Diigo社会化标注功能的本地实现探讨

张 泽[1],张 彤[2]
(1.苏州大学 计算机学院,江苏 苏州 215006;2.中国联通江门分公司,广东 江门 529000)

摘要:通过分析Diigo的社会化标注功能的工作流程,论证了其移植到本地服务器的可行性,给出了本地系统模型。通过使用Ajax和Web Service实现了在企业应用中使用"可标注网页",并给出了具体实现的原型实例。

关键词:社会化书签;社会化标注;Diigo;Ajax;web服务
中图分类号:TP391.4 **文献标识码**:A **文章编号**:1671-5322(2008)03-0030-04

随着Web2.0技术的发展"可写网络"逐步变为现实。这类应用都是由第三方处理和保存数据,当这些技术被应用于企业社团、电子政务等方面,或是内网时,安全、网络等问题使得原有方案无法实现[1-2]。本文对本地服务器实现社会化标注进行了论证分析,并用Ajax和Web Service初步构建了原型机。

1 本地实现社会化标注需求与原理分析

由于系统目标和需求基本明确,所以需求分析相对简单,下面仅简要列出主要功能需求:管理员通过简单设置可以选择是否将当前网页变为可标注;对可标注网页没有特殊要求,支持Html和Shtml等静态页;用户在操作标注前需要进行身份认证,只有合法并有权限的用户方可标注留言;对有合法身份的用户显示页面上已有的书签;当用户鼠标移到书签上,弹出原有的标注信息列表(可能是多条),且用户可继续添加新标注;用户在可标注的页面上选择文字区域后也要弹出标注对话框;用户填写完标注提交后,页面更新并出现新提交的书签;使用BS架构,用户使用标准浏览器即可访问,不安装工具条或其他客户端。

要在本地实现"可标注",最直接的想法就是把内容和标注模块合二为一,让内容页使用服务器脚本,使其自身完成标注功能,但这样作会有很多问题,页面频繁刷新的问题、网页实现上也因此受限,等等。而使用Ajax——通过JavaScript在后台处理数据交换,再操纵DOM和CSS呈现数据,这样刷新问题迎刃而解。

作为一个标注信息(下面成为对象),应包括的内容(属性)有:书签内容——用户选取的页面文字;标注内容——用户写的标注;标注者——标注作者;标注日期——标注日期;网页地址——网页地址;是否公开——标注是否公开;

……

另外需要考虑的约束条件有:每个选定的书签可以有不止1条标注页面呈现时要读取书签而不需要标注1个用户可以有多条标注。

根据上述条件,可以把标注对象分解为3个子对象,分别是书签、标注和用户,由此在数据库上产生有关标注的核心表,即书签信息表(highlight_info_tab)、标注信息表(annotation_info_tab)和用户表(user_tab),由于也忽略了成员身份、权限等管理部分,所以现在涉及的用户表部分只是从user_id找到对应的username,所以不再详述,图2是书签、标注和用户的对象关系图,表1、表2分别是书签信息和标注信息表定义。

表1 highlight_info_tab定义

Table 1 Definition of highlight_info_tab

字段名	类型	备注
highlight_id	int	自动书签编号
highlight_content	nvarchar(MAX)	书签内容
webpage_url	nvarchar(MAX)	网页URL
is_share	bit	是否共享

收稿日期:2008-06-07

作者简介:张泽(1970-),男,江苏苏州人,硕士研究生,主要研究方向为数据库、网络应用,嵌入式系统。

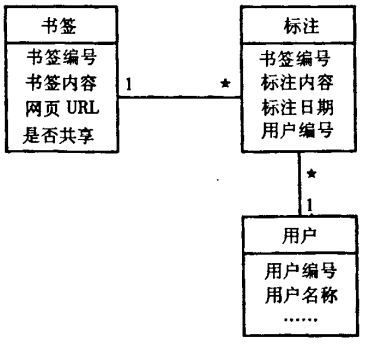

图 2　对象关系图

Fig. 2　Object Model Diagram

表 2　annotation_info_tab 定义

Table 2　Definition of annotation_info_tab

字段名	类型	备注
highlight_id	int	相应书签编号
annotation_note	nvarchar(MAX)	标注内容
annotation_date	datetime	标注日期
user_id	nvarchar(50)	用户编号

2　具体实现

由于篇幅限制，本文只对实现的关键技术原理进行探讨，所以省略了如成员身份认证，成员个人信息管理等部分，下面以普通的 Html 页支持标注为例加以说明：

```
<HTML >
<HEAD > <TITLE > 可标注页 </TITLE >
< script type = " text/javascript" src = " js/myjs. js" > </script >
</HEAD >
<BODY >
<h2 > 这是一个可标注网页 </h2 >
……
```

为简化实现，可使用 MSDN 上得到的 tool tip 和 Web Service2 个 JavaScript 增强 HTC 控件，在客户端调用 Web Service 和产生 Tool Tip 都已被封装成相应的 JavaScript 对象方法，只需调用即可。

服务器端代码中有 3 个 Web Method，分别用于书签查询（wm_getHighlight）、标注查询（wm_getAnnotation）和标注添加（wm_addHighlight），书签查询是用于根据客户端请求返回页面上的所有带书签的位置信息，标注查询是从数据库中查询某条书签的相应标注信息和标注添加则是添加新标注。

Web Service 用 Vs2008 编写，由于篇幅限制全部代码不再列出，下面对有代表性的 wm_getHighlight 方法解释如下：

```
[WebMethod]
public string wm_getHighlight(string szURL) {
string szResult = null;
using (SqlConnection conHighlight = new SqlConnection(
ConfigurationManager. ConnectionStrings[
"annotationDBConnectionString" ]. ConnectionString)) {
SqlDataAdapter daHighlight = new SqlDataAdapter(
" SELECT highlight _ id, highlight _ content FROM highlight_info_tab WHERE webpage_url = '" + szURL + "' ",conHighlight);
DataSet dsHighlight = new DataSet( );
daHighlight. Fill(dsHighlight);
szResult = dsHighlight. GetXml( );
}
return szResult;
}
```

wm_getHighlight 是客户端请求查询当前页的所有书签信息的 Web Method，参数是内容页的的地址字符串，数据库查询后会返回所有 Webpage_url 与参数相同的记录集，利用 DataSet 类的 GetXml，把这些记录集序列化为 xml 格式后返回，客户端的 js 通过 Ajax 的 XMLHttpRequest 对象就可以得到此 xml 序列，从而完成书签数据的读取过程。需要说明的是，在总体设计阶段，设定的传输流是 JSON，因为它更合适 JavaScript 操作，也是 Ajax 推崇的流格式，但在详细设计阶段发现，由于使用的 C#和 ASP. net 并没有操作 JSON 的类库，需要自己写相应的序列化和反序列化方法，如此增加了很多代码工作量。所以改用 xml 实现，这样只需一个 DataSet. GetXml( )，记录集就变出可以被传输的 xml 序列了。而如果是 JSP 或是新一代的 Silverlight，它们都有 JSON 类库，相信使用 JSON 时更好的选择了[3-4]。

wm_getAnnotation 与上述方法类似，只是参数不同。wm_addHighlight 使用 ADO. net 的 SqlCommand 类执行 SQL 的 update 语句，需要执行 2 次

update,分别写入2张表highlight_info_tab和annotation_info_tab(由于只是原型系统,所以只求功能的实现,在实际应用中此处建议用存储过程,这样效率、安全方面等就可以相应提高)。

在客户端,前述的myjs.js文件中有如下部分:

```
//添加样式表链接
document.write("<link rel='stylesheet' type=text/css' href=StyleSheet.css/>");
//添加id为oTipEl的控件,用于启动tooltip控件
document.write("<div id='oTipEl' class='undisplay'></div>");
//添加id为service的div,用于操作webservice
document.write("<div id='service' onresult='onWSresult()' class='undisplay'></div>");
//动态添加tooltip对话框到网页
document.write("<Tooltip:tip element='oTipEl' avoidmouse=false'>");
……//tooltip对话框表单部分
document.write("</Tooltip:tip>");
```

在StyleSheet.css中有如下2节:

```
Tooltip…tip { behavior: url(js/tooltip_js.htc)};
#service { behavior: url(js/webservice.htc)};
```

如此就把Tool Tip和Web Service这2个JavaScript控件加入了,特别的通过隐藏的Service层使用了Ajax技术与服务器进行异步通信,在onload事件响应中[5]:

```
window.onload = function(){
service.useService("..host adress/ws_Highlight.asmx? WSDL","ws_Highlight");
iCallID = service.ws_Highlight.callService("wm_getHighlight",document.URL);
//iCallID是全局变量,由于记录事件id
}
```

然后在id为Service的层上加了由Web Service控件内定义的onresult事件,其响应程序如下:

```
function onWSresult(){
if((!event.result.error) && (iCallID == event.result.id)) {
var doc = new ActiveXObject("Msxml2.DOMDocument");
//此处忽略浏览器兼容问题,实际应用不可省略
doc.loadXML(event.result.value);
nodes = doc.getElementsByTagName("Table");
if(nodes != null){
for(var i=0; i<nodes.length; i++){
var node = nodes[i].getElementsByTagName("annotation_note");
if(node == null){
node = nodes[i].getElementsByTagName("highlight_id");
……
}
else{
document.getElementById("anno_list")……
```

上述代码中的粗体部分,是被封装后的服务器返回值,其后台就是用作为AJAX核心的XMLHTTPRequest对象,详见MSDN Web Service Behavior。从event.result.value获得的就是前面Web Method封装的xml序列,剩余的处理相对简单,就不再赘述了。

由于标注对话框的产生有2种情况,一是鼠标移到已经Highlight的书签上,二是当选定文字,显然只用原来mouseover的响应是不够的,为保持原控件代码的完整性,我们不改.htc文件,而用发moseover消息的办法,让这2个事件发生时都能弹出标注对话框,代码如下:

```
document.body.onmouseup = function(){
//响应mouseup事件,保存选取的文字,并发出mouseover消息
if((document.selection)&&(document.selection.type == "Text")) {
var range = document.selection.createRange();
var str = AnalyzeWords(range.text);
document.getElementById("selectedText").value = null;
if(str.length > 0){
document.getElementById("oTipEl").fireEvent("onmouseover");
document.getElementById("selectedText").value = str;
```

```
}
}
}
function onHighlihtover(szHighlightID){
//书签上 mouseover 响应函数,发送 mouseover 和远程调用 wm_getAnnotation
document. getElementById ( " oTipEl " ). fireEvent( "onmouseover" );
service. useService( ".. host adress/ws_Highlight. asmx? WSDL","ws_Highlight" );
iCallID = service. ws_Highlight. callService( "wm_getAnnotation", szHighlightID);
}
```

经过以上过程"可标注网站"初具雏形,当然还有一些细节处理也比较重要,如用户在选取网页内容时,可能会出现"跨节选取"得问题,即选取的文字不在同一节点上,如果不加处理,会使存储的书签被截断,甚至出错。

3 总结

本系统作为对"社会化书签"、"社会化标注"的功能延伸,在电子政务、企业应用等领域具有一定应用前景。当然,因为本文是在理论分析了系统需求后给出的系统原型,虽然已能实现基本功能,但在细节上还很不完善,如用户管理、系统安全等都还需要进一步改进。

参考文献:

[1] 刘润东. UML 对象设计与编程[M]. 北京:希望电子出版,2001.

[2] 王沛,冯曼菲. 征服 Ajax—Web2.0 开发技术详解[M]. 北京:人民邮电出版社,2006.

[3] Microsoft MSDN - ToolTip Behavior[EB/OL]. http://msdn2. microsoft. com/en - us/library/ms531441(VS. 85). aspx.

[4] Microsoft, MSDN. Web Service Behavior[EB/OL]. http://msdn2. microsoft. com/en - us/library/ms531035(VS. 85). aspx.

[5] 李莹,黎加厚. Diigo 在教学中的应用初探[J]. 远程教育杂志,2007(5):16 - 20.

The Discussion for Local Host Realization of Diigo's Social Annotation

ZHANG Ze[1], ZHANG Tong[2]

(1. College of Computer Science, Soochow University, Jiangsu Suzhou 215006, China;
2. China Unicom Co., LTD, Jiangmen Branch, Guangdong Jiangmen 529000, China)

Abstract: Today, the expansion of Web 2.0 technologies has brought tremendous promotion to the Social Software. The hot line of Web applications is how to use these new technologies. This paper analyzes the work flow of Diego's Social Annotation, presents the feasibility of migrating to local server and gives the local system model. With the Ajax and Web service, this paper describes the detailed example for carrying out the annotated Web pages which fits to enterprise application.

Keywords: social bookmark; social annotation; Diigo; Ajax; Web Service