ศุภชัย สมพานิช

Thaivb.NET

พัฒนาเว็บแอพด้วย

ASP.NET Core

MVC ด้วยภาษา VC# 2015

```
            app.UseBrowserLink();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
        }

        app.UseStaticFiles();

        app.UseMvc(routes =>
        {
            routes.MapRoute
                name: "default",
                template: "{controller=Home}/{action=Index}/{id?}");
        });
    }
}
}
```

- ศึกษาการใช้งาน EF Core แบบ Code First
- อธิบายละเอียด Step By Step
- มีตัวอย่างประกอบทุกหัวข้อ
- ใช้งานร่วมกับฐานข้อมูล SQL Server 2016

Visual Studio

สอบถามปัญหา : www.facebook.com/thaivb.net

พัฒนาเว็บแอพด้วย

ASP.NET Core MVC ด้วยภาษา VC# 2015

ผู้แต่ง : นายศุภชัย สมพานิช

เนื้อหา : 10 บท (326 หน้า)

วันที่พิมพ์ : สิงหาคม 2559

ครั้งที่พิมพ์ : 1st Edition

จัดจำหน่ายโดย : www.se-ed.com

คุณผู้อ่านสามารถสอบถามปัญหา-ติชมเนื้อหาของหนังสือเล่มนี้ได้ที่

https://www.facebook.com/thaivb.net

คำนำ

ในโลกของ .NET Framework เดิม นักพัฒนาสาย .NET จะถูกจำกัดอยู่เพียงแค่บน Windows เท่านั้น แต่ในปัจจุบันไมโครซอฟท์สร้าง .NET Core ขึ้นมา เพื่อให้นักพัฒนาสาย .NET สามารถพัฒนาแอพครบทั้ง 3 แพลตฟอร์มแล้ว นั่นคือ Windows, macOS และ Linux

แม้ว่าเนื้อหาของหนังสือเล่มนี้ เป็นการพัฒนา Web Apps โดยอาศัย ASP.NET Core MVC บน Windows แต่คุณสามารถนำเนื้อหาและหลักการ ไปใช้ใน macOS และ Linux ได้อีกด้วย อาจจะต้องรอเวลาให้องค์ประกอบต่างๆ บน macOS และ Linux ครบถ้วนเช่นเดียวกับบน Windows

หากคุณผู้อ่านท่านใดติดปัญหาในการเขียนโปรแกรม สามารถสอบถามได้ที่แฟนเพจของผู้เขียนได้ที่ www.facebook.com/thaivb.net และอีก 1 ช่องทาง คือ ช่องใน Youtube ค้นหาช่องที่ชื่อว่า Thaivb.NET

ศุภชัย สมพานิช

สิงหาคม 2559

สารบัญ

บทที่ 1 พัฒนา Web Apps ด้วย ASP.NET Core MVC

บทนำ

ในปัจจุบัน การพัฒนา Web Apps ยังคงมีความสำคัญเป็นอย่างมาก คุณมีทางเลือกมากมาย ที่จะพัฒนา Web Apps ขึ้นมา ไม่ว่าจะเป็นการใช้งานภาษา PHP, MEAN Stack, ASP.NET, ASP.NET Core ฯลฯ เป็นต้น

เนื้อหาที่จะนำเสนอในหนังสือเล่มนี้ เป็นการพัฒนา Web Apps ด้วย ASP.NET Core MVC ด้วยภาษา VC# เป็นอีก 1 ทางเลือกที่นักพัฒนา พลาดไม่ได้เป็นอย่างยิ่ง

Web Apps ในโลกของ .NET

ASP.NET MVC เป็นการพัฒนา Web Apps ด้วย ASP.NET (เนื้อหาของเล่มนี้ เลือกใช้ภาษา VC#) โดยอาศัยหลักการของ MVC เข้ามาช่วยแบ่งแยกหน้าที่การทำงานออกเป็น 3 ส่วน กล่าวคือ

- **Models (M)** หมายถึง ส่วนของการจัดเก็บข้อมูล เช่น ระบบจัดเก็บข้อมูลถาวรในฐานข้อมูล, ข้อมูลชั่วคราวที่อยู่ในคลาส, อาร์เรย์, Generic List ฯลฯ เป็นต้น
- **Views (V)** หมายถึง ส่วนแสดงผลที่ปรากฏในบราวเซอร์ให้ผู้ใช้งานเห็น

บทที่ 2 ทำความรู้จักกับโปรเจ็กต์ ASP.NET Core MVC

บทนำ

เนื้อหาในบทนี้ เป็นบทแรกที่เราจะลงรายละเอียดในการพัฒนา Web Apps ด้วย ASP.NET Core MVC โดยที่ผู้เขียนปูพื้นฐานมาบ้างแล้วในบทที่ ๑ เราจะมาเริ่มทำความรู้จักกับชิ้นส่วนแต่ละชิ้นที่เราได้มาจากโปรเจ็กต์แบบ ASP.NET Core MVC

คลาส Program และคลาส Startup จุดเริ่มต้นของโปรเจ็กต์ ASP.NET Core MVC

เมื่อคุณสร้างโปรเจ็กต์แบบ ASP.NET Core MVC มาแล้ว คุณจะได้โปรเจ็กต์ที่มีโครงสร้างค่อนข้างสมบูรณ์ เราจะมาเริ่มทำความรู้จักกับองค์ประกอบต่าง ๆ เหล่านี้ว่า มีที่มาอย่างไร

จุดเริ่มต้นของโปรเจ็กต์ ASP.NET Core MVC อยู่ที่เมธอด Main() ในคลาส Program (ไฟล์ชื่อว่า Program.cs) พบว่า มีการสั่งให้รันคลาส Startup เป็นลำดับแรกด้วยเมธอด UseStartup() ดังรูปที่ 2-1

บทที่ 3 คอนโทรลเลอร์ (Controller)

บทนำ

คอนโทรลเลอร์ (Controller) เป็นแกนหลัก 1 ใน 3 ตามแนวความคิดของ MVC อยู่ในฐานะเป็นตัวกลาง ทำหน้าที่สั่งการงานในด้านต่างๆ ผ่านทางเมธอด ที่เรียกว่า "Action Methods" ผู้เขียนจะลงรายละเอียดของคอนโทรลเลอร์ว่า มีประเด็นใดบ้างที่น่าสนใจ

การสร้างคอนโทรลเลอร์และส่วนแสดงผลใหม่

คุณสามารถสร้างคอนโทรลเลอร์และส่วนแสดงผลใหม่ตามที่คุณต้องการได้เช่นกัน มีขั้นตอนดังนี้

1. ให้คุณสร้างโปรเจ็กต์แบบ ASP.NET Core MVC ขึ้นมาใหม่
2. ต่อมา ให้คุณโฟกัสที่โฟลเดอร์ Controllers ก่อน จากนั้นคลิกขวา เลือก Add > New Item... เพิ่มไฟล์คลาสประเภท MVC Controller Class เข้ามาในโฟลเดอร์ Controllers ของโปรเจ็กต์เรา ดังรูปที่ 3-1

บทที่ 4 การใช้งานส่วนของข้อมูล (Model & ViewModels)

บทนำ

หลักการทำงานอย่างหนึ่งที่คุณต้องพบเจอเสมอ เมื่อคุณเข้ามาสู่โลกของ ASP.NET Core MVC ก็คือ การทำงานร่วมกับข้อมูล กล่าวคือ

- **การทำงานกับข้อมูลชั่วคราว** เช่น การทำงานกับข้อมูลที่เก็บอยู่ในระบบคลาส (Class) เรามักจะสร้างคลาสขึ้นมาใช้งานเอง เพื่อเก็บข้อมูลตามที่เราต้องการ และยังถือว่า ข้อมูลที่เก็บอยู่ในออบเจ็กต์ที่เกิดมาจากคลาส เป็นข้อมูลชั่วคราว
- **การทำงานกับข้อมูลถาวร** เช่น ระบบฐานข้อมูลประเภทต่างๆ เราถือว่า ข้อมูลในกลุ่มนี้ เป็นข้อมูลถาวร

ข้อมูลทั้ง 2 ประเภท ถูกจัดให้อยู่ในส่วนของ "Model" ไม่ว่าคุณจะทำงานกับข้อมูลประเภทใดก็ตาม มีกลไกหนึ่งที่คุณต้องทำให้เป็น นั่นคือ การนำข้อมูลเหล่านี้ ซึ่งอยู่ในส่วนของ Controller นำไปแสดงผลในส่วนของ View

การส่งข้อมูลจาก Controller ไปสู่ View ด้วยคลาส

การทำงานกับส่วนของ Model ลำดับแรกเลยก็คือ ผู้เขียนจะสร้างคลาสที่ชื่อว่า Products ขึ้นมา ทำหน้าที่เก็บข้อมูลสินค้า และต้องการแสดงข้อมูลสินค้าที่มีอยู่ ในส่วนแสดงผล

Example 4-1 การส่งข้อมูลจาก Controller ไปสู่ View ด้วยคลาส

มีขั้นตอนดังนี้

1. เริ่มต้นสร้างโปรเจ็กต์ที่ชื่อว่า UsingModel ขึ้นมาก่อน

บทที่ 5 การใช้งานส่วนแสดงผล View

บทนำ

"View" เป็นกลไกการทำงานลำดับสุดท้ายของการทำงานในรูปแบบ MVC กล่าวคือ ทำหน้าที่แสดงผล ปรากฏในบราวเซอร์ให้ผู้ใช้งานเห็น อาจจะเป็นข้อมูลที่ถูกดึงมาตั้งแต่ส่วนของ Model หรือเป็นข้อมูลที่ได้มาจากการทำงานของส่วน Controller ก็ได้ เราจะมาดูว่า การทำงานในส่วนของ View มีประเด็นใดบ้างที่น่าสนใจ

การแทรกข้อมูลใน View ด้วยวิธี Inject

โดยปกติแล้ว ข้อมูลที่ถูกส่งต่อมาที่ส่วนของ View เกิดจาก

- **Controller (อยู่ในฐานะเป็นผู้ส่ง)** ส่งข้อมูลโดยอาศัยฟังก์ชัน View()
- **View (อยู่ในฐานะผู้รับ)** โดยใช้คำสั่ง @model รับข้อมูลมาแล้ว นำไปแสดงผลในไฟล์ .cshtml ปรากฏในบราวเซอร์ ซึ่งถือเป็นการทำงานขั้นตอนสุดท้าย

แต่คุณสามารถใช้คำสั่ง @inject ทำหน้าที่ "นำข้อมูลจากส่วนของข้อมูล" แทรกเข้ามาที่ส่วนของ View ได้เลย

Example 5-1 การแทรกข้อมูลใน View ด้วยวิธี Inject

เป้าหมายของตัวอย่างนี้ ก็คือ แสดงข้อมูลลูกค้า (แสดงด้วยขั้นตอนตามปกติ) และข้อมูลอื่นๆ ของลูกค้าที่น่าสนใจ (แสดงด้วยวิธี Inject)

1. ผู้เขียนสร้างโปรเจ็กต์ที่ชื่อว่า UsingInject ขึ้นมาก่อน จากนั้นสร้างหน่วยจัดเก็บข้อมูลลูกค้าขึ้นมา อยู่ในฐานะเป็นข้อมูลชั่วคราวในระดับพื้นฐาน นั่นคือ คลาส Customers เก็บอยู่ในโฟลเดอร์ที่ชื่อว่า Models ข้อมูลลูกค้าที่เราต้องการเก็บ ประกอบด้วย

บทที่ 6 ทำความรู้จักกับระบบบริการ (Service) และ Dependency Injection

บทนำ

เมื่อคุณเข้ามาสู่โลกของ MVC มีการทำงานหลักอย่างหนึ่งที่คุณต้องการทำเสมอ นั่นคือ การนำข้อมูลจากส่วนของ Controller ไปสู่ส่วนแสดงผล View ผ่านทางฟังก์ชัน View()

คำว่า "ข้อมูล" ที่ผู้เขียนกล่าวถึง อาจจะมาจากการทำงานภายในของคอนโทรลเลอร์เอง หรือคอนโทรลเลอร์นำมาจากส่วนของข้อมูล Model ก็ได้ เช่น ข้อมูลชั่วคราวที่ได้จากคลาสต่างๆ ที่เราสร้างขึ้นมาใช้งานเอง หรือข้อมูลที่มาจากระบบฐานข้อมูล ซึ่งถือเป็นระบบจัดเก็บข้อมูลแบบถาวร

เนื้อหาในบทนี้ เรากำลังยกระดับข้อมูลที่เราต้องการ ให้กลายเป็น "การบริการด้านข้อมูล" เรียกว่า "Service" และยังรวมไปถึงการปรับปรุงการรับ-ส่งข้อมูลระหว่างส่วนของ Controller กับ View ให้มีประสิทธิภาพเพิ่มมากยิ่งขึ้นอีกด้วย

พื้นฐานการส่งข้อมูลจาก Controller ไปสู่ส่วนแสดงผล View

วิธีการส่งข้อมูลจากส่วน Controller ไปสู่ส่วนแสดงผล View แบบง่ายที่สุด ก็คือ การส่งข้อความธรรมดา โดยอาศัยคลาส ViewData ที่เราคุ้นเคยกันเป็นอย่างดีนั่นเอง

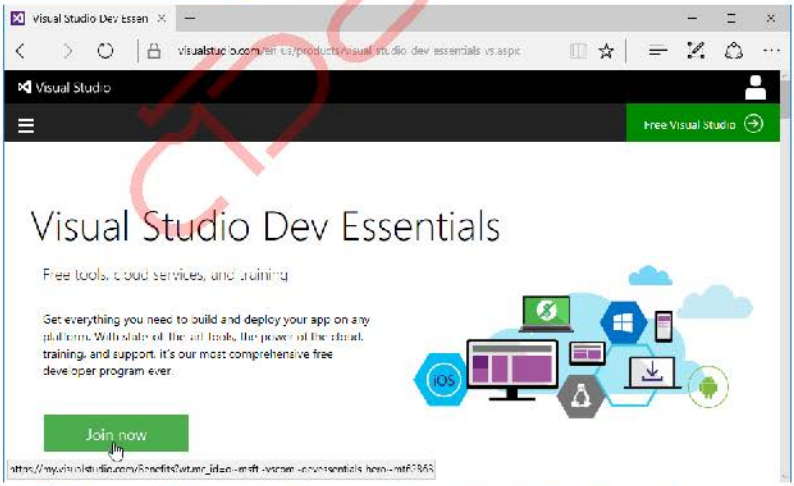
ทุกๆ ครั้งที่คุณสร้างโปรเจ็กต์ ASP.NET Core MVC ขึ้นมา ในคอนโทรลเลอร์ HomeController ที่เมธอด About() หรือเมธอด Contact() จะมีการส่งข้อความที่ชื่อว่า Message โดยอาศัยคลาส ViewData เป็นค่าเริ่มต้น

บทที่ 7 ทำงานกับระบบฐานข้อมูล SQL Server 2016 Developer Edition

บทนำ

ฐานข้อมูล SQL Server 2016 เป็นระบบฐานข้อมูลอีกตัวหนึ่งที่นักพัฒนาในยุคปัจจุบัน จะต้องใช้งานอีก 1 ประเภท ไม่ว่าคุณจะพัฒนาแอพในแพลตฟอร์มใดก็ตาม

ในปัจจุบันไมโครซอฟท์เปิดช่องทางให้นักพัฒนา สามารถใช้งานฐานข้อมูล SQL Server 2016 Developer Edition ได้ฟรี ภายใต้โครงการที่เรียกว่า "Visual Studio Dev Essentials" คุณสามารถสมัครเข้าร่วมโครงการนี้ได้ฟรีที่เว็บไซต์ https://www.visualstudio.com/en-us/products/visual-studio-dev-essentials-vs.aspx



รูปที่ 7-1 แสดงเว็บไซต์ของโปรแกรม Visual Studio Dev Essentials

บทที่ 8 พื้นฐานการทำงานกับระบบฐานข้อมูล SQL Server

บทนำ

การทำงานกับระบบฐานข้อมูล (Database) ถือเป็นหัวข้อหลักอีก 1 เรื่องที่คุณต้องศึกษา เราถือว่า ข้อมูลที่จัดเก็บอยู่ในระบบฐานข้อมูล เป็นการเก็บข้อมูลแบบถาวร สามารถแบ่งฐานข้อมูล SQL Server ได้ 2 รูปแบบ

- ฐานข้อมูล SQL Server ที่มากับ Visual Studio
- ฐานข้อมูล SQL Server ที่เราติดตั้งแยกต่างหาก ในกรณีนี้ คือ ฐานข้อมูล SQL Server 2016 Developer Edtion

วิธีการทำงานกับระบบฐานข้อมูล SQL Server ลำดับแรกที่ผู้เขียนเลือกมานำเสนอในหนังสือเล่มนี้ ก็คือ การทำงานในรูปแบบ "Code First" โดยอาศัย Entity Framework Core เรียกสั้นๆ ว่า "EF Core"

Code First เป็นรูปแบบการทำงานร่วมกับระบบฐานข้อมูล มีจุดเริ่มต้นอยู่ที่การเขียนโค้ดขึ้นมาก่อน โดยอาศัยคลาสที่สร้างขึ้นมา แปลงเป็นตารางในฐานข้อมูล คุณสามารถกำหนดชื่อตาราง, ชนิดข้อมูลแต่ละฟิลด์, เงื่อนไขต่างๆ ฯลฯ ผ่านทางคลาสที่คุณสร้างขึ้นมาทั้งหมด

เห็นได้ว่า รูปแบบนี้แตกต่างไปจากเดิมที่เราต้องสร้างตารางในฐานข้อมูลก่อน แล้วค่อยสร้างโปรเจ็กต์เชื่อมต่อเข้าไป

บทที่ 9 การทำงานกับข้อมูลในฐานข้อมูล SQL Server 2016

บทนำ

เนื้อหาของบทที่แล้ว เป็นการเตรียมความพร้อมให้โปรเจ็กต์ของเราสามารถเชื่อมต่อกับระบบฐานข้อมูลด้วยวิธี Code First ขึ้นอยู่กับว่า คุณจะใช้ฐานข้อมูล SQL Server ใด โดยที่ผู้เขียนขอเลือกใช้ฐานข้อมูล SQL Server 2016 Developer Edition เป็นหลัก โดยใช้ข้อความเชื่อมต่อเก็บอยู่ในไฟล์ appsettings.json ดังต่อไปนี้

appsettings.json

```
{
  "ApplicationInsights": {
    "InstrumentationKey": ""
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "DefaultConnection": {
```

บทที่ 10 การจัดการข้อมูลในฐานข้อมูล SQL Server (CRUD)

บทนำ

เนื้อหาที่ผ่านมา ผู้เขียนนำเสนอวิธีการเรียกดูข้อมูลออกมาจากตารางที่เราสนใจอยู่ เป็นเพียงการแสดงวิธีการใช้งาน ASP.NET Core MVC ร่วมกับระบบฐานข้อมูล SQL Server ด้วยวิธี Code First ในขั้นต้นเท่านั้น

ในทางปฏิบัติแล้ว การทำงานร่วมกับระบบฐานข้อมูล ประกอบด้วยการทำงาน 4 ส่วน คือ

1. **Create (C)** หมายถึง ขั้นตอนการสร้างข้อมูลใหม่ เพื่อนำข้อมูลนั้นๆ ไปเพิ่มลงในฐานข้อมูล
2. **Read (R)** หมายถึง การอ่านข้อมูลออกมาจากระบบฐานข้อมูล
3. **Update (U)** หมายถึง การอัพเดตข้อมูลเดิมที่มีอยู่ ให้กลายเป็นข้อมูลล่าสุด
4. **Delete (D)** หมายถึง การลบข้อมูลออกจากระบบฐานข้อมูล

สำหรับขอบเขตการนำเสนอในบทนี้ นำเสนอการทำงานข้อ 1 ถึงข้อ 3 เท่านั้น เหตุผลก็คือ ในปัจจุบันการลบข้อมูลจริงออกจากระบบฐานข้อมูล จะทำให้ระบบของเราไม่มีหลักฐานหรือข้อมูลอ้างอิงในภายหลัง

การแสดงรายละเอียดข้อมูล

การทำงานลำดับแรกที่จะนำเสนอ ก็คือ การแสดงข้อมูล (Read) ผู้เขียนเลือกใช้รูปแบบที่เราพบเห็นได้โดยทั่วไป นั่นคือ เมื่อผู้ใช้งานคลิกเลือกดูข้อมูลแถวใด ก็จะสั่งให้แสดงรายละเอียดของแถวนั้นๆ ในส่วนแสดงผลตามที่เรากำหนด