Spring Security And Monitoring

Cyril Grossenbacher
Khaufra Maggini

ti&m

Security

ti&m

Security

- Intro
- Authentication Types
- Cors

Security

- **Intro**

- Authentication Types

- Cors

Hypertext Transfer Protocol – History

Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP

ti&m

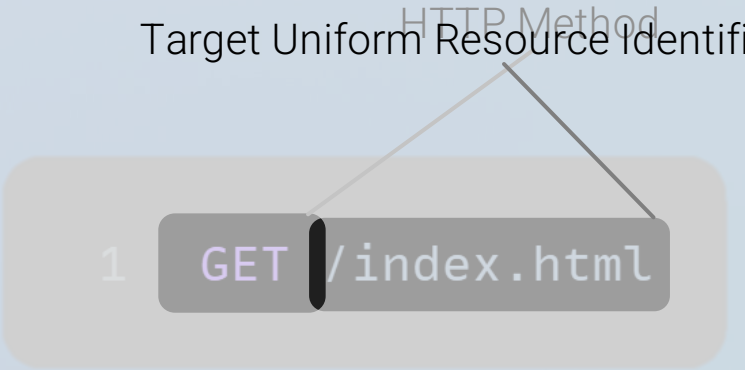
HTTP/0.9 - Overview

- Only possible file type was HTML
- Error Handling over HTML pages
- No Header
- No Status Codes

HTTP/0.9 - HTTP Methods

Method	Description
GET	Retrieve information from server

HTTP/0.9 - Retrieve HTML



ti&m

HTTP/0.9 - Retrieve HTML

```
1   GET /index.html
```

```
1   <html>
2     <h1>
3       Hello World!
4     </h1>
5   </html>
```

HTTP/1.0 - Overview

- Start of versioned protocol
- Headers
- Status code
- Error code

HTTP/1.0 - Retrieve HTML

HTTP Version used to make request

Identification of application requesting resource

```
1  GET /index.html HTTP/1.0
2  User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)
```

HTTP/1.0 - Retrieve HTML

```
1  GET /index.html HTTP/1.0
2  User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)
```

```
1  200 OK
2  Date: Tue, 12 Mai 2023 08:12:31 GMT
3  Server: CERN/3.0 libwww/2.17
4  Content-Type: text/html
5  <html>
6      <h1>
7          Hello World!
8      </h1>
9      <img src="/world.png">
10 </html>
```

When and from whom the Request was sent

The format of the response body

HTTP/1.0 - Retrieve HTML

```
GET /index.html HTTP/1.0
User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)
```

```
200 OK
Date: Tue, 12 Mai 2023 08:12:31 GMT
Server: CERN/3.0 libwww/2.17
Content-Type: text/html
<html>
    <h1>
        Hello World!
    </h1>
    <img src="/world.png">
</html>
```

HTTP/1.0 - Retrieve Image

```
1  GET /world.png HTTP/1.0
2  User-Agent: NCSA_Mosaic/2.0 (Windows 3.1)
```

```
1  200 OK
2  Date: Tue, 12 Mai 2023 08:12:31 GMT
3  Server: CERN/3.0 libwww/2.17
4  Content-Type: image/png
5  
6  AANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAA
7  BGdBTUEAALGPC...
```


ti&m

HTTP/1.0 - Added HTTP Methods

Method	Description
HEAD	Similar to GET but without body information
POST	Request server to accept resources in body

Source: https://datatracker.ietf.org/doc/html/rfc1945#section-8

HTTP/1.0 - HTTP Status codes

Code	Signification	Code	
1xx	Reserved value range for informational response	300	Multiple Choices - Default for any 3xx response
200	OK	301	Moved permanently
201	Created	302	Moved temporarily
202	Accepted – Uncommitted reception	304	Not Modified
204	No Content		

Source: https://datatracker.ietf.org/doc/html/rfc1945#section-8

HTTP/1.0 - HTTP Status codes

Code	Signification	Code	Signification
400	Bad Request	500	Internal Server Error
401	Unauthorized	501	Not Implemented
403	Forbidden	502	Bad Gateway
404	Not found	503	Service unavailable

Source: https://datatracker.ietf.org/doc/html/rfc1945#section-8

HTTP/1.1 - Overview

- Added more methods
- Pipelining for parallel request handling
- Chunking for splitting big payloads
- Content negotiations for simpler connections

HTTP/1.1 - Retrieve HTML

```
GET /index HTTP/1.1
Host: myDomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: text/html,
        application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

Address from where request comes

HTTP/1.1 - Retrieve HTML

```
GET /index HTTP/1.1
Host: mydomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: text/html,
        application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 12 Mai 2023 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 9 Mai 2023 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding

<html>
    ...
```

HTTP/1.1 - Retrieve HTML

```
GET /index HTTP/1.1
Host: mydomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: text/html,
        application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

Should network connection be persisted

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 12 Mai 2023 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 9 Mai 2023 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding

<html>
    ...
```

HTTP/1.1 - Retrieve HTML

```
GET /index HTTP/1.1
Host: mydomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: text/html,
        application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

Result of content negotiation

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 12 Mai 2023 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 9 Mai 2023 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding

<html>
    ...
```

HTTP/1.1 - Retrieve HTML

```
GET /index HTTP/1.1
Host: mydomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: text/html,
        application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 12 Mai 2023 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 9 Mai 2023 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding

<html>
    ...
```

HTTP/1.1 - Retrieve Image

```
GET /static/img/background.png HTTP/1.1
Host: mydomain.org
User-Agent: Mozilla/5.0
            (Macintosh; Intel Mac OS X 10.9; rv:50.0)
            Gecko/20100101 Firefox/50.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://google.com
```

```
200 OK
Age: 9578461
Cache-Control: public, max-age=315360000
Connection: keep-alive
Content-Length: 3077
Content-Type: image/png
Date: Fri, 12 Mai 2023 13:34:46 GMT
Last-Modified: Wed, 10 Mai 2023 18:27:50 GMT
Server: Apache


AAAGQAAABkCAYAAABw4pVUAAAABGdBTUEAALGPC/xhBQ
AANDtJREFUeAHsmQVU...
```


ti&m

HTTP/1.1 – Added HTTP Methods

Method	Description
PUT	Request for update of an already existing server resource
DELETE	Request for deletion of a server resource
TRACE*	Request for loop back of request message

Source: https://datatracker.ietf.org/doc/html/rfc2068#section-9
https://my.f5.com/manage/s/article/K85840901

HTTP/1.1 - Added HTTP Status codes

Code	Signification	Code	
100	Continue	303	See other
101	Switching Protocols	305	Use Proxy
203	Non Authoritative Information		
205	Reset Content		
206	Partial content		

Source: https://datatracker.ietf.org/doc/html/rfc2068#section-10.1

HTTP/1.1 - HTTP Status codes

Code	Signification	Code	Signification
402	Payment required	410	Gone
405	Method not allowed	411	Length required
406	Not Acceptable	412	Precondition failed
407	Proxy authentication required	413	Request entity too large
408	Request timeout	414	Request URI too long
409	Conflict	415	Unsupported media type

Source: https://datatracker.ietf.org/doc/html/rfc2068#section-10.1

HTTP/1.1 - HTTP Status codes

Code	Signification	Code	Signification
504	Gateway timeout	505	HTTP Version not supported

Source: https://datatracker.ietf.org/doc/html/rfc2068#section-10.1

HTTP – Working Group

IETF HTTP Working Group (httpwg.org)

Source: https://datatracker.ietf.org/doc/html/rfc2068#section-10.1

Security

- Intro
- **Authentication Types**
- Cors

Authentication vs. Authorization

Authentication (usually first)	Authorization (usually second)
Is Client who they say they are?	What can Client access? And what not?
Does Client have required credentials?	What level of access does Client have? (CRUD)
	What policies and rules do apply to Client?

Source: https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization

Authentication Factors

Knowledge *something you **know***	Possession *something you **have***	Inherence *something you **are***	Location ***somewhere** you are*	Behavior *something you **do***
Password	Token	Biometric	Network	Point Grid
Secret	Certificate		Geolocation	
Pin				

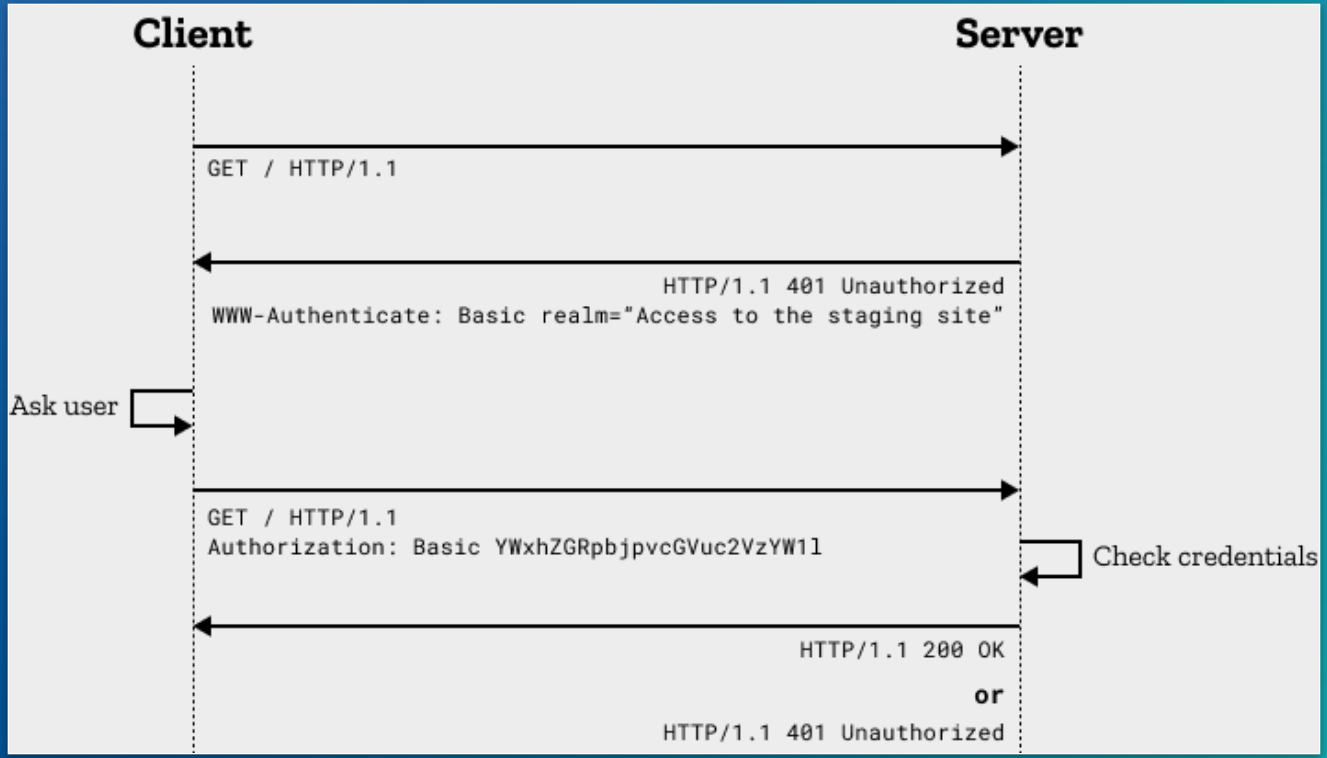
Source: https://www.sumologic.com/glossary/authentication-factor/#:~:text=The%20five%20main%20authentication%20factor,location%20factors%2C%20and%20behavior%20factors.

Multifactor Authentication

ti&m

HTTP Basic Auth

- Minimal security measure
- Base64 encoding not encryption
- Authentication Secret will be sent in HTTP Header
- For UTF-8 support Server has to enable it over HTTP Header



Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

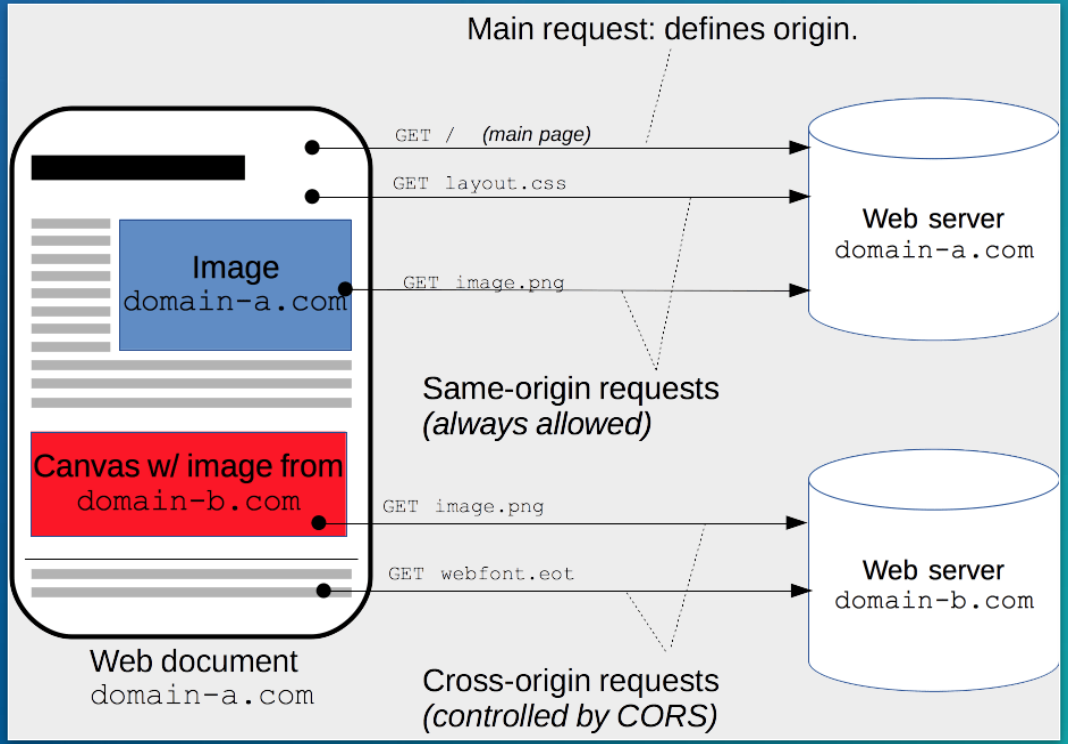
ti&m

Security

- Intro
- Authentication Types
- **Cors**

Cross-Origin Resource Sharing CORS

- Getting Resources from different origins



Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

ti&m

15 min break & Setup the laptops

Repository

ti8m-academy/spring-security-and-monitoring

https://gitlab.ti8m.ch/ti8m-academy/spring-security-and-monitoring

Rest call -> postman

Ide (for run configurations) Intellij Idea

Security – Basic

ti&m

Security – Basic

- Intro
- Configuration
- Default User
- In Memory User
- Password Encoding
- Authorization
- Database Integration
- Annotations

Security – Basic

- **Intro**
- Configuration
- Default User
- In Memory User
- Password Encoding
- Authorization
- Database Integration
- Annotations

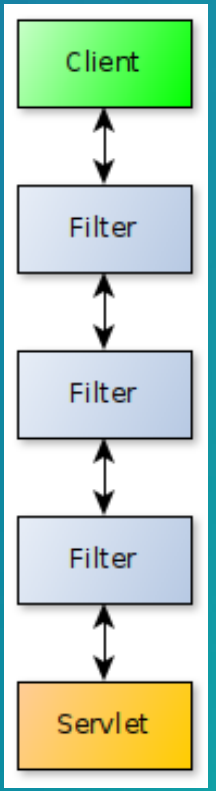
Spring Filter Chain

Spring Filter Chain

- A request from the client to the servlet that menages it pass through multiple filters

Spring Filter Chain

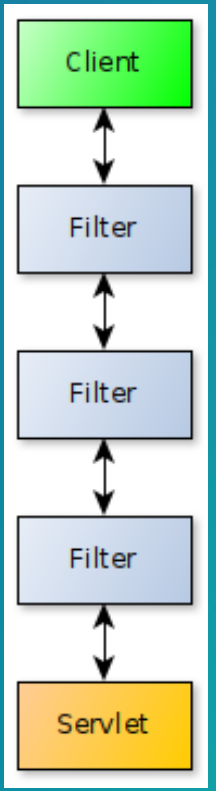
- A request from the client to the servlet that menages it pass through multiple filters



Spring Filter Chain

- A request from the client to the servlet that menages it pass through multiple filters

- Each filter is a bean and can:
  - Handle the request itself, interrupting the chain
  - Modify the request
  - Modify the response

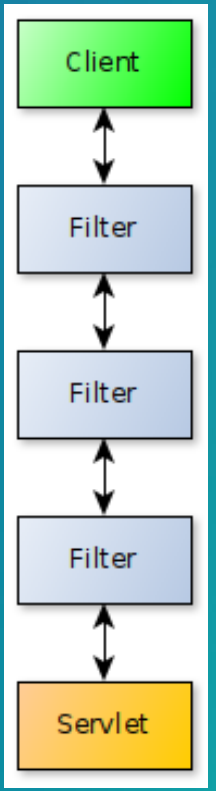


Spring Filter Chain

- A request from the client to the servlet that menages it pass through multiple filters

- Each filter is a bean and can:
  - Handle the request itself, interrupting the chain
  - Modify the request
  - Modify the response

- The order of the filters is very important, it can be managed by
  - ***@Ordered*** annotation
  - Implement ***Ordered***

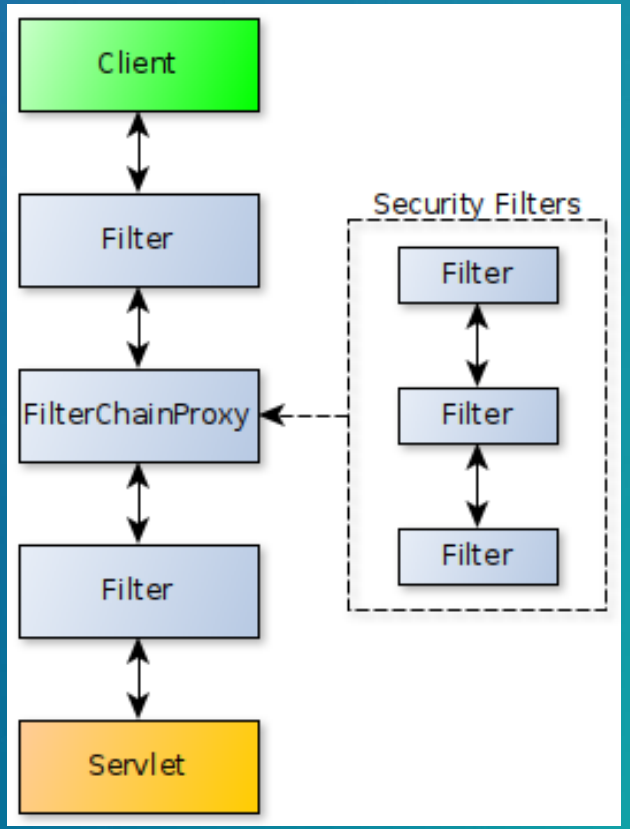


Spring Security Filter

- Spring Security is installed as a filter of this chain

Spring Security Filter

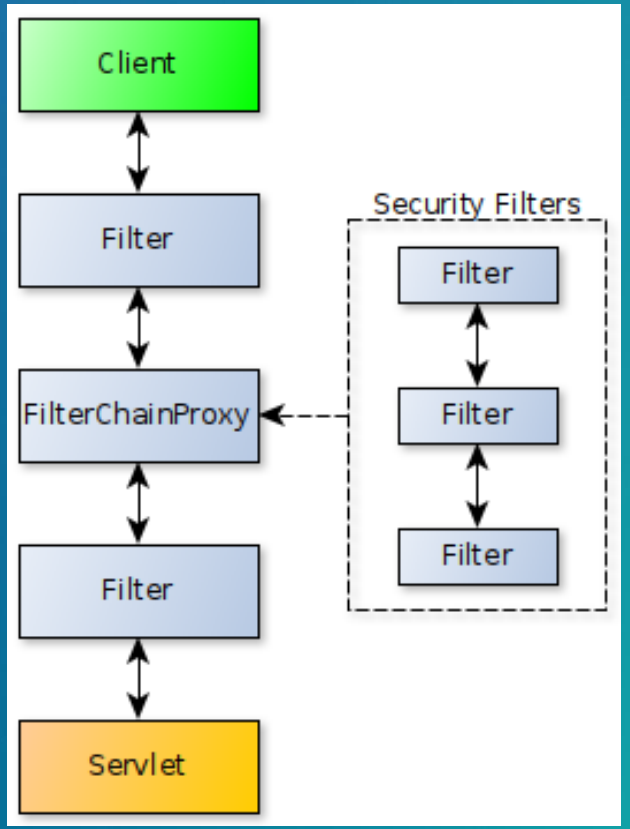
- Spring Security is installed as a filter of this chain



Spring Security Filter

- Spring Security is installed as a filter of this chain

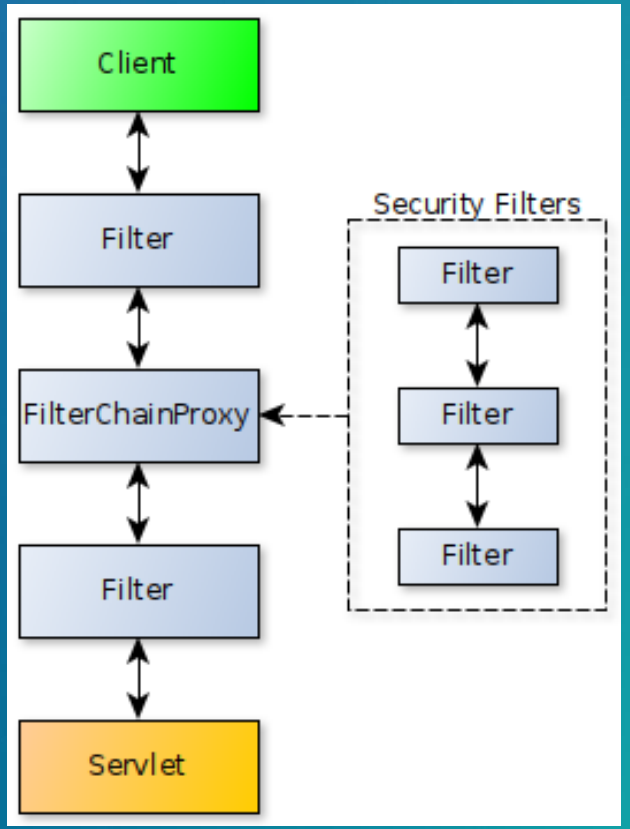
- ***FilterChainProxy*** is the concrete implementation



ti&m

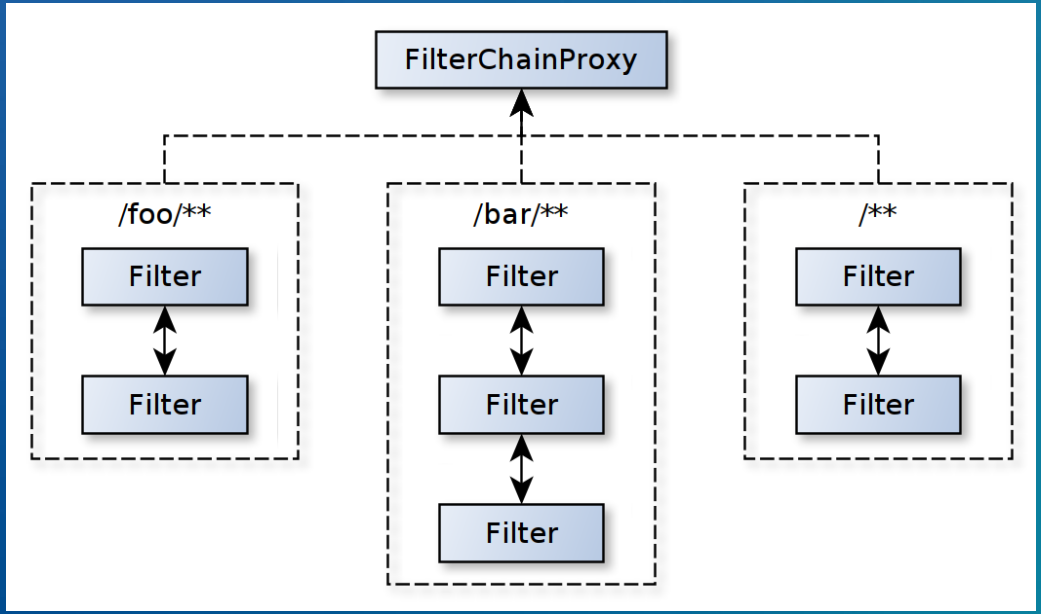
Spring Security Filter

- Spring Security is installed as a filter of this chain
- ***FilterChainProxy*** is the concrete implementation
- Even if from the big picture it is only one filter, in fact it delegates processing to internal filters



Spring Security Filter

- The filter chain proxy dispatches the request to the first matching filter chain



Security – Basic

- Intro
- **Configuration**
- Default User
- In Memory User
- Password Encoding
- Authorization
- Database Integration
- Annotations

Spring Security – Configuration Classes

- **SecurityFilterChain**
  - It defines a single security filter chain used in the ***FilterChainProxy***

Spring Security – Configuration Classes

- **SecurityFilterChain**
  - It defines a single security filter chain used in the ***FilterChainProxy***

- **HttpSecurity** (builder)
  - It allows configuring web security for http requests
  - It defines the rules that applies to a security filter chain

Spring Security – Configuration Bean

```java
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        // define custom sercurity configuration here
        return http.build();
    }
}
```

Spring Security – Configuration Bean – Step 1

- Initialize the http requests authorization configuration

- *HttpSecurity.**authorizeRequests()***

Spring Security – Configuration Bean – Step 2

- Target the http requests types

- Targes
  - .anyRequest()
  - .antMatchers(HttpMethod)
  - .antMatchers(HttpMethod, String...)
  - .antMatchers(String...)

Spring Security – Configuration Bean – Step 3

- Manage the access

- Authorization
  - permitAll()
  - denyAll()
  - authenticated()
  - hasRole(String)
  - hasAnyRole(String...)
  - hasAuthority(String)
  - hasAnyAuthority(String...)

Spring Security – Exercise Basic #1

- Open /default/open to anyone

- Branch: security-basic/exercise_1

Security – Basic

- Intro
- Configuration
- **Default User**
- In Memory User
- Password Encoding
- Authorization
- Database Integration
- Annotations

Spring Security – Basic Auth Default User

- Users are managed by *UserDetails* configuration bean

Spring Security – Basic Auth Default User

- Users are managed by *UserDetails* configuration bean

- When this bean is not provided, a default user is generated
  - Username: user
  - Password: logged in the console (uuid)

Spring Security – Basic Auth Default User

- Default user name and password are configurable on yaml
  - spring.security.user.name
  - spring.security.user.password

Spring Security – Basic Auth Default User

- Default user name and password are configurable on yaml
  - spring.security.user.name
  - spring.security.user.password

- Or can be disable by excluding
  - *UserDetailsServiceAutoConfiguration*

Spring Security – Basic Auth Default User

- In order to correctly authenticate the default basic auth user in the service, it is necessary to activate the basic auth mechanism

- *HttpSecurity.**httpBasic()***

Spring Security – Exercise Basic #2

- Customize the default user
  - Username: default
  - Password: password
- User basic authentication
- Require authenticated requests (except open endpoint)
- Branch: security-basic/exercize_2

Security – Basic

- Intro
- Configuration
- Default User
- **In Memory User**
- Password Encoding
- Authorization
- Database Integration
- Annotations

Spring Security – In Memory User

- It is possible to avoid to use the default user and add multiple users to the service by providing an **InMemoryUserDetailsManager** bean

Spring Security – In Memory User

- It is possible to avoid to use the default user and add multiple users to the service by providing an **InMemoryUserDetailsManager** bean

- The manager takes an array of **UserDetails**

Spring Security – In Memory User

- It is possible to avoid to use the default user and add multiple users to the service by providing an **InMemoryUserDetailsManager** bean

- The manager takes an array of **UserDetails**

- An UserDetails can be generate using the **User** builder

Spring Security – In Memory User - UserDetails

- The **User** builder starts with the user name definition
  - *withUsername(String)*

Spring Security – In Memory User - UserDetails

- The **User** builder starts with the user name definition
  - *withUsername(String)*

- *And can be customized by*
  - *password(String)*
  - *roles(String...)*
  - *authorities(String...)*
  - *disabled(boolean)*

Spring Security – In Memory User - UserDetails

- The **User** builder starts with the user name definition
  - *withUsername(String)*

- *And can be customized by*
  - *password(String)*
  - *roles(String...)*
  - *authorities(String...)*
  - *disabled(boolean)*

- The UserDetails are created by the *build()* method

Security – Basic

- Intro
- Configuration
- Default User
- In Memory User
- **Password Encoding**
- Authorization
- Database Integration
- Annotations

Spring Security – Password Encoding

- The **PasswordEncoder** bean is responsible for encoding and validate passwords

Spring Security – Password Encoding

- The **PasswordEncoder** bean is responsible for encoding and validate passwords

- *It can be generated using the **PasswordEncoderFactories***

```
@Bean
public PasswordEncoder passwordEncoder() {
    return PasswordEncoderFactories.createDelegatingPasswordEncoder();
}
```

Spring Security – Password Encoding

- *When using password in an application it is a best practice to do not have is as raw text but encoded*

Spring Security – Password Encoding

- *When using password in an application it is a best practice to do not have is as raw text but encoded*

- An encoded password looks like
  - *{<encoderType>}<encodedPassword>*

Spring Security – Password Encoding

- *When using password in an application it is a best practice to do not have is as raw text but encoded*

- An encoded password looks like
  - *{<encoderType>}<encodedPassword>*

- Where some encoder types are:
  - noop (raw)
  - bcrypt
  - MD5
  - SHA-256

Spring Security – Exercise Basic #3

- Create the in-memory users
  - Admin
    - password: admin-password
  - User
    - password: user-password

- Verify users can access /default/authenticated endpiont

- Branch: security-basic/exercize_3

Security – Basic

- Intro
- Configuration
- Default User
- In Memory User
- Password Encoding
- **Authorization**
- Database Integration
- Annotations

Spring Security – Authorities

- Represents an individual privilege

Spring Security – Authorities

- Represents an individual privilege
- It has any naming convention (just CONSTANT_CASE)

Spring Security – Authorities

- Represents an individual privilege
- It has any naming convention (just CONSTANT_CASE)
- Example
  - READ_AUTHORITY
  - WRITE_PRIVILEGE
  - CAN_DELETE

Spring Security - Roles

- It is an authorities wrapper

Spring Security - Roles

- It is an authorities wrapper
- It grants automatically the authority *ROLE_<role_name>*

Spring Security - Roles

- It is an authorities wrapper
- It grants automatically the authority *ROLE_<role_name>*
- Example
  - ADMIN (ROLE_ADMIN)
  - STAFF (ROLE_STAFF)
  - USER (ROLE_USER)

Spring Security – Exercise Basic #4

- Add roles to in memory users
  - admin: ADMIN, STAFF, USER
  - staff: STAFF, USER
  - user: USER
- Verity the configuration on /roles
- Branch: security-basic/exercise_4

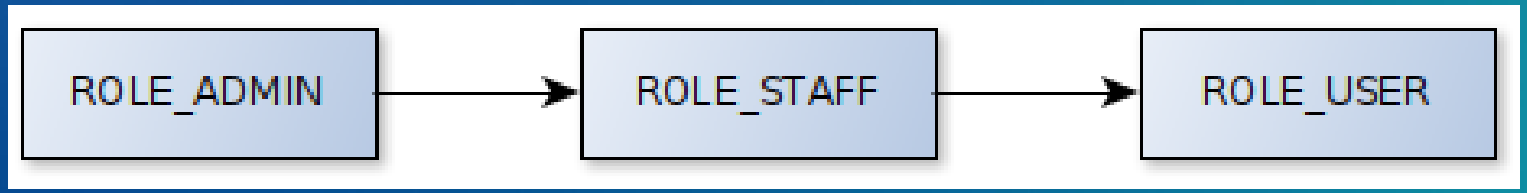
Spring Security – Exercise Basic #4 part 2

- Use the defined roles to secure the endpoints
  - DELETE /message: ADMIN
  - POST /message: STAFF
  - PUT /message: STAFF
  - GET /message: USER

- Branch: security-basic/exercise_4_part_2

Spring Security – Authorities' Hierarchy

- In an hierarchy the upper level authorities include the lower ones
  - ROLE_ADMIN (grants ROLE_STAFF and ROLE_USER)
  - ROLE_STAFF (grants ROLE_USER)
  - ROLE_USER



Spring Security – Hierarchy Definition

- The hierarchy can be defined using a bean

```
@Bean
public RoleHierarchy roleHierarchy() {
    var hierarchy = new RoleHierarchyImpl();
    hierarchy.setHierarchy("..."); // hierarchy definition
    return hierarchy;
}
```

Spring Security – Hierarchy Definition

- The hierarchy can be defined using a bean

```
@Bean
public RoleHierarchy roleHierarchy() {
    var hierarchy = new RoleHierarchyImpl();
    hierarchy.setHierarchy("..."); // hierarchy definition
    return hierarchy;
}
```

- Where the definition string is
  - *<upper> > <lower> [> <lower>]*

Spring Security – Hierarchy Usage

- Define an expression handler that uses the defined hierarchy

```
@Bean
public DefaultWebSecurityExpressionHandler webSecurityExpressionHandler() {
    var expressionHandler = new DefaultWebSecurityExpressionHandler();
    expressionHandler.setRoleHierarchy(roleHierarchy());
    return expressionHandler;
}
```

Spring Security – Hierarchy Usage

- Define an expression handler that uses the defined hierarchy

```
@Bean
public DefaultWebSecurityExpressionHandler webSecurityExpressionHandler() {
    var expressionHandler = new DefaultWebSecurityExpressionHandler();
    expressionHandler.setRoleHierarchy(roleHierarchy());
    return expressionHandler;
}
```

- Add it to the security filter

```
http.authorizeRequests()
        .expressionHandler(webSecurityExpressionHandler());
```

Spring Security – Exercise Basic #4 part 3

- Define the roles' hierarchy
- Simplify the roles requirements in the users' details definition
- Branch: security-basic/exercise_4_part_3

Security – Basic

- Intro
- Configuration
- Default User
- In Memory User
- Password Encoding
- Authorization
- **Database Integration**
- Annotations

Spring Security – Database Integration

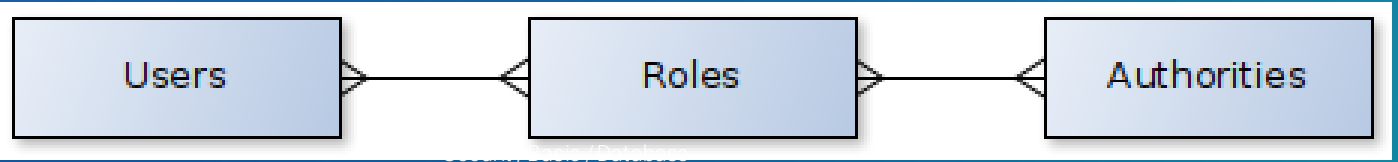
- Needed Information
  - username
  - password
- Optional Information
  - activation status
  - roles / authorities (recommended)

Spring Security – Database Schema

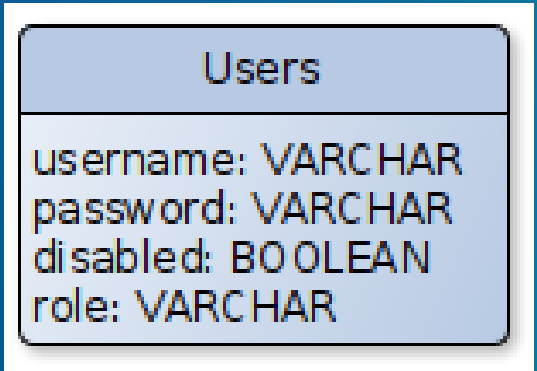
- Full Schema

Spring Security – Database Schema

- Full Schema



- Lazy Schema
  - using hierarchy definition



Spring Security – Database Schema Migration

```sql
CREATE TABLE `users` (
    `username`      VARCHAR(64)     NOT NULL,
    `password`      VARCHAR(128)    NOT NULL,
    `role`          VARCHAR(8)      NOT NULL,
    `disabled`      BOOLEAN         DEFAULT FALSE,
    PRIMARY KEY (`username`)
);

INSERT INTO `users` (`username`, `password`, `role`, `disabled`) VALUES
('admin@example.com','{bcrypt}$2a$12$.dT5J2XuDrAH7Q3Uc.B9Quc855XzELiIyM0QeaRkZxn7G7YBZhyNO','ADMIN', FALSE),
('staff@example.com','{bcrypt}$2a$12$N5e6.2VF101ZKa1KKqFb5.4Eizm8Fv9gSwSFA1UTaMv11LVc.SAzK','STAFF', FALSE),
('user@example.com','{noop}user-password','USER', FALSE),
('disabled@example.com','{noop}disabled-password','USER', TRUE);
```


ti&m

Spring Security – UserDetailsService

- InMemory

```
@Bean
public InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {
    var user = User.withUsername("user").build();
    return new InMemoryUserDetailsManager(user);
}
```

Spring Security – UserDetailsService

- InMemory

```
@Bean
public InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {
    var user = User.withUsername("user").build();
    return new InMemoryUserDetailsManager(user);
}
```

- Service

```
@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return User.withUsername(username).build();
    }
}
```

Spring Security – UserDetails

- Builder

```
User.withUsername(username) /* ... */ .build();
```

Spring Security – UserDetails

- Builder

```
User.withUsername(username) /* ... */ .build();
```

- Constructor

```
new User(username,
         password,
         enabled,
         accountNonExpired,
         credentialsNonExpired,
         accountNonLocked,
         grantedAuthorities);
```

Spring Security – Exercise Basic #5

- Provide the dynamic **UserDetails** using the **UserDetailsService**

- Branch: security-basic/exercise_5

Security – Basic

- Intro
- Configuration
- Default User
- In Memory User
- Password Encoding
- Authorization
- Database Integration
- **Annotations**

Spring Security – Annotations

- Annotations' enabling annotation
  - *@EnableGlobalMethodSecurity*

Spring Security – Annotations

- Annotations' enabling annotation
  - *@EnableGlobalMethodSecurity*

- Configuration
  - ***prePostEnabled (true| false)***
  - ***securedEnabled (true| false)***
  - ***jsr250Enabled (true| false)***

Spring Security – Annotations

- prePostEnabled
  - ***@PreAuthorize @PostAuthorize - method invocation control***
  - ***@PreFilter @PostFilter - collections' response filtering***

Spring Security – Annotations

- prePostEnabled
  - ***@PreAuthorize @PostAuthorize - method invocation control***
  - ***@PreFilter @PostFilter - collections' response filtering***

- securedEnabled
  - ***@Secured - method invocation control***

Spring Security – Annotations

- prePostEnabled
  - ***@PreAuthorize @PostAuthorize - method invocation control***
  - ***@PreFilter @PostFilter - collections' response filtering***
- securedEnabled
  - ***@Secured - method invocation control***
- jsr250Enabled
  - ***@RoleAllowed - method invocation control***

Spring Security – Annotations - [Per|Post]Authorize

- Usage
  - ***@[Pre|Post]Authorize("<rule>")***

Spring Security – Annotations - [Per|Post]Authorize

- Usage
  - ***@[Pre|Post]Authorize("<rule>")***

- Rules
  - ***isAnonymous()***
  - ***isAuthenticated()***
  - ***permitAll()***
  - ***hasRole('<role>')***
  - ***hasAuthority('<authority>')***
  - ***...***

Spring Security – Annotations - Secured

- Usage
  - ***@Secured("<role>")***

Spring Security – Annotations - Secured

- Usage
  - ***@Secured("<role>")***

- Example
  - *ROLE_ADMIN*
  - *ROLE_STAFF*
  - *ROLE_USER*

Spring Security – Annotations - RolesAllowed

- Usage
  - ***@RolesAllowed("<role>")***
  - ***@RolesAllowed({"<role>", "<role>"})***

Spring Security – Annotations - RolesAllowed

- Usage
  - ***@RolesAllowed("<role>")***
  - ***@RolesAllowed({"<role>", "<role>"})***

- Example
  - *ROLE_ADMIN*
  - *ROLE_STAFF*
  - *ROLE_USER*

Spring Security – Exercise Basic #6

- Activate desired annotations with *@EnableGlobalMethodSecurity*

- Secure endpoints with annotations:
  - GET /default/open: open
  - GET /default/authenticated: authenticated (already true, see comments)
  - GET /default/roles: authenticated (already true, see comments)
  - DELETE /message: ADMIN
  - POST /message: STAFF
  - PUT /message: STAFF
  - GET /message: USER

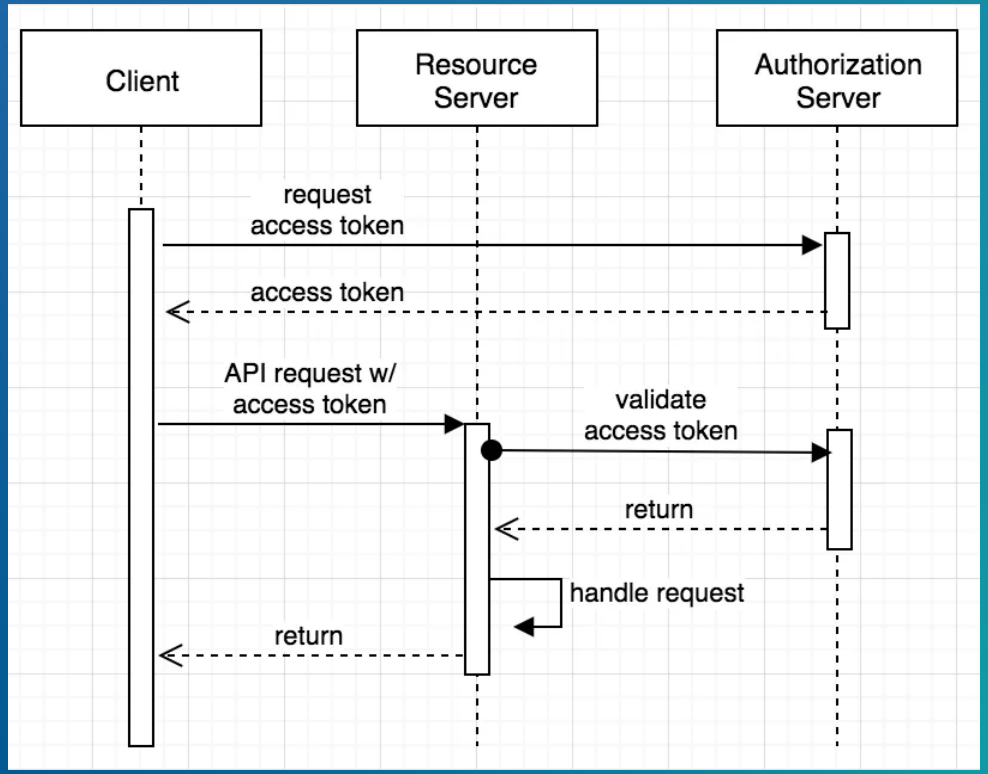
- Branch: security-basic/exercise_6

Lunch Time

Security – OAuth2

OAuth 2.0

- Delegate Authorization of User
- Not usable for authentication



Source: https://developer.okta.com/blog/2018/04/02/client-creds-with-spring-boot

OAuth 2.0 - Authorization Server

https://console.cloud.google.com

- Create Project
- Create Consent Screen
- Create Test user
- Branch: oauth/exercise

15 minutes break

Error Handling

Error Handling

- Default Message Customization
- Exception Management
- Advanced Options

Error Handling

- **Default Message Customization**

- Exception Management

- Advanced Options

Default Not Found Page

- Spring Boot Rest provides an automatic response message if the page searched doesn't exists

```
{
    "timestamp": "2023-04-21T07:11:42.762+00:00",
    "status": 404,
    "error": "Not Found",
    "path": "/error-handling/not-existing"
}
```

Default Error Page

- Given a not handled exception

```
@GetMapping("not-implemented")
public void notImplemented() { throw new CustomNotImplementedException(); }
```

Default Error Page

- Given a not handled exception

```java
@GetMapping("not-implemented")
public void notImplemented() { throw new CustomNotImplementedException(); }
```

- The default return code is 500

```
{
    "timestamp": "2023-05-10T11:48:06.521+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "path": "/error-handling/errors/not-implemented"
}
```

Error Message Customization – Properties

- server.error.**path** (/error)
  - Path of the error controller

Error Message Customization – Properties

- server.error.**path** (/error)
  - Path of the error controller
- server.error.**whitelabel.enabled** (true | false)
  - Provides the default HTML error page
    or relies on the container's default one (for example tomcat)

Error Message Customization – Properties

- server.error.**path** (/error)
  - Path of the error controller
- server.error.**whitelabel.enabled** (true | false)
  - Provides the default HTML error page or relies on the container's default one (for example tomcat)
- server.error.**include-exception** (true | false)
  - Includes the exception type in the default response

Error Message Customization – Properties

- server.error.**path** (/error)
  - Path of the error controller
- server.error.**whitelabel.enabled** (true | false)
  - Provides the default HTML error page or relies on the container's default one (for example tomcat)
- server.error.**include-exception** (true | false)
  - Includes the exception type in the default response
- server.error.**include-stacktrace** (always | never | on-param)
  - Shows the error stacktrace in the default response

Error Message Customization – Properties

- server.error.**path** (/error)
  - Path of the error controller
- server.error.**whitelabel.enabled** (true | false)
  - Provides the default HTML error page or relies on the container's default one (for example tomcat)
- server.error.**include-exception** (true | false)
  - Includes the exception type in the default response
- server.error.**include-stacktrace** (always | never | on-param)
  - Shows the error stacktrace in the default response
- server.error.**include-message** (always | never | on-param)
  - Shows the error message of the exception if present

Error Message Customization – Response Status

- Any custom exception can be annotated with
  - *@ResponseStatus*

Error Message Customization – Response Status

- Any custom exception can be annotated with
  - *@ResponseStatus*

- It Allows to define
  - default HttpStatus to return
  - generic error reason (aka message)

Error Message Customization – DefaultErrorAttributes

Another way to customize the default error message is to provide a Configuration bean that extends *DefaultErrorAttributes*

```
@Configuration
public class CustomErrorAttributes extends DefaultErrorAttributes {
    @Override
    public Map<String, Object> getErrorAttributes(WebRequest webRequest,
                                                  ErrorAttributeOptions options) {
        var errorAttributesMap = super.getErrorAttributes(webRequest, options);
        // do your customizations
        return errorAttributesMap;
    }
}
```

Spring Exception – Exercise #1

- Add exception and message to response with properties configuration
- Customize Status Code and Message with *@ResponseStatus*
- Add locale info using *DefaultErrorAttributes*
- Branch: error-handling/exercise_1

Error Handling

- Default Message Customization
- **Exception Management**
- Advanced Options

Error Message Customization – ResponseStatusException

- Spring 5+ provides a dedicated unchecked exception
  - *ResponseStatusException*

Error Message Customization – ResponseStatusException

- Spring 5+ provides a dedicated unchecked exception
  - *ResponseStatusException*

- It can be used instead of *@ResponseStatus* for externally defined exceptions or just to use inside the control flow

Error Message Customization – ResponseStatusException

- Spring 5+ provides a dedicated unchecked exception
  - *ResponseStatusException*
- It can be used instead of *@ResponseStatus* for externally defined exceptions or just to use inside the control flow
- Constructor's parameters
  - HttpStatus (required)
  - Reason message (optional)
  - Throwable cause (optional)

Error Message Customization – ExceptionHandler

- It is possible to define in each controller a method that takes care of a particular exception using *@ExceptionHandler*

Error Message Customization – ExceptionHandler

- It is possible to define in each controller a method that takes care of a particular exception using *@ExceptionHandler*
- It takes as argument the list of exceptions to handle

Error Message Customization – ExceptionHandler

- It is possible to define in each controller a method that takes care of a particular exception using *@ExceptionHandler*
- It takes as argument the list of exceptions to handle
- The handler method can receive as optional parameters
  - The throwed Exception (base class if multiple)
  - The request HttpServletRequest that generated the error

Error Message Customization – ExceptionHandler

- It is possible to define in each controller a method that takes care of a particular exception using *@ExceptionHandler*
- It takes as argument the list of exceptions to handle
- The handler method can receive as optional parameters
  - The throwed Exception (base class if multiple)
  - The request HttpServletRequest that generated the error
- The handler method can return any type response

Spring Exception– Exercise #2

- Use a try-catch and throw a *ResponseStatusException* with
  - Status code: Locked
  - Reason: The resource cannot be accesses

- Branch: error-handling/exercise_2

Spring Exception– Exercise #2 part 2

- Create a method annotated with *@ExceptionHandler* that manages *CustomLockedException* and returns:
  - Locked status code
  - Response message with
    - Code: RESOURCE_LOCKED
    - Message: The resource cannot be accessed

- Branch error-handling/exercise_2_part_2

Error Message Customization – ControllerAdvice

- A class annotated with *@ControllerAdvice* extends the exception handling to any controller in the application

Error Message Customization – ControllerAdvice

- A class annotated with *@ControllerAdvice* extends the exception handling to any controller in the application

- In the REST context it is possible to use *@RestControllerAdvice*
  - Syntactic sugar for *@ControllerAdvice* + *@ResponseBody*

Error Message Customization – ControllerAdvice

- A class annotated with *@ControllerAdvice* extends the exception handling to any controller in the application
- In the REST context it is possible to use *@RestControllerAdvice*
  - Syntactic sugar for *@ControllerAdvice* + *@ResponseBody*
- This means that a method annotated with *@ExceptionHandler* can be moved in this class and become globally available

Error Handling

- Default Message Customization
- Exception Management
- **Advanced Options**

Advanced Customization – Putting All Together

- What we have
  - A way to manage exception with custom responses (*@ExceptionHandler*)
    - And make it globally (*@RestControllerAdvice*)
  - A way to dynamically return status codes (ResponseStatusException)

Advanced Customization – Putting All Together

- What we have
  - A way to manage exception with custom responses (*@ExceptionHandler*)
    - And make it globally (*@RestControllerAdvice*)
  - A way to dynamically return status codes (ResponseStatusException)

- What we want
  - Provide a consistent error response for our apis
  - Have a dynamic way to define status codes and error messages
  - Have a dynamic way to log errors (!= api error messages)

Advanced Customization – Putting All Together

- Solution
  - Use *@ExceptionHandler* inside *@RestControllerAdvice* to provide custom responses and error logging for standard exceptions

```
@ResponseStatus(HttpStatus.BAD_REQUEST)
@ExceptionHandler(ValidationException.class)
public ErrorMessage handleValidationException(ValidationException ex, WebRequest request) {
    logError(request, ex.getMessage(), LogLevel.ERROR);
    return new ErrorMessage(ErrorCode.MALFORMED_USER_REQUEST, "Not valid request,");
}

@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
@ExceptionHandler(PersistenceException.class)
public ErrorMessage handlePersistenceException(PersistenceException ex, WebRequest request) {
    logError(request, ex.getMessage(), LogLevel.ERROR);
    return new ErrorMessage(ErrorCode.EXECUTION_REQUEST, "Unable to finalyze request.");
}
```

Advanced Customization – Putting All Together

- Solution
  - Use *@ExceptionHandler* inside *@RestControllerAdvice* to provide custom responses and error logging for standard exceptions
  - Create a custom exception that can be thrown from anywhere where it is possible to customize
    - Loggin message
    - Response Message
    - Status Code
    - Etc

```
@Value @With @EqualsAndHashCode(callSuper = true)
public class GenericApiException extends RuntimeException {
    HttpStatus status;
    ErrorCode code;
    String userMessage;
    String internalMessage;
    LogLevel logLevel;

    public static GenericApiException notFound() {
        var defaultMessage = "Nothing found.";
        return new GenericApiException(
                HttpStatus.NOT_FOUND,
                ErrorCode.NOTING_HERE,
                defaultMessage,
                defaultMessage,
                LogLevel.ERROR
        );
    }
}
```

Spring Exception– Exercise #3

- Throw the *GenericApiException* and manage it with *@ExceptionHandler*
  - Status code: NOT_FOUND
  - Logged message: Droids are safe
  - Logging Level: WARNING
  - Response Message: "These are not the droids you are looking for"
  - Response Code: NOTING_HERE

- Branch: error-handling/exercise_2

Advanced

- ResponseEntityExceptionHandler
  - It is an useful base class for any *@ControllerAdvice* *(reccomended)*
  - It already handles for a lot of generic exceptions
    - HttpRequestMethodNotSupportedException
    - HttpMediaTypeNotSupportedException
    - HttpMessageNotReadableException
    - HttpMessageNotWritableException
    - NoHandlerFoundException
    - AsyncRequestTimeoutException
    - ...

Advanced

- ResponseEntityExceptionHandler
  - It is an useful base class for any *@ControllerAdvice* *(reccomended)*
  - It already handles for a lot of generic exceptions
    - HttpRequestMethodNotSupportedException
    - HttpMediaTypeNotSupportedException
    - HttpMessageNotReadableException
    - HttpMessageNotWritableException
    - NoHandlerFoundException
    - AsyncRequestTimeoutException
    - ...
- BasicErrorController
  - Is the controller responsible to manage the */error* page
  - It can be extended to add more customization for errors

Advanced – Use Case

- All endpoints use json but I want to return consistent errors for xml

Advanced – Use Case

- All endpoints use json but I want to return consistent errors for xml
  - On *@ControllerAdvice* (extends *ResonseEntityExceptionHandler*)

```
@Override
protected ResponseEntity<Object> handleHttpMediaTypeNotAcceptable(HttpMediaTypeNotAcceptableException ex,
                                                                  HttpHeaders headers,
                                                                  HttpStatus status,
                                                                  WebRequest request) {
    var errorMessage = new ErrorMessage(ErrorCode.MALFORMED_USER_REQUEST, "Not Supported");
    return new ResponseEntity<>(errorMessage, status);
}
```

Advanced – Use Case

- All endpoints use json but I want to return consistent errors for xml
  - On *@ControllerAdvice* (extends *ResonseEntityExceptionHandler*)

```
@Override
protected ResponseEntity<Object> handleHttpMediaTypeNotAcceptable(HttpMediaTypeNotAcceptableException ex,
                                                                  HttpHeaders headers,
                                                                  HttpStatus status,
                                                                  WebRequest request) {
    var errorMessage = new ErrorMessage(ErrorCode.MALFORMED_USER_REQUEST, "Not Supported");
    return new ResponseEntity<>(errorMessage, status);
}
```

  - On BasicErrorController

```
@Component
public class CustomErrorController extends BasicErrorController {
    public CustomErrorController(ErrorAttributes errorAttributes, ServerProperties serverProperties) {
        super(errorAttributes, serverProperties.getError());
    }

    @RequestMapping(produces = MediaType.APPLICATION_XML_VALUE)
    public ResponseEntity<Map<String, Object>> xmlError(HttpServletRequest request) {
        var options = super.getErrorAttributeOptions(request, MediaType.APPLICATION_XML);
        var body = super.getErrorAttributes(request, options);
        return ResponseEntity.status(super.getStatus(request)).body(body);
    }
}
```

Advanced – Hands On

- Just try the rest call

- Relevant Classes
  - ApplicationExceptionHandler#handleHttpMediaTypeNotAcceptable
  - CustomErrorControlles

- Needed extra configuration
  - error-handling pom -> xml dependency
  - WebConfig -> message converters and default content type

- Branch: error-handling/advanced

15 minutes break

Monitoring

ti&m

Monitoring

- Intro
- Actuator
- Metrics
- Admin

Monitoring

- **Intro**
- Actuator
- Metrics
- Admin

Monitoring

- Why Are we interested in monitoring?
  - Ensure availability
  - Generate alerts
  - Performance analysis

Monitoring

- Why Are we interested in monitoring?
  - Ensure availability
  - Generate alerts
  - Performance analysis

- The monitoring operations can be heavy
  - The service usually provides only the data
  - There is another system that collects, analyses and shows this data to an user

Monitoring

- Intro
- **Actuator**
- Metrics
- Admin

Spring Actuator

- Defines a lot of production-ready features, not only monitoring

Spring Actuator

- Defines a lot of production-ready features, not only monitoring

- Accessible by
  - Http
  - Jmx

Spring Actuator

- Defines a lot of production-ready features, not only monitoring
- Accessible by
  - Http
  - Jmx
- Base url: */actuator*

Spring Actuator – Endpoints

- /health
  - Health information

Spring Actuator – Endpoints

- /health
  - Health information
- /info
  - Application information

Spring Actuator – Endpoints

- /health
  - Health information
- /info
  - Application information
- /flyway, /liquibase
  - Migration information

Spring Actuator – Endpoints

- /health
  - Health information
- /info
  - Application information
- /flyway, /liquibase
  - Migration information
- /metrics
  - Metrics information (just names, data depends on monitoring system)

Spring Actuator – Endpoints

- /health
  - Health information
- /info
  - Application information
- /flyway, /liquibase
  - Migration information
- /metrics
  - Metrics information (just names, data depends on monitoring system)
- /shutdown
  - Can be used to shut down the application

Spring Actuator – Endpoints

- /health
  - Health information
- /info
  - Application information
- /flyway, /liquibase
  - Migration information
- /metrics
  - Metrics information (just names, data depends on monitoring system)
- /shutdown
  - Can be used to shut down the application
- More endpoints are available

Spring Actuator – Configuration

- Generic configuration
  - *management.endopints.enabled-by-default* (*true|false*)

Spring Actuator – Configuration

- Generic configuration
  - *management.endopints.enabled-by-default* *(true|false)*
- Configure single endpoint
  - *management.endopint.<name>.enabled* *(true|false)*

Spring Actuator – Configuration

- Generic configuration
  - *management.endopints.enabled-by-default* *(true|false)*
- Configure single endpoint
  - *management.endopint.<name>.enabled* *(true|false)*

- Expose endpoints
  - *management.endopints.<technology>.exposure.include* *(<name> [,<name>|*])*
  - *management.endopints.<technology>.exposure.exclude* *(<name> [,<name>|*])*
  - *With technology*
    - *web*
    - *jmx*

Spring Actuator – Security

- If Spring Security is in the class-path and no custom security is defined
  - all the endpoints but */health* are automatically secured and CSRF is enabled

Spring Actuator – Security

- If Spring Security is in the class-path and no custom security is defined
  - all the endpoints but */health* are automatically secured and CSRF is enabled

- If a custom *SecurityFilterChain* is defined the auto-configuration is not loaded and the endpoints must be manually secured

Spring Actuator – Security

- If Spring Security is in the class-path and no custom security is defined
  - all the endpoints but */health* are automatically secured and CSRF is enabled

- If a custom *SecurityFilterChain* is defined the auto-configuration is not loaded and the endpoints must be manually secured

- CORS can be configured by using
  - *management.endpoints.web.cors.allowed-origins* *(<url> [,<url>])*
  - *management.endpoints.web.cors.allowed-methods* *(<method> [,<method>])*
  - *etc..*

Spring Monitoring – Exercise #1

- Activate the following endpoints and see the response data
  - Health
  - Info
  - Flyway
  - Env
  - Metrics
  - Beans
  - Mappings

- Branch: monitoring/exercise_1

Monitoring

- Intro
- Actuator
- **Metrics**
- Admin

Spring Actuator – Metrics

- Spring Actuator provides dependency management and auto-configuration for **Micrometer**
  - An application metric facade

Spring Actuator – Metrics

- Spring Actuator provides dependency management and auto-configuration for **Micrometer**
  - An application metric facade

- Supported Monitoring Systems (additional dependencies needed)
  - Dynatrace
  - Elastic
  - Prometheus
  - ...

Spring Actuator – Prometheus

- As metrics example will be used Prometheus
  - Because it exposes an endpoint to get the data and down not need other systems involved

Spring Actuator – Prometheus

- As metrics example will be used Prometheus
  - Because it exposes an endpoint to get the data and down not need other systems involved

- Get endpoint available at */actuator/prometheus*
  - After enabling and micrometer dependency added

Spring Actuator – Prometheus – Metrics

- System metrics
  - jvm_memory_[...], jvm_threads_[...], jvm_buffer_[...]
  - system_cpu_[...]
  - process_[...]
  - disk_[...]

Spring Actuator – Prometheus – Metrics

- System metrics
  - jvm_memory_[...], jvm_threads_[...], jvm_buffer_[...]
  - system_cpu_[...]
  - process_[...]
  - disk_[...]
- Database
  - spring_data_repository_[...]
  - hikaricp_connections_[...]

Spring Actuator – Prometheus – Metrics

- System metrics
  - jvm_memory_[...], jvm_threads_[...], jvm_buffer_[...]
  - system_cpu_[...]
  - process_[...]
  - disk_[...]
- Database
  - spring_data_repository_[...]
  - hikaricp_connections_[...]
- Server
  - tomcat_sessions_[...]
  - http_server_requests_[...]

Spring Monitoring – Exercise #2

- Play with the actuator and service endpoints to see the metrics change
- Branch: monitoring/exercise_2

Monitoring

- Intro
- Actuator
- Metrics
- **Admin**

Spring Boot Admin

- **Spring Boot Admin** is a web application used for monitoring and managing Spring application

Spring Boot Admin

- **Spring Boot Admin** is a web application used for monitoring and managing Spring application

- Each Application is a client and register itself to the admin server

Spring Boot Admin

- **Spring Boot Admin** is a web application used for monitoring and managing Spring application

- Each Application is a client and register itself to the admin server

- The monitoring and managing is given by **Spring Actuator**

Spring Boot Admin – Server

- The server can be enable with the annotation
  - @EnableAdminServer

Spring Boot Admin – Server

- The server can be enable with the annotation
  - @EnableAdminServer

- Spring Security must be configured
  - Default user or more specific way

Spring Boot Admin – Server

- The server can be enable with the annotation
  - @EnableAdminServer
- Spring Security must be configured
  - Default user or more specific way
- Spring Admin configuration must be set to allow client to register

Spring Boot Admin – Server

- The server can be enable with the annotation
  - @EnableAdminServer
- Spring Security must be configured
  - Default user or more specific way
- Spring Admin configuration must be set to allow client to register
- The UI Is then available at the main server path

Spring Boot Admin – Client

- Setup client info (from client to server)
  - Server url
  - Username/Password
  - Username/Password metadata (sent to server to connect to client)

Spring Boot Admin – Client

- Setup client info (from client to server)
  - Server url
  - Username/Password
  - Username/Password metadata (sent to server to connect to client)

- Spring Security must be configured
  - Default user or more specific way

Spring Boot Admin – Client

- Setup client info (from client to server)
  - Server url
  - Username/Password
  - Username/Password metadata (sent to server to connect to client)
- Spring Security must be configured
  - Default user or more specific way
- The actuator endpoints must be exposed and secured

Spring Boot Admin – UI

- The UI is available at the server context path

Spring Boot Admin – UI

- The UI is available at the server context path
- At */applications* are visible all the clients

Spring Boot Admin – UI

- The UI is available at the server context path
- At */applications* are visible all the clients
- At */instances/{id}* is possible to access single client data
  - */details* the overview with info, health, medadata, system statistics
  - */metrics* available metrics, it is possible to select which one to see
  - Etc depending on the exposed endpoints

Spring Boot Admin – Notifications

- It is possible to add a notification system
  - Email
  - PagerDuty
  - OpsGenie
  - HipChat
  - Slack
  - Let's Chat

Spring Monitoring – Exercise #3

- Play with the Admin UI
- Branch: monitoring/exercise_3

So long and thanks for all the fish

Feedback

Weblink: https://de.surveymonkey.com/r/B5NNSMV

Spring Security And Monitoring

Cyril Grossenbacher
Khaufra Maggini

ti&m