Web API Standard

1. **What is Web API?**

* A framework which helps you to build/develop HTTP services.
* Supports HTTP protocol only
* RESTful services
* Utilize in any platforms which supports HTTP

1. **Why to use Web API?**

* To create simple, non-SOAP-based HTTP Services
* Based on HTTP and easy to define, expose and consume in a RESTful way.
* Lightweight architecture and ideal for devices that have limited bandwidth like smartphones.
* Cross platform

1. **Can we use WCF services to build RESTful services?**

Yes, we can still develop RESTful services with WCF.

But two main reasons to use Web API instead of WCF services.

* Web API increases TDD (Test Data Driven) approach in the development of RESTful services.
* If we want to develop RESTful services in WCF, you surely need a lot of config settings, URI templates, contracts & endpoints for developing RESTful services using web API.

**Fact:** It’s a not at all true that ASP.NET Web API has replaced WCF. In fact, it is another way of building non-SOAP based services, i.e., plain XML or JSON string.

1. **Advantages of Web API**

* OData
* Filters
* Content Negotiation
* Self-Hosting
* Routing
* Model Bindings

1. **Compare WCF and Web API**

**Windows Communication Foundation** is designed to exchange standard SOAP-based messages using variety of transport protocols like HTTP, TCP or MSMQ, etc. WCF returns only XML based data.

**ASP.NET API** is a framework for building non-SOAP based services over HTTP only. API can return data in multiple forms like, text, html, json, xml.

1. **Return types in Web API**

* **void** - Return empty 204 (No Content)
* **HttpResponseMessage** - It will convert the response to an HTTP message. Return empty 204 (No Content)
* **IHttpActionResult** - Call ExecuteAsync to create an HttpResponseMessage, then convert to an HTTP response message
* **Some other type** - Write the serialized return value into the response body; return 200 (OK).

1. **Default HTTP response for uncaught exceptions**

500 – Internal Server Error

1. **Web API Routing**

Routing is pattern same as in ASP.Net MVC.

All routes are registered in Route Tables.

**For example:**

Routes.MapHttpRoute(

Name: "ExampleWebAPIRoute",

routeTemplate: “api/{controller}/{id}

defaults: new { id = RouteParameter.Optional}

1. **What is SOAP and benefit of using REST in Web API**

**SOAP** is an XML message format used in web service interactions. It allows to send messages over HTTP, but other transport protocols can be used. It is also an XML-based messaging protocol for exchanging information among computers.

**REST** is used to make fewer data transfers between client and server which make it an ideal for using it in mobile apps. Web API also supports HTTP protocol.

1. **Different ways of using Web API in ASP.Net web forms or MVC**

Web API can be used with ASP.NET Web Form/MVC.

It can be performed in three simple steps:

* Create a Web API Controller,
* Add a routing table to Application\_Start method of Global.sax
* Then you need to make a jQuery AJAX Call to Web API method and get data. Or by HttpClient object in C# back-end development.

1. **Web API Actions and how to give alias to different Actions?**

We can give alias name for Web API action same as in case of ASP.NET MVC by using “ActionName” attribute as follows:

[ActionName("SaveStudentInfo")]

1. **Filters in API**

* **Simple Filter –** defines the method that is used in other filters
* **Action Filter –** used to add extra logic before or after action execution
* **Authentication Filter –** Used to force users to authenticate before action
* **Authorization Filter –** Used to restrict access to action methods to specific users
* **Exception Filter –** Handle exceptions in the Web API
* **Override Filter –** to override the behavior of other filters

Exception filter will be executed when exceptions are unhandled and thrown from a controller method. The reason for the exception can be anything. Exception filters will implement “IExceptionFilter” interface.

1. **Ways to register or call filters (explain global registration)**

Apply On:

* Controller
* Action
* Global level

To register exception filter globally using following code-

GlobalConfiguration.Configuration.Filters.Add(new MyTestCustomerStore.NotImplExceptionFilterAttribute());

1. **Can we return view from API and why?**

No, we can’t return a view from ASP.NET Web API Method. Web API creates HTTP services that render raw data. However, it’s also possible in ASP.NET MVC application.

1. **How can we handle exception in web API?**

By global exception filters or several classes are available in Web API to handle errors. They are HttpError, HttpResponseException, and Registering Exception Filters.

1. **New features in Web API 2.0 and 2.2?**

* Attribute Routing
* Cross-Origin Resource Sharing
* External Authentication
* Open Web Interface NET
* HttpActionResult
* Web API OData

1. **HTTP method types, how to restrict actions?**

* **GET –** Retrieve information/data from server
* **POST –** send data to server
* **PUT –** replaces all current data by sending data to server (update)
* **DELETE –** removes all current data from server
* **PATCH –** applies partial data modifications to the resource on server
* **OPTIONS –** Describes the communication options for the target machine
* **TRACE –** performs a message loop-back test along the path to the target machine
* **HEAD –** Same as a Get but transfers the status line and header section only.

With the help of Attributes (like HTTP verbs), It is possible to implement access restrictions in Web API.

1. **How does it help in testing? What is unit test in API?**

API testing involves testing the application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security.

We can perform a Unit test using Web API tools like Fiddler, Postman.

Here, are some setting to be done if you are using

Fiddler –Compose Tab -> Enter Request Headers -> Enter the Request Body and execute

1. **Difference between Web API and ASP.NET MVC?**

The purpose of **Web API** framework is to generate HTTP services that reach 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.

**ASP.NET MVC** framework is used to develop web applications that generate Views (HTML) as well as data. ASP.NET MVC facilitates in rendering HTML easy.

1. **Difference between ApiController and Controller**

Use Controller to render your normal views.

ApiController action only return data that is serialized and sent to the client.

1. **Attribute routing in Web API**

ASP.NET Web API v2.0 and later supports Attribute Routing along with convention-based approach. In convention-based routes, the route templates are already defined as follows:

Config.Routes.MapHttpRoute(

name: "DefaultApi",

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

defaults: new { id = RouteParameter.Optional }

);

So, any incoming request is matched against already defined routeTemplate and further routed to matched controller action. But it’s really hard to support certain URI patterns using conventional routing approach like nested routes on same controller.

For example, authors have books or customers have orders, students have courses etc.

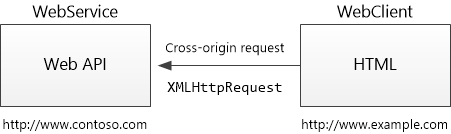
Such patterns can be defined using attribute routing i.e. adding an attribute to controller action as follows:

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

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

1. **What is CORS?**

**Cross Origin Resource Sharing** (CORS) is a W3C standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests while rejecting others.



1. **Explain OWIN**

**OWIN** (Open Web Interface for .NET) is a standard for an interface between .NET Web applications and Web servers. It is a community-owned open-source project.

OWIN aims to decouple the relationship between ASP.NET applications and IIS by defining a standard interface.

**ASP.Net Web API Life Cycle:**

