What is the best way to tell if an object is modified?

Asked 16 years, 3 months ago Modified 12 years, 1 month ago

Viewed 3k times

6

I have an object that is mapped to a cookie as a serialized base-64 string. I only want to write out a new cookie if there are changes made to the object stored in the cookie on server-side.

What I want to do is get a hash code when the object is pulled from the cookie/initialized and compare the original hash code to the hash code that exists just before I send the cookie header off to the client to ensure I don't have to re-serialize/send the cookie unless changes were made.

I was going to override the .NET's `Object.GetHashCode()` method, but I wasn't sure that this is the best way to go about checking if an object is modified.

Are there any other ways I can check if an object is modified, or should I override the `GetHashCode()` method.

Update I decided to accept @rmbarnes's answer as it had an interesting solution to the problem, and because I decided to use his advice at the end of his post and not check for modification. I'd still be interested to hear any

other solutions anyone may have to my scenario however.

c# .net

Share

Improve this question

Follow

edited Aug 29, 2008 at 18:54

asked Aug 29, 2008 at 17:30

Dan Herbert

103k ● 51 ● 192 ● 221

4 Answers

Sorted by: Highest score (default)

3

GetHashCode() should always be in sync with Equals(), and Equals() isn't necessarily guaranteed to check for *all* of the fields in your object (there's certain situations where you want that to not be the case).

Furthermore, GetHashCode() isn't guaranteed to return unique values for all possible object states. It's conceivable (though unlikely) that two object states could result in the same HashCode (which does, after all, only have an int's worth of possible states; see the Pigeonhole Principle for more details).

If you can ensure that Equals() checks all of the appropriate fields, then you could possibly clone the

object to record its state and then check it with Equals() against the new state to see if its changed.

BTW: Your mention of serialization gave me an idea. You could serialize the object, record it, and then when you check for object changing, repeat the process and compare the serialized values. That would let you check for state changes without having to make any code changes to your object. However, this isn't a great solution, because:

1. It's probably very inefficient
2. It's prone to serialization changes in the object; you might get false positives on the object state change.

Share Improve this answer

Follow

answered Aug 29, 2008 at 17:39

Craig Walker

51.6k ● 57 ● 155 ● 212

1

At the end of the object's constructor you could serialize the object to a base 64 string just like the cookie stores it, and store this in a member variable.

When you want to check if the cookie needs recreating, re - serialize the object and compare this new base 64 string against the one stored in a member variable. If it has changed, reset the cookie with the new value.

Watch out for the gotcha - don't include the member variable storing the base 64 serialization in the serialization itself. I presume your language uses

something like a sleep() function (is how PHP does it) to serialize itself, so just make sure the member is not included in that function.

This will always work because you are comparing the exact value you'd be saving in the cookie, and wouldn't need to override GetHashCode() which sounds like it could have nasty consequences.

All that said I'd probably just drop the test and always reset the cookie, can't be that much overhead in it when compared to doing the change check, and far less likelyhood of bugs.

answered Aug 29, 2008 at 18:02

Robin Barnes
13.5k ● 15 ● 45 ● 45

0

I personally would say go with the plan you have.. A good hash code is the best way to see if an object is "as-is".. Theres tons of hashing algorithms you can look at, check out the obvious Wikipedia page on hash functions and go from there..

Override GetHashCode and go for it! Just make sure *ALL* the elements of the information make up part of the hash :)

Share Improve this answer

Follow

answered Aug 29, 2008 at 17:40

Rob Cooper

28.9k ● 26 ● 105 ● 142

0

Seems odd to me why you'd want to store the same object both server side and client side - especially if you're comparing them on each trip.

I'd guess that deserializing the cookie and comparing it to the server side object would be equivalent in performance to just serializing the object again.

But, if you wanted to do this, I'd compare the serialized server side object with the cookie's value and update accordingly. Worst case, you did the serialization for naught. Best case, you did a string compare.

The alternative, deserializing and comparing the objects, has a worst case of deserializing, comparing n fields, and

then serializing. Best case is deserializing and comparing n fields.

Share Improve this answer

Follow

answered Sep 1, 2008 at 20:24

Mark Brackett

85.6k ● 17 ● 111 ● 155