

DWZ 富客户端框架使用手册

目录

- 概述5
 - 设计思路.....5
 - 学习 DWZ 的建议.....5
 - DWZ 区别于其它 JS 框架，最大的优点5
 - 版权声明.....6
 - DWZ 团队介绍.....6
- HTML 扩展7
 - Ajax 链接扩展.....7
 - 当前 navTab 中链接 ajax post 扩展.....7
 - dialog 链接扩展.....7
 - navTab 链接扩展.....8
 - Tab 组件扩展.....9
 - Accordion 组件9
 - 容器高度自适应.....9
 - CSS Table10
 - Table 扩展.....10
 - 在线编辑器.....11
 - 分页组件.....11
 - ajaxTodo 扩展.....12
 - dwzExport 列表数据导出.....12
 - Input alt 扩展.....12
 - Tree 扩展13
 - Panel 扩展.....13
 - 日历控件.....13
 - url 变量替换16
 - checkbox 全选、反选.....17
 - uploadify 多文件上传.....17
 - combox 组件.....18
 - suggest+lookup+主从结构19
 - 查找带回.....19

主从结构.....	20
Ajax 表单.....	20
表单查询.....	21
普通 Ajax 表单提交.....	22
服务器端响应.....	23
文件上传表单提交.....	24
服务器端响应.....	24
Java 服务器端表单处理示例	25
DWZ js 库介绍	27
DWZ 框架初始化.....	27
dwz.core.js	27
dwz.ui.js	27
dwz.ajax.js	27
dwz.alertMsg.js	27
dwz.jDialog.js	27
dwz.accordion.js.....	27
dwz.barDrag.js	27
dwz.navTab.js.....	28
dwz.scrollCenter.js	28
dwz.stable.js.....	28
dwz.cssTable.js.....	28
dwz.tree.js.....	28
dwz.theme.js	28
dwz.validate.method.js	28
dwz.validate.zh.js	28
dwz.contextmenu.js.....	28
dwz.pagination.js	29
dwz.database.js.....	29
dwz.datepicker.js	29
dwz.combox.js.....	29
dwz.checkbox.js	29
dwz.utl.date.js	29
dwz.regional.zh.js.....	29
dwz.validate.method.js	29
Javascript 混淆和压缩.....	30
Javascript 混淆.....	30

Javascript 用 gzip 压缩	31
DWZ 如何中使用打包的 js	31
常见问题及解决.....	32
DWZ 中如何整合第三方 jQuery 插件	32
Error loading XML document: dwz.frag.xml.....	32
IIS 不能用 Ajax 访问*.html 后缀的页面.....	32
多个 navTab 页面或 dialog 页面 ID 冲突，解决方法	33
jQuery1.4.2 和 jquery.validate.js 在 IE 的兼容问题	33
升级 jQuery1.4.2 注意事项.....	33
weblogic 访问 xml 问题	33
如何自定义 DWZ 分页参数参数	34
如何关闭 loading.....	34
DWZ 局部刷新怎样做?.....	34
DWZ 版本升级.....	35
V1.4.7	35
V1.4.6	35
V1.4.5	35
V1.4.4	35
V1.4.3	35
V1.4.2	35
V1.4.1	35
V1.3 Final.....	36
V1.3 RC4	36
V1.3 RC3	36
V1.3 RC2	36
V1.3 RC1	37
V1.2 Final.....	38
V1.2 RC1	38
V1.1.6 Final.....	39
V1.1.6 RC3	39
V1.1.6 RC2	39
V1.1.6 RC1	40
V1.1.5 Final.....	40
V1.1.5 RC3	40
V1.1.5 RC2	40
V1.1.5 RC1	40

V1.1.5 Beta1	40
V1.1.4 Final.....	40
V1.1.3	41
V1.1.2	41
V1.1.1	41
v1.1.0.....	41
v1.0.6.....	42
v1.0.5.....	42
附录	43
附录一 Firebug 介绍.....	43
补充说明和常见问题(xiaosuiba)	44

概述

DWZ 富客户端框架(jQuery RIA framework), 是中国人自己开发的基于 jQuery 实现的 Ajax RIA 开源框架.

DWZ 富客户端框架设计目标是**简单实用、扩展方便、快速开发、RIA 思路、轻量级**

DWZ 框架支持用 html 扩展的方式来代替 javascript 代码, 只要懂 html 语法, 再参考 DWZ 使用手册就可以做 ajax 开发.

开发人员不写 javascript 的情况下, 也能用 ajax 做项目和使用各种 UI 组件. 基本可以保证程序员不懂 javascript, 也能使用各种页面组件和 ajax 技术. 如果有特定需求也可以扩展 DWZ 做定制化开发.

做 ajax 项目时需要写大量的 javascript 才能达到满意的效果. 国内很多程序员 javascript 不熟, 大大影响了开发速度. 使用 DWZ 框架自动绑定 javascript 效果. 不需要开发人员去关心 javascript 怎么写, 只要写标准 html 就可以了.

DWZ 简单扩展了 html 标准, 给 HTML 定义了一些特别的 **class** 和 **attribute**. DWZ 框架会找到当前请求结果中的那些特别的 **class** 和 **attribute**, 并自动关联上相应的 js 处理事件和效果.

DWZ 基于 jQuery 可以非常方便的定制特定需求的 UI 组件, 并以 jQuery 插件的形式发布出来. 如有需要也可做定制化开发.

欢迎大家提出建议, 我们将在下一版本中进一步调整和完善功能.

DWZ 富客户端框架是开源项目, 可以免费获取源码. 希望有多的开发人员使用, 共同推进国内整体 ajax 开发水平.

在线演示地址 <http://j-ui.com/>

在线文档 <http://j-ui.com//doc/dwz-user-guide.pdf>

Google Code 下载: <http://code.google.com/p/dwz/>

设计思路

第一次打开页面时载入界面到客户端, 之后和服务器的交互只是数据交互, 不占用界面相关的网络流量.

支持 HTML 扩展方式来调用 DWZ 组件.

标准化 Ajax 开发, 降低 Ajax 开发成本.

学习 DWZ 的建议

刚接触 DWZ 的人可能感觉 DWZ 文档太少、入门困难, 原因都是没有掌握正确的学习方法. 建议按下面的步骤来学习 DWZ 框架:

- 通读 DWZ 文档, 很多新手提的问题文档中都写了。
- 看 demo 每个组件演示效果和代码（留意组件 html 结构）。
- 建议安装 firebug, 用 firebug 看 html 结构、CSS 和调试 JS 都非常方便。见附录一 firebug 介绍。
- 对于初学者不建议看 DWZ 全部源码, 但还是非常有必要看看 **dwz.ui.js** 和 **dwz.ajax.js**
- 可以从 google code 下载 dwz_thinkphp 版本, 结合 php 后台去理解 DWZ 和服务端端的交互方式

DWZ 区别于其它 JS 框架, 最大的优点

- 完全开源, 源码没有做任何混淆处理, 方便扩展
- CSS 和 js 代码彻底分离, 修改样式方便
- 简单实用, 扩展方便, 轻量级框架, 快速开发

- 仍然保留了 html 的页面布局方式
- 支持 HTML 扩展方式调用 UI 组件，开发人员不需写 js
- 只要懂 html 语法不需精通 js，就可以使用 ajax 开发后台
- 基于 jQuery，UI 组件以 jQuery 插件的形式发布，扩展方便

版权声明

- DWZ 框架的源代码完全开放，在 Apache License 2.0 许可下，可免费应用于个人或商业目的。
- 欢迎各大网站转载下载版本。
- 禁止把 DWZ 框架包装成类外一个 UI 框架出售。

DWZ 团队介绍

DWZ 研发组开发人员目前是 3 人（兼职）

杜权从事 UI 设计工作，有 10 年以上 UI 设计经验。做过至少 1500 个网站的 UI 设计。

吴平主要做 Java web 开发，兼 ajax 开发。一直从事电子商务、企业建站平台开发工作。目前就职于支付宝应用架构师职位。

张慧华主要做 Java web 开发，兼 ajax 开发。以前也是电子商务、企业建站平台开发工作。从 2009 年 4 月开始从事建筑能效评估 IT 解决方案。目前从 Java 开发转型做 HTML5 手机 APP。

以前我们做的大部份 java 项目都用了 Ajax，项目开发过程中经常自己做一些 UI 组件和界面效果。我们对 RIA 非常感兴趣，业余时间就做了 DWZ 富客户端框架。DWZ 框架中的 UI 组件都是我们从做过的大量 web 项目中总结出来的，都是一些非常实用的 UI 组件。

联系方式

杜权 (UI 设计)	d@j-ui.com
吴平 (Ajax 开发)	w@j-ui.com
张慧华 (Ajax 开发)	z@j-ui.com

官方微博 (欢迎加入) <http://weibo.com/dwzui>

jQuery.DWZ-jUI-1 群 (满员) 107983317
jQuery.DWZ-jUI-2 群 (满员) 69611933
jQuery.DWZ-jUI-3 群 (满员) 20866231
jQuery.DWZ-jUI-4 群 (满员) 369203
jQuery.DWZ-jUI-5 群 (满员) 85031937
jQuery.DWZ-jUI-6 群 (欢迎加入) 172602882
jQuery.DWZ-jUI-7 群 (满员) 210322217
jQuery.DWZ-jUI-8 群 (欢迎加入) 139067378

合作电话：010-52897073

HTML 扩展

支持HTML 扩展方式来调用 DWZ 组件

Ajax 链接扩展

``

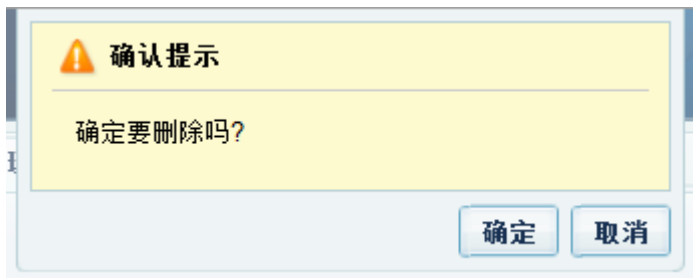
示例: `提示窗口`

当前 navTab 中链接 ajax post 扩展

`删除`

或

`删除`



Title 为可选项, 如果设置, 点击后将调用 `alertMsg.confirm` 与用户交互确认或取消, Title 值为提示信息. Target 值为 `ajaxTodo` 时会自动关联如下 JS.

```
$( "a[target=ajaxTodo]", $p ).each( function() {
    $( this ).click( function( event ) {
        var $this = $( this );
        var title = $this.attr( "title" );
        if ( title ) {
            alertMsg.confirm( title, {
                okCall: function() {
                    ajaxTodo( $this.attr( "href" ) );
                }
            } );
        } else {
            ajaxTodo( $this.attr( "href" ) );
        }
        event.preventDefault();
    } );
} );
```

dialog 链接扩展

``

A 所指向页面将会在 dialog 弹出层中打开, rel 标识此弹出层的 ID, rel 为可选项。

Html 标签扩展方式示例:

`弹出窗口`

或

```
<a href="demo_page1.html" target="dialog" [max=true, mask=true, maxable=true, minable=true,
resizable=true,drawable=true] rel="dlg_page1" title="[自定义标题]" width="800"
height="480">打开窗口一</a>
```

Max 属性表示此dialog打开时默认最大化, mask表示打开层后将背景遮盖. maxable: dialog 是否可最大化,

minable: dialog 是否可最小化,
mixable: dialog 是否可最大化
resizable: dialog 是否可变大小
drawable: dialog 是否可拖动
width: dialog 打开时的默认宽度
height: dialog 打开时默认的高度
width,height分别打开dialog时的宽度与高度.
fresh: 重复打开dialog时是否重新载入数据, 默认值true,
close: 关闭dialog时的监听函数, 需要有boolean类型的返回值,
param: close监听函数的参数列表, 以json格式表示, 例{msg:'message'}

关闭窗口:

在弹出窗口页面内放置<button class="close" value="关闭"></button>即可。

JS 调用方式示例:

```
$.pdialog.open(url, dlgId, title);
```

或

```
$.pdialog.open(url, dlgId, title, options);
```

options:{width:100px,height:100px,max:true,mask:true,mixable:true,minable:true,resizable:true,drawable:true,fresh:true,close:"function", param:"{msg:'message'}"}, 所有参数都是可选项。

关闭dialog层:

```
$.pdialog.close(dialog); 参数dialog可以是弹出层jQuery对象或者是打开dialog层时的dlgId.
```

或者

```
$.pdialog.closeCurrent(); 关闭当前活动层。
```

```
$.pdialog.reload(url, {data:{}, dialogId:"", callback:null})
```

刷新dialogId指定的dialog, url: 刷新时可重新指定加载数据的url, data: 为加载数据时所需的参数。

navTab 链接扩展

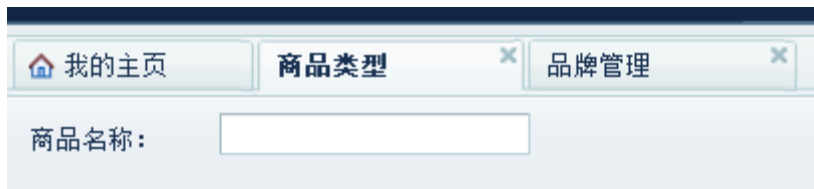
```
<a href="xxx" target="navTab" [rel="tabId"]>
```

示例:

```
<a href="url" target="navTab" >默认页面</a>
```

```
<a href="url" target="navTab" rel="page1" title="自定义标签名" fresh="false">自定义页面</a>
```

```
<a href="url" target="navTab" external="true">iframe 方式打开</a>
```

target=navTab: 自动关联调用 navTab 组件

rel: 为 navtab 的 ID 值, 后续可以用来重载该页面时使用, 如当前页新增或删除数据可以通过该 ID 进行通知 JS 重载。注意 rel 的值区分大小写。

fresh: 表示重复打开 navTab 时是否重新加载数据

external: 为 external="true"或者 href 是外网连接时, 以iframe 方式打开navTab 页面

Js 调用

```
navTab.openTab(tabid, url, { title:"New Tab", fresh:false, data:{} });
```

其中 data: {} json 格式的数据

Tab 组件扩展

开发人员不需写任何 javascript, 只要使用下面的 html 结构就可以.

```
<div class="tabs">
  <div class="tabsHeader">
    <div class="tabsHeaderContent">
      <ul>
        <li class="selected"><a href="#"><span>标题1</span></a></li>
        <li><a href="#"><span>标题2</span></a></li>
      </ul>
    </div>
  </div>
  <div class="tabsContent" style="height:150px;">
    <div>内容1</div>
    <div>内容2</div>
  </div>
  <div class="tabsFooter">
    <div class="tabsFooterContent"></div>
  </div>
</div>
```

Accordion 组件

```
<div class="accordion" [fillSpace="xxxKey"]>
  <div class="accordionHeader">
    <h2><span>icon</span>面板1</h2>
  </div>
  <div class="accordionContent" style="height:200px">
    内容1
  </div>
  <div class="accordionHeader">
    <h2><span>icon</span>面板2</h2>
  </div>
  <div class="accordionContent">
    内容2
  </div>
  <div class="accordionHeader">
    <h2><span>icon</span>面板3</h2>
  </div>
```

```

        <div class="accordionContent">
            内容3
        </div>
    </div>

```

容器高度自适应

容器高度自适应, 只要增加扩展属性 `layoutH="xx"`, 单位是像素.

`LayoutH` 表示容器内工具栏高度. 浏览器窗口大小改变时容器高度自适应, 但容器内工具栏高度是固定的, 需要告诉 js 工具栏高度来计算出内容的高度.

示例:

```

<div class="layoutBox">

    <div layoutH="150">内容</div>

</div>

```

注意: `layoutH="150"` 的高度是根据含有 `class="layoutBox"` 的父容器 `div` 动态更新的

CSS Table

原生 html + CSS 实现, 无 js 处理效果、最简单、最基本、性能最高的 table。

在 `table` 标签上增加 `class="list"`, `table` 外面包一个 `<div layoutH="xx">` 实现 `table` 固定高度

```

<div layoutH="120">
<table class="list" width="98%">
    <thead>
        <tr>
            <th colspan="2">客户信息</th>
            <th colspan="2">基本信息</th>
            <th colspan="3">资料信息</th>
        </tr>
        <tr>
            <th width="80">客户号</th>
            <th width="100">客户名称</th>
            <th width="100">客户划分</th>
            <th>证件号码</th>
            <th align="right" width="100">信用等级</th>
            <th width="100">企业性质</th>
            <th width="100">建档日期</th>
        </tr>
    </thead>

    <tbody>

        <tr>
            <td>iso127309</td>
            <td>北京市政府哟哟哟</td>
            <td>政府单位</td>
            <td>0-0001027766351528</td>
            <td>四等级</td>
            <td>政府单位</td>
            <td>2009-05-21</td>
        </tr>
    </tbody>
</table>

```

</div>

Table 扩展

在 `table` 标签上增加 `class="table"`

```
<table layoutH="170" class="table">
  <thead>
    <tr>
      <th width="80">客户号</th>
      <th width="100">客户名称</th>
      <th align="right">证件号码</th>
      <th width="100">建档日期</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>iso127309</td>
      <td>北京市政府</td>
      <td>0-0001027766351528</td>
      <td>2009-05-21</td>
    </tr>
  </tbody>
</table>
```

在线编辑器

在 `textarea` 标签上增加 `class="editor"`

示例:

```
<textarea class="editor" name="description" rows="15" cols="80">内容</textarea>
```

分页组件

分页思路服务器返回当前页的数据，总条数，再由 js 来生成分页标签。分页是配合服务器端来处理的, 不是存 js 做的分页。

因为如果数据量很大，比如有好几百页，存 js 分页就是悲剧了，存 js 分页是必须一次载入所有数据，性能很慢。

分页组件参数要由服务器传过来 `targetType`, `totalCount`, `numPerPage`, `pageNumShown`, `currentPage`

框架会自动把下面的 form 中 `pageNum` 修改后，ajax 重新发请求。下面这个 form 是用来存查询条件的

```
<form id="pagerForm" action="xxx" method="post">
  <input type="hidden" name="pageNum" value="1" /><!-- 【必须】value=1可以写死-->
  <input type="hidden" name="numPerPage" value="20" /><!-- 【可选】每页显示多少条-->
  <input type="hidden" name="orderField" value="xxx" /><!-- 【可选】查询排序-->
  <input type="hidden" name="orderDirection" value="asc" /><!-- 【可选】升序降序-->

  <!-- 【可选】其它查询条件，业务有关，有什么查询条件就加什么参数。
  也可以在searchForm上设置属性rel="pagerForm"，js框架会自动把searchForm搜索条件复制到
  pagerForm中 -->
  <input type="hidden" name="name" value="xxx" />
  <input type="hidden" name="status" value="active" />
  .....
</form>
```

分页组件处理分页流程:

- 1) pagerForm 中缓存了当前的查询条件，加上一个 pageNum 字段

2) 点击分页时动态修改 pageNum, 重新提交 pagerForm

分页组件使用方法:

```
<div class="pagination" targetType="navTab" totalCount="200" numPerPage="20" pageNumShown="10"
currentPage="1"></div>
```

测试方法, currentPage 从 1 改为 2, 就是第 2 页了, 把上面那句改为:

```
<div class="pagination" targetType="navTab" totalCount="200" numPerPage="20" pageNumShown="10"
currentPage="2"></div>
```

参数说明:

targetType: navTab 或 dialog, 用来标记是 navTab 上的分页还是 dialog 上的分页

totalCount: 总条数

numPerPage: 每页显示多少条

pageNumShown: 页标数字多少个

currentPage: 当前是第几页

注意:

服务器端返回一个页面碎片, 其中包括 pagerForm, table, 和分页的 div。只要把这个页面碎片组装好就行。

ajaxTodo 扩展

navTab 页面上 a 链接添加 target="ajaxTodo" 后框架会自动绑定相应的 ajax 处理。【参考 dwz.ajax.js】

可选 a 链接扩展属性 [title="xxx"] 提示确认信息

示例:

```
<a href="/news.do?method=remove&id=${item.id}" target="ajaxTodo" title="确定要删除吗?">>删除</a>
<a href="/news.do?method=publish&id=${item.id}" target="ajaxTodo">发表</a>
```

框架自动绑定js

```
$( "a[target=ajaxTodo]", $p ).each( function() {
    $( this ).click( function( event ) {
        ajaxTodo( $( this ).attr( "href" ) );
        event.preventDefault();
    } );
} );
```

dwzExport 列表数据导出

链接添加 target="dwzExport" 后框架会自动绑定相应的 ajax 处理。

targetType="navTab" 根据当期 navTab 页面中的 pagerForm 参数导出, 默认

targetType="dialog" 根据当期 dialog 页面中的 pagerForm 参数导出

title="实要导出这些记录吗?" 确认提示信息, 可选项

示例:

```
<a href="doc/dwz-team.xls" target="dwzExport" targetType="dialog" title="实要导出这些记录吗?">导出 EXCEL</a>
```

Input alt 扩展

示例:

```
<input name="xxx" alt="请输入客户名称"/>
```

Tree 扩展

```
<ul class="tree [treeFolder treeCheck [expand|collapse]]" oncheck="kkk">
  <li><a href="#" target="navTab" rel="main" tname="name" tvalue="value" checked="true">第一级菜单项 A</a>
    <ul>
      <li><a href="#" target="dialog" rel="dialog1" tname="name" tvalue="value" checked="true">第二级菜单项 A </a></li>
      <li><a href="#">第二级菜单项 B </a></li>
      <li><a href="#">第二级菜单项 C </a>
        <ul>
          <li><a href="#">第三级菜单项 A </a></li>
          <li><a href="#">第三级菜单项 B </a></li>
        </ul>
      </li>
    </ul>
  </li>
  <li><a href="#">第一级菜单项 B</a></li>
</ul>
```

树结构是按,的嵌套格式构成, 将最顶级的以 class="tree"标识即可。treeFolder, treeCheck, expand|collapse 则为可选的。

treeFolder:在所有树节点前加上 Icon 图标

treeCheck:在所有树节点前加上 checkbox, 此时需要在<a> 加上三个扩展属性 tname="", tvalue="", checked, 其中 tname 与 tvalue 对应该 checkbox 的 name 与 value 属性

checked 表示 checkbox 的默认状态是否 checked.

expand 与 collapse: expand 表示树的所有第一级节点默认是展开状态, collapse 则表示所有第一级节点默认为折叠状态, 当 expand 与 collapse 都没有时默认则会展开第一个节点。

扩展属性 oncheck 是自定义函数, 用来接收点击 checkbox 时返回值, 当点击非子树节点 checkbox 时返回数据格式为: {checked:true|false, items:{name:name, value:value}}, 当点击了树节点 checkbox 时, 此子树节点下所有的 checkbox 都将选中, 同时返回此子树节点下所有的 checkbox 的值, 格式为{checked:true|false, items:{{name:name, value:value}, {name:name, value:value}.....}}

Panel 扩展

```
<div class="panel [close collapse]" [defH="200"|minH="100"]>
  <h1>标题</h1>
  <div>
    内容
  </div>
</div>
```

顶层 div 以 class=" panel" 标识即可, 其中的<h1>为 panel 的标题, <h1>后的<div>则是放置内容的容器.

Class 中的 close 与 collapse 为可选项, close 表示 panel 默认为关闭状态, 没有则默认为打开状态. collapse 再表示此 panel 是否为可折叠的 panel, 没有则此 panel 不可折叠. 扩展属性 defH 则表示 panel 内容部分的固定高度, 没有则 panel 内容部分的高度为实际内容的高度, minH 可以指定 panel 内容部分的最小高度.

日历控件

```
<input type="text" name="xxx" class="date" [dateFmt="yyyy-MM-dd"] [minDate="{%y-80}"]
[maxDate="{%y+5}"] />
```

日期格式:

* Field	Full Form	Short Form
* Year	yyyy (4 digits)	yy (2 digits), y (2 or 4 digits)
* Month	MMM (name or abbr.)	MM (2 digits), M (1 or 2 digits)
*	NNN (abbr.)	
* Day of Month	dd (2 digits)	d (1 or 2 digits)
* Day of Week	EE (name)	E (abbr)
* Hour (1-12)	hh (2 digits)	h (1 or 2 digits)
* Hour (0-23)	HH (2 digits)	H (1 or 2 digits)
* Hour (0-11)	KK (2 digits)	K (1 or 2 digits)
* Hour (1-24)	kk (2 digits)	k (1 or 2 digits)
* Minute	mm (2 digits)	m (1 or 2 digits)
* Second	ss (2 digits)	s (1 or 2 digits)
* AM/PM	a	

定义日期范围属性 minDate,maxDate 静态格式 y-M-d 或 y-M 或 y, 支持以下几种写法:

```
minDate="2000-01-15" maxDate="2012-12-15"

minDate="2000-01" maxDate="2012-12"

minDate="2000" maxDate="2012"
```

定义日期范围属性 minDate,maxDate 动态态格式 %y-%M-%d 或 %y-%M 或 %y, 支持以下几种写法:

```
minDate="{%y-10}-%M-%d" maxDate="{%y}-%M-{%d+1}"

minDate="{%y-10}-%M" maxDate="{%y+10}-%M"

minDate="{%y-10}" maxDate="{%y+10}"
```

示例:

```
<p>
  <label>默认格式: </label>
  <input type="text" name="date1" class="date" readonly="true"/>
  <a class="inputDateButton" href="javascript:;">选择</a>
  <span class="info">yyyy-MM-dd</span>
</p>
<p>
  <label>定义日期范围: </label>
  <input type="text" name="date2" value="2000-01-15" class="date" minDate="2000-01-15"
```

```

maxDate="2012-12-15" readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
</p>
<p>
    <label>自定义日期格式: </label>
    <input type="text" name="date3" class="date" dateFmt="yyyy/MM/dd" minDate="2000-01"
maxDate="2012-06" readonly="true" />
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyy/MM/dd</span>
</p>
<p>
    <label>自定义日期格式: </label>
    <input type="text" name="date4" class="date" dateFmt="dd/MM/yyyy" minDate="2000"
maxDate="2012" readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">dd/MM/yyyy</span>
</p>
<p>
    <label>动态参数minDate: </label>
    <input type="text" name="date5" class="date" dateFmt="dd/MM/yy" minDate="{%y-10}-%M-
%d" maxDate="{%y}-%M-{%d+1}"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">dd/MM/yy</span>
</p>
<p>
    <label>自定义日期格式: </label>
    <input type="text" name="date6" class="date" dateFmt="yyyyMMdd" minDate="2000-01-01"
maxDate="2020-12-31"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyyMMdd</span>
</p>
<p>
    <label>自定义日期格式: </label>
    <input type="text" name="date7" class="date" dateFmt="yyyy年MM月dd日" minDate="2000-
01-01" maxDate="2020-12-31"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyy年MM月dd日</span>
</p>
<p>
    <label>自定义日期格式: </label>
    <input type="text" name="date8" class="date" dateFmt="y年M月d日" minDate="2000-01-01"
maxDate="2020-12-31"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">y年M月d日</span>
</p>
<div class="divider"></div>
<h3>日期 + 时间</h3>
<div class="unit">
    <label>自定义日期格式: </label>
    <input type="text" name="date10" class="date" dateFmt="yyyy-MM-dd HH:mm:ss"
readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyy-MM-dd HH:mm:ss</span>
</div>
<div class="unit">
    <label>自定义日期格式: </label>
    <input type="text" name="date11" class="date" dateFmt="yyyy-MM-dd HH:mm"
readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyy-MM-dd HH:mm</span>

```

```
</div>
<div class="unit">
    <label>自定义日期格式: </label>
    <input type="text" name="date12" class="date" dateFmt="yyyy-MM-dd HH:ss"
readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">yyyy-MM-dd HH:ss</span>
</div>

<div class="unit">
    <label>自定义日期格式: </label>
    <input type="text" name="date13" class="date" dateFmt="y年M月d日 H点" readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">y年M月d日 H点</span>
</div>
<div class="unit">
    <label>自定义日期格式: </label>
    <input type="text" name="date14" class="date" dateFmt="EEE HH:mm:ss" readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">EEE HH:mm:ss</span>
</div>
<div class="unit">
    <label>自定义只有时间: </label>
    <input type="text" name="date15" class="date" dateFmt="HH:mm:ss" readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">HH:mm:ss</span>
</div>
<div class="unit">
    <label>自定义时间: </label>
    <input type="text" name="date16" class="date" dateFmt="HH:mm" mmStep="15"
readonly="true"/>
    <a class="inputDateButton" href="javascript:;">选择</a>
    <span class="info">HH:mm</span>
</div>
```

url 变量替换

HTML 扩展方式navTab, dialog, ajaxTodo 的 url 支持变量替换。例如: `URL_ /edit/id/{xxx}`

大括号内的 xxx 就是变量名，主要功能是结合 table 组件一起使用，下面是 dwz_thinkphp 中用户列表的示例：

下图中的删除、编辑、修改密码都是用了 url 变量替换：

<div>新增 删除 编辑 修改密码</div>			
编号	用户名	昵称	Email
36	zhanghuihua	张慧华	zhanghuihua@sohu.com
4	leader	领导	
3	member	员工	
2	demo	演示	

删除、编辑、修改密码使用了变量 `{sid_user}`
<tbody>中<tr target="sid_user" rel="{ \$vo['id'] }">
当选中的一行时, tr上的rel值会自动替换到url变量中。
注意url变量名 `{sid_user}` 和tr的target="sid_user"保持一致。

代码示例：


```

<a class="delete" href="__URL__/foreverdelete/id/{sid_user}/navTabId/__MODULE__"
target="ajaxTodo" title="你确定要删除吗? " warn="请选择用户"><span>删除</span></a>

<a class="edit" href="__URL__/edit/id/{sid_user}" target="dialog" mask="true" warn="请选择用户"><span>编辑</span></a>

<a class="icon" href="__URL__/password/id/{sid_user}" target="dialog" mask="true" warn="请选择用户"><span>修改密码</span></a>

<table class="list" width="100%" layoutH="116">
  <thead>
    <tr>
      <th width="60">编号</th>
      <th width="100">用户名</th>
      <th>昵称</th>
      <th>Email</th>
      <th width="100">添加时间</th>
      <th width="120">上次登录</th>
      <th width="80">登录次数</th>
      <th width="80">状态</th>
    </tr>
  </thead>
  <tbody>
    <volist id="vo" name="list">
      <tr target="sid_user" rel="{ $vo['id'] }">
        <td>{ $vo['id'] }</td>
        <td>{ $vo['account'] }</td>
        <td>{ $vo['nickname'] }</td>
        <td>{ $vo['email'] }</td>
        <td>{ $vo['create_time'] | date="Y-m-d",### }</td>
        <td>{ $vo['last_login_time'] | date="Y-m-d H:i:s",### }</td>
        <td>{ $vo['login_count'] }</td>
        <td>{ $vo['status'] | showStatus=$vo['id'] }</td>
      </tr>
    </volist>
  </tbody>
</table>

```

checkbox 全选、反选

checkbox 全选、反选。（demo → 表单组件 → 多选框/单选框）

```

<label><input type="checkbox" name="c1" value="1" />选择1</label>
<label><input type="checkbox" name="c1" value="2" />选择2</label>
<label><input type="checkbox" name="c1" value="3" />选择3</label>

<input type="checkbox" class="checkboxCtrl" group="c1" />全选
<button type="button" class="checkboxCtrl" group="c1" selectType="invert">反选</button>

```

uploadify 多文件上传

```

<div id="fileQueue" class="fileQueue"></div>

<input id="testFileInput" type="file" name="image"
  uploader="uploadify/scripts/uploadify.swf"
  cancelImg="uploadify/cancel.png"
  script="ajaxDone.html"
  scriptData="{PHPSESSID:'xxx', ajax:1}"
  folder="/folder"

```

```
fileQueue="fileQueue"
[onComplete="uploadifyComplete"]
[onAllComplete="uploadifyAllComplete"] />
```

参数说明:

uploader: flash组件uploadify.swf的访问路径

cancelImg: 取消按钮使用的图片路径

script: 服务器端处理上传文件的路径

scriptData: 上传文件时需要传递给服务器的其他参数, 是json格式

folder: 是服务器存储文件的目录

fileQueue: 是上传进度显示区域

onAllComplete: 可选参数, 单个文件上传完时触发, 参数有:

event: event 事件对象

Id: 上传进度队列的id

fileObj: 是一个包含上传文件信息的对象, 包括的信息有:

name: 文件名

filePath: 上传文件在服务器端的路径

size: 文件的大小

creationDate: 文件创建的时间

modificationDate: 文件最后更改的时间

type: 是以"."开始的文件扩展名

response: 服务器端处理完上传文件后返回的文本

data: 包含有两个参数的对象,

fileCount: 上传队列中还剩下的文件数

speed: 以KB/s为单位的文件上传平均速度

uploadifyAllComplete: 可选参数, 全部文件上传完成时调用的函数, 参数有:

event: event 事件对象

data: 是一个包含以下信息的对象,

filesUploaded: 已经上传的文件总数

errors: 上传出错的文件总数

allBytesLoaded: 已经上传文件的总大小

speed: 以 KB/s 为单位的上传文件的平均速度

以下 3 个方法是 dwz.ajax.js 中定义的用于文件上传的会调函数:

```
function uploadifyAllComplete(event, data){
    if (data.errors) {
        var msg = "The total number of files uploaded: "+data.filesUploaded+"\n"
            + "The total number of errors while uploading: "+data.errors+"\n"
            + "The total number of bytes uploaded: "+data.allBytesLoaded+"\n"
            + "The average speed of all uploaded files: "+data.speed;
        alert("event:" + event + "\n" + msg);
    }
}

function uploadifyComplete(event, queueId, fileObj, response, data){
    DWZ.ajaxDone(DWZ.jsonEval(response));
}
```

```

}

function uploadifyError(event, queueId, fileObj, errorObj){
    alert("event:" + event + "\nqueueId:" + queueId + "\nfileObj.name:"
        + fileObj.name + "\nerrorObj.type:" + errorObj.type + "\nerrorObj.info:" +
errorObj.info);
}

```

combox 组件

在传统的 select 用 class 定义：class="combox", html 扩展：保留原有属性 name, 增加了属性：ref。

ref 属性则是为了做级联定义的，ref 所指向的则是当前 combox 值改变成引起联动的下一级 combox，ref 用下一级 combox 的 id 属性来赋值。

注意：一般 combox 没必要设置 id 属性，只要级联时需要设置子级 id 等于父级 ref，不同 navTab 和 dialog 中 combox 组件 id 必须唯一

以下是级联示例：

```

<select class="combox" name="province" ref="combox_city" refUrl="city.do?code={value}">
    <option value="all">所有省市</option>
    <option value="bj">北京</option>
    <option value="sh">上海</option>
</select>
<select class="combox" name="city" id="combox_city" ref="combox_area" refUrl="
area.do?code={value}">
    <option value="all">所有城市</option>
</select>
<select class="combox" name="area" id="combox_area">
    <option value="all">所有区县</option>
</select>

```

服务器端返回 json 格式：

```

[
    ["all", "所有城市"],
    ["bj", "北京市"]
]

```

suggest+lookup+主从结构

dwz.database.js 主要功能是数据库操作相关的界面组件。主要分为 2 部分，分别是查找带回和主从结构。

- 查找带回：lookup、suggest、lookup 附件(文件上传带回)、多选查找带回 multLookup 几个 jQuery 插件，以及 \$.bringBack、\$.bringBackSuggest 两个配套查找带回工具方法
- 主从结构：itemDetail

suggest+lookup+主从结构 请参照 demo：界面组件 → 常用组件 → suggest+lookup+主从结构

查找带回

lookup、suggest 都支持联动效果，比如类似选省份、城市联动效果。支持自定义查找带回主键 lookupPk, 可选项默认为 id。

suggest+lookup

部门名称A1: * 🔍 (suggest+lookup)

部门编号A1:

部门名称A2: * 🔍 (suggest, lookup 联动效果)

```
<dl class="nowrap">
  <dt>部门名称A1: </dt>
  <dd>
    <input id="inputOrg1" name="org1.id" value="" type="hidden"/>
    <input class="required" name="org1.orgName" type="text" postField="keyword" suggestFields="orgNum,orgName"
      suggestUrl="demo/database/db_lookupSuggest.html" lookupGroup="org1"/>
    <a class="btnLook" href="demo/database/dwzOrgLookup.html" lookupGroup="org1">查找带回</a>
    <span class="info">(suggest+lookup)</span>
  </dd>
</dl>
<dl class="nowrap">
  <dt>部门编号A1: </dt>
  <dd>
    <input class="readonly" name="org1.orgNum" readonly="readonly" type="text"/>
  </dd>
</dl>
<dl class="nowrap">
  <dt>部门名称A2: </dt>
  <dd>
    <input name="org1_1.id" value="" type="hidden"/>
    <input class="required" name="org1_1.orgName" type="text" suggestFields="orgNum,orgName"
      suggestUrl="demo/database/db_lookupSuggest.html?orgId={inputOrg1}" warn="请选择部门A1" lookupGroup="org1_1"/>
    <a class="btnLook" href="demo/database/dwzOrgLookup.html?orgId={inputOrg1}" warn="请选择部门A1" lookupGroup="org1_1">查找带回</a>
    <span class="info">(suggest, lookup 联动效果)</span>
  </dd>
</dl>
```

这个就是lookupGroup, 可以自定义, 也可以是空字符串, 但必须保持一致

自定义suggest提交参数, 不是必须, 默认是inputValue

这几个input name属性后缀和弹出lookup的dialog中的\$.bringBack方法参数中的key保持一致

用于lookup, suggest联动
父级id="xxx" 必须和子级url中替换变量保持一致

lookup 通过复选框选择多个值查找回带示例:

请参照 dwz-ria 中 demo/database/db_widge.html和 demo/database/dwzOrgLookup2.html 页面

部门名称C: 🔍 (lookup 通过复选框选择多个值查找回带)

部门编号C:

```
<button type="button" multLookup="orgId" warn="请选择部门">选择带回</button>
```

```
<input type="checkbox" name="orgId" value="{id:'1', orgName:'技术部', orgNum:'1001'}"/>
<input type="checkbox" name="orgId" value="{id:'2', orgName:'人事部', orgNum:'1002'}"/>
<input type="checkbox" name="orgId" value="{id:'3', orgName:'销售部', orgNum:'1003'}"/>
```

主从结构

针对主表和从表的数据库结构设计, 实现主从结构复合表单, 动态添加、删除从表字段。

请参照 dwz-ria 中 demo/database/db_widge.html

```
<table class="list nowrap itemDetail" addButton="新建从表1条目" width="100%">
  <thead>
    <tr>
      <th type="text" name="items.itemString" size="12" fieldClass="required">从字符串</th>
      <th type="text" name="items.itemInt" size="12" fieldClass="digits">从整数</th>
      <th type="text" name="items.itemFloat" size="12" fieldClass="number">从浮点</th>
      <th type="date" name="items.itemDate" size="12">从日期</th>
    </tr>
  </thead>
```

```

        <th type="date" format="yyyy-MM-dd HH:mm:ss" name="items.itemDataTime" size="16">从日期时间</th>
        <th type="lookup" name="dwz.items.org.orgName" lookupGroup="items.org"
lookupUrl="xxxUrl" suggestUrl="xxxUrl" suggestFields="orgName" size="12">部门名称</th>
        <th type="enum" name="items.itemEnum" enumUrl="xxxUrl" size="12">从枚举</th>
        <th type="attach" name="dwz.items.attachment.fileName" lookupGroup="items.attachment"
lookupUrl="xxxUrl" size="12">从附件</th>
        <th type="del" width="60">操作</th>
</tr>
</thead>
<tbody></tbody>
</table>

```

<table>标签中class="itemDetail" 必须用于关联主从结构js效果。addButton="xxx"可选默认为"Add New"用来定义添加从表按钮的文字。

<th>标签中: type必填项, type类型有text、date、lookup、enum、attach、del
name必填项, 定义子表字段名称
size可选项, 默认size="12", 定义从表input字段的长度
fieldClass可选项, 用来定义表input字段的class
lookupGroup当type="lookup" 或type="attach"时必填
lookupUrl当type="lookup"时lookupUrl和suggesUrl至少填一项, 当type="attach"时必填
suggestUrl当type="lookup"时lookupUrl和suggesUrl至少填一项
suggestFields当type="lookup"并且有suggestUrl时必填
enumUrl当type="enum"时必填

Ajax 表单

Ajax 表单相关的操作封装在 dwz.ajax.js 中。表单查询、分页、表单提交 js 方法都已经封装在里面了。开发人员自己不需写 js, 按标准使用就可以了。

表单查询

DWZ 中定义表单查询和分页都是用这个 pagerForm 来临时存查询条件。所以需要在查询页面上放下面的 form

```

<form id="pagerForm" action="xxx" method="post">
    <input type="hidden" name="pageNum" value="1" /><!-- 【必须】 value=1可以写死-->
    <input type="hidden" name="numPerPage" value="20" /><!-- 【可选】 每页显示多少条-->
    <input type="hidden" name="orderField" value="xxx" /><!-- 【可选】 查询排序-->
    <!-- 【可选】 其它查询条件, 业务有关, 有什么查询条件就加什么参数-->
    <input type="hidden" name="status" value="active" />
</form>

```

ajax 表单查询

```
<form action="xxx" method="post" onsubmit="return navTabSearch(this)">
```

或

```
<form action="xxx" method="post" onsubmit="return dialogSearch(this)">
```

修改每页显示行数

```

<select class="combox" name="numPerPage"
onchange="navTabPageBreak({numPerPage:this.value})">
    <option value="20">20</option>
    <option value="50">50</option>
    <option value="100">100</option>
    <option value="200">200</option>

```

```
</select>
```

```
/**
 * 处理navTab弹出层上的查询，会重新载入当前navTab
 * @param {Object} form
 */
function navTabSearch(form) {
    navTab.reload(form.action, $(form).serializeArray());
    return false;
}

/**
 * 处理dialog弹出层上的查询，会重新载入当前dialog
 * @param {Object} form
 */
function dialogSearch(form) {
    $.pdialog.reload(form.action, $(form).serializeArray());
    return false;
}

/**
 * 处理navTab中的分页和排序
 * @param args {pageNum:"n", numPerPage:"n", orderField:"xxx"}
 */
function navTabPageBreak(args) {
    var form = _getPagerForm(navTab.getCurrentPanel(), args);
    if (form) navTab.reload(form.action, $(form).serializeArray());
}

/**
 * 处理dialog中的分页和排序
 * @param args {pageNum:"n", numPerPage:"n", orderField:"xxx"}
 */
function dialogPageBreak(args) {
    var form = _getPagerForm($.pdialog.getCurrent(), args);
    if (form) $.pdialog.reload(form.action, $(form).serializeArray());
}
```

ajax 表单查询完整示例:

```
<div class="pageHeader">
    <form action="/render.do?method=search" method="post" onsubmit="return
navTabSearch(this)">
        <input type="hidden" name="resourceStatus" value="${param.resourceStatus}" />
        <input type="hidden" name="pageNum" value="1" />
        <input type="hidden" name="orderField" value="${param.orderField}" />
        <div class="searchBar">
            <div class="searchContent">
                <select name="resourceType">
                    <option value="">全部栏目</option>
                    <c:forEach var="item" items="${model.resourceTypes}">
                        <option value="${item.id}" ${param.resourceType eq
item.id?"selected":""}>${item.name}</option>
                    </c:forEach>
                </select>
                <input name="keywords" type="text" size="25" value="${param.keywords}" />
            </div>
            <div class="subBar">
                <ul>
                    <li><div class="buttonActive"><div class="buttonContent"><button
type="submit">检索</button></div></div></li>
                </ul>
            </div>
        </div>
```

```
</div>
</form>
</div>
```

普通Ajax 表单提交

DWZ 框架Ajax 无刷新表单提交处理流程是:

1. ajax 表单提交给服务器
2. 服务器返回一个固定格式 json 结构
3. js 会调函数根据这个 json 数据做相应的处理

注意:

DWZ 框架默认的 ajax 表单提交都是返回 json 数据, 告诉客户端操作是否成功, 成功或失败提示信息, 以及成功后的处理方式(刷新某个 navTab 或关闭某个 navTab 或 navTab 页面跳转)。

表单提交后服务器操作失败了, 客户端接收 statusCode 和 message 后给出错误提示, 表单页面是不动的。这样可以方便用户看到出错原因后直接修改表单数据再次提交, 而不用重填整个表单数据。当然如果你还是喜欢表单提交后直接载入 html 页面也是没有问题的, 参照 dwz.ajax.js 自己扩展一下也是没问题的。

DWZ 表单提交dwz.ajax.js

- Ajax 表单提交后自动调用默认回调函数, 操作成功或失败提示。

Form 标签上增加 `onsubmit="return validateCallback(this);"`

- Ajax 表单提交后如果需要重新加载某个 navTab 或关闭 dialog, 可以使用 dwz.ajax.js 中事先定义的方法 `navTabAjaxDone/dialogAjaxDone`

注意: 如果表单在 navTab 页面上使用 `navTabAjaxDone`, 表单在 dialog 页面上使用 `dialogAjaxDone`

Form 标签上增加 `onsubmit="return validateCallback(this, navTabAjaxDone) "`

或 `onsubmit="return validateCallback(this, dialogAjaxDone) "`

- Ajax 表单提交后如果需要做一些其它处理也可以自定义一个回调函数 `xxxAjaxDone`。例如下面表单提交成功后关闭当前 navTab, 或者重新载入某个 tab.

Form 标签上增加`onsubmit="return validateCallback(this, xxxAjaxDone) "`

服务器端响应

Ajax 表单提交后服务器端需要返回以下 json 代码:

```
{
    "statusCode": "200",
    "message": "操作成功",
    "navTabId": "",
    "rel": "",
    "callbackType": "closeCurrent",
    "forwardUrl": ""
}
```


以下是 `dwz.ajax.js` 中定义的 `navTabAjaxDone` 和 `dialogAjaxDone` 代码片段:

```
/**
 * navTabAjaxDone是DWZ框架中预定义的表单提交回调函数.
 * 服务器转回navTabId可以把那个navTab标记为reloadFlag=1, 下次切换到那个navTab时会重新载入内容.
 * callbackType如果是closeCurrent就会关闭当前tab
 * 只有callbackType="forward"时需要forwardUrl值
 * navTabAjaxDone这个回调函数基本可以通用了, 如果还有特殊需要也可以自定义回调函数.
 * 如果表单提交只提示操作是否成功, 就可以不指定回调函数. 框架会默认调用DWZ.ajaxDone()
 * <form action="/user.do?method=save" onsubmit="return validateCallback(this,
navTabAjaxDone)">
 *
 * form提交后返回json数据结构statusCode=DWZ.statusCode.ok表示操作成功, 做页面跳转等操作.
statusCode=DWZ.statusCode.error表示操作失败, 提示错误原因.
 * statusCode=DWZ.statusCode.timeout表示session超时, 下次点击时跳转到DWZ.loginUrl
 * {"statusCode":"200", "message":"操作成功", "navTabId":"navNewsLi", "forwardUrl":"","
"callbackType":"closeCurrent"}
 * {"statusCode":"300", "message":"操作失败"}
 * {"statusCode":"301", "message":"会话超时"}
 */
function navTabAjaxDone(json) {
    DWZ.ajaxDone(json);
    if (json.statusCode == DWZ.statusCode.ok) {
        if (json.navTabId) { //把指定navTab页面标记为需要“重新载入”。注意navTabId不能是当前
navTab页面的
            navTab.reloadFlag(json.navTabId);
        } else { //重新载入当前navTab页面
            navTabPageBreak();
        }

        if ("closeCurrent" == json.callbackType) {
            setTimeout(function() {navTab.closeCurrentTab();}, 100);
        } else if ("forward" == json.callbackType) {
            navTab.reload(json.forwardUrl);
        }
    }
}

/**
 * dialog上的表单提交回调函数
 * 服务器转回navTabId, 可以重新载入指定的navTab. statusCode=DWZ.statusCode.ok表示操作成功, 自动关闭
当前dialog
 *
 * form提交后返回json数据结构,json格式和navTabAjaxDone一致
 */
function dialogAjaxDone(json) {
    DWZ.ajaxDone(json);
    if (json.statusCode == DWZ.statusCode.ok) {
        if (json.navTabId) {
            navTab.reload(json.forwardUrl, {}, json.navTabId);
        }
        $.pdialog.closeCurrent();
    }
}
```

示例:

```
<form method="post" action="url" class="pageForm required-validate" onsubmit="return
validateCallback(this);">
```



```

<div class="pageFormContent" layoutH="56">
    <p>
        <label>E-Mail: </label>
        <input class="required email" name="email" type="text" size="30" />
    </p>
    <p>
        <label>客户名称: </label>
        <input class="required" name="name" type="text" size="30" />
    </p>
</div>
<div class="formBar">
    <ul>
        <li>
            <div class="buttonActive"><div class="buttonContent"><button
type="submit">保存</button></div></div>
        </li>
        <li>
            <div class="button"><div class="buttonContent"><button type="Button"
class="close">取消</button></div></div>
        </li>
    </ul>
</div>
</form>

```

文件上传表单提交

因为 Ajax 不支持 `enctype="multipart/form-data"` 所以用隐藏 `iframe` 来处理无刷新表单提交。

```

<form method="post" action="url" class="pageForm required-validate"
enctype="multipart/form-data" onsubmit="return iframeCallback(this);">

```

或

```

<form method="post" action="url" class="pageForm required-validate"
enctype="multipart/form-data" onsubmit="return iframeCallback(this,
[navTabAjaxDone/dialogAjaxDone]);">

```

服务器端响应

DWZ-v1.2版本开始服务器返回和`validateCallback`格式保持一致:

```

{
    "statusCode": "200",
    "message": "操作成功",
    "navTabId": "",
    "rel": "",
    "callbackType": "closeCurrent",
    "forwardUrl": ""
}

```

DWZ-v1.2 以前版本使用隐藏 `iframe` 来处理无刷新表单提交时, 服务器端需要返回以下 js 代码:

```

<script type="text/javascript">
    var statusCode="200";
    var message="操作成功";
    var navTabId="";
    var forwardUrl="";
    var callbackType="closeCurrent"

    var response = {statusCode:statusCode,
        message:message,
        navTabId:navTabId,
        forwardUrl:forwardUrl,
        callbackType:callbackType
    };
    if(window.parent.donecallback) window.parent.donecallback(response);

```

</script>

Java 服务器端表单处理示例

```
public class NewsAction extends BaseAction {

    private NewsManager manager = bf.getManager(BeanManagerKey.newsManager);
    private News news = manager.newNews();
    private Collection<News> newList;

    public String add() {
        return INPUT;
    }

    public String insert() {
        manager.createNews(news);
        return ajaxForwardSuccess(getText("msg.operation.success"));
    }

    public String edit() {
        news = manager.getNews(this.getNewsId());
        return INPUT;
    }

    public String update() {
        News m = manager.getNews(newsId);
        m.copyProperties(news);
        manager.updateNews(m);
        return ajaxForwardSuccess(getText("msg.operation.success"));
    }

    public String publish() {
        manager.publishNews(newsId);
        return ajaxForwardSuccess(getText("msg.operation.success"));
    }

    public String disable() {
        manager.disableNews(newsId);
        return ajaxForwardSuccess(getText("msg.operation.success"));
    }

    public String delete() {
        manager.removeNews(newsId);
        return ajaxForwardSuccess(getText("msg.operation.success"));
    }

}
```

// BaseAction 代码片段

```
public class BaseAction extends ActionSupport {
    private int statusCode = 200;
    private String message = null;
    private String forwardUrl = null;

    private String ajaxForward(int statusCode) {
        this.statusCode = statusCode;
        return OPERATION_DONE;
    }

    protected String ajaxForwardSuccess(String message) {
        this.message = message;
        return ajaxForward(200);
    }
}
```

```

protected String ajaxForwardError(String message) {
    this.message = message;
    return ajaxForward(300);
}

public int getStatusCode() {
    return statusCode;
}

public String getMessage() {
    return message;
}

public String getForwardUrl() {
    return forwardUrl;
}

public void setForwardUrl(String forwardUrl) {
    this.forwardUrl = forwardUrl;
}
}

```

// 工具类判断是否 ajax 请求

```

public class ServerInfo {
    public static boolean isAjax(HttpServletRequest request) {
        if (request != null
            && "XMLHttpRequest".equalsIgnoreCase(request
                .getHeader("X-Requested-With")))
            return true;
        return false;
    }
}

```

DWZ js 库介绍

DWZ 框架初始化

在<head> 引入必要的 js 库

DWZ 框架初始化会自动读取 `dwz.frag.xml` 中的页面组件碎片代码.

`dwz.frag.xml`中定义了一些 dwz 组件碎片和提示信息, 需要初始化到 DWZ 环境中.

注意 `dwz.frag.xml` 路径问题.

假设 `dwz.frag.xml` 放在根目录下, 在<head>标签中调用 `DWZ.init("dwz.frag.xml")`

```
<script type="text/javascript">
$(function() {
    DWZ.init("dwz.frag.xml", {
        loginUrl:"login.html",
        callback:function() {
            initEnv();
            $("#themeList").theme({themeBase:"themes"});
        }
    });
});
</script>
```

dwz.core.js

DWZ 核心库主要功能是 DWZ 初始化, Javascript String 增加了一些扩展方法.

定义 dwz ajax 加载扩展 `loadUrl(url, data, callback)`和 `ajaxUrl(options)`

dwz.ui.js

页面效果初始化, html 扩展绑定 js 效果

dwz.ajax.js

ajax 表单提交封装

dwz.alertMsg.js

➤ 确认提示框

```
alertMsg.confirm("您修改的资料未保存, 请选择保存或取消!", {
    okCall: function(){
        $.post(url, {accountId: accountId}, DWZ.ajaxDone, "json");
    }
});
```

➤ 成功提示框

```
alertMsg.correct('您的数据提交成功!')
```

➤ 错误提示框

```
alertMsg.error('您提交的数据有误, 请检查后重新提交!')
```

➤ 警告提示框

```
alertMsg.warn('您提交的数据有误, 请检查后重新提交!')
```

➤ 信息提示框

```
alertMsg.info('您提交的数据有误, 请检查后重新提交!')
```

dwz.jDialog.js

弹出层组件库

dwz.accordion.js

滑动面板组件库

dwz.barDrag.js

DWZ 左边的活动面板

dwz.navTab.js

导航 **tab** 组件库

navTab API

打开一个标签页 `navTab.openTab(tabid, title, url, [data])`

重新载入标签页，如果无 `tabid` 参数，就载入当前标签页 `navTab.reload(url, data, [tabid])`

获取当前标签页容器 `navTab.getCurrentPanel()`

关闭一个标签页 `navTab.closeTab(tabid)`

关闭当前标签页 `navTab.closeCurrentTab()`

关闭全部标签页 `navTab.closeAllTab()`

dwz.scrollCenter.js

页面容器自动居中组件

dwz.stable.js

table 组件库

dwz.cssTable.js

简单 table 组件库

dwz.tree.js

tree 组件库

dwz.theme.js

切换界面主题风格

dwz.validate.method.js

这是 `jquery.validate.js` 表单验证扩展方法

dwz.validate.zh.js

表单验证本地化

dwz.contextmenu.js

自定义鼠标右键菜单, 先在 `dwz.frag.xml` 加入菜单项定义，下面是 `navTab` 和 `dialog` 两个组件的菜单项定义：

```
<_PAGE_ id="navTabCM"><![CDATA[
<ul id="navTabCM">
    <li rel="closeCurrent">关闭标签页</li>
    <li rel="closeOther">关闭其它标签页</li>
    <li rel="closeAll">关闭全部标签页</li>
</ul>
]]></_PAGE_>
```

```
<_PAGE_ id="dialogCM"><![CDATA[
<ul id="dialogCM">
```

```

        <li rel="closeCurrent">关闭弹出窗口</li>
        <li rel="closeOther">关闭其它弹出窗口</li>
        <li rel="closeAll">关闭全部弹出窗口</li>
    </ul>
]]></_PAGE_>

```

示例:

```

$("body").contextMenu('navTabCM', {
    bindings:{
        closeCurrent:function(t) {
            // TODO
        },
        closeOther:function(t) {
            // TODO
        },
        closeAll:function(t) {
            // TODO
        }
    },
    ctrSub:function(t,m) {
        var mCur = m.find("[rel='closeCurrent']");
        var mOther = m.find("[rel='closeOther']");
        var mAll = m.find("[rel='closeAll']");
        // TODO
    }
});

```

dwz.pagination.js

分页组件库

```

<div class="pagination" targetType="navTab" totalCount="200" numPerPage="20" pageNumShown="10"
currentPage="1"></div>

```

开发人员只要用程序动态生成这个<div>, 不用写 js, 框架自动绑定处理事件。

dwz.database.js

suggest 自动完成的提示框组件

lookup 查找带回组件

itemDetail 主从结构组件

selectedTodo 批量选择操作组件(批量删除, 批量审核...)

dwz.datepicker.js

DWZ 日历控件库

dwz.combox.js

combox 下拉菜单组件, 支持多级联动

dwz.checkbox.js

checkbox 全选、反选。(demo → 表单组件 → 多选框/单选框)

dwz.util.date.js

日期处理工具类

dwz.regional.zh.js

DWZ 本地化

dwz.validate.method.js

jquery.validate.js 扩展

class:

required <input type="text" name="name" class="required"/>

email <input type="text" name="name" class="email"/>

url <input type="text" name="name" class="url"/>

date <input type="text" name="name" class="date"/>

number <input type="text" name="name" class="number"/>

digits <input type="text" name="name" class="digits"/>

creditcard <input type="text" name="name" class="creditcard"/>

attribute:

equalTo: selector <input type="text" name="name" equalTo="#name"/>

maxlength: <input type="text" name="name" maxlength="20"/>

minlength: <input type="text" name="name" minlength="2"/>

实现长度范围时是同时写上 minlength 与 maxlength，此时的提示将是 rangelength 的提示。

max: <input type="text" name="name" max="2"/>

min: <input type="text" name="name" min="2"/>

实现值范围时是同时写上 min 与 max，此时提示将是 range 的提示。

提示内容更改在文件 dwz.regional.zh.js。

参考文档 <http://docs.jquery.com/Plugins/Validation>

Javascript 混淆和压缩

Javascript混淆并用 gzip 压缩后，可以把 300K 的 js 压缩到 40K 左右。

DWZ 混淆和压缩方法：

- 1) 打开bin/gzjs.bat 修改第一行路径为本地文件系统绝对路径
- 2) 执行批处理文件 bin/gzjs.bat

Javascript 混淆

DWZ 混淆工具 bin/ESC.wsf

压缩级别分为 5 种，从 0 到 4

Level 0 :: No compression

Level 1 :: Comment removal

Level 2 :: Whitespace removal

Level 3 :: Newline removal

Level 4 :: Variable substitution

在 WINDOWS 命令行下执行

cscript ESC.wsf -ow menu2.js menu.js 将会把 menu.js 按照 js 压缩级别 2 来压缩（默认 js 压缩级别为 2）为 menu2.js

cscript ESC.wsf -l 3 -ow menu3.js menu.js 将会把 menu.js 按照 js 压缩级别 3 来压缩为 menu3.js

需要注意的是，js 压缩级别 4 会把变量名修改，如果你的 js 中用到了全局变量或者类的话，就不能使用该压缩级别了，否则其它使用你的 js 的文件可能会无法正常运行。

Javascript 用 gzip 压缩

动态的压缩会导致服务器 CPU 占用率过高,现在我想到的解决办法是通过提供静态压缩(就是将 js 预先通过 gzip.exe 压缩好)

传统的 JS 压缩(删除注释,删除多余空格等)提供的压缩率有时还是不尽不意,幸亏现在的浏览器都支持压缩传输(通过设置 http header 的 Content-Encoding=gzip), 可以通过服务器的配置(如 apache)为你的 js 提供压缩传输。

Apache 配制

在 httpd.conf 中加入配制，这样浏览器可以自动解压缩.gzjs

LoadModule mime_module modules/mod_mime.so

AddEncoding x-gzip .gzjs

DWZ 如何中使用打包的 js

在 index.html 中移除全部 dwz.*.js，引入下面 2 个 js 库

```
<script src="bin/dwz.min.js" type="text/javascript"></script>
<script src="javascripts/dwz.regional.zh.js" type="text/javascript"></script>
```

dwz.*.js 打包到 dwz.min.js 步骤:

- 1) 打开 bin/gzjs.bat 修改第一行路径为本地文件系统绝对路径
- 2) 执行批处理文件 bin/gzjs.bat

使用时引入以下 js:

javascripts/speedup.js	【可选】js 加速
javascripts/jquery-1.4.4.js	【必须】jQuery 库

javascripts/jquery.cookie.js 【可选】js 操作 cookie, 目前用于记住用户选择的 theme 风格
javascripts/jquery.validate.js 【必须】表单验证
javascripts/jquery.bgiframe.js 【可选】用于 IE6 弹出层不能盖住 select 问题
xheditor/xheditor-zh-cn.min.js 【可选】xheditor 在线编辑器
uploadify/scripts/swfobject.js 【可选】用于文件批量上传
uploadify/scripts/jquery.uploadify.v2.1.0.js 【可选】用于文件批量上传

bin/dwz.min.js 【必须】 DWZ 框架 js 压缩包
javascripts/dwz.regional.zh.js 【可选】 用于国际化

常见问题及解决

DWZ 中如何整合第三方 jQuery 插件

jQuery 插件一般是 `$(document).ready()` 中初始化

```
$(document).ready(function(){
    // 文档就绪, 初始化 jQuery 插件
});
// 或者缩写形式
$(function(){
    // 文档就绪, 初始化 jQuery 插件
});
```

因为 DWZ RIA 是富客户端思路, 第一次打开时加载界面到浏览器端, 之后和服务器的交互是存数据交互, 不占用界面相关的网络流量。

也就是说, 只需要在一个完整的页面 (通常是起始页, 如 `index.aspx/index.php/index.jsp` 等), 只有这个页面包含完整的 html 结构(`<head><body>`), `<head>` 中引入全部 css、js。其它的页面只需要页面碎片, 就是 `<body></body>` 中的部分。

因为 ajax 加载基本原理是: ajax 加载一段 html 代码片段放到当前页面的某个容器中 (通常是一个 `div`)。当然也可以是 xml 结构、json 结构, 只是在插入到当前页面之前也要转化成 html 代码片段。如果你 ajax 加载一个完整的页面 (就是包括 `<head><body>` 标签的), 插入的当前 document 就有问题了, 因为一个 document 不可能有多个 `<head><body>` 标签。

理解了富客户端思路你也就明白了为什么 DWZ 框架中整合第三方 jQuery 插件不能在 `<head>` 中通过 `$(document).ready()` 初始化。

DWZ 初始化 ajax 加载的页面中的第三方 jQuery 组件:

- 一般插件初始化 `dwz.ui.js` 中的 `initUI` 里面处理, `initUI()` 方法 DWZ 框架封装的 `$.fn.loadUrl()` 自动调用。如果是 jQuery 原生 load 方法需要手动调用 `$.fn.initUI()` 插件。主意 `initUI()` 中初始化是要注意作用域, 里面有一个 `$p` 代表当前 ajax 加载的容器, 只要初始化当前容器新加载的内容就可以了。

```
268     if ($.fn.ajaxTodo) $("a[target=ajaxTodo]", $p).ajaxTodo();
269     if ($.fn.dwzExport) $("a[target=dwzExport]", $p).dwzExport();
```

- 如果一些特殊的 ajax 交互, 自己写回调函数处理

Error loading XML document: dwz.frag.xml

直接用 IE 打开 `index.html` 弹出一个对话框: Error loading XML document: `dwz.frag.xml`

原因: `dwz.frag.xml` 是一个核心文件, 需要加载才可以正常使用。IE ajax load 本地文件有限制, 是 ie 安全级别的问题, 不是框架的问题。

解决方法: 放到 `apache` 或 `iis` 下就可以了。如果不想安装 `apache` 或 `iis` 用 `firefox` 打开就正常了。

IIS 不能用 Ajax 访问 *.html 后缀的页面

Ajax 访问 *.html 后缀的页面在 `Apache` 很好的工作, 但在 `IIS` 不行, `IIS` 下 `firebug` 调试报错 `ajax 405 Method Not Allowed`。

Http 405 原因是 `IIS` 不允许 ajax post 方式访问 *.html 后缀的页。

IIS 在使用 Ajax post 方式请求页面时,一定要动态网页后缀的或者用改用 get 方式! 这是 IIS 的问题, 不是框架 bug。

也可以试试修改 IIS 配置, 添加扩展名 (.html) 的脚本映射。

多个 navTab 页面或 dialog 页面 ID 冲突, 解决方法

如果多个 navTab 页面或 dialog 页面有相同的 ID, 假设这个 ID 为: xxxId

```
$("#xxxId", navTab.getCurrentPanel()); // 获取当前 navTab 中的 xxxId
```

```
$("#xxxId", $.pdialog.getCurrent()); // 获取当前 dialog 中的 xxxId
```

jQuery1.4.2 和 jquery.validate.js 在 IE 的兼容问题

jQuery1.4.2和 jquery.validate.js在 IE 有兼容问题, ajax 表单提交在 IE 不能触发 form onsubmit 事件。

导致 form 提交后跳转到了一个白页面。

升级 jQuery1.4.2 注意事项

jQuery1.4.2 对 json 要求非常严格 key、value 都要用引号抱起来, 否则就无法解析。jQuery1.3.2 以前版本没有这种限制。

```
{ "statusCode": "200", "message": "操作成功" }
```

\$.ajax() 发送ajax请求成功后调用success方法, success根据dataType来解析返回的内容httpData()。分析jQuery1.4.2源码发现dataType="json"的处理方式完全不一样了。1.3.2之前版本是用window.eval()来解析JSON结构, 1.4.2版本添加了parseJSON()方法来解析。估计是window.eval()存在安全漏洞, 1.4.2版本进行了改进, 对JSON格式也要求更严格了。

ECMAScript 5发布有段时间了, 其中就包括了解析JSON的原生API-JSON.parse。许多浏览器已经支持了。主流js库如jQuery, Ext, Prototype都优先使用JSON.parse, 不支持该方法的浏览器则使用new Function或eval。为何优先使用JSON.parse, 我想一个就是性能, 原生的总是要快一些吧。此外JSON.parse较eval也更安全。这里也当然不能落后了, 优先使用JSON.parse, 不行再用new Function方式。最后失败了会给failure的第二个参数msg赋值为"parse json error"

```
parseJSON: function( data ) {
    if ( typeof data !== "string" || !data ) {
        return null;
    }

    data = jQuery.trim( data );

    if ( /^[\],:{}\s]*$/ .test(data.replace(/\\(?:[\\\/bfnrt]|u[0-9a-fA-F]{4})/g, "@")
        .replace(/"[^"\\\n\r]*"|true|false|null|-?\d+(?:\.\d*)?(?:[eE][+\-]?\d+)?/g,
            "]""))
        .replace(/(?:^|:|,)(?:\s*\w+)/g, "")) ) {
        return window.JSON && window.JSON.parse ?
            window.JSON.parse( data ) :
            (new Function("return " + data))();
    } else {
        jQuery.error( "Invalid JSON: " + data );
    }
}
```

weblogic 访问 xml 问题

weblogic访问xml文件, 需要在web.xml中加入下面的声明

```
<mime-mapping>
<extension>xml</extension>
<mime-type>text/xml</mime-type>
</mime-mapping>
```

这时再次访问时webJgoic就给加上contentType了

如何自定义 DWZ 分页参数

pagerForm 默认使用的当前页参数是 pageNum, 每页显示条数 numPerPage, 查询排序字段名 orderField, 升序降序 orderDirection, 更改其它参数需要设置 DWZ.init(pageFrag, options) 的 options["pageInfo"]:

```
<form id="pagerForm" action="xxx" method="post">
    <input type="hidden" name="pageNum" value="1" /></><!-- 【必须】 value=1 可以写死-->
    <input type="hidden" name="numPerPage" value="20" /></><!-- 【可选】 每页显示多少条-->
    <input type="hidden" name="orderField" value="xxx" /></><!-- 【可选】 查询排序字段-->
    <input type="hidden" name="orderDirection" value="asc|desc" /></><!-- 【可选】 升序|降序-->
</form>

<script type="text/javascript">
$(function() {
    DWZ.init("dwz.frag.xml", {
        loginUrl:"login.html", // 跳到登录页面
        statusCode:{ok:200, error:300, timeout:301}, // 【可选】
        pageInfo:{pageNum:"pageNum", numPerPage:"numPerPage", orderField:"orderField",
orderDirection:"orderDirection"}, // 【可选,这里自定义分页参数】
        debug:false, // 调试模式 【true|false】
        callback:function() {
            initEnv();
            $("#themeList").theme({themeBase:"themes"});
        }
    });
});
</script>
```

如何关闭 loading

dwz 的 ajax 方法每次调用都会出现读取数据的 loading, 怎么修改可选的? 我自己写了一个局部更新的 ajax 函数, 结果 loading 太烦人 怎么关掉好?

dwz.ui.js 中注册了 ajax 全局事件:

```
var ajaxbg = $("#background,#progressBar");
ajaxbg.hide();
$(document).ajaxStart(function() {
    ajaxbg.show();
}).ajaxStop(function() {
    ajaxbg.hide();
});
```

\$.ajax() 有个参数 global (Boolean): (默认: true) 是否触发全局 AJAX 事件. 设置为 false 将不会触发全局 AJAX 事件, 如 ajaxStart 或 ajaxStop 可用于控制不同的 Ajax 事件。

DWZ 局部刷新怎样做?

API 调用方式:

```
$("#xxxId").loadUrl(url,data, callback);
```

html 扩展链接方式:

```
<a href="url" target="ajax" rel="xxId"></a>
```

DWZ 版本升级

版本升级如果无特殊说明只要把高版本中的 **dwz.*.js** 全部覆盖、还有 **dwz.frag.xml** 和 **theme** 目录下的 **css** 就可完成升级。

如果新添加了 js 库，需要在 index.html 页面 head 标签中引入。

V1.4.7

解决 dwz.tree.js 那个选中父节点下单个子节点获取不到值问题

V1.4.6

解决 sortDrag 排序出现滚动条的话 滚动出现的部分拖动 一点就跑上面去了

解决 DWZ IE10 表单验证页面 兼容问题，删除 index 页面 `<meta http-equiv="X-UA-Compatible" content="IE=7" />`

升级 xheditor 到 v1.2.1 版

V1.4.5

uploadify 从 2.1 版本升级到 v3.2 版本, 调整 dwz 中 uploadify 2 种 demo（自动上传方式；选择文件后再点击 Upload 按钮上传方式）

修正 navTab, dialog 组件 session 超时处理流程，自动关闭当前 navTab 或 dialog

解决 speedup.js（用于 IE 加速）IE10 中报错问题

修正 dwz.database.js 主从结构中含有日期控件时，dateFmt 格式不一致问题

修正 dwz.database.js 主从结构上传附件，弹出的窗口上传文件之后，带回的文件名不显示出来，原因是该控件中的 items[#index#] 中的 #index# 没有被替换，导致 js 找不到控件，而无从替换

V1.4.4

修复使用 xheditor 插件 IE 下兼容问题：IE 下打开一个含有编辑器的页面，然后关闭，再打开不能录入问题

修复多文件上传插件 uploadify 的 html 扩展方式，java 读取不到数据流问题：原因是以前没有把 input="file" 的 name 属性填充到插件 uploadify 的 fileName 中

保持 navTab 有 pagerForm 的列表页面 reload 查询条件（比如第 5 页上要修改一条记录 修改完了 刷新页数还在第五页）

日历控件添加动态参数 (具体细节请参考本手册：HTML 扩展 -> 日历控件)

添加图表示例

V1.4.3

修复表单验证插件 jquery.validate.js 1.9 版本，在 IE 下重复提交 2 次问题。

V1.4.2

升级表单验证插件 jquery.validate.js 到最新 1.9 版本，解决上 jUI 上一版本中 jQuery1.7.1 和 jquery.valiate 1.7 在 IE 下兼容问题

V1.4.1

调整 suggest+lookup，见文档: HTML 扩展 → suggest+lookup+主从结构

添加拖动排序组件 sortDrag

升级注意更新 dwz.frag.xml、js、css 和表单提交返回的 json 结构添加 confirmMsg 这是 navTabAjaxDone 中 forwardConfirm 时的提示信息，具体细节可以看 dwz.ajax.js 源码和里面的注释

V1.3 Final

升级注意：

- index 页面中<div class="navTab-panel tabsPageContent">添加 class “layoutBox” 改成<div class="navTab-panel tabsPageContent layoutBox">
- 然后更新 js、css、dwz.frag.xml

Changelist:

1. 修复 combox 联动菜单重复发送 ajax 请求问题 s
2. 调整 layoutH= “xx” 的高度根据含有 class=” layoutBox” 的父容器 div 动态更新
3. 修复 navTab 打开外部页面和 iframe 方式打开时，浏览器前进后退问题
 - a. 外部页面
 - b. iframe 方式打开
4. 调整 Lookup、suggest，添加联动效果。自定义查找带回主键 lookupPk, 可选项默认为 id。
5. 添加多选查找带回 multLookup

V1.3 RC4

1. 修改 combox 代码还原 onchane 事件写法，不用 change param 分开写了，修改级联菜单。（请参考本手册“HTML 扩展 → combox 组件”）
2. 修改 dwz.ajax.js 中 ajax 分页、局部刷新相关接口
3. 添加 jUI 组件组合应用 局部刷新分页 demo

V1.3 RC3

1. 修复当左边菜单折叠，然后再展开时，table 的纵向滚动条会消失问题
2. taskBar 弹出框任务栏添加 hover 加亮效果



3. 添加dwzExport 列表数据导出html 扩展，具体介绍请参见本手册 html 扩展部分
导出 EXCEL
4. 简化 index.html 页面，以下代码片段移入 dwz.frag.xml 中
 - taskBar
 - resizable
 - Shadow (阴影层)
 - <div id="alertBackground" class="alertBackground"></div>

- `<div id="dialogBackground" class="dialogBackground"></div>`
- `<div id='background' class='background'></div>`
- `<div id='progressBar' class='progressBar'>数据加载中，请稍等...</div>`

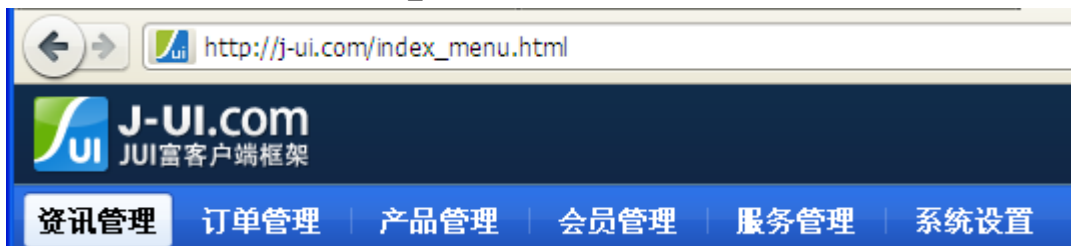
V1.3 RC2

1. 解决 loadUrl 插件 IIS 不能用 Ajax 访问 *.htm 或是 *.html 后缀的页面
2. 日历组件 class="date" 并且自定义 pattern 时和验证冲突问题，pattern 改成 format
3. session 超时，弹出登录框，登录后还能保存当前操作到的状态



V1.3 RC1

1. 添加横向导航栏, 参考示例index_menu.html



2. 添加主从结构组件, 参考示例 db_widget.html 和 dwz.database.js

我的主页 dwz.database.js

dwz.database.js

部门名称: 技术部 (suggest+lookup)

部门编号: 1001

部门名称: (suggest)

部门名称: (lookup)

附件: (lookup附件)

主从结构

新建 1

从字符串	从整数	从浮点	从日期	从日期时间	部门名称	从枚举	从附件	操作
						是		
						是		

保存 关闭

3. 添加suggest自动完成的提示框组件

部门名称A: 技术部 (suggest+lookup)

部门编号A: 1001-技术部
1002-人事部

部门名称B: 1003-销售部
1004-售后部 (gest)

部门名称C: (lookup)

附件: (lookup附件)

- 修复 table 组件当把左边栏收缩后拖动下边的滚动条，内容和题错位问题
- 高级 table 扩展的拖动有 BUG，单击一下就直接往前缩小了一部分
- 修复 nav Tab 组件关于[页面一(外部页面)]，在 tab 标签上右键刷新，就会出现[数据加载中，请稍等...]的 loading 的效果，但不会自动关掉。所有运用 iframe 的页面同样出现此问题的 bug

V1.2 Final

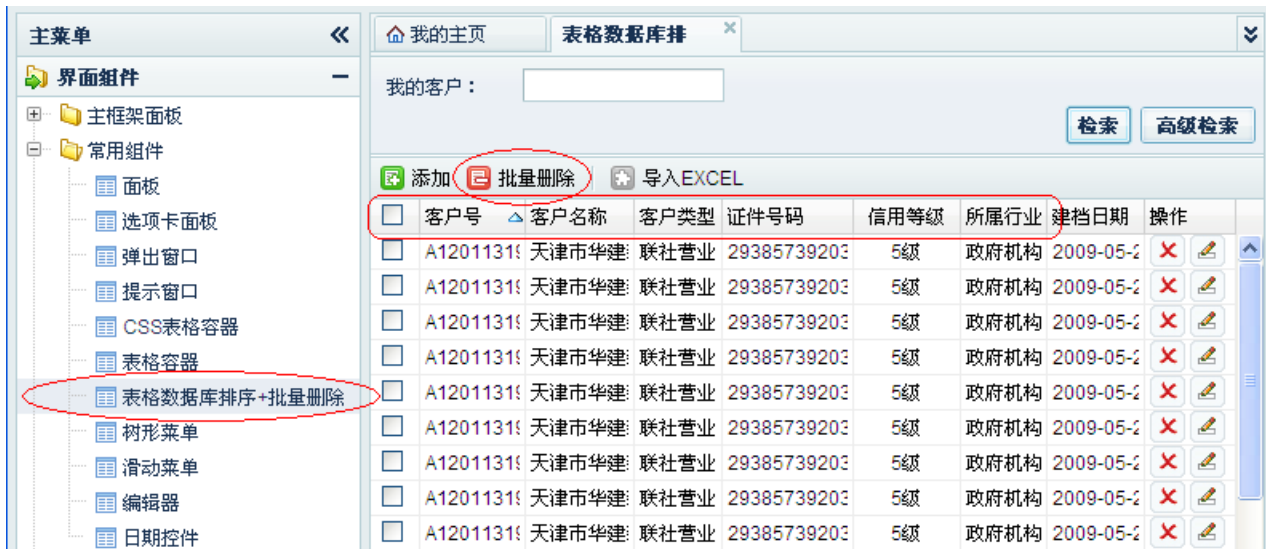
- 添加新主题风格 azure
- 添加 lookup 调用的 dialog 设置 resizable 和 maxable
- green 和 purple 主题的 tree 和 formBar 样式不正确
- 一个页面多个 combobox，在加载的时有几率使两个 combobox 变为相同问题
- combobox 不能用 validation 验证问题
- 解决在 form 表单的<p></p>中使用如果使用 combobox 会造成位置不正确

V1.2 RC1

- 使用隐藏 iframe 来处理无刷新表单提交时，服务器端返回 json 格式和普通 DWZ 普通 ajax 表单提交保持一致（即 validateCallback 和 iframeCallback 服务器端返回 json 格式一致）。具体细节请参考“文件上传表单提交”部分
- 新增关联对象查找带回组件 lookup

部门名称：

- 修改了 dwz.stable.js 解决了 table 表格组件的标题，拉动后，会和下面的记录错位问题。
- 新增表格组件多选批量删除功能
- 新增表格组件点击表头数据库排序功能



- 调整table 表格组件默认宽进和普通的 html table 保持一致。
- table 表格组件添加 TD 内容超大时是否多行显示控制, nowrapTD="false" 时 TD 可以自动换行
`<table class="table" layoutH="138" nowrapTD="false" width="100%">`
- 解决切换主题后，左边的菜单，左右拉动 IE 下失效问题。
- 修复日历控件当日期格式不匹配时初始化失败问题，格式错误时默认为当前日期。
- 解决在 ie 下页面有 xheditor 编辑器时，经过多次编辑后，文本框失效，不能输入问题。

V1.1.6 Final

DWZ 中 jQuery 版本从 1.4.2 升级到 1.4.4

navTab 组件重复打开同一个页面时是否重新加载数据控制: navTab.openTab(tabid, url, { title:"New Tab", fresh:false, data:{} });

解决 dwz.combox.js 中的 select 把 jquery 中的 select 冲突问题

V1.1.6 RC3

日历控件添加自定义选择时间控制功能。

组件 navTab 支持打开外部连接，navTab 组件自动判断如果是外部连接就用 iframe 方式打开。

修复 tab 组件和 inputAlert 组件冲突问题。

xhEditor 升级到最新版本。

V1.1.6 RC2

解决 Input alt 扩展和必填字段 class="required" 冲突问题

修复 uploadify 打开多个 navTab 时出现多个 upload 按钮

修复 table 组件数据量多的时候 调整这个列宽时, IE 下提示 “是否停止脚本运行”

checkbox 全选、反选示例。（demo → 表单组件 → 多选框/单选框）

Tree 组件优化，增加 checkbox 属性 checked，表示 checkbox 默认状态是否 checked，

修改 select combox 组件的默认样式

V1.1.6 RC1

此版本对应的 dwz_thinkphp-1.0RC1, 可以结合 dwz_thinkphp 版本去理解 DWZ 和服务端端的交互方式

DWZ.init() 方法添加 debug 状态，用于 DWZ.debug()

添加 jquery.uploadify 文件上传 HTML 扩展

HTML 扩展方式 navTab, dialog, ajaxTodo 的 url 支持变量替换。例如：`URL /edit/id/{sid_user}`

Table 组件修复切换 navTab 延时问题

添加 dwz.checkbox.js 用于 checkbox 全选、反选

添加 combox 下拉菜单组件(支持多级联动)

V1.1.5 Final

解决 jQuery1.4.2 与 jquery.validate.js 在 IE6 下兼容问题，jQuery 版本升级到 1.4.2

修复 dialog 内容无法复制问题

dialog 弹出后默认居中

添加 session 超时控制选择，跳转到“登录页面”或弹出带屏蔽层的“登录对话框”

navTab 的 openTab(tabid, title, url, [data]) 接口添加 data 参数，并调换 title 和 url 位置

V1.1.5 RC3

navTab 右键菜单添加“刷新标签页”

修复 google 浏览器中日历控件 icon 错位问题，和 button 字体错位问题

修复在弹出窗口 再弹出一个窗口是，新弹出的窗口被遮住问题

V1.1.5 RC2

修复 IE6 下 ajaxTodo 成功后关闭当前 navTab 时 js 出错问题

添加 CSS Table: 原生 html + CSS 实现，无 js 处理效果、最简单、最基本、性能最高的 table。

添加国际化 dwz.regional.zh.js，删除 dwz.validate.zh.js

DWZ 打包 JS，dwz.min.js

V1.1.5 RC1

修复 panel 折叠效果 IE 下错位问题

修复 DWZ 日历控件 IE6 下被 input 和 select 覆盖问题

V1.1.5 Beta1

添加 panel 折叠效果

添加 DWZ 日历控件

V1.1.4 Final

Tree 添加控制默认展开/收缩控制。

jQuery1.4.2 和 jquery.validate.js 在 IE 有兼容问题，ajax 表单提交在 IE 不能触发 form onsubmit 事件。导致 form 提交后跳转到了一个白页面，还原到 jQuery1.3.2

解决 v1.1.3 dialog 上的分页问题。

V1.1.3

修复了一些 v1.1.2 版本 ajax 载入 bug

添加了分页组件

V1.1.2

修改框架初始化方法，添加回调函数来保证，在初始化UI组件之前先载入dwz.frag.xml

```
DWZ.init("dwz.frag.xml", function() {  
    initEnv();  
    $("#themeList").theme({themeBase:"themes"});  
});
```

修复 IE6 下 alertMsg 问题

当前 dialog 添加 reload 方法：\$.pdialog.reload(url, params)

V1.1.1

增加当前 navTab 中链接 ajax post 扩展功能 ajaxTodo

修复 dialog 在 IE 下托动，dialog 中内容自动全选问题

修复 tree 组件折叠图标 bug

修复当前 navTab 上分页通用方法 navTabPageBreak 问题

修复当前navTab上分页跳转通用方法 navTabPageJump 问题

修复navTab中的 table HTML 扩展问题

v1.1.0

增加自定义鼠标右键菜单库 dwz.contextment.js

右键菜单定义在 dwz.frag.xml 文件中

navTab 右键菜单功能

```
<_PAGE_ id="navTabCM"><![CDATA[  
<ul id="navTabCM">  
    <li rel="closeCurrent">关闭标签页</li>  
    <li rel="closeOther">关闭其它标签页</li>  
    <li rel="closeAll">关闭全部标签页</li>  
</ul>  
]]></_PAGE_>
```

taskbar 右键菜单功能

```
<_PAGE_ id="dialogCM"><![CDATA[
<ul id="dialogCM">
    <li rel="closeCurrent">关闭弹出窗口</li>
    <li rel="closeOther">关闭其它弹出窗口</li>
    <li rel="closeAll">关闭全部弹出窗口</li>
</ul>
]]></_PAGE_>
```

v1.0.6

增加 Javascript 混淆和 gzip 压缩

增加银灰色主题风格

修复左边活动面板滑动问题

v1.0.5

增加 Dialog 默认大小设置功能.

Html 标签扩展方式

```
<a class="button" href="demo_page1.html" target="dialog" rel="dlg_page1" title="[自定义标题]"
width="800" height="480">打开窗口一</a>
```

JS 调用方式

```
$.pdialog.open(url, dlgId, title, {width: 500, height: 300});
```

navTab 浏览器前进后退按钮控制

ajax 前进后退控制,DWZ navTab浏览器前进后退功能控制.

增加文件上传表单提交方式演示页面

典型页面 → 文件上传表单提交示例

附录

附录一 Firebug介绍

Firebug 是一个 Firefox 插件。

Firebug 可以直接在 firefox 中修改、调试、查看 CSS, HTML, and JavaScript。

Firebug 也可以跟踪 ajax 请求，可以查看 ajax 请求的 request 和 response 信息。



补充说明和常见问题(xiaosuiba)

作者: xiaosuiba (xiaosuiba@qq.com)

2011 年 6 月

1、我如何在项目中使用 dwz?

手册中有如下说明:

设计思路

第一次打开页面时载入界面到客户端,之后和服务器的交互只是数据交互,不占用界面相关的网络流量.

支持HTML扩展方式来调用DWZ组件.

标准化Ajax开发,降低Ajax开发成本.

也就是说,只需要在一个页面(通常是起始页,如 index.aspx/index.php)包含框架,这里的框架是指 demo 中 index.html 页面的所有元素(<div class="page"可自定义),完整的 html 结构.其它的页面只需要页面碎片,就是 <body></body>中的部分.

2、怎样初始化 dwz?

dwz 是通过 init 函数初始化, index.html 的初始化函数如下:

```
$(function(){
    DWZ.init("dwz.frag.xml", {
        // loginUrl:"loginsub.html", loginTitle:"登录", // 弹出登录对话框
        loginUrl:"login.html", // 跳到登录页面
        statusCode:{ok:200, error:300, timeout:301}, // 【可选】
        pageInfo:{pageNum:"pageNum", numPerPage:"numPerPage", orderField:"orderField", orderDirection:"orderDirection"}, // 【可选】
        debug:false, // 调试模式, 【true|false】
        callback:function(){
            initEnv();
            $("#themeList").theme({themeBase:"themes"});
        }
    });
});
```

dwz.init(pathToDwz_frag_xml, options);

pathToDwz_frag_xml: js 方式能访问到的 dwz.frag.xml, 一定要保证通过这个地址能访问到 dwz.frag.xml 文件, 最好用 serverPath+dwz.frag.xml 的绝对路径方式.

options 是一个对象方式的参数:

loginUrl:当 ajax 的 json 返回 timeout 的时候会跳转到此页面

statusCode:自定义的 json 错误代码, 如果不指定将使用图片所示的默认规则.

pageInfo:这里可以为 pagerForm 指定别名, 比如 pageNum:"currentPage".

Callback:指定初始化完成后的回调函数.有人问如何在打开 dwz 的时候在我的主页加载另一个页面或者打开一个 navTab, 就可以在 callback 里使用 navTab.open("main"), \$.pdialog.open 等等.

3、我如何解析 json 数据来重绘表格/页面?

很多人不明白 dwz 的工作方式, 认为 dwz 的 navTab 的页面看起来是 ajax 方式解析的, 那就要从服务器传回 json 再手动解析. dwz 事实上是这样工作得, 只是他传回的不是一部分数据, 而是整个页面, 然后通过 JSON.eval()加载到 navTab 上, 这个过程对使用者是透明的, 也即你不需要关心页面的数据处理, 以前怎么写的页面, 现在还是怎么写页面. dwz 会将普通请求转换为 ajax 方式(前提是正确使用 dwz 提供的接口).

4、如果不是传输数据, dwz 的 json 是用于哪里?

dwz 的服务器端响应上提到一个服务器端响应 json，很多初学者问这个 json 如何传递数据，用于自己拼接页面等。如第 3 点所说，dwz 的页面是不需要手动处理 ajax 的，这个 json 结构是对 ajax 表单提交、ajax post 链接 (ajaxTodo) 状态的响应，而不涉及具体的页面数据。

5、提交表单或者 ajax post 链接后，如何刷新本 navTab？

一定要记住在返回的 json 中加上要刷新的 navTabId。

6、如何在 ajax 连接扩展中使用回调函数？

手册上没有写，其实从 1.2RC1 开始，ajax link 就有了 callback 属性，用于指定回调函数，如<a target="ajaxTodo" callback="MyOwnFunction"。

7、如何使用 table 和 css table 的排序功能？

Table 的排序功能是手册中没有提到的，其实 dwz 的排序功能相当强大，这里我简单介绍一下流程：

- 1、给要排序的表格<table 中加上 asc="asc" desc="desc"，指定排序别名。
- 2、给要排序的表格表头 th 加上 orderField="fieldName" 属性，这样点击该表头才能出发提交事件。Th 的 class="asc"/class="desc" 会分别显示向上和向下的箭头，这个不是只显示这么简单，往下看。
- 3、在 pageForm 加上 orderField 和 orderDirection，点击排序后提交的依然是 pagerForm，orderField 会绑定点击的 th 的 orderField，而 orderDirection 则会反向绑定 th 的 class，这是 dwz 智能的地方，也就是你不用手动记住状态来反向（感谢细心的作者），class="asc" 就会提交 orderDirection = desc。注意每次要将 orderDirection 绑定回 th 的 class。

8、如何使用 table 和 css table 的分页功能？

分页功能是大家用得比较多，也是不容易理解的一点。这里我凭着自己的理解给初学者讲讲。

手册上讲得很清楚，dwz 不是客户端分页，而是服务器端分页，结合本文第 3 点可以知道 dwz 的分页就是每次将分页数据提交回后台，后台生成分页数据显示到页面上。需要注意的是以下几点：

- 1、分页只需要 pagerForm 与 pagination 两个 dwz 组件，点击分页提交的是 pagerForm。
- 2、pagerForm 用于带查询的分页数据的缓存，说缓存是因为这里的参数都需要自己手动从后台读取绑定（pageNum 除外）。
- 3、Pagination 可以理解为一个页码生成器，他需要 totalCount="200" numPerPage="20" pageNumShown="10" currentPage="1" 几个参数来显示，每次后台需要绑定这几个参数，dwz 不会帮你做什么事情，也就是你想他显示第几页就是第几页。
- 4、点击分页，dwz 将 pagination 的 currentPage 绑定到 pagerForm 的 pageNum，然后提交 pagerForm 到后台。
- 5、初学者可以做这件事情来帮助理解：页面只放 pagerForm 和一个 div 和 pagination，div 每次显示当前的 pageNum 和 pagination，点击分页来好好体验以下这个过程，这对第三点的理解也有帮助。

9、如何使用输入表单客户端验证？

客户端表单验证也是手册上的 js 库介绍中的 dwz.regional.zh.js 一节有介绍。结合 demo 中的 demo_page4.html，简单明了。

10、如何使用 combox 的 ajax 联动？

手册上提及的联动方式，稍微扩展一下就可以达到 ajax 联动的目的。在 select 的 change 时间中，根据当前选择请求下一级的数据就可以了。

11、如何做局部刷新？

1.3 版中已经具有局部刷新功能，手册解释如下：

DWZ局部刷新怎样做？

API调用方式：

```
$("#xxxId").loadUrl(url,data,callback);
```

html扩展链接方式：

```
<a href="url" target="ajax" rel="xxxId"></a>
```

1.2 版可用 loadUrl 方式。

12、如何去掉 dwz 调用 ajax 方法出现的等待图片

```
dwz.ui.js

var ajaxbg = $("#background,#progressBar");
ajaxbg.hide();
$(document).ajaxStart(function() {
    ajaxbg.show();
}).ajaxStop(function() {
    ajaxbg.hide();
});
```

也可以把自己的\$.ajax 的 globe 设置为 false 来屏蔽 ajaxStart 方法。

13、最后谈谈我的看法（xiaosuiba）

dwz 作为一个开源的 ajax 前端框架，为广大的 web 开发者提供了极大的方便，这点对初学者可能还没有这么深刻，不过一些老程序员可是感动得一塌糊涂。可以看出，这个框架倾注了作者极大的心血，而大家的热情就是对这种奉献精神的最好回报。本人接触 dwz 不过区区 1 个月，但是一直坚持和大家一起讨论问题，就是希望有越来越多的人能够使用 dwz，这样才能使其具有长久的生命力。

对于 web 开发老手来说，dwz 很容易上手，对于新手，我想提几条建议：

- 1、先要有基础的 web 知识，手上常备 js 手册和 dwz 手册。
- 2、遇到问题先试着从手册和 demo 里寻找解决办法，尽量不要问手册中已经存在的问题，相信没有人会喜欢一遍一遍回答诸如 navTab 是什么，navTabId 是什么之类的问题。
- 3、相信 dwz 能够用于项目，现在已经有人成功了。所以放手去做吧。
- 4、尽量使用最新版本，作为一个开源项目，dwz 更新是频繁的，通过这种更新来消除 bug，同时引入新的特性。所以请升级你的版本到最新稳定版本或 RC 版本。