**What is MVC Framework?**

The **Model-View-Controller (MVC)** framework is an architectural pattern that separates an application into three main logical components Model, View, and Controller. Hence the abbreviation MVC. Each architecture component is built to handle specific development aspect of an application. MVC separates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC architecture has become popular for designing web applications as well as mobile apps.

چارچوب Model-View-Controller (MVC) یک الگوی معماری است که یک برنامه را به سه مؤلفه اصلی منطقی Model ، View و Controller جدا می کند. از این رو مخفف MVC. هر مؤلفه معماری ساخته شده است تا جنبه توسعه خاص یک برنامه را داشته باشد. MVC منطق تجارت و لایه ارائه را از یکدیگر جدا می کند. به طور سنتی برای رابط های کاربری گرافیکی دسک تاپ (GUI) استفاده می شد. امروزه معماری MVC برای طراحی برنامه های وب و همچنین اپلیکیشن های موبایل محبوبیت پیدا کرده است.

In this tutorial, you will learn more about-

* [History of MVC](https://www.guru99.com/mvc-tutorial.html#2)
* [Features of MVC](https://www.guru99.com/mvc-tutorial.html#3)
* [MVC Architecture](https://www.guru99.com/mvc-tutorial.html#4)
* [MVC Examples](https://www.guru99.com/mvc-tutorial.html#5)
* [Popular MVC web frameworks](https://www.guru99.com/mvc-tutorial.html#6)
* [Advantages of MVC: Key Benefits](https://www.guru99.com/mvc-tutorial.html#7)
* [Disadvantages of using MVC](https://www.guru99.com/mvc-tutorial.html#8)
* [3-tier Architecture vs. MVC Architecture](https://www.guru99.com/mvc-tutorial.html#9)

**History of MVC**

* MVC architecture first discussed in 1979 by Trygve Reenskaug
* MVC model was first introduced in 1987 in the Smalltalk programming language.
* MVC was first time accepted as a general concept, in a 1988 article
* In the recent time, MVC pattern is widely used in modern web applications

**Features of MVC**

* Easy and frictionless testability. Highly testable, extensible and pluggable framework
* Offers full control over your HTML as well as your URLs
* Leverage existing features provided by ASP.NET, JSP, Django, etc.
* Clear separation of logic: Model, View, Controller. Separation of application tasks viz. business logic, Ul logic, and input logic
* URL Routing for SEO Friendly URLs. Powerful URL- mapping for comprehensible and searchable URLs
* Supports for Test Driven Development (TDD)

• قابلیت تست آسان و بدون اصطکاک. چارچوب بسیار قابل آزمایش ، قابل توسعه و قابل نصب  
• کنترل کامل بر HTML و URL های شما را ارائه می دهد  
• از ویژگیهای موجود ارائه شده توسط ASP.NET ، JSP ، Django و غیره استفاده کنید.  
• جداسازی پاک منطق: مدل ، نمایش ، کنترل. جدایی وظایف برنامه یعنی. منطق تجارت ، منطق Ul و منطق ورودی  
• مسیریابی URL برای URL های دوستانه برای SEO. نقشه برداری قدرتمند URL برای URL های قابل فهم و جستجو  
• پشتیبانی از توسعه تست محور (TDD)

**MVC Architecture**

[](https://www.guru99.com/images/1/122118_0445_MVCTutorial1.png)MVC Architecture Diagram

Three important MVC the components are:

* Model: It includes all the data and its related logic
* View: Present data to the user or handles user interaction
* Controller: An interface between Model and View components

سه مؤلفه مهم MVC عبارتند از:  
• مدل: این شامل تمام داده ها و منطق مرتبط با آن است  
• مشاهده: داده ها را به کاربر ارائه داده یا تعامل کاربر را کنترل می کند  
• Controller: واسط بین مؤلفه های Model و View

Let's see each other this component in detail:

**View**

A View is that part of the application that represents the presentation of data.

Views are created by the data collected from the model data. A view requests the model to give information so that it resents the output presentation to the user.

The view also represents the data from chats, diagrams, and table. For example, any customer view will include all the UI components like text boxes, drop downs, etc.

نمای بخشی از برنامه است که نمایش داده ها را نشان می دهد.  
نمایش ها توسط داده های جمع آوری شده از داده های مدل ایجاد می شوند. یک نمای از مدل درخواست می کند تا اطلاعات را به گونه ای ارائه دهد که ارائه خروجی را به کاربر خشمگین کند.  
این نمای همچنین داده های چت ها ، نمودارها و جدول را نشان می دهد. به عنوان مثال ، هر نمای مشتری شامل تمام مؤلفه های UI مانند جعبه متن ، افت فراز و غیره خواهد بود.

**Controller**

The Controller is that part of the application that handles the user interaction. The controller interprets the mouse and keyboard inputs from the user, informing model and the view to change as appropriate.

A Controller send's commands to the model to update its state(E.g., Saving a specific document). The controller also sends commands to its associated view to change the view's presentation (For example scrolling a particular document).

Controller بخشی از برنامه هایی است که تعامل کاربر را کنترل می کند. کنترل کننده ورودی های ماوس و صفحه کلید را از کاربر تفسیر می کند ، مدل و اطلاعات را به منظور اطلاع تغییر می دهد تا در حد مناسب تغییر کند.  
یک کنترلر دستورات را برای به روزرسانی حالت به مدل ارسال می کند (برای مثال ، ذخیره کردن یک سند خاص). کنترل کننده همچنین دستوراتی را برای نمایش نمای خود ارسال می کند (برای مثال پیمایش یک سند خاص).

**Model**

The model component stores data and its related logic. It represents data that is being transferred between controller components or any other related business logic. For example, a Controller object will retrieve the customer info from the database. It manipulates data and send back to the database or use it to render the same data.

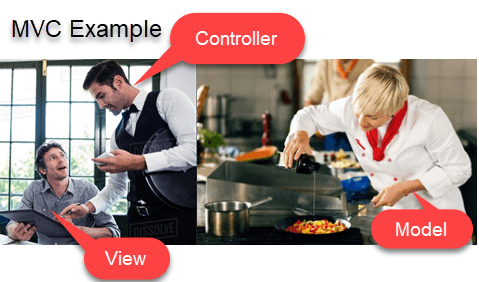
It responds to the request from the views and also responds to instructions from the controller to update itself. It is also the lowest level of the pattern which is responsible for maintaining data.

مؤلفه مدل داده ها و منطق مرتبط با آن را ذخیره می کند. این نمایانگر داده هایی است که بین اجزای کنترل کننده یا هر منطق تجاری مرتبط با آن منتقل می شود. به عنوان مثال ، یک شی Controller اطلاعات مشتری را از پایگاه داده بازیابی می کند. این داده ها را دستکاری کرده و به پایگاه داده می فرستد یا از آن برای ارائه همان داده استفاده می کند.  
  
به درخواست از دیدگاه ها پاسخ می دهد و همچنین به دستورالعمل های کنترل کننده برای بروزرسانی خود پاسخ می دهد. همچنین پایین ترین سطح الگوی مسئول حفظ داده است.

**MVC Examples**

Let's see Model View Controller from daily life:

**Example 1:**

[](https://www.guru99.com/images/1/122118_0445_MVCTutorial2.png)

* Let's assume you go to a restaurant. You will not go to the kitchen and prepare food which you can surely do at your home. Instead, you just go there and wait for the waiter to come on.
* Now the waiter comes to you, and you just order the food. The waiter doesn't know who you are and what you want he just written down the detail of your food order.
* Then, the waiter moves to the kitchen. In the kitchen waiter not prepare your food.
* The cook prepares your food. The waiter is given your order to him along with your table number.
* Cook then prepared food for you. He uses ingredients to cooks the food. Let's assume that your order a vegetable sandwich. Then he needs bread, tomato, potato, capsicum, onion, bit, cheese, etc. which he sources from the refrigerator
* Cook final hand over the food to the waiter. Now it is the job of the waiter to moves this food outside the kitchen.
* Now waiter knows which food you have ordered and how they are served.

In this case,

View= You

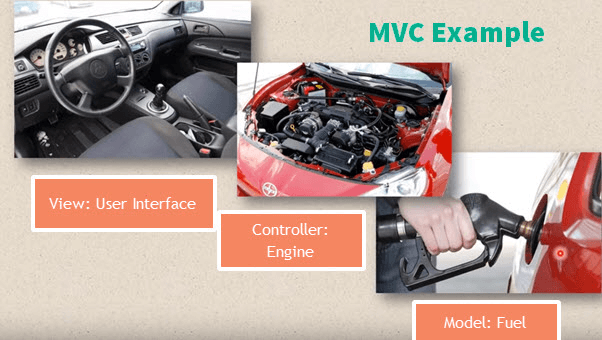
Waiter= Controller

Cook= Model

Refrigerator= Data

Let see one more example,

**Example 2:**

[](https://www.guru99.com/images/1/122118_0445_MVCTutorial3.png)

Car driving mechanism is another example of the MVC model.

* Every car consist of three main parts.
* View= User interface : ( Gear lever, panels, steering wheel, brake, etc.)
* Controller- Mechanism ( Engine)
* Model- Storage ( Petrol or Diseal tank)

Car runs from engine take fuel from storage, but it runs only using mentioned user interface devices.

**Popular MVC web frameworks**

Here, is a list of some popular MVC frameworks.

* [Ruby on Rails](http://www.rubyonrails.org))
* [Django](http://www.djangoproject.org)
* [CakePHP](http://www.cakephp.org)
* [Yii](http://www.yiiframework.com)
* [CherryPy](http://www.cherrypy.org)
* [Spring MVC](http://www.springsource.org)
* [Catalyst](http://www.catalyst.org)
* Rails
* Zend Framework
* CodeIgniter
* Laravel
* Fuel PHP
* Symphony

**Advantages of MVC: Key Benefits**

Here, are major benefits of using MVC architecture.

* Easy code maintenance easy to extend and grow
* MVC Model component can be tested separately from the user
* Easier support for new type of clients
* Development of the various components can be performed parallelly.
* It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
* It only uses a Front Controller pattern which process web application requests through a single controller.
* Offers the best support for test-driven development
* It works well for Web apps which are supported by large teams of web designers and developers.
* Provides clean separation of concerns(SoC).
* Search Engine Optimization (SEO) Friendly.
* All classed and objects are independent of each other so that you can test them separately.
* MVC allows logical grouping of related actions on a controller together.

در اینجا ، مزایای اصلی استفاده از معماری MVC است.  
• نگهداری آسان کد آسان برای گسترش و رشد  
• مؤلفه مدل MVC را می توان جداگانه از کاربر آزمایش کرد  
• پشتیبانی آسانتر از نوع جدید مشتری  
• توسعه مؤلفه های مختلف به صورت موازی قابل انجام است.  
• به شما کمک می کند تا با تقسیم یک برنامه به سه واحد ، از پیچیدگی خودداری کنید. مدل ، نمایش و کنترل کننده  
• این فقط از الگوی Front Controller استفاده می کند که درخواست های برنامه وب را از طریق یک کنترلر واحد پردازش می کند.  
• بهترین پشتیبانی را برای توسعه آزمایش محور ارائه می دهد  
• این برنامه برای برنامه های وب که توسط تیم های بزرگی از طراحان وب و توسعه دهندگان پشتیبانی می شود ، به خوبی کار می کند.  
• جدایی تمیز نگرانی ها (SoC) را فراهم می کند.  
• بهینه سازی موتور جستجو (SEO) دوستانه.  
• همه طبقه بندی شده و اشیاء از یکدیگر مستقل هستند به طوری که می توانید آنها را جداگانه آزمایش کنید.  
• MVC اجازه می دهد تا گروه بندی منطقی اقدامات مرتبط با یک کنترل کننده را با هم گروه بندی کنند.

**Disadvantages of using MVC**

* Difficult to read, change, to unit test, and reuse this model
* The framework navigation can some time complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
* No formal validation support
* Increased complexity and Inefficiency of data
* The difficulty of using MVC with the modern user interface
* There is a need for multiple programmers to conduct parallel programming.
* Knowledge of multiple technologies is required.
* Maintenance of lots of codes in Controller

خواندن ، تغییر ، آزمایش واحد و استفاده مجدد از این مدل دشوار است  
ناوبری چارچوب می تواند برخی اوقات پیچیده باشد زیرا لایه های جدیدی از انتزاع را در اختیار کاربران قرار می دهد که به کاربران نیاز دارد تا با معیارهای تجزیه MVC سازگار شوند.  
• پشتیبانی رسمی برای اعتبارسنجی ندارید  
• افزایش پیچیدگی و ناکارآمدی داده ها  
• مشکل در استفاده از MVC با رابط کاربری مدرن  
• برای انجام برنامه نویسی موازی نیاز به چندین برنامه نویس است.  
• دانش در مورد فن آوری های متعدد مورد نیاز است.  
• نگهداری تعداد زیادی کد در Controller

**3-tier Architecture vs. MVC Architecture**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **3-Tier Architecture** | **MVC Architecture** |
| Communication | This type of architecture pattern never communicates directly with the data layer. | All layers communicate directly using triangle topology. |
| Usage | 3-tier: widely used in web applications where the client, data tiers, and middleware a run on physically separate platforms. | Generally used on applications that run on a single graphical workstation. |

**Summary**

* The MVC is an architectural pattern that separates an application into 1) Model, 2) View and 3) Controller
* Model: It includes all the data and its related logic
* View: Present data to the user or handles user interaction
* Controller: An interface between Model and View components
* MVC architecture first discussed in 1979 by Trygve Reenskaug
* MVC is a highly testable, extensible and pluggable framework
* Some popular MVC frameworks are Rails, Zend Framework, CodeIgniter, Laravel, Fuel PHP, etc.

یک الگوی معماری است که یک برنامه را به 1) مدل ، 2) نمایش و 3) کنترل کننده تقسیم می کند.  
• مدل: این شامل تمام داده ها و منطق مرتبط با آن است  
• مشاهده: داده ها را به کاربر ارائه داده یا تعامل کاربر را کنترل می کند  
• Controller: واسط بین مؤلفه های Model و View  
• معماری MVC برای اولین بار در سال 1979 توسط Trygve Reenskaug مورد بحث و گفتگو قرار گرفت  
• MVC یک چارچوب بسیار قابل آزمایش ، قابل توسعه و قابل نصب است  
• برخی از فریم ورک های محبوب MVC عبارتند از ریل ، Zend Framework ، CodeIgniter ، Laravel ، Fuel PHP و غیره.

https://www.guru99.com/mvc-tutorial.html