**2. Sự khác nhau giữa static readonly và const?**

- Static readonly là hằng số được khởi tạo khi [thực thi chương trình.](http://love4all1080.blogspot.com/)   
- Const là hằng số được khởi tạo khi biên dịch chương trình.

**3. Sự khác nhau giữa String và StringBuilder?**

- String là lớp không thay đổi. Mọi phương thức sẽ tạo ra một biến String mới.   
- StringBuilder là lớp thay đổi. Các phương thức của biến sẽ ảnh hưởng đến đến biến [StringBuilder](http://love4all1080.blogspot.com/) hiện tại.

**4. Sự khác nhau giữa “” và String.Empty?**

- “” sẽ tạo thêm 1 đối tượng khi làm việc với nó (so sánh, copy…)   
- String.Empty không tạo đối tượng nào cả. String.Emtpy nằm trong mscorlib.   
Khi kiểm tra 1 String có rỗng hay không nên dùng String.Length == 0 để có tốc độ nhanh nhất.

**5. Có thể gán một struct bằng null hay không?**

- Có thể nếu bạn lập trình với C# 2.0. Ví dụ: int? var = 1; var = null;

**6. Sự khác nhau giữa == và phương thức Equals?**

- == so sánh theo kiểu giá trị.   
- Equals so sánh theo kiểu tham chiếu.

**10. Sự khác nhau giữa abstract class và interface?**

- Có thể thừa kế từ nhiều interface nhưng chỉ có thể thừa kế từ 1 abstract class.   
- Có thể viết sẵn các thực thi trong Abstract class, nhưng interface thì không.   
- Interface dùng để gom các hành động cần được hiện thực , các khả năng của một đối tượng (ví dụ con người và xe cộ cùng có khả năng đi lại, nhưng con người không cùng tính chất với xe cộ), còn abstract class cho các lớp thừa kế cùng 1 loại, tính chất hay trạng thái (ví dụ đàn ông, phụ nữ, trẻ con, người già đều là con người).   
- Abstract class có tốc độ thực thi nhanh hơn interface.   
- Thêm 1 tính năng mới vào interface sẽ phá vỡ toàn bộ các lớp hiện thực, còn abstract thì không.

**12. Sự khác nhau giữa DataSet và DataReader?**

- DataSet có thể đọc và ghi trong khi DataReader chỉ có thể đọc.

- DataSet có thể chứa nhiều bảng còn DataReader chỉ có thể chứa 1 bảng.

- DataSet làm việc ở chế độ không kết nối còn DataReader làm việc ở chế độ kết nối.

- DataSet có thể di chuyển tới và trở lại còn DataReader thì chỉ có thể di chuyển tới.

- DataSet truy cập chậm hơn DataReader.

**18. Sự khác nhau giữa class, object, struct và interface?**

- Class là cấu trúc dữ liệu định nghĩa bởi người dùng chứa các tính chất và phương thức. Object là một trường hợp cụ thể của class. Class không chiếm bộ nhớ, còn Object được tạo trong bộ nhớ. - Class là cấu trúc dữ liệu định nghĩa bởi người dùng chứa các tính chất và phương thức. Struct là kiểu giá trị được lưu trữ trong stack còn class là kiểu tham chiếu (các object) được lưu trữ trong heap. - Khi struct được khởi tạo không bằng hàm khởi tạo người dùng sẽ phải khởi tạo tất cả các Fields trước khi sử dụng nó.   
- Interface là một hợp đồng gồm một nhóm các phương thức cần được hiện thực.

**34. Sự khác nhau giữa error và exception?**

- Error là lỗi của chương trình còn exception là lỗi chương trình được phát hiện và có thể xử lý.

**Phân biệt HttpApplication, Session, ViewSate và HttpContext?**

- Cả 4 thành phần trên đều dùng để lưu dữ liệu của một Web Application.

- [HttpApplication](http://love4all1080.blogspot.com/) có tác dụng trong toàn bộ quá trình thực thi của Web Application.

- Session có tác dụng trong 1 lần ghé thăm (visit) của người dùng. Nó kết thúc khi gặp timeout.

- ViewState có tác dụng trong trong 1 trang web.

- HttpContext có tác dụng trong 1 request.

**3. Thế nào là thuộc tính ? Phương thức ?**  
Thuộc tính là những biến được khởi tạo có kèm theo cơ chế.  
Phương thức là những hàm được khởi tạo có kèm theo cơ chế  
  
**4. Làm thế nào để sử dụng class?**  
Khởi tạo 1 đối tượng thể hiện của lớp đó.  
  
**5. Có các cơ chế nào ? Nói rõ các cơ chế. Private khác protected chỗ nào ?**  
Có 3 cơ chế: Public, private, protected.  
Public: cho phép truy cập và sử dụng ở phạm vi trong class và ngoài class.  
Protected: cho phép truy cập cái phương thức và thuộc tính trong class, có hỗ trợ sự kế thừa.  
Private: giống private, nhưng không cho phép kế thừa.

**8. Ghi đè là gì ? Tác hại của việc ghi đè là gì ? Làm gì để khắc phục việc ghi đè ? Tầm vực của parent là bao nhiêu cấp ? Làm sao để chống việc ghi đè?**  
– Ghi đè là việc lớp cha và lớp con trùng tên phương thức.  
– Tác hại của việc ghi đè là làm vỡ cấu trúc ban đầu mà lớp cha đã gây dựng nên.  
– Để khắc phục ghi đè ta dùng từ khóa parent:: bên trong phượng thức bị ghi đè.  
– Tầm vực của parent chỉ được 1 cấp. Để chống việc ghi đè ta dùng từ khóa final đặt trước tên phương thức bị ghi đè.  
  
**9. Final là gì ? Diễn giải nó như thế nào ? Nêu ra các chức năng của final.**  
– Final là qui ước phương thức cuối cùng hoặc 1 class cuối cùng.   
– Final dùng để chống việc ghi đè hoàn toàn và chống sự kế thừa.  
  
**10. Nếu sử dụng phương thức private có chống được việc ghi đè không?**  
– Không! Vì cơ chế private chỉ chống sự kế thừa.  
  
**11. Thuộc tính tĩnh là gì? Tác dụng là gì ? Khác thuộc tính thường chỗ nào?**  
**Làm sao để truy xuất thuộc tính tĩnh ? Thuộc tính tĩnh khác hằng chỗ nào ?**  
– Là thuộc tính được qui ước từ khóa static đứng trước tên thuộc tính. Tác dụng là thường dùng để huy động hoặc lặp lại hành động lặp lại nhiều lần nên đỡ tốn tài nguyên.  
– Thuộc tính tĩnh khác thuộc tính thường ở chỗ không cần khởi tạo đối tượng vẫn có thể truy xuất được.  
– Để truy xuất thuộc tính tĩnh dùng từ khóa self::tenthuoctinh  
– Thuộc tính tĩnh khác hằng ở chỗ là giá trị có thể thay đổi, trong khi đó hằng là có giá trị cố định.  
  
**12. Clone là gì ? Sử dụng Clone như thế nào ?**  
– Clone là nhân bản 1 đối tượng đã có sẵn (không phải tạo đối tượng mới).  
– Sử dụng clone khi ta không muốn nó làm thay đổi giá trị của lớp ban đầu.  
  
**13. Lớp trừu tượng là gì ? Dùng adstract để làm gì ? Nhược điểm của abstract là gì? Có cách nào để khắc phục nhược điểm đó không ?**  
– Lớp trừu tượng là lớp chứa các phương thức trừu tượng, các lớp sử dụng lớp trừu tượng phải thực thi các phương thức trừu tượng của lớp trừu tượng đó.  
– Dùng Abstract để quản lý dự án (vừa và nhỏ).  
– Nhược đểm là mang tính đơn hình(chỉ được kế thừa từ 1 lớp). Để khắc phục ta dùng giao diện (interface).  
  
**14. Interface là gì ? Công dụng của nó ?**  
– Interface là giao diện, mang tính đa hình cho phép sự dụng 1 hay nhiều giao diện  
– Công dụng là để quản lý dứ án có qui mô lớn.  
  
**15. Phân biệt sự khác và giống nhau của interface và abstract là gì ?**  
– Giống nhau là để dùng để quản lý dự án  
– Khác nhau: Abstract mang tính đơn hình, chỉ kế thừa được 1 lớp (bản chất là 1 lớp). Còn Interface mang tính đa hình, có thể sử dụng 1 hay nhiều giao diện.  
  
**16. 1 lớp có thể sử dụng abstract và interface Cùng lúc không ?** – Được  
  
**17. 1 lớp có thể sử dụng 2 giao diện có cùng tên hàm không?** – Không  
  
**18. Lazy loading techology là gì ?**  
– Là công nghê tự động nạp 1 file khi khởi tạo 1 đối tượng(tên file trùng tên lớp)

- Phân biệt giữa PartialView và View ? PartialView nó không phài là 1 View hoàn chỉnh. Điểm khác nhau cơ bản của View và Partial View là View sẽ được include vào \_Layout khi return về client, còn Partial View thì without \_Layout.

- Khi nào thì sử dụng PartialView và View ?

**Cách truyền dữ liệu từ Controller về View có những cách nào? ViewBag, ViewData, TempData và  Session? Sự khác nhau giữa chúng như thế nào? Khi nào thì dùng cái nào?**

ASP.NET MVC cung cấp 3 tùy chọn ViewData, ViewBag và TempData để truyền dữ liệu từ controller vào view và trong request kế tiếp. ViewData và ViewBag tương tự như nhau, TempData thực hiện thêm nhiệm vụ. Chúng ta sẽ nói về điểm chính của 3 đối tượng:

**Sự giống nhau giữa ViewBag và ViewData:**

1. Giúp duy trì dữ liệu khi bạn di chuyển từ view vào controller

2. Được dùng để truyền dữ liệu từ controller vào view tương ứng

3. Chu kỳ tồn tại ngắn, giá trị sẽ thành null khi redirect xuất hiện. Đây là bởi vì mục tiêu thiết kế các đối tượng này là để giao tiếp giữa controller và view

**Sự khác nhau giữa ViewBag và ViewData:**

1. ViewData là 1 từ điển các đối tượng (dictionary) kế thừa từ lớp ViewDataDictionary và có thể truy cập dùng chuỗi key.

2. ViewBag là thuộc tính động (dynamic) mang đến thuận lợi từ đặc tính mới của C# 4.0

3. ViewData yêu cầu chuyển kiểu với các kiểu dữ liệu phức tạp và kiểm tra giá trị null để tránh lỗi

4. ViewBag không yêu cầu chuyển kiểu với các kiểu dữ liệu phức tạp

**Ví dụ về ViewBag và ViewData:**

public ActionResult Index()

{

ViewBag.Name = "Ngoc Dinh NGUYEN";

return View();

}

public ActionResult Index()

{

ViewData["Name"] = "Ngoc Dinh NGUYEN";

return View();

}

Trong View:

@ViewBag.Name

@ViewData["Name"]

**TempData:**

TempData cũng là 1 từ điển đối tượng (dictionary) kế thừa từ lớp TempDataDictionary, được lưu trong 1 chu kỳ rất ngắn. Điểm khác biệt chính là chu kỳ sống của đối tượng. TempData giữ thông tin trong khoảng thời gian HTTP Request. Điều này có nghĩa là chỉ từ 1 trang đến 1 trang khác. Điều này cũng hoạt động với 302/303 redirection bởi vì nó cùng HTTP Request. Giúp duy trì dữ liệu khi bạn di chuyển từ 1 controller sang 1 controller khác hoặc từ hành động này sang hành động khác. Nói cách khác, khi bạn redirect, TempData giúp duy trì dữ liệu giữa các redirect. Nó là biến cục bộ dùng trong session. Nó yêu cầu chuyển kiểu cho các kiểu dữ liệu phức tạp, và kiểm tra giá trị null để tránh lỗi. Thông thường nó dùng để lưu error message hoặc validation message.

public ActionResult Index()

{

var model = new Review()

{

Body = "Start",

Rating = 5

};

TempData["ModelName"] = model;

return RedirectToAction("About");

}

public ActionResult About()

{

var model = TempData["ModelName"];

return View(model);

}

## ViewBag-ViewData

* Giống nhau về cách sử dụng của ViewBag và ViewData:
  + Giúp bạn có thể chuyển dữ liệu từ [Controller](http://codehay365.com/hd-lap-trinh-web-asp-net-mvc-5-phan-2/) xuống [View](http://codehay365.com/hd-lap-trinh-web-asp-net-mvc-5-phan-4/)
  + Truyền dữ liệu từ Controller xuống View tương ứng
  + Chỉ sử dụng 1 lần duy nhất, sau khi dùng xong dữ liệu ở hai thằng này được giải phóng.
* Khác nhau giữa ViewBag và ViewData:
  + ViewData là một dạng Dictionary, để truy cập vào biến bạn cần khai báo một key dạng string: ViewData[“Danhmuc”].
  + ViewBag: là một điểm mới trong Net Framework 4.0. Cho phép bạn định nghĩa các biến dạng Dynamic: ViewBag.DanhMuc.
  + ViewData: đòi hỏi bạn cần kiểm tra dữ liệu Null và chuyển đổi kiểu trước khi sử dụng
  + ViewBag: không đòi hỏi kiểm tra hay chuyển đổi kiểu (ép kiểu)

**Cách làm việc Routing của ASP.NET MVC**

* Tất cả cấu trúc URL đều được khai báo trong Global.ascx, chịu trách nhiệm ánh xạ đường dẫn trên trình duyệt.

Trình duyệt yêu cầu một địa chỉ từ Controller Acction được gọi là định tuyến URL (URL routing).

URL routing sẽ chỉ định request tới Controller Action.

URL routing sử dụng một bảng định tuyến để điều khiển các yêu cầu (request).

Bảng định tuyến được tạo khi ứng dụng được chạy lần đầu tiên.

Bảng định tuyến được thiết lập trong file Global.asax

Razor hỗ trợ lập trình viên viết các khối mã lệnh tại các View. Một khối mã lệnh Raror là một phần của View, và nó được viết bằng các ngôn ngữ C# hoặc VB.  
Cú pháp @{<code>}

Làm sao để sử dụng nhiều model trong một view

class ListModel  // class này bạn đề đâu tùy ý. khi dùng thì using tới controller or view là oke

{

   public Sanpham List\_sanpham {get; set ;};

   public Sanpham List\_Khachang {get; set ;};

}

Trong Controller

Public Actionresult index  ()

{

 ListModel  l = new  ListModel   ();

 // truy vấn dữ liệu

  return view (l);

}

**Bundling:** Là một nhóm file mà có thể được tham chiếu bởi một tên duy nhất và được loaded bởi HTTP requestor.

**Minification:** Đó là xư lý để bỏ đi các khoảng trống dư thừa, xuống dòng và comment trong code để giảm đi cỡ và cải thiện thời gian load.

**Tại sao phải dùng 2 kỹ thuật này :**  
Cơ bản các developer sử dụng nhiều files JS và CSS files và dẫn tới vấn đề performance cho web. Bởi vì nhiều file JS và CSS files yêu cầu nhiều HTTP Requests từ trình duyệt dẫn tới suy giảm về hiệu suất và thời gian load cho web.

# **"Web API"** là **"công nghệ khá mới của hãng Microsoft  để xây dựng dịch vụ thành phần phân tán"**. **Web API** là mô hình hỗ trợ MVC: routing, controller, action result, filter, filter, loc container, model binder, unit test, injection.**Web API** hỗ trợ restful đầy đủ các phương thức: Get/Post/put/delete dữ liệu.

# **Web API (.NET 4 trở lên)**

1. Đây là một framework mới giúp cho việc xây dựng các HTTP service rất đơn giản và nhanh chóng
2. Open Source và có thể được sử dụng bởi bất kì client nào hỗ trợ XML, JSON
3. Hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content formats
4. Có thể host trong ứng dụng hoặc trên IIS
5. Kiến trúc lý tưởng cho các thiết bị có băng thông giới hạn như smartphone, tablet
6. Định dạng dữ liệu có thể là JSON, XML hoặc một kiểu dữ liệu bất kỳ

Ưu điểm:

* Cấu hình hết sức đơn giản khi so với WCF
* Performance cao
* Hỗ trợ RESTful đầy đủ
* Hỗ trợ đầy đủ các thành phần [MVC](http://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx) như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test
* Open Source

Nhược điểm:

* Còn rất mới nên chưa có nhiều đánh giá về nhược điểm của Web API

Nói chung, Html.Partial() và Html.RenderPartial() dùng để chèn sub view có nội dung tĩnh hoặc nội dung được render với dữ liệu lấy từ parent view. Ngược lại, Html.Action() và Html.RenderAction() dùng để chèn sub view được render với dữ liệu sinh ra từ kết quả thực thi action method trên controller.

Sự khác biệt duy nhất giữa Html.Partial() và Html.RenderPartial() là Html.Partial() trả về một chuỗi được encoded (MvcHtmlString class) trong khi Html.RenderPartial() render và write trực tiếp sub view vào trong output stream. Điều này cũng đúng cho Html.Action() và Html.RenderAction().  
  
Vì Html.Partial() và Html.Action() trả về một chuỗi nên chúng có thể được viết trực tiếp trong Razor như sau  
  
@Html.Partial("ViewName")  
@Html.Action("ActionName", "ControllerName")  
  
Trong khi đó, Html.RenderPartial() và Html.RenderAction() trả về void nên phải được gọi như là một method trong Razor và do đó phải được đặt trong cặp dấu {} như sau  
  
@{ Html.RenderPartial("ViewName"); }  
@{ Html.RenderAction("ActionName", "ControllerName"); }  
  
Khi truyền view name cho Html.Partial() và Html.RenderPartial() thì asp.net mvc sẽ tìm kiếm sub view ở "~/Views/ControllerName/ViewName.cshtml" trong đó "ControllerName" là tên controller mà gọi parent view hoặc ở "~/Views/Shared/ViewName.cshtml".  
  
Khi sub view cần render không nằm trong những location nêu trên thì phải truyền virtual path cho Html.Partial() và Html.RenderPartial() như ví dụ sau  
  
@Html.Partial("~/Views/Account/SubViews/AccountPartial.cshtml")  
@{ Html.RenderPartial("~/Views/Shared/Account/AccountPartial.cshtml"); }

AngularJS là một framework có cấu trúc cho các ứng dụng web động.

* Scope : là đối tượng có nhiệm vụ giao tiếp giữa controller và view của ứng dụng.
* Controller : xử lí dữ liệu cho đối tượng $scope, từ đây bên views sẽ sử dụng các dữ liệu trong scope để hiển thị ra tương ứng.
* Data-binding : tự động đồng bộ dữ liệu giữa model và view
* Service : là singleton object được khởi tạo 1 lần duy nhất cho mỗi ứng dụng, cung cấp các phương thức lưu trữ dữ liệu có sãn. ($http, $httpBackend, $sce, $controller, $document, $compile, $parse, $rootElement, $rootScope …..)
* Filter : Lọc các tập con từ tập item trong các mảng và trả về các mảng mới.
* Directive : dùng để tạo các thẻ HTML riêng phục vụ những mục đích riêng. AngularJS có những directive có sẵn như ngBind, ngModel…
* Temple : một thành phần của view, hiển thị thông tin từ controller
* Routing : chuyển đổi giữa các action trong controller, qua lại giữa các view.

**Các components chính**

* ng-app : định nghĩa và liên kết một ứng dụng AngularJS tới HTML.
* ng-model : gắn kết giá trị của dữ liệu ứng dụng AngularJS đến các điều khiển đầu vào HTML.
* ng-bind : gắn kết dữ liệu ứng dụng AngularJS đến các thẻ HTML.

**Ưu điểm của angularJS**

* Cung cấp khả năng tạo ra các Single Page Aplication dễ dàng.
* Cung cấp khả năng data binding tới HTML, khiến cho người dùng cảm giác linh hoạt, thân thiện.
* Dễ dàng Unit test
* Dễ dàng tái sử dụng component
* Giúp lập trình viên viết code ít hơn với nhiều chức năng hơn.
* Chạy được trên các loại trình duyệt, trên cả PC lẫn mobile.

**Nhược điểm**

* Không an toàn : được phát triển từ javascript cho nên ứng dụng được viết bởi AngularJS không an toàn. Nên có sự bảo mật và xác thực phía server sẽ giúp ứng dụng trở nên an toàn hơn.
* Nếu người sử dụng ứng dụng của vô hiệu hóa JavaScript thì sẽ chỉ nhìn thấy trang cơ bản.