

CHỦ ĐỀ 2



CONTROLLER

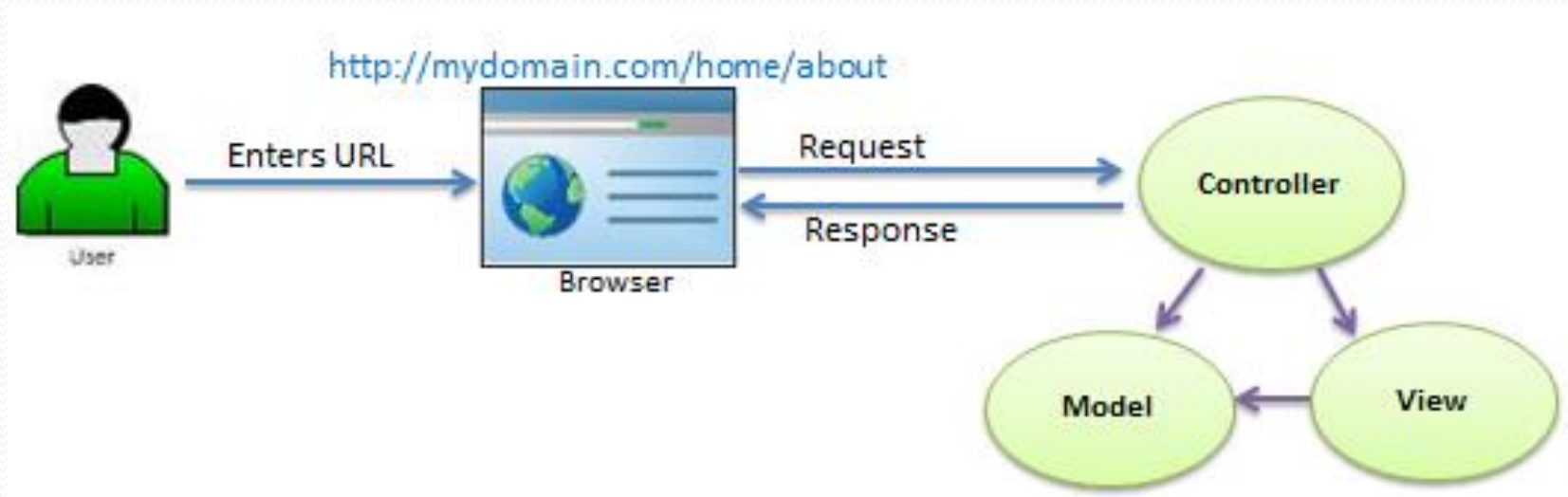
GVGD: PHẠM THỊ KIM NGOAN

Email: ngoanptk@ntu.edu.vn

Nội dung

- Giới thiệu Controllers
- Cách tạo Controllers
- Các phương thức trong Controllers
- Tham số trong Action

1. Giới thiệu Controllers



- Controller chịu trách nhiệm xử lý các **yêu cầu đến**, thực hiện các hoạt động trên Model và chọn View để hiển thị cho người dùng.
- Controller không chứa hoặc lưu trữ dữ liệu cũng như không tạo ra giao diện người dùng.

1. Giới thiệu Controllers

- Trong ASP.NET MVC Framework, Controller là các lớp .NET chứa logic cần thiết để xử lý một yêu cầu.
- Trong ASP.NET MVC, mọi tên lớp của Controller phải **kết thúc** bằng từ "**Controller**".
- Lớp controller trong MVC kế thừa từ lớp **Controller** chứa các phương thức có thể được sử dụng cho các mục đích khác nhau.
- Lớp Controller chứa các phương thức public gọi là các **phương thức hành động** (action).
- Controller và phương thức action xử lý các yêu cầu trình duyệt đến, lấy dữ liệu mô hình cần thiết và trả về cho trình duyệt.

1. Giới thiệu Controllers

- Lớp của Controller có 3 tính năng chính:
 - Phương thức hành động (action)
 - Kết quả hành động (Action result)
 - Bộ lọc (Filter)

Nội dung

- Giới thiệu Controllers
- Cách tạo Controllers
- Phương thức trong Controllers
- Tham số trong Action

2. Tạo Controller

➤ Tạo Controller thực thi **Controller**:

- **Controller**

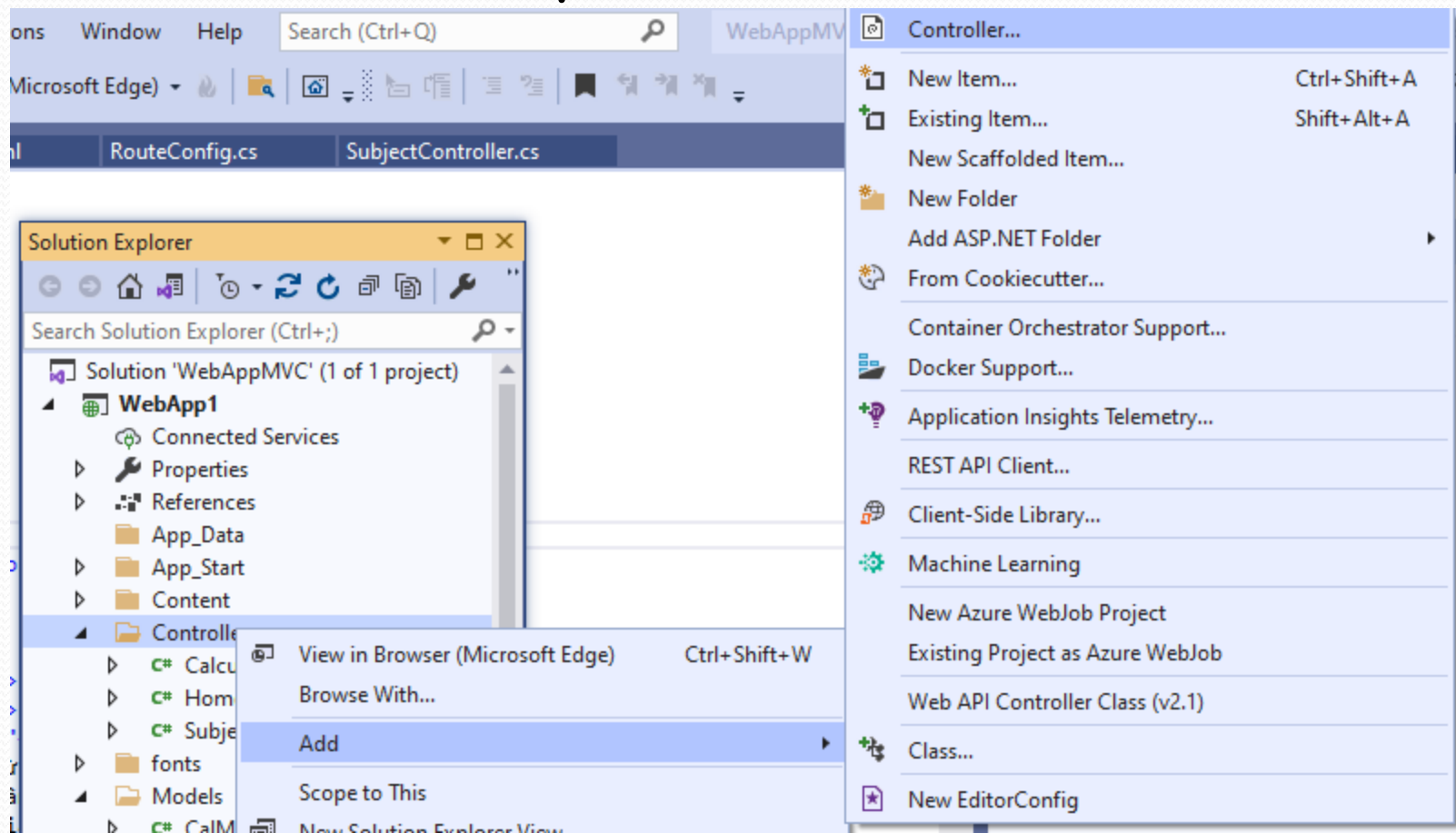
```
public interface Controller {  
    void Execute(RequestContext requestContext);  
}
```

- Tạo lớp trong Controllers thực thi Controller và cài đặt phần thân cho phương thức Execute

```
public class BasicController : Controller  
{  
    public void Execute(RequestContext requestContext)  
    {  
        ...  
    }  
}
```

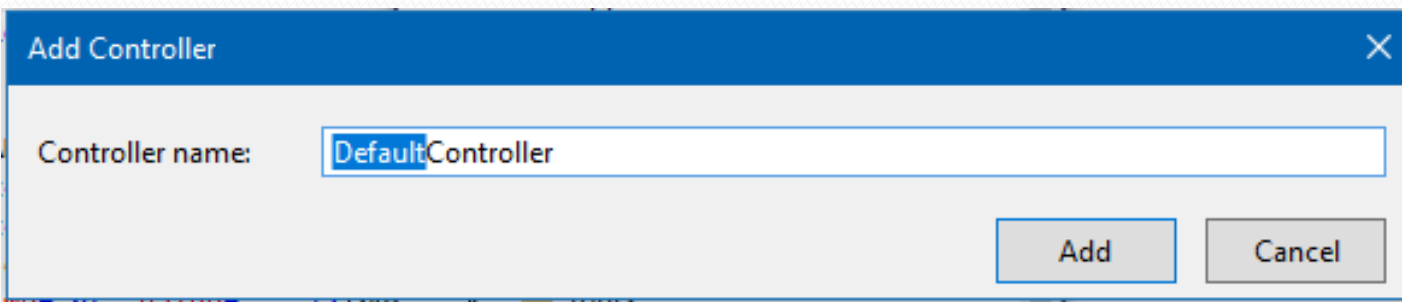
2. Tạo Controller

- Tạo Controller kế thừa từ lớp **Controller**:
 - R_Click trên thư mục Controllers → Add → Controller ...



2. Tạo Controller

- Tạo Controller kế thừa từ lớp **Controller**:
 - Chọn: Empty; read/write action; Entity Framework → Add.
 - Đặt tên Controller - phải có chuỗi controller ở cuối → Add.



Nội dung

- Giới thiệu Controllers
- Cách tạo Controllers
- Phương thức trong Controllers
- Tham số trong Action

3. Phương thức trong Controllers

- Tất cả các phương thức public trong các class của Controller đều gọi là phương thức action.
- Cú pháp phương thức action:

```
<access_modifier> <return_type> <MethodName> ([list of parameters])  
{  
    // body of the method  
}
```

- *<access_modifier>*: Phạm vi truy cập luôn là **public**
- *<return_type>*: Kiểu giá trị trả về của phương thức Action, thông thường sử dụng **ActionResult**.
- *<MethodName>*: Tên phương thức.
- *[list of parameters]*: Danh sách các tham số truyền vào.

3. Phương thức trong Controllers

➤ Một số ràng buộc phương thức action:

- Phạm vi truy cập bắt buộc là **public**.
- **Không** được **overloaded**.
- **Không** được thiết lập **static**.
- Tham số truyền vào, **không** được sử dụng **ref** và **out**.

➤ Action có các nhiệm vụ sau:

- Xác nhận tính chính xác của truy vấn
- Thực thi các lệnh tương ứng với truy vấn
- Lựa chọn loại phản hồi phù hợp

3. Phương thức trong Controllers

➤ Phương thức action mặc định:

- Mỗi controller có phương thức action mặc định được cài đặt trong lớp RouteConfig.
- Phương thức Index là phương action mặc định.

```
public class RouteConfig
{
    1 reference
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

3. Phương thức trong Controllers

- **Gọi phương thức action:** một controller có thể có nhiều phương thức action; có thể gọi phương thức action thông qua địa chỉ **URL** trên trình duyệt bằng **tên controller** và **tên phương thức**.

Cú pháp :

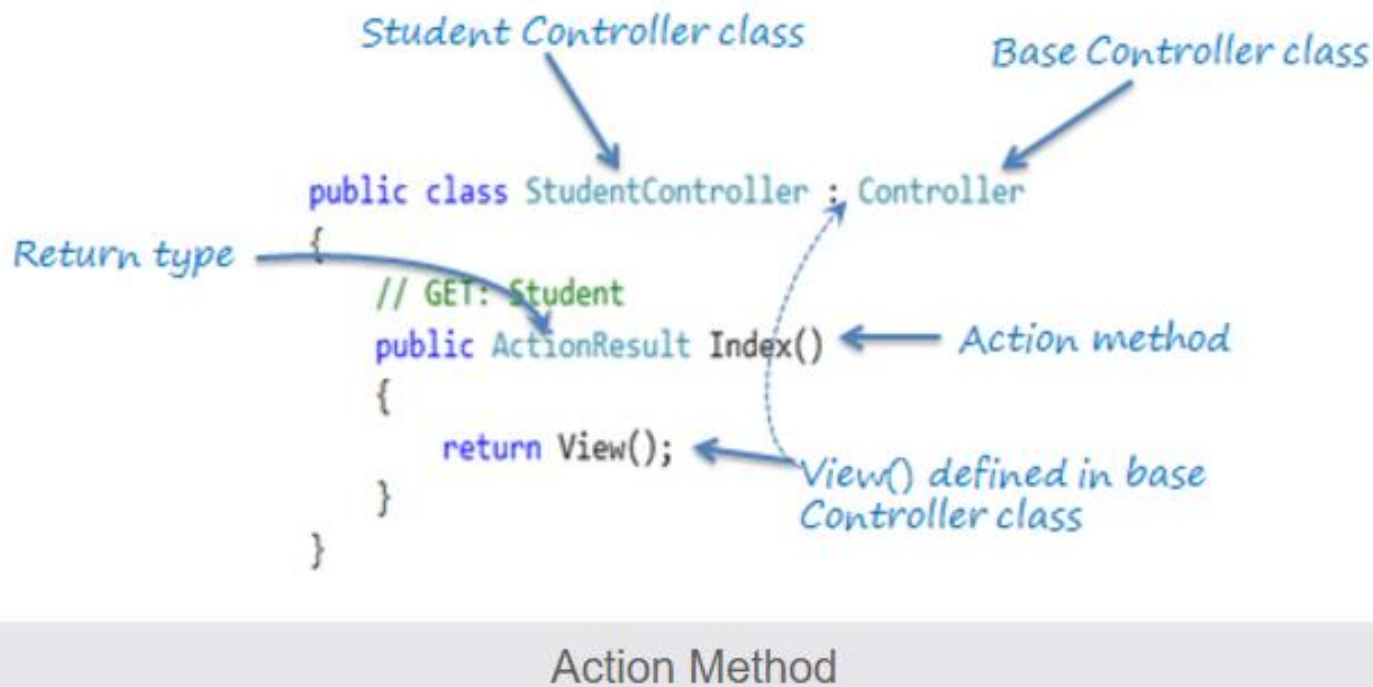
`http://<domain_name>/<controller_name>/<actionmethod_name>`

➤ **Mô tả :**

- `<domain_name>`: Tên domain của ứng dụng.
- `<controller_name>`: Tên controller của ứng dụng, chú ý tên controller không chứa hậu tố Controller.
- `<actionmethod_name>`: Tên phương thức Action của ứng dụng.

3. Phương thức trong Controllers

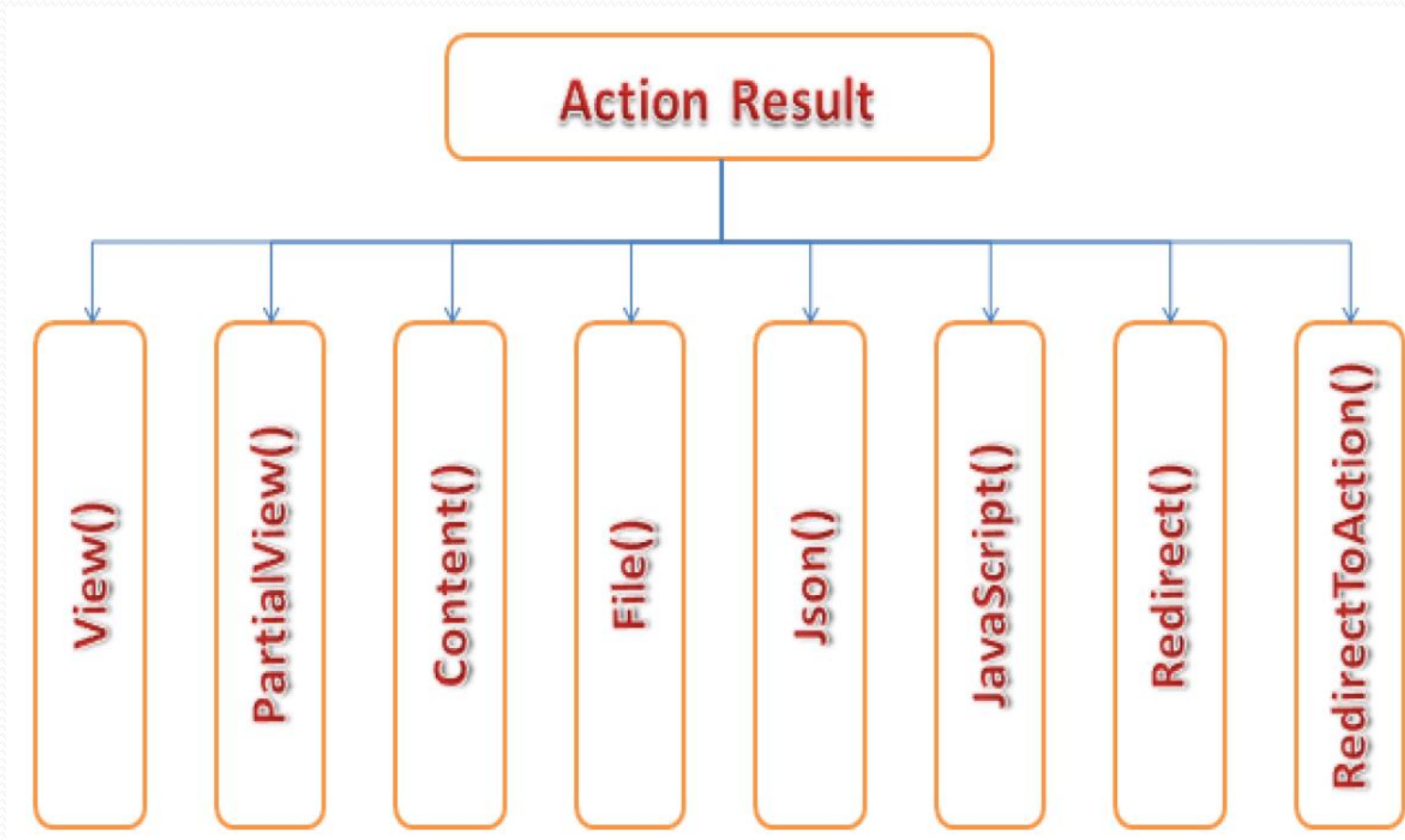
- Ví dụ: một phương thức action



- Gọi phương thức action:
localhost:53013/Student/Index

Kiểu trả về của Action

- Kiểu trả về của phương thức action



Kiểu trả về của Action

- Kiểu trả về của phương thức action:
 - **View()** trả về View được bọc trong layout (Master Page)
 - **PartialView()** giống như View, nhưng sẽ không được bọc trong layout.
 - **Content()** trả về text.
 - **Json()** trả về dữ liệu Json.
 - **Javascript()** trả về nội dung javascript
 - **File()** trả về nội dung file không bao gồm layout.
 - **RedirectToAction()** chuyển sang 1 Action khác.
 - **Redirect()** chuyển sang một url khác.

Kiểu trả về của Action

- Kiểu trả về của phương thức action: lớp ActionResult là lớp cơ sở của các lớp kết quả sau:

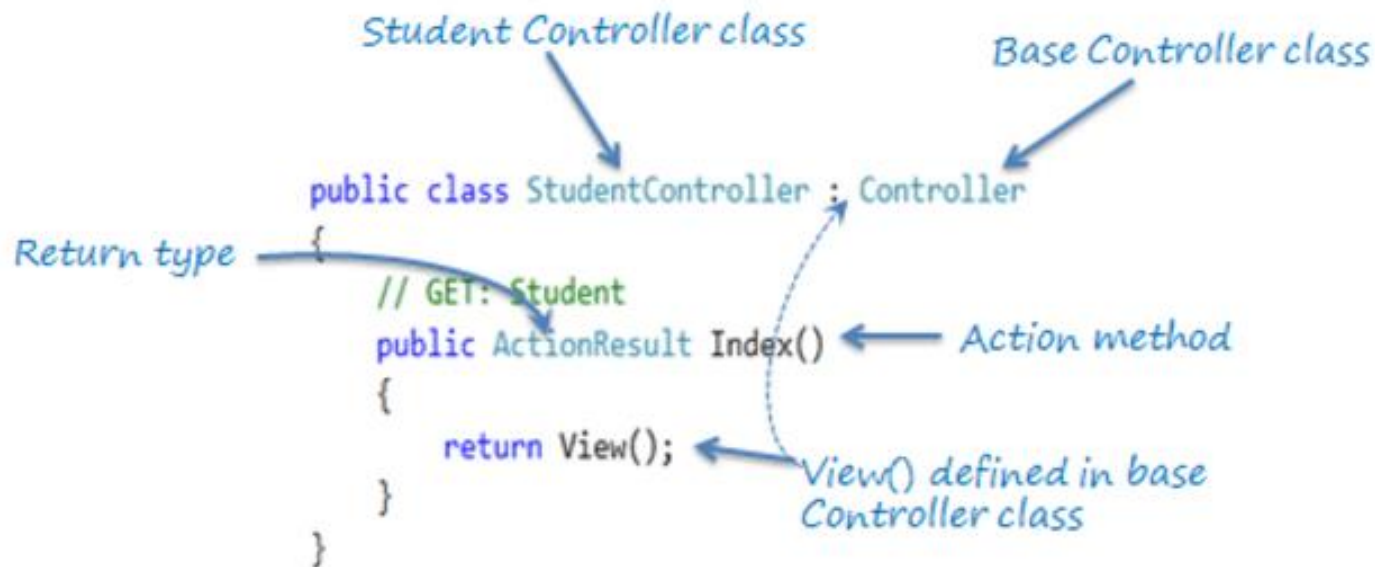
Result Class	Description
ViewResult	Trả về HTML
EmptyResult	Không trả về cái gì cả.
ContentResult	Trả về chuỗi ký tự
FileContentResult/ FilePathResult/ FileStreamResult	Trả về nội dung của tập tin
JavaScriptResult	Trả về JavaScript script.
JsonResult	Trả về một JSON có thể sử dụng trong AJAX
RedirectResult	Chuyển hướng người dùng
RedirectToRouteResult	Chuyển hướng hành động khác.
PartialViewResult	Trả về HTML từ Partial view
HttpUnauthorizedResult	Trả về lỗi và HTTP Code

Phương thức trong Controllers

- Phương thức View() trong lớp Controller cơ sở chứa các phương thức khác nhau, tự động trả về loại kết quả cụ thể như:

Result Class	Description	Base Controller Method
ViewResult	Trả về HTML.	View()
EmptyResult	Represents No response.	
ContentResult	Không trả về cái gì cả.	Content()
FileContentResult, FilePathResult, FileStreamResult	Trả về nội dung của tập tin	File()
JavaScriptResult	Trả về JavaScript script.	JavaScript()
JsonResult	Trả về một JSON có thể sử dụng trong AJAX	Json()
RedirectResult	Chuyển hướng người dùng	Redirect()
RedirectToRouteResult	Chuyển hướng hành động khác.	RedirectToRoute()
PartialViewResult	Trả về HTML từ Partial view	PartialView()
HttpUnauthorizedResult	Trả về lỗi và HTTP Code	

Phương thức trong Controllers



```

Student Controller class
Base Controller class

public class StudentController : Controller
{
    // GET: Student
    public ActionResult Index()
    {
        return View();
    }
}
    
```

Return type

Action method

View() defined in base Controller class

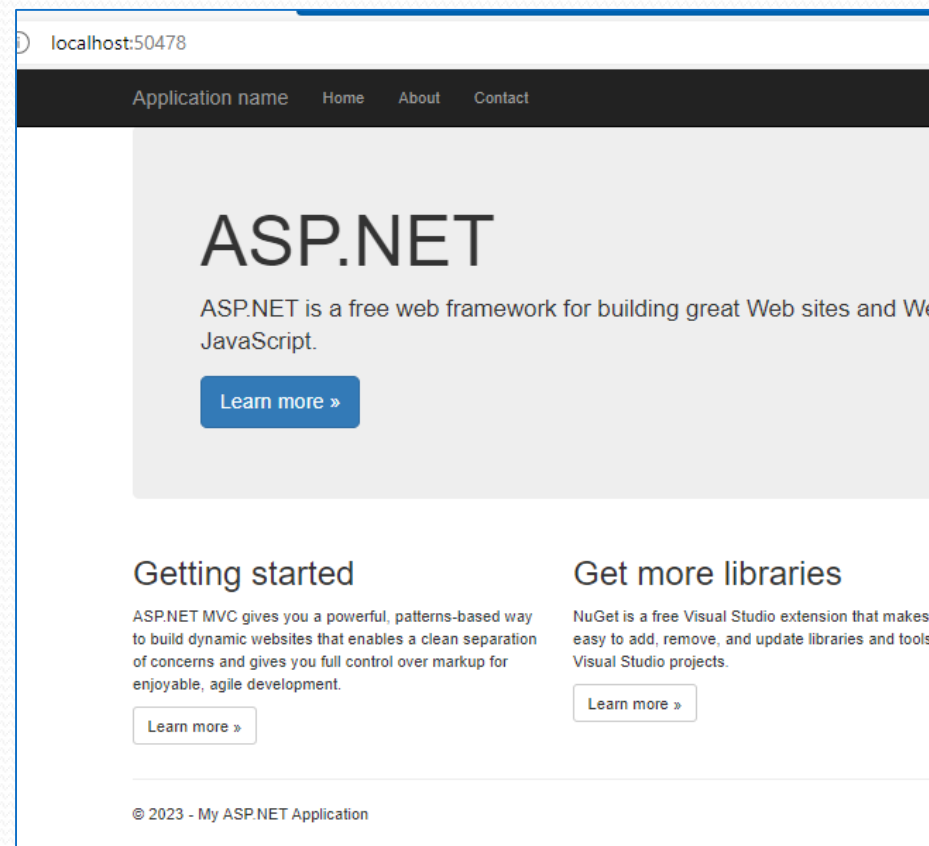
Action Method

- Phương thức Index() của StudentController trong hình trên sử dụng phương thức View() để trả về ViewResult (được lấy từ ActionResult).

Kiểu trả về là ActionResult

- Trả về một View là một trang html và nó có thể truyền tham số hóa model (**có layout**)

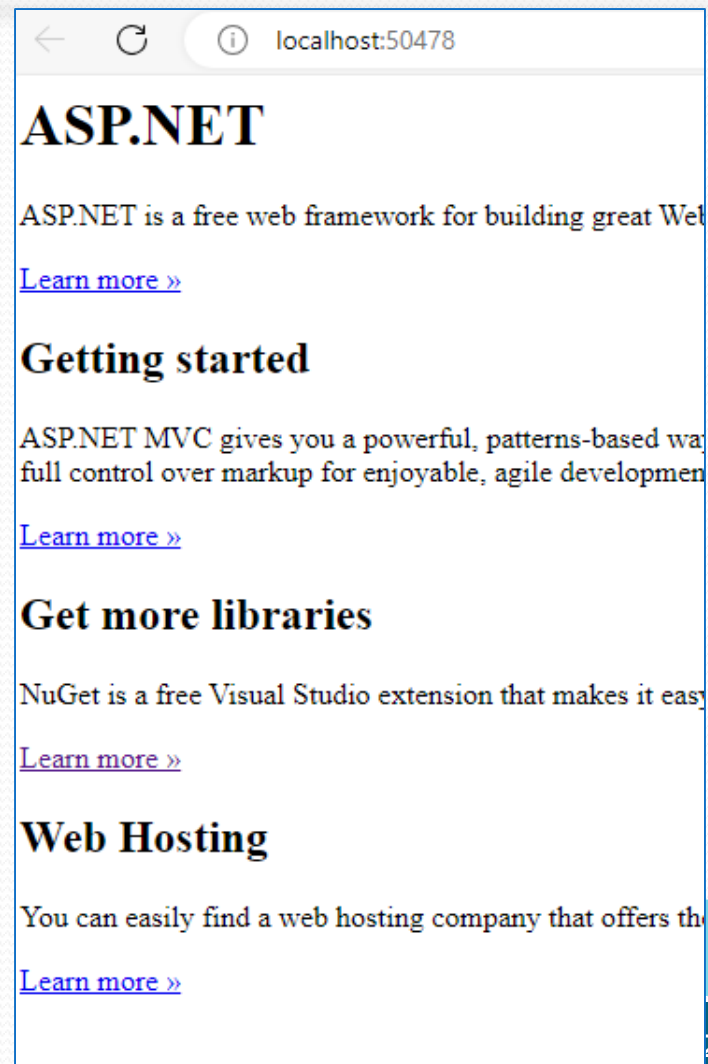
```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```



Kiểu trả về là PartialViewResult

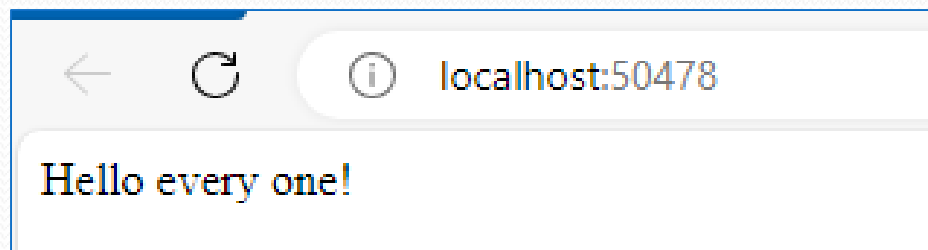
- PartialViewResult trả về một phần nhỏ của View (không có layout)

```
public class HomeController : Controller
{
    public PartialViewResult Index()
    {
        return PartialView();
    }
}
```



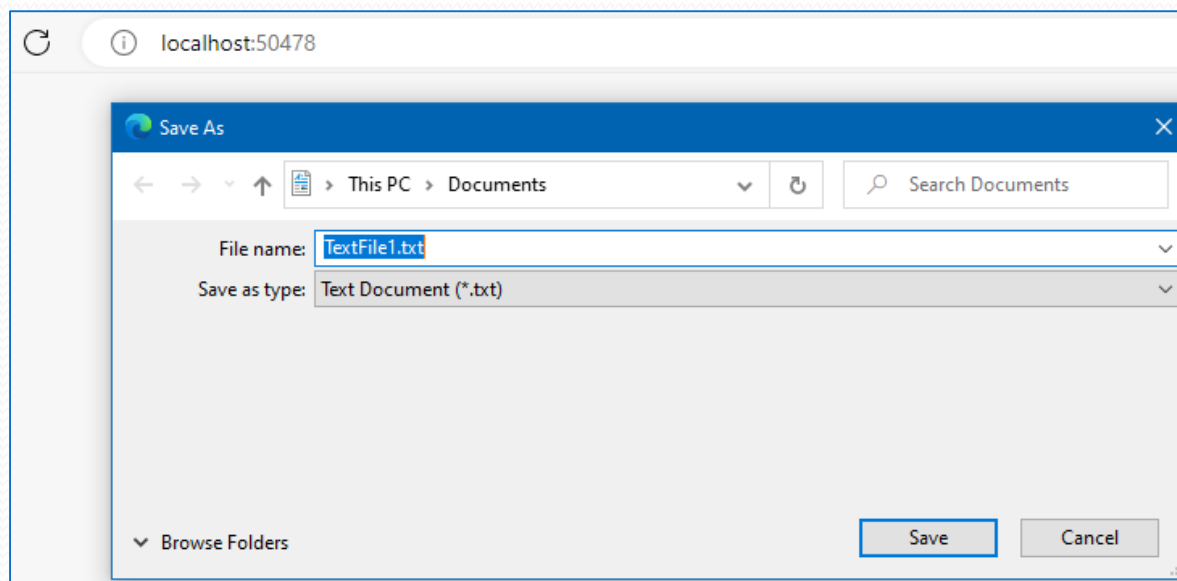
Kiểu trả về là ActionResult

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return Content("Hello every one!");
    }
}
```



Kiểu trả về là FilePathResult

```
public class HomeController : Controller
{
    public FilePathResult Index()
    {
        return File("~/Content/TextFile1.txt", "text/plain", "TextFile1.txt");
    }
}
```



Kiểu trả về là FileContentResult

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return File("~/Content/TextFile1.txt", "text/plain");
    }
}
```



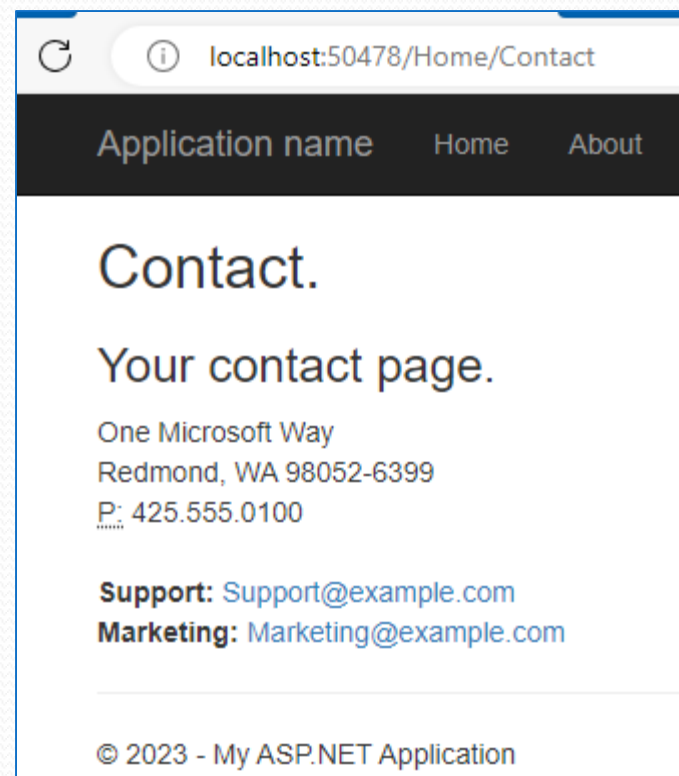
```
localhost:50478

public FilePathResult TestFilePathResult()
{
    string pathFile = Server.MapPath("~/Content");
    string fileName = "TextFile1.txt";
    return File(pathFile, "text/doc", fileName);
}
```

Kiểu trả về là RedirectResult

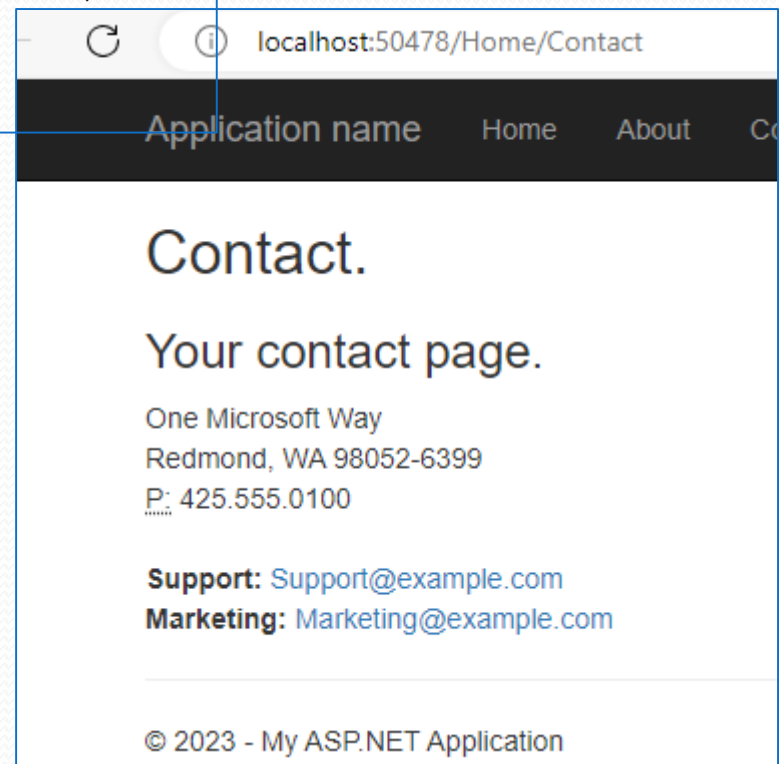
- Action đáp ứng việc chuyển trực tiếp tới một view khác.

```
public class HomeController : Controller
{
    public RedirectResult Index()
    {
        return Redirect("Home/Contact");
    }
}
```



Kiểu trả về là RedirectToRouteResult

```
public class HomeController : Controller
{
    public RedirectToRouteResult Index()
    {
        return RedirectToAction("Contact", "Home");
    }
}
```



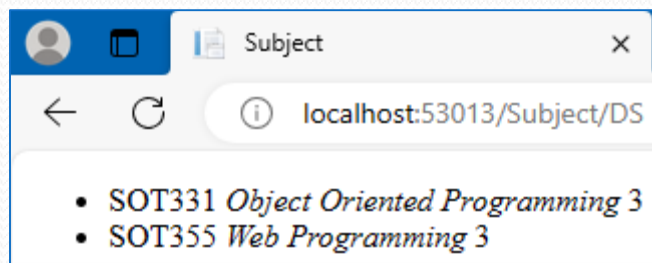
Action Selector

- **Action Selector** là các thuộc tính (Attribute) gắn với các phương thức Action. Action Selector **trợ giúp** cho các Route Engine **lựa chọn chính xác** các **phương thức Action** khi đáp ứng yêu cầu của các Http Request.
- Asp.Net MVC 5 bao gồm các Action Selectors:
 - ActionName
 - NonAction
 - ActionVerb

Action Selector

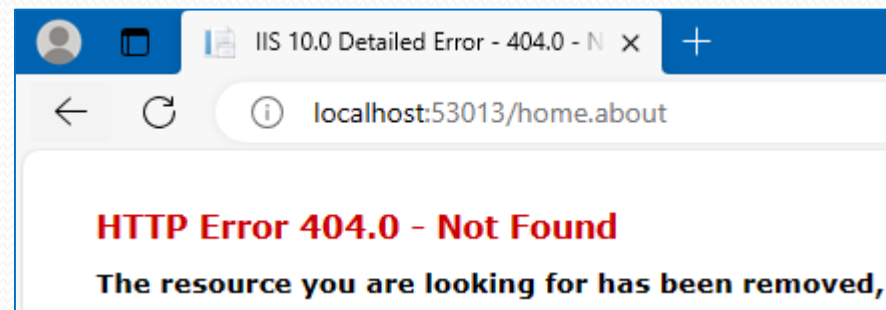
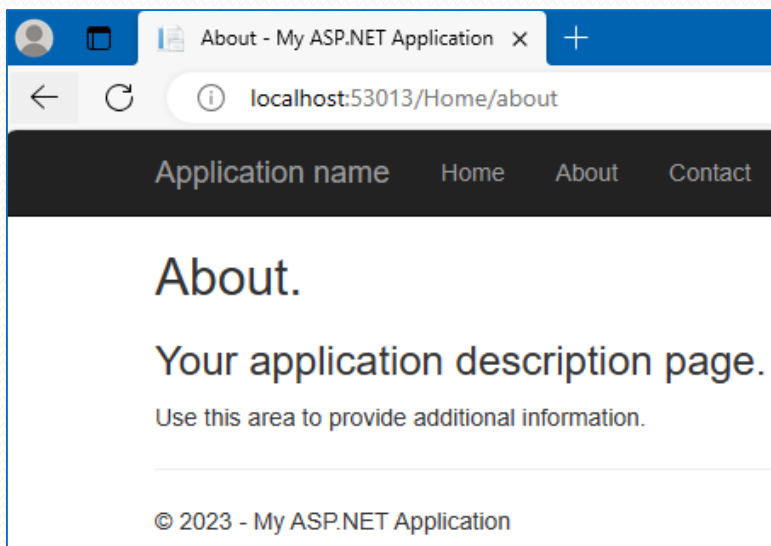
- **Action Name** cho phép sử dụng **tên khác nhau** của phương thức. Khi lập trình, chúng ta không muốn đưa ra tên phương thức thực thi thực sự của Controller ra phía Client.

```
public class SubjectController : Controller
{
    // GET: Subject
    [ActionName("DS")]//dùng tên khác cho action List
    public ActionResult List()
    {
        var model = new List<SubjectModel>()
        {
            new SubjectModel ("SOT331", "Object Oriented Programming", 3),
            new SubjectModel("SOT355","Web Programming",3)
        };
        return View(model);
    }
}
```



Action Selector

- **NonAction** khi muốn phương thức public trong Controller **không** thực hiện vai trò của một **phương thức action**.



```
[NonAction]
public ActionResult About()
{
    ViewBag.Message = "Your application description page.";
    return View();
}
```

Action Selector

- **ActionVerb** cho phép định nghĩa **hai phương thức** khác nhau nhưng **cùng tên**, một phương thức có thể dùng Http Get và phương thức kia thì đáp có thể dùng HttpPost.
- Một số ActionVerb:
 - HttpGet, HttpPost;
 - HttpPut;
 - HttpDelete;
 - HttpOptions & HttpPatch.
- ActionVerb mặc định là **HttpGet**.
- Trong một phương thức Action có thể sử dụng nhiều ActionVerb.

Action Selector

➤ Các phương thức Http:

Phương thức Http	Mô tả
GET	Các tham số trên URL (query string) tự động được thêm vào như các tham số. HttpGet được dùng để nhận một resource từ server.
POST	HttpPost attribute giới hạn action method chấp nhận HTTP Request sử dụng Post verb. Post verb được dùng để tạo mới bản ghi.
PUT	HttpPut attribute giới hạn action method chỉ chấp nhận các HTTP Request sử dụng Put verb. Put verb được dùng để cập nhật hoặc tạo mới tài nguyên.
HEAD	HttpHead attribute giới hạn action method chỉ chấp nhận HTTP Request sử dụng Head verb. Head verb được dùng để nhận các HTTP Header. Phương thức này được định danh cho GET ngoại trừ các server không trả về message body.
OPTIONS	HttpOptions attribute giới hạn action method chỉ chấp nhận các request sử dụng Options verb. Method này nhận thông tin về tùy chọn giao tiếp được hỗ trợ bởi web server.
DELETE	HttpDelete attribute giới hạn Action method chỉ chấp nhận HTTP Request sử dụng Delete verb. Delete verb được dùng để xóa tài nguyên đang tồn tại
PATCH	HttpPatch attribute giới hạn action method chỉ nhận các HTTP Request sử dụng Options verb. Method này sử dụng cho toàn bộ hoặc một phần việc cập nhật tài nguyên.

Nội dung

- Giới thiệu Controllers
- Cách tạo Controllers
- Phương thức trong Controllers
- Tham số trong phương thức Action

4. Tham số của Action

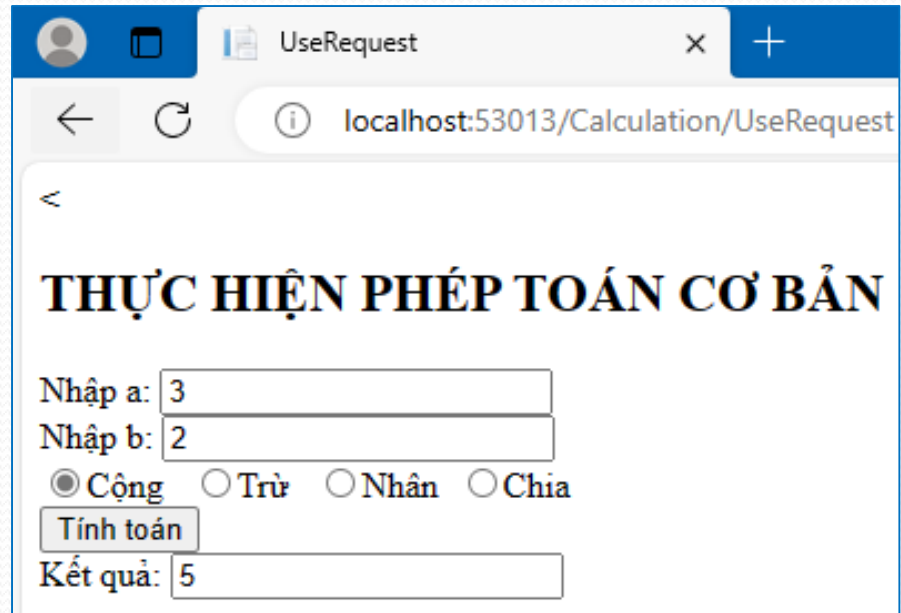
- Các cách tiếp nhận tham số yêu cầu trong MVC:
 - Sử dụng đối tượng ngầm định **Request**
 - Sử dụng đối số của **Action**
 - Sử dụng tham số của **FormCollection**
 - Sử dụng **Model**

4. Tham số của Action

- **Request** dùng để thông tin giữa Server và trình duyệt Client. Trình duyệt dùng đối tượng Request để gửi thông tin cần thiết tới Server. Đối tượng Request là thể hiện của HttpRequest.
- Sử dụng đối tượng ngầm định **Request** trong phương thức action để nhận tham số:
 - String value = **Request**["<tham số>"];
 - String value = **Request.QueryString**["<tham số>"];
 - String value = **Request.Form**["<tham số>"];
 - String value = **Request.Params**["<tham số>"];

Sử dụng Request

➤ Ví dụ dùng Request



The screenshot shows a web browser window with the title "UseRequest". The address bar displays "localhost:53013/Calculation/UseRequest". The page content includes a back button, a title "THỰC HIỆN PHÉP TOÁN CƠ BẢN", two input fields for "Nhập a:" (value 3) and "Nhập b:" (value 2), four radio buttons for "Cộng", "Trừ", "Nhân", and "Chia" (with "Cộng" selected), a "Tính toán" button, and a "Kết quả:" field (value 5).

- Thêm lớp **CalculationController** vào thư mục Controllers
- Tạo thư mục Calculation trong Views
- Tạo **UseRequest.cshtml** trong thư mục Calculation

Sử dụng Request

```
public class CalculationController : Controller
{
    //Sử dụng Request
    public ActionResult UseRequest()
    {
        return View();
    }
    [HttpPost]
    public ActionResult UseRequest(string pt)
    {
        double a = double.Parse(Request["a"]); //Chuyển đổi chuỗi sang số thực
        double b = double.Parse(Request["b"]);
        pt = Request["pt"].ToString();
        switch (pt)
        {
            case "+": ViewBag.KQ = a + b; break;
            case "-": ViewBag.KQ = a - b; break;
            case "*": ViewBag.KQ = a * b; break;
            case "/":
                if (b == 0) ViewBag.KQ = "Không chia được cho 0";
                else ViewBag.KQ = a / b; break;
        }
        return View();
    }
}
```

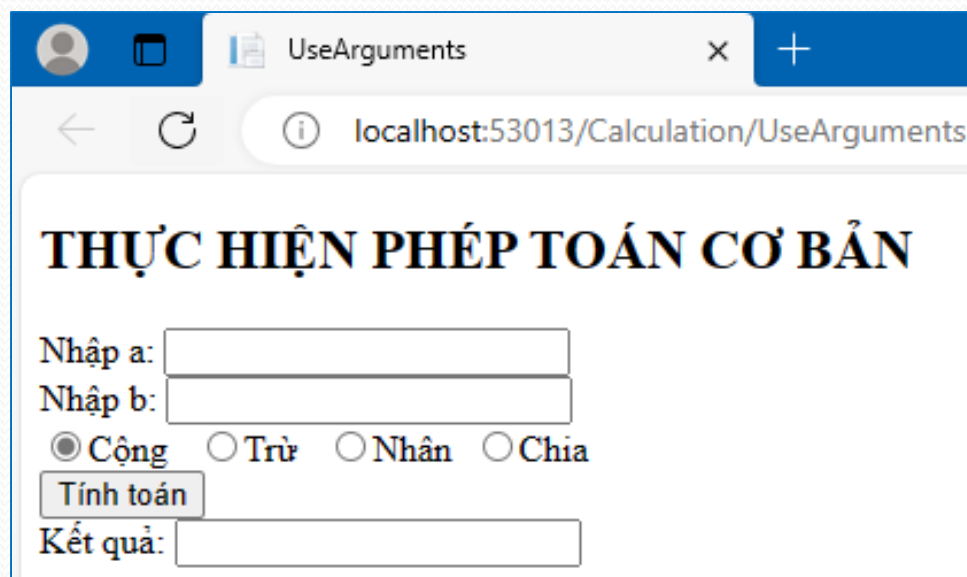
Sử dụng Request

```
<!DOCTYPE html>

<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>UseRequest</title>
</head>
<body>
  <<form name="f1" action="/Calculation/UseRequest" method="post">
    <div class="container">
      <div class="form">
        <h2>THỰC HIỆN PHÉP TOÁN CƠ BẢN</h2>
        Nhập a: <input type="text" name="a" /> <br />
        Nhập b: <input type="text" name="b" /> <br />
        <input type="radio" checked name="pt" value="+" /> Cộng &nbsp;  >
        <input type="radio" name="pt" value="-" /> Trừ &nbsp;  >
        <input type="radio" name="pt" value="*" /> Nhân&nbsp;  >
        <input type="radio" name="pt" value="/" /> Chia&nbsp;  ><br />
        <input type="submit" value="Tính toán" /><br />
        Kết quả: <input type="text" name="kq" value=@ViewBag.KQ />
      </div>
    </div>
  </form>
</body>
</html>
```

Sử dụng đối số của Action

➤ Ví dụ dùng đối số của Action



THỰC HIỆN PHÉP TOÁN CƠ BẢN

Nhập a:

Nhập b:

☒ Cộng ☐ Trừ ☐ Nhân ☐ Chia

Kết quả:

- Thêm phương thức UseArguments vào lớp **CalculationController** trong Controllers.
- Tạo **UseArguments.cshtml** trong thư mục Calculation của Views

Sử dụng đối số của Action

```
public class CalculationController : Controller
{
    //Sử dụng đối số của action

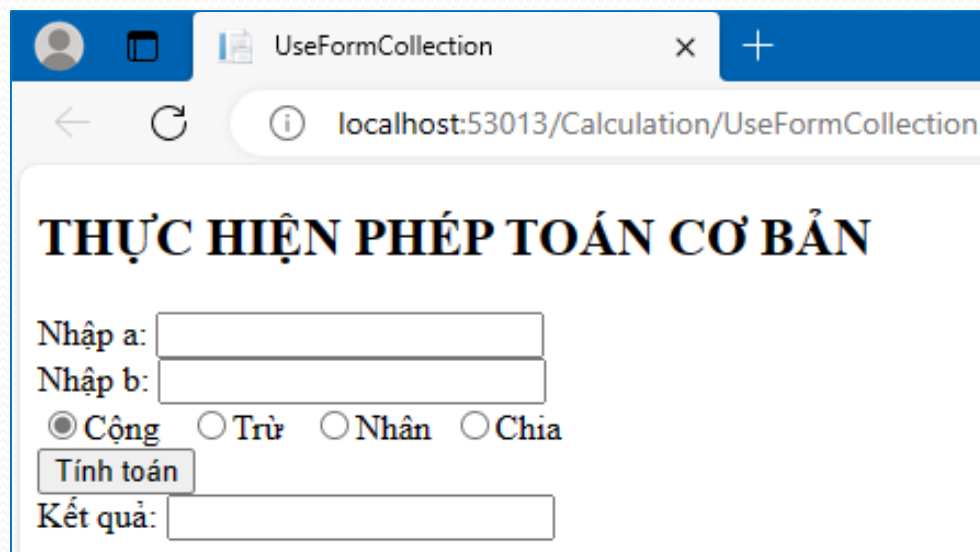
    public ActionResult UseArguments()
    {
        return View();
    }
    [HttpPost]
    public ActionResult UseArguments(double a, double b, string pt = "+")
    {
        switch (pt)
        {
            case "+": ViewBag.KQ = a + b; break;
            case "-": ViewBag.KQ = a - b; break;
            case "*": ViewBag.KQ = a * b; break;
            case "/":
                if (b == 0) ViewBag.KQ = "Không chia được cho 0";
                else ViewBag.KQ = a / b; break;
        }
        return View();
    }
}
```


Sử dụng đối số của Action

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>UseArguments</title>
</head>
<body>
  <form name="f1" action="/Calculation/UseArguments" method="post">
    <div class="container">
      <div class="form">
        <h2>THỰC HIỆN PHÉP TOÁN CƠ BẢN</h2>
        Nhập a: <input type="text" name="a" /> <br />
        Nhập b: <input type="text" name="b" /> <br />
        <input type="radio" checked name="pt" value="+" /> Cộng &nbsp;
        <input type="radio" name="pt" value="-" /> Trừ &nbsp;
        <input type="radio" name="pt" value="*" /> Nhân&nbsp;
        <input type="radio" name="pt" value="/" /> Chia&nbsp;<br />
        <input type="submit" value="Tính toán" /><br />
        Kết quả: <input type="text" name="kq" value=@ViewBag.KQ />
      </div>
    </div>
  </form>
</body>
</html>
```

Sử dụng FormCollection

➤ Ví dụ dùng FormCollection



UseFormCollection

localhost:53013/Calculation/UseFormCollection

THỰC HIỆN PHÉP TOÁN CƠ BẢN

Nhập a:

Nhập b:

☒ Cộng ☐ Trừ ☐ Nhân ☐ Chia

Kết quả:

- Thêm phương thức UseFormCollection vào lớp **CalculationController** trong Controllers.
- Tạo **UseFormCollection.cshtml** trong thư mục Calculation của Views.

Sử dụng FormCollection

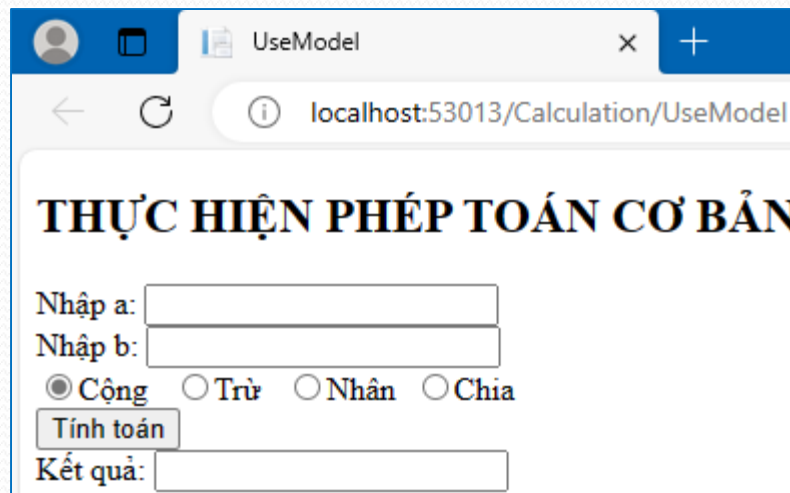
```
public class CalculationController : Controller
{    //Sử dụng FormCollection
    public ActionResult UseFormCollection()
    {        return View();    }
    [HttpPost]
    public ActionResult UseFormCollection(FormCollection f)
    {        double a = double.Parse(f["a"]); //Chuyển đổi chuỗi sang số thực
        double b = double.Parse(f["b"]);
        string pt = f["pt"].ToString();
        switch (pt)
        {
            case "+": ViewBag.KQ = a + b; break;
            case "-": ViewBag.KQ = a - b; break;
            case "*": ViewBag.KQ = a * b; break;
            case "/":
                if (b == 0) ViewBag.KQ = "Không chia được cho 0";
                else ViewBag.KQ = a / b; break;
        }
        return View();
    }
}
```

Sử dụng FormCollection

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>UseArguments</title>
</head>
<body>
    <form name="f1" action="/Calculation/UseFormCollection" method="post">
        <div class="container">
            <div class="form">
                <h2>THỰC HIỆN PHÉP TOÁN CƠ BẢN</h2>
                Nhập a: <input type="text" name="a" /> <br />
                Nhập b: <input type="text" name="b" /> <br />
                <input type="radio" checked name="pt" value="+" /> Cộng &nbsp;
                <input type="radio" name="pt" value="-" /> Trừ &nbsp;
                <input type="radio" name="pt" value="*" /> Nhân&nbsp;
                <input type="radio" name="pt" value="/" /> Chia&nbsp;<br />
                <input type="submit" value="Tính toán" /><br />
                Kết quả: <input type="text" name="kq" value=@ViewBag.KQ />
            </div>
        </div>
    </form>
</body>
</html>
```

Sử dụng Model

➤ Ví dụ dùng Model



The screenshot shows a web browser window with the title 'UseModel'. The address bar displays 'localhost:53013/Calculation/UseModel'. The page content is titled 'THỰC HIỆN PHÉP TOÁN CƠ BẢN'. It features two input fields for 'Nhập a:' and 'Nhập b:'. Below these are four radio buttons for the operations: 'Cộng' (selected), 'Trừ', 'Nhân', and 'Chia'. A 'Tính toán' button is positioned below the radio buttons. At the bottom, there is a 'Kết quả:' label followed by an empty output field.

- Thêm lớp **CalModel** vào thư mục Models.
- Thêm phương thức UseModel vào lớp **CalculationController** trong Controllers.
- Tạo **UseModel.cshtml** trong thư mục Calculation của Views.

Sử dụng Model

➤ Lớp CalModel trong thư mục Models

```
public class CalModels
{
    public double a { get; set; }
    public double b { get; set; }
    public string pt { get; set; }
}
```

Sử dụng Model

```
public class CalculationController : Controller
{    //Sử dụng Model

    public ActionResult UseModel()
    {    return View();    }
    [HttpPost]
    public ActionResult UseModel(CalModels cal)
    {
        switch (cal.pt)
        {
            case "+": ViewBag.KQ = cal.a + cal.b; break;
            case "-": ViewBag.KQ = cal.a - cal.b; break;
            case "*": ViewBag.KQ = cal.a * cal.b; break;
            case "/":
                if (cal.b == 0) ViewBag.KQ = "Không chia được cho 0";
                else ViewBag.KQ = cal.a / cal.b; break;
        }
        return View();
    }
}
```

Sử dụng FormCollection

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>UseArguments</title>
</head>
<body>
    <form name="f1" action="/Calculation/UseModel" method="post">
        <div class="container">
            <div class="form">
                <h2>THỰC HIỆN PHÉP TOÁN CƠ BẢN</h2>
                Nhập a: <input type="text" name="a" /> <br />
                Nhập b: <input type="text" name="b" /> <br />
                <input type="radio" checked name="pt" value="+" /> Cộng &nbsp;  
                <input type="radio" name="pt" value="-" /> Trừ &nbsp;  
                <input type="radio" name="pt" value="*" /> Nhân&nbsp;  
                <input type="radio" name="pt" value="/" /> Chia&nbsp;  <br />
                <input type="submit" value="Tính toán" /><br />
                Kết quả: <input type="text" name="kq" value=@ViewBag.KQ />
            </div>
        </div>
    </form>
</body>
</html>
```


Kết thúc chủ đề 2

