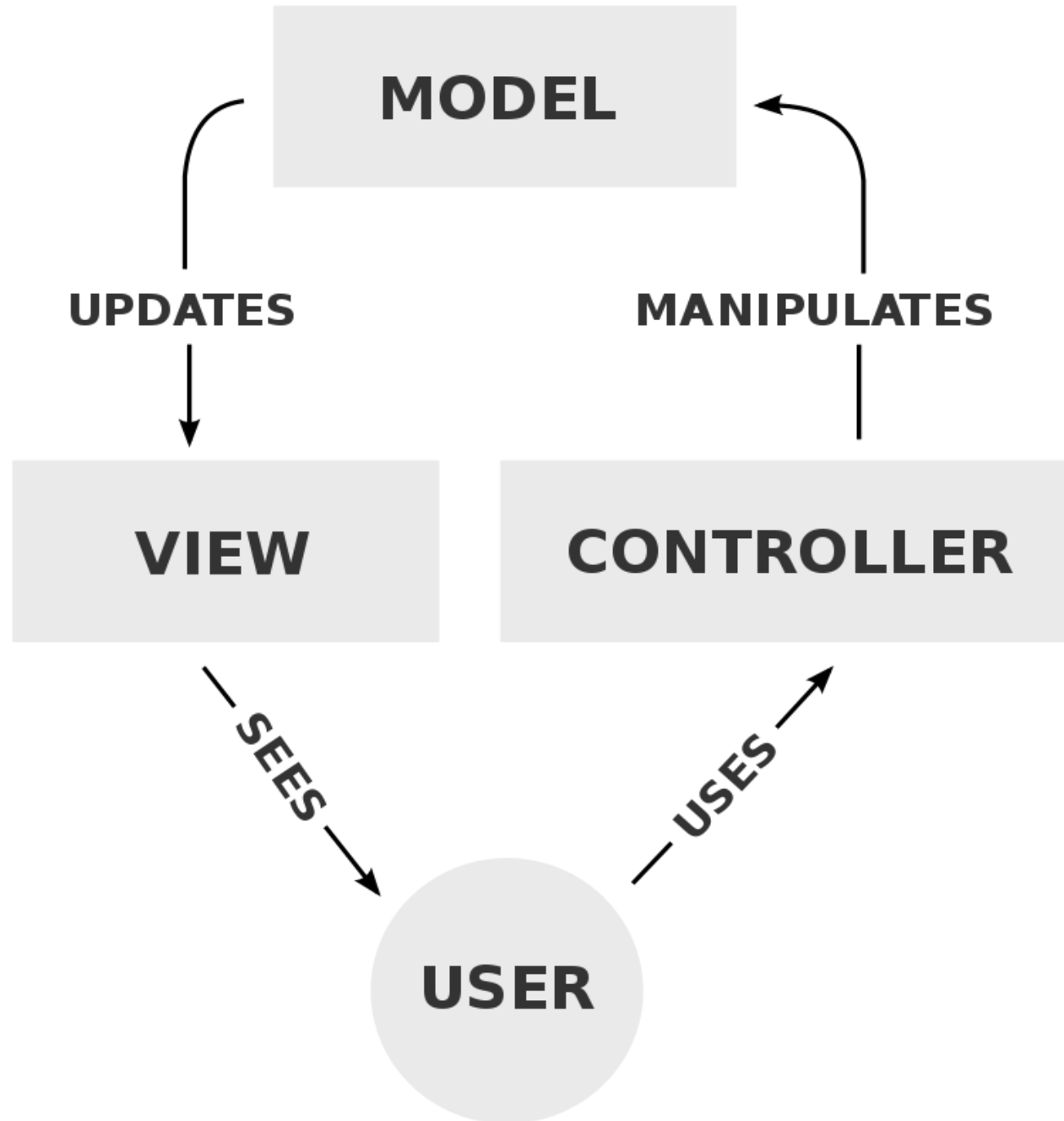


Controller

# 1.MVC pattern

- ជា software design pattern ដែលវាធ្វើការបំបែក application ជា ៣ផ្នែកគឺ model, view និង controller ដើម្បីបង្កើតភាពងាយស្រួលដល់ developer ដើម្បី maintenance application ។



# MVC pattern

- Model(is a data and business logic) ជា ការបង្ហាញទំរង់នៃទិន្នន័យនិង business logic. វាជា អ្នកគ្រប់គ្រងទិន្នន័យរបស់កម្មវិធីដែលមានតួនាទីជាអ្នកទទួលនិងរក្សាទុកទិន្នន័យក្នុងជាតាបេស។
- View(is a User Interface) ជាអ្វីៗដែលអ្នកទស្សនាមើលឃើញដែលវាបង្ហាញទិន្នន័យដោយប្រើប្រាស់នូវ model ទៅកាន់អ្នកទស្សនានិងអោយគេធ្វើការកែប្រែទិន្នន័យបាន។
- Controller (request handler) ជាអ្នកដោះស្រាយនូវសំណើរបស់អ្នកប្រើប្រាស់(user)។ វាទូទៅអ្នកប្រើប្រាស់ធ្វើអន្តរកម្មជាមួយ View តាមរយៈការលើកជាសំណើ URL ត្រឹមត្រូវដែលសំណើទាំងត្រូវបានដោះស្រាយដោយ Controller។ បន្ទាប់មក controller នឹងបង្ហាញនូវចំណើយតាមតាមរយៈ data model ទៅជា view សមរម្យមួយ។

## 2.Controller

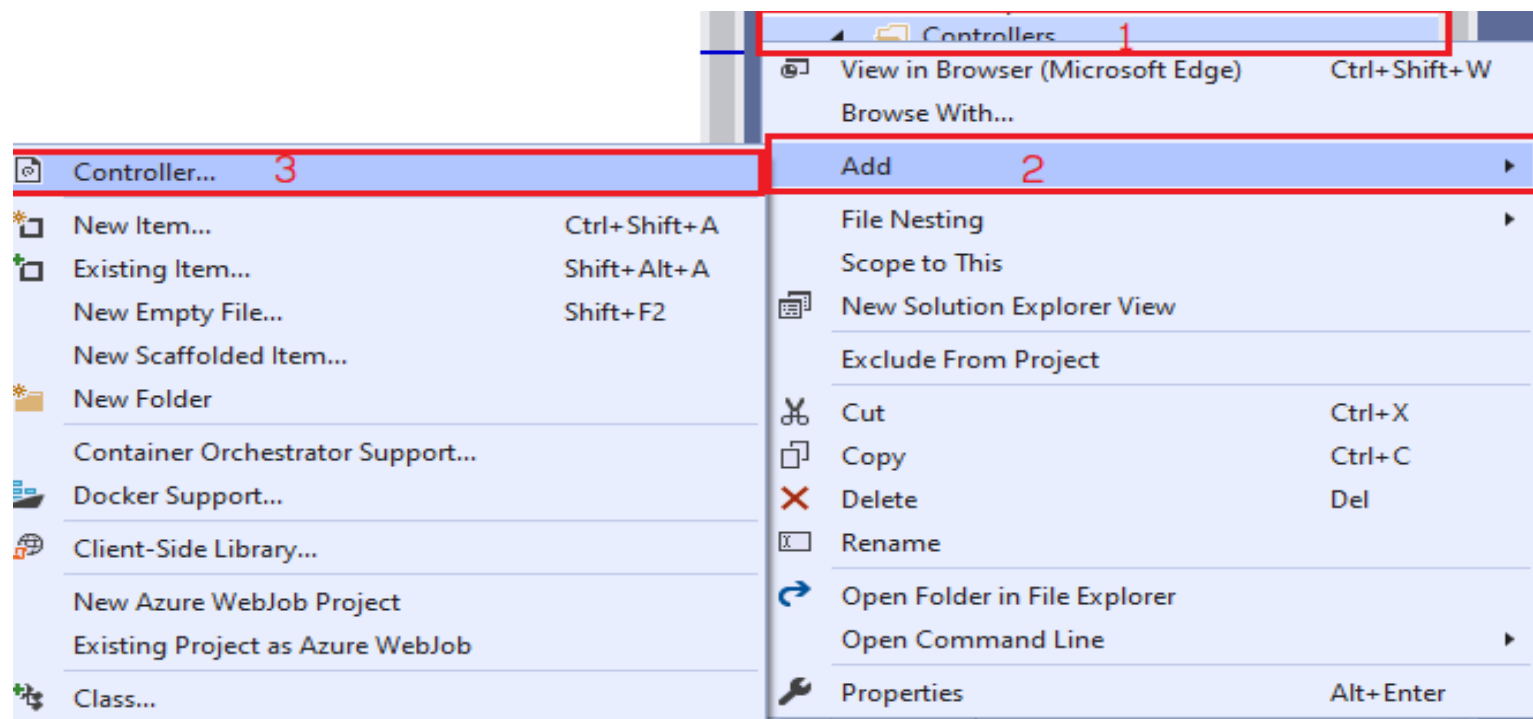
- Controller ជាបណ្តុំ classes ជាច្រើនដែលចាំទទួលដោះស្រាយ នូវសំណើរពី user និង នឹងឆ្លើយនូវតបនូវសំណើររបស់ user ។
- នៅពេលដែល user វាយនូវសំណើរ(request) នៅលើ address bar របស់ browser ពេលនោះ route engine របស់ asp.net mvc នឹងធ្វើការផ្លាស់ប្តូរនូវសំណើរជាមួយ controller's actions method ជាក់លាក់ណាមួយ។

# តួនាទីរបស់ controller

- គ្រប់គ្រង user's request( handle the user's request )
- បង្កើត model ( create model ) វាចំណេញការ ( execute ) នូវ application logic ហើយបង្កើត model ។
- និងបញ្ជូនចំណេញតបទៅ user វិញ ( send the response ) ជាចុងក្រោយវានិងបោះចំណេញត្រឡប់ទៅវិញក្នុងទម្រង់ជា html/json/xml ឬទម្រង់ផ្សេងទៀតដើម្បីប្រើប្រាស់បានឆ្លើយតប។

# ការបង្កើត controller

- ចុចម៉ៅស្តាំលើចត controllers
- ចុចពាក្យថា add → controller
- ជ្រើសរើស controller template → MVC empty → Add
- ដាក់ឈ្មោះ controller (ឈ្មោះ controller ត្រូវបញ្ចប់ដោយ controller )
- រួចហើយចុចពាក្យ → Add



Add New Scaffolded Item





## 2.1.Action method

- គ្រប់ public method ទាំងអស់ដែលគេសរសេរនៅក្នុង controller class គេហៅថា Action Method។ នៅពេលដែលយើងបង្កើត action method ត្រូវតែអនុវត្តន៍តាមរបៀបដូចខាងក្រោម៖
  - គ្រប់ actions method ទាំងអស់ត្រូវតែ public
  - វាមិនអាច overloaded ទេ
  - វាមិនអាចជា static method
  - វាមិនអាចមាន ref ឬ out parameters
  - ត្រូវតែមាន return type

Student Controller class

Base Controller class

```
public class StudentController : Controller
```

Return type

```
{
```

```
// GET: Student
```

```
public ActionResult Index()
```

Action method

```
{
```

```
    return View();
```

View() defined in base  
Controller class

```
}
```

```
}
```

## 2.2.Action parameteres

- ສັງເກດ method ຈຸດເຈາະເຝິກ ດີ້ເປັນ action method ກໍ່ເກີດ parameters ເຝິກສັງເກດເປັນ:
- Simple type
  - Integer
  - Double
  - String
  - Boolean ເປັນເປັນ
- ສັງເກດ complex type
  - Class object

## 2.3. កម្រិត Action method

- [www.host:port/controller\\_name/action\\_name](http://www.host:port/controller_name/action_name)
- [www.host:port/controller\\_name/action\\_name/\[parameters\]](http://www.host:port/controller_name/action_name/[parameters])
- Example
  - [www.localhost/student/index](http://www.localhost/student/index)
  - [www.localhost.com/student/search/1](http://www.localhost.com/student/search/1)

## 2.5.Routing

- Routing module ជាផ្នែកសំខាន់មួយរបស់ asp.net core ដែលវាពិនិត្យមើលនូវរាល់ URLs ដែលចូលមកហើយវានិងហៅ controller's action methodដែលត្រឹមត្រូវដែលទាក់ទងនឹងសំណើរនោះ។
- ឧទាហរណ៍ថាURL ដែលចូលមក [www.localhost/order/list](http://www.localhost/order/list)
  - តំបូងវាឆ្លងកាត់ចូលតាមរយៈ routing module
  - បន្ទាប់មក routing moduleនិងពិនិត្យមើល URL នេះហើយវានិងជ្រើសរើសនូវ controller ណាដែលត្រឹមត្រូវដើម្បីដោះស្រាយសំណើរនេះ (handle the request)
  - ហើយបន្ទាប់មកវានិងហៅ action's method ត្រឹមត្រូវដំណើរការបន្ទាប់។

## 2.5.Return types

- Action method ទទួលខុសត្រូវចំពោះការផ្ញើសំរេបសំរួលផ្ទៃប្រាសនិងបង្កើតការឆ្លើយតបដែលជាចុងក្រោយវាបង្ហាញដល់អ្នកប្រើប្រាស់ដែលគេហៅថាresponse អាចជា html json xml ឬក៏file download។
- Action method អាចបោះតំលៃបានគ្រប់ប្រភេទតាមដែលយើងចង់។
- ជាទូទៅ controller action method គួរតែបោះតំលៃផ្ទៃម្យ៉ាងគេហៅថា action result។ តំលៃដែលបោះមកនិមួយ(html json xml។ល។)មាន action result ផ្ទាល់ខ្លួនរបស់ពួកគេដែលទទួលដំណែល(inherit)បានពី ActionResult base class(abstract class)។

## 2.5.Return types

ActionResult	Helper Method	Description
ViewResult	View	បោះពុម្ពផ្សាយជា HTML markup language
PartialViewResult	PartialView	បង្ហាញជា បំណែកតូចនៃ web page( partial view )
RedirectResult	Redirect	បោះពុម្ពផ្សាយនូវ ការទៅទីតាំងណាមួយ
RedirectToRouteResult	RedirectToRoute	ប្រើសម្រាប់ទៅទីតាំងណាមួយ
RedirectToActionResult	RedirectToAction	ប្រើសម្រាប់ទៅ action ណាមួយ
ContentResult	Content	បោះពុម្ពផ្សាយអក្សរធម្មតា( plain text )
JsonResult	Json	បោះពុម្ពផ្សាយជា Json
EmptyResult	( None )	បោះពុម្ពផ្សាយទទេ

### 3. វិធីទៅជា Routing ?

- វាជាតំណើរការនៃការផ្លាស់ប្តូរ incoming request ជាមួយ controller's action method ជាកំណត់ណាមួយហើយក៏ប្រើវាដើម្បីបង្កើត outgoing urls ផងដែរដែលតំណើរការនេះត្រូវបានដោះស្រាយដោយ middleware ដែល route middleware មានក្នុង

[Microsoft.AspNetCore.Routing](#) Namespace ។

- Routes នីមួយៗផ្តើមឡើងដូចជា ឈ្មោះ, URL pattern( template ), default និង constrain

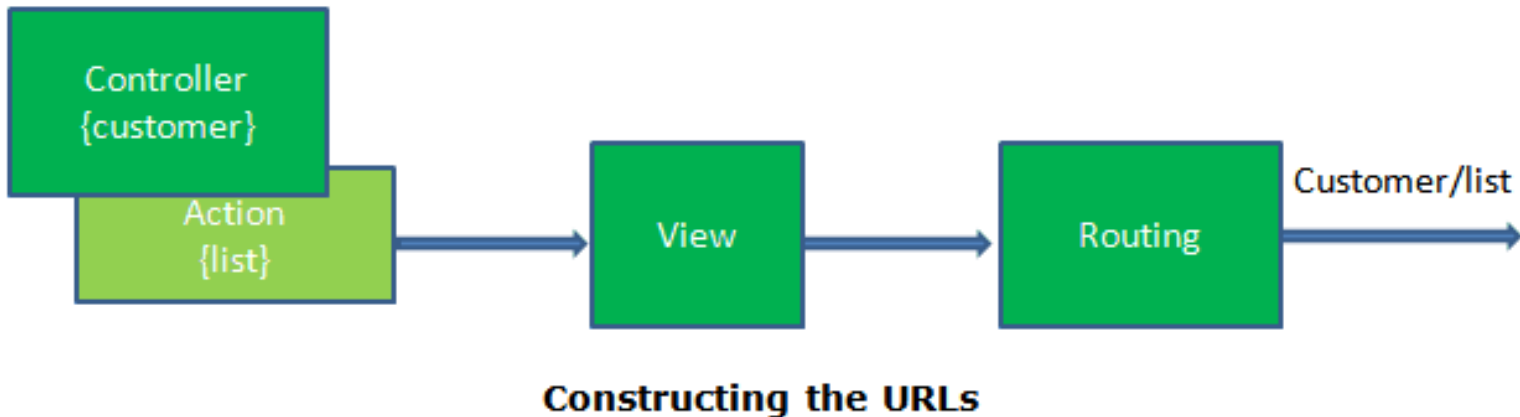
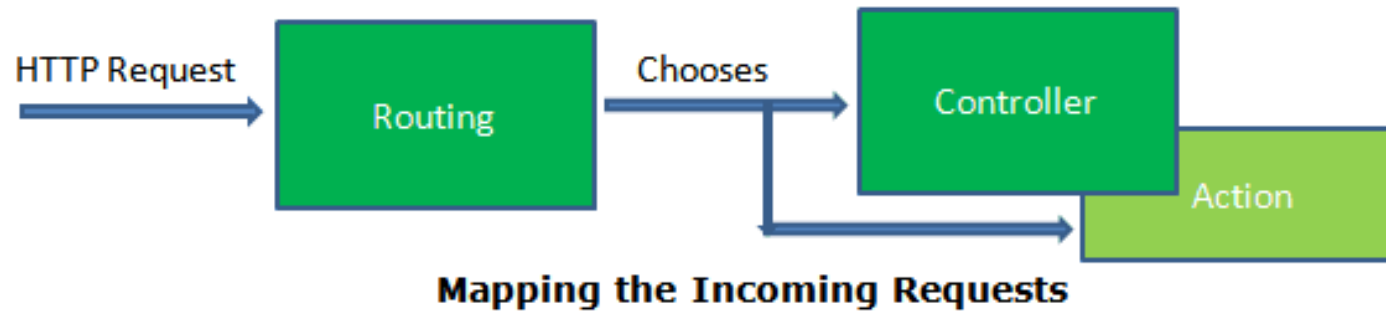


### 3. វិធីសាស្ត្រជា Routing ?

- Route ទទួលខុសត្រូវរៀបចំសំខាន់ពីរគឺ៖
  - ធ្វើការផ្លាស់ប្តូរ user's request ដែលចូលមកជាមួយនិង controller's action method
  - បង្កើត outgoing URL ដែលត្រូវផ្ញើនិង controller's action method

### 3. କିପରିକି Routing ?

#### Routing in ASP.NET MVC Core



## 3.1. ការបង្កើតRoute ?

- មានវិធីសាស្ត្រពីរបីក្នុងបង្កើត Route
  - Conventional-base routing
    - ជាការបង្កើតroute នៅក្នុង startup class ក្នុង startup file
  - និង ការប្រើប្រាស់ attribute routing
    - ជាការបង្កើត route ដោយប្រើប្រាស់route attribute នៅក្នុង controller's action method

## A. Conventional Configuration

- ត្រូវបានបង្កើតនៅក្នុង configure method របស់ startup class
- Routing ត្រូវបានគ្រប់គ្រងដោយ router middleware ដែល asp.net mvc បង្កើត routing middleware ទៅក្នុង middleware pipeline នៅពេលដែលយើងប្រើ app.useRouting( ) និង app.useEndpoint( ) extension methods។

# Adding Service

- ដើម្បីអោយវាដំណើរបានត្រឹមត្រូវយើងបន្ថែមservice ជាមុនសិនដូចខាងក្រោម។

0 references

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```

# Adding Middleware

- ដើម្បីបង្កើត route ត្រូវបន្ថែម middleware ចំនួនពីរគឺ៖
  - App.UseRouting( )
  - និង app.UseEndpoint( ) extension method


```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        //your routes here
    });
}
```

# Syntax

endpoints.MapControllerRoute(

string name,  ស្ម័គ្រច្បាប់ route

string pattern ,  URL pattern

object default,  កំណត់តំលៃដើមរោយ route

object constrain  កំណត់តំលៃ constrain

);

# URL Patterns

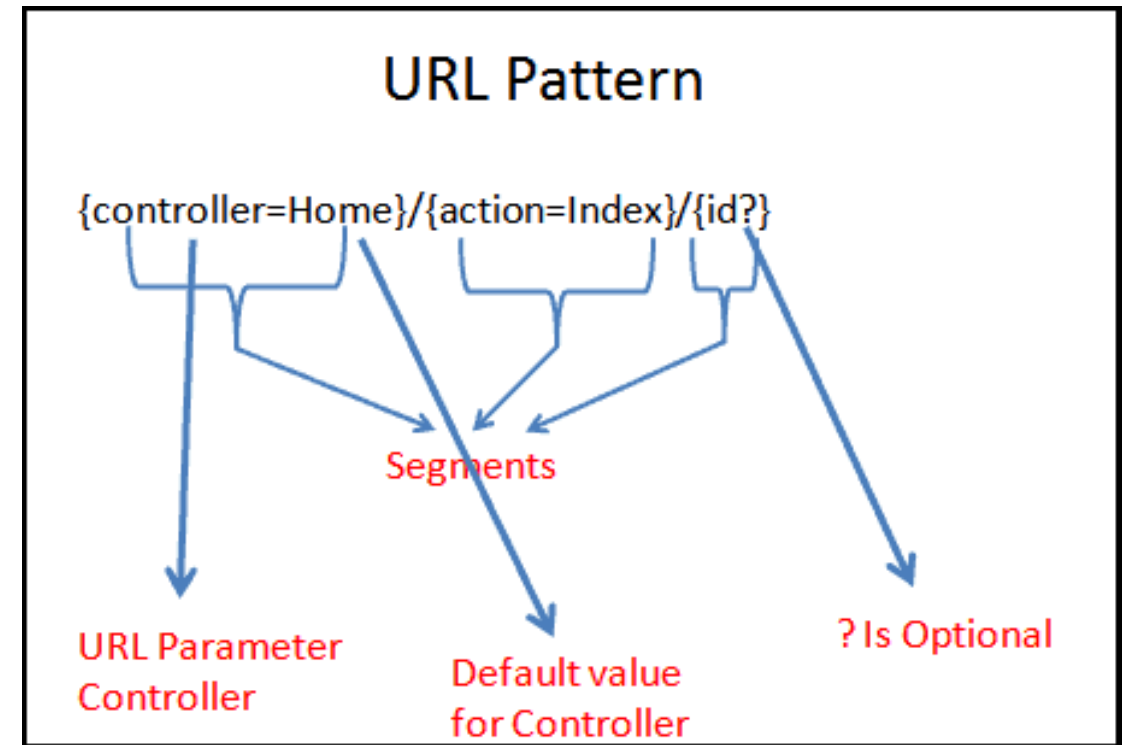
- URL pattern ជាបំណែកតូចនៃ user's request (controller, action method, និង parameteres) ដែលខ្ញុំស្នើសុំដោយសញ្ញា (/)។

`http://localhost:1234/home/index/100`

Controller    Action method    Id parameter value

`http://localhost:1234/home/index`

Controller    Action method





# URL Matching

- រាល់ segments ទាំងអស់នៅក្នុង incoming url គឺវាផ្ទុកផ្ទុកជាមួយ នឹង segmentដែលត្រូវ ផ្ទៀងផ្ទាត់ជាមួយនឹង URL patternដែលជាទូទៅ មានប្លែងទៅជា៖ {controller}/{action}/{parameter} ប៉ុន្តែក៏អាចមាន លើសពីបីក៏បានដែរ។

# URL Matching


## URL Matching


`{controller=Home}/{action=Index}/{id?}`

`www.Example.com/Product/List`

First Segment is mapped to the first segment in the Pattern

# ดูจาบรรณ

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",  เฉพาะรอบส่ route
            pattern: "{controller}/{action}/{id}"
        );
    });
}
```

 URL pattern

# Route matching(routing table)

- តាមឧទាហរណ៍ខាងលើវានឹងបង្កើតបាន routing table ដូចខាងក្រោម៖

URL	controller	action	parameter
localhost/product/save	ProductController	Save	Form data
localhost/product/delete/234	ProductController	Delete	234
localhost/product/edit/123	ProductController	Edit	123
localhost/order/submit	OrderController	Submit	Form data

# ഉദാഹരണം

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "example",
            pattern: "{controller}/{action}/{id?}",
            defaults: new { controller = "Home", action = "Index" }
        );
    });
}
```

# Route matching(routing table)

- តាមឧទាហរណ៍ខាងលើវានិងបង្កើតបាន routing table ដូចខាងក្រោម៖

URL	controller	action	parameter
localhost/home	HomeController	Index	Null
localhost/home/about	HomeController	About	Null
localhost/order/list	OrderController	List	Null
localhost/product/save	ProductController	Save	Null
localhost/product/delete/234	ProductController	Delete	234
localhost/product/edit/123	ProductController	Edit	123
localhost/order/submit	OrderController	Submit	null

# Route Defaults

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseDeveloperExceptionPage();
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapDefaultControllerRoute();
    });
}
```

- វាជួយបង្កើត route ដូចខាងក្រោម៖

Route {controller=Home}/{action=Index}/{id ?}

# Route Defaults

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseDeveloperExceptionPage();
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            "default",
            "{controller=Home}/{action=Index}/{id?}");
    });
}
```

- វាជួយបង្កើត route ដូចខាងក្រោម៖

Route {controller=Home}/{action=Index}/{id ?}



# Route Defaults

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller}/{action}/{id}",
            defaults: new { controller="Product",action="get" }
        );
    });
}
```

---

# Route parameters

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller}/{action}/{id}",
            defaults: new { controller="Product",action="get" }
        );
    });
}
```

---

# Multiple Routes

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseDeveloperExceptionPage();
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            "default",
            "{controller=Home}/{action=Index}/{id?}"
        );
        endpoints.MapControllerRoute(
            "example",
            "blog/{controller}/{action}/{id?}"
        );
    });
}
```

# Routes Area

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseDeveloperExceptionPage();
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            "default",
            "{controller=Home}/{action=Index}/{id?}")
        );
        endpoints.MapAreaControllerRoute(
            "areas",
            "area_name",
            "{area}/{controller=Home}/{action=Index}/{id?}")
        );
    });
}
```

## 3.1.2.Attribute's route

- គេអាចកំណត់ route attribute បានពីរបៀប៖
  - កំណត់ដោយផ្ទាល់ចំពោះ action methodណាមួយ
  - កំណត់ជាមួយ controllerណាមួយ
- ដើម្បីអោយ routing attribute ដំណើរការយើងត្រូវធ្វើរឿងពីរយ៉ាងដូចខាងក្រោម៖
  - បន្ថែម AddControllerWithView() method ក្នុង ConfigureServices() methodនៃ startup class
  - និងបន្ថែម UseRouting() និង UseEndpoint() middlewareក្នុង Configure() methodនៃ startup class

# Attribute's route

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```

// This method gets called by the runtime. Use this method to configure

0 references

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

# កំណត់ជាមួយ action method

```
public class ProductController : Controller
{
    [Route("/")]
    public string Index()
    {
        return "using route attributes.";
    }
}
```

matching

[Route("/")]

0 references

example.com  
example.com/

# កំណត់ជាមួយ action method (multiple route)

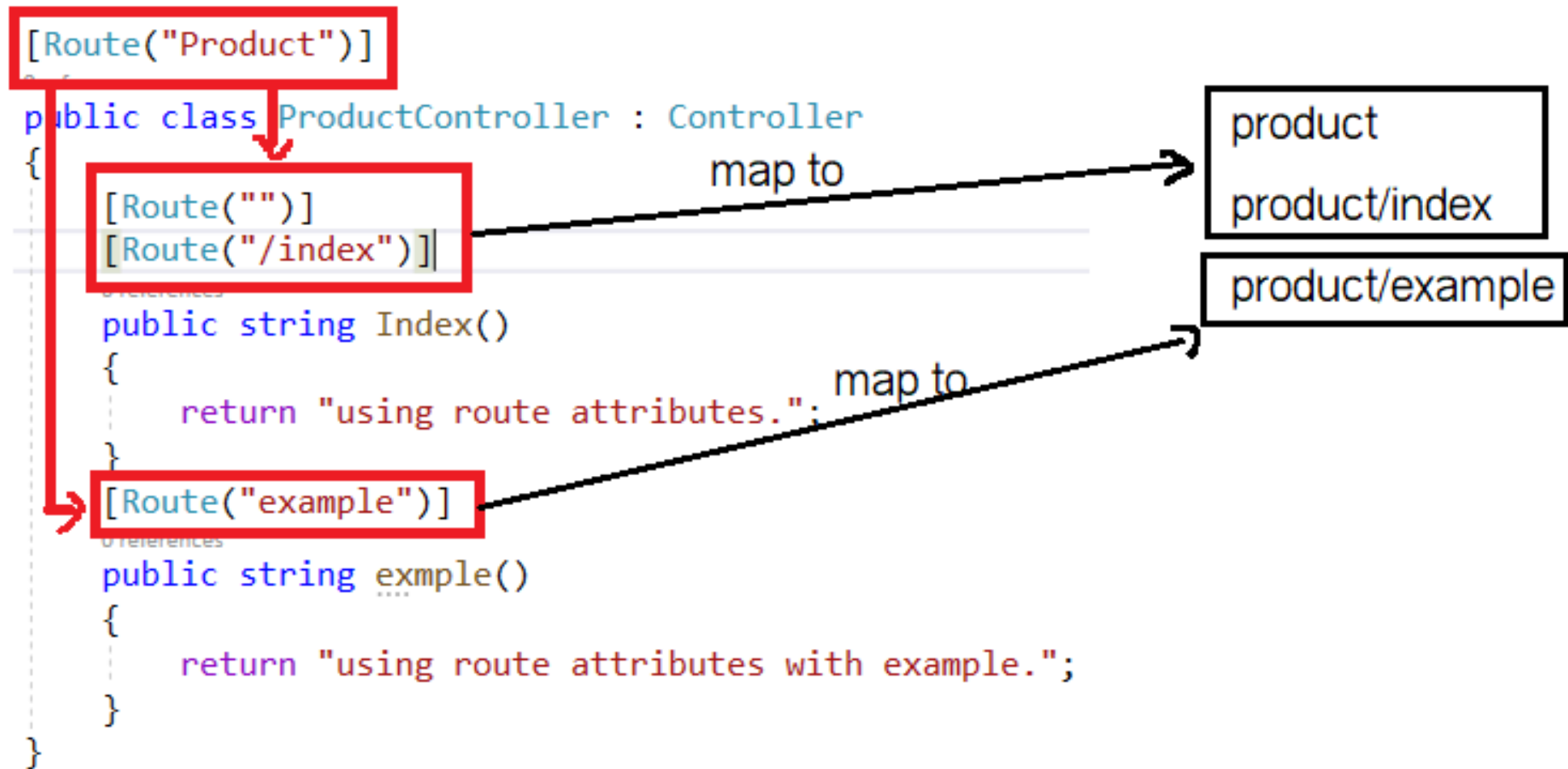
```
public class ProductController : Controller
{
    [Route("/")]
    [Route("/index")]
    [Route("product/index")]
    public string Index()
    {
        return "using route attributes.";
    }
}
```

matching

www.example.com  
www.example.com/index  
www.example.com/product/index



# កំណត់ជាមួយ controller



# Token Replacement

```
[Route("[controller]")]
```

0 references

```
public class ProductController : Controller
```

```
{
```

```
    [Route("")]
```

```
    [Route("[action]")]
```

0 references

```
    public string Index()
```

```
    {
```

```
        return "using route attributes.";
```

```
    }
```

```
    [Route("[action]")]
```

0 references

```
    public string example()
```

```
    {
```

```
        return "using route attributes with example.";
```

```
    }
```

```
}
```

# Token Replacement

```
[Route("[controller]/[action]")]
```

0 references

```
public class ProductController : Controller
```

```
{
```

0 references

```
    public string Index()
```

```
    {  
        return "using route attributes.";  
    }
```

```
    [Route("{name:alpha}")]
```

0 references

```
    public IActionResult example(string name)
```

```
    {
```

```
        return Content($"<h4>route attribute and constrain==> {name}</h4>"  
                        , contentType: "text/html");
```

```
    }
```

```
}
```

# Token Replacement(multiple route)

```
[Route("~/")]
[Route("[controller]")]
[Route("[controller]/[action]")]
```

0 references

```
public class ProductController : Controller
{
```

0 references

```
    public string Index()
    {
        return "using route attributes.";
    }
```

```
    [Route("{name:alpha?}")]
```

0 references

```
    public IActionResult example(string name)
    {
        return Content($"<strong>{name}</strong>", contentType: "text/html");
    }
}
```

# Route constrain

- Route constrain ជួយឱ្យយើងច្រោះឬដាក់កម្រិតលើតម្លៃដែលមិនចង់បានពី controller action's method parameter (The Route Constraints helps us to filter out or restrict the unwanted values from reaching the controller action) ។

# Route constrain

- គេមានវិធីសាស្ត្រចំនួនពីរដើម្បីប្រើនូវ route's constrain:
  - Inline with the URL Parameter
  - Using the Constraint argument of the MapControllerRoute method.

# Inline with the URL Parameter

CONSTRAINT	INLINE	NOTES
int	{id:int}	Constrains a route parameter to represent only 32-bit integer values
alpha	{id:alpha}	Constrains a route parameter to contain only lowercase or uppercase letters A through Z in the English alphabet.
bool	{id:bool}	Constrains a route parameter to represent only Boolean values.
datetime	{id:datetime}	Constrains a route parameter to represent only DateTime values.
decimal	{id:decimal}	Constrains a route parameter to represent only decimal values.
double	{id:double}	Constrains a route parameter to represent only 64-bit floating-point values
float	{id:float}	Matches a valid float value (in the invariant culture - see warning)
guid	{id:guid}	Matches a valid Guid value

# Inline with the URL Parameter

CONSTRAINT	INLINE	NOTES
length(length)	{id:length(12)}	Constrains a route parameter to be a string of a given length or within a given range of lengths.
maxlength(value)	{id:maxlength(8)}	Constrains a route parameter to be a string with a maximum length.
minlength(value)	{id:minlength(4)}	Constrains a route parameter to be a string with a maximum length.
range(min,max)	{id:range(18,120)}	Constraints a route parameter to be an integer within a given range of values.
min(value)	{id:min(18)}	Constrains a route parameter to be a long with a minimum value.
max(value)	{id:max(120)}	Constrains a route parameter to be an integer with a maximum value.
egex(expression)	{ssn:regex(^\\d{{3}}-\\d{{2}}-\\d{{4}}\$)}	Constrains a route parameter to match a regular expression.



# ឧទាហរណ៍

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller}/{action}/{id:int?}"
        );
    });
}
```

# Constraint method of MapControllerRoute

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller}/{action}/{id?}",
            defaults: new { controller="Home", action="Index" },
            constraints: new { id=new IntRouteConstraint() }
        );
    });
}
```

# Constraint method of MapControllerRoute

CONSTRAINT	CLASS	NOTES
int	IntRouteConstraint	Constrains a route parameter to represent only 32-bit integer values
alpha	AlphaRouteConstraint	Constrains a route parameter to contain only lowercase or uppercase letters A through Z in the English alphabet.
bool	BoolRouteConstraint	Constrains a route parameter to represent only Boolean values.
datetime	DateTimeRouteConstraint	Constrains a route parameter to represent only DateTime values.
decimal	DecimalRouteConstraint	Constrains a route parameter to represent only decimal values.
double	DoubleRouteConstraint	Constrains a route parameter to represent only 64-bit floating-point values
float	FloatRouteConstraint	Matches a valid float value (in the invariant culture - see warning)
guid	GuidRouteConstraint	Matches a valid Guid value

# Constraint method of MapControllerRoute

CONSTRAINT	CLASS	NOTES
length(length)	LengthRouteConstraint	Constrains a route parameter to be a string of a given length or within a given range of lengths.
maxlength(value)	MaxLengthRouteConstraint	Constrains a route parameter to be a string with a maximum length.
minlength(value)	MinLengthRouteConstraint	Constrains a route parameter to be a string with a maximum length.
range(min,max)	RangeRouteConstraint	Constrains a route parameter to be an integer within a given range of values.
min(value)	MinRouteConstraint	Constrains a route parameter to be a long with a minimum value.
max(value)	MaxRouteConstraint	Constrains a route parameter to be an integer with a maximum value.
egex(expression)	RegexRouteConstraint	Constrains a route parameter to match a regular expression.



## 5.Action Selector

- ជា attribute មួយដែលគេប្រើប្រាស់សម្រាប់កំណត់លក្ខណៈរបស់ action method របស់ controller។
- គេចែកជាបីគឺ៖
  - ActionName
    - ប្រើដើម្បីស្វែងរក action method
  - NoAction
    - ប្រើសម្រាប់កំណត់ action method ជា private ឬ method ឆ្គង
  - ActionVerb
    - ប្រើសម្រាប់កំណត់ប្រភេទនៃ http request( GET ឬ POST request )

## 5.1.ActionName attribute

- ActionName ជា attribute មួយដែលគេប្រើដើម្បីប្តូរឈ្មោះនូវ action method ។

```
14 [ActionName("changed")]
    0 references
15 public object method1()
16 {
17     return new { name = "Mon Minh", sex = "male" };
18 }
```

## 5.2.NoAction

- NoAction ជា attribute មួយដែលប្រើសម្រាប់កំណត់ action method ជា private ឬ method ធម្មតា។

```
19  [NonAction]
20  0 references
21  public string[] method2()
22  {
23      return new string[] { "Hello", "hi..." };
}
```



## 5.3.ActionVerb

- ActionVerb គេប្រើនៅពេលដែលយើងចង់គ្រប់គ្រង action methods អាស្រ័យលើប្រភេទនៃ http request method ដូចជា httpGet ឬ httpPost។
- Asp.net core មាន httpVerb មួយចំនួនដូចជា៖ GET, POST, PUT, DELETE, OPTIONS, HEAD និង PATCH។
- យើងផ្ដល់ attributes ទាំងនេះជាមួយនឹង controller action method។ ពេលដែល client បញ្ជូនសំណើជាមួយ verb ណាមួយបន្ទាប់មក routing engine នឹងស្វែងរក controller action ដែលបានកំណត់ attribute នោះហើយចាប់ផ្ដើមហៅវា។
- យើងដឹងហើយថា action method មិនអាច overloaded បានទេតែដោយប្រើ HTTP attribute អាចអោយយើង overloaded បាន។

# HTTP attributes

Atributes	ប្រើប្រាស់
httpGet	ទាញយកឯកសារពី server
httpPost	ប្រើសម្រាប់បង្កើត resource ថ្មីនៅលើ server
httpPut	កែប្រែឬបង្កើត resource នៅលើ server
httpDelete	ប្រើសម្រាប់លុប resource ចេញពី server
httpOptions	
httpHead	ទាញយក ព័ត៌មាន HTTP header information
httpPatch	

## 6.ActionResult

- វាជាលទ្ធផលប្រភេទ return type របស់ action method ដែលត្រូវផ្ញើទៅអោយ users លើពេលដែល action method របស់ controller ណាមួយត្រូវបានដំណើរការ។

## 6.ActionResult

- Producing the HTML
- Redirecting the Users
- Returning Files
- Content Results
- Returning Errors and HTTP Codes
- Security Related Results

## 6.1. Produce the HTML

- `ViewResult( )`
  - `View( )` ជា helper method ដែលវាស្វ័យប្រវត្តិរក View នៅក្នុងទីតាំង `Views/<Controller> folder` ដែល `*.cshtml` ស្ថិតនៅនិងបកប្រែដោយប្រើ `Razor view engine`។
- `PatialViewResult( )`
  - `PatialView( )` វាប្រើ model ដើម្បីបង្ហាញផ្នែកណាមួយនៃ View។

## 6.2. Redirecting the Users

- `RedirectResult( )`
  - `Redirect( )`
- `LocalRedirectResult( )`
  - `LocalRedirect( )`
- `RedirectToActionResult( )`
  - `RedirectToAction( )`
- `RedirectToRouteResult( )`
  - `RedirectToRoute( )`

## 6.3. Returning Files

- **FileResult**
- **FileContentResult**
- **FileStreamResult**
- **PhysicalFileResult**
- **သို့မဟုတ် VirtualFileResult**

## 6.4. Content Results

- **JsonResult**
- **ContentResult**
- **သို့မဟုတ် EmptyResult**



## 6.5. Returning Errors and HTTP Codes

- **StatusCodeResult**
- **ObjectResult**
- **OkResult**
- **OkObjectResult**
- **CreatedResult**
- **CreatedAtActionResult**
- **CreatedAtRouteResult**
- **BadRequestResult**
- **BadRequestObjectResult**
- **NotFoundResult**
- **NotFoundObjectResult**
- **UnsupportedMediaTypeResult**
- **NoContentResult**

## 6.6. Security Related Results

- **SignInResult**
- **SignOutResult**
- **ForbiddenResult**
- **ChallengeResult**
- **သို့မဟုတ် UnauthorizedResult**

# Action Result Summary

ActionResult	Helper Method	Description
ViewResult	View	បោះចំលើងជា HTML markup language
PartialViewResult	PartialView	បង្ហាញជា បំណែកតូចនៃ web page( partial view )
RedirectResult	Redirect	បោះចំលើងនូវការទៅទីតាំងណាមួយ
RedirectToRouteResult	RedirectToRoute	ប្រើសម្រាប់ទៅទីតាំងណាមួយ
RedirectToActionResult	RedirectToAction	ប្រើសម្រាប់ទៅ action ណាមួយ
ContentResult	Content	បោះចំលើងអក្សរធម្មតា( plain text )
JsonResult	Json	បោះចំលើងជា Json
EmptyResult	( None )	បោះចំលើងទទេ