What is MVC?

MVC, short for Model-view-controller, is a software pattern originally developed in the 1970's when graphical user interfaces were in the very early phases of development. MVC strives to separate an application into three areas:

The Model

This is where data and business *logic is store*d. In a typical web application, this part of the application would represent e.g. a database table or any other object that your application should deal with.

The View

User naSystem.***out***.println(coach.getdailyFortune());me

Password

Submit

The View is the actual visual representation of your Model. In a typical web application, this would be the page that shows your Model to the user, whether it be a form for inputting the data or an output of the data or a combination of both. Obviously, a View doesn't have to show all parts of the Model, and a Model can have more than one View.

The Controller

The Controller should handle all communication between your Model and your View. In a typical web application, the methods of the Controller are invoked by the user when a page is loaded or a button is clicked. The Controller then updates the Model, if necessary, and then returns a new View (or action, but more about that later), if necessary.

The MVC (Model-View-Controller) design pattern is a design pattern that's actually been around for a few decades, and it's been used across many different technologies, everything from Smalltalk to C++ to Java and now in C# and .NET as a design pattern to use when you're building a user interface.

* The MVC design pattern is a popular design pattern for the user interface layer of a software application.
* In larger applications, you typically combine a model-view-controller UI layer with other design patterns in the application, like data access patterns and messaging patterns.
* These will all go together to build the full application stack.

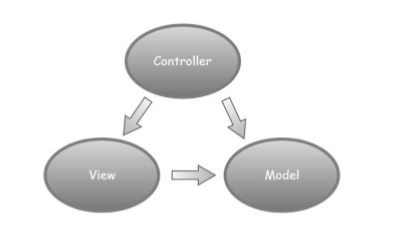
The MVC separates the user interface (UI) of an application into the following three parts −

* **The Model** − A set of classes that describes the data you are working with as well as the business logic.
* **The View** − Defines how the application’s UI will be displayed. It is a pure HTML which decides how the UI is going to look like.
* **The Controller** − A set of classes that handles communication from the user, overall application flow, and application-specific logic.

Idea behind MVC

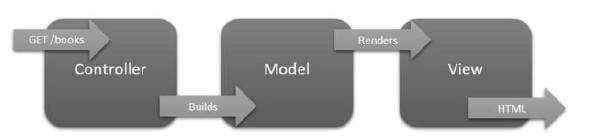
Let us now understand the idea behind MVC.

* The idea is that you'll have a component called the view which is solely responsible for rendering this user interface whether it should be HTML or whether it actually should be a UI widget on a desktop application.
* The view talks to a model, and that model contains all the data that the view needs to display.
* In a web application, the view might not have any code associated with it at all.
* It might just have HTML and then some expressions of where to take the pieces of data from the model and plug them into the correct places inside the HTML template that you've built in the view.



* The controller organizes everything. When an HTTP request arrives for an MVC application, the request gets routed to a controller, and then it's up to the controller to talk to either the database, the file system, or a model.

In MVC, the controller receives an HTTP request, the controller has to figure out how to put together the information to respond to this request. Perhaps the user is directing the browser to the /books URL of the application. So the controller needs to put together the information to display a list of books. In this scenario, the controller will build a model.



* The model doesn't know anything about the HTTP request or the controller.
* The model is only responsible for holding the books information that the user wants to see, as well as any logic associated with that list of books.
* The model is just another Java class we can use and you might have more than one class if you have a complex model.
* Once the model is put together, the controller can then select a view to render the model.
* The view will take the information in the model, like all the books and each book title etc., and it will use that information to construct an HTML page.
* Then that HTML is sent back to the client in the HTTP response and the entire HTTP request and response transaction is completed.

These are the basics of the MVC design pattern and the idea behind this pattern is to keep a separation of concerns. So the controller is only responsible for taking a request and building a model. It is the model that carries the logic and data we need into the view. Then the view is only responsible for transforming that model into HTML.