| | | |
|---|---|---|
| | **ชุดการสอน**<br>**หน่วยสมรรถนะ : UOC 1**ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ | **หน่วยที่1** |
| | **รหัสวิชาชื่อวิชา**31901-2004 **หน่วยกิต 3** | **สัปดาห์ที่...** |
| | **สมรรถนะย่อย : EOC** เข้าใจหลักการ MVC ใน ASP.NET Core | **ทฤษฎี 4 ชม.**<br>**ปฏิบัติ 0 ชม.** |

**1. ผลลัพธ์การเรียนรู้**

1. เข้าใจและอธิบายหลักการพื้นฐานของ ASP.NET Core MVC ได้อย่างถูกต้อง
2. ออกแบบและพัฒนาเว็บแอปพลิเคชันโดยใช้ ASP.NET Core MVC ได้อย่างมีประสิทธิภาพ
3. สร้างและจัดการโครงสร้าง Model-View-Controller ในโปรเจค ASP.NET Core ได้
4. ใช้งาน Routing, Data Annotation, และ Dependency Injection ใน ASP.NET Core MVC ได้อย่างเหมาะสม
5. พัฒนาเว็บแอปพลิเคชันที่สามารถเชื่อมต่อกับฐานข้อมูลและทำ CRUD operations ได้
6. ประยุกต์ใช้ความรู้ในการพัฒนาเว็บแอปพลิเคชันขนาดเล็กถึงขนาดกลางได้อย่างมีประสิทธิภาพ
7. วิเคราะห์และแก้ไขปัญหาที่อาจเกิดขึ้นในการพัฒนาเว็บด้วย ASP.NET Core MVC ได้
8. ใช้เครื่องมือและเทคโนโลยีที่เกี่ยวข้องกับการพัฒนา ASP.NET Core MVC ได้อย่างเหมาะสม

**2. อ้างอิงมาตรฐานอาชีพ** รหัส ICT-TGWL-008B สาขาวิชาชีพอุตสาหกรรมดิจิทัล สาขาซอฟต์แวร์และการประยุกต์ อาชีพนักพัฒนาระบบ ระดับ 4

**3. หน่วยสมรรถนะ : UOC 1** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์

**หน่วยสมรรถนะย่อย : EOC1.1 EOC** เข้าใจหลักการ MVC ใน ASP.NET Core

**4. เกณฑ์การประเมิน (Performance Criteria : PC)**

1.1 ความรู้: อธิบายหลักการและโครงสร้างของ MVC ใน ASP.NET Core ได้อย่างถูกต้อง

1.2 ทักษะ: สามารถสร้างและจัดการโครงสร้าง MVC ในโปรเจค ASP.NET Core ได้

1.3 เจตคติ: ตระหนักถึงความสำคัญของการใช้ MVC ในการพัฒนาเว็บแอปพลิเคชัน

1.4 ประยุกต์การใช้งาน: สามารถประยุกต์ใช้ MVC ในการพัฒนาเว็บแอปพลิเคชันขนาดเล็กได้

**5. ขอบเขตการปฏิบัติ (Range: R)**

1.1 เครื่องคอมพิวเตอร์ที่มีการติดตั้ง Visual Studio และ .NET Core SDK

1.2 เอกสารประกอบการสอนเกี่ยวกับ MVC ใน ASP.NET Core

1.3 ระบบอินเทอร์เน็ตสำหรับการดาวน์โหลดแพ็คเกจและการค้นคว้าเพิ่มเติม

**6. จุดประสงค์เชิงพฤติกรรม**

6.1 ความรู้:

- อธิบายความหมายและหลักการของ MVC ได้
- ระบุหน้าที่ของแต่ละส่วนใน MVC (Model, View, Controller) ได้
- อธิบายวิธีการทำงานของ Routing ใน ASP.NET Core MVC ได้

6.2 ทักษะ:

- สร้างโปรเจค ASP.NET Core MVC ใหม่ได้
- สร้างและจัดการ Controller, Model, และ View ได้อย่างถูกต้อง
- ใช้ Data Annotation ในการกำหนดกฎการตรวจสอบข้อมูลได้

6.3 เจตคติ:

- เห็นประโยชน์ของการแยกส่วนการทำงานตามหลักการ MVC
- มีความกระตือรือร้นในการเรียนรู้และทดลองใช้ MVC ในการพัฒนาเว็บแอปพลิเคชัน

6.4 ประยุกต์การใช้งาน:

- สามารถพัฒนาเว็บแอปพลิเคชันอย่างง่ายโดยใช้ ASP.NET Core MVC ได้
- สามารถเชื่อมต่อฐานข้อมูลและทำ CRUD operations ผ่าน MVC pattern ได้

**7. เนื้อหา**

**1.** Model-View-Controller คืออะไร

1.1 Model-View-Controller (MVC) เป็นรูปแบบการออกแบบสถาปัตยกรรมซอฟต์แวร์ที่ใช้กันอย่างแพร่หลายในการพัฒนาแอปพลิเคชันเว็บ ASP.NET Core ได้นำรูปแบบนี้มาใช้เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่มีโครงสร้างดี ง่ายต่อการบำรุงรักษา และขยายได้

1.2 องค์ประกอบของ MVC

1.2.1 Model เป็นส่วนที่จัดการข้อมูลและตรรกะทางธุรกิจของแอปพลิเคชันประกอบด้วยคลาสที่แทนข้อมูลและกฎทางธุรกิจไม่ขึ้นกับส่วน View หรือ Controller

1.2.2 View รับผิดชอบในการแสดงผลข้อมูลให้กับผู้ใช้ใน ASP.NET Core MVC, Views มักจะเป็นไฟล์ .cshtml ที่ใช้ Razor syntax

1.2.3 Controller ทำหน้าที่เป็นตัวกลางระหว่าง Model และ View จัดการการร้องขอของผู้ใช้ ประมวลผลข้อมูลจาก Model และส่งข้อมูลไปยัง View ในASP.NET Core, Controllers เป็นคลาสที่สืบทอดมาจาก ControllerBase หรือ Controller

1.3 การทำงานของ MVC ใน ASP.NET Core

ผู้ใช้ส่งคำขอผ่านเบราว์เซอร์Routing ใน ASP.NET Core จะส่งคำขอไปยัง Controller ที่เหมาะสมController ประมวลผลคำขอ โดยอาจมีการเรียกใช้ Model เพื่อดึงหรือจัดการข้อมูลController เลือก View ที่เหมาะสมและส่งข้อมูลไปยัง View View แสดงผลข้อมูลและส่งกลับเป็น HTML ไปยังเบราว์เซอร์ของผู้ใช้

1.4 ข้อดีของการใช้ MVC

- แยกส่วนความรับผิดชอบ (Separation of Concerns)
- ง่ายต่อการทดสอบ (Testability)
- ความยืดหยุ่นในการพัฒนา

2. โครงสร้างพื้นฐานและการทำงาน

SQL Server เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีองค์ประกอบหลายอย่างเพื่อจัดการและควบคุมข้อมูลที่เก็บในฐานข้อมูลได้อย่างมีประสิทธิภาพ โครงสร้างหลักของ SQL Server ประกอบด้วยองค์ประกอบดังนี้

2.1 โครงสร้างของ SQL Server:

2.1.1 Tables: โครงสร้างหลักในการเก็บข้อมูล ตารางเป็นโครงสร้างพื้นฐานในการจัดเก็บข้อมูลใน SQL Server ข้อมูลจะถูกจัดเก็บในรูปแบบของแถว (rows) และคอลัมน์ (columns) โดยแต่ละคอลัมน์จะกำหนดประเภทข้อมูล เช่น int, varchar, datetime เป็นต้น ตารางหนึ่งๆ ในฐานข้อมูลสามารถกำหนดความสัมพันธ์กับตารางอื่นๆ ได้โดยใช้คีย์ต่างๆ เช่น Primary Key และ Foreign Key

2.1.2 Stored Procedures:(โปรแกรมจัดเก็บสำเร็จรูป)Stored Procedures เป็นชุดคำสั่ง SQL ที่ถูกจัดเก็บไว้ใน SQL Server ซึ่งสามารถนำมาใช้ซ้ำได้ โดยมีประโยชน์ในการลดความซับซ้อนของการประมวลผลข้อมูล และเพิ่มประสิทธิภาพในการจัดการกับข้อมูล Stored Procedures สามารถรองรับการใช้เงื่อนไข (conditions), การวนซ้ำ (loops), และตัวแปร (variables) ภายในคำสั่งได้ อีกทั้งยังช่วยปรับปรุงความปลอดภัยของระบบเพราะการเข้าถึงข้อมูลสามารถควบคุมได้โดยไม่ต้องเปิดเผยโครงสร้างข้อมูลที่แท้จริง

2.1.3 Indexes: คือโครงสร้างข้อมูลที่ใช้เพื่อเพิ่มความเร็วในการค้นหาและดึงข้อมูลจากตารางในฐานข้อมูล โดยดัชนีจะถูกสร้างขึ้นจากหนึ่งหรือหลายคอลัมน์ของตาราง การใช้งานดัชนีที่เหมาะสมจะช่วยให้การค้นหาข้อมูลทำได้รวดเร็วและมีประสิทธิภาพมากขึ้น อย่างไรก็ตาม การสร้างดัชนีที่มากเกินไปอาจส่งผลกระทบต่อประสิทธิภาพในการเพิ่ม แก้ไข หรือลบข้อมูล เนื่องจากต้องมีการอัปเดตดัชนีอยู่ตลอดเวลา

2.2 สถาปัตยกรรมของ ASP.NET Core: เป็นแพลตฟอร์มสำหรับการพัฒนาเว็บแอปพลิเคชันที่ถูกออกแบบมาให้ทำงานบนหลายแพลตฟอร์ม (cross-platform) และมีโครงสร้างที่ยืดหยุ่นและทันสมัย สถาปัตยกรรมหลักของ ASP.NET Core มีองค์ประกอบสำคัญที่ช่วยในการพัฒนาแอปพลิเคชันที่มีความสามารถในการปรับตัวได้ดี, ง่ายต่อการทดสอบ, และมีประสิทธิภาพสูง ดังนี้

2.2.1 MVC (Model-View-Controller) เป็นสถาปัตยกรรมแบบแยกส่วนที่ช่วยจัดการโครงสร้างของแอปพลิเคชัน โดยแบ่งออกเป็นสามส่วนหลัก:

- Model: ส่วนของแอปพลิเคชันที่จัดการข้อมูลและธุรกิจลอจิก (business logic) โดย Model จะเป็นตัวแทนของข้อมูลที่ได้รับจากฐานข้อมูลและใช้ในการประมวลผลต่างๆ เช่น การคำนวณหรือการตรวจสอบ
- View: ส่วนที่แสดงผลข้อมูลให้กับผู้ใช้งาน เป็นส่วนที่รับผิดชอบในการจัดการการแสดงข้อมูลที่ได้รับจาก Model ไปสู่ผู้ใช้งาน ผ่านการสร้างหน้าเว็บ (HTML) หรืออินเทอร์เฟซอื่นๆ
- Controller: ตัวกลางระหว่าง View และ Model มีหน้าที่รับคำขอ (request) จากผู้ใช้ผ่าน View และทำการประมวลผลโดยการเรียกใช้ Model เพื่อดึงข้อมูล จากนั้นส่งข้อมูลที่ได้ไปยัง View เพื่อแสดงผล

2.2.2 Middleware คือส่วนประกอบที่ถูกใช้เพื่อจัดการคำขอ (request) และคำตอบ (response) ในแอปพลิเคชัน ASP.NET Core เมื่อผู้ใช้ส่งคำขอเข้ามาในแอปพลิเคชัน คำขอนั้นจะผ่าน Middleware หลายตัว ซึ่งจะทำงานเป็นขั้นตอนก่อนที่คำขอนั้นจะไปถึงส่วนที่ประมวลผลจริงๆ เช่น Controller Middleware แต่ละตัวสามารถ:

- จัดการคำขอ (เช่น การตรวจสอบสิทธิ์)
- เปลี่ยนแปลงคำขอหรือคำตอบ
- สร้างคำตอบเองหรือส่งคำขอไปยัง Middleware ตัวถัดไป

2.2.3 Dependency Injection (DI) เป็นแนวคิดที่ช่วยในการจัดการ dependencies ของแอปพลิเคชัน โดยที่ ASP.NET Core นำ DI มาใช้เป็นส่วนหนึ่งของสถาปัตยกรรมหลัก ซึ่งช่วยในการควบคุมการสร้างและการใช้งานของ objects ที่เป็น dependencies ในแอปพลิเคชัน เช่น บริการ (services) หรือ classes อื่นๆ ที่แอปพลิเคชันต้องการใช้งาน การใช้ DI ช่วยให้โค้ดมีความยืดหยุ่นและทดสอบง่ายขึ้น โดยการแยกการสร้างและการใช้งาน objects ออกจากกัน ตัวอย่างของการใช้ Dependency Injection ได้แก่:

- Service Registration: แอปพลิเคชันจะทำการลงทะเบียนบริการต่างๆ ใน Startup.cs เช่น การสร้าง service ในรูปแบบของ Singleton, Scoped, หรือ Transient
- Service Injection: ASP.NET Core จะทำการ inject service ที่ลงทะเบียนไว้เข้าไปยัง Controller, Middleware, หรือส่วนประกอบอื่นๆ ของแอปพลิเคชันที่ต้องการใช้งาน

3. การเชื่อมต่อและการดึงข้อมูล

ในการพัฒนาแอปพลิเคชันด้วย ASP.NET Core การเชื่อมต่อกับฐานข้อมูลเป็นขั้นตอนสำคัญที่ช่วยให้สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ โดยการเชื่อมต่อและดึงข้อมูลใน ASP.NET Core สามารถทำได้ผ่านหลายวิธี หนึ่งในวิธีหลักคือการใช้ Connection String และ Entity Framework Core ซึ่งเป็นเครื่องมือที่รองรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ผ่านการแมปข้อมูลในรูปแบบของวัตถุ (ORM)

3.1 Connection String คือข้อมูลที่ใช้ระบุรายละเอียดการเชื่อมต่อกับฐานข้อมูล เช่น ชื่อเซิร์ฟเวอร์, ชื่อฐานข้อมูล, และข้อมูลรับรอง (credentials) ของผู้ใช้ โดยทั่วไปแล้ว Connection String จะถูกเก็บไว้ในไฟล์การตั้งค่าของแอปพลิเคชัน เช่น appsettings.json

3.2 Entity Framework Core เป็นเฟรมเวิร์กแบบ ORM ที่ได้รับความนิยมใน .NET สำหรับการทำงานกับฐานข้อมูลเชิงสัมพันธ์ เช่น SQL Server, PostgreSQL, และ MySQL EF Core ช่วยให้ผู้พัฒนาสามารถทำงานกับข้อมูลในฐานข้อมูลได้ในลักษณะของวัตถุ (Object-Oriented) โดยไม่ต้องเขียนคำสั่ง SQL โดยตรง EF Core ทำหน้าที่เป็นตัวกลางที่แปลงคำสั่งในโค้ดที่เป็นวัตถุ (C#) ให้กลายเป็นคำสั่ง SQL ที่ทำงานกับฐานข้อมูลได้ โดยผู้พัฒนาสามารถทำงานกับข้อมูลในลักษณะของ "Entities" ซึ่งแต่ละ Entity จะเป็นตัวแทนของตารางในฐานข้อมูล

3.3 ORM (Object-Relational Mapping) คือเทคนิคในการแมป (mapping) ข้อมูลระหว่างฐานข้อมูลเชิงสัมพันธ์ (Relational Database) และวัตถุในภาษาการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) การใช้ ORM ช่วยให้ผู้พัฒนาไม่จำเป็นต้องเขียนคำสั่ง SQL โดยตรง และสามารถจัดการข้อมูลในฐานข้อมูลในลักษณะของวัตถุได้อย่างสะดวกใน .NET หนึ่งใน ORM ที่ได้รับความนิยมมากที่สุดคือ Entity Framework Core ซึ่งทำให้การเชื่อมต่อฐานข้อมูลและการจัดการข้อมูลสามารถทำได้อย่างยืดหยุ่นและสะดวก ผู้พัฒนาสามารถสร้างโครงสร้างฐานข้อมูลจากโค้ด (Code-First) หรือสร้างโครงสร้างของวัตถุจากฐานข้อมูลที่มีอยู่แล้ว (Database-First)

4. Tag Helpers

Tag Helpers ใน ASP.NET Core MVC เป็นคุณลักษณะที่ใช้เพื่อเพิ่มความสามารถในการทำงานร่วมกันระหว่างโค้ด C# และ HTML โดยทำให้โค้ดที่เขียนใน C# สามารถสร้างหรือจัดการ HTML elements ได้ง่ายขึ้น ซึ่งจะช่วยให้การพัฒนาเว็บแอปพลิเคชันสะดวกและเป็นระเบียบมากขึ้น

4.1 คุณสมบัติเด่นของ Tag Helpers

- HTML-like syntax: ใช้รูปแบบการเขียนที่คล้ายกับ HTML ซึ่งทำให้นักพัฒนาสามารถใช้งานได้ง่ายขึ้น ไม่ต้องเขียน C# เพื่อสร้าง HTML tags ด้วยตนเอง
- Integration with Model: สามารถเชื่อมโยงข้อมูลจากโมเดลไปยังฟอร์ม HTML ได้ง่าย เช่น การสร้าง input fields โดยใช้ข้อมูลจากโมเดล
- Built-in and custom Tag Helpers: มี Tag Helpers ที่มาพร้อมกับ ASP.NET Core เช่น asp-for, asp-action, asp-controller และยังสามารถสร้าง Tag Helpers เองได้เพื่อใช้งานตามความต้องการเฉพาะ

4.2 ตัวอย่างการใช้ Tag Helpers

- **Form Tag Helper**
  ช่วยสร้างฟอร์มที่ส่งข้อมูลไปยัง Action ใน Controller

```
<form asp-action "Create" asp-controller "Products">
    <input asp-for "ProductName" />
    <button type "submit">Submit</button>
</form>
```

ตัวอย่างนี้จะสร้างฟอร์มที่ส่งข้อมูลไปยัง Action ชื่อ Create ของ Controller ชื่อ Products และใช้ asp-for เพื่อเชื่อมข้อมูลในฟิลด์กับโมเดล ProductName

- Anchor Tag Helper

  ช่วยสร้างลิงก์ที่เชื่อมโยงไปยัง Action หรือ Controller

```
<a asp-action "Edit" asp-controller "Products" asp-route-id "5">Edit Product</a>
```

  ตัวอย่างนี้จะสร้างลิงก์ไปยัง Action Edit ของ Controller Products โดยส่งค่า id เป็น 5

- Custom Tag Helper

  สามารถสร้าง Tag Helper เองได้โดยสืบทอดจาก TagHelper class

```
public class EmailTagHelper : TagHelper
{
    public string Email { get; set; }
    public override void Process(TagHelperContext context, TagHelperOutput output)
    {
        output.TagName = "a";     // เปลี่ยน tag ให้เป็น <a>
        output.Attributes.SetAttribute("href", $"mailto:{Email}");
        output.Content.SetContent(Email);
    }
}
```

4.3 ข้อดีของการใช้ Tag Helpers

- ช่วยให้โค้ดสะอาดและอ่านง่าย: ไม่ต้องแยกโค้ด HTML และโค้ด C# ออกจากกัน
- สนับสนุนการทำงานของ IntelliSense: ช่วยให้การเขียนโค้ดรวดเร็วขึ้นเพราะมีการแนะนำโค้ด
- รองรับการใช้งานร่วมกับ Razor Views: ทำให้การสร้าง View มีความยืดหยุ่นและสะดวกมากขึ้น

**8. คู่มือ / Data**

1. คู่มือการใช้งาน Visual Studio สำหรับการพัฒนา ASP.NET Core MVC
2. เอกสารอธิบายโครงสร้างและการทำงานของ MVC ใน ASP.NET Core
3. ตัวอย่างโค้ดสำหรับการสร้าง Model, View, และ Controller พื้นฐาน

**9. เอกสารอ้างอิง**

1. Microsoft. (2024). Introduction to ASP.NET Core MVC. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview
2. Freeman, A. (2023). Pro ASP.NET Core MVC. Apress.
3. Smith, S. (2024). ASP.NET Core in Action. Manning Publications.
4. Sanderson, S. (2023). ASP.NET Core Application Development: Building an Application in Four Sprints. Microsoft Press.
5. ASP.NET Core MVC Tutorial. (2024). https://www.tutorialsteacher.com/core/aspnet-core-introduction10. ใบงาน/ใบมอบหมายงาน