



พัฒนาเว็บแอปพลิเคชันด้วย
ASP.NET Core MVC (.NET7)

รู้จักกับ .NET Core

รู้จักกับ .NET Core



ในอดีตตั้งแต่ปี 2002 การพัฒนา
แอปพลิเคชันสาย .NET (Web ,
Desktop) จะทำงานผ่านแพลตฟอร์ม
ที่ชื่อว่า “ .NET Framework ”

รู้จักกับ .NET Core



ใช้ภาษา C# , Visual Basic พร้อม
เครื่องมือที่ชื่อว่า Visual Studio ในการ
พัฒนาโปรเจกต์

ซึ่งการเขียนโปรแกรมนั้นรองรับ
การทำงานแค่ใน Windows เท่านั้น

รู้จักกับ .NET Core



ในปี 2017 ได้มีการพัฒนา .NET อีกรูปแบบหนึ่งขึ้นมาเรียกว่า .NET Core

ทำให้โปรแกรมเมอร์สาย .NET นั้น
ไม่จำเป็นต้องเขียนโปรแกรมใน Windows
อย่างเดียว สามารถเขียนโปรแกรมบน
Windows , macOS , Linux ได้

.NET Framework และ .NET Core



- รองรับการทำงานบน Windows เท่านั้น
- ใช้ Visual Studio (2002 - ปัจจุบัน)



- Cross Platform (รองรับการทำงานบน Windows , macOS , Linux)
- ใช้ Visual Studio (2017 - ปัจจุบัน)

รายละเอียดการอัปเดตเวอร์ชัน

.NET and .NET Core

.NET and .NET Core refer to several technologies including the runtime, the SDK, ASP.NET Core, and Entity Framework Core.

Supported versions

The following table tracks release and end of support dates for .NET and .NET Core versions.

Version	Original release date	Latest patch version	Patch release date	Release type	Support phase	End of support
.NET 7	November 8, 2022	7.0.1	December 13, 2022	STS	Active	May 14, 2024
.NET 6	November 8, 2021	6.0.12	December 13, 2022	LTS	Active	November 12, 2024

<https://dotnet.microsoft.com/en-us/platform/support/policy>



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

ต้องมีพื้นฐานอะไรบ้าง



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

ต้องมีพื้นฐานอะไรบ้าง

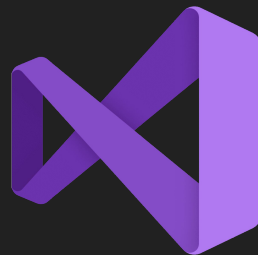


- มีพื้นฐานการเขียนโปรแกรมด้วยภาษา C#
- มีพื้นฐานการเขียนโปรแกรมเชิงวัตถุด้วยภาษา C#
(Class , Method , Object)
- มีพื้นฐานการจัดการฐานข้อมูลเบื้องต้น (SQL)
- มีพื้นฐาน HTML5 , CSS3 , JavaScript , Bootstrap 5
สำหรับนำมาสร้างเว็บแอปพลิเคชัน

เครื่องมือพื้นฐาน

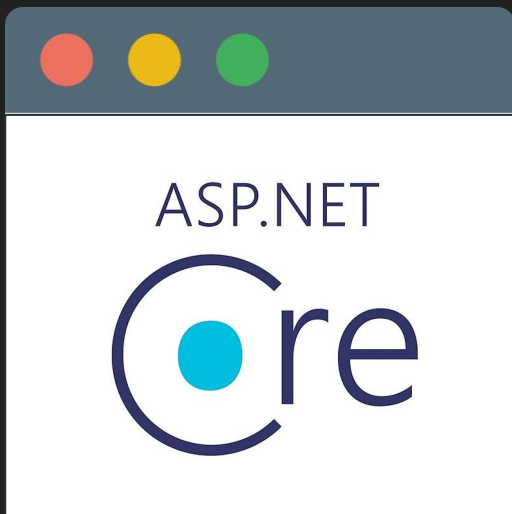
เครื่องมือพื้นฐาน

- Visual Studio 2022
- SQL Server 2022
- Google Chrome



รู้จักกับ ASP.NET Core MVC

ASP.NET Core MVC คืออะไร



เฟรมเวิร์คในการพัฒนาเว็บแอปพลิเคชัน
ที่มีทั้งระบบหน้าบ้าน (Front-End) และ
ระบบหลังบ้าน (Back-End) รวมอยู่ใน
โปรเจกต์เดียวกัน โดยใช้สถาปัตยกรรม
“ MVC (Model-View-Controller) ”

Frontend VS Backend



Frontend คือ การพัฒนาโปรแกรมระบบหน้าบ้าน (UI : User Interface หรือ หน้าตาของแอปพลิเคชัน) โดยผู้ใช้งานสามารถมองเห็นและมีส่วนร่วมหรือโต้ตอบภายใน Web Browser ได้



Backend คือ การพัฒนาโปรแกรมหลังบ้านหรือการทำงานเบื้องหลังในแอป เช่น การทำงานกับฐานข้อมูล เป็นต้น โดยผู้ใช้งานไม่สามารถมีส่วนร่วมหรือโต้ตอบได้



<https://www.youtube.com/c/KongRuksiamOfficial/>



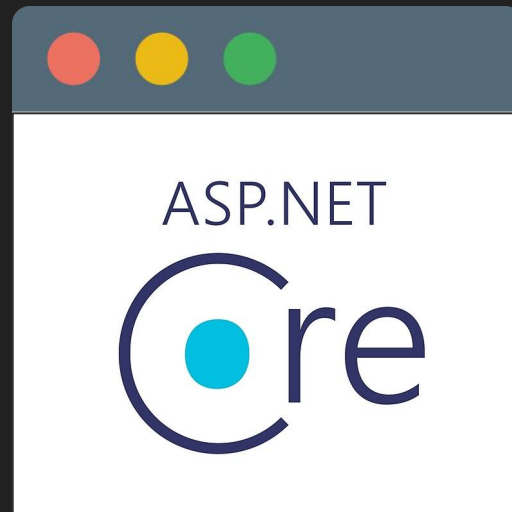
<https://www.facebook.com/KongRuksiamTutorial/>

ภาพรวมระบบ

ภาพรวมระบบ



Frontend (Client)

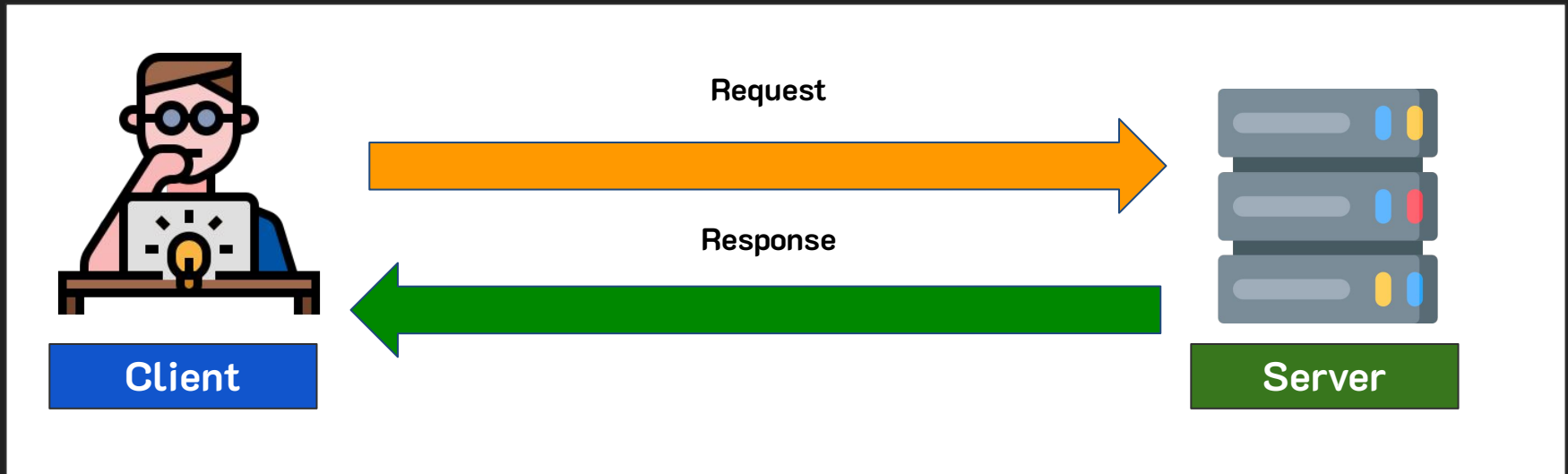


Backend (Server)

คำศัพท์พื้นฐาน

- Client - (ผู้ใช้บริการ)
- Server - (ผู้ให้บริการ)
- Request - (คำขอในการเข้าถึง)
- Response - (ตอบกลับคำขอ)

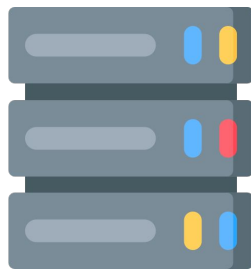
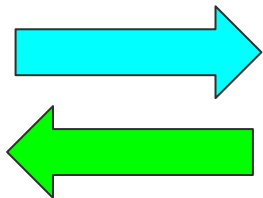
ภาพรวมระบบ



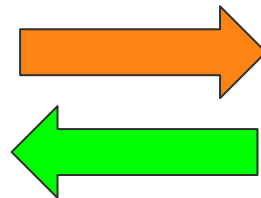
ภาพรวมระบบ



Client



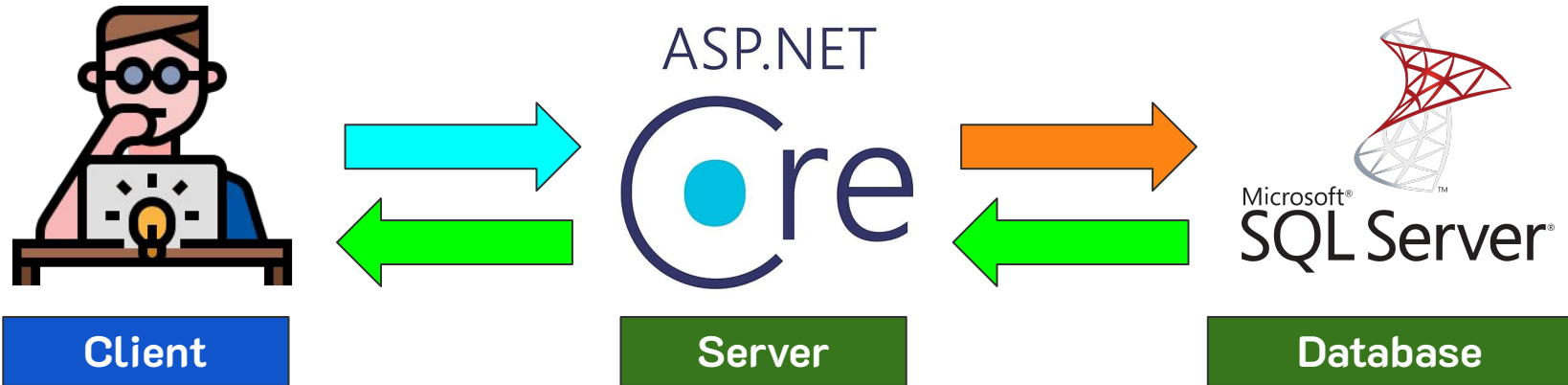
Server

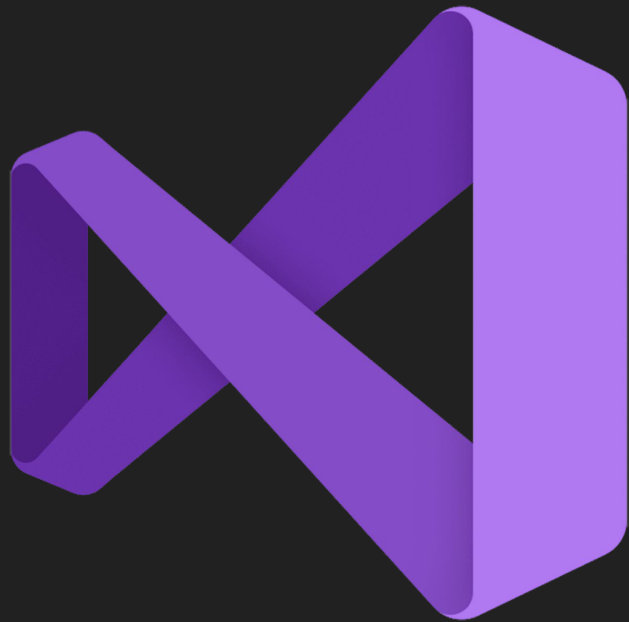


Database



ภาพรวมระบบ



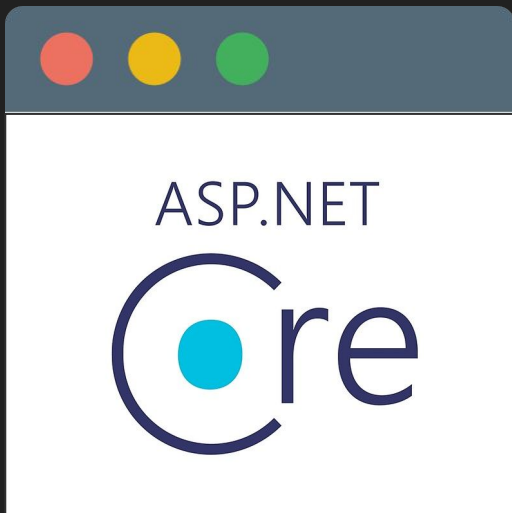


ติดตั้ง Visual Studio

รู้จักกับ MVC

(Model-View-Controller)

ASP.NET Core MVC คืออะไร

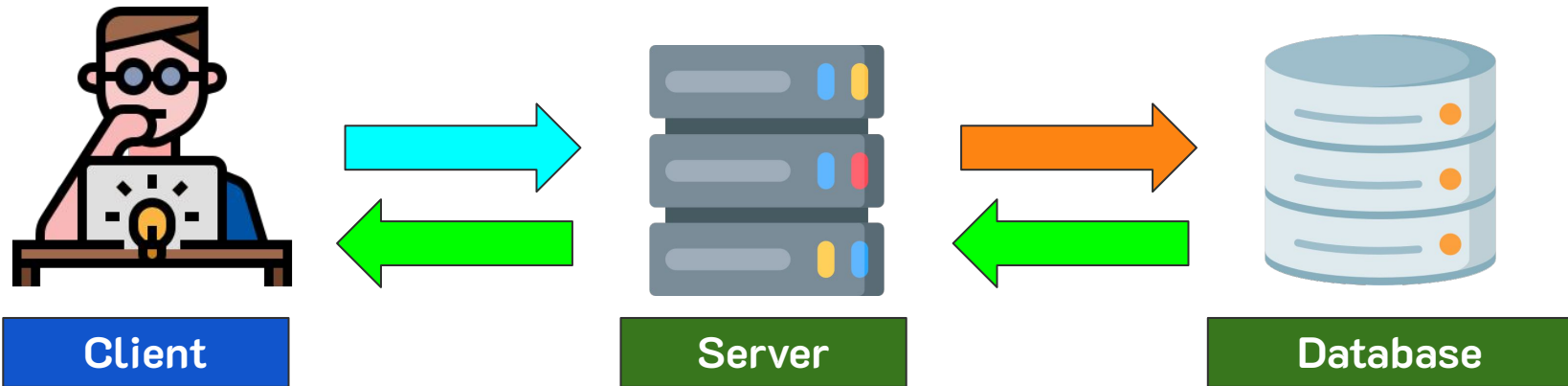


เฟรมเวิร์คในการพัฒนาเว็บแอปพลิเคชัน
ที่มีทั้งระบบหน้าบ้าน (Front-End) และ
ระบบหลังบ้าน (Back-End) รวมอยู่ใน
โปรเจกต์เดียวกัน โดยใช้สถาปัตยกรรม
“ MVC (Model-View-Controller) ”

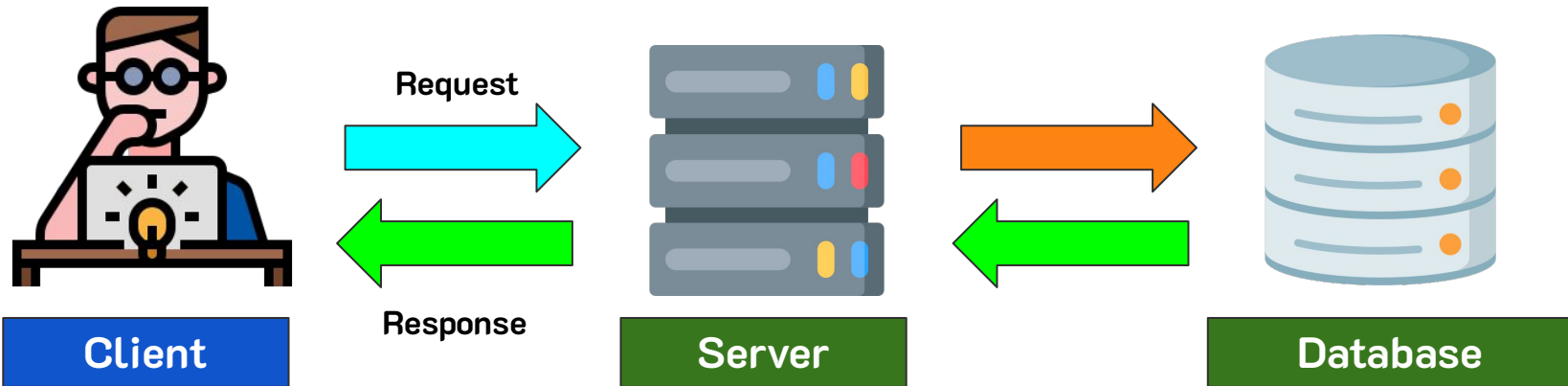
รู้จักกับ MVC

- **Model (M)** คือ ส่วนที่เก็บข้อมูลของ Application (Back-End)
- **View (V)** คือ ส่วนที่ทำงานฝั่ง Front-End หรือหน้าต่างแอปพลิเคชัน เป็นส่วนที่ไว้ใช้แสดงผลข้อมูลด้วย HTML ร่วมกับ Tag Helper
- **Controller (C)** คือ ส่วนประมวลผลคำสั่งต่างๆ ในแอปพลิเคชัน โดยควบคุมการทำงานระหว่าง Model และ View (Back-End)

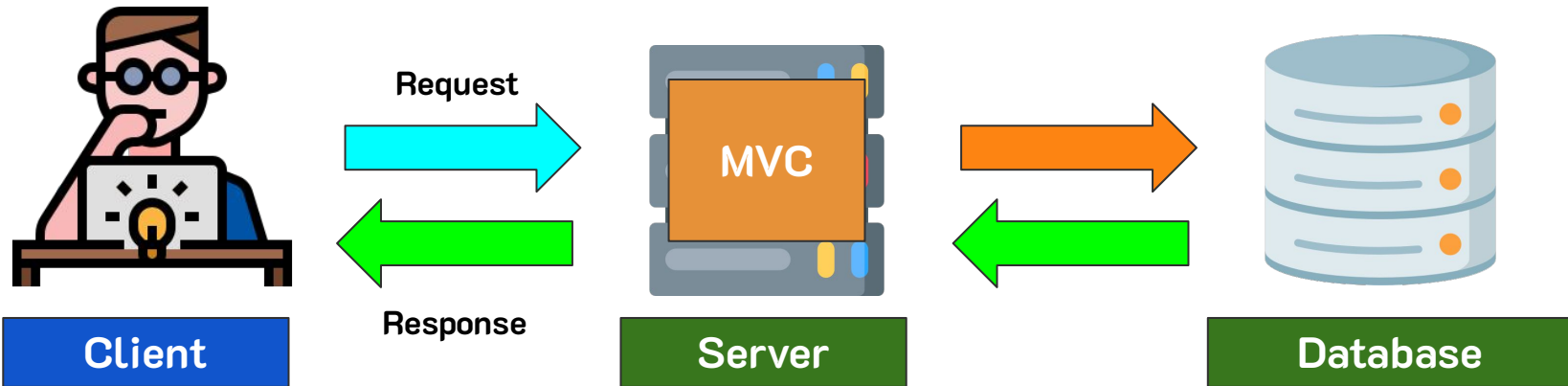
รู้จักกับ MVC



รู้จักกับ MVC



รู้จักกับ MVC



รู้จักกับ MVC



Client

Routing

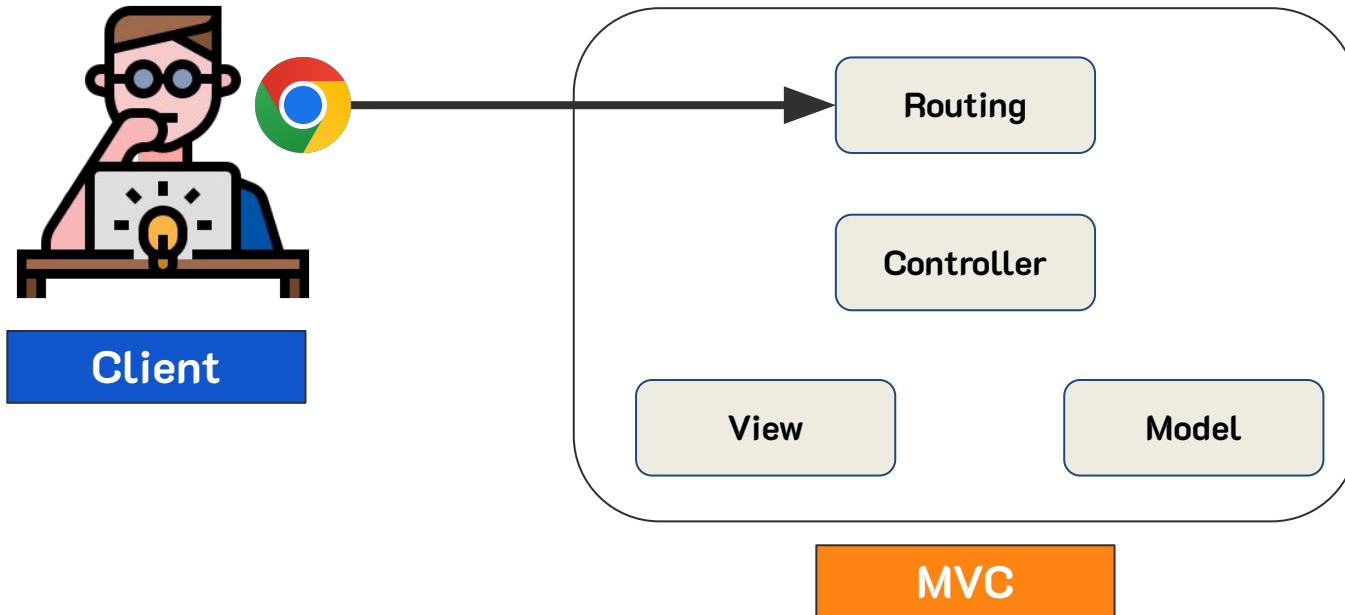
Controller

View

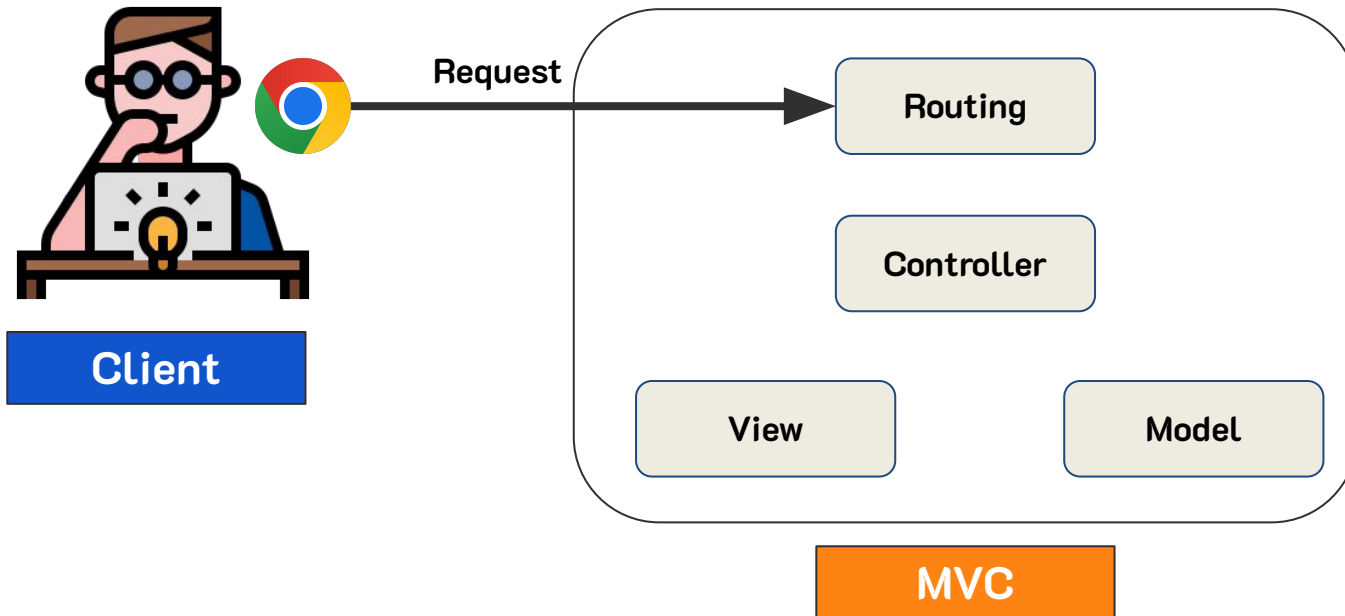
Model

MVC

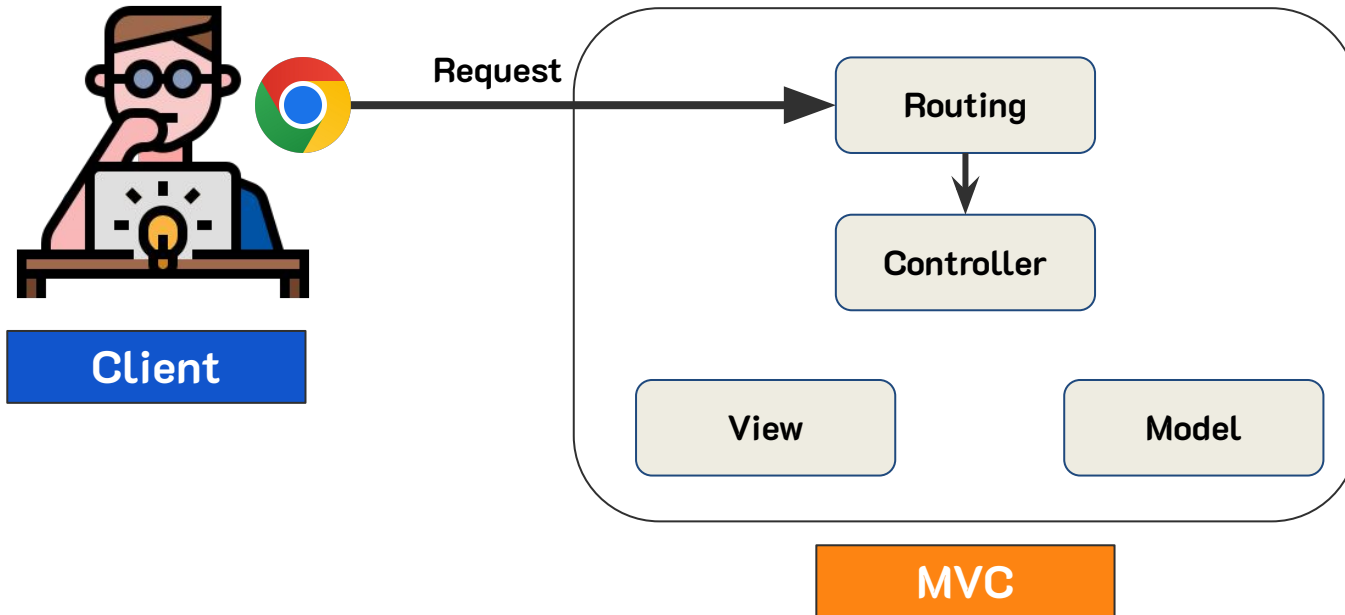
รู้จักกับ MVC



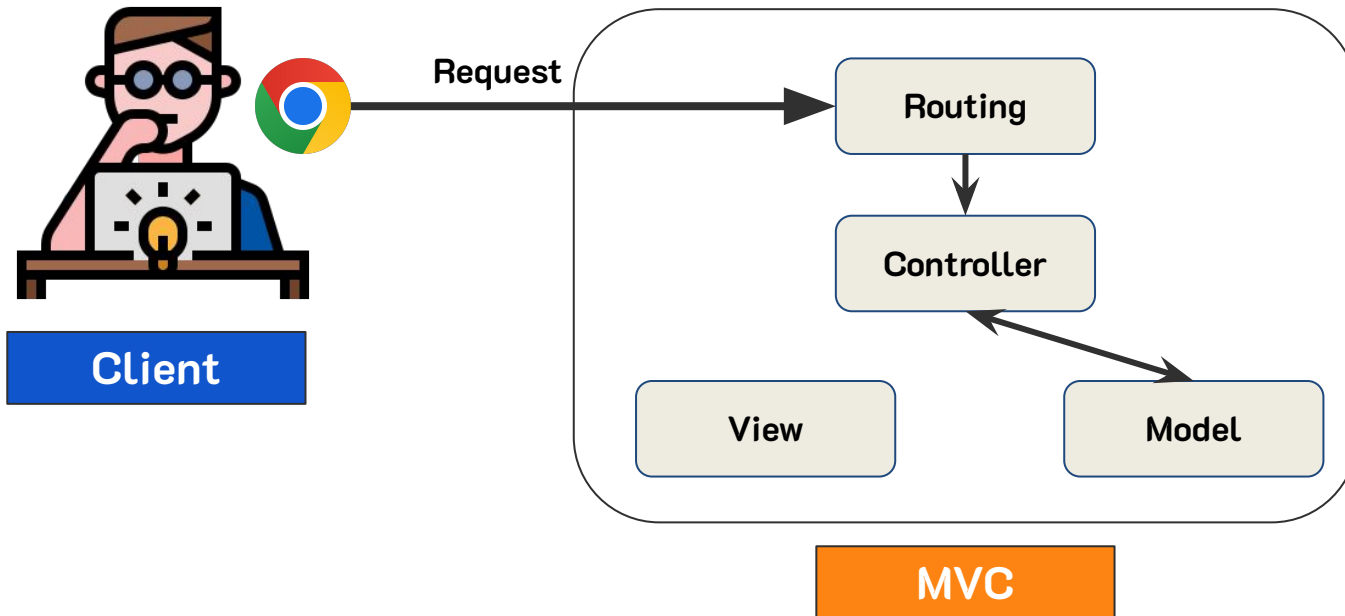
รู้จักกับ MVC



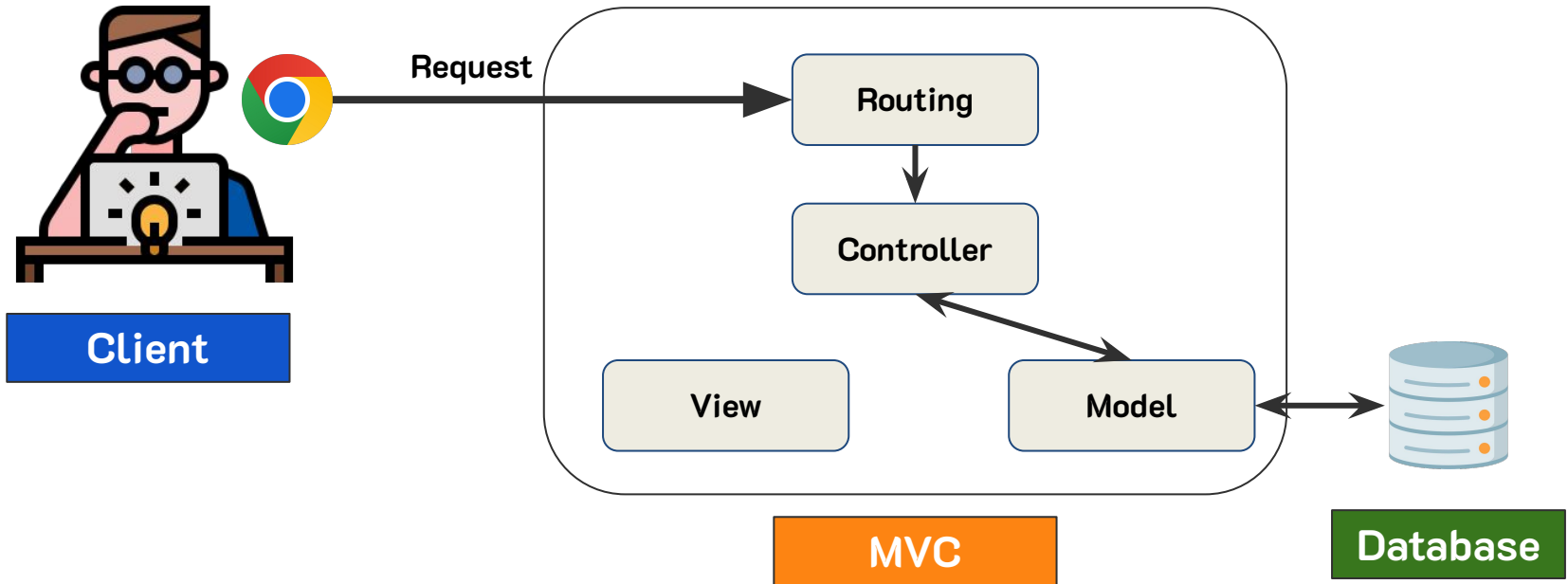
รู้จักกับ MVC



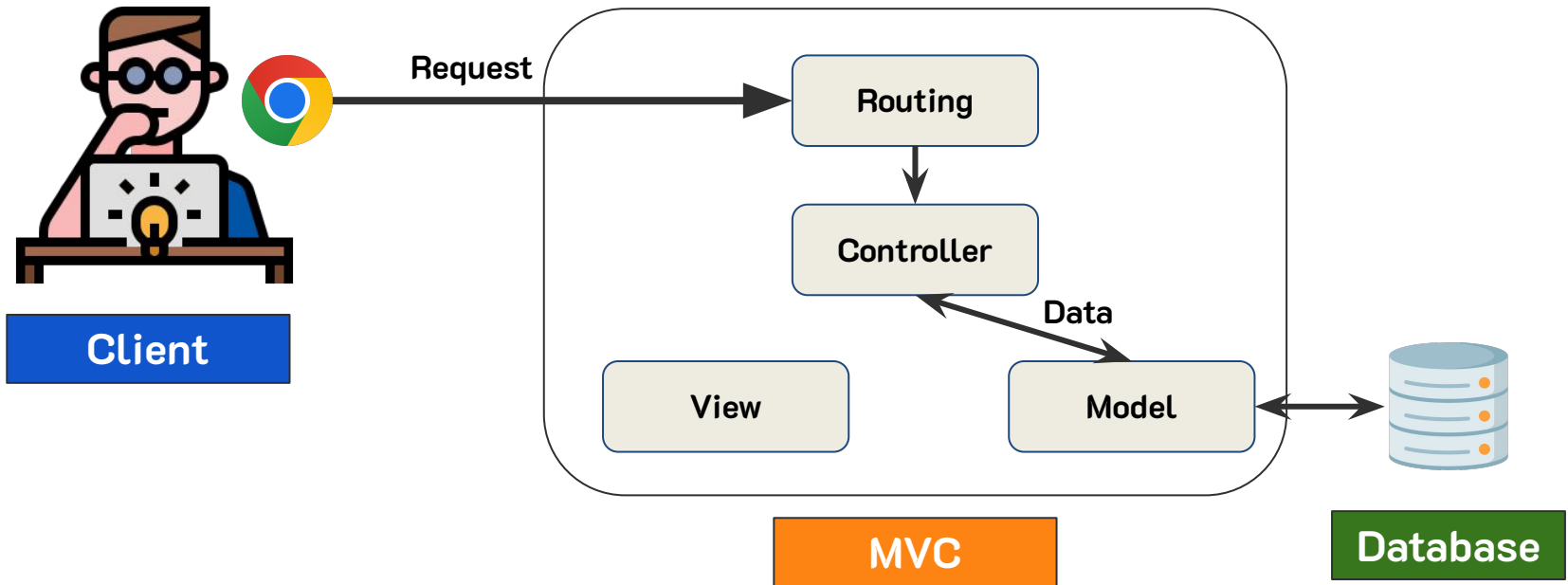
รู้จักกับ MVC



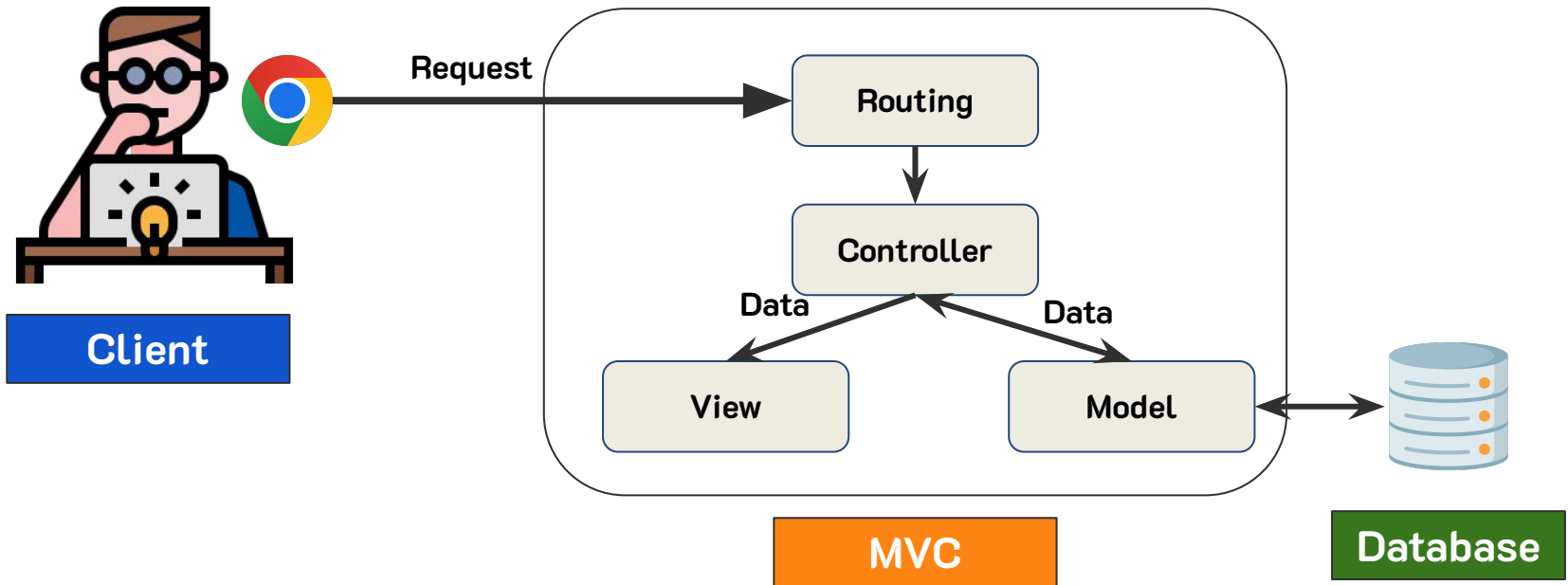
รู้จักกับ MVC



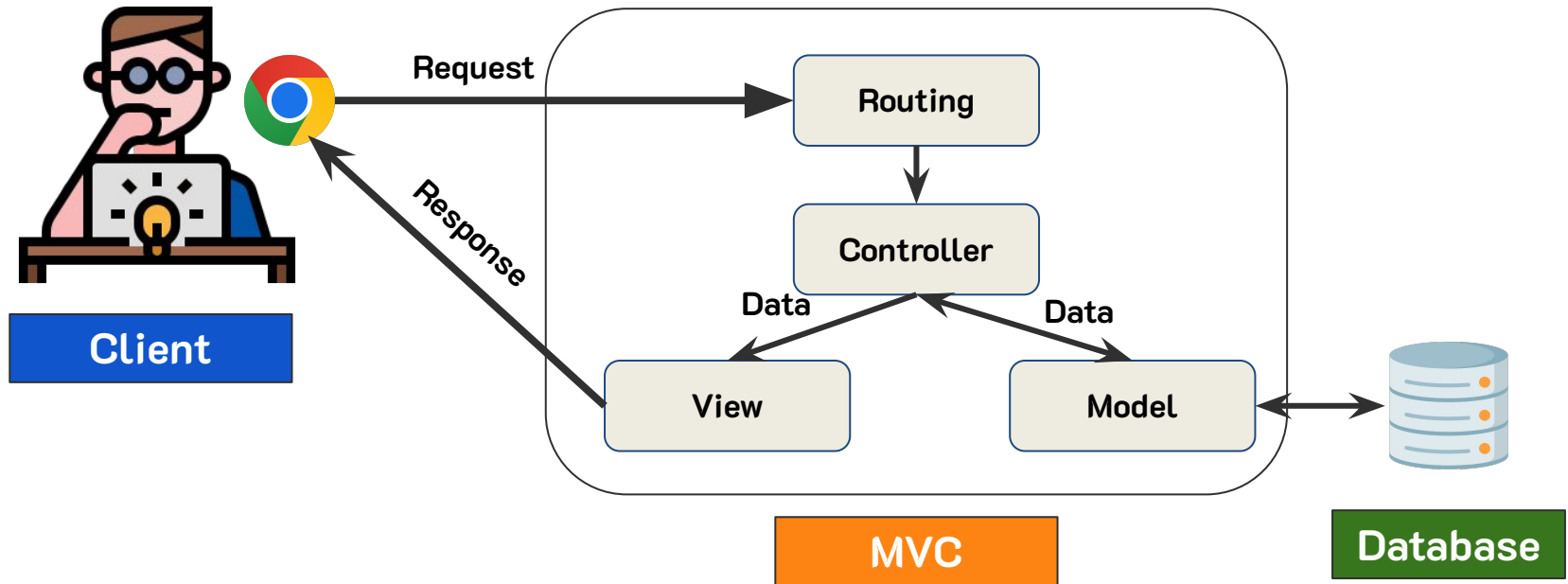
รู้จักกับ MVC



รู้จักกับ MVC



รู้จักกับ MVC



สร้างโปรเจกต์

โครงสร้างโปรเจกต์

โครงสร้างโปรเจกต์

- **Controllers** เป็นโฟลเดอร์ที่เก็บกลุ่ม Controller สำหรับกำหนดการทำงานหรือการประมวลผลข้อมูลในแอปพลิเคชัน
- **Models** เป็นโฟลเดอร์ที่เก็บกลุ่มไฟล์ Model เพื่อใช้งานฐานข้อมูล
- **Views** เป็นโฟลเดอร์ที่เก็บไฟล์สำหรับแสดงผลหน้าเว็บ (.cshtml)
- **wwwroot** เป็นโฟลเดอร์ที่ใช้จัดการ Static File (css , js)
- **Properties** โฟลเดอร์ที่เก็บการตั้งค่ารูปแบบการรันแอปพลิเคชันผ่านไฟล์ที่ชื่อว่า launchSettings.json

โครงสร้างโปรเจกต์

- **appsettings.json** เป็น Configuration File หลักของระบบ เช่นการเพิ่ม Environment Variable ที่ใช้ในโปรเจกต์
- การทำงานเริ่มต้นของโปรเจกต์ ASP.NET Core MVC จะอยู่ในไฟล์ **Program.cs** ซึ่งมีกลุ่มคำสั่งสำหรับจัดการ Request และ Response (**Middleware**) รวมถึงการตั้งค่า Service หรือบริการต่างๆที่ใช้งานในโปรเจกต์ เช่น Routing , Database Service , Static File , Authentication เป็นต้น

โครงสร้างโปรเจกต์

ไฟล์ Program.cs จะมีการตั้งค่าบริการเริ่มต้นสำหรับจัดการ
เส้นทาง (Route) และ คอนโทรลเลอร์ (Controller) ผ่านคำสั่ง

```
app.UseRouting();
```

```
app.MapControllerRoute(
```

```
    name: "default",
```

```
    pattern: "{controller=Home}/{action=Index}/{id?}"
```

```
);
```

โครงสร้างโปรเจกต์

ไฟล์ Program.cs จะมีการตั้งค่าบริการเริ่มต้นสำหรับจัดการ
เส้นทาง (Route) และ คอนโทรลเลอร์ (Controller) ผ่านคำสั่ง

```
app.UseRouting();
```

```
app.MapControllerRoute(
```

```
    name: "default",
```

```
    pattern: "{controller=Home}/{action=Index}/{id?}"
```

```
);
```

โครงสร้างโปรเจกต์

```
app.UseRouting();
```

```
app.MapControllerRoute(
```


```
    name: "default", ← ชื่อ Route
```

```
    pattern: "{controller=Home}/{action=Index}/{id?}"
```

```
);
```

โครงสร้างโปรเจกต์

```
app.UseRouting();  
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}"  
);
```




คอนโทรลเลอร์ (Controller) ที่ทำงาน
เป็นอันดับแรกในตอนเริ่มต้นรันแอปพลิเคชัน

- Home (เขียนแบบย่อ)
- HomeController (เขียนแบบเต็ม)

โครงสร้างโปรเจกต์

```
app.UseRouting();  
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}"  
);
```

- 
- action คือ กระบวนการทำงาน
 - Index () คือชื่อเมธอดที่ทำงานอยู่ใน Home (HomeController)

โครงสร้างโปรเจกต์

```
app.UseRouting();  
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}"  
);
```

พารามิเตอร์ที่ส่งเข้ามาทำงานในเมธอด
สามารถส่งค่าเข้าไปทำงานหรือไม่ส่งก็ได้
(? Optional Parameter)



Routing & Controller

Routing

- **Routing** คือ ส่วนที่ใช้ระบุเส้นทางในการรับส่งข้อมูล
- **Controller** คือ ศูนย์กลางสำหรับรับส่งข้อมูลและการประมวลผลโดยเชื่อมโยงการทำงานระหว่าง Model และ View

โครงสร้าง URL

<https://www.example.com/product/computer>

Routing

- **Routing** คือ ส่วนที่ใช้ระบุเส้นทางในการรับส่งข้อมูล
- **Controller** คือ ศูนย์กลางสำหรับรับส่งข้อมูลและการประมวลผลโดยเชื่อมโยงการทำงานระหว่าง Model และ View

โครงสร้าง URL

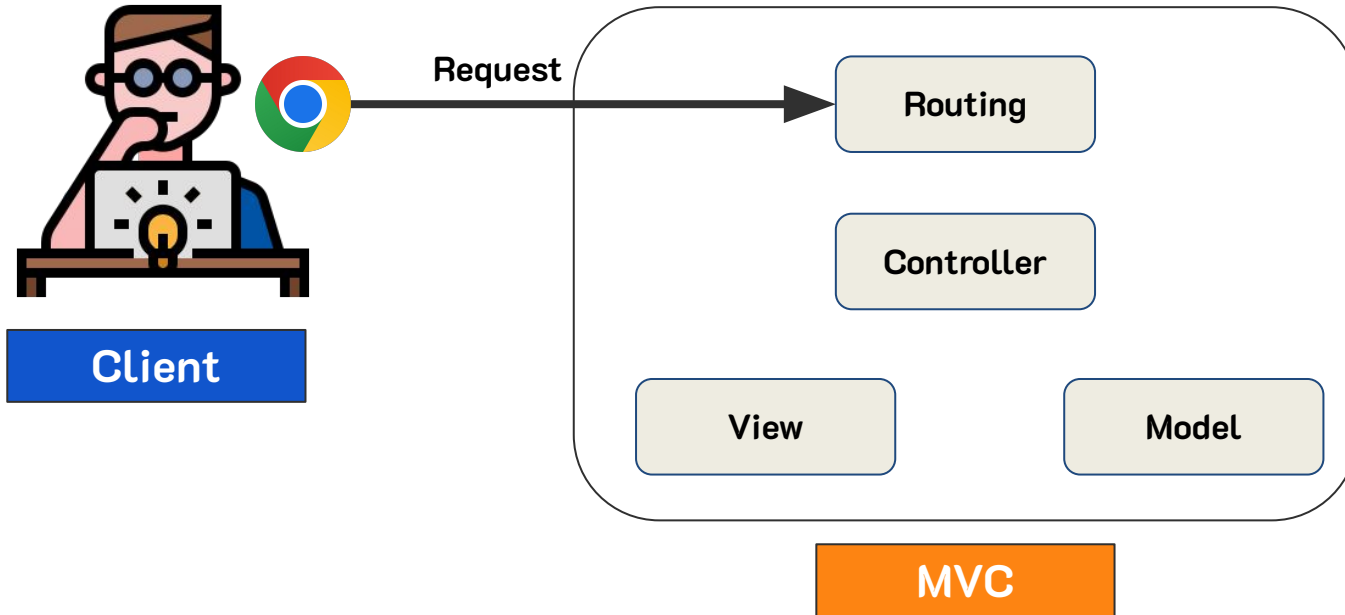
Protocol

Domain

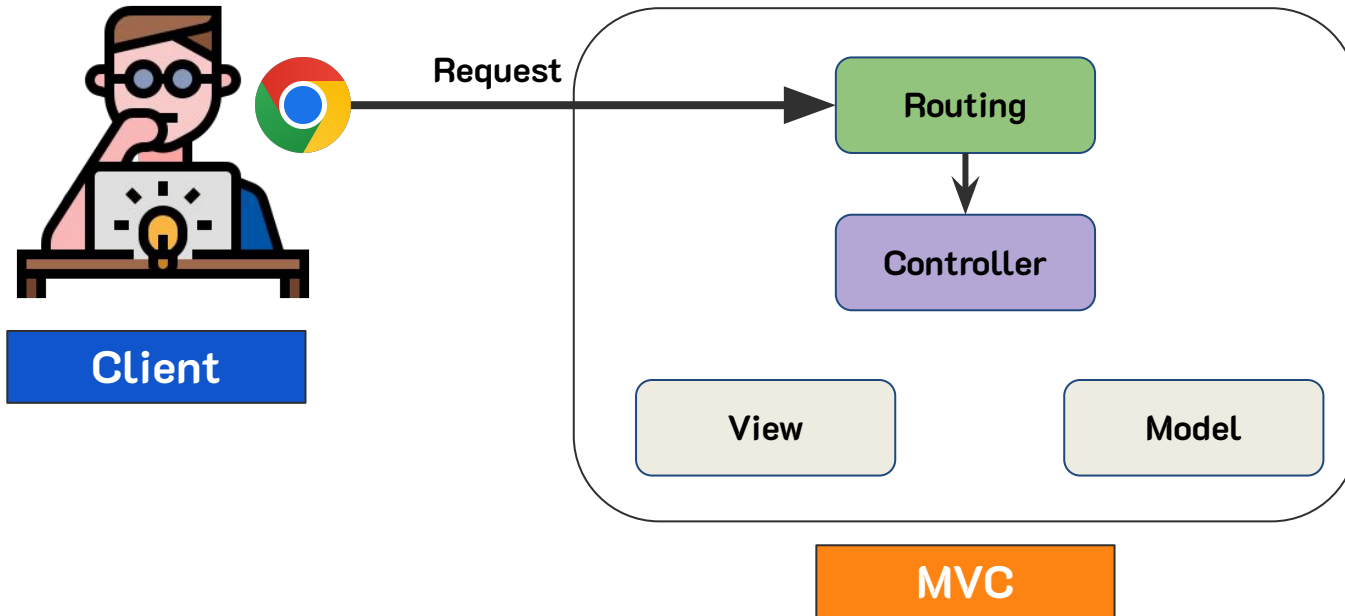
Path

<https://www.example.com/product/computer>

Routing



Routing



Routing & Controller

Controller

Action Method 1

Action Method 2

Action Method 3

Controller (C) คือ ส่วนประมวลผลคำสั่งต่างๆ ในแอปพลิเคชันโดยควบคุมการทำงานระหว่าง Model และ View

ภายใน Controller จะกำหนดกระบวนการทำงานผ่านส่วนที่เรียกว่าเมธอด (**Action Method**)

Routing & Controller

Controller

Action Method 1

Action Method 2

Action Method 3

Routing & Controller

Controller

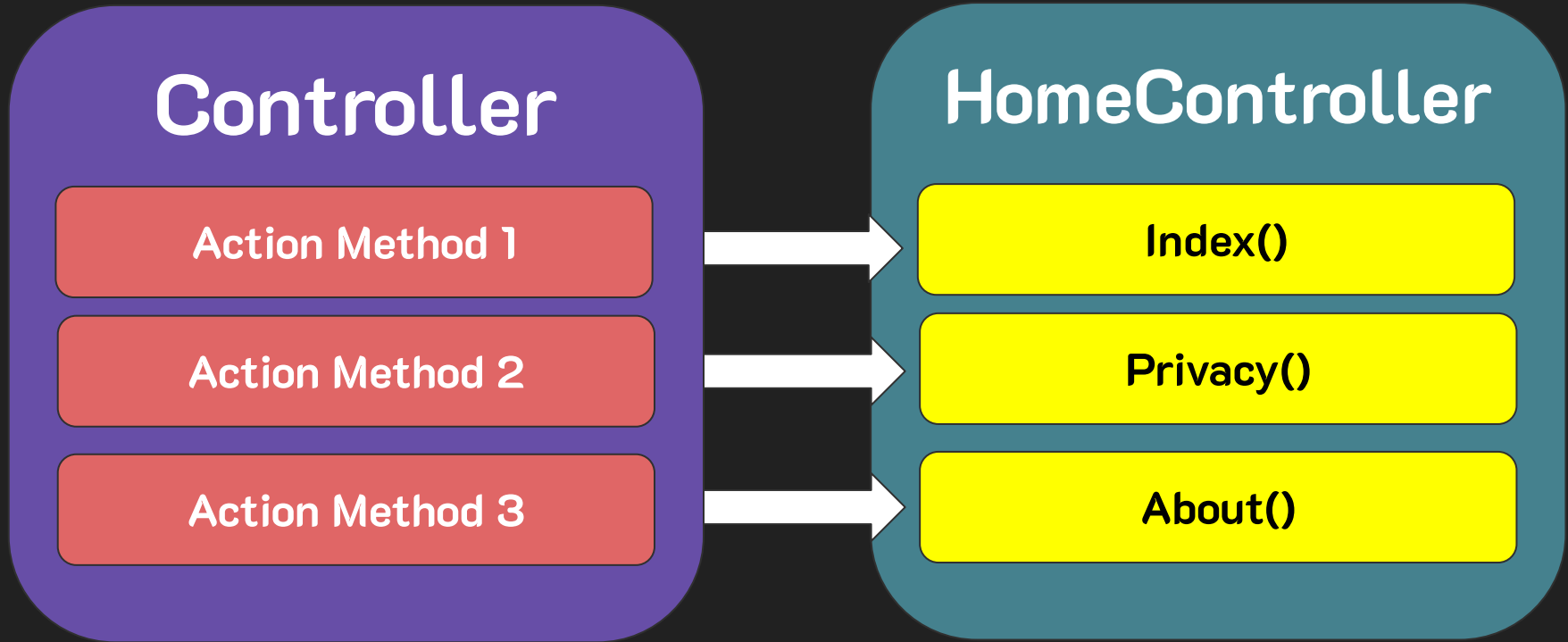
Action Method 1

Action Method 2

Action Method 3

HomeController

Routing & Controller



Routing & Controller

ตัวอย่าง URL	Controller	Action	ID
https://localhost:5000/home/index	HomeController	index	null
https://localhost:5000/home/privacy	HomeController	privacy	null
https://localhost:5000/product/edit/1	ProductController	edit	1
https://localhost:5000/student/detail/5	StudentController	detail	5

Routing & Controller

ตัวอย่าง URL	Controller	Action	ID
https://localhost:5000/home/index	HomeController	index	null
https://localhost:5000/home/privacy	HomeController	privacy	null
https://localhost:5000/product/edit/1	ProductController	edit	1
https://localhost:5000/student/detail/5	StudentController	detail	5

Routing & Controller

ตัวอย่าง URL	Controller	Action	ID
https://localhost:5000/home/index	HomeController	index	null
https://localhost:5000/home/privacy	HomeController	privacy	null
https://localhost:5000/product/edit/1	ProductController	edit	1
https://localhost:5000/student/detail/5	StudentController	detail	5

Routing & Controller

ตัวอย่าง URL	Controller	Action	ID
https://localhost:5000/home/index	HomeController	index	null
https://localhost:5000/home/privacy	HomeController	privacy	null
https://localhost:5000/product/edit/1	ProductController	edit	1
https://localhost:5000/student/detail/5	StudentController	detail	5

Routing & Controller

ตัวอย่าง URL	Controller	Action	ID
https://localhost:5000/home/index	HomeController	index	null
https://localhost:5000/home/privacy	HomeController	privacy	null
https://localhost:5000/product/edit/1	ProductController	edit	1
https://localhost:5000/student/detail/5	StudentController	detail	5

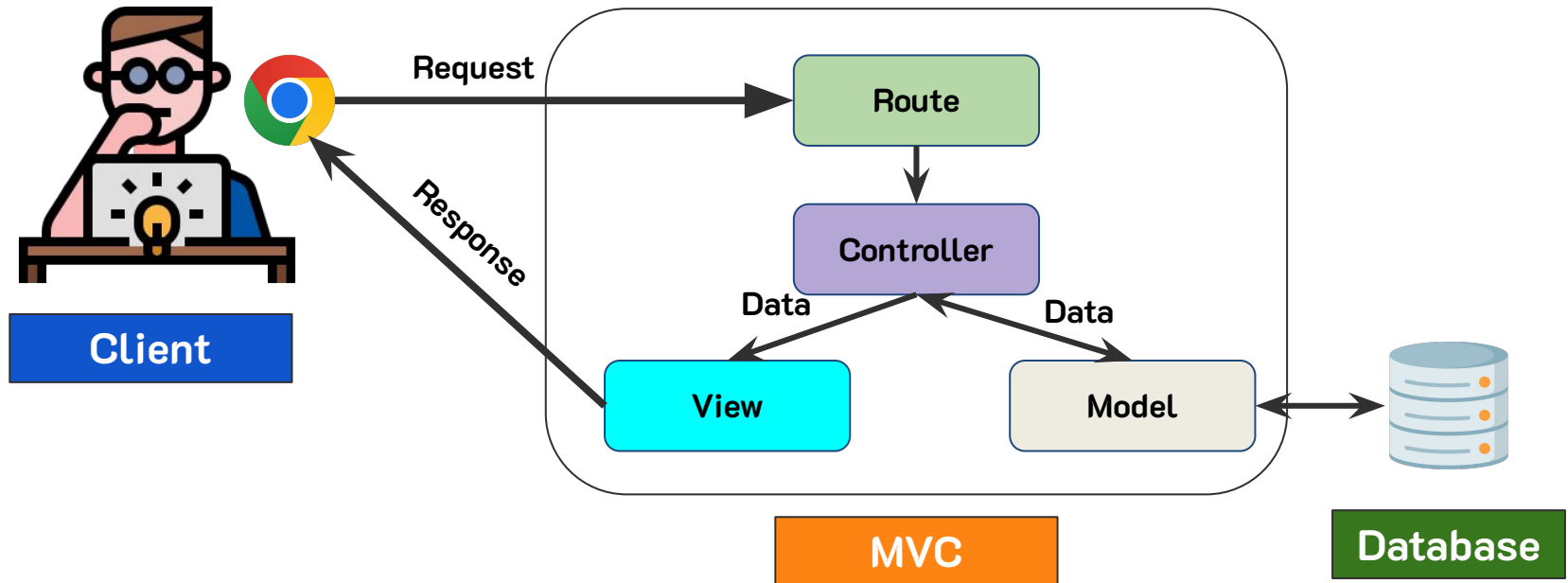
รู้จักกับ View

รู้จักกับ View

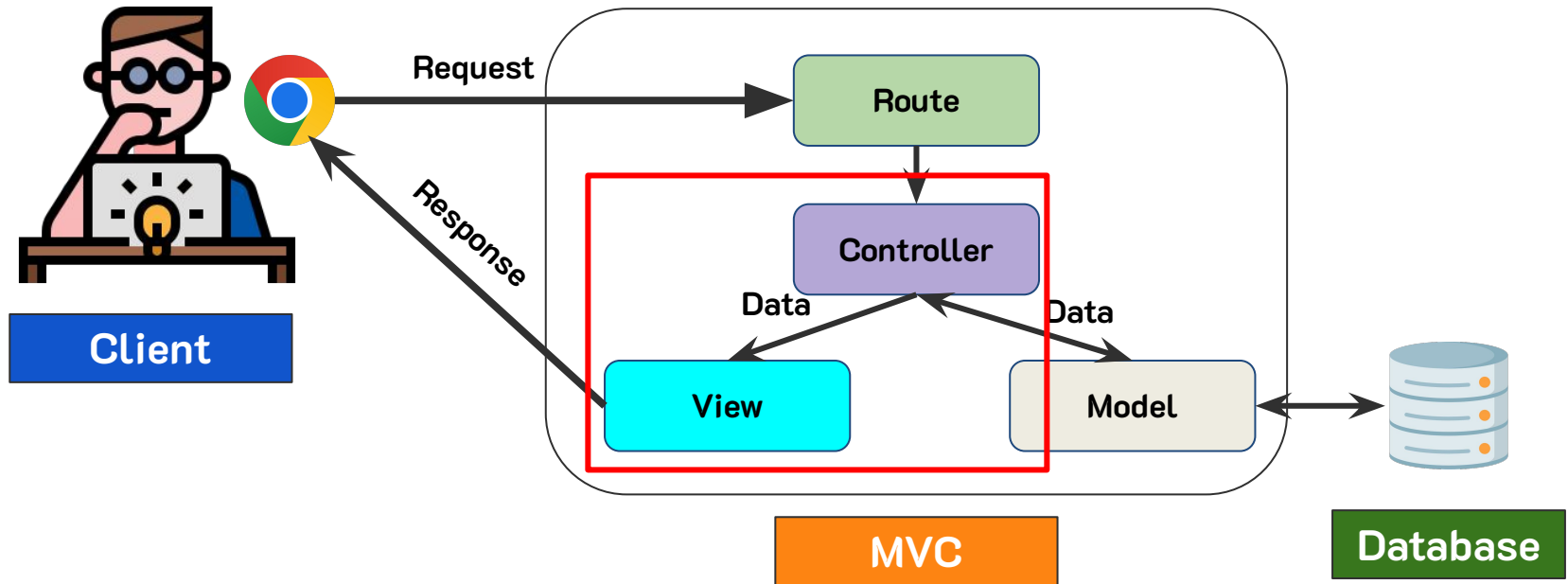


View คือหน้าต่างแอปพลิเคชันเป็นส่วนที่
ไว้ใช้แสดงผลข้อมูลหรือผลลัพธ์จาก
การประมวลผลในหน้าเว็บเพจโดยทำ
งานร่วมกับ HTML และ Tag Helper

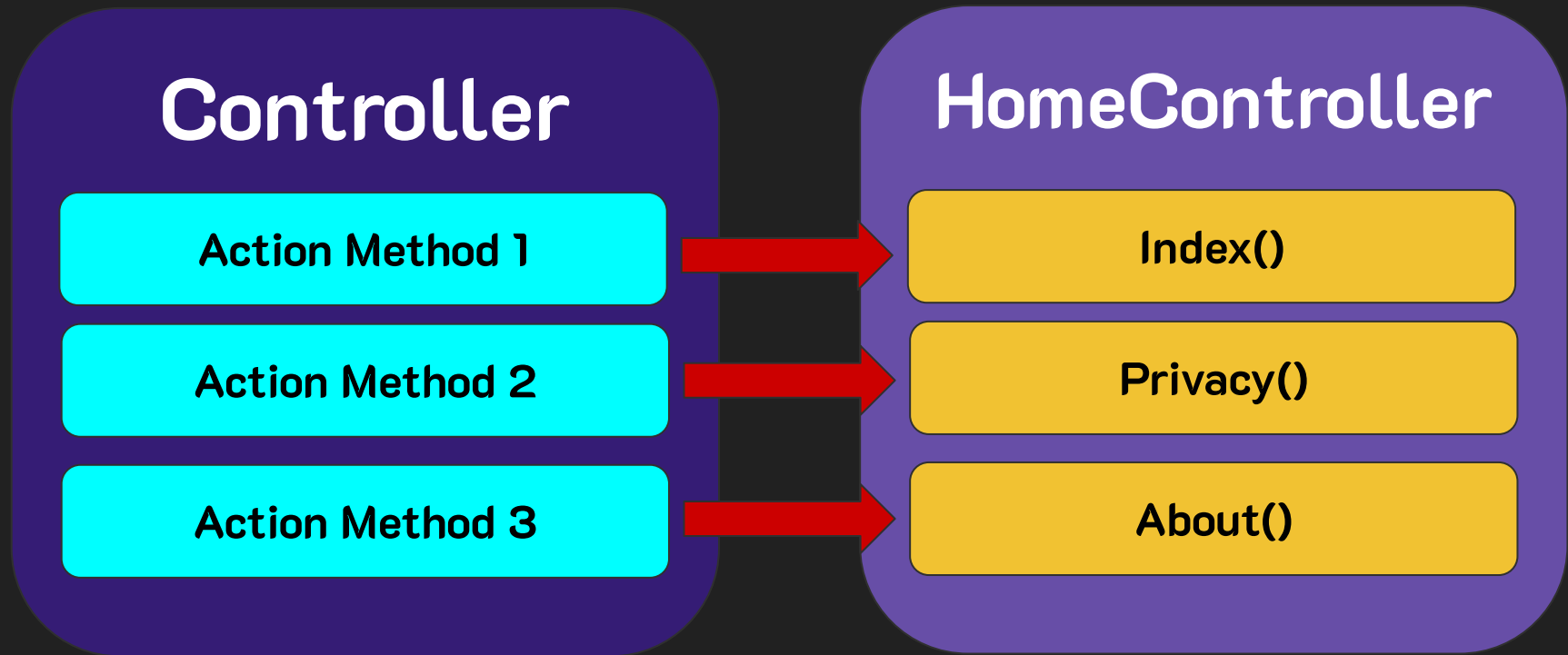
รู้จักกับ View



รู้จักกับ View



ความสัมพันธ์ระหว่าง View และ Controller



ความสัมพันธ์ระหว่าง View และ Controller

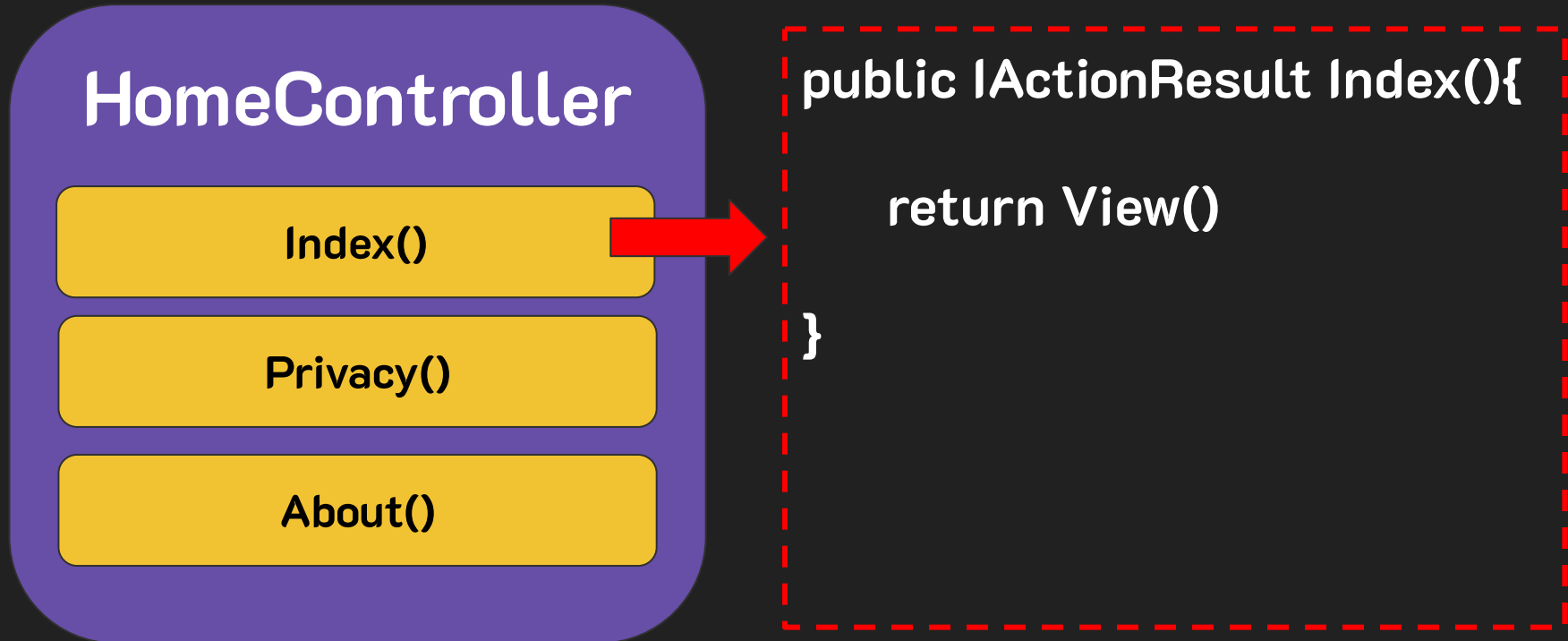
HomeController

Index()

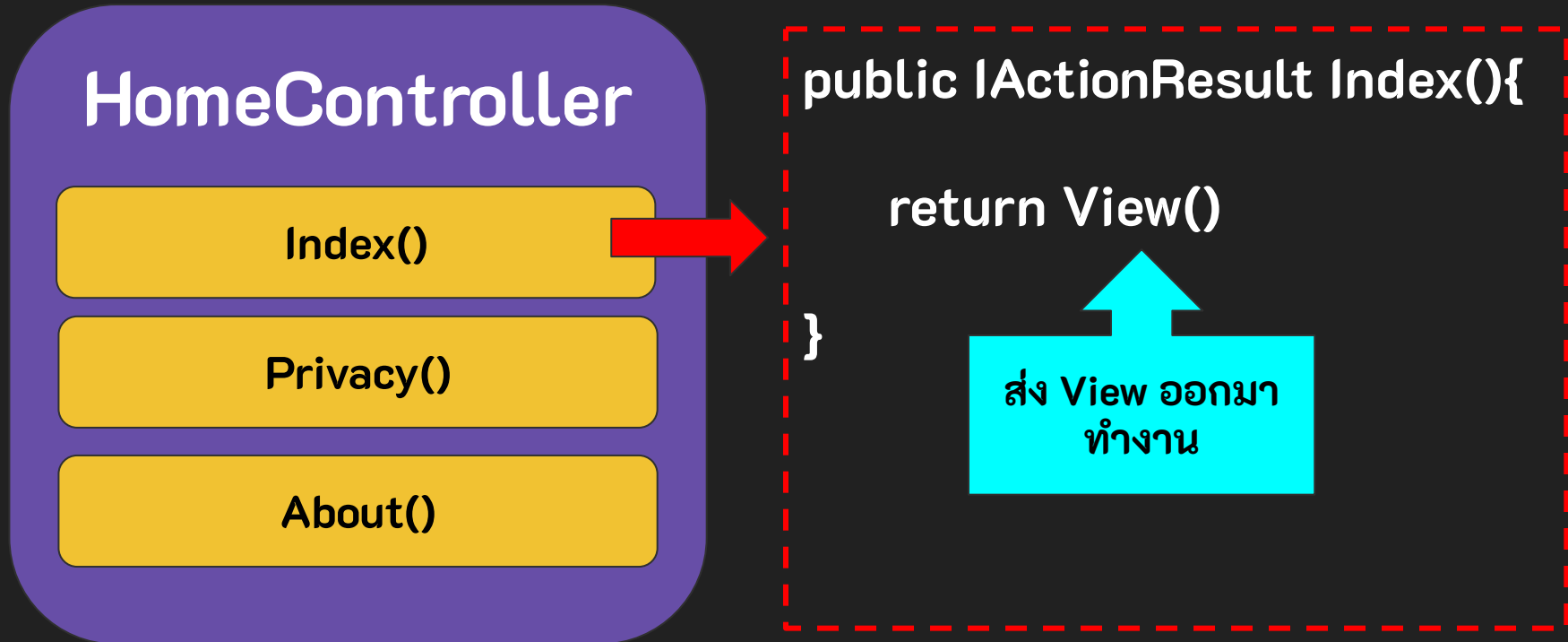
Privacy()

About()

ความสัมพันธ์ระหว่าง View และ Controller



ความสัมพันธ์ระหว่าง View และ Controller



ความสัมพันธ์ระหว่าง View และ Controller

HomeController

```
public IActionResult Index(){  
    return View()  
}
```

ความสัมพันธ์ระหว่าง View และ Controller

HomeController

```
public IActionResult Index(){  
    return View()  
}
```

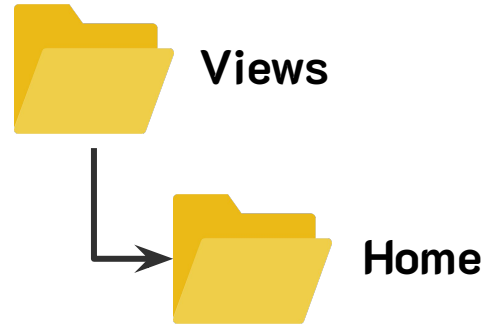


Views

ความสัมพันธ์ระหว่าง View และ Controller

HomeController

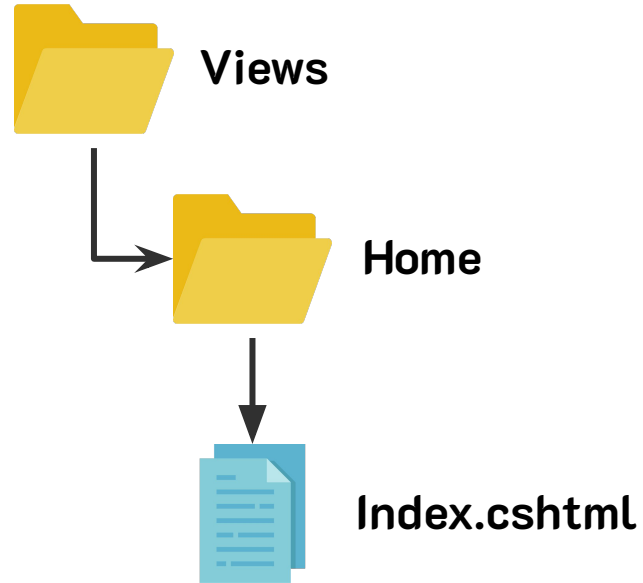
```
public IActionResult Index(){  
    return View()  
}
```



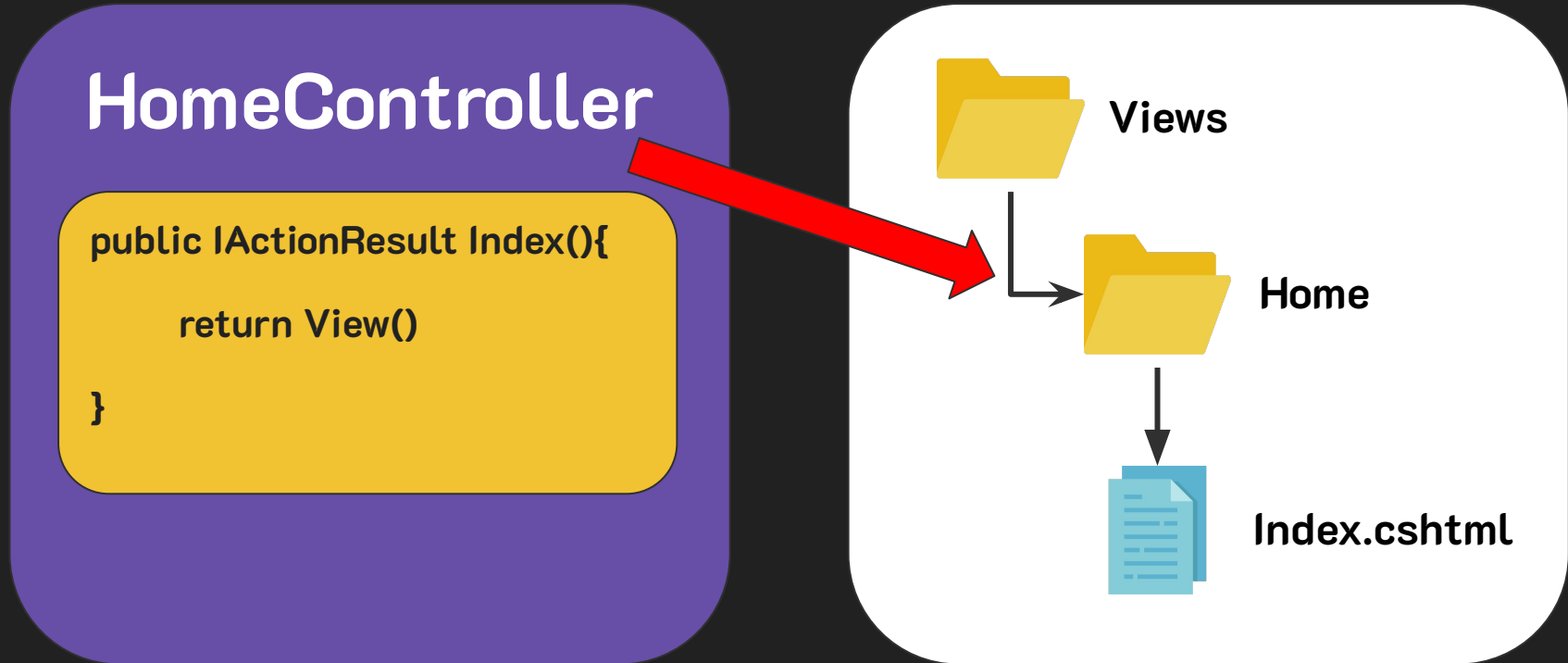
ความสัมพันธ์ระหว่าง View และ Controller

HomeController

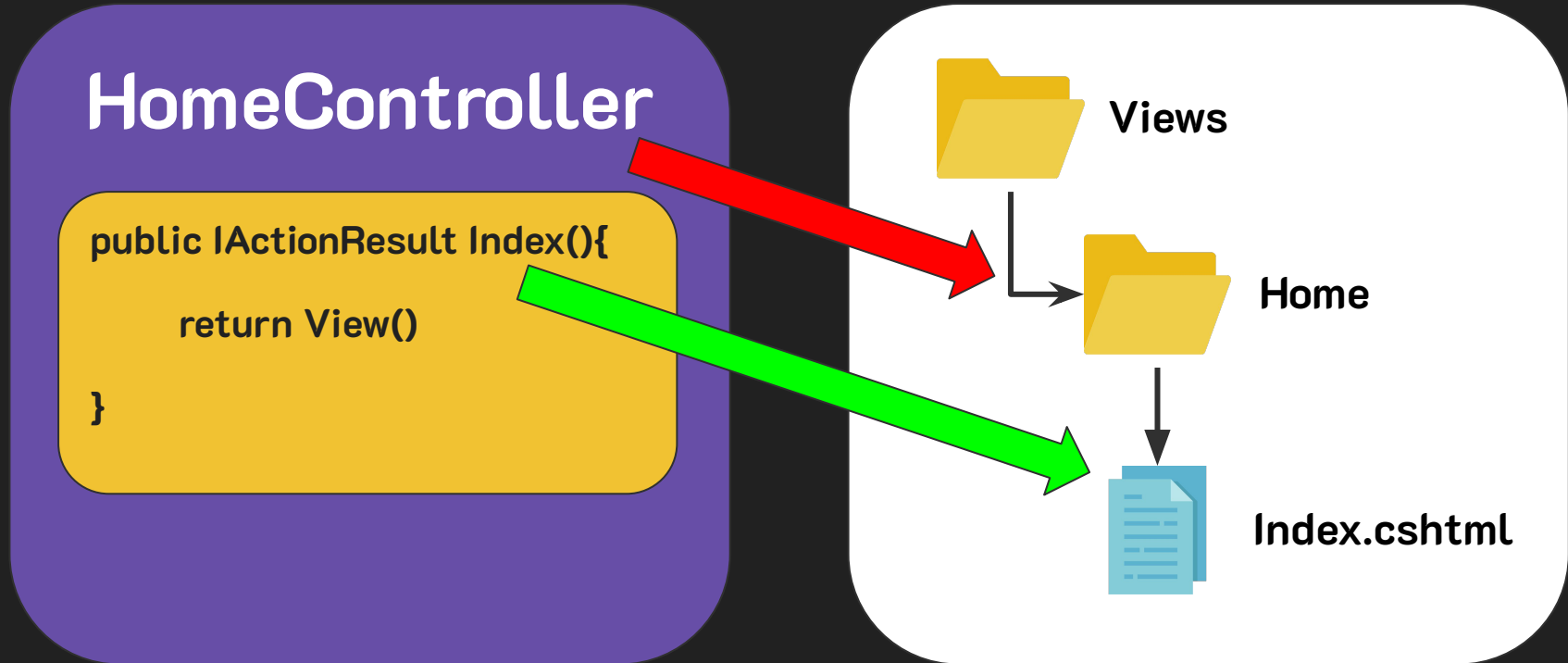
```
public IActionResult Index(){  
    return View()  
}
```



ความสัมพันธ์ระหว่าง View และ Controller



ความสัมพันธ์ระหว่าง View และ Controller



ความสัมพันธ์ระหว่าง View และ Controller

Controller	Action	Views
HomeController	index	Home/index.cshtml
HomeController	privacy	Home/privacy.cshtml
ProductController	computer	Product/computer.cshtml
StudentController	index	Student/index.cshtml

ความสัมพันธ์ระหว่าง View และ Controller

Controller	Action	Views
HomeController	index	Home/index.cshtml
HomeController	privacy	Home/privacy.cshtml
ProductController	computer	Product/computer.cshtml
StudentController	index	Student/index.cshtml

ความสัมพันธ์ระหว่าง View และ Controller

Controller	Action	Views
HomeController	index	Home/index.cshtml
HomeController	privacy	Home/privacy.cshtml
ProductController	computer	Product/computer.cshtml
StudentController	index	Student/index.cshtml

ความสัมพันธ์ระหว่าง View และ Controller

Controller	Action	Views
HomeController	index	Home/index.cshtml
HomeController	privacy	Home/privacy.cshtml
ProductController	computer	Product/computer.cshtml
StudentController	index	Student/index.cshtml

ความสัมพันธ์ระหว่าง View และ Controller

Controller	Action	Views
HomeController	index	Home/index.cshtml
HomeController	privacy	Home/privacy.cshtml
ProductController	computer	Product/computer.cshtml
StudentController	index	Student/index.cshtml

Layout Views

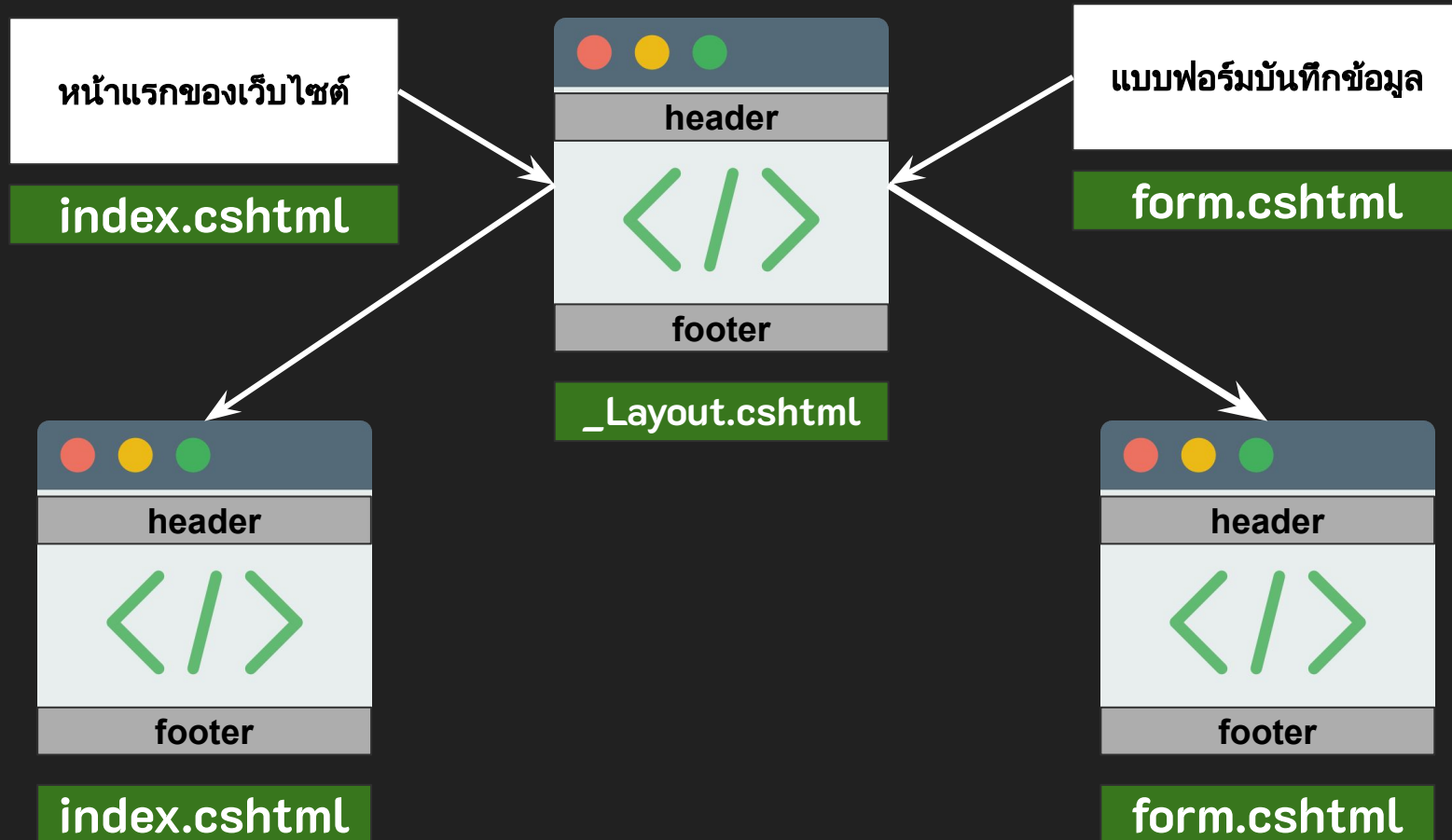
Layout Views

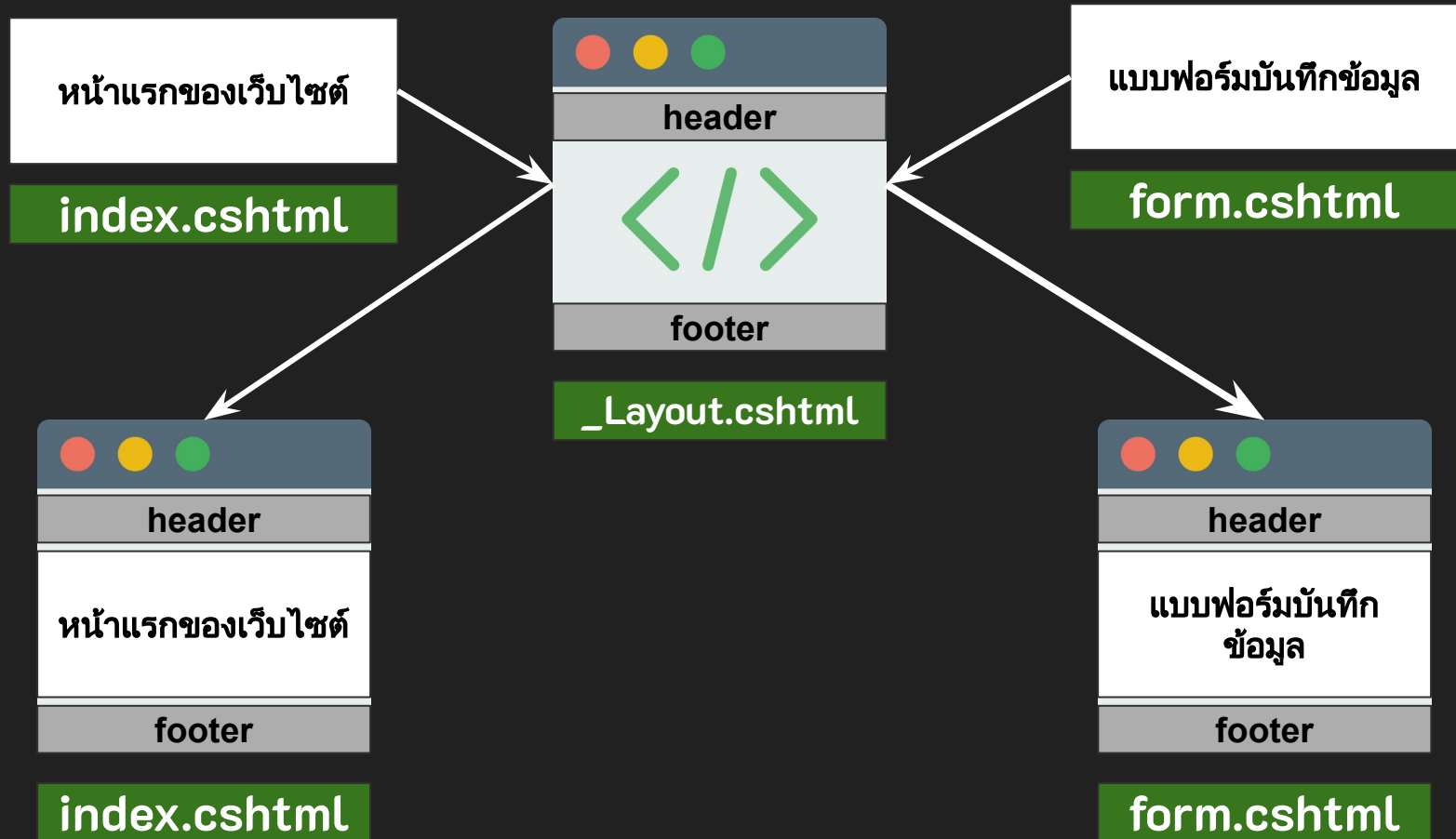
คือ การกำหนดโครงสร้างหลักในหน้าเว็บเพจที่ทุกๆหน้าใช้งานร่วมกันเพื่อลดความซ้ำซ้อนของโค้ด เช่น เมนู (Navbar) , Footer เป็นต้น

โดยการแทรกไฟล์เนื้อหาหลักเข้าไปทำงานในหน้าเว็บเพจ ในโปรเจกต์ ASP.NET Core MVC จะตั้งค่าการทำงานในไฟล์ที่ชื่อว่า

“_Layouts.cshtml”







โครงสร้างของไฟล์ .Layouts.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - BasicASPTutorial</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/BasicASPTutorial.styles.css" asp-append-version="true" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light" />
  </header>
  <div class="container">
    <main role="main" class="pb-3">
      @RenderBody()
    </main>
  </div>
  <footer class="border-top footer text-muted">
    <div class="container">...</div>
  </footer>
  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>
  @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

พื้นที่อ้างอิงตำแหน่งไฟล์ CSS (Bootstrap) และกำหนด title

ส่วนหัวของเว็บเพจ หรือเมนู (Navbar)

พื้นที่แสดงผลเนื้อหาที่ต่างกันในแต่ละหน้า @RenderBody()

พื้นที่อ้างอิงไฟล์ JavaScript Library (Footer)

Bootstrap คืออะไร



Frontend Framework ที่รวม HTML, CSS และ Javascript เข้าด้วยกัน เป็นชุดคำสั่งที่ถูกพัฒนาขึ้นมาเพื่อกำหนดมาตรฐานหรือรูปแบบการพัฒนาเว็บไซต์ในส่วนของ User Interface (UI) และทำให้เว็บรองรับการแสดงผลขนาดหน้าจอที่แตกต่างกันอีกด้วย

จุดเด่นของ Bootstrap



- มี UI ที่สวยงาม
- เรียนรู้ง่ายและเป็นที่นิยมใช้ทั่วโลกมีการพัฒนาและปรับปรุงต่อเนื่องจนถึงเวอร์ชัน 5
- ประหยัดเวลาในการพัฒนาเว็บไซต์
- รองรับการแสดงผลในอุปกรณ์ที่มีหน้าจอต่างขนาดกัน



Tag Helper

คือ การเชื่อมโยง Element ของ HTML5 ให้สามารถทำงานร่วมกับ .NET ได้ ตัวอย่าง เช่น

```
<a class="nav-link text-light"  
    asp-area=""  
    asp-controller="Student"  
    asp-action="Index">หน้าแรก</a>
```

Tag Helper

คือ การเชื่อมโยง Element ของ HTML5 ให้สามารถทำงานร่วมกับ .NET ได้ ตัวอย่าง เช่น

```
<a class="nav-link text-light"  
    asp-area=""  
    asp-controller="Student"  
    asp-action="Index">หน้าแรก</a>
```

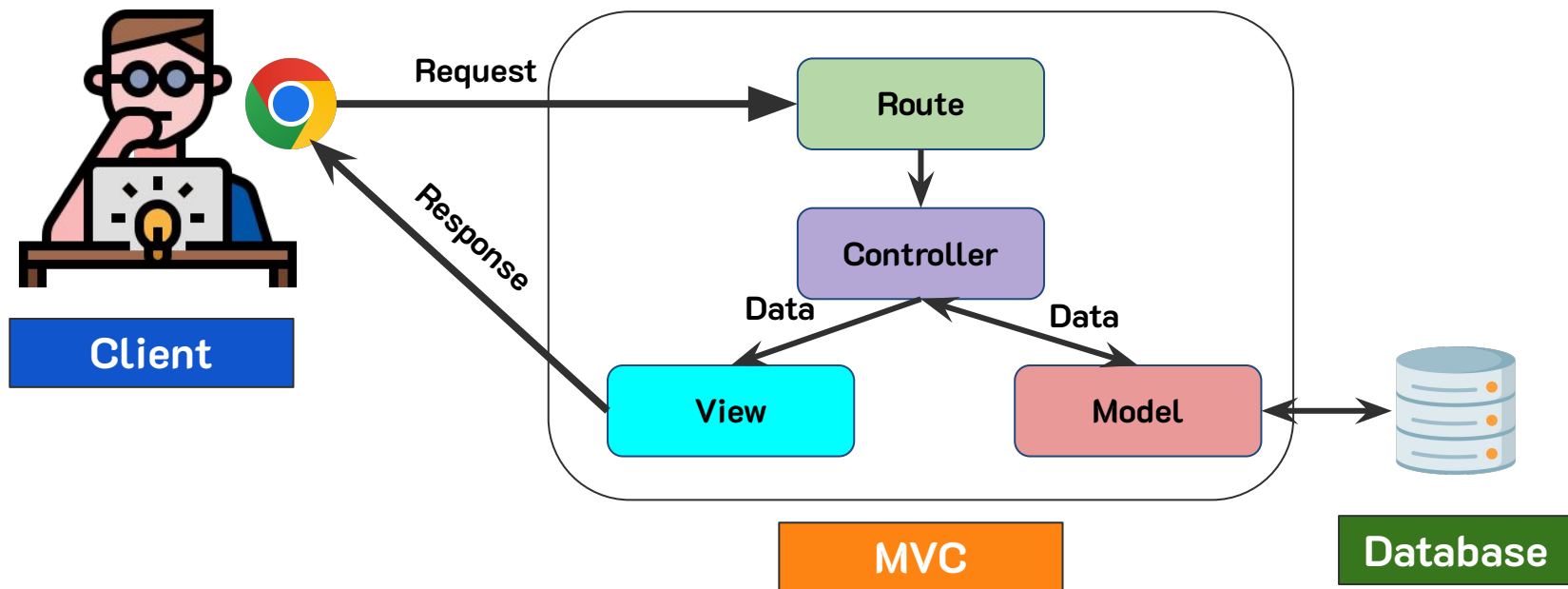

รู้จักกับโมเดล
(Model)

โมเดลคืออะไร

Model (M) คือ ส่วนที่เก็บข้อมูลของ Application โดยจะทำหน้าที่จัดการฐานข้อมูลแทนการใช้คำสั่ง SQL โดยตรง ผ่านการกำหนด Class และ Object ซึ่งจะเรียกส่วนนี้ว่า **ORM (Object Relational Mapping)**



รู้จักกับโมเดล



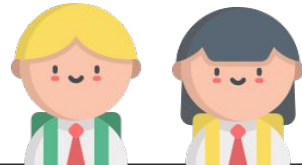
สร้างโมเดล (Model)

ข้อมูลนักเรียน (Student)

Property	Type	ความหมาย
ID	int	รหัสนักเรียน
Name	string	ชื่อนักเรียน
Score	int	คะแนนสอบ

ข้อมูลนักเรียน (Student)

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Score { get; set; }
}
```



สร้าง Object จากโมเดล

รูปแบบการสร้าง Object

1. คำสั่ง **new** เป็นวิธีการสร้าง Object แบบมาตรฐาน
2. **Type Interence** ตัวแปรภาษา C# จะกำหนดชนิดข้อมูลให้กับตัวแปรอัตโนมัติตามข้อมูลที่จัดเก็บ
3. **ใช้ฟังก์ชัน new()** คือ การกำหนดชนิดข้อมูลให้กับตัวแปรก่อน ข้อมูลที่อยู่ในตัวแปรนั้นๆก็จะมีชนิดข้อมูลเดียวกันอัตโนมัติ

รูปแบบการสร้าง Object

1. คำสั่ง new

```
Student s1 = new Student()
```

2. Type Interence

```
var s1 = new Student()
```

3. ใช้ฟังก์ชัน new()

```
Student s1 = new()
```

SQL Server

SQL Server



เป็นระบบจัดการฐานข้อมูลเชิง
สัมพันธ์ (RDBMS: Relational
Database Management System)
หรือ การเก็บข้อมูลในรูปแบบตาราง
และใช้ภาษา SQL เข้ามาจัดการข้อมูล
ในฐานข้อมูล



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

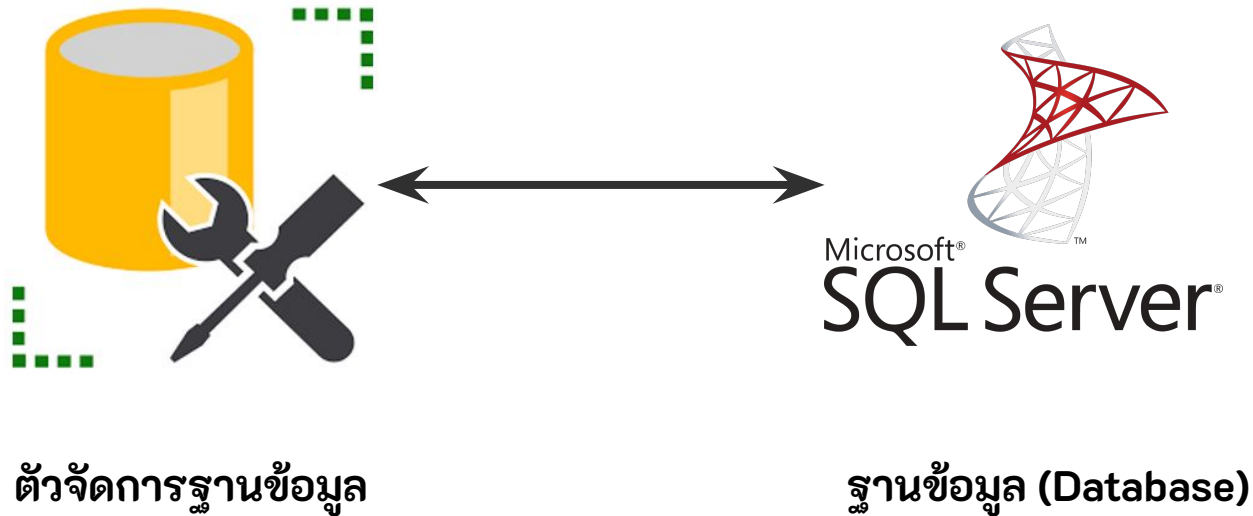
SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS)

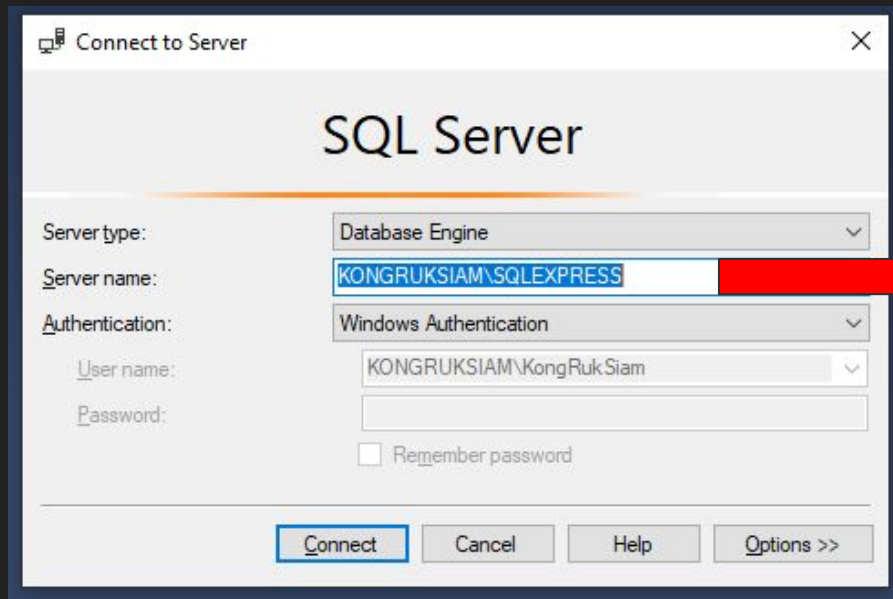


SQL Server Management Studio (SSMS) คือ โปรแกรมสำหรับจัดการฐานข้อมูล สำหรับกำหนดการเข้าถึง , จัดการฐานข้อมูล และ การตั้งค่าฐานข้อมูล เป็นต้น

SQL Server Management Studio (SSMS)



การเชื่อมต่อฐานข้อมูลด้วย SSMS



เชื่อมต่อ SSMS กับ SQL Server

โดยระบุ

ชื่อเครื่อง\ชื่อฐานข้อมูล

ตัวอย่าง

- KONGRUKSIAM\SQLEXPRESS)

Data Annotation



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

Data Annotation

เป็นการกำหนดกฎเกณฑ์หรือรูปแบบการทำงานในระดับโมเดล เช่น

- **Key** คือ กำหนดให้เป็น Primary Key
- **Required** คือ ห้ามเป็นค่าว่าง
- **DisplayName** คือ การกำหนดข้อความกำกับช่อง
- **Range** คือ การระบุช่วงจำนวนของตัวเลข

Connection String



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

Connection String

ชุดข้อความสำหรับระบุตำแหน่งฐานข้อมูลที่จะใช้งานร่วมกับ
โปรเจกต์ ASP.NET Core MVC โดยระบุในไฟล์ appsetting.json

```
"ConnectionStrings": { "DefaultConnection":  
    "Server=ชื่อ Sever;  
    Database=ชื่อฐานข้อมูล;  
    Trusted_Connection=True;  
    TrustServerCertificate=True"  
}
```

Entity Framework Core

Entity Framework Core

Entity Framework Core (EF Core) คือ ไลบรารีสำหรับการจัดการฐานข้อมูล โดยทำงานร่วมกับโมเดล (Model)

EF Core ทำหน้าที่จัดการฐานข้อมูลแทนการใช้คำสั่งSQLโดยตรง กระบวนการทำงานคือ จะทำการ Mapping หรือจับคู่ โมเดลกับตาราง (Table) ที่อยู่ในฐานข้อมูลและให้โมเดลนั้นๆเป็นตัวแทนของตาราง (ORM : Object Relational Mapping)

ติดตั้ง Entity Framework Core

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.SqlServer

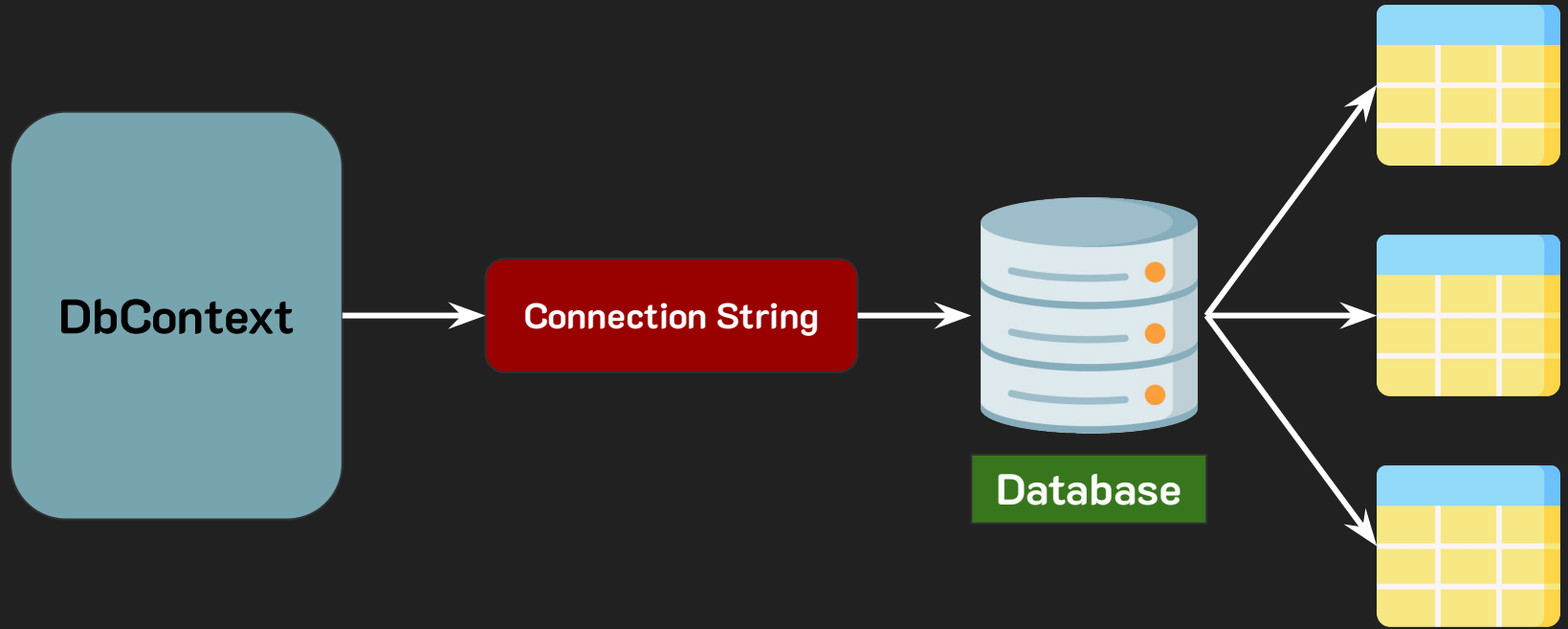
Application DbContext

Application DbContext

คือ การสร้างคลาสที่รับบทบาทเป็นตัวแทนของฐานข้อมูลที่ใช้งานในโปรเจกต์ รวมถึงโมเดลที่จะนำไปสร้างเป็นตารางเก็บในฐานข้อมูลดังกล่าว



Application DbContext



สร้างฐานข้อมูล

ขั้นตอนการสร้างฐานข้อมูล

1. `add-migration <migration-name>`

สร้างไฟล์ migration ที่เก็บโครงสร้างตารางจาก
โมเดล (แปลงโมเดลเป็นตาราง)

2. `update-database`

นำไฟล์ migration ไปใช้งาน



แบบฟอร์มบันทึกข้อมูล



<https://www.youtube.com/c/KongRuksiamOfficial/>



<https://www.facebook.com/KongRuksiamTutorial/>

แบบฟอร์มบันทึกข้อมูล

```
<form method="get|post"  
    asp-controller ="controller"  
    asp-action="method">
```

```
// Element
```

```
</form>
```

แบบฟอร์มบันทึกข้อมูล

method="get หรือ post" (รูปแบบการส่ง)

- **get** ส่งข้อมูลพร้อมแนบข้อมูลไปพร้อมกับ url (ไม่มีความปลอดภัย เพราะข้อมูลถูกมองเห็นและไม่ควรใช้งานร่วมกับข้อมูลที่เป็นแบบ sensitive data)
- **post** ส่งข้อมูลพร้อมซ่อนค่าข้อมูลระหว่างทางที่ส่งไป (มีความปลอดภัย)

Dependency Injection

Dependency Injection

คือ การกำหนดให้คอนโทรลเลอร์สามารถเรียกใช้งานฐานข้อมูลได้
ผ่านตัวแทนของฐานข้อมูลที่ระบุใน DbContext

Dependency Injection

StudentController.cs

```
private readonly ApplicationDbContext _db;
```

```
public StudentController(ApplicationDbContext db){
```

```
    _db = db;
```

```
}
```

บันทึกข้อมูล

บันทึกข้อมูล

คอนโทรลเลอร์ (Controller)

- **HttpPost** ทำหน้าที่รับข้อมูลจากฟอร์มมาใช้งาน
- **ValidateAntiForgeryToken** เพื่อป้องกันการโจมตี (Hack) ผ่านการป้อนข้อมูลในแบบฟอร์มจากผู้ไม่หวังดี เช่น ไม่สามารถแนบหรือรับ Input ที่เป็นรูปแบบ Script ได้ เป็นต้น