HW#3 – İbrahim Çelik

1 – SOAP vs Restful ?

The server-side portion of the web API is a programmatic interface to a defined request-response message system, and is typically referred to as the Web Service. There are several design models for web services, but the two most dominant are SOAP and REST.

SOAP provides the following advantages when compared to REST:

- Language, platform, and transport independent (REST requires use of HTTP)
- Works well in distributed enterprise environments (REST assumes direct point-to-point communication)
- Standardized
- Provides significant pre-build extensibility in the form of the WS* standards
- Built-in error handling
- Automation when used with certain language products

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

- Uses easy to understand standards like swagger and OpenAPI Specification 3.0
- Smaller learning curve
- Efficient (SOAP uses XML for all messages, REST mostly uses smaller message formats like JSON)
- Fast (no extensive processing required)
- Closer to other Web technologies in design philosophy

As one REST API tutorial put it: SOAP is like an envelope while REST is just a postcard.

Certainly a postcard is faster and cheaper to send than an envelope, but it could still be wrapped within something else, even an envelope.

You can just read a postcard too, while an envelope takes a few extra steps, like opening or unwrapping to access what’s inside.

2 - Difference between acceptance test and functional test ?

The difference is between testing the problem and the solution. Software is a solution to a problem, both can be tested.

The functional test confirms the software performs a function within the boundaries of how you've solved the problem. This is an integral part of developing software, comparable to the testing that is done on mass produced product before it leaves the factory. 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. If the software passes this real world test, it's accepted to replace the previous solution. This acceptance test can sometimes only be done properly in production, especially if you have anonymous customers (e.g. a website). Thus a new feature will only be accepted after days or weeks of use.

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 ?

Mocking is used in testing environments. A lot of times you have a part of your program that does something really important, like for example, a bank may have a function that updates someone's account balance. When you're testing this function—which you obviously want to test it to make sure it works correctly—when you're testing it, you don't want to actually change a person's account balance, so you can create a *mock* of a person's account balance, and you can alter that mock.

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. So in simple words, mocking is creating objects that simulate the behavior of real objects.

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.

Empirical studies of real projects found that increasing code coverage above 70-80% is time consuming and therefore leads to a relatively slow bug detection rate. Your goal should depend on the risk assessment and economics of the project. Consider the following factors.

- Cost of failure. Raise your goal for safety-critical systems or where the cost of a failure is high, such as products for the medical or automotive industries, or widely deployed products.
- Resources. Lower your goal if testers are spread thin or inadequately trained. If your testers are unfamiliar with the application, they may not recognize a failure even if they cover the associated code.
- Testable design. Raise your goal if your system has special provisions for testing such as a method for directly accessing internal functionality, bypassing the user interface.
- Development cycle status. Lower your goal if you are maintaining a legacy system where the original design engineers are no longer available.

Many projects set no particular minimum percentage required code coverage. Instead they use code coverage analysis only to save time. Measuring code coverage can quickly find those areas overlooked during test planning.

Defer choosing a code coverage goal until you have some measurements in hand. Before measurements are available, testers often overestimate their code coverage by 20-30%.

5 – HTTP/POST vs HTTP/PUT ?

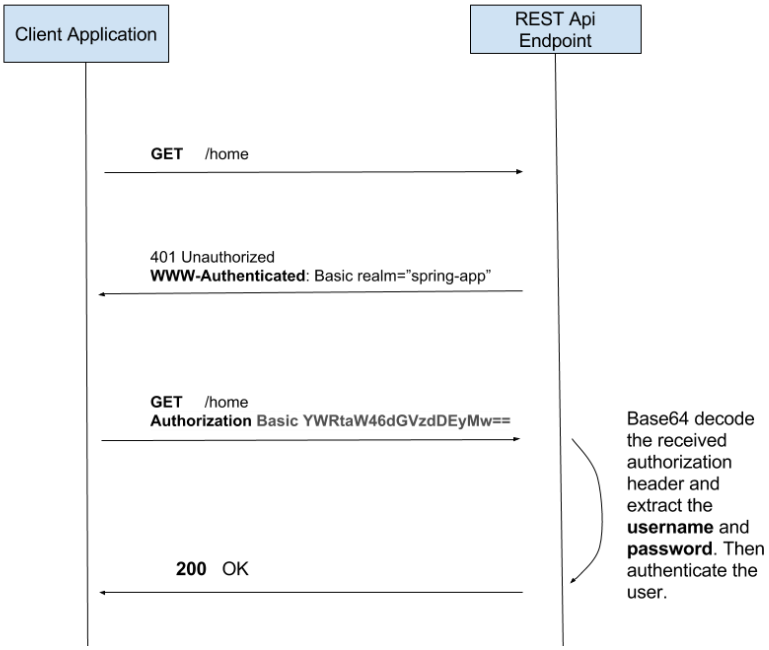
PUT	POST
This method is idempotent.	This method is not idempotent.
PUT method is call when you have to modify a single resource, which is already a part of resource collection.	POST method is call when you have to add a child resource under resources collection.
RFC-2616 depicts that the PUT method sends a request for an enclosed entity stored in the supplied request URI.	This method requests the server to accept the entity which is enclosed in the request.
PUT method syntax is PUT /questions/{question-id}	POST method syntax is POST /questions
PUT method answer can be cached.	You cannot cache PUT method responses.
PUT /vi/juice/orders/1234 indicates that you are updating a resource which is identified by “1234”.	POST /vi/juice/orders indicates that you are creating a new resource and return an identifier to describe the resource.
If you send the same request multiple times, the result will remain the same.	If you send the same POST request more than one time, you will receive different results.
PUT works as specific.	POST work as abstract.
We use UPDATE query in PUT.	We use create query in POST.
In PUT method, the client decides which URI resource should have.	In POST method, the server decides which URI resource should have.

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

An HTTP method is **safe** if a request using this method doesn’t alter the state of the server. In other words, it leads to a read-only operation. By calling a safe method, the client doesn’t request any server change, and therefore won’t create an unnecessary load or burden for the server. Although the servers can still alter their state by keeping logs or statistics. What is important here is browsers can call safe methods without fearing to cause any harm, loss of property, or unusual burden on the origin server; this allows them to perform activities like pre-fetching without risk. Web crawlers also rely on calling safe methods.

- safe methods: GET, HEAD, OPTIONS, TRACE
- unsafe methods: POST, PUT, DELETE, CONNECT, PATCH

7 - How does HTTP Basic Authentication work ?



- User (who is unauthenticated) tries to access the protected/secured REST resource.
- Server examines that the request is from an unauthenticated user and it is for accessing secured resource. Server generates unauthorized response and send back to the client application saying that authentication is required.
- Client application receives the server response and identifies that authentication is required to access the requested resource. In addition, it identifies that supported authentication type is HTTP Basic.
- The client application prepare a base64 encoded string with username and password and include it in Authorization header. Then request the access for the protected resource along with the authorization header.
- In the server end, authorization header will be decoded using base64 and extract the username and password. Then the user will be authenticated with the available authentication provider.
- If the user request is authenticated successfully, the requested protected resource will be sent back to the client application.

8 - Define RestTemplate in Spring ?

The RestTemplate is the central class within the Spring framework for executing synchronous HTTP requests on the client side.

Like Spring JdbcTemplate, RestTemplate is also a high-level API, which in turn is based on an HTTP client. By default, the class java.net.HttpURLConnection from the Java SDK is used in RestTemplate. However, the Spring Framework makes it possible to easily switch to another HTTP client API.

Most of us surely have experience with HttpURLConnection or another HTTP client API. When using it we noticed that for each request the same boilerplate code is generated again and again:

- Creating a URL object and opening the connection
- Configuring the HTTP request
- Executing the HTTP request
- Interpretation of the HTTP response
- Converting the HTTP response into a Java object
- Exception handling

When using RestTemplate all these things happen in the background and the developer doesn't have to bother with it.

Starting with Spring 5, the non-blocking and reactive WebClient offers a modern alternative to RestTemplate. WebClient offers support for both synchronous and asynchronous HTTP requests and streaming scenarios. Therefore, RestTemplate will be marked as deprecated in a future version of the Spring Framework and will not contain any new functionalities.

Before we really get started, I would like to take a closer look at the following points of the project setup:

- Used dependencies
- POJO class Employee
- REST web service for testingFor the

RestTemplate demo project we need the following dependencies in our Spring Boot based application:

```
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>
	<dependency>
		<groupId>org.apache.httpcomponents</groupId>
		<artifactId>httpclient</artifactId>
	</dependency>
	<dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<optional>true</optional>
	</dependency>
</dependencies>
```

The Dependency spring-boot-starter-web is a starter for building web applications. This dependency contains the class RestTemplate, the option to publish REST web services and many other web-related things.

As HTTP client API we use Apache HttpComponents for the following examples. Lombok generates e.g. Getter and Setter and helps us to avoid repeating code.

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

Idempotency is a property of HTTP methods.

A request method is considered idempotent if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request. And it's worthwhile to mention that idempotency is about the effect produced on the state of the resource on the server and not about the response status code received by the client.

To illustrate this, consider the DELETE method, which is defined as idempotent. Now consider a client performs a DELETE request to delete a resource from the server. The server processes the request, the resource gets deleted and the server returns 204. Then the client repeats the same DELETE request and, as the resource has already been deleted, the server returns 404.

Despite the different status code received by the client, the effect produced by a single DELETE request is the same effect of multiple DELETE requests to the same URI.

Finally, requests with idempotent methods can be repeated automatically if a communication failure occurs before the client is able to read the server's response. The client knows that repeating the request will have the same intended effect, even if the original request succeeded, though the response might be different.

Method	**Safe**	**Idempotent**
CONNECT	no	no
DELETE	no	yes
GET	yes	yes
HEAD	yes	yes
OPTION	yes	yes
POST	no	no
PUT	no	yes
TRACE	yes	yes

10 – What is DNS Spoofing ? How to prevent ?

The DNS in and of itself has never been secure. Being created in the 1980s when the Internet was a complete novelty, protection had not been a priority in its design. Throughout time, this has led malicious actors to take advantage of this issue and develop elaborate attack techniques that leverage the DNS, such as DNS spoofing.

DNS spoofing is a cyber-attack in which fake data is introduced into the DNS resolver’s cache, which causes the name server to return an incorrect IP address. In other words, these types of attacks exploit vulnerabilities in domain name servers and redirect traffic towards illegitimate websites.

When a recursive resolver sends a request to an authoritative name server, the resolver has no means of checking the response’s validity. The best the resolver can do is check if the response seems to come from the same IP address where the resolver sent the query in the first place. But relying on the

source IP address of response is never a good idea since the source IP address of a DNS response packet can be easily spoofed.

Security-wise, due to the faulty design of the DNS, a resolver can't identify a fake response to one of its queries. This means cybercriminals could easily pose as the authoritative server that was originally queried by the resolver, spoofing a response that seems to come from that authoritative server.

In a nutshell, an attacker could redirect a user to a malicious site without the user noticing it. In a nutshell, DNS spoofing refers to all attacks that attempt to change the DNS records returned to the user and redirect him/her to a malicious website.

While DNS spoofing attacks are undeniably cunning, they can also be prevented with a few additional security measures, as well as advanced solutions. Here are a couple of actionable tips that will help you prevent an incident of this kind in your enterprise and cover all **attack vectors**.

- Set up DNSSEC
- Look for the secure connection symbol
- Regularly apply patches to DNS servers
- Perform thorough DNS traffic-filtering

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 ?

As per the rule followed by REST architecture, RESTful web services do not maintain a client's state on a server. This restriction is known as Statelessness. Therefore the client shoulders the responsibility of passing its context directly to the server. The server, then stores the stores this context for processing client's requests. For example, if a session is maintained by a server it will be identified by session identifier as it has been passed by the client.

RESTful web service is bound to follow this restriction. In this context you must be remembering that in the chapter RESTful – web services- Method, we have already studied about the special feature of web service methods which do not store a single information which is derived from a client.

13 - What is CSRF attack? How to prevent ?

Cross-site Request Forgery (CSRF/XSRF), also known as sea surf or session riding, refers to an attack against authenticated web applications using cookies. The hacker is able to trick the victim into making a request that the victim did not intend to make. Therefore, the hacker abuses the trust that a web application has for the victim's browser. While Cross-site Request Forgery (CSRF) attacks do not provide anything to an attacker with the response returned from the server, a clever attacker could cause a fair amount of damage, especially when paired with well-crafted social engineering attacks on administrative users.

Upon sending an HTTP request the victim's web browser will include the cookie header. Cookies are typically used to store a user's session identifier so that the user does not have to enter their login credentials for each request, which would obviously be impractical. If the users authenticated session is stored in a browser session cookie that is still valid (a browser window/tab does not necessarily need to be open), and if the application is vulnerable to Cross-site Request Forgery (CSRF), then the

attacker can validates CSRF to launch any desired malicious requests against the website and the server-side code is unable to distinguish whether these are legitimate requests.

CSRF attacks may, for example, be used to compromise online banking by forcing the victim to make an operation involving their bank account. CSRF can also evaluate as Cross-site Scripting (XSS).Therefore, CSRF should be treated as a serious web application security issue, even if it is not currently included in the OWASP Top 10.

Generally there are two primary approaches to protect Cross-site Request Forgery (CSRF) –

- synchronizing the cookie with an anti-CSRF token that has already been provided to the browser.
- Preventing the browser from sending cookies to the web application in the first place.

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

Core components of HTTP requests are:

- HTTP Version – Indicates version
- Request Body – Represents message content
- Request Header – Contains metadata, such as cache settings and client type, for the HTTP request message
- URI – Identifies the resource on the server
- Verb – Indicates HTTP methods such as GET, POST, and PUT

HTTP response has the following core components:

- HTTP Version – Indicates the present version of HTTP
- Response Body – Represents the response message content
- Response Header – Consists of metadata, like content length and server length, for the HTTP response message
- Status/Response Code – Indicates the server status for the requested resource