

Basics of REST

1. What is a REST API?

REST (Representational State Transfer) is an architectural style for designing networked applications using stateless communication and standard HTTP methods.

2. What are the key principles of REST?

- Statelessness
- Client-Server architecture
- Uniform Interface
- Cacheable
- Layered System
- Code on Demand (optional)

3. What is a resource in REST?

A resource is any data that can be identified using a URI. For example, /users/1 is a resource representing user with ID 1.

4. What is the difference between PUT and POST?

- **POST**: Create a new resource.
- **PUT**: Update an existing resource or create it if it doesn't exist (idempotent).

5. What is idempotency in REST?

An operation is idempotent if performing it multiple times produces the same result. GET, PUT, DELETE are idempotent; POST is not.

✓ HTTP Methods & Status Codes

6. List HTTP methods used in REST APIs.

- GET: Retrieve data
- POST: Create data
- PUT: Update/replace data
- PATCH: Partially update data
- DELETE: Remove data

7. What are common HTTP status codes in REST?

- 200 OK – Success
- 201 Created – Resource created
- 204 No Content – Success, no content to return
- 400 Bad Request – Invalid input
- 401 Unauthorized – Auth needed

- 403 Forbidden – Access denied
- 404 Not Found – Resource not found
- 500 Internal Server Error – Server-side error

8. What's the difference between 401 and 403?

- 401 Unauthorized: No or invalid authentication.
- 403 Forbidden: Authenticated but not allowed to access the resource.

9. When do you return 204 No Content?

When the server has successfully fulfilled the request but there's nothing to return (e.g., successful DELETE).

10. Is GET method allowed to modify data?

No. GET must be safe and should only retrieve data without side effects.

✓ URI Design & Best Practices

11. How should REST URIs be structured?

Use nouns and hierarchical URIs. Example:

- /users
- /users/1/orders

12. Should URIs contain verbs?

No. Use HTTP methods for actions. URIs should represent resources, not actions.

13. How do you handle versioning in REST APIs?

- URI versioning: /api/v1/users
- Header versioning: Accept: application/vnd.company.v1+json

14. How do you represent relationships between resources?

Use nested routes or linking. Example: /users/1/orders for user's orders.

15. What is HATEOAS?

Hypermedia As The Engine Of Application State. Clients interact with resources via hyperlinks provided dynamically by the server.

✓ Security & Auth

16. How do you secure a REST API?

- HTTPS
- Authentication (Basic, OAuth2, JWT)
- Input validation
- Rate limiting
- CORS control

17. What is the role of JWT in REST APIs?

JWT (JSON Web Token) is a token format used to represent claims securely between parties and is often used in stateless auth.

18. What is CORS?

Cross-Origin Resource Sharing – a security feature that restricts web pages from making requests to a different domain.

19. What is the difference between Authentication and Authorization?

- Authentication: Who are you?
- Authorization: What are you allowed to do?

20. How do you prevent CSRF in REST APIs?

CSRF is less common in REST (especially with JWT + no cookies). To prevent:

- Use tokens
- Use CORS with strict origin
- Disable cookies for APIs

✓ Advanced Concepts

21. What are some REST API performance optimizations?

- Caching (Cache-Control)
- Pagination
- Filtering/sorting
- Compression (GZIP)
- Asynchronous processing

22. What is content negotiation?

The ability for the client to specify preferred response format via headers like Accept: application/json.

23. What is pagination and how do you implement it?

It splits large responses into chunks. Example query params:

- /users?page=2&size=10

24. How do you handle errors in REST APIs?

- Return structured error response:

json

CopyEdit

```
{
```

```
  "error": "Invalid ID",
```

```
  "code": 400
```

}

- Use proper status codes

25. Difference between synchronous and asynchronous REST calls?

- **Synchronous:** Immediate response expected
- **Asynchronous:** Request is accepted, response delivered later (e.g., 202 Accepted)

✓ Spring Boot Specific (for Java)

26. How do you create a REST controller in Spring Boot?

Use `@RestController` and `@RequestMapping`.

java

CopyEdit

```
@RestController
```

```
@RequestMapping("/api/users")
```

```
public class UserController { ... }
```

27. What does `@RestController` do?

It combines `@Controller` and `@ResponseBody`, making methods return JSON/XML by default.

28. How do you handle exceptions in Spring REST API?

Use `@ControllerAdvice` and `@ExceptionHandler`.

29. How do you validate input in Spring REST API?

Use `@Valid` and validation annotations like `@NotNull`, `@Size` with `BindingResult`.

30. How do you document REST APIs in Spring Boot?

Use **Swagger/OpenAPI** with `SpringDoc` or `Springfox`. It generates interactive API docs.