Web 页面间信息传递方法的效果研究

Research on the Effect of Methods to Transfer Data between Web Pages

杨育标 徐炳文 (广东岭南职业技术学院 广东广州 510301)

摘 要: 在 Web 应用系统的开发中,页面间的信息传递是系统必不可少技术,是否选用合适的传递方法,将影响 Web 应用系统的安全性、可靠性和性能。本文首先对各种 Web 页面间的信息传递技术进行归纳分类,然后就各种类型的传递方法对 Web 应用系统的安全性、可靠性、性能、灵活性、应用范围、应用限制等方面的影响进行分析探讨。

关键词: Web 应用 页面 信息传递 安全性 可靠性

1 引言

随着 Internet 的普及相关技术的进步,越来越多的应用系统转向 B/S 结构或以浏览器为客户端的多层结构。这些 Web 应用程序使用的是 HTTP 协议,客户端与 Web 服务器之间仅是请求与响应之间的关系,服务器不保留与客户端的连接状态,客户端访问任意的两个 Web 页面实际上是与 Web 服务器的两次连接,这就造成了在两个 Web 页面之间传递信息的困难。可喜的是技术的进步,已有多种方法可以实现 Web 页面间信息的传递,但是不同的传递方法在安全性、可靠性、应用范围、灵活性上有所不同。因此,深入研究各种 Web 页面间信息的传递方法及传递效果,对于开发安全性高、性能好的 Web 应用程序有重要的现实意义。

2 Web 页面间传递信息的方法分类

在 Web 页面间传递信息的方法有多种,归纳其来有以下六种类型:

2.1 用 <a> 标记或相应功能的超链接控件进行信息传递

HTML 的 <a> 标记的主要用途是用于导航到某个目标页面,可以同时用来传递信息。具体做法是在 <a> 标记的 href 属性中,把要传递的信息作为 href 参数的一部分,与表示目标页面的 URL 之间用"?"符号(英文的问号)隔开,如果要传递的信息有几个,信息之间用"&"号分隔。下例是一个导航到 navi_page. aspx 页面的链接,其中包含国籍和语言的信息:

```
<a href ="navi_page.aspx? country = China &
Language = Chinese" > 目标页面的超链接 </a>
```

在目标页面,可以使用 Request 对象的 QueryString 属性取出传递过来的信息,例如对上面的 <a> 标记中传递过来的信息,可以用如下的代码取出信息:

```
Dim str1, str2 As String
str1 = Request.QueryString("country")
str2 = Request.QueryString("Language")
```

在目前流行的可视化开发工具中,提供了构建超链接的控件,该类控件的功能本质上与 <a> 标记是相同的,因此也可以用其进行信息传递,其构建传递信息与获取信息的方法与使用 <a> 标记的信息传递方法相同。

2.2 使用 Redirect 方法传递信息

上面信息传送方法中,信息是以硬编码的方式存在 Web 页面中的,Web 页面一旦设计完成,所传递的信息就固定了。使用 Response 对象的 Redirect 方法传送信息,可克服上面的传递方法的灵活性问题,下面的示例把源页面中两处用户的输入传递到目标页面:

```
a = TextBox1.Text
b = TextBox2.Text
Response.Redirect("navi_page.aspx?country =
" + a + "&Language =" + b)
```

获取 Redirect 方法传送的信息与获取 <a> 标记

中传送过来的信息完全相同，例如在目标页面获取源页面中传递过来 a、b 变量的值与获取 <a> 标记中传送过来的信息的代码完全相同，实际上，这两类传递方法都属于所谓的传递"查询字符串"。

2.3 使用 get 方法和 post 方法

这两种方法用于把当前页面中的文本框或隐藏域中的信息发送到目标页面，实现的方法是在源页面的 <form> 标记的 action 属性指定接收信息的目标页面，同时在 method 属性使用 get 或 post 关键字指定发送信息的方式，下面的示例是把当前页面中文本框中的信息传递到 navi_page.asp 页面：

```
<form id = "Form1" action = "navi_page.asp"
method = "get" runat = "server" >
```

获取 get 方法传递的信息也使用 Request 对象的 QueryString 属性，下面的示例代码在目标页面获取源页面中"Text1"文本框中的信息：

```
Data1 = Request. QueryString ("Text1")
```

如果把上面的 get 方法中的 get 关键字改为 post 关键字，就是 post 的传递方法，同样可以在页面间传递信息，但在安全性和性能方面与 get 方法有所不同。在目标页面获取 post 方法传递的信息不是使用 Request 对象的 QueryString 属性来获取，而是使用 Form 属性来获取[1]，下面示例是获取源页面中"Text1"文本框中的值的代码：

```
Data1 = Request. Form("Text1")
```

2.4 Server 对象的 Transfer 方法

在新一代的 Web 应用程序开发平台，出现了新的信息传递方法，.NET 平台的 ASP.NET 中的 Server 对象的 Transfer 方法就是其中的一个代表，它不但可以传递源页面中的基本数据类型数据，还可以传递数组、图片数据和自定义的数据类型的数据，下面是在 Web 页面间传送图片数据的例子。

首先，在源页面定义一个应用程序范围的共享对象：

```
Friend myImage As New _
System. Web. UI. WebControls. Image
```

然后准备图片数据，再用 Transfer 方法导航到目标页面：

```
myImage. ImageUrl = "images\gd30002. jpg"
Server. Transfer("navi_page. aspx")
```

在目标页面，使用 HttpContext 类的对象的 Handler 属性获取 Http 请求，并转换为封装传递的信息类型（即源页面的类型，如下例中的 WebForm1 类），然后就可以通过转换后的对象属性访问所传递的值。以下的示例是获取传递的图片数据并在目标页面的 Image 控件（Image1）中显示出来：

```
Dim sourcepage As WebForm1
sourcepage = CType ( Context. Handler,
WebForm1)
Image1. ImageUrl = sourcepage. myImage. Image-
geUrl
```

2.5 跨越多个 Web 页面的信息传递

在上面的信息传递方法中，信息只能从第一个页面传送到第二个页面，不能在第三个页面获取传递的信息。而在实际应用中，有时需要信息在多个页面中进行传递，例如网上购物系统，用户进入购物站点浏览多个页面并购买多种商品，就需要该用户的信息及用户在各页面所进行的操作结果能在他所浏览的多个页面之间进行传递。解决这类在多个页面之间传递信息可以通过 Web 应用的内置对象来解决，根据应用的范围不同，可以分为 Application 对象，Session 对象和 Cookie。

2.5.1 使用 Session 对象传递信息

Session 对象用于管理一个用户对站点的一次访问的信息，不管在这次访问中访问了多少页面，保存在这个 Session 对象中的信息在这些页面中都有效，能够在任何页面获取这个对象中传递的信息，或修改保存在这个对象中的信息。该对象的使用很简单，可以不先声明而直接使用，下面的示例是在某一页面把作为购物车存储结构的哈希表"MyHashCart"保存在 Session 对象中：

```
Session("Cart") = MyHashCart
```

在其他页面，可以用简单的方法把 Session 对象传递的内容取出，下面的代码是取出 Session 对象所传递的哈希表：

```
Dim MyHashCart As Hashtable
If Session("Cart") Is Nothing Then
MyHashCart = New Hashtable
Else
MyHashCart = Session("Cart")
End If
```

2.5.2 使用 Application 对象传递信息

Application 对象又称全局对象，该类对象的信息能够被所有的用户在任何页面共同使用，它的生命周期是从该应用程序的第一个页面执行开始，直到网站关闭为止。如果是传递全局性的信息就可以使用该类对象，通常是使用 Application 对象存储 Web 应用系统的控制参数，其使用方法与 Session 对象类似，以下的例子是在任意页面检查 Web 服务器的状态变量：

```
Status = Application("Server_Status")
```

2.5.3 使用 Cookie 传递信息

对于特定的计算机用户来说，Cookie 也可以用来在多个页面间传递信息，Cookie 实际上是 Web 服务器保存在用户硬盘上的一段文本，位于计算机的客户端的 Document and Settings 目录下的特定用户目录的 Cookies 子目录下。因此使用 Cookie 在页面间传递信息仅是对特定的计算机的特定操作操作系统用户而言，而不是 Web 应用系统的用户。Cookie 的生命周期可以自行定义，因而可以跨越对 Web 服务器的多次访问，这比 Session 对象的信息传递在时间上有更大的跨度。下例是在 Cookie 中写入信息的示例代码：

```
Response.Cookies("User").Value = "张三"
Response.Cookies("User").Expires = _
DateTime.Now.AddDays(5)
```

下面是在另一次访问中读出以前写入的信息的示例代码：

```
If Not Request.Cookies("User") Is Nothing Then
TextBox1.Text = Request.Cookies("User").Value
End If
```

2.6 通过数据库转接在 Web 页面间传递信息

这种方法的传递是在源页面把信息存入数据库，导航到目标页面之后再把在源页面中存入数据库中的信息取出来，它同样可以跨越多个页面来传递信息。在实际使用中，由于 Web 应用系统中可能有很多的用户同时进行操作，因此存在着如何确定用户与信息的对应关系统的问题，通常是结合 Session 对象一起使用，在源页面把要传递的信息及用户 SessionID 号同时存入数据库，在目标页面根据用户的 SessionID 号把源页面中存入的信息取出。

3 信息传递效果的分析研究

虽然有多种方法可用于 Web 页面间传递信息，但是实现方法的不同，对 Web 应用程序的安全性、可靠性、灵活性、性能的影响也不同，应用范围也有所不同，下面就这些相关问题进行探讨。

3.1 安全性

就安全性方面来探究，在 Web 页面间传递信息的方法中，第 1 种类型的方法有明显的安全缺陷，用这种方法传递的信息，在鼠标指向源页面的超链接时，传递的信息会在浏览器的状态栏中显示出来。在导航到目标页面后，传递的信息还会在浏览器的地址栏中显示出来，显然这种方法不能用来传递机密信息。第 2 种类型的方法和第 3 种类型的 get 方法也有安全性问题，在导航到目标页面过程中，传递的信息也会在浏览器的地址栏中显示出来。除了信息暴露的安全问题外，这些类型的传递方法还有非法的数据修改的安全问题，当传递的信息显示在地址栏时用户可以修改其信息，并把被非法修改的信息发送到目标页面，如果这些信息是目标页面的查询参数，就可能使用户看到未经授权查看的内容。此外，这些方法都是以字符串的形式传递信息，不能传递二进制信息，而在实际应用中，用标准加密法加密的信息是以二进制的形式存在的，因此从信息加密的角度看，这些方法也不利于提高传递信息的安全性。

使用 Post 方法来传递源页面中的文本框或隐藏域中的信息具有信息隐藏的功能，信息在传递过程中不会暴露出来，也可以防止非法的数据修改，但它也无法传递以二进制形式存在的加密信息。

Cookie 的传递信息方法也有明显的安全性缺陷，因为传递的信息是存放在客户机硬盘的固定位置上，不仅信息暴露，而且可被篡改，因此不适合传递重要的信息。

在防止信息暴露和信息非法修改方面，transfer 方法、数据库转接方法等都有很好的表现，它们还可以传递多种类型的数据，如加密的二进制数据，这使信息的传递更加安全。相比之下，.NET 平台的 Transfer 方法，在安全性上更好，不仅传递的信息不暴露，甚至导航到目标页面后，目标页面的 URL 也不显示出来，仍然显示源页面的 URL。

3.2 可靠性

不同的传递方法对 Web 应用程序的可靠性的影响主要表现在数据类型的转换引起的不可靠性、非法修改引起的不可靠性、以及数据量超限引起的不可靠性。

实际的业务过程中，需要传递的信息有多种多样的数据类型，如：数值、字符、日期、对象等，而前 3 种类型传递方法的，都是把信息作为字符串来传递，即对于非字符串的数据也是作为字符串来传递，在目标页面接收到后，再转换为原来的数据类型，这有可能出现字符串不能正确地转换成数值或日期数据的情况，从而导致系统出错。对于所谓的“查询字符串”的传递方法，还会因其传递的信息出现在地址栏，由此引起的非法修改信息而使系统出错。

值得注意的是有些传递方法对传递的信息量是有限制的，当传递的信息量超出限制就可能引起信息的丢失或者系统出错。上述传递方法的前两种类型和第 3 种类型的 get 传递方法，传递的信息量就是有限制的，虽然信息量的限制根据浏览器的版本不同有所差异，但通常就是几 KB。因此，如果使用 redirect 方法和 get 方法来传递用户在文本框中的输入内容，设计时就要考虑到户的输入数据量是有很大差异的，在程序中应对用户的输入数据进行长度验证，避免信息传递的不完整。

在传递方法中可以避免类型转换的有 transfer 方法、数据库转接方法、Session 对象等传递方法，它们可以传递各种类型的数据。若要传递大数据量的信息可采用 Post 方法、transfer 方法和数据库转接方法。

3.3 对性能的影响

不同传递方法对性能的影响体现在两个方面，一是对 Web 服务器性能的影响，另一是对用户请求面页的响应速度的影响。

使用 Application 对象和 Session 对象进行的信息传递，是通过 Web 服务器的内存来实现的，从用户感觉的反应速度来看是很快的，但大量的使用这些对象来传递信息或传递大数据量的信息，将占用大量的服务器内存，增加 Web 服务器的负担，导致服务器性能的下降，特别是 Application 对象由于其生命周期很长，对服务器的影响更加显著。

第 1 至第 4 种类型的方法，其传递信息存在于访问请求或 Web 页面中，这些方法的传递信息仅在进行处理时短暂地占用服务器的内存和 CPU，对服务器的性能影响较小，但每次都要从请求或 Web 页面中提取信息，因此从用户感觉来看，反应速度稍慢。

使用数据库转接的方法，需要经过较多的中间环节，包括数据库的连接和存储操作，因而从用户感觉来看，反应速度较慢，但对服务器的性能的影响不是很大。当传递的信息量很大时，因为它不占用 Web 服务器的内存资源，将有利于保证 Web 应用系统的性能和可扩展性。

3.4 灵活性与简单性

在简单性和灵活性方面，第一种类型的传递方法实现起来是最简单的，但它缺乏灵活性，要传递的内容只能在页面的设计中确定。其他类型的信息传递方法，都具有好的灵活性，能够传递用户动态设定的内容。就实现难度来说，Transfer 方法和数据库转接等方法，实现起来稍为复杂。

3.5 应用范围及限制

为了正确的使用 Web 页面间信息的传递方法，还要注意不同传递方法的应用范围和限制。Application 对象传递的信息可以在所有用户、所有页面之间进行，具有最广泛的应用范围，但这种方法的信息共享有可能导致信息的相互覆盖，影响传递信息的准确性，因此使用该类对象要考虑周到，避免一个用户的信息被另一用户的信息覆盖。使用 Session 对象的信息的传送方法可以跨越多个页面传递信息，但只有同一用户才能够存取。使用 Cookie 的传递方法能在某台客户机和服务器之间传递信息，并可定义有效期，因此该方式的有效范围大于 Session 方法的应用范围。但当客户端的浏览器禁止使用 Cookie 时，信息的传递将无法进行。数据库转接的方法就时间上或页面范围方面来讲，没有什么限制，但它总是要与其他的对象结合使用，因此其应用范围实际上取决于其结合使用的对象的生命周期。除此之外的其他方法的应用范围都是只能在某特定的两个页面间传递信息。

应用限制的另一个要注意的问题是不同的 Web 应用开发平台和工具，其传递信息方法可能有所变化，例如，在 ASP. NET 中已经不能用第三种类型的方法来导航到目标页面，因而也不能使用这种类型的方法来在 Web 页面间传递信息。

4 结束语

从上面的分析探讨可知，Web 页面间进行信息传递的方法有很多种，不同的实现方法在安全性、稳定性、性能、灵活性方面有所不同，各有其优、缺点。在

（下转第 50 页）

(上接第 41 页)

Web 应用系统的开发中,究竟应选择何种实现方法?这要根据实际情况,包括所选择的开发平台及工具来确定,没有统一的选择标准。事实上,大多数的应用系统会同时使用多种信息传递方法,例如在进行统计和系统的全局控制时,使用 Application 对象来传递信息;在用户管理上使用 Session 对象跟踪用户的操作结果,此外还会选用其他的一种以上的仅能在特定的两个 Web 页面之间传递信息的方法。要强调的是不管选用什么方法进行页面间传递信息,要充分了解该方法所带来的影响,并采取适当的措施来弥补,同时做好测试和检验工作。

参考文献

1 邹天思,孙明丽,庞娅娟. ASP 开发技术大全[M]. 北京:人民邮电出版社, 2007. 272 – 298.