Anything on the web like web applications is exposed to the open world of the Internet, they are vulnerable to security threats. Only authorized personnel should have access to Web pages, files, and other classified resources. There are often several layers of security, such as firewalls, proxy servers, JVM security, etc., but, if access is to be controlled, application-level security should also be applied. Hence, Spring Security, a part of the Spring Framework, provides a means for applying a layer of security to Java applications.

What is Spring Security?

Spring Security is essentially just a bunch of servlet filters that enable Java applications to include authentication and

authorization functionality. It is one of the most powerful, and highly customizable access-control frameworks (security framework) that provide authentication, authorization, and other security features for Java EE (Enterprise edition) based enterprise applications. The real power of Spring Security lies in its ability to be extended to meet custom needs. Its main responsibility is to authenticate and authorize incoming requests for accessing any resource, including rest API endpoints, MVC (Model-View-Controller) URLs, static resources, etc.

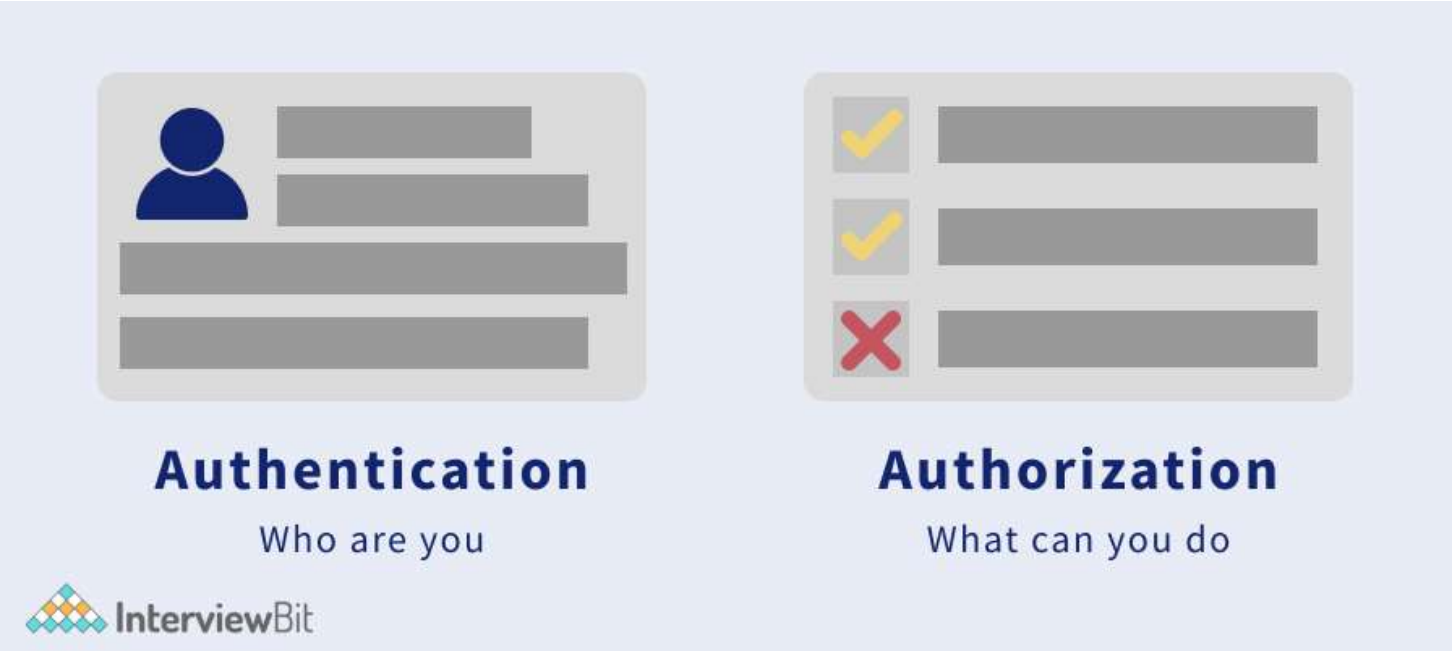
Spring Security Interview Questions for Freshers

1. What are some essential features of Spring Security?

Some essential **features** of Spring Security include:

- Supports authentication and authorization in a flexible and comprehensive manner.
- Detection and prevention of attacks including session fixation, clickjacking, cross-site request forgery, etc.
- Integrate with Servlet API.
- Offers optional integration with Spring Web MVC (Model-View-Controller).
- Java Authentication and Authorization Service (JAAS) is used for authentication purposes.
- Allows Single Sign-On so that users can access multiple applications with just one account (username and password).

2. What is Spring security authentication and authorization?



- **Authentication:** This refers to the process of verifying the identity of the user, using the credentials provided when accessing certain restricted resources. Two steps are involved in authenticating a user, namely identification and verification. An example is logging into a website with a username and a password. This is like answering the question Who are you?
- **Authorization:** It is the ability to determine a user's authority to perform an action or to view data, assuming they have successfully logged in. This ensures that users can only access the parts of a resource that they are authorized to access. It could be thought of as an answer to the question Can a user do/read this?

3. What do you mean by basic authentication?

RESTful web services can be authenticated in many ways, but the most basic one is basic authentication. For basic authentication, we send a username and password using the HTTP [Authorization] header to enable us to access the resource. Usernames and passwords are encoded using base64 encoding (not encryption) in Basic Authentication. The encoding is not secure since it can be easily decoded.

Syntax:

```
Value = username:password
Encoded Value = base64(Value)
Authorization Value = Basic <Encoded Value>
//Example: Authorization: Basic VGVzdFVzZXI6dGVzdDEyMw==
//Decode it'll give back the original username:password U
```

You can download a PDF version of Spring Security Interview Questions.

Download PDF

4. What do you mean by digest authentication?

RESTful web services can be authenticated in many ways, but advanced authentication methods include digest authentication. It applies a hash function to username, password, HTTP method, and URI in order to send credentials in encrypted form. It generates more complex cryptographic results by using the hashing technique which is not easy to decode.

Syntax:

```
Hash1=MD5(username:realm:password)
Hash2=MD5(method:digestURI)
response=MD5(Hash1:nonce:nonceCount:cnonce:qop:Hash2)
//Example, this got generated by running this example
Authorization: Digest username="TestAdmin", realm="admin-
```

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

As far as security is concerned, session management relates to securing and managing multiple users' sessions against their request. It facilitates secure interactions between a user and a service/application and pertains to a sequence of requests and responses associated with a particular user. Session Management is one of the most critical aspects of Spring security as if sessions are not managed properly, the security of data will suffer. To control HTTP sessions, Spring security uses the following options:

- SessionManagementFilter.
- SessionAuthneticationStrategy

With these two, spring-security can manage the following security session options:

- Session timeouts (amount of time a user can remain inactive on a website before the site ends the session.)
- Concurrent sessions (the number of sessions that an authenticated user can have open at once).
- Session-fixation (an attack that permits an attacker to hijack a valid user session).

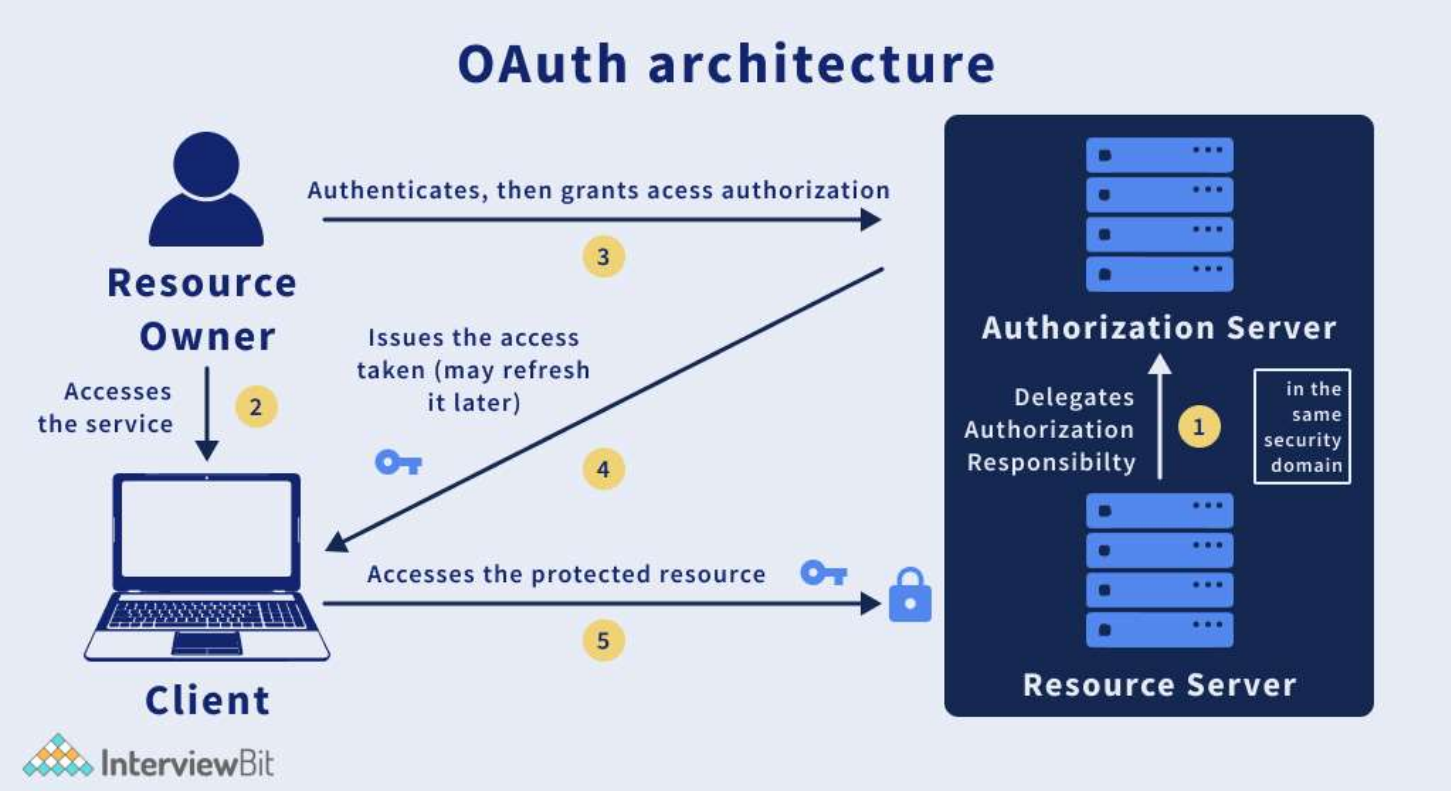
6. Explain SecurityContext and SecurityContext Holder in Spring security.

There are two fundamental classes of Spring Security: SecurityContext and SecurityContextHolder.

- **SecurityContext:** In this, information/data about the currently authenticated user (also known as the principal) is stored. So, in order to obtain a username or any other information about the user, you must first obtain the SecurityContext.
- **SecurityContextHolder:** Retrieving the currently authenticated principal is easiest via a static call to the SecurityContextHolder. As a helper class, it provides access to the security context. By default, it uses a ThreadLocal object to store SecurityContext, so SecurityContext is always accessible to methods in the same thread of execution, even if SecurityContext isn't passed around.

7. Explain spring security OAuth2.

A simple authorization framework, OAuth 2.0, permits client applications to access protected resources via an authorization server. Using it, a client application (third party) can gain limited access to an HTTP service on behalf of the resource owner or on its own behalf.



In OAuth2, four roles are available as shown below:

- **Resource Owner/User:** The owner of a resource, i.e., the individual who holds the rights to that resource.
- **Client:** The application requests an access token (represents a user's permission for the client to access their

data/resources), then accesses the protected resource server after receiving the access token.

- **Authorization Server:** After successfully authenticating the resource owner and obtaining authorization, the server issues access tokens to the client.
- **Resource Server:** It provides access to requested resources. Initially, it validates the access tokens, then it provides authorization.

8. What do you mean by OAuth2 Authorization code grant type?

The term "grant type" in OAuth 2.0 refers to the way an application gets an access token. The authorization code flow is one of several types of grants defined by OAuth 2.0. This grant is used by both web applications and native applications to obtain an access token after a user authorizes the application. As opposed to most other grant types, it requires the application to first launch a browser to begin the process/flow. The process involves the following steps:

- The application opens a browser to direct the user to an OAuth server.
- Upon seeing the authorization prompt, the user approves the application's request.

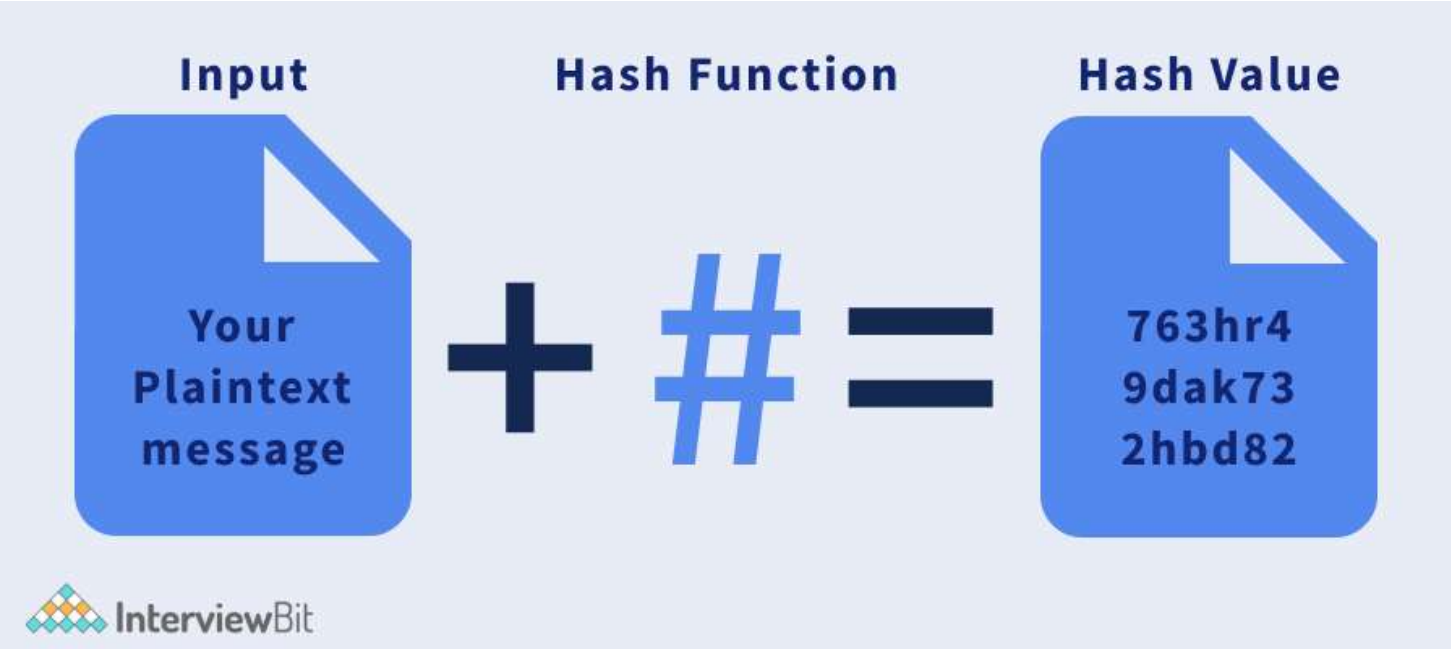
- Upon approval, the user is redirected back to the application with an authorization code in the query string.
- Application exchange authorization codes for access tokens.

9. What is method security and why do we need it?

Simply put, Spring method security lets us add or support authorization at the method level. Spring security checks the authorization of the logged-in user in addition to authentication. Upon login, the ROLE of the user is used to determine which user is authorized to access the resource. When creating a new user in WebSecurityConfig, we can specify his ROLE as well. A security measure applied to a method prevents unauthorized users and only allows authentic users. The purpose of method level security is not to facilitate users who have access but to prevent unauthorized users from performing activities beyond their privileges and roles. Method level security is implemented using AOP (**Aspect-Oriented Programming**).

10. What do you mean by HASHING in spring security?

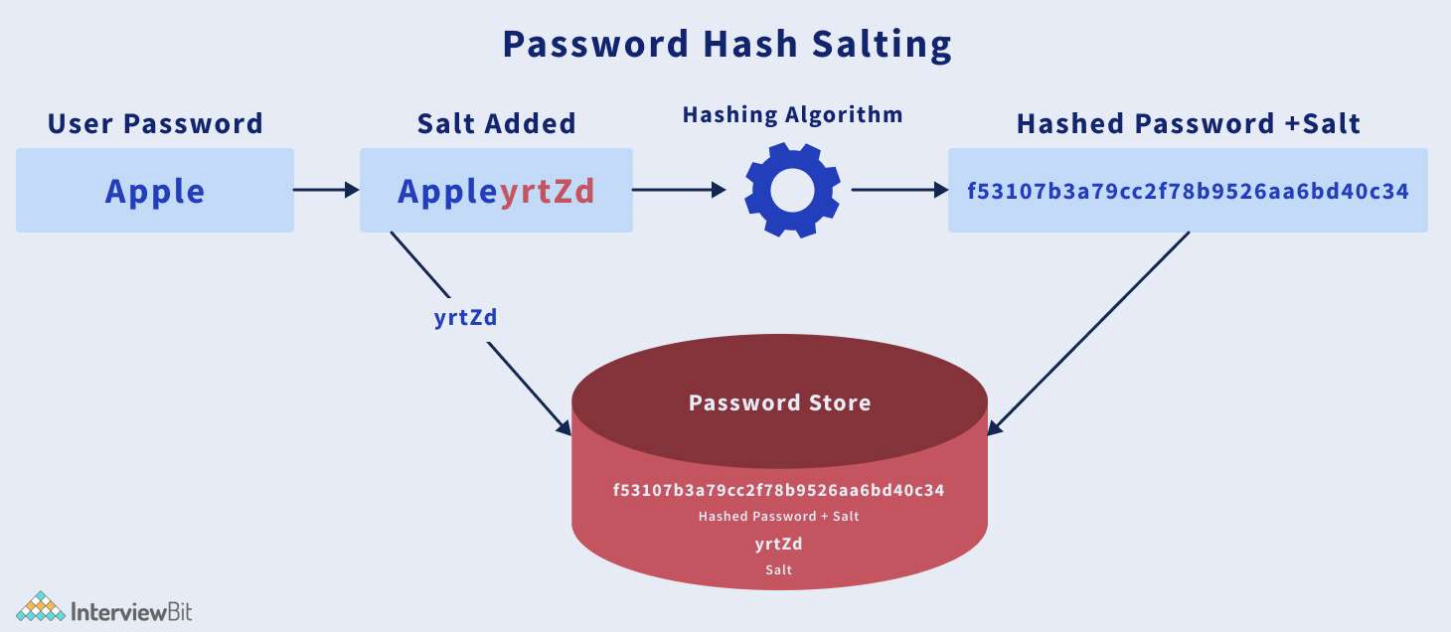
Databases often suffer from security problems when storing passwords. Plain text passwords cannot be stored in your database because then anyone who has access to the database would know the passwords of every user. The solution to this problem is to store encrypted passwords in a database. This is called password hashing.



As part of a general security concept, Hashing involves encoding a string according to the hashing algorithm used. MD4, MD5, SHA (Security Hashing Algorithm) like SHA256 SHA128, etc., are some of the hashing algorithms that can be applied. The hashing method should take the password as input and return a hashed string, which should be stored in a database rather than plain text.

11. Explain salting and its usage.

Spring Security automatically applies salting since version 3.1. Salting is the process of combining random data with a password before password hashing. Salt improves hashing by increasing its uniqueness and complexity without increasing the requirements for users, thereby reducing password attacks. Hashed passwords are then stored in a database, along with salt. Your application will be protected from Dictionary-Attack by using salting. With Salt, you can add an extra string to the password to make it more difficult for hackers to crack it.



12. What is PasswordEncoder?

Password encoding is provided by Spring Security using the PasswordEncoder interface. This interface defines two methods:

- **encode():** It converts a plain password into an encoded form.
- **matches():** It compares an encoded password from the database with a plain password (input by the user) that's been encoded using the same salting and hashing algorithm as the encoded password.

13. Explain AbstractSecurityInterceptor in spring security?

In Spring Security, the AbstractSecurityInterceptor handles the initial authorization of incoming requests. AbstractSecurityInterceptor has two concrete implementations:

- **FilterSecurityInterceptor:** It will authorize all authenticated user requests.
- **MethodSecurityInterceptor:** This is crucial for implementing method-level security. It allows us to secure our program at the method level.

14. Is security a cross-cutting concern?

Spring Security is indeed a cross-cutting concern. Spring security is also using Spring AOP (Aspect Oriented Programming) internally. A cross-cutting concern is one that

applies throughout the whole application and affects it all. Below are some cross-cutting concerns related to the enterprise application.

- Logging and tracing
- Transaction management
- Security
- Caching
- Error handling
- Performance monitoring
- Custom Business Rules

Spring Security Interview Questions for Experienced

15. What is SpEL (Spring Expression Language)?

Spring Framework 3.0 introduced Expression Language/ SpEL. In Spring Expression Language (SpEL), queries and manipulations of object graphs are possible at runtime. You can use it with XML and annotation-based Spring configurations. JSP EL, OGNL, MVEL and JBoss EL are some of the expression languages available, but SpEL provides additional features including string template functionality and method invocation.

Example:

```
import org.springframework.expression.Expression;
import org.springframework.expression.ExpressionParser;
import org.springframework.expression.spel.standard.SpelE
public class WelcomeTest
{
    public static void main(String[] args)
    {
        ExpressionParser parser = new SpelExpressionParse
        Expression exp = parser.parseExpression("'WELCOME
        String message = (String) exp.getValue();
        System.out.println(message);
        //OR
       //System.out.println(parser.parseExpression("'Hell
   }
}
```

Output:

```
WELCOMEtoSPEL
```

16. Name security annotations that are allowed to use SpEL.

Some security annotations that are allowed to use SpEL include:

- @PreAuthorize
- @PreFilter
- @PostAuthorize
- @PostFilter

These provide expression-based access control. In Spring Security, @PreAuthorize is one of the most powerful annotations that allows you to use SpEL. But the old @Secured annotation cannot use it, for example you cannot write @Secured("hasRole('ROLEADMIN')"), but you can do @PreAuthorize("hasRole('ROLEADMIN')").

17. Explain what is AuthenticationManager in Spring security.

A Spring Security component called AuthenticationManager tells "How authentication will happen". Because the how part of this question depends on which authentication provider we are using for our application, an AuthenticationManager contains references to all the AuthenticationProviders. AuthenticationManager is the strategy interface for authentication, which has only one method:

```
public interface AuthenticationManager  {
    Authentication authenticate(Authentication authentica
```

```
    throws AuthenticationException;
}
```

AuthenticationManagers can perform one of three actions in their authenticate() method:

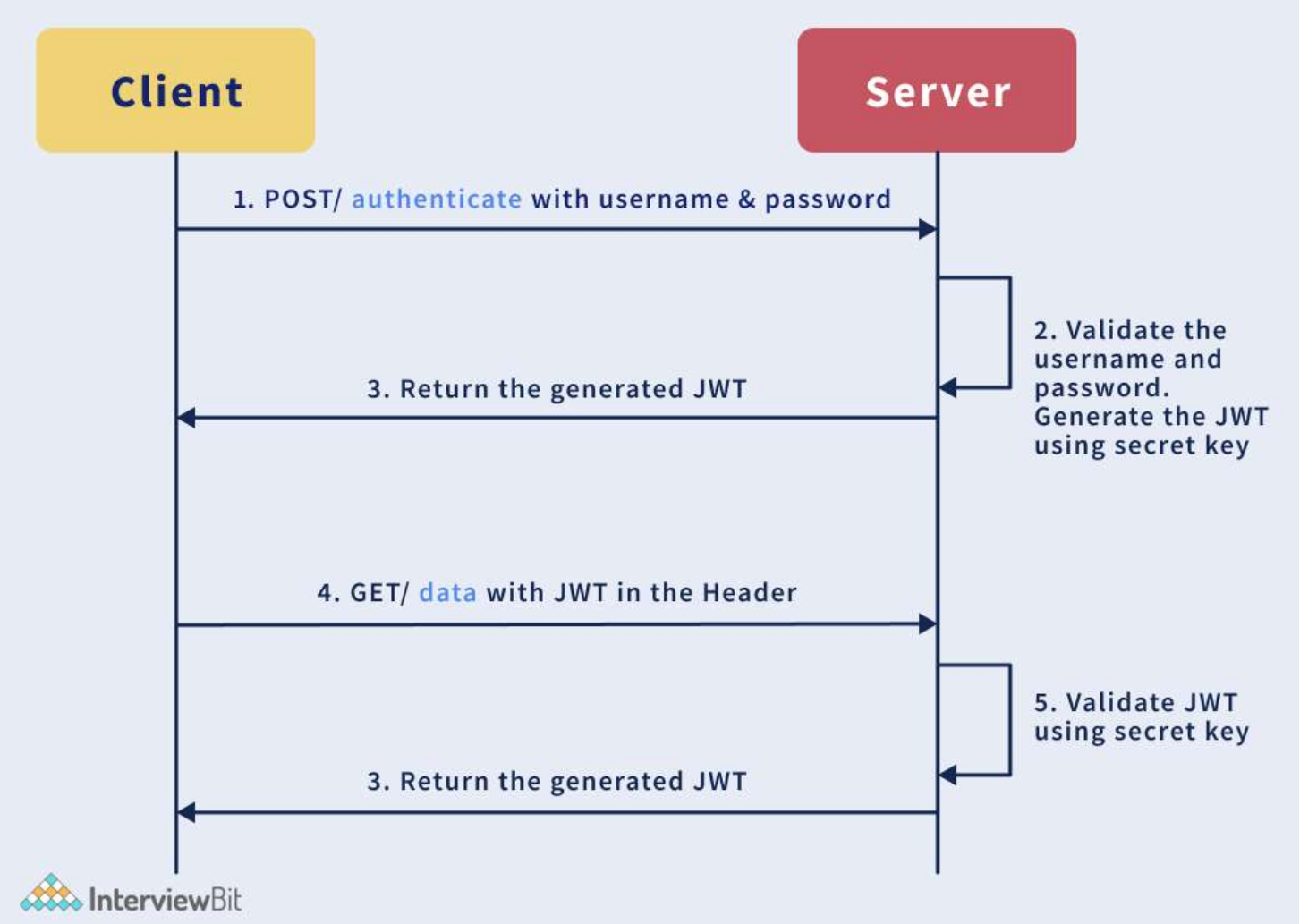
- If it can verify that the input represents a valid principal, it will return an Authentication (normally authenticated=true).
- If the input is believed to represent an invalid principal, it will throw an AuthenticationException.
- If it is unable to decide, it will return null.

18. Explain what is ProviderManager in Spring security.

The default implementation of AuthenticationManager is ProviderManager. It does not handle the authentication request itself, rather delegates the authentication process to a list of configured AuthenticationProviders. Each authenticationprovider in turn is queried to see if it can handle the authentication request.

19. What is JWT?

JWT (JSON Web Tokens) are tokens that are generated by a server upon user authentication in a web application and are then sent to the client (normally a browser). As a result, these tokens are sent on every HTTP request, allowing the server to verify or authenticate the user's identity. This method is used for authorizing transactions or requests between client and server. The use of JWT does not intend to hide data, but rather ensure its authenticity. JWTs are signed and encoded, instead of encrypted. A cryptographic algorithm is used to digitally sign JWTs in order to ensure that they cannot be altered after they are issued. Information contained in the token is signed by the server's private key in order to ensure integrity.



- Login credentials are sent by the user. When successful, JWT tokens (signed by private key/secret key) are sent back by the server to the client.
- The client takes JWT and inserts it in the Authorization header to make data requests for the user.
- Upon receiving the token from the client, the server simply needs to compare the signature sent by the client to the one it generated with its private key/secret key. The token will be valid once the signatures match.

Three parts make up JSON Web Tokens, separated by a dot (.). The first two (the header and the payload) contain Base64-URL

encoded JSON, while the third is a cryptographic signature.

For example:

```
eyJhbGciOfefeiI1NiJ9.eyJuYW1lIjdgdfeENvZGVyIn0.5dlp7GmziL
```

Take a look at each of the sections:

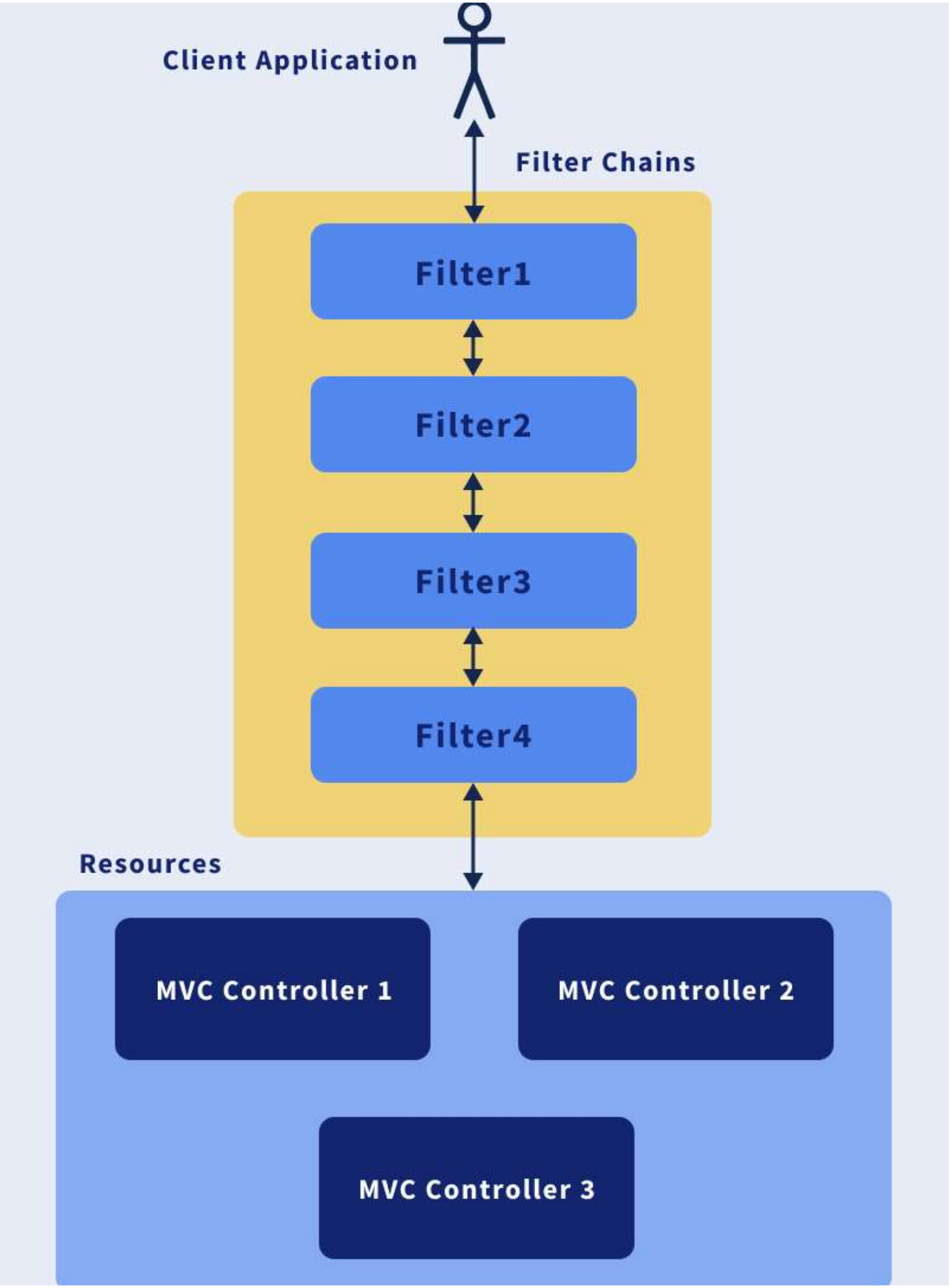
```
eyJhbGciOfefeiI1NiJ9   #header
eyJuYW1lIjdgdfeENvZGVyIn0   #payload
5dlp7GmziL2dfecegse4mtaqv0_xX4oFUuTDh14KuF   #signature
```

20. What is Spring Security Filter Chain?

Spring Security executes most of its security features using the filter chain. Spring security is driven through servlet filters in web applications. A servlet filter intercepts requests before they reach the protected resource (e.g., a Spring controller). As a result, every request for a protected resource will be processed through a spring security filter chain for completing authentication and authorization purposes.

21. Explain how the security filter chain works.

Here's how filters work in a web application:


Client Application
Filter Chains
Filter1
Filter2
Filter3
Filter4
Resources
MVC Controller 1
MVC Controller 2
MVC Controller 3

- **Step 1:** The client first sends a request for a resource (MVC controller). The application container creates a filter chain for handling and processing incoming requests.
- **Step 2:** Each HttpServletRequest passes through the filter chain depending upon the request URI. (We can configure whether the filter chains should be applied to all requests or to the specific request URI).
- **Step 3:** For most web applications, filters perform the following functions:
    - Modify or Change the HttpServletRequest/HttpServletResponse before it reaches the Spring MVC controller.
    - Can stop the processing of the request and send a response to the client, such as Servlets not allowing requests to specific URI's.

22. Name some predefined filters used in spring security and write their functions.

Filter chains in Spring Security are very complex and flexible. They use services such as UserDetailsService and AuthenticationManager to accomplish their tasks. It is also important to consider their orders since you might want to

verify their authenticity before authorizing them. A few of the important security filters from Spring's filter chain are listed below in the order they occur:

- **SecurityContextPersistenceFilter:** Stores the SecurityContext contents between HTTP requests. It also clears SecurityContextHolder when a request is finished.
- **ConcurrentSessionFilter:** It is responsible for handling concurrent sessions. Its purpose is to refresh the last modified time of the request's session and to ensure the session hasn't expired.
- **UsernamePasswordAuthenticationFilter:** It's the most popular authentication filter and is the one that's most often customized.
- **ExceptionTranslationFilter:** This filter resides above FilterSecurityInterceptor in the security filter stack. Although it doesn't perform actual security enforcement, it handles exceptions thrown by the security interceptors and returns valid and suitable HTTP responses.
- **FilterSecurityInterceptor:** It is responsible for securing HTTP resources (web URIs), and raising or throwing authentication and authorization exceptions when access is denied.

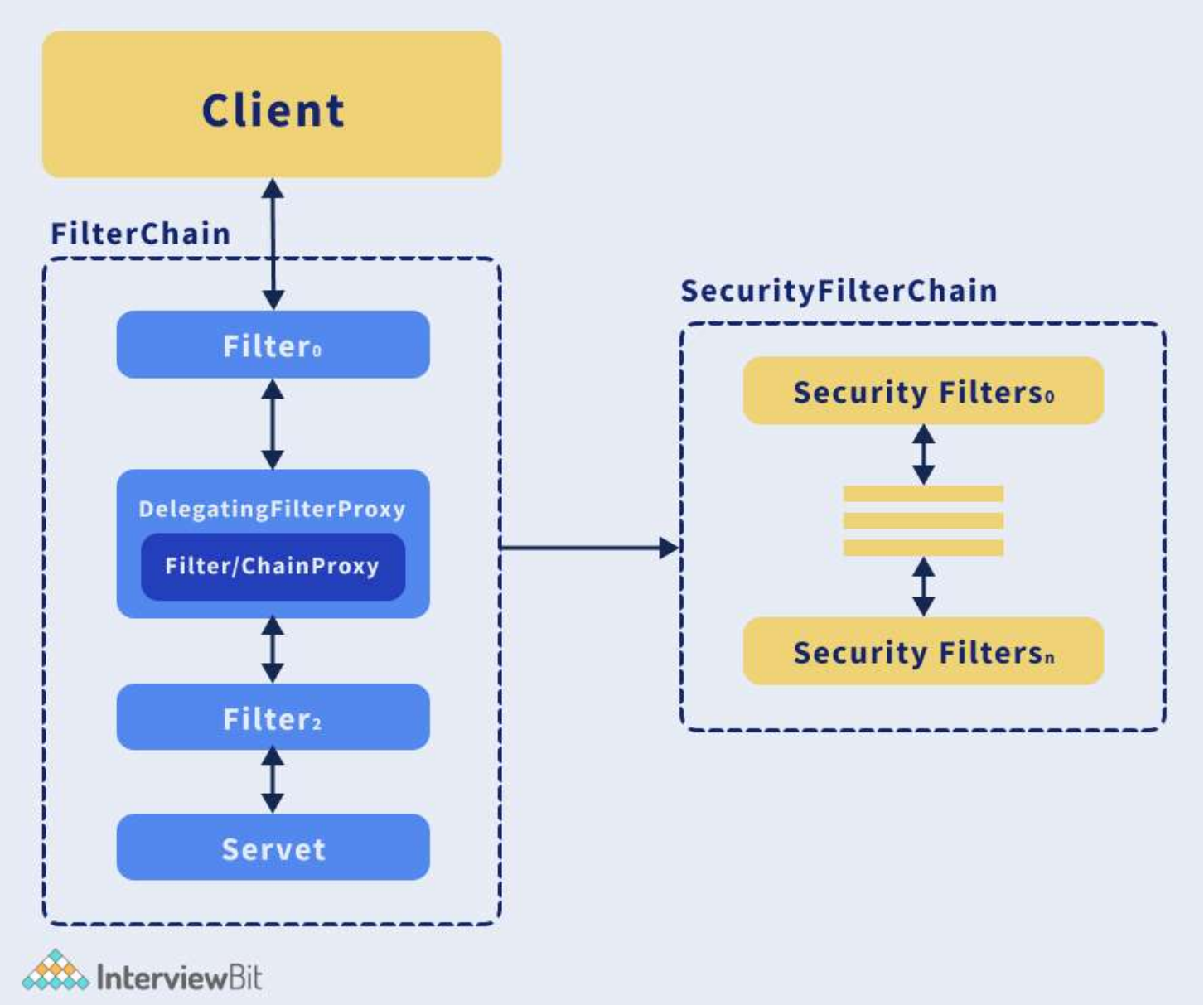
23. What do you mean by principal in Spring security?

The principal is actually the currently logged in user that is using the application. Information/data about the principal (currently authenticated user) is stored in the SecurityContext of the application. As a helper class, SecurityContextHolder provides access to the security context. By default, it uses a ThreadLocal object to store SecurityContext, so SecurityContext is always accessible to methods in the same thread of execution, even if SecurityContext isn't passed around explicitly.

24. Can you explain what is DelegatingFilterProxy in spring security?

A servlet filter must be declared in the web.xml file so that it can be invoked before the request is passed on to the actual Servlet class. DelegatingFilterProxy is a servlet filter embedded in the spring context. It acts as a bridge between web.xml (web application) and the application context (Spring IoC Container). DelegatingFilterProxy is a proxy that delegates an incoming request to a group of filters (which are not managed as spring beans) provided by the Spring web framework. It provides full

access to the Spring context's life cycle machinery and dependency injection.



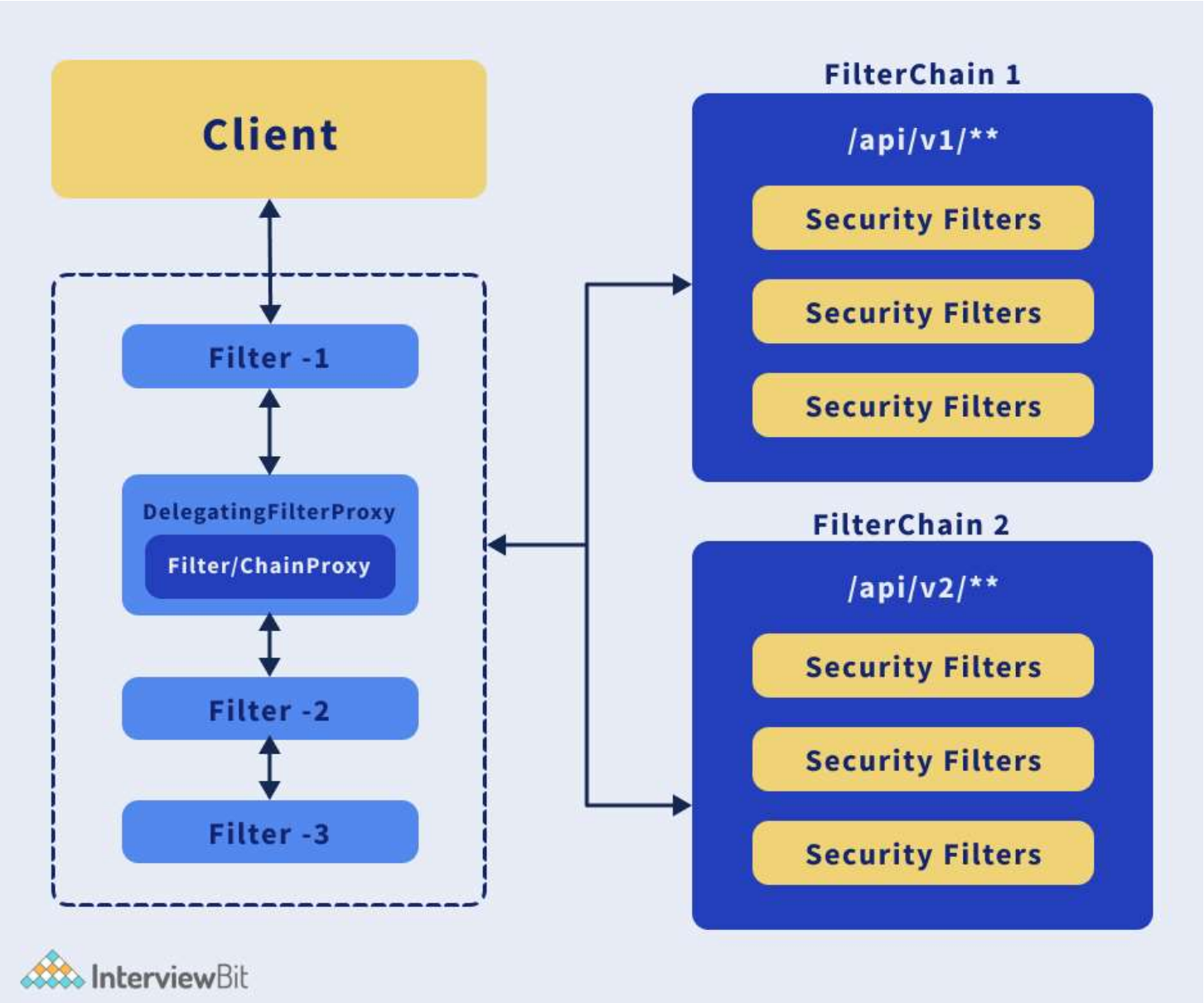
Whenever a request reaches the web application, the proxy ensures that the request is delegated to Spring Security, and, if everything goes smoothly, it will ensure that the request is directed to the right resource within the web application. The following example demonstrates how to configure the DelegatingProxyFilter in web.xml:

```
 <?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <filter>
        <filter-name>springSecurityFilterChain</filter-na
        <filter-class>
            org.springframework.web.filter.DelegatingFilt
        </filter-class>
    </filter>

    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-na
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

25. Can you explain what is FilterChainProxy in spring security?

FilterChainProxy is another servlet filter designed to invoke the appropriate filters based on the path of the incoming request. It contains information about the security filters that make up the security filter chain. It is not directly executed, but it is started by the DelegatingFilterProxy.



26. What is the intercept-url pattern and why do we need it?

<Intercept-url> is used to configure authorizations or access-controls in a Spring Security application. It is used to restrict access to a particular URL. The majority of web applications using Spring Security usually have just a few intercept-URLs because their security needs are quite less.

Example: Basic Spring security using intercept URL

```
<http realm="Example" use-expressions="false">
   <intercept-url pattern="/index.jsp" access="IS_AUTHENT
   <intercept-url pattern="/login.jsp*" access="IS_AUTHEN
   <intercept-url pattern="/admin/*" access="ROLE_ADMIN"/
   <intercept-url pattern="/trade/*" access="ROLE_TRADER"
   <intercept-url pattern="/**" access="ROLE_USER,ROLE_AD
<http-basic/>
```

In this case, index.jsp and admin.jsp can be accessed without authentication. Anything with admin in the URL requires ROLE_ADMIN access, and anything with trade in the URL requires ROLE_TRADER access.

27. Does order matter in the intercept-url pattern? If yes, then in which order should we write it?

Yes, ordering is crucial when we have multiple intercept-URL patterns. Multiple intercept URLs should be written from more specific to less specific. As intercept-URL patterns are processed in the order they appear in a spring security configuration file, the URL must match the right pattern.

28. State the difference between ROLE_USER and ROLE_ANONYMOUS in a spring intercept-url configuration.

- **ROLE_USER:** It has no relevance unless you assign it to your users as soon as they are authenticated. You are responsible for loading the roles (authorities) for each authenticated user.
- **ROLE_ANONYMOUS:** When a configuration uses Spring Security's "anonymous authentication" filter, ROLE_ANONYMOUS is the default role assigned to an anonymous (unauthenticated) user. ROLE_ANONYMOUS is enabled by default. However, it would be better if you used the expression isAnonymous() instead, which has the same meaning.

29. State the difference between @PreAuthorize and @Secured in Spring security.

A variety of security options are available with Spring Framework. This framework offers many useful tools or methods for securing applications. In order to provide method-level security, @Secured and @PreAuthorize are the most commonly used annotations. Compared to @Secured, @PreAuthorize is quite new but becoming well known very

fast. There aren't many differences between @Secured and @PreAuthorize; they're nearly identical. However, @PreAuthorize is considerably more powerful than @Secured.

@PreAuthorize

We can access the methods and properties of SecurityExpress @PreAuthorize.

It can work with Spring EL.

It supports multiple roles in conjunction with AND ope

For example:

```
@PreAuthorize("hasRole('ROLE_role1') and hasRole(
```

Add the following line to spring-security.xml and sprin

30. State the difference between @Secured and @RolesAllowed.

@RolesAllowed: It is a Java standard annotation (JSR250) (i.e., not only spring security). Because this annotation only supports role-based security, it is more limited than the @PreAuthorize annotation. To enable the @RolwesAllowed annotation in your code, add the following line to spring-security.xml and spring boot.

```
XML: <global-method-security jsr250-annotations="enabled"
```

```
Spring boot: @EnableGlobalMethodSecurity(jsr250Enabled =
```

@Secured: It is a Spring specific annotation. There is more to it than just role-based security. It secures methods implemented by beans (objects whose life-cycle is managed by the Spring IoC). However, Spring Expression Language (SpEL) is not supported for defining security constraints. To enable the @Secured annotation in your code, add the following line to spring-security.xml and spring boot.

```
XML: global-method-security secured-annotations="enabled"
```

```
Spring boot: @EnableGlobalMethodSecurity(securedEnabled=t
```

Conclusion

Spring Security is one of the most popular, powerful, and highly customizable access-control frameworks (security framework) that provide authentication, authorization, and other security features for enterprise applications. In this article, we have compiled a comprehensive list of Spring Security Interview questions, which are typically asked during interviews. In addition to checking your existing Spring Security skills, these questions serve as a good resource for reviewing some important concepts before you appear for an interview. It is suitable for both freshers as well as experienced developers and tech leads.