**HW#5 GULNİSA ASLAN gulnisaslan@gmail.com/@hotmail.com**

## **1** – What are Authentication and Authorization ?

| Authentication | Authorization |
|---|---|
| In authentication process, the identity of users are checked for providing the access to the system. | While in authorization process, person's or user's authorities are checked for accessing the resources. |
| In authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process |
| It needs usually user's login details | While it needs user's privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines **What permission do user have?** |

## **2** - What is Hashing in Spring Security ?

Hashing is the process of generating a string, or hash, from a given message using a mathematical function known as a cryptographic hash function.

## **3** - What is Salting and why do we use the process of Salting ?

In cryptography, a **salt** is random data that is used as an additional input to a one-way function that hashes data,
a password or passphrase. Salts are used to safeguard passwords in storage

## **4** - What is “intercept-url” pattern ?

Most web applications using Spring Security only have a couple of intercept-urls because they only have very basic security requirements. You need to have unauthenticated access to the login and login-error screens and usually some aspect of the public site, so that can be a few URL patterns.

## 5 - What do you mean by session management in Spring Security ?

HTTP session related functonality is handled by a combination of the SessionManagementFilter and the SessionAuthenticationStrategy interface, which the filter delegates to. Typical usage includes session-fixation protection attack prevention, detection of session timeouts and restrictions on how many sessions an authenticated user may have open concurrently.

## **6** – Why we need Exception Handling ?

The worst situation is if your application crashes while the user is doing any important work, especially if their data is lost. To make the user interface robust, it is important to handle Java exceptions to prevent the application from unexpectedly crashing and losing data. There can be many causes for a sudden crash of the system, such as incorrect or unexpected data input. For example, if we try to add two users with duplicate IDs to the database, we should throw an exception since the action would affect database integrity.

## **7** - Explain what is AuthenticationManager in Spring security ?

The main job of this component is to delegate the authenticate() call to the correct AuthenticationProvider. An application can have multiple AuthenticationProvider s, few of which are DaoAuthenticationProvider, LdapAuthenticationProvider, OpenIDAuthenticationProvider, etc. ). The Authentication Manager decides which Authentication Provider to delegate the call to by calling the “supports()” method on every available AuthenticationProvider. If the “supports()” method returns true then that AuthenticationProvider supports the Authentication type and is used to perform authentication

## **8** - What is Spring Security Filter Chain ?

A Filter is a plain object that can be used to intercept the HTTP requests and responses coming in to and going out from your application. We can use these filter objects to perform pre-processing before HTTP requests reach the controllers. Also we can perform post processing after HTTP requests are sent out by the controllers.

## **9** – What are the differences between OAuth2 and JWT ?

**JWT (JSON Web Tokens)**- It is just a token format. JWT tokens are JSON encoded data structures contains information about issuer, subject (claims), expiration time . It is signed for tamper proof and authenticity and it can be encrypted to protect the token information using symmetric or asymmetric approach. JWT is simpler than SAML 1.1/2.0 and supported by all devices and it is more powerful than SWT(Simple Web Token).

**OAuth2** - OAuth2 solve a problem that user wants to access the data using client software like browse based web apps, native mobile apps or desktop apps. OAuth2 is just for authorization, client software can be authorized to access the resources on-behalf of end user using access token.

## **10** - What is method security and why do we need it ?

In simple terms, Spring method security allows us to support / add authorization supports at the method level. On a high level, we can configure which roles are allowed to access what method within the same service class

## **11** – What Proxy means and how and where can be used ?

Proxy provides tom connect computer a way safe internet.

If blocked a website needs quickly surfing and If you try to hide our datas from no body.

## **12** – Waht is Wrapper Class and where can be used ?

The **wrapper class in Java** provides the mechanism *to convert primitive into object and object into primitive*.

Since J2SE 5.0, **autoboxing** and **unboxing** feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

# Use of Wrapper classes in Java

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

- **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
- **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
- **Synchronization:** Java synchronization works with objects in Multithreading.
- **java.util package:** The java.util package provides the utility classes to deal with objects.
- **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

## **13** – What is SSL ? What is TLS ? What is the difference ? How can we use them ?

SSL stands for Secure Sockets Layer and, in short, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals from reading and modifying any information transferred, including potential personal details. The two systems can be a server and a client (for example, a shopping website and browser) or server to server (for example, an application with personal identifiable information or with payroll information).

TLS (Transport Layer Security) is just an updated, more secure, version of SSL. We still refer to our security certificates as SSL because it is a more commonly used term, but when you are buying SSL from DigiCert you are actually buying the most up to date TLS certificates with the option of ECC, RSA or DSA encryption.

## **14** - Why do you need the intercept-url ?

Yes, with intercept-url you specify the urls which have to be secured, and also the type of the access (like the role which is necessary etc.)

| Expression | Description |
|---|---|
| `hasRole([role])` | Returns `true` if the current principal has the specified role. |
| `hasAnyRole([role1,role2])` | Returns `true` if the current principal has any of the supplied roles (given as a comma-separated list of strings) |
| `principal` | Allows direct access to the principal object representing the current user |
| `authentication` | Allows direct access to the current `Authentication` object obtained from the `SecurityContext` |
| `permitAll` | Always evaluates to `true` |
| `denyAll` | Always evaluates to `false` |
| `isAnonymous()` | Returns `true` if the current principal is an anonymous user |
| `isRememberMe()` | Returns `true` if the current principal is a remember-me user |
| `isAuthenticated()` | Returns `true` if the user is not anonymous |
| `isFullyAuthenticated()` | Returns `true` if the user is not an anonymous or a remember-me user |