FastAPI Interview Questions and Answers

Here are 20 commonly asked FastAPI interview questions and answers to prepare you for your interview:

1. What is FastAPI?

FastAPI is a Python web framework that is designed to be fast and easy to use. It is built on top of the Starlette web framework and utilizes the Pydantic library for data validation. FastAPI is also fully compatible with the OpenAPI standard, which makes it easy to document and test your API endpoints.

2. Can you explain the architecture of FastAPI?

FastAPI is built on top of Starlette, which is a lightweight ASGI framework. FastAPI then adds additional features on top of Starlette, such as automatic data validation, serialization, and documentation.

3. How do you define a route in FastAPI?

In FastAPI, routes are defined using decorators. For example, to define a route that returns a list of users, you would use the @get decorator:

```
@get("/users")
def list_users():
return [{"username": "jane"}, {"username": "joe"}]
```

4. What are some advantages of using FastAPI over Flask?

FastAPI is built on top of Starlette, which is a lightweight ASGI framework. This makes it ideal for building high-performance web applications. Additionally, FastAPI comes with built-in support for data validation, authentication, and documentation.

5. Can you give me an example of how to use HTTP methods with routes in FastAPI?

You can use HTTP methods with routes in FastAPI by specifying the methods argument with a list of methods when you add a route. For example, if you want to add a route that can be accessed with the GET and POST methods, you would do the following:

```
@app.get("/my-route", methods=["GET", "POST"])
def my_route():
return "Hello, world!"
```

6. What's the difference between path, query, and body parameters?

Path parameters are used to specify the resource that you are requesting, and are typically used to identify a specific object.

Query parameters are used to specify the parameters of a query, and are typically used to filter a result set.

Body parameters are used to specify the body of a request, and are typically used to provide data to be used in the request.

7. What is UTF8JSONResponse?

UTF8JSONResponse is a class used in FastAPI to return responses that are encoded in UTF-8 and formatted as JSON. This is useful for APIs that need to support international characters, as UTF-8 is a widely used encoding that can represent most languages.

8. What is uvicorn? Why should it be used with FastAPI?

Uvicorn is a web server that is specifically designed for use with the FastAPI web framework. It is built on top of the asyncio library and provides a very fast and efficient way to serve web applications.

9. Can I upload files in FastAPI? If yes, then how?

Yes, you can upload files in FastAPI. There are a few different ways to do it, but the most common way is to use the FileUpload class. This class allows you to specify the maximum file size, the allowed file types, and the storage location for the uploaded files.

10. Is there any easy way to add documentation for APIs when using FastAPI?

Yes, FastAPI includes a built-in documentation system that makes it easy to document your APIs. Simply add a few lines of code to your FastAPI application and the documentation will be generated automatically.

11. What are OpenAPI specifications and why are they important?

OpenAPI specifications are important because they provide a standard, language-agnostic way of describing REST APIs. This allows developers to more easily understand how an API works, and also allows for tools to be built that can automatically generate code or documentation based on the OpenAPI specification.

12. Do clients need to support JSON schema or OpenAPI specifications to interact with your API?

No, clients do not need to support JSON schema or OpenAPI specifications to interact with your API. However, doing so would certainly make things easier, as it would allow them to automatically generate code to interact with your API.

13. Can you tell me what automatic validation is? How does it work?

Automatic validation is a feature of FastAPI that automatically validates incoming requests against a defined schema. This ensures that only valid requests are processed, and helps to prevent malicious input from causing errors or unexpected behavior.

14. What is dependency injection?

Dependency injection is a technique for decoupling the dependencies of a software component from the component itself. This allows the component to be more easily reused and testable. In the context of FastAPI, dependency injection is used to inject the dependencies of a route handler into the handler itself. This allows the route handler to be more easily reused and testable.

15. What are asynchronous requests in context with FastAPI?

Asynchronous requests are those that are executed in a separate thread from the main thread of execution. This allows the main thread to continue processing other requests while the asynchronous request is being processed. FastAPI provides support for asynchronous requests, which can improve the performance of your application.

16. What is UvicornWorker?

UvicornWorker is a child process created by Uvicorn when it starts up. It is responsible for handling requests from clients.

17. What are some common error codes returned by FastAPI?

Some common error codes that may be returned by FastAPI include:

-400: Bad Request
-401: Unauthorized
-403: Forbidden
-404: Not Found
-405: Method Not Allowed
-500: Internal Server Error

18. What is the best way to test endpoints that have been created using FastAPI?

There are a few different ways to test endpoints created with FastAPI. One way is to use the built-in test client that is provided. Another way is to use a tool like Postman.

19. Are there any performance differences between using standard Python dicts and typing classes?

There can be some performance differences between using standard Python dicts and typing classes, depending on how they are used. For example, if you are using a standard dict as a data structure, then accessing items by key will be faster than using a typing class. However, if you are using a typing class to define the structure of your data, then the compiler can optimize your code better, leading to better performance.

20. Is it possible to return multiple types of response from one endpoint using FastAPI?

Yes, it is possible to return multiple types of response from one endpoint using FastAPI. This can be accomplished by using the @Produces decorator to specify the types of response that the endpoint can return.