



LẬP TRÌNH C# 5

BÀI 4: CRUD – IDENTITY AUTHENTICATION

- ⦿ CRUD – Code First
- ⦿ Dependency Injection trong CRUD
- ⦿ Identity In ASP.NET Core MVC Authentication



- ❑ Viết tắt của 4 từ tiếng Anh: Create, Read, Update, Delete
- ❑ CRUD là 4 tính năng quan trọng nhất để làm việc với Database của một Website



- ❑ Asp.net core sử dụng syntax linq hiện thực các thao tác crud hoặc scaffolding- tự động sinh code controller và view dựa trên thông tin model
- ❑ Ví dụ model cho crud gồm Department & Employee có quan hệ một – nhiều

```
public class Employee
{
    public int Id { get; set; }
    public int DepartmentId { get; set; }
    public string Name { get; set; }
    public string Designation { get; set; }

    public Department Department { get; set; }
}

public class Department
{
    public int Id { get; set; }
    public string Name { get; set; }

    public ICollection<Employee> Employee { get; set; }
}
```

❑ Insert a Single Record

```
public IActionResult InsertSingle()
{
    var dept = new Department()
    {
        Name = "Designing1"
    };
    using (var context = new CompanyContext())
    {
        context.Department.Add(dept);
        context.SaveChanges();
    }
    return View("Common");
}
```

❑ Inserting Multiple Records

```
public async Task<IActionResult> InsertMultiple()
{
    var dept1 = new Department() { Name = "Development" };
    var dept2 = new Department() { Name = "HR" };
    var dept3 = new Department() { Name = "Marketing" };

    using (var context = new CompanyContext())
    {
        context.AddRange(dept1, dept2, dept3);
        await context.SaveChangesAsync();
    }

    return View("Common");
}
```

❑ Inserting Related Records: Entity Framework Core sẽ tự nhận biết mối quan hệ và insert các dữ liệu liên quan

```
public async Task<IActionResult> InsertRelated()
{
    var dept = new Department()
    {
        Name = "Admin"
    };
    var emp = new Employee()
    {
        Name = "Matt",
        Designation = "Head",
        Department = dept
    };
    using (var context = new CompanyContext())
    {
        context.Add(emp);
        await context.SaveChangesAsync();
    }

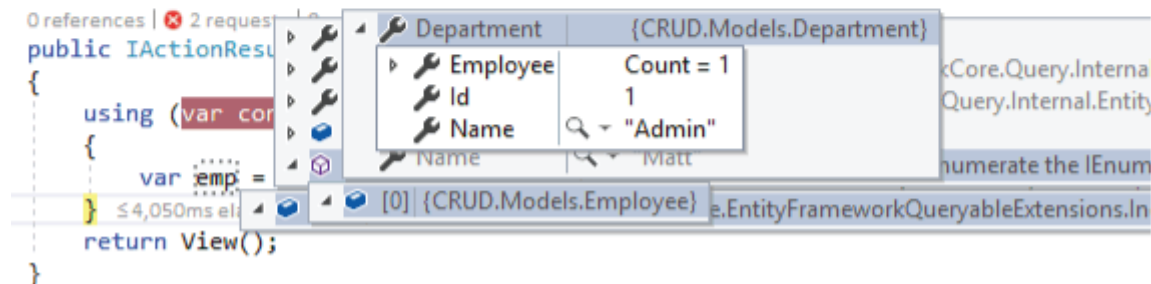
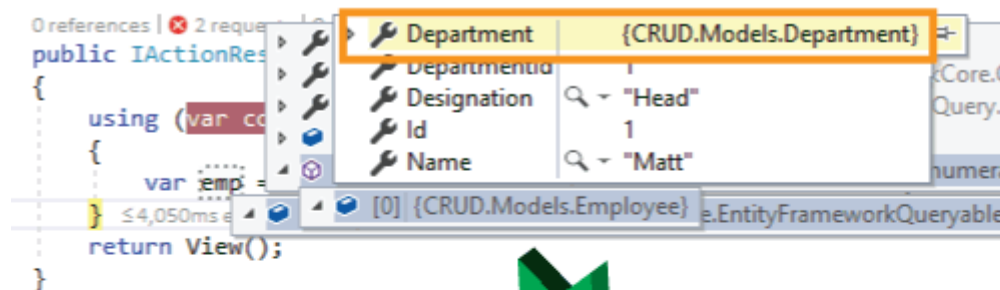
    return View("Common");
}
```

❑ Eager Loading trong EF Core:

- ❖ load tất cả các entity trong 1 câu lệnh, tất cả các entity con sẽ được load ra trong 1 lần gọi duy nhất
- ❖ quá trình trong đó một truy vấn cho một kiểu thực thể cũng tải các thực thể liên quan như một phần của truy vấn, do đó chúng ta không cần phải thực hiện một truy vấn riêng cho các thực thể liên quan
- ❖ Sử dụng phương thức Include, sẽ trả về các entity liên quan như một phần của câu query và một lượng lớn dữ liệu sẽ được load ra 1 lần.

❑ Eager Loading trong EF Core:

```
public async Task<IActionResult> EagerRead()
{
    Employee emp;
    using (var context = new CompanyContext())
    {
        emp = await context.Employee.Where(e => e.Name == "Matt")
                                   .Include(s => s.Department)
                                   .FirstOrDefaultAsync();
    }
    return View("Common");
}
```



- ❑ Lazy Loading: trì hoãn việc tải các dữ liệu liên quan, cho đến khi bạn yêu cầu cụ thể
 - ❖ Cài Microsoft.EntityFrameworkCore.Proxies package
 - ❖ Sử dụng phương thức UseLazyLoadingProxies để cho phép tạo proxies trong phương thức OnConfiguring

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseLazyLoadingProxies().UseSqlServer(@"Server=thepv;Data
    }
}
```

❑ Lazy Loading

- ❖ Sử dụng từ khóa `virtual` cho tất cả các thuộc tính điều hướng như sau

```
public class Employee
{
    public int Id { get; set; }
    public int DepartmentId { get; set; }
    public string Name { get; set; }
    public string Designation { get; set; }

    public virtual Department Department { get; set; }
}

public class Department
{
    public int Id { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Employee> Employee { get; set; }
}
```

❑ Lazy Loading


❖ Xử lý ở controller

```
public async Task<IActionResult> LazyRead()
{
    Employee emp;
    using (var context = new CompanyContext())
    {
        emp = await context.Employee.Where(e => e.Name == "Matt")
                                   .FirstOrDefaultAsync();
        string deptName = emp.Department.Name;
    }
    return View("Common");
}
```

❖ Kết quả dữ liệu Department được load kèm với Employee

References

```
public async Task<IActionResult> LazyRead()
{
    Employee emp;
    using (var context = new CompanyContext())
    {
        emp = await context.Employee.Where(e => e.Name == "Matt")
                                   .FirstOrDefaultAsync();
        string deptName = emp.Department.Name;
    }
    return View("Common");
}
```



The screenshot shows the code in Visual Studio. A variable `deptName` is declared and assigned the value of `emp.Department.Name`. A tooltip is visible for `deptName`, showing its value as `"Admin"`. A large yellow arrow points to the `deptName` variable.

□ Update Single

```
public async Task<IActionResult> UpdateSingle()
{
    var dept = new Department()
    {
        Id = 100,
        Name = "Designing"
    };
    using (var context = new CompanyContext())
    {
        context.Update(dept);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```

□ Update Multiple Records

```
public async Task<IActionResult> UpdateMultiple()
{
    var dept1 = new Department()
    {
        Id = 1,
        Name = "New Designing"
    };
    var dept2 = new Department()
    {
        Id = 2,
        Name = "New Research"
    };
    List<Department> modifiedDept = new List<Department>() { dept1, dept2, dept3 };

    using (var context = new CompanyContext())
    {
        context.UpdateRange(modifiedDept);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```

□ Update Related Records

```
public async Task<IActionResult> UpdateRelated()
{
    var dept = new Department()
    {
        Id = 8,
        Name = "Admin_1"
    };
    var emp = new Employee()
    {
        Id = 1,
        Name = "Matt_1",
        Designation = "Head_1",
        Department = dept
    };
    using (var context = new CompanyContext())
    {
        context.Update(emp);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```

❑ Delete Single

```
public async Task<IActionResult> DeleteSingle()
{
    var dept = new Department()
    {
        Id = 1
    };
    using (var context = new CompanyContext())
    {
        context.Remove(dept);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```

❑ Delete Multiple

```
public async Task<IActionResult> DeleteMultiple()
{
    List<Department> dept = new List<Department>()
    {
        new Department() {Id=1},
        new Department() {Id=2},
        new Department() {Id=3}
    };
    using (var context = new CompanyContext())
    {
        context.RemoveRange(dept);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```

❑ Delete Related Records: tùy thuộc vào cách thiết lập ràng buộc các domain class bằng Fluent API sẽ có các Delete behaviors

1. **Cascade**: Delete the Child Records both in client & Database.
2. **ClientCascade**: Delete the Child Records both in the client only.
3. **SetNull**: Set Foreign Key as NULL in both in Client & Database.
4. **ClientSetNull**: Set Foreign Key as NULL only in the Client
5. **NoAction**: Default behavior on the client and No action on the database
6. **ClientNoAction**: No action on the client and on the database
7. **Restrict**: Same as **NoAction**. The migrations script will generate (Restrict or Non) instead of NoAction.

❑ Delete Related Records

```
entity.HasOne(d => d.Department)
    .WithMany(p => p.Employee)
    .HasForeignKey(d => d.DepartmentId)
    .onDelete(DeleteBehavior.Cascade)
    .HasConstraintName("FK_Employee_Department");
```

```
public async Task<IActionResult> DeleteRelated()
{
    using (var context = new CompanyContext())
    {
        Department dept = context.Department.Where(a => a.Id == 1008)
            .Include(x => x.Employee).FirstOrDefault();
        context.Remove(dept);
        await context.SaveChangesAsync();
    }
    return View("Common");
}
```




DEMO

- Hiện thực các ví dụ





LẬP TRÌNH C# 5

BÀI 4: CRUD – IDENTITY AUTHENTICATION (P2)

- ❑ Cần áp dụng DI trong asp.net core vì các lợi ích mà nó mang lại.
- ❑ Ví dụ cho model Employee

```
[Key]
```

```
4 references | 0 exceptions
```

```
public int EmployeeId { get; set; }
```

```
[Column(TypeName = "nvarchar(250)")]
```

```
[Required(ErrorMessage = "This field is required.")]
```

```
[DisplayName("Full Name")]
```

```
5 references | 0 exceptions
```

```
public string FullName { get; set; }
```

```
[Column(TypeName = "varchar(10)")]
```

```
[DisplayName("Emp. Code")]
```

```
5 references | 0 exceptions
```

```
public string EmpCode { get; set; }
```

```
[Column(TypeName = "varchar(100)")]
```

```
5 references | 0 exceptions
```

```
public string Position { get; set; }
```

```
[Column(TypeName = "varchar(100)")]
```

```
[DisplayName("Office Location")]
```

```
5 references | 0 exceptions
```

```
public string OfficeLocation { get; set; }
```

❑ Tạo Interface quy định các phương thức crud

```
List<Employee> getAll();  
2 references | 0 exceptions  
Employee AddOrEdit(int id = 0);  
2 references | 0 exceptions  
void Delete(int? id);  
1 reference | 0 exceptions  
Employee Add([Bind("EmployeeId,FullName," +  
"EmpCode,Position,OfficeLocation")] Employee employee);  
0 references | 0 exceptions  
Employee Edit([Bind("EmployeeId,FullName," +  
"EmpCode,Position,OfficeLocation")] Employee employee);
```

❑ Tạo class EmployeeService hiện thực crud thông qua DbContext

```
public class EmployeeService : IEmployeeService  
{  
    private readonly EmployeeContext _context;  
    0 references | 0 exceptions  
    public EmployeeService(EmployeeContext context)  
    {  
        _context = context;  
    }  
}
```

- ❑ Implement các phương thức đã khai báo trong `IEmployeeService`

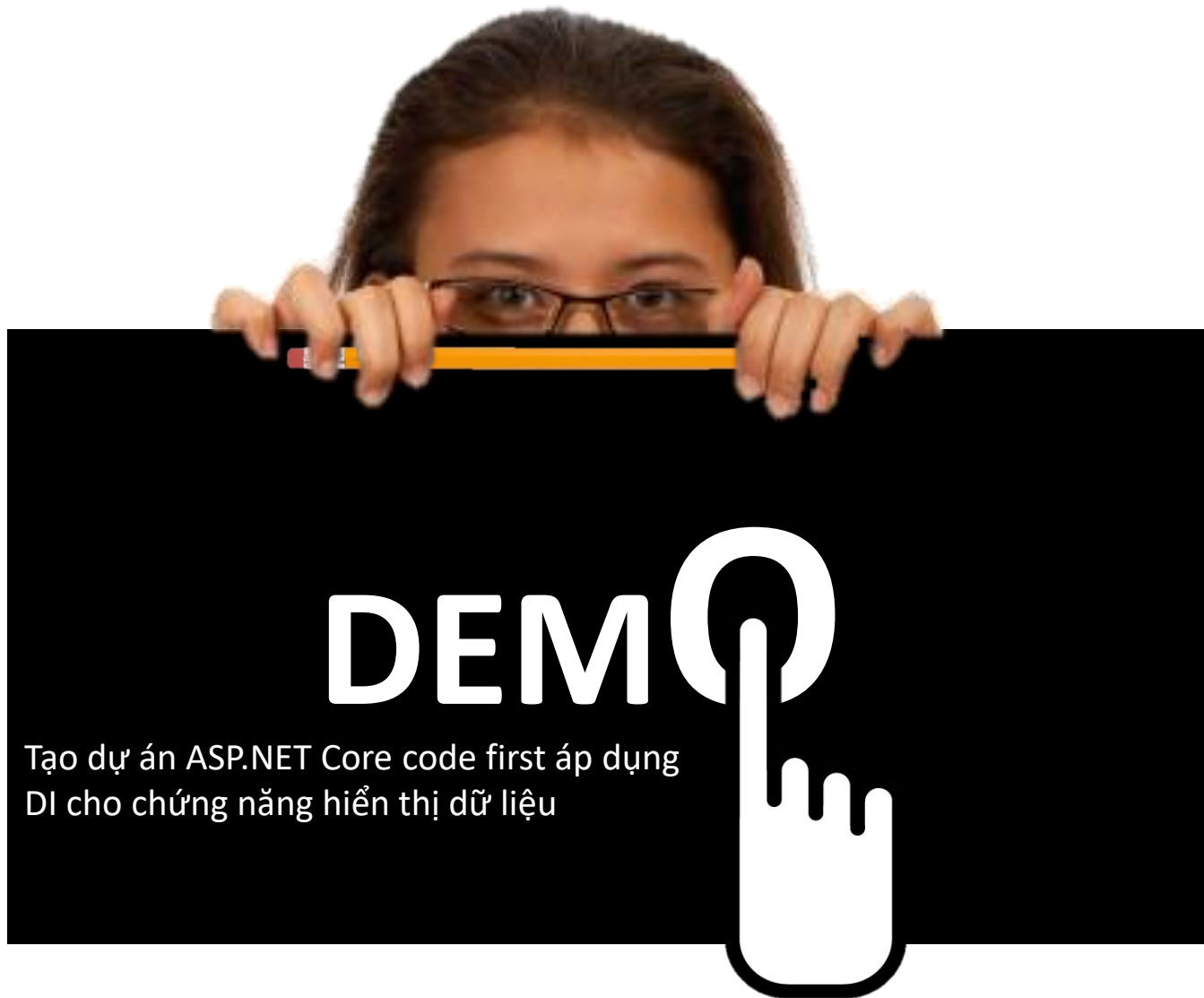
```
public List<Employee> getAll()
{
    return _context.Employees.ToList();
}
1 reference | 0 exceptions
public Employee Add(int id = 0)
{
    if (id == 0)
        return new Employee();
    else
        return _context.Employees.Find(id);
}
1 reference | 0 exceptions
public Employee Add([Bind("EmployeeId,FullName," +
    "EmpCode,Position,OfficeLocation")] Employee employee)
{
    if (employee.EmployeeId == 0)
        _context.Add(employee);
    else
        _context.Update(employee);
    _context.SaveChanges();
    return employee;
}
```

❑ Sử dụng interface trong controller

```
private IEmployeeService _employeeService;  
0 references | 0 exceptions  
public EmployeeController(IEmployeeService employeeService)  
{  
    _employeeService = employeeService;  
}  
  
// GET: Employee  
2 references | 0 requests | 0 exceptions  
public IActionResult Index()  
{  
  
public IActionResult Add(int id = 0)  
{  
    if (id == 0)  
        return View(new Employee());  
    else  
        return View(_employeeService.AddOrEdit(id));  
}  
  
[HttpPost]  
[ValidateAntiForgeryToken]  
0 references | 0 requests | 0 exceptions  
public IActionResult Add([Bind("EmployeeId,FullName,EmpCode,Position,OfficeLocation")])  
{  
    if (ModelState.IsValid)  
    {  
        _employeeService.AddOrEdit(employee);  
        return RedirectToAction(nameof(Index));  
    }  
    return View(employee);  
}
```

□ View hiển thị

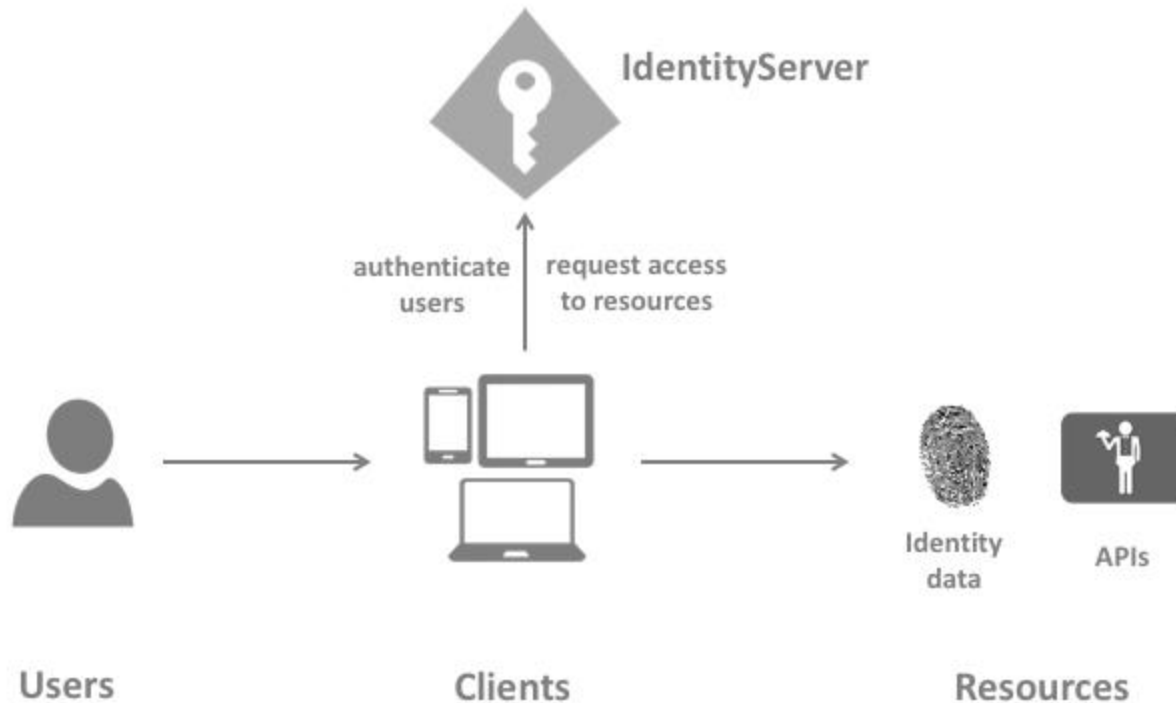
```
@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.FullName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EmpCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Position)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.OfficeLocation)
        </td>
    </tr>
}
```



- ❑ Thành phần (built-in) của [ASP.NET](#) Core, quản lý tài khoản người dùng trong ứng dụng. Nó cung cấp các tính năng cần thiết để quản lý tài khoản user (tạo, xóa, sửa), vai trò (role, phân quyền), claim, đăng ký, đăng nhập, reset password ...
- ❑ Sử dụng được với những provider bên ngoài giống như: Facebook, Google, Twitter, Microsoft account



- ❑ Ứng dụng Identity phát triển chức năng registration và Authentication login





□ Tạo ứng dụng


Create a new ASP.NET Core web application


.NET Core


ASP.NET Core 3.1


 **Empty**
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

 **API**
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

 **Web Application**
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

 **Web Application (Model-View-Controller)**
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

 **Angular**
A project template for creating an ASP.NET Core application with Angular

 **React.js**

Authentication
No Authentication
[Change](#)

Advanced
☒ Configure for HTTPS
☐ Enable Docker Support
(Requires [Docker Desktop](#))

Linux

☐ Enable Razor runtime compilation

Author: Microsoft
Source: Templates 3.1.7

Back

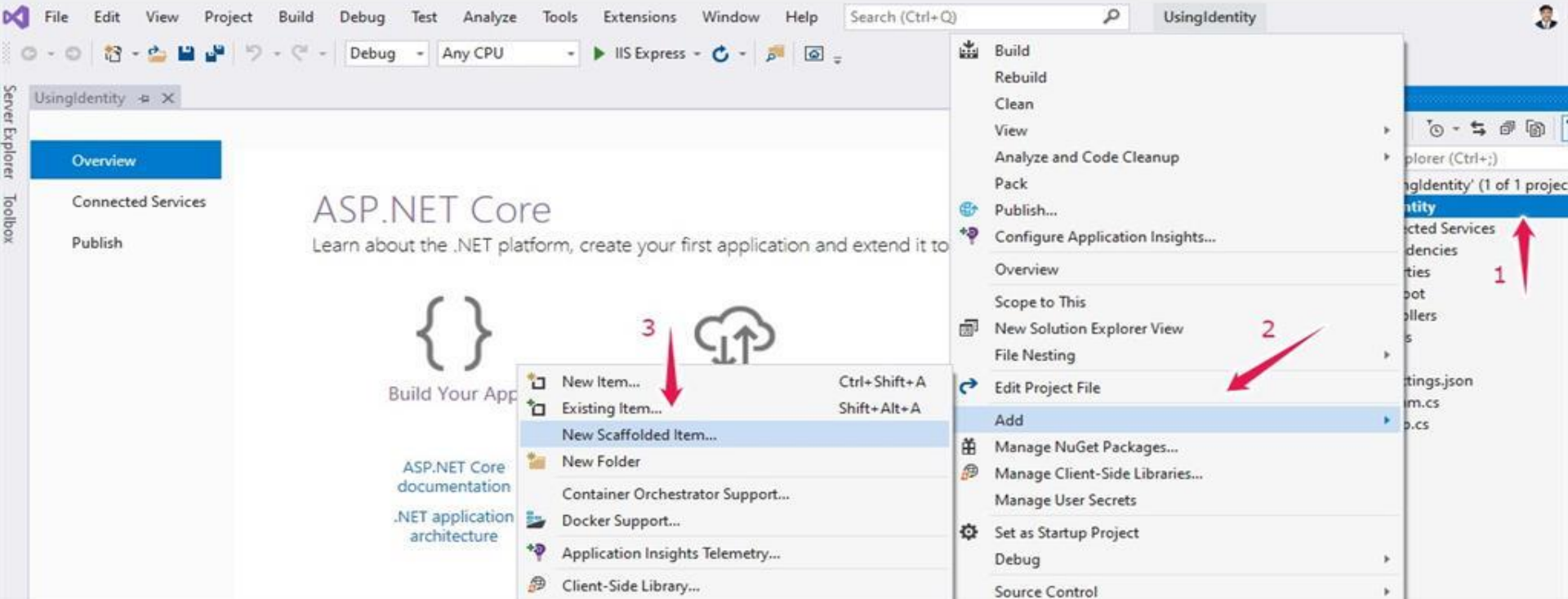
Create

[Get additional project templates](#)

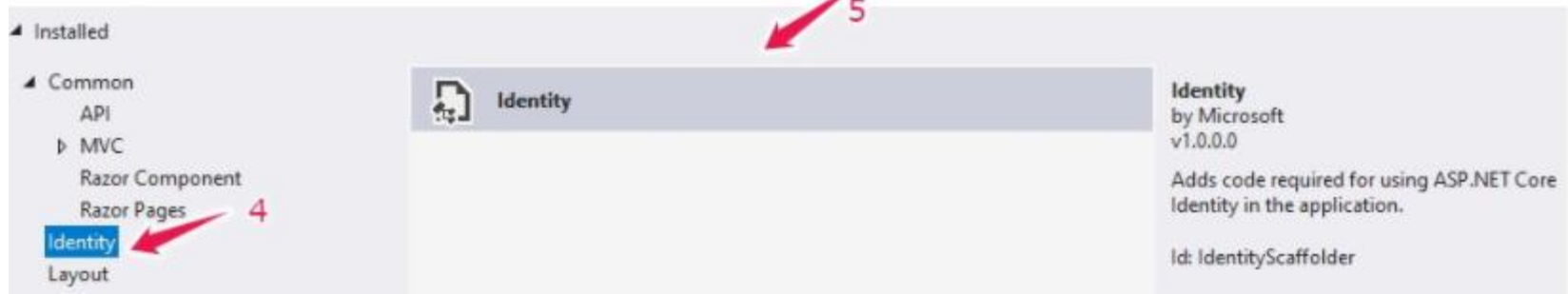
IDENTITY IN ASP.NET CORE MVC

AUTHENTICATION

□ Scaffold Identity



Add New Scaffolded Item



❑ Chọn login, Register và DbContext

Add Identity

Select an existing layout page, or specify a new one:

/Areas/Identity/Pages/Account/Manage/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

☐ Override all files

Choose files to override

- | | | |
|--|---|---|
| <input type="checkbox"/> Account\StatusMessage | <input type="checkbox"/> Account\AccessDenied | <input type="checkbox"/> Account\ConfirmEmail |
| <input type="checkbox"/> Account\ConfirmEmailChange | <input type="checkbox"/> Account\ExternalLogin | <input type="checkbox"/> Account\ForgotPassword |
| <input type="checkbox"/> Account\ForgotPasswordConfirmation | <input type="checkbox"/> Account\Lockout | <input type="checkbox"/> Account>Login |
| <input type="checkbox"/> Account>LoginWith2fa | <input type="checkbox"/> Account>LoginWithRecoveryCode | <input type="checkbox"/> Account\Logout |
| <input type="checkbox"/> Account\Manage\Layout | <input type="checkbox"/> Account\Manage\ManageNav | <input type="checkbox"/> Account\Manage\StatusMessage |
| <input type="checkbox"/> Account\Manage\ChangePassword | <input type="checkbox"/> Account\Manage\DeletePersonalData | <input type="checkbox"/> Account\Manage\Disable2fa |
| <input type="checkbox"/> Account\Manage\DownloadPersonalData | <input type="checkbox"/> Account\Manage\Email | <input type="checkbox"/> Account\Manage\EnableAuthenticator |
| <input type="checkbox"/> Account\Manage\ExternalLogins | <input type="checkbox"/> Account\Manage\GenerateRecoveryCodes | <input type="checkbox"/> Account\Manage\Index |
| <input type="checkbox"/> Account\Manage\PersonalData | <input type="checkbox"/> Account\Manage\ResetAuthenticator | <input type="checkbox"/> Account\Manage\SetPassword |
| <input type="checkbox"/> Account\Manage\ShowRecoveryCodes | <input type="checkbox"/> Account\Manage\TwoFactorAuthentication | <input type="checkbox"/> Account\Register |
| <input type="checkbox"/> Account\RegisterConfirmation | <input type="checkbox"/> Account\ResetPassword | <input type="checkbox"/> Account\ResetPasswordConfirmation |

Data context class:

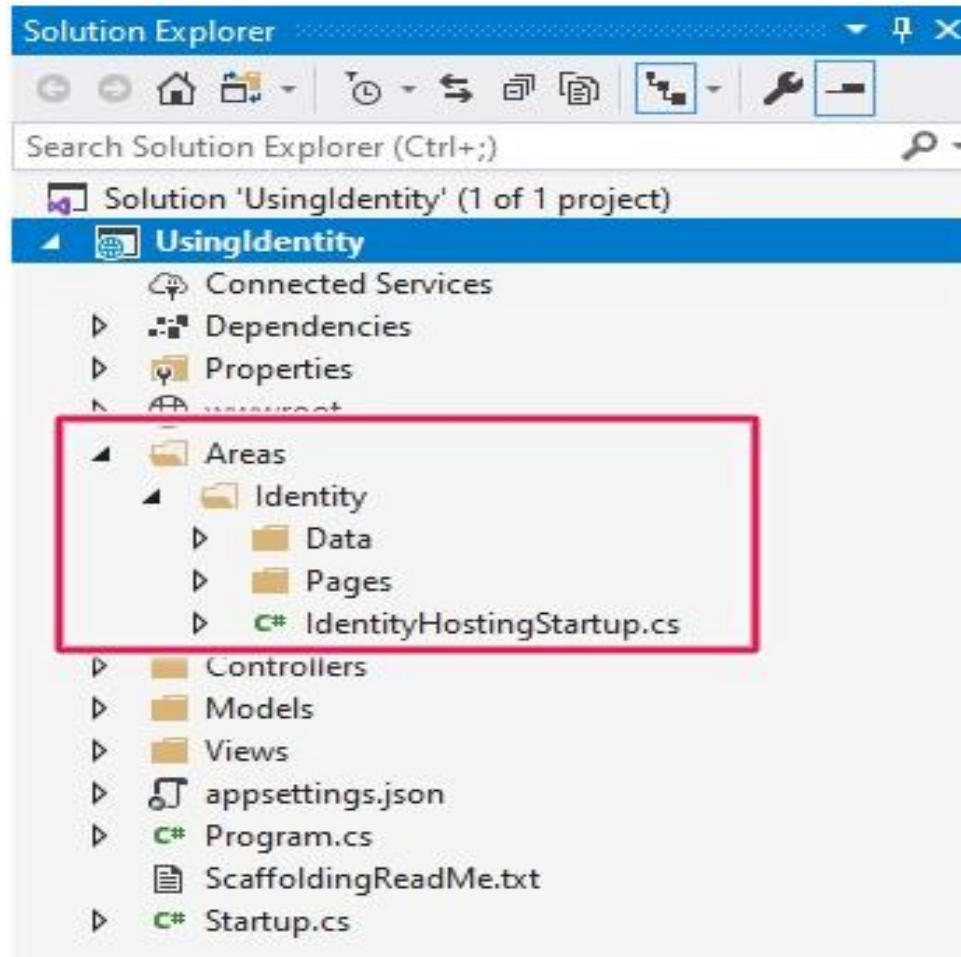
☐ Use SQLite instead of SQL Server

User class:

Add

Cancel

❑ Scaffold Identity thành công



❑ Thêm user authentication vào ứng dụng

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddRazorPages();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
app.UseAuthentication();

app.UseAuthorization();

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

IDENTITY IN ASP.NET CORE MVC

AUTHENTICATION

- ❑ Cập nhật các thông tin User tùy ý trong model UsingIdentityUser

```
public class UsingIdentityUser : IdentityUser
{
    [PersonalData]
    [Column(TypeName = "nvarchar(100)")]
    public string Firstname { get; set; }
    [PersonalData]
    [Column(TypeName = "nvarchar(100)")]
    public string LastName { get; set; }
}
```

- ❑ Cấu hình chuỗi kết nối

```
"ConnectionStrings": {
    "UsingIdentityContextConnection": "Server=ThePV;Database=UsingIdentityDB;"
}
```

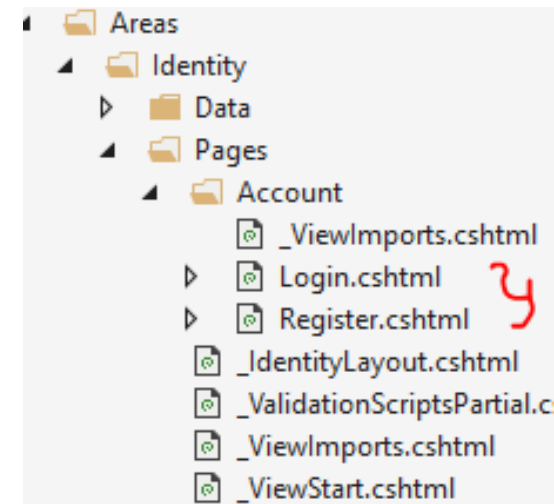
- ❑ Kết nối đến csdl trong class IdentityHostingStartup

```
public void Configure(IWebHostBuilder builder)
{
    builder.ConfigureServices((context, services) => {
        services.AddDbContext<UsingIdentityContext>(options =>
            options.UseSqlServer(
                context.Configuration.GetConnectionString("UsingIdentityContextConnection")
            )
    })
}
```


IDENTITY IN ASP.NET CORE MVC

AUTHENTICATION

□ Cập nhật view login và register thích hợp



UsingIdentity

Register Login

Welcome

Fpoly to Identity In ASP.NET Core MVC Authentication

❑ Form registration và login sẵn sàng

Log In

Sign Up

Register

- The First Name field is required.
- The Last Name field is required.
- The Email field is required.
- The Phone Number field is required.
- The Password field is required.

First Name

The First Name field is required.

Last Name

The Last Name field is required.

Email

The Email field is required.

Phone Number

The Phone Number field is required.

Password

The Password field is required.

Confirm password

Register



DEMO

- Hiện thực các ví dụ



Tổng kết bài học

- ◎ CRUD – Code First
- ◎ Dependency Injection trong CRUD
- ◎ Identity In ASP.NET Core MVC Authentication





KẾT THÚC