

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ PHÁT TRIỂN WEBSITE
(LẬP TRÌNH WEBSITE VỚI ASP.NET MVC 5)

CHƯƠNG 4:

ĐIỀU KHIỂN DỮ LIỆU - CONTROLLER



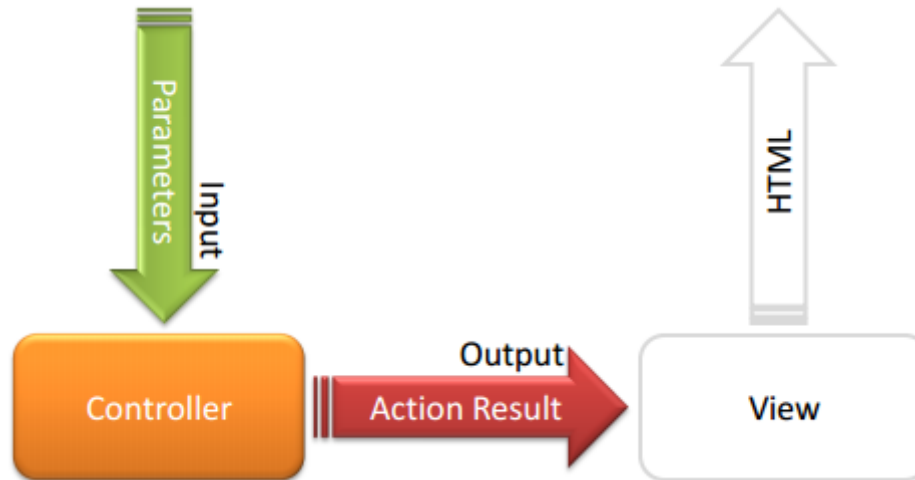
Giảng Viên: ThS. Dương Thành Phết

Email: phetcm@gmail.com

**Website: phetcm@gmail.com -
www.thayphet.net**

Tel: 091815867

GIỚI THIỆU



- ✓ Input: Tham số ?, Nhận tham số ?
- ✓ Output: Lựa chọn View ?

MỤC TIÊU

Kiến thức:

- Trình bài được các về tham số và cách truyền và tiếp nhận tham số của Controller như: *Request, FormCollection, Đối số Action, Model*
- Trình bày được công dụng của lớp *ActionResult()* trong MVC

Kỹ năng:

- Thực hiện được các ứng dụng truyền tham số như: Máy tính cá nhân, Upload File, Gửi mail, iệc truy cập với LINQ

NỘI DUNG

1. Tham số

2. Tiếp nhận tham số

- Request
- FormCollection
- Đối số Action
- Model

3. Ứng dụng: Upload file; Gửi mail

4. Action Result: Text, View, Action, Url, File,...

5. Action Selectors

6. Action Filters

1. THAM SỐ

Tham số yêu cầu từ người dùng được cung cấp dưới 2 dạng: Form field hoặc Query String

Query String

```
<a href="/Student/Register?Id=SV01&Name=Tuấn&Marks=7">Tuấn</a>  
<a href="/Student/Register/SV02?Name=Phương&Marks=8">Phương</a>
```

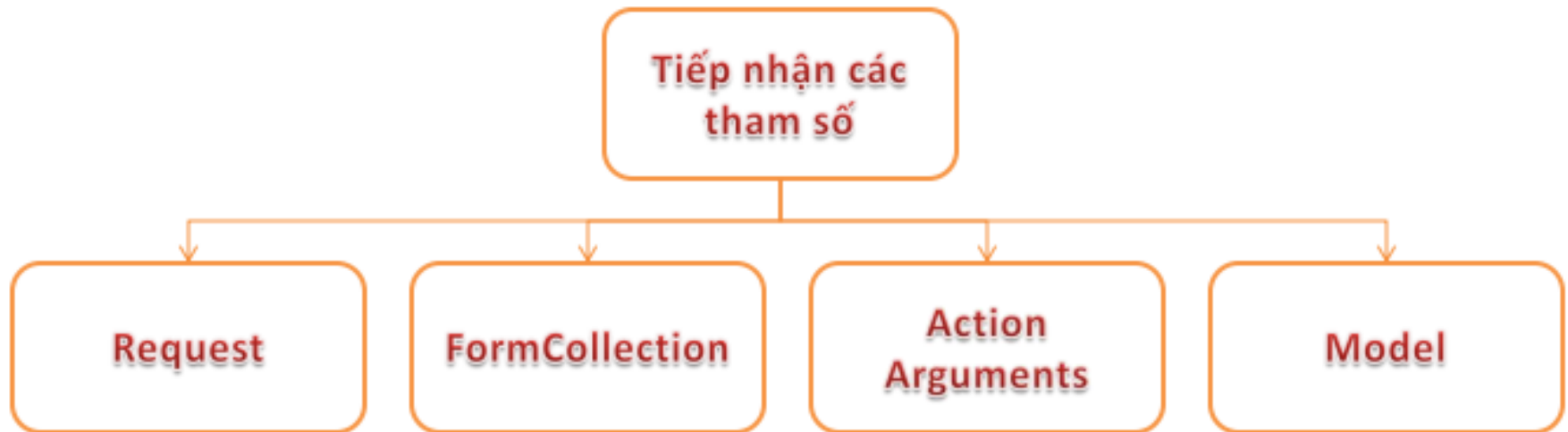
Form field

```
<form action="/Student/Register" method="post">  
  <div>Id</div> <input name="Id" />  
  <div>Name</div> <input name="Name" />  
  <div>Marks</div> <input name="Marks" />  
  <hr />  
  <input type="submit" value="Register" />  
</form>
```

2. TIẾP NHẬN THAM SỐ

Trong MVC có 4 cách để nhận tham số:

- ✓ Sử dụng đối tượng ngầm định Request
- ✓ Sử dụng đối số của Action
- ✓ Sử dụng tham số FormCollection
- ✓ Sử dụng Model



2. TIẾP NHẬN THAM SỐ

2.1. Sử dụng Request

Trong phương thức hành động có thể viết một trong số cách sau đây 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>"];

Ví dụ sau sẽ nhận tham số

```
string Id = Request["Id"];  
string Name = Request["Name"];  
double Marks = Convert.ToDouble(Request["Marks"]);
```

2. TIẾP NHẬN THAM SỐ

2.2. Sử dụng FormCollection

- ✓ Tập hợp các tham số form vào đối số FormCollection của Action. Chỉ nhận được các trường form.
- ✓ Ví dụ nhận tham số form có tên txtName

```
public ActionResult UseFormCollection(FormCollection Fields)
{
    string Id = Fields["Id"];
    string Name = Fields["Name"];
    double Marks = Convert.ToDouble(Fields["Marks"]);
    return View();
}
```

Cách lấy tương đương với Request.Form

```
var value = Request.Form["Name"];
```

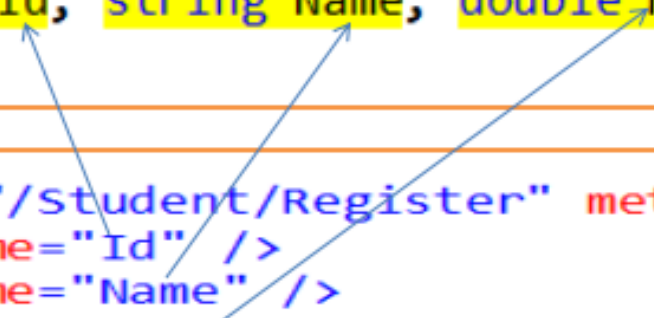

2. TIẾP NHẬN THAM SỐ

2.3. Sử dụng đối số action

- ✓ Định nghĩa tham số cho Action để nhận tham số cùng tên.
- ✓ Ví dụ nhận 2 tham số txtUserName và txtPassword:

```
public ActionResult UseArgument  
    (string Id, string Name, double Marks=0){...}
```

```
<form action="/Student/Register" method="post">  
    <input name="Id" />  
    <input name="Name" />  
    <input name="Marks" />  
    <input type="submit" value="Register" />  
</form>
```


A diagram with three blue arrows pointing from the HTML input names to the C# parameters. One arrow points from 'Id' in the HTML to 'string Id' in the C# code. Another arrow points from 'Name' in the HTML to 'string Name' in the C# code. A third arrow points from 'Marks' in the HTML to 'double Marks=0' in the C# code.

2. TIẾP NHẬN THAM SỐ

2.4. Sử dụng model

- ✓ Tạo lớp Model chứa thuộc tính cùng tên với tham số.
- ✓ Sử dụng lớp này làm đối số cho Action để nhận tham số cùng tên với thuộc tính.

```
public class StudentInfo
{
    public string Id { get; set; }
    public string Name { get; set; }
    public double Marks { get; set; }
}
```

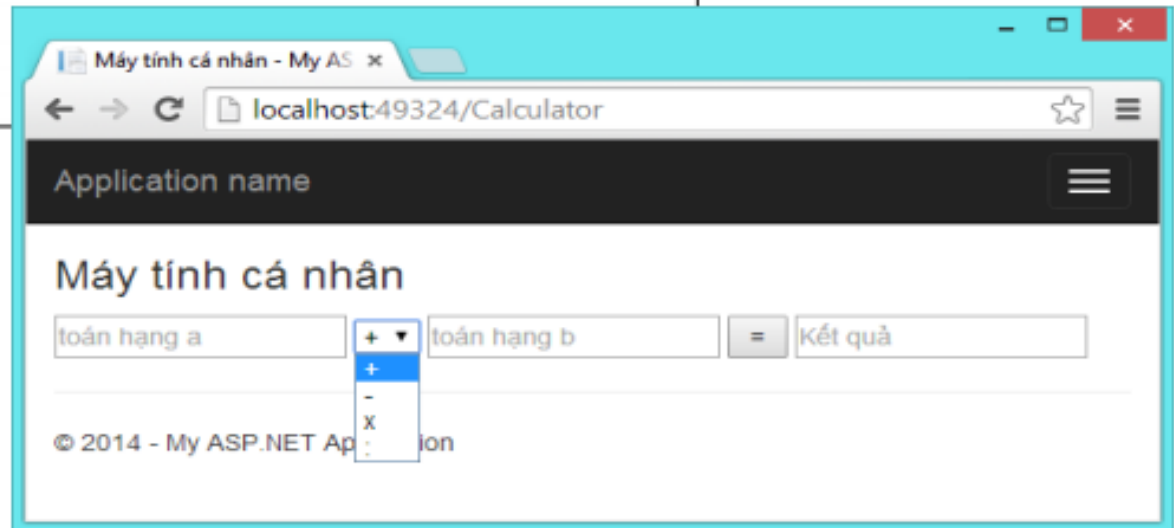


```
public ActionResult UseModel(StudentInfo model){...}
```

3. ỨNG DỤNG

3.1. Máy tính cá nhân

```
public ActionResult Calculate(double a = 0, double b = 0, char o = '+')
{
    switch (o)
    {
        case '+':
            ViewBag.KetQua = a + b;
            break;
        case '-':
            ViewBag.KetQua = a - b;
            break;
        case 'x':
            ViewBag.KetQua = a * b;
            break;
        case '/':
            ViewBag.KetQua = a / b;
            break;
    }
    return View("Index");
}
```



3. ỨNG DỤNG

3.2. Đọc/Ghi file văn bản

Độc/ghi x

← → ↻ localhost ☆ ≡

Application name ≡

Độc/ghi file

Mã sinh viên

Họ và tên

Điểm trung bình

Lưu Mở

Đã ghi vào file !

Độc/ghi x

← → ↻ localhost ☆ ≡

Application name ≡

Độc/ghi file

Mã sinh viên

Họ và tên

Điểm trung bình

Lưu Mở

Đã đọc từ file !

3. ỨNG DỤNG

3.2. Đọc/Ghi file văn bản (tt)

```
public ActionResult Save(String Id, String Name, double Marks)
{
    String path = Server.MapPath("~/StudentInfo.txt");
    String[] lines = {Id, Name, Marks.ToString() };
    System.IO.File.WriteAllLines(path, lines);

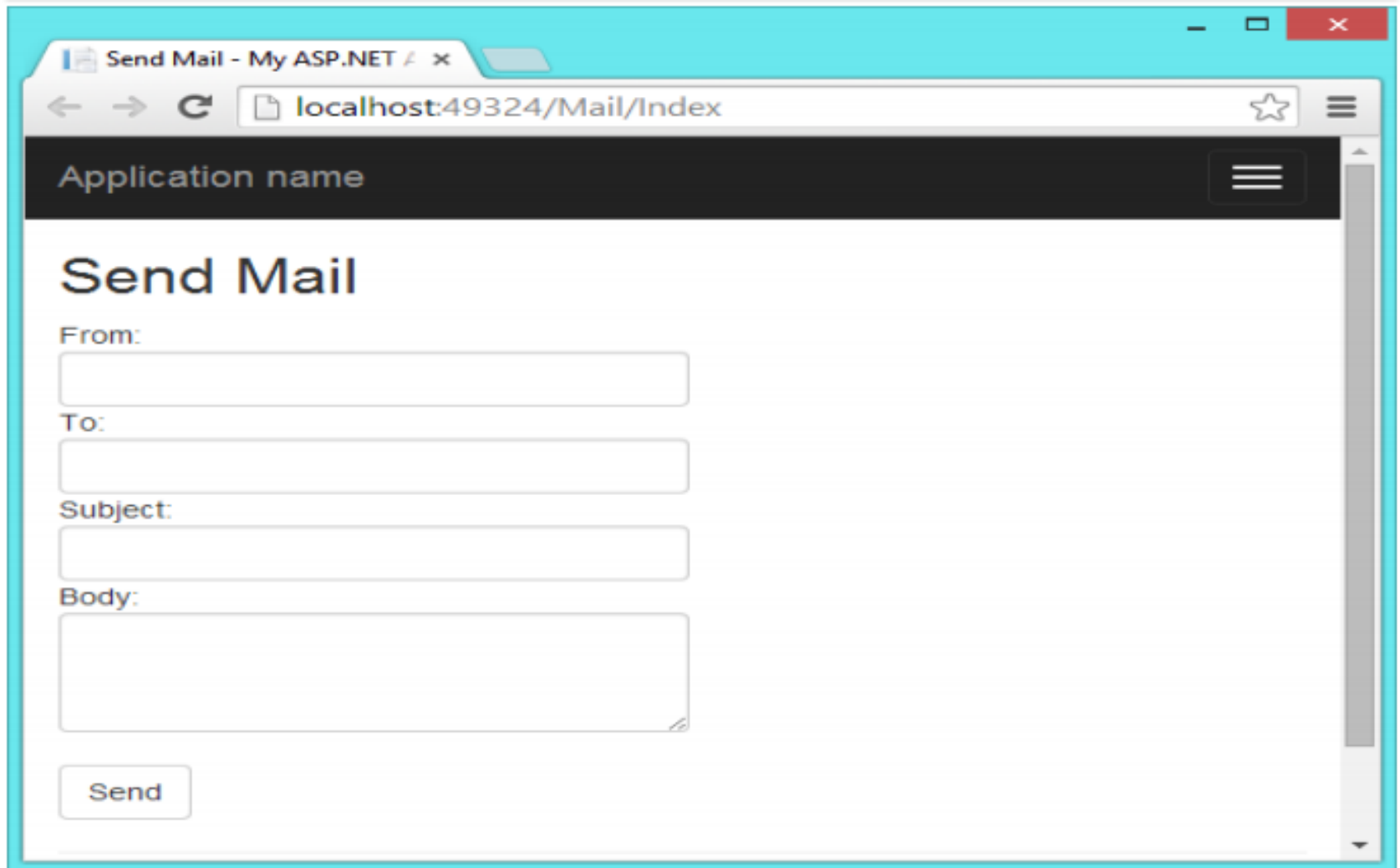
    return View("Index");
}
```

```
public ActionResult Open()
{
    String path = Server.MapPath("~/StudentInfo.txt");
    String[] lines = System.IO.File.ReadAllLines(path);
    ViewBag.Id = lines[0];
    ViewBag.Name = lines[1];
    ViewBag.Marks = Convert.ToDouble(lines[2]);

    return View("Index");
}
```

3. ỨNG DỤNG

3.3. Send Mail



The screenshot shows a web browser window with the title 'Send Mail - My ASP.NET / x'. The address bar displays 'localhost:49324/Mail/Index'. The page has a dark header with the text 'Application name' and a hamburger menu icon. The main content area is titled 'Send Mail' and contains a form with the following fields:

- From:** A text input field.
- To:** A text input field.
- Subject:** A text input field.
- Body:** A large text area for the email content.

At the bottom left of the form is a 'Send' button.

3. ỨNG DỤNG

3.3. Send Mail (tt)

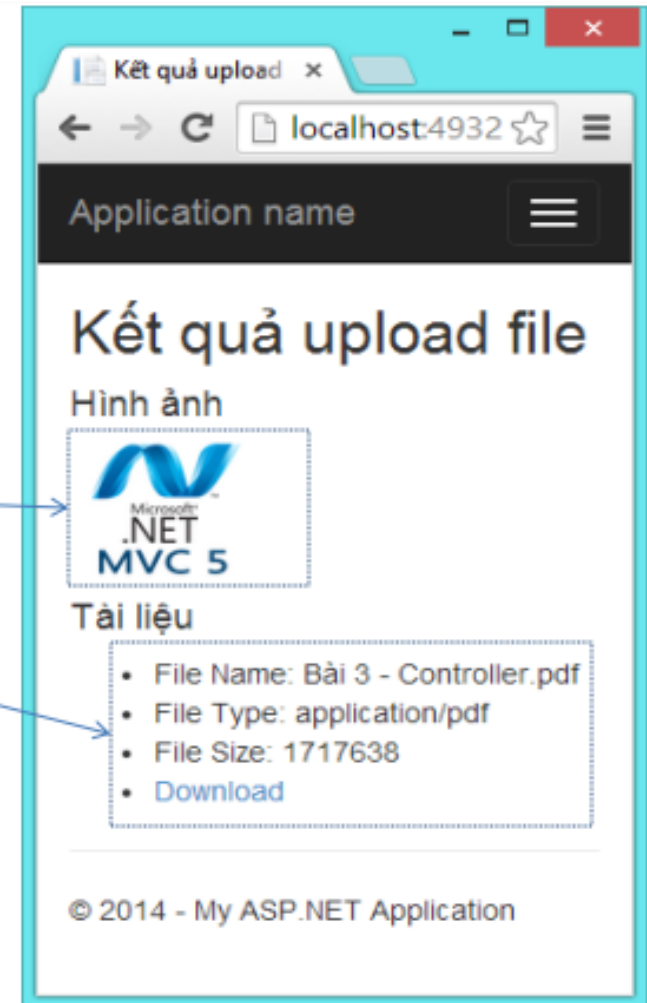
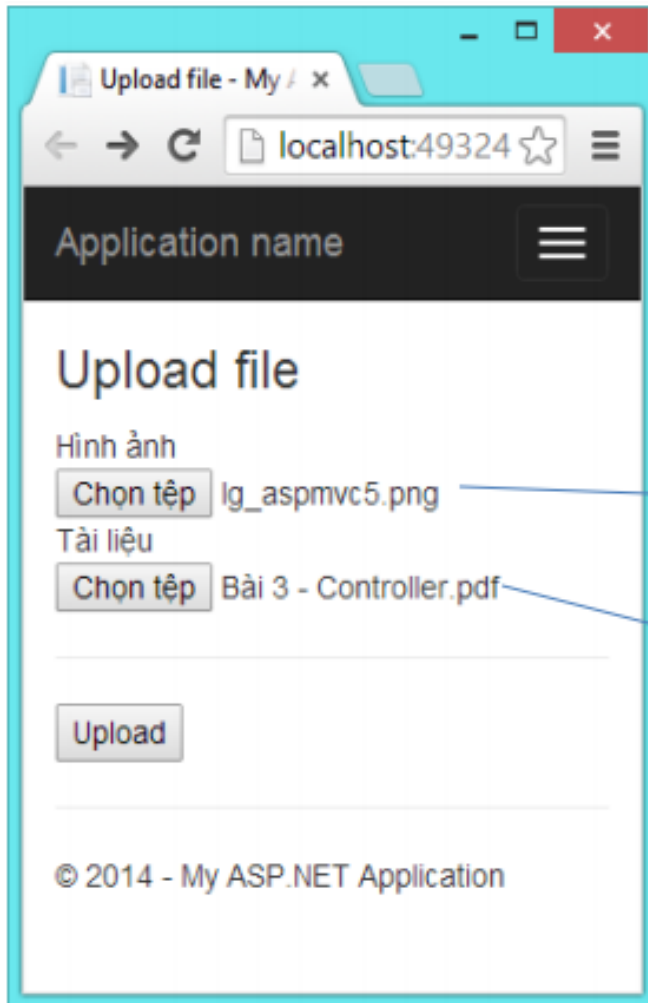
```
public ActionResult Send(MailInfo model)
{
    // Cấu hình thông tin gmail
    var mail = new SmtpClient("smtp.gmail.com", 25)
    {
        Credentials = new NetworkCredential("your@gmail.com", "password"),
        EnableSsl = true
    };

    // Tạo email
    var message = new MailMessage();
    message.From = new MailAddress(model.From);
    message.ReplyToList.Add(model.From);
    message.To.Add(new MailAddress(model.To));
    message.Subject = model.Subject;
    message.Body = model.Body;

    // Gửi mail
    mail.Send(message);
    return View("Index");
}
```

3. ỨNG DỤNG

3.4. Upload File



3. ỨNG DỤNG

3.4. Upload File(tt)

```
<form action="/FileUpload/Upload"
      method="post" enctype="multipart/form-data">
  <div>Hình ảnh</div>
  <input name="image" type="file" />

  <div>Tài liệu</div>
  <input name="document" type="file" />

  <hr />
  <input type="submit" value="Upload" />
</form>
```

```
public ActionResult Upload()
{
    var f = Request.Files["document"];
    if (f != null && f.ContentLength > 0)
    {
        var path = Server.MapPath("~/UploadFiles/" + f.FileName);
        f.SaveAs(path);

        ViewBag.FileName = f.FileName; // tên file
        ViewBag.FileType = f.ContentType; // Kiểu file
        ViewBag.FileSize = f.ContentLength; // Kích thước file
    }
    ...
}
```

3. ỨNG DỤNG

3.4. Upload File (tt)

Giới hạn kích thước file

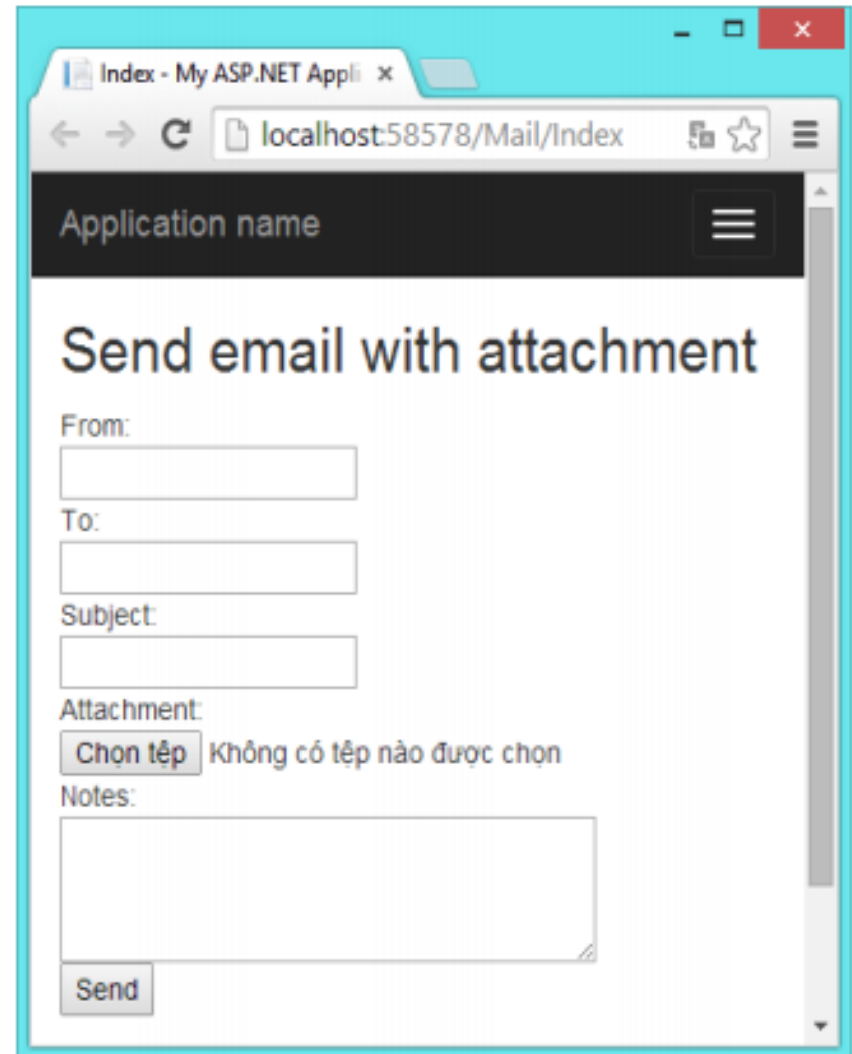
- ✓ Mặc định 2MB
- ✓ Cấu hình giới hạn

```
<system.web>  
  <compilation debug="true" targetFramework="4.5" />  
  <!--20MB-->  
  <httpRuntime targetFramework="4.5" maxLength="20480" />  
</system.web>
```

3. ỨNG DỤNG

Bài tập 1

Kết hợp gửi email và upload file để xây dựng trang web gửi email có attach file



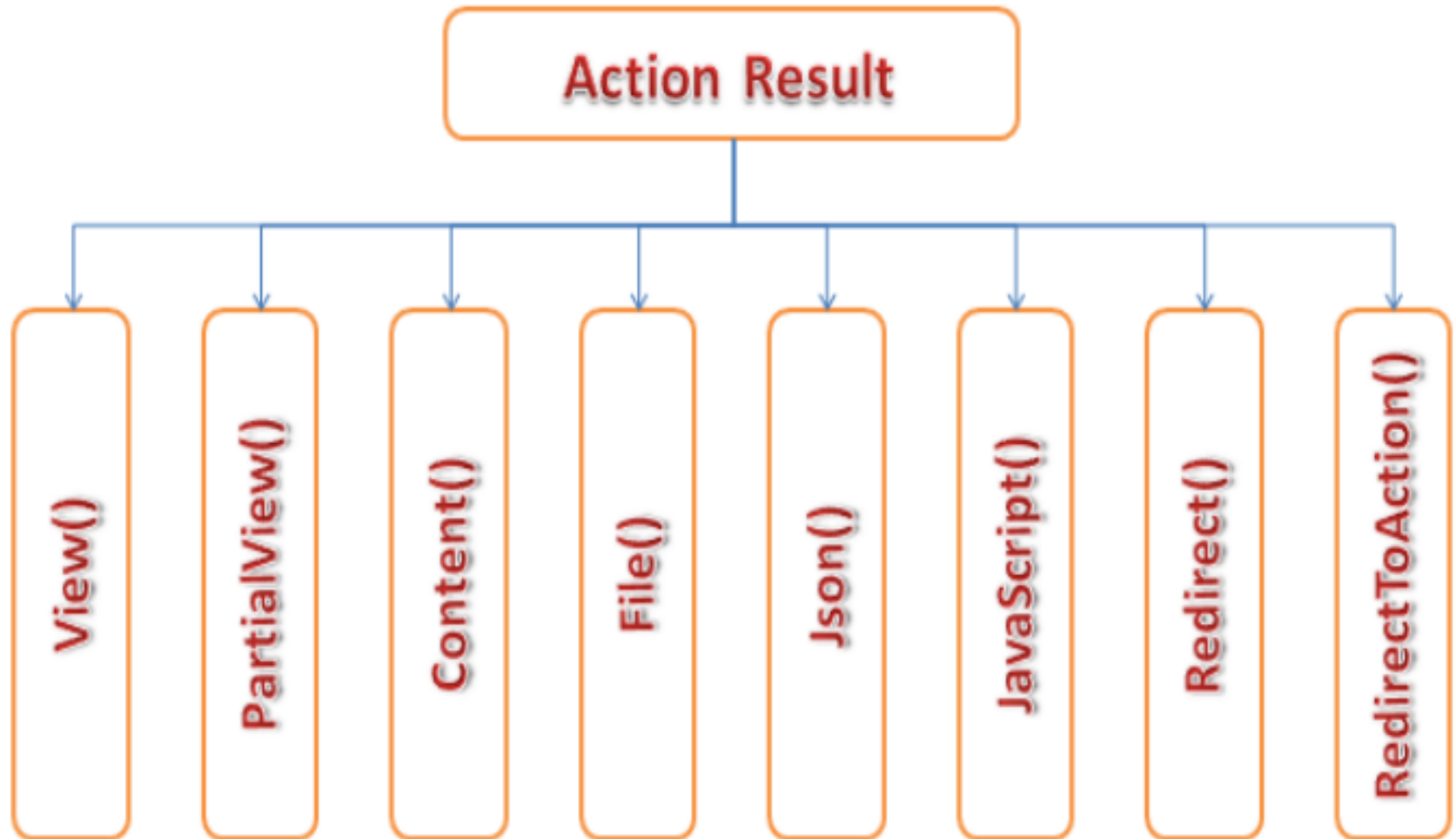
The screenshot shows a web browser window with the address bar displaying 'localhost:58578/Mail/Index'. The page title is 'Index - My ASP.NET Appli'. The main content area has a dark header with 'Application name' and a hamburger menu icon. Below the header, the title 'Send email with attachment' is displayed. The form includes fields for 'From:', 'To:', and 'Subject:'. The 'Attachment:' section has a 'Chọn tệp' button and the text 'Không có tệp nào được chọn'. There is a large text area for 'Notes:' and a 'Send' button at the bottom.

3. ỨNG DỤNG

Bài tập 2: Nhận và lưu dữ liệu vào File

- ✓ Tiếp nhận form thông tin học viên và lưu vào file
 - Mã học viên
 - Họ và tên
 - Giới tính
 - Ngày sinh
 - Học phí
 - Hình
 - Ghi chú
- ✓ Đọc thông tin học viên từ file và xuất ra form
- ✓ Gợi ý:
 - Sử dụng model để nhận tham số yêu cầu
 - Sử dụng `File.WriteAllLines()` để lưu thông tin nhân sự

4. ACTION RESULTS

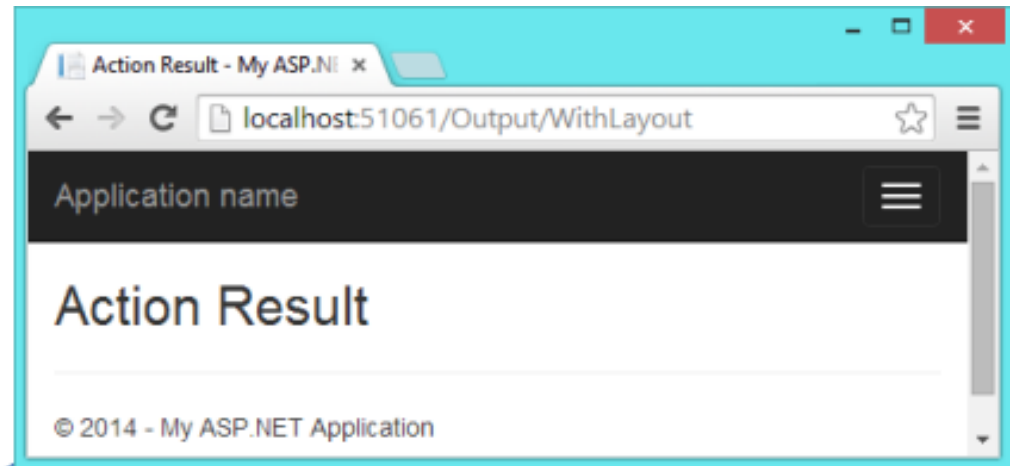


4. ACTION RESULTS

Tên Result	Mô tả	Hàm sử dụng
ContentResult	Trả về chuỗi	Content()
FileContentResult/ FilePathResult/ FileStreamResult	Trả về nội dung file	File()
JavaScriptResult	Trả về nội dung JavaScript	JavaScript()
JsonResult	Trả về dữ liệu dạng Json	Json()
RedirectResult	Chuyển sang URL mới	Redirect()
RedirectToRouteResult	Chuyển sang 1 action hoặc 1 action của controller khác	RedirectToRoute() RedirectToAction()
ViewResult	Chuyển sang View để hiển thị	View()
PartialViewResult	Chuyển sang View để hiển thị không layout	PartialView()

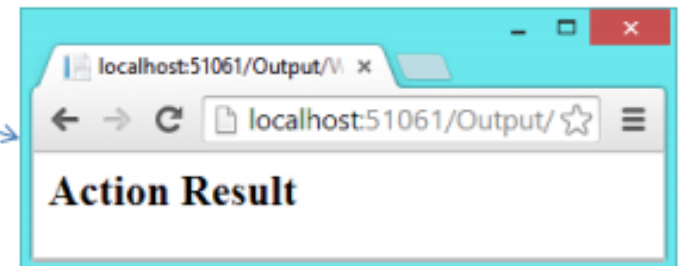
4. ACTION RESULTS

4.1.View() và PartialView()



```
public ActionResult WithLayout()
{
    return View("Index");
}

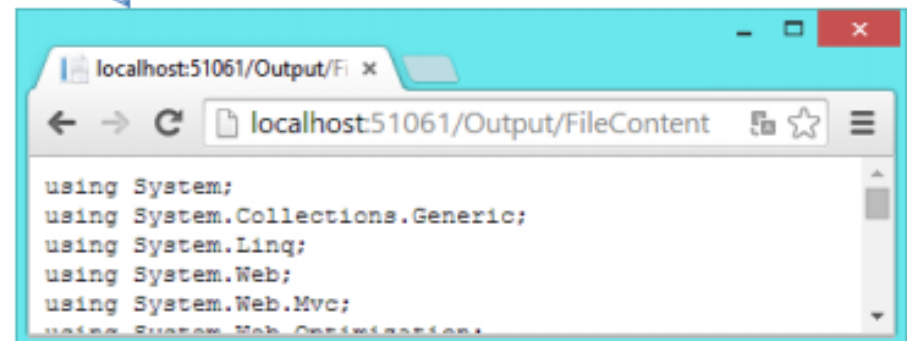
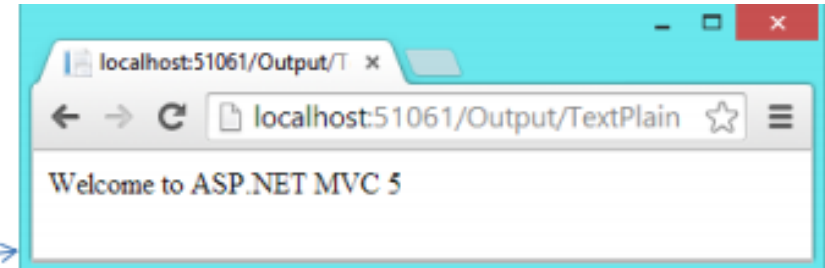
public ActionResult WithoutLayout()
{
    return PartialView("Index");
}
```



4. ACTION RESULTS

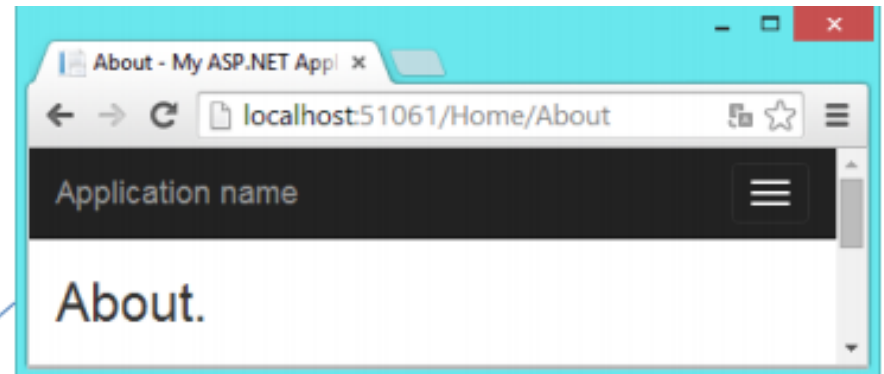
4.2. Content() & File ()

```
public ActionResult TextPlain()  
{  
    return Content("Welcome to ASP.NET MVC 5");  
}  
  
public ActionResult FileContent()  
{  
    return File("~/Global.asax.cs", "text/plain");  
}
```



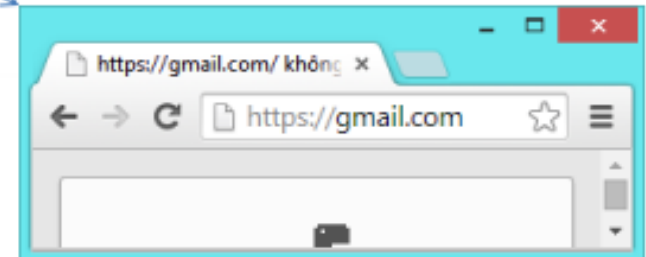
4. ACTION RESULTS

4.3. RedirectToAction() VÀ Redirect()



```
public ActionResult RedirectToAction()  
{  
    return RedirectToAction("About", "Home");  
}
```

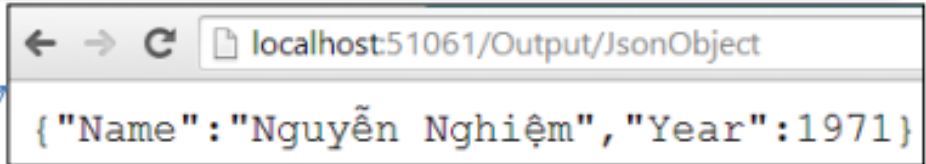
```
public ActionResult RedirectToUrl()  
{  
    return Redirect("http://gmail.com");  
}
```



4. ACTION RESULTS

4.4. JSON()

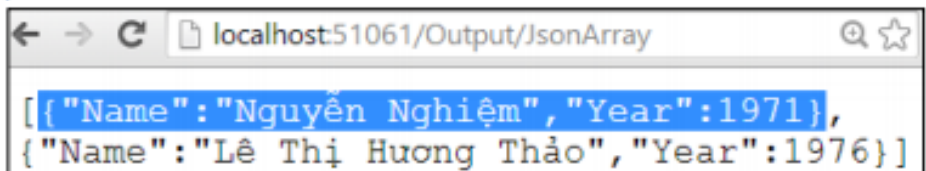
```
public ActionResult JsonObject()  
{  
    var data = new { Name = "Nguyễn Nghiệm", Year = 1971 };  
    return Json(data, JsonRequestBehavior.AllowGet);  
}
```



localhost:51061/Output/JsonObject

```
{"Name": "Nguyễn Nghiệm", "Year": 1971}
```

```
public ActionResult JsonArray()  
{  
    ArrayList data = new ArrayList();  
    data.Add(new { Name = "Nguyễn Nghiệm", Year = 1971 });  
    data.Add(new { Name = "Lê Thị Hương Thảo", Year = 1976 });  
  
    return Json(data, JsonRequestBehavior.AllowGet);  
}
```



localhost:51061/Output/JsonArray

```
[{"Name": "Nguyễn Nghiệm", "Year": 1971}, {"Name": "Lê Thị Hương Thảo", "Year": 1976}]
```

5. ACTION SELECTORS

- ✓ **[HttpPost]**
- ✓ **[HttpGet]**
- ✓ **[ActionName(name)]**
- ✓ **[ChildActionOnly]**

5. ACTION SELECTORS

Một Action sau khi được định nghĩa có thể được triệu gọi theo cả **POST** và **GET**

```
public ActionResult MyAction()
```

- ✓ Nếu muốn chỉ có gọi với **POST** hay **GET** thì đánh dấu Action với **[HttpPost]** hay **[HttpGet]**

```
[HttpGet]
```

```
public ActionResult MyAction()
```

```
[HttpPost]
```

```
public ActionResult MyAction(MyModel model)
```

5. ACTION SELECTORS

- ✓ Đổi tên giao dịch một Action

[ActionName("OtherName")]

public ActionResult MyAction()

- ✓ Chỉ cho phép sử dụng **@Html.Action()** mà không cho phép gọi trực tiếp

[ChildActionOnly]

public ActionResult MyAction()

6. ACTION FILTER

- ✓ Action filter được sử dụng để thực hiện nhiệm vụ xảy ra tại thời điểm thực hiện khác nhau của một action.
- ✓ Một số Action Filter được cung cấp sẵn bởi MVC5
 - [ValidateInput]
 - [Authorize]
 - [ValidateAntiForgeryToken]
 - [HandleError]
- ✓ Ngoài ra chúng ta có thể tự viết Action Filter cho mục đích riêng

[InitializeSimpleMembership]

- ✓ Bộ lọc có thể sử dụng để lọc
 - Một action đơn lẻ
 - Các action trong một Controller
 - Tất cả các action trong dự án

6. ACTION FILTER

ValidateInput

- ✓ Cho phép hay không dữ liệu HTML được gửi đến Action Send.
 - Nếu cho phép: bạn có thể nhập HTML vào form
 - Ngược lại sẽ nhận thông báo lỗi
- ✓ Mã sau cho phép gửi HTML đến Action Send()

```
public class HomeController : Controller
{
    [ValidateInput(false)]
    public ActionResult Send(Mail mail)
    {
```

6. ACTION FILLTER

Authorize

- ✓ Bộ lọc [Authorize] được sử dụng để kiểm tra việc truy xuất các action với thẩm quyền phù hợp.
- ✓ Có 3 dạng thường sử dụng
 - **[Authorize]** Chỉ cho phép truy xuất sau đăng nhập
 - **[Authorize(users="u1,u2...")]** Chỉ cho phép truy xuất với user u1, u2...
 - **[Authorize(roles="r1,r2...")]** Chỉ cho phép truy xuất với các user có vai trò r1, r2...

6. ACTION FILTER

Authorize

❑ Lọc từng action riêng

```
[Authorize(Roles="Administrator")]—  
public ActionResult Manage()  
{  
    return View();  
}
```

Phải đăng nhập với user có vai trò là Administrator trước khi gọi action

```
[Authorize(Users = "nnghiem")]——  
public ActionResult Config()  
{  
    return View();  
}
```

Phải đăng nhập với user name là nnghiem trước khi gọi action

```
[Authorize]——  
public ActionResult MyAccount()  
{  
    return View();  
}
```

Phải đăng nhập trước khi gọi action

6. ACTION FILTER

AllowAnonymous

- ❑ Lọc tất cả các action trong controller

Tất cả mọi Action phải đăng nhập mới được phép truy xuất

[Authorize]

[InitializeSimpleMembership]

```
public class AccountController : Controller  
{
```

```
    public ActionResult ChangePassword()...
```

[AllowAnonymous]

```
    public ActionResult Login(string returnUrl)...
```

Cho phép truy
xuất nặc danh

6. ACTION FILTER

Lọc hết các Action trong dự án

- ❑ B1: bổ sung filter trong App_Start/FilterConfig.cs

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new HandleErrorAttribute());
}
```

- ❑ B2: đăng ký filter trong Global.asax

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();

    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

6. ACTION FILTER

Custom Action Filter

- ✓ Bộ lọc tùy biến được sử dụng để thực hiện giám sát việc thực của Action tại các thời điểm khác nhau theo yêu cầu của riêng mình.
- ✓ Có bốn thời điểm cần được giám sát trong 1 action
 - **OnActionExecuting()**: Trước lúc action chạy
 - **OnActionResulting()**: Trước lúc trả kết quả
 - **OnActionResulted()**: Sau khi đã trả kết quả
 - **OnActionExecuted()**: Sau khi action chạy
- ✓ Sau khi định nghĩa action, bạn có thể sử dụng như các action hỗ trợ sẵn trong MVC5

6. ACTION FILLTER

Định nghĩa Action Filler

```
public class HitCounterAttribute: ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        base.OnActionExecuting(filterContext);
    }

    public override void OnActionExecuted(ActionExecutedContext filterContext)
    {
        base.OnActionExecuted(filterContext);
    }

    public override void OnResultExecuting(ResultExecutingContext filterContext)
    {
        base.OnResultExecuting(filterContext);
    }

    public override void OnResultExecuted(ResultExecutedContext filterContext)
    {
        base.OnResultExecuted(filterContext);
    }
}
```

6. ACTION FILTER

Sử dụng Action Filter

- ❑ Lọc tất cả các Action trong 1 Controller

```
[HitCounter]
public class UploadController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

- ❑ Lọc tất cả các Action trong mọi Controller

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new HandleErrorAttribute());
        filters.Add(new HitCounterAttribute());
    }
}
```

6. ACTION FILLTER

Demo HitCounter

Sử dụng file để lưu số lần truy cập của mỗi trang trong website

Dictionary<String, int>

➔ để lưu danh sách truy cập

HẾT