HOL : Spring Object Lifecycle

Task 1. Lab

1. In Package Explorer > right-click > New > Java Project
   1)Project Name : SpringLifecycle
   2)JRE
      -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
   3)Uncheck [Create module-info.java file]
   4)Next
   5)Finish

2. src > right-click > New > Package
   1)Name : com.example
   2)Finish

3. POJO 객체 생성
   1)src/com.example > New > Class
   2)Name : Student

```
package com.example;

import java.util.List;

public class Student {
   private String name;
   private int age;
   private List<String> hobbys;
   private double height;
   private double weight;
}
```

4. Java Project를 Spring Project로 변환
   1)SpringLifecycle Project > right-click > Configure > Convert to Maven Project
      -Project : /SpringLifecycle
      -Group Id : SpringLifecycle
      -Artifact Id : SpringLifecycle
      -version : 0.0.1-SNAPSHOT
      -Packaging : jar
      -Finish

   2)SpringLifecycle Project > right-click > Spring > Add Spring Project Nature

   3)pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.10</version>
   </dependency>
</dependencies>
```

   4)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

5. Lombok library 추가
1)https://mvnrepository.com/에서 'lombok'으로 검색
2)'Project Lombok' click
3)1.18.20 click
4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.10</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

5)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS 확인

6. Student.java 수정
1)Student.java

```
package com.example;

import java.util.List;

import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@RequiredArgsConstructor
@ToString
public class Student {
  @NonNull private String name;
  @NonNull private Integer age;
  @NonNull private List<String> hobbys;
  @Setter private double height;
  @Setter private double weight;
}
```

7. SpringLifecycle/resources folder 생성
1)SpringLifecycle project > right-click > New > Source Folder
2)Folder name : resources
3)Finish

8. Bean Configuration XML 작성
1)SpringLifecycle/resources > right-click > New > Spring Bean Configuration File
2)File name : applicationContext.xml
3)Namespaces에서 util check할 것
3)Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:util="http://www.springframework.org/schema/util"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="student1" class="com.example.Student">
      <constructor-arg value="백두산" />
      <constructor-arg value="25" />
      <constructor-arg>
         <util:list>
            <value>독서</value>
            <value>영화감상</value>
            <value>요리</value>
         </util:list>
      </constructor-arg>
      <property name="height" value="165" />
      <property name="weight">
         <value>45</value>
      </property>
   </bean>
</beans>
```

9. com.example.MainClass.java

```
package com.example;

import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
   public static void main(String[] args) {
      GenericXmlApplicationContext context = new
      GenericXmlApplicationContext();

      context.load("classpath:applicationContext.xml");
      context.refresh();

      Student student1 = context.getBean("student1", Student.class);
      System.out.println(student1);

      context.close();
   }
}
```

10. 실행
1)MainClass > right-click > Run As > Java Application
Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0, weight=45.0)

11. Java Annotation 방식 사용하기

1)src/com.example > right-click > New > Class

2)Name : ApplicationConfig

```
package com.example;

import java.util.Arrays;
import java.util.List;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ApplicationConfig {
   @Bean
   public Student student1() {
      List<String> list = Arrays.asList("독서", "영화감상", "요리");
      Student student = new Student("백두산", 25, list);
      student.setHeight(165.0);
      student.setWeight(45.0);
      return student;
   }
}
```

3)com.example.MainClass2 생성

-src/com.example > right-click > New > Class

-Name : MainClass2

```
package com.example;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext
;

public class MainClass2 {
   public static void main(String[] args) {
      AnnotationConfigApplicationContext ctx = new
      AnnotationConfigApplicationContext(ApplicationConfig.class);
      Student student1 =  ctx.getBean("student1", Student.class);
      System.out.println(student1);

      Student student2 =  ctx.getBean("student1", Student.class);
      System.out.println(student1 == student2);
      ctx.close();
   }
}
```

4)실행 결과

```
Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,
weight=45.0)
true
```

12. @Component 방식으로 Student 객체 변경

1)Student.java

```
package com.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import lombok.Getter;
import lombok.ToString;

@ToString
@Getter
@Component(value = "student1")
public class Student {
    @Value("한라산")
    private String name;
    @Value("35")
    private Integer age;
    @Value("등산, 게임, 독서")
    private List<String> hobbys;
    @Value("162.5")
    private double height;
    @Value("49.2")
    private double weight;
}
```

2)com.example.ApplicationConfig 변경

```
package com.example;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan(basePackages = {"com.example"})
public class ApplicationConfig {}
```

3)resources/applicationContext.xml 삭제

4)com.example.MainClass2 실행 결과

```
Student(name=한라산, age=35, hobbys=[등산, 게임, 독서], height=162.5,
weight=49.2)
true
```

13. JUnit 5를 사용한 DI test class(JUnit5Test.java) 작성

1)src > right-click > New > Package

2)Name : com.example.test

3)Finish

4)com.example.test > right-click > New > JUnit Test Case

5)Select [New JUnit Jupiter test]

6)Name : JUnit5Test

7)Finish

8)Select [Perform the following action: Add Juni 5 library to the build path
9)OK

```
package com.example.test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.example.ApplicationConfig;
import com.example.Student;

class JUnit5Test {
   private AnnotationConfigApplicationContext context;

   @BeforeEach
   public void init() {
      this.context = new
      AnnotationConfigApplicationContext(ApplicationConfig.class);
   }

   @Test
   public void test1() {
      Student student = this.context.getBean("student1", Student.class);
      assertEquals("한라산", student.getName());
   }
}
```

4)right-click > Run As > JUnit Test
5)결과 -> Junit View에 초록색 bar

Task 2. Lab

1. In Package Explorer > right-click > New > Java Project
   1)Project Name : SpringLifecycle1
   2)JRE
      -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
   3)Uncheck [Create module-info.java file]
   4)Next
   5)Finish

2. src > right-click > New > Package
   1)Package name : com.example

3. POJO 객체 생성
   1)com.example > right-click > New > Class
   2)Name : Student.java

```
package com.example;
```

```
public class Student{
   private String name;
   private int age;
}
```

4. Java Project를 Spring Project로 변환

1)SpringLifecycle1 Project > right-click > Configure > Convert to Maven Project

-Project : /SpringLifecycle1
-Group Id : SpringLifecycle1
-Artifact Id : SpringLifecycle1
-version : 0.0.1-SNAPSHOT
-Packaging : jar
-Finish

2)SpringLifecycle1 Project > right-click > Spring > Add Spring Project Nature

3)pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.10</version>
   </dependency>
</dependencies>
```

4)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

5. Lombok library 추가

1)https://mvnrepository.com/에서 'lombok'으로 검색
2)'Project Lombok' click
3)1.18.20 click
4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.10</version>
   </dependency>
   <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
   <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.20</version>
      <scope>provided</scope>
   </dependency>
</dependencies>
```

5)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

6. Student.java lombok Annotation 붙이고, InitializingBean, DisposableBean interface 구현하기

```
package com.example;

import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

import lombok.AllArgsConstructor;
import lombok.ToString;

@AllArgsConstructor
@ToString
public class Student implements InitializingBean, DisposableBean{
   private String name;
   private int age;

   @Override
   public void destroy() throws Exception {
      System.out.println("방금 Bean이 소멸됐습니다.");
   }
   @Override
   public void afterPropertiesSet() throws Exception {
      System.out.println("방금 Bean이 생성됐습니다.");
   }
}
```

7. SpringLifecycle1/resources folder 생성
   1)SpringLifecycle1 project > right-click > New > Source Folder
   2)Folder name : resources
   3)Finish

8. Bean Configuration XML 작성
   1)SpringLifecycle1/resources > right-click > New > Spring Bean Configuration File
   2)File name : applicationContext.xml
   3)Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="student" class="com.example.Student">
      <constructor-arg value="백두산" />
      <constructor-arg value="25" />
   </bean>

</beans>
```

9. MainClass 작성
   1)com.example > right-click > New > Class
   2)Name : MainClass

3)Finish

```
package com.example;

import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        GenericXmlApplicationContext context = new
        GenericXmlApplicationContext();
        context.load("classpath:applicationContext.xml");
        context.refresh();

        Student student = context.getBean("student", Student.class);
        System.out.println(student);
        context.close();
    }
}
```

10. 실행

1)MainClass > right-click > Run As > Java Application

```
방금 Bean이 생성됐습니다.
Student (name=백두산, age=25)
방금 Bean이 소멸됐습니다.
```

11. @PostConstruct, @PreDestroy 이용하기

1)https://mvnrepository.com에서 'javax annotation'으로 검색

2)[Javax Annoation API] click

3)1.3.2 click

4)dependency copy하여 pom.xml에 paste

```
<!--
https://mvnrepository.com/artifact/javax.annotation/javax.annotation-api -->
<dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>javax.annotation-api</artifactId>
    <version>1.3.2</version>
</dependency>
```

5)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

6)Student2 class 생성하기

-com.example > right-click > New > Class

-Name : Student2

```
package com.example;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;

import lombok.AllArgsConstructor;
import lombok.ToString;
```

```
@AllArgsConstructor
@ToString
public class Student2 {
   private String name;
   private int age;

   @PostConstruct // Bean이 생성단계에서 해야할 일 기술
   public void initTest() {
      System.out.println("방금 객체가 생성됐습니다.");
   }

   @PreDestroy // Bean이 소멸할 때 해야할 일 기술
   public void destroyTest() {
      System.out.println("방금 객체가 소멸됐습니다.");
   }
}
```

12. resources/applicationContext.xml 수정하기
   1)Namespaces Tab click
   2)[context] check

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd">

   <!-- 첫번째 방법 -->
   <context:annotation-config/>
   <bean id="student2" class="com.example.Student2">
      <constructor-arg value="백두산" />
      <constructor-arg value="25" />
   </bean>

   <!-- 두번째 방법-->
   <bean
   class="org.springframework.context.annotation.CommonAnnotationBeanPostPr
   ocessor" />

   <bean id="student2" class="com.example.Student2">
      <constructor-arg value="백두산" />
      <constructor-arg value="25" />
   </bean>

   <!-- 세번째 방법 -->
   <bean id="student2" class="com.example.Student2" init-method="initTest"
   destroy-method="destroyTest">
      <constructor-arg value="백두산" />
      <constructor-arg value="25" />
   </bean>
</beans>
```

13. MainClass 수정
   1)com.example.MainClass.java

```
package com.example;

import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        GenericXmlApplicationContext context = new
        GenericXmlApplicationContext();
        context.load("classpath:applicationContext.xml");
        context.refresh();

        Student2 student2 = context.getBean("student2", Student2.class);
        System.out.println(student2);
        context.close();
    }
}
```

2)실행
-MainClass > right-click > Run As > Java Application

```
방금 객체가 생성됐습니다.
Student2(name=백두산, age=25)
방금 객체가 소멸됐습니다.
```

Task 3. Lab

1. In Package Explorer > right-click > New > Java Project
   1)Project Name : SpringScopeDemo
   2)JRE
      -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
   3)Uncheck [Create module-info.java file]
   4)Next
   5)Finish

2. src > right-click > New > Package
   1)Name : com.example
   2)Finish

3. com.example.Student class 생성
   1)com.example > right-click > New > Class
   2)Name : Student
   3)Finish

```
package com.example;

public class Student{
    private String name;
    private int age;
}
```

4. Java Project를 Spring Project로 변환
   1)SpringScopeDemo Project > right-click > Configure > Convert to Maven Project
      -Project : /SpringScopeDemo

-Group Id : SpringScopeDemo
-Artifact Id : SpringScopeDemo
-version : 0.0.1-SNAPSHOT
-Packaging : jar
-Finish

2)SpringScopeDemo Project > right-click > Spring > Add Spring Project Nature

3)pom.xml 파일에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.10</version>
  </dependency>
</dependencies>
```

4)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS 확인

5. Lombok library 추가
1)https://mvnrepository.com/에서 'lombok'으로 검색
2)'Project Lombok' click
3)1.18.20 click
4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.10</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

5)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS 확인

6. Student.java lombok Annotation 붙이기

```
package com.example;

import lombok.AllArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@AllArgsConstructor
```

```
654     @Setter
655     @ToString
656     public class Student {
657        private String name;
658        private int age;
659     }
660
661
662  7. SpringLifecycle1/resources folder 생성
663     1)SpringLifecycle1 project > right-click > New > Source Folder
664     2)Folder name : resources
665     3)Finish
666
667
668  8. Bean Configuration XML 작성
669     1)SpringLifecycle1/resources > right-click > New > Spring Bean Configuration File
670     2)File name : applicationContext.xml
671     3)Finish
672
673        <?xml version="1.0" encoding="UTF-8"?>
674        <beans xmlns="http://www.springframework.org/schema/beans"
675              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
676              xsi:schemaLocation="http://www.springframework.org/schema/beans
                 http://www.springframework.org/schema/beans/spring-beans.xsd">
677
678           <bean id="student" class="com.example.Student" scope="singleton">
679              <constructor-arg value="백두산" />
680              <constructor-arg value="25" />
681           </bean>
682
683        </beans>
684
685
686  9. MainClass 생성하기
687     1)com.example > right-click > New > Class
688     2)Name : MainClass
689
690        package com.example;
691
692        import org.springframework.context.support.AbstractApplicationContext;
693        import org.springframework.context.support.GenericXmlApplicationContext;
694
695        public class MainClass {
696           public static void main(String[] args) {
697              AbstractApplicationContext context = new
                 GenericXmlApplicationContext("classpath:applicationContext.xml");
698
699              Student student = context.getBean("student", Student.class);
700              System.out.println(student);
701              System.out.println("----------------------------------");
702
703              Student student1 = context.getBean("student", Student.class);
704              student1.setName("북한산");
705              student1.setAge(55);
706              System.out.println(student1);
707              System.out.println("----------------------------------");
```

```
            if(student.equals(student1)) System.out.println("Equals");  //Print Equals
            else System.out.println("Different");
            context.close();
        }
    }
```

10. Java Application 실행 결과

```
Student [name=백두산, age=25]
---------------------------------
Student [name=북한산, age=55]
---------------------------------
Equals
```

11. ApplicationConfig와 MainClass2 생성하기

1)com.example > right-click > New > Class
2)Name : ApplicationConfig
3)Finish

```
package com.example;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component
public class ApplicationConfig {
    @Bean
    @Scope("prototype")
    public Student student() {
        Student student = new Student("한라산", 35);
        return student;
    }
}
```

4)com.example > right-click > New > Class
5)Name : MainClass2
6)Finish

```
package com.example;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass2 {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context =
                new AnnotationConfigApplicationContext(ApplicationConfig.class);

        Student student = context.getBean("student", Student.class);
        System.out.println(student);
        System.out.println("---------------------------------");

        Student student1 = context.getBean("student", Student.class);
```

```
            student1.setName("북한산");
            student1.setAge(55);
            System.out.println(student1);
            System.out.println("---------------------------------");

            if(student.equals(student1)) System.out.println("Equals");  //Print Equals
            else System.out.println("Different");
            context.close();
        }
    }
```

7)실행결과

```
Student(name=한라산, age=35)
---------------------------------
Student(name=북한산, age=55)
---------------------------------
Different
```