API Key Authentication

Methods to Implement APIKey Auth :-

1. Custom Middleware
2. Custom Attribute

1. Custom Middleware :-

a. Store key in appsettings.json

```
"Apikey" : " GUID "
```

b. Create Service :-

→ IApiKeyValidation.cs

```
public interface IApiKeyValidation {
    bool IsValidApikey(string userApiKey);
}
```

→ ApiKeyValidation.cs

```
public class ApikeyValidation : IApiKeyValidation
{
    private readonly IConfiguration _configuration;
    public ApikeyValidation(IConfiguration config)
    {
        _configuration = config;
    }
    public bool IsValidApikey(string userApiKey)
    {
        if(String.IsNullOrWhiteSpace(userApikey))
        {
            return false;
```

Date / / Page 5

```
string? apiKey = _configuration.GetValue<string>(Constants.ApiConstants.ApiKeyName);

if (apiKey == null || apiKey != userApiKey)
{
    return false;
}
return true;
}
```

(c) Create a constants folder

```
public class ApiConstants
{
    public const string ApiKeyHeaderName = "X-Api-Key";
    public const string ApiKeyName = "ApiKey";
}
```

(d) Create Custom Middleware

```
public class ApiKeyMiddleware
{
    // code on github
}
```

(e) Register Services in Program.cs

Note As we are injecting IApiKeyValidation service in the middleware's Constructor

We should register our service as AddSingleton.

Because middleware components run outside of the scope of request pipeline & their life time / Page / Date is Independent of the request scope.

builder.Services.AddTransient<IApiKeyValidation, ApiKeyValidation>();

app.UseMiddleware<ApiKeyMiddleware>();

⊛ Filters in .NET

→ Filters allow code to run before or after specific stages in the request processing pipeline.

→ Custom filters can be created to handle cross cutting concern. ex-
- Authorization
- Logging
- Caching
- Error handling exceptions.

→ Request processing stages are

→ Middleware → Controller → Action → View

→ Run filters anywhere between

⇒ Filter Types :-

① Authorization Filter :-
Runs first
determine whether the user is authorized for the request. Short ckt the pipeline if request is not authorized

Date __/__/__ Page _____

② Resource filter :-
Run after authorization

OnResourceExecuting(): runs code before the rest of filter pipline.

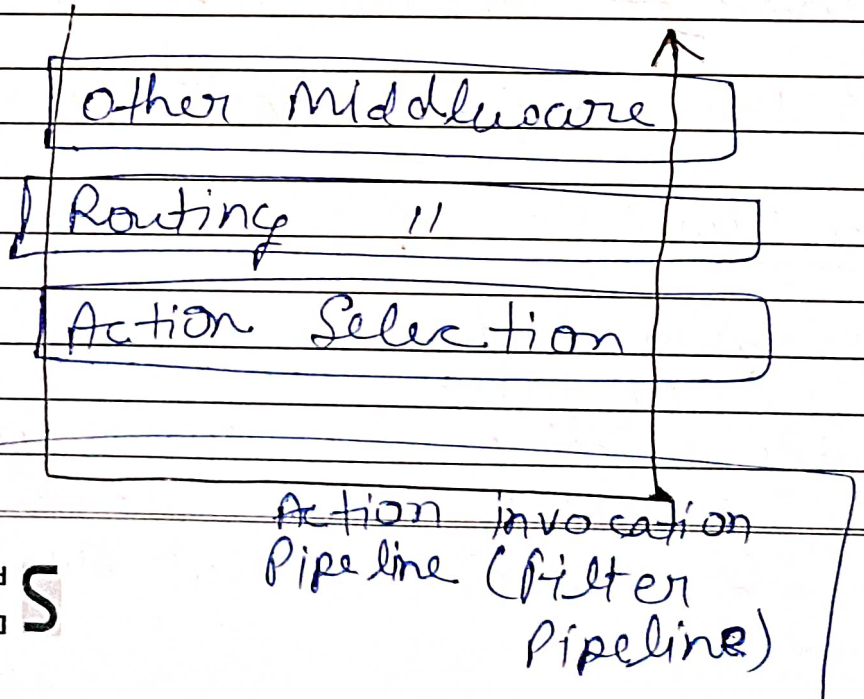
OnResourceExecuted(): after the rest of pipline has completed.

③ Action Filter :-
run immediately before & after an action method is called. Can change the arguments passed into an action. can change the result

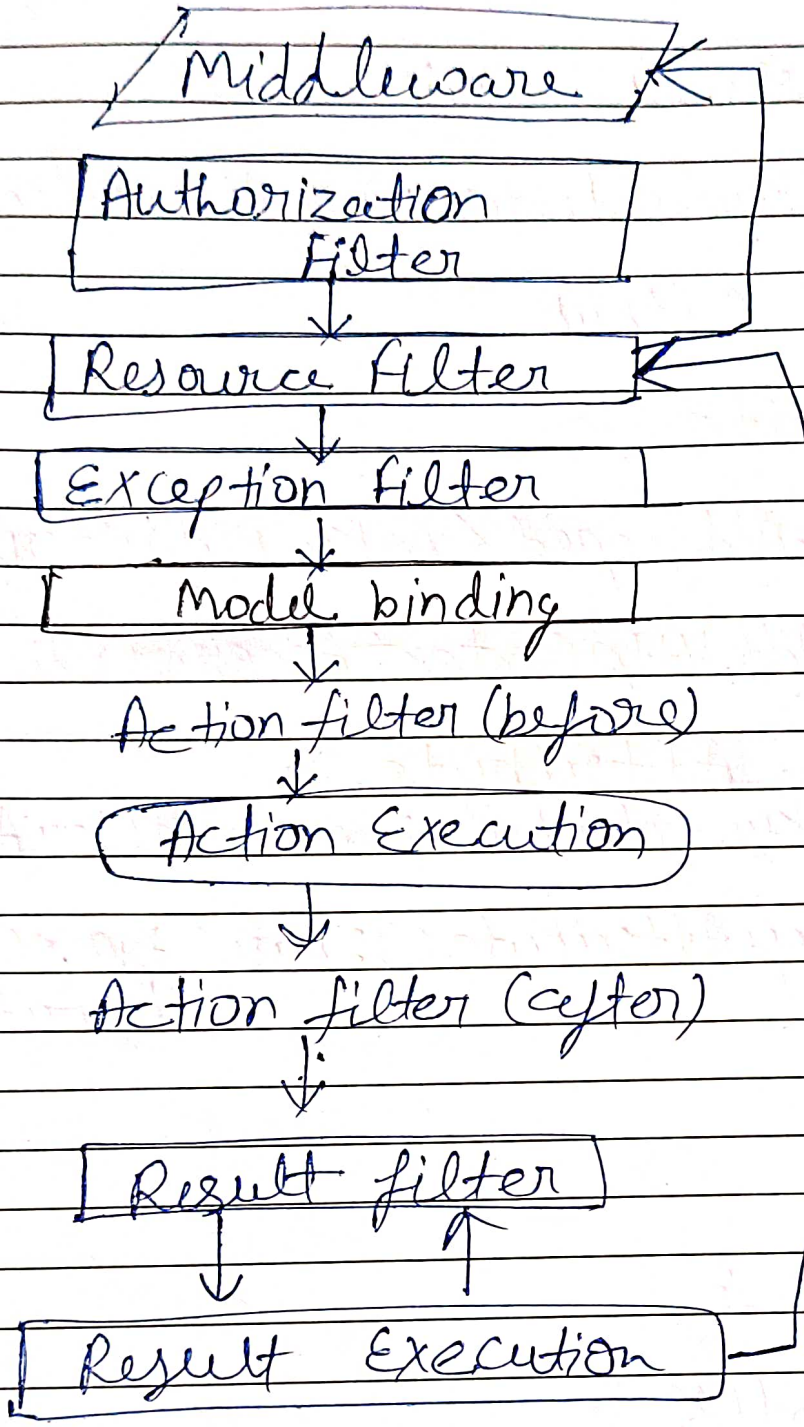
④ Endpoint Filter :- run immediately before and after an action methods is called

⑤ Exception filter :- global exception handling

⑥ Result Filter :- apply global policies to unhandled exceptions that occur before the response body has been written to.



filter types interaction



② Using Custom Attribute :-

ⓐ Create a custom Filter :-

```
public class ApiKeyAuthFilter : IAuthorizationFilter
{
        //code on Github
}
```

ⓑ Register the filter :-

```
builder.Services.AddScoped<ApiKeyAuthFilter>();
builder.Services.AddHttpContextAccessor();
```

ⓒ Create custom Attribute :-

```
public class ApiKey.Attribute : ServiceFilterAttribute
{
    public ApiKeyAttribute() : base(typeof(
                                ApiKeyAuthFilter))
    {
    }
}
```

ⓓ Use Attribute :-

```
[ApiKey]
ActionMethod(){ }
```