What is ASP.NET Web API?

ASP.NET Web API is a framework that simplifies building HTTP services for broader range of clients (including browsers as well as mobile devices) on top of .NET Framework.
Using ASP.NET Web API we can create non-SOAP based services like plain XML or JSON strings etc. with many other advantages including:

- Create resource-oriented services using the full features of HTTP.
- Exposing services to a variety of clients easily like browsers or mobile devices etc.

What are the advantages of using ASP.NET Web API?

Using ASP.NET Web API has a number of advantages, but core of the advantages are:

- It works the HTTP way using standard HTTP verbs like GET, POST, PUT, DELETE etc for all CRUD operations.
- Complete support for routing.
- Response generated in JSON or XML format using MediaTypeFormatter.
- It has the ability to be hosted in IIS as well as self-host outside of IIS.
- Supports Model binding and Validation.
- Support for OData.
- and more....

For implementation on performing all CRUD operations using ASP.NET Web API, click here.

What new features are introduced in ASP.NET Web API 2.0?

More new features introduced in ASP.NET Web API framework v2.0 are as follows:

- Attribute Routing
- External Authentication
- CORS (Cross-Origin Resource Sharing)
- OWIN (Open Web Interface for .NET) Self Hosting
- IHttpActionResult
- Web API OData

You can follow a good Web API new feature details on Top 5 New Features in ASP.NET Web API 2 here.

WCF Vs ASP.NET Web API?

Actually, **Windows Communication Foundation** is designed to exchange standard SOAP-based messages using variety of transport protocols like HTTP, TCP, NamedPipes or MSMQ etc.
On the other hand, **ASP.NET API** is a framework for building non-SOAP based services over HTTP only.
For more details, please follow here.

Is it true that ASP.NET Web API has replaced WCF?

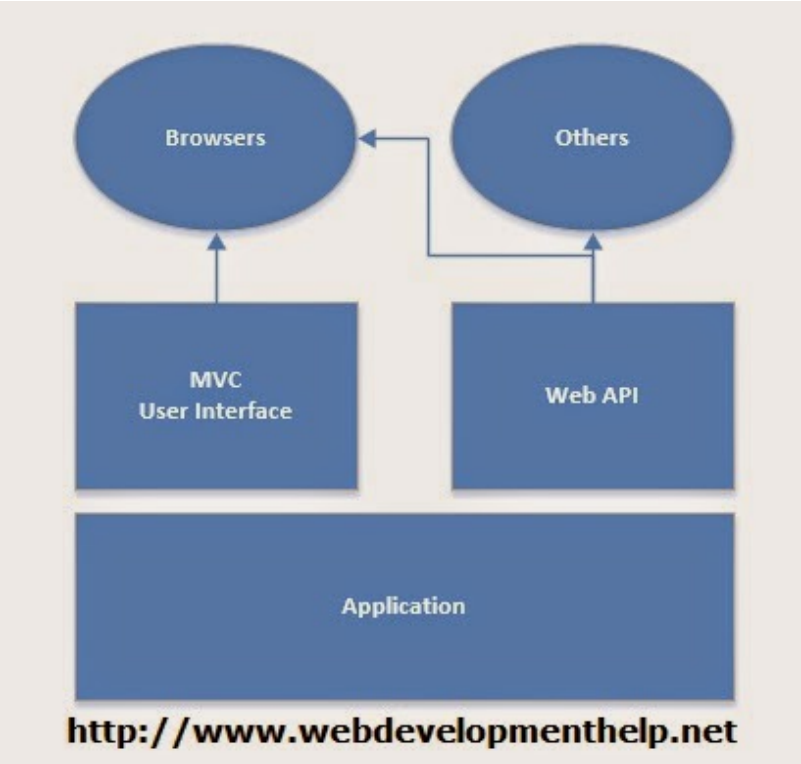
It's a misconception that ASP.NET Web API has replaced WCF. It's another way of building non-SOAP based services, for example, plain XML or JSON string etc.
Yes, it has some added advantages like utilizing full features of HTTP and reaching more clients such as mobile devices etc.
But WCF is still a good choice for following scenarios:

- If we intended to use transport other than HTTP e.g. TCP, UDP or Named Pipes.
- Messag Queuing scenario using MSMQ.
- One-way communication or Duplex communication

A good understanding for WCF(Windows Communication Foundation), please follow WCF Tutorial.

MVC Vs ASP.NET Web API?

As in previous ASP.NET Web API Interview Questions, we discussed that purpose of Web API framework is to generate HTTP services that reaches more clients by generating data in raw format, for example, plain XML or JSON string. So, ASP.NET Web API creates simple HTTP services that renders raw data.
On the other hand, ASP.NET MVC framework is used to develop web applications that generates Views as well as data. ASP.NET MVC facilitates in rendering HTML easy.



For **ASP.NET MVC Interview Questions**, follow the link.

How to return View from ASP.NET Web API method?

(A tricky Interview Question) No, we can't return view from ASP.NET Web API Method. As we discussed in earlier interview question about difference between ASP.NET MVC and Web API that ASP.NET Web API creates HTTP services that renders raw data. Although, it's quite possible in ASP.NET MVC application.

How to restrict access to Web API method to specific HTTP Verb?

Attribute programming plays it's role here. We can easily restrict access to an ASP.NET Web API method to be called using a specific HTTP method. For example, we may required in a scenario to restrict access to a Web API method through HTTP POST only as follows:

```
[HttpPost]
public void UpdateStudent(Student aStudent)
{
    StudentRepository.AddStudent(aStudent);
}
```

Can we use Web API with ASP.NET Web Form?

Yes, ASP.NET Web API is bundled with ASP.NET MVC framework but still it can be used with ASP.NET Web Form.
It can be done in three simple steps as follows:

1. Create a Web API Controller.
2. Add a routing table to Application_Start method of Global.asax.
3. Make a jQuery AJAX Call to Web API method and get data.
   jQuery call to Web API for all CRUD (Create, Retrieve, Update, Delete) operations can be found here.

How we can provide an alias name for ASP.NET Web API action?

We can provide an alias name for ASP.NET Web API action same as in case of ASP.NET MVC by using "ActionName" attribute as follows:

```
[HttpPost]
[ActionName("SaveStudentInfo")]
public void UpdateStudent(Student aStudent)
{
    StudentRepository.AddStudent(aStudent);
}
```

In this ASP.NET Tutorial, we covered most important Interview Questions on ASP.NET Web API framework. Hopefully, it will be helpful for Web API developer Interview but along with these questions, do the practical implementation as much as you can. In **Practical guide to ASP.NET Web API**, you can find a good step by step approach for understanding and implementing ASP.NET Web API services.

ASP.NET Web API 2 has been released with a number of new exciting features. In this web development post, I'll try to discuss new features of it which can be considered the top 5.

1. Attribute Routing

Along with convention-based routing, Web API 2 now supports attribute routing as well.

In case of convention-based routing, we can define multiple route templates. When a request comes, it will be matched against already defined route templates, and forwarded to specific controller action according to matched template.

You can see the following default route template in routing table for Web API:

Config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{Controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

This routing approach has benefits that all routing templates are defined at one common location but for certain URI patterns, it really becomes difficult to support (like nested routing on same controller).

With ASP.NET Web API 2, we can easily support above mentioned URI pattern and others as well. Following shows an example of a URI pattern with attribute routing.

URI Pattern –> books/1/authors

[Route("books/{bookId}/authors")]

public IEnumerable<Author> GetAuthorByBook(int bookId) { ….. }

---

2. CORS – Cross Origin Resource Sharing

Normally, browsers don't allow making cross-domain calls due to same-origin policy and we know that. So, what exactly is CORS (Cross Origin Resource Sharing)?

CORS is a mechanism that allows a web page to make an AJAX call to a domain other than the domain which actually rendered that specific web page. CORS is compliant with W3C standards and now ASP.NET Web API has support for it in version 2.

3. OWIN (Open Web Interface for .NET) self hosting

ASP.NET Web API 2 comes with a new self hosting package i.e. *Microsoft.AspNet.WebApi. OwinSelfHost*.

According to http://owin.org/
"*OWIN defines a standard interface between .NET web servers and web applications. The goal of the OWIN interface is to decouple server and application, encourage the development of simple modules for .NET web development, and, by being an open standard, stimulate the open source ecosystem of .NET web development tools.*"

So, according to above description, OWIN is an ideal option for self hosting a web application in a process other than IIS process.

There are a number of OWIN implementations like Giacomo, Kayak, Firefly etc. available (some may be partial or outdated) but Katana is the recommended one for Microsoft servers and Web API frameworks.

4. IHttpActionResult

Along with the existing two approaches of creating response from controller action, ASP.NET Web API 2 now supports another way of doing the same. *IHttpResponseMessage* is basically an interface which acts as a factory for *HttpResponseMessage*. It's very powerful because it extensify web api. Using this approach we can compose any specific type of response.
Please follow the link to know how to serve HTML with *IHttpActionResult*.

5. Web API OData

The Open Data Protocol (OData) is actually a web protocol for querying and updating data. ASP.NET Web API 2 has added support for *$expand*, *$select*, and *$value* options for OData. By using these options, we can control the representation that is returned from the server.

- ***$expand***: Normally, response doesn't include related entities if we query an OData collection. By using $expand, we can get related entities inline in response.
- ***$select***: It's used if we wanted to include subset of properites in response instead of all.
- ***$value***: It allows to return raw value of the property instead returning in OData format.

What is WebAPI?

Web API is the technology by which you can expose data over HTTP following REST principles

2. With WCF also you can implement REST, So why WebAPI?

WCF was brought in to implement SOA, never the intention was to implement REST.
WebAPI is built from Scratch and the only goal is to create HTTP services using REST.

3. What is the difference between ASP.NET MVC Vs ASP.NET WebAPI?

MVC vs ASP.NET WebA

ASP.NET MVC	ASP.NET WebAPI
used to create web applications that returns	used to create HTTP services, It returns only data.

both view and data	
return data in json format using jsonResult	return data in JSON, XML
Requests are mapped to actions name.	Requests are mapped to the actions based on HTTP verbs

4. WebAPI where is the proxy?

Web API doesn't make it easy for consumers to generate a service client like a SOAP WSDL does.
If you're only ever going to have .NET clients it's not a big deal because they can share the contract obj
implement, but other language clients will need to manually create their client objects if you don't use S

5. What are the Advantages using WebAPI?

OData support (via Queryable attribute)
Content Negotiation
Filters
Model binding and validation
Ability to self host outside of IIS
Link generation to related resources that incorporates routing rules
Full support for routes/routing
Ability to create custom help and test pages using IApiExplorer

6. What are the Http methods used to implement a RESTful API

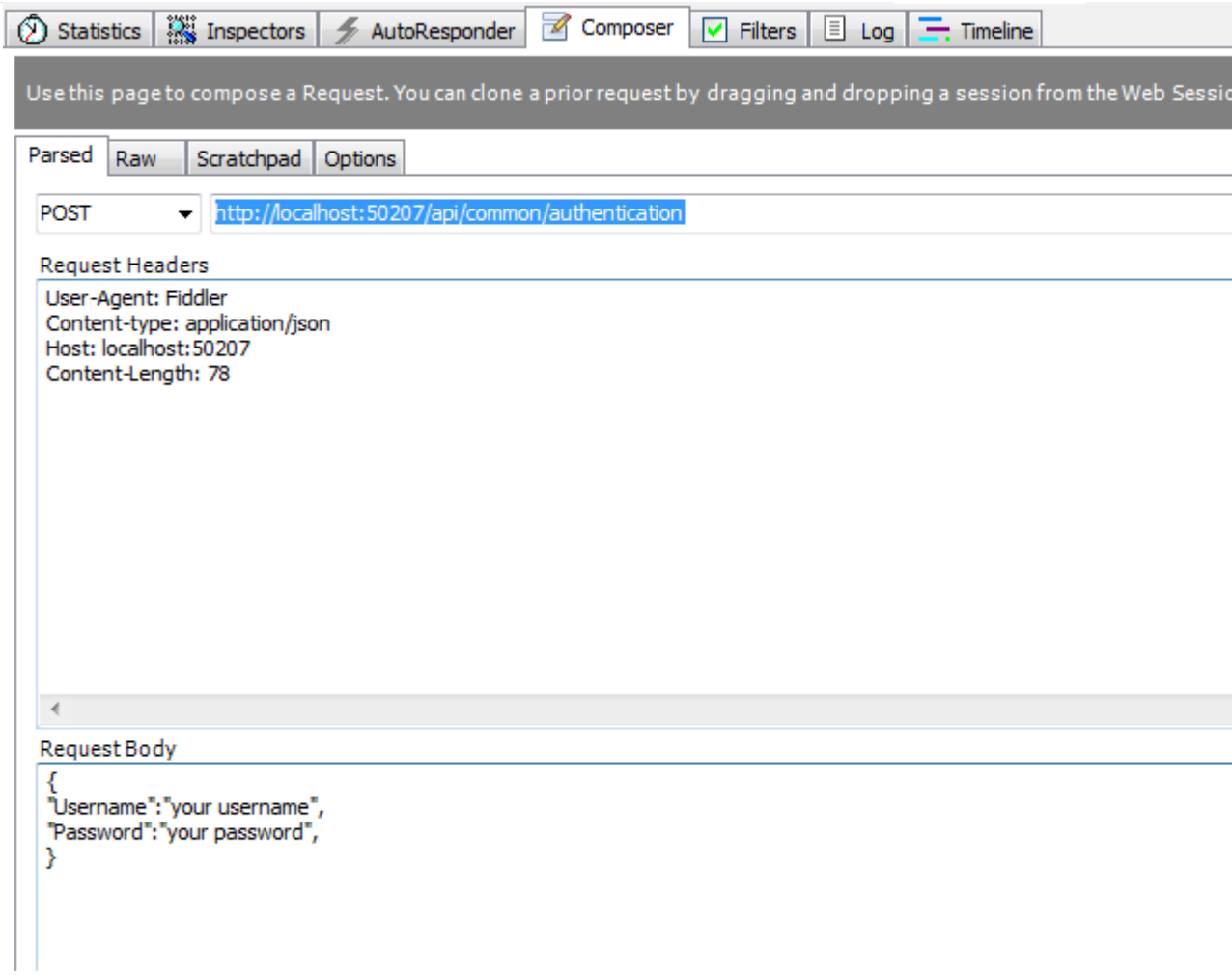
RESTful API HTTP meth

Resource	GET	PUT
Collection URI, such as `http://example.com/resources`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire with another col
Element URI, such as `http://example.com/resources/ item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addre of the collection doesn't exist, **crea**

The PUT and DELETE methods are idempotent methods. The GET method is a safe method (or nullipo
http://en.wikipedia.org/wiki/Representational_State_Transfer#RESTful_web_services

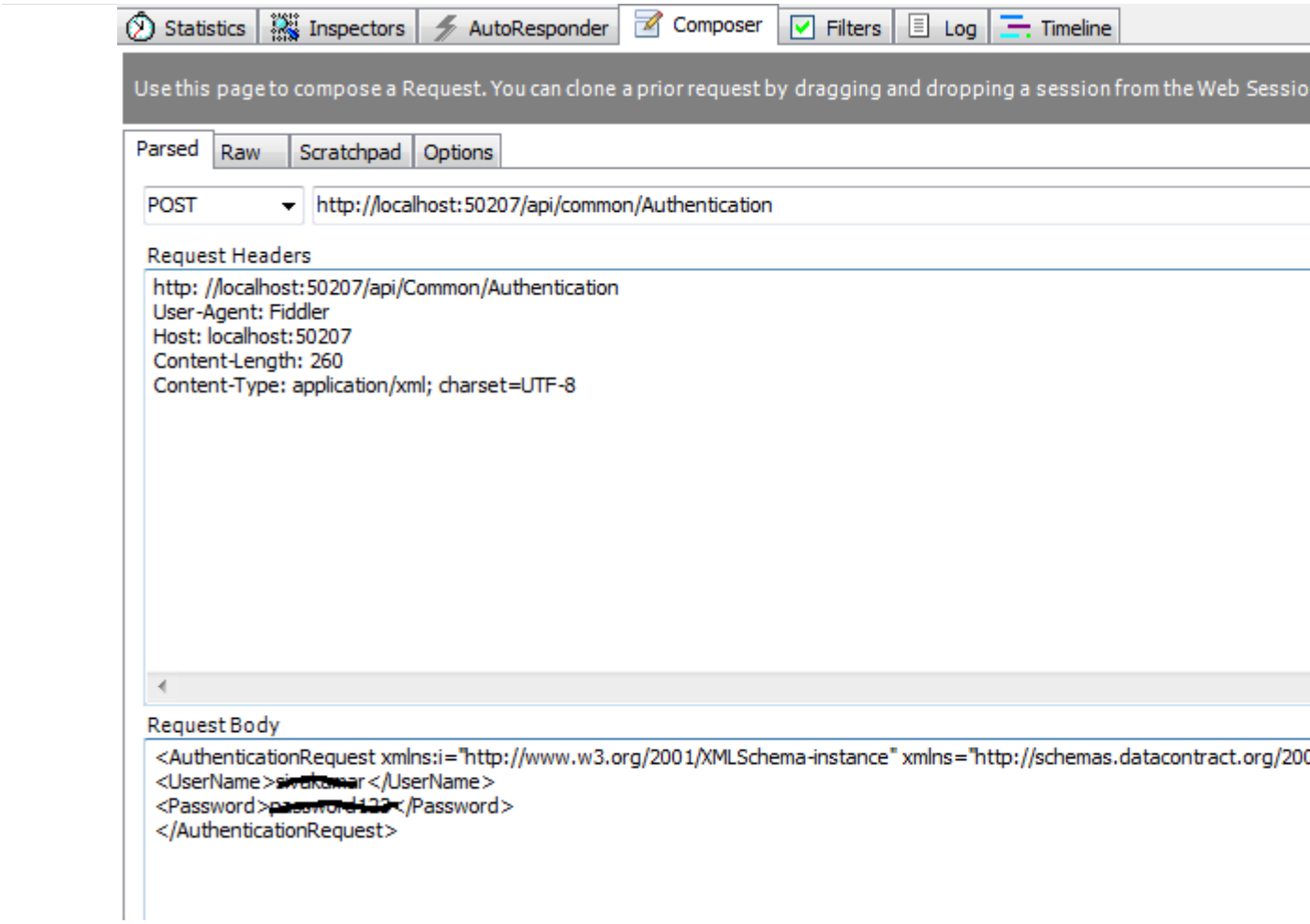
7. How to unit test the WebAPI method using Fiddler - Content-type:application/json?

Fiddler 2 tool -> Compooser Tab - > Enter Request Headers
- > Enter Request Body



8. How to unit test the WebAPI method using Fiddler - Content-type:application/xml?

Fiddler 2 tool -> Compooser Tab - > Enter Request Headers
- > Enter Request Body



ASP.NET WEB API is a framework for building HTTP services that can be consume by a broad range of clients including browsers, mobiles, iphone and tablets. It is very similar to ASP.NET MVC since it contains the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection. But it is not a part of the MVC Framework.

It is a part of the core ASP.NET platform and can be used with MVC and other types of Web applications like ASP.NET WebForms. It can also be used as a stand-alone Web services application.

Web API Features

1. It supports convention-based CRUD Actions since it works with HTTP verbs GET,POST,PUT and DELETE.
2. Responses have an Accept header and HTTP status code.
3. Responses are formatted by Web API's MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.
4. It may accepts and generates the content which may not be object oriented like images, PDF files etc.

5. It has automatic support for OData. Hence by placing the new [Queryable] attribute on a controller method that returns IQueryable, clients can use the method for OData query composition.
6. It can be hosted with in the applicaion or on IIS.
7. It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection that makes it more simple and robust.

Why to choose Web API ?

1. If we need a Web Service and don't need SOAP, then ASP.Net Web API is best choice.
2. It is Used to build simple, non-SOAP-based HTTP Services on top of existing WCF message pipeline.
3. It doesn't have tedious and extensive configuration like WCF REST service.
4. Simple service creation with Web API. With WCF REST Services, service creation is difficult.
5. It is only based on HTTP and easy to define, expose and consume in a REST-ful way.
6. It is light weight architecture and good for devices which have limited bandwidth like smart phones.
7. It is open source.

What do you think?

I hope you have enjoyed the article and would able to have a clear picture about Web API. I would like to have feedback from my blog readers. Your valuable feedback, question, or comments about this article are always welcome.

ASP.NET MVC is an open source framework built on the top of Microsoft .NET Framework to develop web application that enables a clean separation of code. ASP.NET MVC framework is the most customizable and extensible platform shipped by Microsoft.

ASP.NET MVC is a lightweight and highly testable open source framework for building highly scalable and well designed web applications. Here is the list of released version history of ASP.NET MVC Framework with theirs features.

ASP.NET MVC1

1. Released on Mar 13, 2009
2. Runs on .Net 3.5 and with Visual Studio 2008 & Visual Studio 2008 SP1
3. MVC Pattern architecture with WebForm Engine
4. Html Helpers
5. Ajax helpers
6. Routing
7. Unit Testing

ASP.NET MVC2

1. Released on Mar 10, 2010
2. Runs on .Net 3.5, 4.0 and with Visual Studio 2008 & 2010
3. Strongly typed HTML helpers means lambda expression based Html Helpers
4. Templated Helpers
5. Support for Data Annotations Attribute
6. Client-side validation
7. UI helpers with automatic scaffolding & customizable templates
8. Attribute-based model validation on both client and server
9. Overriding the HTTP Method Verb including GET, PUT, POST, and DELETE
10. Areas for partitioning a large applications into modules
11. Asynchronous controllers

ASP.NET MVC3

1. Released on Jan 13, 2011
2. Runs on .Net 4.0 and with Visual Studio 2010
3. The Razor view engine
4. Improved Support for Data Annotations
5. Remote Validation
6. Compare Attribute
7. Sessionless Controller
8. Child Action Output Caching
9. Dependency Resolver
10. Entity Framework Code First support
11. Partial-page output caching
12. ViewBag dynamic property for passing data from controller to view
13. Global Action Filters
14. Better JavaScript support with unobtrusive JavaScript, jQuery Validation, and JSON binding
15. Use of NuGet to deliver software and manage dependencies throughout the platform
16. Good Intellisense support for Razor into Visual Studio

ASP.NET MVC4

1. Released on Aug 15, 2012
2. Runs on .Net 4.0, 4.5 and with Visual Studio 2010SP1 & Visual Studio 2012
3. ASP.NET Web API
4. Enhancements to default project templates
5. Mobile project template using jQuery Mobile
6. Display Modes
7. Task support for Asynchronous Controllers
8. Bundling and minification
9. Support for the Windows Azure SDK

ASP.NET MVC5

1. Released on 17 October 2013

2. Runs on .Net 4.5, 4.5.1 and with Visual Studio 2013
3. One Asp.Net
4. Asp.Net Identity
5. ASP.NET Scaffolding
6. Authentication filters - run prior to authorization filters in the ASP.NET MVC pipeline
7. Bootstrap in the MVC template
8. ASP.NET Web API2

C# pronounced as “See Sharp”. It is object oriented programming language developed by Microsoft, which runs under .NET platform. Its syntax is similar to C++ or Java. Its most recent version C# 5.0 was released on August 15, 2012. It is widely used for developing web application, windows application, smart phone apps and games etc.

Evolution History of C#

C# 1.0

1. Managed Code
2. IDE - Visual Studio 2002, 2003
3. .NET Framework - 1.0, 1.1

C# 2.0

1. Generics
2. Static Classes
3. Partial types
4. Anonymous methods
5. Iterators
6. Nullable types
7. Asymmetric Property and Indexer Accessors
8. Delegate Inference
9. Covariance and Contra-variance
10. IDE - Visual Studio 2005
11. .NET Framework - 2.0

C# 3.0

1. Implicit types (var)
2. Partial Methods
3. Object and collection initializers
4. Auto-Implemented properties
5. Anonymous types
6. Extension methods
7. LINQ
8. Query expressions
9. Lambda expressions
10. Expression trees
11. IDE - Visual Studio 2008

12. .NET Framework - 3.5

C# 4.0

1. Dynamic binding
2. Named arguments
3. Optional Parameters
4. Generic Covariance and Contra-variance
5. COM Interop
6. IDE -Visual Studio 2010
7. .NET Framework - 4.0

C# 5.0

1. Asynchronous methods
2. Caller info attributes
3. IDE - Visual Studio 2012, 2013
4. .NET Framework - 4.5, 4.5.1

What do you think?

I would like to have feedback from my blog readers. Your valuable feedback, question, or comments about this article are always welcome.

AngularJS is an open-source JavaScript framework developed by Google. It helps you to create single-page applications or one-page web applications that only require HTML, CSS, and JavaScript on the client side. It is based on MV-* pattern and allow you to build well structured, easily testable, and maintainable front-end applications.

AngularJS has changed the way to web development. It is not based on jQuery to perform its operations. In spite of using ASP.NET Web form, ASP.NET MVC, PHP, JSP, Ruby on Rails for web development, you can do your complete web development by using most powerful and adaptive JavaScript Framework AngularJS. There is no doubt, JavaScript frameworks like AngularJS, Ember etc. are the future of web development.

Why Angular JS?

There are numerous of reasons to choose AngularJS as you web development framework. Some of them are listed below:

1. It is based on MVC pattern which helps you to properly organize your web apps or web application.
2. It extends HTML by attaching directives to your HTML markup with new attributes or tags and expressions in order to define very powerful templates.
3. It also allows you to create your own directives, crafting reusable components that fill your needs and abstracting away all the DOM manipulation logic.
4. It supports two-way data binding i.e. connects your HTML (views) to your JavaScript objects (models) seamlessly. In this way your model will be immediately reflected into your view without the need for any DOM manipulation or event handling (with jQuery).
5. It encapsulates the behavior of your application in controllers which are instantiated with the help of dependency injection.
6. It supports services that can be injected into your controllers to use some utility code to fulfill your need.

   For example, it provides $http service to communicate with REST service.

7. It helps you to structure and test your JavaScript code very easily.
8. Also, AngularJS is mature community to help you. It has widely support over the internet.

Why this is called AngularJS

Html has angle brackets i.e. <, > and ng sound like Angular. That's why it is called AngularJS

AngularJS Developmet IDE

AngularJS development can be done with the help of following IDEs.

1. Visual Studio 2012, 2013, 2015 or higher
2. Eclipse
3. WebStorm
4. Sublime Text
5. TextMate

Note

1. AngularJS has no dependency on jQuery library. But it can be used with jQuery library.
2. By default AngularJS use jQLite which is the subset of jQuery. If you want to use jQuery then simply load the jQuery library before loading the AngularJS. By doing so, Angular will skip jQLite and will started to use jQuery library.

Browser support

The latest version of AngularJS 1.3 support Safari, Chrome, Firefox, Opera 15+, IE9+ and mobile browsers (Android, Chrome Mobile, iOS Safari, Opera Mobile).

AngularJS 1.3 has dropped support for IE8 but AngularJS 1.2 will continue to support IE8.

What do you think?

I hope you will have better understanding of AngularJS. I would like to have feedback from my blog readers. Your valuable feedback, question, or comments about this article are always welcome.

http://www.dotnetanalyst.com/FAQs/WebAPI.aspx

http://www.dotnetanalyst.com/FAQs/mvc.aspx

http://www.dotnetanalyst.com/FAQs/WebAPIAdvanced.aspx