**Chapter 7 - Spring Security**

Web application security can be grouped under

* Authentication
* Authorization web requests
* Authorizing method calls
* Authorizing access to individual domain object

When securing an application following concepts are very important

1. Principal: This is the user, system or a device that that can perform an action within a system.
2. Credentials: these are the identification key that a principal user to confirm its identity.
3. Authentication: is the process of verifying the validity of a principal’s credentials.
4. Authorization: is a process of verifying if a principal is authorized to perform request action on the application.
5. Secure Item is the term used to describe a application resources that is secured.

Spring is popular because it helps, its modules are completely decoupled. One can change the modules are swap then with other modules to meet specific requirements. Spring implements security as cross cutting concerns, it implemented using AOP (aspect-oriented programming).

In-order to implement security in spring one need to perform the following

1. Declare a security filter
2. Define a spring security context
3. Configure Authentication and Authorization

In spring the security is implemented by chain of security beans , delegatingFilterProxy bean delegates the calls to a list of chained security filter beans and acts as an interceptor for secured requests. These chained proxy filters have the following responsibilities , these filter are replaceable at runtime.

1. Driving authentication
2. Enforcing authorization
3. Managing logouts
4. Maintaining security context in httpSession

**What is the effect of annotating a method with @Secured("ROLE\_ADMIN")?**

in a Spring secured application, where @EnableGlobalMethodSecurity(securedEnabled = true) is used on a configuration class, it causes the class containing the method to be wrapped in a secure proxy to restrict access only to users with the ROLE\_ADMIN