目录

5.7 Holding strategy for SecurityContextHolder

```
@GetMapping("/hello")
public String hello() {
	SecurityContext context = SecurityContextHolder.getContext();
	Authentication a = context.getAuthentication();
	return "Hello, " + a.getName() + "!";
}
```

5.8 Holding strategy for SecurityContextHolder(Easy Way)

```
@GetMapping("/hello")
public String hello(Authentication a) {
	return "Hello, " + a.getName() + "!";
}
```

5.9 Using a holding strategy for asynchronous calls

```
@GetMapping("/bye")
@Async
public void goodbye() {
	SecurityContext context = SecurityContextHolder.getContext();
	String username = context.getAuthentication().getName();
	// do something with the username
}
```

```
@Configuration
@EnableAsync
public class ProjectConfig {
	@Bean
	public InitializingBean initializingBean() {
		return () -> SecurityContextHolder.setStrategyName(
		SecurityContextHolder. MODE_INHERITABLETHREADLOCAL);
	}

}
```

- Being Asynchronized, the get mapping must be always void, return nothing at the end.

- Besides , SecurityContextHolder. MODE_INHERITABLETHREADLOCAL SecurityContextHolder.MODE_GLOBAL; SecurityContextHolder.MODE_THREADLOCAL; are available to us to change different functions.\
- SecurityContextHolder.MODE_GLOBAL allows all threads to access security context together.

```
@GetMapping("ciao")
public String ciao() throws Exception{
	Callable<String> task = ()->{
	SecurityContext context = SecurityContextHolder.getContext();
	return context.getAuthentication().getName();
	};

	ExecutorService e = Executors.newCachedThreadPool();

	try{
	var contextTask = new DelegatingSecurityContextCallable<>(task);
	return "Ciao, "+e.submit(contextTask).get()+"!";
	}finally{
	e.shutdown();
	}
}
```

```
@GetMapping("/hola")
public String hola() throws Exception {
	Callable<String> task = () -> {
	SecurityContext context = SecurityContextHolder.getContext();
	return context.getAuthentication().getName();
	};

	ExecutorService e = Executors.newCachedThreadPool();
	e = new DelegatingSecurityContextExecutorService(e);

	try {
	return "Hola, " + e.submit(task).get() + "!";
	} finally {
	e.shutdown();
	}
}
```

HTTP Basic

The Realm name is used to set the name for the HTTP basic authentication realm for that directory and subdirectories. It is presented to the browser by the server on each request, and the browser knows which stored password to send to the server based on

the combination of site-name and realm-name. Without this mechanism there would be no way to require different passwords for different "areas" of a web site because there would be no differentiating criteria other than site name.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
        http.httpBasic(c -> {
        c.realmName("OTHER");
        });
        http.authorizeRequests().anyRequest().authenticated();
}
```

NOTE: Realmname only shown in header and when the access is unauthorized.

HTTPBasic Success and Failure Response

```
public class CustomEntryPoint implements AuthenticationEntryPoint {
@Override
public void commence(
HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse,
AuthenticationException e)
throws IOException, ServletException {
        httpServletResponse
        .addHeader("message", "Luke, I am your father!");
        httpServletResponse
        .sendError(HttpStatus.UNAUTHORIZED.value());
        }
}
```

```
@Override
protected void configure(HttpSecurity http)
throws Exception {
        http.httpBasic(c -> {
        c.realmName("OTHER");
        c.authenticationEntryPoint(new CustomEntryPoint());
        });

        http.authorizeRequests()
        .anyRequest()
        .authenticated();
}
```

Form Login

```
@Configuration
public class ProjectConfig extends WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http)
        throws Exception {
                http.formLogin();
                http.authorizeRequests().anyRequest().authenticated();
        }
}
```

Note: a default login page is set in spring boot

Access to certain HTML Page using spring boot

```
@Controller
public class HelloController {
	@GetMapping("/home")
	public String home() {
		return "home.html";
	}
}
```

```
@Override
protected void configure(HttpSecurity http)
throws Exception {
	http.formLogin()
	.defaultSuccessUrl("/home", true);
	http.authorizeRequests()
	.anyRequest().authenticated();
}
```

```
@Component
public class CustomAuthenticationSuccessHandler
implements AuthenticationSuccessHandler {
	@Override
	public void onAuthenticationSuccess(
	HttpServletRequest httpServletRequest,
	HttpServletResponse httpServletResponse,
	Authentication authentication)
	throws IOException {
	var authorities = authentication.getAuthorities();
	var auth =
	authorities.stream()
	.filter(a -> a.getAuthority().equals("read"))
	.findFirst();
	if (auth.isPresent()) {
	httpServletResponse
	.sendRedirect("/home");
	} else {
	httpServletResponse
	.sendRedirect("/error");
	}
	}
}
```

```
@Component
public class CustomAuthenticationFailureHandler
implements AuthenticationFailureHandler {
	@Override
	public void onAuthenticationFailure(
	HttpServletRequest httpServletRequest,
	HttpServletResponse httpServletResponse,
	AuthenticationException e) {
	httpServletResponse
```

```
        .setHeader("failed", LocalDateTime.now().toString());
        }
}
```

```
@Configuration
public class ProjectConfig
extends WebSecurityConfigurerAdapter {
        @Autowired
        private CustomAuthenticationSuccessHandler authenticationSuccessHandler;
        @Autowired
        private CustomAuthenticationFailureHandler authenticationFailureHandler;

        @Override
        protected void configure(HttpSecurity http)
        throws Exception {

        http.formLogin()
        .successHandler(authenticationSuccessHandler)
        .failureHandler(authenticationFailureHandler)
        .and()
        .httpBasic();

        http.authorizeRequests()
        .anyRequest().authenticated();
        }
}
```