

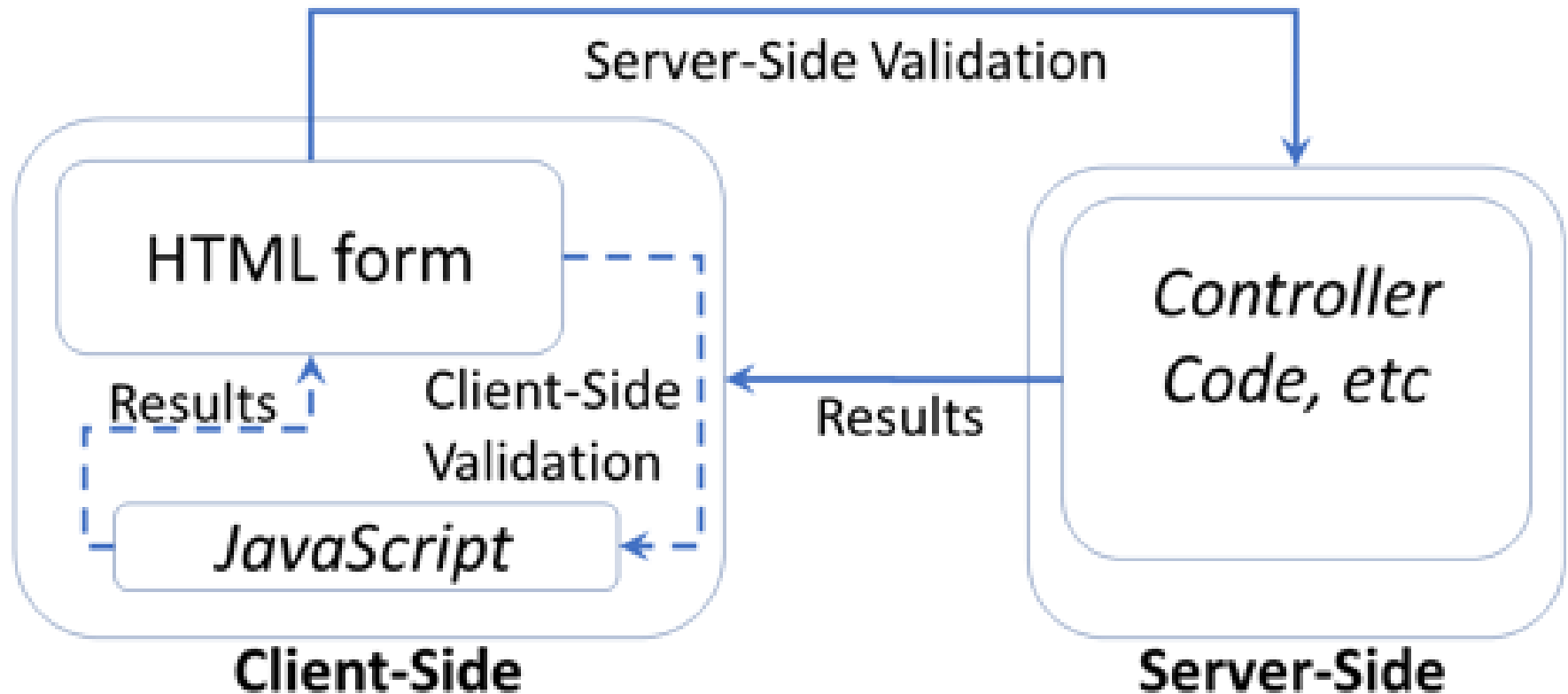
LẬP TRÌNH C# 5

BÀI 3: DATA VALIDATION - DEPENDENCY INJECTION

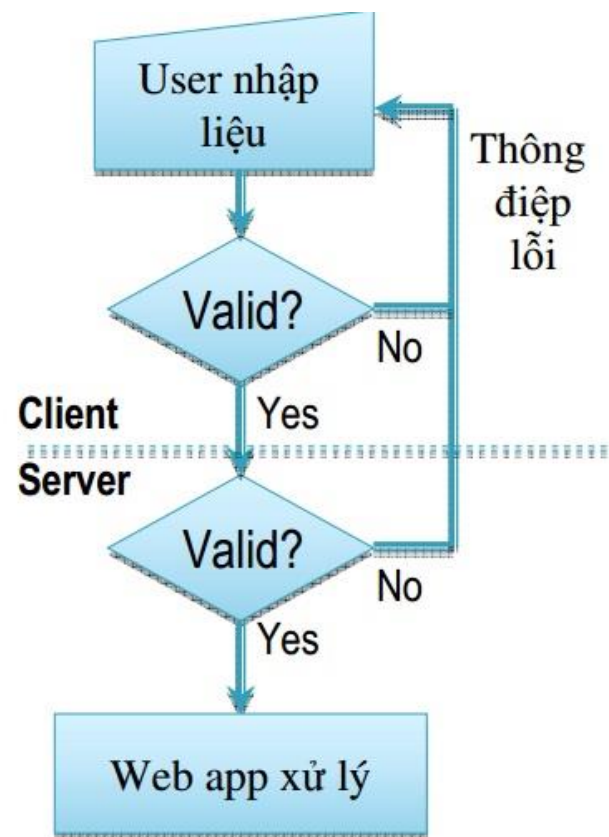
- ⊙ Data Validation
- ⊙ Model Validation
- ⊙ Validation Tag Helpers
- ⊙ Dependency Injection



- ❑ Phương pháp kiểm tra tính hợp lệ và giúp người dùng nhập dữ liệu chuẩn theo yêu cầu của hệ thống
- ❑ Định hướng hỗ trợ người dùng hoàn thiện điền thông tin gửi server
- ❑ Giảm nguy cơ xâm nhập các mã độc nhằm mục đích phá hoại hệ thống
- ❑ 2 phương pháp Validation thông dụng: client-side validation và server-side validation
- ❑ Trong asp.net core, có thể validation các nguồn HTML Form, Route, Query String, Request Body, Services, ...



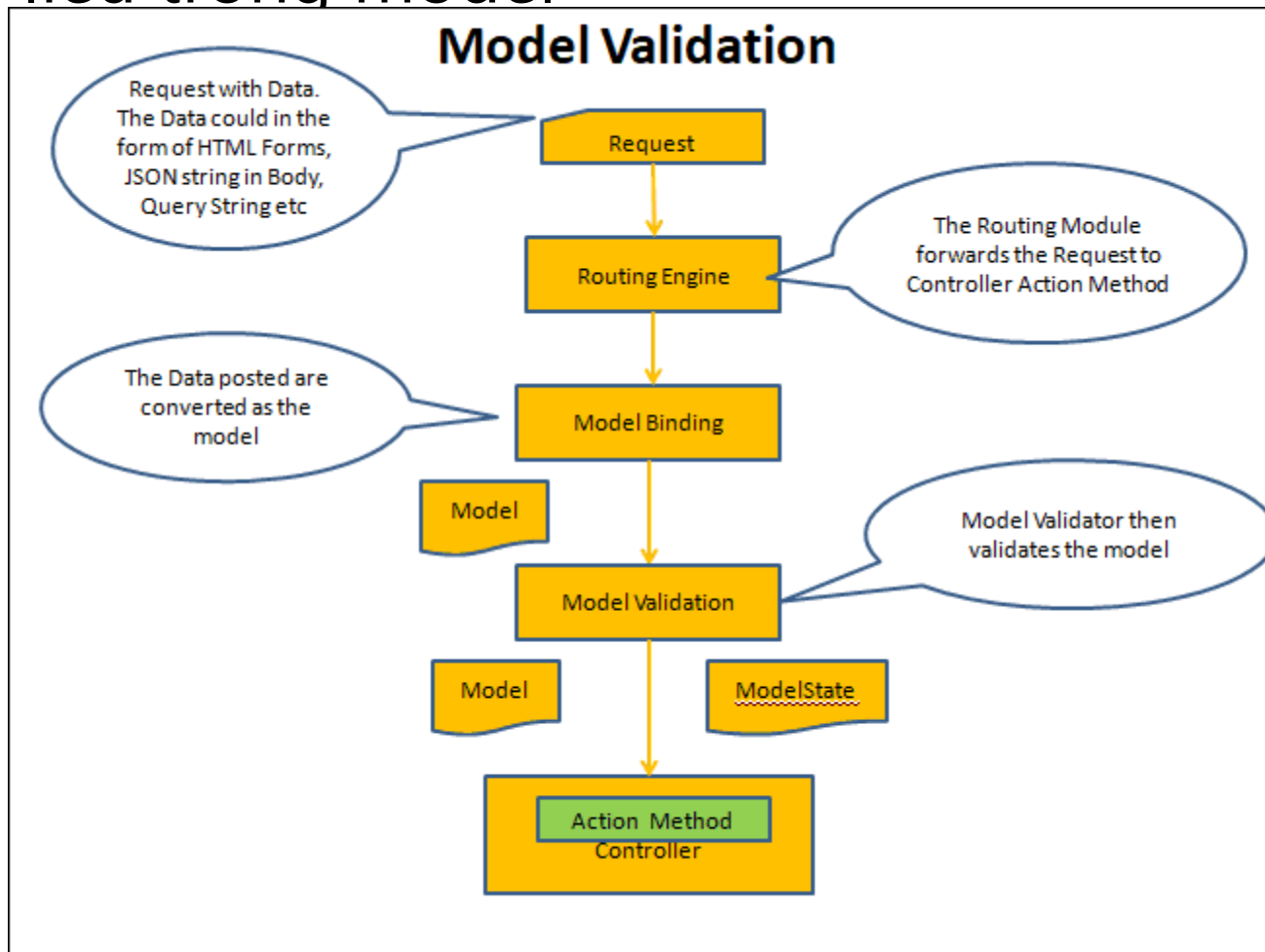
- ❑ Nhằm hợp lệ dữ liệu trước khi được submit lên server
- ❑ Vì việc kiểm tra tiến hành phía trình duyệt client nên phản hồi nhanh hơn và gần như là ngay lập tức
- ❑ Làm giảm lỗi trước khi dữ liệu được gửi lên server-side
- ❑ Giúp tăng sự trải nghiệm



- ❑ Client-side validation cung cấp trải nghiệm người dùng tốt hơn nhưng không tin cậy. Nó có thể lỗi do một trong các lý do sau:
 - ❖ Javascript có thể bị tắt ở trình duyệt
 - ❖ Người dùng có thể gửi trực tiếp dữ liệu đến người dùng mà không sử dụng ứng dụng hoặc sử dụng một số các trình chỉnh sửa request có hại.
 - ❖ Khi Javascript có lỗi thì kết quả là dữ liệu được đưa vào hệ thống mà có thể không hợp lệ
- ❑ Giúp hệ thống ngăn chặn được các mối nguy hại và lỗ hổng bảo mật khi bị tấn công vào server
- ❑ Sử dụng công nghệ, ngôn ngữ lập trình phía server

- ❑ Validations trong ASP.NET Core được thực hiện dựa trên các kỹ thuật Route Constraints, ModelState, Attributes, Tag Helpers, jQuery Unobtrusive Validation
- ❑ Validation Attributes (Model Validation): thường được áp dụng trên các Properties trong Model classes, kiểm tra dữ liệu để đảm bảo rằng dữ liệu đó đáp ứng các yêu cầu đặt ra
- ❑ Validation Tag Helpers: sử dụng tại view để cung cấp các thông báo trực quan tới client
- ❑ jQuery Unobtrusive Validation: thư viện tích hợp trong asp.net core hỗ trợ client validation

- ❑ ASP.NET Core cung cấp cơ chế kiểm tra dữ liệu thông qua Model Validator, nó dùng các attribute để kiểm tra dữ liệu trong model



❑ Một số Validation Attributes thông dụng

| Attribute | Description |
|-------------------|---|
| CreditCard | This validates that the property has a credit card format. |
| Compare | This attribute validates that two property in model class match like password and compare password. |
| EmailAddress | This validates the property has email address format. |
| Phone | This validates that the property has a telephone number format. |
| Range | This validates that the property value within a specified range. |
| RegularExpression | This validates that the property value matches a specified regular expression. |
| Required | This validates that the field is not null |
| StringLength | This validates that a string property value doesn't exceed a specified length limit. |
| Url | This validates that the property has a URL format. |
| Remote | This validates input on the client by calling an action method on the server |

❑ Các Validation Attributes được gắn vào properties trong các domain class và map với asp-for Tag Helper

❑ Ví dụ Validation Attributes cho Engineers class

```

namespace ThEngineeringProjects.Models
{
    17 references
    public class Engineers
    {
        9 references
        public int Id { get; set; }

        [Required]
        [MaxLength(50, ErrorMessage = "Username exceeds the required limit.")]
        3 references
        public string Username { get; set; }

        [Required]
        [RegularExpression(@"^[a-zA-Z0-9_+]+\@[a-zA-Z0-9]+\.[a-zA-Z0-9-]+\.$",
            ErrorMessage = "Invalid Email Address.")]
        3 references
        public string Email { get; set; }

        [Required]
        [MaxLength(50, ErrorMessage = "Full name exceeds the required limit.")]
        [Display(Name = "Full Name")]
        3 references
        public string FullName { get; set; }

        [Required]
        3 references
        public DeptEnum? Department { get; set; }

        [Required]
        9 references
        public UniEnum? University { get; set; }

        0 references
        public string Bio { get; set; }

        0 references
        public string PersonalAddress { get; set; }
    }
}
    
```

- ❑ Model Binder sẽ bind và kiểm tra dữ liệu nhận từ HTTP Request → tạo một đối tượng ModelState chứa các danh sách thông báo lỗi được tạo ra bởi Model Binder và đặt thuộc tính IsValid là false trước khi gọi action method trong controller
- ❑ Cần kiểm tra ModelState.IsValid để biết xem quá trình kiểm tra có hợp lệ hay không để action method có thể trả về danh sách lỗi nếu cần.

```
if (ModelState.IsValid)
{
    //Model is valid. Call Service Layer for further processing of data
} else {
    //Validation failed. Return the model to the user with the relevant error messages
}
```

- ❑ ModelState hỗ trợ danh sách Properties và Method dùng quản lý lỗi rất mạnh:
<https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.modelbinding.modelstatedictionary?view=aspnetcore-5.0>
- ❑ Ví dụ xử lý ModelState trong controller

```
[HttpPost]
References
public IActionResult Registration(Engineers engineer)
{
    if (ModelState.IsValid)
    {
        Engineers newEngineer = _engineersRepository.AddNewEngineer(engineer);
        return RedirectToAction("info", new { id = newEngineer.Id });
    }

    return View();
}
```

- ❑ Validation Message Tag Helper: sử dụng `` tag kết hợp `asp-validation-for` Tag Helper để hiển thị một thông báo lỗi
- ❑ Validation Summary Tag Helper: sử dụng `<div>` kết hợp `asp-validation-summary` Tag Helper (All, ModelOnly, None) hiển thị danh sách thông báo lỗi

The diagram shows a form with two input fields and a submit button. The first field is labeled 'Quantity' and contains the value '0'. Below it, a red error message states: 'The field Quantity must be between 1 and 1000.' The second field is labeled 'Currency To' and is empty. Below it, a red error message states: 'The Currency To field is required.' A 'Submit' button is located below the 'Currency To' field. To the left of the form, two labels with arrows point to the error messages: 'Validation Message Tag Helpers' points to the error for the 'Quantity' field, and 'Validation Summary Tag Helper' points to the summary of errors below the 'Submit' button. The summary of errors is a list of three items: 'The Currency To field is required.', 'Not a valid currency code', and 'The field Quantity must be between 1 and 1000.'

Quantity

0

The field Quantity must be between 1 and 1000.

Currency To

The Currency To field is required.

Submit

- The Currency To field is required.
- Not a valid currency code
- The field Quantity must be between 1 and 1000.

❑ Ví dụ Validation errors

```
<form asp-controller="Home" asp-action="Registration" method="post">

    <div>
        <label asp-for="Username"></label>
        <input asp-for="Username" />
        <span asp-validation-for="Username"></span> ①
    </div>

    <div>
        <label asp-for="Email"></label>
        <input asp-for="Email" />
        <span asp-validation-for="Email"></span> ②
    </div>

    <div>
        <label asp-for="FullName"></label>
        <input asp-for="FullName" />
        <span asp-validation-for="FullName"></span> ③
    </div>

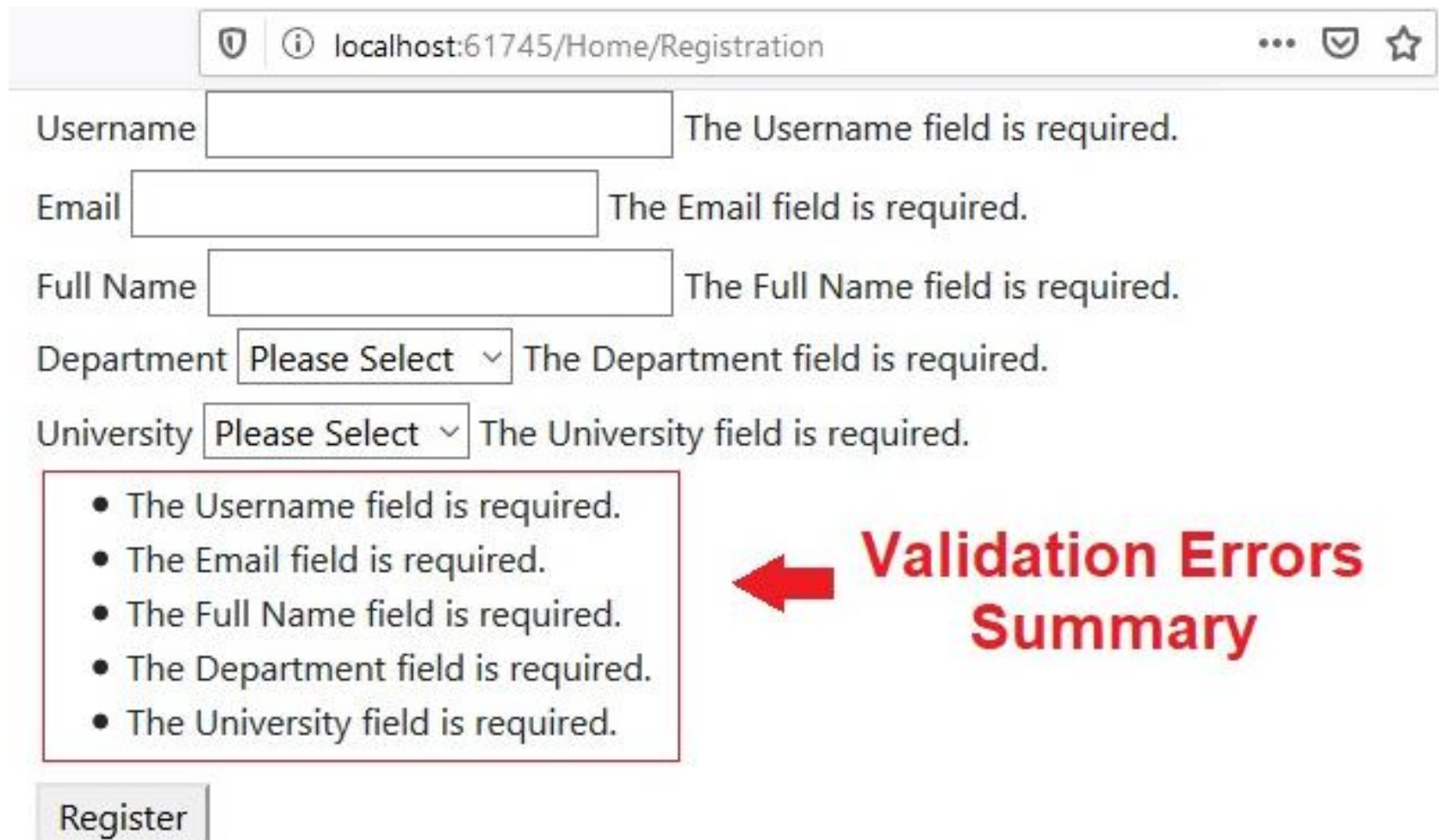
    <div>
        <label asp-for="Department"></label>
        <select asp-for="Department"
            asp-items="Html.GetEnumSelectList<DeptEnum>()">
            <option value="">Please Select</option>
        </select>
        <span asp-validation-for="Department"></span> ④
    </div>

    <div>
        <label asp-for="University"></label>
        <select asp-for="University"
            asp-items="Html.GetEnumSelectList<UniEnum>()">
            <option value="">Please Select</option>
        </select>
        <span asp-validation-for="University"></span> ⑤
    </div>

    <div asp-validation-summary="All"></div> ⑥

    <div>
        <button type="submit">Register</button>
    </div>
</form>
```


❑ Chạy kiểm tra các thông báo lỗi



localhost:61745/Home/Registration

Username The Username field is required.

Email The Email field is required.

Full Name The Full Name field is required.

Department The Department field is required.

University The University field is required.

- The Username field is required.
- The Email field is required.
- The Full Name field is required.
- The Department field is required.
- The University field is required.

Register

Validation Errors Summary

- ❑ Thư viện tích hợp với Asp.net core
- ❑ Thư viện unobtrusive client-side cũng sử dụng các Validation Attributes để kiểm tra các thuộc tính trên phía client
- ❑ Unobtrusive validation sử dụng jquery.validate.unobtrusive.js. Thư viện này được xây dựng dựa trên jquery.validate.js, cần import tất cả vào view

```
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js"></script>  
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.16.0/jquery.validate.min.js"></script>  
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.6/jquery.validate.unobtrusive.min.js"></script>
```


❑ Thêm Validation Attribute vào thuộc tính model

```
[Required(AllowEmptyStrings = false, ErrorMessage = "Please enter the name")]  
[StringLength(maximumLength: 25, MinimumLength = 10, ErrorMessage = "Length must be between 10 to 25")]  
public string Name { get; set; }
```

❑ Tạo controller

```
[HttpPost]  
public IActionResult Create(ProductEditModel model, [FromQuery] string test)  
{  
    string message = "";  
    if (ModelState.IsValid){  
        message = "product " + model.Name + " created successfully";  
    }  
    else {  
        return View(model);  
    }  
    return Content(message);  
}
```

❑ Thêm view hiển thị thông báo

```
<form asp-controller="home" asp-action="create" method="post">

    <div asp-validation-summary="ModelOnly">
        <span>Please correct the following errors</span>
    </div>

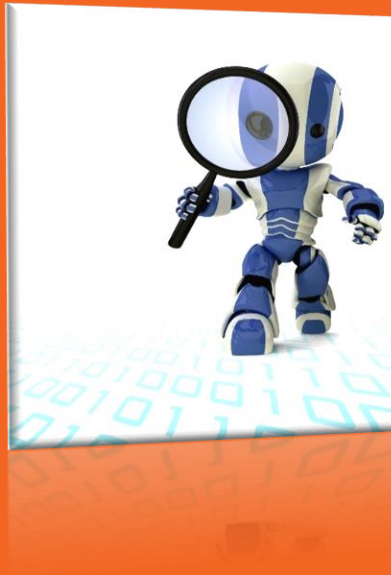
    <label asp-for="Name"></label>
    <input asp-for="Name"/>
    <span asp-validation-for="Name"></span>
    <br />
    <input type="submit" name="submit" />
</form>
```



DEMO

- Hiện thực các ví dụ



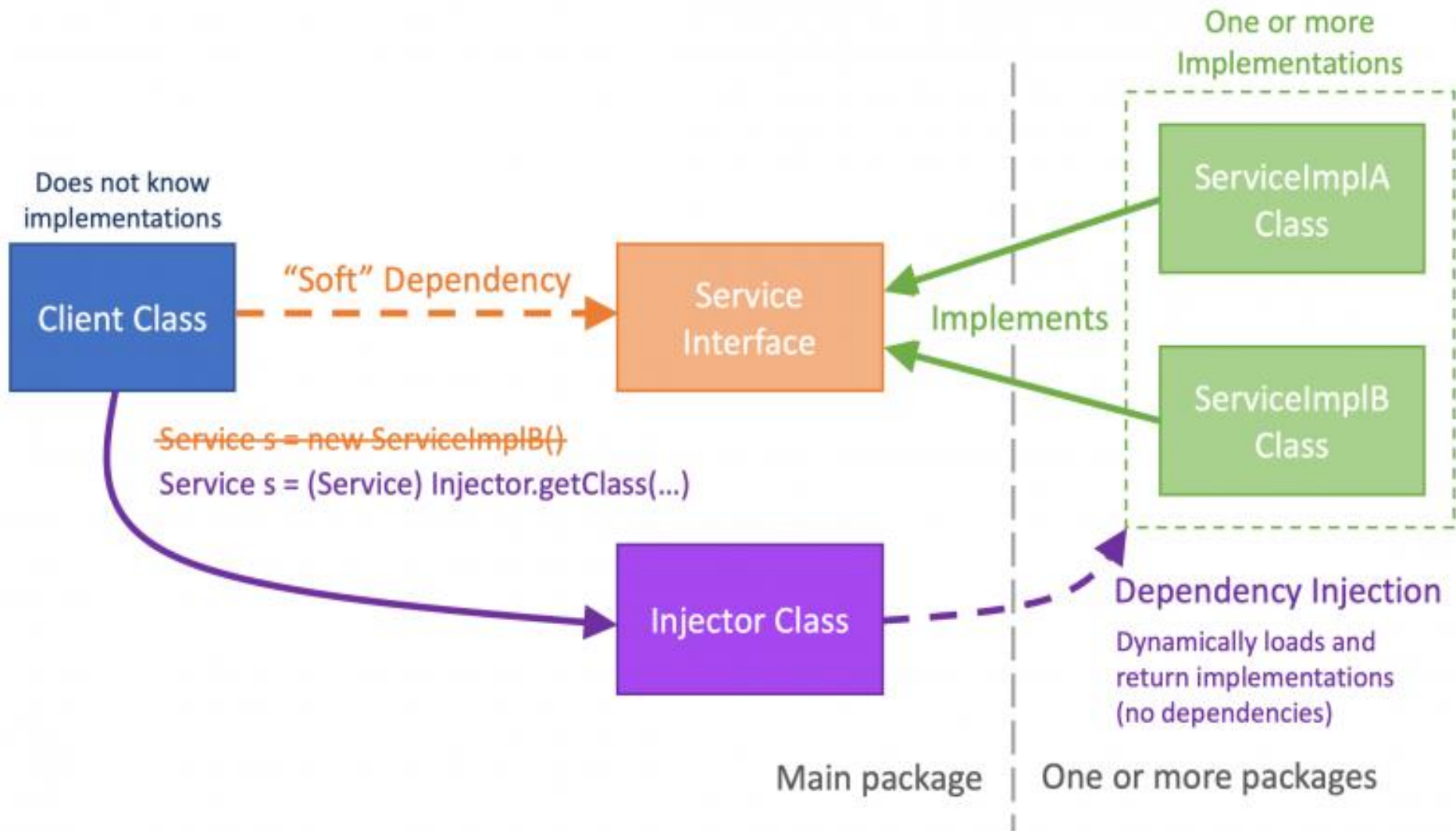


LẬP TRÌNH C# 5

BÀI 3: DATA VALIDATION - DEPENDENCY INJECTION (P2)

- ❑ Dependency Injection là 1 kỹ thuật, 1 design pattern cho phép xóa bỏ sự phụ thuộc hard-code và làm cho ứng dụng dễ mở rộng và maintain
- ❑ Các module cấp cao không nên phụ thuộc vào các modules cấp thấp. Cả 2 nên phụ thuộc vào abstraction (interface)
- ❑ Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation)
- ❑ DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng hơn trong việc thay đổi module, bảo trì code và testing
- ❑ Việc Module nào gắn với interface nào sẽ được config trong file Startup.cs

DEPENDENCY INJECTION



❑ Ví dụ Dependency Injection

```
public class ProductViewModel
{
    public int Id { get; set; }
    public string Name { get; internal set; }
}
```

Tạo Interface và class implement

```
public interface IProductService
{
    List<ProductViewModel> getAll();
}

public class ProductService : IProductService
{
    public List<ProductViewModel> getAll()
    {
        return new List<ProductViewModel>
        {
            new ProductViewModel {Id = 1, Name = "Pen Drive" },
            new ProductViewModel {Id = 2, Name = "Memory Card" },
            new ProductViewModel {Id = 3, Name = "Mobile Phone" },
            new ProductViewModel {Id = 4, Name = "Tablet" },
            new ProductViewModel {Id = 5, Name = "Desktop PC" } ,
        };
    }
}
```

- ❑ ASP.NET Core Dependency Injection dùng DI Container tạo ra thể hiện của ProductService trong controller bằng cách khai báo thể hiện của interface và cập giá trị của thể hiện này trong constructor

```
private IProductService _productService;  
  
public HomeController(IProductService productService)  
{  
    _productService = productService;  
}  
  
public IActionResult Index()  
{  
    _productService = new ProductService();  
    return View(_productService.All());  
}
```


- ❑ Đăng ký service với Dependency Injection container thông qua phương thức AddTransient

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddTransient<IProductService, ProductService>();
}
```

- ❑ View hiển thị

```
@model List<DependencyInjection.Models.ProductViewModel>;
<h2>Index</h2>
```

```
@foreach (var product in Model)
{
    <p>@product.Id @product.Name</p>}
}
```

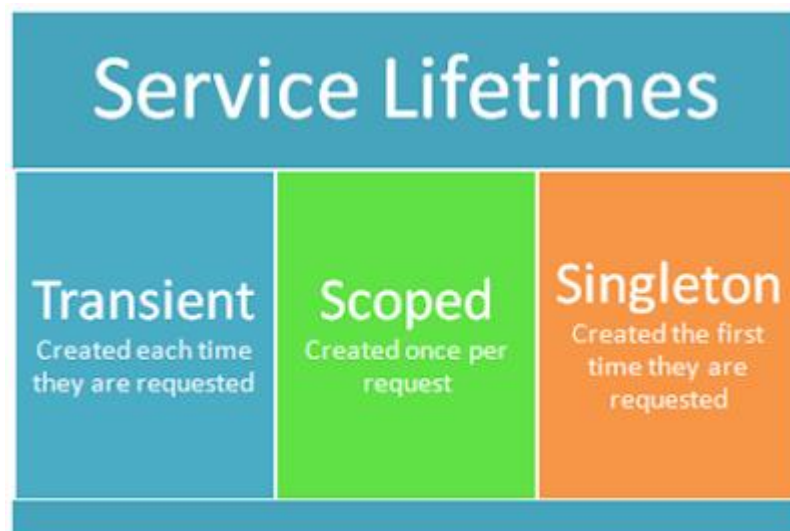


DEMO

- Hiện thực các ví dụ

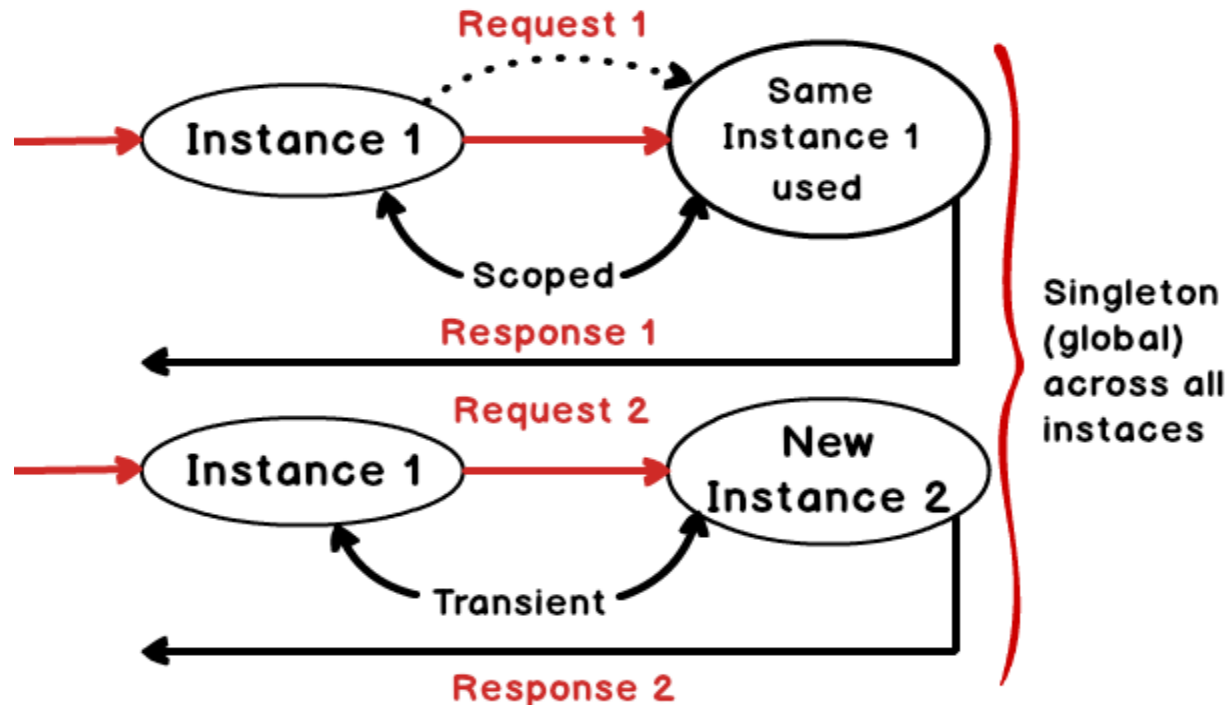


- ❑ Trong ASP .NET Core thì việc hiểu rõ việc khởi tạo các services sử dụng DI (Dependency Injection) container là vô cùng quan trọng.
- ❑ Tùy vào từng service và mục đích sử dụng trong ứng dụng khác nhau mà ta sẽ đăng ký với DI container theo các cách khác nhau. ASP .NET Core cung cấp 3 cách



DEPENDENCY INJECTION LIFETIME

- ❑ Transient: Instance được khởi tạo mỗi lần tạo service
- ❑ Scoped: Instance được khởi tạo mỗi scope. (Scope ở đây chính là mỗi request gửi đến ứng dụng). Trong cùng một scope thì service sẽ được tái sử dụng.
- ❑ Singleton: Instance của service được tạo duy nhất lúc khởi chạy ứng dụng và được dùng ở mọi nơi.



❑ Ví dụ quản lý Dependency Injection Lifetime

Tạo Interface và class implement

```
public interface ITransientService
{
    0 references | 0 exceptions
    Guid GetID();
}

0 references
public interface IScopedService
{
    0 references | 0 exceptions
    Guid GetID();
}

0 references
public interface ISingletonService
{
    0 references | 0 exceptions
    Guid GetID();
}
```

```
public class SomeService : ITransientService,
    IScopedService,
    ISingletonService
{
    Guid id;
    0 references | 0 exceptions
    public SomeService()
    {
        id = Guid.NewGuid();
    }

    3 references | 0 exceptions
    public Guid GetID()
    {
        return id;
    }
}
```

Lập trình C#

Inject các service vào controller

```
public class HomeController : Controller
```

```
{
```

```
    ITransientService _transientService1;
```

```
    ITransientService _transientService2;
```

```
    IScopedService _scopedService1;
```

```
    IScopedService _scopedService2;
```

```
    ISingletonService _singletonService1;
```

```
    ISingletonService _singletonService2;
```

```
0 references | 0 exceptions
```

```
public HomeController(ITransientService transientService1,
```

```
    ITransientService transientService2,
```

```
    IScopedService scopedService1,
```

```
    IScopedService scopedService2,
```

```
    ISingletonService singletonService1,
```

```
    ISingletonService singletonService2)
```

```
{
```

```
    _transientService1 = transientService1;
```

```
    _transientService2 = transientService2;
```

```
    _scopedService1 = scopedService1;
```

```
    _scopedService2 = scopedService2;
```

```
    _singletonService1 = singletonService1;
```

```
    _singletonService2 = singletonService2;
```

```
}
```

```
public IActionResult Index()
```

```
{
```

```
    ViewBag.message1 = "First Instance " +  
        _transientService1.GetID().ToString();
```

```
    ViewBag.message2 = "Second Instance " +  
        _transientService2.GetID().ToString();
```

```
    ViewBag.message3 = "First Instance " +  
        _scopedService1.GetID().ToString();
```

```
    ViewBag.message4 = "Second Instance " +  
        _scopedService2.GetID().ToString();
```

```
    ViewBag.message5 = "First Instance " +  
        _singletonService1.GetID().ToString();
```

```
    ViewBag.message6 = "Second Instance " +  
        _singletonService2.GetID().ToString();
```

```
    return View();
```

```
}
```

□ Tạo view kiểm tra

```
<h3>Transient Service</h3>
```

```
@ViewBag.message1
```

```
</br>
```

```
@ViewBag.message2
```

```
<h3>Scoped Service</h3>
```

```
@ViewBag.message3
```

```
</br>
```

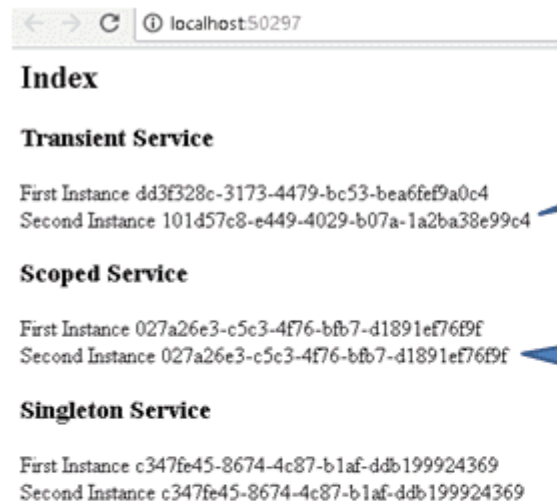
```
@ViewBag.message4
```

```
<h3>Singleton Service</h3>
```

```
@ViewBag.message5
```

```
</br>
```

```
@ViewBag.message6
```



← → ↻ localhost:50297

Index

Transient Service

First Instance dd3f328c-3173-4479-bc53-bea6fef9a0c4
Second Instance 101d57c8-e449-4029-b07a-1a2ba38e99c4

Scoped Service

First Instance 027a26e3-c5c3-4f76-bfb7-d1891ef76f9f
Second Instance 027a26e3-c5c3-4f76-bfb7-d1891ef76f9f

Singleton Service

First Instance c347fe45-8674-4c87-b1af-ddb199924369
Second Instance c347fe45-8674-4c87-b1af-ddb199924369

Always returns the new instance

Instance is created only once per request and shared across the request I.e. why you have same Ids generated
New ids are generated, when you click on refresh button

Only one instance is created and shared across the application.
Click on Refresh button, the ids will remain the same

- ❑ Thảo luận, Vậy nên sử dụng cái nào?
- ❑ Quy tắc:
 - ❖ Không bao giờ inject Scoped & Transient service vào Singleton service
 - ❖ Không bao giờ inject Transient Service vào Scope Service



DEMO

- Hiện thực các ví dụ



Tổng kết bài học

- ◎Data Validation
- ◎Model Validation
- ◎Validation Tag Helpers
- ◎Dependency Injection





KẾT THÚC