Web Api

1. What is web api ?

It is a .net technology which is exposed on http for data communication.

2. What is rest ?

Rest is basically a software architecture which helps computer system to communicate on web following http protocol, it provides a window to the client to access the resource as it is.

Characteristics / Features of rest

SEPARATION OF CLIENT AND SERVER (UNIFORM INTERFACE)

The implementation of the client and the implementation of the server can be done independently without each knowing about the other. This means that the code on the client side can be changed at any time without affecting the operation of the server, and the code on the server side can be changed without affecting the operation of the client.

STATELESSNESS

The server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages

COMMUNICATION BETWEEN CLIENT AND SERVER

Clients send requests to retrieve or modify resources, and servers send responses to these requests

CACHEABLE

An effective cache can reduce the number of client-server interactions, which contributes positively to the performance of the system. At least, from a user's point of view.

3. What is restful services ?

All the services which follow the REST Principle are called restful services.

4. When to use WCF ?

1. It can be used when we need SOAP based data access protocol
2. Choose WCF when you want to create a service that should support special scenarios such as Simplex, Duplex Communication, messages queues.
3. Select WCF when you want to produce a service that uses quick transport channels when available, such as Named Pipes, TCP, or perhaps even UDP

5. When to use Web API ?

1. Select Web API when you want to produce resource-related services over HTTP that utilizes the complete features of HTTP (such as response/request headers, URIs, versioning, caching, and numerous content formats).
2. Select Web API when you want to display your service to a wide variety of clients with mobiles, browsers, and tablets.
3. It is light weight architecture and good for devices which have limited bandwidth like smart phones.

MVC ASP.NET	WEB Api
It is used to create web application that returns both view and data.	It used to create http service that returns only data
The request are mapped to action based on action name	The request are mapped to action depending on Http verb
Content negotiation is not supported	Web api supports content negotiation, the best possible response data that can be supported by client.

WEB API	WCF
It operates on the rest principle	It operates on the SOAP principle
It supports only Http	It support various protocols like TCP, HTTP, HTTPS.
It is light weight architecture and good for devices which have limited bandwidth like smart phones.	The main issue with WCF is, its tedious and extensive configuration.

6. Lifecycle of MVC

The life cycle of MVC is the combination of multiple request cycle, right from the application starts to application stops

Request life cycle

1. Whenever application receives the request the routing module is responsible to match the incoming url to routes that we define in application.
2. The MVC framework handles converting of route data to controller that can handle the request
3. After this the component action invoker finds appropriate action method to be executed
4. Once the result is prepared, the next step is result execution. If the result is view type the view engine is called and is responsible to finding and rendering the view
5. If the result is not view the result executer will execute its own and the final output is the response.

7. Routing

It is the technique of routing the incomming request to particular action

Conventional based routing

It uses route template to determine which action of the controller to execute.

Attribute routing

It is the technique of giving customized url name to the action, controller, area to be invoked , which makes the url more readable.

8. Web Api Controller

It is the controller which handles the incomming request and sends appropriate request back to client as response

9. Attributes

It is used to decorate or to provide extra information to any specific action or property. E.g. Route, OutputCache, NonAction, Required, strnLength, Message, Regex, etc.

10. HttpVerb

HttpGet, HttpPost, HttpDelete, HttpPut

GET	POST
It is used to get data from server	It is used to send data to server (to add new object)
Query string is send in the url and is visible in the browser’s address bar	Query string is sent in the Http message body.
Get request can be cached	Post request cannot be cached
It remains in the browsers history	It does not remain in the browser history
It can be bookmarked	It cannot be bookmarked
It should never be used when dealing with sensitive data	It should be used when dealing with sensitive data.
It has length length restriction	It has no length restriction

11. Put vs Post

1. Put is idempotent where as post is not.
2. Making multiple identical requests has the same effect as making a single request. (It is generally used to update existing object)
3. Making multiple request using post will have multiple effect as per the number of request accordingly.

12. Content Negotiation

It is a layer in Web Api which determines what kind of data response will be supported by client. Which is specified in the accept headers

13. What are return types of Web Api response

1. Void
2. HttpResponseMessage
3. IHttpActionResult
4. Other type or Complex type

- Void

When API return type is void, it will return an empty HTTP response.

- HttpResponseMessage

1. Web API converts the return value directly into an HTTP response message.
2. We can set the status code, content as per our requirement.

```
public HttpResponseMessage Get()

{

    HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.OK, "value1");

    return response;

}
```

- IHttpActionResult

IHttpActionResult interface acts like a factory for HttpResponseMessage.

There are some advantages of using the IHttpActionResult over HttpResponseMessage

1. The code is cleaner and easier to read
2. Unit testing for controller action methods become easy
3. It comes with custom built in responses
    a. Ok
    b. NotFound
    c. Exception
    d. Unauthorized
    e. BadRequest
    f. Conflict
    g. Redirect

- Other type or Complex type

1. A Web API can return other or custom complex type
2. Web API serialize value and return it into the response body
3. The response status code 200 OK
4. But in this approach we cannot directly return an error code, still we can throw an HttpResponseException for error codes

- Exception Handling in Web Api

Exception can be handled generically by using Exception filter where we get HttpActionExecutedContext we can return the error response in this which can be shown to the client.

Secondly we can handle error specific to action by using HttpResponseException.

```
Throw new HttpResponseException(response);
```

14. Filters in Web Api

- Authentication Filter

It contains two methods

AuthenticationAsync

Here the credentials of client are authenticated.

ChallengeAsync

1. If the authentication criteria is not met a challenge data is sent to the client by the server in order to generate a different response
2. Challenge-response protocols are one way to fight against replay attacks where an attacker listens to the previous messages and resends them at a later time to get the same credentials as the original message.

- Authorization Filter

- Action Filter

- Exception Filter

15. Type of authentication

1. Forms
2. Passport (e.g. gmail login applicable to youtube, Drive.)
3. Windows
4. token based authentication

16. HttpRequestMessage

The HttpRequestMessage class contains headers, the HTTP verb, and potentially data. This class is commonly used by developers who need additional control over HTTP requests.

It is used for following common things

1. To examine transport information
2. To explicitly set request properties

The instance of HttpRequestMessage is created and various properties is set and sent as parameter to HttpClient

17. HttpResponseMessage

1. Represents an HTTP response message including headers, the status code, and data.
2. A common way to get an HttpResponseMessage is the from the return value for one of the DeleteAsync, GetAsync, PostAsync , PutAsync, or SendRequestAsync methods on the HttpClient object.
3. It contains properties like status code, Content, Headers, Source, Version.

18. Hosting

Web api are generally hosted on IIS or self hosted, the other platform where they can be hosted are azure, amazon.

19. Dependency Injection

It is a software design pattern to develop loosely coupled code. It is one of the efficient way to reduce tight coupling.

Advantages of DI –

1. Reduce class coupling
2. Increase code reusability
3. Improves code maintainability
4. Improves application testing

20. Types of Dependency Injection

- Constructor Injection

```
Public interface IGame
{
   void Play();
}

Public class GameOne : IGame
{
       Public void Play()
       {
              Console.WriteLine("Playing Game One");
       }
 }
```

```
Public class GameTwo : IGame
{
      Public void Play()
        {
              Console.WriteLine("Playing Game Two");
        }
 }

Public class Xbox
{
      IGame iGame;

        public Xbox(IGame igame)
        {
              iGame = igame;
        }

        Public void PlayGame()
        {
              iGame.Play();
        }
 }

Static void Main(string[] args)
{
        Xbox objXbox = newXbox(new GameOne());
        objXbox.PlayGame();
        Xbox objXbox1 = newXbox(new GameTwo());
        objXbox1.PlayGame();
 }
```

- Property Injection

```
Public interface IGame
{
        void Play();
}

Public class GameOne : IGame
{
        Public void Play()
        {
            Console.WriteLine("Playing Game One");
        }
 }


Public class GameTwo : IGame
{
        Public void Play()
        {
            Console.WriteLine("Playing Game Two");
        }
 }
```

```
Public class Xbox
{
        Private IGame iGame;

        Public IGame Game { set { this.iGame = value; } }

        Public void PlayGame()
        {
                iGame.Play();
        }
}

Static void Main(string[] args)
{
        Xbox objXbox = newXbox();
        objXbox.Game = newGameOne();
        objXbox.PlayGame();
        Xbox objXbox1 = newXbox();
        objXbox.Game = newGameTwo();
        objXbox1.PlayGame();
}
```

- Method Injection

```
Public interface IGame
{
        void Play();
}

Public class GameOne : IGame
{
        Public void Play()
        {
              Console.WriteLine("Playing Game One");
        }
 }


Public class GameTwo : IGame
{
        Public void Play()
        {
              Console.WriteLine("Playing Game Two");
        }
 }

Public class Xbox
{
        Private IGame iGame;

        Public void PlayGame(IGame igame)
        {
                this.iGame = igame;
                iGame.Play();
        }
}
```

```
        staticvoid Main(string[] args)
        {
                Xbox objXbox = newXbox();
                objXbox.PlayGame(newGameOne());
                Xbox objXbox1 = newXbox();
                objXbox1.PlayGame(newGameTwo());
        }
```

21. Owin

It is a standard interface between .net web servers and web applications. It is used to decouple server and application which has led to development of simple modules for .net web development and being the open source, to stimulate open source tools development

Features of owin

Loose Coupling
Creating loose coupling has the advantage of using components as per requirement and also possible to swap it without effecting the application.

Flexibility
We have the option of selecting the different components and easily replacing one component with another one.We can mix and match different components as we see fit based on our requirement.

Portable
Owin lets us us connect the existing components with new components which are not restricted to any particaular vendor

An application built upon OWIN specification has the following layers

Host is responsible for starting the process. It also creates the pipeline by fetching the information from our application. It fetches the configuration information from our application to create the pipeline.

Server binds to a port and listens for the requests. Once it receives the request it passes the request in the pipeline that our application has configured.IIS is an example of both Host and Server .

Application framework provides a development model to our application.MVC is a an example of one such framework.

Application is the application that we are developing.

OWIN is just a specification and there could be many different implementations of OWIN.Katana is an example of one such implementation. OWIN specification is open source ,Katana is an OWIN implementation by Microsoft.

22. CORS

Cross origin resource resource sharing is the technique of allowing request from different domain. It cn be achieved by allowing the cors by setting it through web.config or setting it through webapi class while serving resource

```
<httpProtocol>
      <customHeaders>
        <add name="Access-Control-Allow-Origin" value="*" />
        <add name="Access-Control-Allow-Headers" value="Content-Type" />
        <add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE, OPTIONS" />
      </customHeaders>
</httpProtocol>
```