Chapter 3

Routing
Routine is the process of matching incoming HTTP requests by checking the HTTP method url and invoking corresponding endpoints. Routing is accomplished with two individual methods that are UseRouting() and UseEndPoints(). In this routing first needs to be executed using the UseRouting() method and then UseEndPoints() method.

UseRouting()
This method is responsible for mapping incoming requests with corresponding Endpoints.

Ex

```
app.UseRouting();
```

UseEndPoints()
This method is responsible for executing the endpoint method selected by the UseRouting() method. This method accepts lambda expressions where you need to further define endpoints related to the method.

Ex

```
app.UseEndpoints(endpoints =>
{
    //add endpoints
});
```

Map(), MapGet() and MapPost()

Map()
Request is used to map endpoints with incoming requests any incoming request mapped with endpoints.map get executed and respective endpoint is executed.

MapGet()
If you want to restrict incoming requests based on request type for get only then you can use MapGet().

MapPost()
If you want to restrict incoming requests based on request type for post only then you can use MapPost().

Ex

```
app.UseEndpoints(async endpoints =>
{
    //add end points

    endpoints.Map("", async (context) =>
    {
        await context.Response.WriteAsync("Select map 1 or map 2");
    });

    endpoints.MapGet("map1", async (context) =>
    {
        await context.Response.WriteAsync("In map 1");
    });

    endpoints.MapPost("map2", async (context) =>
    {
        await context.Response.WriteAsync("In map 2");
    });
});
```

GetEndPoint()
Get end point method is used to get an object of mapped endpoint. We can use this method if we want to implement some functionality before getting some endpoint is called. This method we need to write after use.Routing() because then only it will be able to get endpoint objects.

Route Parameter
So for example we have one URL where we are getting user details

/employee/profile/john

Above URL is responsible for providing employee details for john so employee can be different like john, mac, tom etc so and rest url is same it is literal text text that is vary is called as route parameter

/employee/profile/{employeename}

Route Values
Routevalues are used to get access to pass parameters through route parameters progamitically.

```
app.UseEndpoints(endpoints =>
{
    //add end points

    endpoints.Map("file/{filename}.{extension}", async context =>
    {
        await context.Response.WriteAsync("In Files");
    });

    endpoints.Map("employee/profile/{employeename}", async context =>
    {
        string? employeename =
Convert.ToString(context.Request.RouteValues["employeename"]);

        await context.Response.WriteAsync($"Employee name is
{employeename}");
    });
});
```

Default Parameter
If you want to assign default parameter to URL you can do that simple my assigning default value to route parameters

```
endpoints.Map("employee/profile/{employeename=admin}", async context =>
    {
        string? employeename =
Convert.ToString(context.Request.RouteValues["employeename"]);

        await context.Response.WriteAsync($"Employee name is
{employeename}");
    });
```

In the above example we are assigning admin as default value to employee name.

Optional Parameter
You can also make parameters as an optional by adding ? to parameter
Ex

```
endpoints.Map("employee/profile/{employeename?}", async context =>
    {
if(context.Request.RouteValues.ContainsKey(“emloyeename”))
{
        string? employeename =
```

```
Convert.ToString(context.Request.RouteValues["employeename"]);

        await context.Response.WriteAsync($"Employee name is
{employeename}");
}
else
{
  await context.Response.WriteAsync($"Employee name is null or empty");
}
    });
```

Route Constraint
If you want to restrict any specific type of value in parameter this concept is called as route constraint. Constraint is nothing but the condition which specifies this parameter value should be like this type.
Ex

“products/details/{id:int}”

Below are constraints that we can implements with route

Int : for integer values

Bool : for boolean values that match with true, false, TRUE, FALSE

Datetime : matches with valid datetime values with format “yyyy:MM:dd hh:mm:ss tt” and “MM/dd/yyy hh:mm:ss tt”.

```
  //datetime contraint
    endpoints.Map("daily/digestreport/{reportdate:datetime}", async
context =>
    {
        DateTime dt =
Convert.ToDateTime(context.Request.RouteValues["reportdate"]);
        await context.Response.WriteAsync($"Report date is
{dt.ToShortDateString()}");
    });
```

Decimal : matches with decimal values

Long : long matches with valid long values.

Guid : match with valid guid id

Minlength : restrict parameter to have at least specific length.
Ex:
"employee/profile/{employeename:minlength(3)=admin}"

maxlength: restrict parameter to specific length only
"employee/profile/{employeename:maxlength(10)}"

length(min, max) : restrict parameter to have at least min length and restrict to specific max length
"employee/profile/{employeename:length(3,10)}"
"employee/profile/{empid:range(0,10000)}"

regex(expression) : regular expression is a pattern where you can write a set of rules to implement regular expression.

```
 //regular expression

endpoints.Map("sales-report/2023/{year:int:min(1900)}/{month:regex(^(apr
|jul|oct|jan)$)}", async context =>
    {
        int year = Convert.ToInt32(context.Request.RouteValues["year"]);
        string month =
Convert.ToString(context.Request.RouteValues["month"]);
        await context.Response.WriteAsync($"Year {year} and month is
{month}");

    });
```

Custom Constraint Class
So if you want to apply some constraints where you want to implement logic and use it in multiple places you can create a custom constraint class. Custom constraint class implements interface called IRouteConstraint which has Match method.

Example

Custom Contraint Class

```
using System.Text.RegularExpressions;

namespace RoutingExample.CustomConstraint
{
    public class MonthsCustomConstraints : IRouteConstraint
    {
        public bool Match(HttpContext? httpContext, IRouter? route,
string routeKey, RouteValueDictionary values, RouteDirection
routeDirection)
        {
            if(!values.ContainsKey(routeKey))
            {
                return false;
            }
```

```
            Regex regex = new Regex("^(apr|jul|oct|jan)$");
            string? month = Convert.ToString(values[routeKey]);
            if(regex.IsMatch(month))
            {
                return true;
            }
            return false;
        }
    }
}
```

Using custom constraints in Program.cs

```
using RoutingExample.CustomConstraint;

var builder = WebApplication.CreateBuilder(args);

//register custom constraints
builder.Services.AddRouting(options =>{
    options.ConstraintMap.Add("months",
typeof(MonthsCustomConstraints));
});

var app = builder.Build();

app.UseRouting();
app.UseEndpoints(endpoints =>
{
endpoints.Map("sales-report/2023/{year:int:min(1900)}/{month:months}",
async context =>
    {
        int year = Convert.ToInt32(context.Request.RouteValues["year"]);
        string? month =
Convert.ToString(context.Request.RouteValues["month"]);
        await context.Response.WriteAsync($"Year {year} and month is
{month}");

    });
});
app.Run();
```

UseStaticFiles()
UserStaticFiles() is middleware that enables dot net core applications to serve static files if we are using an application. By default there is a folder with the name wwwroot which contains mostly all the static files. But it is not mandatory you can have this folder as per your requirement. To add new custom folder need to configure into usestatic file middleware

Ex

```
app.UseStaticFiles(new StaticFileOptions()
{
    FileProvider = new PhysicalFileProvider(
        Path.Combine(builder.Environment.ContentRootPath,
"myrootfolder")
    )
});
```

Questions for this assignment

1. What is Routing?
2. How Routing works in ASP.NET Core?
3. What are the important route constraints?
4. What is the purpose of the wwwroot folder?
5. How do you change the path of wwwroot folder?