Spring Study - 스프링 핵심 원리 이해

스프링 핵심 원리 이해 4

Platform Engineering 김보배

모든 Bean 조회

```java
package com.test.proxy;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.ApplicationContext;

@SpringBootTest
public class StudyTest {

    @Autowired
    ApplicationContext ac;

    @DisplayName("모든 빈을 조회한다")
    @Test
    void getAllBean() {
        String[] allBeanNames = ac.getBeanDefinitionNames();
        for (String beanName : allBeanNames) {
            System.out.println(beanName);
        }
    }

}
```

같은 타입을 반환하는 Bean이 있다면 어떻게 될까?

```java
public class StudyTest {

    AnnotationConfigApplicationContext ac = new AnnotationConfigApplicationContext(TestConfig.class);

    @Test
    @DisplayName("부모 타입으로 조회시, 자식이 둘 이상 있으면, 중복 오류가 발생한다")
    void findBeanByParentTypeDuplicate() {
        assertThrows(NoUniqueBeanDefinitionException.class, () -> ac.getBean(TestService.class));
    }

    @Test
    @DisplayName("부모 타입으로 조회시, 자식이 둘 이상 있으면, 빈 이름을 지정하면 된다")
    void findBeanByParentTypeBeanName() {
        TestService testService1 = ac.getBean("testServiceImpl1", TestService.class);
        TestService testService2 = ac.getBean("testServiceImpl2", TestService.class);

        assertThat(testService1).isInstanceOf(TestServiceImpl1.class);
        assertThat(testService2).isInstanceOf(TestServiceImpl2.class);
    }


    @Configuration
    static class TestConfig {

        @Bean
        public TestService testServiceImpl1() {
            return new TestServiceImpl1();
        }

        @Bean
        public TestService testServiceImpl2() {
            return new TestServiceImpl2();
        }

    }
}
```

같은 Type bean이 중복됐어도 나는 특정 Bean 기본으로 사용할거야 !

```java
public class StudyTest {

    AnnotationConfigApplicationContext ac = new AnnotationConfigApplicationContext(TestConfig.class);

    @Test
    @DisplayName("자식이 2개 이상일 때 부모 타입으로 조회하더라도 Primary bean이 존재하면, 오류가 발생하지 않는다")
    void findPrimaryBeanByParentType() {
        TestService testService = ac.getBean(TestService.class);
        assertThat(testService).isInstanceOf(TestServiceImpl1.class);
    }

    @Configuration
    static class TestConfig {

        @Primary
        @Bean
        public TestService testServiceImpl1() {
            return new TestServiceImpl1();
        }

        @Bean
        public TestService testServiceImpl2() {
            return new TestServiceImpl2();
        }

    }
}
```

실제 환경에선 Bean이 중복되면 어떻게 될까 ?

```java
@Configuration
public class RestTemplateConfig {

    @Bean
    public RestTemplate restTemplateNoProxy() {
        return new RestTemplate();
    }

    @Bean
    public RestTemplate restTemplateProxy() {
        var proxy = new HttpHost("www.proxy.com", 9999);
        var routePlanner = new DefaultProxyRoutePlanner(proxy);
        var httpClient = HttpClients.custom()
                .setRoutePlanner(routePlanner)
                .build();

        var factory = new HttpComponentsClientHttpRequestFactory();
        factory.setHttpClient(httpClient);
        return new RestTemplate(factory);
    }

}
```

(보통) 실제환경에선 ApplicationContext로 모든 bean을 확인하진 않지

-> 직접 중복된 bean을 만들고 주입받아 보자

```java
@Service
public class SomeService {

    private final RestTemplate client;

    public SomeService(RestTemplate client) {
        this.client = client;
    }

}
```

Could not autowire. There is more than one bean of 'RestTemplate' type.
Beans: restTemplateNoProxy (RestTemplateConfig.java)
restTemplateProxy (RestTemplateConfig.java)

Add qualifier ⌥⇧↵ More actions... ⌥↵

Use Qualifier !

```java
@Configuration
public class RestTemplateConfig {

    @Bean("noproxy")
    public RestTemplate restTemplateNoProxy() {
        return new RestTemplate();
    }

    @Bean("proxy")
    public RestTemplate restTemplateProxy() {
        var proxy = new HttpHost("http://proxy_server.com", 9999);
        var routePlanner = new DefaultProxyRoutePlanner(proxy);
        var httpClient = HttpClients.custom()
                .setRoutePlanner(routePlanner)
                .build();

        var factory = new HttpComponentsClientHttpRequestFactory();
        factory.setHttpClient(httpClient);
        return new RestTemplate(factory);
    }

}
```

```java
@Service
public class OtherService {

    private final RestTemplate client;

    public OtherService(@Qualifier("noproxy") RestTemplate client) {
        this.client = client;
    }


    public String sendRequest() {
        return this.client.getForObject("https://www.naver.com", String.class);
    }

}
```

```java
@Service
public class SomeService {

    private final RestTemplate client;

    public SomeService(@Qualifier("proxy") RestTemplate client) {
        this.client = client;
    }


    public String sendRequest() {
        return this.client.getForObject("https://www.naver.com", String.class);
    }

}
```

많은 의존성 추가 -> 중복되는 Configuration, Bean이 생길텐데 ....

```java
@Configuration(proxyBeanMethods = false)
@ConditionalOnClass(ApacheHttpClient.class)
@ConditionalOnMissingBean(CloseableHttpClient.class)
@ConditionalOnProperty(value = "feign.httpclient.enabled", matchIfMissing = true)
@Conditional(HttpClient5DisabledConditions.class)
protected static class HttpClientFeignConfiguration {

    private final Timer connectionManagerTimer = new Timer(
            "FeignApacheHttpClientConfiguration.connectionManagerTimer", true);

    @Autowired(required = false)
    private RegistryBuilder registryBuilder;

    private CloseableHttpClient httpClient;

    @Bean
    @ConditionalOnMissingBean(HttpClientConnectionManager.class)
    public HttpClientConnectionManager connectionManager(
            ApacheHttpClientConnectionManagerFactory connectionManagerFactory,
            FeignHttpClientProperties httpClientProperties) {
        final HttpClientConnectionManager connectionManager = connectionManagerFactory.newConnectionManager(
                httpClientProperties.isDisableSslValidation(), httpClientProperties.getMaxConnections(),
                httpClientProperties.getMaxConnectionsPerRoute(), httpClientProperties.getTimeToLive(),
                httpClientProperties.getTimeToLiveUnit(), this.registryBuilder);
        this.connectionManagerTimer.schedule(new TimerTask() {
            @Override
            public void run() {
                connectionManager.closeExpiredConnections();
            }
        }, 30000, httpClientProperties.getConnectionTimerRepeat());
        return connectionManager;
    }

  ...
}
```

Conditional

@Conditional

@ConditionalOnClass

@ConditionalOnBean

@ConditionalOnMissingClass

@ConditionalOnMissingBean

@ConditionOnProperty

@ConditionOnResource

...

Ex) FeignAutoConfiguration - HttpClientFeignConfiguration

대....박인데?
이젠 원하는 부분이 있다면 bean을 갈아끼워서 활용도 할 수 있겠네...?