# 1. SOAP vs Restful ?

Security: While it can be easier and faster to provide security on SOAP, this can be complicated for REST. When it comes to security for REST, it is a bit weak compared to SOAP.

Data Structure: While we can work with REST with JSON, XML and even TEXT, we should use XML with SOAP. REST can be more useful in this way. With JSON, you can perform operations with smaller data. If data sizes are important for your application, using REST will be suitable for you.

Application Speed: If you want your application to run faster, it will be beneficial to use REST.

Support: Developer tools for SOAP are better. We can find more resources to help us use it. REST has less documentation than SOAP.

# 2. Difference between acceptance test and functional test ?

Functional testing - test the product, verifying that it has the qualities you've designed or build (functions, speed, errors, consistency, etc.)

Acceptance testing - test the product in its context, this requires (simulation of) human interaction, test it has the desired effect on the original problem(s).

# 3. What is Mocking ?

In general, every mock created for unit testing is called mocking. however mock is only one of the test double types.

Other types of test doubles are dummy, stub, spy and fake.

# 4. What is a reasonable code coverage % for unit tests (and why) ?

100%. You might choose this because you want to be sure everything is tested. This doesn't give you any insight into test quality, but does tell you that some test of some quality has touched every statement (or branch, etc.) Again, this comes back to degree of confidence: If your coverage is below 100%, you know some subset of your code is untested.

Some might argue that this is silly, and you should only test the parts of your code that are really important. I would argue that you should also only maintain the parts of your code that are really important. Code coverage can be improved by removing untested code, too.

99% (or 95%, other numbers in the high nineties.) Appropriate in cases where you want to convey a level of confidence similar to 100%, but leave yourself some margin to not worry about the occasional hard-to-test corner of code.

80%. I've seen this number in use a few times, and don't entirely know where it originates. I think it might be a weird misappropriation of the 80-20 rule; generally, the intent here is to show that most of your code is tested. (Yes, 51% would also be "most", but 80% is more reflective of what most people mean by most.) This is appropriate for middle-ground cases where "well-tested" is not a high priority (you don't want to waste effort on low-value tests), but is enough of a priority that you'd still like to have some standard in place.

# 5. HTTP/POST vs HTTP/PUT ?

POST and PUT are both HTTP methods used to send data to the server.

POST is only used to send data to a specific resource and what to do with the data is up to the server. In PUT, the same resource is accessed with the same address and if there is content, it is replaced with the incoming data, if there is no content, new content is created. With PUT, it is mostly preferred to send file-based content to the server.

# 6. What are the Safe and Unsafe methods of HTTP ?

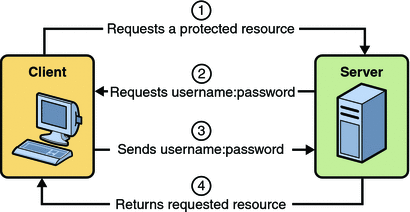
HTTP methods can be both safe or unsafe. By safe, it means the methods are not expected to change any resource on the server and by unsafe it means the methods are expected to change some resource on the server.

safe methods: GET, HEAD, OPTIONS, TRACE

unsafe methods: POST, PUT, DELETE, CONNECT, PATCH

# 7. How does HTTP Basic Authentication work ?

HTTP Basic Authentication is a method for the client to provide a username and password when making a request. In this authentication method, which can be set up for a specific file or the entire workspace on the server side, the client must enter the correct login information.



# 8. Define RestTemplate in Spring ?

RestTemplate is the default class in Spring library to handle synchronous HTTP requests on client side.

*RestTemplate restTemplate = new RestTemplate();*

*User forObject = restTemplate.getForObject("http://localhost:8080/users/1", User.class);*

# 9. What is idempotant and which HTTP methods are idempotant ?

If a method is called once and the result is the same when it is called more than once, it is an idempotent method. For example: GET, PUT and DELETE.

**GET**

When we send a request like /students/1 to the server, the record with id 1 will be returned. Repeating this request more than once will not change the result. Therefore, GET is an idempotent method.

**PUT**

/students/1

body: {'name':'Ahmet', 'surname':'Acar'}

When we send a request to the server as above, the text Alnıçık expression will be updated as Ahmet Acar. When we send the request once again, there will be no change when the value in the database is Ahmet Acar. This is also an idempotent method.

**DELETE**

/students/1

When we send a request to the server as above, the record with id 1 will be deleted. When we call the relevant request again, no action will be taken because there is no record with an id of 1. This is also an idempotent method.

# 10. What is DNS Spoofing ? How to prevent ?

DNS spoofing is a type of cyber attack that aims to trick internet users into sharing sensitive information by allowing them to reach a different site that looks like the original site instead of the one they target

Here are a few precautions website owners and DNS service providers can take:

1. Tools built against DNS cache poisoning

2. Domain name system security extensions

3. End-to-end encryption

Security measures for users:

1. Never click on a link you don't know

2. Scan your computer regularly for malware

3. Clear your DNS cache to resolve the poisoning

4. Use a Virtual Private Network (VPN)

# 11. What is content negotiation ?

Content Negotiation is a concept specific to the HTTP protocol. If we translate it literally, we can say that it is a content agreement or negotiation between the client and the server. Its purpose is to be able to serve content in different document types with the same URI. In other words, in more general terms, we can say that the form of reference is determined by the users.

# 12. What is statelessness in RESTful Web Services ?

Stateless: On the server side, no client-related context or session is kept. Every request made by the client carries the necessary information for the server to give a response, that is, all kinds of state are kept by the client, and if needed, it is reported to the server in the request. This is also important for Scalability because it makes it unnecessary for the server to store any state between requests and makes resource management easier. Visibility is important because the information contained in a single request is sufficient to understand the purpose of the request.

Of course, being stateless also has some disadvantages. The client has to add the necessary information in each request, which increases the network traffic. This also makes it difficult for the server to control the consistency of the application's behavior, because requests with different content may come from many different clients, putting more load on the server in terms of validation.

REST is stateless, meaning it does not store state information. Let's open a little more; In REST standards, extra header information is not stored in the data transferred between the client-server, the details of the client are not found, this information is not carried between the client-server.

# 13. What is CSRF attack? How to prevent ?

It is the name given to the attacks made on the system by sending parameters via URL addresses in web applications.

**CSRF Vulnerability Prevention**

**Precautions on the User's Side**

One of the most important and constantly expressed measures is not to click on links that have not verified the security of the source. Do not click on links in e-mails that you are not sure from.

Get access by scanning the links on malicious analysis platforms and checking the domain.

Clear the cache data regularly. (cookie, site data)

**Precautions on the System Side**

Requests made by users should be received with the POST method.

CSRF is prevented with tokens specially produced for the user. This token is reproduced in every transaction and stored in the form, making it impossible for the attacker to guess this token. The important thing in this part is to use a correct pattern and reliable cryptography methods when generating tokens. There are multiple token method creation methods.

# 14. What are the core components of the HTTP request and HTTP response ?

**There are 5 major components for HTTP Request.**

Verb − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.

URI − Uniform Resource Identifier (URI) to identify the resource on server.

HTTP Version − Indicate HTTP version, for example HTTP v1.1 .

Request Header − Contains metadata for the HTTP Request message as key-value pairs. For example, client ( or browser) type, format supported by client, format of message body, cache settings etc.

Request Body − Message content or Resource representation

**Every HTTP response contains four key elements.**

Status/Response Code – These are response codes issued by a server to a client’s request. For example, 404 means Page Not Found, and 200 means Response is OK.

HTTP Version – describes HTTP version, for example-HTTP v1.1.

Response Header – Includes information for the HTTP response message. For example, Content-type, Content-length, date, status and server type.

Response Body – It contains the data that was requested by a client to server.