8 SPRING SECURITY

Security

- Sobald man spring-boot-starter-security in der Maven pom des Projektes aufnimmt,
  - sind alle HTTP-Endpunkte der Applikation abgesichert.

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

Security

- Out of the Box wird eine User mit namen User und zufälligem Password zu testzwecken erzeugt.

Security Konfiguration

- Wie Authentifizierungs- und Authorisierung detaillieren?
  - Konfigurationen mittels einer Ableitung der Klasse **WebSecurityConfigurerAdapter:**

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

	@Override
	@Bean
	public AuthenticationManager authenticationManagerBean() throws Exception {
		return super.authenticationManagerBean();
	}

	@Override
	public void configure(HttpSecurity http) throws Exception {
		//@formatter:off
		http
			.httpBasic()
			.and()
			.authorizeRequests()
				.antMatchers("/secured").hasAnyRole("USER")
				.anyRequest().permitAll();
		//@formatter:on
	}

	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws Exception {
		//@formatter:off
		auth.inMemoryAuthentication()
			.withUser("habuma").password("password").authorities("ROLE_USER", "ROLE_ADMIN")
			.and()
			.withUser("izzy").password("password").authorities("ROLE_USER");
		//@formatter:on
	}


}
```

Security Konfiguration


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

	@Override
	@Bean
	public AuthenticationManager authenticationManagerBean() throws Exception {
		return super.authenticationManagerBean();
	}

	@Override
	public void configure(HttpSecurity http) throws Exception {
		//@formatter:off
		http
			.httpBasic()
			.and()
			.authorizeRequests()
				.antMatchers("/secured").hasAnyRole("USER")
				.anyRequest().permitAll();
		//@formatter:on
	}

	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws Exception {
		//@formatter:off
		auth.inMemoryAuthentication()
			.withUser("habuma").password("password").authorities("ROLE_USER", "ROLE_ADMIN")
			.and()
			.withUser("izzy").password("password").authorities("ROLE_USER");
		//@formatter:on
	}


}
```

Drittsysteme einbinden

- Neben der Möglichkeit ein inMemoryUserstore zu verwenden,
  - lassen sich Drittsysteme dafür einbinden
    - LDAP
    - …

Drittsysteme einbinden

- Methodensecurity in den Servicekomponenten ist auch möcglich nachdem man dies enabled hat.

```
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class MethodSecurityConfig {
}

@Service
public class GreetingService {
        @PreAuthorize("hasRole('ADMIN')")
        public String adminGreeting() {
                return "Hello, Admin!";
        }
}
```

OAuth2

- Oftmals möchte man für die Securityaufgaben Tokens verwenden.
- OAuth 2 ist eine Spezifikation für dieses Verfahren.
- Ein Endbenutzer (User oder **Resource Owner**) kann mit Hilfe dieses Protokolls einer Anwendung (**Client** oder Relying Party) den Zugriff auf seine Daten erlauben (Autorisierung), die von einem anderen Dienst (**Resource Server**) bereitgestellt werden, ohne geheime Details seiner Zugangsberechtigung (Authentifizierung) dem Client preiszugeben.

OAuth2

- Der Endbenutzer kann so Dritten gestatten, in seinem Namen einen Dienst zu benutzen.
- Typischerweise wird dabei die Übermittlung von Passwörtern an Dritte vermieden.
- Dabei bezieht eine Clientapplikation Berechtigungstickets (Token) von einem **Authorizationserver,** gegenüber den sie sich authentifizieren muss.

OAuth2

- Die **Tokens** besitzen eine Lebensdauer und werden dann bei jedem Request gegen einen **Resourcenserver** (hält die schützenswerten Informationen) mitgeschickt.

- Der **Resourceserver** prüft dann die Validität des Tokens (verbindet sich meist mit dem **Authorizationserver**) und kann daraus Authentifizierung- und Autorisierungsprüfungen vornehmen.

OAuth2

- Um unterschiedliche Anwendungsfälle optimal abdecken zu können, wurden vier Genehmigungsprozesse in OAuth 2 definiert
  - authorization code
  - Implicit
  - resource owner password credentials
  - client credentials