Tìm hiểu về MVC

1. Thành phần của MVC

1. **Model**
    - **Chức năng: Đóng gói dữ liệu và logic nghiệp vụ (business logic).**
    - **Nhiệm vụ chính:**
        - **Quản lý trạng thái (state) của ứng dụng.**
        - **Thực hiện các thao tác với cơ sở dữ liệu, tính toán, xác thực nghiệp vụ.**
        - **Phát ra thông báo (event) khi dữ liệu thay đổi để View có thể cập nhật.**
2. **View**
    - **Chức năng: Hiển thị giao diện người dùng (UI).**
    - **Nhiệm vụ chính:**
        - **Lắng nghe sự thay đổi từ Model và render lại (hoặc cập nhật) nội dung phù hợp.**
        - **Tiếp nhận đầu vào (input) từ người dùng (như click, nhập liệu) và gửi tiếp cho Controller.**
3. **Controller**
    - **Chức năng: Điều phối luồng điều khiển giữa View và Model.**
    - **Nhiệm vụ chính:**
        - **Nhận các sự kiện (event) hoặc dữ liệu đầu vào từ View.**
        - **Xử lý logic điều khiển (chọn model nào sẽ thay đổi, view nào sẽ render, …).**
        - **Gọi phương thức của Model để cập nhật dữ liệu, rồi chọn View phù hợp để hiển thị kết quả.**

□ Tách biệt trách nhiệm: Model quản lý dữ liệu, View hiển thị giao diện, Controller điều phối luồng.

- **Quy trình hoạt động: Người dùng tương tác với View → View gửi dữ liệu đến Controller → Controller xử lý và cập nhật Model → Model thay đổi và phát sự kiện → View lắng nghe sự kiện, cập nhật giao diện.**
- **Ưu điểm: Rõ ràng, dễ phân công công việc, tái sử dụng code, hỗ trợ kiểm thử tốt.**
- **Nhược điểm: Phức tạp với ứng dụng nhỏ, nhiều lớp gây rườm rà, không chuẩn hoá chặt chẽ giữa các framework.**

2. Quy trình hoạt động điển hình

1. **Người dùng tương tác với View (ví dụ: nhấn nút "Lưu", nhập dữ liệu,…).**
2. **View gửi sự kiện hoặc yêu cầu đó cho Controller.**
3. **Controller xử lý yêu cầu:**
    - **Kiểm tra tính hợp lệ đầu vào (validation).**
    - **Gọi Model để thay đổi dữ liệu hoặc thực hiện nghiệp vụ.**
4. **Model cập nhật dữ liệu (cơ sở dữ liệu, trạng thái bên trong…) và phát sự kiện "data changed".**
5. **View lắng nghe sự kiện từ Model và cập nhật giao diện tương ứng.**

---

3. Các đặc điểm chính

Đặc điểm	**Mô tả**
Tách biệt trách nhiệm	**Mỗi thành phần chỉ lo một việc: Model lo dữ liệu, View lo giao diện, Controller lo luồng.**
Tái sử dụng (Reusability)	**Model và View có thể được tái sử dụng trên nhiều ứng dụng hoặc giao diện khác nhau.**
Dễ bảo trì (Maintainable)	**Khi thay đổi giao diện không ảnh hưởng tới logic, và ngược lại; đơn giản hóa việc nâng cấp.**
Dễ mở rộng (Extensible)	**Thêm tính năng mới, thêm view mới mà không "đụng chạm" quá sâu đến phần khác.**

Đặc điểm	Mô tả
Hỗ trợ testing	Có thể viết unit test riêng cho Model (business logic) mà không cần quan tâm tới UI.
Giảm coupling	Controller đóng vai trò trung gian, giúp Model và View "lỏng" hơn, không phụ thuộc trực tiếp.

---

4. Ưu – nhược điểm

Ưu điểm

- **Rõ ràng: Quy ước phân chia rõ ràng, giúp nhóm dev dễ hình dung ai làm gì.**
- **Phân công công việc dễ dàng: Front-end dev xây View, back-end dev xây Model.**
- **Tái sử dụng code: Model độc lập, có thể dùng lại trong API, console app,…**
- **Test dễ: Có thể mock các layer khác để test từng phần.**

Nhược điểm

- **Phức tạp với ứng dụng nhỏ: Nếu app chỉ vài màn hình đơn giản, MVC có thể "overkill".**
- **Quá nhiều lớp: Dễ gây boilerplate, cấu trúc thư mục phình to với các file Controller, View, Model.**
- **Không chuẩn hoá chặt chẽ: Tùy framework, người dev có thể đưa quá nhiều logic vào Controller hoặc View.**