Day 1 — Introduction to Spring Security & Project Setup

Goal of the Day

Understand what Spring Security is, why it's used, and set up a base Spring Boot project ready for implementing security in the coming days.

---

1. What is Spring Security?

Spring Security is a powerful and customizable authentication and access-control framework for Java applications, part of the Spring ecosystem.

It handles:

- **Authentication** → Verifying *who* you are.
- **Authorization** → Checking *what* you can do.
- **Protection** → Preventing attacks like CSRF, session fixation, clickjacking, etc.

Why use it?

- Secure login & logout handling.
- Integration with databases, OAuth2, JWT, LDAP, etc.
- Easy role-based access control.
- Highly customizable.

<!-- Spring Security -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>

<!-- Spring Web -->
<dependency>
<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!-- DevTools (Optional - Hot Reload) -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
<optional>true</optional>
</dependency>

---

2. Today's Deliverables

By the end of today:

- You'll have a **Spring Boot project** with Spring Security dependency added.
- You'll understand the **default behavior** of Spring Security.

---

3. Step-by-Step Work

Step 1 — Create a New Spring Boot Project

Use **Spring Initializr**

- **Project:** Maven
- **Language:** Java
- **Spring Boot Version:** Latest stable (e.g., 3.3.x)
- **Dependencies:**
  - Spring Web
  - Spring Security
  - Spring Boot DevTools (optional, for fast reload)

Folder Structure:

```
src/main/java/com/example/securitydemo
src/main/resources/application.properties
```

Step 2 — Run the Project

1. Open the project in your IDE (IntelliJ/Eclipse/VS Code).
2. Run the main class:

```
@SpringBootApplication
public class SecurityDemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(SecurityDemoApplication.class, args);
    }
}
```

1. Open browser → `http://localhost:8080`

   You'll see a **login page** automatically generated by Spring Security.

Step 3 — Observe Default Behavior

Without writing a single line of security configuration:

- Spring Security **locks down all endpoints**.
- Default login:
  - Username → `user`
  - Password → Generated in console logs.

Example console output:

```
Using generated security password: 1a2b3c4d-xxxx
```

Step 4 — Key Notes for Day 1

- Spring Security auto-applies **Basic Authentication** when no configuration is provided.
- The login form you see is **auto-generated**.
- Tomorrow, we'll **create our own security configuration** and control access.

---

4. Homework

1. Create a simple REST controller:

```
@RestController
public class HomeController {
    @GetMapping("/")
    public String home() {
        return "Welcome to Spring Security!";
    }
}
```

1. Test it → You'll be redirected to the login page before accessing it.
2. Take a screenshot of your login page and post it as "Day 1 Progress."

✅ **End of Day 1 — Base project with default Spring Security is ready.**

Tomorrow: **Customizing authentication & user details.**

created by Pravin Sonwane - https://youtube.com/@programmingwithpravin?si=1FPrRHc6INPAYaRo