Software Development

Presented By:
Khaled Gamal

ASP.NET Core

MVC

Section 4

Outlines

- **Passing Data From views to controllers**
    - **Request object**
    - **Parameters**
    - **IformCollection**

Teams Code is *bgdylhk*

Passing Data from Views to Controllers

Passing Data From views to controllers

- You can send data from view to controller using many ways like:
  - The From property Request object which is a property to the controller and a property in the HttpContext object.
  - Parameters (the action method accepts the same parameters)
  - IFormCollection or FormCollection object (the action method accepts a parameter of this type)

The Request object

- The ***Request*** object has view input field values in name/value pairs. When we create a submit button, the request type POST is created and calls the POST method.
- You can access the object directly or by the ***HttpContext*** object, which is the object constructed by the ASP.NET Core web server (Kestrel).
- The ***HttpContext*** is used by the application as a sort of storage box for a single request.
- Anything that's specific to this particular request and the subsequent response can be associated with it and stored in it, such as properties of the request, request-specific services, data that's been loaded, or errors that have occurred.
- The web server fills the initial HttpContext with details of the original HTTP request and other configuration details and then passes it on to the rest of the application.

The Request object

- You can access the **Form** property to reach the submitted data from a view using **indexer**.
- You will need type casting.
- Your code will be:

CS (in controller)

```
[HttpGet]
public IActionResult RequestObject()
{
return View();
}
[HttpPost]
[ActionName("RequestObject")]
public IActionResult RequestObjectP()
{
int num1 = Convert.ToInt32(Request.Form["Num1"]);
int num2 = Convert.ToInt32(Request.Form["Num2"]);
int num3 = Convert.ToInt32(Request.Form["Num3"]);
ViewBag.res = num1 + num2 + num3;
return View("RequestObject");
}
```

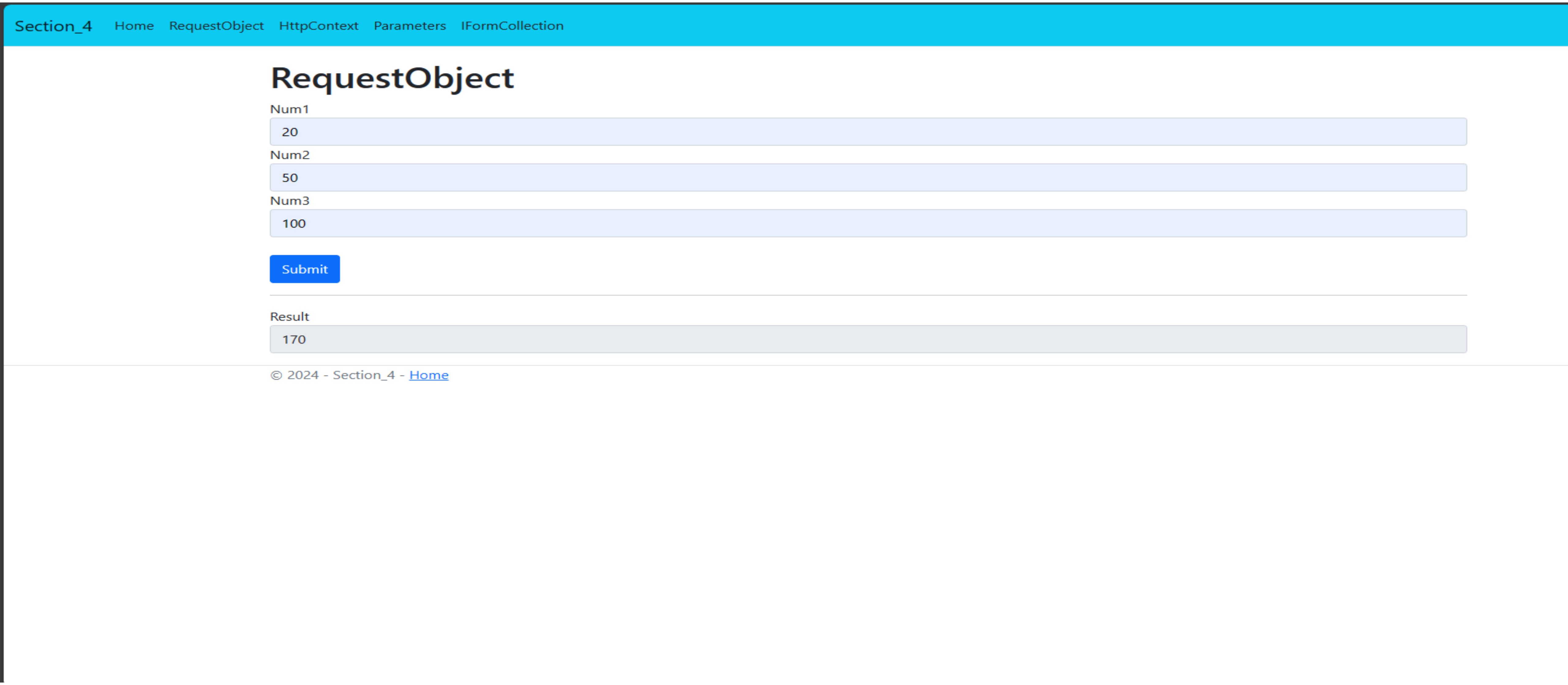
CSHTML (in View)

```
@{
ViewData["Title"] = "RequestObject";
}
<h1>RequestObject</h1>
<form method="post" asp-controller="home" asp-action="RequestObject">

<label>Num1</label>
<input type="text" class="form-control" placeholder="Enter Num1" name="Num1" />
<label>Num2</label>
<input type="text" class="form-control" placeholder="Enter Num2" name="Num2" />
<label>Num3</label>
<input type="text" class="form-control" placeholder="Enter Num3" name="Num3" />
<br />
<input type="submit" name="submit" class="btn btn-primary" value="Submit" />

<hr />
<label>Result</label>
<input type="text" value="@ViewBag.res" class="form-control" placeholder="Result"
name="Result" readonly />
</form>
```

The Request object



RequestObject

Num1	20
Num2	50
Num3	100

Result	170

© 2024 - Section_4 - Home

The Request object

- You can access the **Request** object also in the **HttpContext** object and access the **Form** property to reach the submitted data from a view using **indexer**.
- You will need type casting.
- Your code will be:

CS (in controller)

```
[HttpGet]
public IActionResult HttpContextObject()
{
return View();
}
[HttpPost]
[ActionName("HttpContextObject")]
public IActionResult HttpContextObjectP()
{
int num1 =
Convert.ToInt32(HttpContext.Request.Form["Num1"]);
int num2 =
Convert.ToInt32(HttpContext.Request.Form["Num2"]);
int num3 =
Convert.ToInt32(HttpContext.Request.Form["Num3"]);
ViewBag.res = num1 + num2 + num3;
return View("HttpContextObject");
}
```

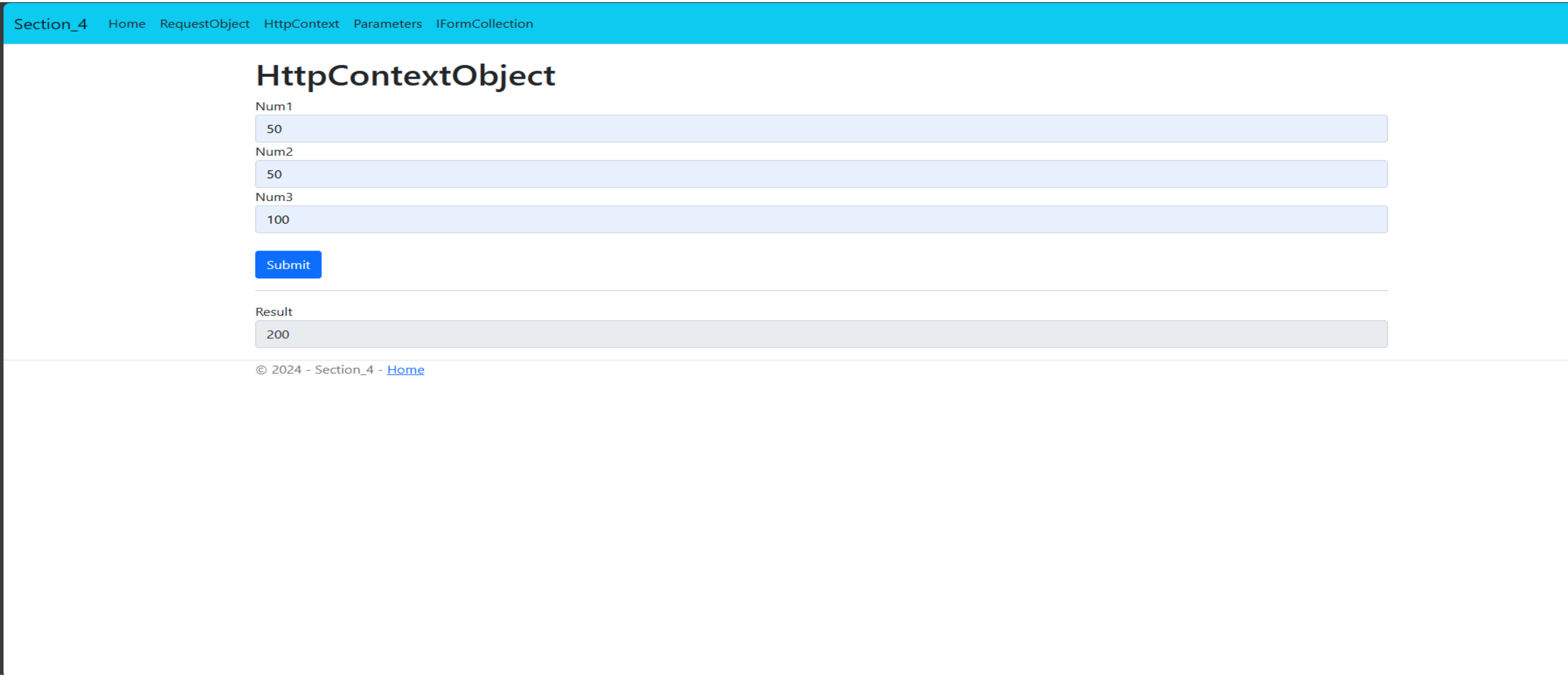
CSHTML (in View)

```
@{
ViewData["Title"] = "HttpContextObject";
}
<h1>HttpContextObject</h1>
<form method="post" asp-controller="home" asp-action="HttpContextObject">

<label>Num1</label>
<input type="text" class="form-control" placeholder="Enter Num1" name="Num1" />
<label>Num2</label>
<input type="text" class="form-control" placeholder="Enter Num2" name="Num2" />
<label>Num3</label>
<input type="text" class="form-control" placeholder="Enter Num3" name="Num3" />
<br />
<input type="submit" name="submit" class="btn btn-primary" value="Submit" />

<hr />
<label>Result</label>
<input type="text" value="@ViewBag.res" class="form-control"
placeholder="Result" name="Result" readonly />
</form>
```

The Request object



Section_4 Home RequestObject HttpContext Parameters IFormCollection

HttpContextObject

Num1	50
Num2	50
Num3	100

Result	200

© 2024 - Section_4 - Home

Parameters

- You can send data from view to controller (action) as parameters to action.
- You only need to name the controls that hold these data with the same name as the action parameters.
- Your code will be:

Parameters

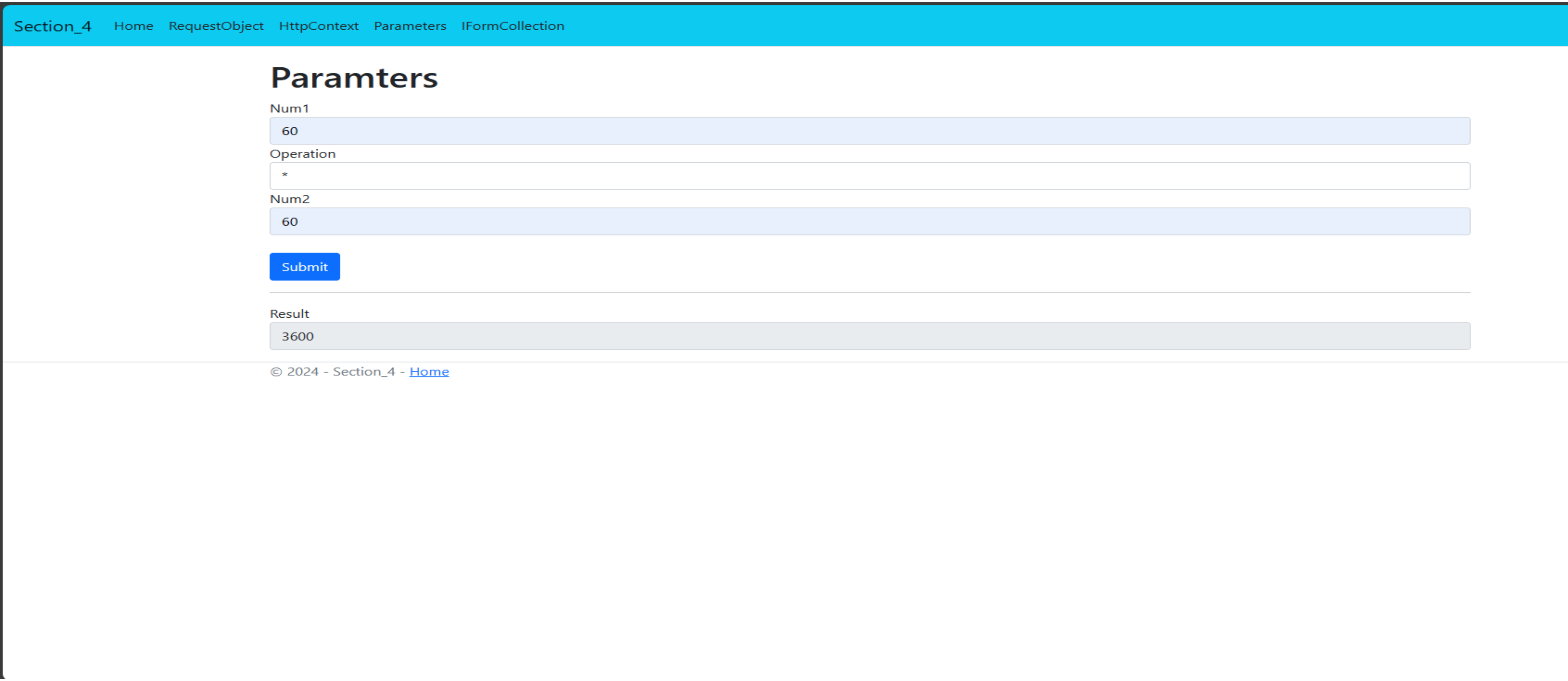
CS (in controller)

```
[HttpGet]
public IActionResult Paramt()
{
return View();
}
[HttpPost]
public IActionResult Paramt(int Num1, string OP, int Num2)
{
if (OP == "+")
ViewBag.res = Num1 + Num2;
else if (OP == "-")
ViewBag.res = Num1 - Num2;
else if (OP == "*")
ViewBag.res = Num1 * Num2;
else if (OP == "/")
ViewBag.res = Num1 / Num2;
return View();
}
```

CSHTML (in View)

```
@{
    ViewData["Title"] = "Parameters";
}
<h1>Paramters</h1>
<form method="post">
<label>Num1</label>
<input type="text" class="form-control" placeholder="Enter Num1" name="Num1" />
<label>Operation</label>
<select class="form-control" name="OP">
<option>+</option>
<option>-</option>
<option>*</option>
<option>/</option>
</select>
<label>Num2</label>
<input type="text" class="form-control" placeholder="Enter Num2" name="Num2" />
<br />
<input type="submit" name="submit" class="btn btn-primary" value="Submit" />
<hr />
<label>Result</label>
<input type="text" value="@ViewBag.res" class="form-control"
placeholder="Result" name="Result" readonly />
</form>
```

Parameters


Section_4 Home RequestObject HttpContext Parameters IFormCollection

Paramters

Num1	60
Operation	*
Num2	60

Submit

Result	3600

© 2024 - Section_4 - Home

IFormCollection - FormCollection

- You can send data from view to the controller (action) so, the action can receive them as a parameter of type IFormCollection or FormCollection.
- In the action you can access the data in the same way as the request object form using indexer.
- You will need type casting also.
- You only need to name the controls that hold these data with the same name as the action parameters.
- Your code will be:

IFormCollection - FormCollection

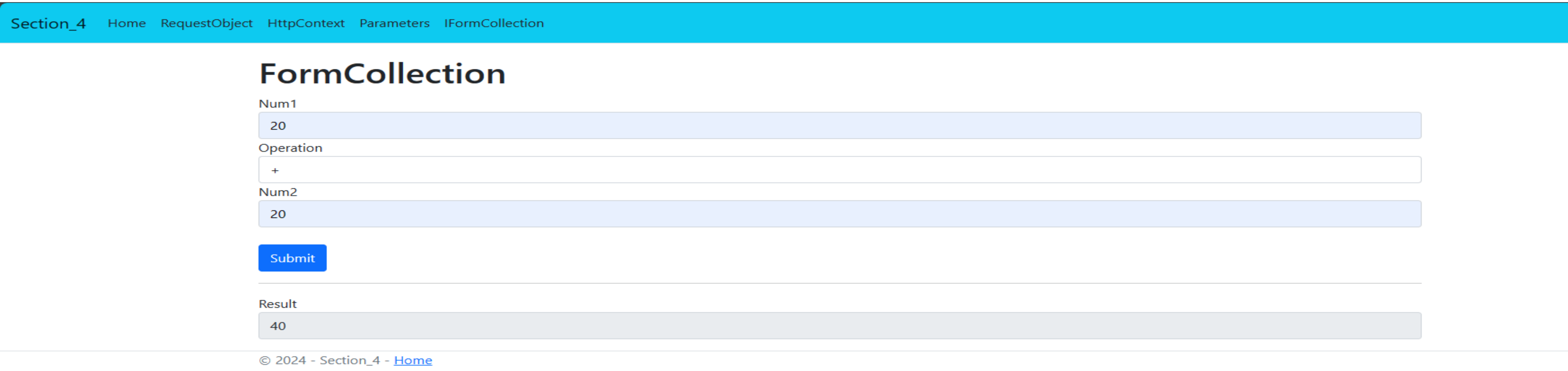
CS (in controller)

```
[HttpGet]
public IActionResult FrmCollection()
{
return View();
}
[HttpPost]
public IActionResult FrmCollection(IFormCollection
vals/*FormCollection vals */)
{
double n1 = Convert.ToDouble(vals["Num1"]);
double n2 = Convert.ToDouble(vals["Num2"]);
if (vals["OP"] == "+")
ViewBag.res = n1 + n2;
else if (vals["OP"] == "-")
ViewBag.res = n1 - n2;
else if (vals["OP"] == "*")
ViewBag.res = n1 * n2;
else if (vals["OP"] == "/")
ViewBag.res = n1 / n2;
return View();
}
```

CSHTML (in View)

```
@{
    ViewData["Title"] = "FormCollection";
}
<h1>FormCollection</h1>
<form method="post">
<label>Num1</label>
<input type="text" class="form-control" placeholder="Enter Num1" name="Num1" />
<label>Operation</label>
<select class="form-control" name="OP">
<option>+</option>
<option>-</option>
<option>*</option>
<option>/</option>
</select>
<label>Num2</label>
<input type="text" class="form-control" placeholder="Enter Num2" name="Num2" />
<br />
<input type="submit" name="submit" class="btn btn-primary" value="Submit" />
<hr />
<label>Result</label>
<input type="text" value="@ViewBag.res" class="form-control"
placeholder="Result" name="Result" readonly />
</form>
```

IFormCollection - FormCollection



FormCollection

Num1

20

Operation

+

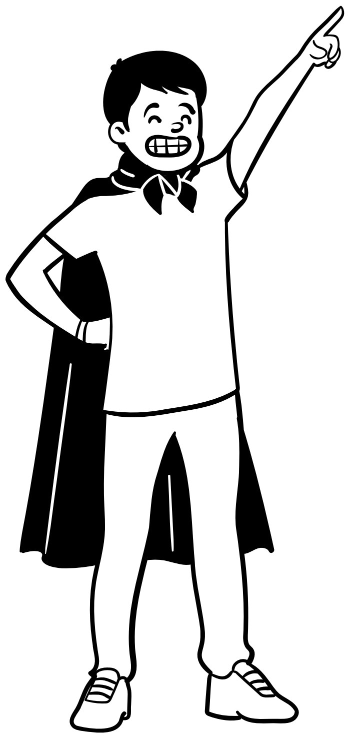
Num2

20

Result

40

© 2024 - Section_4 - Home



Any Questions?

THANK YOU!