第35卷 第22期 计 算 机 工 程 2009年11月
Vol.35 No.22 Computer Engineering November 2009
·软件技术与数据库· 文章编号：1000—3428(2009)22—0088—03 文献标识码：A 中图分类号：TP311

ASP.NET 页面间传值综述

范明虎，樊 红
(武汉大学测绘遥感信息工程国家重点实验室，武汉 430079)

摘 要：对 ASP.NET 中各种页面间传值方法进行综述，阐述实现这些方法所必需的对象的概念、作用和特点，讨论这些方法的优缺点，通过实例代码给出这些方法的基本应用，并从生命周期、应用范围、可传递的数据类型和数据量 4 个方面对这些方法进行比较和分析。介绍了实现这些方法时常用的 2 种页面重定向方法。

关键词：页面；变量；传值；重定向；比较；分析

Overview of Value-passing Between Pages layout in ASP.NET

FAN Ming-hu, FAN Hong
(State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079)

【Abstract】This paper reviews a variety of methods of value-passing between pages in ASP.NET Necessary objects for implementing these methods are expatiated on their concepts, roles and characteristics. The advantages and disadvantages of these methods are discussed. The basic applications of these methods are given through examples. A brief analysis and comparison for these methods is made from four aspects: life cycle, scope of applications, data type and data size. Two commonly used methods of redirection in implementing these methods are introduced.

【Key words】pages layout; variable; value-passing; redirection; comparison; analysis

在 Web 应用程序中，页面之间交换数据(传值)是一种基本应用，它的实现效率对程序性能有直接的影响。传统的解决方案是通过 Get 或 Post 方法来实现的。但在 ASP.NET 中，这 2 种方法的使用发生了变化。因为 ASP.NET 使用了基于事件驱动的编程模型，页面采用了 PostBack 技术(发回给自己)，一些传统的方法必须做出一定的调整才能适应它。同时，ASP.NET 以其优秀的架构提供了更多可用于页面间传值的方法。这些方法各具特色，比较充分地适应了各种情况下页面间传值的需求。

1 页面重定向

页面间的传值方法总是与一定的页面重定向方法相联系，而且有些传值方法只能用某种重定向方法才能实现。ASP.NET 中有多种页面重定向的方法，这里仅介绍下文中用到(也是最常用的)的 2 种。

(1)Redirect 方法

将客户端重定向到新的页面，只是简单地终止当前页面，并转入新的页面开始执行，对转入的页面无任何限制。

(2)Transfer 方法

终止当前页的执行，并为当前请求开始执行新的页面。把执行流程从当前页面转到同一服务器中的另一页面，但是新的页面仍然使用当前页面创建的应答流。

这两种方法的主要区别是：(1)Redirect 方法可以实现任意网页间的重定向，而 Transfer 方法只能实现同一服务器中 ASPX 页面间的重定向。(2)Redirect 方法不会保留当前页面的表单(Form)数据和查询字符串(QueryString)，而 Transfer 方法则可以保留这些数据。

2 传值方法

为了简化问题的讨论，本文作如下约定：示例基于 C# 语言；源页和目标页已经导入了必要的命名空间；源页面设为 Src.aspx，值的传送点设在控件 Button1 的 Click 事件中，待传送的数据保存在控件 TextBox1 中；目标页面设为 Dst.aspx，值的接收点设在该页面的 Page_Load 事件中，接收的数据显示在控件 Label1 中；不深入讨论与之相关的编程问题，所有示例仅给出基本实现方法，所有代码在 Visual Studio 2005 中调试通过。

2.1 QueryString 的使用

最简单最常用的传值方法，与传统的 Get 方法相对应，使用变化相对较小。数据直接在 URL 中以明文传递，对于用户来说是可见的，但是可以加密。

此方法的优点是简单、方便，缺点是能传递的数据量少，而且不能传递对象。示例如下：

```
Src.aspx：
Private void Button1_Click(object sender, EventArgs e)
{    …
     //将待传送的数据保存到 URL 中
     strUrl=“Dst.aspx?id=”+TextBox1.Text;
     //重定向到目标页面
     Response.Redirect(strUrl);
}
Dst.aspx：
private void Page_Load(object sender, EventArgs e)
{    …
     //接收数据并显示在 Label1 控件中
     Label1.Text = Request.QueryString[“id”];
```

作者简介：范明虎(1974－)，男，博士研究生，主研方向：网络信息系统；樊 红，教授、博士

收稿日期：2009-06-17 **E-mail：**fanminghu@21cn.com

```
}
```

因为方法的描述完全相同，为节省篇幅，下文示例只给出数据传送和接收的实现代码。

2.2 Form 的使用

通过表单中的控件来传递数据，与传统的 Post 方法相对应，使用变化相对较大。

该方法的优点是可以传递大量数据，并且可以访问源页面控件中的数据；缺点是操作比较复杂，生命周期仅限于当前请求。示例如下：

```
Src.aspx：
public string id{get{return TextBox1.Text;}} //定义公共属性
Server.Transfer("Dst.aspx"); //重定向到目标页面
Dst.aspx：
//HTML 中引入 PreviousPageType 指令
<%@ PreviousPageType VirtualPath="~/Src.aspx" %>
//代码中获取 Form 数据
Label1.Text=PreviousPage.id;
```

该方法的实现灵活多样。文献[1]中给出了使用 Transfer 进行传值的方法，其他一些文献中给出了使用隐藏域或者隐藏字段进行传值的方法，笔者认为其实都是使用 Form 进行传值。因为传值过程中使用的数据载体是 Form，而不是其他东西。

2.3 Session 的使用

Sessionss 对象为当前用户会话提供信息。通过其属性可以方便地设置和检索 ASP.NET 会话状态变量，这些变量的值在会话持续期间保留不变，并且可以被同一个会话中的所有页面访问，这就为页面传值提供了途径和方便。

该方法的特点是[2]：(1)用户数据存储在服务器端；(2)ASP.NET 中的 Session 对象也可以不依赖 Cookie 而正常工作；(3)可以直接存储对象，且无大小限制；(4)有效期与用户的活动时间有关，具体为用户活动时间+自定义延迟时间。该方法的缺点是：由于 Web 应用程序对每个用户都会生成 Session 变量，因此它会随着用户数量的增多而加重服务器的负担。

如果数据量比较小，Sessionss 对象是保存只需要在当前对话中保持的特定数据的极好位置。实例代码如下：

```
Src.aspx：
Session[“id”]=TextBox1.Text;
Response.Redirect(“Dst.aspx”);
Dst.aspx：
Label1.Text=Session[“userid”].ToString();
Session.Remove(“id”); //清除 Session 变量
```

2.4 Application 的使用

Application 对象提供了对应用程序状态的访问。与 Session 对象相比，两者对数据的存储与访问都相似，不同之处在于：每个 Web 应用程序只生成一个 Application 实例，并应用于所有的用户和会话；有效期为整个 Web 应用程序的生命周期。因此，该方法非常适合存储那些数量少、不随用户变化而变化的常用数据。

在多用户并发访问控制方面，Apllication 对象采用了锁定机制，但是它串行化了对 Application 对象的访问，而这对于应用程序来说有可能形成严重的性能瓶颈。示例如下：

```
Src.aspx：
Application[“id”]=TextBox1.Text;
Response.Redirect(“Dst.aspx”);
Dst.aspx：
Application.Lock(); //加锁
Label1.Text=Application[“userid”].ToString();
Application.Unlock(); //解锁
Application.Remove(“id”); //清除 Application 变量
```

2.5 Cookie 的使用

Cookie 是网站存放在用户机器上的一小块信息，主要用来保存一些与用户相关的东西。

浏览器一般都对 Cookie 的使用进行了一些限制[3]：(1)每个域最多只能在一台用户机器上存储 20 个 Cookie；(2)每个 Cookie 的总尺寸不能超过 4 096 Byte；(3)一台用户的机器上 Cookie 的总数不能超过 300 个。

该方法的缺点是：并非所有的浏览器都支持 Cookie；用户可以禁用和删除它；信息采用明文保存，安全性差。

当特定的用户需要特定的数据片，并且需要把数据在某个可变的时段中保持的时候，Cookie 就非常方便。实例代码如下：

```
Src.aspx：
Response.Cookies[“id”].Value=TextBox1.Text;
Response.Redirect(“Dst.aspx”);
Dst.aspx：
Label1.Text=Request.Cookies[“id”].Value;
//清除 Cookie 变量
Response.Cookies[“id”].Expires=Now.AddDays(-1);
//清除当前域名(和路径)下的所有 Cookie
Response.Cookies[cookie].Expires=Now.AddDays(-1);
```

2.6 Static(静态)变量的使用

准确地说，是使用类的公共静态变量。这种变量在应用程序运行期间分配固定的存储空间，它依类而不依类实例而存在，使用时直接通过类名即可访问。在 ASP.NET 中，每个页面对应一个类，因此，可以利用它们的公共静态变量在页面间传递数据。

该方法的优点是如果善加利用，则可以有效提高数据传递效率，缺点是如果滥用，则会导致用户或页面间数据紊乱，造成极大的隐患。示例如下：

```
public static string id; //在类 Src 中定义公共静态变量
Src.aspx：
id=TextBox1.Text;
Response.Redirect("Dst.aspx");
Dst.aspx：
Label1.Text=Src.id;
```

2.7 Cache 的使用

应用程序中的缓存机制用于将需要大量服务器资源来创建的对象存储在内存中，以此大大改进应用程序的性能。这个机制同样可以用来传值。

与其他方法不同的是，该方法需要设置缓存项优先级和缓存时间。因为当系统内存缺乏时，缓存机制会自动移除很少使用或优先级较低的项，从而造成传值失败。

该方法的优点是传递数据的大小和数量无限制，速度快。缺点是缓存机制的操作相对比较复杂。

在 ASP.NET 中，缓存机制通过 Cache 类实现。示例如下：

```
Src.aspx：
Cache["id"]=TextBox1.Text;
Response.Redirect("Dst.aspx");
Dst.aspx：
if(Cache["id"]!=null) Label1.Text=Cache["id"].ToString();
Cache.Remove("id"); //移除缓存项
```

如果 Cache["id"]为空，则传值失败。可使用如下方法实

现过期策略(以 10 min 过期为例)：

```
Cache.Insert("id",TextBox1.Text,null,
Cache.NoAbsoluteExpiration,new TimeSpan(0,10,0));
```

2.8 Context 的使用

Context 对象包含与当前页面相关的信息，提供对整个上下文的访问，包括请求、响应、以及上文中的 Session 和 Application 等信息。可以使用此对象在网页之间共享信息，从而实现页面间的传值。

与使用 Form 的方法类似，该方法也能保持大量的数据，缺点也相同，但使用方法相对比较简单。示例如下：

```
Src.aspx：
Context.Items["id"]=TextBox1.Text;
Server.Transfer("Dst.aspx");
Dst.aspx：
Label1.Text=Context.Items["id"].ToString();
Context.Items.Remove("id"); //移除项
```

2.9 ViewState 的使用

ViewState 是 ASP.NET 用来在同一页面的多个请求之间保存和还原服务器控件视图状态的一种机制。与传统的“同一页面”不同，ASP.NET 中“同一页面”的每一个请求都会导致服务器重新生成该页面，但是新生成的页面并不包含原来页面的数据。ViewState 的任务就是保存原来页面中服务器控件视图状态的数据供新页面使用。从这个意义上讲，ViewState 也可以看作是一种在页面间传递数据的工具。

ViewState 的工作原理是：作为一个隐藏的窗体字段在客户端和服务器之间传递，可见，滥用 ViewState 会加重页面回传的负担，从而降低应用程序的性能。此外，ViewState 也能被控件、页面和应用程序禁用。示例如下(在同一页面中)：

```
ViewState["id"]=TextBox1.Text; //数据保存
Label1.Text=ViewState["id"].ToString(); //数据取出
ViewState.Remove("id"); //数据移除
```

2.10 Web.config 和 Machine.config 的使用

Web.config 文件用来设置每个 ASP.NET 应用程序的一些属性，Machine.config 文件用来设置所有应用程序基础信息。

这 2 种文件保存的数据一般都很小，多为明文，特别适合保存一些字符串常量，如数据库连接信息。此外，Web.config 文件是可以扩展的，因此，也可以用来传递变量。

因为这 2 种文件会被自动缓存，所以不存在因磁盘 I/O 产生的性能瓶颈问题。要注意的是文件中某些设置会导致文件被修改后 Web 应用程序的重启(详情见文献[4])。示例如下：

```
Src.aspx：
WebConfigurationManager.AppSettings.Set("id",
TextBox1.Text);
Dst.aspx：
Label1.Text=WebConfigurationManager.AppSettings["id"];
//数据移除
WebConfigurationManager.AppSettings.Remove("id");
```

2.11 其他方法

在其他方法中应用最多最广的就是数据库了，比较常见的还有文件。相对于数据库来讲，文件需要自定义格式，还有可能因磁盘 I/O 而产生性能瓶颈，设计和使用的工作量都比较大，除非必要，很少应用。

数据库和文件在程序中应用广泛，几乎可以存储任何类型、任何尺寸和数量的任何信息，而且可以永久保存。

考虑到数据库和文件都是通用技术，此处不作详述。

对于其他传值方法，或者应用极少，或者是上述方法的变种，这里不作讨论。

3 比较与分析

基于以上论述，下面通过表 1 来简单比较和分析这些方法的异同。

表 1 传值方法比较

传值方法	生命周期	应用范围	数据类型	数据大小
使用 QueryString	与目标页有关	单用户	字符串	1KB
使用 Form	当前请求	单用户	全部	不限
使用 Session	当前会话	单用户	全部	不限
使用 Application	应用程序	全局	全部	不限
使用 Cookie	可自定义	单用户	字符串	4KB
使用 Static 变量	应用程序	全局	全部	不限
使用 Cache	可自定义	全局	全部	不限
使用 Context	当前请求	单用户	全部	不限
使用 ViewState	当前页	单用户	全部	不限
使用 Web.config 和 Machine.config	可永久	全局	字符串	不限
使用数据库或文件	永久	全局	全部	不限

为了简化问题的讨论，此处不考虑权限管理和人为因素的影响，因为它们会大大复杂化这些方法的应用，对它们的讨论超出了本文的范围。

表中的“当前请求”指目标页的第 1 次加载到目标页回发操作或作废前；“当前会话”指用户活动时间+自定义延迟时间；“应用程序”指应用程序的生命周期。数据大小的“不限”并非指无限，因为要受存储介质的限制。在程序中动态写入 Web.config 中的数据默认不保存到该文件中，应用程序重启后会消失。

通过前文的论述和上表的比较可以看出：传递无需保密的简单数据使用 QueryString 最方便；通过控件直接传递数据只能使用 Form；传递只与具体用户有关的数据使用 Session 最合适；要想在所有的用户间共享数据使用 Application 比较方便，而共享数据如果是字符串常量则使用 Web.config 也是很好的选择；Static 变量可以长时间保持数据，而 Cache 会自动清除数据，Context 还具有访问请求、会话以及 Application 的能力；如果要长期保存数据，则只能在 Cookie、Web.config、数据库或文件之中选择；除使用数据库或文件之外，数据传递速度都很快，只要设计得当，一般不会形成性能瓶颈。总之，每种方法都有其自身的特点与优势，实际应用中选取何种方法应该根据具体情况具体分析，不能一概而论。

4 结束语

无论何种传值方法，都有其适用的条件和范围，都有其优缺点。在实际应用中，应该根据需求，仔细、全面地权衡各种方法的利弊，比较它们的应用效果，才能做出较佳的选择。一般来说，要考虑的主要因素有数据的保密性、类型、数据量的大小、使用范围和保持时间长短等。既要满足需求，又要尽可能简单易行，还要兼顾应用程序的性能优化。在实践中，还要不断尝试，不断分析和总结经验，才能充分理解和灵活运用这些方法，从而更好地解决各种实际问题。

参考文献

[1] 陈启祥, 左 强. ASP.NET 页面间传值方法研究[J]. 计算机工程, 2006, 32(8): 113-114.

[2] Walther S. ASP.NET 2.0 揭秘(卷 2)[M]. 谭振林, 黎 志, 译. 北京: 人民邮电出版社, 2007.

[3] Zakas N C. JavaScript 高级程序设计[M]. 曹 力, 张 欣, 译. 北京: 人民邮电出版社, 2006.

[4] Evjen. ASP.NET 2.0 高级编程(特别版)[M]. 杨 亚, 译. 北京: 清华大学出版社, 2007.

编辑 索书志