

DWZ 富客户端框架使用手册

目录

- 概述 4
 - 设计思路 4
 - 学习 DWZ 的建议 4
 - DWZ 区别于其它 JS 框架，最大的优点 4
 - 版权声明 5
 - DWZ 研发组介绍 5
- HTML 扩展 6
 - Ajax 链接扩展 6
 - 当前 navTab 中链接 ajax post 扩展 6
 - dialog 链接扩展 6
 - navTab 链接扩展 7
 - Tab 组件扩展 8
 - Accordion 组件 8
 - 容器高度自适应 8
 - CSS Table 9
 - Table 扩展 9
 - 在线编辑器 10
 - 分页组件 10
 - navTabTodo 扩展 11
 - Input alt 扩展 11
 - Tree 扩展 11
 - Panel 扩展 12
 - 日历控件 12
 - url 变量替换 13
 - checkbox 全选、反选 14
 - uploadify 多文件上传 14
 - combox 组件 15
- Ajax 表单 16
 - 表单查询 16
 - 普通 Ajax 表单提交 18
 - 服务器端响应 18
 - 文件上传表单提交 20

服务器端响应	20
Java 服务器端表单处理示例.....	20
DWZ js 库介绍.....	22
DWZ 框架初始化.....	22
dwz.core.js.....	22
dwz.ui.js.....	22
dwz.ajax.js	22
dwz.alertMsg.js.....	22
dwz.jDialog.js.....	22
dwz.accordion.js	22
dwz.barDrag.js	22
dwz.navTab.js	23
dwz.scrollCenter.js.....	23
dwz.stable.js	23
dwz.cssTable.js.....	23
dwz.tree.js	23
dwz.theme.js	23
dwz.validate.method.js.....	23
dwz.validate.zh.js.....	23
dwz.contextmenu.js.....	23
dwz.pagination.js	24
dwz.datepicker.js	24
dwz.combox.js	24
dwz.checkbox.js	24
dwz. regional.zh.js.....	24
dwz.validate.method.js.....	24
Javascript 混淆和压缩.....	25
Javascript 混淆	25
Javascript 用 gzip 压缩.....	26
DWZ 如何中使用打包的 js.....	26
常见问题及解决	27
Error loading XML document: dwz.frag.xml.....	27
IIS 不能使用 Ajax 解决方案	27
多个 navTab 页面或 dialog 页面 ID 冲突，解决方法	27
jQuery1.4.2 和 jquery.validate.js 在 IE 的兼容问题	27
升级 jQuery1.4.2 注意事项	27
weblogic 访问 xml 问题.....	28

DWZ 版本升级	28
版本命名规则	28
V1.2 RC1	28
V1.1.6 Final	29
V1.1.6 RC3	29
V1.1.6 RC2	29
V1.1.6 RC1	29
V1.1.5 Final	30
V1.1.5 RC3	30
V1.1.5 RC2	30
V1.1.5 RC1	30
V1.1.5 Beta1	30
V1.1.4 Final	30
V1.1.3	31
V1.1.2	31
V1.1.1	31
v1.1.0.....	31
v1.0.6.....	31
v1.0.5.....	32
附录	33
附录一 Firebug 介绍.....	33

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://dwz.duqn.com>

在线文档 <http://dwz.duqn.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 和服务器端的交互方式

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 研发组介绍

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。

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。

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) msn:duqn@hotmail.com QQ:8560685
吴平 (Ajax开发) msn:wupinggone@hotmail.com QQ:465046815
张慧华 (Ajax开发) msn:zhanghuihua@msn.com QQ:350863780
jQuery.DWZ-UI超级群(满员) 69611933
jQuery.DWZ-UI-1群(满员) 107983317
jQuery.DWZ-UI-2群(满员) 369203
jQuery.DWZ-UI-3群(欢迎加入) 20866231

HTML 扩展

支持 HTML 扩展方式来调用 DWZ 组件

Ajax 链接扩展

``

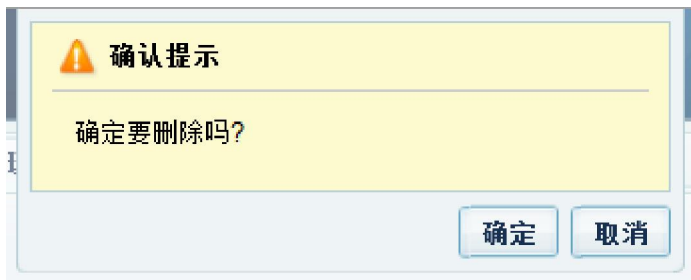
示例: `提示窗口`

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 `navTabTodo` 时会自动关联如下 JS。

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

`打开窗口一`

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
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,dlid) 刷新dlid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

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>
```



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

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 `layout="xx"`, 单位是像素。

Layout 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 layout="150">内容</div>`

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 class="tableList" layoutH="xx">` 实现 table 固定高度

```
<div class="tableList" 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>北京市政府哟呀哟</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-->
  <!-- 【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>
```

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。

navTabTodo 扩展

navTab 页面上 a 链接添加 `target="navTabTodo"` 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=${item.id}" target="navTabTodo" title="确定要删除吗?">>
删除</a>
<a href="/news.do?method=publish&id=${item.id}" target="navTabTodo">发表</a>
```

框架自动绑定js

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

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]]" onclick="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" [pattern="yyyy-MM-dd"] [yearstart="-80"]
[yearend="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	

示例:

```
<p>
  <label>默认格式: </label>
  <input type="text" name="date1" class="date" />
  <a class="inputDateButton" href="#">选择</a>
</p>
<p>
  <label>定义年份: </label>
  <input type="text" name="date2" class="date" yearstart="-80" yearend="5"/>
  <a class="inputDateButton" href="#">选择</a>
</p>
<p>
  <label>自定义日期格式: </label>
  <input type="text" name="date3" class="date" pattern="yyyy/MM/dd" />
  <a class="inputDateButton" href="#">选择</a>
</p>
<p>
```

```

<label>自定义日期格式: </label>
<input type="text" name="date5" class="date" pattern="dd/MM/yy" />
<a class="inputDateButton" href="#">选择</a>
</p>
<p>
<label>自定义日期+时间: </label>
<input type="text" name="date7" class="date" pattern="yyyy年MM月dd日 HH:mm:ss" />
<a class="inputDateButton" href="#">选择</a>
</p>
<p>
<label>自定义日期+时间: </label>
<input type="text" name="date8" class="date" pattern="yyyy年MM月dd日 HH:mm" />
<a class="inputDateButton" href="#">选择</a>
</p>
<p>
<label>自定义日期+时间: </label>
<input type="text" name="date9" class="date" pattern="y年M月d日 HH点" />
<a class="inputDateButton" href="#">选择</a>
</p>

```

url 变量替换

HTML 扩展方式 navTab, dialog, navTabTodo 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:

 新增  删除  编辑  修改密码			
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="__URL__" rel="{sid_user}">`

当选中一行时, `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="__URL__"`保持一致.

代码示例:

```

<a class="delete" href="__URL__/foreverdelete/id/{sid_user}/navTabId/__MODULE__"
target="__URL__" 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, change, param, value。

change 属性替换原有的 onchange 属性, 功能与原有 onchange 属性一样, 定义了两种方式向 change 所指定的函数传参数, 一种是直接传所选择的 option 的 value, 例: change(value), 另一种是以 json 格式传参数, param 属性即是用来定义此 json 的 key, 例: change({pname:value}); 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 name 来赋值。value 属性

则是用来指定 combox 的默认值。级联时 change 指定的函数的返回值需是一个二维数组，如：new Array(["value","text"], ["value","text"]...);

以下则级联示例：

```
<select class="combox" name="province" ref="city" change="setCity" value="北京">
  <option value="">所有省市</option>
  <option value="北京" selected="selected">北京</option>
  <option value="上海">上海</option>
  <option value="天津">天津</option>
  <option value="重庆">重庆</option>
  <option value="广东">广东</option>
</select>
<select class="combox" name="city">
  <option value="">所有城市</option>
</select>
```

非级联 json 传值示例：

```
<select class="combox" name="numPerPage" change="navTabPageBreak" param="numPerPage">
  <option value="20">20</option>
  <option value="50">50</option>
  <option value="100">100</option>
  <option value="200">200</option>
</select>
```

```
/**
 * 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());
}
```

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)">
```



```

/**
 * 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>
    </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": "",
    "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 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>

```

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":"",
    "callbackType":"closeCurrent",
    "forwardUrl":""
}

```

Java 服务器端表单处理示例

```

public class UserAction extends BaseAction {

    public ActionForward changePwd(ActionMapping mapping, BaseActionForm bForm,
        HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        UserForm form = (UserForm) bForm;

```

```

        if (form.getOldPassword() == null
            || "".equals(form.getOldPassword().trim())) {
            request.setAttribute("statusCode", 300);
            request.setAttribute("message", this.getMessage(request,
                "msg.oldpassword.invalid"));
            return mapping.findForward("done");
        }

        User user = AppContextHolder.getContext().getUser();

        try {
            UserManager uManager = BusinessFactory.getFactory()
                .getUserManager();
            uManager.changePassword(user, form.getOldPassword(), form
                .getPassword());
            request.setAttribute("statusCode", 200);
            request.setAttribute("message", this.getMessage(request,
                "msg.operation.success"));
        } catch (PasswordNotCorrectException e) {
            request.setAttribute("statusCode", 300);
            request.setAttribute("message", this.getMessage(request,
                "msg.password.incorrect"));
        }

        return mapping.findForward("done");
    }

    protected ActionForward callbackForward(ActionMapping mapping,
        HttpServletRequest request) {
        if (ServerInfo.isAjax(request))
            return mapping.findForward("done");
        return mapping.findForward("callbackDone");
    }
}

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.

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.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.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

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 不能使用 Ajax 解决方案

Ajax 访问*.htm 或是*.html 后缀的网页在 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 访问*.htm 或是*.html 后缀的网页,于是将请求的页面后缀改为.php,SUCCESS!

现在总结一下,IIS 在使用 Ajax 请求网页时,一定要动态网页后缀的! 这是 IIS 的问题, 不是框架 bug。

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,:{}\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*\[)/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weblogic就给加上contentType了

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。

版本命名规则

Alpha 版

alpha 演示版就是比 beta 还早的测试版，一般都是内部测试的版本。

Beta 版

公开测试版，这个阶段的版本会一直加入新的功能。

RC 版

RC 候选发布版本 接近正式版了，Candidate 是候选人的意思，用在软件上就是候选版本。Release.Candidate. 就是发行候选版本。和 Beta 版最大的差别在于 Beta 阶段会一直加入新的功能，但是到了 RC 版本，几乎就不会加入新的功能了，而主要着重于除错！

Final 版

最终版，正式版本

V1.2 RC1

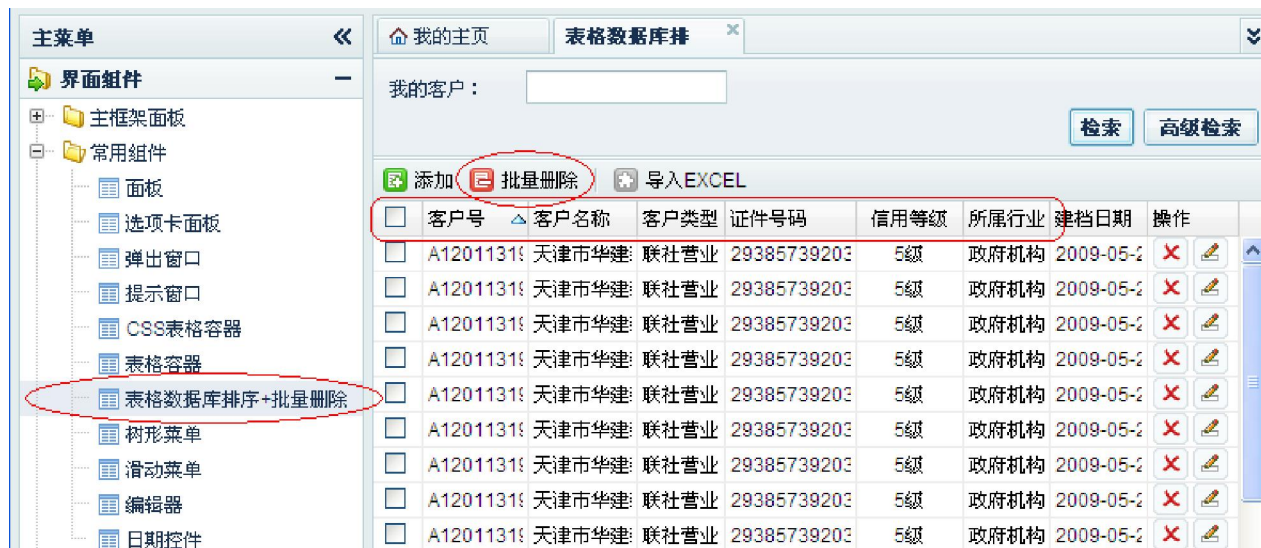
1. 使用隐藏 iframe 来处理无刷新表单提交时，服务器端返回 json 格式和普通 DWZ 普通 ajax 表单提交保持一致（即 **validateCallback** 和 **iframeCallback** 服务器端返回 json 格式一致）。具体细节请参考“文件上传表单提交”部分
2. 新增关联对象查找带回组件 lookup

部门名称：

人事部 

3. 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。
4. 新增表格组件多选批量删除功能

5. 新增表格组件点击表头数据库排序功能



6. 调整 table 表格组件默认宽进和普通的 html table 保持一致。
7. 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%">`
8. 解决切换主题后, 左边的菜单, 左右拉动 IE 下失效问题。
9. 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。
10. 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, navTabTodo 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 navTabTodo 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 navTabTodo

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.contextmenu.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。

