Static fields vs Session variables

Asked 11 years, 10 months ago Modified 7 years, 11 months ago

Viewed 22k times

33

So far I've been using Session to pass some variables from one page to another. For instance user role. When a user logs in to the web application the role id of the user is kept in Session and that role is checked at different parts of the application. I've recently started thinking why not use static members instead. I can store the same information in a static field and easily access it anywhere in my application (well anywhere the namespace in which that static field resides is included.) I know that using Session variables comes handy sometimes, such that:

1. Any kind of data can be stored in Session. It then must be casted however.But static fields accept data with the correct datatype only.
2. Session variables will expire after a certain time which is the behavior we need in many cases.

Apart from the above, are there any other reasons why I should not use static fields to store data and have it available everywhere?

c# asp.net .net session-variables static-members

Share

Improve this question

Follow

edited Feb 6, 2013 at 7:26

Jacek

12k ● 25 ● 75 ● 127

asked Feb 6, 2013 at 7:09

Mikayil Abdullayev

12.4k ● 27 ● 129 ● 214

Hi, you can refer to stackoverflow.com/questions/1563171/… – whastupduck Feb 6, 2013 at 7:12

3 don't ever used static field for user related data. They should only be used where you have to maintain something regarding Application wide – Kamran Shahid Feb 6, 2013 at 7:17

4 Answers

Sorted by: Highest score (default)

47

No, using static variables for this is *not* the way to go:

- If your AppDomain is recycled, all your static variables will be "reset"
- Static variables don't scale horizontally - if you load-balance your application, a user who hits one server then a different one won't see the data store in the static variables in the first server
- Most importantly, static variables will be shared by *all* access to that server... it won't be on a per-user basis at all... whereas from your description, you wouldn't want user X to see user Y's information.

Fundamentally, you have two choices for propagating information around your application:

- Keep it client-side, so each request gives the information from the previous steps. (This can become unwieldy with large amounts of information, but can be useful for simple cases.)
- Keep it server-side, ideally in some *persistent* way (such as a database) with the client providing a session identifier.

If you can use load-balancing to keep all users going to the same server, and *if* you don't mind sessions being lost when the AppDomain is recycled[1] or a server going down, you can keep it in memory, keyed by session ID... but be careful.

---

[1] There may be mechanisms in ASP.NET to survive this, propagating session information from one AppDomain to another - I'm not sure

Share Improve this answer

Follow

edited Feb 5, 2016 at 9:22

Rahul Nikate

6,327 ● 5 ● 43 ● 56

answered Feb 6, 2013 at 7:13

Jon Skeet

1.5m ● 889 ● 9.3k ● 9.3k

---

3 I now realize that I have to make so many amendments to my code because I've used static fields in various places. Thank

you Jon for broad explanation. – Mikayil Abdullayev Feb 6, 2013 at 7:22

13

They are two very different things.

- Session can be used out of process (important for load balancing)
- Session can be more durable because of its out of process capabilities.
- ASP.Net automatically manages Session concurrency.
- Access to static variables needs to be manually synchronized.
- **Static is global to the entire app domain.** If you set the value of a static field/property for one user, all users will get the same value. Not the desired behavior in your scenario.

> Any kind of data can be stored in Session. It then must be casted however.But static fields accept data with the correct datatype only.

It's often helpful to abstract Session values with a helper class. This can improve testability and also allows you to strongly type properties and perform the cast in the internals of the class.

Example:

```
public List<int> UserRoles
{
    get
    {
        // optionally check that the value is indeed i
        // will throw
        return (List<int>)Session["UserRoles"];
    }
}
```

See also:

- C# Static variables - scope and persistence
- does aspx provide special treatment for c# static variables

Share Improve this answer

Follow

edited May 23, 2017 at 12:17

Community Bot

1 ● 1

answered Feb 6, 2013 at 7:12

Tim M.

54.3k ● 14 ● 124 ● 166

3 You saved me from a catastrophe. – Mikayil Abdullayev Feb 6, 2013 at 7:16

9

You forget one thing (I think)

Static data will be the same for all users of the application, while session is "per user".

So for your "User Role" scenario, I would expect funny results ;)

Share Improve this answer

Follow

answered Feb 6, 2013 at 7:12

Raphaël Althaus

60.5k ● 6 ● 104 ● 125

6

Static fields would be shared across all users.
In as web environment, you will have several threads running together.

Updating any static members would need proper concurrency control. Done wrong, this would slow down your site performance significantly.

Sessions can be moved out of process and shared across a web farm.

Out of proc sessions would exist even if your app server crashed.

Share Improve this answer

Follow

answered Feb 6, 2013 at 7:12

nunespascal

17.7k ● 2 ● 44 ● 46

+1 for mentioning concurrency. Even if a static member is the right choice (sometimes they are good for shared data) access must be synchronized properly. – Tim M. Feb 6, 2013 at 7:18