How to skip action execution from an ActionFilter?

37

5

Is it possible to skip the whole action method execution and return a specific `ActionResult` when a certain condition is met in `OnActionExecuting` ?

asp.net-mvc action-filter

edited Jan 13 '16 at 14:43
BartoszKP
27.4k 10 74 110

asked Mar 23 '12 at 9:59
user49126
466 5 21 43

4 Answers

22

You can use filterContext.Result for this. It should look like this:

```
public override void OnActionExecuting(ActionExecutingContext filterContext)
{
    //Check your condition here
    if (true)
    {
        //Create your result
        filterContext.Result = new EmptyResult();
    }
    else
        base.OnActionExecuting(filterContext);
}
```

answered Mar 23 '12 at 11:21
tpeczek
21.5k 3 63 71

Why do you skip base.OnActionExecuting when your condition is true? In my case I need that to run before I can set the Result. – xr280xr Jun 16 '15 at

34

See my download sample and MSDN article Filtering in ASP.NET MVC.

You can cancel filter execution in the `OnActionExecuting` and `OnResultExecuting` methods by setting the `Result` property to a non-null value.

Any pending `OnActionExecuted` and `OnActionExecuting` filters will not be invoked and the invoker will not call the `OnActionExecuted` method for the cancelled filter or for pending filters.

The `OnActionExecuted` filter for previously run filters will run. All of the `OnResultExecutingand` `OnResultExecuted` filters will run.

The following code from the sample shows how to return a specific `ActionResult` when a certain condition is met in `OnActionExecuting` :

```
if (filterContext.RouteData.Values.ContainsValue("Cancel"))
{
    filterContext.Result = new RedirectResult("~/Home/Index");
    Trace.WriteLine(" Redirecting from Simple filter to /Home/Index");
}
```

edited May 3 '14 at 3:14
Alex
1,108 2 14 35

answered Mar 23 '12 at 16:29
RickAndMSFT
11.5k 5 40 57

2 It should be noted that if you have more than one filter and the order of the filters matters you should specify the "Order" parameter when registering your filter so you can control the execution order as Rick has outlined in his details about how the Result filterContext.Result property behaves. – Nick Bork Mar 23 '12 at 16:36

Good point Nick - but see my section Filter Order - the order property only applies to filters in the same class. Auth always run first, exception, last. – RickAndMSFT Mar 27 '12 at 16:38

2

You can use the following code here.

```
public override void OnActionExecuting(ActionExecutingContext filterContext)
 {
```

```
            ...
            filterContext.Result = new RedirectToAction(string action, string controller)
            return;
        }
        ...
    }
```

RedirectToAction will redirect you the specific action based on the condition.

answered May 13 '14 at 6:41
Mehul Vaghela
313 2 18

RedirectToAction is a method (at least in MVC5) so you can't new() it. – rumblefx0 Sep 1 '15 at 14:08

1 A worthless answer added long after the original (and correct) answers, and it's wrong as `filterContext.Result` must be a sub class of `ActionResult` . How this ever got up-voted is a mystery. – Steve Pettifer Mar 22 '18 at 14:47

0

If anyone is extending `ActionFilterAttribute` in MVC 5 API, then you must be getting `HttpActionContext` instead of `ActionExecutingContext` as the type of parameter. In that case, simply set `httpActionContext.Response` to new `HttpResponseMessage` and you are good to go.

I was making a validation filter and here is how it looks like:

```
    /// <summary>
    /// Occurs before the action method is invoked.
    /// This will validate the request
    /// </summary>
    /// <param name="actionContext">The http action context.</param>
    public override void OnActionExecuting(HttpActionContext actionContext)
    {
        ApiController ctrl = (actionContext.ControllerContext.Controller as
 ApiController);
        if (!ctrl.ModelState.IsValid)
        {
            var s = ctrl.ModelState.Select(t => new { Field = t.Key, Errors =
 t.Value.Errors.Select(e => e.ErrorMessage) });
            actionContext.Response = new System.Net.Http.HttpResponseMessage()
```

```
                StatusCode = (System.Net.HttpStatusCode)422
            };
        }
    }
```

answered Nov 23 '18 at 13:56

Anup Sharma
1,179 10 21