**C# Global.asax文件详解**

　　Global.asax 文件，有时候叫做 ASP.NET 应用程序文件，提供了一种在一个中心位置响应应用程序级或模块级事件的方法。你可以使用这个文件实现应用程序安全性以及其它一些任务。下面让我们详细看一下如何在应用程序开发工作中使用这个文件。   
  
　　**概述**   
  
　　Global.asax 位于应用程序根目录下。虽然 Visual Studio .NET 会自动插入这个文件到所有的 ASP.NET 项目中，但是它实际上是一个可选文件。删除它不会出问题——当然是在你没有使用它的情况下。.asax 文件扩展名指出它是一个应用程序文件，而不是一个使用 aspx 的 ASP.NET 文件。   
  
　　Global.asax 文件被配置为任何（通过 URL 的）直接 HTTP 请求都被自动拒绝，所以用户不能下载或查看其内容。ASP.NET 页面框架能够自动识别出对Global.asax 文件所做的任何更改。在 Global.asax 被更改后ASP.NET 页面框架会重新启动应用程序，包括关闭所有的浏览器会话，去除所有状态信息，并重新启动应用程序域。   
  
　　**编程**   
  
　　Global.asax 文件继承自HttpApplication 类，它维护一个HttpApplication 对象池，并在需要时将对象池中的对象分配给应用程序。Global.asax 文件包含以下事件：   
  
　　·Application\_Init：在应用程序被实例化或第一次被调用时，该事件被触发。对于所有的HttpApplication 对象实例，它都会被调用。   
  
　　·Application\_Disposed：在应用程序被销毁之前触发。这是清除以前所用资源的理想位置。   
  
　　·Application\_Error：当应用程序中遇到一个未处理的异常时，该事件被触发。   
  
　　·Application\_Start：在HttpApplication 类的第一个实例被创建时，该事件被触发。它允许你创建可以由所有HttpApplication 实例访问的对象。   
  
　　·Application\_End：在HttpApplication 类的最后一个实例被销毁时，该事件被触发。在一个应用程序的生命周期内它只被触发一次。   
  
　　·Application\_BeginRequest：在接收到一个应用程序请求时触发。对于一个请求来说，它是第一个被触发的事件，请求一般是用户输入的一个页面请求（URL）。   
  
　　·Application\_EndRequest：针对应用程序请求的最后一个事件。   
  
　　·Application\_PreRequestHandlerExecute：在 ASP.NET 页面框架开始执行诸如页面或 Web 服务之类的事件处理程序之前，该事件被触发。   
  
　　·Application\_PostRequestHandlerExecute：在 ASP.NET 页面框架结束执行一个事件处理程序时，该事件被触发。   
  
　　·Applcation\_PreSendRequestHeaders：在 ASP.NET 页面框架发送 HTTP 头给请求客户（浏览器）时，该事件被触发。   
  
　　·Application\_PreSendContent：在 ASP.NET 页面框架发送内容给请求客户（浏览器）时，该事件被触发。   
  
　　·Application\_AcquireRequestState：在 ASP.NET 页面框架得到与当前请求相关的当前状态（Session 状态）时，该事件被触发。   
  
　　·Application\_ReleaseRequestState：在 ASP.NET 页面框架执行完所有的事件处理程序时，该事件被触发。这将导致所有的状态模块保存它们当前的状态数据。   
  
　　·Application\_ResolveRequestCache：在 ASP.NET 页面框架完成一个授权请求时，该事件被触发。它允许缓存模块从缓存中为请求提供服务，从而绕过事件处理程序的执行。   
  
　　·Application\_UpdateRequestCache：在 ASP.NET 页面框架完成事件处理程序的执行时，该事件被触发，从而使缓存模块存储响应数据，以供响应后续的请求时使用。   
  
　　·Application\_AuthenticateRequest：在安全模块建立起当前用户的有效的身份时，该事件被触发。在这个时候，用户的凭据将会被验证。   
  
　　·Application\_AuthorizeRequest：当安全模块确认一个用户可以访问资源之后，该事件被触发。   
  
　　·Session\_Start：在一个新用户访问应用程序 Web 站点时，该事件被触发。   
  
　　·Session\_End：在一个用户的会话超时、结束或他们离开应用程序 Web 站点时，该事件被触发。   
  
　　这个事件列表看起来好像多得吓人，但是在不同环境下这些事件可能会非常有用。   
  
　　使用这些事件的一个关键问题是知道它们被触发的顺序。Application\_Init 和Application\_Start 事件在应用程序第一次启动时被触发一次。相似地，Application\_Disposed 和 Application\_End 事件在应用程序终止时被触发一次。此外，基于会话的事件（Session\_Start 和 Session\_End）只在用户进入和离开站点时被使用。其余的事件则处理应用程序请求，这些事件被触发的顺序是：   
  
　　·Application\_BeginRequest   
  
　　·Application\_AuthenticateRequest   
  
　　·Application\_AuthorizeRequest   
  
　　·Application\_ResolveRequestCache   
  
　　·Application\_AcquireRequestState   
  
　　·Application\_PreRequestHandlerExecute   
  
　　·Application\_PreSendRequestHeaders   
  
　　·Application\_PreSendRequestContent   
  
　　·<<执行代码>>   
  
　　·Application\_PostRequestHandlerExecute   
  
　　·Application\_ReleaseRequestState   
  
　　·Application\_UpdateRequestCache   
  
　　·Application\_EndRequest   
  
　　这些事件常被用于安全性方面。下面这个 C# 的例子演示了不同的Global.asax 事件，该例使用Application\_Authenticate 事件来完成通过 cookie 的基于表单（form）的身份验证。此外，Application\_Start 事件填充一个应用程序变量，而Session\_Start 填充一个会话变量。Application\_Error 事件显示一个简单的消息用以说明发生的错误。

|  |
| --- |
| protected void Application\_Start(Object sender, EventArgs e) {  　Application["Title"] = "Builder.com Sample";  }  protected void Session\_Start(Object sender, EventArgs e) {  　Session["startValue"] = 0;  }  protected void Application\_AuthenticateRequest(Object sender, EventArgs e) {  　// Extract the forms authentication cookie  　string cookieName = FormsAuthentication.FormsCookieName;  　HttpCookie authCookie = Context.Request.Cookies[cookieName];  　if(null == authCookie) {  　　// There is no authentication cookie.  　　return;  　}  　FormsAuthenticationTicket authTicket = null;  　try {  　　authTicket = FormsAuthentication.Decrypt(authCookie.Value);  　} catch(Exception ex) {  　　// Log exception details (omitted for simplicity)  　　return;  　}  　if (null == authTicket) {  　　// Cookie failed to decrypt.  　　return;  　}  　// When the ticket was created, the UserData property was assigned  　// a pipe delimited string of role names.  　string[2] roles  　roles[0] = "One"  　roles[1] = "Two"  　// Create an Identity object  　FormsIdentity id = new FormsIdentity( authTicket );  　// This principal will flow throughout the request.  　GenericPrincipal principal = new GenericPrincipal(id, roles);  　// Attach the new principal object to the current HttpContext object  　Context.User = principal;  }  protected void Application\_Error(Object sender, EventArgs e) {  　Response.Write("Error encountered.");  } |

　　这个例子只是很简单地使用了一些Global.asax 文件中的事件；重要的是要意识到这些事件是与整个应用程序相关的。这样，所有放在其中的方法都会通过应用程序的代码被提供，这就是它的名字为Global 的原因。   
  
　　这里是前面的例子相应的 VB.NET 代码：

|  |
| --- |
| Sub Application\_Start(ByVal sender As Object, ByVal e As EventArgs)  　Application("Title") = "Builder.com Sample"  End Sub  Sub Session\_Start(ByVal sender As Object, ByVal e As EventArgs)  　Session("startValue") = 0  End Sub  Sub Application\_AuthenticateRequest(ByVal sender As Object, ByVal e As  EventArgs)  　’ Extract the forms authentication cookie  　Dim cookieName As String  　cookieName = FormsAuthentication.FormsCookieName  　Dim authCookie As HttpCookie  　authCookie = Context.Request.Cookies(cookieName)  　If (authCookie Is Nothing) Then  　　’ There is no authentication cookie.  　　Return  　End If  　Dim authTicket As FormsAuthenticationTicket  　authTicket = Nothing  　Try  　　authTicket = FormsAuthentication.Decrypt(authCookie.Value)  　　Catch ex As Exception  　　’ Log exception details (omitted for simplicity)  　　Return  　End Try  　Dim roles(2) As String  　roles(0) = "One"  　roles(1) = "Two"  　Dim id As FormsIdentity  　id = New FormsIdentity(authTicket)  　Dim principal As GenericPrincipal  　principal = New GenericPrincipal(id, roles)  　’ Attach the new principal object to the current HttpContext object  　Context.User = principal  End Sub   Sub Application\_Error(ByVal sender As Object, ByVal e As EventArgs)  　Response.Write("Error encountered.")  End Sub |

　　**资源**   
  
　　Global.asax 文件是 ASP.NET 应用程序的中心点。它提供无数的事件来处理不同的应用程序级任务，比如用户身份验证、应用程序启动以及处理用户会话等。你应该熟悉这个可选文件，这样就可以构建出健壮的ASP.NET 应用程序。