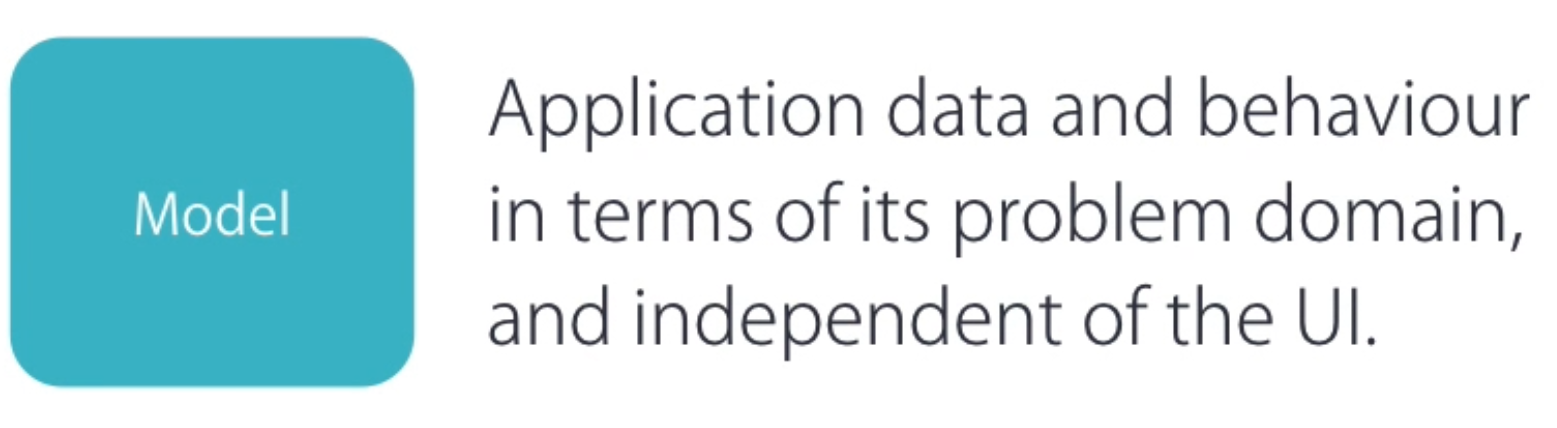
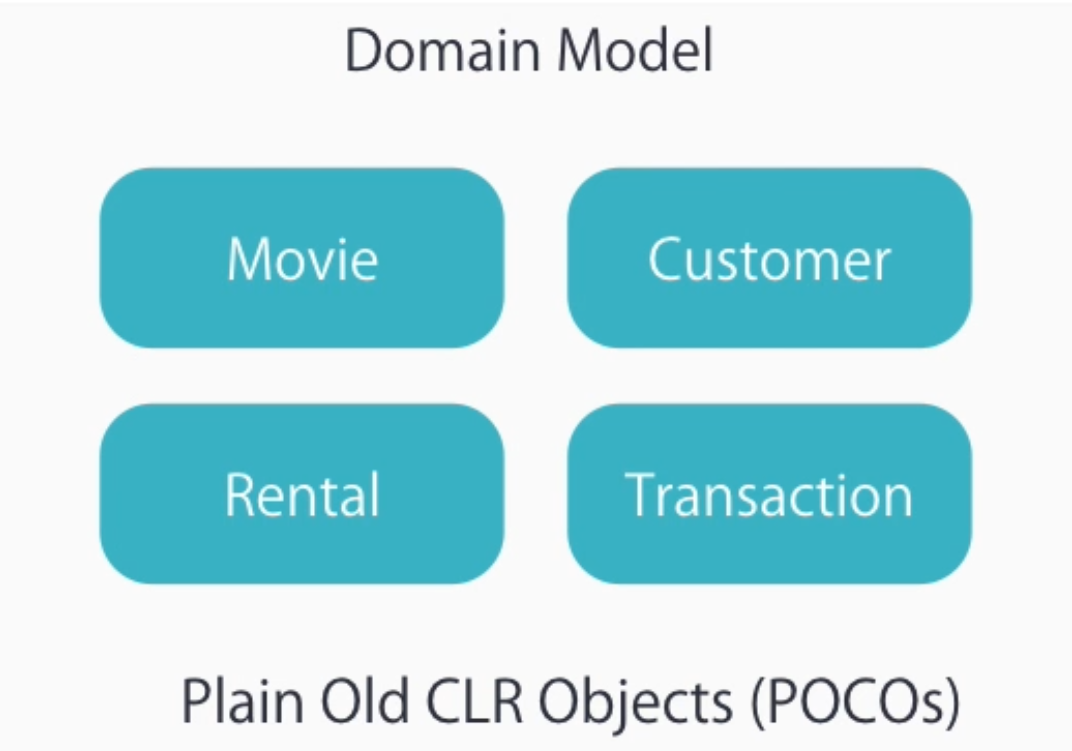
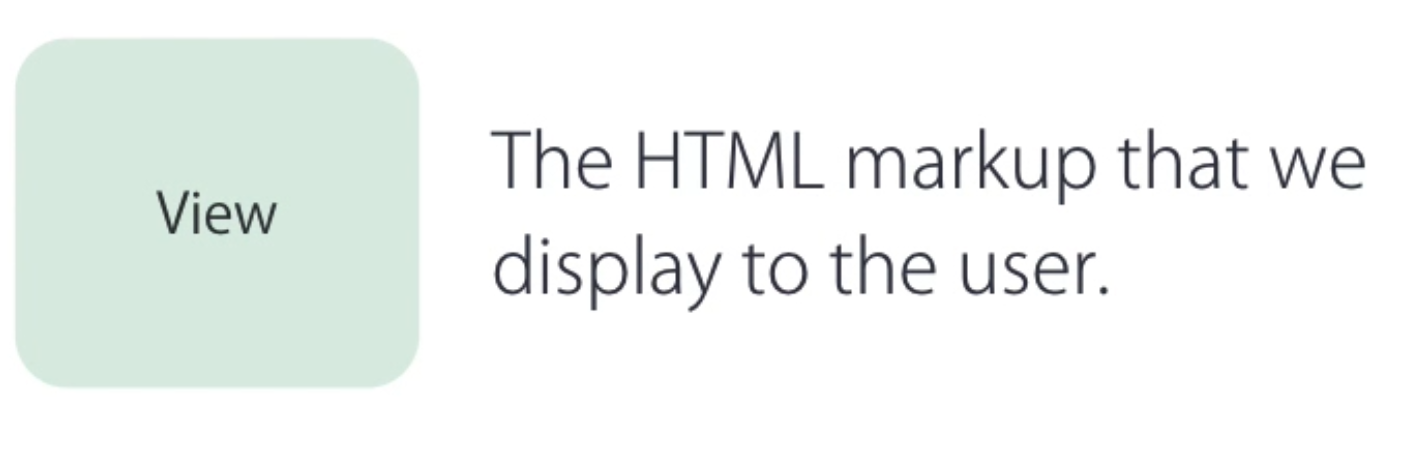
**MVC Pattern**

**M – Model** : is class, …

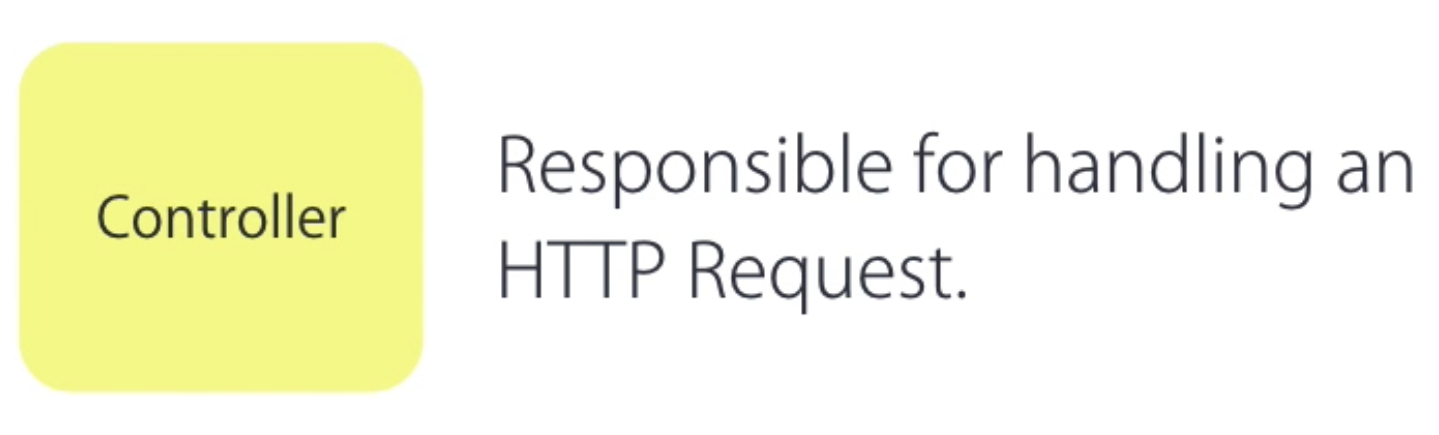




**V – View:**

****

C – Controller



Extension some tools

* Visual studio productivity power tool.
* Web essentials

**CREATE PROJECT**

1. ASP.NET Web Application

* Add to source control ( use for Git )

2. Select template MVC

* App\_Start -> router config

public static void RegisterRoutes(**RouteCollection** routes)

        {

            routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

            routes.MapRoute(

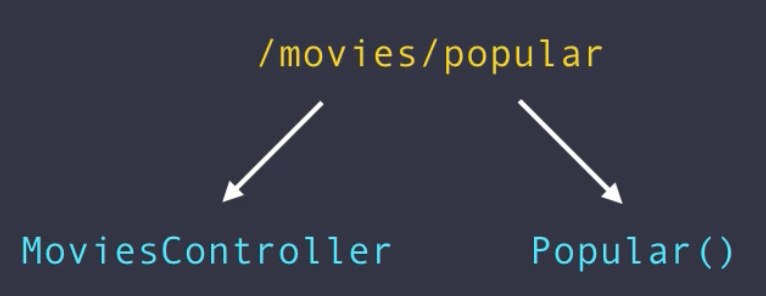
                name: "Default",

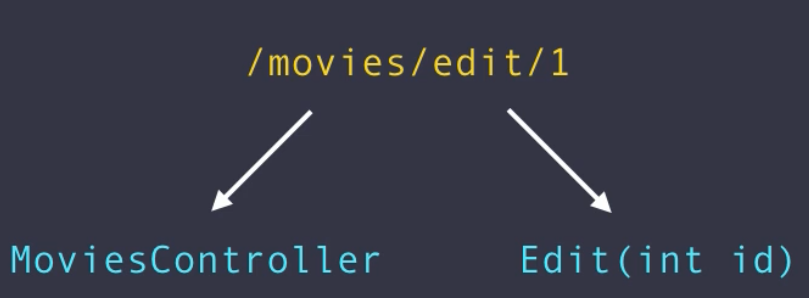
                url: "{controller}/{action}/{id}",

                defaults: new { controller = "Home", action = "Index", id = **UrlParameter**.Optional }

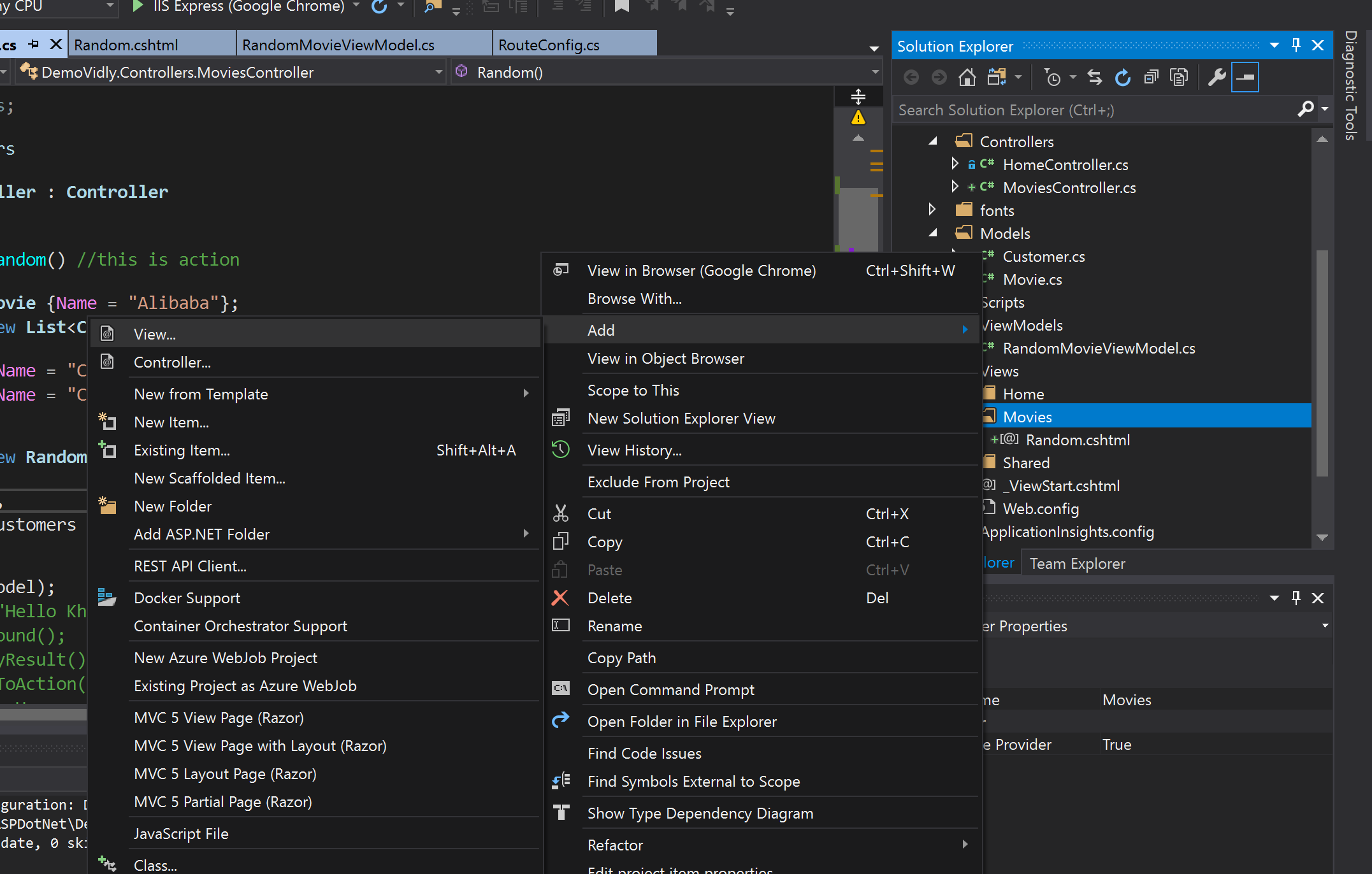
            );

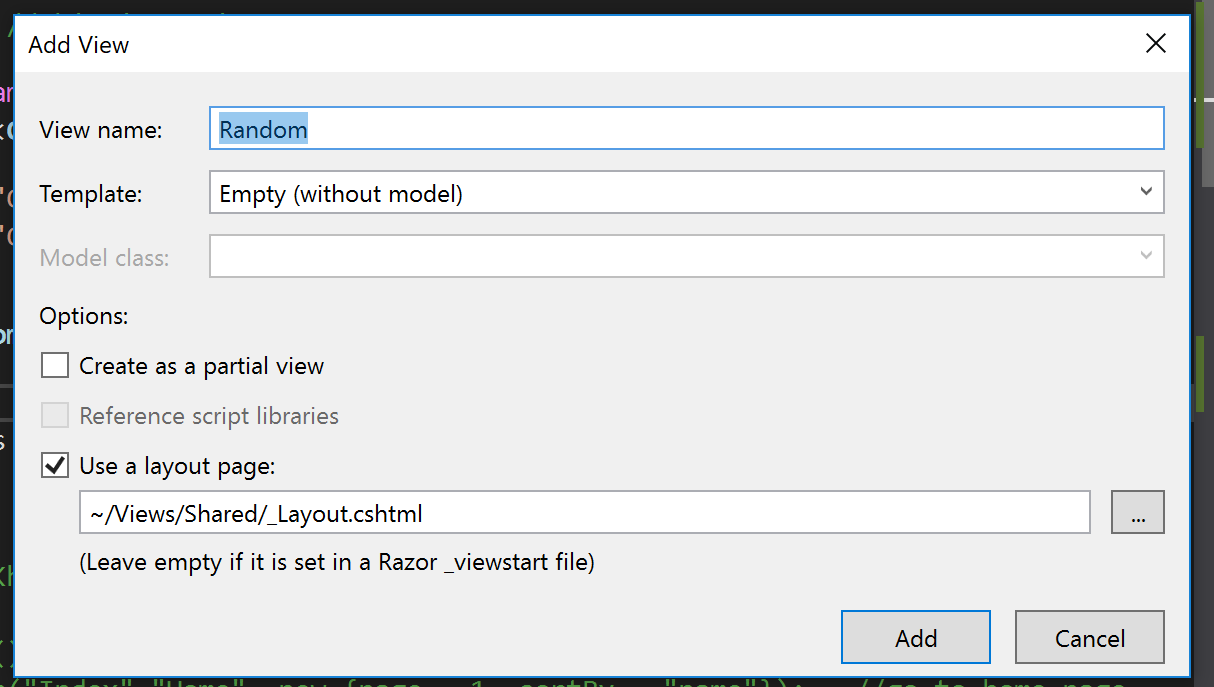
        }





* Create class in Models folder -> name Movie
* Create controller in Controller folder -> name MoviesController
* public class **MoviesController** : **Controller**
* {
* // GET: Movies/Random
* public **ActionResult** Random()
* {
* var movie = new **Movie** {Name = "Alibaba"};
* return View(movie); // transfer param
* }
* }
* Must be create View in View folder -> movies





Random is name of action in MoviesController

@model DemoVidly.Models.**Movie** //if you want to transfer param from view()

@{

    ViewBag.Title = "Random";

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

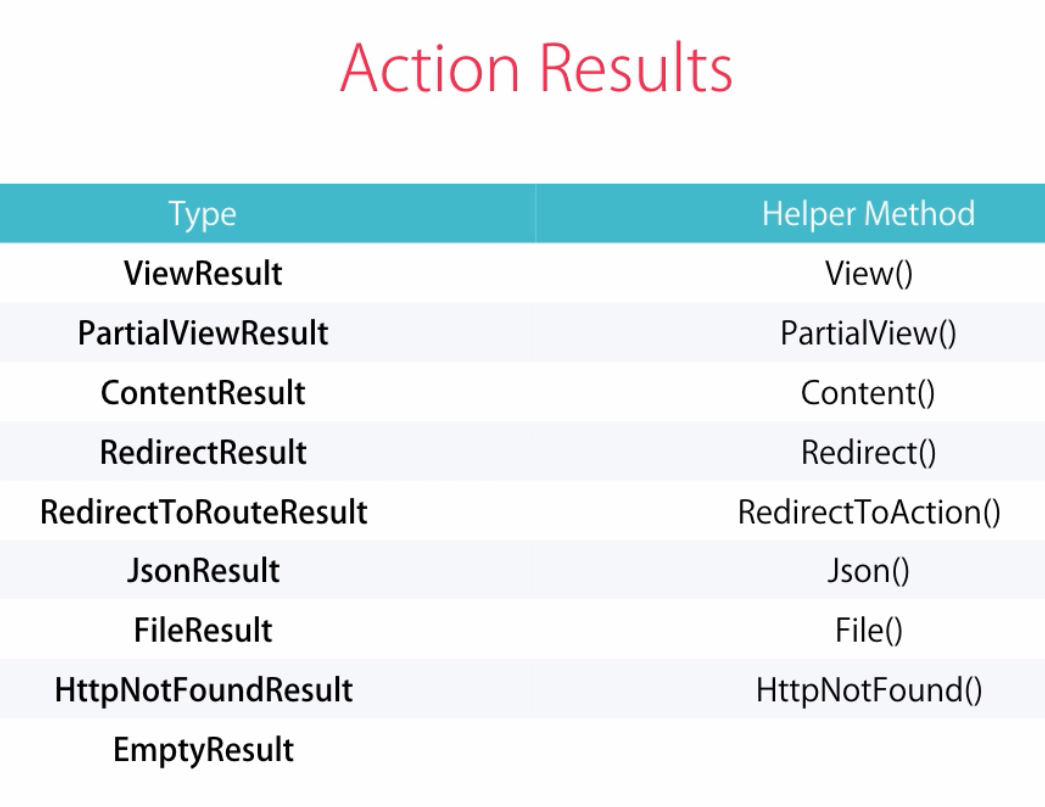
<h2>@Model.Name</h2> // transfer param

Bootstrap

* In Content folder contain all bootstrap theme
* Bootswatch.com
* After add the bootstrap-lumen ( bootstrap theme ) to content folder, must be go to folder App\_Start -> BundleConfig.cs -> change the StyleBundle
* bundles.Add(new **StyleBundle**("~/Content/css").Include(
* "~/Content/bootstrap-lumen.css",
* "~/Content/site.css"));

**Action Result**

Action Result is actually a data type. When it is used with action method, it is called return type. As you know, **an action is referred to as a method of the controller,** **the Action Result is the result of action when it executes.** In fact, **Action Result is a return type.** This return type has many other derived types.



Ctrl + Shift + b : built project not open new tab of browser

Ctrl + f5 : built project and open new tab of browser

namespace DemoVidly.Controllers

{

    public class **MoviesController** : **Controller**

    {

        // GET: Movies/Random

        public **ActionResult** Random()    //this is action

        {

            var movie = new **Movie** {Name = "Alibaba"};

            //return View(movie);

            //return Content("Hello Khanh");

            //return HttpNotFound();

            //return new EmptyResult();

            return RedirectToAction("Index","**Home**", new {page = 1, sortBy = "name"});   //go to home page

             // Index: action , Home: controler,

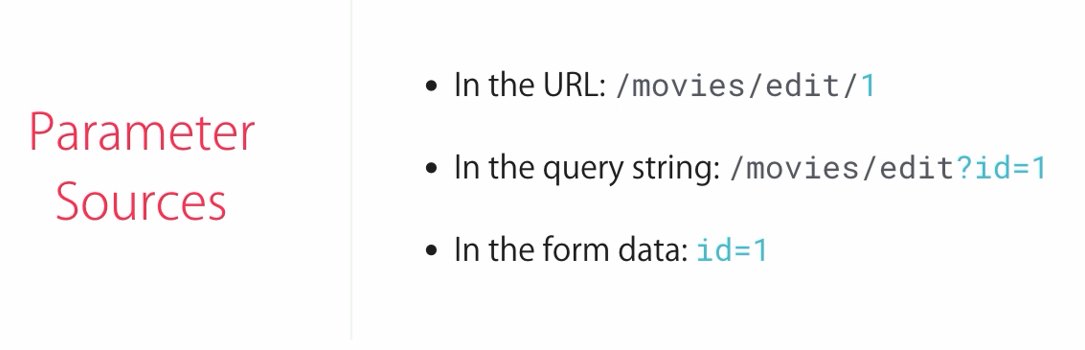
new {page = 1, sortBy =  "name"} : active param Url

        }

    }

}

**Action Parameters**



public class **MoviesController** : **Controller**

    {

        // GET: /movies/edit/5

( movies : controller , edit: action, 5: parameter )

        public **ActionResult** Edit(int id)

        {

            return Content("Id = " + id);

        }

    }

Note: the default parameter is id. url: "{controller}/{action}/{id}"

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 = "Home", action = "Index", id = **UrlParameter**.Optional }

            );

        }

    }

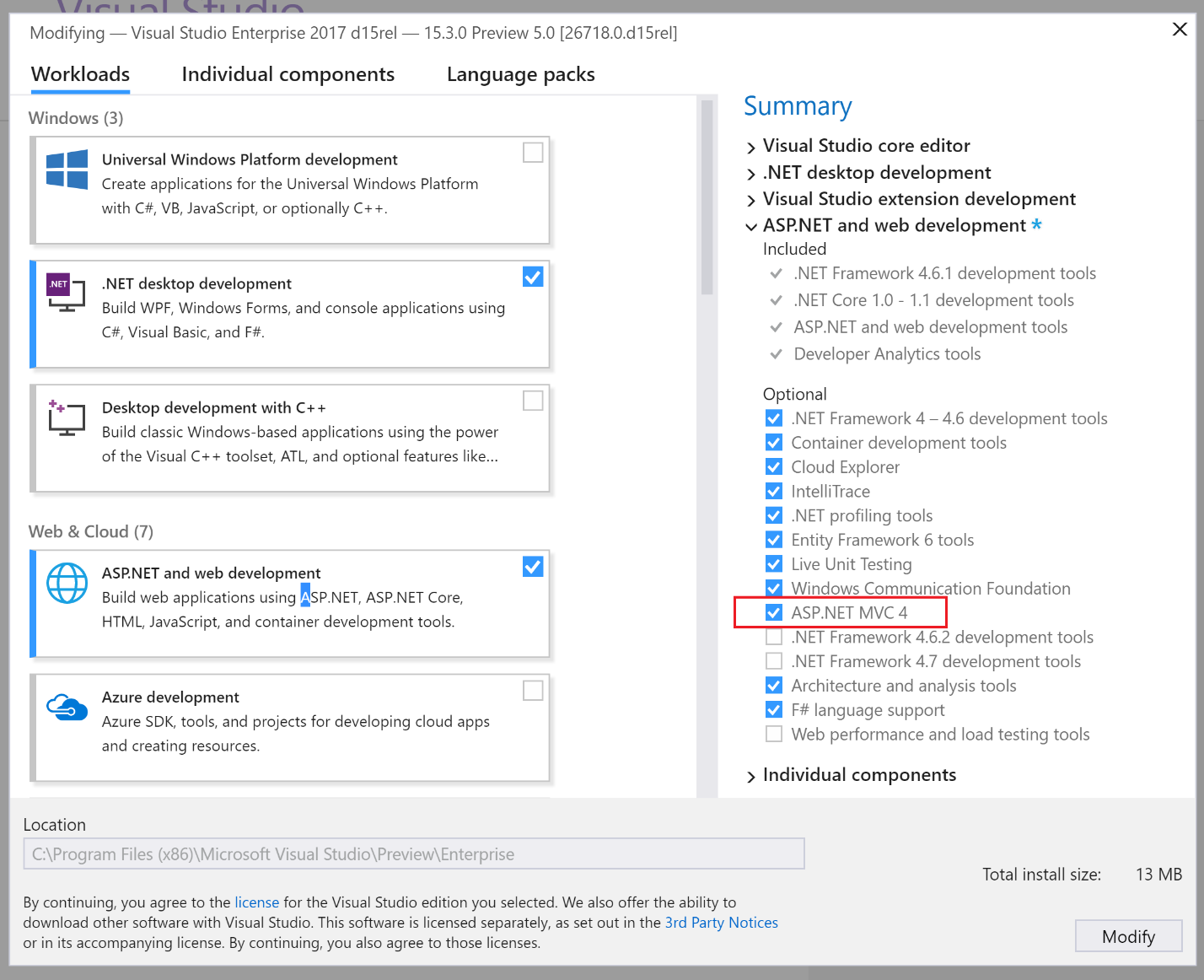
**mvcaction4 - >**

public **ActionResult** Action()

        {

            return View();

        }



**Custom Routes**

routes.MapRoute(

                "MoviesByReleaseDate",

                "movies/released/{year}/{month}",

                new {controller = "Movies", action = "ByReleaseDate"},

                new {year = @"/d{4}" , month = @"/d{2}"});

            //this is constraints, year must be have 4 number, month must be have 2 number

ASP.NET MVC Attribute Route Constraints

public class **RouteConfig**

    {

        public static void RegisterRoutes(**RouteCollection** routes)

        {

            routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

            routes.MapMvcAttributeRoutes();

            /\*

            routes.MapRoute(

                "MoviesByReleaseDate",

                "movies/released/{year}/{month}",

                new {controller = "Movies", action = "ByReleaseDate"},

                new {year = @"/d{4}" , month = @"/d{2}"});

            //this is constraints, year must be have 4 number, month must be have 2 number

            \*/

            routes.MapRoute(

                name: "Default",

                url: "{controller}/{action}/{id}",

                defaults: new { controller = "Home", action = "Index", id = **UrlParameter**.Optional }

            );

        }

    }

Go to:

public class **MoviesController** : **Controller**

[**Route**("movies/released/{year}/{month:regex(\\d{2}) : range(1,12)}")]

//use with attribute route

        public **ActionResult** ByReleaseDate(int year, int month)

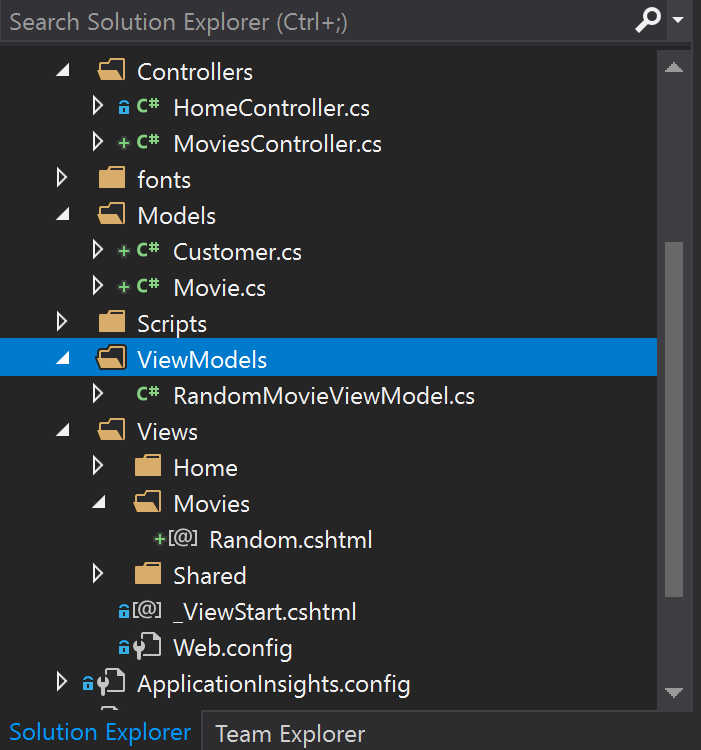
        {

            return Content(year + "/" + month);

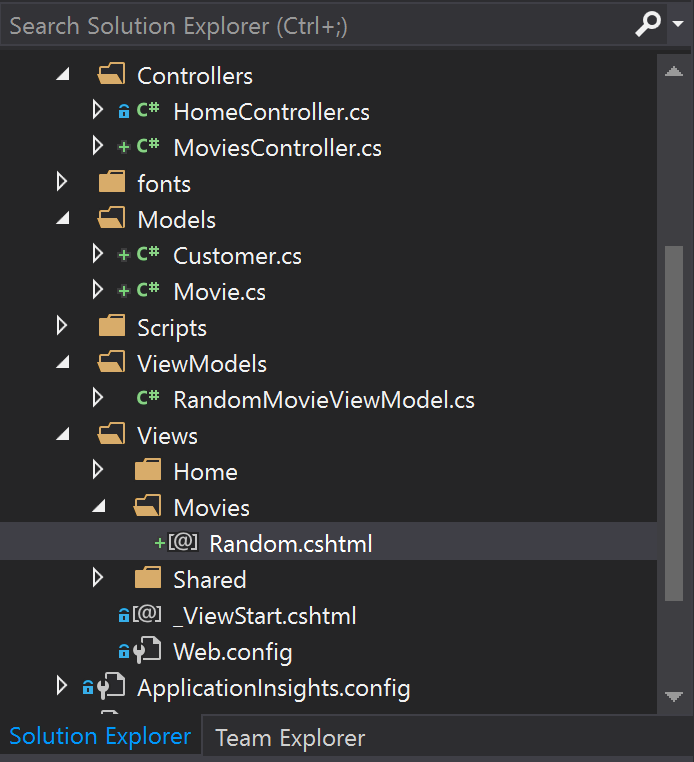
        }

**View**

**Passing Data To View**



* Create ViewModels folder and create class RandomMovieViewModel.cs with properties is what you want to transfer to Views-Movies-Random.cshtml
* public class **RandomMovieViewModel**
* {
* public **Movie** Movie { get; set; }
* public **List**<**Customer**> Customers { get; set; }
* }
* Go back to MoviesController add
* // GET: Movies/Random
* public **ActionResult** Random() //this is action (Views)
* {
* var movie = new **Movie** {Name = "Alibaba"};
* var customers = new **List**<**Customer**>
* {
* new **Customer**{Name = "Customer 1"},
* new **Customer**{Name = "Customer 2"}
* };
* var viewModel = new **RandomMovieViewModel**
* {
* Movie = movie,
* Customers = customers
* };
* return View(viewModel);
* }
* In View folder -> Random.cshtml add some code



@model  DemoVidly.ViewModels.**RandomMovieViewModel**

@{

    ViewBag.Title = "Random";

    Layout = "~/Views/Shared/\_Layout.cshtml";

}

<br/>

@\*

    This is comment

    On multiple lines

\*@

@{

    var className = Model.Customers.Count > 1 ? "alert alert-dismissible alert-info" : null;

}

<h2 class="@className">@Model.Movie.Name</h2>

@if (Model.Customers.Count == 0)

{

    <p> No one has to rented this movie before </p>

}

else

{

    <ul class="list-group">

        @foreach (var customer in Model.Customers)

        {

        <li class="list-group-item d-flex justify-content-between align-items-center">

            @customer.Name

            <span class="badge badge-primary badge-pill">14</span>

        </li>

        }

    </ul>

}

**Partial View**

Problem can’t create patial view

<https://stackoverflow.com/questions/40051981/packages-config-null-or-empty-package-id>

Create partial view name \_NarBar and

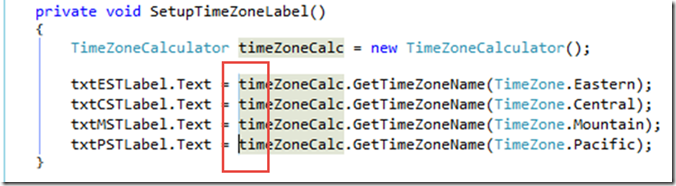
use code below in layout where you want to use partial view

@Html.Partial("\_NarBar")

Multi-cursor in Visual Studio

<https://blogs.msdn.microsoft.com/mschray/2014/09/03/multi-cursor-in-visual-studio/>

If the lines are consecutive select the location in the first line and then use Shift-Alt and the up and down arrow keys to select multiple lines.  In the image below see the light blue highlight before the time…. indicating these four rows are selected.

[](https://msdnshared.blob.core.windows.net/media/MSDNBlogsFS/prod.evol.blogs.msdn.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/01/56/23/metablogapi/5287.image_0A28BA36.png)

If you prefer use Shift-Alt and click multiple times for the row you want to select.

If you want more power control check out the free [multi-editing extension for Visual Studio](http://visualstudiogallery.msdn.microsoft.com/2beb9705-b568-45d1-8550-751e181e3aef).

**Adding-active-tag-to-navigation-list-in-an-asp-net-mvc**

<https://stackoverflow.com/questions/214583/adding-active-tag-to-navigation-list-in-an-asp-net-mvc-master-page>

I made myself a helper method to handle this type of thing. In the code behind of my master page (could be pushed of to an extension method ... probably a better approach), I put the following code.

protected string ActiveActionLinkHelper(string linkText, string actionName, string controlName, string activeClassName)

{

if (ViewContext.RouteData.Values["action"].ToString() == actionName &&

ViewContext.RouteData.Values["controller"].ToString() == controlName)

return Html.ActionLink(linkText, actionName, controlName, new { Class = activeClassName });

return Html.ActionLink(linkText, actionName, controlName);

}

Then, I just call it in my page like so:

<%= ActiveActionLinkHelper("Home", "Index", "Home", "selected")%>

In My code:

@{

var newClassCustomer = ViewContext.RouteData.Values["controller"]. ToString() == "Customers" ? "active" : null;

}

   <li class="nav-item @newClassCustomer">

        @Html.ActionLink("Customers", "Index", "**Customers**", new {id = "wasClick"}, new { @class = "nav-link" })

   </li>

<https://stackoverflow.com/questions/3399701/how-do-i-get-the-routedata-associated-with-the-parent-action-in-a-partial-view>

It looks like off of the ControllerContext (from which ViewContext derives) you can get the ParentActionViewContext:

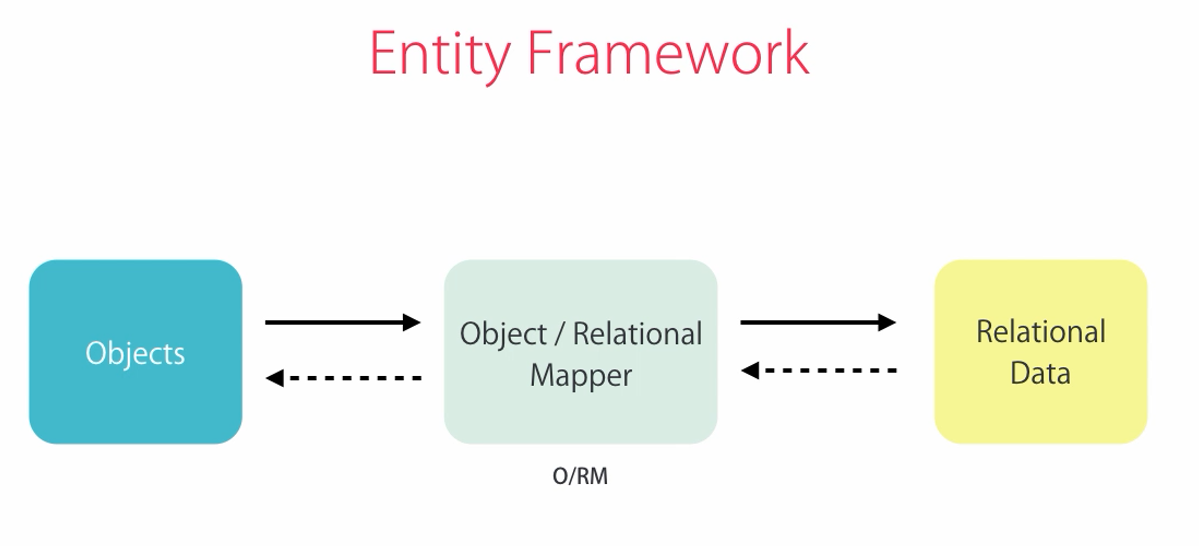
ViewContext.ParentActionViewContext.RouteData["controller"]

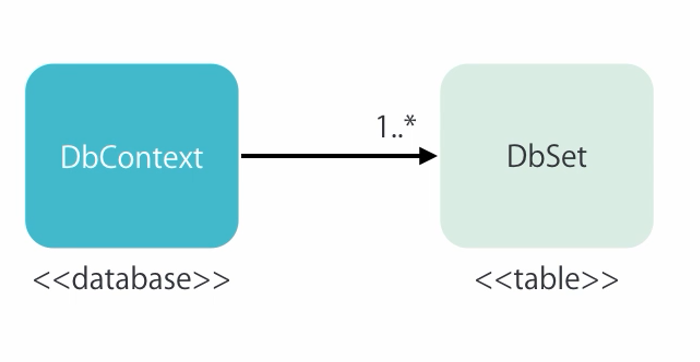
**Summary**

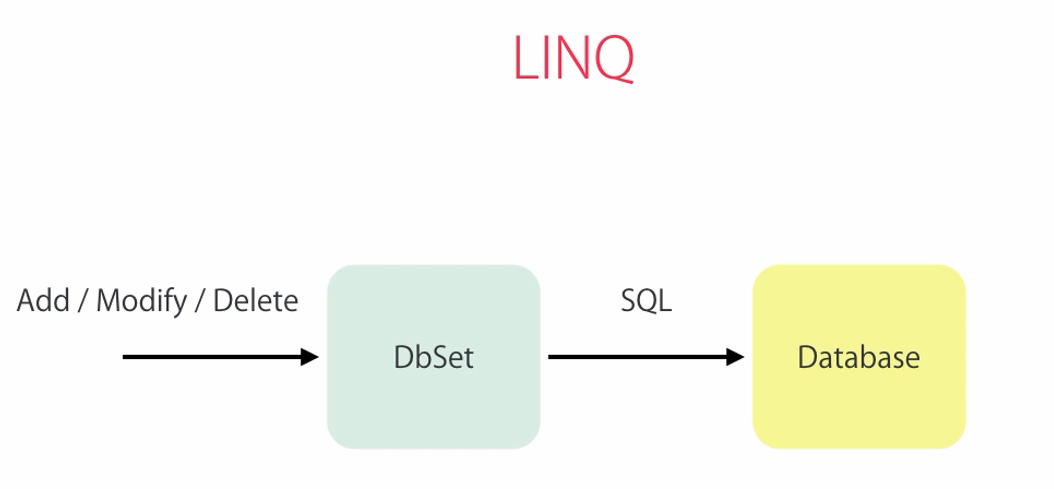
* Action Results
* Action Parameters
* Convention-based Routing
* Attribute Routing
* Razor View

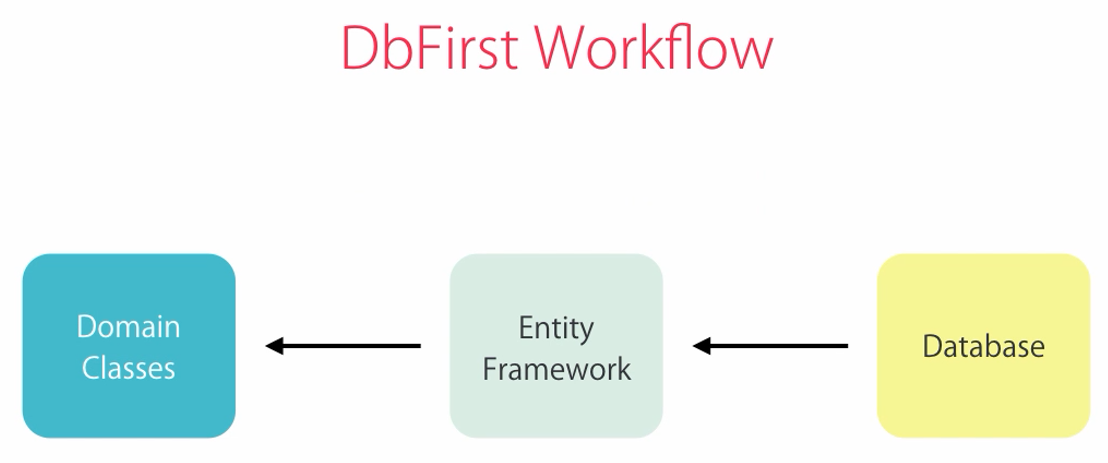
**Working with Data**

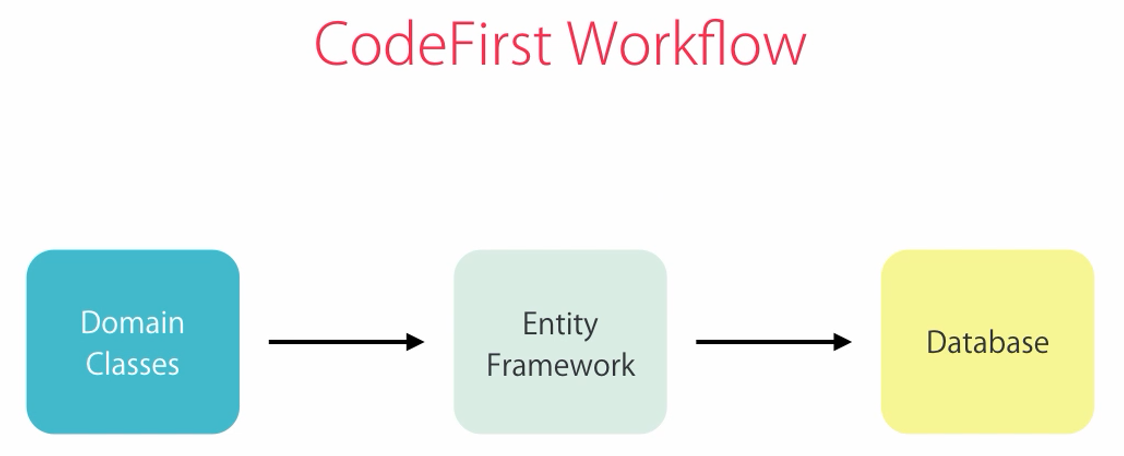
Entity Framework











Code-first Migrations  
add-migration <name>  
add-migration <name> -force (to overwrite the last migration)   
update-database

Seeding the Database  
Create a new empty migration and use the Sql method:  
Sql(“INSERT INTO …”)

Overriding Conventions – System.ComponentModel.DataAnnotations  
[Required]  
[StringLength(255)]  
public string Name { get; set; }

Querying Objects  
public class MoviesController   
{  
 private ApplicationDbContext \_context;

public MoviesController()  
 {  
 \_context = new ApplicationDbContext();  
 }

protected override Dispose()  
 {  
 \_context.Dispose();  
 }

public ActionResult Index()   
 {  
 var movies = \_context.Movies.ToList();  
 …   
 }  
}

LINQ Extension Methods  
\_context.Movies.Where(m => m.GenreId == 1)  
\_context.Movies.Single(m => m.Id == 1);  
\_context.Movies.SingleOrDefault(m => m.Id == 1);  
\_context.Movies.ToList();

Eager Loading  
\_context.Movies.Include(m => m.Genre);

# [HTML.ActionLink method](https://stackoverflow.com/questions/200476/html-actionlink-method)

<https://stackoverflow.com/questions/200476/html-actionlink-method>

Html.ActionLink(article.Title,

"Item", // <-- ActionMethod

"Login", // <-- Controller Name.

new { article.ArticleID }, // <-- Route arguments.

null // <-- htmlArguments .. which are none. You need this value

// otherwise you call the WRONG method ...

// (refer to comments, below).

)

-------------------------

When you want to load in Index View data from table joint with root table, must be use Eager Loading, use Eager Loading must be use Include ( expression ) in Controller, must be using System.Data.Entity

<tbody>

        @foreach (var customer in Model)

        {

            <tr>

                <td>@Html.ActionLink(customer.Name, "Detail", "**Customers**", new { id = customer.Id }, null)</td>

                <td>@customer.MembershipType.DiscountRate %</td>

            </tr>

        }

    </tbody>

private **ApplicationDbContext** \_context;

        public **MoviesController**()

        {

            \_context = new **ApplicationDbContext**();

        }

        protected override void Dispose(bool disposing)

        {

             \_context.Dispose();

        }

// GET: Customers

        public **ActionResult** Index()

        {

            var customers = \_context.Customers.Include(c => c.MembershipType).ToList();

            return View(customers);

        }

**Sử dụng dispose trong entity framework**

## Đảm bảo rằng kết nối đến CSDL đã được đóng

Để đảm bảo các kết nối đến CSDL đã được đóng và tài nguyên do chúng chiếm giữ đã được giải phóng, bạn phải đảm bảo đối tượng context phải bị hủy. Đó là vì sao chúng ta sửa lại phương thức Dispose ở cuối lớp StudentController trong file StudentController.cs, như bạn thấy trong ví dụ dưới đây:

protected override void Dispose(bool disposing)

{

    db.Dispose();

    base.Dispose(disposing);

}

Lớp Controller cơ sở đã implement IDisposable, do vậy đoạn code này đơn giản thêm một phương thức override lại Dispose(bool) để thực hiện việc giải phóng đối tượng context.

<https://blog.jongallant.com/2012/10/do-i-have-to-call-dispose-on-dbcontext/>

<https://stackoverflow.com/questions/15666824/entity-framework-and-context-dispose>

using (var context = new SchoolDBEntities())

{

var L2EQuery = from st in context.Students

where st.StudentName == "Bill"

select st;

var student = L2EQuery.FirstOrDefault<Student>();

}

Đầu tiên bạn phải tạo một đối tượng của lớp context đó là SchoolDBEntities. Bạn nên khởi tạo nó ở using() để một khi nó bên ngoài phạm vi thì nó sẽ tự động gọi phương thức Dispose() của DbContext. Cả hai cú pháp trên, context đều trả về IQueryable.

Building Form

@using (Html.BeginForm("Create", "**Customers**"))

{

}

Html.BeginForm("Create", "**Customers**") is <form> tag of html

Out of using block, this will call the Depose Object and close tag </form>

Set label for form

[**Display**(Name = "Date of Birth")]

        public DateTime? Birthday { get; set; }

@using (Html.BeginForm("Create", "**Customers**"))

{

    <div class="form-group">

        @Html.LabelFor(m => m.Name)

        @Html.TextBoxFor(m => m.Name, new { @class = "form-control" })

    </div>

    <div class="form-group">

        @Html.LabelFor(m => m.Birthday)

        @Html.TextBoxFor(m => m.Birthday, new { @class = "form-control" })

    </div>

    <div class="form-check">

        <label class="form-check-label">

            @Html.CheckBoxFor(m => m.IsSubscribedToNewsLetter, new { @class = "form-check-input" }) Subcribed to Newsletter?

        </label>

    </div>

}