Does This ASP.NET Consultant Know What He's Doing?

Asked 16 years, 2 months ago Modified 14 years, 9 months ago Viewed 2k times

9

The IT department of a subsidiary of ours had a consulting company write them an ASP.NET application. Now it's having intermittent problems with mixing up who the current user is and has been known to show Joe some of Bob's data by mistake.

The consultants were brought back to troubleshoot and we were invited to listen in on their explanation. Two things stuck out.

First, the consultant lead provided this pseudo-code:

```
void MyFunction()
{
    Session["UserID"] = SomeProprietarySessionManagementLookup();
    Response.Redirect("SomeOtherPage.aspx");
}
```

He went on to say that the assignment of the session variable is asynchronous, which seemed untrue. Granted the call into the lookup function could do something asynchronously, but this seems unwise.

Given that alleged asynchronousness, his theory was that the session variable was not being assigned before the redirect's inevitable ThreadAbort exception was raised. This faulure then prevented SomeOtherPage from displaying the correct user's data.

Second, he gave an example of a coding best practice he recommends. Rather than writing:

```
int MyFunction(int x, int x)
{
    try
    {
        return x / y;
    }
    catch(Exception ex)
    {
        // log it
        throw;
    }
}
```

the technique he recommended was:

```
int MyFunction(int x, int y, out bool isSuccessful)
{
  isSuccessful = false;

  if (y == 0)
      return 0;

  isSuccessful = true;

  return x / y;
}
```

This will certainly work and could be better from a performance perspective in some situations.

However, from these and other discussion points it just seemed to us that this team was not well-versed technically.

Opinions?

c# asp.net exception session-variables

Share
Improve this question
Follow

edited Oct 2, 2008 at 20:59
Ross
47k ● 39 ● 123 ● 173

asked Oct 2, 2008 at 20:37
marc
3,328 ● 5 ● 30 ● 33

22 Answers

Sorted by: Highest score (default)

19

Rule of thumb: If you need to ask if a consultant knows what he's doing, he probably doesn't ;)

And I tend to agree here. Obviously you haven't provided much, but they don't seem terribly competent.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:39
Serafina Brocious
30.6k ● 12 ● 91 ● 115

it looks like they demonstrated quite clearly that they are not competent - fire them and hire me instead ;-) – Steven A. Lowe Oct 2, 2008 at 21:28

1 Often I've found that people who need to ask don't know what they are doing either. If you don't *know* that the guy is wrong, you don't know what you are doing either. :) – brian d foy Oct 6, 2008 at 1:01

"if you need to ask ..." doesn't necessarily mean the guy doesn't know what he'd doing, but it does indicate a problem in your relationship with this consultant that need to be resolved.
– John MacIntyre Jan 27, 2009 at 23:08

14

I would agree. These guys seem quite incompetent.

(BTW, I'd check to see if in "SomeProprietarySessionManagementLookup," they're using static data. Saw this -- with behavior *exactly as you describe* on a project I inherited several months ago. It was a total head-slap moment when we finally saw it ... And wished we could get face to face with the guys who wrote it ...)

Share Improve this answer Follow

answered Oct 2, 2008 at 20:43

John Rudy
37.8k ● 14 ● 66 ● 101

Thanks for the input. They don't seem overly interested in what we think but I'll put it on the table. Thus far we've not been invited to look at any source code. – marc Oct 2, 2008 at 20:49

1 Agreed ... If ever refused to show code to a client who was paying for it, I'd get fired. And probably beaten up with a rubber chicken. – John Rudy Oct 2, 2008 at 20:58

I think it's mostly our "child" organization that doesn't want to share the code - a turf thing...
– marc Oct 2, 2008 at 21:14

1 +1 Asynchronous is nonsense, whatever the method is doing, the Session variable is set synchronously. Static variables, possibly from a developer who's discovered the singleton pattern are the reason. Don't ask for source: look at the code with Lutz Reflector!
– to StackOverflow Oct 3, 2008 at 17:47

Asynchronous might not be total nonsense, but the consultant-provided pseudocode is wrong, but it's pseudocode after all. See my answer for more info. – Robert Paulson Oct 6, 2008 at 0:55

12

If the consultant has written an application that's supposed to be able to keep track of users and only show the correct data to the correct users and it doesn't do that, then clearly something's wrong. A good consultant would find the problem and fix it. A bad consultant would tell you that it was asynchronicity.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:42

Randy
4,023 ● 21 ● 25

Exactly. Who cares what the problem really is. The only real answer here is for the consultant to fix it. – ChrisLively Oct 2, 2008 at 21:21

8

On the asynchronous part, the only way that could be true is if the assignment going on there is actually an indexer setter on Session that is hiding an asynchronous call with no callback indicating success/failure. This would seem to be a HORRIBLE design choice, and it looks like a core class in your framework, so I find it highly unlikely.

Usually asynchronous calls have a way to specify a callback so you can determine what the result is, or if the operation was successful. The documentation for Session should be pretty clear though on if it is actually hiding an asynchronous call, but yeah... doesn't look like the consultant knows what he is talking about...

The method call that is being assigned to the Session indexer cannot be asynch, because to get a value asynchronously, you HAVE to use a callback... no way around that, so if there is no explicit callback, it's definitely not asynch (well, internally there could be an asynchronous call, but the caller of the method would perceive it as synchronous, so it is irrelevant if the method internally for example invokes a web service asynchronously).

For the second point, I think this would be much better, and keep the same functionality essentially:

```
int MyFunction(int x, int y)
{
    if (y == 0)
    {
        // log it
        throw new DivideByZeroException("Divide by zero attempted!");
    }

    return x / y;
}
```

Share

Improve this answer

Follow

edited Oct 2, 2008 at 21:02

answered Oct 2, 2008 at 20:57

Mike Stone

44.6k ● 30 ● 114 ● 140

I'm pretty sure he's talking about ASP.NET Session which doesn't behave that way. – Cade Roux Oct 2, 2008 at 21:04

Yeah, I just don't have the docs open and I wanted to be clear that the assignment COULD be asynchronous, but more than likely isn't (and easy to tell if it isn't) – Mike Stone Oct 2, 2008 at

21:06

FYI, with regards to throwing an exception in MyFunction... don't bother with performance of exceptions unless you are expecting an exception on every invoke in like a tight loop that is executed a lot... microperformance optimization like that makes the code unmaintainable – Mike Stone Oct 2, 2008 at 21:34

From the OP's example, the spec seems to define dividing by zero to return 0 (hence, no need to throw the exception... otherwise you wouldn't need try/catch at all; just return x/y; and let the framework throw any exceptions) – Steven Evers Mar 8, 2010 at 20:50

5

For the first point, that does indeed seem bizarre.

On the second one, it's reasonable to try to avoid division by 0 - it's entirely avoidable and that avoidance is simple. However, using an out parameter to indicate success is only reasonable in certain cases, such as int.TryParse and DateTime.TryParseExact - where the caller can't easily determine whether or not their arguments are reasonable. Even then, the return value is usually the success/failure and the out parameter is the result of the method.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:40

Jon Skeet
1.5m ● 889 ● 9.3k ● 9.3k

Yeah, avoiding exceptions at all costs is just silly. They oftentimes make the code a lot easier to understand and thus easier to maintain, so trying to skirt around the exception like that to me smells of someone unnecessarily doing micro-performance optimization (a BAD THING) – Mike Stone Oct 2, 2008 at 21:37

4

Asp.net sessions, if you're using the built-in providers, won't accidentally give you someone else's session. `SomeProprietarySessionManagementLookup()` is the likely culprit and is returning bad values or just not working.

> ```
> Session["UserID"] = SomeProprietarySessionManagementLookup();
> ```

First of all assigning the return value from an asynchronously SomeProprietarySessionManagementLookup() just wont work. The consultants code probably looks like:

```
public void SomeProprietarySessionManagementLookup()
{
    // do some async lookup
    Action<object> d = delegate(object val)
    {
```

```
        LookupSession(); // long running thing that looks up the user.
        Session["UserID"] = 1234; // Setting session manually
    };

    d.BeginInvoke(null,null,null);
}
```

The consultant isn't totally full of BS, but they have written some buggy code. Response.Redirect() does throw a ThreadAbort, and if the proprietary method is asynchronous, asp.net *doesn't know to wait* for the asynchronous method to write back to the session before asp.net itself saves the session. This is probably why it sometimes works and sometimes doesn't.

Their code might work if the asp.net session is in-process, but a state server or db server wouldn't. It's timing dependent.

I tested the following. We use state server in development. This code works because the session is written to before the main thread finishes.

```
Action<object> d = delegate(object val)
{
    System.Threading.Thread.Sleep(1000);  // waits a little
    Session["rubbish"] = DateTime.Now;
};

d.BeginInvoke(null, null, null);
System.Threading.Thread.Sleep(5000);      // waits a lot

object stuff = Session["rubbish"];
if( stuff == null ) stuff = "not there";
divStuff.InnerHtml = Convert.ToString(stuff);
```

This next snippet of code **doesn't work** because the session was already saved back to state server by the time the asynchronous method gets around to setting a session value.

```
Action<object> d = delegate(object val)
{
    System.Threading.Thread.Sleep(5000);  // waits a lot
    Session["rubbish"] = DateTime.Now;
};

d.BeginInvoke(null, null, null);

// wait removed - ends immediately.
object stuff = Session["rubbish"];
if( stuff == null ) stuff = "not there";
divStuff.InnerHtml = Convert.ToString(stuff);
```

The first step is for the consultant to make their code synchronous because their *performance trick* didn't work at all. If that fixes it, have the consultant properly implement using the Asynchronous Programming Design Pattern

Share
Improve this answer
Follow

edited Mar 8, 2010 at 20:29

answered Oct 2, 2008 at 21:46
Robert Paulson
18k ● 6 ● 36 ● 53

I agree with you on this "Asp.net sessions, if you're using the built-in providers, won't accidentally give you someone else's session. SomeProprietarySessionManagementLookup() is the likely culprit and is returning bad values or just not working" – Saif Khan Oct 6, 2008 at 1:45

I know this question is really old but I imagine you're giving that developer/team wayyyyy too much credit. – Chris Marisic Mar 8, 2010 at 13:48

2

I agree with him in part -- it's definitely better to check y for zero rather than catching the (expensive) exception. The out bool isSuccessful seems really dated to me, but whatever.

re: the asynchronous sessionid buffoonery -- may or may not be true, but it sounds like the consultant is blowing smoke for cover.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:43
Danimal
7,710 ● 8 ● 48 ● 57

A valid point, but the code is still low quality. If you're going to allow a 0 divsor, then catch it before the division and log it, but don't count on someone writing client code to "do the right thing" when a false success flag is passed back to them. If you're going to require that client code reacts to the success flag, why not just require that client code not pass a 0 in the first place? If that were part of the interface contract, then an unhandled exception would be the proper way to handle div/zero. – Paul Keister Mar 8, 2010 at 20:49

1

Cody's rule of thumb is dead right. If you have to ask, he probably doesn't.

It seems like point two its patently incorrect. .NET's standards explain that if a method fails it should throw an exception, which seems closer to the original; not the consultsant's suggestion. Assuming the exception is accurately & specifically describing the failure.

Share
Improve this answer
Follow

edited May 23, 2017 at 12:14
Community Bot
1 ● 1

answered Oct 2, 2008 at 20:44
Steve Duitsman
2,781 ● 5 ● 27 ● 39

1

The consultants created the code in the first place right? And it doesn't work. I think you have quite a bit of dirt on them already.

The asynchronous answer sounds like BS, but there may be something in it. Presumably they have offered a suitable solution as well as pseudo-code describing the problem they themselves created. I would be more tempted to judge them on their solution rather than their expression of the problem. If their understanding is flawed their new solution won't work either. Then you'll know they are idiots. (In fact look round to see if you have a similar proof in any other areas of their code already)

The other one is a code style issue. There are a lot of different ways to cope with that. I personally don't like that style, but there will be circumstances under which it is suitable.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:48
Simon
80.6k ● 26 ● 92 ● 119

1

They're wrong on the async.

The assignment happens and then the page redirects. The function can start something asynchronously and return (and could even conceivably alter the Session in its own way), but whatever it does return has to be assigned in the code you gave before the redirect.

They're wrong on that defensive coding style in any low-level code and even in a higher-level function unless it's a specific business case that the 0 or NULL or empty string or whatever should be handled that way - in which case, it's always successful (that successful flag is a nasty code smell) and not an exception. Exceptions are for exceptions. You don't want to mask behaviors like this by coddling the callers of the functions. Catch things early and throw exceptions. I think Maguire covered this in Writing Solid Code or McConnell in Code Complete. Either way, it smells.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:52
Cade Roux
89.6k ● 40 ● 184 ● 266

FYI, C# has indexers that can hide assignments as actual methods, and so the Session[]= call could potentially fire off an asynch assignment... but this is unbelievably unlikely, and easy to determine by looking at the documentation (this would NOT be hidden from the docs).
– Mike Stone Oct 2, 2008 at 20:59

I'm pretty sure he's talking about ASP.NET Session which doesn't behave that way.
– Cade Roux Oct 2, 2008 at 21:03

1

This guy does not know what he is doing. The obvious culprit is right here:

```
Session["UserID"] = SomeProprietarySessionManagementLookup();
```

Share Improve this answer Follow

answered Oct 2, 2008 at 21:19
MusiGenesis
75.3k ● 41 ● 197 ● 338

The consultant blamed ASP.NET for the swapped session information while inaccurately stating that session variables are assigned asynchronously. Far more likely that the proprietary session management code is flawed (in addition to being totaly superfluous).
– MusiGenesis Oct 6, 2008 at 2:15

Believe me, if ASP.NET's built-in session management system were prone to intermittently swapping user data, the uproar over this would have been tremendous. – MusiGenesis Oct 6, 2008 at 2:17

0

I have to agree with John Rudy. My gut tells me the problem is in SomeProprietarySessionManagementLookup().

.. and your consultants do not sound to sure of themselves.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:51
phreakre
197 ● 1 ● 1 ● 7

0

Storing in Session in not async. So that isn't true unless that function is async. But even so, since it isn't calling a BeginCall and have something to call on completion, the next line of code wouldn't execute until the Session line is complete.

For the second statement, while that could be used, it isn't exactly a best practice and you have a few things to note with it. You save the cost of throwing an exception, but wouldn't you want to know that you are trying to divide by zero instead of just moving past it?

I don't think that is a solid suggestion at all.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:53

Tom
1,611 ● 10 ● 11

Quite strange. On the second item it may or may not be faster. It certainly isn't the same functionality though.

0

Share Improve this answer Follow

answered Oct 2, 2008 at 21:00

BlackWasp
4,961 ● 2 ● 33 ● 42

Typical "consultant" bollocks:

0

1. The problem is with whatever SomeProprietarySessionManagementLookup is doing
2. Exceptions are only expensive if they're thrown. Don't be afraid of `try..catch`, but throws should only occur in *exceptional* circumstances. If variable `y` **shouldn't** be zero then an `ArgumentOutOfRangeException` would be appropriate.

Share
Improve this answer
Follow

edited Oct 2, 2008 at 21:07

answered Oct 2, 2008 at 20:50

Duncan Smart
32k ● 11 ● 70 ● 72

but an exception is indeed thrown when you try to divide by zero -- the call stack is wound up, etc. – Danimal Oct 2, 2008 at 20:52

I'm guessing your consultant is suggesting use a status variable instead of exception for error handling is a better practice? I don't agree. How often does people forgot or too lazy to do error checking for return values? Also, pass/fail variable is not informative. There are more things can go wrong other than divide by zero like integer x/y is too big or x is NaN. When things go wrong, status variable cannot tell you what went wrong, but exception can. Exception is for exceptional case, and divide by zero or NaN are definitely exceptional cases.

0

Share Improve this answer Follow

answered Oct 2, 2008 at 21:16

Alvin
10.4k ● 9 ● 39 ● 50

0

The session thing is possible. It's a bug, beyond doubt, but it could be that the write arrives at whatever custom session state provider you're using after the next read. The session state provider API accommodates locking to prevent this sort of thing, but if the implementor has just ignored all that, your consultant could be telling the truth.

The second issue is also kinda valid. It's not quite idiomatic - it's a slightly reversed version of things like int.TryParse, which are there to avoid performance issues caused by throwing lots of exceptions. But unless you're calling that code an awful lot, it's unlikely it'll make a noticeable difference (compared to say, one less database query per page etc). It's certainly not something you should do by default.

Share Improve this answer Follow

answered Oct 2, 2008 at 21:39

Thom
2,483 ● 2 ● 30 ● 34

0

If SomeProprietarySessionManagementLookup(); is doing an asynchronous assignment it would more likely look like this:

```
SomeProprietarySessionManagementLookup(Session["UserID"]);
```

The very fact that the code is assigning the result to Session["UserID"] would suggest that it is not supposed to be asynchronous and the result should be obtained before Response.Redirect is called. If SomeProprietarySessionManagementLookup is returning before its result is calculated they have a design flaw anyway.

The throw an exception or use an out parameter is a matter of opinion and circumstance and in actual practice won't amount to a hill of beans which ever way you do it. For the performance hit of exceptions to become an issue you would need to be calling the function a huge number of times which would probably be a problem in itself.

Share Improve this answer Follow

answered Oct 3, 2008 at 13:42

Martin Brown
25.3k ● 16 ● 85 ● 130

If the consultants deployed their ASP.NET application on your server(s), then they may have deployed it in uncompiled form, which means there would be a bunch of

0

*.cs files floating around that you could look at.

If all you can find is compiled .NET assemblies (DLLs and EXEs) of theirs, then you should still be able to decompile them into somewhat readable source code. I'll bet if you look through the code you'll find them using static variables in their proprietary lookup code. You'd then have something very concrete to show your bosses.

Share Improve this answer Follow

answered Oct 6, 2008 at 23:02

MusiGenesis
75.3k ● 41 ● 197 ● 338

0

This entire answer stream is full of typical programmer attitudes. It reminds me of Joel's 'Things you should never do' article (rewrite from scratch.) We don't really know anything about the system, other than there's a bug, and some guy posted some code online. There are so many unknowns that it is ridiculous to say "This guy does not know what he is doing."

Share Improve this answer Follow

answered Jan 26, 2009 at 2:20

Shawn
19.8k ● 20 ● 100 ● 153

0

Rather than pile on the Consultant, you could just as easily pile on the person who procured their services. No consultant is perfect, nor is a hiring manager ... but at the end of the day the real direction you should be taking is very clear: **instead of trying to find fault you should expend energy into working collaboratively to find solutions**. No matter how skilled someone is at their roles and responsibilities they will certainly have deficiencies. If you determine there is a pattern of incompentencies then you may choose to transition to another resource going forward, but assigning blame has never solved a single problem in history.

Share Improve this answer Follow

answered Mar 8, 2010 at 20:35

Keith Adler
21.2k ● 29 ● 122 ● 191

-1

On the second point, I would not use exceptions here. Exceptions are reserved for exceptional cases.
However, division of anything by zero certainly does not equal zero (in math, at least), so this would be case specific.

Share Improve this answer Follow

answered Oct 2, 2008 at 20:49

Alf Zimmerman
305 ● 1 ● 2 ● 9