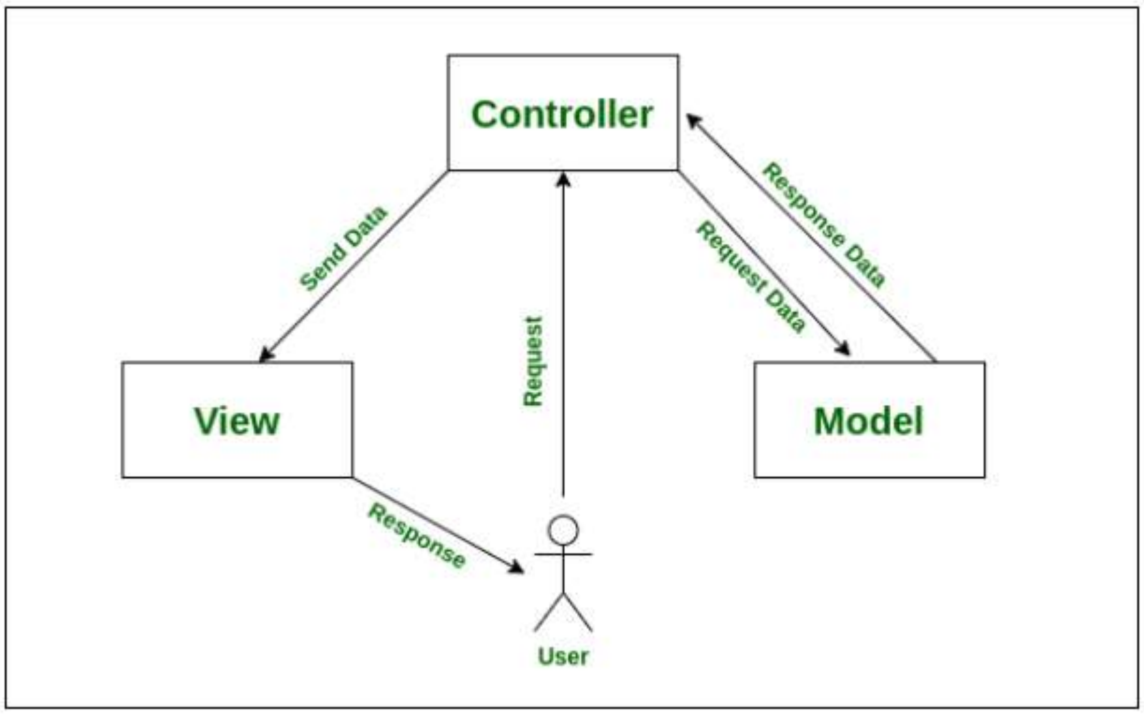
MVC Design Pattern

The **Model View Controller** (MVC) design pattern specifies that an application consists of **a data model, presentation information, and control information**. The pattern requires that each of these be **separated into different objects**. MVC is more of an **architectural pattern**, but not for complete application. MVC mostly relates to the **UI / interaction layer of an application**. You're still going to need a business **logic layer, maybe some service layer, and a data access layer**.

UML Diagram MVC Design Pattern



Design components

- The **Model** contains the **pure application data and pure logic describing how to present the data to a user.** (It's just a data that is shipped across the application like for example from **back-end server view and from front-end view to the database**. In java programming, Model can be represented by the use **of POJO (Plain-old-java-object)** which is a simple java class.
- The **View presents the model's data to the user**. The view knows how to access the model's data, but it **does not know what this data means or what the user can do** to

manipulate it. View just represent, displays the application's data on screen. View page are generally in the format of **.html or .jsp in java programming** (which is flexible).

- The **Controller** exists between the view and the model. It listens to **events triggered by the view (or another external source) and executes the appropriate reaction to these events**. In most cases, the reaction is to **call a method on the model**. Since the view and the model are connected through a **notification mechanism**, the result of this **action is then automatically reflected in the view.**

Advantages

- **Multiple developers can work simultaneously** on the model, controller and views.
- MVC enables **logical grouping of related actions** on a controller together. The **views for a specific model are also grouped** together.
- Models can have **multiple views**.
- The overall components of an application are **easily manageable & are less dependent on each other for proper functioning of application.**

Disadvantages

- The **framework navigation** can be complex because it introduces new layers of abstraction and **requires users to adapt to the decomposition criteria of MVC**.
- Knowledge on multiple technologies becomes the norm. Developers using MVC need to be **skilled in multiple technologies**.