1. Lifecycle
  1)Spring Container 생성
    GenericXmlApplicationContext context = new GenericXmlApplicationContext();

  2)Spring Container 설정
    context.load("classpath:applicationContext.xml");
    context.refresh(); //생성자를 사용하지 않을 때는 반드시 refresh() 할 것

  3)Spring Container 사용
    Student student = context.getBean("student", Student.class);
    System.out.println(student);

  4)Spring Container 종료
    context.close();

2. Lab
  1)In Package Explorer > right-click > New > Java Project
    -Project Name : SpringLifecycle

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

  3)com.example.Student.java

```
    package com.example;

    import java.util.ArrayList;

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

      public Student(String name, int age, ArrayList<String> hobbys) {
        this.name = name;
        this.age = age;
        this.hobbys = hobbys;
      }

      public void setName(String name) {
        this.name = name;
      }

      public void setAge(int age) {
        this.age = age;
      }

      public void setHobbys(ArrayList<String> hobbys) {
        this.hobbys = hobbys;
      }

```

```
55        public void setHeight(double height) {
56          this.height = height;
57        }
58
59        public void setWeight(double weight) {
60          this.weight = weight;
61        }
62
63        @Override
64        public String toString() {
65          return String.format("Student [name=%s, age=%s, hobbys=%s, height=%s,
            weight=%s]", name, age, hobbys, height,
66              weight);
67        }
68      }
69
70    4)Java Project를 Spring Project로 변환
71      -SpringLifecycle Project > right-click > Configuration > Convert to Maven Project
72        --Project : /SpringLifecycle
73        --Group Id : SpringLifecycle
74        --Artifact Id : SpringLifecycle
75        --version : 0.0.1-SNAPSHOT
76        --Packaging : jar
77        --Finish
78        --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
79
80      -SpringLifecycle Project > right-click > Spring > Add Spring Project Nature
81        --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
82
83      -pom.xml 파일에 Spring Context Dependency 추가하기
84        <version>0.0.1-SNAPSHOT</version>
85        <dependencies> <--- dependencies element 추가
86          <dependency> <---여기에 paste
87            <groupId>org.springframework</groupId>
88            <artifactId>spring-context</artifactId>
89            <version>4.3.24.RELEASE</version>
90          </dependency>
91        </dependencies>
92
93      -pom.xml > right-click > Run As > Maven install
94        [INFO] BUILD SUCCESS 확인
95
96    5)SpringLifecycle/resources folder 생성
97      -SpringLifecycle project > right-click > Build Path > Configure Build Path
98      -Source Tab > Add Folder
99      -SpringLifecycle click
100     -Create New Folder > Folder name : resources > Finish > OK
101     -SpringLifecycle/resources(new) 확인
102     -Apply and Close
103
104   6)Bean Configuration XML 작성
105     -SpringLifecycle/resources > right-click > New > Other > Spring > Spring Bean Configuration
        File
106     -File name : applicationContext.xml > Finish
```

```
107
108     <?xml version="1.0" encoding="UTF-8"?>
109     <beans xmlns="http://www.springframework.org/schema/beans"
110       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
111       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
112
113       <bean id="student1" class="com.example.Student">
114         <constructor-arg value="한지민" />
115         <constructor-arg value="25" />
116         <constructor-arg>
117           <list>
118             <value>독서</value>
119             <value>영화감상</value>
120             <value>요리</value>
121           </list>
122         </constructor-arg>
123         <property name="height" value="165" />
124         <property name="weight">
125           <value>45</value>
126         </property>
127       </bean>
128     </beans>
129
130   7)com.example.MainClass.java
131     package com.example;
132
133     import org.springframework.context.support.GenericXmlApplicationContext;
134
135     public class MainClass {
136       public static void main(String[] args) {
137         GenericXmlApplicationContext context = new GenericXmlApplicationContext();
138
139         context.load("classpath:ApplicationContext.xml");
140         context.refresh();
141
142         Student student1 = context.getBean("student1", Student.class);
143         System.out.println(student1);
144
145         context.close();
146       }
147     }
148
149   8)실행
150     -MainClass > right-click > Run As > Java Application
151
152
153 3. Spring Bean Lifecycle
154   1)context.refresh() 할 때 bean 초기화 과정에서는 InitializingBean interface의 afterPropertiesSet() 또는
      @PostConstruct annotation 을 호출
155     -ctx.load("classpath:applicationContext.xml"); 에서 Bean이 초기화되고,
156     -ctx.refresh() 할 때 Bean이 생성된다.
157     -즉 Bean이 초기화될 때 afterPropertiesSet()이 호출된다.
158
```

2)context.close() 할 때 bean 소멸 과정에서는 DisposableBean interface의 destory() 또는 @PreDestroy() 를 호출

3)만일 close()할 때 컨테이더가 소멸되면 그 안의 모든 bean은 자동 소멸된다.
-반면 bean만 소멸하고자 한다면 student.destory()를 이용하면 된다.

4. Lab

1)In Package Explorer > right-click > New > Java Project
-Project Name : SpringLifecycle1

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

3)InitializingBean, DisposableBean interface 이용하기
-com.example.Student.java

```
package com.example;

import java.util.ArrayList;

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

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

  public Student(String name, int age) {
    this.name = name;
    this.age = age;
  }

  @Override
  public String toString() {
    return String.format("Student [name=%s, age=%s]", name, 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.");
  }
}
```

4)Java Project를 Spring Project로 변환
-SpringLifecycle1 Project > right-click > Configuration > Convert to Maven Project
--Project : /SpringLifecycle1
--Group Id : SpringLifecycle1
--Artifact Id : SpringLifecycle1
--version : 0.0.1-SNAPSHOT

```
212         --Packaging : jar
213         --Finish
214         --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
215
216       -SpringLifecycle1 Project > right-click > Spring > Add Spring Project Nature
217         --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
218
219       -pom.xml 파일에 Spring Context Dependency 추가하기
220         <version>0.0.1-SNAPSHOT</version>
221         <dependencies> <--- dependencies element 추가
222           <dependency> <---여기에 paste
223             <groupId>org.springframework</groupId>
224             <artifactId>spring-context</artifactId>
225             <version>4.3.24.RELEASE</version>
226           </dependency>
227         </dependencies>
228
229       -pom.xml > right-click > Run As > Maven install
230         [INFO] BUILD SUCCESS 확인
231
232     5)SpringLifecycle1/resources folder 생성
233       -SpringLifecycle1 project > right-click > Build Path > Configure Build Path
234       -Source Tab > Add Folder
235       -SpringLifecycle1 click
236       -Create New Folder > Folder name : resources > Finish > OK
237       -SpringLifecycle1/resources(new) 확인
238       -Apply and Close
239
240     6)Bean Configuration XML 작성
241       -SpringLifecycle1/resources > right-click > New > Other > Spring > Spring Bean
          Configuration File
242       -File name : applicationContext.xml > Finish
243
244         <?xml version="1.0" encoding="UTF-8"?>
245         <beans xmlns="http://www.springframework.org/schema/beans"
246           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
247           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd">
248
249           <bean id="student" class="com.example.Student">
250             <constructor-arg value="한지민" />
251             <constructor-arg value="25" />
252           </bean>
253
254         </beans>
255
256       -com.example.MainClass.java
257
258         package com.example;
259         import org.springframework.context.support.GenericXmlApplicationContext;
260
261         public class MainClass {
262           public static void main(String[] args) {
263             GenericXmlApplicationContext context = new GenericXmlApplicationContext();
```

```
264             context.load("classpath:applicationContext.xml");
265             context.refresh();
266
267             Student student = context.getBean("student", Student.class);
268             System.out.println(student);
269             context.close();
270           }
271         }
272
273     7)@PostConstruct, @PreDestroy 이용하기
274       -com.example.Student2.java
275
276         package com.example;
277
278         import javax.annotation.PostConstruct;
279         import javax.annotation.PreDestroy;
280
281         public class Student2 {
282           private String name;
283           private int age;
284
285           public Student2(String name, int age) {
286             this.name = name;
287             this.age = age;
288           }
289
290           @Override
291           public String toString() {
292             return String.format("Student [name=%s, age=%s]", name, age);
293           }
294
295           @PostConstruct  <--bean이 생성단계에서 해야할 일 기술
296           public void initTest(){
297             System.out.println("방금 객체가 생성됐습니다.");
298           }
299
300           @PreDestroy     <--bean이 소멸할 때 해야할 일 기술
301           public void destroyTest(){
302             System.out.println("방금 객체가 소멸됐습니다.");
303           }
304         }
305
306       -resources/applicationContext.xml
307         <?xml version="1.0" encoding="UTF-8"?>
308         <beans xmlns="http://www.springframework.org/schema/beans"
309           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
310           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd">
311
312           <!-- <context:annotation-config/>  첫번째 방법-->
313
314           <!-- 두번째 방법-->
315           <bean
              class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor"
```

```
        />

        <bean id="student2" class="com.example.Student2">
          <constructor-arg value="설운도" />
          <constructor-arg value="50" />
        </bean>

        <!-- 세번째 방법 -->
        <!-- <bean id="student2" class="com.example.Student2" init-method="initTest"
        destroy-method="destoryTest">
          <constructor-arg value="설운도" />
          <constructor-arg value="50" />
        </bean> -->
      </beans>
```

-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();
  }
}
```

5. Spring Bean Scope

1)Spring Container가 생성되고, Spring Bean이 생성될 때, 생성된 Spring Bean은 Scope를 가지고 있다.

2)scope이란 쉽게 생각해서 해당하는 객체가 어디까지 영향을 미치는지 결정하는 것이라고 생각하면 된다.

-singleton : 스프링 컨테이너에 단 한 개의 빈 객체만 존재, 기본값

-prototype : Bean을 사용할 때마다 객체를 생성

-request : Http 요청마다 빈 객체를 생성 WebApplicationContext 에서만 적용 가능

-session : Http Session 마다 빈 객체를 생성한다. WebApplicationContext에서만 적용 가능

6. Lab

1)In Package Explorer > right-click > New > Java Project

-Project Name : SpringScopeDemo

2)src > right-click > New > Package

-Package name : com.example

3)com.example.Student.java

```
package com.example;
```

```
368     import java.util.ArrayList;
369     import org.springframework.beans.factory.DisposableBean;
370     import org.springframework.beans.factory.InitializingBean;
371
372     public class Student{
373       private String name;
374       private int age;
375
376       public Student(String name, int age) {
377         this.name = name;
378         this.age = age;
379       }
380
381       public void setName(String name) {
382         this.name = name;
383       }
384
385       public void setAge(int age) {
386         this.age = age;
387       }
388
389       @Override
390       public String toString() {
391         return String.format("Student [name=%s, age=%s]", name, age);
392       }
393     }
394
395   4)Java Project를 Spring Project로 변환
396     -SpringScopeDemo Project > right-click > Configuration > Convert to Maven Project
397       --Project : /SpringScopeDemo
398       --Group Id : SpringScopeDemo
399       --Artifact Id : SpringScopeDemo
400       --version : 0.0.1-SNAPSHOT
401       --Packaging : jar
402       --Finish
403       --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
404
405     -SpringScopeDemo Project > right-click > Spring > Add Spring Project Nature
406       --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
407
408     -pom.xml 파일에 Spring Context Dependency 추가하기
409       <version>0.0.1-SNAPSHOT</version>
410       <dependencies> <--- dependencies element 추가
411         <dependency> <---여기에 paste
412           <groupId>org.springframework</groupId>
413           <artifactId>spring-context</artifactId>
414           <version>4.3.24.RELEASE</version>
415         </dependency>
416       </dependencies>
417
418     -pom.xml > right-click > Run As > Maven install
419       [INFO] BUILD SUCCESS 확인
420
421   5)SpringScopeDemo/resources folder 생성
```

```
422     -SpringScopeDemo project > right-click > Build Path > Configure Build Path
423     -Source Tab > Add Folder
424     -SpringScopeDemo click
425     -Create New Folder > Folder name : resources > Finish > OK
426     -SpringScopeDemo/resources(new) 확인
427     -Apply and Close
428
429   6)Bean Configuration XML 작성
430     -SpringScopeDemo/resources > right-click > New > Other > Spring > Spring Bean
        Configuration File
431     -File name : applicationContext.xml > Finish
432
433     <?xml version="1.0" encoding="UTF-8"?>
434     <beans xmlns="http://www.springframework.org/schema/beans"
435       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
436       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
437
438       <bean id="student" class="com.example.Student" scope="singleton">
439         <constructor-arg value="한지민" />
440         <constructor-arg value="25" />
441       </bean>
442
443     </beans>
444
445   7)com.example.MainClass.java
446
447     package com.example;
448
449     import org.springframework.context.support.AbstractApplicationContext;
450     import org.springframework.context.support.GenericXmlApplicationContext;
451
452     public class MainClass {
453       public static void main(String[] args) {
454         AbstractApplicationContext context = new
            GenericXmlApplicationContext("classpath:ApplicationContext.xml");
455
456         Student student = context.getBean("student", Student.class);
457         System.out.println(student);
458         System.out.println("--------------------------------");
459
460         Student student1 = context.getBean("student", Student.class);
461         student1.setName("설운도");
462         student1.setAge(55);
463         System.out.println(student1);
464         System.out.println("--------------------------------");
465
466         if(student.equals(student1)) System.out.println("Equals");  //Print Equals
467         else System.out.println("Different");
468         context.close();
469       }
470     }
471
472
```

7. <import>
1)Spring 기반의 Application은 단순한 <bean> 등록 외에도 Transaction관리, 예외처리, 다국어 처리 등 복잡하고 다양한 설정이 필요하다.
2)이런 모든 설정을 하나의 파일로 모두 처리할 수도 있지만, 그렇게 하면 스프링 설정 파일이 너무 길어지고 관리가 어려워진다.
3)결국, 기능별 여러 XML 파일로 나누어 설정하는 것이 더 효율적인데, 이렇게 분리하여 작성한 설정 파일들을 하나로 통합할 때 <import>를 사용.

```
context-datasource.xml
<beans>
  ...
</beans>

context-transaction.xml
<beans>
  ...
</beans>
```

4)이럴 경우 2개의 XML 설정파일을 하나로 통합하자.

```
<beans>
  <import resource="context-datasource.xml" />
  <import resource="context-transaction.xml" />
</beans>
```

8. <bean>의 기타 속성들
1)init-method 속성
-Servlet Container는 web.xml 파일에 등록된 Servlet 클래스의 객체를 생성할 때 기본생성자만 인식한다.
-따라서 생성자로 Servlet 객체의 멤버변수를 초기화할 수 없다.
-그래서 Servlet에서는 init()를 재정의하여 멤버변수를 초기화할 수 있다.
-Spring Container 역시 Spring 설정 파일에 등록된 Class를 객체 생성할 때 기본 생성자를 호출한다.
-따라서 객체를 생성할 때 멤버변수를 초기화하는 작업이 필요하다면, Servlet의 init() 같은 메소드가 필요하다.
-이를 위해 Spring 에서는 <bean>에 init-method 속성을 지원한다.

```
<bean id="hello" class="com.example.Hello" init-method="initMethod" />
```

Hello.java

```
package com.example;

public class Hello {
  public void initMethod(){
    System.out.println("초기화 작업");
  }
}
```

-Spring Container는 <bean>에 등록된 Class 객체를 생성한 후에 init-method 속성으로 지정된 initMethod() 메소드를 호출한다.
-이 메소드는 멤버변수에 대한 초기화를 처리한다.

2)destroy-method 속성
-init-method와 마찬가지로 <bean>에서 destroy-method 속성을 이용하여 Spring Container가 객체를 삭제하기 전에 호출될 임의의 메소드를 지정할 수 있다.

```
<bean id="hello" class="com.example.Hello" destroy-method="destroyMethod" />
```

Hello.java

```
package com.example;

public class Hello {
  public void initMethod(){
    System.out.println("초기화 작업");
  }
  public vodi destroyMethod(){
    System.out.println("객체 삭제 처리 작업");
  }
}
```

-Container가 종료되기 직전에 Container는 자신이 관리하는 모든 객체를 삭제하는데, 이때 destroy-method 속성으로 지정한 destoryMethod() 메소드는 객체가 삭제되기 직전에 호출된다.

3)lazy-init 속성

-ApplicationContext를 이용하여 Container를 구동하면 Container가 구동되는 시점에 Spring 설정 파일에 등록된 <bean> 들을 생성하는 즉시 로딩(pre-loading) 방식으로 동작한다.

-그런데 어떤 <bean>은 자주 사용되지 않으면서 메모리를 많이 차지하여 시스템에 부담을 주는 경우도 있다.

-따라서 Spring 에서는 Container가 구동되는 시점이 아닌 해당 <bean>이 사용되는 시점에 객체를 생성하도록 lazy-init="true"로 설정하면 Spring Container는 해당 <bean>을 미리 생성하지 않고 Client가 요청하는 시점에 생성한다.

-결국, 메모리 관리를 더 효율적으로 할 수 있게 해 준다.