MVC API Return Result - ExecuteResultAsync

(ActionContext) method

What is the usage of ExecuteResultAsync(ActionContext) method?

1. IActionResult is **the result (object) returned by your controller action method**. ASP.NET Core executes this result by invoking the IActionResult::ExecuteResultAsync(ActionContext) method.
2. Action method might return different kinds of IActionResult, such as **JsonResult** (which sends a json to client when executed), **BadRequestResult**(which sends 400 response when executed) and so on.
3. You can also create your own implementation of IActionResult. Just be aware that the ExecuteResultAsync(ActionContext) method **should write bytes to the HTTP Response** . For example, I created a [custom IActionResult to return CSV file](https://stackoverflow.com/a/52492851/10091607) for your reference. This is a simple implementation that doesn't contains too much complicated logic.

Some action results are rather complicated that they introduce a new IActionResultExecutor<TResult> interface to deal with these process, for example, the ObjectResult employ an IActionResultExecutor<ObjectResult> to do that. When creating your own implementation, whether to use an IActionResultExecutor<ObjectResult> is up to you.

How the MVC will used the method?

1. WebApp developers don't need to invoke thisIActionResult::ExecuteResultAsync(ActionContext) method manually. This is a method that will be invoked by the MVC/RazorPage subsystem.
2. If you're interested, the whole process is:
   * a coming request received
   * match current request against the pre-defined routes (route table or graph). If matched :
     + we know the controller name, action name, and other route data.
     + Since we've known the controller name and action name, ASP.NET Core generates an instance of ActionDescriptor that describes the target C# action method, such as the parameters.
   * Since ASP.NET Core has known the Controller/Action and routes data, it creates an instance of **IActionInvoker** to invoke that action method pipline (including the filters, for more details, see [official docs](https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-3.1)):
     + Action Method returns an instance of IActionResult
     + Before invoking the IActionResult::ExecuteResultAsync(ActionContext), invoke Result Filters OnResultExecuting() method.
     + invoke IActionResult::ExecuteResultAsync(ActionContext)
     + After that, invoke the Result Filters OnResultExecuted() method.

*where is the ActionContext parameter from? from the Http request?*

Firstly, HttpContext is built by the underlying server. It contains a Request property that mimics the HTTP Request.

Next, we'll get another two objects after selecting the action:

* RouteData: the route data, e.g. current area name, current page name, e.t.c.
* ActionDescriptor: a description about the current action that are matched with current route.

With the above three objects, ASP.NET Core creates the ActionContext by simply new it. For example, the IRouter-based Routing system creates an actionContext as below:( [see source code](https://github.com/aspnet/AspNetCore/blob/c565386a3ed135560bc2e9017aa54a950b4e35dd/src/Mvc/Mvc.Core/src/Routing/MvcRouteHandler.cs#L69-L78))

// create action context

var actionContext = new ActionContext(context.HttpContext, routeData, actionDescriptor);

// create action invoker

var invoker = \_actionInvokerFactory.CreateInvoker(actionContext);

if (invoker == null){ throw ...;}

// invoke the pipeline

return invoker.InvokeAsync();