CÔNG NGHỆ WEB

NGÔN NGỮ KỊCH BẢN PHÍA SERVER
ASP.NET MVC

Biên soạn: Chu Thị Hường – Bộ môn HTTT – Khoa CNTT

NỘI DUNG

- Giới thiệu một số ngôn ngữ kịch bản phía Server
- Ngôn ngữ ASP.NET
- Công nghệ ASP.NET
- Mô hình MVC

MỘT SỐ NGÔN NGỮ KỊCH BẢN PHÍA SERVER

- PHP (Hypertext Preprocessor):
  - PHP là một kịch bản trên phía trình chủ, có các phiên bản trên hệ điều hành window và Linux.
  - Có thể dễ dàng nhúng vào trang HTML.
  - PHP thường kết hợp với hệ quản trị CSDL MySQL phù hợp với các doanh nghiệp vừa và nhỏ.

MỘT SỐ NGÔN NGỮ KỊCH BẢN PHÍA SERVER

- PHP (Hypertext Preprocessor):
- JSP là một kịch bản trên trình chủ với nền tảng dựa trên ngôn ngữ lập trình Java.
- ASP.Net Framework dựa trên nền tảng ngôn ngữ lập trình .NET.

- Classic ASP - Active Server Pages
  - ASP (Active Server Page) được Microsoft giới thiệu vào năm 1996 và trở lên thông dụng từ ngày đó
  - ASP là công nghệ cho phép các kịch bản trong trang web được thực hiện bởi một máy chủ Internet.
  - Các trang có đuôi mở rộng là .asp.

NGÔN NGỮ ASP.NET

- ASP.NET:
  - ASP.NET là một thế hệ mới của ASP. Nó không tương thích với Classis ASP, nhưng ASP.NET có thể bao gồm ASP.
  - Tăng hiệu quả lập trình thông qua các đặc điểm:
    - Dễ dàng lập trình
    - Lựa chọn ngôn ngữ đơn giản
    - Hỗ trợ công cụ tuyệt vời
    - Nhờ nền tảng vững vàng và tài nguyên phong phú của .Net Framework với hơn 5000 class (lớp) bao gồm XML, Data access, File upload…

NGÔN NGỮ ASP.NET

- ASP.NET:
  - Tăng khả năng thực hiện và tính ổn định
  - Dễ dàng triển khai
  - ASP.Net cho phép ta tự động cập nhật các thành phần đã biên dịch mà không cần phải khởi động lại các Web Server.
- ASP.NET Razor
  - Razor là một cú pháp đánh dấu mới và đơn giản để nhúng code server vào các trang web ASP.NET, giống như ASP cổ điển.
  - Razor có khả năng của ASP.NET truyền thống, nhưng là dễ học và sử dụng hơn.

- Web Pages (with Razor syntax):
  - Mô hình lập trình đơn giản nhất, và trang web sẽ là tập hợp nhiều trang web
  - Sử dụng mã server-side ngay bên trong các trang web
  - Không có vòng đời của trang như Web Forms, nó chỉ đơn giản là xử lý mã lệnh từ trên xuống dưới

- Web Forms (traditional ASP.NET):
  - Mô hình lập trình hướng sự kiện
  - Cung cấp nhiều các server side controls, chúng đóng gói HTML, JavaScript và CSS bên trong.
  - Quản lý trạng thái của các control nhờ vào cơ chế post-back và view state
  - Nhược điểm: Cấu trúc của nó khiến nó khó có thể được áp dụng unit-test hoàn toàn và tách biệt lập trình viên khỏi kiến thức về CSS, HTML và mô hình lập trình state-less (không lưu giữ trạng thái của các điều khiển) truyền thống của lập trình web.

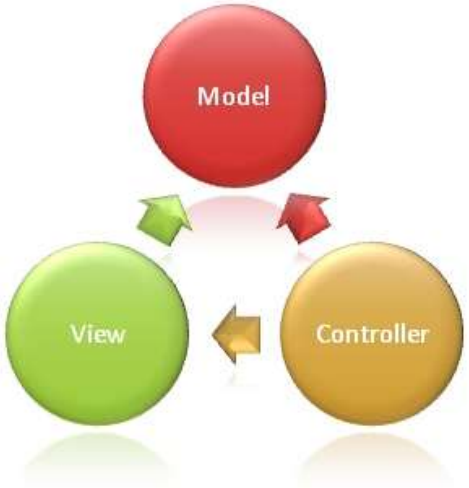
- MVC (Model View Controller):
  - Phát triển ứng dụng theo mô hình test-driven, và áp dụng mẫu lập trình SoC (Separation of Concerns).
  - Không cung cấp các điều khiển được đóng gói sẵn như server-side control của ASP.NET Web Froms, thay vào đó bạn bạn phải hiểu biết sâu hơn về HTML và giao thức HTTP.

ASP.NET SERVER TECHNOLOGIES

- ASP.NET Development Tools:
  - WebMatrix
  - Visual Web Developer
  - Visual Studio
- ASP.NET File Extensions
  - Classic ASP files có đuôi mở rộng .asp
  - ASP.NET files có đuôi mở rộng .aspx
  - ASP.NET files với Razor C# syntax có đuôi mở rộng .cshtml
  - ASP.NET files với Razor VB syntax có đuôi mở rộng .vbhtml

MÔ HÌNH MVC

- ASP.NET MVC là framework cho xây dựng ứng dụng web áp dụng Model-View-Controller pattern.
- MVC chia user interface (UI) của ứng dụng thành 3 khía cạnh chính:
  - Model
  - Controller
  - View

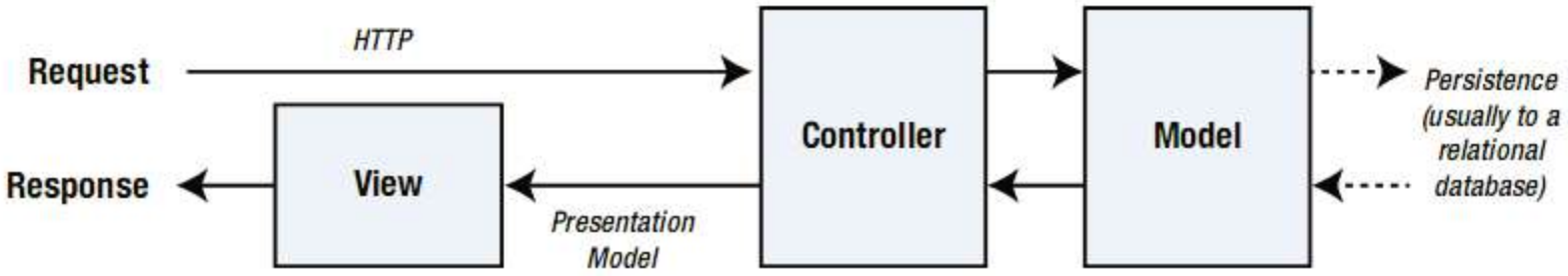


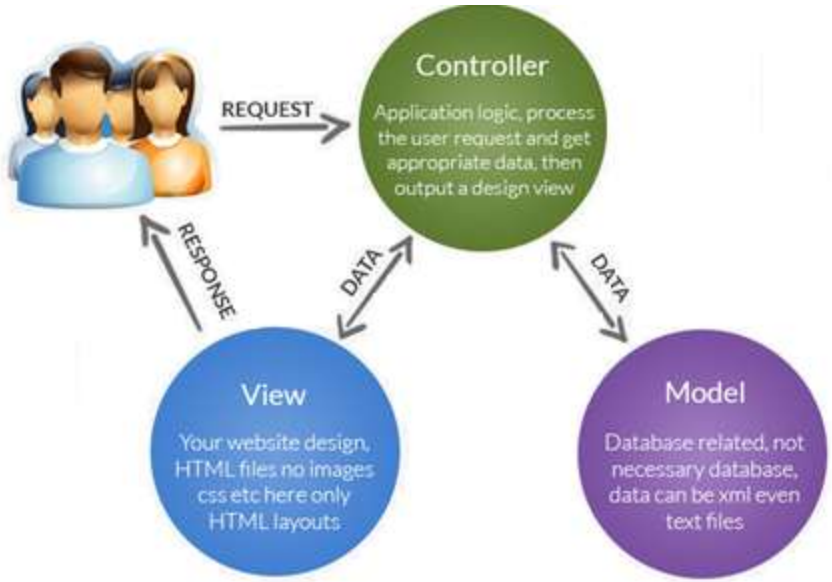
MÔ HÌNH MVC

- ASP.NET MVC
  - The Model: Gồm các classes mô tả dữ liệu cũng như các quy tắc nghiệp vụ cho việc thay đổi và thao tác dữ liệu. Với ASP.NET MVC gần Data Access Layer, có thể dùng tool Entity Framework or Nhibernate.
  - View: Định nghĩa application's UI sẽ được hiển thị như thế nào? Cụ thể trong ASP.NET MVC là các template cho HTML.
  - Controller: Là các class that quản lý mối quan hệ giữa View và Model. Đáp ứng user input, báo cho Model và decides view nào được render.

MÔ HÌNH MVC

- Tương tác trong ứng dụng MVC



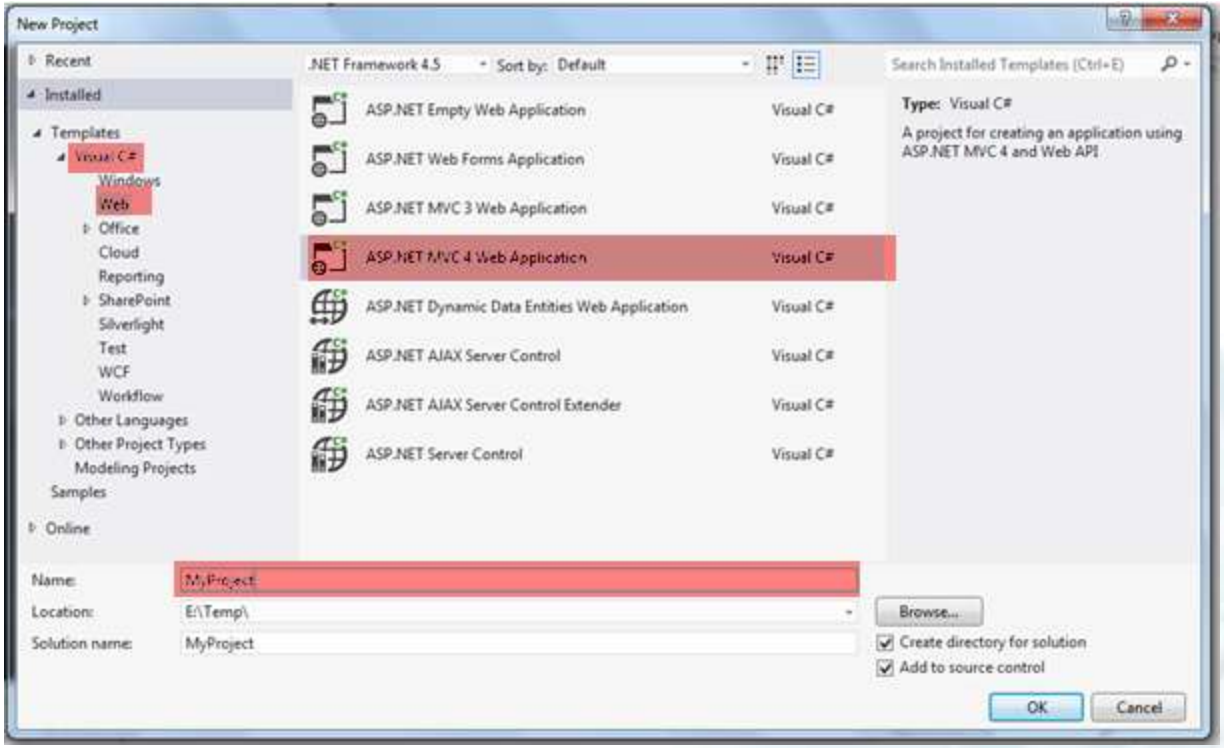


MÔ HÌNH MVC

- ASP.NET MVC
  - The Model: Gồm các classes mô tả dữ liệu cũng như các quy tắc nghiệp vụ cho việc thay đổi và thao tác dữ liệu. Với ASP.NET MVC gần Data Access Layer, có thể dùng tool Entity Framework.
  - View: Định nghĩa application's UI sẽ được hiển thị như thế nào? Cụ thể trong ASP.NET MVC là các template cho HTML.
  - Controller: Là các class that quản lý mối quan hệ giữa View và Model. Đáp ứng user input, báo cho Model và decides view nào được render.

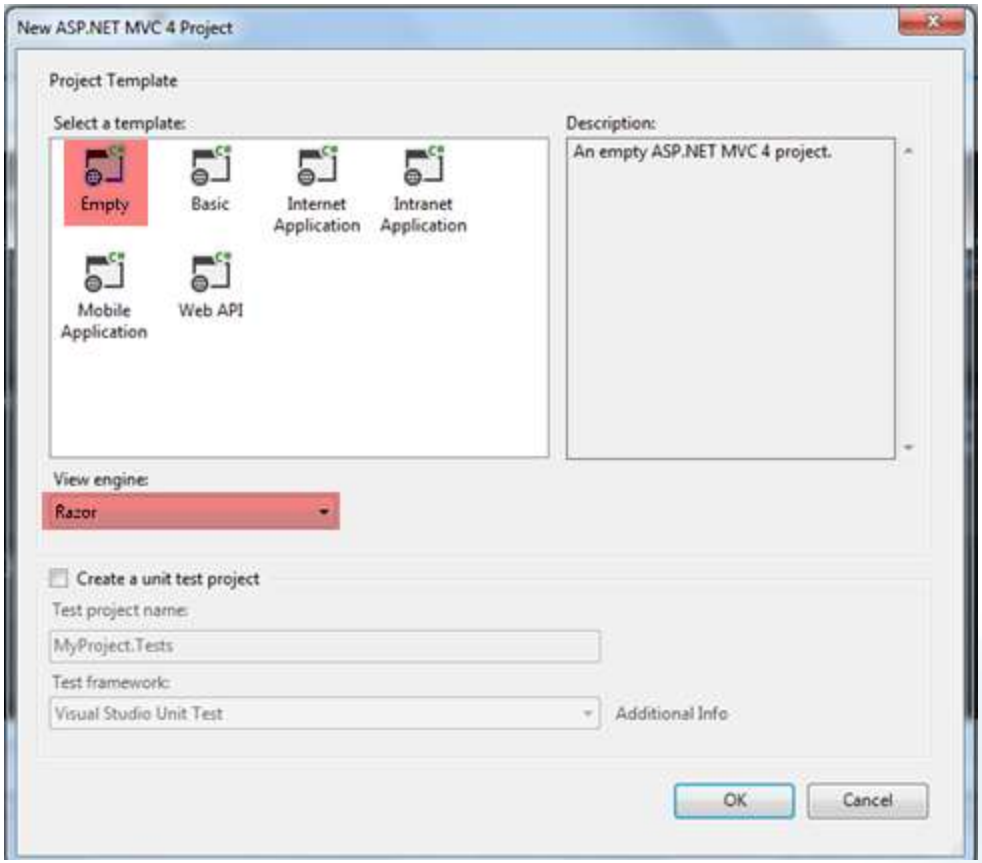
MÔ HÌNH MVC

- Tạo ứng dụng ASP.NET MVC
  - File/New/Project



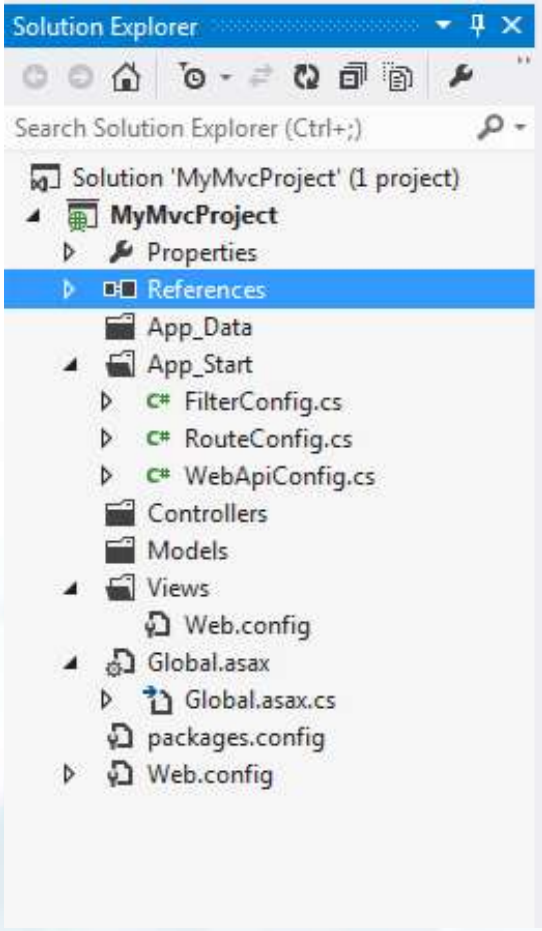
MÔ HÌNH MVC

- Tạo ứng dụng ASP.NET MVC



MÔ HÌNH MVC

- Cấu trúc ứng dụng ASP.NET MVC: Có các thư mục sau:
  - /Controllers: Chứa các class Controller điều khiển URL requests
  - /Models: Chứa các classes biểu diễn, thao tác data và business objects
  - /Views: Chứa template UI, chịu trách nhiệm cho việc rendering output, như HTML
  - /Scripts: thư viện và scripts (.js)



MÔ HÌNH MVC

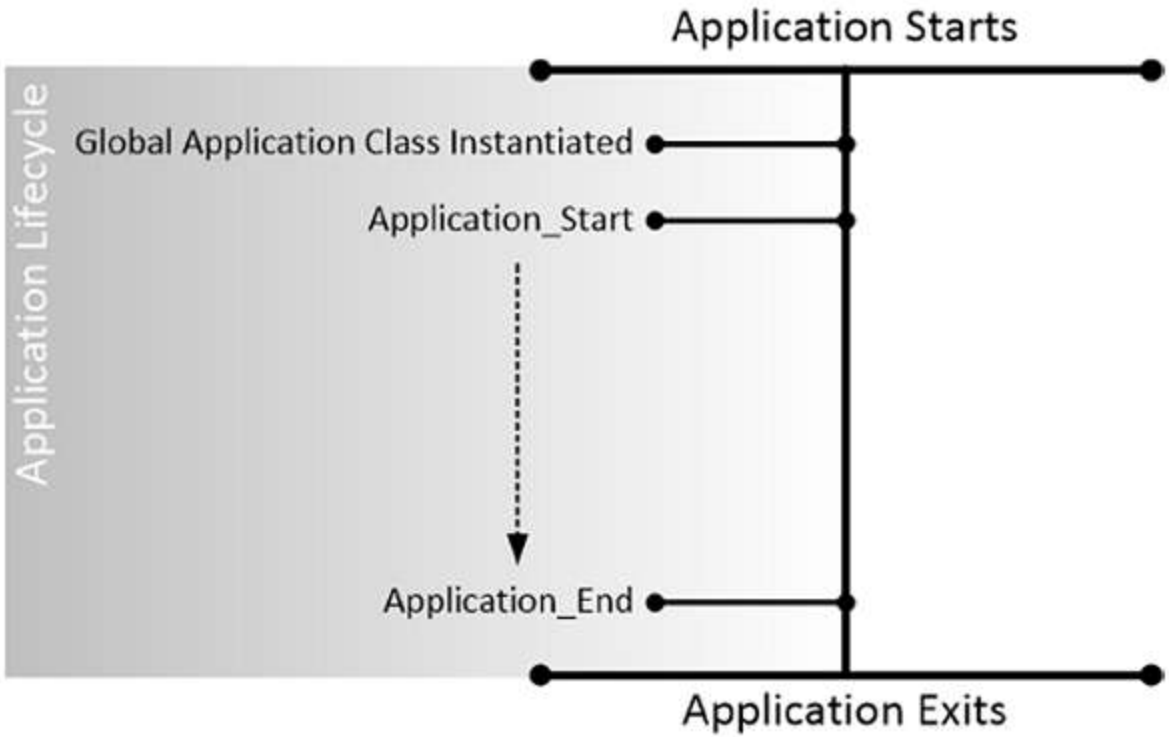
- Cấu trúc ứng dụng ASP.NET MVC
  - /fonts: Bootstrap template system bao gồm một vài custom web fonts
  - /Content: CSS, images, site content, ...
  - /App_Data: Lưu trữ file data ta muốn read/write
  - /App_Start: configuration code cho các tính năng như Routing, bundling, Web API

- Thêm các Packages to the Project
  - Visual Studio cung cấp NuGet package cho phép cung cấp các truy cập các catalog của packages mở rộng cho việc phát triển .NET application.
  - Right click trên cửa sổ Solution/Manage NuGet packages...
  - Chọn các gói cần cài

- Thêm các Packages to the Project

ASP.NET Life Cycles

- ASP.NET Application Life Cycle:
  - Được tính khi bắt khởi tạo ứng dụng đến khi ứng dụng bị dừng.



ASP.NET Life Cycles

- ASP.NET Application Life Cycle:
  - File Global chứa thông tin ở mức ứng dụng
    - Application_Start(): Được gọi khi ứng dụng bắt đầu
    - Application_End(): Được gọi khi ứng dụng kết thúc

ASP.NET Life Cycles

- ASP.NET Request Life Cycle:
  - Request life cycle được mô tả bởi chuỗi các sự kiện cho một tiến trình request từ khi nó yêu cầu đến khi response được gửi.
  - Các sự kiện này có thể được sử dụng khi ta tạo các moduls và điều khiển nó.

ASP.NET Life Cycles

- ASP.NET Request Life Cycle:
  - Một số sự kiện (Sử dụng trong file Global)
    - BeginRequest
    - AuthenticateRequest
    - PostAuthenticateRequest
    - AuthorizeRequest
    - ResolveRequestCache
    - PostResolveRequestCache
    - AcquireRequestState
    - PostAcquireRequestState
    - EndRequest

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Context objects cung cấp thông tin về ứng dụng, current request, và response đang chuẩn bị cho nó. Nó cũng cung các truy cập quan trọng của ASP.NET platform services như security và state data.
  - Lớp trung tâm của context là System.Web.HttpContext

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - System.Web.HttpContext:
    - Application
    - Cache
    - Current
    - Request
    - Response
    - Session
    - Server
    - Trace,...

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Request và HttpRequest:
    - Browser
    - ContentType
    - MapPath(path)
    - Headers
    - HttpMethod
    - MapPath(path)
    - UserHostAddress
    - UserHostName,...

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Request và HttpRequest:
    - Cookies
    - MapPath(path)
    - Headers
    - HttpMethod
    - MapPath(path)
    - Params
    - Form
    - QueryString(Thường không sử dụng trực tiếp trong MVC),...

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Request và HttpRequest:

```
// GET: /ContextObject/
public ActionResult Index(){
    if (Request.Cookies["UserInfo"] == null){
                Response.Write("Không Persitent Cookies ");
     }
     else{
          Response.Write("Persitent Cookies  :" +
          Request.Cookies["UserInfo"]["UserName"]);
     }
     return View();
}
```

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Response và HttpResponse:
    - BufferOutput
    - Charset
    - Cache
    - Clear()
    - ClearContent()
    - ContentEncoding
    - IsClientConnected
    - Write(data),...

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Response và HttpResponse:

```
[HttpPost]
public ActionResult Index( Users model) {
    HttpCookie userInfoCookie = new HttpCookie("UserInfo");
    userInfoCookie.Values["UserName"] = model.UserName;
    userInfoCookie.Values["LastVisit"] =
DateTime.Now.ToString();
    userInfoCookie.Expires = DateTime.Now.AddDays(15);
    Response.Cookies.Add(userInfoCookie);
    return View();
 }
```

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Server và HttpServerUtility:
    - MachineName
    - ScriptTimeout
    - Execute
    - HtmlEncode
    - HtmlDecode
    - MapPath
    - UrlEncode
    - UrlDecode,
    - ...

ASP.NET Life Cycles

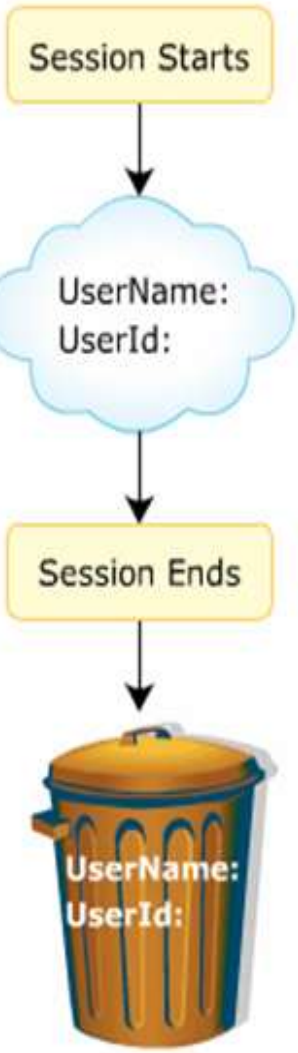
- ASP.NET Context Objects:
  - Đối tượng Application và HttpApplicationState:

```
//Global
 protected void Session_Start(){
     Application["Num"]=(int)Application["Num"]+1;
  }
 //Controller
 public ActionResult Index(){
   ViewBag.SoNguoi=HttpContext.Application["Num"];
   return View();
 }
```

ASP.NET Life Cycles

- ASP.NET Context Objects:
  - Đối tượng Session và HttpSessionState:

```
public RedirectToRouteResult AddToCart( string id){
      var product = new SanPhamFunction().FindEntity(
      var cart = (Cart)Session[CartSession];
      if (cart != null){
          cart.AddItem(product, 1);
           Session[CartSession] = cart; }
      else {
           cart = new Cart();
           cart.AddItem(product, 1);
           Session[CartSession] = cart; }
           return RedirectToAction("Index");
 }
```



ASP.NET Life Cycles

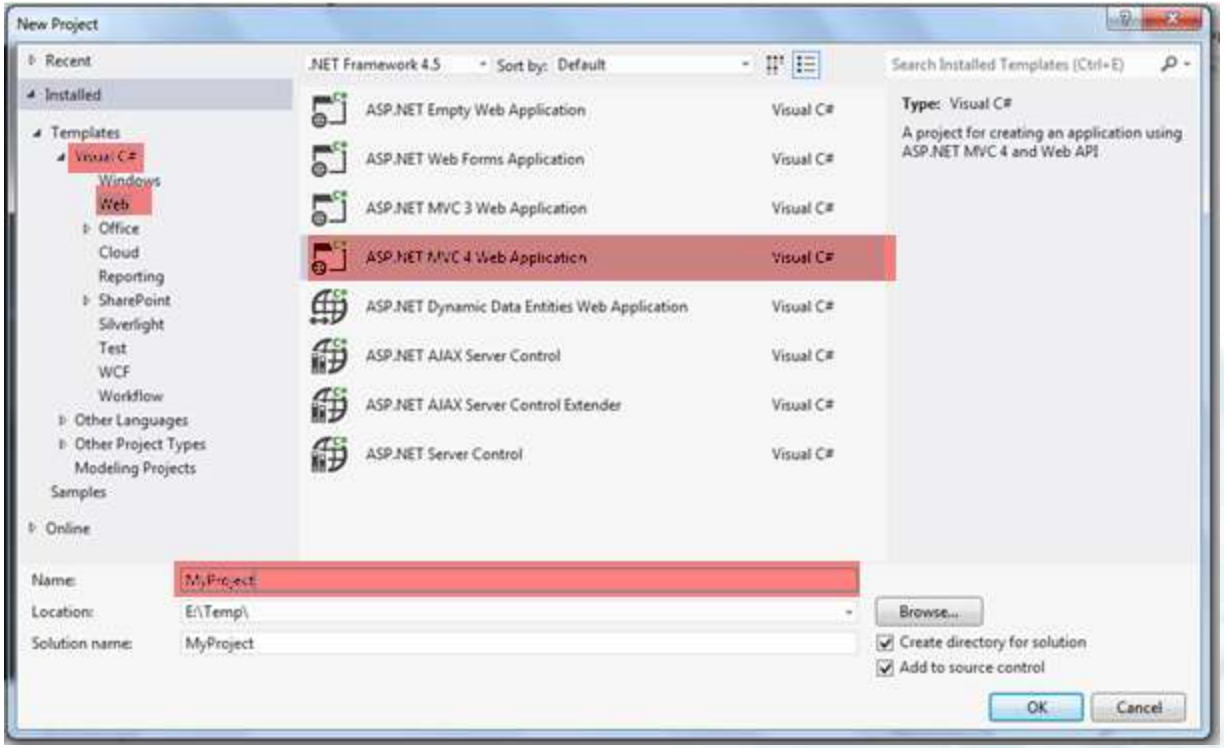
- ASP.NET Context Objects:
  - Đối tượng Session và HttpSessionState:

```
public static class SessionPersister {
    public static string UserName
    {   get{  if (HttpContext.Current == null){
                    return string.Empty;}
var sessionVar = HttpContext.Current.Session["Name"];
              if (sessionVar != null){
                    return sessionVar as string;
               }
               return null;}
            set{
HttpContext.Current.Session["Name"] = value;
            }
        }
}
```

THẢO LUẬN – CÂU HỎI

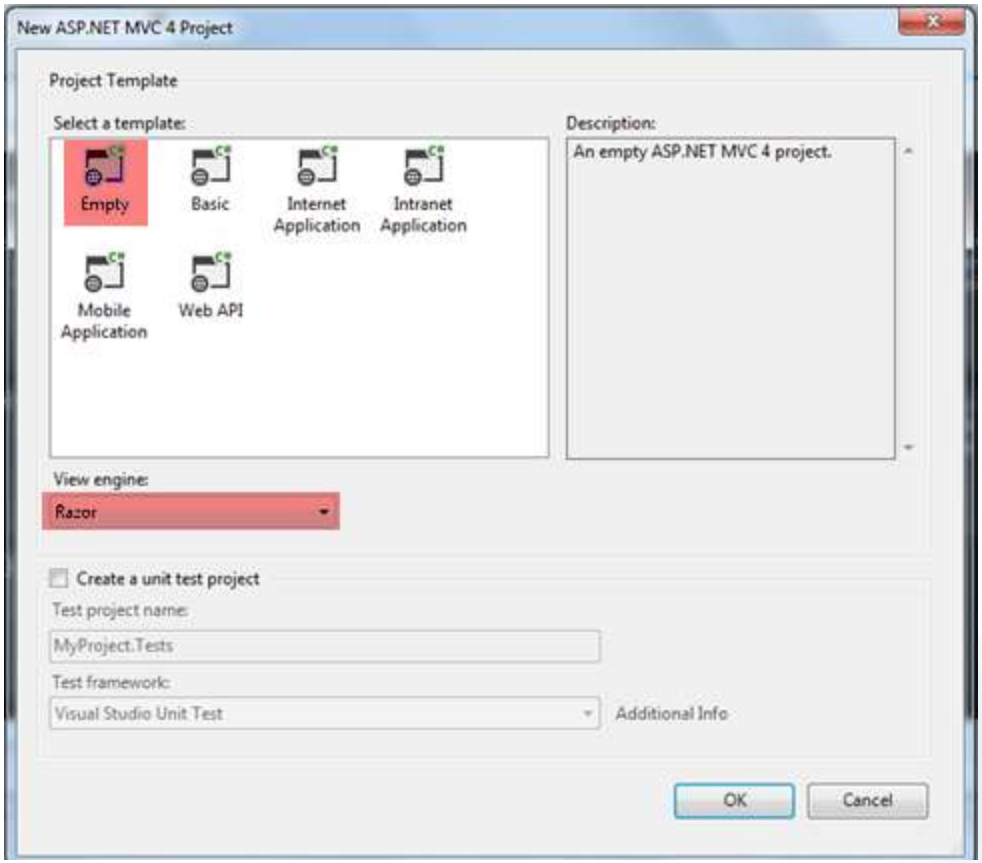
BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo ứng dụng ASP.NET MVC
  - File/New/Project



BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo ứng dụng ASP.NET MVC



BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo Models: Right click trên Models tạo class SanPhamModels. Và tạo 2 class sau:

```
public class SanPhamModels
{
        public int MaSP { get; set; }
        public string TenSP { get; set; }
        public decimal GiaSP { get; set; }
 }
```

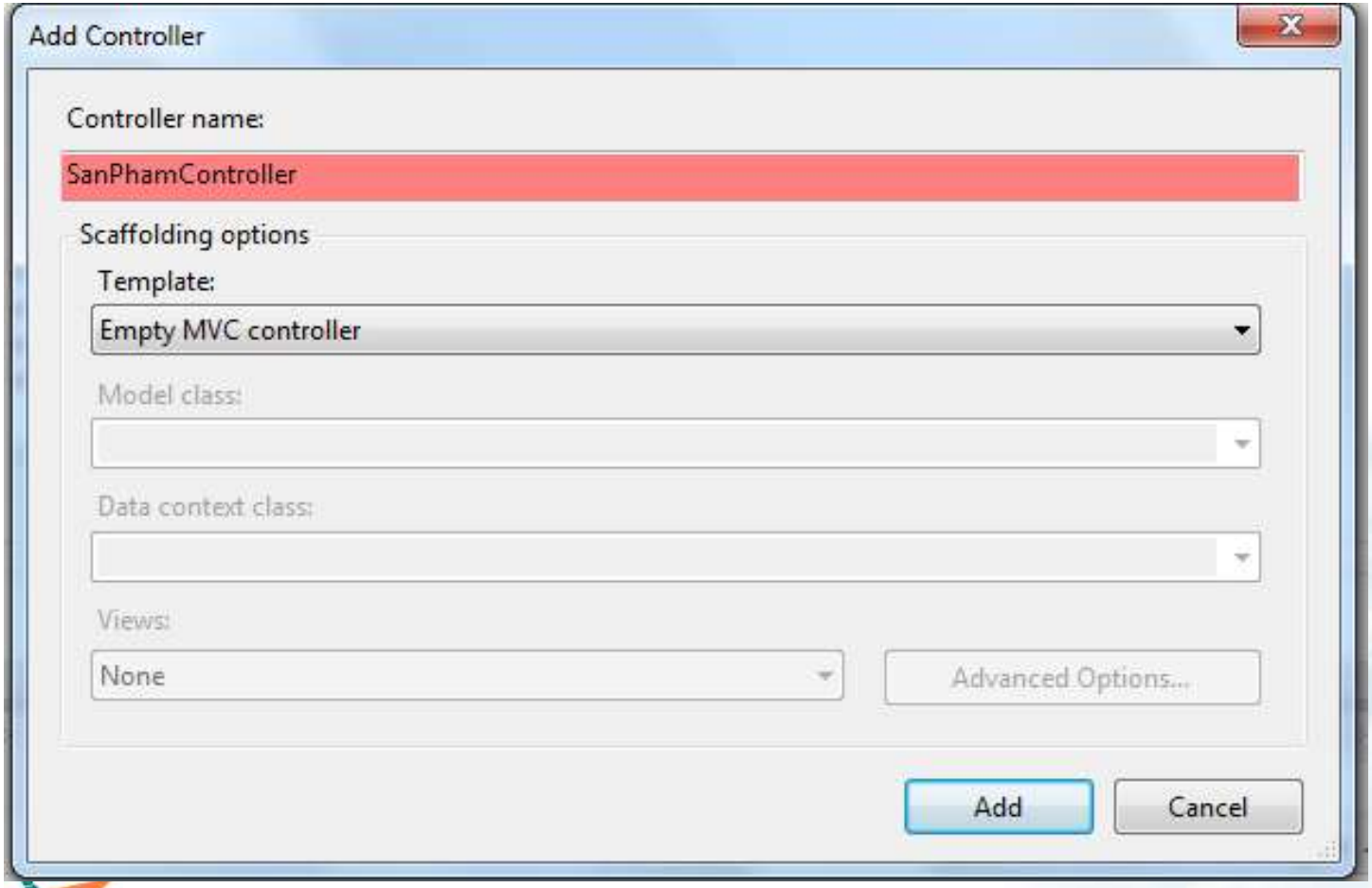
BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo Models: Right click lên Models tạo class SanPhamModels.cs

```
namespace MyMvcProject.Models
{
    public class SanPhamModels
    {
        public int MaSP { get; set; }
        public string TenSP { get; set; }
        public decimal GiaSP { get; set; }
    }
    public class SanPham
    {
        public SanPham()
        {
            _listSanPham.Add(new SanPhamModels { MaSP = 1, TenSP = "HP", GiaSP = 14 });
            _listSanPham.Add(new SanPhamModels { MaSP = 2, TenSP = "Asus", GiaSP = 12 });
            _listSanPham.Add(new SanPhamModels { MaSP = 3, TenSP = "Del", GiaSP = 13 });
            _listSanPham.Add(new SanPhamModels { MaSP = 4, TenSP = "Vaio", GiaSP = 16 });
        }
        public List<SanPhamModels> _listSanPham = new List<SanPhamModels>();
        public void CreateSanPham(SanPhamModels sp)
        {
            _listSanPham.Add(sp);
        }
    }
}
```

BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo Controller: Right click lên Controller



BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo Controller: Viết code

```
namespace MyMvcProject.Controllers
{
    public class SanPhamController : Controller
    {
        private static SanPham _sanpham = new SanPham();
        //
        // GET: /SanPham/

        public ActionResult Index()
        {
            ViewBag.ListSP = _sanpham._listSanPham;
            return View();
        }
        public ActionResult SanPhamAdd()
        {
            return View();
        }
        [HttpPost]
        public ActionResult SanPhamAdd(SanPhamModels sp)
        {
            _sanpham._listSanPham.Add(sp);
            return View();
        }

    }
}
```

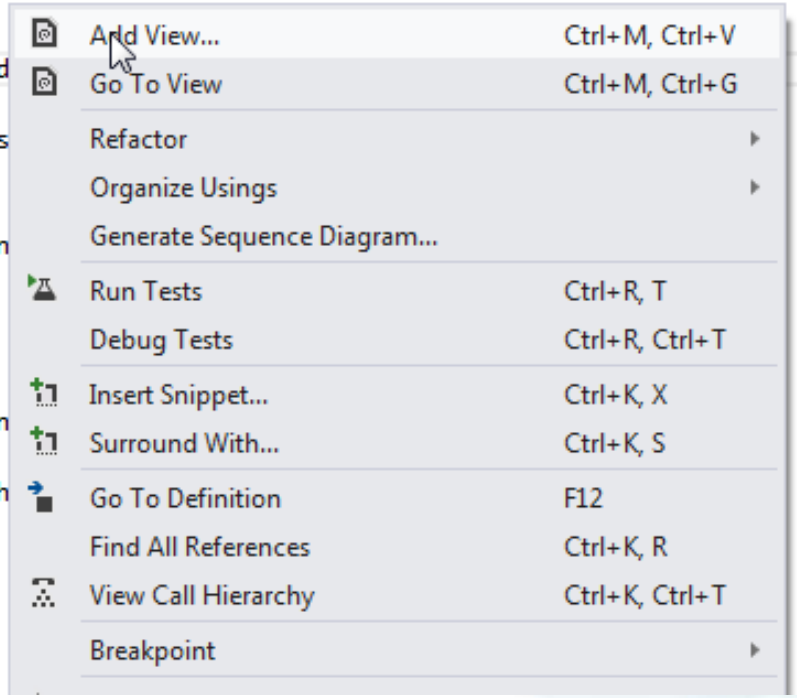
BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo View: Trong cửa sổ thiết kế controller, right click

```
namespace MyMvcProject.Controllers
{
    public class SanPhamController : Controller
    {
        private static SanPham _sanpham = new SanPham();
        //
        // GET: /SanPham/

        public ActionResult Ind
        {
            ViewBag.ListSP = _s
            return View();
        }
        public ActionResult San
        {
            return View();
        }
        [HttpPost]
        public ActionResult San
        {
            _sanpham._listSanPh
            return View();
        }

    }
}
```



BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo Controller: Viết code

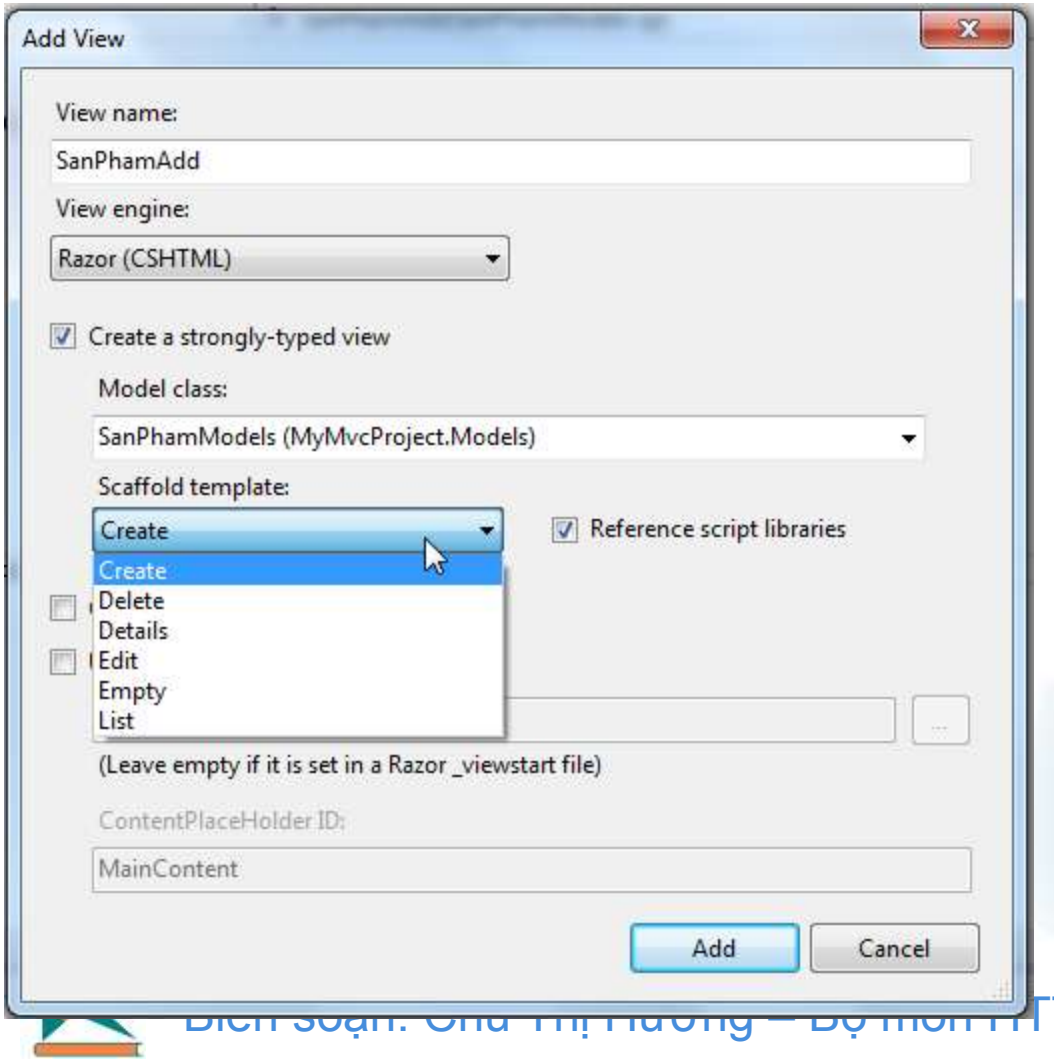
```
namespace MyMvcProject.Controllers
{
    public class SanPhamController : Controller
    {
        private static SanPham _sanpham = new SanPham();
        //
        // GET: /SanPham/

        public ActionResult Index()
        {
            ViewBag.ListSP = _sanpham._listSanPham;
            return View();
        }
        public ActionResult SanPhamAdd()
        {
            return View();
        }
        [HttpPost]
        public ActionResult SanPhamAdd(SanPhamModels sp)
        {
            _sanpham._listSanPham.Add(sp);
            return View();
        }

    }
}
```

BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo View:



BÀI TẬP TẠO ỨNG DỤNG MVC

- Tạo View:

```
@model MyMvcProject.Models.SanPhamModels

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <table>
            <tr><th>Mã sản phẩm</th><th>Tên sản phẩm</th> <th>Giá</th></tr>
            @foreach (MyMvcProject.Models.SanPhamModels a in (ViewBag.ListSP as IEnumerable<MyMvcProject.Models.SanPhamModels>)) {
                <tr><td>@a.MaSP</td>
                <td>@a.TenSP</td>
                <td>@a.GiaSP</td></tr>
            }
        </table>
    </div>
</body>
</html>
```

BÀI TẬP TẠO ỨNG DỤNG MVC

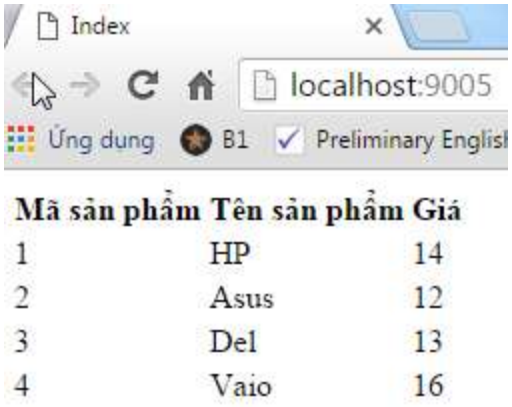
- Cấu hình chạy mặc định: Cấu hình lại trong file RouteConfig

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

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

BÀI TẬP TẠO ỨNG DỤNG MVC

- Chạy thử nghiệm ứng dụng:



Mã sản phẩm	Tên sản phẩm	Giá
1	HP	14
2	Asus	12
3	Del	13
4	Vaio	16

BÀI TẬP TẠO ỨNG DỤNG MVC

- Chạy thử nghiệm ứng dụng với Bootstrap:

```
<head>
    <meta name="viewport" content="width=device-width" />
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />
    <link href="~/Content/bootstrap-theme.min.css" rel="stylesheet" />
</head>
<body>
 <div class="panel panel-primary">
        <h5 class="panel-heading">Results</h5>
        <table class="table table-condensed table-striped">
            <tr><th>Mã sản phẩm</th>
                <th>Tên sản phẩm</th> <th>Giá</th></tr>
            @foreach (dynamic a in ViewBag.ListSP ) {
                <tr><td>@a.MaSP</td>
                <td>@a.TenSP</td>
                <td>@a.GiaSP</td></tr>
            }
        </table>
   </div>
```

BÀI TẬP TẠO ỨNG DỤNG MVC

- Chạy thử nghiệm ứng dụng với Bootstrap: