Chapter 5
Overview of Model Binding

In application we receive requests with input data in various sources like query string, route parameters, from body etc. It is the responsibility of server side code to retrieve that input parameters. We have a model binding concept in dot net which is responsible for retrieving data from various sources and making this data available for parameters. Model binding is a feature that reads values from http requests and passes them as arguments to the action method. So whenever a request hits the first model binding process is initiated and it happens in sequence of preference from first to last for below sources.

1. Form Fields
2. Request body
3. Route data
4. Query string parameters

[FromQuery]
Used to specify that you want to retrieve parameter from query string

```
public IActionResult ActionMethodName([FromQuery] type parameter)
{

}
```

[FromRoute]
Used to specify that you want to retrieve parameter from route

```
public IActionResult ActionMethodName([FromRoute] type parameter)
{

}
```

Models
Model is a class that represents structure of data as properties that you would like to receive from the request and send the response. Also known as POCO (Plain Old CLR Objects). Model binding here works in such a way that when a request is made model binding first creates an object of model class and maps parameters with class' properties automatically.

Model Validations
In the old school approach we used to write all validation code in the controller's action method which is length and repetitive thing. But nowadays we can use data annotations to apply validation on model class properties to avoid any repetition.

```
class ModelClassName
{
      [Attribute] //apply validation
public type Property {get; set;}
}
```

ModelState
Model state is property of controller base class that is available for all action methods in the controller which is used to check status of validation. After the model binding model validation stage occurs which checks all validation and if there are any validation errors then it will be stored in model state. It contains three properties mainly IsValid, Values and ErrorCount
IsValid - Specify if there is at least one validation error or not.
Values - Contains each model property with corresponding error.
ErrorCount - Returns number of errors.
Ex

```
public IActionResult Index(Person p)
       {
           if(!ModelState.IsValid)
           {
               List<string> errorsList =
                  //new List<string>();
               ModelState.Values.SelectMany(value =>
               value.Errors).Select(err =>
               err.ErrorMessage).ToList() ;
               string errors = string.Join("\n", errorsList);
               return BadRequest(errors);
           }
          return Content($"{p}");
       }
```

Model Validation

- **[Required(ErrorMessage=”value”)]**
  Specifies that property value is required cant taek blank or empty

- **[StringLength(int maximumLength, MinimumLength=value, ErrorMessage=”Value”)]**
  Specifies min and max length allowed in string

- **[Range(int minimum, int maximum, ErrorMessage=”value”)]**
  Specifies min and max allowed for numerical values

- **[RegularExpression(string pattern, ErrorMessage=”value”)]**
  Specifies the valid pattern

- **[EmailAddress(ErrorMessage=”value”)]**
  Specifies that the value should be a valid email address.

- **[Phone(ErrorMessage=”values”)]**
  Specifies that the value should be valid phone number

- **[Compare("Password", ErrorMessage ="Password and Confirm password is not matching")]**
  For comparing one property with other

- **CustomValidator**
  You can also add validation logic as per your requirement

Bind and BindNever

- **[Bind] -** Bind attribute specifies that only specified properties should be included in the model binding. Benefit of this is we can avoid over posting of nor required properties.

- **[BindNever] -** Bind never attribute specifies that the specified property should not be included in model binding.

FromBody

It enables the input formatters to read data from request body as JSON or XML only

```
Public IActionResult ActionMethodName([FromBody] type parameter)
{
}
```

To enable Input formatter we need to register below server in startup class or program class
`builder.Services.AddControllers().AddXmlSerializerFormatters();`

Questions

1. What is model binding in ASP.NET CORE?
2. How validation works in ASP.NET CORE MVC and how they follow DRY principle?