您的潜力，我们的动力

Microsoft
微软（中国）有限公司

ASP.NET状态管理

讲师：邵志东

msdn

MSDN Webcasts

课前准备

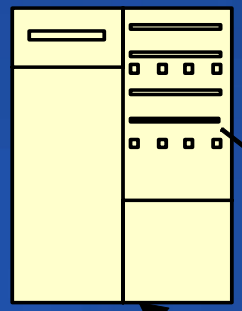
- Dot Net FrameWork
- VS.NET 2002/2003
- C#/VB.NET

- Level 200

议程

- 状态管理概述
- 基于客户端的状态管理
- 基于服务器的状态管理

ASP.NET工作原理



②Web 服务寻找指令文件(.aspx)

③ASP.NET 代码被发送给公共语言运行时进行编译

④HTML 流返回给浏览器和指令

①客户请求 Web 页

⑤浏览器处理 HTML 并显示页面

客户机

aspnet_isapi.dll

Http协议—“无状态协议”

- Web服务器每分钟对上千个用户进行管理的一种方式就是执行所谓的“无状态”连接。只要有一个希望浏览器返回一个页面、图象或其他资源的请求，就发生以下事情：
  - 连接到服务器
  - 告诉服务器想要的页面、图象或者其他项
  - 服务器发送请求的资源
  - 服务器切断连接，把用户忘的干干净净。

WEB页面处理过程

- 页面的一次往返处理：用户对Server Control的一次操作，就可能引起页面的一次往返处理：页面被提交到服务器端，执行响应的事件处理代码，重建页面，然后返回到客户端
- 页面重建：每一次页面被请求，或者页面事件被提交到服务器，asp.net运行环境将执行必要的代码，重建整个页面，把结果页面送到浏览器，然后抛弃页面的变量、控件的状态和属性等等页面信息。
- 页面处理内部过程：
  - Page_load ：IsPostBack属性判定页面是否为第一次被请求
  - 事件处理 ：这一阶段处理表单的事件
  - Page_Unload ：这个阶段页面已经处理完毕，需要做些清理工作。一般地，你可以在这个阶段关闭打开的文件和数据库链路，或者释放对象

ASP.NET Web Form 的“连续”和“有状态”假象

- ASP.NET 的设计者们，从实际访问者的角度重新考虑了这一过程：访问者打开一个页面，点击一个按钮，看到新的画面……这一切似乎都是连续的。
- 这种连续性假象是由 ASP.NET 页框架、页及其控件实现的。回发后，控件的行为必须看起来是从上次 Web 请求结束的地方开始的。
- 另一方面，对于 Web Form 中的 TextBox，ASP.NET 也让它们具有了状态，可以知道上一个 loop 和这一个 loop 之间的 TextBox 值的变化；如果变化，可能会触发 TextBox 的 TextChanged 事件。这同样是 ASP.NET 特意实现的一个假象。

议程

- 状态管理概述
- 基于客户端的状态管理
- 基于服务器的状态管理

基于客户端的状态管理

- 视图状态
- 隐藏的窗体域
- Cookie
- 查询字符串

2.1视图状态

ASP.NET 使用了 ViewState 视图状态，是所有控件的一个属性。如果你查看 Web Form 产生的 HTML 代码，可以看到一个名为 ViewState 的隐藏字段，ASP.NET 将状态信息以 Hash 的方式存储在这里。通过它，可以在下一次回发时知道回发前各控件的状态。

ASP.NET 服务器控件的生命周期

1. 初始化 - **Init** 事件 (OnInit 方法)
2. 加载视图状态 - **LoadViewState** 方法
3. 处理回发数据 - **LoadPostData** 方法
4. 加载 - **Load** 事件 (OnLoad 方法)
5. 发送回发更改通知 - **RaisePostDataChangedEvent** 方法
6. 处理回发事件 - **RaisePostBackEvent** 方法
7. 预呈现 - **PreRender** 事件 (OnPreRender 方法)
8. 保存视图状态 - **SaveViewState** 方法
9. 呈现 - **Render** 方法
10. 处置 - **Dispose** 方法
11. 卸载 - **UnLoad** 事件 (OnUnLoad 方法)

视图状态

- 启用视图状态
  EnableViewState = “true”
  默认为true，如果为false，那么该控件和子控件的视图状态就不会被串行化。
- 可以在视图状态中存储的类型
  Int32,Bool,String,Color,Array,ArrayList,Unit及其以上类型的HashTable对象。
- 视图状态与安全
  视图状态串行化的字符串表达式作为明文来往返传送。这是不安全的。 在视图状态中决不能保存任何信息（例如口令、连接字符、文件路径）。

DEMO1

视图状态

2.2隐藏域

- 隐藏域不会显示在用户的浏览器中，但我们可以象设置标准控制的属性那样设置其属性。当一个网页被提交给服务器时，隐藏域的内容和其他控制的值一块儿被送到HTTP Form集合中。隐藏域可以是任何存储在网页中的与网页有关的信息的存储库，隐藏域在其value属性中存储一个变量，而且必须被显性地添加在网页上。
- ASP.NET中的HtmlInputHidden控制提供了隐藏域的功能。

DEMO2

隐藏域

隐藏域使用注意事项

- 隐藏域在其值属性中存储单个变量，并且必须被显式添加到页上。然后可以将值插入到隐藏域。
- 为了在页处理期间隐藏域的值可供使用，必须使用 HTTP POST 方法提交该页。
- 恶意用户可以很容易地查看和修改隐藏域的内容。请不要在隐藏域中存储任何敏感信息或保障应用程序正确运行的信息。

2.3 Cookie集合

2.3.1 什么是Cookie?

- 由网络服务器发送出来以存储在网络浏览器上的小量信息
- Cookie是把与用户和网站相关的信息存储比会话时间还长的一种方式
- Cookie存储在用户的硬盘上（一般存储在Web浏览器软件所在的文件夹上，称为Cookies）

2.3.2 Cookie的用途

- 用户的个人配置
- 注册和“Remember me”
- 弹出窗口

2.3.4 Cookies是如何工作的

- IE存储在C:\Documents and Settings\<Username>\Cookies下
- IE选项中的“隐私”选项下修改cookie设置，也可以在“常规”选项卡下选择“删除cookie”

2.3.5 如何使用Cookie

- 使用Response对象设置Cookie状态
  - Response.Cookies[“UserName”].Value = “张三”;
- 使用Request对象读取已有的Cookie
  - string strName = Request.Cookies[“UserName].Value;
- 清除：

  Response.Cookies[“UserName”].Value=null;

  Response.Cookies[“UserName”].Expires= new System.DateTime(1999,10,12);

2.3.6 Cookie的属性

- Value:值，是String类型的
- Domain：设置这个属性后，只有在这个域下才能访问该Cookie。

  例如： Response.Cookies[“UserName”].Domain = “.Webcast.com.cn”;//指定只有以“.Webcast.com.cn”结尾的域可以访问本Cookie
- Path：该属性指定哪些路径下的页面可以访问此Cookie。
- Expires：指定Cookie过期的日期

DEMO3

Cookie的使用

2.4查询字符串

- 查询字符串提供了一种简单而受限制的维护状态信息的方法，我们可以方便地将信息从一个网页传递给另一个网页 。
- 带有查询字符串的URL如下所示：
  http://www.examples.com/list.aspx?categoryid=1&productid=101
- 使用：

```
string categoryid, productid;
categoryid=Request.Params[“categoryid”];
productid=Request.Params[“productid”];
```

DEMO4

Post方法和
查询字符串的使用

使用查询字符串的注意事项

- 大多数浏览器和客户端装置都把URL的长度限制在255个字符长。
- 查询值是通过URL传递给互联网的，因此，在有些情况下，安全就成了一个大问题。
- 我们只能使用HTTP-Get提交该互联网网页，否则就不能从查询字符串获得需要的值。

议程

- 状态管理概述
- 基于客户端的状态管理
- 基于服务器的状态管理

基于服务器的状态管理

信息存储在服务器上，尽管其安全性较高，但会占用较多的web服务器资源。 服务器端通常用以下方式实现状态管理：

- Aplication对象
- Session对象

3.1 Application状态

- 应用程序级别的状态存取
- 变量状态的存储和提取
  - Application[“Name”] = “张三”
  - string strUserName = Application[“Name”];
- 同时访问要加锁：
  - Application.Lock();
  - Application.Unlock();

DEMO5

Application对象的使用

Application对象的使用建议

- 对于频繁使用的数据使用改对象
- 不要把太多的信息放在该对象中
- 如果站点有很大的通信量，建议使用Web.Config

3.2、Session

3.2.1 什么是Session（会话）

- 对网站的一次访问
- 超时后，自动结束会话

3.2.2 什么时候用Session？

- 购物篮－网络用户决定购买的商品列表
- 用户信息－访问者的姓名
- 用户设置－个性化界面

ASP和ASP.NET会话区别

- ASP中用于标识会话的120位会话ID总是作为一个Cookie存储到浏览器中。所以一旦用户的安全策略禁用了Cookie，Session对象便无数据可用。
- ASP.NET的会话实现弥补了这个缺陷，它允许“无Cookie”的会话，以及在服务器之外存储会话数据。ASP.NET会话状态模块在Web.config文件中像下面这样配置：
  - <sessionState mode="InProc" cookieless="false" timeout="20" />
  - 在这个例子中，mode属性设为InProc（默认值），表明会话状态要由ASP.NET存储到内存中，而且不用Cookie来传递会话ID。

3.2.3 Session的属性和方法

- TimeOut属性：获取和设置会话结束之前的时间段，以分钟位单位。默认为20分钟。
- Abandon()：结束当前会话。会话中的所有信息都被清空
- Clear():删除当前会话中的所有信息，但不结束会话
- IsNewSession：如果会话是在用户访问页面时创建的，则这个属性返回true。当会话需要对某些数据进行初始化后才能使用时，就可以使用这个属性。

3.2.4 如何用Session存储状态

- 和Application类似
  - Session[“Name”] = “张三”;
- 和Application的区别：
  - Application：应用程序级别的状态存储
  - Session：会话级别的状态存储

DEMO6

Session的使用

Application和Session对象的区别

- 作用域不同
  - Application对象是针对所有用户都生效，
  - Session对象则相反，每个用户都有自己的Session对象，它的生命周期起始于服务器产生对用户请求页面的相应，终止于用户断开与服务器的连接。

3.3数据库

- 当存储特定于用户的信息并且信息存储较大时，通常的做法是使用数据库技术维护状态。对于维护长期的状态或维护即使在服务器必须重新启动的情况下仍必须保留的状态，数据库存储尤其有用。
- 数据库将使我们能够存储大量的与Web应用程序中的状态相关的信息，有时，用户会使用唯一的ID频繁地访问数据库，我们可以将它存储在数据库中，在对网站中网页的多次请求中使用。

小结

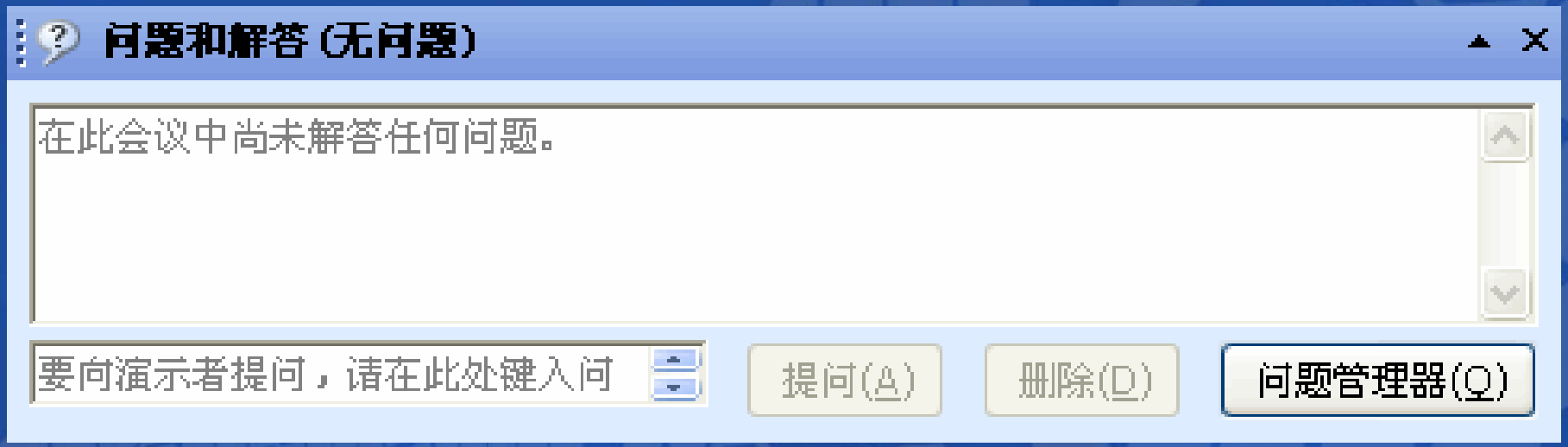
- 状态管理概述
- 基于客户端的状态管理
- 基于服务器的状态管理

更多信息.....

- MSDN网站： http://msdn.microsoft.com
- 程序员大本营:**www.csdn.net**

Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts