Value type variables are not nullable, they don’t store null

Reference type variable store what by default null;

String name = null;

If you want to store null in value type variables , we use ? after variable name

Int? n = null;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication61.Controllers

{

public class SecondController : Controller

{

// GET: Second

public ActionResult Index()

{

return View();

}

public ActionResult Index1(int? id)

{

// Value type variables store their defautal value

// They are not nullable

if (id != null)

{

ViewBag.id = id;

}

return View();

}

}

}

@{

ViewBag.Title = "Index1";

}

<h2>Index1 of Second </h2>

@{ if (ViewBag.id != null)

{

@ViewBag.id

}

else

{

<b> There is no ID </b>

}

}

RouteConfig.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace WebApplication61

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Route1",

url: "{controller}/{action}/{id}/{name}",

defaults: new { controller = "Second", action = "Index1",

id = UrlParameter.Optional , name = UrlParameter.Optional}

);

routes.MapRoute(

name: "Route2",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Second", action = "Index1", name = UrlParameter.Optional }

);

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

}

}

public ActionResult Index2(int? id, string name)

{

// Value type variables store their defautal value

// They are not nullable

if (id != null)

{

ViewBag.id = id;

}

if(name!=null)

{

ViewBag.name = name;

}

return View();

}

@{

ViewBag.Title = "Index2";

}

<h2>Index2</h2>

@{ if (ViewBag.id != null)

{

@ViewBag.id

}

else

{

<b> NO ID PASSED </b>

}

if (ViewBag.name != null)

{

@ViewBag.name

}

else

{

<b> NO name has been passed </b>

}

}

Pass data from View to Controller

<https://www.c-sharpcorner.com/UploadFile/3d39b4/getting-data-from-view-to-controller-in-mvc/>

1. Using Traditional approach
2. Using the FormCollection Object
3. Using the Parameters
4. Strongly type model binding to view

Add 2 numbers. We will enter numbers in form

1. **Using Traditional approach**

Created a controller

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication61.Controllers

{

public class AddController : Controller

{

public ActionResult AddNumbers()

{

return View();

}

public ActionResult AddNumbersMethod()

{

int x = Convert.ToInt16( Request["txtNo1"]);

int y = Convert.ToInt16(Request["txtNo2"]);

ViewBag.result = x + y;

return View();

}

}

}

We added a View AddNumbers

@{

ViewBag.Title = "AddNumbers";

}

<h2>AddNumbers</h2>

<form action="AddNumbersMethod" method="post">

Enter No 1 : <input type="number" name="txtNo1" />

<br />

Enter No 2 : <input type="number" name="txtNo2" />

<br />

<button type="submit"> Add </button>

</form>

---------------------

Then we added a View AddNumbersMethod

@{

ViewBag.Title = "AddNumbersMethod";

}

<h2>AddNumbersMethod</h2>

@{if(ViewBag.result!=null)

{

<b> @ViewBag.result </b>

}

}