Architecture

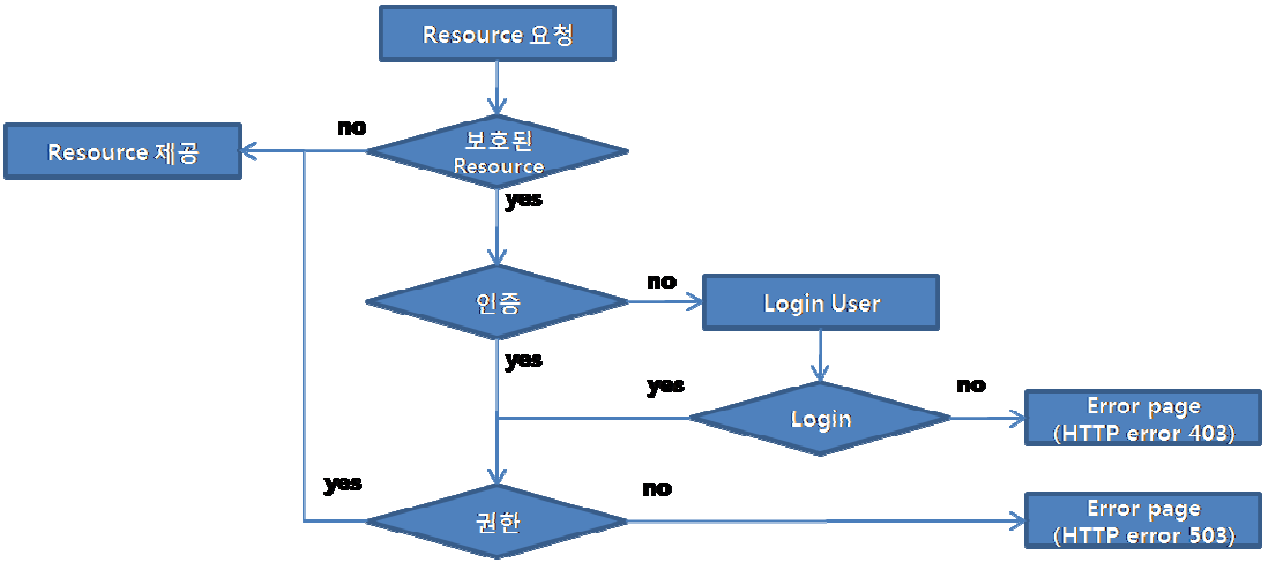
Summary

This document shows the e- government development framework's Spring Security basic architecture and environment configuration.
The Server Security in the e- government development framework uses DB based JDBC certification instead of XML based certification when the container starts to operate.

Description

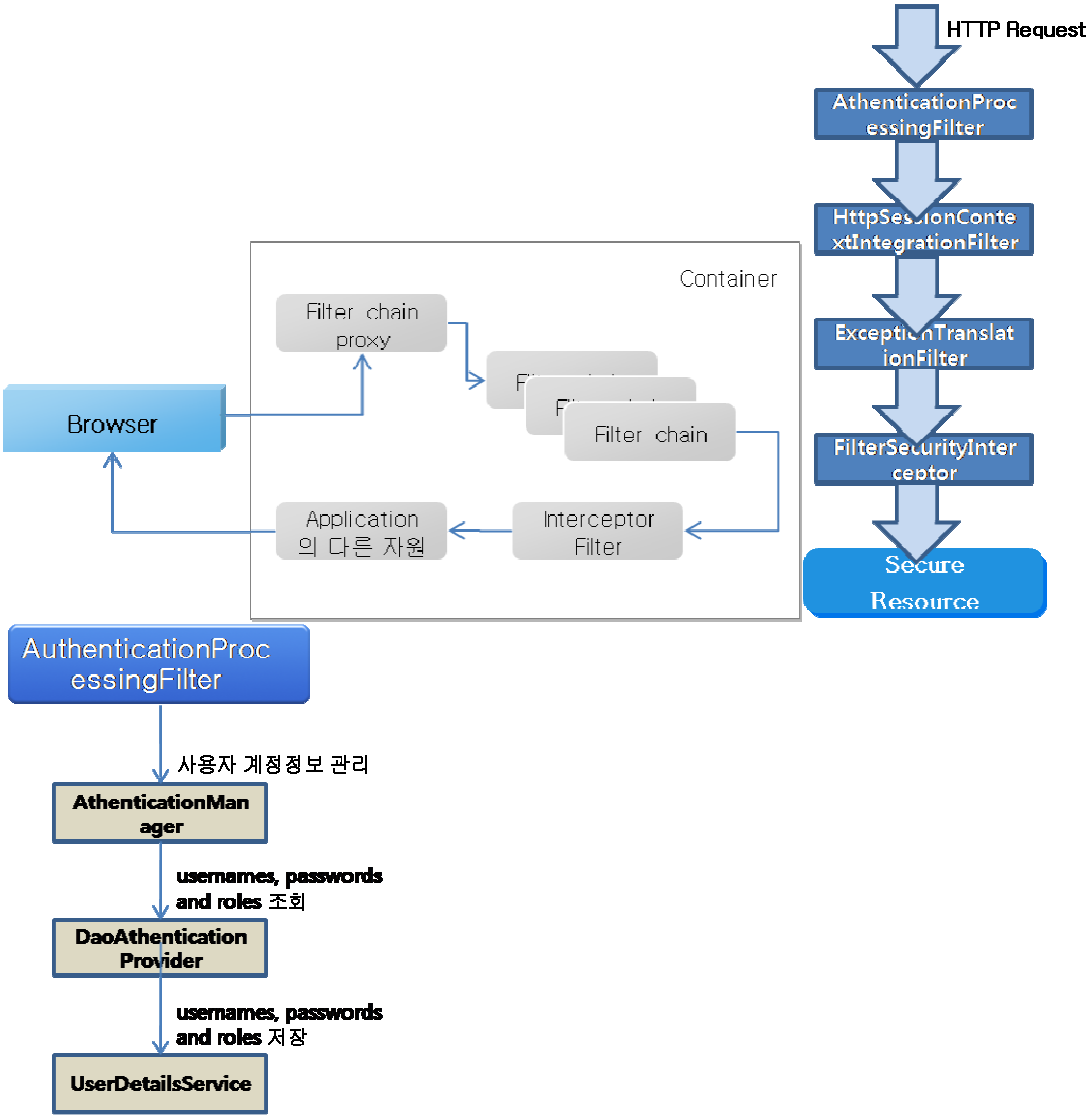
Spring Security architecture

Web application certification process



1. Click the link
2. Determine if the resource is being protected by a request.
3. Not certificated. redirect to HTTP response error or a specific page
4. web page login form or X509 certificate depending on the certification mechanism
5. Request the content of input form to HTTP post or HTTP header that contains certification details
6. Determine the credential is valid
    - Go the next process if valid
    - Re- request ID information if not valid (go back)
7. If there is a permission to access the protected resources, the request is successful. If there is not, it is the forbidden 403 HTTP error.

Spring Security Filter Chain



- Information created by Spring Security – Use SecurityContextHoler to get SecurityContext
- For use in various types environment (e.g. distributed application), SecurityContext is stored by using ThreadLocal object created within SecurityContextHolder.
- ThreadLocal object can contain the required status information only at the current thread.
- It is not suitable to re- create SecurityContext which carries out the same role every time the user requests in the web environment → Use httpSessionContextIntegrationFilter to carry out recording (bringing) SecurityContext information in ThreadLocal (stored in Session)

Security Configuration

Required libraries

- spring- security- core- 2.0.4.jar
- spring- security- taglibs- 2.0.4.jar

Register web.xml

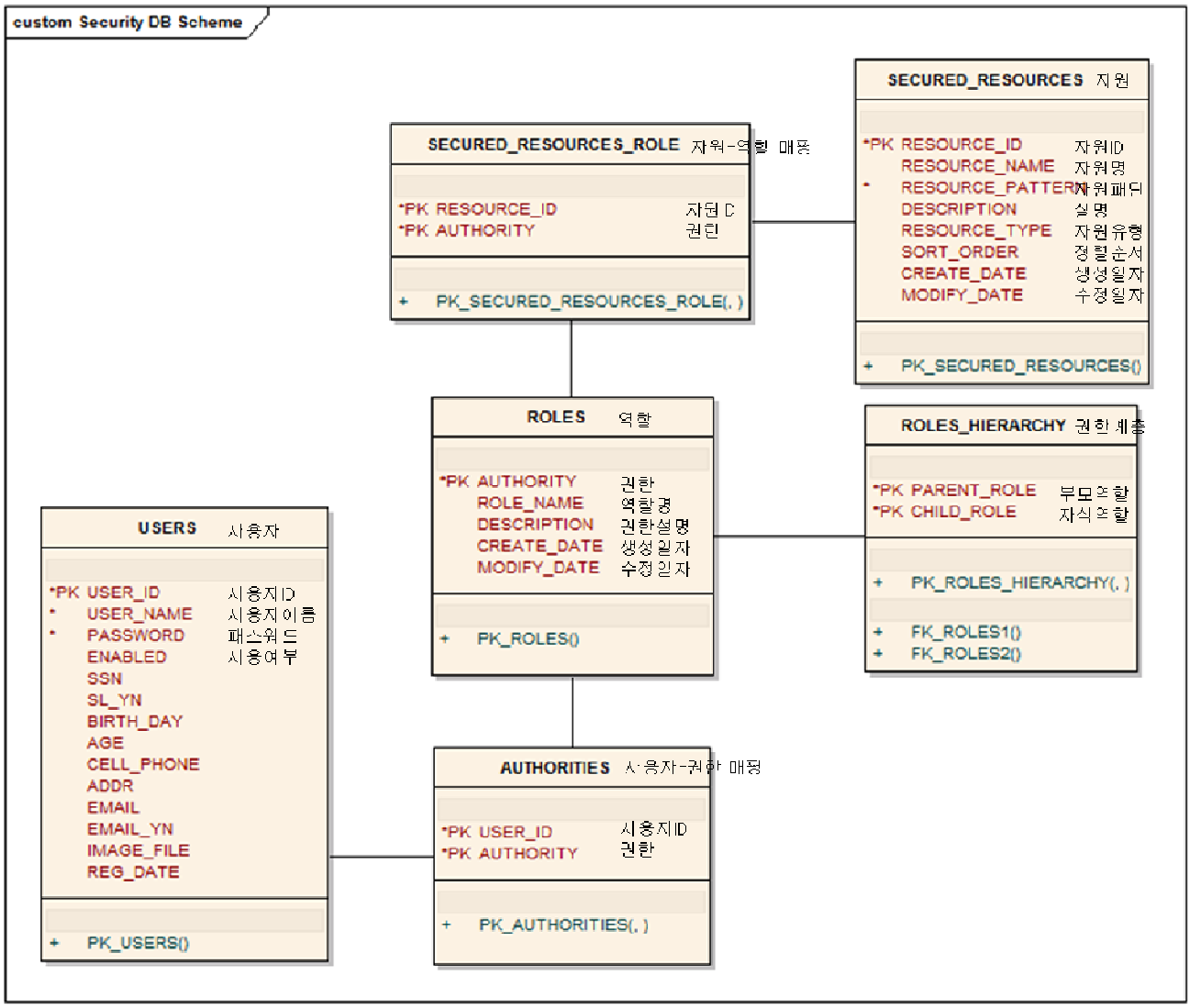
- Register org.springframework.web.filter.DelegatingFilterProxy: Spring Framework class that represents the filter implement registered as Spring bean in Application Context.
- Ensure that all web request be delivered to Spring Security's DelegatingFilterProxy.
- DelegatingFilterProxy is a general- purpose class ensuring that web requests will be delivered to different filters based on different URL patterns.
- These mandated filters are managed in the Application Context, thus enjoying the advantages of dependency injection.

example

```
<filter>
        <filter- name>springSecurityFilterChain</filter- name>
        <filter- class>org.springframework.web.filter.DelegatingFilterProxy</filter- class>
</filter>
<filter- mapping>
        <filter- name>springSecurityFilterChain</filter- name>
        <url- pattern>/*</url- pattern>
</filter- mapping>
```

Main tables

The user table and user permission table are related with user certification and the user permission table includes tables such as roles, resources, role layer, etc.



DaoAuthenticationProvidor

User table

```
CREATE TABLE USERS (
        USERNAME VARCHAR(50) NOT NULL,
        PASSWORD VARCHAR(50) NOT NULL,
        ENABLED BIT NOT NULL,
        CONSTRAINT PK_USERS PRIMARY KEY(USERNAME)
);
```

- Required field: USERNAME(user ID), PASSWORD(user password), ENABLED(account use)
- Other fields : use the remaining information of the table for session processing. (e.g. Resident Registration Number, address, birthday, etc.)

User permission table

```
CREATE TABLE AUTHORITIES (
        USERNAME VARCHAR(50) NOT NULL,
        AUTHORITY VARCHAR(50) NOT NULL,
        CONSTRAINT PK_AUTHORITIES PRIMARY KEY(USER_ID,AUTHORITY),
        CONSTRAINT FK_USERS FOREIGN KEY(USER_ID) REFERENCES USERS(USER_ID),
        CONSTRAINT FK_ROLES3 FOREIGN KEY(AUTHORITY) REFERENCES ROLES(AUTHORITY)
);
```

- Required filed : USER_ID(user ID), AUTHORITY(permission)

Role table

```
CREATE TABLE ROLES (
        AUTHORITY VARCHAR(50) NOT NULL,
        ROLE_NAME VARCHAR(50),
        DESCRIPTION VARCHAR(100),
        CREATE_DATE DATE,
        MODIFY_DATE DATE,
        CONSTRAINT PK_ROLES PRIMARY KEY(AUTHORITY)
);
```

AUTHORITY	***DESCRIPTION***
IS_AUTHENTICATED_ANONYMOUSLY	Anonymous user
IS_AUTHENTICATED_REMEMBERED	REMEMBERED user
IS_AUTHENTICATED_FULLY	Certified user
ROLE_RESTRICTED	Restricted user
ROLE_USER	user
ROLE_ADMIN	manager
ROLE_A	A business
ROLE_B	B business

Role layer table

Table that stores the layered structure of a role

```
CREATE TABLE ROLES_HIERARCHY (
        PARENT_ROLE VARCHAR(50) NOT NULL,
        CHILD_ROLE VARCHAR(50) NOT NULL,
        CONSTRAINT PK_ROLES_HIERARCHY PRIMARY KEY(PARENT_ROLE,CHILD_ROLE),
        CONSTRAINT FK_ROLES1 FOREIGN KEY(PARENT_ROLE) REFERENCES ROLES(AUTHORITY),
        CONSTRAINT FK_ROLES2 FOREIGN KEY(CHILD_ROLE) REFERENCES ROLES (AUTHORITY)
);
```

CHILD_ROLE	***PARENT_ROLE***
ROLE_ADMIN	ROLE_USER
ROLE_USER	ROLE_RESTRICTED
ROLE_RESTRICTED	IS_AUTHENTICATED_FULLY
IS_AUTHENTICATED_FULLY	IS_AUTHENTICATED_REMEMBERED
IS_AUTHENTICATED_REMEMBERED	IS_AUTHENTICATED_ANONYMOUSLY

ROLE_ADMIN	ROLE_A
ROLE_ADMIN	ROLE_B
ROLE_A	ROLE_RESTRICTED
ROLE_B	ROLE_RESTRICTED

Protected resource table

```
CREATE TABLE SECURED_RESOURCES (
        RESOURCE_ID VARCHAR(10) NOT NULL,
        RESOURCE_NAME VARCHAR(50),
        RESOURCE_PATTERN VARCHAR(300) NOT NULL,
        DESCRIPTION VARCHAR(100),
        RESOURCE_TYPE VARCHAR(10),
        SORT_ORDER INTEGER,
        CREATE_DATE DATE,
        MODIFY_DATE DATE,
        CONSTRAINT PK_RECURED_RESOURCES PRIMARY KEY(RESOURCE_ID)
);
```

Use url, method and pointcut to protect resources.

RESOURCE_ID	***RESOURCE_PATTERN***
web- 000001	\ A/test\ .do\ Z
web- 000002	\ A/sale/.*\ .do\ Z
web- 000003	\ A/cvpl/((?!EgovCvplLogin\ .do).)*\ Z
mtd- 000001	egovframework.rte.sample.service.EgovSampleService.updateSample
mtd- 000002	egovframework.rte.sample.service.EgovSampleService.deleteSample
mtd- 000003	execution(* egovframework.rte.sample..service.*Service.insert*(..))

Protected resource role table

Table mapped between protected resources and roles

```
CREATE TABLE SECURED_RESOURCES_ROLE (
        RESOURCE_ID VARCHAR(10) NOT NULL,
        AUTHORITY VARCHAR(50) NOT NULL,
        CONSTRAINT PK_SECURED_RESOURCES_ROLE PRIMARY KEY(RESOURCE_ID,AUTHORITY),
        CONSTRAINT FK_SECURED_RESOURCES FOREIGN KEY(RESOURCE_ID) REFERENCES
SECURED_RESOURCES(RESOURCE_ID),
        CONSTRAINT FK_ROLES4 FOREIGN KEY (AUTHORITY) REFERENCES ROLES(AUTHORITY)
);
```

N. References