Session 11

ASP.NET Core MVC Filters

Session Overview

- Explain different filters
- Describe implementations of filters
- List various examples for filters

Introduction to Filters

Filters are performed prior to, post, or during execution of an action method.

Filters facilitate execution of tasks that are repeated in multiple controllers or within their action methods.

Filters can be implemented using built-in attributes or customized attributes.

Filters can be defined by implementing the filter interface or, it can be done by inheriting or overriding the methods of the available filter attribute class.

When to Use Filters?

- Authentication
- Authorization (User or Role-based)
- Login details of users
- Handling of errors
- Data caching
- Data In/Out
- Logging

Different Types of Filters (1-2)

- Authorization filters
- Resource Filter
- Action filters
- Exception Filter
- Result filters

Different Types of Filters (2-2)

Filter Type	Interfaces	Description
Authorization	IAuthorizationFilter	Controller or action limits access to the controller or action to any authenticated user
Resource	IResourceFilter	Resource filters are useful to implement caching or otherwise short-circuit the filter pipeline for performance reasons
Action	IActionFilter	Executed prior to or subsequent to intended actions
Result	IResultFilter	Executed subsequent to results of actions
Exception	IExceptionFilter	Executed only if any other filter, action method, or result of actions shows an exception

Types of ASP.NET Core MVC Filters

Authorization Filters

- Components of Authorization filter include `AuthorizeAttribute` and `AllowAnonymousAttribute`
- They are part of the `Microsoft.ASPNetCore.Authorization` namespace

Resource Filters

Resource filters are commonly used to implement caching.

Resource filters take care of the request post authorization.

They can run code before and after the rest of the filter is executed, before the model binding happens.

Resource filters are useful to short-circuit most of the work a request is doing.

Action Filters

Action filters are executed prior to an action run or subsequent to an action run.

`IActionFilter` interface provides for an action filter with a method called `OnActionExecuting`, which gets executed prior to an action.

Method `OnActionExecuted` gets executed subsequent to an action run.

Result Filters

Result filters are executed prior to or subsequent to results of actions. Different types of Result filters are:

- ViewResult
- PartialView Result
- RedirectTo RouteResult
- RedirectResult
- ContentResult
- JsonResult
- FileResult
- EmptyResult

Exception Filters

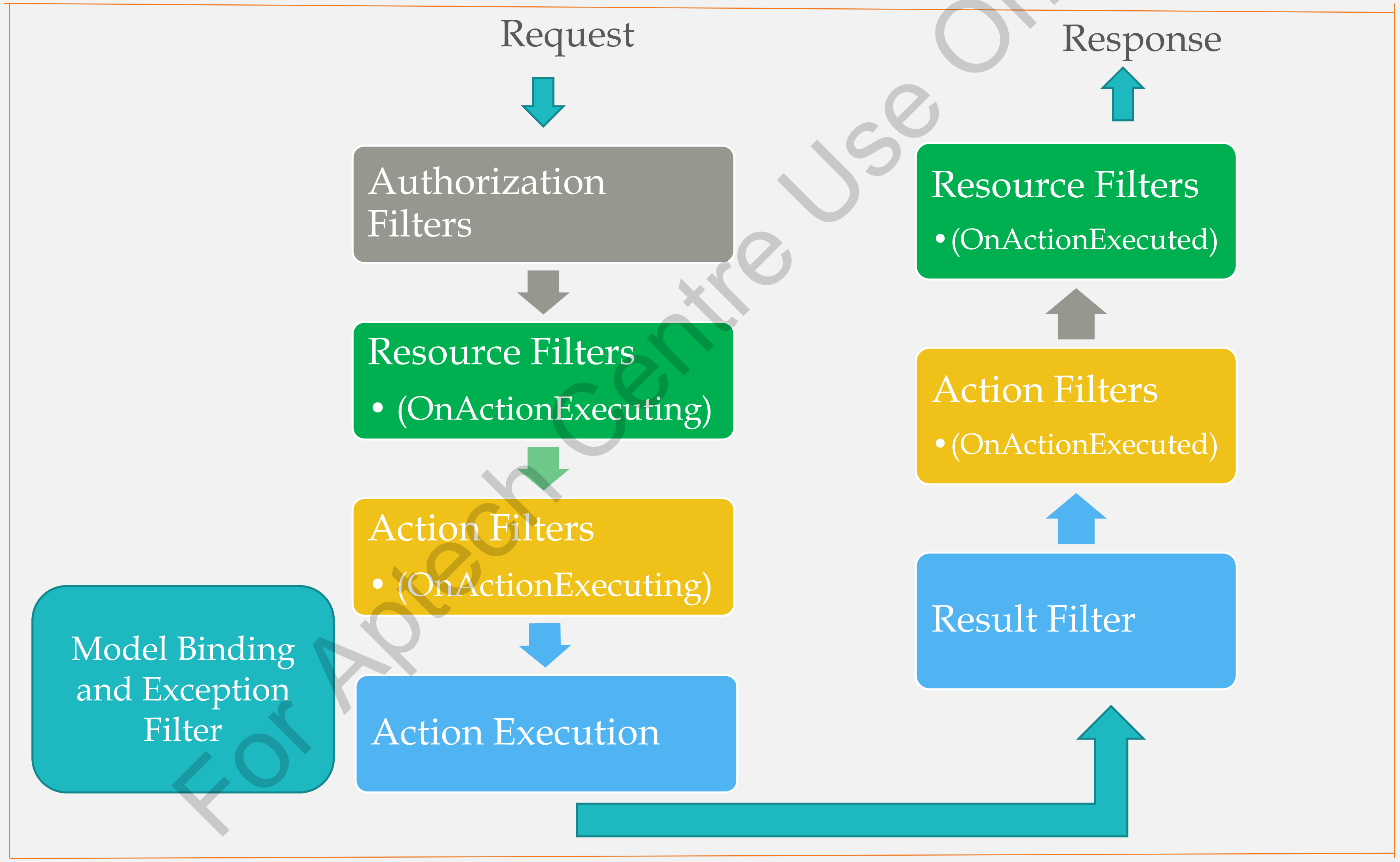
Exceptions are generated during execution of actions or filters. This results in calling of Exception filters.

IExceptionFilter aids in creation of an Exception filter.

The Exception filter allows for the OnException method.

The OnException method comes into play under the condition of an exception.

Order of Filter Execution



Configuring Filters

A custom filter can be configured into an application using three different levels as follows:

Global level

- ➢ A developer can restrict access for every Web API controller by adding the `AuthorizeAttribute` filter to the global filter list. In the `Startup.cs` class, add the code snippet for setting global authorization for all actions.

Action Filter for Validation

A controller action or a complete controller can be associated with an attribute. It can be an Action filter that modifies the way the action is executed.

Action Filter for Handling Error

By overriding `OnActionExecuted` method and `ActionExecutedContext` developers can handle an exception in a managed way. When the controller generates an error, the application is forwarded to a custom message.

More on Resource Filter

- Resource filters take care of the request post authorization.
- Override OnResourceExecuting method of IResourceFilter filter attribute.
- In case developers want to short-circuit any action method, they can implement this attribute.

Summary

- ASP.NET Core MVC provides support to execute custom code in the form of filters.
- These filters can be general or customized. The intended action can be allowed to be performed prior to or subsequent to controller actions.
- There are five types of filters - Authorization, Resource, Action, Exception, and Result. ASP.NET framework has special interfaces to support the function of filters.
- The specific order in executing them are Authentication, Authorization, Action, and Result filters.
- Configuration of Action filters is possible in three levels - Global level, Controller level, and Action level. The design and implementation of custom filters for various scenarios is also possible.