

# Re: How do Large Scale Web Service Applications Maintain Session State?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2007-02/msg00051.htm>

---

- *From:* "Joseph Geretz" <[jgeretz@xxxxxxxxxx](mailto:jgeretz@xxxxxxxxxx)>
  - *Date:* Sat, 10 Feb 2007 19:33:52 -0500
- 

Hi John,

Here's are a few specific example, in our case. First of all, our application implements application defined authentication. To keep every transaction absolutely stateless, this will require authentication on every single application call. Won't it be much more efficient to allow the client application to authenticate once against the database and then to maintain that authenticated state for every transaction which is submitted by that authenticated client, until the client logs off, or the time-out period expires?

Second, and tangentially related issue, our application defines detailed user profiles which define what a particular user can and cannot do. Again, absolute statelessness on the Server would require us to check the database on every single transaction to authorize the transaction. Our concept is to cache these profiles on the server in order to increase performance.

Naturally, I agree with you that specific application classes (i.e. classes providing direct services to application clients) on the server should not maintain state, since this limits their usability to a single client. However, the caching of session state on the server is a different issue and can have tremendous benefits for performance, without compromising scalability, if done correctly (as far as I can imagine). Our proposed architecture would implement a stateful Web Service server application, which is something different than stateful Web Service classes.

Again, if this is fundamentally a bad idea, I'd be interested to learn why. But if others have implemented server side caching successfully, I am interested to hear about that as well.

Thanks!

– Joseph Geretz –

"John Saunders" <[john.saunders@trizetto.com](mailto:john.saunders@trizetto.com)> wrote in message [news:e9f0XyWTHHA.4744@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:e9f0XyWTHHA.4744@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Re: How do Large Scale Web Service Applications Maintain Session State?

Radek,

Perhaps you could describe, in general, what it was about your application which prevented you from using a stateless model?

Thanks,  
John

"Radek Cerny" <radek.cerny@xxxxxxxxxxxxxxxxxxxx> wrote in message  
[news:uJzhqoVTHHA.4756@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:uJzhqoVTHHA.4756@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Joseph,

I completely understand your puzzlement... we went through the same issues and finally decided that we will build stateful server sessions, and have been very happy with the results. We do pass a session token as a parameter on every call, and have only 1 asmx file with a handful of methods (public – there are another 20 or so administrative and instrumentation).

There is a whitepaper showing the topology, but there is much more as well.

<http://www.c1s.com.au/C1prod/files/Whitepaper.pdf>

Having systems deployed through web services allows clients to access via rich [thin] clients, browsers via AJAX, etc.

Cheers,

Radek

"Joseph Geretz" <jgeretz@xxxxxxxxxx> wrote in message  
[news:Oe2Gk0HTHHA.2212@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:Oe2Gk0HTHHA.2212@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Doesn't anyone use server side caching to manage session state?

If there's something fundamentally wrong with this approach, please let me know.

Thanks!

– Joseph Geretz –

"Joseph Geretz" <jgeretz@xxxxxxxxxx> wrote in message  
[news:unvWicBTHHA.4252@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:unvWicBTHHA.4252@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

I've been looking at two approaches for the maintenance of Session state for a Web Service application.

## Re: How do Large Scale Web Service Applications Maintain Session State?

One approach uses the old familiar Session object which I've used in the past for Web applications. As far as I can see, the Session approach is non-standard since Web Services are supposed to be agnostic with respect to their clients. It seems that cookies are outside the Web Service standard; therefore, such a Web Service application won't work for those clients which are not equipped to shuttle the Session cookie back and forth.

The second approach I have researched uses the Context.Cache object, plus a unique session ID which is shuttled back and forth as a parameter on every method call. The unique session ID is of course, the index into the Cache object for the retrieval of session related data.

I see two drawbacks to this approach. The first issue, is that this approach affects the parameter signature practically every single public method in the entire application. OK, this is perhaps a minor detail. For all I know, this may be standard practice?

The second issue however, is that for a large scale Web Services application, supporting thousands of concurrent sessions, the Cache memory consumption on the server is liable to be quite large and this could be a problem.

So I'm wondering, what is the best way to do this?

Tangentially, I'm also curious how you typically engineer your Web Service applications. Would the server side application consist of a single ASMX page with all application methods in a single page, or would you break your application into separate pages? I'm thinking that

Re: How do Large Scale Web Service Applications Maintain Session State?

from the standpoint of team development  
alone, the latter method is the  
correct approach.

I appreciate any advice which you can offer.

Thanks!

Joseph Geretz