SPRING SECURITY

Session 5

OBJECTIVES

- Explain the Spring Security framework and Java Configuration
- Explain how to use Spring Security concepts in a Spring Web application
- Describe how to use validation in Spring-supported Web applications

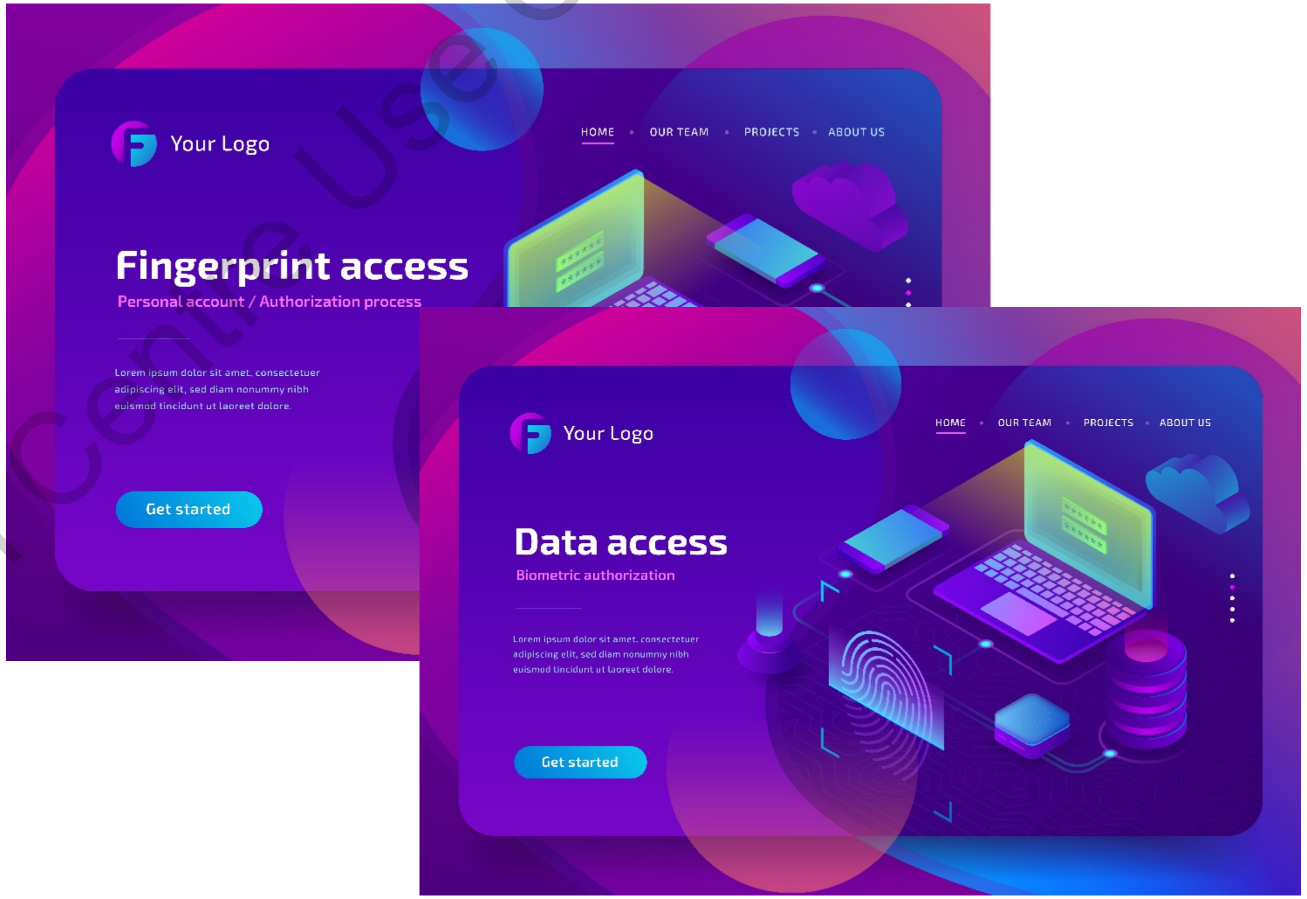
SPRING SECURITY – MAJOR OPERATIONS (1-3)

- Spring Security is a powerful and highly customizable Java/Java EE framework.
- Spring Security framework provides the following to secure Spring-based enterprise applications:
  - Authentication
  - Authorization

Secure Login to a Web Application

SPRING SECURITY – MAJOR OPERATIONS (2-3)

- **Authentication:** Process of validating a user, device, or system for who they claim to be.
- **Authorization:** Process of deciding whether a user, device, or system is permitted to perform an action within the application.


Your Logo
HOME
OUR TEAM
PROJECTS
ABOUT US
Fingerprint access
Personal account / Authorization process
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore.
Get started
Your Logo
HOME
OUR TEAM
PROJECTS
ABOUT US
Data access
Biometric authorization
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore.
Get started

Authentication and Authorization Processes

SPRING SECURITY – MAJOR OPERATIONS (3-3)

- Spring Security supports a wide range of authentication models and authorization capabilities.
- Spring Security provides its own set of authentication features.
- It also supports authentication integration with various technologies, such as HTTP BASIC authentication headers, Lightweight Directory Access Protocol (LDAP), and so on.

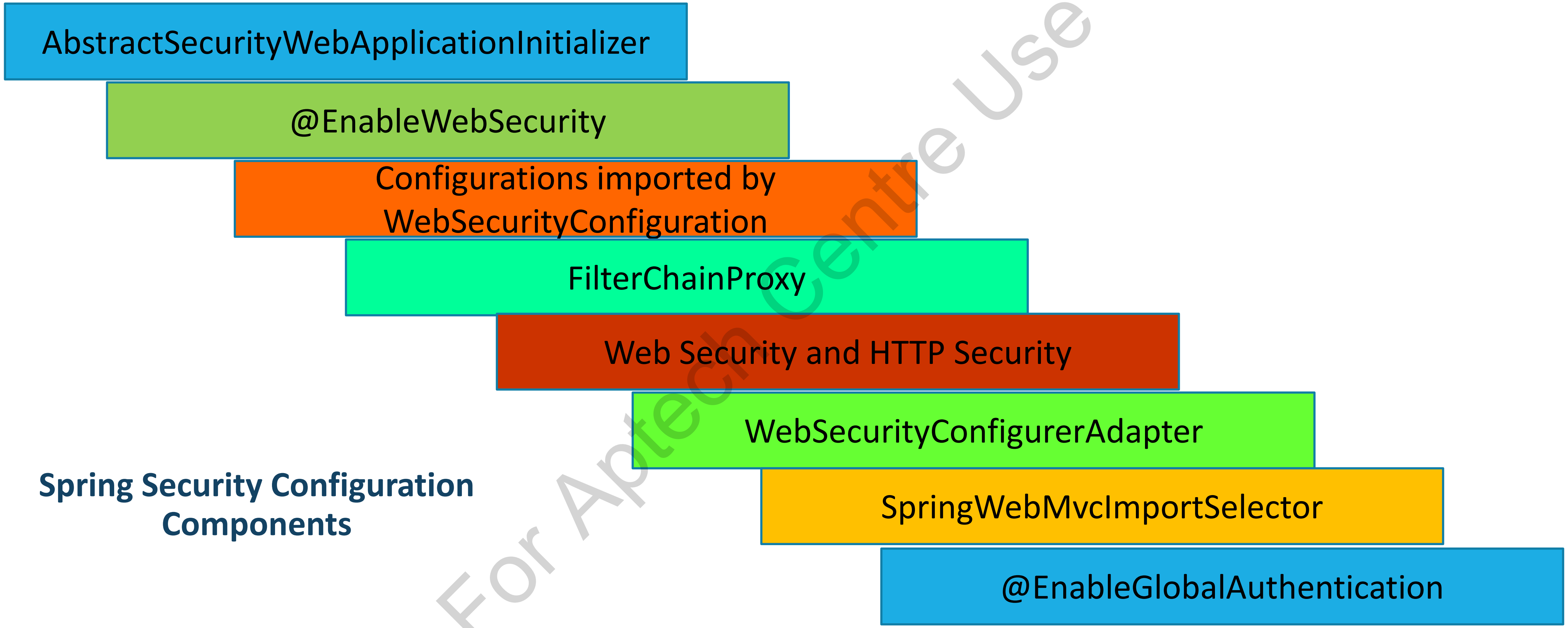
Authorizing Web request
(servlet specification Web pattern security)

Authorizing whether methods can be invoked
(EJB container managed security)

Authorizing access to individual domain object instances
(file system security)

Spring Security Authorization Capabilities

SPRING SECURITY CONFIGURATION – SECURITY SETUP


AbstractSecurityWebApplicationInitializer
@EnableWebSecurity
Configurations imported by WebSecurityConfiguration
FilterChainProxy
Web Security and HTTP Security
WebSecurityConfigurerAdapter
SpringWebMvcImportSelector
@EnableGlobalAuthentication

Spring Security Configuration Components

SPRING SECURITY CONFIGURATION – NAMESPACE CONFIGURATION

- A namespace is a separate program region that developers can use to restrict the scope of variables, functions, classes, and so on, within a Java program.

Namespace	Description
aop	Provides the elements that can be used to declare Spring aspects and use them as proxies.
beans	Allows developers to declare and configure beans and establish a connection among them.
context	Provides elements to support dependency injection in a program. Spring containers use them to auto-detect and auto-wire beans.
Mvc	Provides MVC framework support.

Spring Namespaces

JAVA CONFIGURATION

- Spring Security Java Configuration support enables developers to easily configure authentication and authorization security processes without the use of any XML.

Authenticates all the URLs used in the application

Generates a login form and authenticates the user

Allows the user to logout

Provides session fixation protection

Integrates with the following Servlet API methods

Java Configuration Security Features

JAVA CONFIGURATION – AUTHENTICATE AND AUTHORIZE

- The `HTTP.authorizeRequests()` method helps developers to secure URLs in their Web application.
- Selected authentication model can be configured via the `<authentication-manager></authentication-manager>` tag and the `AuthenticationProvider` interface in the Web application.

LOGGING INTO WEB APPLICATION

- Use the following code to configure the login page for their Web application:

```
<form-login login-page='/login.jsp' default-target-url='/home.jsp'
always-use-default-target='true' />
```

USER'S AUTHENTICATION

- The `Authentication` interface represents the token for an authentication request.
- This interface extends the `Principal` and `Serializable` super interfaces.

AbstractAuthenticationToken
AnonymousAuthenticationToken
CasAssertionAuthenticationToken
CasAuthenticationToken
JaasAuthenticationToken
OpenIDAuthenticationToken
PreAuthenticatedAuthentication
RunAsUserToken
TestingAuthenticationToken
UsernamePasswordAuthenticationToken

Authentication Interface Classes

HTTP SECURITY

- HTTP Security is a default security configuration provided by Spring Security framework.
- Following code snippet represents the default HTTP Security configuration:

```
protected void configure(HttpSecurity http) throws Exception {
      http
            .authorizeRequests()
                  .anyRequest().authenticated()
                  .and()
            .formLogin()
                  .and()
            .httpBasic();
}
```

HANDLING LOGOUTS

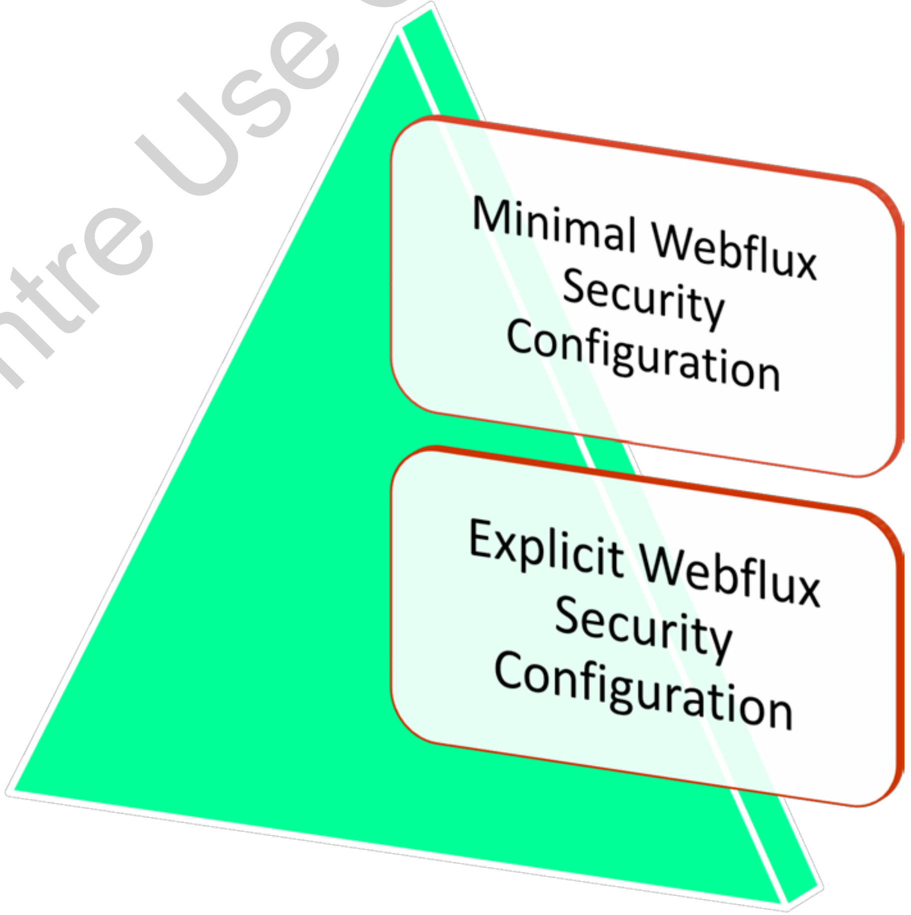
- Use the `WebSecurityConfigurerAdapter` interface to apply default logout capabilities.
- Logout capabilities can be customized as per the user's requirements.

Invalidates the HTTP Session	Clears RememberMe authentication
Clears the SecurityContextHolder	Redirect the user to the required page. By default, it is either login or logout page

WebSecurityConfigurerAdapter Interface's Tasks

WEBFLUX SECURITY

- Webflux module contains support for:
  - ❑ Reactive HTTP and WebSocket clients.
  - ❑ Reactive server Web applications, such as REST, HTML browser, and WebSocket style interactions.


Minimal Webflux Security Configuration
Explicit Webflux Security Configuration

Custom Webflux Configurations

OAUTH 2.0 LOGIN AND MULTIPLE HTTP SECURITY

OAuth 2.0

- Industry-standard protocol that provides specific authorization flows, and used for applications, mobile devices, and living room devices.
- With Spring framework, it provides the capability to login to an application by using existing account at an OAuth 2.0 pre-defined Provider, such as GitHub.

Multiple HTTP Security

- Spring Security framework allows to configure multiple HTTP Security instances by extending the `WebSecurityConfigurationAdapter` multiple times.

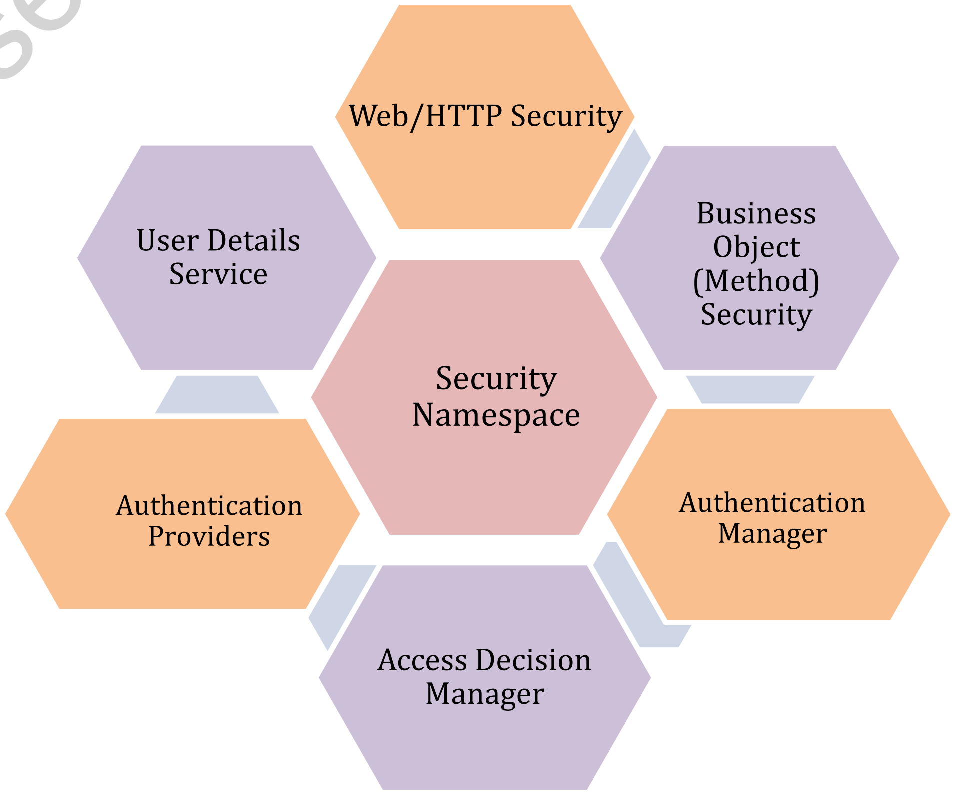
METHOD SECURITY

- Spring Security provides ways to add authorization semantics to service layer methods.
- Following code snippet shows how to use a custom `MethodSecurityExpressionHandler`:

```
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class MethodSecurityConfig extends GlobalMethodSecurityConfiguration {
      @Override
      protected MethodSecurityExpressionHandler createExpressionHandler() {
            // ... create and return custom MethodSecurityExpressionHandler ...
            return expressionHandler;
      }
}
```

SECURITY NAMESPACE CONFIGURATION

- Security Namespace configuration allows to improve Spring beans application context syntax by providing several shortcuts to hide complexity of the framework.



Types of Security Namespaces

VALIDATION AND BEAN VALIDATION

Validation

- Validation means ensuring that the application or software is doing what it is intended to do, and the best way to do this is to accept user data in a Web application.
- Can validate the user input at client-side as well as at server-side, and by default, Spring framework supports JSR-303 specification for bean validation.

Bean Validation

- The Bean Validation specification defines a framework for declared constraints on JavaBean classes, fields, and properties.
- Few standard JSR annotations include, @NotNull, @AssertTrue, @Min, @Max, @Email, and so on.

USING SPRING VALIDATOR INTERFACE

- Spring provides a flexible and extensible mechanism known as `Validator` interface for validation.
- Primarily used to validate application-specific command objects.
- The Validator (`org.springframework.validation.Validator`) interface provides the following two methods:
  - **`supports(Class)`**, which refers to the class or instance of a class this validator supports.
  - **`validate(Object, org.springframework.validation.Errors)`**, which refers to the object being validated that reports the validation error.

RESOLVING CODES TO ERROR MESSAGE

- When a validation is executed, Spring returns error messages in the `BinderResult` object of the `BinderResult` interface in Controller.

ResourceBundle MessageSource

It delegates message handlings to the underlying JDK ResourceBundle instance.

ReloadableResourceBundle MessageSource

It reloads message sources that are changed during runtime and can read XML properties.

BinderResult Interface Classes

BEAN MANIPULATION AND BEAN WRAPPER

- Bean manipulation is the process of getting and setting values from Java beans.
- The `org.springframework.beans` package contains interfaces and classes for manipulating beans.
- BeanWrapper creates a wrapper/envelope which contains an object and helps to transfer that object from source to destination without knowing about the object.

SUMMARY

- Spring Security framework focuses on authentication and authorization and supports validation.
- Authentication is the process of validating a user's, device's, or some other system's identity.
- Authorization is the process of deciding permissions and access rights to an authenticated user, device or a system.
- Spring Security Java Configuration support enables developers to easily configure authentication and authorization security processes without XML.
- HTTP Security is a default security configuration provided by Spring Security framework.
- Webflux is a Spring Framework module that contains support for reactive HTTP and WebSocket clients as well as for reactive server Web applications.