.NET CORE MVC

Model-View-Controller[[1]](#endnote-1)

Views

[Layout view](https://dotnettutorials.net/lesson/layout-view-in-asp-net-core-mvc/)

[Razor syntax](https://www.tutorialsteacher.com/mvc/razor-syntax)

[Parital view](https://dotnettutorials.net/lesson/partial-view-result-in-asp-net-core-mvc/)

[Dependency Injection](https://dotnettutorials.net/lesson/asp-net-core-dependency-injection/)[[2]](#endnote-2)

[Singleton vs. Transient vs. Scoped](https://dotnettutorials.net/lesson/singleton-vs-scoped-vs-transient-services-in-asp-net-core/)[[3]](#endnote-3)

[State Management](https://dotnettutorials.net/lesson/differences-between-cookies-and-sessions-in-asp-net-core-mvc/)[[4]](#endnote-4)

[Cookies](https://dotnettutorials.net/lesson/cookies-in-asp-net-core-mvc/)

[Sessions](https://dotnettutorials.net/lesson/sessions-in-asp-net-core-mvc/)

[Fluent API Validation](https://dotnettutorials.net/lesson/fluent-api-validation-examples-in-asp-net-core-mvc/)

[Routing](https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-7.0)[[5]](#endnote-5)

[Model Binding](https://dotnettutorials.net/lesson/model-binding-asp-net-core/)[[6]](#endnote-6)

Bind Attribute

Custom Binding

Data Annotation Validation[[7]](#endnote-7)

[\*Data Annotations vs. Fluent API](https://dotnettutorials.net/lesson/data-annotations-vs-fluent-api-in-asp-net-core-mvc/)(FluentValidation)[[8]](#endnote-8)

Model Validation[[9]](#endnote-9)

[Filters](https://dotnettutorials.net/lesson/filters-in-asp-net-core-mvc/)[[10]](#endnote-10)

[Authorization Filters](https://dotnettutorials.net/lesson/authorization-filters-in-asp-net-core-mvc/)

[Action Filters](https://dotnettutorials.net/lesson/action-filters-in-asp-net-core-mvc/)

[Exception Filters](https://dotnettutorials.net/lesson/exception-filter-in-asp-net-core-mvc/)

[Result Filters](https://dotnettutorials.net/lesson/result-filters-in-asp-net-core-mvc/)

[Tag Helpers](https://dotnettutorials.net/lesson/tag-helpers-in-asp-net-core-mvc/)

[Html Helpers](https://dotnettutorials.net/lesson/html-helpers-in-asp-net-core-mvc/)

Data Passing Techniques

[Strongly-type](https://dotnettutorials.net/lesson/strongly-typed-view-asp-net-core-mvc/)

[ViewData](https://dotnettutorials.net/lesson/viewdata-asp-net-core-mvc/)

[ViewBag](https://dotnettutorials.net/lesson/viewbag-asp-net-core-mvc/)

[ViewModel](https://dotnettutorials.net/lesson/view-model-asp-net-core-mvc/)

[TempData](https://dotnettutorials.net/lesson/tempdata-in-asp-net-core-mvc/)

[Areas](https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0#areas)

[Action Method](https://www.tutorialsteacher.com/mvc/action-method-in-mvc)[[11]](#endnote-11)

[Action Selectors](https://www.tutorialsteacher.com/mvc/action-selectores-in-mvc)[[12]](#endnote-12)

[ActionVerbs](https://www.tutorialsteacher.com/mvc/actionverbs-in-mvc)[[13]](#endnote-13)

GHI CHÚ

1. [Model-View-Controller](https://dotnettutorials.net/lesson/introduction-asp-net-core-mvc/):

   Là một Design Pattern

   [Model](https://dotnettutorials.net/lesson/model-asp-net-core-mvc/):

   Truy xuất, xử lý, lưu trữ dữ liệu trong database

   Cung cấp / phản hồi dữ liệu cho controller

   [View](https://dotnettutorials.net/lesson/views-asp-net-core-mvc/):

   Hiển thị giao diện

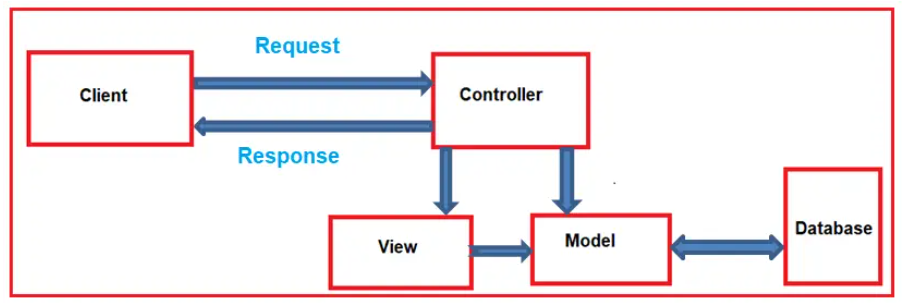
   Tương tác với người dùng

   [Controller](https://dotnettutorials.net/lesson/controllers-asp-net-core-mvc/):

   Xử lý request của người dùng, điều hướng Model / View

   Quyết định response được trả về từ request của người dùng

   Cập nhật Model khi người dùng tương tác, hiển thị thông qua View

    [↑](#endnote-ref-1)
2. [Dependency Injection](https://dotnettutorials.net/lesson/asp-net-core-dependency-injection/) (DI):

   Là một Design Pattern

   Inject (chèn) dependency (phụ thuộc) object vào trong class

   Nguyên tắc IoC (Inversion of Control) để inject class từ bên ngoài

   .Net core, DI container được tích hợp sẵn để quản lý các dependencies [↑](#endnote-ref-2)
3. [Singleton vs. Transient vs. Scoped](https://dotnettutorials.net/lesson/singleton-vs-scoped-vs-transient-services-in-asp-net-core/)

   Lifetime:

   Bắt đầu khi khởi tạo new hoặc trên scope , stack/ heap

   Kết thúc khi delete / ra khỏi scope / không được reference

   Scope:

   Phạm vi của biến, phương thức, object

   Block Scope {}, Method Scope, Class Scope, Namespace Scope, Global Scope

   Ownership:

   Là sở hữu, nói đến object/hàm/thread chịu tránh nhiệm giải phóng tài nguyên

   C# sử Garbage collocter, khi đối tượng không reference sẽ giải phóng.

   Singleton Service:

   Instance được inject chỉ khởi tạo một lần

   Khi inject sẽ là inject reference đến instance đó, không tạo thêm instance mới

   Phù hợp các service có thể reused – tái sử dụng (database connections, configurations settings, … )

   Transient Service:

   Mỗi lần inject sẽ tạo ra một instance mới

   Scoped Service:

   1 resquest 1 instance (HTTP resquest)

   Inject instance nếu cùng 1 lần request => trrong cùng request thì instance giống nhau

   Phù hợp maintain state (duy trì trạng thái) cho 1 request

   |  |  |  |
   | --- | --- | --- |
   | **Singleton** | **Transient** | **Scoped** |
   | One instance for the entire application lifetime. | A new Instance each time it is requested | One Instance per request |
   | Use for shared, long-lived services. | Use for maintaining state within a request. | Use for lightweight, stateless services. |

   [↑](#endnote-ref-3)
4. [State Management](https://dotnettutorials.net/lesson/differences-between-cookies-and-sessions-in-asp-net-core-mvc/):

   Sự khác nhau giữa sessions và cookies

   |  |  |
   | --- | --- |
   | Sessions | Cookies |
   | Stored on the server | Stored on the client’s browser |
   | Short time, closes the browser or after a period of inactivity configured | Long time, for the duration configured by the client's browser or until manually deleted |
   | secure than cookies | cross-site scripting (XSS) and cross-site request forgery (CSRF) |
   | Larger amounts of data | Small amounts of data (4KB) |
   | Session data is only accessible on the server side | Cookies are accessible on both the client side (JavaScript) and server-side (ASP.NET Core) |

   Use Cases

   Cookies: Cookies are commonly used to store user preferences, authentication tokens, and other small pieces of data that need to persist across requests.

   Sessions: Ideal for sensitive data that shouldn’t persist on the client’s machine, like shopping cart contents or current user state in multi-page forms. [↑](#endnote-ref-4)
5. [Routing](https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-7.0):

   Cơ chế quyết định request được xử lý bởi controller nào

   Map request vào Controller, Action method

   Tạo URL tương ứng Controller, Action method [↑](#endnote-ref-5)
6. [Model Binding](https://dotnettutorials.net/lesson/model-binding-asp-net-core/):

   Ánh xạ dữ liệu từ HTTP Request thành parameters của method

   Query string, body, headers, … [↑](#endnote-ref-6)
7. Data Annotation Validation:

   [Data Annotation](https://dotnettutorials.net/lesson/data-annotations-in-asp-net-core-mvc/):

   Là các Attribute được khai báo để ràng buộc dữ liệu trên model

   [Data Annotation Attributes](https://dotnettutorials.net/lesson/data-annotation-attributes-in-asp-net-core-mvc/):

   Requried, StringLength, Range, Email. Regular Expression, …

   [Display & DisplayFormat Attributes](https://dotnettutorials.net/lesson/displaying-and-formatting-attributes-in-asp-net-core-mvc/) [↑](#endnote-ref-7)
8. \*[Data Annotations vs. Fluent API](https://dotnettutorials.net/lesson/data-annotations-vs-fluent-api-in-asp-net-core-mvc/)(FluentValidation):

   |  |  |
   | --- | --- |
   | Data Annotations | Fluent API (FluentValidation) |
   | Uses attributes placed directly on model properties. | Uses a chained method approach. |
   | Limited expressiveness | Highly expressive |
   | Create a new attribute class. | Easily supports custom validation |
   | Slightly faster for basic scenarios since there’s no additional overhead. | It might have a minor performance overhead due to its more dynamic nature, but this isn’t very important in most real-world scenarios. |
   | The code can become cluttered for models with numerous attributes, impacting readability. | Centralizes validation rules, making it potentially easier to read and manage, especially when complex rules exist. |

   [↑](#endnote-ref-8)
9. Model Validation:

   Kiểm tra dữ liệu theo các Validation Attribute trên model trước khi lưu vào database

   Data Annotation là một phần của Model Validation

   ModelState.IsValid, ghi nhận lỗi khi validation

   [↑](#endnote-ref-9)
10. Filters:

    In the ASP.NET Core framework, filters and attributes are used to add additional functionality to controllers and action methods, such as authentication, authorization, caching, and exception handling.

    Xử lý features / logic trước, sau của controller’s action method

    Can thiệp vào quá trinh xử lý HTTP request

    Các loại filters

    Authorization filter [Authorize]

    Action filter

    Exception filter [HandleErrorr]

    Result filter [OutputCatche] [↑](#endnote-ref-10)
11. Action Method

    [Action Results](https://dotnettutorials.net/lesson/action-results-in-asp-net-core-mvc/):

    Kiểu response chỉ định xử lý resquest

    Nguyên tắc

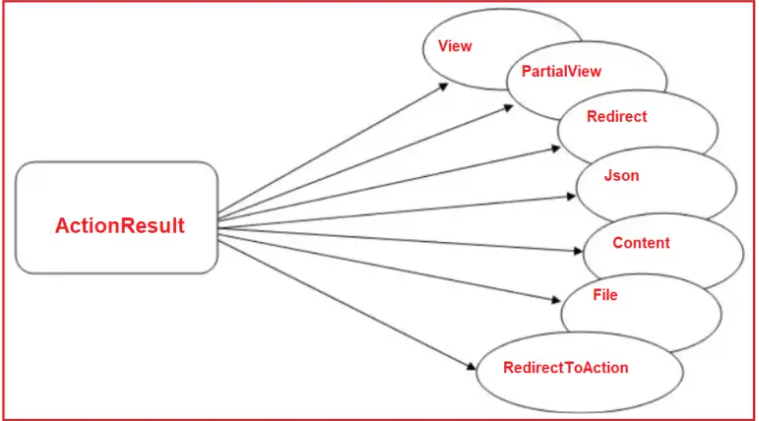
    Các Action method phải là public

    Không được overloaded

    Không là static method

    IactionResult là class base của tất cả result types từ một action method returns

    Cho phép sử dụng các parameters từ request

     [↑](#endnote-ref-11)
12. [Action Seletors](https://www.tutorialsteacher.com/mvc/action-selectores-in-mvc):

    Các Attribute có thể áp dụng cho các Action method

    ActionName, NoAction, ActionVerbs [↑](#endnote-ref-12)
13. [ActionVerbs](https://www.tutorialsteacher.com/mvc/actionverbs-in-mvc):

    The MVC framework includes HttpGet, HttpPost, HttpPut, HttpDelete, HttpOptions, and HttpPatch action verbs.

    Dùng để xử lý các HTTP requests

    Là thuộc tính gắn trên action method [↑](#endnote-ref-13)