Web API Interview Questions

1. What is ASP.NET Web API?

Answer: ASP.NET Web API is a framework for building HTTP services that can be consumed by various clients, including web browsers and mobile devices. It is designed to work with HTTP and is particularly suitable for RESTful service development.

2. Explain the difference between ASP.NET Web API and WCF (Windows Communication Foundation).

Answer:

- ASP.NET Web API is designed for building HTTP-based services that are lightweight and RESTful, primarily targeting web and mobile clients.
- WCF, on the other hand, is a comprehensive framework for building various types of services, including SOAP-based services, TCP services, and more. It is more complex and versatile but may be overkill for simple HTTP services.

3. What are the key HTTP verbs used in ASP.NET Web API, and what are their meanings?

Answer:

- `GET`: Used to retrieve data from the server.
- `POST`: Used to create a new resource on the server.
- `PUT`: Used to update an existing resource on the server.
- `DELETE`: Used to delete a resource on the server.
- `PATCH`: Used to partially update a resource on the server.

4. Explain the purpose of routing in ASP.NET Web API.

Answer: Routing in ASP.NET Web API determines how incoming HTTP requests should be mapped to specific controller actions. It allows you to define URL patterns and map them to specific actions in your API controllers.

5. What is content negotiation in ASP.NET Web API?

Answer: Content negotiation is the process by which ASP.NET Web API selects the appropriate response format (e.g., JSON or XML) based on the client's requested format (typically specified in the `Accept` header of the HTTP request).

6. What is Model-View-Controller (MVC) in the context of ASP.NET Web API?

Answer: MVC is an architectural pattern that separates the application into three interconnected components: Model (data), View (presentation), and Controller (logic). In the context of ASP.NET Web API, it primarily refers to the Controller component, which handles HTTP requests and generates HTTP responses.

7. Explain attribute routing in ASP.NET Web API.

Answer: Attribute routing is a feature that allows you to define routes using attributes on your controller actions. This provides a more declarative and intuitive way to specify how incoming requests should be mapped to actions.

8. What is Cross-Origin Resource Sharing (CORS), and why is it important in a Web API?

Answer: CORS is a security feature that allows web pages from one domain to request and access resources on a different domain. In a Web API, CORS is essential for enabling cross-domain requests, such as those made by web applications running in a browser while interacting with the API hosted on a different domain.

9. How can you secure an ASP.NET Web API?

Answer: There are several ways to secure an ASP.NET Web API:

- Token-based authentication (e.g., JWT).
- OAuth or OAuth2 for authentication and authorization.
- API keys.
- SSL/TLS for secure communication.
- Role-based or claims-based authorization.

10. Explain the purpose of the `[FromBody]` and `[FromUri]` attributes in Web API parameter binding.

Answer:

- `[FromBody]` is used to bind parameters from the request body. It is typically used with complex objects in POST and PUT requests.
- `[FromUri]` is used to bind parameters from the URI (query string) of the request. It is commonly used for GET requests to retrieve data.

11. What is the role of the `HttpResponseMessage` class in Web API responses?

Answer: `HttpResponseMessage` is a class used to create HTTP responses in Web API. It allows you to set the response status code, headers, and content, making it a flexible way to create customized HTTP responses.

12. What is versioning in Web API, and how can it be implemented?

Answer: Versioning in Web API involves managing different versions of the API to ensure backward compatibility. It can be implemented using various methods, including URI versioning, custom request headers, or using a query parameter to specify the version.

13. Explain the difference between JSON and XML as response formats in Web API.

Answer:

- JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write. It is widely used for Web API responses due to its simplicity and compactness.
- XML (Extensible Markup Language) is another data interchange format that is more verbose and structured. It provides additional metadata and is often used in legacy systems.

14. What is dependency injection, and why is it important in Web API development?

Answer: Dependency injection is a design pattern that promotes loose coupling between components in an application. In Web API development, it is important because it allows you to inject dependencies, such as database context or

services, into controllers, making the code more maintainable, testable, and flexible.

15. Explain the purpose of the `ActionResult` class in ASP.NET Web API.

Answer: The `ActionResult` class in ASP.NET Web API is a base class for all return types from controller actions. It allows you to return various types of responses, such as `Ok`, `NotFound`, `BadRequest`, or custom response types, making it versatile for handling different HTTP status codes and content types.

16. How can you handle errors and exceptions in Web API?

Answer: You can handle errors and exceptions in Web API by using global exception filters, custom exception handling middleware, or by implementing error response models for consistent error messages.

17. What are OData controllers, and how are they different from regular Web API controllers?

Answer: OData controllers in Web API are used to create OData-compliant APIs. They provide features for querying data using the OData protocol, which allows clients to filter, order, and project data efficiently. Regular Web API controllers do not offer OData-specific features out of the box.

18. Explain the concept of content negotiation in Web API, and how does it work?

Answer: Content negotiation is the process of determining the response format (e.g., JSON or XML) based on the client's preferences specified in the `Accept` header of the HTTP request. Web API uses media formatters to serialize the data into the requested format, ensuring that clients receive responses in their preferred format.

19. What is the purpose of attribute-based routing in Web API, and how do you define routes using attributes?

Answer: Attribute-based routing allows you to define routes for your Web API controllers and actions using attributes. You can use the `[Route]` attribute on your controllers or actions to specify the URI template for each route.

20. How can you implement authentication and authorization in ASP.NET Web API?

Answer: You can implement authentication and authorization in Web API using various methods, such as:

- OAuth2 or JWT for authentication.
- Role-based or claims-based authorization.
- Custom authorization filters.
- API keys or API tokens for authentication.

21. Explain the difference between POST and PUT HTTP verbs.

Answer:

1. **POST:** Used to create a new resource on the server. The server generates a unique identifier for the resource, and the response typically includes the newly created resource's URI.
2. **PUT:** Used to update an existing resource on the server. The client specifies the resource's URI, and the request replaces the existing resource with the updated content.

22. When should you use a GET request, and when should you use a POST request?

Answer: Use a GET request when you want to retrieve data from the server without causing any side effects. Use a POST request when you want to create or submit data to the server, and this action may result in side effects, such as creating a new resource on the server.

23. What are the most commonly used HTTP verbs, and what do they represent?

Answer: The most commonly used HTTP verbs are:

- **GET:** Used for retrieving data from the server.
- **POST:** Used for creating new resources on the server.
- **PUT:** Used for updating existing resources on the server.
- **DELETE:** Used for removing resources from the server.
- **PATCH:** Used for making partial updates to resources.
- **HEAD:** Similar to GET but retrieves only the headers of a resource without the actual content.

24. **What is idempotence in the context of HTTP methods, and which HTTP methods are idempotent?**

Answer: Idempotence means that making the same request multiple times has the same effect as making it once. The idempotent HTTP methods are GET, PUT, and DELETE. POST and PATCH are not idempotent because they can result in different outcomes with each request.

25. **How does the OPTIONS HTTP method work, and what is its primary use?**

Answer: The OPTIONS method is used to retrieve information about the communication options for the target resource, such as which HTTP methods are supported or what headers are accepted. It is often used for determining the capabilities of a server and is a part of Cross-Origin Resource Sharing (CORS) implementation.

26. **Explain the purpose of the HEAD HTTP method and how it differs from GET.**

Answer: The HEAD method is similar to GET but only retrieves the headers of a resource, not the actual content. It is used to check the availability and metadata of a resource without downloading the full content, which can be useful for optimizing performance and reducing data transfer.

27. **When is it appropriate to use the DELETE HTTP method, and what is the expected outcome of a successful DELETE request?**

Answer: The DELETE method is used to request the removal of a resource on the server. A successful DELETE request results in the resource being deleted from the server, and subsequent requests to the resource's URI will return a 404 Not Found response.

28. **Explain the role of the PATCH HTTP method and its use cases in resource modification.**

Answer: The PATCH method is used for making partial updates to a resource on the server. It allows clients to send only the changes or modifications to a resource, rather than sending the entire resource. This can be more efficient when working with large or complex resources.

29. How does the PUT method differ from the PATCH method in terms of resource updates?

Answer:

- **PUT:** The PUT method is used to completely replace the existing resource with the new content provided in the request. If the resource already exists, it is overwritten entirely with the new data.
- **PATCH:** The PATCH method is used to make partial updates to a resource. It applies modifications to the resource while preserving the existing data that is not explicitly modified in the request.

30. What is the role of the POST HTTP method in RESTful web services, and how is it different from PUT?

Answer:

- **POST:** The POST method is used to create a new resource on the server. The server typically generates a unique identifier for the resource, and the response includes the new resource's URI. It does not require the client to specify the resource's URI.
- **PUT:** The PUT method is used to update an existing resource on the server. The client must specify the resource's URI in the request, and the existing resource is replaced with the new content provided.