类型约束
* 主键类型为varchar2(40)
* 日期字段类型为varchar2(30)
* 数据字典类型为varchar2(40)
* 超大文本类型必须设定为LONG
  1. 字段注释

字段的注释默认会读取为该字段的名称，字段的详细注释在字段名称之后以逗号隔开。

* 1. 外键

创建表时不得创建外键关联

* 1. 多对多对应关系
  2. 一对多对应关系

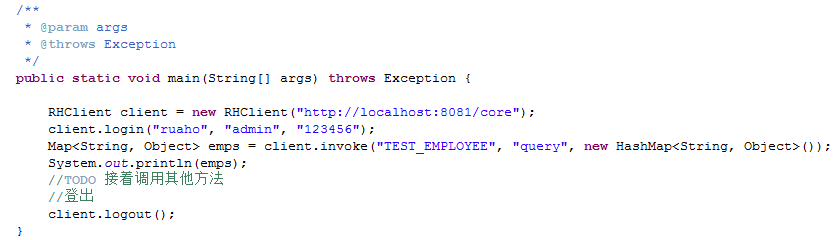
1. WebService

WebService的发布及调用，有以下5种方式：

* 1. 通过rhclient调用平台服务

1、不需要做任何配置

2、调用方法前先认证，调用后登出



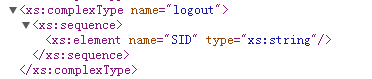
* 1. 生成wsdl调用平台服务

注意：

生成的wsdl地址为：<http://server:port/path/服务名.wsdl.ws>

xsd地址为：<http://server:port/path/服务名.xsd.ws>。

web service方法中（除login方法外），第一个参数（不需要配置，自动生成）总是SID，指的是登录用户的sessionId，从xsd文档中能看到，如：



1、配置SY\_ORG\_LOGIN的login方法，

因为调用前必须登录认证，因此我们先配置登录服务。



WebService结果定义了方法的返回值类型，此方法返回登录用户的sessionId，为string类型。



上面的配置定义了login方法的入参。

以下为logout方法：



生成的wsdl见本文最后。

1. 需要对服务进行配置，在此以TEST\_EMPLOYEE服务的delete方法为例进行说明。  
2. 根据wsdl文件生成相应的类文件。

使用这些类文件，就可以像调用本地的类一样调用WebService提供的方法了。

此处生成SY\_ORG\_LOGIN和上文提到的TEST\_EMPLOYEE的：

在命令行中：

wsimport -s E:\work\workspace\eclipse\rhclient\ws\login http://localhos

t:8081/core/SY\_ORG\_LOGIN.wsdl.ws

wsimport -s E:\work\workspace\eclipse\rhclient\ws\employee http://localhos

t:8081/core/TEST\_EMPLOYEE.wsdl.ws

运行完后，在E:\work\workspace\eclipse\rhclient\ws的login和employee中会生成一堆类文件供用户本地调用。

4、本地进行调用，代码示例：



* 1. 通过wsdl调用第三方WebService

请参考上一节的3、4 。

* 1. 通过WSClient调用平台服务

1. 不需要对方法进行特殊配置
2. 代码示例：



* 1. 通过WSClient调用第三方WebService

与“通过WSClient调用平台服务”相同。如：



一些配置项的说明



描述返回结果的数据类型：

1. 简单类型

xs:string xs:int xs:boolean

1. 复杂类型

目前配置上还有问题，待解决。

Wsdl文档示例

1. <http://localhost:8081/core/SY_ORG_LOGIN.wsdl.ws>

<?xml version="1.0" encoding="UTF-8"?>

<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws.ruaho.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://ws.ruaho.com/" name="SY\_ORG\_LOGIN">

<types>

<xsd:schema>

<xsd:import namespace="http://ws.ruaho.com/" schemaLocation="http://localhost:8081/core/SY\_ORG\_LOGIN.xsd.ws"/>

</xsd:schema>

</types>

<message name="login">

<part name="parameters" element="tns:login"/>

</message>

<message name="loginResponse">

<part name="parameters" element="tns:loginResponse"/>

</message>

<message name="logout">

<part name="parameters" element="tns:logout"/>

</message>

<message name="logoutResponse">

<part name="parameters" element="tns:logoutResponse"/>

</message>

<message name="Exception">

<part name="fault" element="tns:Exception"/>

</message>

<portType name="SY\_ORG\_LOGIN">

<operation name="login">

<input message="tns:login"/>

<output message="tns:loginResponse"/>

<fault message="tns:Exception" name="Exception"/>

</operation>

<operation name="logout">

<input message="tns:logout"/>

<output message="tns:logoutResponse"/>

<fault message="tns:Exception" name="Exception"/>

</operation>

</portType>

<binding name="SY\_ORG\_LOGINPortBinding" type="tns:SY\_ORG\_LOGIN">

<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>

<operation name="login">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

<fault name="Exception">

<soap:fault name="Exception" use="literal"/>

</fault>

</operation>

<operation name="logout">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

<fault name="Exception">

<soap:fault name="Exception" use="literal"/>

</fault>

</operation>

</binding>

<service name="SY\_ORG\_LOGINService">

<port name="SY\_ORG\_LOGINPort" binding="tns:SY\_ORG\_LOGINPortBinding">

<soap:address location="http://localhost:8081/core/SY\_ORG\_LOGIN.ws"/>

</port>

</service>

</definitions>

1. <http://localhost:8081/core/SY_ORG_LOGIN.xsd.ws>

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:tns="http://ws.ruaho.com/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://ws.ruaho.com/">

<xs:element name="login" type="tns:login"/>

<xs:element name="loginResponse" type="tns:loginResponse"/>

<xs:element name="logout" type="tns:logout"/>

<xs:element name="logoutResponse" type="tns:logoutResponse"/>

<xs:element name="Exception" type="tns:Exception"/>

<xs:complexType name="login">

<xs:sequence>

<xs:element name="cmpyCode" type="xs:string"/>

<xs:element name="loginName" type="xs:string"/>

<xs:element name="password" type="xs:string"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="loginResponse">

<xs:sequence>

<xs:element name="return" type="xs:string"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="logout">

<xs:sequence>

<xs:element name="SID" type="xs:string"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="logoutResponse">

<xs:sequence/>

</xs:complexType>

<xs:complexType name="Exception">

<xs:sequence>

<xs:element name="message" type="xs:string" minOccurs="0"/>

</xs:sequence>

</xs:complexType>

</xs:schema>

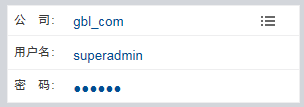
1. 常用配置
   1. 系统配置项

**使用的服务**

系统配置项使用的服务为SY\_COMM\_CONFIG。

1. 对于所有公司都使用的配置项。可维护为公共配置项。

用全局公司管理员登录，在“系统配置（公共）”中维护数据。

1. 各个公司专有的配置项，从本公司登录后，在“系统配置”中维护。公司的配置会覆盖公共配置。



**获取方式**

1. 配置项设为“前端使用”，可在前台进行获取。获取时，需要增加前缀C\_，如：System.getVar(“C\_SY\_PAGE\_TITLE”);
2. 后台获取方式：Context.getSyConf(key, 默认值))

**配置项**

按照key的英文字母顺序排列。配置项中的value，开发人员需要根据项目需求自己进行配置，如邮箱的配置，文件、搜索及报表服务器的配置等。

很多配置项在需要相应功能的时候，才用得上。当有相应异常提示信息时，在此进行查找并加上配置即可。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 配置key | 配置value | 说明 |
| 意见显示排序规则 | CM\_MIND\_SORT\_TYPE | TYPE | TIME 按照时间倒排序 TYPE 按照意见类型排序，意见类型下面再按时间倒排 |
| 发送邮件提醒的邮件服务器地址 | MAIL\_MESSAGE\_HOST | smtp.exmail.qq.com | 发送邮件提醒的邮件服务器地址 |
| 发送邮件提醒的邮箱是否要求用户验证 | MAIL\_MESSAGE\_IF\_AUTH | true | 发送邮件提醒的邮箱是否要求用户验证 |
| 发送邮件系统的邮箱密码 | MAIL\_MESSAGE\_PASSWORD | 1234 | 发送邮件系统的邮箱密码 |
| 发送邮件提醒的邮箱地址 | MAIL\_MESSAGE\_SENDER | 测试<test1@cplh.com.cn> | 发送邮件提醒的邮箱地址 |
| 发送邮件提醒的邮箱用户名 | MAIL\_MESSAGE\_USER\_NAME | test1@ cplh.com.cn | 发送邮件提醒的邮箱用户名 |
| 提醒消息发送邮件实现类 | MSG\_SENDER\_IMPL\_PREFIX\_EMAIL | com.rh.core.comm.msg.MailMsgSender | 提醒消息发送邮件实现类 |
| 提醒消息待办发送实现类 | MSG\_SENDER\_IMPL\_PREFIX\_TODO | com.rh.core.comm.msg.TodoMsgSender | 提醒消息中发送类型选择待办发送实现类 |
| 总公司层级 | ORG\_ZGS\_LEVEL | 1 |  |
| 日志异步处理调度间隔 | SY\_ASYNC\_LOG\_HANDLER\_INTERVAL | 120000 | 日志异步处理调度间隔,单位为秒。默认为60000ms(1分钟) |
| 日志异步处理队列最大值 | SY\_ASYNC\_LOG\_HANDLER\_MAX\_MSG\_COUNT | 300 | 日志异步处理队列上限，当队列数据量达到该上限时则会进行处理。默认200 |
| 头部banner的配置 | SY\_BANNER\_CONFIG | {'banner':'/sy/comm/page/img/head\_logo\_rbcx.gif','bannerBack':'/sy/comm/page/img/head\_bg\_rbcx.png','bannerTabBack':'/sy/comm/page/img/rh-head-menu.png'} | banner:含login的图片 bannerBack:整个banner的背景图片 bannerTabBack:banner下方的tab条的背景 |
| Banner图标主页配置 | SY\_BANNER\_ICON | {"portal":{"url":"","name":"门户","menuId":""}, "msg":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=SY\_MSG\_CENTER&model=view","name":"消息中心","menuId":"SY\_MSG\_CENTER\_\_@CMPY\_CODE@"}, "new":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=2uyWMoDQB0LX2cL6lOQxEo&model=view","name":"资讯中心","menuId":"SY\_NEWS\_CENTER\_\_@CMPY\_CODE@"}, "time":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=SY\_COMM\_MEMO\_PAD&model=view","name":"时间中心","menuId":"SY\_TIME\_CENTER\_\_@CMPY\_CODE@"}, "wf":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=SY\_WORKFLOW\_CENTER&model=view","name":"流程中心","menuId":"SY\_WORKFLOW\_CENTER\_\_@CMPY\_CODE@"}, "job":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=SY\_TASK\_CENTER&model=view","name":"任务中心","menuId":"SY\_TASK\_CENTER\_\_@CMPY\_CODE@"}, "social":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=3vcRyLKjt6YVkMO2Y1wHsy&model=view","name":"社区中心","menuId":"SY\_CMMU\_CENTER\_\_@CMPY\_CODE@"}, "doc":{"url":"SY\_COMM\_TEMPL.show.do?pkCode=SY\_DOC\_CENTER&model=view","name":"文档中心","menuId":"SY\_DOC\_CENTER\_\_@CMPY\_CODE@"}} | portal:值为系统处理，不需要url和menuId 其它可以选增相应图标显示 |
| 是否显示banner区的全文搜索框 | SY\_BANNER\_SEARCH | true |  |
| 控制信息 | SY\_COMM\_ADDRESS\_CTR\_INFO | USER\_MOBILE: | 键值对格式串，格式值为 控制字段：显示值，多个键值对间用“,”分隔。  如 SER\_MOBILE:保密，表示，USER\_MOBILE 使用“保密”显示， 此配置项同“SY\_COMM\_ADDRESS\_SECURED\_CLASS”配置项配合使用。 例子：USER\_MOBILE:保密, USER\_NAME:\* |
| 通讯录默许开放对象 | SY\_COMM\_ADDRESS\_OPEN\_OBJ | {'DEPTS':'cplh\_leader','ROLES':'RGSLD','USERS':'2nUDp31wt8QHRUFsg8LkE\_'} | 可以对通讯录信息进行开放性设置，即通讯录对哪些特殊的部门、角色或个人进行开放不进行控制；配置值：为JSON格式数据，参数为DEPT、ROLS、USERS三项的编码值，暂不支持其它； 配置值例子如：{'DEPTS':'1,2','ROLES':'22,33','USERS':'zhangjinwei'} |
| 通讯录权限控制类 | SY\_COMM\_ADDRESS\_SECURED\_CLASS | com.rh.core.comm.txl.DefTxlSecurer | 通讯录权限控制默认实现类，配置值可自己实现，extends AbstractTxlSecurer类，实现getSecuredUserBean(Bean userBean)方法，即可使用自己的规则进行控制，若配置值为空，则默认使用“DefTxlSecurer”类 可以同 SY\_COMM\_ADDRESS\_CTR\_INFO配合使用 |
| 批量保存日志时间间隔 | SY\_COMM\_BATCH\_SAVE\_LOG\_INTERVAL | 300 | 单位为:秒 |
| 批量保存日志最大数据量 | SY\_COMM\_BATCH\_SAVE\_LOG\_MAX\_SIZE | 500 | 当日志数据缓存到${SY\_COMM\_BATCH\_SAVE\_LOG\_MAX\_SIZE}条时，进行批量保存 |
| 文件根路径 | SY\_COMM\_FILE\_ROOT\_PATH | smb://samba:123456@192.168.100.42/public/ | 每个项目都需要设置 |
| 是否启用菜单缓存 | SY\_COMM\_MENU\_CACHE | true | 启用菜单缓存后，只有当更新用户相关权限信息后才重新生成菜单文件，缺省为true。 |
| 提醒数据的查询时间范围 | SY\_COMM\_REMIND\_TIME\_RANGE | 5 | 按时间提醒数据的查询时间范围，即与当前时间相差几分钟的数据能被查询出来，准备发送 |
| 提醒消息实时发送数据的过期时间 | SY\_COMM\_REMIND\_TIME\_VALID | 30 | 提醒消息实时发送数据的过期时间，用于去掉很久之前提交的已失去时效的提醒数据 |
| 上班时间设定 | SY\_COMM\_WORK\_TIME | 08:30:00,11:30:00,13:00:00,18:00:00 | 分别为上午上班时间，下班时间  下午上班时间，下班时间 |
| 待办提醒更新时间 | SY\_DESK\_MSG\_INTERVAL | 300000 | 待办提醒更新时间，单位毫秒，默认值为5分钟即300000ms |
| 是否显示工作台的搜索框 | SY\_DESK\_SEARCHICON | false |  |
| 系统开发用户 | SY\_DEV\_USERS | admin,superadmin | 系统开发用户，可双击卡片字段的配置。预留以后开发用户的一些功能时的开关 |
| 最大导出数量 | SY\_EXP\_NUM | 5000 | 导出功能最大一次导出数据量。 |
| 大字体显示 | SY\_FONT\_BIG | true | 为true，则系统将启用大字体风格 |
| 首页菜单显示位置 | SY\_HOMEMENU\_LOC | 2 | 1:顶部菜单，2：左侧菜单 |
| home首页的配置 | SY\_HOME\_CONFIG | {'tTitle':'系统首页','url':'SY\_COMM\_TEMPL.show.do?pkCode=1v84qOett8QWe8sJvkW9UW','closeFlag':false,'menuFlag':3} | 有此参数则直接进入相应系统的home页 |
| 系统URL根路径 | SY\_HTTP\_URL | http://oa.cplh.com | 非80端口的要求包含端口号，不需要包含虚路径，此配置用于后台没有用户交互的程序，有用户交互的程序自动从request中获取。 |
| IM集成即时消息类名 | SY\_HUB\_IM | com.rh.core.hub.im.ImRtx | 缺省为空，不集成即时通讯工具，如果集成RTX，配置值为：com.rh.core.hub.im.ImRtx |
| IM服务器端口号 | SY\_HUB\_IM\_PORT | 8012 | 调用RTX服务器的端口号 |
| IM服务器IP地址 | SY\_HUB\_IM\_SERVIP | 192.168.100.41 | 集成RTX服务器的IP地址，如果为本机，则为127.0.0.1 |
| 是否启用智能索引服务 | SY\_HUB\_SEARCH | true | 是否启用搜索，true启用，false不启用（缺省） |
| 系统登录认证服务类 | SY\_ORG\_LOGIN\_MODULE\_CLASS |  | 缺省不填写为：com.rh.core.auth.login.PasswordLoginModule  ldap为：com.rh.core.auth.login.LdapLoginModule |
| 系统登录认证服务名 | SY\_ORG\_LOGIN\_MODULE\_NAME |  | 名称为空：使用缺省的用户名密码认证服务；  ldap：使用ldap认证  custom：使用自定义认证服务 |
| 是否自动将用户登录名转为小写 | SY\_ORG\_LOGIN\_NAME\_LOWERCASE | true | 是否自动将用户登录名转为小写，缺省为true，以小写的英文名作为登录名。 |
| 是否显示手机登录链接 | SY\_MB\_LINK | true |  |
| 公共角色编码 | SY\_ORG\_ROLE\_PUBLIC | RPUB | 系统提供公共角色支持，每个用户缺省都具有此角色，无需单独在角色中添加用户，在授权管理中可以通过对公共角色设置一些所有人都有的功能权限。 |
| 超级公司编码 | SY\_ORG\_SUPER\_CMPY | 1 | 超级公司是系统可以统筹管理其他公司的公司，缺省超级公司编码为：1 |
| 登录后浏览器标题 | SY\_PAGE\_TITLE |  | page页浏览器标题 |
| 项目级扩展js文件 | SY\_PJ\_JS |  | 工程级扩展js文件,项目公用js定义文件,如：/sy/test.js |
| 搜索服务文件服务器 | SY\_PLUG\_SEARCH\_FILE\_SERVER | http://172.16.0.30:8888/searchserver/file/ | 搜索服务-文件服务器地址 |
| 搜索服务分词服务器地址 | SY\_PLUG\_SEARCH\_SEG\_SERVER | http://172.16.0.30:8888/searchserver/seg | 搜索服务分词服务器地址 |
| 搜索服务器地址 | SY\_PLUG\_SEARCH\_SERVER | http://172.16.0.30:8888/searchserver/rhsearch | 搜索服务器地址 |
| 搜索服务器拼写检查服务器地址 | SY\_PLUG\_SEARCH\_SPELLCHECK\_SERVER | http://172.16.0.30:8888/searchserver/spellcheck/ | 搜索服务器拼写检查服务器地址 |
| 搜索服务搜索建议服务器地址 | SY\_PLUG\_SEARCH\_SUGGEST\_SERVER | http://172.16.0.30:8888/searchserver/suggestion | 搜索服务搜索建议服务器地址 |
| web爬虫服务器地址 | SY\_PLUG\_SEARCH\_WEBCRAWLER\_SERVER | http://172.16.0.110:1976/ | web爬虫服务器地址 |
| 即时通讯服务器地址 | SY\_PLUG\_WEBIM\_SERVER |  | http://172.16.0.30:7070/http-bind/ |
| 系统管理角色 | SY\_ROLE\_ADMIN | RADMIN | 此角色下的用户缺省可以看到所有数据，不会判断数据权限。 |
| 无Session可以展示的字典 | SY\_SERV\_DICT\_NOSESSION | SY\_ORG\_CMPY | 多个字典逗号分隔 |
| SQL批量处理单次数量 | SY\_SQL\_BATCH\_SIZE | 10000 | 缺省值为10000，出于性能考虑，执行批量处理的SQL超过此设定后会自动分段执行。 |
| 单点登录服务器地址 | SY\_SSO\_URL | http://172.16.0.127:8080/cas | 单点登录服务器地址 |
| 系统风格定义 | SY\_STYLE\_DEF | {'SS\_STYLE\_MENU':'pageBody-bluelue','SS\_STYLE\_BACK':'bodyBack-blue','SS\_STYLE\_BLOCK':'conHeaderTitle-gray'} | 系统默认风格定义 |
| 节日风格覆盖 | SY\_STYLE\_FEST | {'SS\_STYLE\_MENU':'pageBody-festival','SS\_STYLE\_BACK':'bodyBack-festival','SS\_STYLE\_BLOCK':'conHeaderTitle-festival'} | 此配置值优先级 > 个人风格(优先级) > 系统默认风格(优先级) |
| tab颜色 | SY\_TAB\_COLOR | {'1':{'li':'grayLi','a':'grayA'}} | 一共5种，可选取其中几种作为指定的颜色：{'1':{'li':'qingLi','a':'qingA'},'2':{'li':'yellowLi','a':'yellowA'},  '3':{'li':'grayLi','a':'grayA'},'4':{'li':'blueLi','a':'blueA'},'5':{'li':'redLi','a':'redA'},'6':{'li':'lightLi','a':'lightA'},}; |
| 系统标签最大个数 | SY\_TAB\_MAX | 10 | 最外层系统的标签打开个数限制 |
| 是否启用顶部下拉提醒面板 | SY\_TOP\_ALERT | true |  |
| 用户密码加密方式 | SY\_USER\_PASSWORD\_ENCRYPT | MD5 | 密码加密方式，支持DES（缺省）、MD5、SHA |
| 用户初始密码 | SY\_USER\_PASSWORD\_INIT | 123456 | 创建用户的缺省密码 |
| 即时通讯启用标志 | SY\_WBIM\_FLAG | true | 即时通讯启用标志 true为启用，默认不启用 |
| 催办单代字的默认值 | SY\_WFE\_REMIND\_CODE | 催办 |  |
| 自动发催办的用户 | SY\_WFE\_REMIND\_USER | admin | 对超时的节点发催办的时候，发送人的loginName |
| 返回上一节点的字段设定 | SY\_WF\_BACK\_NODE\_NAME | NODE\_NAME 节点名称 NODE\_MEMO 节点描述 如果不填，按钮就是 返回XX |  |
| 一个人是否直接送交 | SY\_WF\_DIRECT\_SEND | true | 添加全局系统配置，实现当角色下只有一人时，是否直送可根据需要配置。 true 直接送交 false 出机构树，让用户选择这个人，确定之后再发送 |
| XDOC的URL路径 | SY\_XDOC\_URL | http://localhost:8080/xdoc | XDOC服务的URL地址，用到XDOC的必须配置 |

* 1. 首页配置

**平台目前支持两种默认打开首页的方式，根据情况选择其中一种配置即可。**具体配置方法和效果如下：

* + 1. 默认打开首页，保留工作台
       1. 效果图



* + - 1. 配置步骤

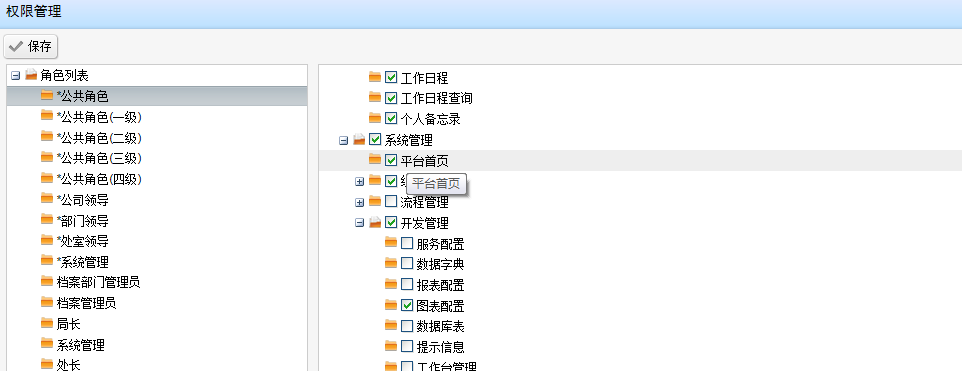
1. 配置平台首页菜单

菜单配置如下图所示，设定“显示类型”为“桌面图标”或“桌面或手机图标”（这两种显示类型可以在工作台中显示），桌面设置的属性为工作台显示的相关设定。



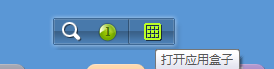
1. 菜单授权

在权限管理下，将菜单授权给可以相应的角色（为保证所有人都打开首页，应授权给公共角色），如下图所示。



1. 在工作台设置“平台首页”为默认应用。

打开应用盒子：



在桌面应用中点击平台首页，将平台首页显示在工作台桌面上。



刷新页面或重新登录，然后再次打开应用盒子，在默认应用中，点击选中“平台首页”。

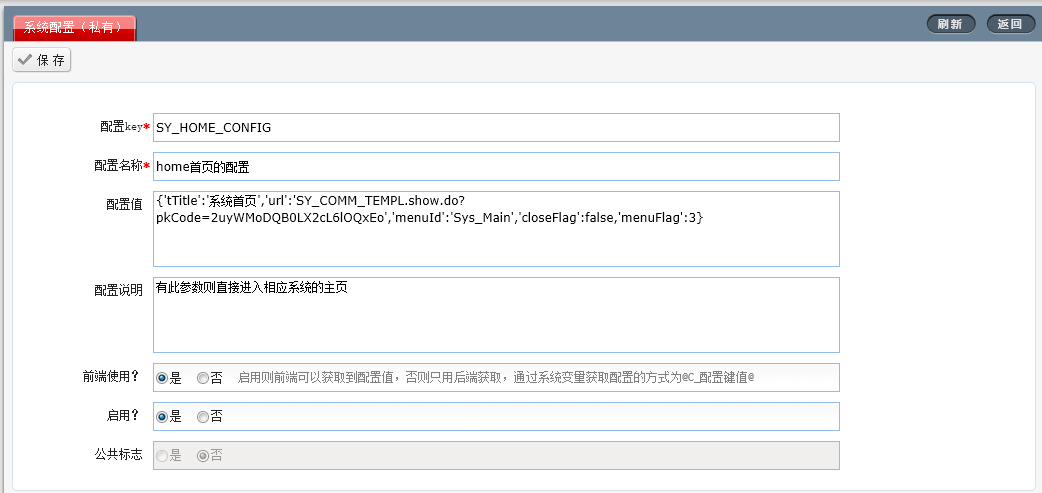


1. 配置完成，重新登录即可默认打开首页。
   * 1. 默认打开首页，不保留工作台
        1. 效果图



* + - 1. 配置步骤

1. 用admin登录，在系统设置-系统配置中，添加SY\_HOME\_CONFIG配置项，具体配置如下图所示：



配置值的JSON含义如下：

{'tTitle':'name','url':'SY\_COMM\_TEMPL.show.do?pkCode=2uyWMoDQB0LX2cL6lOQxEo','menuId':'Sys\_Main','closeFlag':false,'menuFlag':3}

tTitle：首页的tab页名称。

url：首页的链接。

menuId：首页所对应的菜单的Id；如首页SY\_COMM\_TEMPL.show.do?pkCode=2uyWMoDQB0LX2cL6lOQxEo在配置菜单时的菜单Id为Sys\_Main，则menuId就要配置为Sys\_Main。

closeFlag：设定首页tab是否可以关闭。

menuFlag：1、显示当前层级菜单 2、显示全部菜单 3、隐藏左侧菜单 4、没有对应菜单，但是显示菜单栏

1. 配置登录页的隐藏参数

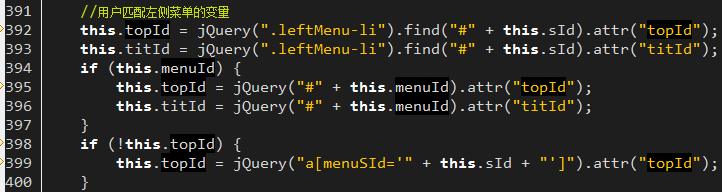
打开登录页面文件（默认为index.jsp），找到隐藏域

<input type="hidden" id="rhDevIndex" value="true"/>

设定value=false或删除此隐藏域

1. 配置完成，重新登录即可默认打开首页

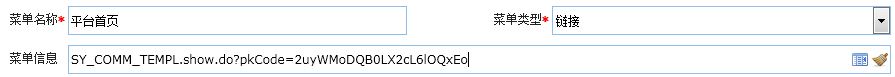
注：匹配左侧菜单的细节请参考rh.ui.openTab.js



* + 1. 首页配置解析

SY\_HOME\_CONFIG中配置的首页对应着一个菜单。

**首页菜单：**

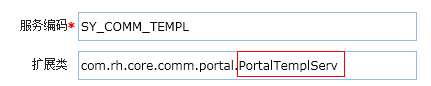


可以看出使用的是“门户模板”服务。

“菜单信息”中的pkCode参数对应着配置好的一个门户模板数据。

对“门户模板”服务的说明：SY\_COMM\_TEMPL的配置为：





扩展类的show方法，跳转到这个页面：/sy/comm/home/portalTempl.jsp。

**新增门户模板：**

保存数据后才可进行可视化编辑。



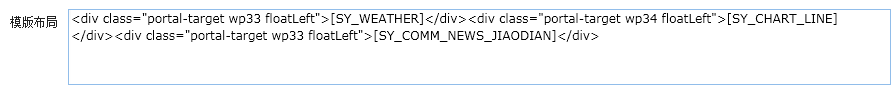
要先选择布局、再选择组件（组件是通过“**模板组件**”服务配置的）







可视化设计完成后：



直接写模板布局效果一样。

**代码分析：**

portalTempl.jsp



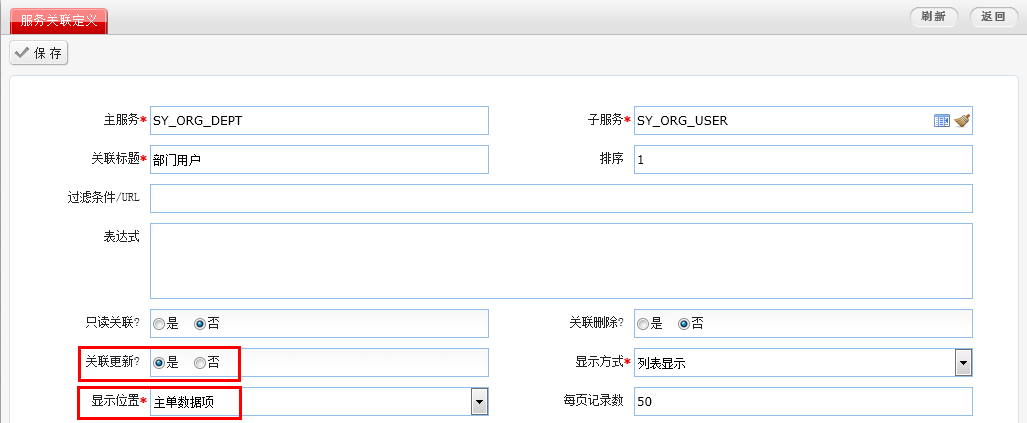


incl-templ.js作为首页主要的js，主要处理首页模板的业务逻辑，包括各个模板组件的处理。如“待办”列表的处理，“已办”列表的处理。

* 1. 一体化组件配置
     1. 配置工作

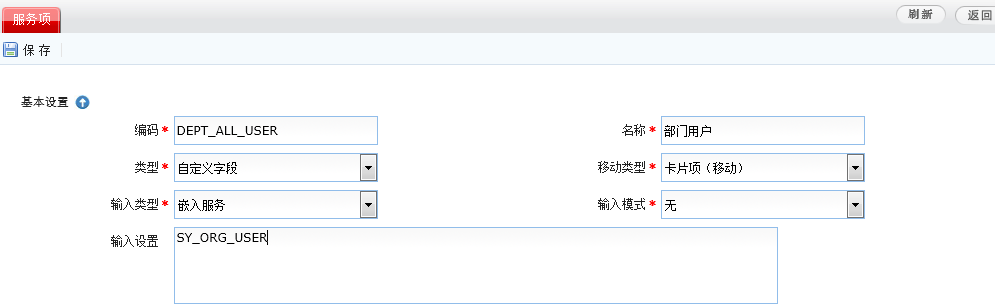
1. 配置关联服务





配置关联服务相关信息，其中“显示位置”设置为“主单数据项”,配置关联更新为“是”。

1. 配置服务字段



定义一个类型为“自定义字段”的服务字段，输入类型为“嵌入服务”，输入模式“无”，然后在输入设置中填写关联的子服务ID，如图中示例的“SY\_ORG\_USER”。

* 1. 消息提醒配置
     1. 消息提醒

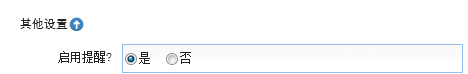
消息提醒是指在工作台上显示该服务的消息总数，其显示方式如下所示：



其中“667”即为消息总数。

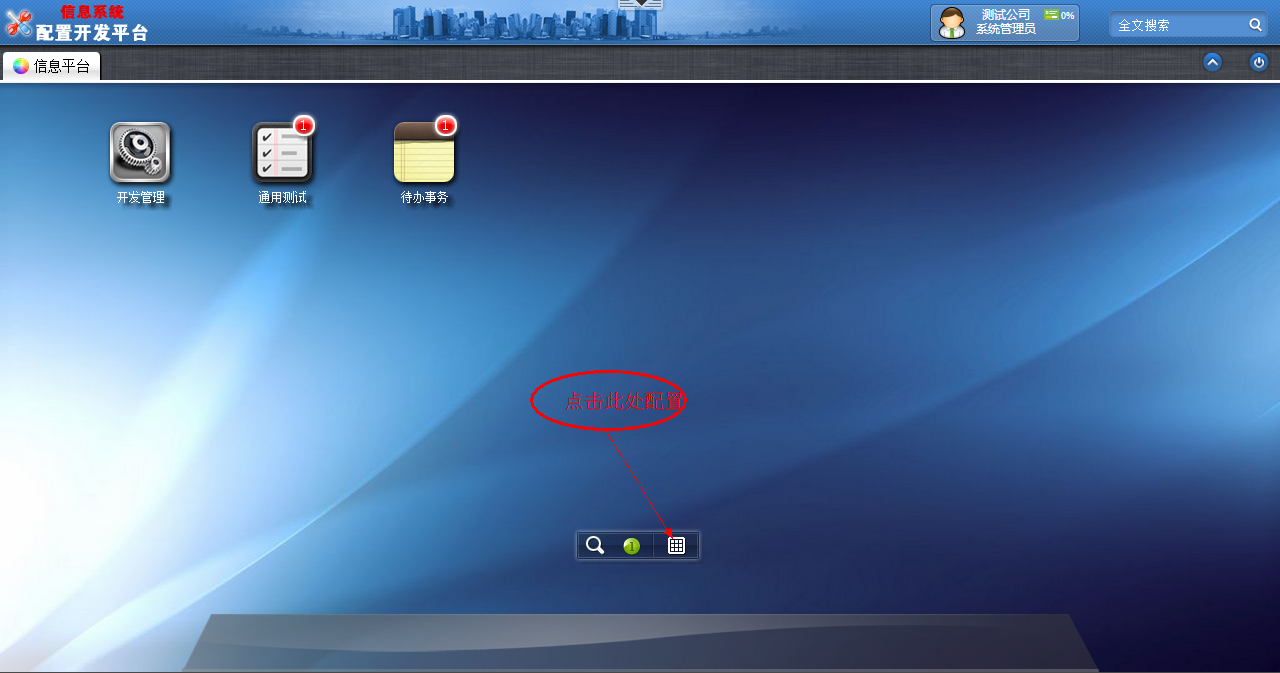
* + 1. 菜单配置

在对应菜单定义中选中“启用提醒”：



图标上显示的数字是菜单对应服务count方法的返回值。

* + 1. 桌面配置





将“通知设置”中的“是否接受来自应用的消息提醒”勾中。

* 1. 顶部下拉面板配置

在“系统设置”——“系统配置”中添加如下配置项：



* 1. 自定义登录配置
     1. 平台默认的登录逻辑

平台提供的默认的登录认证为：基于用户名、密码的登录认证。

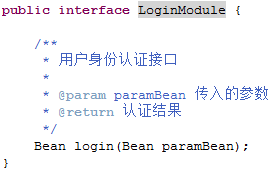
* + 1. 自定义登录逻辑

如果要自定义自己的登录逻辑，首先在“系统配置”服务中增加两条配置：





自定义的登录认证类需实现LoginModule接口。

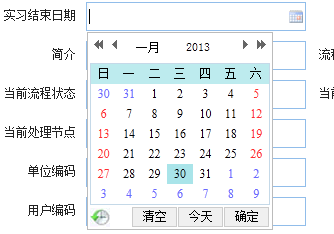


平台还提供了该接口的一个实现类，通常我们直接继承该类即可。

* 1. 日期框配置
     1. 日期框的基本配置



显示效果如下

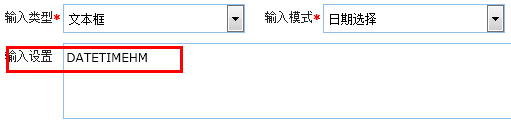


值的显示为



* + 1. 对日期框的格式进行控制

如显示“2013-01-08 21:12:48”等格式的数据，需要在“输入设置”中进行配置，如

、

预定义的类型有：

|  |  |
| --- | --- |
| 配置 | 显示格式 |
| DATETIME | yyyy-MM-dd HH:mm:ss |
| DATETIMEH | yyyy-MM-dd HH |
| DATETIMEHM | yyyy-MM-dd HH:mm |
| YEAR | yyyy |
| MONTH | yyyy-MM |
| TIME | H:mm |
| 其他 | 见下面，自定义日期格式配置 |

* + 1. 自定义日期格式配置

预定义的日期格式不能满足需求时，需要配置自定义日期格式。

|  |  |
| --- | --- |
| 配置 | 显示格式 |
| CUSTOM,,,yyyy年MM月 | yyyy年MM月 |
| CUSTOM,,,yyyy年MM月dd日 | yyyy年MM月dd日 |

* + 1. 例子

1、前面的日期不能大于后面的日期。如图，要求，开始日期 < 结束日期



在card扩展js中，加入事件：

\_viewer.getItem("PRACTICE\_START").obj.unbind("click").bind("click", **function**(){

WdatePicker({maxDate:"#F{$dp.$D('" + \_viewer.servId + "-PRACTICE\_END')}"});

});

\_viewer.getItem("PRACTICE\_END").obj.unbind("click").bind("click", **function**(){

WdatePicker({minDate:"#F{$dp.$D('" + \_viewer.servId + "-PRACTICE\_START')}"});

});

2、最大日期为今天

\_viewer.getItem("PRACTICE\_START").obj.unbind("click").bind("click", function(){

WdatePicker({maxDate:"%y-%M-%d"});

});

3、其他复杂场景

代码的具体含义及更复杂使用，请自行参考<http://www.my97.net/dp/demo/index.htm>。

* 1. 只读配置
     1. 列表只读而卡片非只读配置

该服务配置时必须在“方法&按钮”中做以下配置：save和modify



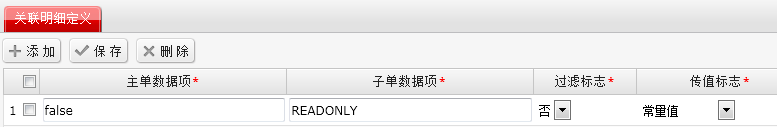


* + 1. 主服务只读而子服务非只读配置

应用场景：主服务的数据不需要维护，但在主服务的card页面，看到的子表数据要能维护。

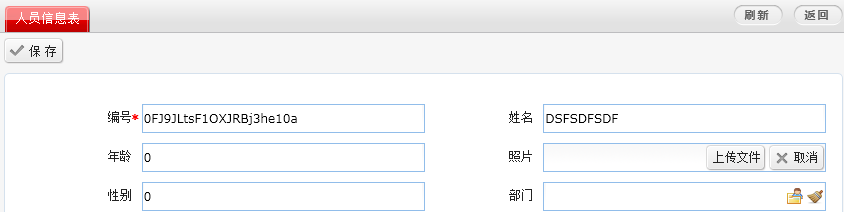
需要做以下配置：

在主服务的“关联服务定义”中，维护主子服务的关联关系，在“关联明细定义”中，增加一项配置：



* + 1. 主服务非只读而子服务只读配置

应用场景：主服务可正常编辑，但在主服务的card页面，子服务（主单下列表或主单标签）数据只读。如图：



需要在主服务的扩展js中写上：

if(条件){

\_viewer.tabReadOnly(关联子服务的servId);

}

* 1. 系统提示信息使用
     1. 通用使用方式

1. Tip.show(msg, p arentFlag, scopeObj);//提示信息条，定时会隐藏, msg显示顶部提示信息,parentFlag 父层对象的标识
2. Tip.showAttention(msg, parentFlag, scopeObj);//提示信息条，定时会隐藏,msg 显示顶部提示信息,parentFlag 父层对象的标识
3. Tip.showError(msg, parentFlag, scopeObj);//提示信息条，定时会隐藏,msg显示顶部提示信息,parentFlag父层对象的标识
4. Tip.showLoad(msg, parentFlag, appendObj, timeout, scopeObj);//加载信息提示,msg显示顶部提示信息, parentFlag父层对象的标识
5. Tip.clear();//外层清除提示信息
6. Tip.clearLoad();//删除加载提示信息条

详细使用情况见：tool.js

* + 1. 在扩展list.js中使用

\_viewer.listBarTipError("提示信息");

* + 1. 在扩展card.js中使用

\_viewer.cardBarTipError("提示信息");

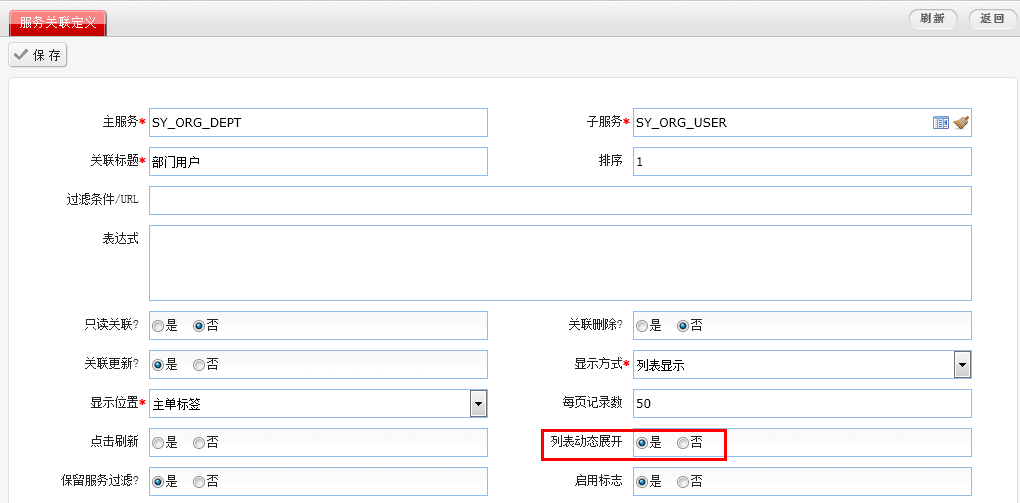
* 1. 主子表组件配置

1. 显示效果如下：



1. 配置

在服务关联定义配置中，设定“列表动态展开”为“是”，如下图所示：

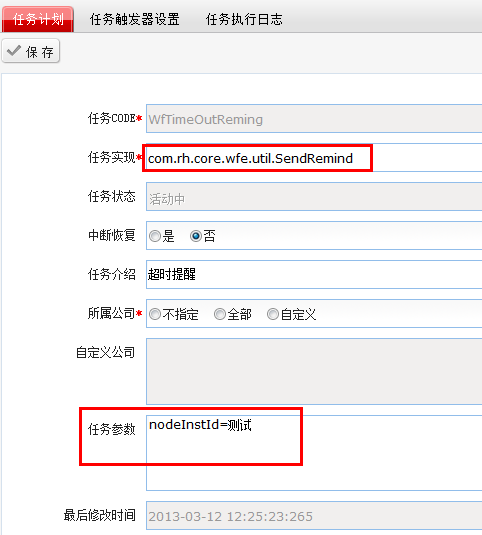


* 1. 任务调度的使用配置
     1. 任务配置



任务调度对应的服务为SY\_COMM\_SCHED（任务计划），也可以直接在“服务配置”中进行查询然后预览。

在列表上点击“添加”，



其中，

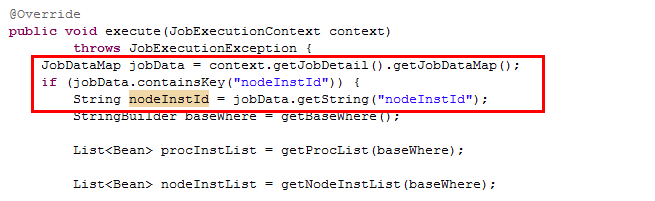
“任务实现”是开发人员的Job类，必须实现org.quartz.Job接口。

“任务参数”中配置在job实现类需用到的参数，多个参数之间逗号分隔，如

* + 1. 任务类的编写

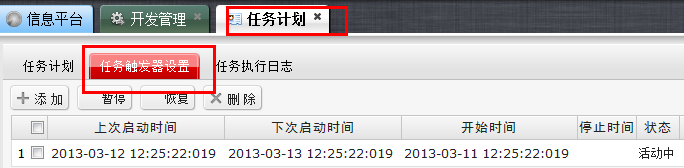
Job类要实现Job接口，





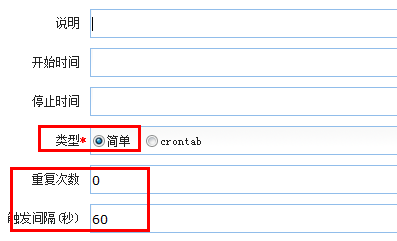
红框内的代码展示的是如何获取“任务参数”。

* + 1. 调度执行

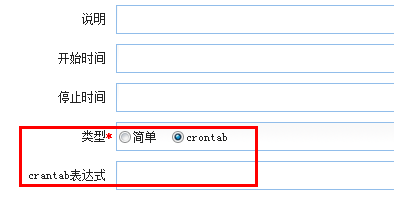


添加时，可看到有简单任务调度及复杂任务调度两种：

简单任务调度：



复杂任务调度：



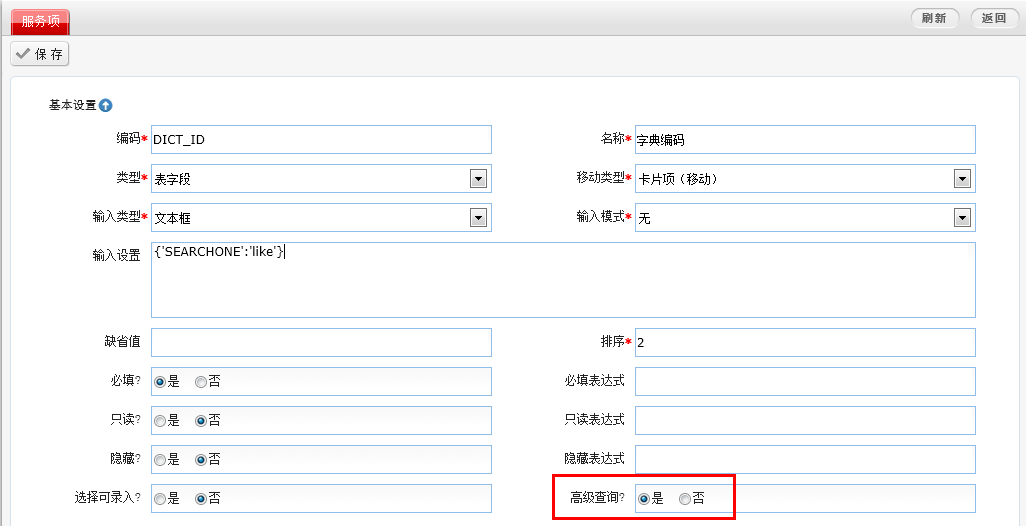
表达式的写法是[Quartz表达式](http://www.cnblogs.com/archie2010/archive/2011/03/15/1984820.html)，如：



具体写法请google。

* 1. 高级查询配置
     1. 设定高级查询字段

设定服务项中“高级查询”为“是”



* + 1. 查询字段高级配置

一般情况下，设定“高级查询”为“是”即可满足大部分需求，各类型字段的默认查询方式为：

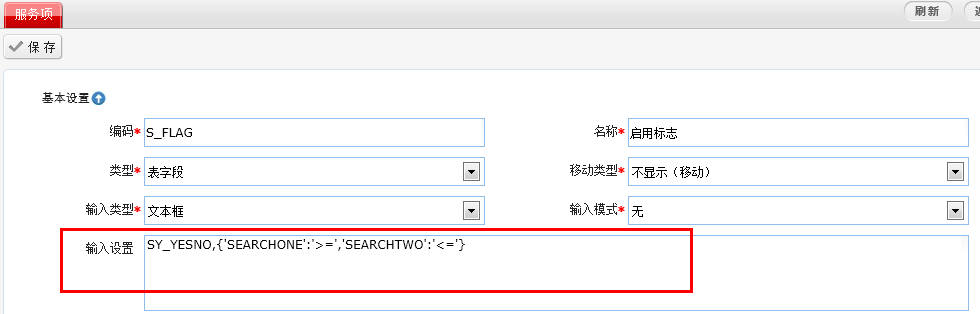
文本框：模糊匹配（like）

数字：等于

日期：区间（从。。到。。）

单选下拉框、单选、多选框及查询选择：根据选择条数为“等于”或“IN”

**另外系统支持自定义配置，设定“输入设置”**



格式如下：

{'SEARCHTYPE':'multi', 'SEARCHONE':'>=', 'SEARCHTWO':'<=' }

各配置项说明如下：

1、SEARCHTYPE：空或multi，设定为multi支持下拉框和单选框在高级查询中多选，转换为复选框。显示效果如下：



2、SEARCHONE和SEARCHTWO：配置字段匹配操作符，目前支持文本框、日期和数字，只配置SEARCHONE则为单个控件，如果两个都配置则为区间查询（区间查询目前只支持数字和日期）。

文本类型支持：'like'、'not like'和'='

日期和数字类型支持：'>='、'>'、'<='、'<'、'='

例1：字段类型为文本框

配置{'SEARCHONE':'not like'}，则显示效果如下：



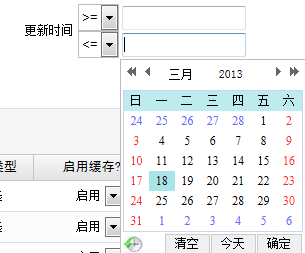
例2：字段类型为数字文本框

配置{'SEARCHONE':'>=', 'SEARCHTWO':'<='}，则显示效果如下：



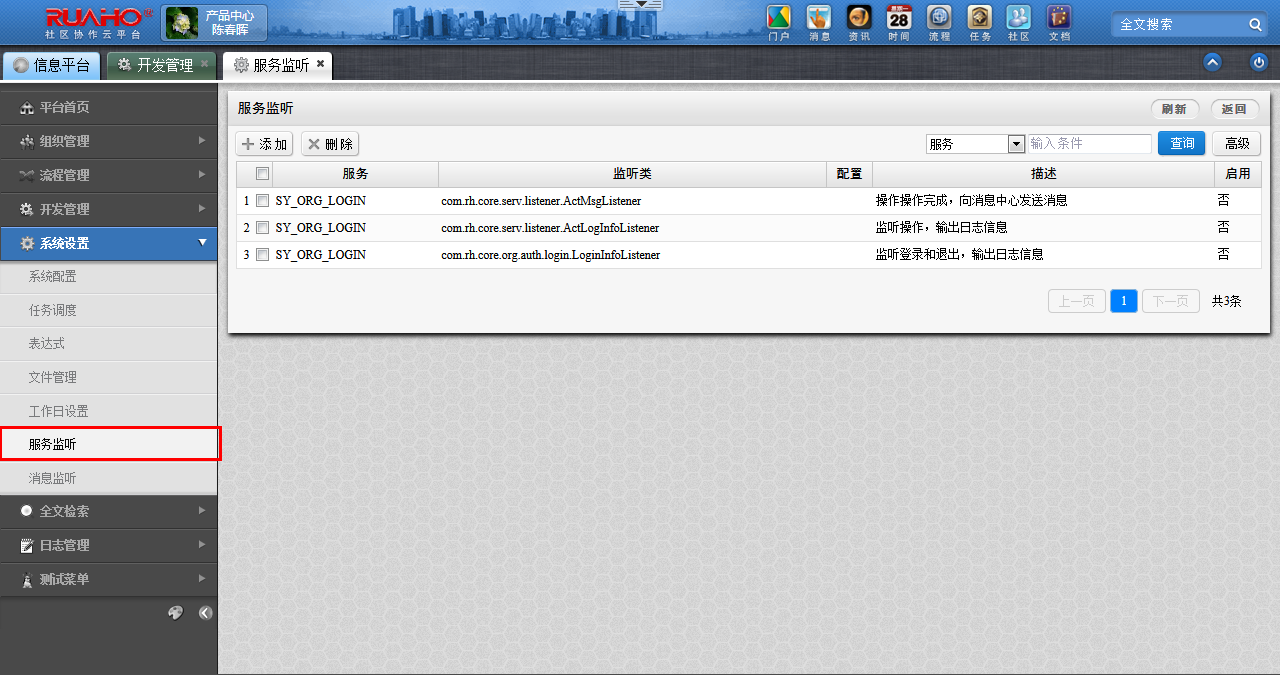
例3：字段类型为日期选择

配置{'SEARCHONE':'>=', 'SEARCHTWO':'<='}，则显示效果如下：



* 1. 工作流监听与RTX通知
     1. 工作流监听配置

平台对工作流监听的实现是基于平台提供的“服务监听”功能，打开平台的“系统设置”-“服务监听”页面：



添加一条新的服务监听配置，其配置信息如下：



其中服务“SY\_WFE\_PROC\_DEF”即是流程的服务，所有系统配置的流程均在此服务中运行，com.rh.core.wfe.listener.WorkFlowRtxNotifyListener是我们实现的服务监听类。关于服务监听类的实现方式和要求，请查看服务监听文档。WorkFlowRtxNotifyListener提供的接口如下所示：

* public void init(String serv, String conf)

WorkFlowRtxNotifyListener在平台启动的过程中将被实例化唯一一次，在实例化之后，该方法将被调用；

* public void beforeStart(ParamBean paramBean)

该方法在每次流程启动之前都会被调用，paramBean是流程启动时的参数bean，从该参数bean中可以获取到流程的定义、流程中的数据、流程发起人等信息；

* public void afterStart(ParamBean paramBean, OutBean result)

该方法在每次流程启动之后都会被调用，paramBean是流程启动时的参数bean，result是流程启动之后返回的结果bean；

* public void beforeToNext(ParamBean paramBean)

该方法在每次流程流转到下一个节点前调用，paramBean是流程流转时的参数bean，包含了流程的实例信息；

* public void afterToNext(ParamBean paramBean, OutBean result)

该方法在每次流转到下一个节点后调用，paramBean是流转时的参数bean，result是流程流转后的返回结果；

* public void beforeFinish(ParamBean paramBean)

该方法在流程结束之前被调用，paramBean是流程结束时的参数bean；

* public void afterFinish(ParamBean paramBean, OutBean result)

该方法在流程结束之后被调用，paramBean是流程结束的时候的参数bean，result是流程结束后的结果。

WorkFlowRtxNotifyListener的简单实现类如下所示：

package com.rh.core.wfe.listener;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import com.rh.core.base.Bean;

import com.rh.core.plug.im.ImRtx;

import com.rh.core.serv.OutBean;

import com.rh.core.serv.ParamBean;

import com.rh.core.serv.ServMgr;

import com.rh.core.serv.ServUtils;

/\*\*

\* 工作流RTX通知

\*

\* @author 陈春晖

\*

\*/

public class WorkFlowRtxNotifyListener {

private static Log log = LogFactory.getLog(WorkFlowRtxNotifyListener.class);

/\*\*

\* 流程定义

\*/

private static final String WORK\_FLOW\_DEFINITION = "\_WF\_PROC\_DEF\_BEAN\_";

/\*\*

\* 流程实例

\*/

private static final String WORK\_FLOW\_INSTANCE = "\_WF\_PROC\_INST\_";

/\*\*

\* 流程操作实例

\*/

private static final String WORK\_FLOW\_ACT\_INSTANCE = "\_WF\_ACT\_INST\_";

private static final ImRtx RTX = new ImRtx();

/\*\*

\* 初始化，该监听类实例化后自动调用

\*

\* @param serv 服务

\* @param conf 配置

\*/

public void init(String serv, String conf) {

if(RTX.init()){

log.debug("RTX初始化成功");

}else{

log.error("RTX初始化失败");

}

}

/\*\*

\* 流程启动前执行

\*

\* @param paramBean

\*/

public void beforeStart(ParamBean paramBean){

RTX.sendNotify("1002", "流程启动前", getStartWorkFlowUser(paramBean).getStr("USER\_NAME")

+ " 将要启动一个新流程： " + getWorkFlowTitleInStart(paramBean));

}

/\*\*

\* 流程启动后执行

\*

\* @param paramBean

\* @param result

\*/

public void afterStart(ParamBean paramBean, OutBean result){

RTX.sendNotify("1002", "流程启动成功", getStartWorkFlowUser(paramBean).getStr("USER\_NAME")

+ " 成功启动一个新流程： " + getWorkFlowTitleInStart(paramBean));

}

/\*\*

\* 流程流转到下一个节点前执行

\*

\* @param paramBean

\*/

public void beforeToNext(ParamBean paramBean){

RTX.sendNotify("1002", "流程即将流转到下一个节点", "流程实例编码：" + paramBean.getStr("PI\_ID") +

"\n节点实例编码：" + paramBean.getStr("NI\_ID"));

}

/\*\*

\* 流程流转到下一个节点后执行

\*

\* @param paramBean

\* @param result

\*/

public void afterToNext(ParamBean paramBean, OutBean result){

RTX.sendNotify("1002", "流程即将流转到下一个节点", "流程实例编码：" + paramBean.getStr("PI\_ID") +

"\n节点实例编码：" + paramBean.getStr("NI\_ID") +

"\n发送到部门：" + result.getStr("TO\_DEPT\_NAME") +

"\n发送到角色：" + result.getStr("TO\_ROLE\_NAME") +

"\n发送到用户：" + result.getStr("TO\_USER\_NAME") +

"\n节点名称：" + result.getStr("NODE\_NAME"));

}

/\*\*

\* 流程办结前执行

\*

\* @param paramBean

\*/

public void beforeFinish(ParamBean paramBean){

RTX.sendNotify("1002", "流程办结", "流程实例：" + paramBean.getStr("PI\_ID") +

"\n节点实例：" + paramBean.getStr("NI\_ID"));

}

/\*\*

\* 流程办结后执行

\*

\* @param paramBean

\* @param result

\*/

public void afterFinish(ParamBean paramBean, OutBean result){

if(result.getStr("rtnstr").equals("success")){

RTX.sendNotify("1002", "流程办结", "流程实例：" + paramBean.getStr("PI\_ID"));

}

}

/\*\*

\* 获取流程启动人

\*

\* @param userCode

\* @return

\*/

private OutBean getStartWorkFlowUser(ParamBean paramBean){

return ServMgr.act(

"SY\_ORG\_USER",

ServMgr.ACT\_BYID,

new ParamBean()

.setServId("SY\_ORG\_USER")

.setId(paramBean.getBean(WORK\_FLOW\_DEFINITION).getStr("S\_USER"))

);

}

/\*\*

\* 获取流程标题

\*

\* @param paramBean

\* @return

\*/

private String getWorkFlowTitleInStart(ParamBean paramBean){

Bean bean = paramBean.getBean(WORK\_FLOW\_DEFINITION);

return ServUtils.replaceSysAndData(bean.getStr("BIND\_TITLE"), bean);

}

/\*\*

\* 获取当前节点名称

\*

\* @param outBean

\* @return

\*/

private String getNodeNameInStart(OutBean outBean){

return outBean.getBean(WORK\_FLOW\_INSTANCE).getBean(WORK\_FLOW\_ACT\_INSTANCE).getStr("NODE\_NAME");

}

/\*\*

\* 获取流程绑定的服务id

\*

\* @param paramBean

\* @return

\*/

private String getServiceIdInStart(ParamBean paramBean){

return paramBean.getBean(WORK\_FLOW\_DEFINITION).getStr("SERV\_ID");

}

/\*\*

\* 获取当前绑定的数据的主键

\*

\* @param paramBean

\* @return

\*/

private String getDataIdInStart(ParamBean paramBean){

return paramBean.getBean(WORK\_FLOW\_DEFINITION).getId();

}

@Override

protected void finalize() throws Throwable {

RTX.close();

super.finalize();

}

}

* + 1. RTX通知

在上述代码中，我们可以看到RTX的关联类为com.rh.core.plug.im.ImRtx,在使用之前我们必须对该类进行实例化并初始化。其发送通知的接口为：

/\*\*

\* 发送普通消息提醒

\*

\* **@param** receivers 接收人(多个接收人以逗号分隔)

\* **@param** title 消息标题

\* **@param** msg 消息内容，如果带连接，采用特殊格式如：[腾讯|http://www.qq.com]

\* **@return** true:发送成功 false:操作不成功

\*/

**public** **boolean** sendNotify(String receivers, String title, String msg)

* 1. 邮件发送
     1. 使用到的类

com.rh.core.util.email.MailSender

* + 1. 示例代码

MailSender mailSender = new MailSender("smtp.exmail.qq.com","test1@staff.ruaho.com","ruaho1234");

mailSender.setMailTo("yangjinyun@staff.zotn.com");

mailSender.setBody("test<b><font color='#FF0000'>能不能有中文呢！</font></b>");

mailSender.setSubject("test");

mailSender.setAuth(true);

mailSender.setBodyIsHTML(true);

mailSender.setDebug(true);

mailSender.setMailFrom("admin<test1@staff.ruaho.com>");

mailSender.addFile("测试文件.doc", "E:/书签定义.docx");

构造方法中的host、user、 password信息，最好通过“系统配置”进行配置，不要在代码中写死，这样方便上线维护。

* 1. cas单点登录配置
     1. 服务器端配置
        1. 准备工作

cas官方网站为：<http://www.jasig.org/cas>；

当前平台使用的cas server 版本是cas-server-3.5.2-release。将下载的zip文件解压缩，将cas-server-webapp-3.5.2.war部署到web服务器，作为单点登录服务器, 部署路径为/cas。

* + - 1. 添加数据源配置

在WebRoot\WEB-INF\deployerConfigContext.xml文件中添加以下内容：

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName">

<value>oracle.jdbc.OracleDriver</value>

</property>

<property name="url">

<value>jdbc:oracle:thin:@localhost:1521:orcl</value>

</property>

<property name="username"><value>dev\_cdp</value></property>

<property name="password"><value>cdp</value></property>

</bean>

* + - 1. 定义密码加密方式

在WebRoot\WEB-INF\deployerConfigContext.xml文件中添加以下内容：

<bean id="passwordEncoder"

class="org.jasig.cas.authentication.handler.DefaultPasswordEncoder"

autowire="byName">

<constructor-arg value="MD5"/>

</bean>

当前平台使用的加密方式为MD5。

配置authenticationManager下面的authenticationHandlers属性：

<bean

class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler">

<property name="dataSource" ref="dataSource" />

<property name="sql"

value="select USER\_PASSWORD from sy\_org\_user where USER\_LOGIN\_NAME = ?

AND CMPY\_CODE = 'ruaho' and S\_FLAG = 1" />

<property name="passwordEncoder" ref="passwordEncoder"/>

</bean>

其中sql语句是从数据库用户表中查询出登录用户密码，CMPY\_CODE为当前登录人的所属公司编码，系统可以根据需要对此sql语句进行更改。

* + - 1. 获取用户信息

修改attributeRepository的配置，通过jdbc查询用户的详细信息，可以把用户表或用户的所属组织机构或角色等查询出来：

<bean id="attributeRepository" class="org.jasig.services.persondir.support.jdbc.SingleRowJdbcPersonAttributeDao">

<constructor-arg index="0" ref="dataSource"/>

<constructor-arg index="1"

value="select USER\_CODE,USER\_NAME,USER\_EMAIL,USER\_MOBILE,

USER\_LOGIN\_NAME,USER\_LOGIN\_TYPE,USER\_OFFICE\_PHONE,

CMPY\_CODE, DEPT\_CODE,USER\_IDCARD,USER\_WORK\_NUM

from sy\_org\_user where {0} and S\_FLAG = 1 "/>

<property name="queryAttributeMapping">

<map>

<entry key="username">

<list><value>USER\_LOGIN\_NAME</value></list>

</entry>

</map>

</property>

<property name="resultAttributeMapping">

<map>

<entry key="USER\_CODE" value="USER\_CODE"/>

<entry key="USER\_NAME" value="USER\_NAME"/>

<entry key="USER\_EMAIL" value="USER\_EMAIL"/>

<entry key="USER\_MOBILE" value="USER\_MOBILE"/>

<entry key="USER\_LOGIN\_NAME" value="USER\_LOGIN\_NAME"/>

<entry key="USER\_LOGIN\_TYPE" value="USER\_LOGIN\_TYPE"/>

<entry key="CMPY\_CODE" value="CMPY\_CODE"/>

<entry key="DEPT\_CODE" value="DEPT\_CODE"/>

<entry key="USER\_IDCARD" value="USER\_IDCARD"/>

<entry key="USER\_WORK\_NUM" value="USER\_WORK\_NUM"/>

</map>

</property>

</bean>

配置authenticationManager中credentialsToPrincipalResolvers属性：

<bean

class="org.jasig.cas.authentication.principal.UsernamePasswordCredentialsToPrincipalResolver">

<property name="attributeRepository" ref="attributeRepository"/>

</bean>

默认cas登录服务器没有把用户信息传到客户端中，因此需要修改WEB-INF\view\jsp\protocol\2.0\casServiceValidationSuccess.jsp文件，修改为：

<%@ page session="false"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

<cas:authenticationSuccess>

<cas:user>${fn:escapeXml(assertion.chainedAuthentications[fn:length(assertion.chainedAuthentications)-1].principal.id)}</cas:user>

<c:if test="${not empty pgtIou}">

<cas:proxyGrantingTicket>${pgtIou}</cas:proxyGrantingTicket>

</c:if>

<c:if test="${fn:length(assertion.chainedAuthentications) > 1}">

<cas:proxies>

<c:forEach var="proxy" items="${assertion.chainedAuthentications}"

varStatus="loopStatus" begin="0"

end="${fn:length(assertion.chainedAuthentications)-2}" step="1">

<cas:proxy>${fn:escapeXml(proxy.principal.id)}</cas:proxy>

</c:forEach>

</cas:proxies>

</c:if>

<c:if

test="${fn:length(assertion.chainedAuthentications[fn:length(assertion.chainedAuthentications)-1].principal.attributes) > 0}">

<cas:attributes>

<c:forEach var="attr"

items="${assertion.chainedAuthentications[fn:length(assertion.chainedAuthentications)-1].principal.attributes}">

<cas:${fn:escapeXml(attr.key)}>${fn:escapeXml(attr.value)}</cas:${fn:escapeXml(attr.key)}>

</c:forEach>

</cas:attributes>

</c:if>

</cas:authenticationSuccess>

</cas:serviceResponse>

* + - 1. 取消https验证

修改WebRoot/WEB-INF/spring-configuration/ticketGrantingTicketCookieGenerator.xml文件中配置为：

<bean id="ticketGrantingTicketCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"

p:cookieSecure="false"

p:cookieMaxAge="-1"

p:cookieName="CASTGC"

p:cookiePath="/cas" />

其中cookieSecure设置为false。

* + 1. 客户端配置
       1. 准备工作

cas客户端使用版本为：cas-client-3.2.1-release，将下载的zip解压缩之后，将cas-client-core-3.2.1.jar文件加入编译路径中。

将平台提供的com.rh.client.AuthenticationFilter加入到编译路径中。

* + - 1. 添加单点登录过滤配置

在客户端web.xml中添加如下配置：

<context-param>

<param-name>serverName</param-name>

<param-value>http://172.16.0.113:8080</param-value>

</context-param>

<filter>

<filter-name>CAS Single Sign Out Filter</filter-name>

<filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Single Sign Out Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<listener>

<listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-class>

</listener>

<filter>

<filter-name>CAS Authentication Filter</filter-name>

<filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>

<init-param>

<param-name>casServerLoginUrl</param-name>

<param-value>http://172.16.0.113:9090/cas/login</param-value>

</init-param>

</filter>

<filter>

<filter-name>CAS Validation Filter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<param-value>http://172.16.0.113:9090/cas</param-value>

</init-param>

</filter>

<filter>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>

</filter>

<filter>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

<filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Authentication Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS Validation Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter>

<filter-name>local</filter-name>

<filter-class>com.rh.client.AuthenticationFilter</filter-class>

<init-param>

<param-name>EXCLUDE\_URL\_PATTERNS</param-name>

<!-- 多个值之间以分号间隔 -->

<param-value>.\*logout.\*</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>local</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

其中各配置项说明如下：

* serverName : 当前客户端应用地址；
* casServerLoginUrl ：cas服务器登录地址；
* casServerUrlPrefix ： cas服务器地址；
* EXCLUDE\_URL\_PATTERNS ： 避免cas过滤的地址正则表达式，一般默认是退出链接；
  + - 1. 平台登出配置

修改WebRoot/ sy/base/frame/engines/rhPageView.js文件中“退出”按钮的事件响应方法（大约在520行）：

jQuery("#loginOut").on("click", function(event) {//退出

var resultData = FireFly.logout();

if (resultData[UIConst.RTN\_MSG].indexOf(UIConst.RTN\_OK) == 0) {

//若开启单点登录需要使用下句

var homeUrl = FireFly.getContextPath() + "/logout.jsp";

Tools.redirect(homeUrl);

}

event.stopPropagation();

return false;

});

将var homeUrl = FireFly.getContextPath() + "/logout.jsp"之前的注释放开，使得改行代码得以执行。

修改WebRoot/logout.jsp文件，修改为：

<%

response.sendRedirect("http://172.16.0.123:8082/cas/logout");

%>

其中跳转地址为cas服务器登出地址。

* + - 1. 客户端过滤器

com.rh.client.AuthenticationFilter是平台提供的单点登录客户端过滤器，其主要逻辑在方法

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

不同客户端在使用时需要根据自身逻辑重写此方法。

* 1. 图表使用配置

图表组件使用HighCharts图表（<http://www.highcharts.com/>）实现， 具体配置使用方法可参考以下说明和系统自带的图表示例。

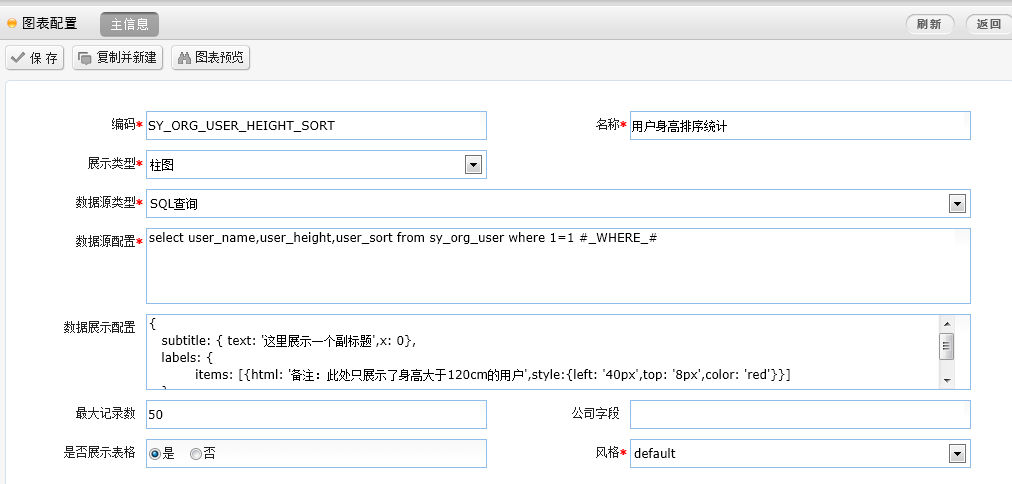
* + 1. 图表配置

图表配置路径：



* + - 1. 基础配置

图表配置主信息界面如下：



编码：图表的唯一标识

名称：图表名称

展示类型：目前支持柱图、饼图、线图、面图、条形图、时间图以及表格形式展示。

数据源类型：支持“SQL查询”和“服务”。

数据源配置：根据数据源类型的不同，配置也不相同。

一、“服务”类型，支持两种写法：

(1) 服务名(默认调用此服务的query方法)

(2) 服务名.方法名.do?data={}，如：SY\_ORG\_USER.methodA.do?data={}

data为JSON格式的字符串参数，后台接收时自动放入ParamBean中。

二、“SQL查询”类型：

编写SQL如：select USER\_NAME,USER\_HEIGHT from SY\_ORG\_USER where USER\_HEIGHT>160 \_WHERE\_

支持系统级变量(如：@DEPT\_CODE@)替换。

**注意：\_WHERE\_为带参数查询时的替换关键字，如果图表需要参数过滤，则需要在“SQL查询”类型的数据源配置中增加\_WHERE\_**

数据展示配置：自定义配置信息，JSON格式的字符串，如图表增加副标题：{subtitle: { text: '这里展示一个副标题',x: 0}}，更多高级配置参考highcharts的API。

是否展示表格：是否在图表下方展示表格数据。

风格：系统内置的几种图表风格。

参考系统内置的示例配置完图表主信息后，点击“保存”按钮。进行图表数据项配置：



数据项配置界面如下：



类型：“X轴”和“Y轴”，“X轴”为分类轴，“Y轴”为数据轴。如：本例中，用户为“X轴”，表示按用户分类，身高和排序号为“Y轴”。即：按用户分类，展示各用户的身高和排序信息。**图表配置至少需要一个“X轴”和一个“Y轴”**。

编号：与“服务”的字段或“SQL查询”的查询字段相对应。

展示类型：针对每个“Y轴”，都可以单独指定它的展示类型，目前支持“线图”、“面图”、“柱图”和“不展示”。“不展示”表示，此数据列不在图表中展示。业务是哪个一般不在图表中展示的，会在表格展示对应的列。同理，不在表格中展示的，会在图表中展示。否则配置此列则没有业务意义。

数据项配置：对“X轴”或“Y轴”的自定义配置信息，JSON格式的字符串，如label标签旋转45°：{'labels':{'rotation':45}}，更多高级配置参考highcharts的API。

字典编码：针对查询出来的列（编号对应的列），部分是数据字典的编码，需要转换为字典名称显示，此处配置此编号对应的数据字典即可展示名称。

对应URL：点击图表打开的链接。

显示合计：针对表格的配置，在列的最后增加合计值。

表格显示：配置此编号对应的列数据是否在表格中显示。

* + - 1. 如何单独引用

**如果想在iframe中或菜单中配置，引用方式为：**

**SY\_COMM\_INFO.chart.do?\_PK\_=SY\_ORG\_USER\_HEIGHT\_SORT**

**其中\_PK\_为图表配置的编码。**

**系统中模板组件中有部分图表的引用示例，引用方式就是以上方式。引用效果如下图红框所示：**

****

* + 1. 带参数查询的图表配置

一般情况下，图表展示需要参数查询过滤，系统中图表的参数查询是通过与“报表配置”结合使用的。配置路径：



* + - 1. 基础配置

点击进入“报表配置”列表页面



“添加”报表配置，界面如下：



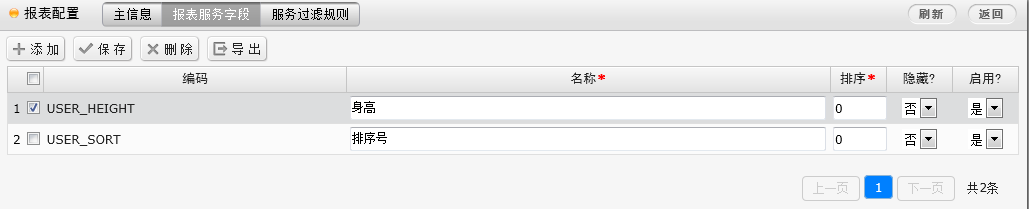
报表编码：唯一编号，配置引用时需使用，如引用此图表查询界面的方式为：

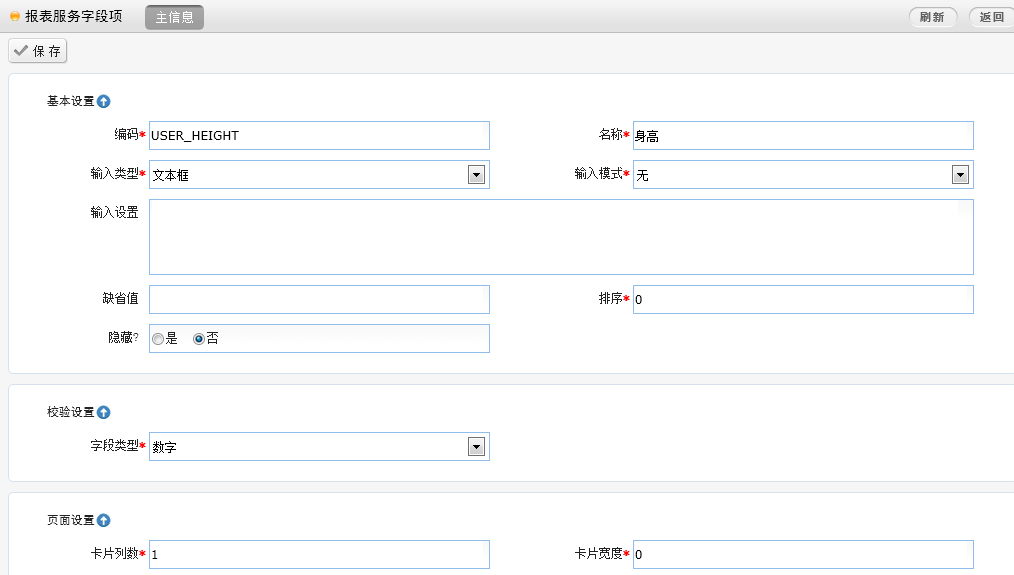
/sy/comm/page/REPORT\_TEST\_CHART.show.do。

报表类型：此处选择为“图表”。

报表服务路径：填写图表组件的编码，如我们上面配置的图表编码：SY\_ORG\_USER\_HEIGHT\_SORT。

“报表服务字段”和“服务过滤规则”和服务的配置相同，此处不详细说明，配置界面如下：





配置完成后，点击“报表预览”，效果如下：



其中红色方框为配置的“查询区”，点击“开始查询”展示的图表即我们在“图表配置”中配置的图表。

**注意：如果图表需要参数过滤，在图表配置时，需要在“SQL查询”类型的数据源配置中增加\_WHERE\_**

* + - 1. 如何单独引用

**如果想在iframe中或菜单中配置，引用方式为：**

**REPORT\_TEST\_CHART.show.do，**

**其中REPORT\_TEST\_CHART为报表服务的编码。**

* 1. 修改密码功能配置

平台中实际已提供了用户登录后修改个人信息，包括修改个人密码的功能，只是因为菜单配置上没将“修改密码”功能配置出来。

下图为配置好菜单前的显示情况：

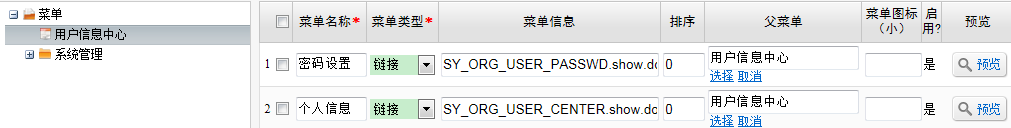


基本知识：点击用户头像后出现的“基本信息”页面使用的服务为SY\_ORG\_USER\_CENTER。修改用户密码的服务为“SY\_ORG\_USER\_PASSWD”。

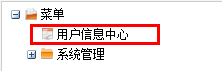
下面需要做的就是将这两个服务的菜单配置在同一根节点下。

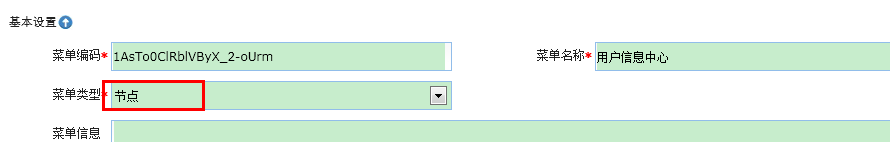
配置开始：

先来个总图



1. 在“菜单管理”中建立一个独立的根节点。





1. 配置“SY\_ORG\_USER\_CENTER”的菜单

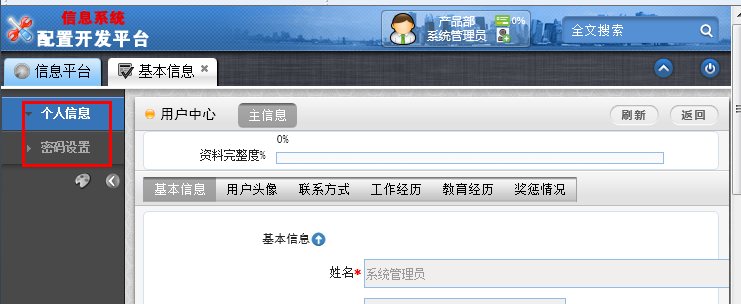


\*\* 菜单编码的构成：SY\_USER\_INFO\_\_（双下划线） + 公司编码

1. 配置“SY\_ORG\_USER\_PASSWD”的菜单



1. 完成。记得授权。展示一下效果



* 1. 配置菜单图标

如何为菜单配置自定义的图标呢？如图所示，主要分为配置小图标、大图标。



* + 1. 配置小图标（菜单图标）

1. 在数据字典SY\_COMM\_MENU\_ICON中新增一项。



1. 在common.css中新增一组样式，以上面的hetong为例。新增的样式为：

/\*合同\*/

.ui-state-default .leftMenu-hetong {

background: url(images/icons/rh-leftMenu-iconsAll.png) no-repeat 0px 0px;

border: 0px red solid;

}

.ui-state-active .leftMenu-hetong {

background: url(images/icons/rh-leftMenu-iconsAll.png) no-repeat -16px 0px;

}

.ui-state-default:hover .leftMenu-hetong {

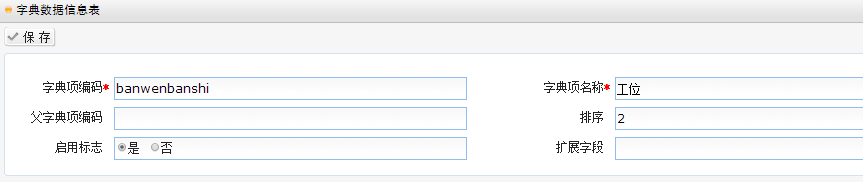
background: url(images/icons/rh-leftMenu-iconsAll.png) no-repeat -16px 0px;

}

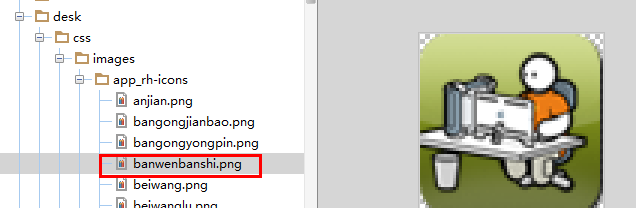
1. 将上面样式中用到的图片放在/sy/theme/default/images/icons/目录下。
   * 1. 配置大图标（桌面图标）

有2步。

1. 在数据字典SY\_COMM\_DESK中新增一项。



1. 在/sy/comm/desk/css/images/app\_rh-icons/放图标的图片，后缀为png，名字与“字典项编码”相同。



1. 平台前端
   1. 视图
      1. 卡片视图
         1. js文件

/sy/base/frame/engines/rhCardView.js

* + 1. 列表视图
       1. js文件

/sy/base/frame/engines/rhListView.js

* + 1. 字典选择弹出框视图
       1. js文件

/sy/base/frame/engines/rhDictTreeView.js

* + 1. 工作流卡片视图
       1. js文件

/sy/base/frame/engines/rhWfCardView.js

* + 1. 首页视图
       1. js文件

/sy/base/frame/engines/rhPageView.js

* + 1. 门户视图
       1. js文件

/sy/base/frame/engines/rhPortalView.js

* + 1. 下拉框视图
       1. js文件

/sy/base/frame/engines/rhSelectListView.js

* 1. 组件
     1. 卡片
     2. 表单
  2. 其他

1. 平台后端
   1. 参数
      1. ParamBean
      2. OutBean
   2. 服务扩展
      1. BaseServ
      2. CommonServ
   3. 服务调用
   4. 数据库
      1. SqlBean
      2. 数据库链接
      3. 多数据源

平台中进行多数据源查询需要按照以下方式：  
1. 获取到对应数据源的数据库连接：

Bean dsBean = Context.getDSBean(dsName)；

Connection connection = ((DataSource) dsBean.get(DS.DS)).getConnection();

2. 调用SqlExecutor的带Connection参数方法，如下所示：

Context.getExecutor(dsName).query(connection, sql)

由于平台在多数据源的实现上存在瑕疵，在使用服务之外的数据源时必须显式获取到对应的数据库连接。

1. 系统约束
   1. 公司编码

公司编码长度不能超过20，请在建立公司时就设定好编码，后续不能修改，否则公司所有数据失效。

* 1. 分组框

服务字段配置时，隐藏字段序号不能大于第一个分组框本身的序号（否则页面出现空白分组框）