Chapter 4
Introduction to Controllers

Controller is a class that is used to group up a set of actions or action methods. Controller is a class which is inherited from controllers base class. To add a controller simple we need to right click on add and select controller. After adding controller class need to register controller as service in program.cs. Like below

```
builder.Services.AddControllers();
```

After registering the server further we need to use the map controller method build method. This method is responsible for calling useRouting and useEndpoint internally.

```
app.MapControllers();
```

Controller Naming and Inheritance Rule

Should either or both

- Class name should suffixed with controller
- The [Controller] attribute is applied to class or its base class

Responsibilities for Controllers

- **Reading request -** Extracting data values from requests such as query string parameters, request body , request cookies , request headers etc.
- **Validation -** Validating incoming requests.
- **Invoking Method -** Responsible for invoking actual business logic method.
- **Response -** Choosing what kind of response has to be sent and preparing response accordingly.

ContentResult
ContentResult can represent any type of response based on the specified MIME type. MIME represents types of content such as text, html, json, application, xml, pdf etc.

```
[Route("")]
        public ContentResult Index()
        {
            return Content("<h1>Hello from index</h1>", "text/html");
        }
```

JsonResult
JsonResult can represent an object in JavaScript Object Notation format.
Eg {“firstName” : “James”, “lastName” : “Smith”, “age”:25}

```
 [Route("person")]
        public JsonResult Person()
        {
            Person p = new Person();

            p.Id = 1;
            p.FirstName = "Durga";
            p.LastName = "Mohite";
            p.Age = 35;

            return new JsonResult(p);
        }
```

File Result
It is used to return file as result there are again two types
Virtual File Result - When file is located in wwwroot folder at that time we can use virtual file result

```
[Route("file")]
        public VirtualFileResult File()
        {
            return new
VirtualFileResult("/Top-500-DotNet-Interview-Questions-2022-PDF
(1).pdf", "application/pdf");
        }
```

Physical File Result - When file is not stored in wwwroot folder or outside of the project at that time we can use physical file result

```
 [Route("physcialfile")]
        public PhysicalFileResult PhysicalFile()
        {
            return new
PhysicalFileResult(@"C:\Users\DURGA\Downloads\Table Invoice.pdf",
"application/pdf");
        }
```

FileContentResult - In real world scenarios files are stored in a database in byte format to return that we can use file content result.

```
[Route("filebyte")]
        public FileContentResult ByteFile()
        {
            byte[] bytes =
System.IO.File.ReadAllBytes(@"C:\Users\DURGA\Downloads\Table
```

```
Invoice.pdf");
            return new FileContentResult(bytes, "application/pdf");
        }
```

IActionResult
IActionResult is the parent interface which is responsible for returning various types of results. It is recommended to use IActionResult as a return type of action method because it is the parent of all results that we want to return.

Status Code Result
Status code sends an empty response with specific status code. Each status code represents a specific meaning.
Eg

return new StatusCodeReslut(status_code);

return new BadRequestResult();

return new UnauthorizedResult();

return new NotFoundResult();

- **Redirection:**
  - `RedirectResult`: Redirect to any URL.
  - `RedirectToActionResult`: Redirect to a specific action within your app.
  - `LocalRedirectResult`: Redirect to a local URL within the same app.

Questions
What is Controller?
What is an Action Method?
Explain different types of Action Results in asp.net core?
What's the HttpContext object? How can you access it within a Controller?