DWZ 富客户端框架使用手册

目录

概述 3

DWZ区别于其他的js库，最大的优点 3

设计思路 3

版权声明 3

DWZ研发组介绍 3

HTML扩展 5

Ajax链接扩展 5

当前navTab中链接ajax post扩展 5

dialog链接扩展 5

navTab链接扩展 5

Tab组件扩展 6

容器高度自适应 6

Table扩展 6

在线编辑器 7

分页组件 7

navTabTodo扩展 8

Input alt扩展 8

Tree扩展 9

Ajax表单 9

表单查询 9

普通Ajax表单提交 10

文件上传表单提交 12

Java服务器端表单处理示例 12

DWZ js库介绍 14

DWZ框架初始化 14

dwz.core.js 14

dwz.ajax.js 14

dwz.alertMsg.js 14

dwz.jDialog.js 14

dwz.barDrag.js 14

dwz.navTab.js 14
dwz.scrollCenter.js 14
dwz.stable.js 15
dwz.tree.js 15
dwz.theme.js 15
dwz.ui.js 15
dwz.validate.method.js 15
dwz.validate.zh.js 15
dwz.contextmenu.js 15
dwz.pagination.js 16
Javascript混淆和压缩 17
Javascript混淆 17
Javascript 用gzip压缩 17
常见问题及解决 18
Error loading XML document: dwz.frag.xml 18
IIS不能使用Ajax解决方案 18
jQuery1.4.2 和jquery.validate.js在IE的兼容问题 18
升级jQuery1.4.2 注意事项 18
weblogic访问xml问题 19
DWZ版本升级 19
V1.1.4 19
V1.1.3 19
V1.1.2 19
V1.1.1 19
v1.1.0 20
v1.0.6 20
v1.0.5 20

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。 让开发人员不写javascript的情况下，也能用ajax做项目和使用各种UI组件.

DWZ框架支持用html扩展的方式来代替javascript代码，只要董html语法，再参考DWZ使用手册就可以做ajax开发.
DWZ框架基本可以保证程序员不董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做定制化开化.

一般做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也不是太熟，大大影响了开发速度. 使用DWZ就解决这一问题，使用DWZ框架自动邦定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使用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/

DWZ区别于其他的js库，最大的优点

- 简单实用，扩展方便，轻量级框架
- 仍然保留了 html 的页面布局方式
- 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

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, 不会占用界面相关的网络流量.
- 支持 HTML 扩展方式来调用 DWZ 组件.
- 设计目标是简单实用扩展方便.
- 标准化 Ajax 开发, 降低 Ajax 开发成本.

版权声明

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 UI 设计经验。

吴平主要做 Java web 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，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 组件都是从我们做过的大量 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
DWZ QQ群 369203 107983317

HTML扩展

支持 HTML 扩展方式来调用 DWZ 组件

Ajax链接扩展

<a href=”xxx” target=“ajax” [rel=“boxId”]>

示例：提示窗口

当前navTab中链接ajax post扩展

```
<a href="user.do?method=remove" target="navTabTodo">删除</a>
```

dialog链接扩展

<a href=”xxx” target=“**dialog**” [rel=“**dialogId**”]>

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：

```
<a href="w_dialog.html" target="dialog" rel="page2">弹出窗口</a>
```

或

```
<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>
```

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, {width: 500, height: 300});
```

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。
```

navTab链接扩展

<a href=”xxx” target=“navTab” [rel=“tabId”]>

示例：提示窗口

Tab组件扩展

开发人员不需写任何 javacsript, 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>
```

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: 内容

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>
```

分页组件使用方法：

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

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

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】

示例：

```
<a href="/news.do?method=remove&id=${item.id}" target="navTabTodo">删除</a>
<a href="/news.do?method=publish&id=${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]]">
	<li><a href="#">第一级菜单项 A</a>
		<ul>
			<li><a href="#">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<ul>,<li>的嵌套格式构成，将最顶级的<ul>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，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。

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)">
```

navTabSearch()用于 navTab 页面上的查询，会重新再入当前 navTab

dialogSearch()用于 dialog 弹出层上的查询, 会重新再入当前 dialog

```
function navTabSearch(form){
	navTab.reload(form.action, $(form).serializeArray());
	return false;
}
function dialogSearch(form){
	$.pdialog.reload(form.action, $(form).serializeArray());
```

```
	return false;
}
```

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 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做一些其它处理也可以自定义一个回调函数. 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, navTabAjaxDone)"`

```
/**
 * 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tab标记为reloadFlag=1, 下次切换到那个tab时会重新载入内容.
 * callbackType如果是closeCurrent就会关闭当前tab
 * 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="/userAction?method=save" onsubmit="return validateCallback(this,
navTabAjaxDone)">
 *
 * form提交后返回json数据结构statusCode=200表示操作成功, 做页面跳转等操作. statusCode=300表示操作失
败, 提示错误原因.
```

```
 * {"statusCode":"200", "message":"操作成功", "navTabId":"navNewsLi", "forwardUrl":"",
"callbackType":"closeCurrent"}
 * {"statusCode":"300", "message":"操作失败"}
 */
function navTabAjaxDone(json){
	ajaxDone(json);
	if (json.statusCode == 200){
		if (json.navTabId){
			navTab.reloadFlag(json.navTabId);
		} else {
			navTabPageBreak();
		}

		if ("closeCurrent" == json.callbackType) {
			navTab.closeCurrentTab();
		} else if ("forward" == json.callbackType) {
			navTab.reload(json.fowrwardUrl);
		}
	}
}


/**
 * 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=200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 == 200){
		if (json.navTabId){
			navTab.reload(null, {}, 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"
onclick="navTab.closeCurrentTab()">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 ifarmeCallback(this);">
```

或

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

服务器端需要返回

```
<script type="text/javascript">
      var statusCode="${statusCode}";
      var message="${message}";
      var navTabId="${navTabId}";

      var response = {statusCode:statusCode,
                  message:message,
                  navTabId:navTabId
            }
      if(window.parent.donecallback) window.parent.donecallback(response);
</script>
```

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", 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.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}, 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

dwz.barDrag.js

DWZ 左边的活动面板

dwz.navTab.js

导航 tab 组件

dwz.scrollCenter.js

页面容器自动居中组件

dwz.stable.js
table 组件

dwz.tree.js
tree 组件

dwz.theme.js
切换界面主题风格

dwz.ui.js
页面效果初始化，html 扩展绑定 js 效果

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(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

```
<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 , 不用写 js, 框架自动绑定处理事件。

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 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,现在我想到的解决辨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

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 laod 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。

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。

DWZ1.1.4 版本还原到 jQuery1.3.2

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paseJSON()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。
```

```
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*\[)+/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weblgoic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** 全部覆盖低版本的 js 和 **dwz.frag.xml** 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。

V1.1.4

此版本是目前最稳定版。

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.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