**C# and Backend**

Table of Contents

[**1.** **LINQ** 2](#_Toc188285169)

[**2.** **SORT mongodb driver** 2](#_Toc188285170)

[**3.** **Sort có thể sort 1 hay nhiều fields** 2](#_Toc188285171)

[**4.** **Gộp nhiều câu filter với BsonDocument** 2](#_Toc188285172)

[**5.** **Webhook vs. API** 3](#_Toc188285173)

[**6.** **MVC pattern** 3](#_Toc188285174)

[**7.** **N-Layer Architecture** 3](#_Toc188285175)

[**8.** **Data Transfer Object (DTO)** 3](#_Toc188285176)

[**9.** **So sánh MVC, N-Tier và Three-Layer** 3](#_Toc188285177)

1. **LINQ**

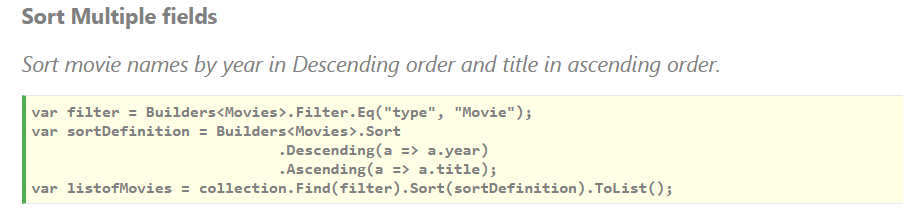
 var listOfNames = new List<string>()  
{  
    "John Doe",  
    "Jane Doe",  
    "Jenna Doe",  
    "Joe Doe"  
};

- The Query syntax (giống kiểu SQL)  
var qNames = from name in listOfNames where name.Length <= 8 select name;

- the Method syntax ( sử dụng lamda, giống arrow func và giống C# )

var mNames = listOfNames.Where(name => name.Length <= 8);

1. **SORT mongodb driver**



1. **Sort có thể sort 1 hay nhiều fields**

Filter 1 document có chứa 1 field là array

Có thể dùng Filter.ElemMatch

Filter.ElemMatch(param1, param2)

Filter.ElemMatch(\_ => \_.HandleableDepartments, Builders<MGTFExDepartmentHandlingModel>.Filter.In(h => h.Status, status));

Param1: **\_ => \_.HandleableDepartments** là field có chứa array

Param2: **Builders<MGTFExDepartmentHandlingModel>.Filter.In(h => h.Status, status)** là 1 filter, filter này sẽ cần 1 field trong array để so sánh

1. **Gộp nhiều câu filter với BsonDocument**



1. **Webhook vs. API**

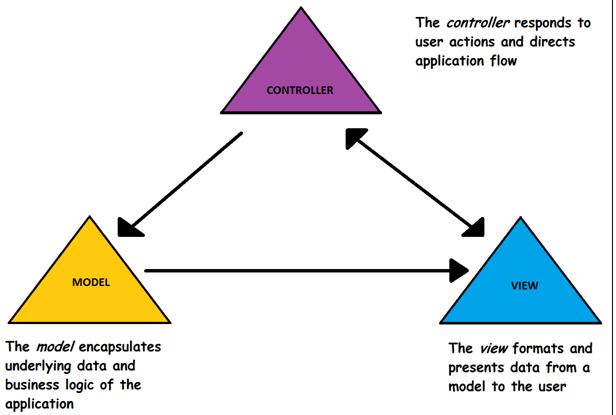
|  |  |
| --- | --- |
| **Webhooks** | **API** |
| POST to the webhook with some JSON data, and then it'll do something internally.  They are **API requests**, but they **help the server** instead of the client  Webhooks **take in data** and do internal work | API would be the team whose job it is to respond to inquiries from external parties  Clients make requests to an API from the client  API endpoints **return data** so that the client asking for that data can do something with it |

1. **MVC pattern**

MVC (Model-View-Controller) It separates an application into **three interconnected components**

* **M-Model**: manage the data, impliment the databasee such as store, get, update, delete data
* **V-View**: render, present the user interface
* **C-Controller**: handle actions, authorize the user, routing the View corresponding with data
* **Flow of Control in MVC**:

The user click a button => the controller handle that action and command the Model process the data => the view will update the UI from Model

* **Separation of Concerns**: seperate the logic, UI, data to make them easy to maintain and test
* **Triangular flow**: 

1. **N-Layer Architecture**

Presentation Layer (controller layer): handle request, intercept authorize user, routing to the correspond Business

Business Logic Layer (Business/ Service Layer): impliment data, modify data, call another services

Data Access Layer (Entity/Repository Layer): retrive, saved, update Data.

1. **Data Transfer Object (DTO)**

* Dùng để giấu cấu trúc của data, encapsulate data
* lọc ra các field cần thiết để trả về cho client,
* validate field data

1. **So sánh MVC, N-Tier và Three-Layer**

Một điểm quan trọng của layered architecture là các layer này phải **xếp chồng lên nhau**. Nguyên tắc là layer trên gọi được xuống (hoặc phụ thuộc vào) các layer dưới mà không phải ngược lại. Ví dụ controller có thể gọi service, service gọi cho repository, nhưng repository không thể gọi cho 2 thằng kia được. Và cũng không bắt buộc phải là 3 layer, có thể thêm các layer khác tùy nhu cầu.

Còn MVC chỉ là **architectural pattern**, tuy cũng chia 3 phần nhưng **không có tính xếp lớp** như layered architecture, mà thiên về kiểu tách chức năng hơn (separation of concerns). Do đó các **pattern này có thể dùng chung với layered architecture** được. Trong các app winform hay web dạng SSR, thì phần model trong MVC vẫn có thể access đến các layer khác như service hoặc repository chẳng hạn.

