1.在範本窗格中,選取已安裝的範本展開 Visual C# 節點。 底下 Visual C#,

選取 Web。 在專案範本清單中,**[ASP.NET Core Web 應用程式]**。 將專案命名

為"**dotNetCore\_DatabaseFirst**” ,然後按一下確定。

2.在 [建立新的 ASP.NET Core Web 應用程式] 對話方塊中,選取**[Web Application Model-View\_Controller]**範本。然後按一下 [建立] 。

3.在 [方案總管] 中,以滑鼠右鍵按一下 **dotNetCore\_DatabaseFirst** 專案資料夾。 從操作功

能表中,按右鍵選取[**管理 NuGet 套件**] ,依序加入

「Microsoft.EntityFrameworkCore.SqlServer」、

「Microsoft.EntityFrameworkCore.Tools」這兩個套件,按[**安裝**]。

\*注意,兩者版本需一致,不然會發生錯誤。

4.在選單點選 **[工具]->[ NuGet 套件管理員]->[套件管理員主控台]**

**Default Project->dotNetCore\_DatabaseFirst** 專案

**dotNetCore\_DatabaseFirst** 專案->設為起始專案

5. 在主控台 Console要下的指令↓

Scaffold-DbContext "Server=.\sqlexpress;Database=Northwind;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Force -UseDatabaseNames -Tables Customers,Orders,"Order Details",Products

Scaffold-DbContext 為剛剛安裝 Tools 套件的指令,DB 連線字串依自己情況決

定

-OutputDir 指定要輸出到專案根目錄下哪個資料夾,本文為 Models 資料夾

-Force 為工具產出的.cs 檔要強制覆寫現存檔案(DB 欄位 Schema 異動後,就給

這個)

-Tables 參數,指定只需要載入哪些資料表(以逗號區隔),留意大小寫最好和 DB

裡一樣避免出現黃字警告,沒給-Tables 參數的話,預設抓 DB 全部的 Table

-UseDatabaseNames,程式碼產生出來的類別名稱要和資料庫裡的一模一樣

6. 在上一步驟會建立的 [Models] 資料夾, 並產生5個檔案 Customers.cs、Orders.cs、Order\_Details.cs、Products.cs、及 NorthwindContext.cs。

7. 接著到 NorthwindContext.cs 裡,註解 OnConfiguring 方法,微軟建議改

使用相依性注入(Dependency Injection)方式設定 DB 連線字串。

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

// if (!optionsBuilder.IsConfigured)

// {

//#warning To protect potentially sensitive information in your connection string, you should move it out of source code. See http://go.microsoft.com/fwlink/?LinkId=723263 for guidance on storing connection strings.

// optionsBuilder.UseSqlServer("Server=.\\sqlexpress;Database=Northwind;Trusted\_Connection=True;");

// }

}

8.在 Appsettings.json 加入 DB 連線字串。

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"DefaultConnection": "Server=.\\sqlexpress;Database=Northwind;Trusted\_Connection=True;"

},

"AllowedHosts": "\*"

}

9. 在 [方案總管] 中,修改**dotNetCore\_DatabaseFirst** 專案資料夾的 Startup.cs 的

ConfigureServices 方法。

public void ConfigureServices(IServiceCollection services)

{

//Add EntityFramework support for SqlServer.

services.AddEntityFrameworkSqlServer();

//Add ApplicationDbContext with Connection options

//using dotNetCore\_DatabaseFirst.Models;

//using Microsoft.EntityFrameworkCore;

services.AddDbContext**<NorthwindContext>**(options =>

options**.UseSqlServer**(Configuration.GetConnectionString(**"DefaultConnection"**))

);

services.AddControllersWithViews();

}

10. 在 [方案總管] 中,以滑鼠右鍵按一下**dotNetCore\_DatabaseFirst** 專案資料夾。 按右鍵選

擇[建置](Build)然後選擇 [建置方案](Bulid Solution)

注意:此步驟需建置完成才能建立 Controller

11. 在方案總管 中,以滑鼠右鍵按一下 在上一步驟建立的 [Controllers] 資料

夾上按一下滑鼠右鍵,然後選取功能表選項選取 新增,然後選取 Controller

12. 以滑鼠右鍵按一下 Controllers 資料夾。

• 選取 [新增] > [新增 Scaffold 項目]。

• 選取 **[MVC Controller With Views, Using Entity Framework]**,然後選取 [新

增]。



13. 在 [**MVC Controller With Views, Using Entity Framework**] 對話方塊中:

⚫ 模型類別:**[Customers (dotNetCore\_DatabaseFirst.Models) ]**。

⚫ 資料內容類別: **[ NorthwindContext(dotNetCore\_DatabaseFirst.Models) ]**。

⚫ 控制器名稱:**[CustomersController]**

按一下 [新增]。

14.Scaffolding 會建立名為**CustomersController.cs** 在 Controllers 資料夾中的

檔案。內含自動產生的 Index, Details, Create, Edit, Delete 等5個CRUD Actions

public class CustomersController : Controller

{

private readonly **NorthwindContext** \_context;

public CustomersController(NorthwindContext context)

{

**\_context = context**;

}

// GET: Customers

public **async** **Task<IActionResult>** Index()

{

return View(**await** \_context.Customers.ToList**Async**());

}

// GET: Customers/Details/5

public **async Task<IActionResult>** Details(string **id**)

{

if (id == null)

{

return NotFound();

}

var **customers** = **await** \_context.Customers

.FirstOrDefaultAsync(m => m.CustomerID == id);

if (customers == null)

{

return NotFound();

}

return View(**customers**);

}

// GET: Customers/Create

public IActionResult Create()

{

return View();

}

// POST: Customers/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

**[HttpPost]**

[ValidateAntiForgeryToken]

public **async Task<IActionResult>** Create([Bind("CustomerID,CompanyName,ContactName,ContactTitle,Address,City,Region,PostalCode,Country,Phone,Fax")] Customers **customers**)

{

if (ModelState.IsValid)

{

\_context.Add(customers);

**await** \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(**customers**);

}

// GET: Customers/Edit/5

public **async Task<IActionResult>** Edit(string id)

{

if (id == null)

{

return NotFound();

}

var **customers** = **await** \_context.Customers.FindAsync(id);

if (customers == null)

{

return NotFound();

}

return View(**customers**);

}

// POST: Customers/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

**[HttpPost]**

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(string id, [Bind("CustomerID,CompanyName,ContactName,ContactTitle,Address,City,Region,PostalCode,Country,Phone,Fax")] Customers **customers**)

{

if (id != customers.CustomerID)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.**Update**(customers);

**await** \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CustomersExists(customers.CustomerID))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(**customers**);

}

// GET: Customers/Delete/5

public async **Task<IActionResult>** Delete(string id)

{

if (id == null)

{

return NotFound();

}

var customers = **await** \_context.Customers

.FirstOrDefaultAsync(m => m.CustomerID == id);

if (customers == null)

{

return NotFound();

}

return View(customers);

}

// POST: Customers/Delete/5

[**HttpPost**, **ActionName("Delete")**]

[ValidateAntiForgeryToken]

public **async Task<IActionResult>** DeleteConfirmed(string id)

{

var **customers** = **await** \_context.Customers.FindAsync(id);

\_context.Customers.Remove(customers);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CustomersExists(string id)

{

return \_context.Customers.Any(e => e.CustomerID == id);

}

15.View -> Solution Explorer -> 在 Views 資料夾--> 檢視 Customers資料夾

會自動產生 Index.cshtml, Details.cshtml, Create.cshtml, Edit.cshtml, Delete.cshtml

等 5 個Views

16.選擇專案名稱**dotNetCore\_DatabaseFirst** (不要使用IIS Express)→Debug → Start Debugging

17.輸入[https://localhost:5001/customers/index](https://localhost:5001/customers)