ASP.NET MVC 5 APPLICATION LIFECYCLE – HIGH-LEVEL VIEW

HttpApplication
Processing Pipeline

HTTP Request

This document shows the lifecycle of every ASP.NET MVC application, beginning from receiving the HTTP request from the client to sending the HTTP response back to the client. It is designed both as an education tool for those who are new to ASP.NET MVC and also as a reference for those who need to drill into specific aspects of the application.

HttpApplication
Processing Pipeline

Controller Creation

Routing

MvcHandler

HTTP Response

Authentication and Authorization

Model Binding

Action Method Invocation

Result Execution (View, etc.)

Action Invocation (with Filters)

ASP.NET MVC 5 APPLICATION LIFECYCLE – DETAIL VIEW

The diagram shows relevant HttpApplication stages to help you understand where MVC integrates into the ASP.NET application lifecycle. In addition, for the overridable methods on the Controller object that are part the MVC lifecycle, the diagram identifies when these methods are invoked in the processing pipeline and why you might want to override them. You may or may not have the need to override any one method, but It is important for you to understand their role in the application life cycle so that you can write code at the appropriate life cycle stage for the effect you intend.

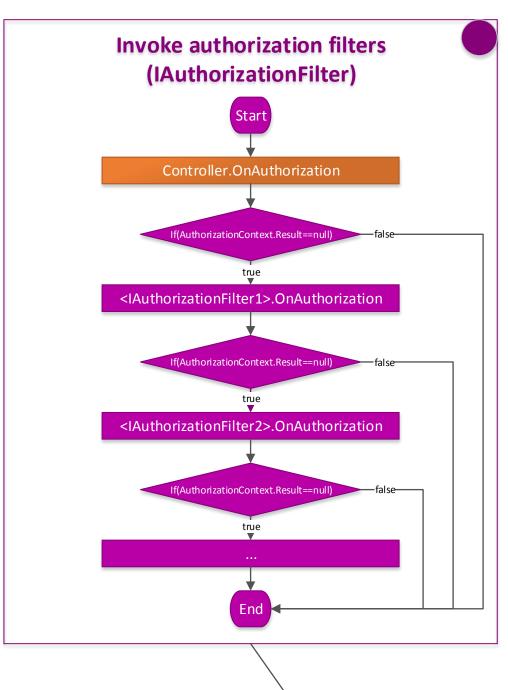## HttpApplication Processing Pipeline

HTTP Request

Application_Start (in Global.asax.cs)
- Tasks to perform:
  - Common tasks such as register routes, bundles, and filters
  - Advance tasks such as set custom controller factory

HttpApplication.Init
- Override to subscribe to HttpApplication events

IHttpModule.Init

HttpApplication.PostResolveRequestCache

UrlRoutingModule.PostResolveRequestCache
- Match request to defined route
- Retrieve MvcHandler as the HttpHandler for the request (stored at HttpApplication.Context.Handler)
- Store the encapsulated route data (at HttpApplication.Context.Request.RequestContext)

HttpApplication.BeginProcessRequest

HttpApplication.EndProcessRequest

MvcHandler executes the controller action

HTTP Response

Color Legend
- HttpApplication
- Routing
- Controller
- Authentication Filters
- Authorization Filters
- Authentication Challenges
- Action Execution
- Result Execution

HttpApplication Processing Pipeline

IControllerFactory.CreateController — Called by the MvcHandler to create the named controller. By default, DefaultControllerFactory is used. To register a custom IControllerFactory, use ControllerBuilder.Current.SetDefaultControllerFactory in Application_Start (in Global.asax.cs)

<Named>Controller.ctor

Controller.BeginExecute — Override to perform processing before anything is done on the controller

Controller.Initialize — Override to perform custom initialization before ControllerBase.Initialize sets Controller.ControllerContext with the encapsulated route data.

Controller.BeginExecuteCore

Controller.CreateTempDataProvider — Override to use a custom TempData provider. SessionStateTempDataProvider is used by default.

Controller.CreateActionInvoker — Override to use a custom action invoker. AsyncControllerActionInvoker is used by default.

Invoke authentication filters (IAuthenticationFilter)

If(IAuthenticationChallengeContext.Result==null) — false / true

Invoke authorization filters (IAuthorizationFilter)

If(AuthorizationContext.Result==null) — false / true

Perform model binding on parameters

Invoke authentication challenges (IAuthenticationFilter)

Invoke action with action filters (IActionFilter)

Invoke authentication challenges (IAuthenticationFilter)

Controller.EndExecute

Controller.EndExecuteCore

Execute result with result filters (IResultFilter)

Execute result

Controller.Dispose

HttpApplication.BeginProcessRequest

HttpApplication.EndProcessRequest

Invoke authentication filters (IAuthenticationFilter)

Start

Controller.OnAuthentication

If(AuthenticationContext.Result==null) — false / true

<IAuthenticationFilter1>.OnAuthentication

If(AuthenticationContext.Result==null) — false / true

<IAuthenticationFilter2>.OnAuthentication

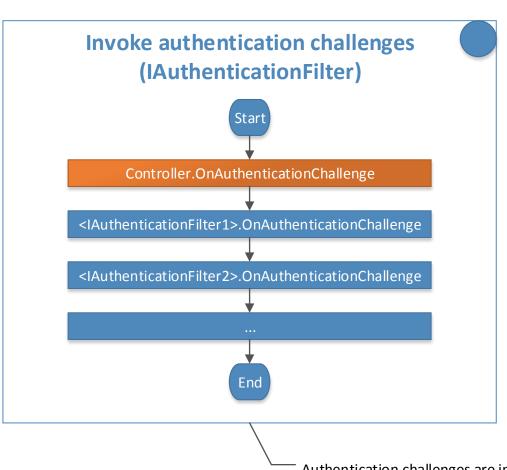If(AuthenticationContext.Result==null) — false / true

...

End

Use IAuthenticationFilter to authenticate a user action toward the intended resources. Authentication filters execute in order until one of the filters returns a non-null result.

Invoke authorization filters (IAuthorizationFilter)

Start

Controller.OnAuthorization

If(AuthorizationContext.Result==null) — false / true

<IAuthorizationFilter1>.OnAuthorization

If(AuthorizationContext.Result==null) — false / true

<IAuthorizationFilter2>.OnAuthorization

If(AuthorizationContext.Result==null) — false / true

...

End

Use IAuthorizationFilters to authorize a user action toward the intended resources. Authorization filters execute in order until one of the filters returns a non-null result.

Invoke authentication challenges (IAuthenticationFilter)

Start

Controller.OnAuthenticationChallenge

<IAuthenticationFilter1>.OnAuthenticationChallenge

<IAuthenticationFilter2>.OnAuthenticationChallenge

...

End

Authentication challenges are invoked whenever authentication or authorization filters return a result to indicate failure. They are also invoked after the action method executes in case the authentication scheme requires it, such as for server authentication to the client. Use OnAuthenticationChallenge to create the ActionResult you intend to send to the client when authentication challenges occur. When authentication challenges are invoked, all registered IAuthenticationFilter contribute to the result.

Execute action method with action filters (IActionFilter)

HttpApplication.BeginProcessRequest

Controller.OnActionExecuting
<ActionFilter1>.OnActionExecuting
<ActionFilter2>.OnActionExecuting
...

Controller implements IActionFilter (by inheriting ActionFilterAttribute). OnActionExecuting methods are executed in order before the action method itself.

<Named>Controller.<ActionAsync>

Asynchronous action methods (modified with the **async** keyword and returns Task<T>) are executed in HttpApplication.**BeginProcessRequest**

Controller.EndExecute

Controller.EndExecuteCore

Called in HttpApplication.EndProcessRequest to end the controller execution. For asynchronous action methods, this portion of the pipeline may be invoked in a different worker thread.

<Named>Controller.<Action>

Synchronous action methods (not modified with the **async** keyword) are executed in HttpApplication.**EndProcessRequest**

...
<ActionFilter2>.OnActionExecuted
<ActionFilter1>.OnActionExecuted
Controller.OnActionExecuted

OnActionExecuted methods are executed in reverse order after the action method itself.

HttpApplication.EndProcessRequest

Execute results with result filters (IResultFilter)

Controller.OnResultExecuting
<ResultFilter1>.OnResultExecuting
<ResultFilter2>.OnResultExecuting
...

Controller implements IResultFilter (by inheriting ActionFilterAttribute). OnResultExecuting methods are executed in order before the result is executed.

Invoke action results (ActionResult.ExecuteResult)

View, PartialView / Other ActionResult types

Find view from view engines (RazorView, etc.) Render view

Write to HTTP response

After authentication challenges are executed as a result of failed authentication or failed authorization, the result is executed immediately without any result filters.

...
<ResultFilter2>.OnResultExecuted
<ResultFilter1>.OnResultExecuted
Controller.OnResultExecuted

OnResultExecuted methods are executed in reverse order after the result is executed