Action Filtering in ASP.NET MVC Applications

.NET Framework 4 5 out of 8 rated this helpful

In ASP.NET MVC, controllers define action methods that usually have a one-to-one relationship with possible user interactions, such as clicking a link or submitting a form. For example, when the user clicks a link, a request is routed to the designated controller, and the corresponding action method is called.

Sometimes you want to perform logic either before an action method is called or after an action method runs. To support this, ASP.NET MVC provides action filters. Action filters are custom attributes that provide a declarative means to add pre-action and post-action behavior to controller action methods.

MVC Action Filter Types

ASP.NET MVC provides the following types of action filters:

- Authorization filter, which makes security decisions about whether to execute an action method, such as performing authentication or validating properties of the request. The AuthorizeAttribute class is one example of an authorization filter.
- Action filter, which wraps the action method execution. This filter can perform additional processing, such as providing extra data to the action method, inspecting the return value, or canceling execution of the action method.
- Result filter, which wraps execution of the ActionResult object. This filter can perform additional processing of the result, such as modifying the HTTP response. The OutputCacheAttribute class is one example of a result filter.
- Exception filter, which executes if there is an unhandled exception thrown somewhere in action method, starting with the authorization filters and ending with the execution of the result. Exception filters can be used for tasks such as logging or displaying an error page. The HandleErrorAttribute class is one example of an exception filter.

How To Apply an Action Filter

Typically, an action filter is an attribute that implements the abstract FilterAttribute class. Some action filters, such as AuthorizeAttribute and HandleErrorAttribute, implement the FilterAttribute class directly. These action filters are always called before the action method runs.

Other action filters, such as OutputCacheAttribute, implement the abstract ActionFilterAttribute class, which enables the action filter to run either before or after the action method runs.

You can use the action filter attribute to mark any action method or controller. If the attribute marks a controller, the action filter applies to all action methods in that controller.

The following example shows the default implementation of the **HomeController** class. In the example, the **HandleError** attribute is used to mark the controller. Therefore, the filter applies to both action methods in the controller.

C#

```
[HandleError]
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewData["Message"] = "Welcome to ASP.NET MVC!";

        return View();
    }

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

Related Topics

Title	Description
AuthorizeAttribute	Describes how to use the **Authorize** attribute to control access to an action method.
OutputCacheAttribute	Describes how to use the **OutputCache** attribute to provide output caching for an action method.
HandleErrorAttribute	Describes how to use the **HandleError** attribute to handle exceptions that are thrown by an action method.
Creating Custom Action Filters	Describes how to implement custom action filters.
How to: Create a Custom Action Filter	Explains how to add a custom action filter to an MVC application.

See Also

Other Resources
ASP.NET MVC 2

ASP.NET MVC Overview

Community Additions

Filters in MVC

summarized the filters in one single POC.

http://pratapreddypilaka.blogspot.in/2012/03/custom-filters-in-mvc-authorization.html

Authorize Filter

Action Filter

Result Filter

Exception Filter

But dint understand which case we can use ResultExecuted() event? Can some one make me understand it.

PratapReddy
4/17/2012

...providing extra data to the action method

In the first section, "MVC Action Filter Types," the second bulleted item describing an Action filter says: "This filter can perform additional processing, such as providing extra data to the action method..."

I was curious about how exactly this extra data can be passed? I was hoping for some obivious and hopefully strongly-typed way to do this, but it seems that one is expected to do this through the ViewData dictionary. Please follow-up to this if there is a better or alternative way.

P.S., the MVC 3 on this same subject is here: http://msdn.microsoft.com/en-us/library/gg416513%28VS.98%29.aspx

Funka!
2/17/2011