**MVC Architecture**

It is important to distinguish between the MVC architectural pattern and the ASP.NET MVC Framework.

The MVC pattern is not new—it dates back to 1978 and the Smalltalk project at Xerox PARC—but it has gained enormous popularity today as a pattern for Web applications, for the following reasons:

User interaction with an MVC application follows a natural cycle: the user takes an action, and in response the application changes its data model and delivers an updated view to the user. And then the cycle repeats.

This is a convenient fit for

Web applications delivered as a series of HTTP requests and responses.

Web applications necessitate combining several technologies (databases, HTML, and executable code, for example), usually split into a set of tiers or layers. The patterns that arise from these combinations map naturally onto the concepts in MVC.

**Extensibility**

The MVC Framework is built as a series of independent components that satisfy a .NET **interface** or that are built on **an abstract base class.**

You can easily replace components, such as the routing system, the view engine, and the controller factory, with a different one of your own implementation.

In general, the MVC Framework gives you three options for each component:

Use the *default* implementation of the component as it stands (which should be enough for most applications).

Derive a ***subclass***of the default implementation to tweak its behavior.

*Replace* the component entirely with a new **implementation of the interface** or abstract base class.

**Tight Control over HTML and HTTP**

ASP.NET MVC produces clean, standards-compliant markup. Its built-in HTML helper methods produce standards-compliant output, but there is a more significant philosophical change compared with Web Forms.

Instead of generating out swathes of

HTML over which you have little control, the MVC Framework encourages you to craft simple, elegant markup styled with CSS.

Of course, if you do want to throw in some ready-made widgets for complex UI elements such as date pickers or cascading menus, ASP.NET MVC’s “no special requirements” approach to markup makes it easy to use best-of-breed UI libraries such as

jQuery UI or the Bootstrap CSS library. ASP.NET MVC meshes so well with jQuery, for

example, that Microsoft ships jQuery as a built-in part of the default Visual Studio ASP.NET MVC project template, along with other popular libraries, such as Bootstrap,

Knockout and Modernizr

**Testability**

The MVC architecture gives you a great start in making your application maintainable and testable because you naturally separate

different application concerns into independent pieces. Yet the ASP.NET MVC designers didn’t stop there. To support unit testing,

they took the framework’s component-oriented design and made sure that each separate piece is structured to meet the

requirements of unit testing and mocking tools.

They added Visual Studio wizards to create unit test projects on your behalf, which can be integrated with open source unit test

tools such as NUnit and xUnit as well as the test tools that are included in Visual Studio,

**Powerful Routing System**

The style of URLs has evolved as Web application technology has improved. URLs like this one:

/App\_v2/User/Page.aspx?action=show%20prop&prop\_id=82742

are increasingly rare, replaced with a simpler, cleaner format like this:

/to-rent/chicago/2303-silver-street

There are some good reasons for caring about the structure of URLs.

First, search engines give weight to keywords found in a URL.

A search for “rent in Chicago” is much more likely to turn up the simpler URL.

Second, many Web users are now savvy enough to understand a URL, and appreciate the option of navigating by typing it into their browser’s address bar.

Third, when someone understands the structure of a URL, they are more likely to link to it, share it with a friend, or even read it aloud over the phone.

Fourth, it doesn’t expose the technical details, folder, and file name structure of your application to the public Internet, so you are free to change the underlying implementation without breaking all your incoming links.

**Modern API**

Microsoft’s .NET platform has evolved with each major release, supporting – and even defining – the state-of-the-art aspects of

modern programming.

ASP.NET MVC 5 is built for .NET 4.5.1, so its API can take full advantage of recent language and runtime innovations, including the **await keyword, extension methods, lambda expressions, anonymous and dynamic types, and Language Integrated Query (LINQ).**

Many of the MVC Framework’s API methods and coding patterns follow a cleaner, more

expressive composition than was possible with earlier platforms.

**ASP.NET MVC Is Open Source**

Unlike previous Microsoft Web development platforms, you are free to download the original source code for ASP.NET MVC, and even modify and compile your own version of it. This is invaluable when your debugging trail leads into a system component

and you want to step into its code (and even read the original programmers’ comments).

It is also useful if you are building an advanced component and want to see what development possibilities exist, or how the built-in components actually work