Spring Fundamentals

Spring Security

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions

Software University

sli.do

#java-web

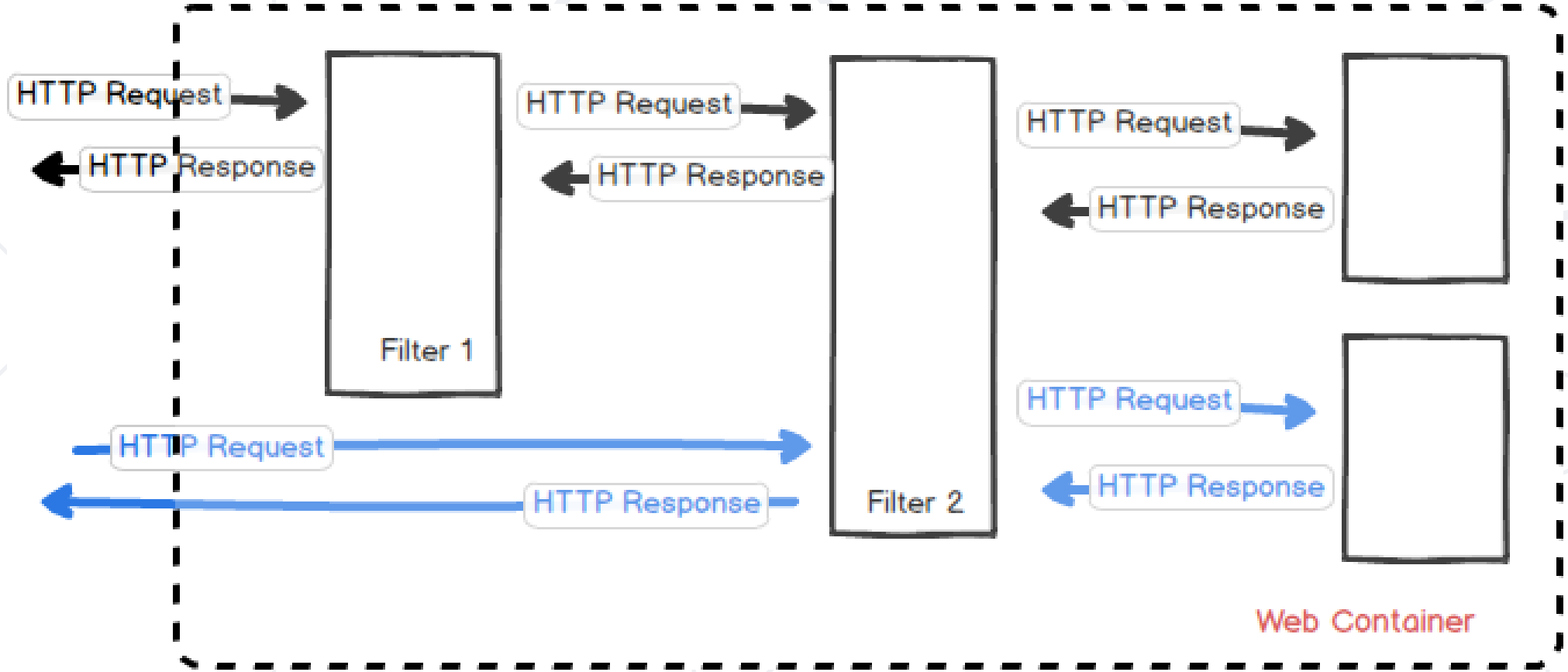
Table of Contents

Filters and Interceptors

Filters

- A filter is an object used to **intercept** the HTTP **requests** and **responses** of your application
- We can perform two operations at two instances:
  - Before sending the **request** to the controller
  - Before sending a **response** to the client

Filters Diagram



Filter Example(1)

GreetingFilter.java

```
@Component
public class GreetingFilter implements Filter {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
FilterChain filterChain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;

        request.getSession().setAttribute('name', 'Pesho');

        filterChain.doFilter(request, response);
    }
}
```

Filter Example(2)

HomeController.java

```
@Controller
public class HomeController {

    @GetMapping('/')
    public ModelAndView index(ModelAndView modelAndView, HttpSession session) {
        modelAndView.setViewName('index');
        modelAndView.addObject('name', session.getAttribute('name'));

        return modelAndView;
    }
}
```

Filter Example(3)

Software University

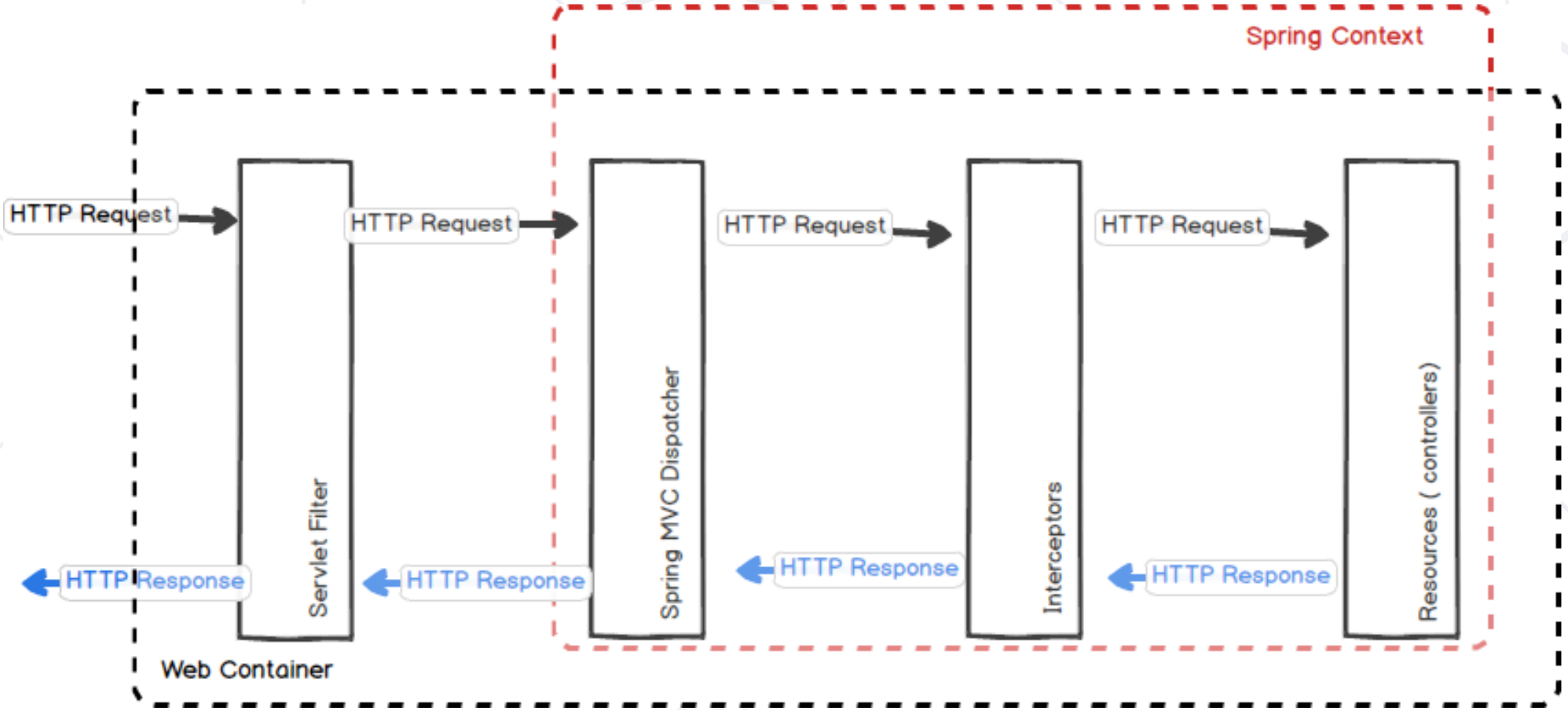
index.html

```
<!DOCTYPE html>
<html lang='en' xmlns='http://www.w3.org/1999/xhtml' xmlns:th='http://www.thymeleaf.org'>
<head>
    <meta charset='UTF-8'>
    <title>Filter Demo</title>
</head>
<body>
    <h1 th:text='|Hello, ${name}!|'></h1>
</body>
</html>
```

Interceptor

- A **Filter** is used in the **web layer only** as it is defined in web.xml. We can not use it out of web context
- While Spring **Interceptors** are defined in the Spring context
- The **interceptor** include **three** main **methods**:
  - **preHandle**: executed before the execution of the target resource
  - **afterCompletion**: executed after the execution of the target resource (after rendering the view)
  - **postHandle**: Intercept the execution of a handler

Interceptor Diagram



Interceptor Example

LogingInterceptor

```
public class LogingInteceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
       FilterChain filterChain, Object handler) throws IOException, ServletException {
              //Log some information ...

              retrun true;
        }
}
```

Register Interceptor in Configuration

- To use interceptors we need to register them

```
@Configuration
public class WebConfiguration implements WebMvcConfigurer {

    private final MyInterceptor myInterceptor;

    public WebConfiguration(MyInterceptor myInterceptor) {
        this.myInterceptor = myInterceptor;
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(myInterceptor);
    }
}
```

Spring Security

What is Spring Security?

Software University

- A **powerful** and **highly customizable** authentication and access-control framework
- It is the de-facto **standard** for securing **Spring-based** applications
- Focuses on providing both **authentication** and **authorization** to Java applications

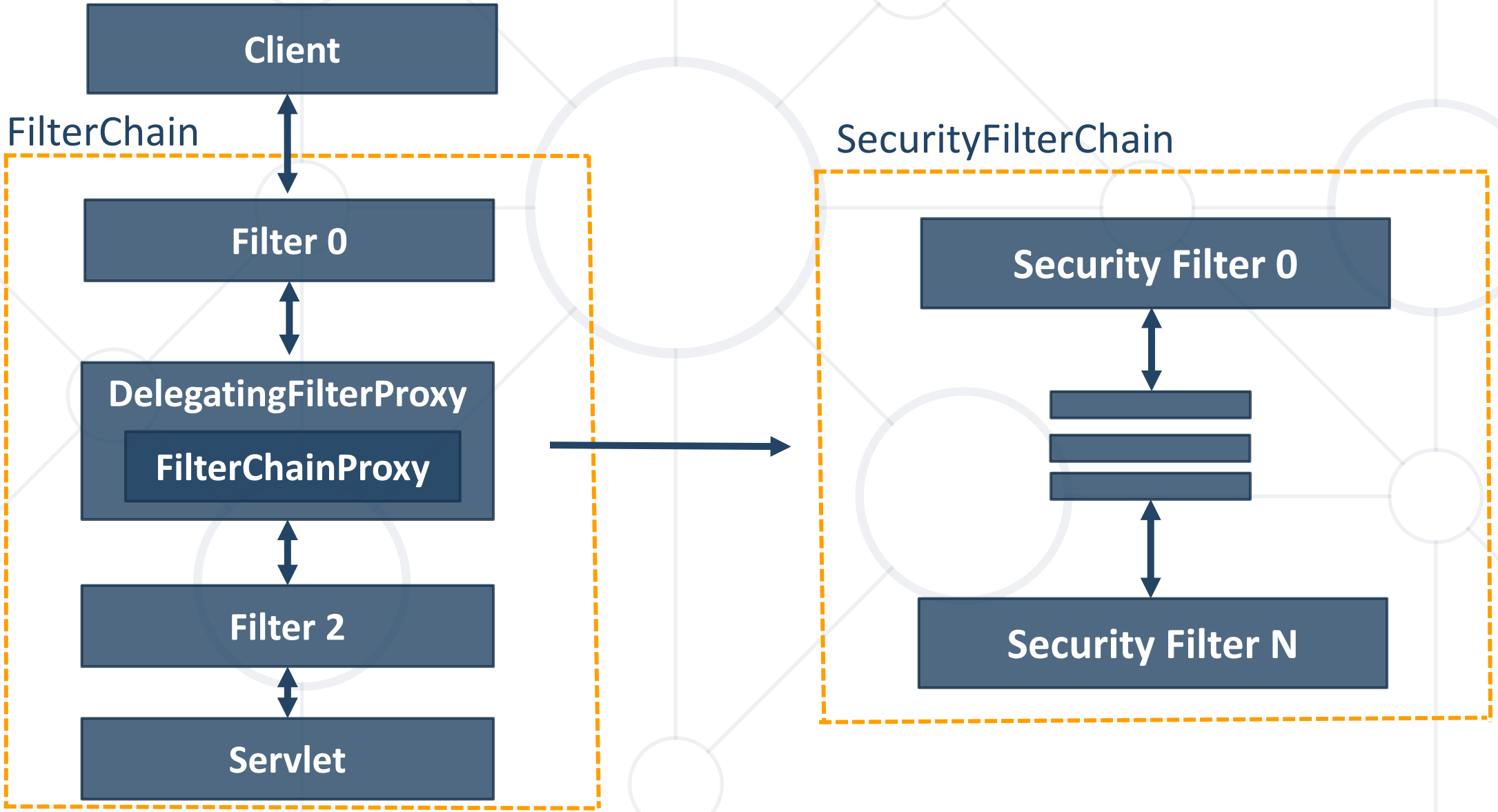
Spring Security

- **Authentication**
  - Who is logged in

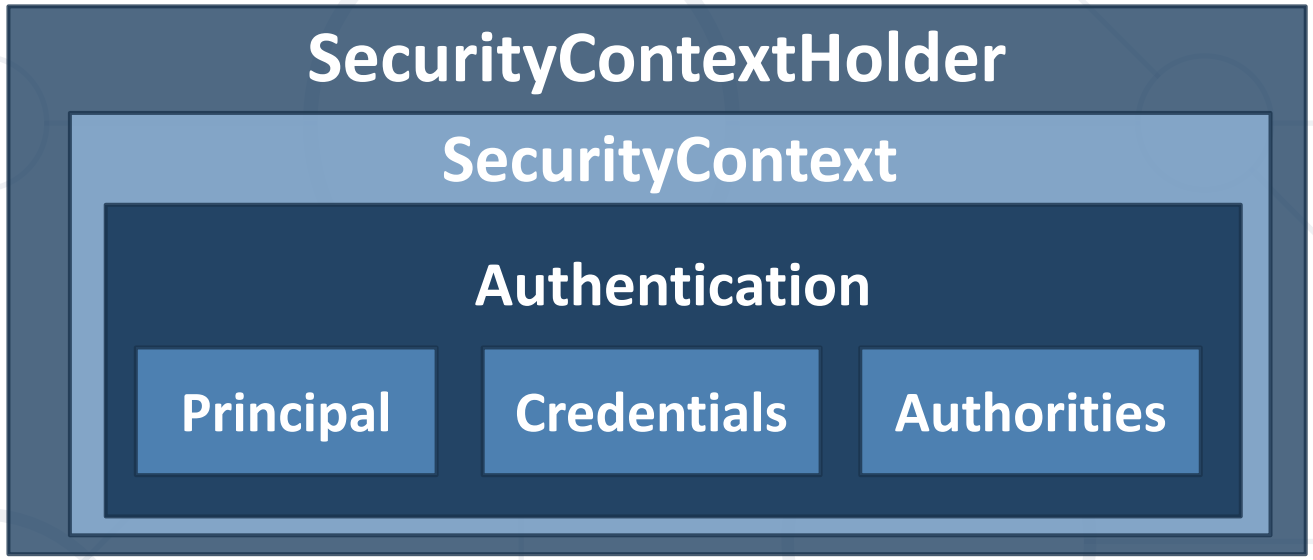
- **Authorization**
  - What you are allowed to do

Software University

Spring Security Filter Chain

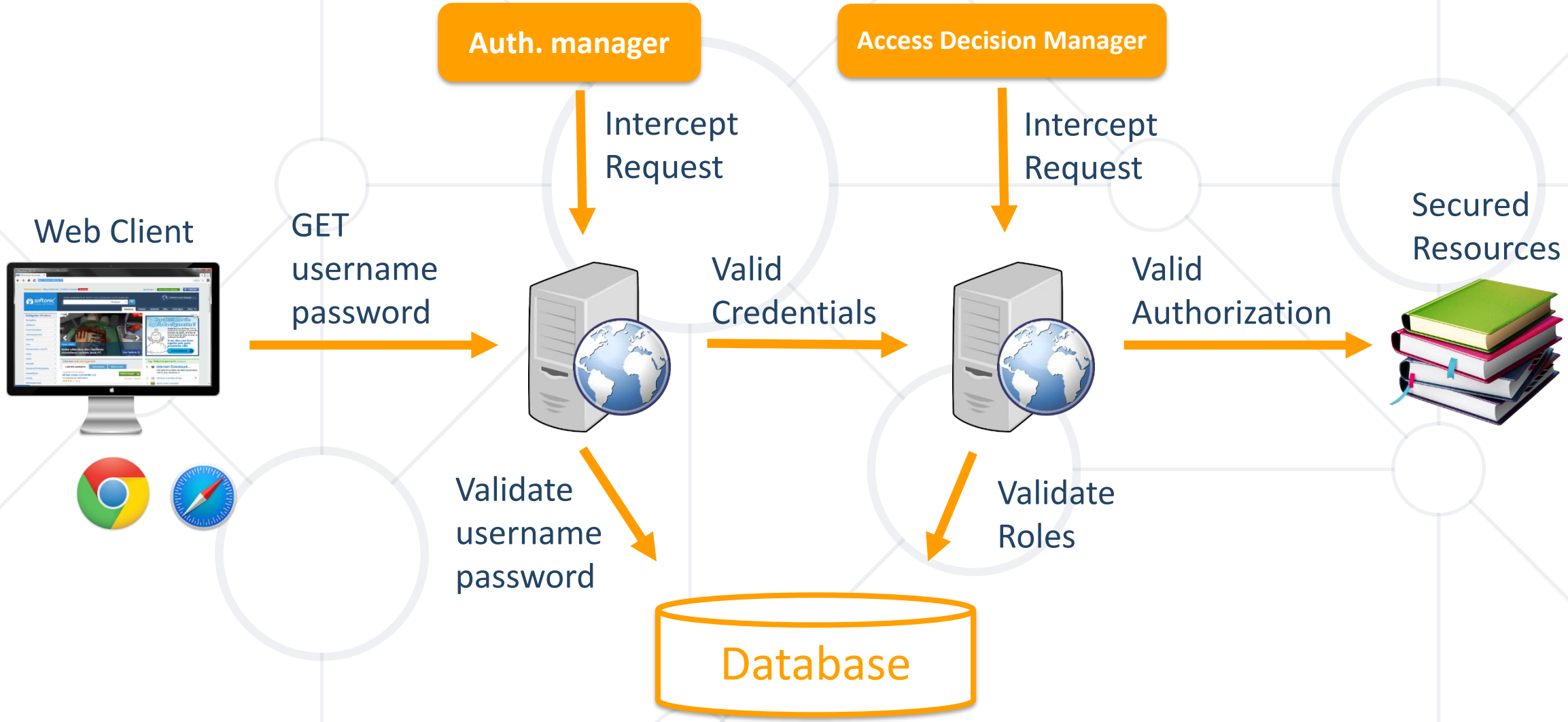


Security Context and Authentication



- At the heart of Spring Security's authentication model is the **SecurityContextHolder**
- It contains the **SecurityContext**

Spring Security Mechanism



Spring Security Maven/Gradle

- Adding **Spring Security**

pom.xml

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

build.gardle

```
dependencies {
      implementation 'org.springframework.boot:spring-boot-starter-
security'
}
```

Spring Security Configuration (1)

- Creating the **SecurityFilterChain** bean.

SecurityConfiguration.java

```
@Configuration
public class SecurityConfiguration {
  @Bean
  public SecurityFilterChain filterChain(HttpSecurity httpSecurity){
     //Configuration goes here
  }
}
```

Spring Security Configuration (2)

- Create the **SecurityFilterChain**

SecurityConfiguration.java

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity httpSecurity){
    http
     .authorizeRequests()
     .antMatchers('/', '/register').permitAll()
     .anyRequest().authenticated();
     …
    return http.build();
}
```

Authorize Requests

Permit Routes

Require Authentication

Build SecurityFilterChain

Registration – User

- We need to implement **UserDetails** interface

UserDetails.java (Spring)

```
public interface UserDetails {
	Collection<? extends GrantedAuthority>
getAuthorities();
	String getPassword();
	String getUsername();
	boolean isAccountNonExpired();
	boolean isAccountNonLocked();
	boolean isCredentialsNonExpired();
	boolean isEnabled();
}
```

```
UserDetails ud =
	User.
	    withUsername(..).
	    password(..).
	    authorities(..).
	    build();
```

Registration – Roles

- Implementing the **GrantedAuthority** interface.

Role.java

```
public class Role implements GrantedAuthority {
    private String authority;
}
```

Role Interface

SimpleGrantedAuthority

- If we want, we can use **SimpleGrantedAuthority** instead of creating Role class
- Is a basic concrete **implementation** of a **GrantedAuthority**
- Stores a **String representation** of an **authority** granted to the Authentication object

UserDetailsService

- Implementing the **UserDetailsService** interface.

UserServiceImpl.java

```
public class UserDetailsServiceImpl implements UserDetailsService {

    public UserDetailsServiceImpl() {
      …
    }

    @Override
    public UserDetails loadUserByUserName(String userName) {
      // get the user and map to UserDetails
    }
}
```

PasswordEncoder and UserDetailsService (1)

- Expose as beans

UserServiceImpl.java

```
@Configuration
public class SecurityConfig {

  @Bean
  public UserDetailsService userDetailsService(UserRepository
                                                 userRepository) {
    return new UserDetailsServiceImpl(userRepository);
  }

  …
}
```

PasswordEncoder and UserDetailsService (2)

- Expose as beans

UserServiceImpl.java

```
@Configuration
public class SecurityConfig {

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new Pbkdf2PasswordEncoder();
  }

  …
}
```

Login Mechanism

Web Client

GET localhost:8080

Session Cookie

GET localhost:8080

Session Cookie

Create Session

Validate Session

Login – Configuration

SecurityConfiguration.java

```
.and()
  .formLogin().loginPage('/login').permitAll()
  .usernameParameter('username')
  .passwordParameter('password')
```

login.html

```
<input type='text' name='username'/>
<input type='text' name='password'/>
```

Login – UserService

UserServiceImpl.java

User Service Interface

```
@Service
public class UserServiceImpl implements UserDetailsService {

    // Some userServiceImpl logic

    @Override
    public UserDetails loadUserByUsername(String username)
                                throws UsernameNotFoundException {
      //...
    }
}
```

Login – Controller

LoginController.java

```
@Controller
public class LoginController {
    @GetMapping('/login')
    public String getLoginPage(@RequestParam(required = false) String
error, Model model) {
        if(error != null){
            model.addAttribute('error', 'Error');
        }

        return 'login';
    }
}
```

Error Handling

Logout

Software University

SecurityConfiguration.java

```
.and()
.logout().logoutSuccessUrl('/login?logout').permitAll()
```

Logout. No Controller is required

Remember Me

SecurityConfiguration.java

```
.and()
      .rememberMe()
      .rememberMeParameter('remember')
      .key('remember Me Encryption Key')
      .rememberMeCookieName('rememberMeCookieName')
      .tokenValiditySeconds(10000)
```

login.html

```
<input name='remember' type='checkbox' />
```

Principal

- This is the **currently logged user**

UserController.java

```
@GetMapping('/user')
public String getUser(Principal principal){
    System.out.println(principal.getName());
    return 'user';
}
```

Print Logged-In username

Pre / Post Authorize

- Grant Access to specific methods

SecurityConfiguration.java

```
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfiguration extends
WebSecurityConfigurerAdapter {
}
```

Enables PreAuthorize

UserService.java

```
public interface UserService extends UserDetailsService {
    @PreAuthorize('hasRole('ADMIN')')
    void delete();
}
```

Requires Admin Role to execute

No Access Handling

SecurityConfiguration.java

```
.and()
.exceptionHandling().accessDeniedPage('/unauthorized')
```

AccessController.java

```
@GetMapping('/unauthorized')
public String unauthorized(){
    return 'unauthorized';
}
```

Cross-Site Request Forgery

Spring CSFR Protection

```
.csrf()
    .csrfTokenRepository(csrfTokenRepository())

private CsrfTokenRepository csrfTokenRepository() {
  HttpSessionCsrfTokenRepository repository = new
HttpSessionCsrfTokenRepository();
  repository.setSessionAttributeName("_csrf");
  return repository;
}
```

form.html

```
<input type='hidden' th:name='${_csrf.parameterName}'
th:value='${_csrf.token}' />
```

Thymeleaf Security

Thymeleaf Security

- Functionality to Thymeleaf

pom.xml

```
<dependency>
      <groupId>org.thymeleaf.extras</groupId>
      <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>
```

Principal

Software University

index.xml

```
<!DOCTYPE html>
<html lang='en'
      xmlns:th='http://www.thymeleaf.org'
      xmlns:sec='http://www.thymeleaf.org/extras/spring-security'>
<body>
<div sec:authentication='name'>
    The value of the 'name' property of the authentication object
should appear here.
</div>
</body>
</html>
```

Show the username

Roles

Software University

index.xml

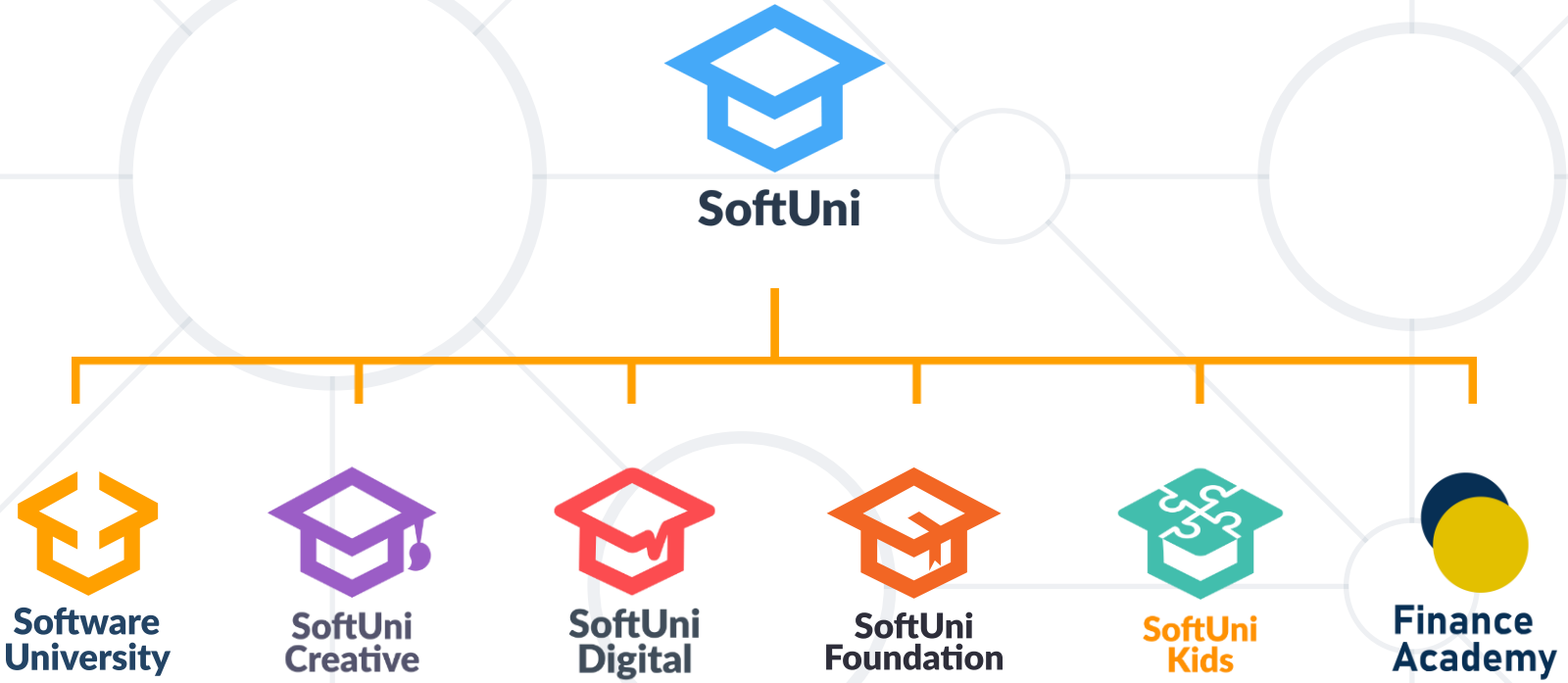
```
<!DOCTYPE html>
<html lang='en'
      xmlns:th='http://www.thymeleaf.org'
      xmlns:sec='http://www.thymeleaf.org/extras/spring-security'>
<body>
<div sec:authorize='hasRole('ADMIN')'>
    This content is only shown to administrators.
</div>
</body>
</html>
```

Show if you are admin

Summary

- What is the difference between **Filters** and **Interceptors**
- What is **Spring Security** and how to implement it
- How to use **Thymeleaf Security**

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg