레디스-세션-사용시-시큐리티-세션-이벤트-로깅-가이드

Table of Contens

레디스 세션

- 캐모마일은 기본 세션(WAS : Tomcat)에 의존하고 있습니다.
- 레디스 세션을 사용하기 위해서는 다음 의존성이 필요합니다.

```markdown
- spring-session-data-redis
```

> *spring-session 프로젝트는 서버에 저장되는 세션의 한계를 넘어 다양한 세션 관리를 위한 프로젝트이다.*

수정사항 1

- `HttpSessionEventPublisher` 가 정상적으로 등록되지 않는 문제

해결방안

- 다음 클래스 파일을 프로젝트에 추가합니다.
  - 클래스패스에서 덮어씁니다.

```java
package net.lotte.chamomile.configure.common;

import java.util.ArrayList;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.LocalVariableTableParameterNameDiscoverer;
import org.springframework.security.web.session.HttpSessionEventPublisher;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import net.lotte.chamomile.commons.aop.interceptor.ServiceLogInterceptor;
import net.lotte.chamomile.core.ApplicationContextProvider;
import net.lotte.chamomile.core.exception.web.DefaultExceptionResolver;
import net.lotte.chamomile.core.exception.web.ExceptionTransfer;
import net.lotte.chamomile.core.exception.web.service.ExceptionService;
import net.lotte.chamomile.core.servlet.mvc.method.annotation.ServiceRequestMappingHandlerMapping;
import net.lotte.chamomile.core.servlet.mvc.test.ServiceTestInterceptor;
import net.lotte.chamomile.core.web.servlet.handler.ServiceHandlerInterceptor;
import net.lotte.chamomile.integration.service.management.PropertyBasedServiceIdProvider;

@Configuration
public class WebConfigConfiguration implements WebMvcConfigurer {

        private final Logger logger = LoggerFactory.getLogger(getClass());
        @Value("${chmm.exception.errorPage:/error}")
        private String errorPage;

        @Value("${chmm.requestMappingSync.enabled:true}")
        private String requestMappingSyncEnabled;

        @Bean
        HttpSessionEventPublisher httpSessionEventPublisher() {
                return new HttpSessionEventPublisher();
        }

        @Bean
        @ConditionalOnProperty(name = "chmm.requestMappingSync.enabled", havingValue = "true", matchIfMissing = true)
        public ServiceRequestMappingHandlerMapping serviceRequestMappingHandlerMapping() {
                logger.info("register ServiceRequestMappingHandlerMapping");
                return new ServiceRequestMappingHandlerMapping();
        }
        @Bean
        public net.lotte.chamomile.commons.excel.MultiSheetExcelViewer multiSheetExcelViewer() {
                return new net.lotte.chamomile.commons.excel.MultiSheetExcelViewer();
        }

        @Bean
        public net.lotte.chamomile.commons.spreadsheet.MultiSheetExcelViewer multisheetSpreadSheetView() {
                return new net.lotte.chamomile.commons.spreadsheet.MultiSheetExcelViewer();
        }

        @Bean
        @ConditionalOnProperty(name = "chmm.requestMappingSync.enabled", havingValue = "true", matchIfMissing = true)
        public ServiceHandlerInterceptor serviceHandlerInterceptor() {
                logger.info("register ServiceHandlerInterceptor");
                return new ServiceHandlerInterceptor();
        }

        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(new ServiceLogInterceptor())
                        .addPathPatterns("/**")
                        .excludePathPatterns("/resources/**");
                if (Boolean.parseBoolean(requestMappingSyncEnabled)) {
                        registry.addInterceptor(new ServiceTestInterceptor())
                                .addPathPatterns("/**")
                                .excludePathPatterns("/resources/**");
                        registry.addInterceptor(serviceHandlerInterceptor())
                                .addPathPatterns("/**")
                                .excludePathPatterns("/resources/**");
                } else {
                        logger.info("requestMappingSyncEnabled is false({})", requestMappingSyncEnabled);
                }
        }

        @Bean
        public DefaultExceptionResolver defaultExceptionResolver(MessageSource messageSource,
                                                                 ExceptionService exceptionService,
                                                                 ExceptionTransfer exceptionLogInfo,
                                                                 ExceptionTransfer exceptionInfoSave) {

                List<ExceptionTransfer> exceptionTransfers = new ArrayList<>();
                exceptionTransfers.add(exceptionLogInfo);
                exceptionTransfers.add(exceptionInfoSave);

                DefaultExceptionResolver defaultExceptionResolver = new DefaultExceptionResolver();
                defaultExceptionResolver.setMessageSource(messageSource);
                defaultExceptionResolver.setExceptionService(exceptionService);
                defaultExceptionResolver.setExceptionTransfers(exceptionTransfers);
                defaultExceptionResolver.setErrorPage(errorPage);
                return defaultExceptionResolver;
        }

        @Bean
        public ApplicationContextProvider ApplicationContextProvider() {
                return new ApplicationContextProvider();
        }

        @Bean
        public LocalVariableTableParameterNameDiscoverer localVariableTableParameterNameDiscoverer() {
                return new LocalVariableTableParameterNameDiscoverer();
        }

        @Bean(name = "propertyServiceIdProvider")
        public PropertyBasedServiceIdProvider propertyServiceIdProvider() {
                return new PropertyBasedServiceIdProvider();
        }

}
```

Info

주요 변경사항은 `HttpSessionEventPublisher` 가 SevletListener 로 우선 등록 되어 있는 설정을 우회

language-diff

```diff
    @Bean
-   public ServletListenerRegistrationBean<HttpSessionEventPublisher> httpSessionEventPublisher() {
-       return new ServletListenerRegistrationBean<HttpSessionEventPublisher>(new HttpSessionEventPublisher());
+   public HttpSessionEventPublisher httpSessionEventPublisher() {
+           return new HttpSessionEventPublisher();
        }
```

수정사항 2

- LogOut, TimeOut 이벤트를 spring security의 SessionDestroyedEvent 로 감지할 수 없는 문제
  - 이유: Redis가 세션에 대한 이벤트를 발행하므로 RequestContextHolder 에서 로그아웃 요청 정보를 인지하지 못함.

해결방안

- 다음 클래스 프로젝트에 추가

language-java
```java
package net.lotte.chamomile.commons.aop.interceptor;

import java.util.List;

import javax.servlet.http.HttpSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.MDC;
import org.springframework.context.ApplicationListener;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.session.SessionDestroyedEvent;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;
import org.springframework.web.context.request.RequestContextHolder;

@Component
public class SessionDestoryListener implements ApplicationListener<SessionDestroyedEvent> {

        final Logger logger = LoggerFactory.getLogger("UserLoginLogger");

        @Override
        public void onApplicationEvent(SessionDestroyedEvent event) {

                List<SecurityContext> contexts = event.getSecurityContexts();

                UserDetails ud;
                if (contexts.isEmpty() == false) {
                        for (SecurityContext ctx : contexts) {

                                ud = (UserDetails) ctx.getAuthentication().getPrincipal();

                                if (MDC.get("userAction") == null) {
                                        userAction();
                                }

                                String sessionId = ((HttpSession) event.getSource()).getId();
                                String userName = ud.getUsername();
                                MDC.put("userName", userName);
                                MDC.put("sessionId", sessionId);
                                logger.trace("");
                        }
                }

        }

        private void userAction() {
                if (RequestContextHolder.getRequestAttributes() == null) {
                        MDC.put("userAction", "TimeOut");
                } else {
                        MDC.put("userAction", "LogOut");
                }
        }

}
```

Info

캐모마일에서 수정된 사항은

language-java
```java
if (MDC.get("userAction") == null) {
    userAction();
}
```

이미 userAction 정보가 있으면 해당 내용에 대해 판단하지 않도록 함.

Spring Security 의 세션 이벤트에서는 timeout, logout 을 구별할 수 없음.

- Redis Sesseion Event 를 받는 리스너 추가

```java
package net.lotte.sample;

import org.slf4j.MDC;
import org.springframework.context.ApplicationListener;
import org.springframework.session.events.SessionDeletedEvent;
import org.springframework.session.events.SessionDestroyedEvent;
import org.springframework.session.events.SessionExpiredEvent;
import org.springframework.stereotype.Component;

@Component
public class RedisSessionDestroyListener implements ApplicationListener<SessionDestroyedEvent> {

	@Override
	public void onApplicationEvent(SessionDestroyedEvent event) {
		if (event instanceof SessionDeletedEvent) {
			MDC.put("userAction", "LogOut");
		}
		if (event instanceof SessionExpiredEvent) {
			MDC.put("userAction", "TimeOut");
		}
	}
}
```

Info

spring session 의 세션 이벤트에서는 Delete, Expried 를 구별할 수 있으나, 사용자 이름 등 을 확인 할 수 없으므로 로그를 출력하는 부분은 SessionDestoryListener 에게 위임.