**新光人寿**

**页面设计规范及标准白皮书**

**V1.1**

Sinosoft

**中科软科技股份有限公司**

**寿险事业部**

**2017年11月13日**

**关于本文件**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 主 题 | | 页面设计规范及标准白皮书 | | | | |
| 说　　明 | | 主要描述LIS7系统页面设计规范及标准的文档 | | | | |
| 适用对象 | | 基于LIS7框架基础进行项目开发的决策者和程序师 | | | | |
| 修 订 历 史 | | | | | | |
| 版 本 | 章 节 | | 类 型 | 日 期 | 作 者 | 说 明 |
| V0.1 |  | | C | 2017-09-10 | 赵雪峰 | 新增 |
| V0.2 |  | | U | 2017-09-13 | 徐云鹏 | 修订及定稿 |
| V1.0 |  | | U | 2017-09-30 | 徐云鹏 | 修订及定稿 |
| V1.1 |  | | U | 2017-11-10 | 韩雪  徐云鹏 | 修订及定稿 |
|  |  | |  |  |  |  |

说明：类型－新增（C）、修改（U）、删除（D）；

**评审记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **评 审 记 录** | | | |
| **角 色** | **签 名** | **日 期** | **说 明** |
| PM | 刘昊 | 2017-09-25 | 审查v0.2 |
| PM | 刘昊 | 2017-10-06 | 审查v1.0 |
| PM | 刘昊 | 2017-11-13 | 审查v1.1 |

**目 录**

[前言 6](#_Toc498506526)

[文档说明 6](#_Toc498506527)

[1 页面权限控制 7](#_Toc498506528)

[1.1 使用者账号 7](#_Toc498506529)

[1.1.1 usrdt 7](#_Toc498506530)

[1.2 选单 7](#_Toc498506531)

[1.2.1 menuitemdt 7](#_Toc498506532)

[1.2.2 usrmenu 7](#_Toc498506533)

[1.3 按钮 8](#_Toc498506534)

[1.3.1 guidt 8](#_Toc498506535)

[1.3.2 usrsec 8](#_Toc498506536)

[2 项目程序结构说明 10](#_Toc498506537)

[2.1 项目结构 10](#_Toc498506538)

[2.2 本项目程序结构 11](#_Toc498506539)

[3 命名规则 12](#_Toc498506540)

[3.1 档案命名 12](#_Toc498506541)

[3.2 变量命名 12](#_Toc498506542)

[4 建档说明 13](#_Toc498506543)

[5 JSP前台页面开发规范 19](#_Toc498506544)

[5.1 Jsp页面完整范例 19](#_Toc498506545)

[5.2 JSP自订标签 20](#_Toc498506546)

[5.2.1 使用说明 20](#_Toc498506547)

[5.3 自订标签使用规范 20](#_Toc498506548)

[5.3.1 标签公共属性介绍 20](#_Toc498506549)

[5.3.2 内容块标签 20](#_Toc498506550)

[5.3.3 表格标签 21](#_Toc498506551)

[5.3.4 普通输入框 22](#_Toc498506552)

[5.3.5 下拉选择框 23](#_Toc498506553)

[5.3.6 唯读框 25](#_Toc498506554)

[5.3.7 日期输入框 26](#_Toc498506555)

[5.3.8 按钮标签 27](#_Toc498506556)

[5.3.9 隐藏输入框 29](#_Toc498506557)

[5.3.10 密码输入框 30](#_Toc498506558)

[5.3.11 Textarea内容块 31](#_Toc498506559)

[5.3.12 Grid列表 32](#_Toc498506560)

[5.3.13 数字输入框 33](#_Toc498506561)

[6 JS写法规范 35](#_Toc498506562)

[6.1 JS文件新建范例 35](#_Toc498506563)

[6.2 属性选择器 35](#_Toc498506564)

[6.3 按钮的提交编写 35](#_Toc498506565)

[6.3.1 使用$.post方式提交 37](#_Toc498506566)

[6.3.2 GET请求 38](#_Toc498506567)

[6.4 民国年转换 38](#_Toc498506568)

[6.4.1 Js中进行转换 38](#_Toc498506569)

[6.4.2 Controller档中进行的转换： 39](#_Toc498506570)

[6.5 页面上显示提示信息 39](#_Toc498506571)

[6.5.1 showMessage使用 39](#_Toc498506572)

[6.5.2 doMessageGo使用 39](#_Toc498506573)

[7 JSON配置规范 41](#_Toc498506574)

[7.1 Json档新建范例 41](#_Toc498506575)

[7.2 LisGrid的设定档 41](#_Toc498506576)

[7.2.1 表头 42](#_Toc498506577)

[7.2.2 Model 43](#_Toc498506578)

[7.2.3 单选和多选 43](#_Toc498506579)

[7.2.4 各列的属性 44](#_Toc498506580)

[7.2.5 清单的样式 46](#_Toc498506581)

[7.2.6 初始化完成事件 47](#_Toc498506582)

[7.2.7 API 47](#_Toc498506583)

[7.2.8 选中一行事件 48](#_Toc498506584)

[7.3 校验规则配置 48](#_Toc498506585)

[7.3.1 pageUrl 49](#_Toc498506586)

[7.3.2 rules 49](#_Toc498506587)

[7.3.3 常见的校验 50](#_Toc498506588)

[7.3.4 与其他组件相关的校验 51](#_Toc498506589)

[7.3.5 自订校验 52](#_Toc498506590)

[7.3.6 添加自订校验属性 54](#_Toc498506591)

[8 controller文件规范 56](#_Toc498506592)

[8.1 写法规范 56](#_Toc498506593)

[8.2 请求与后台访问 56](#_Toc498506594)

[8.3 响应 58](#_Toc498506595)

[9 调用后台方法规范 59](#_Toc498506596)

[9.1 页面数据准备 59](#_Toc498506597)

[9.2 访问js方法 59](#_Toc498506598)

[9.3 Controller档中的操作 60](#_Toc498506599)

[10 权限动态配置 62](#_Toc498506600)

[11 清单查询书写方式 63](#_Toc498506601)

[11.1 语法 63](#_Toc498506602)

[11.2 说明 65](#_Toc498506603)

[12 Nt\_rec与列表资料转换 66](#_Toc498506604)

[12.1 使用Nt\_rec初始化页面列表 66](#_Toc498506605)

[12.2 封装选中的列表中的数据爲Nt\_rec 66](#_Toc498506606)

[12.2.1 Json中的配置 66](#_Toc498506607)

[12.2.2 Js中的代码 67](#_Toc498506608)

[12.2.3 Controller档中的操作 68](#_Toc498506609)

[13 页面之间的值传递 69](#_Toc498506610)

[13.1 功能目的 69](#_Toc498506611)

[13.2 功能实现 69](#_Toc498506612)

[13.2.1 储存值 69](#_Toc498506613)

[13.2.2 取值 69](#_Toc498506614)

[14 通过点击按钮跳转页面 71](#_Toc498506615)

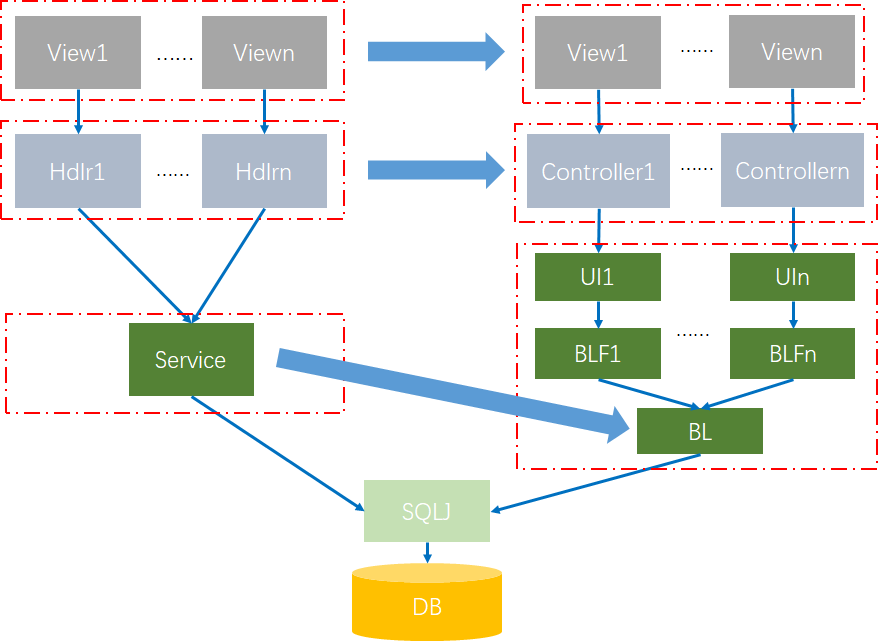
[15 commit和rollback 72](#_Toc498506616)

# 前言

文档说明

本文档编写的目的是爲了使开发人员，通过本文档可以顺利的进行开发工作。本文档将从页面开始，介绍开发人员在使用LIS7架构开发的过程中，可能需要用到的方法。

如下图所示，本文档主要说明前台，即view和controller的开发规范。其中UI,BLF,BL的部分爲后台。



1. 页面权限控制

因为前端使用者的账号会影响该使用者可以看到的程序选单内容、按钮是否能按的权限，所以在这边先简单介绍记录这些信息table的比较重要的栏位，以利后面开始写JSP时可以比较容易了解为什么要这样写指令。

* 1. 使用者账号
     1. usrdt

核心系统使用者主档，用来记录使用者的相关信息及权限。

* user\_code：使用者代号。
* center\_code：使用者单位代号。
* start\_date：使用者的账号启用日。
* end\_date：使用者的账号停用日。
* usr\_group\_id：使用者的选单编号，用来关联usrmenu的menu\_tree\_id栏位。
* related\_dummy\_user：使用者的按钮权限群组，用来关联usrsec的user\_code栏位。
  1. 选单
     1. menuitemdt

设定各节点(node)的主档，开发人员通常只需要定义好自己新开发完成的程序节点即可，节点要放的位置给使用者自行决定。

* menu\_item\_id：节点的代号，命名规则为：程序代号+\_n(n从1开始)。
* URL：节点的超连结路径；空值代表是子目录，非空值代表是子功能。
* menu\_item\_name：节点显示在画面上的名称。
  + 1. usrmenu

设定使用者可以看到的选单内容档。

* menu\_tree\_id：选单树的代号。
* parent\_menu\_id：父节点的代号。
* menu\_item\_id：子节点的代号。
* display\_seq：子节点的显示顺序。

**父节点**



**当按下最上一排的分页或子目录选单时**

**会用menuItemId当作父节点产生选单**

**menuitemdt.URL不为null的子节点**

**menuitemdt.URL为null的子节点**

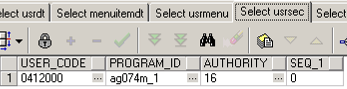
* 1. 按钮
     1. guidt

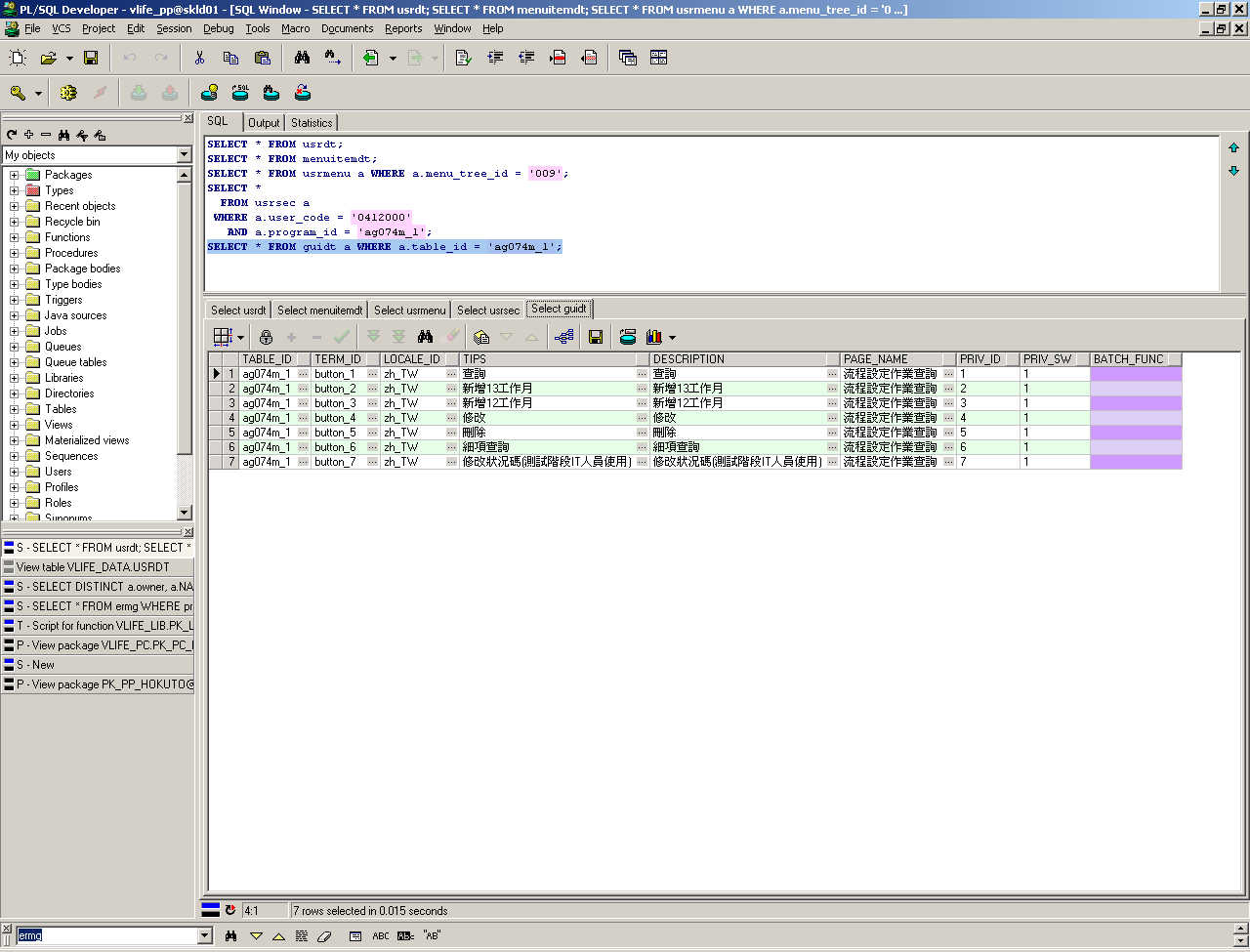
设定子程序作业(子节点)画面内需管控的或于sys没有定义的按钮；其中需控管的按钮为程序各作业入口页面的按钮。

* table\_id：节点的代号。
* term\_id：按钮编号，若需控管的按钮需从1开始编号，数字用完换英文字母；若不需控管的按钮可以从10开始编号。
* description：按钮显示在画面上的文字。
* priv\_id：按钮的权限编号，会等于term\_id的最后一码，若为null代表没有控管。
* priv\_sw：权限控管开关，1代表要控管。
  + 1. usrsec

设定使用者的按钮权限群组可以使用各节点的哪些按钮。

* user\_code：使用者的按钮权限群组，可以用来关联usrdt的related\_dummy\_user找出有哪些使用者有相同的按钮权限。
* program\_id：节点的代号，用来关联guidt的table\_id。
* authority：使用者的按钮权限群组在该节点可以使用的按钮权限编号字符串，会是guidt的priv\_id所组成的字符串。

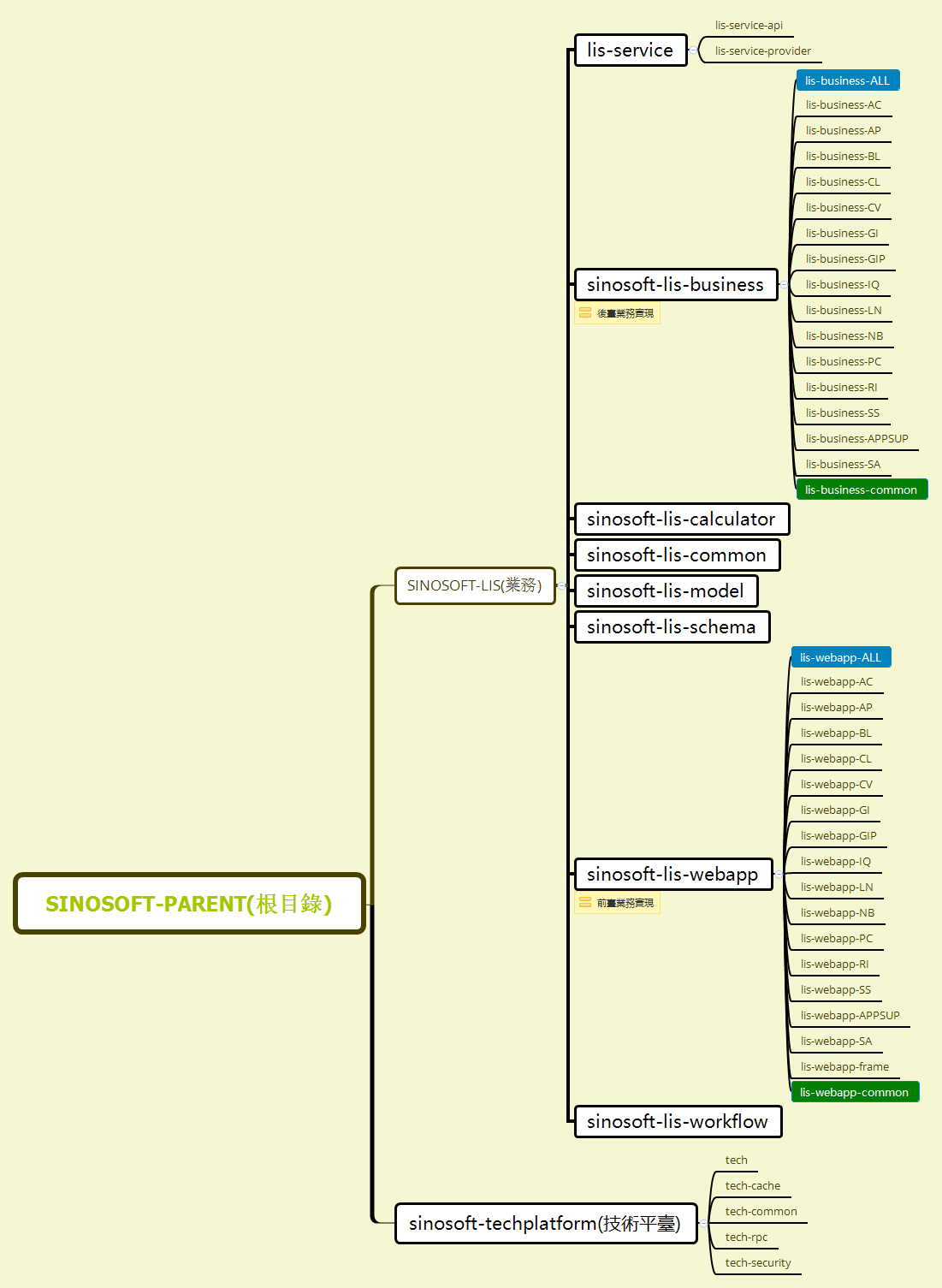




1. 项目程序结构说明
   1. 项目结构

针对本项目，LIS程序结构修改为13线后，项目结构为，每一线有对应的前后台项目，每一线的前台都依赖lis-webapp-common且都依赖在lis-webapp-ALL中；每一线的后台都依赖lis-business-common且都依赖在lis-business-ALL中。lis-webapp-ALL将lis-business-ALL依赖进去，达到前后台一起的目的。SA模块为系统管理是必备模块。Webapp-frame为前台公共的页面。

下图中，sinosoft-lis-business是后台UI、BLF、BL档放置的地方，sinosoft-lis-webapp是前台jsp，js，json，controller档放置的地方。



* 1. 本项目程序结构

本项目程序结构沿用LIS7原有程序结构不变。

1. 命名规则
   1. 档案命名

每个M的命名规范同UIX系统的命名规则。详情请参照8个template各view的命名规则。即第一个页面爲Br开头，新增页爲Ad开头，修改页爲Up开头。如下所示：

BrPc089mView1.jsp。

其中Br是前缀，如还有Ad，Up等。Pc089m是当前页面所属的m，View1是页面的编号，还有View2,View3等。

* 1. 变量命名

用到的变量一部分从uix进行迁移而来,自己定义的变量遵循以下规则:

变量名和方法名小写开始的驼峰命名，例如：studentParentName

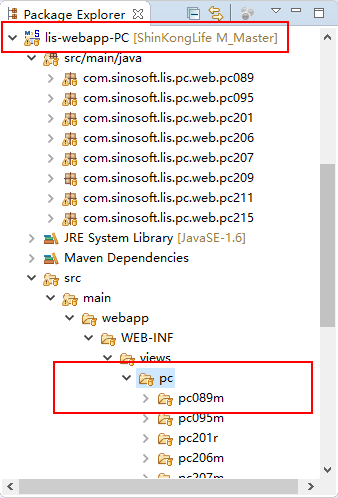
常量名用全大写：MAX\_PARAMETER\_COUNT = 100

枚举类名参考普通类名，枚举变量使用全大写。

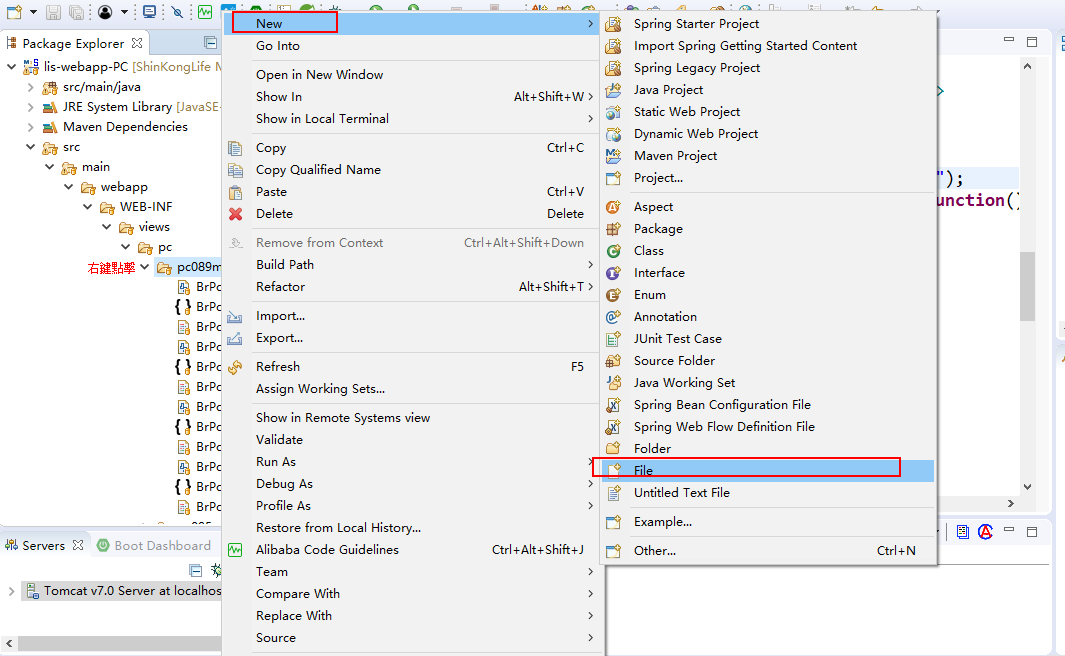
1. 建档说明

前台文档包含jsp，js，json，controller文档，需要在开发某个M时新建这些文档。新建文档的方法，如下所示：

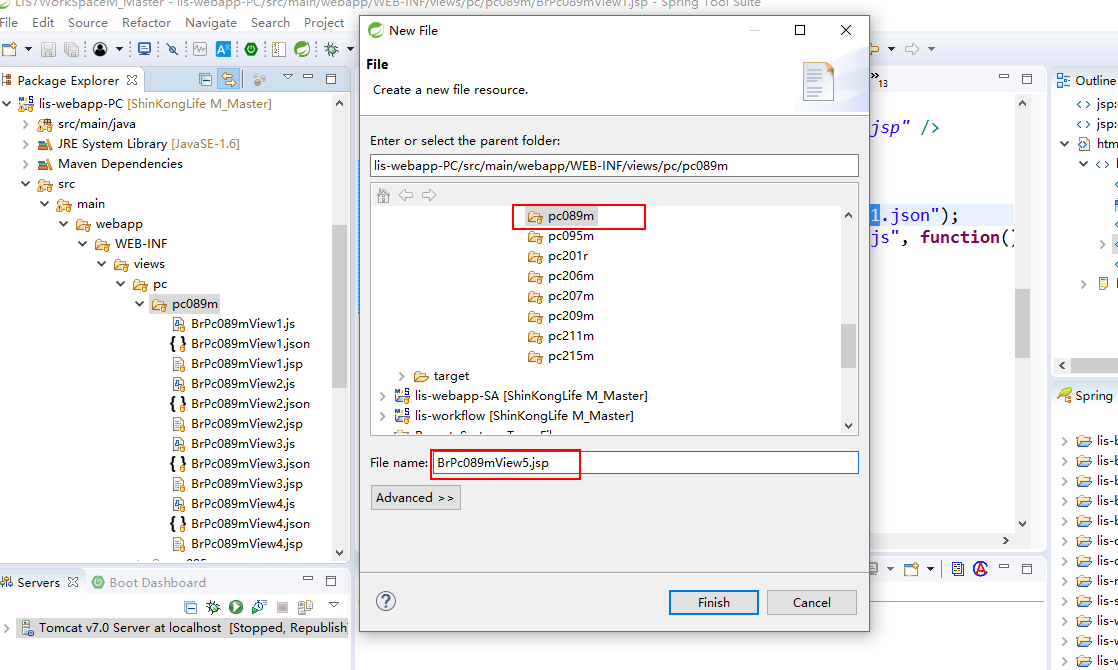
例如现在要在pc线下添加功能页面，需要添加pc线下的某个M的资料夹来放置jsp，js，json档。如，pc089m



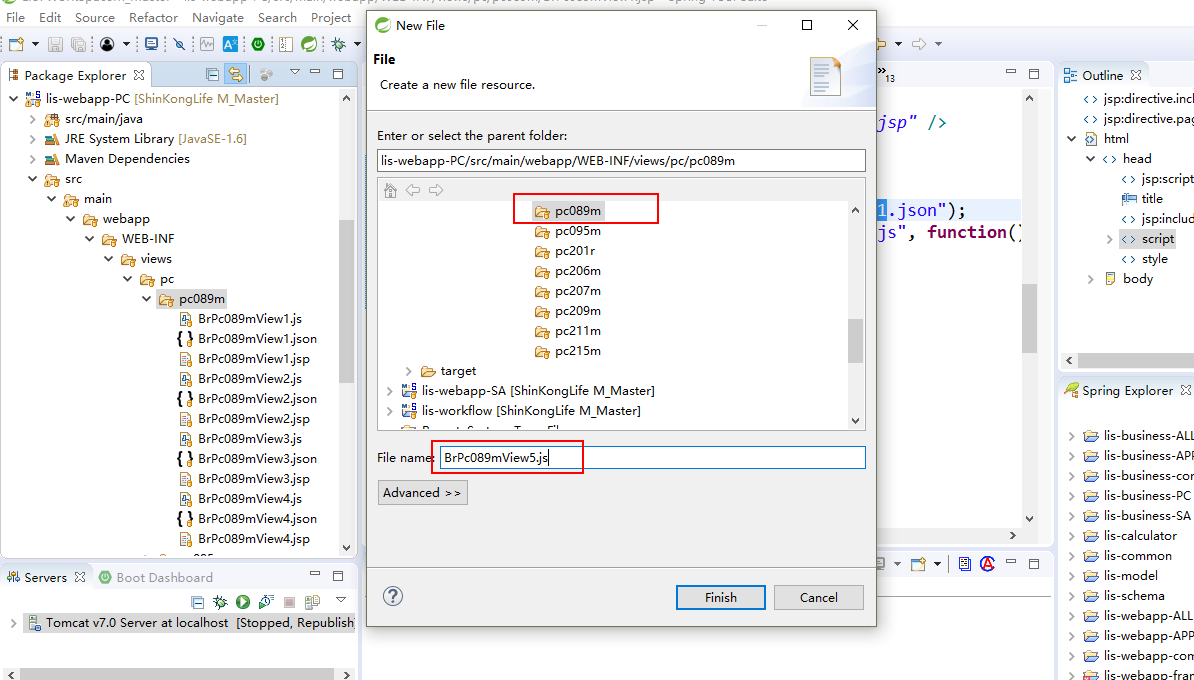
具体建立jsp,js,json三个档的方式爲：



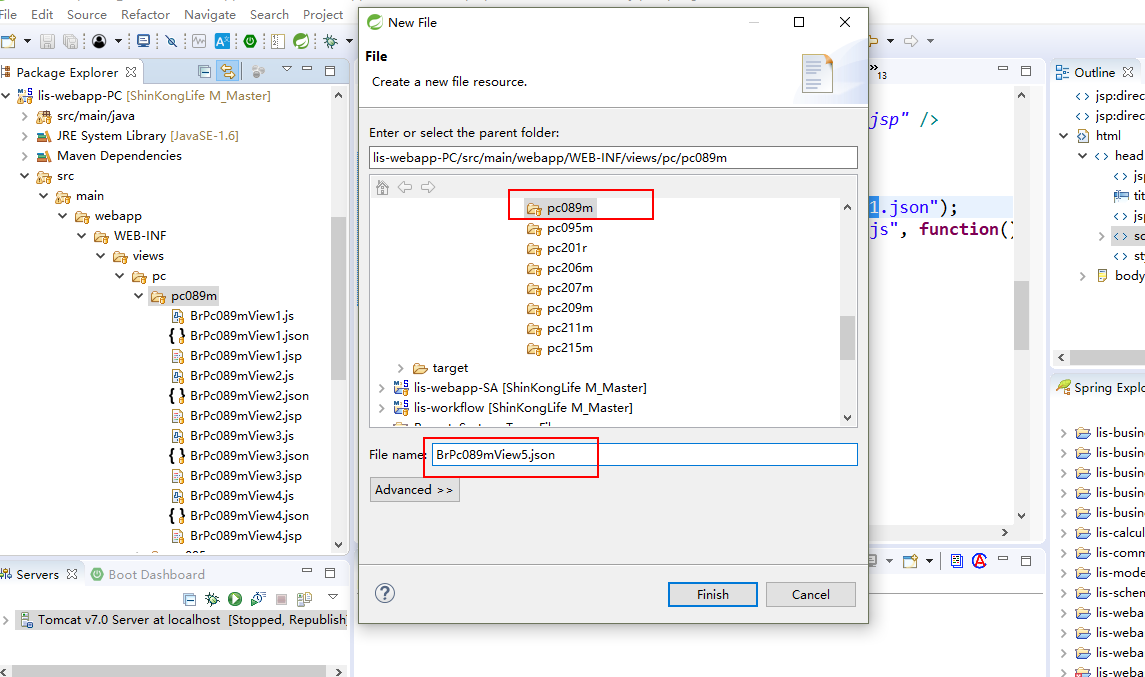
Jsp档案：



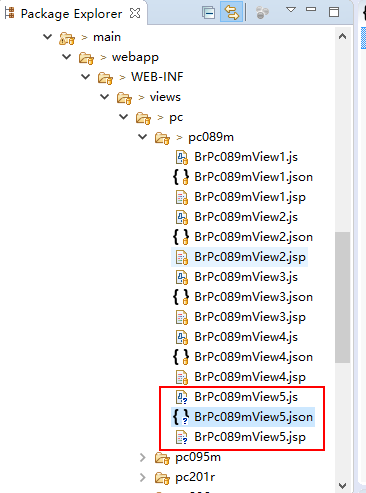
Js档案：



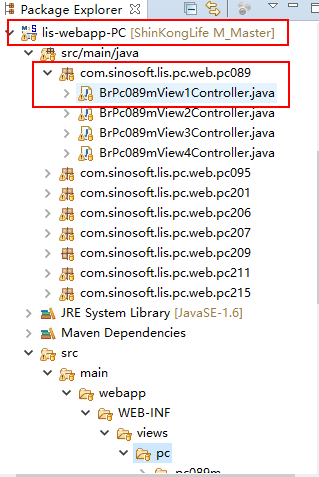
Json档案：



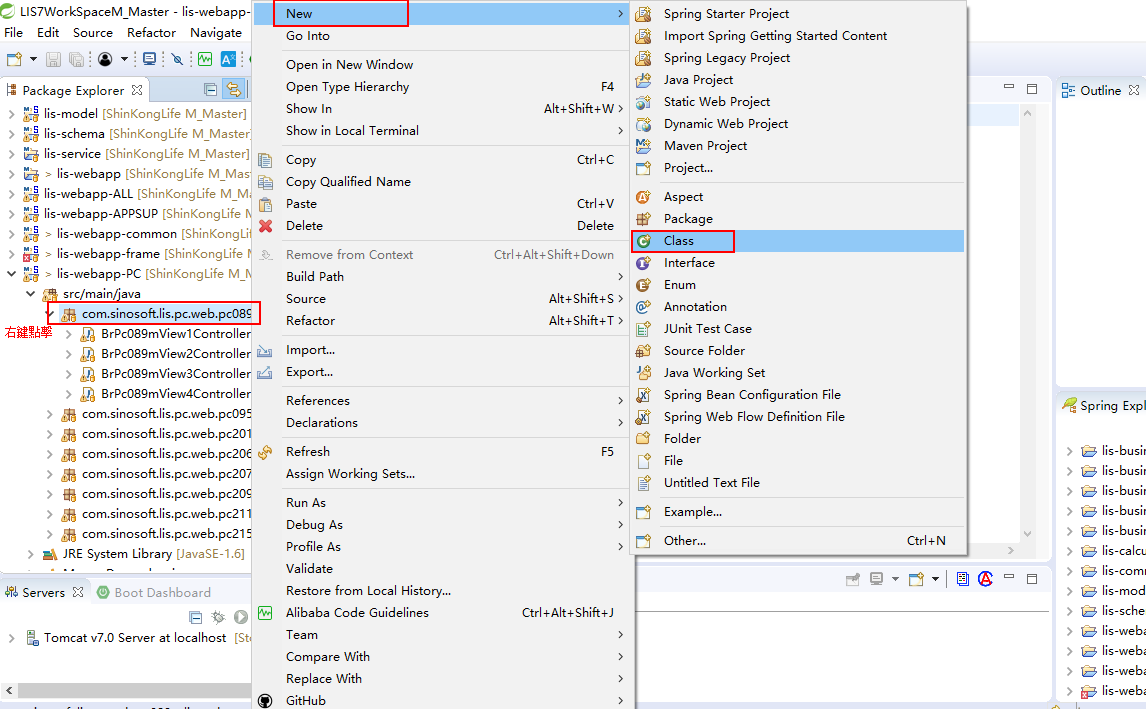
最后建成完毕后：

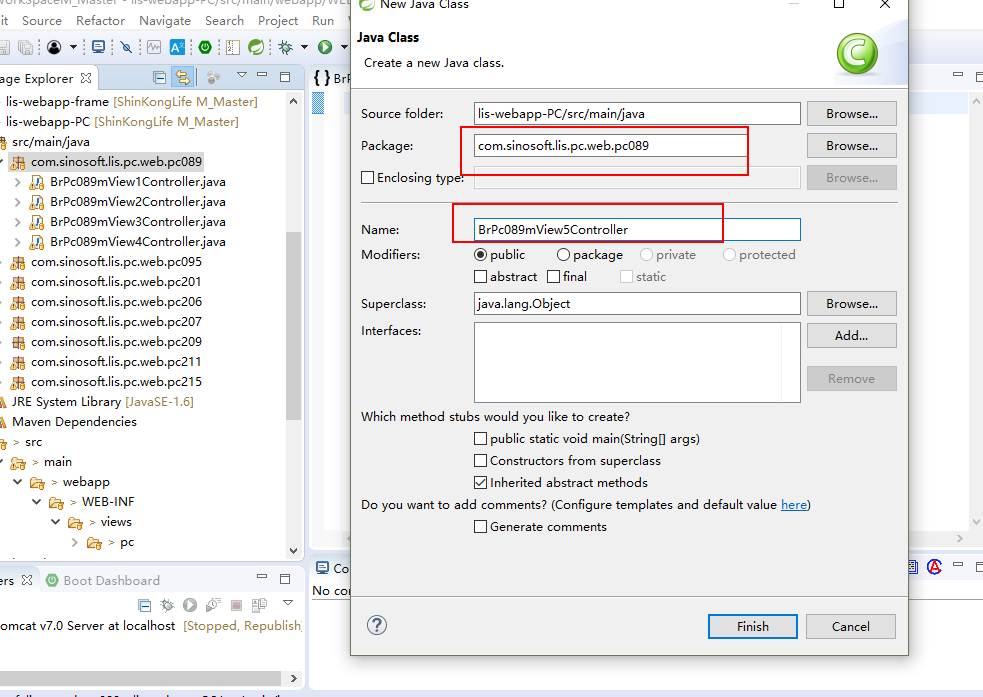


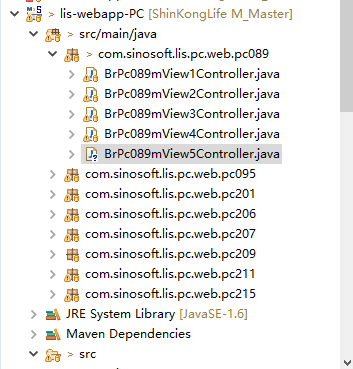
并在src/main/java/com/sinosoft/lis/pc/web/资料夹下建立相应的controller文档：



Controller档案是java文档，建立方式如下：







1. JSP前台页面开发规范
   1. Jsp页面完整范例

建立完毕jsp档后，需要将一些公共的代码贴过来，所要粘贴的代码如下所示：

|  |
| --- |
| <!-- 标签库引入文档 -->  <%@ include file="/WEB-INF/includes/PageInclude.jsp"%>  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <html>  <head>  <%request.setCharacterEncoding("utf-8");response.setContentType("text/html;charset=utf-8"); %>  <title>页面名称</title>  <!-- 引用档案所在jsp -->  <jsp:include page="/WEB-INF/includes/PageHeader.jsp" />  <script>  //需要在页面初始化时引入的js和json档  $(document).ready(function(){  loadJS("pc/pc089m/BrPc089mView1.js", function() {  //引入档案成功后的后续动作  });  loadJson("pc/pc089m/BrPc089mView1.json");  });  </script>  </head>  <body>  <div id='lis'>  <div id='Info' style="display:none">  <L:common prelabel="Info" >  <span id="returnInfo"></span>  </L:common>  </div>  <!-- 页面具体内容 -->  </div>  <!-- 存放token档案 -->  <%@ include file="/WEB-INF/includes/PageFooter.jsp"%>  </body>  </html> |

* 1. JSP自订标签
     1. 使用说明

为了提高代码的复用率和规范，抽象类别AbstractTagLib封装了许多功能。页面上的标签尽可能要使用自订标签，可以使得jsp部分更加整洁，易于维护。

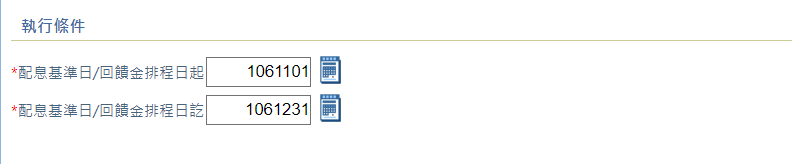
* 1. 自订标签使用规范
     1. 标签公共属性介绍

以下属性是每个自定义标签都有的属性，可根据需要使用。必须要有的属性会用红色字体进行重点标注，黑色爲根据需要的选用属性（jsp公共标签可能不适用下列属性值的说明，需要根据公共标签语法自行设定属性值）：

|  |  |
| --- | --- |
| id | 页面上控制项的唯一标示，页面上控制项的id值不能重复。 |
| name | 页面上控制项的名字属性，页面上控制项的name值可以重复。 |
| prelabel | 页面上控制项前边显示的名称。L:button控制项，可不使用本属性 |
| disabled | 定义控制项是否可用 |

* + 1. 内容块标签

1 展示效果



内容块标签

2 标签语法

|  |
| --- |
| <body>  <L:common prelabel="执行条件" id="divDo">  /\*\*\*\*\*\*\*\*\*中间的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:common>  </body> |

3 说明

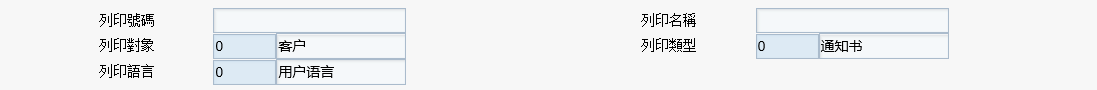
该自订标签名为 L:common（在lis7.tld中定义，下同）。

prelabel的值是内容块的标签的名称。

id的值是内容块的id。内容块的左上角有收放按钮，块周有分界线。

* + 1. 表格标签

1 展示效果



表格标签

2 标签语法

|  |
| --- |
| <L:common prelabel="打印定义" id="divPrint1">  <L:table>  <tr>  /\*\*\*\*\*\*\*\*\*一行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table>  </L:common> |

3 说明

该自订标签名为 L:table，用于将内容像表格一样排列。

<tr>标签代表一行的内容，有几个<tr></tr>的标签对，代表有几行。

* + 1. 普通输入框

1 展示效果



普通输入框

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:input id="PrintName" name="PrintName" prelabel="打印名称"/>  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:input，用于展示一个输入框。在表格(L:table)中使用时将会自动加上td属性，用于表示该输入框是单独一列。

4 特别属性

placeholder是该输入框的水印内容，如提示输入（请输入初审员姓名）等。

onclick：代表该输入框的onclick属性，即单击事件。如 onclick=”doOperate(‘addInsured’)”。

ondblclick：代表该输入框的ondblclick属性，即双击事件。如 ondblclick= ”doOperate(‘add’)”。

value: 代表该框的value值。

size：代表该输入框的长度。

controller:用于动态赋予权限。在第11章中有具体用法。

5 控制项赋值、取值

赋值：$("# PrintName ").val(“需要赋予的值”);

取值：var printName = $("# PrintName ").val();

6 校验规则

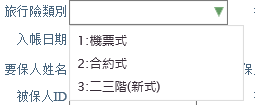
详细配置见13章。

* + 1. 下拉选择框

1 展示效果



下拉时效果图



下拉选择框

2 标签语法

|  |
| --- |
| <L:codeselect id="TaType" name="TaType"  prelabel="旅行险类别" codetype="ta\_type" /> |

3 说明

该自订标签名为 L:codeselect，用于展示一个从指定代码中选择内容的输入框。

4 特别属性

placeholder是该输入框的水印内容，如提示输入（请选择管理机构）等。

codetype：代表该下拉代码的名称，用于后台查找下拉清单的内容。

initValue：代表该输入框的初始化值，即载入时的预设值。

readonly：代表该下拉输入框的readonly属性，即是否唯读属性。如readonly=”true”。

controller:用于动态赋予权限。在第11章中有具体用法。

childData: 显示哪些选项、不显示哪些选项,childData=”0$9$C”那么下拉清单将不显示9和C ,childData=”1$9$C”那么下拉清单只显示9和C

Selectfunc:当选择某一下拉选项时,触发此函数相当于onChange事件

5 控制项赋值、取值

同普通输入框的赋值、取值

6 校验规则

详细配置见13章

7 其他用法

ａ）刷新下拉资料

$('.lisCode').LisCode();

这里的$('.lisCode')是jQuery的语法，用于选择所有属性class有lisCode的页面物件。下拉选择框，<L:codeselect />都会被选择。

这个方法在页面初始化时会执行一次，后续根据开发人员需求，比如通过js代码给控制项赋值之后，需要在赋值后调用一下这个方法，刷新下拉资料框中的值。

６ 下拉资料准备

页面初始化成功后，从缓存中根据页面上的下拉控制项的codetype，得到在ldcode资料表和trmval资料表中配置的相应的资料。

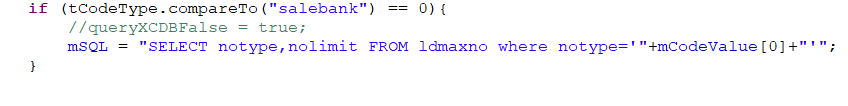
没有在ldcode或者trmval中配置的下拉资料，可在lis-business-common\src\main\java\com\sinosoft\lis\common\business\

CodeQueryBLNew.java类中添加，支援带参数的查询。带参数的查询方式是在codetype中添加“?”和参数值，比如”agentcode?0010,0013”，在codequeryblnew中使用相应的参数就好。

下面是一个带参数的例子，具体的配置如下

jsp：

Codequeryblnew:



参数个数没有限制，在CodequeryBLNew.java中用mCodeValue数组和页面上的参数一一对应，拼接到sql语句中。

* + 1. 唯读框

1 展示效果



唯读框

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:readonly id="PrintID" name="PrintID" prelabel="打印号码"/>  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:readonly，用于展示一个唯读的输入框。

4 特别属性

placeholder：同普通输入框的placeholder属性。

value：代表该框的value值（该值不可以通过页面输入更改）。

onclick：同普通输入框的onclick属性。

ondblclick：同普通输入框的ondblclick属性。

5 控制项赋值、取值

同普通输入框的赋值、取值

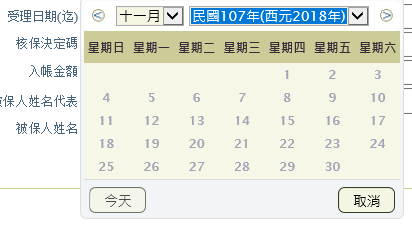
6

* + 1. 日期输入框

1. 展示效果



展开后



日期输入框

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:date id="endDate" name="endDate" prelabel="受理日期(迄)" />  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:date，用于展示一个带日期元件的输入框。当选中该框时，将弹出日期元件，离开后日期元件隐藏。日期控制项的对齐方式现爲右对齐。

4 特别属性

placeholder：同普通输入框的placeholder属性。

onclick：同普通输入框的onclick属性。

ondblclick：同普通输入框的ondblclick属性。

size：代表该输入框的长度。

datetype:如果值爲TW则显示民国年，没有此属性或者爲CN则爲公元年，不可填写其他选项。

5 控制项赋值、取值

同普通输入框的赋值、取值

* + 1. 按钮标签

1 展示效果



按钮表格

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:button id="agin" name="agin" prelabel="FullName" url="ta003m\_1#button\_1" onclick="agin();"/>  </tr>  </L:table> |

3 说明

按钮自订标签名为 L:button，定义一个页面上的按钮。

4 特别属性

onclick：代表该输入框的onclick属性，即单击事件。如 onclick=”doOperate(‘addInsured’)”。

ondblclick：代表该输入框的ondblclick属性，即双击事件。如 ondblclick= ”doOperate(‘add’)”。

url：属性值是根据guidt资料表中的table\_id#term\_id拼写出来的。用来控制按钮的权限和国际化。

按钮的权限爲当前页面所属菜单的url拼接上term\_id。即，如果某页面的访问url爲“/pc/pc089m/BrPc089mView3”，那么按钮的权限爲/pc/pc089m/BrPc089mView3/term\_id。这是程序自动拼接的，只需要在访问controller时的url跟/pc/pc089m/BrPc089mView3/term\_id保持一致。

具体如下图，四点一致原则:



controller:用于动态赋予权限。在第11章中有具体用法。

* + 1. 隐藏输入框

1 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:hidden id="hiddenAddress" prelabel="隐藏内容" />  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

2 说明

该自订标签名为 L:hidden，用于表示一个隐藏的输入框。隐藏框可以在js中进行赋值操作。

3 控制项赋值、取值

同普通输入框的赋值、取值

4 使用实例

* + 1. 密码输入框

1 展示效果



密码输入框

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:password prelabel="PassWord" id="password" placeholder="PassWord" />  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:password，用于展示一个密码输入框。

4 特别属性

placeholder是该输入框的水印内容，如提示输入（请输入初审员姓名）等。

size：代表该输入框的长度。

maxlength：代表该输入框允许输入文本的最大长度。

onclick：代表该输入框的onclick属性，即单击事件。如 onclick=”doOperate(‘addInsured’)”。

ondblclick：代表该输入框的ondblclick属性，即双击事件。如 ondblclick= ”doOperate(‘add’)”。

* + 1. Textarea内容块

1 展示效果



长文本输入框

2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:textarea prelabel="备注" id="remark" name="remark" tdcolspan="10" cols="100" rows="2" />  </tr>  /\*\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:textarea，用于展示一个多行文字方块。

4 特别属性

tdcolspan：代表该文字方块在table中占据几列，默认是10列（因为一般文字方块都比较长），相当于<td colspan=”10”><textarea></textarea>。

cols：是该文字方块的宽度（列数），默认是100

rows：是该文字方块默认的高度（行数），默认是2

placeholder是该文字方块的水印内容，如提示输入（请输入备注）等。

value：代表该文字方块的预设内容。

onclick：代表该文字方块的onclick属性，即单击事件。如 onclick=”doOperate(‘addInsured’)”。

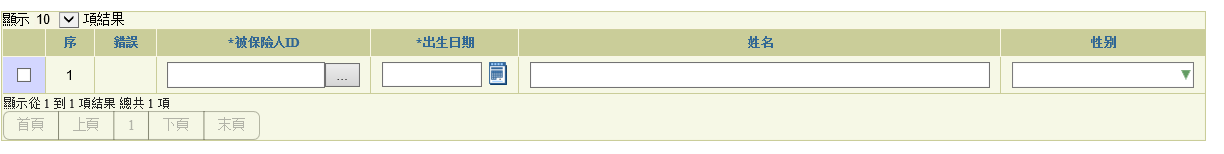
ondblclick：代表该文字方块的ondblclick属性，即双击事件。如 ondblclick= ”

doOperate(‘add’)”。  
5 控制项赋值、取值

同普通输入框的赋值、取值

* + 1. Grid列表

1 展示效果



GRID

2 标签语法

|  |
| --- |
| <body>  <L:grid id="agentImpartList"></L:grid>  </body> |

3 说明

该自订标签名为 L:grid，用于展示一个LisGrid，即列表。

id代表该LisGrid的id值。

其他的LisGrid属性由JSON档来配置（详见第8章）。

该标签生成的HTML内容是

|  |
| --- |
| <table id="agentImpartList"></table> |

* + 1. 数字输入框

1 展示效果



2 标签语法

|  |
| --- |
| <L:table>  <tr>  <L:sumnum id="number" prelabel="金额" />  </tr>  /\*\*\*\*\*\*\*\*其他行的自订标签，略\*\*\*\*\*\*\*\*\*\*\*/  </L:table> |

3 说明

该自订标签名为 L:sumnum，用于展示一个纯数字文本。

id代表该input框的id值。

prelabel是该输入框的前置标签（和内容块的prelabel属性一样，是用于实现多语言化的）。

placeholder是该输入框的水印内容，如提示输入（请输入初审员姓名）等。

除了上述例子中几个属性外，还有以下属性。

name：代表该输入框的name属性。

onclick：代表该输入框的onclick属性，即单击事件。如 onclick=”doOperate(‘addInsured’)”。

ondblclick：代表该输入框的ondblclick属性，即双击事件。如 ondblclick= ”doOperate(‘add’)”。

pointafter:代表在小数点后可以存在的最大位数。

pointbefore：代表在小数点之前可以存在的最大位数。

isthousands：代表是否使用千分位进行数字展示，其中“Y”代表使用千分位展示。

1. JS写法规范
   1. JS文件新建范例

所有js档在新建后，需要添加如下代码，其他方法需要自行实现：

|  |
| --- |
| //jQuery 当前页面中的body  var $body=$("body");  //提示信息div  var $info = $("#Info");  //提示信息具体内容  var $returnInfo = $("#returnInfo");  //列表中的数据  var rowData = null; |

* 1. 属性选择器

LIS7环境在js中对页面上的控制项进行赋值或者取值操作。语法使用jQuery的标准语法，现在仅作获取控制项方式的介绍，其他jQuery的用法，请参照jQuery使用文档。

获取控制项的方式，如下所示:

$('#submitBtn')，

$:jQuery属性选择器

#:id选择器。

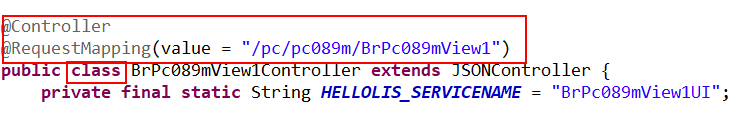
上述语句的功能是得到一个id爲submitBtn的控制项。

* 1. 按钮的提交编写

以下js请求controller时，url的书写规范爲：

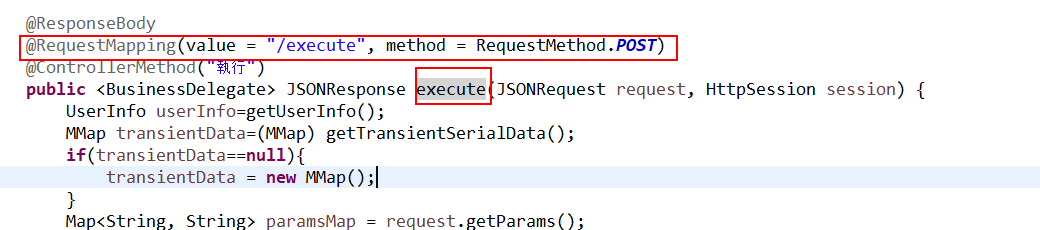
在controller档中（第9章介绍controller），类别上的RequestMapping和方法上的RequestMapping决定前台方法访问当前controller方法的请求位址。原则上，每个view对应一个controller，本view中的所有按钮和方法对应的controller，都需要是本页面对应的controller。

如BrPc089mView1Controller.java类别的RequestMapping如下：

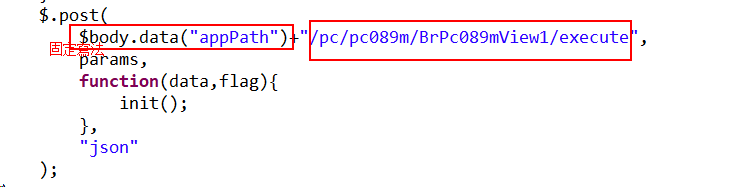


BrPc089mView1Controller.java中execute方法的RequestMapping如下：method=RequestMethod.POST。代表，本方法只能用post请求访问。

method=RequestMethod.GET，需要使用get请求访问。在7.3中分别说明这两种请求。Get请求一般用于按钮跳页使用，参见第14章内容。



则js访问此controller的execute方法的方式爲：



如果是点击按钮进行controller方法请求，还需要遵循四点一致原则：



* + 1. 使用$.post方式提交

这种是post请求：

|  |
| --- |
| $.post(url, params, function(data, flag) {  submitCallback(data, flag, $rolePanel);  $roleGrid.LisGrid("refresh");  }, "json"); |

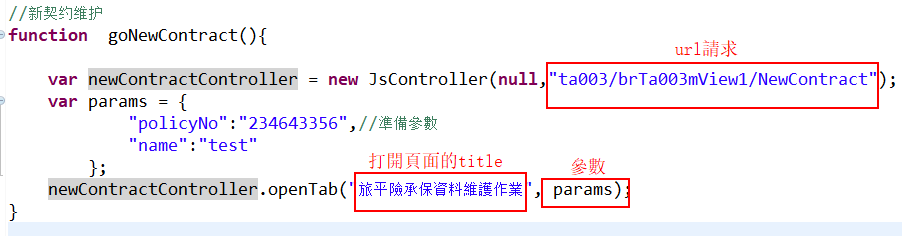
其中，url是要提交请求的url，params是以键值对保存的参数列表。写法如下：

|  |
| --- |
| var params = {  oOperator:activity,  PrintID:$("#PrintID").val(),//得到id爲PrintID控制项的value下同  PrintName:$("#PrintName").val(),  PrintObject:$("#PrintObject").val(),  PrintType:$("#PrintType").val(),  LanguageType:$("#LanguageType").val()  }; |

* + 1. GET请求

Get请求一般用于点击按钮进行跳页的情况。

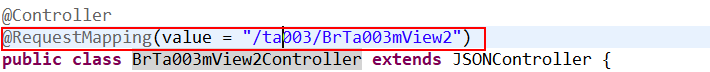
Js中方法如下所示：



Controller中的操作：



要跳转页面的controller：



* 1. 民国年转换
     1. Js中进行转换

公元年转民国年：year2tw(“2017-11-14”);结果：1061114

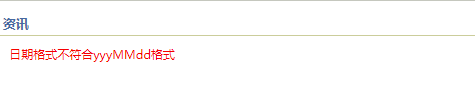
* + 1. Controller档中进行的转换：

民国年转公元年：

String date = DateUtil.getSqlDate(“1061114”); 结果：2017-11-14

* 1. 页面上显示提示信息
     1. showMessage使用

1、如果前台校验需要出现下图所示功能：



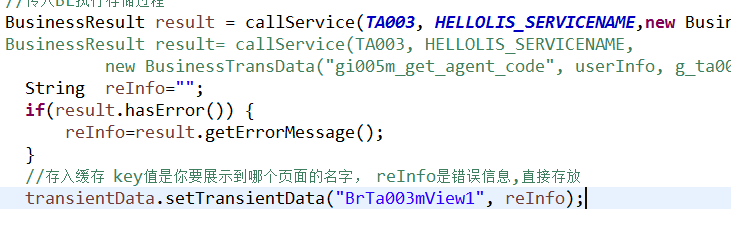
则在js相应方法内使用showMessage(messagetype,message)方法传入参数第一个是信息类型有Error或者Info按照自己的需求写自己需要类型，第二个是错误提示信息。

2、如果是alert弹框提示，就直接获得错误提示信息，弹出提示信息就行了。即：alert(“info”);

* + 1. doMessageGo使用

这个信息是后台传出的错误信息。

在controller档中添加如下代码：

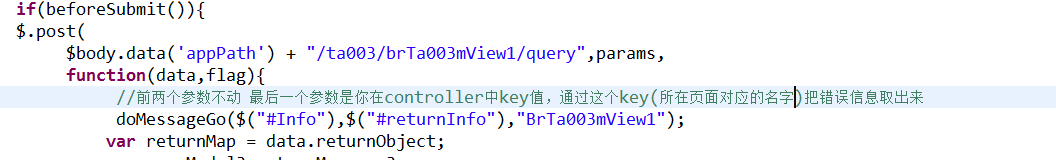


前台js调用:

1、首先可能是进入页面直接会调用。

2、点击某个按钮之后调用。（BrTa003mView1爲某个页面的名称，即BrTa003mView1.jsp

去掉.jsp后的部分。）

doMessageGo($("#Info"),$("#returnInfo"),"BrTa003mView1");

1. JSON配置规范
   1. Json档新建范例

JSON档中包含页面的清单和校验的资料，将在花括号{}中被配置也可以什么都不配置只写{}。花括号中的语法规则，遵循json的书写方式。

## LisGrid的设定档

以ContractApply.json为例。

内容如下

|  |
| --- |
| "grids" : [//grids阵列开始  {//第一个grid配置内容开始  "id" : "agentImpartList",//页面中LisGrid的id  "options" : {//LisGrid的初始化配置  "title" : '业务员告知 Agent Impart',//表头标题，支援多语言  "params" : {//该参数用于获取表中资料，包括sql获取和model获取  model : "policy.customerImparts(customerNoType=0)"  },  "selectType" : 0,//每行前是否有单选框或多选框  "selectFunc" : function(rowData) {//单选或多选触发的事件  },  "editType" : "multLine",//是否可编辑，编辑风格  "cols" : [ //表的每列的初始化内容，阵列，代表每列顺序。  {"sTitle" : "告知版别",//列标题，可多语言化  "mData" : "impartVer",//列资料的名称，对应sql栏位元元元元或者模型  "sClass" : "center",//列的展示格式  "mRule":{//列的校验规则  "required":true,//必填  "maxlength":6//最大长度  }},  /\*\*其他相似内容略\*\*/  ]  }  },  /\*\*其他相似LisGrid内容略\*\*/  ],//grids阵列结束 |

在LisGrid的配置内容中，除了id属性以外只有一个名为options的属性。options属性是用于配置LisGrid的所有初始化功能，如表头，对应的资料资源，各列的属性，是否显示单选或多选按钮（以及选中事件），是否可以编辑，初始化完成后执行函数等。下面说明各个可以配置的功能。

### 表头



图 41 表头

|  |
| --- |
| "options" : {//LisGrid的初始化配置  "title" : '业务员告知 Agent Impart'} |

* + 1. Model

配置Model资料资源的方法如下：

"options" : {//LisGrid的初始化配置

"params" : {//该参数用于获取表中资料，通过 model获取

model : "policy"

},

只有一个属性：model。其中policy代表模型物件的名称。在js中进行model的初始化，即可初始化列表。具体方式请参照12章。

### 单选和多选

配置单选或者多选的方式如下：

其中selectType的值有三种取值0，1，2。

0代表没有单选和多选的列。如图：



图 43 无单选多选列

1代表单选列。如图：



图 44 单选列

2代表多选列。如图：



图 45 多选列

SelectFunc是点击选中一行时回应的函数，该函数只有一个参数，行资料rowData。如上例中的配置，点击行时，会alert客户号码（insuredNo）。如图。

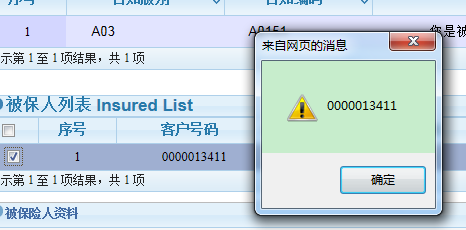


图 46 选中事件

### 各列的属性

ContractApply.json设定档如下：

|  |
| --- |
| "options" : {//LisGrid的初始化配置  "cols" : [ //表的每列的初始化内容，阵列，代表每列顺序。  {"sTitle" : "告知版别",//列标题，可多语言化  "mData" : "impartVer",//列资料的名称，对应模型  "sClass" : "center",//列格式，如果本列值爲"sClass" : "center notVisible",列隐藏  "mRule":{//列的校验规则  "required":true,//必填  "maxlength":6//最大长度  }}, |

cols属性是一个物件阵列，每一个物件代表一个列的配置信息，包括列标题，列资料名称，列的展示格式，列的校验规则。

预设第一列是单选或多选按钮（如果有的话），然后是序号，接下去从左往右将逐个排列设定档中的列。其中sTitle表示列的标题（可以多语言化），mData表示列的资料名称，sClass表示列的展示格式，mRule表示校验规则。

mData的大小写：mData的值必须与资料模型的属性名称一致。如：”mData”:”impartVer”与policy.customerImparts.impartVer的impartVer必须相等，大小写也一致。

sClass目前用于使列内容居中，如”sClass”:”center”。

mRule是一个物件，代表这列要校验的规则。

|  |
| --- |
| “mRule”:{  “required”:true,//表示内容不能为空，  “minlength”:10,//表示内容最小长度  “maxlength”:10,//表示内容最大长度  “inputTwice”:true,//表示要求手动输入两次相同内容（如身份证号）  “email”:true,//表示内容要符合email格式  “url”:true,//表示内容要符合url格式  “date”:true,//表示内容要符合date格式  “number”:true,//表示内容要为数位（整数或者小数）  “digits”:true,//表示内容要为整数  “min”:1,//表示内容最小值为1，如月份  “max”:12,//表示内容最大值为12，如月份  “checkFunc”：function(){},//使用自订函数来判断是否符合校验规则，传递两个参数：内容和当前校验的物件  “grid”：true//表示当前物件是一个LisGrid物件，要对其清单进行校验。  } |

各个规则之间是and的关系，就是只要有一个规则不通过，该列的校验就不通过。提供了常见的required，min，max等校验，也提供了code，grid等和LisCode元件，LisGrid元件相关的校验，还提供了一个自订校验的界面checkFunc，这里的所有规则和LisValidation一致，详细内容可以看LisValidation的说明。

### 清单的样式

清单支援多样式展现。包括下拉样式，输入框与按钮样式，时间选择样式和普通编辑方块样式。

样式需要在cols属性里添加。

|  |
| --- |
| cols : [  {"sTitle" : "保障号码",  "mData" : "COVERAGENO",  "sClass" : "center"},  {"sTitle" : "保险日数",  "mData" : "ISSUEDDAYS",  "sClass" : "center"}  ] |

1. 下拉样式

在以上的cols中添加属性

|  |
| --- |
| {"sTitle" : "性别",  "mData" : "SEX",  "sClass" : "center",  "codeType" : "sex"  }  //"codeType" : "sex"既可以是本列变爲下拉清单，下边只介绍需要添加的属性 |

1. 时间样式

|  |
| --- |
| "tDate" : "true" |

1. 按钮样式

|  |
| --- |
| "tButton" : "true" |

### 初始化完成事件

在LisGrid初始化结束后可以调用一个指定函数，配置如下：

|  |
| --- |
| "options" : {//LisGrid的初始化配置  "initedFunc" : function(grid){//是否可编辑，编辑风格  $(grid).LisGrid(“getSelected”,0);  }} |

initedFunc表示当LisGrid初始化完成后立刻执行该函数，参数是初始化好的LisGrid物件（HTML类型物件，没有用$包装）。

如例子中，初始化完成后，清单将选中第一行资料。

### API

$('#agentImpartList').LisGrid(“query”);//重新载入表格资料

$('#agentImpartList').LisGrid(“getSeleted”);//获得选中行的内容

$('#agentImpartList').LisGrid(“getSeleted”,0);//选中第1行，并返回内容

$('#agentImpartList').LisGrid(“refresh”);//重新查询表格资料，当使用sqlinfo

做为资料来源时，会重新查询后台。

$('#agentImpartList').LisGrid(“initedFunc”);//返回初始化函数

$('#agentImpartList').LisGrid(“getAllRowData”);//返回所有行的资料

### 选中一行事件

|  |
| --- |
| "options" : {//LisGrid的初始化配置  "selectType" : 0,//每行前是否有单选框或多选框  "selectFunc" : function(rowData) {//选中单选和多选后出发事件  alert(rowData.insuredNo);  }  } |

如上图，选中一行后会执行selectFunc属性配置的方法，参数爲选中的这一行的资料。如图，rowData.insuredNo即爲这行中某一列的值。

* 1. 校验规则配置

以ContractApply.json为例。

内容如下

|  |
| --- |
| "validation" : {//validation物件开始  "pageUrl":"apply/contractApply",//校验页面的url  "rules":[//校验规则阵列  {//校验规则物件，包括物件id、规则和对应的post动作  "fieldName":"agentImpartList",//要校验的页面物件id  "grid":true,//进行grid校验  "forRoles":["addAppnt","modifyAppnt"]},//执行校验的post动作名称  {"fieldName":"PolAppntDate",  "required":true,//进行非空校验  "date":true,//进行日期校验  "checkFunc":polAppntDate,//进行自订函数的校验  "forRoles":["addAppnt","modifyAppnt"]},  {"fieldName":"ManageCom","required":true,"forRoles":["addAppnt","modifyAppnt"]},  /\*\*其他rules略\*\*/  ]  }//validation对象结束 |

LisValidation的配置是在JSON档中的。JSON档被JS读取成物件后，其属性validation就是LisValidation的配置。下面说明validation的各个属性。

* + 1. pageUrl

表示当前页面的url，暂时没有用于程序逻辑处理。

* + 1. rules

rules的配置如下：

|  |
| --- |
| "rules":[//校验规则阵列  {//校验规则物件，包括物件id、规则和对应的post动作  "fieldName":"agentImpartList",//要校验的页面物件id  "grid":true,//进行grid校验  {"fieldName":"ManageCom","required":true}  ] |

Rules是一个rule的阵列，每一个rule包含2种属性，一个是用于确定校验物件的

（fieldName），剩余的则是用于校验的规则（如required,grid等）。

fieldName是校验对象的id。

其他的用于校验的规则有：

“required”:true,//表示物件的内容不能为空，

“minlength”:10,//表示内容最小长度

“maxlength”:10,//表示内容最大长度

“inputTwice”:true,//表示要求手动输入两次相同内容（如身份证号）

“email”:true,//表示内容要符合email格式

“url”:true,//表示内容要符合url格式

“date”:true,//表示内容要符合date格式

“number”:true,//表示内容要为数位（整数或者小数）

“digits”:true,//表示内容要为整数

“min”:1,//表示内容最小值为1，如月份

“max”:12,//表示内容最大值为12，如月份

“checkFunc”：function(){},//使用自订函数来判断是否符合校验规则，传递两个参数：内容和当前校验的物件

各个规则之间是and的关系，就是只要有一个规则不通过，该列的校验就不通过。

校验规则提供了常见的required，min，max等校验，也提供了code，grid等和LisCode元件，LisGrid元件相关的校验，还提供了一个自订校验的界面checkFunc。

按照以上的分类结合上例进行说明。

* + 1. 常见的校验

|  |
| --- |
| {"fieldName":"ManageCom",  "required":true,  "forRoles":["addAppnt","modifyAppnt"]}, |

该配置的说明：id为ManageCom的输入框是校验物件，校验规则是required，即非空，校验的post动作名称是addAppnt，modiftyAppnt。就是post这两个对应名称的url之前，要对ManageCom进行非空校验。

如果要加上日期校验，可以这样写：

|  |
| --- |
| {"fieldName":"ManageCom",  "required":true,  "date":true  }, |

这样就可以在校验非空之后，校验日期格式是否正确了。同样的要进行最大最小值，最大最小长度，email，url，number，digits等校验只要直接添加就可以了。

* + 1. 与其他组件相关的校验

|  |
| --- |
| {//校验规则物件，包括物件id、规则和对应的post动作 "fieldName":"agentImpartList",//要校验的页面物件id  "grid":true,//进行grid校验 |

该配置的说明：校验物件是id为agentImaprtList的物件，校验规则是grid，即对LisGrid物件校验，校验的post动作名称是addAppnt，modiftyAppnt。该规则表示post这两个对应名称的url之前，要对id为agentImaprtList的物件进行grid校验。

Grid校验是针对LisGrid物件中各个储存格进行校验，储存格的规则在LisGrid的配置中写明。在LisGrid每列属性中（该属性在grids:options:cols）mRule代表列的规则校验，如

|  |
| --- |
| "options" : {//LisGrid的初始化配置  "cols" : [ //表的每列的初始化内容，阵列，代表每列顺序。  {"sTitle" : "告知版别",//列标题，可多语言化  "mData" : "impartVer",//列资料的名称，对应sql栏位元元元或者模型  "sClass" : "center",//列的展示格式  "mRule":{//列的校验规则  "required":true,//不能为空  "maxlength":6//最大长度  }}, |

mRule包含的属性是该列要进行校验的规则，该规则写法和LisValidation的设定档的写法一致（此处不需要写fieldName和forRoles）。

上面的例子包括了两个部分，一个是LisGrid中的列属性要有校验规则（mRule），另一个是LisValidation的rules属性中要有fieldName为grid物件id的规则，且校验规则必须是grid:true。以上例来看，当post动作为addAppnt或者modifyAppnt时，id为appntImpartList的LisGrid将执行校验，执行的校验规则是grid。该LisGrid执行grid校验，该规则将会遍历LisGrid的储存格，将有mRule属性的列进行储存格校验。从上例可以看到，列标题为“告知版别”的储存格都将会进行非空校验（required）和最大文本长度校验（maxlength）。

除了与LisGrid相关的校验规则外，还有与LisCode相关的校验规则：code。

注：Code还没有实现。

* + 1. 自订校验

|  |
| --- |
| { "fieldName":"PolAppntDate",  "required":true,//进行非空校验  "date":true,//进行日期校验  "checkFunc":polAppntDate,//进行自订函数的校验  "forRoles":["addAppnt","modifyAppnt"]},  自订规则的名称是checkFunc，就是检查函数。在本例中，其对应的polAppntDate应该在ContractApply.js档中实现。如下：  // Apply date cann't be future  var polAppntDate = function(value, $element) {  $element.data("checkFuncTip", "Apply date cann't be future");  if (new Date(value) > new Date())// value must be a valid date  return false;  return true;  }; |

这样，当post动作为addAppnt或者modifyAppnt时，校验id为PolAppntDate的文字方块。先校验非空规则和日期格式规则，然后校验自订规则checkFunc。checkFunc属性的值是polAppntDate，这是一个全域的变量，而且是个函数（在ContractApply.js中）。

该规则polAppntDate要做两个事情，一个是给$element加上checkFuncTip的资料，用于显示提示信息，一个是返回一个boolean值，true表示通过校验，false表示不通过校验。这个函数有两个参数，一个是待校验的值（一般来说就是文字方块的值），一个是校验物件（已用$符号包装过）。

这里的规则是该值（申请投保日期）不能晚于今日，如果出错则如下图提示：



图 49 自订校验

如果有其他复杂逻辑，比如证件号码的校验：当证件类型为无证件时，不用校验，其他证件类型都要进行非空校验和重复录入校验。截图和代码如下：





图 50 复杂的自订校验

其中9代表无证件，1代表有证件。

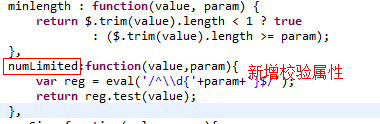
|  |
| --- |
| // When AppntIDType is difference, appntIDNo validation is difference  var appntIDNo = function(value, $element) {  $element.data("checkFuncTip", null); //提示信息要清空或者覆盖，不然上次检验信息会残留  var idType = $("#AppntIDType").val();  var flag = true;  if (idType != "9") {//有证件的时候要校验  var temp = $element.LisValidation("method", "required");//非空校验，不通过的话，会自动添加提示信息  flag = flag & temp;  temp = $element.LisValidation("method", "inputTwice");//重复输入校验  flag = flag & temp;  }  return flag;  }; |

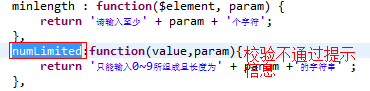
在这里要进行非空校验，可以直接使用LisValidation提供的功能来校验，同时该功能会完成提示信息的生成，同样的进行重复输入校验也可以这样完成。

* + 1. 添加自订校验属性

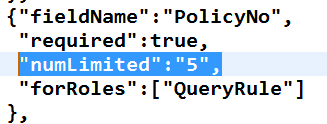
为了使lis7系统能更好的符合uix系统的校验规则,lis7需要添加校验规则

比如input框新增numLimited(只能输入0~9所组成且长度为的字符串)规则,那么只需在jQuery.LisValidation.js中新增校验规则即可





自订校验标签使用:





注：在校验前把提示信息清空，避免内容不断迭加。

1. controller文件规范
   1. 写法规范

Controller用于接收页面上所有的请求，常见的有Post,Get请求。用于返回页面url和回应post请求。

请求页面的两个重要的功能：对应的url和动作（Post,Get），获取参数处理请求（函数体）。

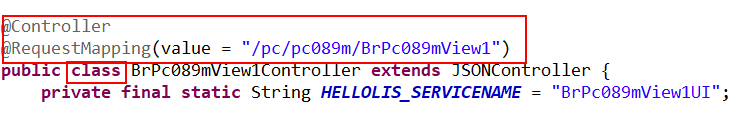
新建立的controller档案需要粘贴如下代码，并需要修改类名和返回的页面url的名称：

|  |
| --- |
| @Controller //要添加@Controller注解  @RequestMapping(value = "/pc/pc089m/BrPc089mView2")//对应的映射  public class BrPc089mView2Controller extends JSONController {  //把类别名改成自己新建的类别名  //新建一个RequestMapping爲空，请求方式爲GET的方法，这样当请求  //爲类别上的RequestMapping时，会自动执行本方法，进行页面访问  @RequestMapping(value = "", method = RequestMethod.GET)  @ControllerMethod("goPc089mView2")  public String gotoIndex(ServletRequest request) {  //返回页面的路径，会自动将下方url中自动拼接爲成一个jsp档 //的完整路径，达到访问页面的目的  return "pc/pc089m/BrPc089mView2";  }  } |

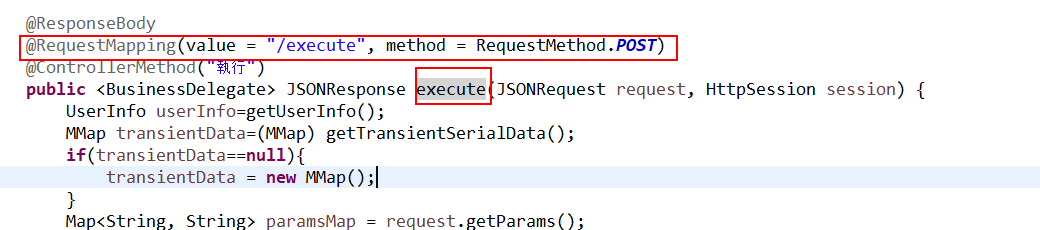
* 1. 请求与后台访问

在准备完页面的数据之后，要将页面上的数据传递到controller中，（controller在页面设计规范及标准白皮书中有详细介绍），controller中类别上的RequestMapping和方法上的RequestMapping决定前台方法访问当前controller方法的请求位址。

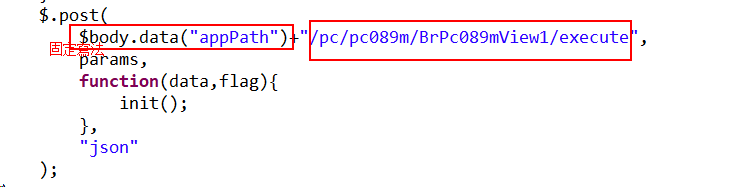
如BrPc089mView1Controller.java类别的RequestMapping如下：



BrPc089mView1Controller.java中execute方法的RequestMapping如下：

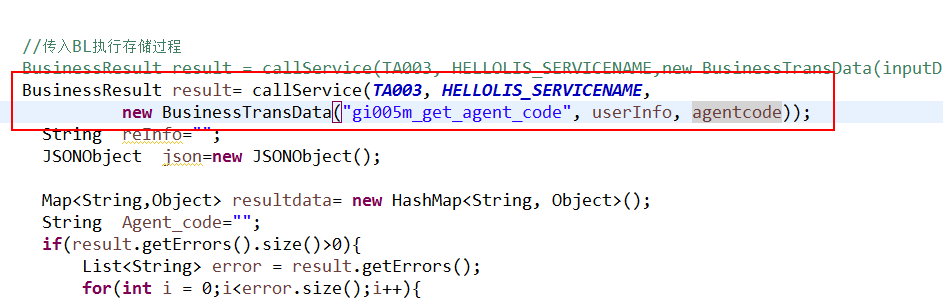


则js访问此controller的execute方法的方式爲：



并在controller进行相应的封装，具体是说有的值需要放入到type实体类别中。

备准好数据之后，执行callService的方法，如下：



callService()方法有3个参数，第一个参数，是所属模块，即TA003；第二个参数，即后台UI即service的名称，如上图：HELLOLIS\_SERVICENAME，爲存储的后台UI即service的名称；第三个参数是BusinessTransData类别，内部封装了两个固定参数，和多个不固定参数：两个固定参数分别是要执行的BL中的方法和UserInfo信息。多个不固定参数是要执行存储过程的参数。

BusinessResult类别是后台逻辑执行后，传回来的返回值。

返回值是一个map，

Map<String,Object> map =result.getReturnData();

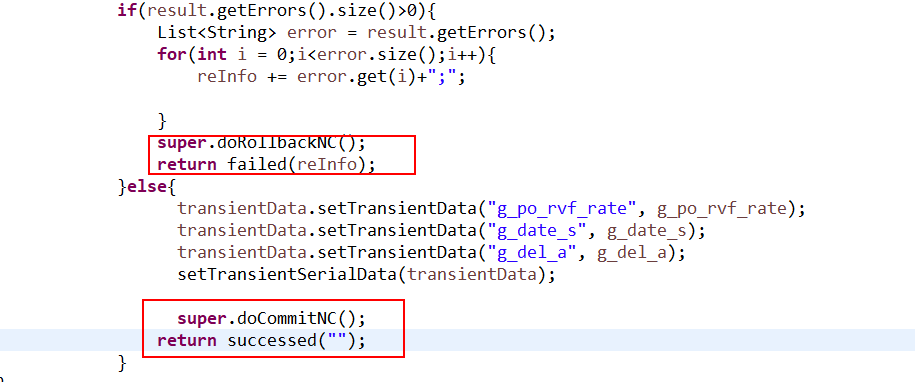
后续详细调用后台的逻辑，请参照程序开发规范及标准白皮书.docx的第5章内容的

controller部分。

* 1. 响应

在controller获得后台返回的处理结果之后，根据13章介绍的页面之间传递至的方式进行资料存储，把得到的资料传往js，并在js中根据13章所说明的方法得到controller中的资料。

最终，在controller中解决完所有逻辑之后，根据有没有错误，调用相应的方法，并return。如下所示：



1. 调用后台方法规范

在页面需要将数据传往后台，从而调用相应存储过程时，需要对后台进行逻辑调用。方法如下：

* 1. 页面数据准备

在页面上进行相应数据的填写，并且点击相应的按钮，通过onclick或者ondblclick方法访问js中的某个方法。

* 1. 访问js方法

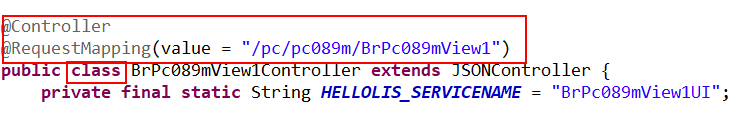
点击按钮进入到js的某方法之后，在js的方法中进行数据的准备，类似的方法如下所示：

|  |
| --- |
| function Execute(){  var params = {  g\_tran\_date\_str:$("#g\_tran\_date\_str").val(),  g\_tran\_date\_end:$("#g\_tran\_date\_end").val()  };  //调用controller的方式有三种，都可以使用，在第7.3中有详细介绍  $.post(  $body.data("appPath")+"/pc/pc089m/BrPc089mView4/button\_1",  params,  function(data,flag){  var returnMap = data.returnObject;  //调用controller后，返回结果的处理  },  "json"  );  } |

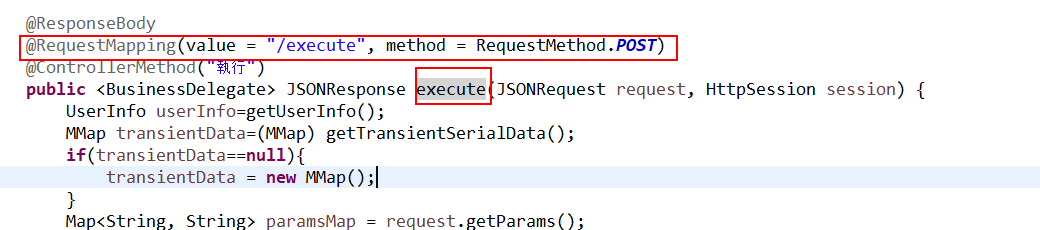
* 1. Controller档中的操作

在准备完页面的数据之后，要将页面上的数据传递到controller中，（controller在页面设计规范及标准白皮书中有详细介绍），controller中类别上的RequestMapping和方法上的RequestMapping决定前台方法访问当前controller方法的请求位址。

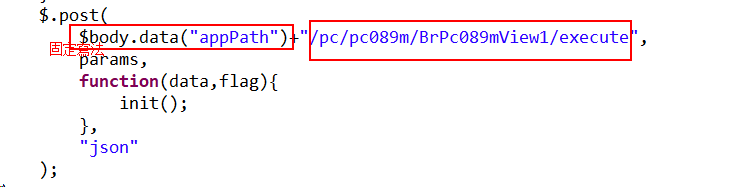
如BrPc089mView1Controller.java类别的RequestMapping如下：



BrPc089mView1Controller.java中execute方法的RequestMapping如下：



则js访问此controller的execute方法的方式爲：



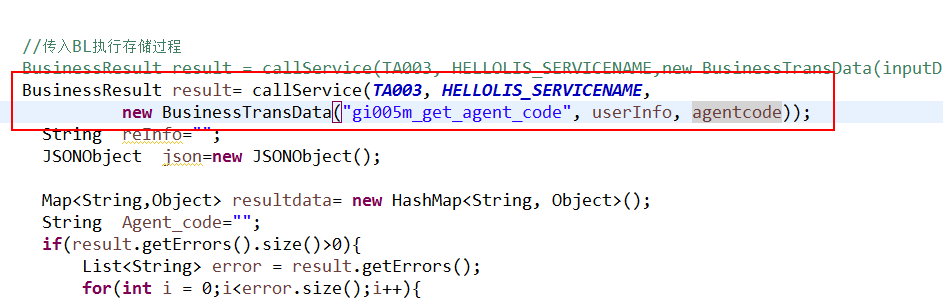
通过10.2传过来的js的数据是一个map，获得的方式爲：

Map<String, String> data=request.getParams();

controller中得到map，并将map中的数据拿出，进行相应的封装，具体是说有的值需要

放入到type实体类别中。

备准好数据之后，执行callService的方法，如下：



callService()方法有3个参数，第一个参数，是所属模块，即TA003；第二个参数，即后台UI即service的名称，如上图：HELLOLIS\_SERVICENAME；第三个参数是BusinessTransData类别，内部封装了两个固定参数，和多个不固定参数：两个固定参数分别是要执行的BL中的方法和UserInfo信息。多个不固定参数是要执行存储过程的参数。

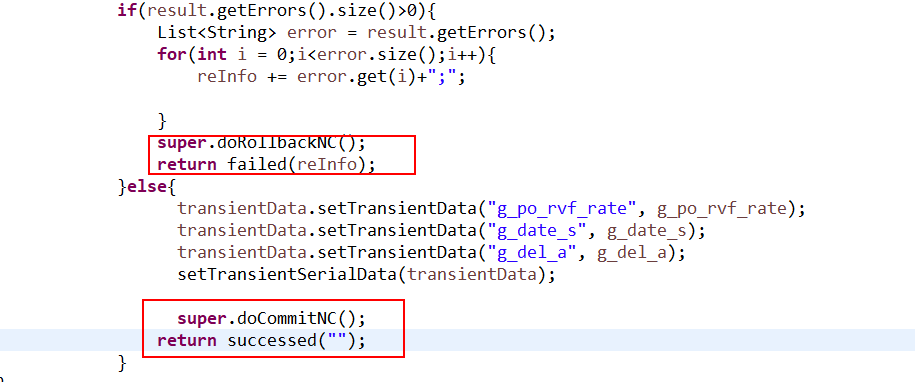
BusinessResult类别是后台逻辑执行后，传回来的返回值。

返回值是一个map，

Map<String,Object> map =result.getReturnData();

后续详细调用后台的逻辑，请参照程序开发规范及标准白皮书.docx的第5章内容的controller部分。

最终，在controller中解决完所有逻辑之后，根据有没有错误，调用相应的方法，并return。如下所示：



1. 权限动态配置

根据四点一致原则



按钮中的权限只需要配置成图中所示的一致原则，即jsp中按钮url属性中，#号后边的term\_id，数据库中guidt资料表中存在的term\_id,js中请求方法的requestMapping（7.3节有说明）和controller设定的requestMapping（9.2.1节有说明）一致即可。

但是触发访问后台动作的不是按钮，或者虽然是按钮，但是没有在数据库中配置，需要动态赋予用户权限的，则需要在控制项中添加controller属性。

比如，如果需要动态添加/pc/pc089m/BrPcO89mView1/addSomething权限，则需要在某个控制项中，比如说访问后台的动作是一个输入框触发的，则需要对输入框添加controller属性，值爲/pc/pc089m/BrPcO89mView1/addSomething，即

<L:input id=’test’ onclick=’doSomething’ prelabel =’测试控制项’

controller=’ /pc/pc089m/BrPcO89mView1/addSomething’/>

这样，在页面初始化时，会自动将/pc/pc089m/BrPcO89mView1/addSomething权限添加到用户权限集合中。

1. 清单查询书写方式

LISGrid清单的初始化，需要使用sql进行查询资料的，使用如下方法：

* 1. 语法

1、在json档中配置清单的信息，上文已经介绍，重点部分爲加颜色的部分：配置一个model来初始化列表。

|  |
| --- |
| "grids" : [//grids阵列开始  {//第一个grid配置内容开始  "id" : "policyList",//页面中LisGrid的id  "options" : {//LisGrid的初始化配置  "title" : '业务员告知 Agent Impart',//表头标题  "params" : {//该参数用于获取表中资料，包括sql获取和model获取  model : "brpc235model"  },  "selectType" : 0,//每行前是否有单选框或多选框  "selectFunc" : function(rowData) {//单选或多选触发的事件  },  "editType" : "myEditRow",//是否可编辑，编辑风格  "cols" : [ //表的每列的初始化内容，阵列，代表每列顺序。  {"sTitle" : "告知版别",//列标题，可多语言化  "mData" : " IMPARTVER",//列资料的名称，对应sql栏位元元元或者模型  "sClass" : "center",//列的展示格式  "mRule":{//列的校验规则  "required":true,//必填  "maxlength":6//最大长度  }},  /\*\*其他相似内容略\*\*/  ]  }  },  /\*\*其他相似LisGrid内容略\*\*/  ],//grids阵列结束 |

3、controller中的书写方法

在controller中需要在方法体中通过callService来调用后台查询逻辑，如下图所示：



Js档中的处理

Controller返回list之后，在js中进行如下代码操作：



上图中，给返回值添加的key值要跟json清单中配置的model值相同。

* 1. 说明

以上语句，在json中配置的列需要和查询语句中查询的栏位个数一致,并且，在json中列的名字需要都是大写，即"mData" : "IMPARTVER"。

1. Nt\_rec与列表资料转换
   1. 使用Nt\_rec初始化页面列表

在controller中，得到执行BL逻辑（参照《程序开发规范及标准白皮书》）返回的Nt\_rec的值以后，执行如下图所示操作：

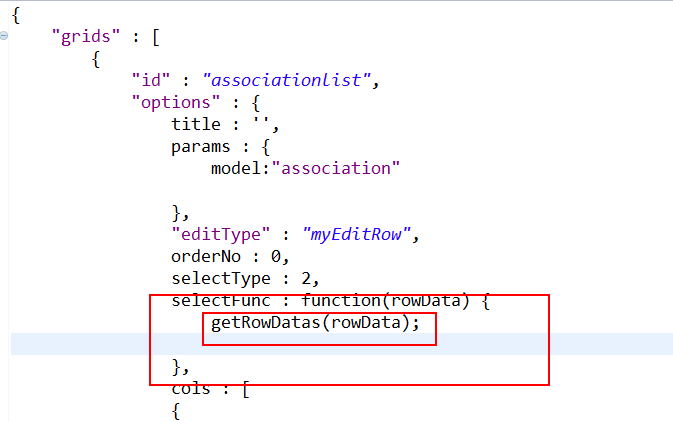


在js中执行类似12.1中的js中的处理。

* 1. 封装选中的列表中的数据爲Nt\_rec
     1. Json中的配置

在json中添加选中事件，如下图（具体json配置的说明，参照第8章）：

调用getRowDatas(rowData),该方法在js中进行定义。



* + 1. Js中的代码

在js中定义全域变量 var rowDatas;

选中的列表数据会存放到rowDatas中，在请求controller时，将rowDatas封装到传往controller的参数中，如下所示：

|  |
| --- |
| **function** getRowDatas(data){  rowDatas=data;  }  **function** lisTest(){  //准备传往controller的数据  **var** params={  "gridData":rowDatas  };  $.post(  $body.data('appPath') + "/ta003/brTa003mView1/lisTest",params,  **function**(data,flag){  doMessageGo($("#Info"),$("#returnInfo"),"brTa003mView1");  },  "json"  );  } |

* + 1. Controller档中的操作

|  |
| --- |
| @RequestMapping(value = "/lisTest", method = RequestMethod.***GET***)  @ControllerMethod("")  **public** String lisTest(JSONRequest request) **throws** SQLException {    UserInfo userInfo=getUserInfo();    Map<String, String> paramsMap = request.getParams();  //以下方法爲封装得到的gridData  JSONArray jsa = **new** JSONArray();  jsa=JSONArray.*fromObject*(paramsMap.get("gridData"));//封装爲JSONArray    Nt\_rec\_ta003m4a Nt\_rec=**new** Nt\_rec\_ta003m4a();  //循环JSONArray放到实体类别rec,进一步放到nt\_rec中  **for** (**int** i = 0; i < jsa.size(); i++) {  JSONObject jsonOject=jsa.getJSONObject(i);  Rec\_ta003m4a rt = (Rec\_ta003m4a)JSONUtil.*jSONObject2Rec*(jsonOject, Rec\_ta003m4a.**class**);  Nt\_rec.add(rt);  }  /\*  \*其他逻辑操作  \*/  **return** "ta/ta003/BrTa003mView2";  } |

1. 页面之间的值传递
   1. 功能目的

页面有时有跳转的操作。在本页面跳转下一个页面，有时还需要跳转回来操作，此时需要页面上已经填写的值还存在，所以，需要在页面进行跳转的时候，将页面上输入或者查出的值暂时存储起来，供页面初始化赋值使用。

* 1. 功能实现
     1. 储存值

在页面输入的值或者是从数据库查出的值，需要暂存起来时，在页面相对应的controller档里，透过以下语句将值存下：

|  |
| --- |
| MMap transientData=new MMap();//数据存放map  //根据key-value存放数据到MMap中  transientData.setTransientData("p\_liaroc\_qry", p\_liaroc\_qry);  // g\_ta003m8a对应数据库中的type类型  transientData.setTransientData("g\_ta003m8a", g\_ta003m8a);  ………//其他set操作  //将MMap以sessionid+usercode作爲key存入到缓存中  setTransientSerialData(transientData); |

* + 1. 取值

在js中通过指定写法可以取出值，用来初始化页面时给页面上的栏位赋值。书写方式如下：

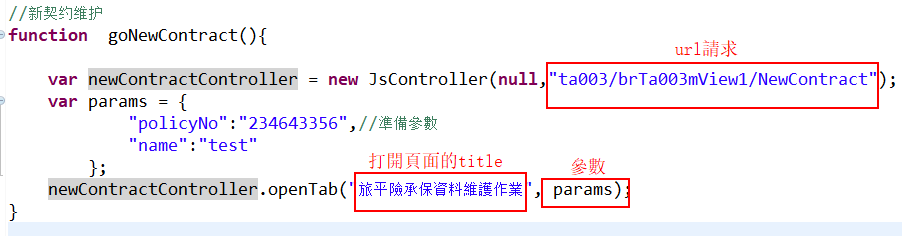
|  |
| --- |
| var p\_liaroc\_qry =getTransientData("p\_liaroc\_qry ");  $("#p\_liaroc\_qry").val(p\_liaroc\_qry); $("#EntryDateS").val(year2tw(g\_ta003m8a.entry\_date\_s));  $("#EntryDateE").val(year2tw(g\_ta003m8a.entry\_date\_e));  //year2tw把公元年转换爲民国年 |

1. 通过点击按钮跳转页面

通过点击按钮来访问js中的某个方法：

在js方法中使用get方法进行对controller中方法的请求。

Js中方法如下所示：



Controller中的方法：



1. commit和rollback

在执行存储过程需要进行commit和rollback操作时，需要在controller中进行相应的方法调用。提供清空页面间缓存的方式和不清空页面间缓存的方式。

方式分别如下：

|  |  |
| --- | --- |
| super.doCommitNC(); | 提交数据库不清空缓存 |
| super.doRollbackNC(); | 回滚数据库不清空缓存 |
| super.doCommit(); | 提交数据库并清空缓存 |
| super.doRollback(); | 回滚数据库并清空缓存 |