**What is ASP.NET Web API?**

The term API stands for Application Programming Interface. [**ASP.NET Web API**](https://dotnettutorials.net/lesson/web-api-architecture/) is a framework that makes it easy to build Web APIs, i.e. HTTP based services on top of the .NET Framework. [**ASP.NET Web API**](https://dotnettutorials.net/lesson/web-api-architecture/) is an ideal platform for building Restful services. These services can then be consumed by a broad range of clients like

1. Browsers
2. Mobile applications
3. Desktop applications
4. IOTs

**What is the Rest?**

REST stands for **Representational State Transfer**. This is an architectural pattern used for exchanging data over a distributed environment. In Rest, there is something called Client and Server, and the data can be exchanged between the client and server over a distributed environment. Distributed environment means the client can be on any platform like Java, .NET, PHP, etc. and the server can also be on any platform like Java, .NET, PHP, etc. The REST architectural pattern treats each **service as a resource** and a client can access those resources by using HTTP Protocol methods such as GET, POST, PUT, PATCH, and DELETE.

**What is Not Rest?**

1. It is not a protocol
2. It is not a standard
3. It is not a replacement for SOAP

**What are the RESTful services?**

If you are preparing for ASP.NET Web API Interviews then definitely you have to prepare this Web API Interview Question**.**REST stands for Representational State Transfer. The REST was first introduced in the year 2000 by Roy Fielding as part of his doctoral dissertation. REST is an architectural pattern for exchanging data over a distributed environment. REST architectural pattern treats each service as a resource and a client can access these resources by using HTTP protocol methods such as GET, POST, PUT, PATCH, and DELETE. The REST architectural pattern specifies a set of constraints that a system should be adhered to. Here are the REST constraints.

**1. Client-Server Constraint:**

This is the first constraint. This constraint specifies that a Client sends a request to the server and the server sends a response back to the client. This separation of concerns supports the independent development of both client-side and server-side logic. That means client applications and server applications should be developed separately without any dependency on each other. A client should only know resource URIs and that’s all. Severs and clients may also be replaced and developed independently as long as the interface between them is not altered.

**2. Stateless Constraint:**

The next constraint is the Stateless Constraint. The stateless constraint specifies that the communication between the client and the server must be stateless between requests. That means the server should not be storing anything on the server related to the client. The request from the client should contain all the necessary information so that the server can identify the client and can process that request. This ensures that each request can be treated independently by the server.

**3. Cacheable Constraint:**

In real-time applications, some data provided by the server is not changed that frequently like the list of Countries, the list of States,  the list of cities, and some master data. The Cacheable Constraint says that let the client know how long this data is good for so that the client does not have to come back to the server for that data over and over again.

**4. Uniform Interface Constraint:**

The Uniform Interface Constraint defines an interface between the client and the server. To understand the uniform interface constraint, first, we need to understand what a resource is and the HTTP verbs such as GET, PUT, POST, PATCH, and DELETE. In the context of a RESTFUL Service, resources typically represent data entities. The Product, Employee, Customer, Country, State, City, etc. are all resources. The HTTP verb (GET, PUT, POST, PATCH, and DELETE) that is sent with each request tells the API what to do with the resource. Each resource is identified by a specific URI (Uniform Resource Identifier).

**5. Layered System:**

REST allows us to use a layered system architecture where we deploy the APIs in server A, and store data on server B and authenticate requests in server C. For example, a client cannot ordinarily tell whether it is connected directly to the server or to an intermediary along the way.

**What are the differences between REST and SOAP?**

It is one of the most frequently asked Web API Interview Questions in Web API Interviews.

The difference between REST and SOAP is given below:

1. SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.
2. The SOAP is an XML-based protocol whereas REST is not a protocol rather it is an architectural pattern i.e. resource-based architecture.
3. SOAP has specifications for both stateless and state-full implementation whereas REST is completely stateless.
4. SOAP enforces message format as XML whereas REST does not enforce message format as XML or JSON.
5. The SOAP message consists of an envelope that includes SOAP headers and body to store the actual information we want to send whereas REST uses the HTTP build-in headers (with a variety of media-types) to store the information and uses the HTTP Methods such as GET, POST, PUT, PATCH, and DELETE  to perform CRUD operations.
6. SOAP uses interfaces and named operations (i.e. Service Contract and Operation Contract) to expose the service whereas to expose resources (service) REST uses URI and methods like (GET, PUT, POST, PATCH, and DELETE).
7. SOAP Performance is slow as compared to REST.

**What are the Differences between WCF and Web API? When to choose one over the other?**

The first important point is, we can use both WCF and ASP.NET Web API to develop restful services. In fact, WCF comes first, and then ASP.NET Web API/

WCF (Windows Communication Foundation) is one of the choices available in the .NET Framework for developing both SOAP and REST services. The problem with WCF is that a lot of configuration is required to turn a WCF service (SOAP service) into a REST service. So the more natural choice for developing REST services is [**ASP.NET Web API**](https://dotnettutorials.net/lesson/web-api-architecture/) In fact, ASP.NET Web API is specifically designed for this purpose i.e. for developing Restful Services.

WCF is more suitable for developing services that are **transport/protocol** independent. For example, our requirement is to build one service that can be consumed by 2 different clients – Let’s say, a Java client and a .NET client. Java client wants the transport protocol to be HTTP and message format to be XML for interoperability, whereas the .NET client expects the protocol to be TCP and the message format to be binary for performance. For this scenario, WCF is the right choice. What we need to do here is, create a single WCF service, and then configure 2 endpoints one for each client (i.e. one for the Java client and the other one for the .NET client).

There is nothing wrong to use WCF to develop RESTFUL services. It’s just that it’s a bit more complex and configuration can be a headache. So, if you are stuck with .NET Framework 3.5 or you have an existing SOAP service that you must support as well as you want to add REST to reach more clients, then you need to go for WCF.

If you don’t have the limitation of .NET Framework 3.5 and you want to create a brand new restful service then go with [**ASP.NET Web API**](https://dotnettutorials.net/course/asp-net-web-api/).

**WCF**

1. It is a framework used for developing SOA (service-oriented applications).
2. WCF can only be consumed by clients, which can understand XML. WCF supports protocols like – HTTP, TCP, Named Pipes, etc.

**ASP.NET Web API**

1. It is a framework that helps us to develop HTTP Based services i.e. Restful Services.
2. Web API is an open-source platform.
3. It supports most of the MVC features which keep Web API over WCF.

**When do we need to choose ASP.NET Web API?**

Today, a web-based application is not enough to reach its customers. Now a day, people have become very smart; they are using iPhones, mobiles, tablets, etc. devices in their daily life. These devices are having a lot of apps that actually make their life easy. Actually, we are moving from the web towards the apps world.

So, if we like to expose our service data to the browsers as well as to all these modern device apps in a fast, simple, and secure way, then we should have an API that is compatible with browsers as well as with all these modern devices.

The [**ASP.NET WEB API**](https://dotnettutorials.net/course/asp-net-web-api/) is a great framework for building HTTP services i.e. Restful Services that can be consumed by a broad range of clients including browsers, mobiles, iPhones, and tablets, etc.

**What is the difference between WCF and WEB API and WCF REST and Web Service?**

The .NET framework has a number of technologies that allow us to create HTTP services such as Web Service, WCF, WCF Rest, and now WEB API. Following are the differences between these four:

**Web Service:**

1. The Web Service is based on SOAP and it returns the data in XML format.
2. It supports only the HTTP protocol.
3. Web Service is not open-source but it can be consumed by any client who understands XML.
4. It can only be hosted on IIS.

**WCF:**

1. WCF is also based on SOAP and it also returns the data in the form of XML.
2. Unlike Web service, WCF supports different types of protocols (transport protocol) such as TCP, Named Pipes, HTTP, HTTPS, and MSMQ.
3. The main problem with the WCF service it required lots of configuration which is a headache for a developer.
4. Like Web Service, WCF is also not open source but it can be consumed by any client who understands XML.
5. WCF can be host within the application or on IIS or using Windows service.

**WCF Rest:**

1. To use a WCF service as WCF Rest service we have to enable webHttpBindings.
2. WCF Rest supports the HTTP verbs such as GET and POST by using the [WebGet] and [WebInvoke] attributes respectively.
3. To use other HTTP verbs you have to do some configuration in the IIS so that it will accept the request of that particular verb on the .svc file
4. It supports different data formats such as XML, JSON, and Atom format.

**WEB API:**

1. The Web API Framework is a new framework that is basically used for developing HTTP-based services in an easy and simple way.
2. Unlike WCF Rest Service, it uses full HTTP features such as URIs, request/response headers, caching, versioning, and various data formats.
3. The ASP.NET Web API also supports most of the MVC features such as routing, controllers, actions, filter, model binders, IOC container, dependency injection, unit testing which makes it more simple and robust.
4. WEB API Services can be hosted on IIS or within the application
5. The Responses in Web API Services are formatted by MediaTypeFormatter into JSON, XML, or any custom format you want.

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

The REST uses **fewer data transfers** between **client** and **server** which makes REST ideal for use in mobile apps. The ASP.NET Web API supports HTTP Protocols and thereby it reintroduces the old way of HTTP verbs for communication.

**What are the Differences between WCF Rest and Web API?**

**WCF Rest**

1. We need to use **WebHttpBinding** to be enabled for WCF Rest.
2. For each method, there have to be attributes like – “**WebGet**” and “**WebInvoke**”

**Web API**

1. Unlike WCF Rest, we can use full features of HTTP in Web API.
2. Web API can be hosted in IIS or in an application.

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

Using [**ASP.NET Web API**](https://dotnettutorials.net/course/asp-net-web-api/) has a number of advantages, but the core advantages are as follows:

1. It supports all the HTTP features and allows all standard HTTP verbs such as GET, POST, PUT, PATCH, and DELETE to perform CRUD operations
2. Complete support for Routing which is good for SEO.
3. It supports content negotiation i.e. as per the client request, the server sends the response in that format (if possible). The Response generated in JSON or XML format using **[MediaTypeFormatter](https://dotnettutorials.net/lesson/media-type-formatters-web-api/)**
4. It has the ability to be hosted in IIS as well as self-host outside of IIS
5. Supports Model binding and Validation

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

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

1. Attribute Routing
2. External Authentication (third party authentication)
3. CORS (Cross-Origin Resource Sharing)
4. OWIN (Open Web Interface for .NET) Self Hosting
5. IHttpActionResult

**What are the differences between ASP.NET MVC and ASP.NET Web API?**

This is one of the frequently asked ASP.NET Web API Interview Questions. The following are some of the differences between ASP.NET MVC and ASP.NET Web API

**MVC**

1. ASP.NET MVC Framework basically used to create a web application by following the MVC (Model-View-Controller) design pattern, in which we can build web pages.
2. The action method can return both data and view. It only returns data in JSON format using JsonResult
3. All requests are mapped to the respective action based on the action method names.
4. Content-negotiation not supported.

**Web API**

1. ASP.NET Web API Framework is used to develop Restful services that can be consumed by different clients.
2. Web API returns data in a particular format like JSON, XML, or any other based upon the Accept header in the request. It does not return the view to the client.
3. All requests are mapped to actions using HTTP verbs.
4. Content-negotiation supported.

**Note:** We can mix Web API and MVC controller in a single project to handle advanced AJAX requests which may return data in JSON, XML, or any other format, and building a full-blown HTTP service. Typically, this will be called Web API self-hosting.

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

It’s not true. It’s a misconception that ASP.NET Web API has replaced WCF. ASP.NET Web API is just another way of building non-SOAP-based services, for example, plain XML or JSON string, etc.

Yes, ASP.NET Web API has some added advantages like utilizing the full features of HTTP (HTTP Built-in Header and HTTP Verbs such as GET, POST, PUT, PATCH, and DELETE, HTTP Status codes such as 500, 200, 404, etc.) and reaching more clients such as mobile devices, Tables, IoTs, Browsers, etc. But WCF is still a good choice for the following scenarios:

1. If you want to develop transport protocol-oriented services other than HTTP, such as TCP, UDP, or Named Pipes, etc. If HTTP only then go with Web API.
2. Message Queuing scenario using MSMQ
3. One-way communication or Duplex communication

**Explain Media Formatters in Web API 2**

It is one of the frequently asked ASP.NET Web API Interview Questions. As we know that Web API Framework handles the data in JSON or XML formats (or any other formats) based on the Accept and Content-Type header values.

**But do you know how does the ASP.NET Web API Framework handles these different formats?** The answer is by using **Media Type Formatters**.

The Media-Type Formatters are classes that are responsible for serializing the request/response data. Based on the accept and content-type header values the ASP.NET Web API Framework can understand the request data format and sends the data in the format which the client expects.

Technically **MediaTypeFormatter**is an abstract class from which **[JsonMediaTypeFormatter](https://dotnettutorials.net/lesson/media-type-formatters-web-api/)** and **[XmlMediaTypeFormatter](https://dotnettutorials.net/lesson/media-type-formatters-web-api/)** classes inherit from. JsonMediaTypeFormatter handles JSON data and XmlMediaTypeFormatter handles XML data.

**How to return JSON from Web API Service irrespective of the Accept header value?**

To do so, we need to remove the XmlFormatter from the Register() method of WebApiConfig.cs file which is present inside the App\_Start folder. To remove XmlFormatter include the following piece of code that will completely remove XmlFormatter which forces ASP.NET Web API to always return data in JSON format irrespective of the Accept header value in the client request. Use this technique when you want your service to support only JSON and not XML.

**config.Formatters.Remove(config.Formatters.XmlFormatter);**  
With this change in place inside the Register method of the WebApiConfig class, irrespective of the Accept header value (application/xml or application/json), the Web API service is always going to return the data in JSON format.

**How to return XML data from Web API Service irrespective of the Accept header value?**

To do so, we need to remove the JsonFormatter from the Register() method of WebApiConfig.cs file which is present inside the App\_Start folder. To remove we can include the following piece of code that will completely remove JsonFormatter which forces ASP.NET Web API to always return XML irrespective of the Accept header value in the client request. Use this technique when you want your service to support only XML and not JSON.

**config.Formatters.Remove(config.Formatters.JsonFormatter);**  
With this change in place inside the Register method of the WebApiConfig class, irrespective of the Accept header value (application/xml or application/json), the Web API service is always going to return the data in XML format.

**How to return JSON instead of XML from Web API Service when a request is made from the browser?**

So here is what we want the service to do

1. When a request is issued from the browser, the ASP.NET Web API service should return the data in JSON format instead of XML.
2. When a request is issued from a tool like a fiddler or Postman, the Accept header value should be respected. That means if the Accept header value is set to **application/xml** the service should return **XML** and if the Accept header value is set to **application/json** the service should return the data in JSON.

There are 2 ways to achieve this

**Approach1:**

Include the following line in Register() method of WebApiConfig.cs file in the App\_Start folder. This tells ASP.NET Web API to use JsonFormatter when a request is made for text/html which is the default for most browsers. The problem with this approach is that the Content-Type header of the response is set to text/html which is misleading.

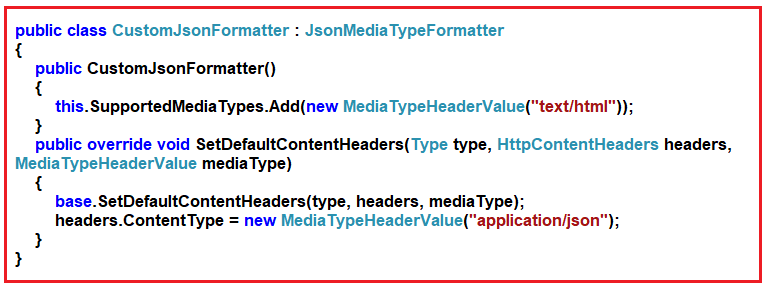
**config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue(“text/html”));**

**Approach2:**

Include the following class in WebApiConfig.cs file in the App\_Start folder.

**Register Formatter:**

Place the following line in Register() method of WebApiConfig.cs file in App\_Start folder



**config.Formatters.Add(new CustomJsonFormatter());**  
With these 2 changes, when a request is issued from the browser you will get JSON formatted data and the Content-Type header of the response is also set to application/json. If you are using tools like a fiddler and if you set Accept header to application/xml you will still get XML formatted data.

**Why do we need Web API to develop RESTful services as we can also develop RESTful services using WCF?**

Yes, it is absolutely possible to develop RESTful services using WCF (Windows Communication Foundation). But there are few reasons why people are moving from WCF to WEB API for developing restful services. Some of them are as follows.

1. The Web API increases the TDD (Test Driven Development) approach in the development of RESTful services.
2. If we want to develop RESTful services using WCF, then we need to do a lot of configuration settings, URI templates, contracts & endpoints. WCF basically used to develop SOA (Service Oriented Architecture) based applications.

**What are the important return types supported in ASP.NET Web API?**

In ASP.NET Web API Application, the controller action methods can return the following:

1. **Void** – It simply returns empty content
2. **HttpResponseMessage** – It will convert the response message to an HTTP message.
3. **IHttpActionResult** – It internally calls the ExecuteAsync method to create an HttpResponseMessage
4. **Other types** – You can also write the serialized return value into the response body. For example, you want to return Excel files.

**Which .NET framework supports Web API?**

The .NET Framework 4.0 and above version supports ASP.NET Web API.

**Which protocol ASP.NET Web API supports?**

The ASP.NET Web App supports the one and only HTTP protocol.

**ASP.NET Web API uses which open-source library for JSON serialization?**

The ASP.NET Web API Framework uses the Json.NET library for JSON serialization.

**By default, Web API sends an HTTP response with which status code for an uncaught exception?**

It will send the response with HTTP Status 500 – Internal Server Error

**How do you construct HtmlResponseMessage in ASP.NET Web API?**

**public** **class** HomeController : ApiController

**{**

**public** HttpResponseMessage Get**()**

**{**

HttpResponseMessage response = Request.CreateResponse**(**HttpStatusCode.OK, "value"**)**;

response.Content = new StringContent**(**"Testing", Encoding.Unicode**)**;

response.Headers.CacheControl = new CacheControlHeaderValue**()**

**{**

MaxAge = TimeSpan.FromMinutes**(**20**)**

**}**;

**return** response;

**}**

**}**

**What is Routing in ASP.NET Web API?**

The ASP.NET Web API Routing module is responsible for mapping the incoming HTTP requests to a particular controller action method. Based on the incoming requests the Web API uses URI and HTTP verbs to select the action method.

**How the Web API Framework handle an incoming HTTP Request?**

When the ASP.NET Web API Framework receives an HTTP request, it tries to match the URI against one of the route templates available in the routing table. If no route template matches the URI, then Web API Framework returns a 404 error to the client who actually makes the request. Once a matching route is found in the Route Table, the Web API Framework then selects the controller and the action to be executed.

**What is SOAP?**

SOAP stands for Simple Object Access Protocol and it is an XML-based protocol. SOAP has specifications for both stateless and state-full implementation. It is also an XML-based messaging protocol for exchanging information among computers. The SOAP message consists of an envelope that includes SOAP headers and body to store the actual information we want to send. It supports different types of protocols such as HTTP, TCP, etc.

**How Can we assign an alias name for ASP.NET Web API Action?**

We can give alias name for Web API action using the “ActionName” attribute as follows:

**[**HttpPost**]**

**[**ActionName**(**"StudentAdd"**)]**

**public** **void** AddStudents**(**Student aStudent**)**

**{**

StudentRepository.AddStudent**(**aStudent**)**;

**}**

**What is Content Negotiation in Web API?**

This is one of the frequently asked ASP.NET Web API Experienced Interview Questions and Answers. One of the standards of the REST service is that the client should have the ability to decide in which format they want the response – whether they want the response in XML or JSON etc. This is called [**Content Negotiation in Web API**](https://dotnettutorials.net/lesson/content-negotiation-web-api/). Web API Content Negotiation means the client and server can negotiate. Always It is not possible to return data in the requested format by the Server. That’s why it is called negotiation, not demand. In such cases, the Web API Server will return the data in the default format.

**What is CORS?**

Before understanding CORS, first, we need to understand the same-origin policy. Browsers allow a web page to make AJAX requests only within the same domain. The Browsers does not allow a web page from making AJAX requests to another domain. This is called the same-origin policy.

CORS is a W3C standard that allows us to get away from the same-origin policy adopted by the browsers that restrict access from making AJAX requests from one domain to another domain. You can enable CORS for your Web API using the respective Web API package (depending on the version of Web API in use).

**What is Web API Attribute Routing?**

The ASP.NET Web API 2 supports a new type of routing called attribute routing. Attribute routing means attributes are used to define routes. The Attribute routing provides more control over the URIs by defining routes directly on the actions and controllers.

**Why do we need Attribute Routing in Web API?**

The convention-based routing makes it hard to support certain URI patterns that are common in RESTful APIs. For example, resources often contain child resources such as Customers have orders, movies have actors, books have authors, etc. It’s natural to create URIs that reflect these relations.

**What is Authentication and Authorization in Web API?**

Once you create a Web API Service, then the most important thing that you need to take care of is security means you need to control access to your Web API Services.

Authentication is the process of identifying the user. For example, one user let’s say James logs in with his username and password, and the server uses his username and password to authenticate James.

Authorization is the process of deciding whether the authenticated user is allowed to perform an action on a specific resource (Web API Resource) or not. For example, James (who is an authenticated user) has the permission to get a resource but does not have the permission to create a resource.

**What is an HTTP Message handler in ASP.NET Web API Application?**

An HTTP Message Handler in Web API is a class that receives an HTTP request and returns an HTTP response. The Message Handler is derived from the abstract HttpMessageHandler class. The HTTP Message handlers are good for cross-cutting concerns (such as authentication and authorization) that operate at the level of HTTP messages rather than controller actions. For example, a Custom HTTP Message handler might do the following things in a Web API Application.

1. **Read or modify the HTTP request headers.**
2. **Add a response header to the HTTP response.**
3. **Validate the requests before they reach the controller (i.e. Authentication and Authorization).**

**Why Web API versioning is required?**

Once you develop and deploy a Web API service then different clients start consuming your Web API services. As you know, day by day the business grows and once the business grows then the requirement may change, and once the requirement change then you may need to change the services as well, but the important thing you need to keep in mind is that you need to do the changes to the services in such a way that it should not break any existing client applications who already consuming your services.

This is the ideal scenario when the Web API versioning plays an important role. You need to keep the existing services as it is so that the existing client applications will not break, they worked as it is, and you need to develop a new version of the Web API service which will start consuming by the new client applications.

**What are the Different options available in Web API to maintain the versioning?**

The different options that are available to maintain versioning are as follows

1. URI’s
2. Query String
3. Version Header
4. Accept Header
5. Media Type

**What are Request Verbs or HTTP Verbs?**

In RESTful service, we can perform all types of CRUD (Create, Read, Update, Delete) Operation. In REST architecture, it is suggested to have a specific Request Verb or HTTP verb on the specific type of the call made to the server. Popular Request Verbs or HTTP Verbs are mentioned below:

1. **HTTP GET:** This HTTP verb is Used to get the resource only.
2. **HTTP POST:** This HTTP verb is Used to create a new resource.
3. **HTTP PUT:** This HTTP verb is Used to update an existing resource.
4. **HTTP PATCH:** This HTTP verb is Used to update an existing resource.
5. **HTTP DELETE:** This HTTP verb is Used to Delete an existing resource.

**Note:** PUT and PATCH are not similar. If you are updating few columns in your database then use PATCG and if you are updating all the data then use PUT.

**What do you mean by Parameter Binding in Web API?**

The Parameter Binding means how the Web API Framework binds the incoming HTTP request data to the parameters of an action method of a Web API controller. The ASP.NET Web API action methods can take one or more parameters of different types. An action method parameter can be either of a complex type or primitive type. The Web API Framework binds the action method parameters either with the URL’s query string or from the request body of the incoming HTTP Request based on the parameter type.

By default, if the parameter type is of the primitive type such as int, bool, double, string, GUID, DateTime, decimal, or any other type that can be converted from the string type then Web API Framework sets the action method parameter value from the query string. And if the action method parameter type is a complex type then Web API Framework tries to get the value from the request body.

But we can change this default behavior of the parameter binding process by using [FromBody] and [FromUri] attributes.

1. **FromBody**: This will force the Web API Framework to get the value from the request body.
2. **FromUri**: This will force the Web API Framework to get the data from the URI (i.e. Route data or Query String)

**What is the use of Authorize Attribute?**

The ASP.NET Web API Framework provided a built-in authorization filter, i.e. Authorize Attribute. This filter checks whether the user is authenticated or not. If not, the user will see 401 Unauthorized HTTP Status Code.

**How to Enable CORS in Web API?**

If we are going to consume the ASP.NET Web API Service using Jquery Ajax from another domain, then we need to enable CORS in the Web API application. Without enabling CORS, it is not possible to access the service from another domain using AJAX call. Enabling CORS in Web API is a two steps process.

**Step1:**Install **Microsoft.AspNet.WebApi.Cors** package.

**Step2:**Once you installed the **Microsoft.AspNet.WebApi.Cors**package then includes the following 2 lines of code in the **Register()**method of **WebApiConfig**class which is present inside the **App\_Start**folder of your project.

**EnableCorsAttribute cors = new EnableCorsAttribute(“\*”, “\*”, “\*”);**  
**config.EnableCors();**

**What is Basic HTTP Authentication?**

This is one of the frequently asked ASP.NET Web API Experienced Interview Questions and Answers. Basic HTTP Authentication is a mechanism, where the user is authenticated through the service in which the client needs to pass the username and password in the HTTP Authorization request headers. The credentials are formatted as the string “username:password: based encoded.

**What is ASP.Net identity?**

ASP.Net identity is the membership management framework given by Microsoft which can be easily integrated with ASP.NET Web API. This helps us in building a secure HTTP service.

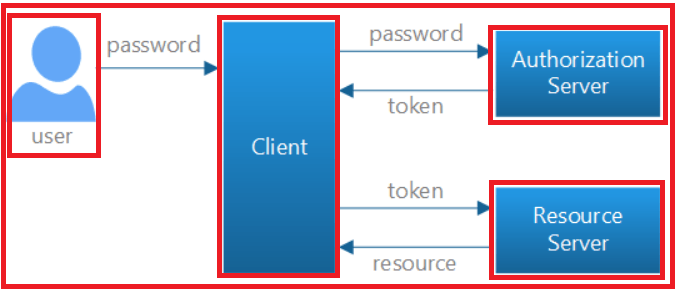
**What is Token Based Authentication in Web API?**

Nowadays, the use of Web API is increasing in a rapid manner. So as a developer we should know how to develop Web APIs. Only developing Web APIs is not enough if there is no security. So, it also very important to implement security for all types of clients (such as Browsers, Mobile Devices, Desktop applications, and IoTs) who are going to use our Web API services.

The most preferred approach nowadays to secure the Web API resources is by authenticating the users in the Web API server by using the **signed token** (which contains enough information to identify a particular user) which needs to be sent to the server by the client in each and every request. This is called the **Token-Based Authentication** approach.

**How does the Token-Based Authentication work?**

This is one of the frequently asked ASP.NET Web API Experienced Interview Questions and Answers. In order to understand how token-based authentication works, please have a look at the following diagram.



**The Token-Based Authentication works as Follows:**

1. The user enters his credentials (i.e. the username and password) into the client (here client means the browser or mobile devices, etc).
2. The client then sends these credentials (i.e. username and password) to the Authorization Server.
3. Then the Authorization Server authenticates the client credentials (i.e. username and password) and generates and returns an access token. This Access Token contains enough information to identify a user and also contains the token expiry time.
4. The client application then includes the **Access Token in the Authorization header** of the HTTP request to access the restricted resources from the Resource Server until the token is expired.

**What is a Refresh Token?**

A **Refresh Token** is a special kind of token that can be used to obtain a new renewed access token that allows access to the protected resources. You can request the new access tokens by using the Refresh Token in Web API until the Refresh Token is blacklisted.

**Why we need Refresh Token in Web API?**

The idea of using the refresh token is to issue a short-lived access token (up to 30 minutes) for the first time and then use the refresh token to obtain a new access token and use that access token to access the protected resources.

So, the user needs to provide the username and password along with the client info (i.e. the client id and client secret) to authenticate himself, and if the information provided by the user is valid, then a response contains a short-lived access token along with a long-lived refresh token gets generated.

The refresh token is not an access token it is just an identifier for the access token**.** Now once the access token is expired, the user can use the refresh token to obtain another short-lived access token and so on.

==========

**What is HMAC Authentication?**

The HMAC stands for Hash-based Message Authentication Code. From the full form of HMAC, we need to understand two things one is Message Authentication Code and the other one is Hash-Based. So HMAC is a mechanism that is used for creating a Message Authentication Code by using a Hash Function.

The most important thing that we need to keep in mind is that while generating the Message Authentication Code using Hash Function we need to use a Shared Secret Key. Moreover, the Shared Secret Key must be shared between the Client and the Server involved in sending and receiving the data.

**Why do we need HMAC Authentication in Web API?**

The main uses of HMAC Authentication in Web API are as follows.

1. **Data integrity:** It means the data sent by the client to the server has not tampered.
2. **Request origination:** The request comes to the server from a trusted client.
3. **Not a replay request:** The request is not captured by an intruder and being replayed.