

**วิทยาลัยเทคโนโลยีภาคตะวันออก (อีเทค)**

**แผนการสอน / แผนการจัดการเรียนรู้โมดูลฐานสมรรถนะอาชีพ**

**หลักสูตรประกาศนียบัตรวิชาชีพชั้นสูง พุทธศักราช 2567**

ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ

กลุ่มอาชีพซอฟต์แวร์และการประยุกต์

สาขาวิชา เทคโนโลยีสารสนเทศ

รหัส 31901-2004 การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End

**จัดทำโดย**

**นายเอกรัตน์ อุไรโรจน์**

**คำนำ**

**รายวิชา "การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End" เป็นวิชาที่มุ่งเน้นการพัฒนาทักษะและความรู้ที่จำเป็นสำหรับการสร้างซอฟต์แวร์ด้านหน้า (Front-End) ที่ทันสมัยและมีประสิทธิภาพ ในยุคดิจิทัลที่เทคโนโลยีมีการเปลี่ยนแปลงอย่างรวดเร็ว การเข้าใจและสามารถประยุกต์ใช้เทคโนโลยี Front-End ได้อย่างชำนาญนั้นเป็นทักษะที่มีความต้องการสูงในอุตสาหกรรมซอฟต์แวร์**

**แผนการจัดการเรียนรู้นี้ครอบคลุมทั้งภาคทฤษฎีและปฏิบัติ โดยเริ่มตั้งแต่หลักการพื้นฐานไปจนถึงเทคนิคขั้นสูงในการพัฒนา Front-End ผู้เรียนจะได้เรียนรู้การใช้ ASP.NET Core MVC ร่วมกับเทคโนโลยี Front-End อื่นๆ เช่น JavaScript frameworks (Vue, React, Angular) และ CSS frameworks ต่างๆ อีกทั้งได้ฝึกฝนการทำงานเป็นทีม การพัฒนาแบบบูรณาการ และการเชื่อมต่อกับ RESTful API และ Web Services**

**นอกจากนี้ หลักสูตรยังครอบคลุมถึงการจัดการฐานข้อมูล การรักษาความปลอดภัย การแก้ไขข้อผิดพลาด และการทดสอบซอฟต์แวร์ ซึ่งเป็นทักษะที่สำคัญสำหรับนักพัฒนา Front-End มืออาชีเป้าหมายของแผนการจัดการเรียนรู้นี้คือการเตรียมความพร้อมให้ผู้เรียนสามารถก้าวเข้าสู่การเป็นนักพัฒนา Front-End ที่มีความสามารถ มีความเข้าใจในหลักการและแนวปฏิบัติที่ดีในการพัฒนาซอฟต์แวร์ และพร้อมที่จะเรียนรู้เทคโนโลยีใหม่ๆ ที่จะเกิดขึ้นในอนาคต**

**นายเอกรัตน์ อุไรโรจน์**

**สารบัญ**

**หน้า**

คำนำ 1

สารบัญ 2

ชื่อวิชา 3

ผังการกำหนดหน่วยสมรรถนะ 4

ตารางรายการสอน 5

หน่วยสมรรถนะย่อย 6

**รหัสโมดูล** 31901-2004 **ชื่อโมดูล** การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End CODE 3021

**เวลาเรียน** 4 **คาบ/สัปดาห์** **จำนวน** 4 **หน่วยกิต**

**ระดับชั้น** ปวส. **ปีที่** 1 **สาขา** เทคโนโลยีสารสนเทศ

**จุดประสงค์รายวิชา**

1. เข้าใจการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
2. มีทักษะในการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
3. มีเจตคติและกิจนิสัยที่ดีในการปฏิบัติงานด้วยความละเอียดรอบคอบ รับผิดชอบ การสื่อสาร

การคิดเชิงนวัตกรรมและการทำงานเป็นทีม

1. มีความสามารถประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ

**สมรรถนะรายวิชา**

1. ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ
2. พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ
3. ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ
4. จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End
5. ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ

**คำอธิบายรายวิชา**

ศึกษาและปฏิบัติเกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ความหมาย ความสำคัญ ประโยชน์ของการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End เลือกใช้ภาษาและเทคโนโลยีสมัยใหม่ในการพัฒนา (Vue/React/

Angular/ฯลฯ) การทำงานร่วนกันเป็นทีม (Version control System) ดำเนินการพัฒนาโปรแกรมแบบ Integration ตกแต่งเอกสารด้วย UI Component Library/CSS Framework เขียนโปรแกรมเชื่อมต่อ RESTful API/Web Services เขียนโปรแกรมติดต่อฐานข้อมูลแบบ SQL/NoSQL การทำ Authentication ในรูปแบบ JSON Web Token/Session หาจุดผิดพลาด ตามบันทึกข้อผิดพลาดแก้ไขข้อผิดพลาดของโปรแกรม ทดสอบการแก้ไขข้อผิดพลาดของโปรแกรมอ่าน Functional/Program Specification/UML เขียนโปรแกรมตาม Functional/Program Specification/ UML ออกแบบการทดสอบ Integration Test ดำเนินการทดสอบโปรแกรมแบบ Integration Test จัดทำรายงาน ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น จัดทำคู่มือการใช้งานโปรแกรม ตรวจสอบความถูกต้องของคู่มือการใช้งานโปรแกรมการส่งมอบซอฟต์แวร์เพื่อให้สามารถใช้งานได้

แผนผังกำหนดหน่วยโมดูลฐานสมรรถนะ

ตารางรายการสอน

|  |  |  |
| --- | --- | --- |
| รายการ | จำนวนชั่วโมง | สัปดาห์ที่ |
| **สมรรถนะที่ 1** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ | | |
| สรรถนะย่อยที่ 1.1 เข้าใจหลักการ MVC ใน ASP.NET Core | 2 | 1 |
| สรรถนะย่อยที่ 1.2 เรียนรู้การใช้งาน SQL Server กับ ASP.NET Core | 1 | 1 |
| สรรถนะย่อยที่ 1.3 เข้าใจระบบการยืนยันตัวตนและการจัดการสิทธิ์ | 1 | 1 |
| **สมรรถนะที่ 2** พัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการและกระบวนการ | | |
| 2.1 สร้างโครงสร้างโปรเจค ASP.NET Core MVC | 4 | 2 |
| 2.2 ออกแบบและสร้างฐานข้อมูล SQL Server | 4 | 3 |
| 2.3 พัฒนาระบบเข้าสู่ระบบและจัดการสิทธิ์ | 3 | 4 |
| 2.4 พัฒนาส่วนแสดงผลและค้นหาหนังสือ | 8 | 5-8 |
| 2.5 พัฒนาระบบจัดการหลังบ้าน CRUD | 8 | 9-12 |
| **สมรรถนะที่ 3**.ทดสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์ตามหลักการและกระบวนการ | | |
| 3.1 ทดสอบการทำงานของระบบเข้าสู่ระบบ | 1 | 4 |
| 3.2 ทดสอบการค้นหาและแสดงผล | 4 | 5-8 |
| 3.3 ทดสอบระบบจัดการหลังบ้าน | 4 | 9-12 |
| **สมรรถนะที่ 4.**จัดทำคู่มือการใช้งานตามหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End | | |
| 4.1 จัดทำคู่มือการใช้งานสำหรับผู้ใช้ทั่วไป | 4 | 5-8 |
| 4.2 จัดทำคู่มือการใช้งานสำหรับผู้ดูแลระบบ | 4 | 9-12 |
| **สมรรถนะที่ 5.**ประยุกต์ใช้หลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ในงานอาชีพ | | |
| 5.1 ออกแบบและพัฒนาโครงงานในรายวิชา | 8 | 13-14 |
| 5.2 นำเสนอโครงงานในรายวิชา | 4 | 15 |
| รวม | 60 | 15 |

หน่วยสมรรถนะที่ 1 ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ที่** | **สมรรถนะย่อย  (EOC)** | **ขอบเขตการปฏิบัติงาน  (R)** | **เกณฑ์การประเมิน (PC)** | **เครื่องมืออุปกรณ์** | **หลักฐาน (EV)** | |
| **1.3** | **ศึกษาการใช้งานโปรแกรมที่พัฒนาขึ้น** | 1. ความถูกต้องของการทำงาน  2. การออกแบบโครงสร้าง MVC  3. การออกแบบ UI/UX (10%) | 1. ระบุการทำงานของโปรแกรม  2. ระบุการใช้งานโปรแกรม  3. ระบุข้อจำกัดการใช้งานโปรแกรม | - เครื่องคอมพิวเตอร์  - visual studio code | - แบบบรรจุภัณฑ์ / ตัวละครเกม | ประเมินผลการ  ทดสอบภาคปฏิบัติ  (ใบงาน) ผ่าน  เกณฑ์ร้อยละ 80 |
| **ขั้นตอนการประเมิน (AG)** | |
| จัดห้องทดสอบภาคปฏิบัติโดยการ  เตรียมใบงานหรือสถานการณ์จำลอง | |

**ชุดการสอนตามหน่วยสมรรถนะ**

**รหัสวิชา** 31901-2004 **ชื่อวิชา** การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End **หน่วยกิต** 4 รหัสหน่วย

**หน่วยสมรรถนะ** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ

**หน่วยสมรรถนะย่อย** เข้าใจหลักการ MVC ใน ASP.NET Core

**เกณฑ์การประเมิน**

1. แบบทดสอบความเข้าใจเกี่ยวกับหลักการ MVC
2. โปรเจคจำลองโดยใช้ ASP.NET Core MVC
3. การนำเสนอและอธิบายโครงสร้าง MVC ที่สร้างขึ้น

**บทนำ**

Model-View-Controller คืออะไร

1. Model-View-Controller (MVC) เป็นรูปแบบการออกแบบสถาปัตยกรรมซอฟต์แวร์ที่ใช้กันอย่างแพร่หลาย

ในการพัฒนาแอปพลิเคชันเว็บ ASP.NET Core ได้นำรูปแบบนี้มาใช้เพื่อช่วยให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่มีโครงสร้างดี ง่ายต่อการบำรุงรักษา และขยายได้

1. องค์ประกอบของ MVC
   1. Model เป็นส่วนที่จัดการข้อมูลและตรรกะทางธุรกิจของแอปพลิเคชันประกอบด้วยคลาสที่แทนข้อมูลและกฎทางธุรกิจไม่ขึ้นกับส่วน View หรือ Controller
   2. View รับผิดชอบในการแสดงผลข้อมูลให้กับผู้ใช้ใน ASP.NET Core MVC, Views มักจะเป็นไฟล์ .cshtml ที่ใช้ Razor syntax
   3. Controller ทำหน้าที่เป็นตัวกลางระหว่าง Model และ View จัดการการร้องขอของผู้ใช้ ประมวลผลข้อมูลจาก Model และส่งข้อมูลไปยัง View ในASP.NET Core, Controllers เป็นคลาสที่สืบทอดมาจาก ControllerBase หรือ Controller
2. 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. ข้อดีของการใช้ MVC

* แยกส่วนความรับผิดชอบ (Separation of Concerns)
* ง่ายต่อการทดสอบ (Testability)
* ความยืดหยุ่นในการพัฒนา
* สนับสนุนการพัฒนาแบบขนาน (Parallel Development)

**ชุดการสอนตามหน่วยสมรรถนะ**

**รหัสวิชา** 31901-2004 **ชื่อวิชา** การพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End **หน่วยกิต** 4 รหัสหน่วย

**หน่วยสมรรถนะ** ประมวลความรู้เกี่ยวกับหลักการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยี Front-End ตามหลักการ

**หน่วยสมรรถนะย่อย** เรียนรู้การใช้งาน SQL Server กับ ASP.NET Core

**เกณฑ์การประเมิน**

1. แบบทดสอบความเข้าใจเกี่ยวกับหลักการ MVC
2. โปรเจคจำลองโดยใช้ ASP.NET Core MVC
3. การนำเสนอและอธิบายโครงสร้าง MVC ที่สร้างขึ้น

**บทนำ**

1. โครงสร้างพื้นฐานและการทำงาน

SQL Server เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีองค์ประกอบหลายอย่างเพื่อจัดการและควบคุมข้อมูลที่เก็บในฐานข้อมูลได้อย่างมีประสิทธิภาพ โครงสร้างหลักของ SQL Server ประกอบด้วยองค์ประกอบดังนี้

* 1. โครงสร้างของ SQL Server:
     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. Stored Procedures:(โปรแกรมจัดเก็บสำเร็จรูป)Stored Procedures เป็นชุดคำสั่ง SQL ที่ถูกจัดเก็บไว้ใน SQL Server ซึ่งสามารถนำมาใช้ซ้ำได้ โดยมีประโยชน์ในการลดความซับซ้อนของการประมวลผลข้อมูล และเพิ่มประสิทธิภาพในการจัดการกับข้อมูล Stored Procedures สามารถรองรับการใช้เงื่อนไข (conditions), การวนซ้ำ (loops), และตัวแปร (variables) ภายในคำสั่งได้ อีกทั้งยังช่วยปรับปรุงความปลอดภัยของระบบเพราะการเข้าถึงข้อมูลสามารถควบคุมได้โดยไม่ต้องเปิดเผยโครงสร้างข้อมูลที่แท้จริง
     3. Indexes: คือโครงสร้างข้อมูลที่ใช้เพื่อเพิ่มความเร็วในการค้นหาและดึงข้อมูลจากตารางในฐานข้อมูล โดยดัชนีจะถูกสร้างขึ้นจากหนึ่งหรือหลายคอลัมน์ของตาราง การใช้งานดัชนีที่เหมาะสมจะช่วยให้การค้นหาข้อมูลทำได้รวดเร็วและมีประสิทธิภาพมากขึ้น อย่างไรก็ตาม การสร้างดัชนีที่มากเกินไปอาจส่งผลกระทบต่อประสิทธิภาพในการเพิ่ม แก้ไข หรือลบข้อมูล เนื่องจากต้องมีการอัปเดตดัชนีอยู่ตลอดเวลา
  2. สถาปัตยกรรมของ ASP.NET Core: เป็นแพลตฟอร์มสำหรับการพัฒนาเว็บแอปพลิเคชันที่ถูกออกแบบมาให้ทำงานบนหลายแพลตฟอร์ม (cross-platform) และมีโครงสร้างที่ยืดหยุ่นและทันสมัย สถาปัตยกรรมหลักของ ASP.NET Core มีองค์ประกอบสำคัญที่ช่วยในการพัฒนาแอปพลิเคชันที่มีความสามารถในการปรับตัวได้ดี, ง่ายต่อการทดสอบ, และมีประสิทธิภาพสูง ดังนี้
     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
  + 1. 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
  + 1. 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 เช่นการสร้าง service ในรูปแบบของ Singleton, Scoped, หรือ Transient
* Service Injection: ASP.NET Core จะทำการ inject service ที่ลงทะเบียนไว้เข้าไปยัง Controller, Middleware, หรือส่วนประกอบอื่นๆ ของแอปพลิเคชันที่ต้องการใช้งาน

1. การเชื่อมต่อและการดึงข้อมูล

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

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