Asp.Net Web API Interview Questions

[https://www.tutorialsteacher.com/webapi/what-is-web-api#google\_vignette](https://www.tutorialsteacher.com/webapi/what-is-web-api%23google_vignette)

**What is Web API?**

Before we understand what is Web API, let's see what is an API (Application Programing Interface).

To put it in simple terms, **API** is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.

Web API as the name suggests, is an API over the web or internet which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc. For example, Twitter's [REST APIs](https://dev.twitter.com/rest/public) provide programmatic access to read and write data using which we can integrate twitter's capabilities into our own application.

## **ASP.NET Web API**

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends **data as a response instead of html view**. It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.

## **What is the DOM?**

The DOM is a W3C (World Wide Web Consortium) standard.

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

The W3C DOM standard is separated into 3 different parts:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

## **What is the HTML DOM?**

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

* The HTML elements as **objects**
* The **properties** of all HTML elements
* The **methods** to access all HTML elements
* The **events** for all HTML elements

In other words:**The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

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

* ASP.NET Web API is a framework for building HTTP services that can be consumed by various clients, including web browsers, mobile devices, and desktop applications.

**What are the key features of ASP.NET Web API?**

* Key features include support for RESTful architecture, content negotiation, model binding, routing, and self-hosting.

**What is REST and how does it relate to ASP.NET Web API?**

* REST (Representational State Transfer) is an architectural style for designing networked applications. ASP.NET Web API is designed to make it easy to build RESTful services that adhere to the principles of REST, such as using HTTP methods (GET, POST, PUT, DELETE) for CRUD operations.

**Explain content negotiation in ASP.NET Web API.**

* Content negotiation is the process by which the client and server decide on the most appropriate representation of a resource. In Web API, it is done through the Accept header in the HTTP request. The server can return data in different formats (e.g., JSON, XML) based on the client's preference.

**What is the role of Web API routing?**

* Web API routing maps HTTP requests to action methods in your controllers. It allows you to define URL patterns and route data to appropriate controller actions.

**What is model binding in Web API?**

* Model binding is the process of mapping HTTP request data to the parameters of a Web API action method. It simplifies the extraction of data from the request, such as query string parameters, form data, or JSON payloads.

**Explain the use of HTTP verbs in Web API.**

* HTTP verbs (GET, POST, PUT, DELETE, etc.) are used to perform CRUD operations on resources in a RESTful Web API. For example, GET is used to retrieve data, POST to create data, PUT to update data, and DELETE to remove data.

**What is the difference between ASP.NET Web API and WCF (Windows Communication Foundation)?**

* Web API is designed for building RESTful services over HTTP, while WCF is a broader communication framework that supports various communication protocols, including HTTP, TCP, and MSMQ.

**How can you enable Cross-Origin Resource Sharing (CORS) in ASP.NET Web API?**

* You can enable CORS by installing the **Microsoft.AspNet.WebApi.Cors** NuGet package and configuring CORS policies in your Web API project. This allows your API to be accessed from different domains.

**What is attribute routing in ASP.NET Web API?**

* Attribute routing allows you to define routing rules directly in your controller actions using attributes. This provides more fine-grained control over routing compared to convention-based routing.

**Explain the concept of dependency injection in Web API.**

* Dependency injection is a design pattern that allows you to inject dependencies (e.g., services or repositories) into your Web API controllers, making them more testable and maintainable. You can use tools like the ASP.NET Core built-in DI container or third-party libraries like Unity or Autofac for dependency injection.

**What is the role of IHttpActionResult in Web API?**

* IHttpActionResult is an interface in Web API that represents an HTTP response. It allows you to return various HTTP status codes and content from your API actions in a flexible and consistent manner.

**What is the purpose of DTOs (Data Transfer Objects) in ASP.NET Web API?**

* DTOs are used to transfer data between the client and the server in a structured format. They help in reducing over-fetching or under-fetching of data, optimizing data transfer in Web API responses.

**Explain the concept of versioning in Web API.**

* API versioning is the practice of managing changes to your API by providing different versions of the same resource. There are different strategies for versioning, including URI versioning, header versioning, and query string versioning.

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

* You can secure a Web API using various methods, such as API keys, OAuth2, JWT (JSON Web Tokens), or basic authentication. Additionally, you can implement authorization through role-based or claims-based security.

**What is token-based authentication in Web API, and how does it work?**

* Token-based authentication involves the use of tokens (e.g., JWTs) for securing Web API endpoints. The client typically obtains a token by authenticating with a token server, and then includes this token in the headers of HTTP requests for authorization.

**Explain the concept of throttling in Web API.**

* Throttling is a mechanism to limit the number of requests a client can make to your Web API within a certain time frame. It is used to prevent abuse or overloading of the API. You can implement throttling using built-in mechanisms or third-party libraries.

**What is the role of middleware in ASP.NET Core Web API?**

* Middleware in ASP.NET Core Web API is used for handling request and response processing. It can perform tasks like authentication, routing, exception handling, and logging. Middleware components can be added to the request/response pipeline.

**How can you handle exceptions in ASP.NET Web API?**

* You can handle exceptions in Web API using various techniques, such as global exception filters, exception handling middleware, and custom error responses. It's important to handle exceptions gracefully to provide meaningful error messages to clients.

**Explain the concept of content negotiation in Web API and how it relates to media formatters.**

* Content negotiation in Web API involves selecting the appropriate media type for the response based on client preferences. Media formatters are responsible for serializing and deserializing data between C# objects and various media types (e.g., JSON, XML). Content negotiation uses media formatters to determine the response format.

**How can you optimize performance in ASP.NET Web API?**

* Performance optimization can be achieved by using techniques such as caching, reducing unnecessary database calls, implementing pagination, asynchronous programming, and optimizing the use of media formatters.

**What is different between REST and SOAP api?**

REST (Representational State Transfer) and SOAP (Simple Object Access Protocol) are two distinct approaches to building web services, and they have several differences in terms of architecture, communication, and usage. Here's a comparison of REST and SOAP APIs:

1. **Architectural Style**:
   * **REST**: REST is an architectural style that is based on a set of constraints for designing networked applications. It relies on the principles of statelessness, resource-based interactions, and using standard HTTP methods (GET, POST, PUT, DELETE) for operations.
   * **SOAP**: SOAP, on the other hand, is a protocol. It is a standardized messaging protocol with a set of rules for structuring messages, including the use of XML and namespaces.
2. **Message Format**:
   * **REST**: REST typically uses lightweight data formats such as JSON and XML for data exchange. It is more flexible in terms of the message format.
   * **SOAP**: SOAP uses XML as its message format. This can make it more rigid and verbose compared to REST.
3. **Transport Protocol**:
   * **REST**: REST primarily uses HTTP as the transport protocol, making it suitable for web-based applications. It can also use other protocols but is most commonly associated with HTTP.
   * **SOAP**: SOAP can be transported over various protocols, including HTTP, SMTP, TCP, and more. It is not tied to a specific transport protocol.
4. **Security**:
   * **REST**: REST relies on the underlying transport layer security (e.g., HTTPS) for securing communication. Security features are often implemented independently in REST services.
   * **SOAP**: SOAP has built-in security features and can use various security standards like WS-Security to ensure message-level security.
5. **State**:
   * **REST**: REST is designed to be stateless, meaning each request from a client to the server must contain all the information needed to understand and process the request.
   * **SOAP**: SOAP can be used in both stateless and stateful communication scenarios. It provides more options for managing session state.
6. **Complexity**:
   * **REST**: REST is simpler and easier to understand. It uses basic HTTP methods and status codes for communication.
   * **SOAP**: SOAP is more complex and requires adherence to a strict specification. It has a higher learning curve due to its extensive features and standards.
7. **Performance**:
   * **REST**: REST is often considered more efficient for simple, read-heavy operations due to its lightweight message format and statelessness.
   * **SOAP**: SOAP can be less efficient due to its XML message format and additional processing required for parsing XML.
8. **Usage**:
   * **REST**: REST is commonly used for web and mobile applications that require quick and straightforward communication. It's widely adopted for public APIs and web services.
   * **SOAP**: SOAP is often used in enterprise-level applications and scenarios where a high level of security and reliability is needed, such as in banking and healthcare systems.

In summary, REST and SOAP have different architectural styles, message formats, transport protocols, and use cases. The choice between them depends on the specific requirements and constraints of the application you are developing or the API you are interacting with.

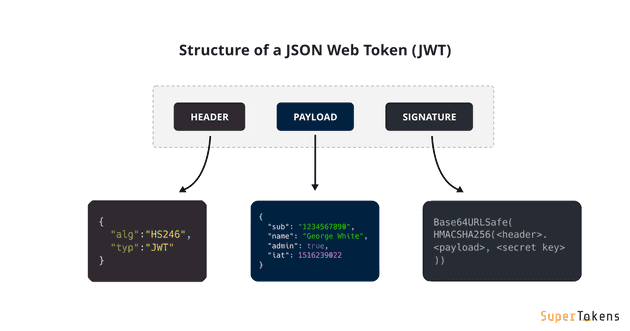
**What is JWT Token?**

JWT stands for "JSON Web Token." It is a compact, self-contained means of representing information between two parties in a way that can be verified and trusted. JWTs are often used for authentication and authorization in web applications and APIs.

## **Structure of a JWT**

A JWT contains three parts:

* **Header**: Consists of two parts:
  + The signing algorithm that’s being used.
  + The type of token, which, in this case, is mostly “JWT”.
* **Payload**: The payload contains the claims or the JSON object.
* **Signature**: A string that is generated via a cryptographic algorithm that can be used to verify the integrity of the JSON payload.

[](https://supertokens.com/static/b0172cabbcd583dd4ed222bdb83fc51a/9af93/jwt-structure.png)