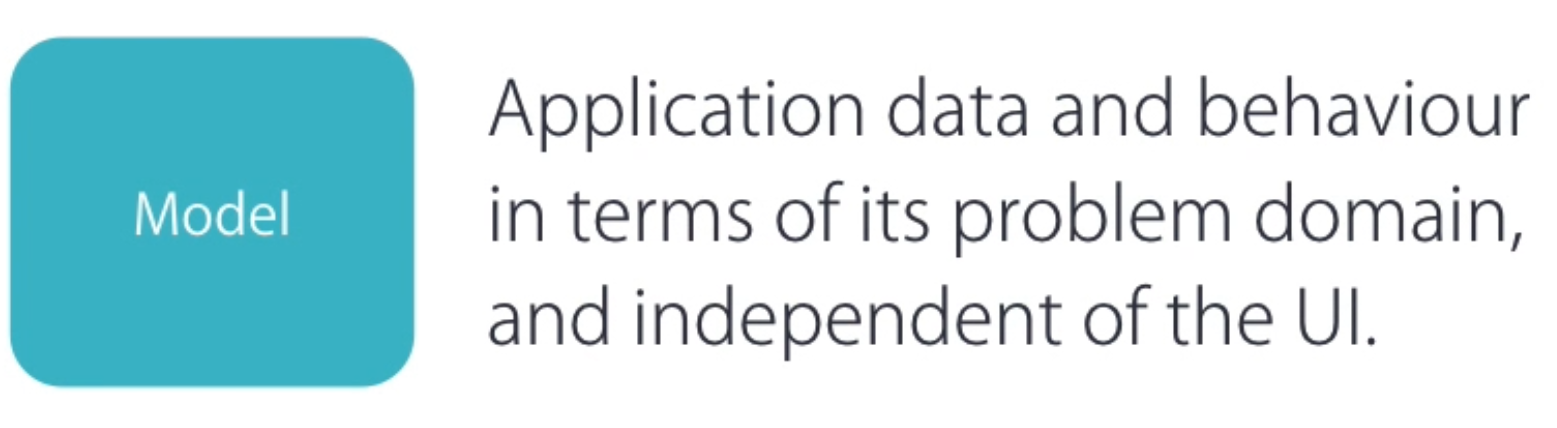
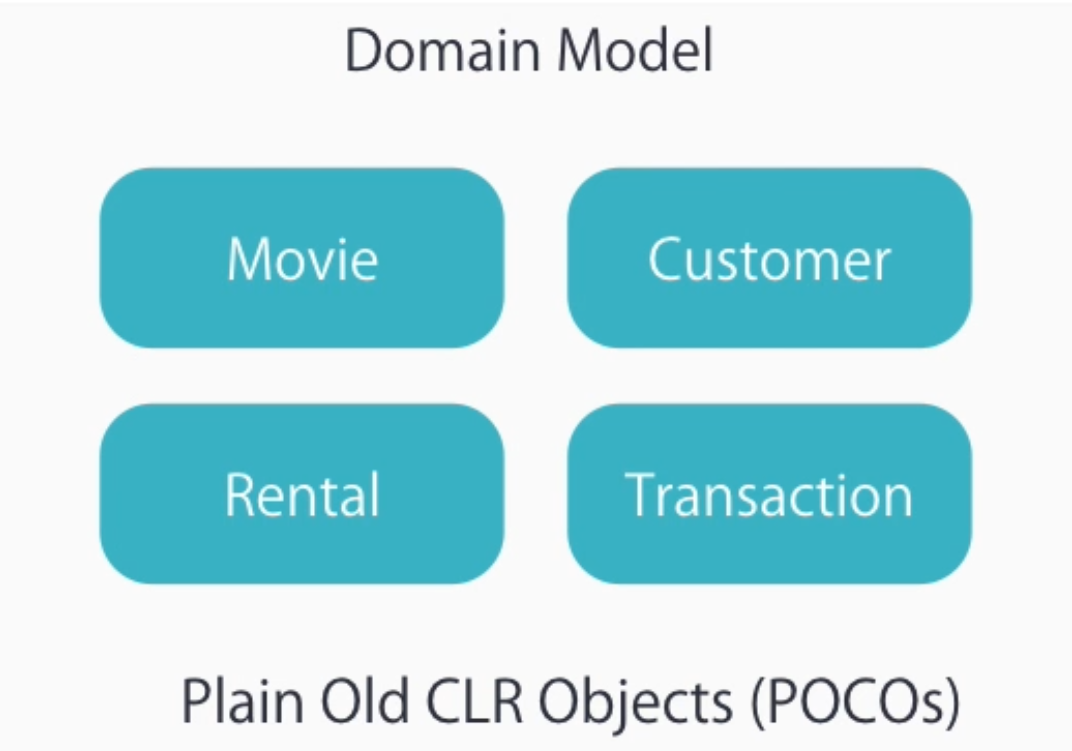
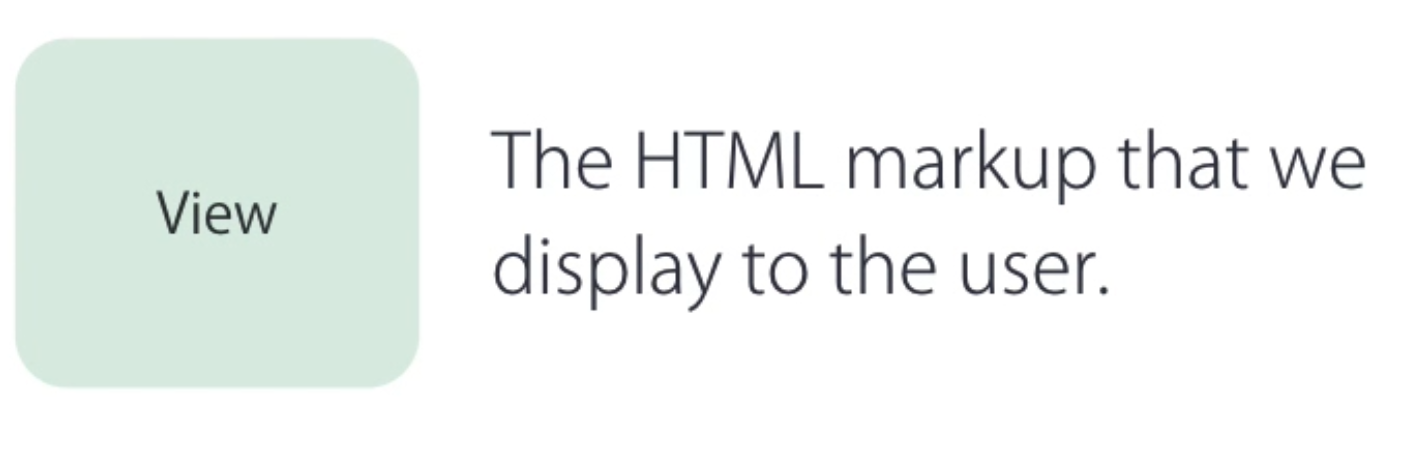
**MVC Pattern**

**M – Model** : is class, …

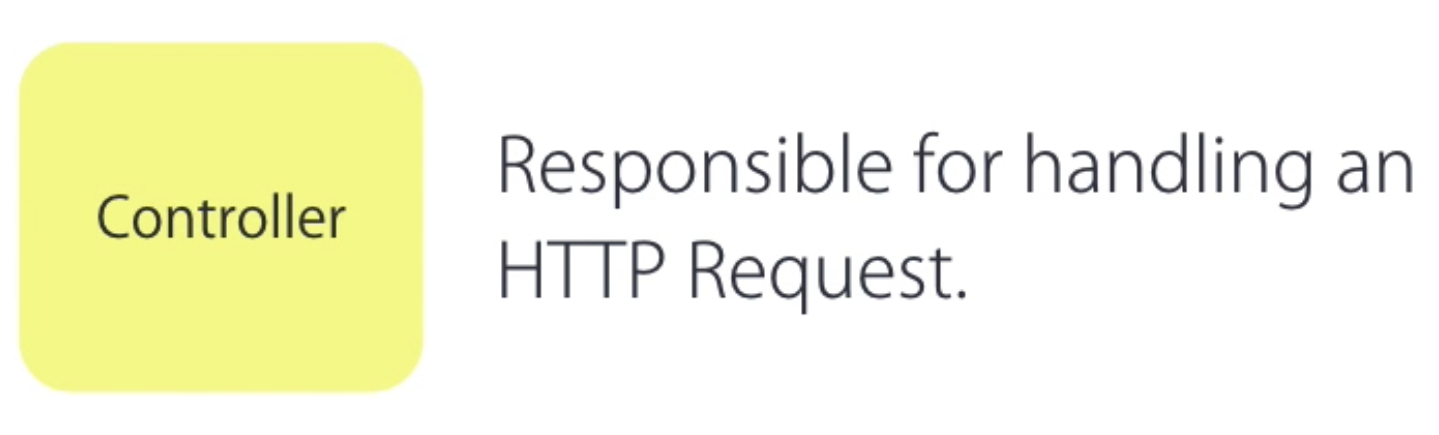




**V – View:**

****

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(

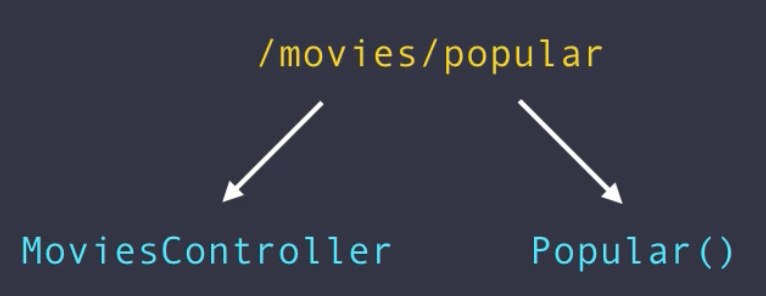
                name: "Default",

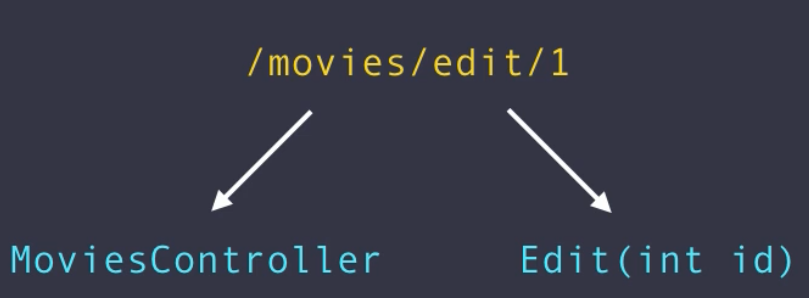
                url: "{controller}/{action}/{id}",

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

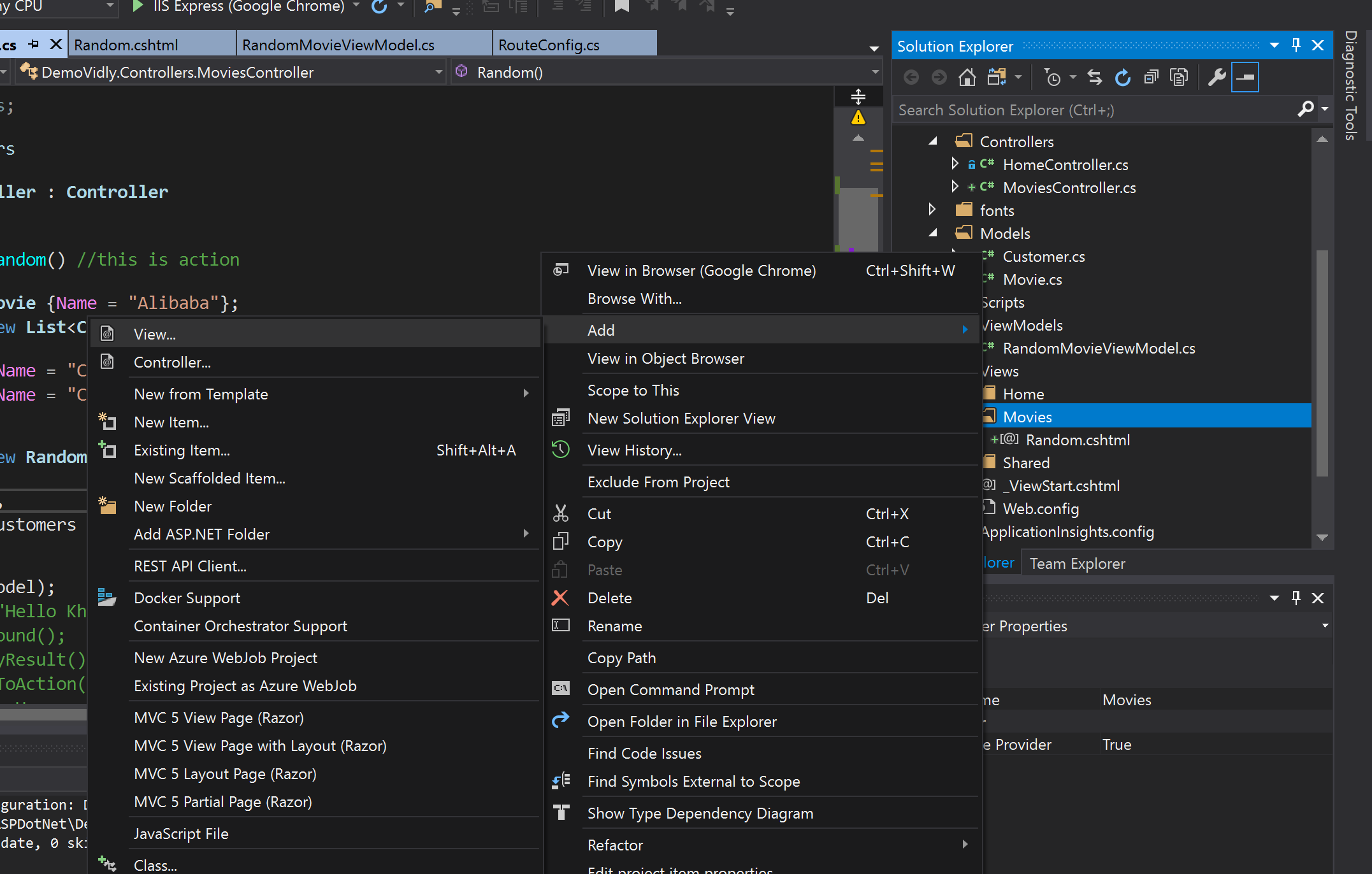
            );

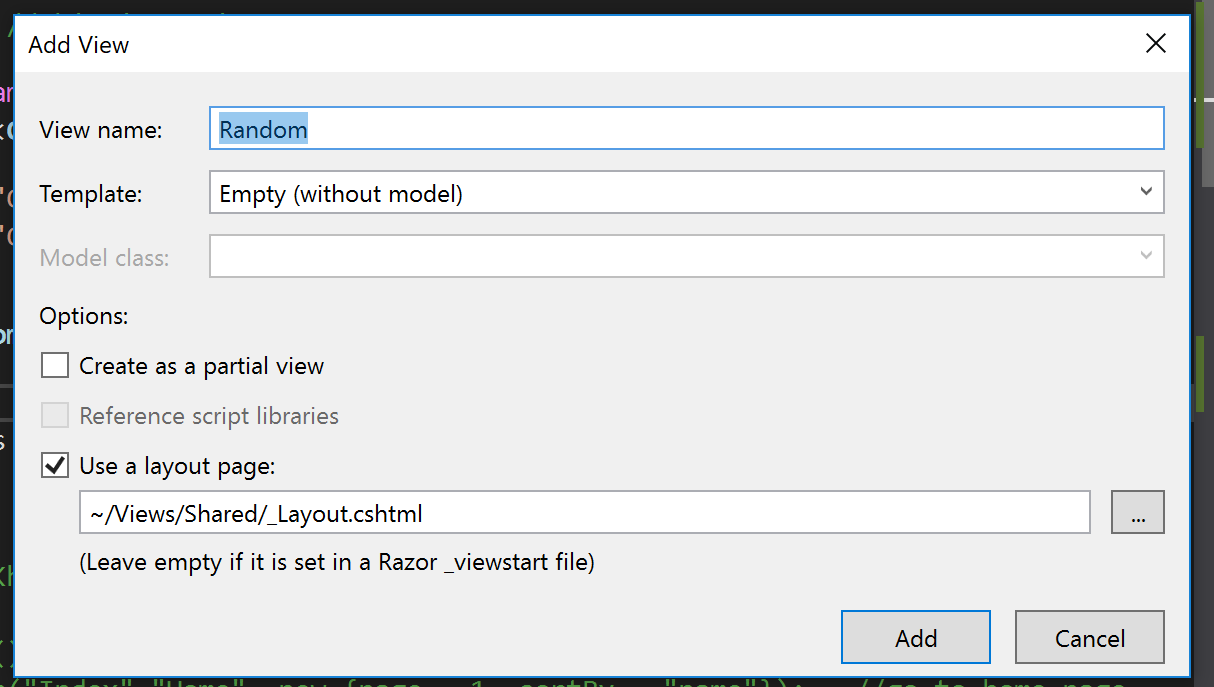
        }





* 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";

}

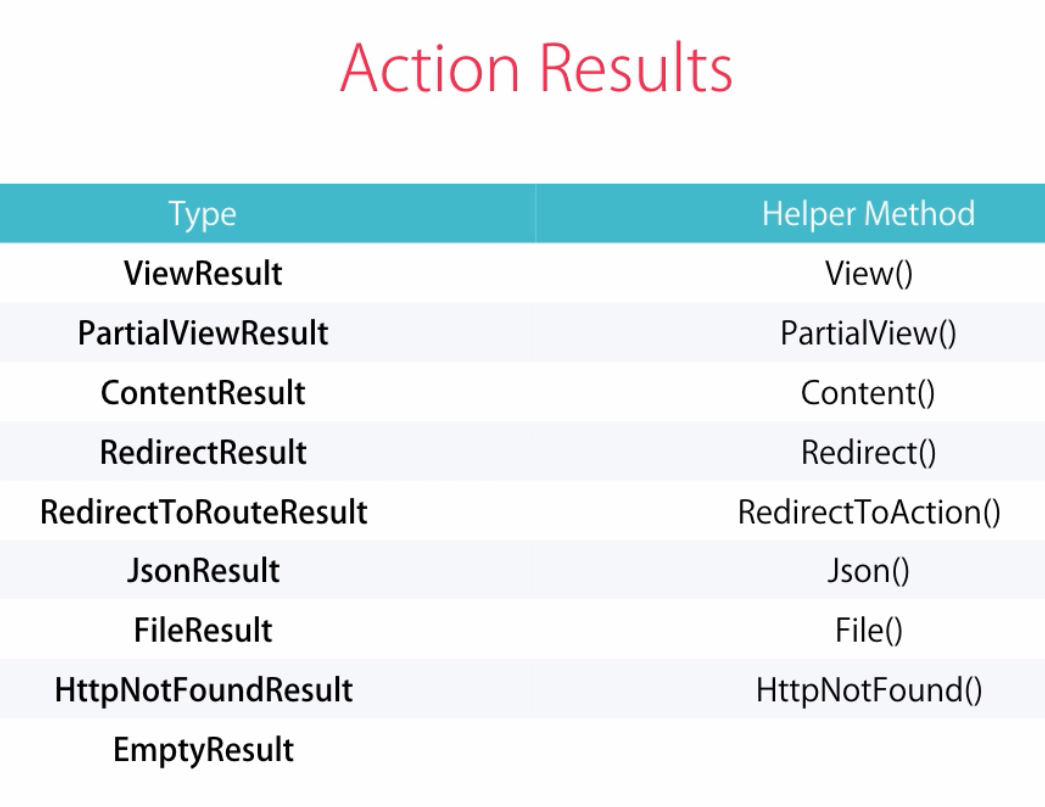
<h2>@Model.Name</h2> // transfer param

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,

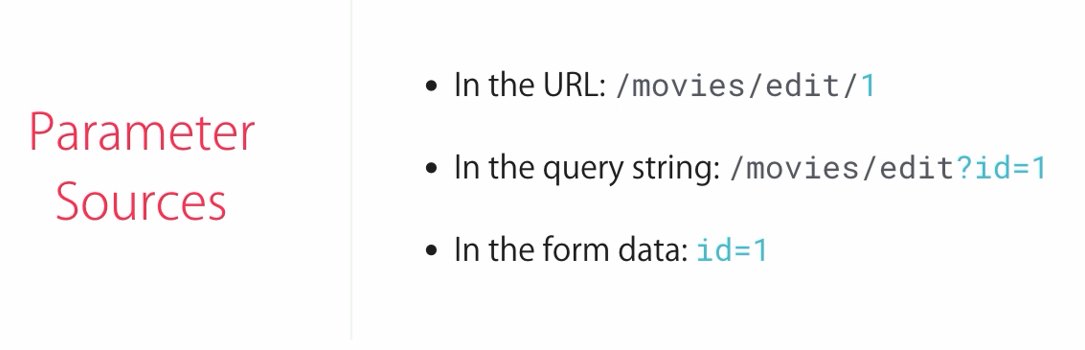
new {page = 1, sortBy =  "name"} : active param Url

        }

    }

}

**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 }

            );

        }

    }

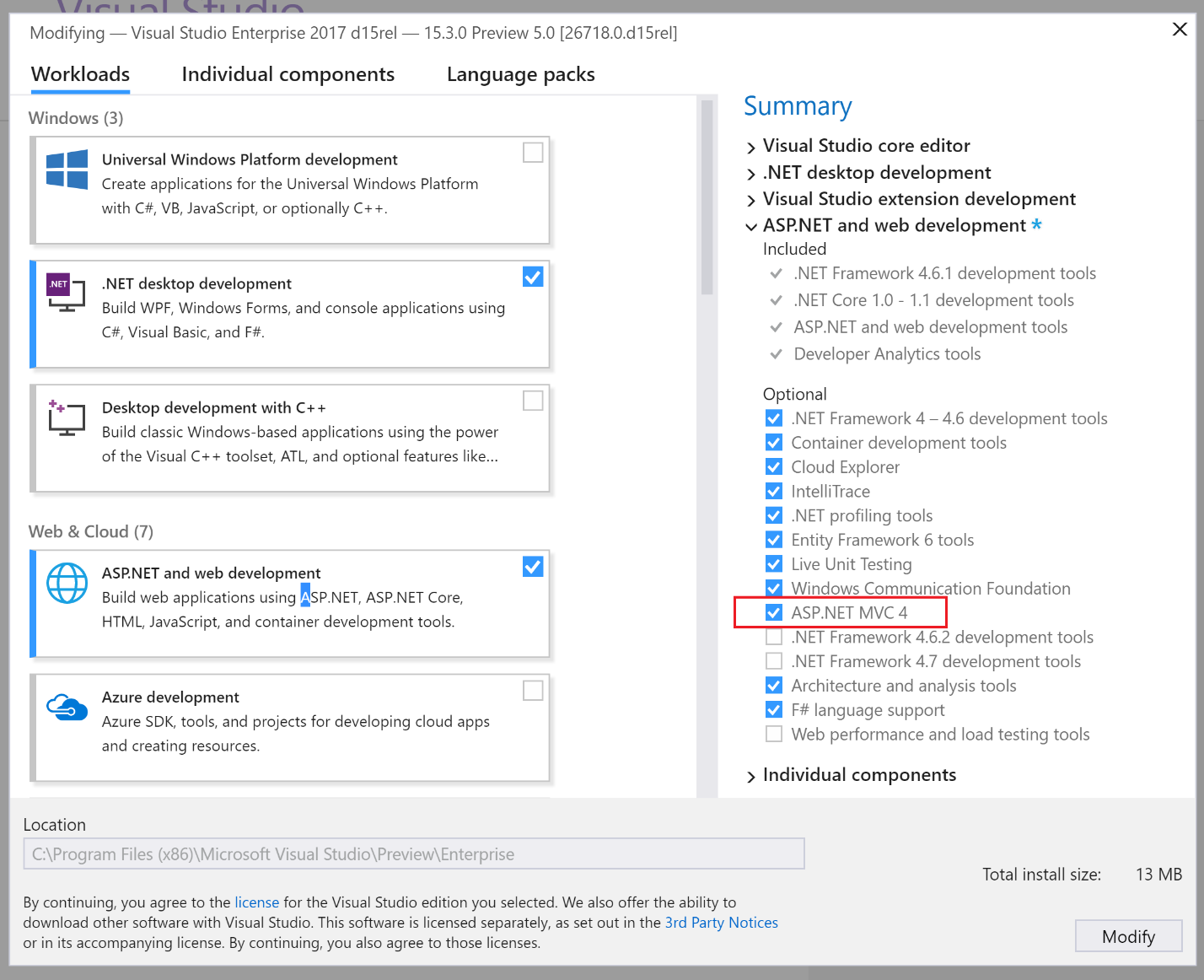
**mvcaction4 - >**

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)

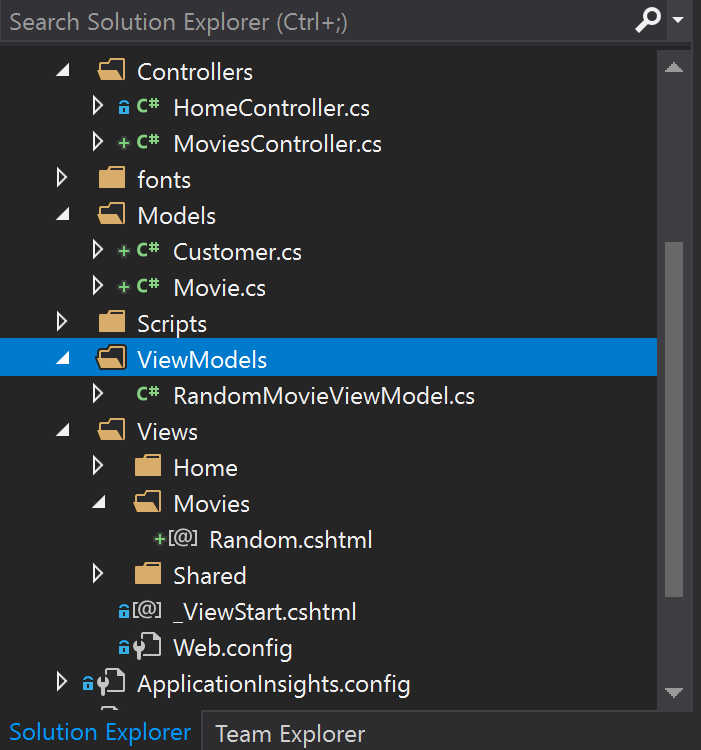
        {

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

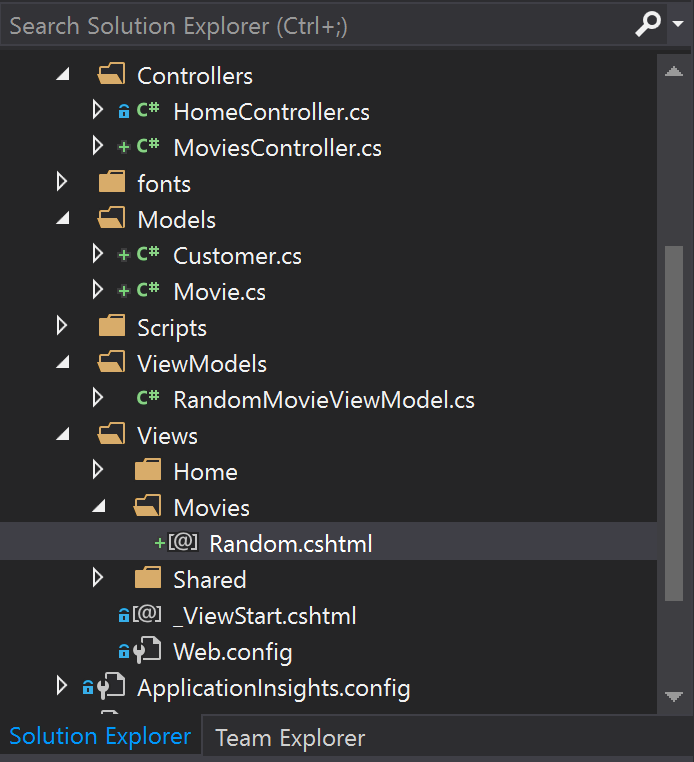
        }

**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

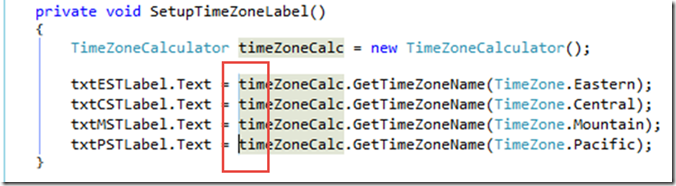
use code below in layout where you want to use partial view

@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:

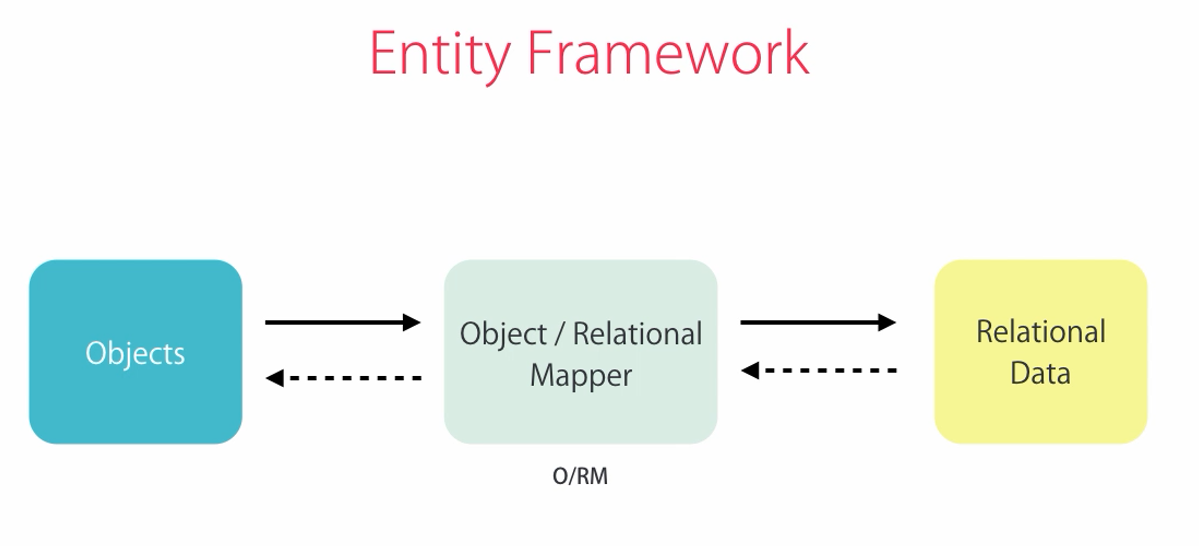
ViewContext.ParentActionViewContext.RouteData["controller"]

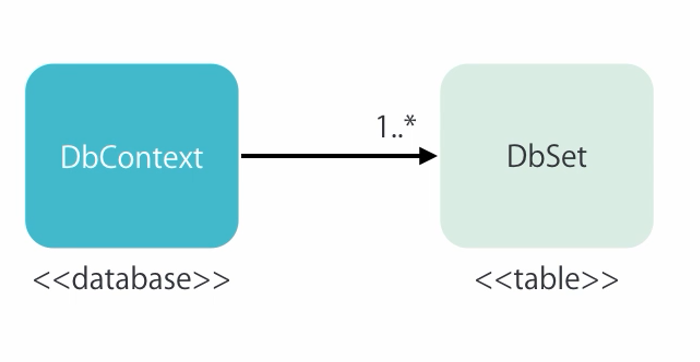
**Summary**

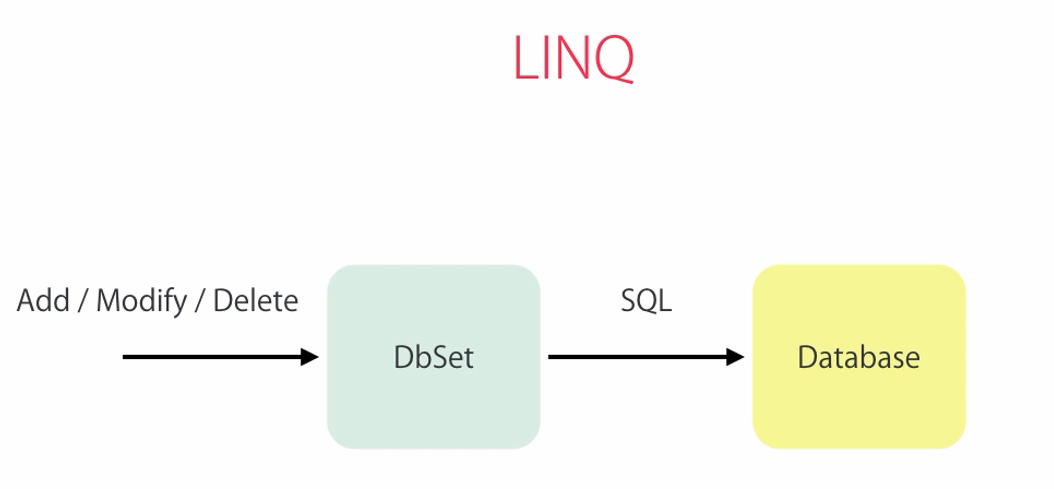
* Action Results
* Action Parameters
* Convention-based Routing
* Attribute Routing
* Razor View

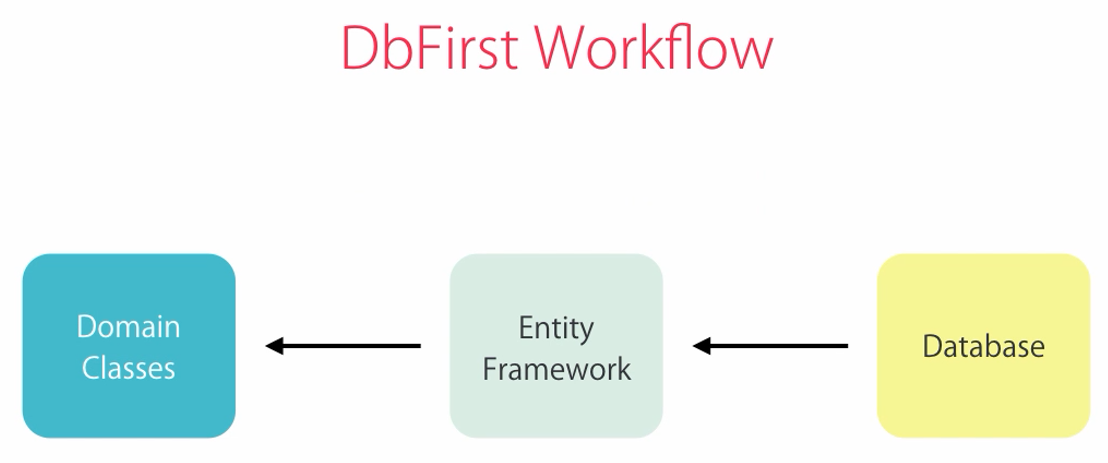
**Working with Data**

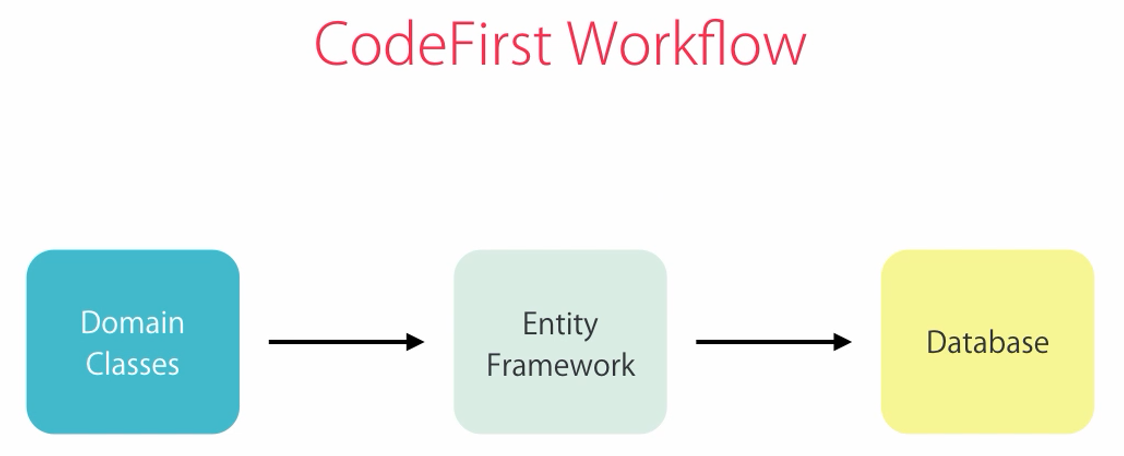
Entity Framework











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("Save","**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 in Model class**

[**Display**(Name = "Date of Birth")]

        public DateTime? Birthday { get; set; }

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

{

    <div class="form-group">

        @Html.LabelFor(m => m.Customer.Name)

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

    </div>

    <div class="form-group">

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

        @Html.TextBoxFor(m => m.Customer.Birthday, "{0: dd/MM/yyyy}", new { @class = "form-control" })

    </div>

    <div class="form-group">

        @Html.LabelFor(m => m.Customer.MembershipTypeId)

        @Html.DropDownListFor(m => m.Customer.MembershipTypeId, new **SelectList**(Model.MembershipTypes, "Id", "Name"), "Select Membership Type", new { @class = "form-control" })

    </div>

    <div class="form-check form-group">

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

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

        </label>

    </div>

    @Html.HiddenFor(m => m.Customer.Id) //get Id to form tranfer to controller

    <button type="submit" class="btn btn-primary">

        Save

    </button>

}

**Drop Down List**

<div class="form-group">

        @Html.LabelFor(m => m.Customer.MembershipTypeId)

        @Html.DropDownListFor(m => m.Customer.MembershipTypeId, new **SelectList**(Model.MembershipTypes, "Id", "Name"), "Select Membership Type", new { @class = "form-control" })

    </div>

**Set display for label on model class**

[**Display**(Name = "Membership Types")]

        public byte MembershipTypeId { get; set; }

        [**Display**(Name = "Date of Birth")]

        public DateTime? Birthday { get; set; }

**Ienumerable<> or List<>**

If you want to transfer more than one object model to view must be create folder ViewModel and create class include 2 object -> 2 properties for use to View

namespace MovieShop.ViewModels

{

    public class NewCustomerViewModel

    {

        public IEnumerable<**MembershipType**> MembershipTypes { get; set; }

//public List<MembershipType> MembershipTypes { get; set; }

public **Customer** Customer { get; set; }

        public string Title

        {

            get

            {

                if (Customer != null && Customer.Id != 0)

                {

                    return "Edit Customer";

                }

                return "New Customer";

            }

        }

    }

}

* Because in the View we don’t need method of List<> like Add, Remove, Insert. We need only the loop iterates ( foreach …) so we use IEnumerable, later if you want to add you can use the ToList method because the Collection interface IEnumerable

**Update model controller**

[**HttpPost**] //Only post

        public **ActionResult** Save(**Customer** customer)  //model binding

        {

            if (customer.Id == 0)

            {

                \_context.Customers.Add(customer);   //This save on memory not on database, after that mus be SaveChange()

            }

            else

            {

                var customerInDb = \_context.Customers.Single(c => c.Id == customer.Id);

         /\*

         TryUpdateModel(customerInDb);   //this way will open the security hall and check all properties of Model

         TryUpdateModel(customerInDb, "", new string[]{"Name", "Email"});

         //this way change properties Name and Email, but later if you change the properties on model class must be manual fix in here

         \*/

//Mapper.Map(customer, customerInDb); //Install-Package AutoMapper

                customerInDb.Name = customer.Name;

                customerInDb.Birthday = customer.Birthday;

                customerInDb.MembershipTypeId = customer.MembershipTypeId;

                customerInDb.IsSubscribedToNewsLetter = customer.IsSubscribedToNewsLetter;

            }

            \_context.SaveChanges();   //Must be use it to save on database

            return RedirectToAction("Index", "**Customers**");

        }

**Form Controller**

Tranfer to from value of MembershipTypes use for droplist, no need to tranfer the customer ( when edit must be transfer also customer )

public **ActionResult** CustomerForm()

        {

            var membershipTypes = \_context.MembershipTypes.ToList();

            var viewModel = new **CustomerFormViewModel**

            {

                MembershipTypes = membershipTypes,

                Customer = new **Customer**()

//this will make default value like Id = 0;

/\*

 instead of having a straight integer ID field, make that an autonumber ([IDENTITY](https://msdn.microsoft.com/en-GB/library/ms186775.aspx)) column. If you make it IDENTITY, you can leave the product.Id = 0 and EF will generate it automatically. After dbContext.SaveChanges has been called, product.PRODUCT\_ID will have been set for you, if you need to ID of the created object.

\*/

            };

            return View(viewModel);

        }

Or if you don’t want they make auto default all value can do like this in Form.cshtml

@Html.Hidden("Customer.Id", (Model.Customer != null) ? Model.Customer.Id : 0)

 //this will add Customer.Id = 0 from the beginning

Or in ViewModel create class **MovieFormViewModel** get all properties you need for that form, create contructor to add model when create object

namespace MovieShop.ViewModels

{

    public class **MovieFormViewModel**

    {

        public IEnumerable<**Genre**> Genres { get; set; }

        public int? Id { get; set; }

//get value from class movie, so can make it null

        [**Required**]

        public string Name { get; set; }

        [**Required**]

        [**Display**(Name = "Genre")]

        public byte? GenreId { get; set; }

        [**Display**(Name = "Realease Date")]

        [**Required**]

        public DateTime? ReleaseDate { get; set; }

        [**Display**(Name = "Number In Stock")]

        [**Required**]

        [**Range**(1, 20)]

        public byte? NumberInStock { get; set; }

        public string Title

        {

            get

            {

                if (Id != 0)

                {

                    return "Edit Movie";

                }

                return "New Movie";

            }

        }

        public **MovieFormViewModel**()

        {

            Id = 0;

        }

        public **MovieFormViewModel**(**Movie** movie)

        {

            Id = movie.Id;

            GenreId = movie.GenreId;

            Name = movie.Name;

            ReleaseDate = movie.ReleaseDate;

            NumberInStock = movie.NumberInStock;

        }

    }

}

public **ActionResult** MovieForm()

        {

**MovieFormViewModel** movieForm = new **MovieFormViewModel**

            { //Contructor no param

                Genres = \_context.Genres.ToList(),

            };

            return View(movieForm);

        }

        public **ActionResult** Edit(int id)

        {

            var movie = \_context.Movies.SingleOrDefault(m => m.Id == id);

            if (movie == null)

            {

                return HttpNotFound();

            }

**MovieFormViewModel** movieForm = new **MovieFormViewModel**(movie)

            { //Contructor with param

                Genres = \_context.Genres.ToList()

            };

            return View("MovieForm", movieForm);

}

**Edit Controller**

Tranfer to View with 2 properties (Membership Type and Customer)

public **ActionResult** Edit(int id)

   {

       var customer = \_context.Customers.SingleOrDefault(c => c.Id == id);

       if (customer == null)

          {

                return HttpNotFound();

          }

        var viewModel = new **CustomerFormViewModel**

            {

                Customer = customer,

                MembershipTypes = \_context.MembershipTypes.ToList()

            };

        return View("CustomerForm", viewModel);

    }

<https://docs.microsoft.com/en-us/ef/core/what-is-new/ef-core-2.0>

## Change Tracking

### Attach can track a graph of new and existing entities

**EF Core supports automatic generation of key values through a variety of mechanisms**. When using this feature, **a value is generated if the key property is the CLR default--usually zero or null.** This means that a graph of entities can be passed to DbContext.Attach or DbSet.Attach and EF Core will mark those entities that have a key already set as Unchanged while those entities that do not have a key set will be marked as Added. This makes it easy to attach a graph of mixed new and existing entities when using generated keys. DbContext.Update and DbSet.Update work in the same way, except that entities with a key set are marked as Modified instead of Unchanged.

**Validation**

1. Add DataAnnotation in model class (object class)

[**Required**]

        [**StringLength**(255)]

        public string Name { get; set; }

1. Use ModelState in action in controller to check validation

public **ActionResult** Save(**Customer** customer)  //model binding

        {

            if (!ModelState.IsValid)

            {

                var viewModel = new **CustomerFormViewModel**

                {

                    Customer = customer,

                    MembershipTypes = \_context.MembershipTypes.ToList(),

                };

                return View("CustomerForm", viewModel);

            }

…

}

1. Set for validate in form and make CSS

<div class="form-group">

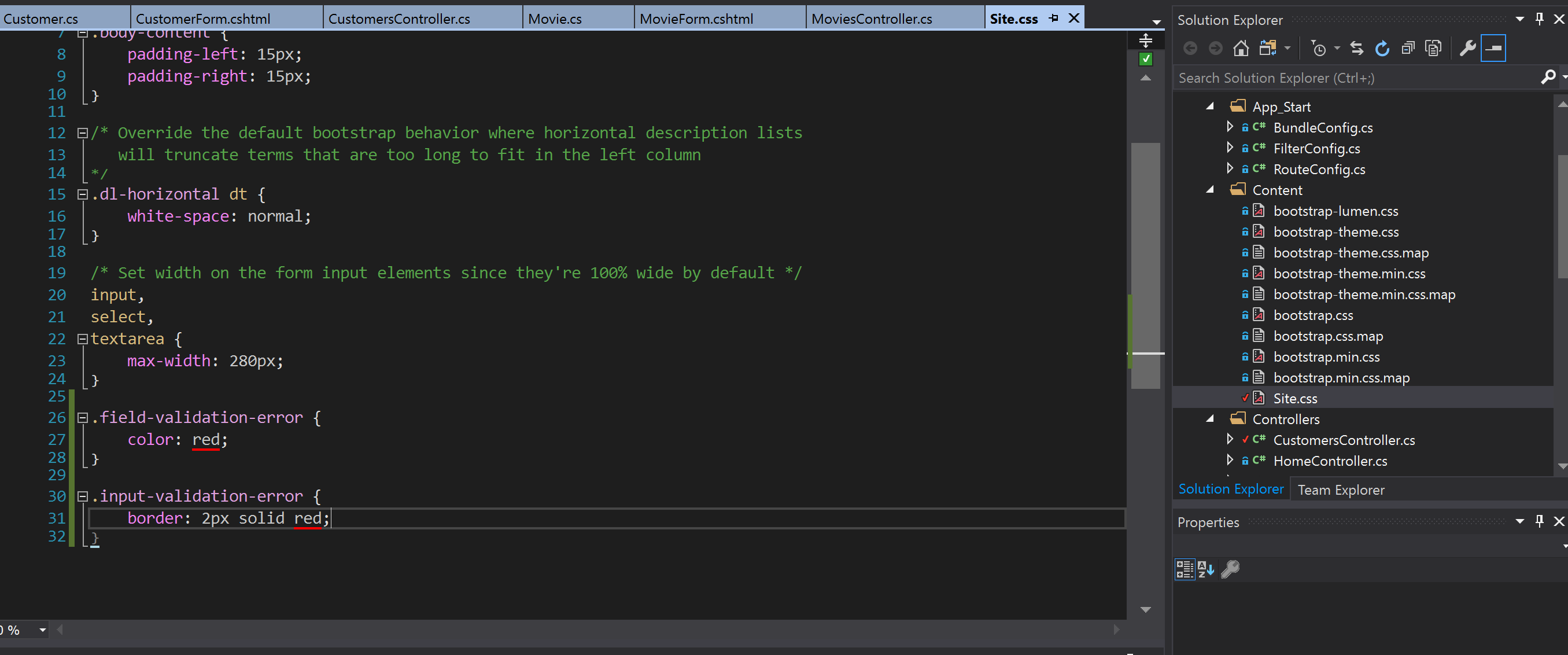
        @Html.LabelFor(m => m.Customer.Name)

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

        @Html.ValidationMessageFor(m => m.Customer.Name) // validate Name

</div>

CSS make in Content folder -> Site.CSS



Data Annotations  
• [Required]  
• [StringLength(255)]  
• [Range(1, 10)]  
• [Compare(“OtherProperty”)]  
• [Phone]  
• [EmailAddress]  
• [Url]  
• [RegularExpression(“…”)]

If you want to custom the error message:

[**Required**(ErrorMessage = "Please enter customer's name.")]

        [**StringLength**(255)]

        public string Name { get; set; }

**Custom Validation**

1. Create class model name Min18YearOfAnMember: To check brithday must be more than 18 year old, inherit class ValidationAttribute Of DataAnnotation

namespace MovieShop.Models

{

    public class **Min18YearsIfAMember** : **ValidationAttribute**

    {

        protected override **ValidationResult** IsValid(object value,

**ValidationContext** validationContext)

        {

            var customer = (**Customer**) validationContext.ObjectInstance;

            //it give we the way to access Customer object (Customer class  have properties use this validation)

            if (customer.MembershipTypeId == 0 ||

customer.MembershipTypeId == 1)

            {

                return **ValidationResult**.Success;

            }

            if (customer.Birthday == null)

            {

                return new **ValidationResult**("Birthday is required");

            }

            var age = DateTime.Now.Year - customer.Birthday.Value.Year;

            if (age <= 18)

            {

                return new **ValidationResult**("Customer should be at

least 18 year old");

            }

            return **ValidationResult**.Success;

        }

    }

}

//if (customer.MembershipTypeId == 0 || customer.MembershipTypeId == 1)

  if (customer.MembershipTypeId == **Membershiptype**.Unknown || customer.MembershipTypeId == **Membershiptype**.PasAsYouGo)

Can change the 0 and 1 by the static file in class like below if you don’t want to use enum, if use enum must be unbox type (byte)Membershiptype.Unknown

namespace MovieShop.Models

{

    public class **Membershiptype**

    {

        public byte Id { get; set; }

        public short SignUpFee { get; set; }

        public byte DurationInMonth { get; set; }

        public byte DiscountRate { get; set; }

        public string Name { get; set; }

        public static readonly byte Unknown = 0;

        public static readonly byte PasAsYouGo = 1;

    }

}

1. Apply this validation at Customer Model Class

[**Display**(Name = "Date of Birth")]

        [**Min18YearsIfAMember**]

        public DateTime? Birthday { get; set; }

1. Add message validation in View Form

<div class="form-group">

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

    @Html.TextBoxFor(m => m.Customer.Birthday, "{0: dd/MM/yyyy}",

new { @class = "form-control" })

    @Html.ValidationMessageFor(m => m.Customer.Birthday)

</div>

1. Make All validation on the top of form (Option)

@Html.ValidationSummary()      @\*Make All validation on the top of form\*@

@Html.ValidationSummary(true, "Please fix the error below")

**Client-side Validation**

* Immediate feedback
* No waste of server-side resources

Client-side Validation working with:

using System.ComponentModel.DataAnnotations;

[**Required**(ErrorMessage = "Please enter customer's name.")]

        [**StringLength**(255)]

        public string Name { get; set; }

Add in CustomerForm.cshtml

@section scripts

{

    @Scripts.Render("~/bundles/jqueryval")

}

// "~/bundles/jqueryval" get from BundleConfig.cs in App\_Start bundles.Add(new **ScriptBundle**("~/bundles/jqueryval").Include(

                        "~/Scripts/jquery.validate\*"));

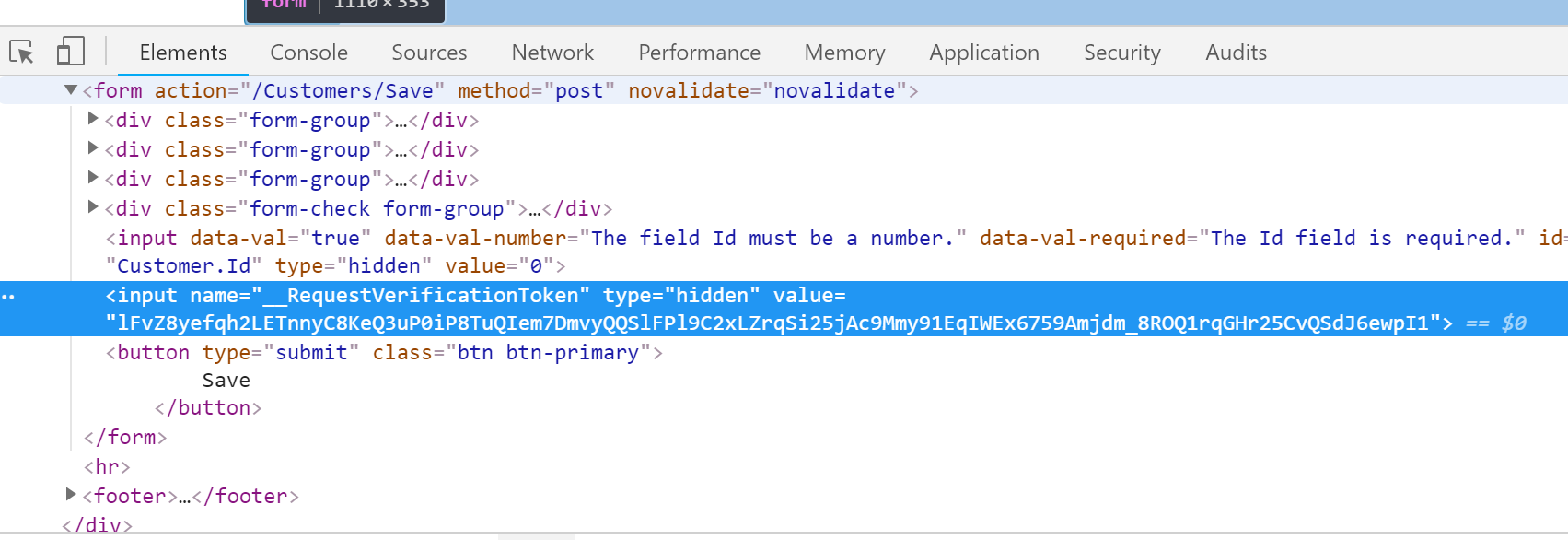
**CSRF – Cross-Site Request Forgery**

<https://toidicodedao.com/2016/11/29/csrf-cu-lua-ngoan-muc/>

In form use code: ( in CustomerForm.cshtml )

@Html.AntiForgeryToken()

This code will add the the value token in cookies, use it to validation form



Now we already have the token value for validation,

Add ValidateAntiForgeryToken in CustomerController.cs for method want to Post

[**HttpPost**] //Only post

        [**ValidateAntiForgeryToken**]  //Check must be post with value token

        public **ActionResult** Save(**Customer** customer)  //model binding

        {

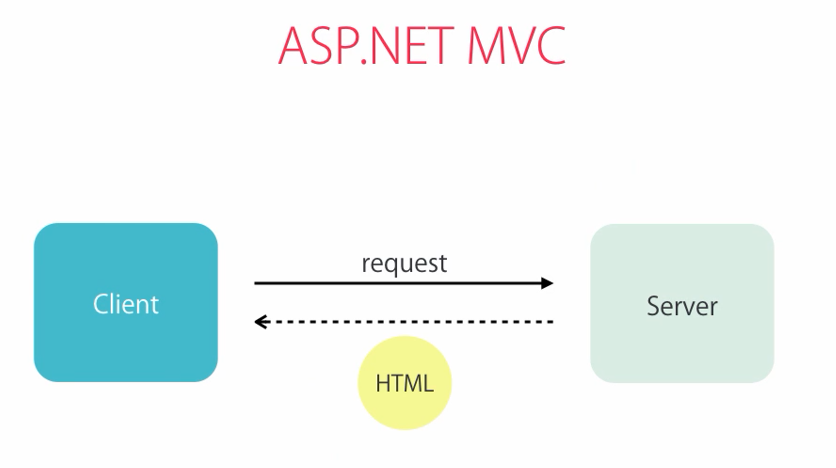
……

}

**SUMMARY**

* Data Annotations
* Custom Validation
* Client-side Validation
* Anti-forgery Tokens

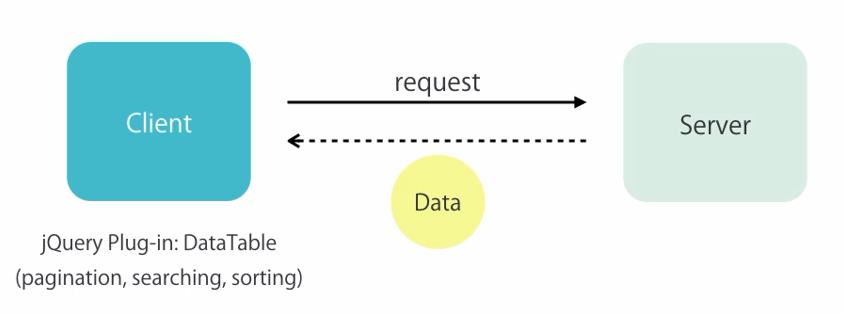
**ASP.NET MVC and ASP.NET Web API**



Client : View ( User see )

Request: Controller ( User want and ask the request to server )

HTML: Model, data ( server give back raw data and will generating markup in the client)



**Benefits (of gennerating markup on the client)**

* Less server resources (improve scalability)
* Less bandwidth (improve performance)
* Support for a broad range of clients (use also for web, mobile, table )

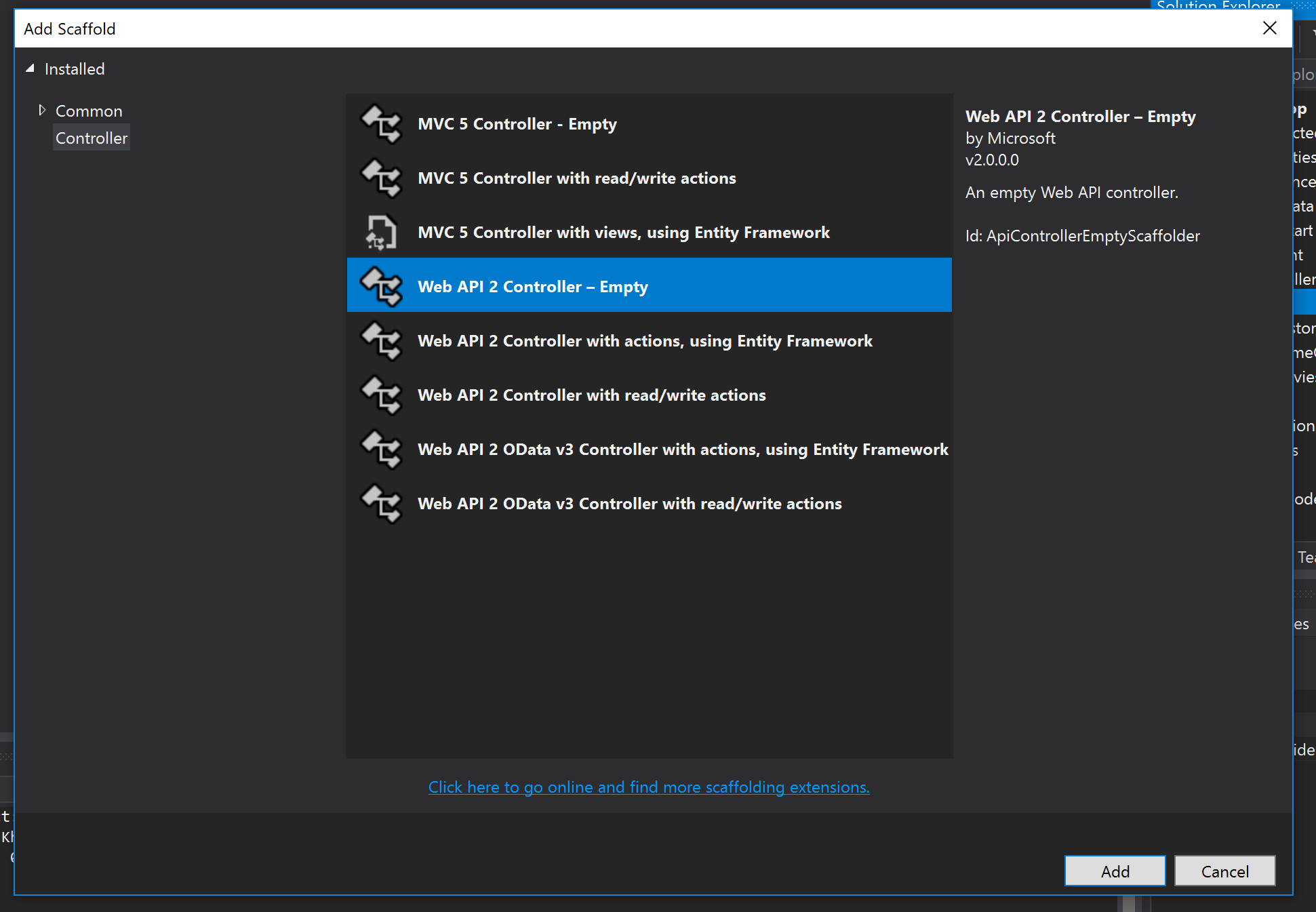
|  |  |
| --- | --- |
| C:\Users\Mai Khanh\Pictures\Saved Pictures\WebServerAPI.PNG | C:\Users\Mai Khanh\Pictures\Saved Pictures\APIServer.PNG |

**RESTful Convention**

REST = **Re**presentational **S**tate **T**ransfer



1. Create API folder in Controllers folder
2. From API folder, add Controller with Web API 2 Controller –Empty



1. Add the following lines to Application\_Start mothod ( in Global.asax.cs )

GlobalConfiguration.Configure(WebApiConfig.Register)

namespace MovieShop

{

    public class **MvcApplication** : System.Web.**HttpApplication**

    {

        protected void Application\_Start()

        {

// add config Web API

            GlobalConfiguration.Configure(WebApiConfig.Register);

**AreaRegistration**.RegisterAllAreas();

**FilterConfig**.RegisterGlobalFilters(GlobalFilters.Filters);

**RouteConfig**.RegisterRoutes(**RouteTable**.Routes);

**BundleConfig**.RegisterBundles(BundleTable.Bundles);

        }

    }

}

1. In API class

namespace MovieShop.Controllers.API

{

    public class **CustomersController** : **ApiController**

    {

        private **ApplicationDbContext** \_context;

        public **CustomersController**()

        {

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

        }

        //GET /api/customers

        public IEnumerable<**Customer**> GetCustomers()

        {

            return \_context.Customers.ToList();

        }

        //GET /api/customers/1

        public **Customer** GetCustomer(int id)

        {

       var customer = \_context.Customers.SingleOrDefault(c => c.Id == id);

            if (customer == null)

            {

                throw new **HttpResponseException**(HttpStatusCode.**NotFound**);

            }

            return customer;

        }

        //POST /api/customers

        [**HttpPost**]

        public **Customer** CreateCustomer(**Customer** customer)

//can rename PostCustomer and no need mark [HttpPort]

        {

            if (!ModelState.IsValid)   //this check from DataAnnotion

            {

               throw new **HttpResponseException**(HttpStatusCode.**BadRequest**);

            }

            \_context.Customers.Add(customer);

            \_context.SaveChanges();

            return customer;

        }

        //PUT /api/customers/1

        [**HttpPut**]

        public void UpdateCustomer(int id, **Customer** customer)

        {

            if (!ModelState.IsValid)

            {

               throw new **HttpResponseException**(HttpStatusCode.**BadRequest**);

            }

            var customerInDb = \_context.Customers.SingleOrDefault(c => c.Id == id);

            if (customerInDb == null)

            {

                throw new **HttpResponseException**(HttpStatusCode.**NotFound**);

            }

       //Mapper.Map(customer, customerInDb); //Install-Package AutoMapper

            customer.Name = customerInDb.Name;

            customer.Birthday = customerInDb.Birthday;

            customer.IsSubscribedToNewsLetter = customerInDb.IsSubscribedToNewsLetter;

            customer.MembershipTypeId = customerInDb.MembershipTypeId;

            \_context.SaveChanges();

        }

        //DELETE /api/customers/1

        [**HttpDelete**]

        public void DeleteCustomer(int id)

        {

  var customerInDb = \_context.Customers.SingleOrDefault(c => c.Id == id);

if (customerInDb == null)

        {

              throw new **HttpResponseException**(HttpStatusCode.**NotFound**);

           }

            \_context.Customers.Remove(customerInDb);

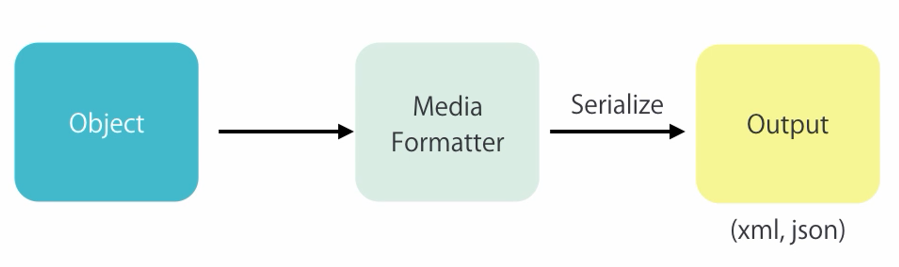
            \_context.SaveChanges();

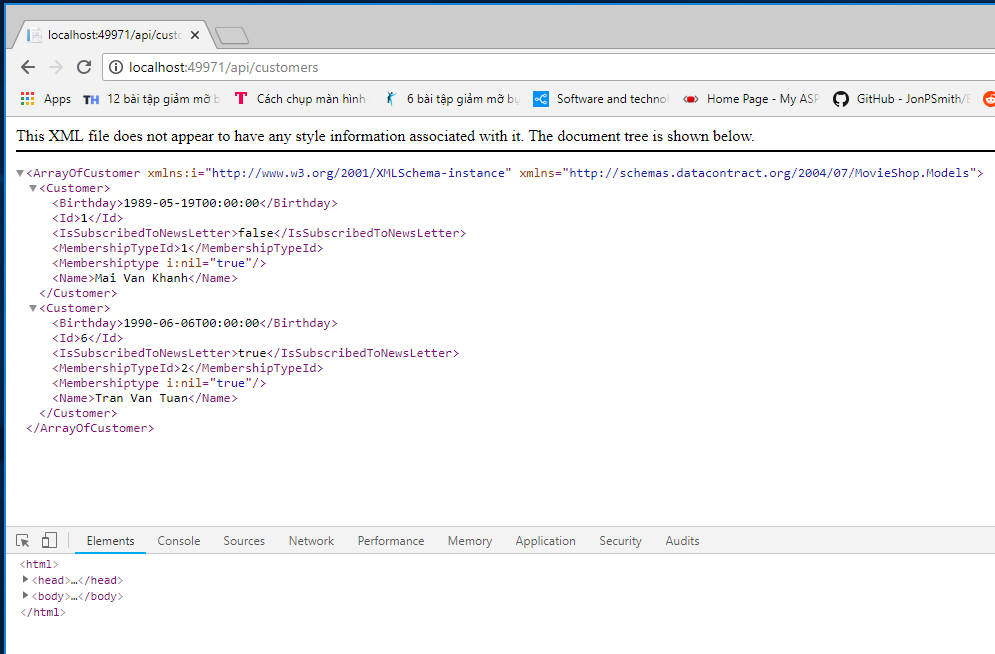
        }

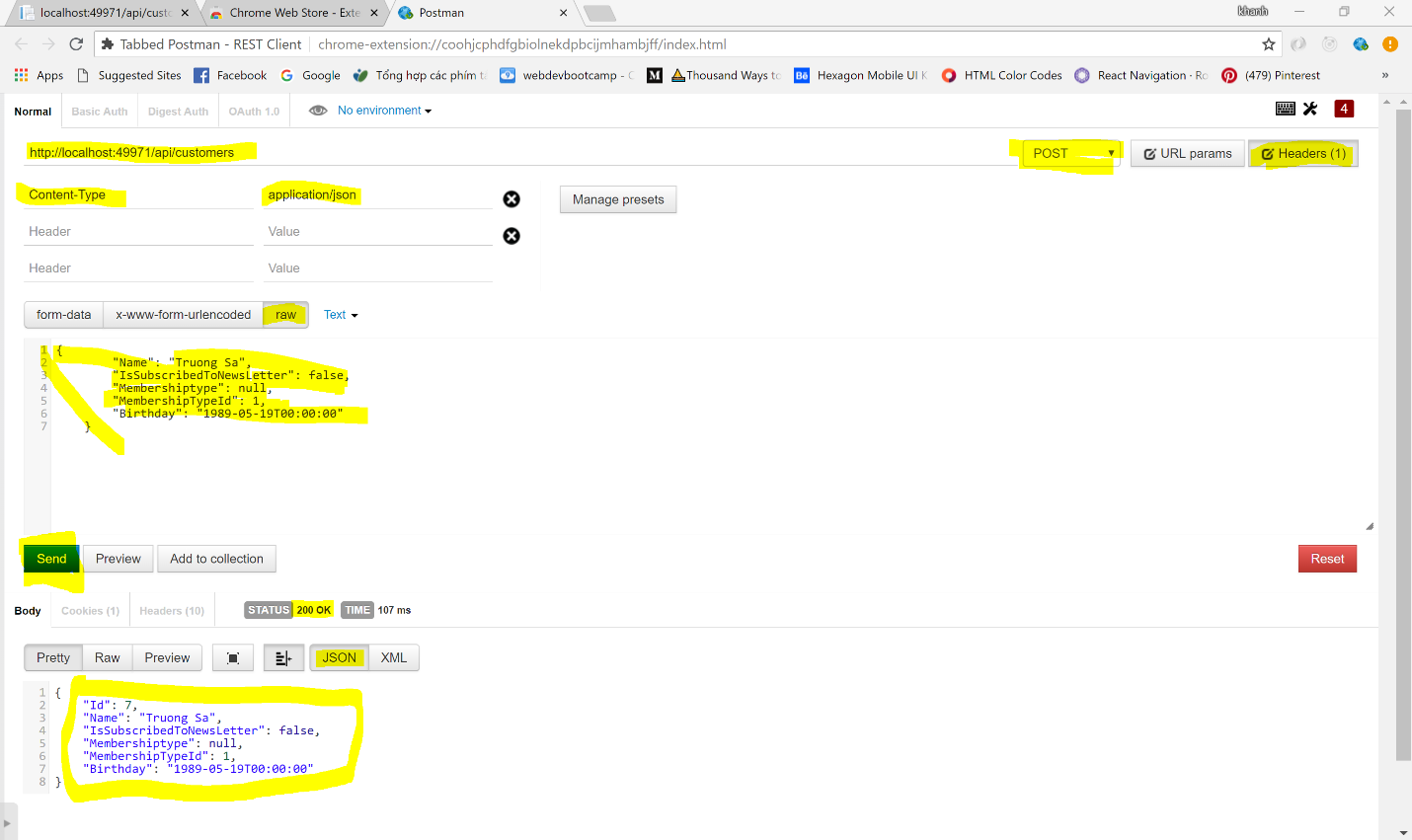
    }

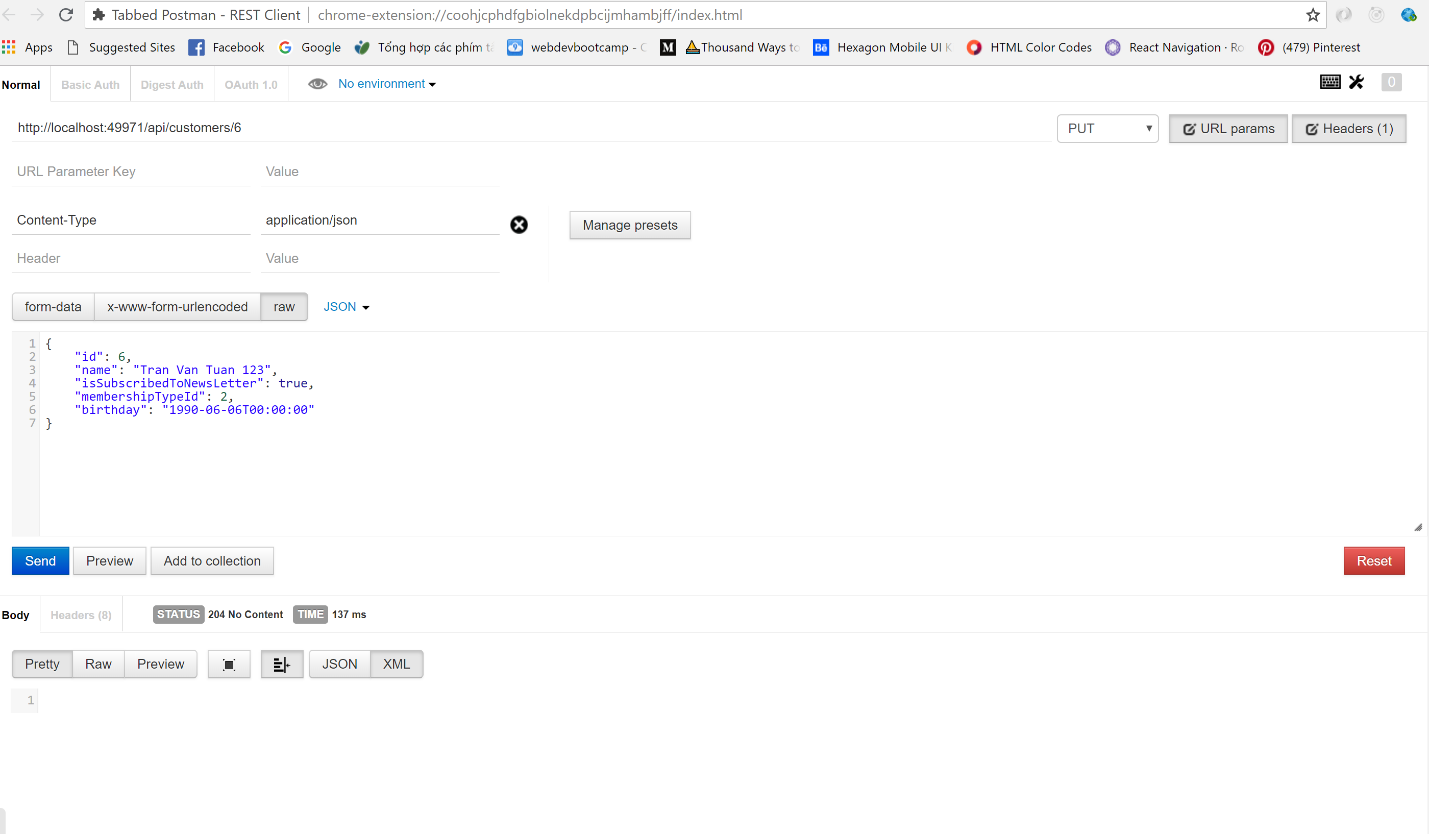
}

Go to browser check <http://localhost:49971/api/customers>









**Why use DTO ( Data Tranfer Object )**

The Data Transfer Object "DTO", is a simple serializable object used to transfer data across multiple layers of an application.

The fields contained in the DTO are usually primitive types such as strings, boolean, etc.

One of the great things about the DTO compared to a DataSet is that the DTO does not have to directly match a data table or view. The DTO can aggregate fields from another DTO.

If not use DTO, and use DataSet The client will receive data that maps directly to your database tables. However, that's not always a good idea.

Sometimes you want to change the shape of the data that you send to client. For example, you might want to:

* Remove circular references (see previous section).
* Omit some properties in order to reduce payload size.

you can define a data transfer object (DTO). A DTO is an object that defines how the data will be sent over the network.

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-4>

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>

<https://www.codeproject.com/Articles/8824/C-Data-Transfer-Object>

<https://stackoverflow.com/questions/5465560/why-should-i-not-make-a-class-serializable>

<https://softwareengineering.stackexchange.com/questions/350067/is-it-good-practice-to-use-entity-objects-as-data-transfer-objects>

1. Create folder Dtos and class CustomerDto

namespace MovieShop.Dtos

{

    public class **CustomerDto**

    {

        public int Id { get; set; }

        [**Required**]

        [**StringLength**(255)]

        public string Name { get; set; }

        public bool IsSubscribedToNewsLetter { get; set; }

        public byte MembershipTypeId { get; set; }

//        [Min18YearsIfAMember]

        public DateTime? Birthday { get; set; }

    }

}

1. Create in Controller folder, API folder and class name CustomerController.cs

Using automapper : Install-Package automapper.

**Config automapper at Global.asax,**

public class **MvcApplication** : System.Web.**HttpApplication**

{

    protected void Application\_Start()

    {

        //Config AutoMapper

**Mapper**.Initialize(cfg => {

            cfg.CreateMap<**Customer**, **CustomerDto**>();

            cfg.CreateMap<**CustomerDto**, **Customer**>();

        });

// add config Web API

        GlobalConfiguration.Configure(WebApiConfig.Register);

**AreaRegistration**.RegisterAllAreas();

**FilterConfig**.RegisterGlobalFilters(GlobalFilters.Filters);

**RouteConfig**.RegisterRoutes(**RouteTable**.Routes);

**BundleConfig**.RegisterBundles(BundleTable.Bundles);

    }

}

Create CustomerController class **inherit** from ApiContrller

namespace MovieShop.Controllers.API

{

    public class **CustomersController** : **ApiController**

    {

        private **ApplicationDbContext** \_context;

        public **CustomersController**()

        {

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

        }

        //GET /api/customers

        public IEnumerable<**CustomerDto**> GetCustomers()

//Use Object is CustomerDto

        {

            return \_context.Customers.ToList().Select(**Mapper**.Map<**Customer**, **CustomerDto**>); //Use Delegate to map customer to customerDto

        }

        //GET /api/customers/1

        public IHttpActionResult GetCustomer(int id) //Use IHttpActionResult to return the ActionResult replace for

Return object is CustomerDto

        {

       var customer = \_context.Customers.SingleOrDefault(c => c.Id == id);

            if (customer == null)

            {

                return NotFound();

            }

            return Ok(**Mapper**.Map<**Customer**, **CustomerDto**>(customer));

//map customer to customerDto

        }

        //POST /api/customers

        [**HttpPost**]

        public IHttpActionResult CreateCustomer(**CustomerDto** customerDto)  //Use IHttpActionResult to return the ActionResult

        {

            if (!ModelState.IsValid)        //this check from DataAnnotion

            {

                return BadRequest();   //this method come from IHttpActionResult helper Method of class Http.Result

            }

            var customer = **Mapper**.Map<**CustomerDto**, **Customer**>(customerDto);  //map customerDto to customer

            \_context.Customers.Add(customer);

            \_context.SaveChanges();

            customerDto.Id = customer.Id; //because Id is identity

            return Created(new **Uri**(Request.RequestUri + "/" + customer.Id), customerDto );

            //Created(Uri location, T content) | Uri location : /api/customers/10

        }

        //PUT /api/customers/1

        [**HttpPut**]

        public void UpdateCustomer(int id, **CustomerDto** customerDto)

        {

            if (!ModelState.IsValid)

            {

               throw new **HttpResponseException**(HttpStatusCode.**BadRequest**);

            }

   var customerInDb = \_context.Customers.SingleOrDefault(c => c.Id == id);

            if (customerInDb == null)

            {

                throw new **HttpResponseException**(HttpStatusCode.**NotFound**);

            }

            // Mapper.Map(customerDto, customerInDb); //Install-Package AutoMapper

            /\*

            customerDto.Name = customerInDb.Name;

            customerDto.Birthday = customerInDb.Birthday;

            customerDto.IsSubscribedToNewsLetter = customerInDb.IsSubscribedToNewsLetter;

            customerDto.MembershipTypeId = customerInDb.MembershipTypeId;

            \*/

**Mapper**.Map<CustomerDto, Customer>(customerDto, customerInDb);

//Update value from customerDto to customer

            \_context.SaveChanges();

        }

        //DELETE /api/customers/1

        [**HttpDelete**]

        public void DeleteCustomer(int id)

        {

var customerInDb = \_context.Customers.SingleOrDefault(c => c.Id == id);

            if (customerInDb == null)

            {

                throw new **HttpResponseException**(HttpStatusCode.**NotFound**);

            }

            \_context.Customers.Remove(customerInDb);

            \_context.SaveChanges();

        }

    }

}

**Camel Casting for JSon**

1. Open WebApiConfig.cs in App\_Start folder

public static class WebApiConfig

    {

        public static void Register(**HttpConfiguration** config)

        {

            //Config to Camel Casting for using Json

 var settings = config.Formatters.JsonFormatter.SerializerSettings;

//enable config

 settings.ContractResolver = new **CamelCasePropertyNamesContractResolver**();

 settings.Formatting = Formatting.**Indented**;

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(

                name: "DefaultApi",

                routeTemplate: "api/{controller}/{id}",

                defaults: new { id = **RouteParameter**.Optional }

            );

        }

    }

**Summary**

* Postman Chrome Extension
* Data Transfer Objects
* Auto Mapper
* Action Results
* Camel Notation

**Client – Side Development**

Talk ablout jQuery

<https://www.impressivewebs.com/jquery-tutorial-for-beginners/>

Use jQuery ajax to make the request

@section scripts

{

    <script>

        $(document).ready(function () {

    //This is to prevent any jQuery code from running before the document is finished loading (is ready).

         $("#customer .js-delete").on("click",

          function() {

            var button = $(this);

//this is button element call by $("#customer .js-delete")

            if (confirm("Are you sure you want to delete this customer?"))  {

                   $.ajax({ //this is jQuery ajax

                        url: "api/customers/" + button.attr("data-customer-id"),

          //url: A string containing the URL to which the request is sent.

                           method: "DELETE",

                           success: function() { //callback function

                                //console.log("Success");

                                button.parents("tr").remove();

//Remove row with delete button

                            }

                         });

                    }

             });

         });

    </script>

}

**Using Library Bootbox.js**

bootboxjs.com

install-package bootbox –version:4.3.0

Config bootbox in BundleConfig.cs

bundles.Add(new **ScriptBundle**("~/bundles/bootstrap").Include(

          "~/Scripts/bootstrap.js",

          "~/scripts/bootbox.js"    //Add bootbox in BundlesConfig

    ));

Use bootbox.confirm ( message, callback)

@section scripts

{

    <script>

        $(document).ready(function () {

            //This is to prevent any jQuery code from running before the document is finished loading (is ready).

            $("#customer .js-delete").on("click",

                function() {

                    var button = $(this);

//this: button element call by $("#customer .js-delete")

                    bootbox.confirm("Are you sure you want to delete this customer?",

                        function(result) {

                            $.ajax({ //this is jQuery ajax

                                url: "api/customers/" + button.attr("data-customer-id"),

          //url: A string containing the URL to which the request is sent.

                                method: "DELETE",

                                success: function() { //callback function

                                    //console.log("Success");

                                    button.parents("tr").remove();

//Remove row with delete button

                                }

                            });

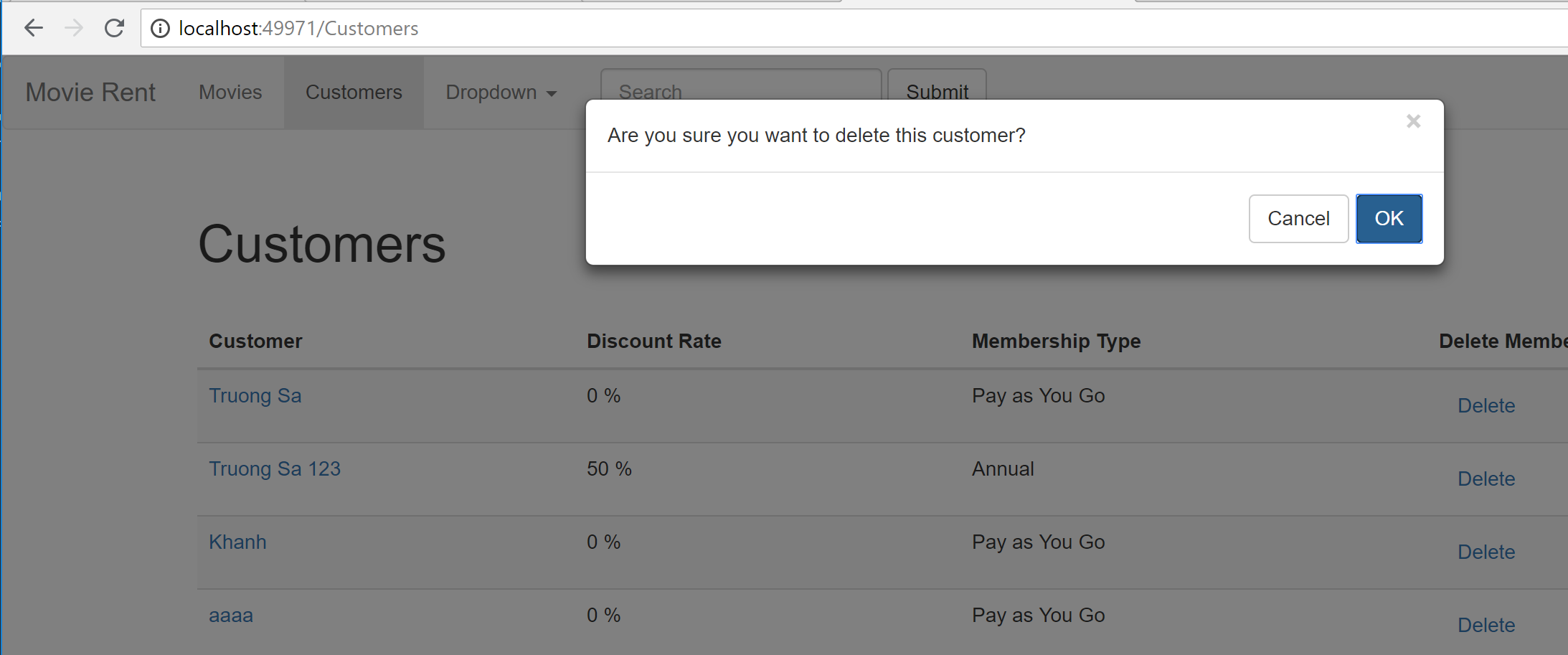
                        });

                });

        });

    </script>

}



**Use filter for element #customer**

With code

$("#customer .js-delete").on("click", function() { }

We will lost memory for all button select element like that.

Better we use filter ".js-delete" for element $("#customer")

An event-delegation approach attaches an event handler to only one element, the tbody, and the event only needs to bubble up one level (from the clicked .js-delete to #customer):

<http://api.jquery.com/on/>

$("#customer").on("click", ".js-delete" function() { }

**Use jQuery.datatables**

<https://datatables.net/>

<https://datatables.net/manual/ajax>

<https://datatables.net/manual/api>

In 4 minutes learn to increase jQuery Datatable performance

<https://www.youtube.com/watch?v=OPE5U25AO0U>

* install-package jquery.datatables -version:1.10.11

With the short data we can install jquery.datatables, config them in bundle.cs, and write this code in index.cshtml

$("#customer").DataTable();

@section scripts

{

    <script>

        $(document).ready(function () {

            //This is to prevent any jQuery code from running before the document is finished loading (is ready).

            $("#customer").DataTable();

//This code of DataTable make sort, filter, ...

            $("#customer").on("click",".js-delete",

                function() {

                    var button = $(this);

//this is button element call by $("#customer .js-delete")

                    bootbox.confirm("Are you sure you want to delete this customer?",

                        function (result) {

                            if (result) {

                                $.ajax({ //this is jQuery ajax

                                    url: "api/customers/" + button.attr("data-customer-id"),

         //url: A string containing the URL to which the request is sent.

                                    method: "DELETE",

                                    success: function() {

//callback function

                                        //console.log("Success");

                                        button.parents("tr").remove();

//Remove row with delete button

                                    }

                                });

                            }

                        });

                });

        });

    </script>

}

<table id="customer" class="table table-striped table-hover ">

    <thead>

        <tr>

            <th>Customer</th>

            <th>Discount Rate</th>

            <th>Membership Type</th>

            <th>Delete Member</th>

        </tr>

    </thead>

    <tbody>

        @foreach (var customer in Model)

        {

            <tr>

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

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

                <td>@customer.Membershiptype.Name</td>

                @\* <td>@Html.ActionLink("Delete", "Delete", "Customers", new { id = customer.Id }, new {@class="btn btn-danger"})</td> \*@

                <td>

                    <button data-customer-id="@customer.Id" class="btn btn-link js-delete">Delete</button>

                </td>

            </tr>

        }

    </tbody>

</table>

With a lot of customer we must be get customer from API, **DataTable with Ajax Source**

<https://datatables.net/manual/ajax>

<https://datatables.net/manual/api>

@section scripts

{

    <script>

        $(document).ready(function () {

            //This is to prevent any jQuery code from running before the document is finished loading (is ready).

            var table = $("#customers").DataTable({

                ajax: {

                    url: "/api/Customers",

                    dataSrc: ""

                },

                columns: [

                    {

                        data: "name",

                        render: function (data, type, customer) {

                            return "<a href='/customers/edit/" +

customer.id + "'>" + data + "</a>";

                        }

                    },

                    {

                        data: "membershipType.name"

                    },

                    {

                        data: "id",

                        render: function (data) {

                            return "<button class='btn btn-link

js-delete' data-customer-id=" + data + "> Delete </button>";

                        }

                    }

                ]

            });

            $("#customers").on("click",

                ".js-delete",

                function () {

                    var button = $(this);

//this is button element call by $("#customer .js-delete")

                    bootbox.confirm("Are you sure you want to delete this customer?",

                        function (result) {

                            if (result) {

                                $.ajax({ //this is jQuery ajax

                                    url: "api/customers/" + button.attr("

data-customer-id"),

     //url: A string containing the URL to which the request is sent.

                                    method: "DELETE",

                                    success: function () {

//callback function

                       //button.parents("tr").remove();

//Remove row with delete button

                       table.row(button.parents("tr")).remove().draw();

//Remove row from table define by datatable

//draw like SaveChange();

                                    }

                                });

                            }

                        });

                });

        });

    </script>

}

.draw(): When you perform an action such as adding or deleting a row, changing the sorting, filtering or paging characteristics of the table you'll want DataTables to update the display to reflect these changes. This function is provided for that purpose. <https://datatables.net/reference/api/draw()>

Table only need <tbody> empty space </tbody>

<table id="customers" class="table table-striped table-hover ">

    <thead>

        <tr>

            <th>Customer</th>

            <th>Membership Type</th>

            <th>Delete</th>

        </tr>

    </thead>

    <tbody>

        @\*

          @foreach (var customer in Model)

          {

            <tr>

               <td>

@Html.ActionLink(customer.Name, "Edit", "Customers", new {id = customer.Id}, null)

</td>

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

               <td>@customer.MembershipType.Name</td>

               // <td>@Html.ActionLink("Delete", "Delete", "Customers", new { id = customer.Id }, new {@class="btn btn-danger"})</td>

             <td>

                 <button data-customer-id="@customer.Id" class="btn btn-link js-delete">Delete</button>

             </td>

             </tr>

            }

        \*@

    </tbody>

</table>