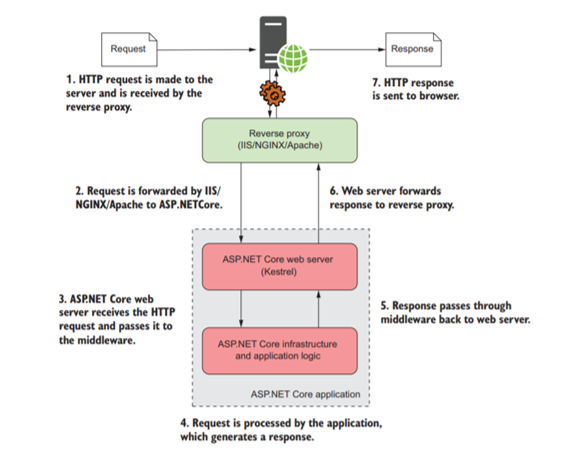
Ch1 – Introduction

ASP.NET Core is a new web framework built with modern software architecture practices and modularization as its focus.

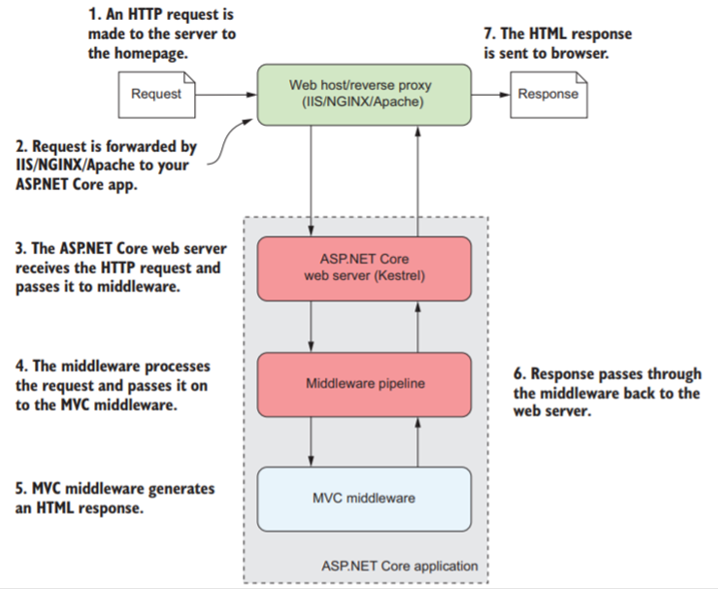
It’s best used for new, “green-field” projects with few external dependencies.

Fetching a web page involves sending an HTTP request and receiving an HTTP response.

ASP.NET Core allows dynamically building responses to a given request.

ASP.NET Core can run on both .NET Framework and .NET Core.

Ch2->5 – Middleware & Routing



Middleware: Pipeline of request handlers. Requests are run from

the top down till one of them is accepted then the rest won’t be

called. Example of an MVC **Startup.cs**

**public void Configure(IApplicationBuilder app, IHostingEnvironment env){**

**if (env.IsDevelopment()){**

**app.UseDeveloperExceptionPage();**

**}//Shows developer the error message in case of errors**

**else{**

**app.UseExceptionHandler("/Home/Error");**

**app.UseHsts();**

**}//shows users a developer-made error page**

**app.UseHttpsRedirection();//redirect HTTP to HTTPS**

**app.UseStaticFiles();//handles any reuqested images, JS, and CSS**

**app.UseCookiePolicy();**

**app.UseMvc(routes =>{**

**routes.MapRoute(**

**name: "default",**

**template: "{controller=Home}/{action=Index}/{id}");**

**routes.MapRoute(**

**name: "second",**

**template: "{controller=contname}/{action=funcName}/{a}/{b}/{c}/{d}");**

**routes.MapRoute(**

**name: "thirdA",**

**template: "{controller=cont2}/{attrib1}/{a2}");**

**});//Add a new maproute for every page & their GET requests**

**}**

Example of actions inside a controller:

**namespace Quiz.Controllers{**

**public class StudentController : Controller{**

**public string Index(string name, int age=1, string hobbies="na"){**

**//Called by default when no action is specified**

**return name+" is "+age+" with hobbies: "+hobbies;**

**}**

**public string noid(string name = "empty", int age=1, string hobbies="none"){**

**//Returns text on a blank page**

**return name+" is " + age +" with hobbies: " + hobbies;**

**}**

**public IActionResult withid(int id = 0, string name = "default", int age = 1, string hobbies = "none"){**

**//Fills items into the viewbag to be used at the vew**

**ViewBag.name = name;**

**ViewBag.id = id;**

**ViewBag.age = age;**

**ViewBag.hobbies = hobbies;**

**//Redirects to the named view**

**return View(“viewname”);**

**//NoteL if View() is empty, the view with the same name as**

**//the action is used**

**}**

**}**

Similar to middleware, routes will be matched from the top down

till one matches. Example with default:

**Cathces all**

**remaining attribs**

**routes.MapRoute(**

**name: "thirdA",**

**template: "{controller=cont2}/{attrib1}/{attrib2?}/{\*others}");**

**defaults: new {controller = “cname” , action = “actname”});**

**Means this attribute is optional**

Instead of returning a view, we can return one of:

**ViewResult(viewname) —Generates an HTML view.**

**RedirectResult(URL) —Sends a 302 HTTP redirect response to automatically**

**send a user to a specified URL.**

**RedirectToAction(actionname) —Calls on the speccified action**

**NotFoundResult() —Sends a raw 404 HTTP status code as the response.**

We can specify if this action is intended for post or get requests only:

**[HttpPost]**

**public string Index(string name, int age=1, string hobbies="na"){…}**

**[HttpGet]**

**public string Index(string name, int age=1, string hobbies="na"){…}**

Ch6 – Binding Models

MVC uses C# classes (models) to send & receive data between views. When a model is set to be used on a page, upon sending a new request, the controller will attempt to fill this model from **Form values** with matching names, **Route values** from the URL, and **Querry Strings** (in that order). This check will happen for every attribute within the set class individually.

Example model:

namespace Exercise1.Models{

public class Student{

[DisplayName("First Name")][StringLength(maximumLength:20,MinimumLength =3)]

public string Fname { get; set; }

[DisplayName("Last Name")][StringLength(maximumLength: 20, MinimumLength = 3)]

public string Lname { get; set; }

[DataType("int")]

public int Age { get; set; }

[Required][Range(0,100)]

public int Grade { get; set; }

}

}

Note: if after all checks an attribute has no value, a default value will be used.

Some useful data bindings:

[CreditCard] [EmailAddress] [Phone] [RegularExpression(“Regex”)]

[Url] [Required(ErrorMessage=“Field is Required”)] [Compare]