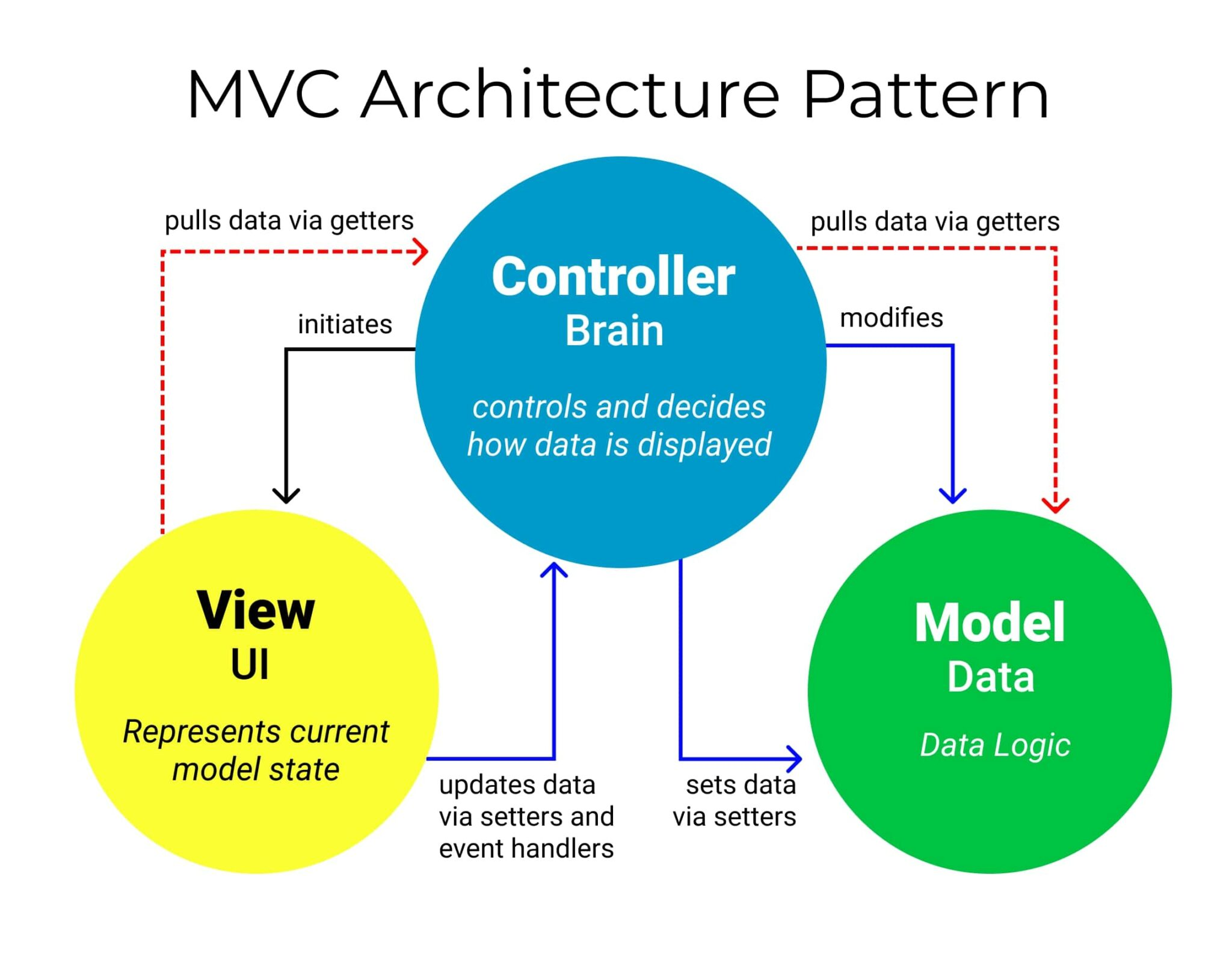
# **Tìm hiểu mô hình MVC**

Mô hình **MVC (Model-View-Controller)** là một kiến trúc phần mềm phổ biến, thường được sử dụng trong phát triển ứng dụng, đặc biệt là ứng dụng web. Nó giúp tách biệt logic nghiệp vụ, giao diện người dùng và luồng điều khiển, làm cho mã nguồn dễ bảo trì, mở rộng và kiểm thử. Dưới đây là giải thích chi tiết về mô hình MVC.



## **Các thành phần của MVC**

### **1. Model (Mô hình)**

* **Chức năng**: Đại diện cho dữ liệu và logic nghiệp vụ. Model quản lý việc truy xuất, lưu trữ và cập nhật dữ liệu.
* **Ví dụ**: Trong ứng dụng quản lý sinh viên, Model có thể là lớp Student (chứa thông tin sinh viên) hoặc StudentDAO (truy cập cơ sở dữ liệu).
* **Vai trò**:
  + Lưu trữ trạng thái ứng dụng (dữ liệu).
  + Thực hiện các thao tác CRUD (tạo, đọc, cập nhật, xóa).
  + Độc lập với giao diện và luồng điều khiển.

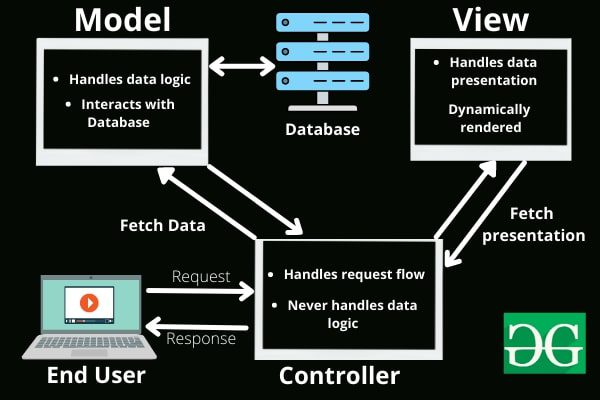
### **2. View (Giao diện)**

* **Chức năng**: Hiển thị dữ liệu từ Model và thu thập yêu cầu từ người dùng để gửi đến Controller.
* **Ví dụ**: Trang HTML, JSP hoặc template như Thymeleaf trong ứng dụng web.
* **Vai trò**:
  + Trình bày dữ liệu trực quan cho người dùng.
  + Không chứa logic nghiệp vụ, chỉ tập trung vào hiển thị và nhận đầu vào.
  + Có thể có nhiều View cho cùng một Model (ví dụ: web và mobile).

### **3. Controller (Bộ điều khiển)**

* **Chức năng**: Nhận yêu cầu từ View, xử lý bằng cách tương tác với Model, rồi cập nhật View để phản hồi.
* **Ví dụ**: Servlet hoặc lớp trong framework như Spring MVC.
* **Vai trò**:
  + Điều phối luồng hoạt động của ứng dụng.
  + Xử lý đầu vào từ người dùng và gọi các phương thức trong Model.
  + Quyết định View nào được sử dụng để hiển thị kết quả.

## **Luồng hoạt động của MVC**



1. Người dùng tương tác với **View** (nhấn nút, gửi form).
2. **Controller** nhận và phân tích yêu cầu.
3. **Controller** gọi **Model** để lấy hoặc cập nhật dữ liệu.
4. **Model** xử lý dữ liệu và trả kết quả.
5. **Controller** gửi dữ liệu đến **View**.
6. **View** hiển thị kết quả cho người dùng.

## **Ưu điểm của MVC**

* **Tách biệt trách nhiệm**: Model, View, Controller có nhiệm vụ riêng, giúp mã nguồn dễ đọc và bảo trì.
* **Dễ kiểm thử**: Kiểm thử từng thành phần độc lập.
* **Hỗ trợ phát triển song song**: Các nhóm có thể làm việc đồng thời trên từng phần.
* **Tái sử dụng mã**: Model và Controller có thể dùng lại cho nhiều View.

## **Nhược điểm của MVC**

* **Phức tạp với ứng dụng nhỏ**: Áp dụng MVC cho ứng dụng đơn giản có thể không cần thiết.
* **Khó hiểu cho người mới**: Luồng điều khiển và tương tác giữa các thành phần có thể gây khó khăn.

## **Ví dụ minh họa**

Trong ứng dụng web quản lý sinh viên:

* **Model**: Lớp Student (mã số, tên, tuổi) và StudentDAO (truy cập dữ liệu).
* **View**: Trang JSP hiển thị danh sách sinh viên hoặc form thêm sinh viên.
* **Controller**: Servlet nhận yêu cầu "/listStudents", gọi StudentDAO, rồi gửi dữ liệu đến JSP để hiển thị.

## **Kết luận**

Mô hình MVC là một kiến trúc mạnh mẽ, phù hợp với các ứng dụng web phức tạp. Nó giúp tổ chức mã nguồn rõ ràng, dễ bảo trì và mở rộng, là nền tảng quan trọng cho người học phát triển ứng dụng.