HOMEWORK-3

RÜMEYSA TÜRKER

1 – SOAP vs Restful?

- Rest is more flexible and useful than XML-based SOAP, as it supports data types such as JSON, HTML, and XML.
- Soap has to use a W3C standard WSDL language to generate its requests. Rest does not require this, requests can be created via the URI. REST is easier to use and design because it doesn't need a language.
- In terms of security, there are ready-made functions for SOAP, and its more complexity makes it more advantageous in this regard.
- Both use the HTTP protocol and provide communication between server-client. However, HTTP is required for REST and SOAP can also support protocols such as TCP/IP STMP.
- REST uses simple HTTP method for caching request and is easier than SOAP. SOAP makes mixed XML requests for this process.

2 - Difference between acceptance test and functional test?

If your aim is to test only one function regardless of the environment and side effects; functional testing.

If your aim is to check whether the expected operations from the software can be performed; acceptance test.

A functional test verifies that the product actually works as you (the developer) think it does.

Acceptance tests verify the product actually solves the problem it was made to solve. This can best be done by the user (customer), for instance performing his/her tasks that the software assists with.

Functional testing is performed before acceptance testing

3 - What is Mocking?

Mocking is creating objects that simulate the behavior of real objects. We use mocking in unit testing. A object that you want to test may have dependencies on other complex objects. To isolate the behavior of the object you want to test you replace the other objects by mocks that simulate the behavior of the real objects.

There are a lot of frameworks used for mocking in unit testing such as PowerMock, Mockito, EasyMock, etc.

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

Code coverage of 70-80% is a reasonable goal for system test of most projects with most coverage metrics. Use a higher goal for projects specifically organized for high testability or that have high failure costs. Minimum code coverage for unit testing can be 10-20% higher than for system testing.

5 – HTTP/POST vs HTTP/PUT?

An HTTP PUT is supposed to accept the body of the request, and then store that at the resource identified by the URI.

An HTTP POST is more general. It is supposed to initiate an action on the server.

HTTP PUT: PUT puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one. PUT is idempotent, but paradoxically PUT responses are not cacheable.

HTTP POST: POST sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource. The POST method is not idempotent, however POST responses are cacheable so long as the server sets the appropriate Cache-Control and Expires headers.

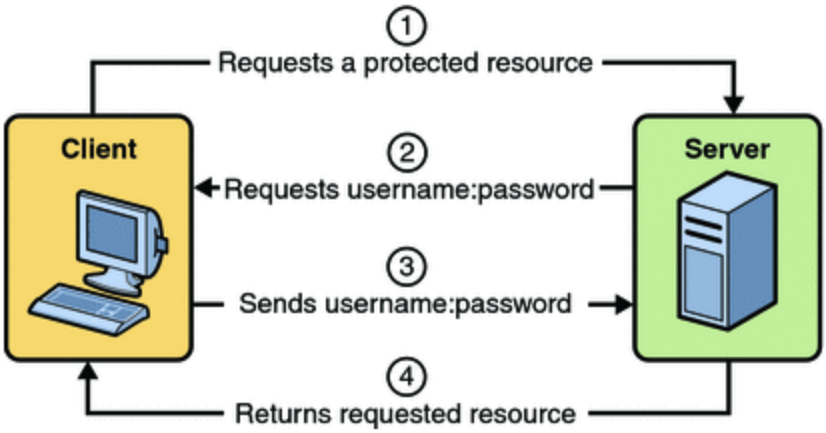
6 - What are the Safe and Unsafe methods of HTTP?

An HTTP method is safe if it doesn't alter the state of the server. In other words, a method is safe if it leads to a read-only operation. Several common HTTP methods are safe: GET, HEAD, or OPTIONS. All safe methods are also idempotent, but not all idempotent methods are safe. For example, PUT and DELETE are both idempotent but unsafe.

7 - How does HTTP Basic Authentication work?

HTTP Basic Authentication requires that the server request a user name and password from the web client and verify that the user name and password are valid by comparing them against a database of authorized users. When basic authentication is declared, the following actions occur:

1. A client requests access to a protected resource.
2. The web server returns a dialog box that requests the user name and password.
3. The client submits the user name and password to the server.
4. The server authenticates the user in the specified realm and, if successful, returns the requested resource.



HTTP Basic Authentication

8 - Define RestTemplate in Spring?

Rest Template is used to create applications that consume RESTful Web Services. You can use the exchange() method to consume the web services for all HTTP methods.

RestTemplate is a synchronous client to perform HTTP requests. It uses a simple, template method API over underlying HTTP client libraries such as the JDK HttpURLConnection, Apache HttpComponents, and others.

Since Spring 5.0, a new client WebClient is available that can be use do create both synchronous and asynchronous requests. In the future releases, RestTemplate will be deprecated in favour of WebClient.

9 – What is idempotant and which HTTP methods are idempotant?

An HTTP method is idempotent if an identical request can be made once or several times in a row with the same effect while leaving the server in the same state. In other words, an idempotent method should not have any side-effects (except for keeping statistics). Implemented correctly, the **GET, HEAD, PUT**, and **DELETE** methods are idempotent, but not the POST method. All safe methods are also idempotent.

10 – What is DNS Spoofing? How to prevent?

DNS (Domain Name Service) spoofing is the process of poisoning entries on a DNS server to redirect a targeted user to a malicious website under attacker control. The DNS attack typically happens in a public Wi-Fi environment but can occur in any situation where the attacker can poison ARP (Address Resolution Protocol) tables and force targeted user devices into using the attacker-controlled machine as the server for a specific website. It's the first step in a sophisticated phishing attack on public Wi-Fi, and it can also trick users into installing malware on their devices or divulge sensitive information.

How to Prevent DNS Spoofing

Any user that accesses the internet from public Wi-Fi is vulnerable to DNS spoofing. To protect from DNS spoofing, internet providers can use DNSSEC (DNS security). When a domain owner sets up DNS entries, DNSSEC adds a cryptographic signature to the entries required by resolvers before they accept DNS lookups as authentic.

11 – What is content negotiation ?

In HTTP, ***content negotiation*** is the mechanism that is used for serving different representations of a resource to the same URI to help the user agent specify which representation is best suited for the user (for example, which document language, which image format, or which content encoding).

12 – What is statelessness in RESTful Web Services?

Statelessness means that every HTTP request happens in complete isolation. When the client makes an HTTP request, it includes all information necessary for the server to fulfill the request.

The server never relies on information from previous requests from the client. If any such information is important then the client will send that as part of the current request.

As per the REST architecture, a RESTful Web Service should not keep a client state on the server. This restriction is called ***Statelessness***. It is the responsibility of the client to pass its context to the server and

then the server can store this context to process the client's further request. For example, session maintained by server is identified by session identifier passed by the client.

13 - What is CSRF attack? How to prevent?

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site when the user is authenticated. A CSRF attack works because browser requests automatically include all cookies including session cookies. Therefore, if the user is authenticated to the site, the site cannot distinguish between legitimate authorized requests and forged authenticated requests.

How to pretend?

Security experts propose many CSRF prevention mechanisms. This includes, for example, using a referer header, using the HttpOnly flag, sending an X-Requested-With custom header using jQuery, and more. Unfortunately, not all of them are effective in all scenarios. In some cases, they are ineffective and in other cases, they are difficult to implement in a particular application or have side effects. The following implementations prove to be effective for a variety of web apps while still providing protection against CSRF attacks. For more advanced CSRF prevention options, see the CSRF prevention cheat sheet managed by OWASP.

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

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

1. ***Verb –*** Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
2. ***URI –*** Uniform Resource Identifier (URI) to identify the resource on server.
3. ***HTTP Version –*** Indicate HTTP version, for example HTTP v1.1 .
4. ***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.
5. ***Request Body –*** Message content or Resource representation

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

1. **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.
2. **HTTP Version –** describes HTTP version, for example-HTTP v1.1.
3. **Response Header –** Includes information for the HTTP response message. For example, Content-type, Content-length, date, status and server type.
4. **Response Body –** It contains the data that was requested by a client to server.