利用隐藏域实现PHP页面的事件机制

姜林美,詹玲超,黄继风
(上海师范大学,上海 200234)

摘 要:在开发网页和Web应用时,使用像用RAD工具开发Windows应用程序一样的事件处理机制来响应网页加载、按钮点击等事件会使得页面的程序结构规整清晰,提高页面代码的可读性和可重用性。文中提出了一个在PHP编程中非常简单易行的事件机制的实现方法,并给出了示例代码及注释。

关键词:PHP;事件;隐藏域;串行化;正则表达式
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2006)10-0068-03

PHP Event Mechanism Based on Hidden Field

JIANG Lin-mei, ZHAN Ling-chao, HUANG Ji-feng
(Shanghai Normal Univ., Shanghai 200234, China)

Abstract: It is well-known that the structure of a Web application coded with event mechanism to process some events such as loading a page, clicking a button and changing the selection of a drop-down box could be clearer, more readable and more reusable than the one coded simply with sequential sentences. So, a solution for implementing PHP event mechanism is brought forth in this paper; meanwhile, some illustrational codes and essential comments are provided.

Key words: PHP; event; hidden field; serialization; regular expression

0 引 言

使用RAD工具(比如Delphi, VB, VC)开发过Windows应用程序(事实上,使用RAD工具是现在程序员开发软件的首选)的程序员都知道应用程序在执行过程中会触发各种各样的事件,而他们(程序员)开发时所需要做的大多数事情就是编写这些事件被触发时所要执行的代码。而有过纯粹使用Windows消息处理机制进行Windows编程经验的程序员一定有过使用大量的switch...case语句来响应各种Windows消息的苦恼,因为这种方式相比之事件机制,程序员要处理的事情更多、更繁琐。

众所周知,动态网页是一种B/S结构,而且在客户机与服务器间是无连接状态的[1]。虽然在网页中也有各种各样的事件,但那仅限于在客户端响应的事件,这些事件并不被传递到服务器端,所以服务端对这些事件一无所知。然而,对于Web应用来说,客户端的数据往往要被提交到服务器端来处理,那么如何让服务器端来简便地接收客户端的数据处理请求呢?下面就来介绍如何利用HTML表单(FORM)的隐藏域以及正则表达式等技术来实现客户端到服务器端的事件处理机制。

收稿日期:2005-12-22
作者简介:姜林美(1976-),男,福建三明人,硕士研究生,主要从事图像及应用软件开发的研究;黄继风,副教授,研究方向为图形、图像处理;数字水印技术;地图匹配等。

1 基本思想

本方法的基本思想是:将客户端触发的事件串行化到表单的一个隐藏域,页面数据提交到服务器端后,服务器端反串行化(解析)隐藏域中的数据并形成相应的事件[2,3]。串行化事件到隐藏域的工作在客户端由一个Javascript类[4](ViewState)来完成,一个Javascript函数(VsEvent)用来进行串行化前的预处理,这两个类写在一个JS文件(viewstate.js)中;反串行化隐藏域数据的工作在服务器端由一个PHP类[1](CViewState)来完成,然后另一个PHP类(CPhpPage)负责利用反串行化出来的数据形成事件调用,这两个类写在同一个PHP文件(viewstate.php)中。每个具体网页的服务器端代码均写在继承自CPhpPage类的一个子类中,由于事件传递与形成过程在基类CPhpPage中完成,这样每个具体网页就自动拥有了事件机制。

2 使用事件机制的页面程序结构

```
<? php
require_once("viewstate.php");      //包含页面基类和反串行化类的代码
class CThisPage extends CPhpPage    //具体页面均继承自CPhpPage
{
    function CThisPage()
    {
```

```
        $this->CPhpPage(false);      //显式调用基类的构造函数
        $this->Dispatch();           //派发事件
    }
    function OkClicked($para=null)   //客户端触发的点击 Ok 按钮事件
    {
      ......                         //在此写响应点击 Ok 按钮的具体代码
    }
    function CancelClicked($para=null)   //客户端触发的点击 Cancel 按钮事件
    {
      ......                         //在此写响应点击 Cancel 按钮的具体代码
    }
    function Data3Changed($para=null)    //客户端触发的下拉框选项更改事件
    {
      ......                         //在此写响应下拉框选项更改的具体代码
    }
      ......                         //其它事件
} //end of CthisPage
$page=new CThisPage();               //构造页面类的唯一实例
?>
<html>
<head><title>事件机制实例</title>
<script language="javascript" src="script/viewstate.js"></script>   <!--包含 View State 类和 VsEvent 函数的代码-->
<script language=javascript>
function PreDeal(action)    <!--在响应事件前在客户端可做一些处理-->
{
  if(…)
    return false;    <!--返回 false 表示不触发服务器端事件-->
  ……
  return true;    <!--返回 true 表示将触发服务器端事件-->
}
</script>
</head>
<body>
<form name="main" method="post" action="#">
<input type=hidden name="_viewstate" value="<?echo $page->VsGetString();?>">    <!--隐藏域-->
Data1:<input type="text" name="data1"><br>    <!--数据项 1-->
Data2:<input type="text" name="data2"><br>    <!--数据项 2-->
Data3:<select name="data3" onchange="javascript:VsEvent('Data3Changed()',predeal);"><!--数据项 3-->
    <option value="1">opt1</option><!--选择改变后将调用服务器端的 Data3Changed()函数-->
    <option value="2">opt2</option>
</select><br>
……      <!--其它内容-->
<input type="button" value="Ok" onclick="javascript:VsEvent('OkClicked()',predeal);"><!--点击后将调用服务器端的 OkClicked()函数-->
<input type="button" value="Cancel" onclick="javascript:VsEvent('CancelClicked()',predeal);"><!--点击后将调用服务器端的 CancelClicked()函数-->
</form>
</body>
</html>
```

正如读者所见,该页面的程序结构非常简单、清晰、易懂。采用本方法,程序员只需专注于具体事件代码的实现,而不用理会数据传递以及调用事件函数的细节。下一小节将详细介绍事件机制的详细实现方法。

3　事件机制的具体实现

3.1　Javascript

事件机制的实现离不开客户端的脚本代码,有不同的客户端脚本语言可用,如:VBSript,JScript 等。文中采用的是被大多数浏览器支持的 Javascript 语言[4]。

3.2　触发事件

服务器端的事件由客户端调用 Javascript 函数 VsEvent 来触发,一般而言,该函数在客户端的一个事件(比如点击按钮和改变下拉框的选项)被触发时调用,如上文的示例所示。

3.3　串行化

串行化的作用在于将有不同数据结构的变量以文本流的方式保持在一个域中,在网页中一般是用来保持键值对[5]。在本事件机制方法中,串行化将客户端触发的事件保持在一个隐藏域中提交到服务器端。在第 2 节的示例中,当点击 Ok 按钮时,客户端调用 Javascript 函数 VsEvent('OkClicked()',predeal),该函数首先调用 predeal 函数,如果 predeal 函数返回真值(true),则将串行化一个字符串"ACTION=OkClicked()"到隐藏域(_viewstate)中。这里 ACTION 是事件名称,而 OkClicked 是服务器端的事件处理函数的函数名。可以为不同的事件定义不同的事件名(这对需要同时触发多个事件时有用),比如下拉框选项改变事件可命名为 SEL_CHANGE,则下拉框选项改变事件串行化到隐藏域中就是"SEL_CHANGE=Data3Changed()"。还可以给事件处理函数提供参数,如:VsEvent('OkClicked(para)',predeal),这样,就为服务器端反串行化出来的事件处理函数提供了一个字符型的值为'para'的参数。

文中用到的串行化方法非常简单,即将各个变量转化成字符串并连接起来,各变量的串之间用特殊的字符串(文中命名为"分隔串",本方法使用"#|#")分隔开来。比如有两个变量 $a=1$, $b=2$ 则它们将被串行化成"#|#a=1#|#b=2#|#"。

注:为简洁起见,假设数据本身不含"分隔串"。进行复杂的应用编程时,如果被串行化的数据本身含有"分隔串"则需要转义处理。

3.4 反串行化

相对于串行化,反串行化的作用即在于将保持在一个域中的文本流数据提取出来转换成有意义的数据结构变量中的数据。在本事件机制方法中,反串行化将客户端提交上来的隐藏域中的数据解析成键值对放入数组变量中。在第 2 节的示例中,Ok 按钮被点击后,服务器端通过反串行化客户端提交的隐藏域(_viewstate)中的数据,将"ACTION=OkClicked(para)"放入 CViewState 类的成员数组变量 $_vs 中,即有 $_vs["ACTION"]="OkClicked(para)"。具体实现时,在 CViewState 类的构造函数中进行反串行化。

在反串行化时,需要用到一个很方便的字符串处理工具:正则表达式[1,4]。利用正则表达式可以很方便地将串行化的数据抽取出来放入结构化变量中。文中实现的方法在 PHP 类 CViewState 的构造函数中完成反串行化,代码如下:

```
function CViewState( $_viewstate)      //$_viewstate 是客户端提交上来的隐藏域中的数据
  {
    $this->_vs=array();      //成员数组变量 $vs 用以保存反串行化出来的数据
    $this->_viewstate= $_viewstate;
    $pattern="/* #-# *?/";      //正则表达式的模式串,对应上述的分隔串
    $pairs=preg_split( $pattern, $_viewstate);
    $pattern="/([^]+)=(.+)/";      //正则表达式的模式串,用于分解键值对
    foreach( $pairs as $value)      //循环以解析出键值对到成员数组变量中
    {
      preg_match( $pattern, $value, $regs);
      if(! empty( $regs[1]))
      {
        $this->_vs[ $regs[1]]= $regs[2];
      }
    }
  } //end of CViewState
```

在将串行化数据反串行化到数组中后,还需要进一步利用正则表达式以解析出事件处理函数,这个过程由 CPhpPage 类的一个成员函数 ParseEvent 来完成,该函数在 CPhpPage 类的构造函数中被调用。ParseEvent 函数实现代码如下:

```
function ParseEvent( $e)      //$e 是事件名,如 ACTION,SEL_CHANGE(见 3.3 节)
{
  $pattern="/#-#". $e."=([^\()+]\(([^\])*)\)#\|#/";      //正则表达式的模式串
  preg_match( $pattern, $this->_viewstate, $regs);//将事件处理函数的函数名和其参数分离出来并保存在
  $this->_event[ $regs[1]]= $regs[2];      //CPhpPage 类的成员数组变量中
}
```

3.5 调用事件处理函数

在完成了上述步骤之后,最后要做的就是调用被触发的事件的事件处理函数。PHP 的页面基类 CPhpPage 的成员函数 Dispatch()负责完成这项工作。该函数的实现如下:

```
function Dispatch()
{
  foreach( $this->_event as $key=>$value)      //轮询客户端触发的所有事件
  {
    if( $key)      //判断事件处理函数是否有被实现
    {
      $this->$key( $value);      //调用事件处理函数
    }
  }
}
```

注意:上面代码中 $key 其实就是 3.4 节解析出来的事件处理函数名,$value 就是事件处理函数的参数。

4 结束语

在使用 PHP 进行网页和 Web 应用开发时,采用文中所述的事件机制,可以避免使用 if...else 语句或 switch 语句来进行繁琐的分支选择处理,摒弃了机械的流式代码,不但简化了程序员的许多工作,而且使页面的程序结构更为模块化,程序的可读性和可重用性大大增强。

为了表述上的简洁,以上所述内容将串行化及反串行化的作用仅局限于保持事件,其实可以将许多其它结构化变量(如多维数组和类等)的数据串行化到隐藏域用以在客户端和服务器端传递数据。当然这样的话,要求客户端和服务器端都既要有串行化的功能代码也要有反串行化的功能代码。另外本机制可以很容易地运用到 ASP 及其它脚本语言中去。相比较 ASP.NET 或 PRADO 来说,ASP.NET 或 PRADO 的整套机制极其复杂,并且具有由于高度封装而导致的进行客户端的控制较麻烦的缺点;而本方法在使用的简单性上具有相当的优势(程序员只需花些许精力即能掌握),并且保持了客户端控制的灵活性。

(下转第 73 页)

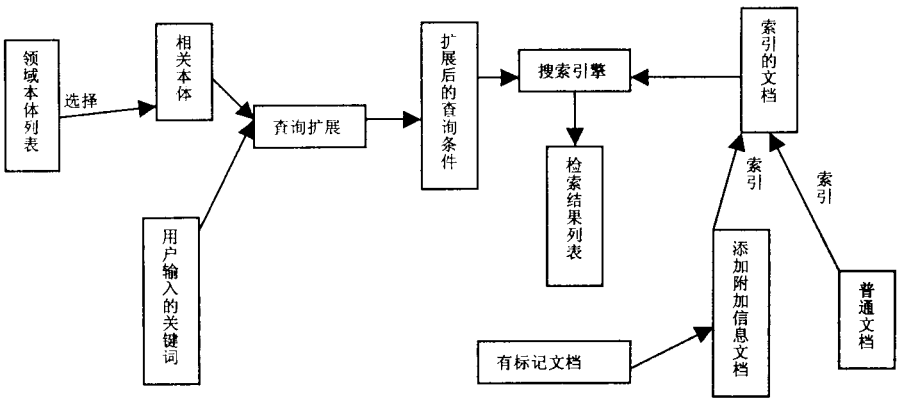

图 3　基于语义的 Web 信息检索模型

入的关键词进行文中所述的 3 种查询扩展,同时通过对扩展以后的各个查询条件的使用频度进行统计,将查询条件按其使用频度进行排序,以方便用户进行选择,搜索引擎根据用户选择的查询条件进行搜索。若用户输入的关键字不是领域本体中的概念,则不需要进行查询扩展。为了减少用户查询的响应时间,可以考虑在领域本体建立以后,即领域本体所蕴含的知识确定以后,就对本体中的概念进行查询扩展,将扩展结果保存于信息库中,这样可以使得在线处理变得相当简单,即只需从信息库中提取相应概念的查询扩展。由于考虑到目前的搜索引擎无法对包含有语义标记的文档进行索引,对带有语义标记的文档进行了文中所述的在文档中添加附加信息的处理,在处理的过程中还运用了本体的推理机制,找出了文档内的隐藏信息,将获得的隐藏信息同样经过处理后添加到文档中,使得搜索引擎可以对这些附加信息进行索引,这样使得搜索引擎对文档的索引更能反映文档的真实内容。该模型考虑了从两个方面解决传统搜索引擎用户检索返回结果相关度不高的问题,即用户查询的"忠实表达"及搜索引擎的索引能否揭示 Web 文档本质的角度。

4　相关问题

4.1　本体相关问题

面向检索的本体在内容和结构上应该符合检索系统的特点,应该考虑利用已有的主题词表、分类表或其他概念体系来半自动地生成本体,以减少重复劳动。同时随着知识的更新,会出现新的词汇,本体应该能够通过分析网络信息和用户的提问来进行自动更新。例如增加新的概念,对原有概念增加新的属性,重构概念之间的关系等。

4.2　Web 页面语义标注

对当今大量的 Web 文档进行语义标注是实现语义 Web 技术应用的前提,人工标注费时费力,所以有必要研究半自动化或自动化的语义标注方法。目前,基于本体的 Web 文档标注的项目有 SHOE(Simple HTML Ontology Extension)、WebKB(采用概念图表的本体来标注文档的人工标注方法)和采用与文档内容相关的基于知识的本体来标注文档的方法。这些方法都是通过人工处理来标注文档,虽然可以提高标注的准确度,但文档的修改和新文档的产生都需要重新标注。文献[5]采用一种类似于 OntoSeek 项目中采用的辞典,在原有本体上添加语言属性的半自动化的标注方法。

5　结　论

通过分析得出了将语义网络技术与传统搜索引擎相结合的检索模型,以解决搜索引擎返回结果相关度不高及网络过渡发展阶段多种类型 Web 文档(带有及不带有语义标记的文档)的检索的问题,对基于语义的检索系统的实现具有一定的指导意义。

参考文献:

[1] 北京奕天锐新科技有限公司 . 第 3 代搜索引擎初显锋芒[EB/OL]. http://www. 21cnbj. com/industrynews/native2003/2003-05-29-57. html. 2003.

[2] 刘遵雄. 搜索引擎的智能化发展趋势[J]. 科技情报开发与经济,2004(6):211-212.

[3] Berners-Lee T, Hendler J, Lassila O. The Semantic Web [EB/OL]. http://www. sciam. com/print_version. cfm? articleID=00048144-10D2-1C70-84A9809EC588EF21.

[4] Reagle J. RDF in XHTML[Z]. W3C Task Force Document, 2003.

[5] 贺　娇. 基于术语本体的网页标引方法[J]. 情报杂志,2004(3):28-29.

(上接第 70 页)

文中所述方法在 UUDynamics 公司的 iStar 产品的开发上得到实际的应用。

参考文献:

[1] Welling L, Thomson L. PHP and MySQL Web Development [M]. U. S. A: Pearson Education, Inc, 2005.

[2] 陈惠贞,陈俊荣. ASP. net 程序设计[M]. 北京. 中国铁道出版社,2004.

[3] 周世雄. . NET 经典范例教程[M]. 北京. 清华大学出版社,2004.

[4] Powell T, Schneider F. JavaScript 2.0 - The Complete Reference, Second Edition[M]. U. S. A: McGraw-Hill/Osborne, 2004.

[5] Dudney B, Lehr J, Willis B, et al. Mastering JavaServer Faces [M]. U. S. A: Wiley Publishing, Inc, 2004.