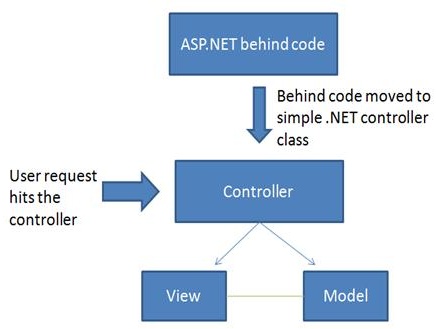
**<http://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx>**

**ASP.NET MVC Overview**

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating Web applications. The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication. The MVC framework is defined in the System.Web.Mvc assembly.

The code-behind is moved to a simple .NET class called Controller. Any user request first comes to the Controller class, the Controller class then invokes the model, and attaches the model to the view for display to the end user.



As this controller class is a simple .NET class we can reuse and also do unit testing easily.

MVC is a standard design pattern that many developers are familiar with. Some types of Web applications will benefit from the MVC framework. Others will continue to use the traditional ASP.NET application pattern that is based on Web Forms and postbacks. Other types of Web applications will combine the two approaches; neither approach excludes the other.

The MVC framework includes the following components:

* **Models.**

Model objects are the parts of the application that implement the logic for the application's data domain. Often, model objects retrieve and store model state in a database. For example, a Product object might retrieve information from a database, operate on it, and then write updated information back to a Products table in a SQL Server database.

In small applications, the model is often a conceptual separation instead of a physical one. For example, if the application only reads a dataset and sends it to the view, the application does not have a physical model layer and associated classes. In that case, the dataset takes on the role of a model object.

* **Views.**

Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data. An example would be an edit view of a Products table that displays text boxes, drop-down lists, and check boxes based on the current state of a Product object.

* **Controllers.**

Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. For example, the controller handles query-string values, and passes these values to the model, which in turn might use these values to query the database.

[**When to Create an MVC Application**](javascript:void(0))

You must consider carefully whether to implement a Web application by using either the ASP.NET MVC framework or the ASP.NET Web Forms model. The MVC framework does not replace the Web Forms model; you can use either framework for Web applications. (If you have existing Web Forms-based applications, these continue to work exactly as they always have.)

Before you decide to use the MVC framework or the Web Forms model for a specific Web site, weigh the advantages of each approach.

**Advantages of an MVC-Based Web Application**

The ASP.NET MVC framework offers the following advantages:

* It makes it easier to manage complexity by dividing an application into the model, the view, and the controller.
* It does not use view state or server-based forms. This makes the MVC framework ideal for developers who want full control over the behavior of an application.
* It uses a Front Controller pattern that processes Web application requests through a single controller. This enables you to design an application that supports a rich routing infrastructure.
* It provides better support for test-driven development (TDD).
* It works well for Web applications that are supported by large teams of developers and for Web designers who need a high degree of control over the application behavior.

**Advantages of a Web Forms-Based Web Application**

The Web Forms-based framework offers the following advantages:

* It supports an event model that preserves state over HTTP, which benefits line-of-business Web application development. The Web Forms-based application provides dozens of events that are supported in hundreds of server controls.
* It uses a Page Controller pattern that adds functionality to individual pages. For more information, see [Page Controller](http://go.microsoft.com/fwlink/?LinkId=141779).
* It uses view state on server-based forms, which can make managing state information easier.
* It works well for small teams of Web developers and designers who want to take advantage of the large number of components available for rapid application development.
* In general, it is less complex for application development, because the components (the Page class, controls, and so on) are tightly integrated and usually require less code than the MVC model.

**Other Features**

The controller gets the first hit and loads the model. Most of the time we would like to pass the model to the view for display purposes. As an ASP.NET developer your choice would be to use session variables, view state, or some other ASP.NET session management object.

The problem with using the ASP.NET session or view state object is the scope. ASP.NET session objects have session scope and view state has page scope. For MVC we would like to see the scope limited to the controller and the view. In other words we would like to maintain data when the hit comes to the controller and reaches the view and after that the scope of the data should expire.

That’s where the new session management technique has been introduced in the ASP.NET MVC framework, i.e., **ViewData**.

**Labs**

[**http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in#Day\_4:-\_JSON,\_JQuery,\_State\_management\_and\_Asynch\_controllers**](http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in#Day_4:-_JSON,_JQuery,_State_management_and_Asynch_controllers)