Cache vs Session state in ASP.NET: Replace Session with Cache

Published 9/3/2010 by Vivek Thakur

Most web applications use both Session state as well as the Cache. We all know that ASP.NET Cache is more of a global storage cauldron for in-memory objects, where as the Session object is limited to each user. Now in a recent project, we faced a tricky issue with our internal Session usage. We were storing user roles inside the session, and wanted to refresh the role list when the user-role assignment changes via the admin section. Since Session objects cannot be accessed globally, there was no decent way to refresh the contents stored there. Besides this, we face other issues too with the Session object:

1. Managing sessions in a web farm is very tricky indeed. I dont like the concept of "sticky" sessions and other kludges to make them work across a farm. And out-proc Session storage was not suitable for our project, besides I would not really use it because of the performance strain it can cause.

2. Session data and its expiry is not very flexible.

Don't get me wrong, Sessions can work great for some projects, but in our scenario we had to find an alternative as we wanted to turn off the session state completely. We found our savior in the Cache object! It was perfect, and given the flexiblity it offers, we could use it in any way we want. We used global events to refresh cached data in the memory, and the multitude of expiry techniques made our life much easier. We turned off the session state (in web.config) and realized we had more flexibility apart from being able to scale in a web farm scenario. So I would recommend going for a Cache provider framework if you feel session state is not for fit for your project. And the best part is, starting ASP.NET 4.0 the Cache object is abstracted out from System.Web assembly into its own separate namespace, System.Runtime.Caching. This makes it possible for you to use the same framework in a non-web environment!

tags : asp, Cache, net, session state