ASP .NET MVC NonAction meaning

Asked 8 years, 4 months ago Active 1 year, 2 months ago Viewed 47k times

36

2

Can anybody please tell me why should I use NonAction? I mean say I have a form with several submit values: Update, Delete or Insert. Since all the submit buttons have the same form in common I'm switching the submit value inside the controller and act accordingly.

Like this:

```
public ActionResult asd(string submitButton){
     switch(submitButton){
         case "Insert":
             return Insert();
         // bla bla bla
     }
}

[NonAction]
public ActionResult Insert(){
    // some code inside here
    return View();
}
```

Once again, why should I use NonAction instead of something like this:

```
public void Insert(){
    // some code inside here
}
```

c# asp.net-mvc actionmethod

edited Sep 22 '12 at 19:59
Erik Funkenbusch
81.8k 24 164 262

asked Jun 17 '11 at 12:15
Shaokan
3,146 12 49 75

8 Why make Insert public if you aren't planning on invoking it from anywhere but the asd method? – Gromer Dec 21 '11 at 20:56

1 Most of the answers are correct, but none of those is mentioning the reason behind the existence of the `ChildActionOnly/NonAction` attributes, i.e. their purpose, which was a means to have a "mini MVC" cycle within the main request's MVC cycle. Example would be a view that needs to render/embed a bit more complex partial view. So, rather than calling a `<partial>` directly, one can call a `NonAction/ChildActionOnly` decorated controller action method using `Html.RenderAction()`, to perform that "mini MVC" cycle which returns the html markup of a rendered partial view. – Mladen B. Aug 1 at 8:12

Seems to me that such methods should be in another class and not a controller – Mark Schultheiss Oct 5 at 14:08

10 Answers

50

You can omit the `NonAction` attribute but then the method is still invokable as action method.

From the MSDN site (ref):

> By default, the MVC framework treats all public methods of a controller class as action methods. If your controller class contains a public method and you do not want it to be an action method, you must mark that method with the NonActionAttribute attribute.

edited Sep 22 '12 at 0:47
Taudris
870 1 8 19

answered Jun 17 '11 at 12:19
ReFocus
1,151 1 13 23

1 Ah alright, if you create a public void inside a controller class then when you type the url the page returns blank. Therefore, as you (MSDN actually :D) stated all public methods are treated as actionresults. So you should use NonActionAttribute whenever you don't want your method to be displayed unless you call it :) Thanks! – Shaokan Jun 17 '11 at 12:25

Exactly. In pre-releases of the MVC framework it was the other way around. You had to explicitly define methods as actionmethod using the `ActionMethod` attribute. From the RTM release all methods are treated as Action Method unless you use the `NonAction` attribute. – ReFocus Jun 17 '11 at 12:39

13

It is worth noting that the need to use `[NonAction]` applies only to public methods. Protected and private methods are not treated as actions. Since your `Update` / `Delete` / `Insert` methods are helpers for `asd()`, a private method would be more appropriate for your scenario:

```
public ActionResult asd(string submitButton){
    switch(submitButton){
        case "Insert":
            return Insert();
        // bla bla bla
    }
}

ActionResult Insert(){
    // some code inside here
}
```

edited Oct 13 '12 at 7:19

answered Sep 22 '12 at 0:29

Taudris

870 1 8 19

Reading Haack's Article

5

> Any public method in a controller class is callable via URL.

Sometimes you may need to avoid this. For example, if you implement some interface and you may not want to call that public method you can mark as `NonAction`

```
public interface IEmployee
{
 void Save(Employee e);
 bool Validate(Employee e);
}

public class EmployeeController:Controller, IEmployee
{
  public void Save(Employee e){
  }

  [NonAction]
  public void Validate(Employee e){
  }
}
```

edited Sep 27 '13 at 16:04

answered Sep 27 '13 at 15:42

Murali Murugesan
17.6k 15 58 109

5

If you don't use the [NonAction] attribute then someone can call your action directly instead of having to go through the 'asd' function

answered Jun 17 '11 at 12:19

devprog
190 1 1 10

2

I just used [NonAction] in our web api, to decorate a bunch of Controller Methods (endpoints) because we had a last minute decision that we will postpone the delivery of the specific endpoints.

So it is useful, if you want to avoid exposing an API endpoint, but still want to keep the implementation for later.

So I used this attribute and it saved me a lot of time.

I will just remove it in the next release and this will simply be there!

answered Nov 16 '17 at 13:54

cnom
1,260 2 12 41

1 Just wondering what benefit this has over commenting out the method? – Backwards_Dave Aug 21 '18 at 8:01

2 Commented out code is not a good practice generally, but a more situation is this: Imagine you did all the implementation and some unit tests for this method. This way you dont have to comment all these out! – cnom Nov 30 '18 at 13:06

2

NonAction attribute makes an action non accessible from the navigation bar. For example if you have an action which deletes items in database, you have to add the `NonAction` attribute to make it not accessible by users.

edited May 8 '18 at 13:33

kayess
3,339 8 24 43

answered Aug 4 '14 at 18:48

Bilel Chaouadi
26 1 1 6

0

This is indicate that a controller method is not an action method.Example: [NonAction] public void IndexTest() { // Do some thing } this is very useful attribute when visibility of controller 's method cannot be changed to private.

answered Oct 30 '17 at 6:17

Ajeet Malviya
440 1 4 8

0

Firstly, think of an ActionResult simply as a particular type of construct that is returned by MVC, that happens to provide a particular convenience in terms of the way that ActionResult can be internally handled within the MVC framework. So the fact that something is an ActionResult *doesn't* necessarily mean "*this should be publicly available*". In fact, **any public method in an MVC controller will be treated as an action method, whether or not it returns an ActionResult**.

So, simply having a return type that *isn't* an ActionResult will *not* necessarily prevent that method from being exposed as a publicly available action that can be called via a URL.

There may be many reasons you don't want to expose a method as an action that can be invoked via a url, and in the case that you want to 'protect' against this, that's when you use the [NonAction'] attribute.

answered Jun 2 '17 at 5:32

Chris Halcrow
13.4k 6 87 104

0

It's an attribute which is used on the methods which are defined by public access modifiers. Actually MVC Framework treats all public methods as URLs but in case you don't want this then you have to decorate a method with the non action attribute. The same thing may be achieved by making the method private.

An example of NonAction Attribute is given below. http://yogeshdotnet.com/non-action-attribute-in-mvc/

edited Aug 21 '18 at 8:03

Backwards_Dave
3,089 11 37 77

answered Feb 14 '17 at 15:22

Yogesh Sharma
21 1

If you did not want to invoke some action methods then you have to mark it with a attribute `[NonAction]` or by making it private

-1

```
public ActionResult Index(){
    return View();
}

[NonAction]
public ActionResult Countries(List<string>countries){
    return View(countries);
}
```

you can copy the code and paste it and see the result.thanks

edited Oct 30 '17 at 6:00

Arash GM

8,918 4 49 71

answered Jun 21 '17 at 18:45

Hamid Nawaz

1 3