Controller trong ASP.Net MVC

Nội dung

- Định nghĩa và giải thích về Controller
- Giải thích Controller làm việc như thế nào
- Giải thích cách triệu gọi các Action
- Giải thích về cơ chế điều hướng
- Cấu trúc của URL

Controller

- Trong ASP.Net MVC, Controller thường làm các việc sau:
  - Quản lý luồng công việc trong ứng dụng
  - Tiếp nhận yêu cầu từ client, thực thi các đoạn mã lệnh để giải quyết yêu cầu và trả về kết quả đáp ứng yêu cầu đến từ client.
  - Giao tiếp với thành phần Model để lấy dữ liệu và lựa chọn View tương ứng để hiển thị dữ liệu cho client
- Các Controller được tạo ra kế thừa từ class Controller trong System.Web.Mvc namespace
- Controller cho phép tách phần xử lý logic nghiệp vụ khỏi phần hiển thị dữ liệu

Controller

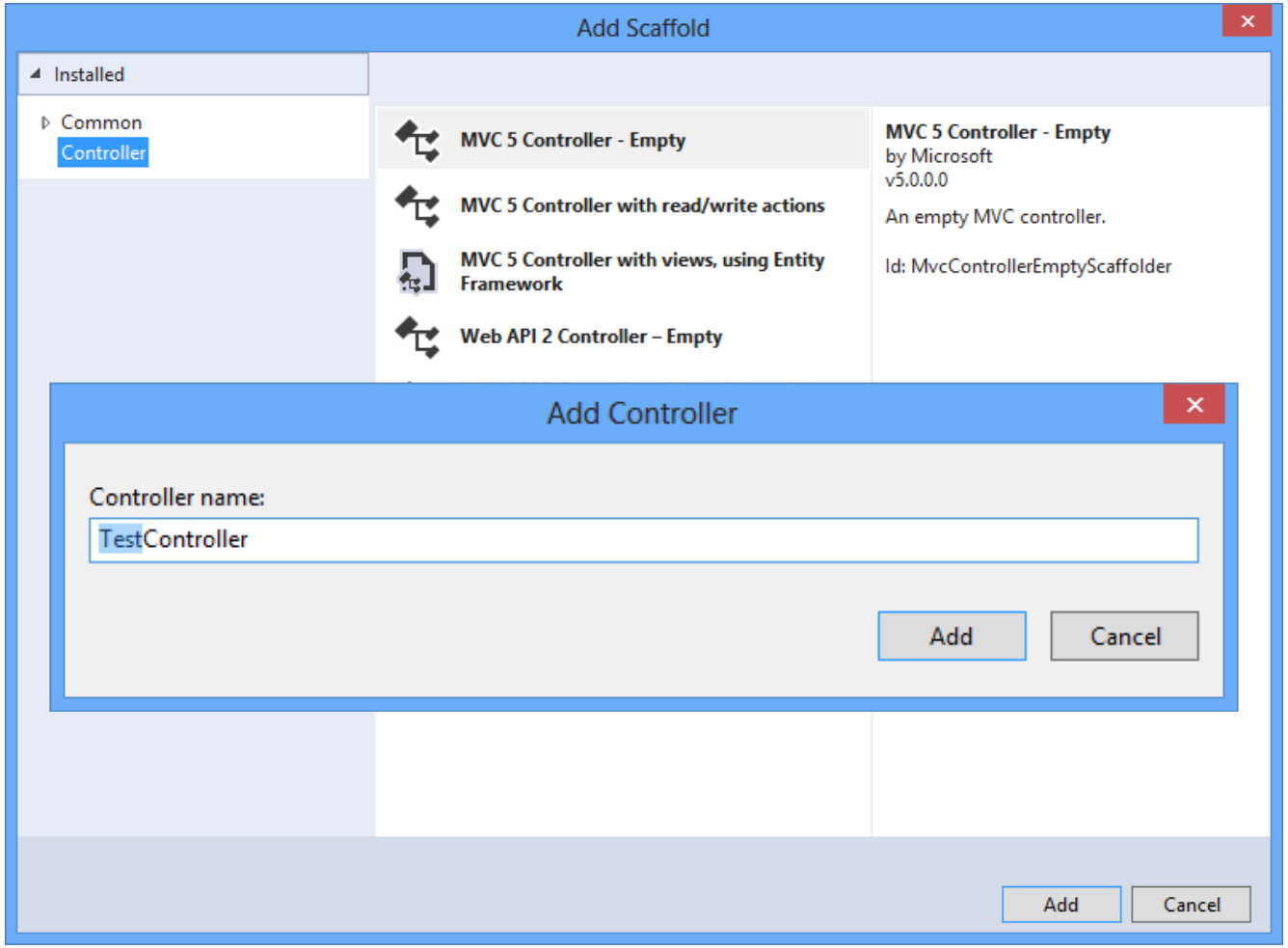
- Trong ASP.Net MVC, Controller chịu trách nhiệm:
  - Chứa các phương thức (Action) sẽ được triệu gọi để đáp ứng các yêu cầu từ client
  - Kiểm soát dữ liệu được gửi kèm cùng yêu cầu từ client trước khi gọi và chạy phương thức xử lý yêu cầu.
  - Nhận dữ liệu được gửi kèm cùng yêu cầu từ client và truyền cho phương thức xử lý yêu cầu như là các tham số.
  - Bắt lỗi và xử lý các ngoại lệ (Exception) do các phương thức phát sinh ra.
  - Lựa chọn các View tương ứng để hiển thị dữ liệu kết quả của phương thức được triệu gọi.

Controller

- Trong ASP.Net MVC lớp cơ sở Controller thuộc namespace System.Web.Mvc là lớp cơ sở của tất cả các Controller trong ứng dụng.
- Các Controller được tạo ra trong ứng dụng kế thừa từ lớp Cotroller cơ sở để thực hiện các thao tác cơ bản đã được chỉ định sẵn của một Controller trong MVC framework
- Để tạo một Controller trong C# cần tạo một lớp kế thừa từ lớp Controller. Việc này có thể được thực hiện:
  - Viết mã lệnh tạo Class thủ công
  - Sử dụng các tiện ích của Visual Studio IDE để tạo Controller theo cấu trúc cây thư mục được định sẵn.

Tạo Controller

- Click chuột phải vào thư mục Controller trong cửa sổ Solution
- Chọn Add -> Controller
- Chọn MVC 5 Controller – Empty
- Click Add để mở hộp thoại Add Controller
- Nhập tên cho Controller
- Click Add thì một file có phần mở rộng .cs với tên vừa đặt sẽ được tạo ra trong thư mục Controllers

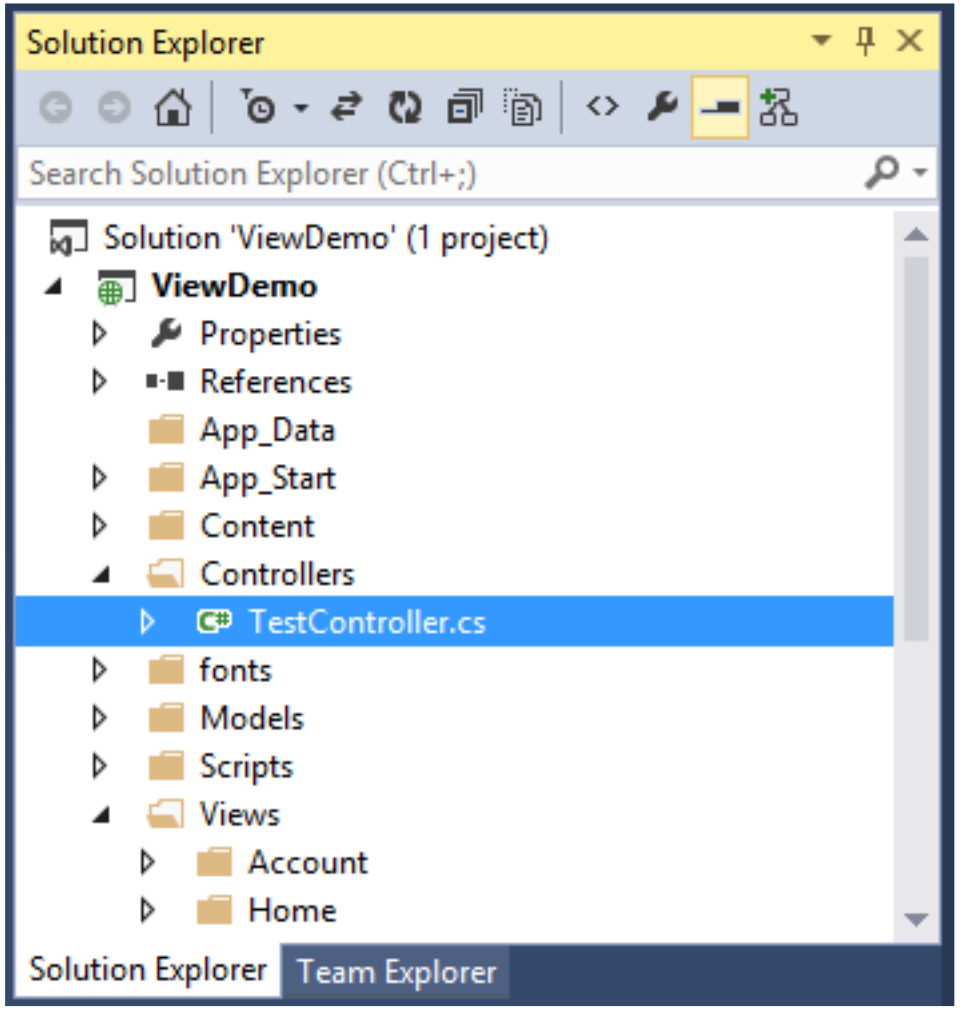


Tạo Controller

- Hình bên là cửa sổ Solution Explorer hiển thị file mã nguồn .cs của controller mới tạo trong thư mục Controllers
- Cũng có thể dùng cú pháp sau để khai báo một Controller mới

```
using System.Web.Mvc;
public class <Controller_Name>Controller:Controller
{
      //Some code
}
```

- Trong đó, Controller_Name là tên của Controller muốn tạo.



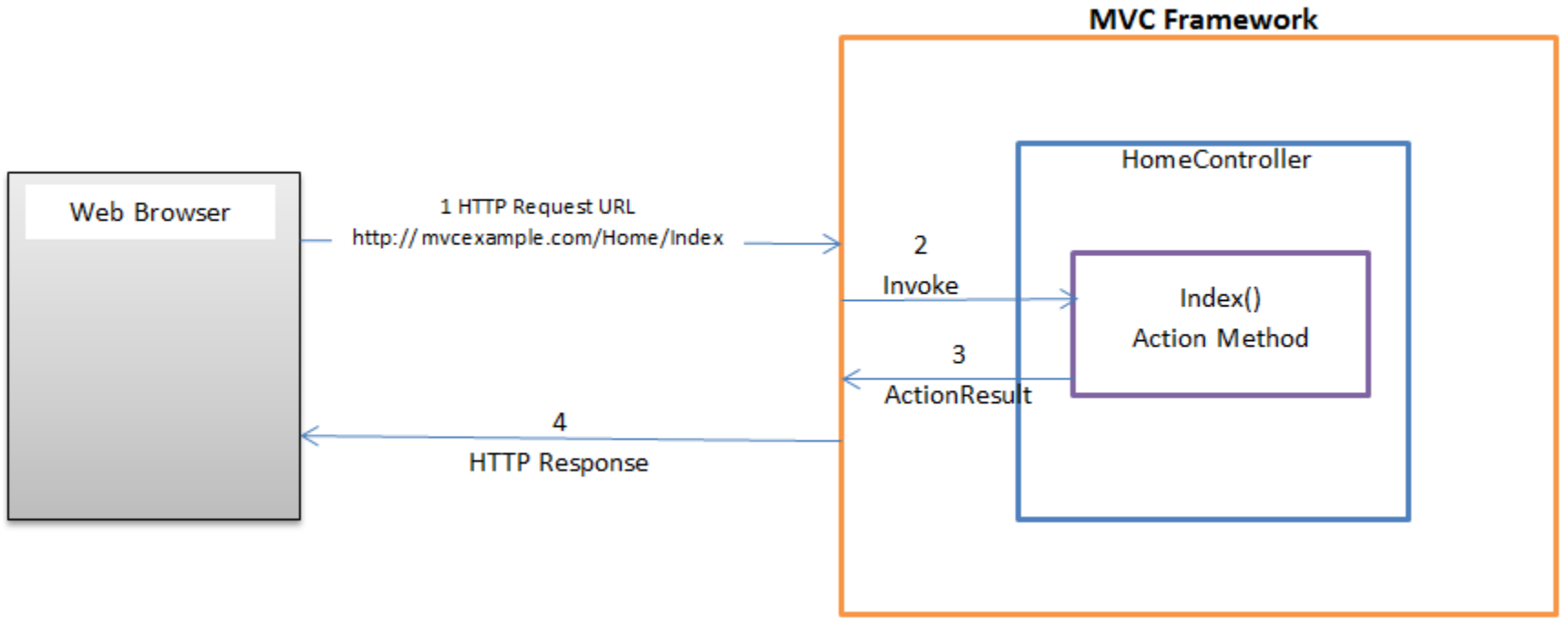
Tạo Controller

- Mã lệnh của TestController mới tạo

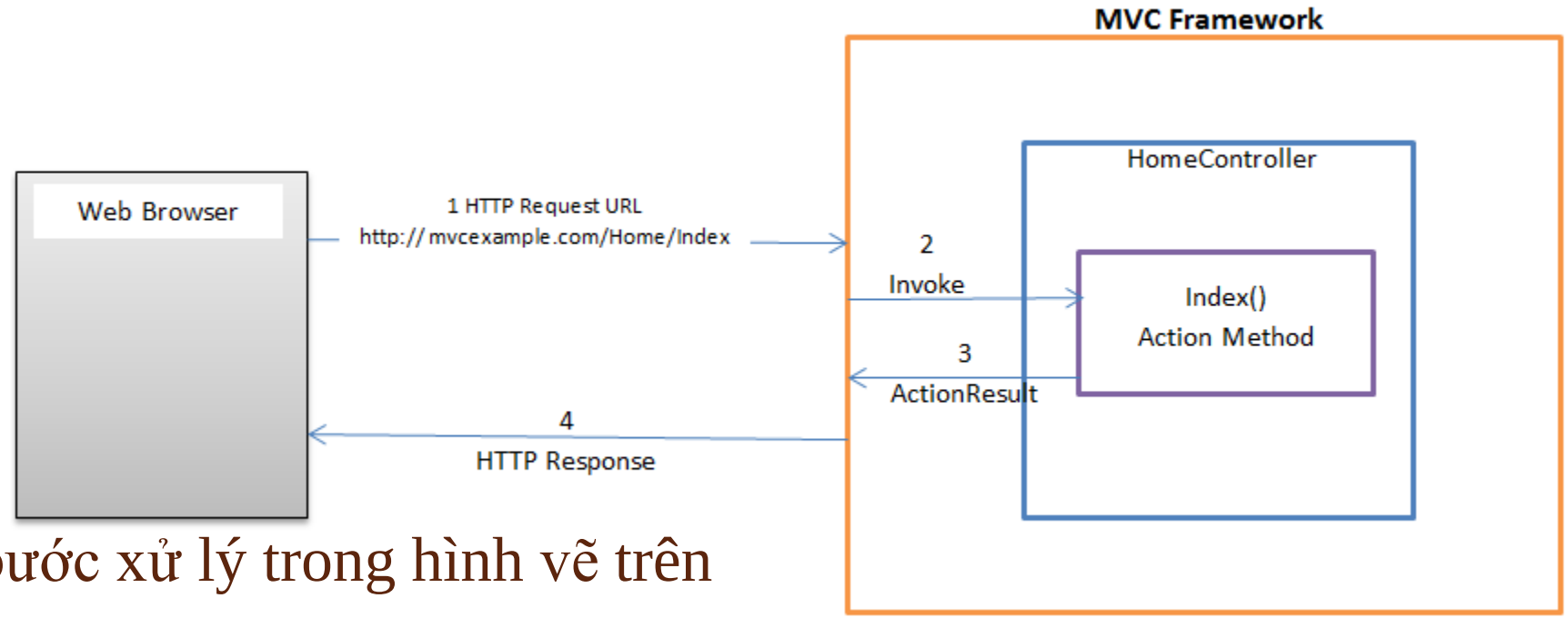
```
using System.Web.Mvc
public class TestController : Controller
{
        //Some code
}
```

Action

- Một Controller có thể chứa một hay nhiều các phương thức hay còn gọi là các Action
- Các Action đảm nhiệm việc xử lý các yêu cầu được gửi tới Controller
- Thông thường, các Action trả về dữ liệu là một đối tượng dẫn xuất từ lớp trừu tượng ActionResult



Action



- Trình tự các bước xử lý trong hình vẽ trên
  1. Trình duyệt gửi yêu cầu Http Request tới máy chủ
  2. MVC Framework triệu gọi Action trong Controller tương ứng với URL của Http Request
  3. Phương thức Action được thực thi và trả về kết quả là ActionResult
  4. MVC Framework chuyển ActionResult thành Http Response trả về trình duyệt

Action

- Một số chú ý khi thêm phương thức Action:
  - Phải được định nghĩa là public
  - Không được định nghĩa là static
  - Không được nạp chồng (Overloading) theo tham số
- Cú pháp tạo phương thức Action trong Controller

```
public ActionResult <ActionMethod_Name>()
{
/*Code to execute logic and return the result as
ActionResult*/
}
```

- Trong đó, ActionMethod_Name là tên Action muốn tạo.

Action

- Đoạn mã lệnh sau tạo hai Action tên là Index và About trong HomeController

```
using System.Web.Mvc;
public class HomeController : Controller
{
    public ActionResult Index()
    {
        /*Code to execute logic and return the result as
        ActionResult*/
    }
    public ActionResult About()
    {
        /*Code to execute logic and return the result as
        ActionResult*/
    }
}
```

- Hai Action đều được định nghĩa public và trả về ActionResult

Action

- Đa số các Action trả về ActionResult nhưng cũng có thể giá trị trả về của Action là các kiểu khác như: String, int, bool…

```
using System.Web.Mvc;
public class HomeController : Controller
{
    public bool IsValid()
    {
        /*Code to execute logic and return a bool*/
    }
    public String Contact()
    {
        /*Code to execute logic and return a String*/
    }
}
```

- Trên đây là phần định nghĩa hai phương thức action tên là IsValid trả về giá trị bool, Contact trả về giá trị String

Action

- ActionResult:
  - Là lớp trừu tượng cho các loại kết quả khác nhau trả về từ Action
  - Các loại ActionResult là các file chứa mã HTML kết hợp với các mã kịch bản Client-side hoặc Server-side để phản hồi đối với yêu cầu của người dùng.
- Bảng dưới đây trình bày một số lớp kế thừa từ ActionResult, là các loại kết quả trả về khác nhau từ phương thức action

Class	Giải thích
View	Trả về một View là một tài liệu HTML
PartialView	Trả về một PartialView là một phần của một View chính
Empty	Trả về một phản hồi rỗng
Redirect	Điều hướng phản hồi sang một Action khác
Json	Trả về đối tượng JSON (JavaScript Object Notation). JSON là một dạng chuẩn mở để lưu trữ và trao đổi thông tin dạng văn bản.

Action

- Bảng dưới đây trình bày một số lớp kế thừa từ ActionResult, là các loại kết quả trả về khác nhau từ phương thức action

Class	Giải thích
JavaScript	Trả về JavaScript sẽ thực thi phía trình duyệt
Content	Trả về Content dựa vào kiểu Content đã định nghĩa ví dụ như XML
FileContent	Trả về nội dung nhị phân
FileStream	Trả về nội dung của file thông qua một đối tượng Stream
FilePath	Trả về một file

Triệu gọi phương thức action

- Trong một ứng dụng ASP.Net MVC, có thể tạo nhiều phương thức action trong một Controller.
- Triệu gọi phương thức action trong Controller bằng cách chỉ rõ URL cho trên trình duyệt bao gồm tên Controller và tên Action muốn gọi.
- URL sau chỉ rõ cách gọi một Action.

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

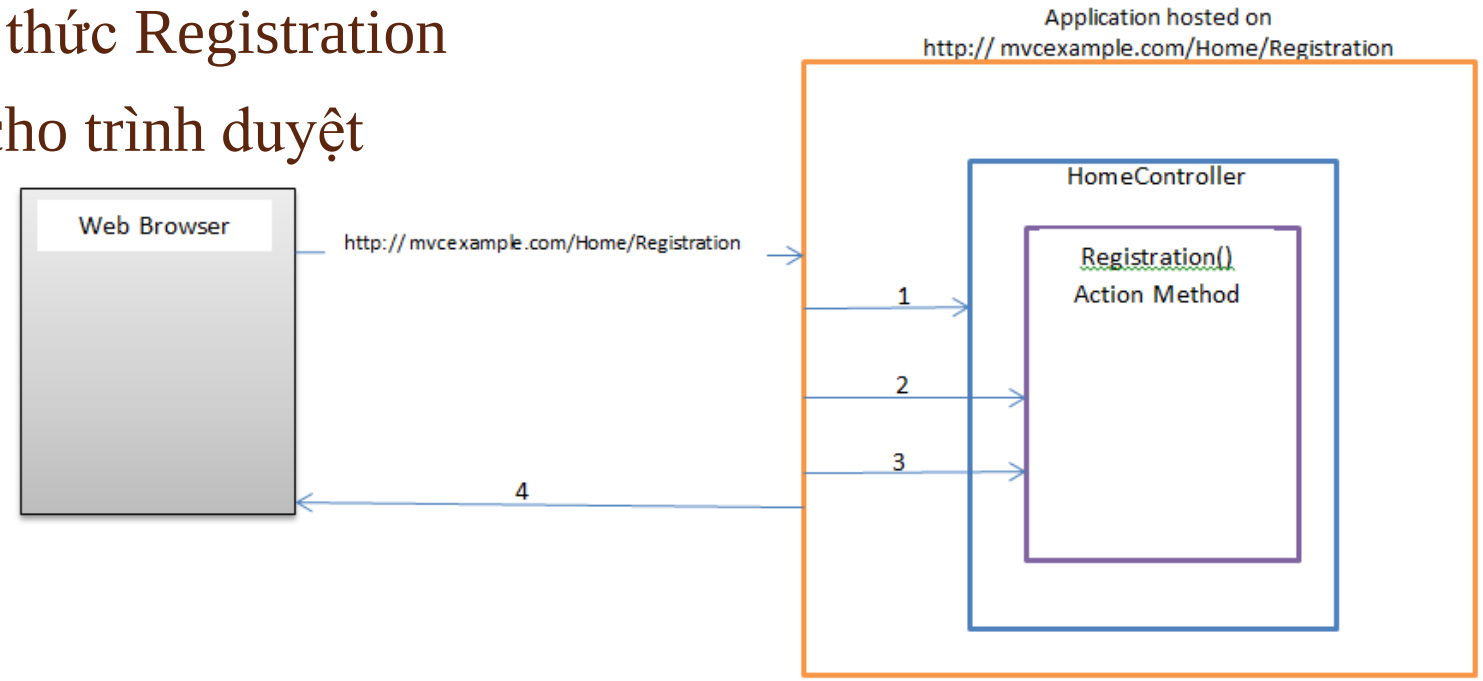
- Trong đó,
  - <domain_name> là tên miền của ứng dụng
  - <controller_name> là tên của Controller (không bao gồm phần đuôi "Controller")
  - <actionmethod_name> là tên của Action muốn gọi

Triệu gọi phương thức action

- Xét URL sau

http:// mvcexample.com/Home/Registration

- Khi URL được gửi cho ứng dụng thông qua trình duyệt web, MVC Framework thực hiện các bước sau:
  - Tìm kiếm lớp có tên HomeController
  - Tìm đến phương thức Registration bên trong lớp HomeController
  - Thực thi phương thức Registration
  - Trả về phản hồi cho trình duyệt



Truyền tham số

- Qua URL, ngoài tên trang web người dùng có thể gửi lên máy chủ các thông tin khác
- Xét URL

```
http://www.mvcexample.com/student/details?Id=006
```

  - URL này sẽ gọi phương thức Detail trong lớp StudentController
  - Ngoài ra URL còn chứa tham số Id với giá trị là 006
- Theo logic, phương thức Detail sẽ tiếp nhận tham số Id kiểu String để trả về bản ghi thông tin Sinh viên có Id tương ứng
- Đoạn mã dưới đây là phương thức action Detail tiếp nhận tham số Id kiểu String

```
public ActionResult Details(string Id)
{
   /*Return student records based on the Id
   parameter as an ActionResultobject*/
}
```

Điều hướng/Định tuyến (Routing)

- MVC Framework hỗ trợ cơ chế Routing cho phép định dạng URL và điều hướng chính xác URL đó với phần mã lệnh tương ứng
- Trong ứng dụng ASP.Net MVC, phần Định tuyến (Routing) cho phép:
  - Chỉ định ứng dụng sẽ xử lý và phản hồi như thế nào đối với các yêu cầu Http
  - Điều hướng chính xác yêu cầu Http đó tới Controller và Action tương ứng
- Hai chức năng chính của Routing:
  - Định nghĩa cấu trúc URL tương ứng với từng Action trong Controller
  - Điều hướng yêu cầu của người dùng tới chính xác Action trong Controller tương ứng khi nhận được URL gửi lên từ phía trình duyệt

Điều hướng/Định tuyến (Routing)

- Routing được thực hiện bằng cách
  - Tạo mẫu định tuyến (route)
  - Đăng ký mẫu định tuyến với bảng định tuyến trong MVC Framework
- Trong một ứng dụng ASP.Net MVC có thể có nhiều mẫu định tuyến trong bảng định tuyến
- Khi tạo ứng dụng ASP.Net MVC trong Visual Studio IDE, một mẫu định tuyến ngầm định được tự động tạo ra trong file RouteConfig.cs

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

- Ở trên, routes là đối tượng thuộc lớp System.Web.Routing.RouteCollection biểu diễn một tập các mẫu định tuyến trong ứng dụng ASP.Net MVC
- MapRoute là phương thức chỉ định rằng:
  - Tên mẫu định tuyến là: Default
  - Định dạng URL tương ứng là: <Tên_Contrller>/<Tên_Action>/<Id_Nếu_có>
  - Mẫu định tuyến ngầm định Controller tên là Home, Action tên là Index với Id nếu có
- Mẫu định tuyến ngầm định được áp dụng khi URL được gửi lên bởi trình duyệt không đúng với định dạng đã được định nghĩa, ví dụ như:
  - Người dùng nhập URL không đúng tên Controller hoặc Action
  - Người dùng chỉ nhập URL chỉ gồm phần <Tên_Miền> của ứng dụng

Đăng ký Định tuyến ngầm định

- Trong một ứng dụng ASP.Net MVC có một file tên là Global.asax
  - Khởi tạo ứng dụng với các chức năng cơ bản được định nghĩa bởi MVC Framework
  - Chứa lớp MVCApplication với phương thức Application_Start() nơi đăng ký bảng định tuyến ngầm định sẽ được áp dụng khi có URL được gửi lên bởi trình duyệt.

```
using System.Collections.Generic;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
namespaceUrlsAndRoutes {
    public class MvcApplication : System.Web.HttpApplication {
        protected void Application_Start() {
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            /*Code for registering other MVC components*/
        }
    }
}
```

URL Pattern

- URL Pattern
  - Khi định nghĩa mẫu định tuyến cần khai báo URL Pattern
  - Được bộ định tuyến so sánh với URL mà người dùng gửi lên bởi trình duyệt
  - Bao gồm nhiều phần phân định nhau bởi dấu "/". Ví dụ `"{controller}/{action}/{id}"`
- URL sau sẽ phù hợp với mẫu URL được định nghĩa ở trên

  `http://www.mvcexample.com/student/records/36`
- Khi bộ định tuyến so khớp URL ở trên với mẫu URL được định nghĩa trong bảng định tuyến bằng cách:
  - Gắn student là giá trị của phần {controller}
  - Gắn records là giá trị của phần {action}
  - Gắn 36 là giá trị của {id}

URL Pattern

- URL Pattern cũng có thể chứa thêm các thành phần phức hợp khác ví dụ:

```
“student/{action}/{id}“
```

- Một số URL khớp với mẫu URL nêu trên:

```
http://www.mvcexample.com/student/records/36
http://www.mvcexample.com/student/delete/21
http://www.mvcexample.com/student/view/23
```

Sắp xếp Bảng định tuyến

- Trong một ứng dụng ASP.Net MVC có thể có nhiều mẫu định tuyến được đăng ký trong bảng định tuyến
- Trong trường hợp này ta cần cấu hình thứ tự mà các mẫu định tuyến sẽ được so khớp bởi bộ định tuyến.
- Bộ định tuyến sẽ thực hiện so khớp URL gửi lên từ trình duyệt với lần lượt từng mẫu định tuyến trong bảng bắt đầu từ mẫu đầu tiên.
- Khi một mẫu định tuyến trong bảng khớp với URL gửi lên thì công việc so khớp sẽ dừng

Sắp xếp Bảng định tuyến

- Đây là ví dụ bảng định tuyến phức hợp

```
routes.MapRoute(
    name: "general",
    url: "{controller}/{action}",
    defaults: new { controller = "Home", action = "Index"}
);
routes.MapRoute(
    name: "manager",
    url: "Manager/{action}",
    defaults: new { controller = "Manager", action = "Browse"}
);
```

- Bảng định tuyến này có 2 mẫu định tuyến trong đó ngầm định sẽ gắn tên Controller là Home, tên Action là Index
- Mẫu định tuyến thứ hai bắt buộc URL phải có chữ "Manager" và gắn tên Controller là "Manager", tên Action là Browse