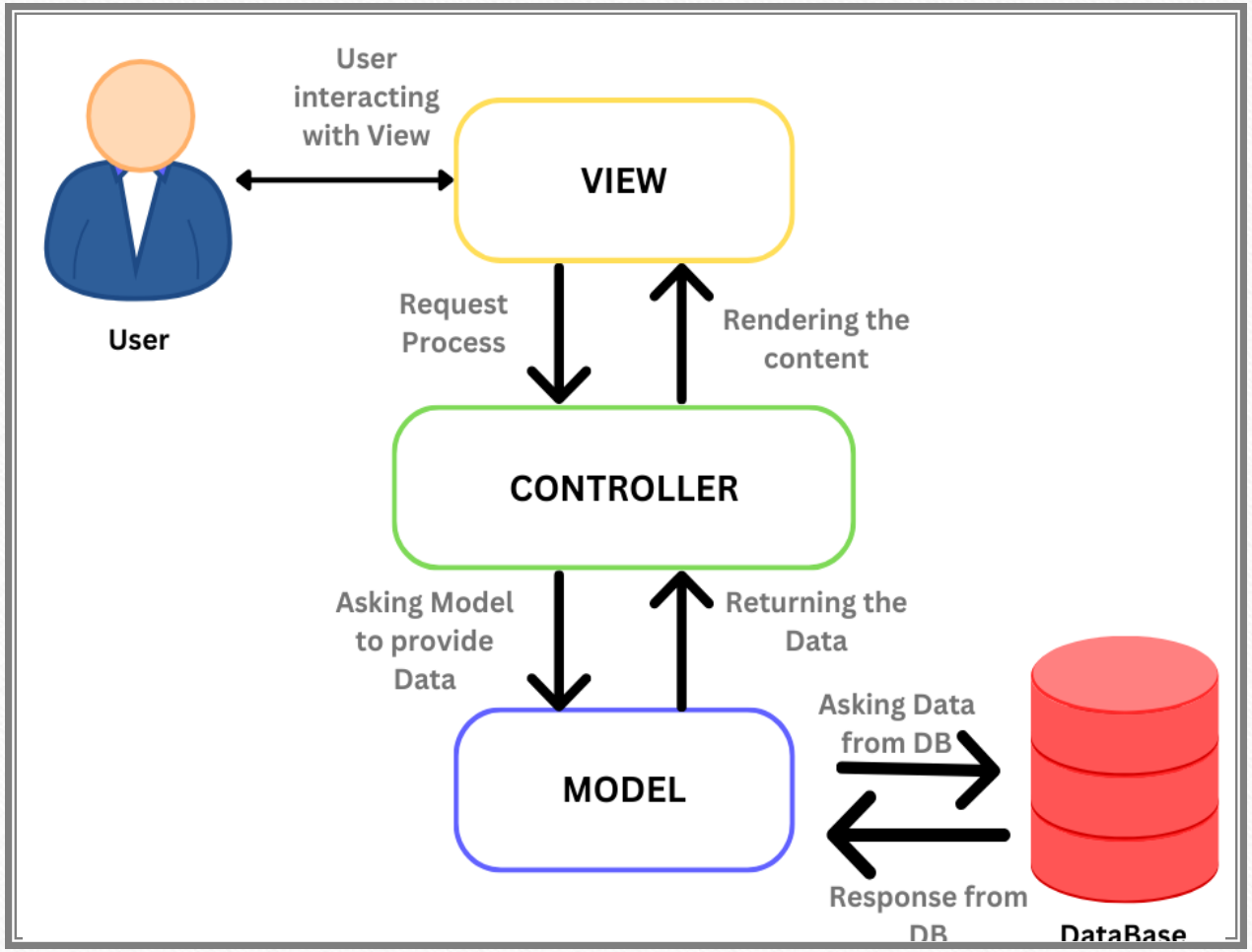
Basic Overview of ASP.NET MVC - .NET 7

Contents

- Introduction to MVC
- Routing
- Action Result
- View Data
- Razor Syntax
- Partial View
- Validation
- Environment Setup – GitHub

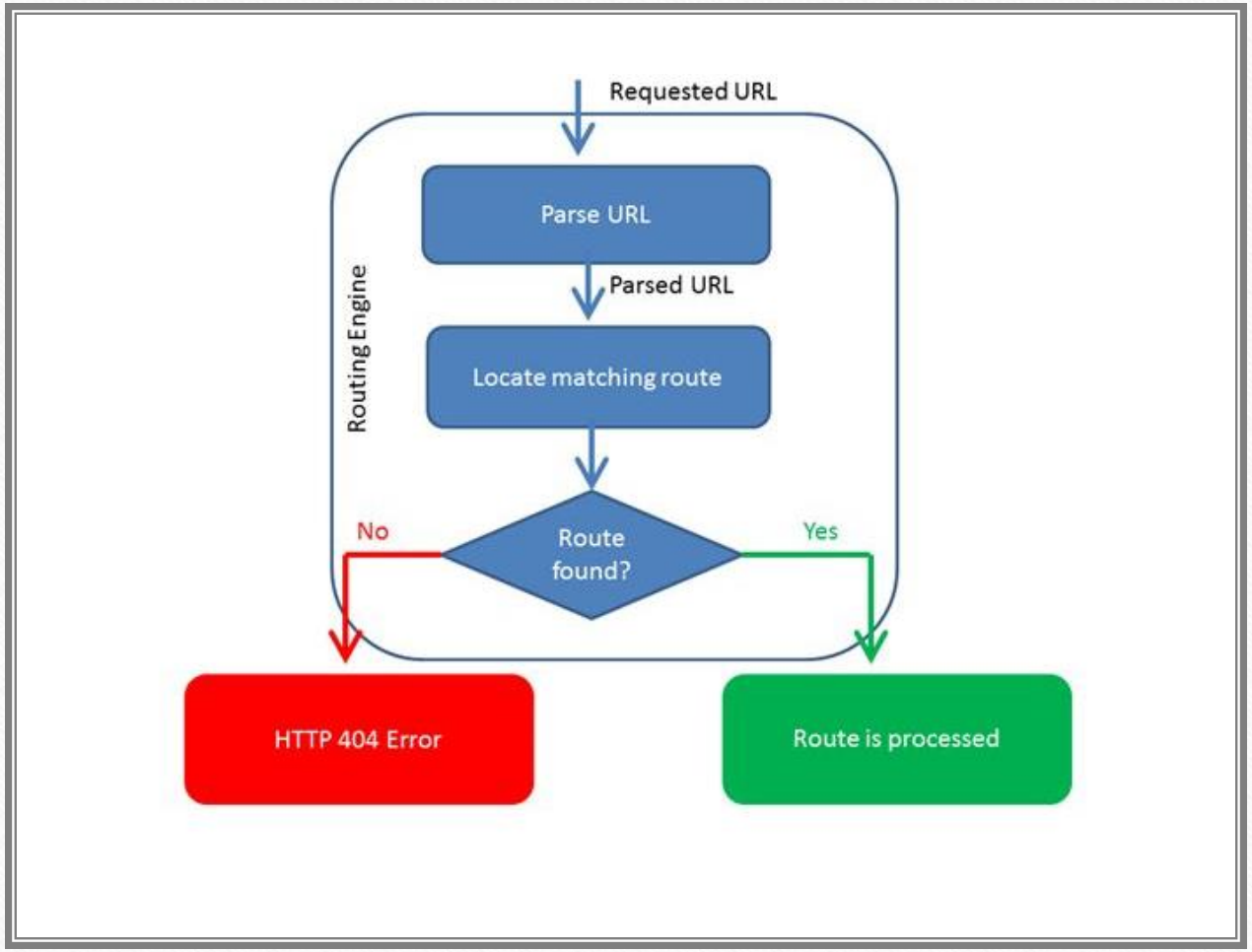
What is MVC?

- The MVC is an open-source framework built on top of the Microsoft .NET Framework to develop a web application that enables a clean code separation.
- MVC stands for Model, View, and Controller.
- It separates the code of UI from logic using controllers and views.
- Supports AJAX and Cross-browser compatible jQuery libraries.
- Supports XML and JSON encoding for clean, RESTful services.
- MVC provides routing capability that allows URLs to map to actions in controllers.



Model | View | Controller

- **Model**: The business and data logic of the application is isolated and encapsulated in models. All application logic is written in models.
- **View**: The views are responsible for rendering the model data into HTML markup. They contain presentation logic that manipulates the data and generates HTML markup based on Model data.
- **Controller**: The controller receives input from a user, calls business logic for processing the request, and then sends a response back to the client. The controller validates input data from the client and updates the model with any changes.

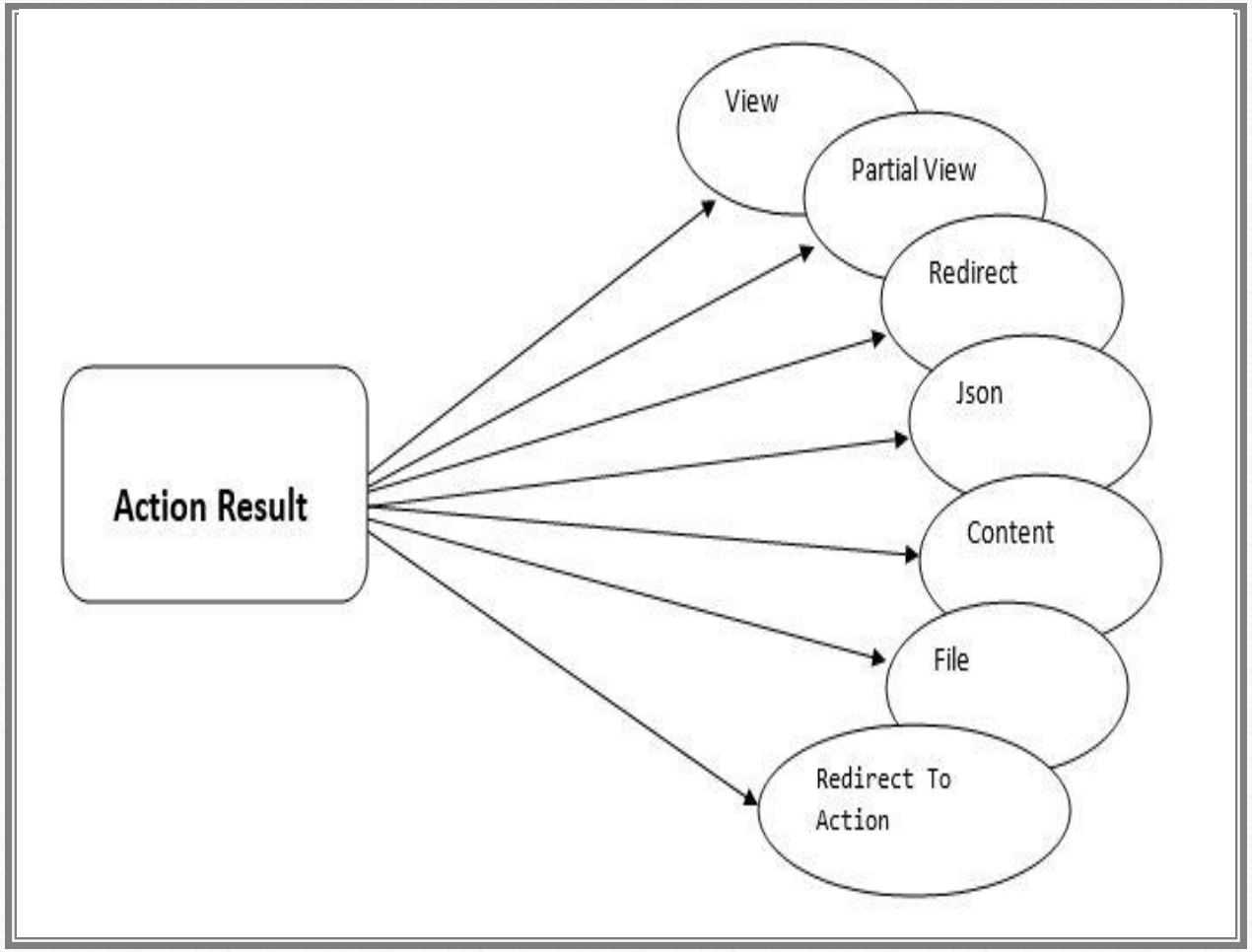


Routing

- Routing is a process of mapping the browser request to the controller action and return response back.
- Each MVC application has default routing for the default HomeController. We can set custom routing for newly created controller.
- The RouteConfig.cs file is used to set routing for the application.

Routing Using Attribute

- A route attribute is defined on top of an action method.
- We can also define route on action method and controller name.
- Syntax to add route at action method:
  - [Route(“Mvctest")]
- We can also define an optional parameter in the URL pattern by defining a question mark (“?") to the route parameter. We can also define the default value by using parameter=value.
  - [Route(“Mvctest /{ customerName ?}”)]
  - [Route(“Mvctest /{ customerName =0036952}”)]



Action Result

- All the public methods inside a Controller which respond to the URL are known as Action Methods. Action Result is the return type of an action method.
- It is the base class for all types that an action method returns.
- View, Partial View, Redirect, Json, Content, File, Redirect To Action, etc. are derived from the abstract Action Result class and these types can also be used as the return type of an action method.

ViewBag, ViewData & TempData

- **ViewBag:**
  - ViewBag is a dynamic object to passes the data from the Controller to View. This will pass the data as a property of the object ViewBag. And we have no need to typecast to read the data or for null checking. The scope of ViewBag is permitted to the current request, and the value of ViewBag will become null while redirecting.
- **ViewData:**
  - ViewData is a dictionary object to pass the data from Controller to View, where data is passed in the form of a key-value pair. Typecasting is required to read the data in View if the data is complex, and we need to ensure a null check to avoid null exceptions. The scope of ViewData is like ViewBag, and it is restricted to the current request, and the value of ViewData will become null while redirecting.
- **TempData:**
  - TempData is a dictionary object to pass the data from one action to another action in the same Controller or different Controllers. Usually, the TempData object will be stored in a session object. Tempdata is also required to typecast and for null checking before reading data from it. TempData scope is limited to the next request, and if we want TempData to be available even further, we should use Keep and Peek.

Example Controller

```
1  Public ActionResult Index()
2  {
3      ViewBag.Title = "Welcome";
4      return View();
5  }
```

Example View

```
1  <h2>@ViewBag.Title</h2>
```

Example Controller

```
1  Public ActionResult Index()
2  {
3      ViewData["Title"] = "Welcome";
4      return View();
5  }
```

Example View

```
1  <h2>@ViewData["Title"]</h2>
```

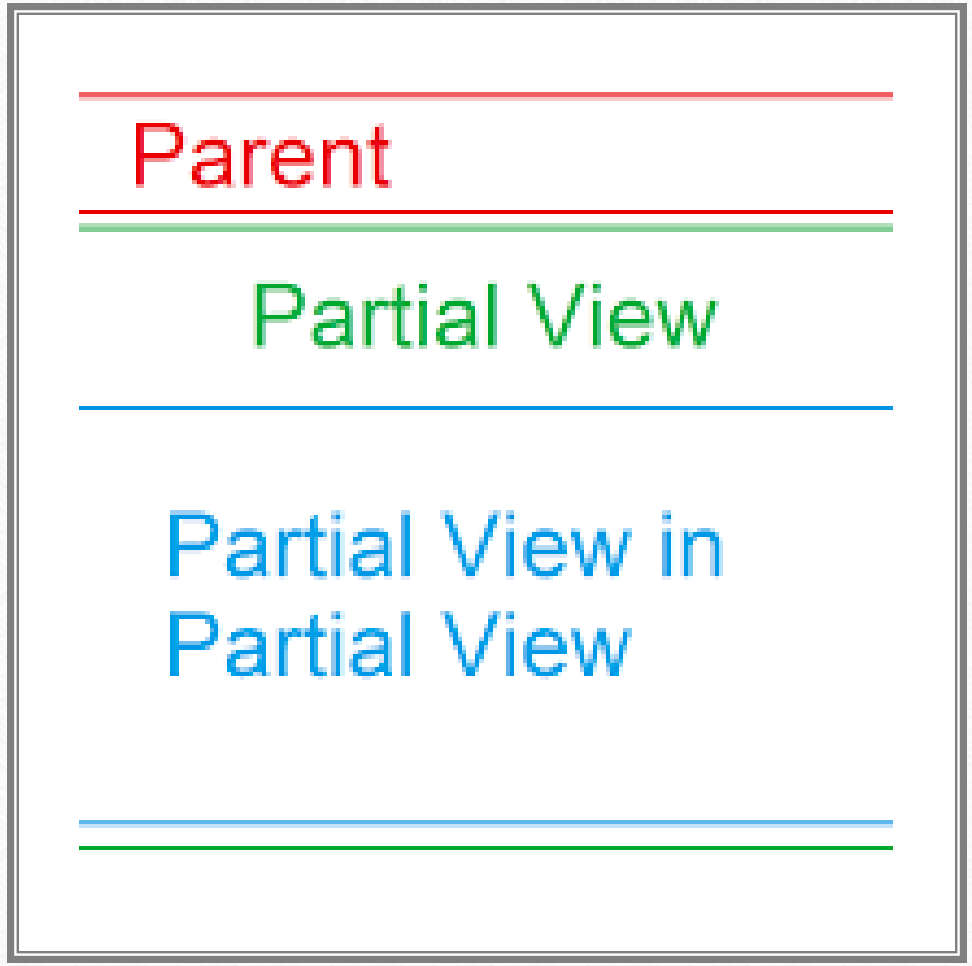
Example Controller

```
1  Public ActionResult Index()
2  {
3      TempData["Data"] = "I am from Index action";
4      return View();
5  }
6
7  Public string Get()
8  {
9      return TempData["Data"] ;
10 }
```

Razor Syntax

- Razor is a markup syntax that lets .NET based code into web pages.
- The Razor syntax consists of Razor markup, C#, and HTML.
- Files containing Razor generally have a .cshtml file extension.
- Razor code blocks are enclosed in @{ … }
- Inline expressions (variables and functions) start with @

```
01. @{
02.     ViewBag.Title = "Index";
03. }
04.
05. <h3>Razor Programming</h3>
06.
07. @for (int i = 1; i < 10; i++)
08. {
09.     <b>@i</b>
10. }
```



Partial Views

- Partial view in MVC is a special type of view that returns the portion of view content.
- It can be reusable in multiple views. It is used to help the duplication and reduce code.
- Partial view is a mostly reusable component so it's good practice to create a partial view inside a shared folder.

Validations

- MVC framework provides built-in annotations that we can apply on Model properties. It validates input and display appropriate message to the user. Below are some commonly used validation annotations:
  1. **Required:** It is used to make a required field.
  2. **StringLength:** It defines a maximum length for a string field.
  3. **Range:** It is used to set a maximum and minimum value for a numeric field.
  4. **RegularExpression:** It is used to associate regular expression for a field.
  5. **MaxLength:** It is used to set max length for a field.
  6. **MinLength:** It is used to set min length for a field.
  7. **Compare:** It is used to compare field to any other value.
  8. **CustomValidation:** It is used to validate the field by using custom validation function.
  9. **ValidationSummary:** It is used to specify summary of validation.

GitHub Account Setup

- Please go through the below links to setup your GitHub account:
  - https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account
  - https://www.wikihow.com/Create-an-Account-on-GitHub

Thank you!