ASP.NET中的状态管理

ASP.NET State Management

赵若斌 （中国科技大学研究生院 100039）

朱建方 （北京工商大学 100037）

摘要：本文对ASP.NET中几种常用的状态管理方法进行了分析比较，包括ViewState、Session、Cookie、Application、Cache的使用范围、使用方法以及各自的特点。

关键词：ViewState Session Cookie Application Cache

1 引言

状态管理指在应用程序生存期内或用户和应用程序交互的过程中数据的保持。ASP.NET程序使用无状态的HTTP协议，HTTP协议本身不保持状态，因此ASP.NET需要应用状态管理机制保持程序状态。本文介绍ASP.NET中页面状态管理、用户状态管理和应用程序状态管理几种常用的方法。

2 Web页面状态管理

Web页面状态管理指页面级数据的保持。ASP.NET中可以使用服务器控件和ViewState保持页面级数据。

2.1 服务器控件

HTML没有维护窗体元素状态的内置机制，因此在页面刷新时，象文本框的输入数据、列表框选择的项目等状态信息会丢失。而ASP.NET服务器控件在默认情况下，利用内置的ViewState（视图状态）功能，能自动维护自己的状态信息，在页面刷新时，服务器控件的状态信息，如文本框输入数据、列表框选择的项目等，可以在重建的页面中保持。

2.2 ViewState

我们可以直接使用StateBag类的实例ViewState保存页面状态，它以键/值对集合的形式存储数据，可以通过指定关键字来读写数据项。

例如：插入数据项：ViewState（"Sport"）="Basketball"

检索数据项：Dim SporName As String

SporName=CType（ViewState（"Sport"），String）

2.3 服务器控件和ViewState比较

服务器控件和ViewState可用于回送（post back）时页面数据的保持。回送（post back）是指ASP.NET页面向自己发出HTTP POST请求，经服务器处理后重新生成页面并返回。服务器控件和ViewState中的状态信息不保存在服务器上，而是经过编码以隐藏字段的形式一起保存在返回页面中，服务器控件用来保持用户的输入数据，ViewState用来保存表单的自定义值。ViewState功能要占用处理器时间，发送额外的信息，因此在不需要时，可以用EnableViewState属性关闭该功能。例如：关闭整个页面ViewState：<% @ Page EnableViewState = "false" %>；关闭服务器控件ViewState：<asp: TextBox id= "name" EnableViewState ="false" runat= "Server"/>

3 用户状态管理

用户状态管理指跨HTTP请求保持用户数据，ASP.NET中常用Session和Cookie两种方式。

3.1 Session

(1) Session对象的使用。Session对象保存单个用户和特定的Web应用程序相互作用时的数据信息（会话信息）。Session由创建它的用户使用，不能在不同的用户间共享，它以键/值对集合的形式存储数据，可以通过指定关键字来读写Session中的数据项。

例如：Session（"MySessionVar"）="Hellow world"

SessionValue=Session（"MySessionVar"）

(2) 标识会话。为了在用户及其Session对象之间建立联系，ASP.NET为每个用户分配了一个唯一标识符SessionID，用来标识用户，鉴别存储在服务器上的Session值。ASP.NET中有两种方式从用户端向服务器传送SessionID：

• 有Cookie方式：SessionID以Cookie的形式存储在用户机器上，用户在向服务器的请求中提交SessionID，服务器利用SessionID访问用户存储在Session中的信息。

• 无Cookie方式：SessionID自动被添加到返回页面中所有的URL中，当用户使用包含SessionID的URL提出请求时，ASP.NET能够从中提取SessionID并将请求映射到合适的Session。

通过无Cookie方式传递SessionID，使ASP.NET程序在用户浏览器不支持或关闭Cookie功能时，可以继续使用会话状态。

(3) Session的存储。在ASP.NET中Session有三种存储模式：

① InProc模式：Session存储在应用程序进程中。这种模式Session的存取在进程内，系统开销小，速度快。但是这使Session被限制在单一服务器单一进程中，不适用于WebFarm，系统缺乏伸缩性，在应用程序崩溃或IIS服务器重启时，会话状态将丢失。

② StateServer模式：Session存储在一个独立于IIS进程的专用状态进程中，状态进程既可以和IIS服务器在同一机器上运行，也可以在不同机器上运行。这种模式session的管理具有独立进程的可靠性，在应用程序崩溃或IIS服务器重启时，会话状态可以恢复，适合用于WebFarm，系统具有较好的伸缩性。但是Session的存取是跨进程的，所以相对于InProc模式系统开销要大，速度上要低。

③ SQLServer模式：Session存储在SQL Server数据库中，ASP.NET管理了使用SQLServer的必要步骤。这种模式需要读写数据库，从速度上来说要低于前两种方式，但是这种模式最可靠，在机器崩溃时可以使用SQLServer中的数据恢复会话状态，适合用于WebFarm，常常是一些商业运行环境中会话状态管理的首选。

(4) 配置Session，ASP.NET通过配置文件中<sessionState>元素管理会话状态。<sessionState>元素的语法如下：

```
<sessionState
    mode = "Off | Inproc | StateServer | SQLServer"
    stateConnectionString = "IP address : port number"
    sqlConnectionString = "sql connection string"
    Cookieless= "false | ture"
    timeout= "number" />
```

其中mode属性用来指定session存储类型，选Off项则关闭会话状态；stateConnectionString属性在mode属性选择了StateServer时，才是有效和必要的，它指定了状态服务器的IP地址和使用的端口；sqlConnectionString属性在mode属性选择了SQLServer时，才是有效和必要的，它指定存储状态信息的数据库的连接字符串，包括数据库的IP地址，用户名和密码；Cookieless属性指定session是否使用Cookie；Timeout属性用来设定超时的时间，以分钟为单位。

Session要占用服务器的资源，不需要时，可以用mode属性的Off选项关闭整个应用程序@˜如：<% page EnableSessionState = "false" %>。

3.2 Cookie

Cookie是一种在客户端存储有关用户或网站信息的方式。Web浏览器软件会把HTTP响应中的Cookie数据存储到用户端Cookie中，或者把存储在Cookie中的数据加载到HTTP请求中。通常使用Response对象和Request对象来设置和检索Cookie。Cookie读写有两种情况：

(1) 只有一个值的Cookie，设置和检索时必须使用Value属性。例如：

设置：Response.Cookies("MyCookie").Value="Single Cookie"

检索：Request. Cookies("MyCookie").Value

(2) 另一种是包含多个键/值对的Cookie，设置和检索时要指定关键字名称。例如：

设置：Response.Cookies("SportCookie")("Username")= "Chris"

Response.Cookies("SportCookie")("Favorite") = "Basketball"

检索：Request. Cookies("SportCookie")("Favorite")

使用Expires属性设置Cookie的有效期，可以将Cookie保存在硬盘上，建立持久性Cookie，

例如：将SportCookie作废时间设置为代码执行后一个月：Response.Cookies("SportCookie").Expires=DataTime.Now.AddMonths(1)，如果建立Cookie时没有明确设定有效期，建立的是临时性Cookie，Cookie保存在浏览器内存中，关闭浏览器Cookie值将丢失。

3.3 Session和Cookie的比较

Session和Cookie都可以用来存储用户信息，

但它们有不同特点。首先，Session在服务器端存储信息，占用服务器端的资源；Cookie将信息存储在用户端，虽然不占用服务器资源，但是Cookie的使用需要客户端的配合，当用户的浏览器不支持或关闭Cookie时就无法使用。其次，Cookie适于存储简单的文本信息，只能存储有限量的信息（一般情况下不超过4KB）；而Session可以存储任意类型的信息（包括对象），且Session本身对存储数据的大小没有限制。

4 应用程序状态管理

ASP.NET应用程序状态管理是指应用程序级的数据存储，这些数据可以在整个应用程序中使用，被应用程序的所有用户共享。常用的管理方式有Application和Cache。

4.1 Application

Application对象存在于ASP.NET进程中，生存期与应用程序相同，为各应用程序私有，不支持独立于ASP.NET进程外的数据存储。

(1) Application对象的使用。Application以键/值对的集合形式存储数据。可以通过指定关键字来读写数据项。

例如：写入数据项：Application ("MyData") = "This is some data to Application "读取数据项：myApplicationData= Application ("MyData")

(2) Application对象访问的同步，Application对象被应用程序的用户共享，使用Lock()和Unlock()函数进行同步。例如：

```
Application.Lock()
Application("HitCounter")= Application ("HitCounter")+1
Application.Unlock()
```

4.2 Cache

Cache对象也存在于ASP.NET进程中，生存期与应用程序相同，它不但具有与Application类似的功能，还有强大的数据操纵功能。

(1) Cache对象的使用。Cache对象以键/值对集合的形式存储数据，可以通过指定关键字来插入和检索数据项。

例如：插入数据项：Cache("MyData")= "This is some data to cache"

读取数据项：myCacheData=CType(Cache ("MyData"), String)

(2) Cache对象数据操纵功能。

① 基于依赖性的终止：可以使Cache中数据项的有效性依赖于某个文件、另一个数据项或定义的时间戳，当它依赖的文件或数据项发生变化时，或者超过设定的时间戳，该项内容就会被从Cache中删除。

② 资源管理 管理系统能够根据实际需要，自动删除Cache中不重要的项，释放内存资源。

③ 支持回调：允许在应用程序中指定一个方法，当从Cache中删除数据项时，调用该方法。

要使用这些功能操纵Cache中数据项，在添加数据项时必须使用Add方法或Insert方法，因为这两种方法能够提供操纵数据项所需的参数，它们在本质上是相同的。只是使用Add方法时要求输入全部八个参数，并返回一个对象；而Insert方法经重载提供了具有不同个数参数的版本，参数最多的一种版本如同Add方法，但是它不返回对象。例如：

- 插入数据项"CustomerData"，使它依赖于名为"cust. xml"的XML文件，即当XML文件发生改变时，ASP.NET将自动从Cache中删除该项。

```
Dim dependencies As new CacheDenpendency (Server.MapPath("cust. xml"))
Insert ("CustomerData", CustomerObject, dependencies)
```

- 插入数据项"CustomerData"，Insert ("CustomerData", CustomerObject, Nothing, DateTime.Now.AddMinutes(10),TimeSpan.Zero)，其中参数DateTime.Now.AddMinutes(10)提供了基于依赖性终止功能所需的绝对期满时间，数据项将在插入后十分钟被删除。

4.3 Application和Cache的比较

Application和Cache都可以用于应用程序状态管理，但它们也有不同特点。首先，基于依赖性的终止功能，使Cache能够精确的控制如何以及何时更新或清除缓存中的数据，这是Application不具备的。其次，Cache可以进行内部的锁定管理，因此不需要象Application那样明确的调用Lock()和Unlock()进行同步管理。最后，Cache的资源管理功能会自动删除Cache中的某些项目，因此在检索Cache项之前，一般需要首先判断该项是否还在Cache中，而Application中的项目不会被自动删除。

5 结束语

本文就ASP.NET中常用的一些状态管理方法进行了归纳分析。状态管理是Web程序设计中的一个重要问题和难点，在实际应用中，需要根据各种状态管理方式的应用范围和特点，以及应用程序的使用环境等多方面来考虑使用的状态管理方式。

参考文献

1 ASP.NET技术内幕/ [美] Scott Worley著，王文龙、刘湘宁译，人民邮电出版社，2002。

2 ASP.NET程序员参考手册/ [英] Jason Bell，[英] Mike Clark等著，赵彦敏译，清华大学出版社，2002。

3 ASP.NET高级编程/ [美] Richard Anderson，[美] Brian Francis等著，王毅、杨浩等译，清华大学出版社，2002。

4 ASP.NET从入门到精通/ [美] Chris Payne著，赵斌等译，人民邮电出版社，2002。