```
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
79      -SpringLifecycle Project > right-click > Spring > Add Spring Project Nature
80
81      -pom.xml 파일에 Spring Context Dependency 추가하기
82        <version>0.0.1-SNAPSHOT</version>
83        <dependencies>
84          <dependency>
85            <groupId>org.springframework</groupId>
86            <artifactId>spring-context</artifactId>
87            <version>4.3.24.RELEASE</version>
88          </dependency>
89        </dependencies>
90
91      -pom.xml > right-click > Run As > Maven install
92        [INFO] BUILD SUCCESS 확인
93
94    5)SpringLifecycle/resources folder 생성
95      -SpringLifecycle project > right-click > Build Path > Configure Build Path
96      -Source Tab > Add Folder
97      -SpringLifecycle click
98      -Create New Folder > Folder name : resources > Finish > OK
99      -SpringLifecycle/resources(new) 확인
100     -Apply and Close
101
102   6)Bean Configuration XML 작성
103     -SpringLifecycle/resources > right-click > New > Other > Spring > Spring Bean Configuration
        File
104     -File name : applicationContext.xml > Finish
105
106     <?xml version="1.0" encoding="UTF-8"?>
```

```
107     <beans xmlns="http://www.springframework.org/schema/beans"
108       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
109       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
110
111       <bean id="student1" class="com.example.Student">
112         <constructor-arg value="한지민" />
113         <constructor-arg value="25" />
114         <constructor-arg>
115           <list>
116             <value>독서</value>
117             <value>영화감상</value>
118             <value>요리</value>
119           </list>
120         </constructor-arg>
121         <property name="height" value="165" />
122         <property name="weight">
123           <value>45</value>
124         </property>
125       </bean>
126     </beans>
127
128   7)com.example.MainClass.java
129     package com.example;
130
131     import org.springframework.context.support.GenericXmlApplicationContext;
132
133     public class MainClass {
134       public static void main(String[] args) {
135         GenericXmlApplicationContext context = new GenericXmlApplicationContext();
136
137         context.load("classpath:applicationContext.xml");
138         context.refresh();
139
140         Student student1 = context.getBean("student1", Student.class);
141         System.out.println(student1);
142
143         context.close();
144       }
145     }
146
147   8)실행
148     -MainClass > right-click > Run As > Java Application
149
150   9)결과
151     Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
152
153
154 3. Spring Bean Lifecycle
155   1)context.refresh() 할 때 bean 초기화 과정에서는 InitializingBean interface의 afterPropertiesSet() 또는
      @PostConstruct annotation 을 호출
156     -ctx.load("classpath:applicationContext.xml"); 에서 Bean이 초기화되고,
157     -ctx.refresh() 할 때 Bean이 생성된다.
158     -즉 Bean이 초기화될 때 afterPropertiesSet()이 호출된다.
```

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)JSR 250(Java platform용 공용 annotation)은 다양한 Java 기술에서 사용되는 표준 annotation을 정의한다.
5)@PostConstructor와 @PreDestory 역시 JSR 250에 속해있다.
-http://jcp.org/en/jsr/detail?id=250 참조

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

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

```
264       import org.springframework.context.support.GenericXmlApplicationContext;
265
266       public class MainClass {
267         public static void main(String[] args) {
268           GenericXmlApplicationContext context = new GenericXmlApplicationContext();
269           context.load("classpath:applicationContext.xml");
270           context.refresh();
271
272           Student student = context.getBean("student", Student.class);
273           System.out.println(student);
274           context.close();
275         }
276       }
277
278   8)실행
279     -MainClass > right-click > Run As > Java Application
280     방금 bean이 생성됐습니다.
281     Student [name=한지민, age=25]
282     ...
283     방금 bean이 소멸됐습니다.
284
285   9)@PostConstruct, @PreDestroy 이용하기
286     -com.example.Student2.java
287
288       package com.example;
289
290       import javax.annotation.PostConstruct;
291       import javax.annotation.PreDestroy;
292
293       public class Student2 {
294         private String name;
295         private int age;
296
297         public Student2(String name, int age) {
298           this.name = name;
299           this.age = age;
300         }
301
302         @Override
303         public String toString() {
304           return String.format("Student [name=%s, age=%s]", name, age);
305         }
306
307         @PostConstruct  <--bean이 생성단계에서 해야할 일 기술
308         public void initTest(){
309           System.out.println("방금 객체가 생성됐습니다.");
310         }
311
312         @PreDestroy     <--bean이 소멸할 때 해야할 일 기술
313         public void destroyTest(){
314           System.out.println("방금 객체가 소멸됐습니다.");
315         }
316       }
317
```

```
318     -resources/applicationContext.xml
319       <?xml version="1.0" encoding="UTF-8"?>
320       <beans xmlns="http://www.springframework.org/schema/beans"
321         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
322         xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
323
324         <!-- 첫번째 방법 -->
325         <!-- <context:annotation-config/>
326         <bean id="student2" class="com.example.Student2">
327           <constructor-arg value="설운도" />
328           <constructor-arg value="50" />
329         </bean> -->
330
331         <!-- 두번째 방법-->
332         <!--<bean
          class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor"
          />
333
334         <bean id="student2" class="com.example.Student2">
335           <constructor-arg value="설운도" />
336           <constructor-arg value="50" />
337         </bean> -->
338
339         <!-- 세번째 방법 -->
340         <!-- <bean id="student2" class="com.example.Student2" init-method="initTest"
          destroy-method="destroyTest">
341           <constructor-arg value="설운도" />
342           <constructor-arg value="50" />
343         </bean> -->
344       </beans>
345
346     -com.example.MainClass.java
347
348       package com.example;
349
350       import org.springframework.context.support.GenericXmlApplicationContext;
351
352       public class MainClass {
353         public static void main(String[] args) {
354           GenericXmlApplicationContext context = new GenericXmlApplicationContext();
355           context.load("classpath:applicationContext.xml");
356           context.refresh();
357
358           Student2 student2 = context.getBean("student2", Student2.class);
359           System.out.println(student2);
360           context.close();
361         }
362       }
363
364   10)실행
365     -MainClass > right-click > Run As > Java Application
366     방금 bean이 생성됐습니다.
367     Student [name=설운도, age=50]
```

```
368     ...
369     방금 bean이 소멸됐습니다.
370
371
372 5. Spring Bean Scope
373   1)Spring Container가 생성되고, Spring Bean이 생성될 때, 생성된 Spring Bean은 Scope를 가지고 있다.
374   2)scope이란 쉽게 생각해서 해당하는 객체가 어디까지 영향을 미치는지 결정하는 것이라고 생각하면 된다.
375     -singleton : 스프링 컨테이너에 단 한 개의 빈 객체만 존재, 기본값
376     -prototype : Bean을 사용할 때마다 객체를 생성
377     -request : Http 요청마다 빈 객체를 생성 WebApplicationContext 에서만 적용 가능
378     -session : Http Session 마다 빈 객체를 생성한다. WebApplicationContext에서만 적용 가능
379
380
381 6. Lab
382   1)In Package Explorer > right-click > New > Java Project
383     -Project Name : SpringScopeDemo
384
385   2)src > right-click > New > Package
386     -Package name : com.example
387
388   3)com.example.Student.java
389
390     package com.example;
391
392     public class Student{
393       private String name;
394       private int age;
395
396       public Student(String name, int age) {
397         this.name = name;
398         this.age = age;
399       }
400
401       public void setName(String name) {
402         this.name = name;
403       }
404
405       public void setAge(int age) {
406         this.age = age;
407       }
408
409       @Override
410       public String toString() {
411         return String.format("Student [name=%s, age=%s]", name, age);
412       }
413     }
414
415   4)Java Project를 Spring Project로 변환
416     -SpringScopeDemo Project > right-click > Configuration > Convert to Maven Project
417       --Project : /SpringScopeDemo
418       --Group Id : SpringScopeDemo
419       --Artifact Id : SpringScopeDemo
420       --version : 0.0.1-SNAPSHOT
421       --Packaging : jar
```

```
422       --Finish
423       --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
424
425     -SpringScopeDemo Project > right-click > Spring > Add Spring Project Nature
426       --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
427
428     -pom.xml 파일에 Spring Context Dependency 추가하기
429       <version>0.0.1-SNAPSHOT</version>
430       <dependencies> <--- dependencies element 추가
431         <dependency> <---여기에 paste
432           <groupId>org.springframework</groupId>
433           <artifactId>spring-context</artifactId>
434           <version>4.3.24.RELEASE</version>
435         </dependency>
436       </dependencies>
437
438     -pom.xml > right-click > Run As > Maven install
439       [INFO] BUILD SUCCESS 확인
440
441   5)SpringScopeDemo/resources folder 생성
442     -SpringScopeDemo project > right-click > Build Path > Configure Build Path
443     -Source Tab > Add Folder
444     -SpringScopeDemo click
445     -Create New Folder > Folder name : resources > Finish > OK
446     -SpringScopeDemo/resources(new) 확인
447     -Apply and Close
448
449   6)Bean Configuration XML 작성
450     -SpringScopeDemo/resources > right-click > New > Other > Spring > Spring Bean
        Configuration File
451     -File name : applicationContext.xml > Finish
452
453     <?xml version="1.0" encoding="UTF-8"?>
454     <beans xmlns="http://www.springframework.org/schema/beans"
455       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
456       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd">
457
458       <bean id="student" class="com.example.Student" scope="singleton">
459         <constructor-arg value="한지민" />
460         <constructor-arg value="25" />
461       </bean>
462
463     </beans>
464
465   7)com.example.MainClass.java
466
467     package com.example;
468
469     import org.springframework.context.support.AbstractApplicationContext;
470     import org.springframework.context.support.GenericXmlApplicationContext;
471
472     public class MainClass {
473       public static void main(String[] args) {
```

```
        AbstractApplicationContext context = new
        GenericXmlApplicationContext("classpath:applicationContext.xml");

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

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

```
524     <bean id="hello" class="com.example.Hello" init-method="initMethod" />
525
526     Hello.java
527
528       package com.example;
529
530       public class Hello {
531         public void initMethod(){
532           System.out.println("초기화 작업");
533         }
534       }
535
536     -Spring Container는 <bean>에 등록된 Class 객체를 생성한 후에 init-method 속성으로 지정된
        initMethod() 메소드를 호출한다.
537     -이 메소드는 멤버변수에 대한 초기화를 처리한다.
538
539   2)destroy-method 속성
540     -init-method와 마찬가지로 <bean>에서 destroy-method 속성을 이용하여 Spring Container가 객체를 삭제
        하기 전에 호출될 임의의 메소드를 지정할 수 있다.
541
542     <bean id="hello" class="com.example.Hello" destroy-method="destroyMethod" />
543     Hello.java
544
545       package com.example;
546
547       public class Hello {
548         public void initMethod(){
549           System.out.println("초기화 작업");
550         }
551         public vodi destroyMethod(){
552           System.out.println("객체 삭제 처리 작업");
553         }
554       }
555
556     -Container가 종료되기 직전에 Container는 자신이 관리하는 모든 객체를 삭제하는데, 이때 destroy-method 속성
        으로 지정한 destoryMethod() 메소드는 객체가 삭제되기 직전에 호출된다.
557
558   3)lazy-init 속성
559     -ApplicationContext를 이용하여 Container를 구동하면 Container가 구동되는 시점에 Spring 설정 파일에 등록
        된 <bean> 들을 생성하는 즉시 로딩(pre-loading) 방식으로 동작한다.
560     -즉, singleton 범위의 bean은 기본적으로 사전(pre) instance화되므로 Spring container의 instance가 생성될
        때 singleton 범위 bean의 instance도 함께 생성된다.
561     -그런데 어떤 <bean>은 자주 사용되지 않으면서 메모리를 많이 차지하여 시스템에 부담을 주는 경우도 있다.
562     -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.
563     -결국, 메모리 관리를 더 효율적으로 할 수 있게 해 준다.
564
565       <bean id="lazyBean" class="example.lazyBean" lazy-init="true" />
```