## **ASP.NET v2.0 - AppDomain recycles, more common than before**

I stumbled across a very interesting find yesterday morning and [Todd Carter](http://blogs.msdn.com/toddca/)confirmed it by sending a [link to his blog entry](http://blogs.msdn.com/toddca/archive/2005/12/01/499144.aspx).  (it's a good read)

First let me cover briefly what an AppDomain is. ASP.NET (.NET Winforms too but I won't cover that here) starts and runs within a sandbox called an AppDomain. Each Application on a server will run its own AppDomain which allows separation between different parts of the site for better security isolation and availability. If a folder is marked as an application, it will live within a different AppDomain than the root of the site. This information is not easily seen in IIS6 and earlier versions of IIS, but AppDomains are there just the same.

What this means though is that if the AppDomain is restarted, then there is a performance hit while the AppDomain is started up and pages are recompiled on their first hit. Furthermore, session state that is running In-Process is lost. In-Process, or "InProc" session state is the default setting.

In ASP.NET v1.x, there is an AppDomain restart when global.asax, web.config or anything related to the /bin folder is 'touched'. By 'touched', I mean renaming, creating, or updating the file in any way. And of course an IIS6 application pool recycle or a restart of IIS would cause the old AppDomain to shut down and a new one to start. This all made sense because those were application files and any change to them would make significant changes to the running AppDomain.

While troubleshooting an unrelated issue on a v2.0 site yesterday, I noticed that there was an AppDomain restart simply by creating a non-application folder. Through further testing I found that adding folders, deleting folders and renaming folders caused the AppDomain recycle. Furthermore, if many dozen files were added in short succession to each other, that would also case the same restart. Of course the app\_{folders} will cause a restart, and that makes sense, but completely non-application related folders also cause this same behavior.

It turns out that this was a purposeful change with v2.0 and was done so that stale content doesn't remain in place after a significant change. I guess there were enough situations where that happened that they changed the situations that caused an AppDomain recycle to be far more aggressive than v1.x of the framework.

What does this mean? Unfortunately quite a bit. People that used to make quick updates to their websites thinking that it wouldn't affect performance of unrelated parts of the site now need to consider that any changes to folders will cause the entire AppDomain to be restarted. Also a large number of changes to files within the site will do the same. (I'm not sure at this time the threshold that triggers a restart based on file changes). This will immediately kill all InProc session state and cause a performance hit on the site.

I'm normally not one to complain about .NET, there are so many great things to say about it. But in this situation, I think this is a step backward. At this time, there is no way to disable this aggressive response to file and folder changes. The linkd.exe solution that Todd gave in the blog entry is a solution, but it's not an easy one and isn't reasonable in many situations.

I don't have a good solution for this, but at least I am now aware of the affects of various file and folders changes and can watch out for that when making site updates or troubleshooting sites.

**Added later:**It's now possible to disable file change notifications completely by making a registry change and restarting IIS.   The downside to disabling it completely is that whenever you make a change to web.config or other system folder, you will need to restart the app pool manually, otherwise the settings won't take effect.  See the registry information section of this page:<http://support.microsoft.com/kb/911272>

This forums thread is another good one and covers how to make a global.asax post that will disable just some notifications while leaving the core ones in place.  I haven't tested it personally but another person on the thread confirmed that it worked: <http://forums.asp.net/p/1310976/2581558.aspx>

Another potential option is to delay an AppDomain recycle when pushing changes to the server so that you can get a full upload of your content before it starts recycling.  Otherwise you may get rapid recycles during deployment:<http://msdn.microsoft.com/en-us/library/e1f13641.aspx>  Search for maxWaitChangeNotification and waitChangeNotification.

[Tess](http://blogs.msdn.com/tess/default.aspx) has a good post on AppDomain recycles now too which is a good read:<http://blogs.msdn.com/tess/archive/2006/08/02/asp-net-case-study-lost-session-variables-and-appdomain-recycles.aspx>

Posted: [Feb 21 2006, 09:20 AM](http://weblogs.asp.net/owscott/archive/2006/02/21/ASP.NET-v2.0-_2D00_-AppDomain-recycles_2C00_-more-common-than-before.aspx) by [OWScott](http://weblogs.asp.net/members/OWScott.aspx) | with [36 comment(s)](http://weblogs.asp.net/owscott/archive/2006/02/21/ASP.NET-v2.0-_2D00_-AppDomain-recycles_2C00_-more-common-than-before.aspx#comments)

Filed under: [ASP.NET](http://weblogs.asp.net/owscott/archive/tags/ASP.NET/default.aspx)