## **Features of FastAPI**

Here are some of the key features that makes FastAPI one of the best Python web frameworks to date:

* **High Performance**: FastAPI, as compared to other python web frameworks, has better performance. The fact that FastAPI is a speed-oriented framework, it ranks high. Leveraging Starlette features, building APIs and production-ready codes, everything boils up to higher speed and improved performance.
* **Fast to code**: With production-ready code, developers need not create anything from scratch. This expedites the process by about 200% to 300%. To elaborate, FastAPI uses Starlette to run and offers a lot more features helpful when building APIs for serialisation and/or data validation. This reduces the time required to code as the framework has everything prepared in advance.
* **Minimal bugs**: Since the framework comes with an auto-completion feature, most of the code is automatically fed into the piece. This reduces all possibilities of error, enhancing the quality of output.
* **Easy-to-understand**: FastAPI isn't anything but Modern Python. There is no new syntax and is basically an advanced version of Python 3.6. Meaning that anyone with a significant understanding of the language can work on the framework.
* **Simple Documentation**: FastAPI automatically generates OpenAPI documentation with no effort from the developer. The same can be located under the application's/doc path. The documentation has detailed information on the endpoints of API, returned codes, response and parameters, etc.
* **Editor-support**: The entire framework is designed keeping in mind autocompletion. It is both simple, and easy to use, empowering developers with production-ready code. The automatic interactive documentation further allows testing the code directly from the browser.
* **Based on open standards**: One of the biggest advantages of having FastAPI as a [Python](https://www.hackertrail.com/interview/python-interview-questions-top-30-asked/" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank) web framework is its open standards.  That is, the framework is fully compatible with [OpenAPI](https://github.com/OAI/OpenAPI-Specification" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank) (previously known as Swagger), facilitating the creation of APIs (in addition to parameter declaration, body requests, etc) and [JSON Schema](https://json-schema.org/" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank) accounting for automated documentation of the data model.

## **Comparison of FastAPI with Django and Flask**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ****Django**** | ****FLASK**** | ****FastAPI**** |
| ****Community**** | Thanks to its popularity, [Django](https://www.hackertrail.com/talent/backend/top-django-interview-questions/" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank) has a huge community of developers. | Next to [Django](https://www.hackertrail.com/talent/backend/top-django-interview-questions/" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank), Flask comes second in terms of community support. | FastAPI is relatively a new framework and hence, has limited community support. |
| ****Packages**** | As a full-stack web development framework, Django has tons of packages that help in the development of compelling web applications. | Flask has a lesser number of packages, considering that it is primarily used for the creation of minimalistic applications. | FastAPI emphasizes building websites faster, and so doesn’t incorporate many packages. |
| ****Performance**** | A mega framework with tons of packages and libraries, Django isn't the best in terms of performance. | Flask being a micro web framework offers better performance as compared to Django. | FastAPI is one of the fastest web frameworks known to the development world. Besides, the native async support adds to the efficiency of the framework. |
| ****Async Support**** | The initial version of Django lacked asynchronous support. However, the latest release seems to extend support to asynchronous calls, but with limited latency. | Flask, like Django, wasn't designed to process requests asynchronously. Adhering to the need, recent usage has been implementing asyncio to support async features. | FastAPI is one of the very few [Python](https://www.hackertrail.com/interview/python-interview-questions-top-30-asked/" \t "https://www.hackertrail.com/talent/backend/fastapi-all-you-need-to-know-about-this-trending-python-web-framework/_blank) frameworks that provide native async support. This is one of the reasons why the development time is least. |
| ****Learning Curve**** | A framework that includes dozens of packages, and libraries, Django is a bit complicated to understand. | A pretty straightforward language, learning the framework is easy. | FastAPI is the simplest of all allowing developers to get started like a pro. |

## **When should you choose FastAPI over Flask/Django**

Even though FastAPI is a recently-created Python web framework, it seems to have gathered tremendous support from developers across the globe. Primary reasons include:

1. **Native async support**: Since the FastAPI web framework is used for the creation of ML instances and applications, the native async support eliminates inference latency.
2. **Improved latency**: Being a high-performance framework, the overall latency identified for the framework is better as compared to Flask and Django.
3. **Production-ready codes**: Coming with auto validation, and short defaults, with FastAPI web framework, developers have the ease to build web applications without modifying the code.
4. **High Performance**: Compatible with Starlette and Pydantic, developers have access to their core features. Needless to state that Pydantic is one of the fastest libraries and hence, the overall performance improves, making FastAPI the choice for web development.
5. **Simpler learning curve**: As a minimalistic web framework, learning and understanding the same is pretty easy.

### **1) Explain what is REST and RESTFUL?**

REST represents REpresentational State Transfer; it is a relatively new aspect of writing web API.

RESTFUL is referred for web services written by applying REST architectural concept are called RESTful services, it focuses on system resources and how state of resource should be transported over HTTP protocol to different clients written in different language. In RESTFUL web service HTTP methods like GET, POST, PUT and DELETE can be used to perform CRUD operations.

### **2) Explain the architectural style for creating web API?**

The architectural style for creating web api are

* HTTP for client server communication
* XML/JSON as formatting language
* Simple URI as the address for the services
* Stateless communication

### **3) Mention what tools are required to test your web API?**

SOAPUI tool for SOAP WS and Firefox “poster” plugin for [RESTFUL services](https://www.guru99.com/restful-web-services.html).

### **4) Mention what are the HTTP methods supported by REST?**

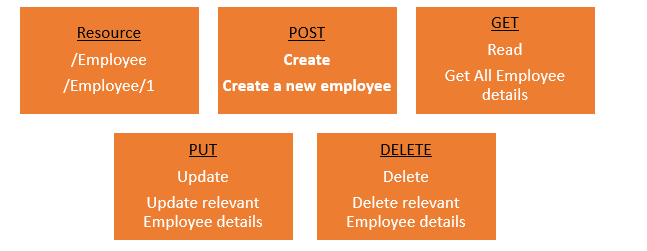
HTTP methods supported by REST are:

* ****GET:****It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.
* ****POST:****It submits information to the service for processing; it should typically return the modified or new resource
* ****PUT:****At the request URL it update the resource
* ****DELETE:****At the request URL it removes the resource
* ****OPTIONS:****It indicates which techniques are supported
* ****HEAD:****About the request URL it returns meta information

Let’s take a look from a perspective of just a single record. Let’s say there was an employee record with the employee number of 1.

The following actions would have their respective meanings.

1. ****POST****– This would not be applicable since we are fetching data of employee 1 which is already created.
2. ****GET****– This would be used to get the details of the employee with Employee no as 1 using the RESTful web service
3. ****PUT****– This would be used to update the details of the employee with Employee no as 1 using the RESTful web service
4. ****DELETE**** – This is used to delete the details of the employee with Employee no as 1



### **5) Mention whether you can use GET request instead of PUT to create a resource?**

No, you are not supposed to use PUT for GET. GET operations should only have view rights, while PUT resource is used for updating a data.

### **6) Mention what are resources in a REST architecture?**

Resources are identified by logical URLs; it is the key element of a RESTful design. Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

### **7) Mention what is the difference between AJAX and REST?**

| **AJAX** | **REST** |
| --- | --- |
| In Ajax, the request are sent to the server by using XMLHttpRequest objects. The response is used by the JavaScript code to dynamically alter the current page. | REST have a URL structure and a request/response pattern the revolve around the use of resources. |
| Ajax is a set of technology; it is a technique of dynamically updating parts of UI without having to reload the page. | REST is a type of software architecture and a method for users to request data or information from servers. |
| Ajax eliminates the interaction between the customer and server asynchronously. | REST requires the interaction between the customer and server. |

### **8) Mention some key characteristics of REST?**

Some key characteristics of REST includes

* REST is stateless, therefore the SERVER has no state (or session data)
* With a well-applied REST API, the server could be restarted between two calls as every data is passed to the server
* Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

### **9) Mention what are the different application integration styles?**

The different integration styles include

* Shared database
* Batch file transfer
* Invoking remote procedure (RPC)
* Swapping asynchronous messages over a message oriented middle-ware (MOM)

### **10) Explain how JAXB related to RESTful web API?**

JAXB stands for java arch for XML binding.

### **11) Mention what is the difference between PUT and POST?**

“PUT” puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one

POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource

PUT is idempotent meaning, invoking it any number of times will not have an impact on resources.

However, POST is not idempotent, meaning if you invoke POST multiple times it keeps creating more resources

### **12) Mention which markup language can be used in restful web api?**

JSON and XML are the two markup language that can be used in restful web api

### **13) Mention what is the difference between RPC or document style web services? How you determine to which one to choose?**

In document style web services, we can transport an XML message as part of SOAP request which is not possible in RPC style web service. Document style web service is most appropriate in some application where XML message behaves as document and content of that document can alter and intention of web service does not rely on the content of XML message.

### **14) Mention what is JAX-WS and JAX-RS?**

Both JAX-WS and JAX-RS are libraries (APIs) for doing communication in various ways in Java. JAX-WS is a library that can be used to do SOAP communication in JAVA, and JAX-RS lets you do the REST communication in JAVA.

### **15) List out the tools or API for developing or testing web api?**

Testing tools for web services for REST APIs includes

* Spring REST web service using MVC
* Jersey API
* CXF
* Axis
* Restlet,

### **16) Mention what is the difference between SOAP and REST?**

| **SOAP** | **REST** |
| --- | --- |
| SOAP is a protocol through which two computer communicates by sharing XML document. | Rest is a service architecture and design for network-based software architectures. |
| SOAP permits only XML | REST supports many different data formats |
| SOAP based reads cannot be cached | REST reads can be cached |
| SOAP is like custom desktop application, closely connected to the server | A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it |
| SOAP is slower than REST | REST is faster than SOAP |
| It runs on HTTP but envelopes the message | It uses the HTTP headers to hold meta information |

* [GOOGLE ALGORITHM CHANGES](https://www.bitarray.io/google-algorithm-tracker/)

[HOME](https://www.bitarray.io/)

[PYTHON](https://www.bitarray.io/python/" \o "View all posts filed under python)

# Fast API in Python

“Fast API is used for generating REST API which is used for web services. It is completely open-source and the use of this web framework is done for building various kinds of python APIs one of which is REST API which is consistent, cacheable, and stateless in behavior and creates the standards for the transfer of data required for communication between servers and clients.

In this article, we will try to focus on the subtopics of Fast API to understand it better. These subtopics include an overview of Fast API, a Comparison between the framework of Fast API and Flask, a simple program of “Hello Bit Array” using Fast API with GET and POST, building blocks of Fast API and the future scope. Let’s begin with our learning.

### Overview of Fast API

Fast API is the web framework with extremely high performance and suitable for the modern age requirements which can prove of enormous helps in building the application programming interfaces that are APIs along with the use of Python support for versions 3.6 or higher. These APIs are built on the type hints of python’s standards. One of the specialties about Fast API is that it has its inbuilt support for the async characteristics and features that are present in python 3.6 and higher which makes its functionality extremely fast in performance.  
Due to existing shortcomings in the DRF and Flask frameworks, Sebastian Ramirez had created Fast API which was further released in 2018. The tool used for the creation of Fast API includes Pydantic and starlette. Nowadays, Fast API is used for building the APIs and apps in some of the huge tech companies including Microsoft, Netflix and Uber.

### Some of the features of Fast API are as mentioned below

• ****Robustness**** – The documentation available is automated and interactive which helps in creating the production-ready code.  
• ****Extreme performance**** – Reading the word, Fast API we can automatically conclude that it will be very fast in its working that is where it derived its name from. Among all the frameworks available in python right now, this one is the fastest of all.  
• ****Fast coding –****The increase in the speed of developing the application when made use of Fast API is approximately 200% to 300%.  
• ****Intuitive interface**** – The availability of vast documentation and support of the editor makes it very easy for learning the design in Fast API. Also, using its available functionalities is very easy.  
• ****Compatibility**** – The framework is compatible and works absolutely fine with the schema of JSON, swagger, or even called OpenAPI now and the open standards declared for the application programming interface that is APIs.  
• ****Reduction in the number of bugs****  –  It is considered to be decreasing the occurrences of the bugs present in the system to around 40%.  
• ****Type hints functionality****  – Conversion and data validation can be done smoothly by making the use of type hinting functionality.  
• ****Creation of plugins –****Dependency injection helps in making the creation of plugins easier.

### Comparison between the framework of Fast API and Flask

Flask is another framework available in python which is actually referred to as a microframework that has its features including authentication, caching and the Object Relation Mapping that has an extensive implementation in the building of web apps with the help of python language. This framework is as well very scalable, fast and easy.

The below table list outs the advantages of using flask and fast API.

|  |  |
| --- | --- |
| **Flask** | **Fast API** |
| Intuitive and easy interface for using. | Exception Handling |
| Flexible and components can be changed at our convenience. | Coding support of asynchronous nature |
| Availability of development server that is built-in. | Validation of data |
| It has its applications in projects such as static websites, social networks, e-commerce systems, social media bots, etc. | It has its applications in projects such as machine learning model deployment, management of internal crises and web applications where authentication, login system and account management is done. |

### Simple program of Hello Bit Array using Fast API

There are four HTTP methods also referred to as operations sometimes which include PUT, DELETE, GET and POST. In GET method, we can pass certain parameters in the query string while the POST method expects the input data in the form of a JSON input file in python. You can use any one of them depending on how much data you want to pass from client to API and how sensitive it is.

### GET HTTP method

We will consider one simple example which will demonstrate the use of GET method/ operation. This will be a simple API returning the data in the JSON format and will contain “Hello Bit Array” message in it. We will create the python file named ****main.py**** which will contain the following code in it –

#Required libraries should be imported

**from** *fastapi* **import** FastAPI

#Fast API instance needs to be created

sampleBitArrayApp = FastAPI()

#Choose and create the required path operation (Here "/")

@sampleBitArrayApp.get("/")

#Function created earlier should be defined

**def** func():

**return** {"Customized Message": "Hello Bit Array"}

In order to create API, we had carried out four major steps involving importing of necessary libraries, creating a new instance of Fast API, defining the operation of the path where you want your output to be seen along with HTTP method such as GET or POST and then define the function corresponding to path operation as in our case a message.  
Now, to confirm if our API is working, you can enter the following command on the terminal by navigating to the directory where main.py lies.

$ uvicorn main:sampleBitArrayApp  -reload

INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)

INFO: Started reloader process [484138] using statreload

INFO: Started server process [484140]

INFO: Waiting **for** application startup.

INFO: Application startup complete.

INFO: 127.0.0.1:43896 - "GET / HTTP/1.1" 200 OK

Now, time to run on browser and check the resultant on link http://127.0.0.1:8000

**

**POST HTTP method**

Now, for a POST request, let us consider this example

**from** fastapi **import** FastAPI

**from** *pydantic* **import** BaseModel

sampleBitArrayApp = FastAPI()

**class** value\_of\_pages\_content(BaseModel):

Number\_of\_pages\_for\_article1:int

Number\_of\_pages\_for\_article2:int

Number\_of\_pages\_for\_article3:int

Number\_of\_pages\_for\_article4:int

@sampleBitArrayApp.post('/show\_content')

**def** show\_content(content: value\_of\_pages\_content):

**return**({"content":[content.Number\_of\_pages\_for\_article1,content.Number\_of\_pages\_for\_article2,content.Number\_of\_pages\_for\_article3,content.Number\_of\_pages\_for\_article4]})

When we go for accessing the URL http://127.0.0.1/show\_content.

We get following output in the response content when we pass respective values for Number\_of\_pages\_for\_article1, Number\_of\_pages\_for\_article2, Number\_of\_pages\_for\_article3 and Number\_of\_pages\_for\_article4 –

$curl http://127.0.0.1:8000/show\_content -X POST -d '{"Number\_of\_pages\_for\_article1":1, "Number\_of\_pages\_for\_article2":2, "Number\_of\_pages\_for\_article3":3,"Number\_of\_pages\_for\_article4":4}' -H "Content-Type: application/json"

{"content":[1,2,3,4]}

****Building blocks of Fast API****  
Some of the most important concepts when learning Fast API include request body, query parameters and path parameters. Let’s have a detailed discussion on it.

****Request body****  
It’s is the data that is supposed to be transferred from the client to the target API you are creating. Pydantic models are used in fast API to inculcate this. Let us consider one example to understand how it can be done

**from** typing **import** Optional

**from** *pydantic* **import** BaseModel

**from** *fastapi* **import** FastAPI

**class** Article(BaseModel):

topic: str

definition: Optional[str] = **None**

numberOfWords: int

writer: Optional[str] = **None**

sampleBitArrayApp = FastAPI()

@sampleBitArrayApp.post(“/articles/”)

**def** create\_article(article: Article):

**return** article

In the above example, we first imported the packages that were required, and then the request data model that we will be using was declared. Further, we went for creating a new instance of the class of FastAPI that we have created and POST path is created to which we will be adding our newly created request model.  
The output of the above program is –

$ curl http://127.0.0.1:8000/articles/ -X POST -d '{"topic":"FastAPI in python","definition":"Features - how to use","numberOfWords":1010, "writer": "Bitarray\_user"}' -H "Content-Type: application/json" 2>/dev/null|jq

{

"topic": "FastAPI in python",

"defination": null,

"numberOfWords": 1010,

"writer": "Bitarray\_user"

}

**Query parameters**  
These are optional in nature and when the declaration of path parameters is not done for functional parameters than by default they are treated as the query parameters.

Let us have one example to get insights –

**from** fastapi **import** FastAPI

sampleBitarrayApp = FastAPI()

article\_topics = [{"article\_topic": "Redis"}, {"article\_topic": "Fast API"}, {"article\_topic": "PostgreSQL"}]

@sampleBitarrayApp.get("/articles/")

**def** read\_articles(begin: int, stop: int):

**return** article\_topics[begin : begin + stop]

The query parameters are present in the link after the presence of “?” which acts as a separator. All the query parameters are in the format of key-value pairs and are separated among themselves by using an ampersand & symbol. The query parameters in the above URL  ****begin**** which has the value of 0 and ****stop**** with 10 as its value.

****output:****

$ curl "http://127.0.0.1:8000/articles/?begin=0&stop=3" 2>/dev/null|jq

[

{

"article\_topic": "Redis"

},

{

"article\_topic": "Fast API"

},

{

"article\_topic": "PostgreSQL"

}

]

**Path parameters**

We can make the scop of the API to a single source by using the path parameters in it which in turn results in permission that nobody is needed to create when building resource finder or similar API. All the path parameters are enclosed inside the curly parenthesis that are {} as it helps in controlling the aspect of representation done for the particular resources. The position of path parameters is present before the query string and end till the endpoint. An example related to the same is

**from** fastapi **import** FastAPI

sampleBitarrayApp = FastAPI()

@sampleBitarrayApp.get("/articles/{article\_topic}")

**def** read\_article(article\_topic):

**return** {"article\_topic": article\_topic}

Here the article\_topic is the path parameter which is further passed to the function of read\_articles as an argument.

****Output:****

curl "http://127.0.0.1:8000/articles/3"

{"article\_topic":"3"}

### 1. What is REST?

REST stands for Representational State Transfer.

### 2. What is a REST API?

An API is an application programming interface, which is a software-to-software interface that allows otherwise separate applications to interact and share data. In a REST API, all data is treated as resources, each one represented by a unique uniform resource identifier (URI).

### 3. What do you mean by RESTful web services?

REST API is also known as RESTful web services that follow the REST architecture.

### 4. What are cache-control headers?

Cache-control headers are used to control catching and to attain caching ability. The most commonly used cache-control headers are public, private, and No-Store.

### 5. What are the features of RESTful web services?

REStful web services have the following unique features:

* Client-server decoupling
* Communication support
* Lightweight
* Uniform interface
* Stateless
* Layered system
* Cacheable
* Code on demand

### 6. What is the definition of messaging in terms of RESTful web services?

In REST API web services, when a REST client wants to send a message to the server, it can be sent in an HTTP request form, and the same applies to the server. This kind of communication is called messaging in REST.

### 7. Explain ‘Addressing’ in RESTful web services.

The process of locating various types of resources with the help of a URL on the REST server is known as ‘addressing’ in RESTful web services. Usually, single or multiple resources are addressed by resources.

### 8. Why are REST services easily scalable?

REST services are scalable due to the statelessness that they do not store data on the server even though they are requested and do not require much communication.

### 9. What are Idempotent methods?

Idempotent methods are known to return the same outcome even after the same request has been made multiple times, and it avoids errors caused by duplicate requests on the client side.

### 10. How can RESTful web services be tested?

The RESTful web services can be tested with the help of tools such as Swagger and Postman, which enable users to inspect query parameters, response headers, and headers, documentation of the endpoints, and conversion of endpoints to XML and JSON.

### 11. What are payloads in RESTful web services?

Payloads are the request data passed through the POST or GET method and found in the message’s body of an HTTP request in RESTful web services.

### 12. What is the maximum payload size that can be sent in POST methods?

Theoretically, there is no such maximum limit for payload size that can be sent in POST methods. However, payloads with larger sizes can consume larger bandwidth. Thus the server could take more time to proceed with the request.

### 13. Which protocol does REST APIs use?

Protocols are used to communicate with clients where REST APIs use HTTP protocol for it.

### 14. In REST APIs, which markup languages are used to represent the resources?

The resources in REST APIs are represented with the help of XML (extensible markup language) and JSON (JavaScript Object Notation).

### 15. Differentiate POST and PUT methods.

POST Method

* POST can create a resource on the server.
* POST is not idempotent.
* POST responses are cacheable.

PUT Method

* PUT is used to replace a resource at a specific URI with another resource.
* PUT is idempotent that it will only result in one resource even after calling it multiple times.
* PUT responses are not.

### 16. Which HTTP request methods are supported by REST?

REST supports various types of HTTP request methods such as GET, POST, PUT, DELETE, HEAD, OPTIONS, ETC.

### 17. What is CRUD?

CRUD stands for “Create, Read, Update, Delete.”

### 18. The main parts of an HTTP response

The main parts of the HTTP response are the HTTP version, Status line, HTTP Response Header, and HTTP Response body.

### 19. What are the most common HTTP response status codes you see while working in REST API?

Some of the most common response status codes are 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable, etc.

### 20. What is a resource?

In REST, A resource is an object with a label and accessible on the server. Resources consist of associated data, a list of methods, and a relationship with other resources on the server.

### 21. What is a URI?

URI stands for ‘Uniform Resource Identifier.

### 22. What is caching in the REST API?

REST API stores a copy of a server response in a particular location of computer memory to retrieve the server response fast in the future. This method is temporary and called "catching."

### 23. What’s a real-world example of a REST API?

1. Public REST APIs are harnessed by weather apps to display weather information and share the related data.
2. Airlines use APIs to expose the flight times and prices to allow travel and ticketing sites for businesses.
3. Public transportation services use APIs to make their data publicly open to make it available for mapping and navigation apps in real-time.

### 24. What is the difference between REST and SOAP?

REST(Representational State Transfer)

* It is an architectural design pattern used to develop web services.
* It is faster in speed and more cacheable.
* It inherits only the security measures concerning the protocol that have been implemented.

SOAP (Simple Object Access Protocol)

* It is a strict protocol used to build secure APIs.
* It is slower in speed and not cacheable.
* It is able to define its own security measures.

### 25. What do you understand about JAX-RS?

It is a Java-based specification implemented for RESTful services and defined by JEE.

### 26. Disadvantages of RESTful web services?

* RESTful web services are stateless and do not maintain session simulation responsibility as the client side does not provide a particular session id for it.
* REST is not able to impose the security restriction inherently. However, it inherits them with the help of implementing protocols. Thus, the integration of SSL/TLS authentication needs to be done very carefully for better security measures of the REST APIs.

### 27. Advantages of REST

* HTTP makes the implementation of REST easy.
* REST fits in the existing infrastructure of the web, thus the web application can easily implement the REST. XML and JSON web technologies make REST easy to learn.
* The client and server communication is stateless, thus the integration is easy to build and scalable, and manageable with respect to time.
* The REST architecture can adapt to a huge variety of cases due to its flexibility.
* The lightweight architecture of REST makes it easy to build the applications faster as compared to other types of APIs.
* REST can be tested easily in the browser with the help of API testing tools.

### 28. How do you keep REST APIs secure?

REST APIs can be kept secure with the help of safety measures such as Authentication and authorization, API Server Validation, TSl/SSL Encryption, Rate-limiting for DDoS attacks, and sensitive information such as username, password, or authentication token should not be visible in URIs

### 29. What are “Options” in REST APIs?

It is an HTTP method used to fetch the supported HTTP options or operations that help clients to choose the options in REST APIs. Cross-Origin Resource Sharing (CORS) uses the REST option method.

### 30. Different types of API architectures

There are other two API architectures used, SOAP (Simple Object Access Protocol), and RPC (Remote Procedure Call)

### 31. What are the different application integration styles?

The different application integration styles are Shared database, Batch file transfer, Invoking remote procedure (RPC), and Swapping asynchronous messages over a message-oriented middleware (MOM).

### 32. How is JAXB related to RESTful web API?

JAXB is a Java arch used for XML binding in RESTful web API.

### 33. What is AJAX?

AJAX stands for  Asynchronous javascript and XML.

### 34. What does the HEAD method in REST APIs do?

The HEAD method is used to return the HTTP Header in read-only form and not the Body.

### 35. Which frameworks can JAX-RS implement in the RESTful web?

JAX-RS is used to implement frameworks such as Jersey, RESTEasy, Apache, and CFX.

### 36. What are HTTP status codes and their meaning?

* Code 200: success.
* Code 201:resource has been successfully created.
* Code 204: no content in the response body.
* Code 404: no method available.

### 37. What is a ‘Resource’?

‘Resource’ is defined as an object of a type that includes image, HTML file, text data, and any type of dynamic data.

### 38. Why is the proper representation of resources required?

Proper representations of resources in the proper format allow the client to easily understand the format and determine the identification of resources easily.

### 39. How to design Resources representation for RESTful web services?

* It should be easy to understand for the client and server.
* It should be complete irrespective of its format structure.
* It should consider the link of the resources to other resources and handle it carefully.

### 40. Important aspects of RESTful web services implementation.

* ResourcesRequest
* Headers
* Request Body
* Response Body
* Status codes