Spring Security

Spring Security Model

- Spring security defines framework for security
- Implemented using servlet filters in the background
- Two methods of securing a web app:
    - declarative and programmatic

Spring Security with Servlet Filters

1. Servlet Filters are used to per-process/ post process web requests
2. Servlet Filters can route web requests based on security logic
3. Spring provides a bulk of security functionality with servlet filters.

Spring Security Overview
SS-SpringSecurity Overview - 2:15

Web Browser -------- Spring Security Filters ----------> Protected web Resource
<------- ----------

	---->	My app	----> user
	<----	Security	<---- passwords
		Configuration	roles

Spring Security in Action
SS-SpringSecurity Overview - 2:50

Security Concepts

1. Authentication
    a. Check user id and password with credentials stored in app/db

2. Authorization
    a. Check to see if the user has an authorized role

Declarative Security
Define application's security constraints in configuration

1. All Java config (@Configuration, No XML)
2. Or Spring XML config

Provides separation of concerns between application code and security

Programmatic Security
Spring Security Provides an API for custom application coding
Provides greater customization for specific app requirements

Different Login Methods

- HTTP basic authentication
  SS video 1 - 5:56
  Built in Dialog from Browser

- Default Login Form
  - Spring security provides a default login form.

  SS video 1 - 6: 32

- Custom Login Form
  - HTML + CSS + etc

  SS - video 1 - 6: 55

Authentication and Authorization

- In-memory
- JDBC
- LDAP
- Custom/Pluggable
- others...

Spring Security Demo

SS - video 2 - 00:35

Full Flow of our Demo APP

Spring Security Example

1. Generate project using spring initializr
    a. Dependencies
        i. Web
        ii. Security
        iii. actuator

2. Import this project in Eclipse
    a. Inside java.project package
    b. Create a package config
        i. Create the config file - SecurityConfig {It is a java class}

```
public class SpringSecurity {

}
```

            a. Add @Configuration Notation

```
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringSecurity {

}
```

            b. Extends WebSecurityConfigurerAdapter

```
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigur
erAdapter;

@Configuration
public class SpringSecurity extends WebSecurityConfigurerAdapter{

}
```

c. Add @EnableWebSecurity

```
@EnableWebSecurity
@Configuration
public class SpringSecurity extends WebSecurityConfigurerAdapter{

}
```

d. Override few methods

i. configure(HttpSecurity httpSecurity){
}

```
@EnableWebSecurity
@Configuration
public class SpringSecurity extends WebSecurityConfigurerAdapter{

    @Override
    protected void configure(HttpSecurity httpSecurity) {


    }

}
```

e. Give implementation of configure method

```
  @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception{

    httpSecurity
            .authorizeRequests()
            .anyRequest()
            .permitAll()
            .and().httpBasic();

    httpSecurity.csrf().disable();

    }
```

3. Create Some REST points
    a. Example

```
@RestController
@RequestMapping("/rest/hello")
public class HelloRestController {

        @GetMapping
        public String hello() {
        return "Hello from HelloRest Controller";
        }
}
```

4. Run the Application
    a. localhost:8080/rest/hello
    b. You can see the output

5. **Right Now there is no security**

6. For applying security - **InMemoryAuthentication**
    a. Override a method configure

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        auth.inMemoryAuthentication()
        .withUser("{noop}neerajvyas").password("test1").and()
        .withUser("{noop}deepakvyas").password("test2");
}
```

    b. Change in configure HttpSecurity
        permitAll() → **fullyAuthenticated()**

```
@Override
    protected void configure(HttpSecurity httpSecurity) throws Exception{

    httpSecurity
            .authorizeRequests()
            .anyRequest()
            .fullyAuthenticated()
            .and().httpBasic();
```

```
httpSecurity.csrf().disable();

}
```

c.

7. s