Sheryians Coding School

REST-API

REST-API

Understanding REST APIs

REST (Representational State Transfer) is an architectural style for designing networked applications. It relies on stateless, client-server communication over HTTP using standard methods and status codes. RESTful APIs are designed around resources, which can be anything from users and products to documents.

Key Concepts

1. **Resources:** Everything that can be accessed via a RESTful API is considered a "resource." Each resource has a unique identifier (URI).
2. **Representations:** Resources are transferred in some representation like JSON or XML.
3. **Stateless Communication:** Each request from client to server contains all needed information; the server does not store any state about the client session.
4. **HTTP Methods:** REST APIs use standard HTTP methods to perform actions on resources.
5. **HTTP Status Codes:** Servers use HTTP status codes to indicate the outcome of a client's request.

HTTP Methods

HTTP methods define the action you want to perform on a resource. Here are the most common methods:

Method	Description	Idempotent?
GET	Retrieves a resource or a list of resources. Should no modify data on the server.	Yes
POST	Creates a new resource. The request body contains t data for the new resource.	No
PUT	Updates a resource by replacing it with new data. Requires complete new representation in the reques body.	Yes

REST-API

Method	Description	Idempotent?
PATCH	Updates a resource by partially modifying it. Require only the modified fields in the request body.	Yes
DELETE	Deletes a resource.	Yes

Important Notes on Methods:

- **Idempotence:** An idempotent method produces the same result if called once or multiple times with the same request (e.g., **GET , PUT , PATCH , DELETE**).POST is generally not considered idempotent.
- **Safe methods:** safe method should not modify server data (**GET , HEAD ,OPTIONS**).

HTTP Status Codes

Status codes are three-digit numbers the server uses to indicate the outcome of a client's request. They are categorized into five classes:

Status Code Range	Meaning
1xx (Informational)	The request was received, continuing process
2xx (Success)	The request was successfully received, understood, and accepted.
3xx (Redirection)	Further action needs to be taken by the client to fulfill th request.
4xx (Client Error)	The client sent a request with invalid syntax or which cou not be fulfilled.
5xx (Server Error)	The server failed to fulfill an apparently valid request. These errors typically indicate problems with the server backend services.

Common Status Codes:

Status codes are three-digit numbers the server uses to indicate the outcome of a client's request. They are categorized into five classes:

Code	Category	Name / Description	Use Case
200	Success	OK: The request was successful.	A successful GET request
201	Success	Created: A new resource has been created.	A successful POST request
204	Success	No Content: The request was successful, but there's no content to return.	DELETE request when the resource is removed successfully
301	Redirection	Moved Permanently: The resource has moved to a new URL.	Redirect old URL to new one
302	Redirection	Found: The resource has been found at a different URL (temporary redirect).	Temporary redirect to different URL
304	Redirection	Not Modified: The resource hasn't changed since the last request.	Conditional GET request to prevent unnecessary transfer
400	Client Error	Bad Request: The request was malformed or invalid.	Invalid request body or missing parameters

Code	Category	Name / Description	Use Case
401	Client Error	Unauthorized: User is not authenticated.	Accessing a protected resource
403	Client Error	Forbidden: User authenticated but not authorized.	Permission denied
404	Client Error	Not Found: Resource does not exist.	Invalid URL
405	Client Error	Method Not Allowed: HTTP method not supported.	POST on read-only endpoint
409	Client Error	Conflict: Request conflicts with resource state.	Deleting related entities
422	Client Error	Unprocessable Entity: Semantic errors in request	Duplicate email or invalid data
500	Server Error	Internal Server Error: Something went wrong on server.	Generic server-side failure
501	Server Error	Not Implemented: Feature not supported by server.	Unsupported operation
503	Server Error	Service Unavailable: Server temporarily unavailable.	Server overload or maintenance

REST-API

RESTful API Design Best Practices:

- Use nouns to represent resources (e.g., /users , /products).
- Use plural nouns for collections (e.g., /users).
- Use HTTP methods according to their semantics (GET for read, POST for create, etc.).
- Use status codes appropriately to convey the outcome of the request.
- Keep your APIs consistent and predictable.
- Design stateless APIs that do not rely on session storage on the server.

Interview Questions and Answers:

Q1: What is a REST API?
Answer:- REST (Representational State Transfer) is an architectural style for designing networked applications. It relies on stateless client-server communication using standard HTTP methods and status codes. RESTful APIs are designed around resources (like data), which can be accessed using methods like GET , POST , PUT , PATCH , DELETE .

Q2: What are the common HTTP methods used in REST APIs, and what do they mean?
Answer:-
- GET : Retrieve a resource or a list of resources.
- POST : Create a new resource.
- PUT : Update a resource by replacing it with new data.
- PATCH : Update a resource by partially modifying it.
- DELETE : Delete a resource.

Q3: Explain the difference between PUT and PATCH .
Answer:-
- PUT : Used to completely replace a resource. It expects the entire resource representation in the request body.
- PATCH : Used to partially modify a resource. It only requires the fields that need to be updated in the request body.

Q4: What are HTTP status codes, and why are they important in REST APIs?
Answer:- HTTP status codes are three-digit numbers the server sends to the client to indicate the outcome of a request. They are important because they allow clients to understand whether a request was successful, failed, or needs more action. They also help in troubleshooting API issues.

Q5: Give some examples of common HTTP status codes and when they are used?
Answer:-
- 200 OK : The request was successful (e.g., retrieving a resource).
- 201 Created : A new resource was successfully created (e.g., POST request).
- 400 Bad Request : The request was invalid.
- 401 Unauthorized : The client needs authentication (API key, credentials).
- 403 Forbidden : The client is authenticated but does not have permission to
- access the resource.
- 404 Not Found : The resource does not exist.
- 500 Internal Server Error : A server-side error occurred.

Q6: What does it mean for a method to be idempotent? Which HTTP methods are idempotent?
Answer:- An idempotent method produces the same result regardless of how many times it's called with the same request. GET , PUT , PATCH , and DELETE are idempotent methods. POST is usually not.

Q7: What does it mean for a method to be safe method? which HTTP methods are safe?
Answer:- A safe method should not modify any data on the server. GET , HEAD , and OPTIONS methods are safe method.

Q8: If an API operation fails, should the API return error status code?
Answer:- Yes. API should always return proper error status code with a descriptive message. For ex.

- Use 400 Bad Request for invalid request data.
- Use 404 Not Found if resource is not available.
- Use 500 Internal Server Error when a unexpected error happens in server.

Q9: What are some best practices for designing RESTful APIs?
Answer:-

- Use nouns to represent resources and pluralize collections.
- Use HTTP methods according to their semantics.
- Use status codes appropriately to convey the outcome of the request.
- Keep your APIs consistent and predictable.
- Design stateless APIs.

Q10: How do you handle errors in RESTful APIs?
Answer:-

- Use proper HTTP status codes to indicate the type of error.
- Include a descriptive error message in the response body (e.g., in JSON format).
- Log errors on the server side for debugging purposes.