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   9)결과
153     Student [name=한지민, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
154
155
156 3. Spring Bean Lifecycle
157     1)context.refresh() 할 때 bean 초기화 과정에서는 InitializingBean interface의 afterPropertiesSet() 또는
        @PostConstruct annotation 을 호출
158       -ctx.load("classpath:applicationContext.xml"); 에서 Bean이 초기화되고,
```

-ctx.refresh() 할 때 Bean이 생성된다.
-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. 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

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

```
264         public class MainClass {
265           public static void main(String[] args) {
266             GenericXmlApplicationContext context = new GenericXmlApplicationContext();
267             context.load("classpath:applicationContext.xml");
268             context.refresh();
269
270             Student student = context.getBean("student", Student.class);
271             System.out.println(student);
272             context.close();
273           }
274         }
275
276     8)실행
277       -MainClass > right-click > Run As > Java Application
278       방금 bean이 생성됐습니다.
279       Student [name=한지민, age=25]
280       ...
281       방금 bean이 소멸됐습니다.
282
283     9)@PostConstruct, @PreDestroy 이용하기
284       -com.example.Student2.java
285
286         package com.example;
287
288         import javax.annotation.PostConstruct;
289         import javax.annotation.PreDestroy;
290
291         public class Student2 {
292           private String name;
293           private int age;
294
295           public Student2(String name, int age) {
296             this.name = name;
297             this.age = age;
298           }
299
300           @Override
301           public String toString() {
302             return String.format("Student [name=%s, age=%s]", name, age);
303           }
304
305           @PostConstruct  <--bean이 생성단계에서 해야할 일 기술
306           public void initTest(){
307             System.out.println("방금 객체가 생성됐습니다.");
308           }
309
310           @PreDestroy     <--bean이 소멸할 때 해야할 일 기술
311           public void destroyTest(){
312             System.out.println("방금 객체가 소멸됐습니다.");
313           }
314         }
315
316       -resources/applicationContext.xml
317         <?xml version="1.0" encoding="UTF-8"?>
```

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

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;

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

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

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

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

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

  4)Java Project를 Spring Project로 변환
    -SpringScopeDemo Project > right-click > Configuration > Convert to Maven Project
      --Project : /SpringScopeDemo
      --Group Id : SpringScopeDemo
      --Artifact Id : SpringScopeDemo
      --version : 0.0.1-SNAPSHOT
      --Packaging : jar
      --Finish
      --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.

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

```
473
474         Student student = context.getBean("student", Student.class);
475         System.out.println(student);
476         System.out.println("---------------------------------");
477
478         Student student1 = context.getBean("student", Student.class);
479         student1.setName("설운도");
480         student1.setAge(55);
481         System.out.println(student1);
482         System.out.println("---------------------------------");
483
484         if(student.equals(student1)) System.out.println("Equals");  //Print Equals
485         else System.out.println("Different");
486         context.close();
487       }
488     }
489
490
491 7. <import>
492   1)Spring 기반의 Application은 단순한 <bean> 등록 외에도 Transaction관리, 예외처리, 다국어 처리 등 복잡하고
      다양한 설정이 필요하다.
493   2)이런 모든 설정을 하나의 파일로 모두 처리할 수도 있지만, 그렇게 하면 스프링 설정 파일이 너무 길어지고 관리가 어려워진
      다.
494   3)결국, 기능별 여러 XML 파일로 나누어 설정하는 것이 더 효율적인데, 이렇게 분리하여 작성한 설정 파일들을 하나로 통합
      할 때 <import>를 사용.
495
496     context-datasource.xml
497     <beans>
498       ...
499     </beans>
500
501     context-transaction.xml
502     <beans>
503       ...
504     </beans>
505
506   4)이럴 경우 2개의 XML 설정파일을 하나로 통합하자.
507     <beans>
508       <import resource="context-datasource.xml" />
509       <import resource="context-transaction.xml" />
510     </beans>
511
512
513 8. <bean>의 기타 속성들
514   1)init-method 속성
515     -Servlet Container는 web.xml 파일에 등록된 Servlet 클래스의 객체를 생성할 때 기본생성자만 인식한다.
516     -따라서 생성자로 Servlet 객체의 멤버변수를 초기화할 수 없다.
517     -그래서 Servlet에서는 init()를 재정의하여 멤버변수를 초기화할 수 있다.
518     -Spring Container 역시 Spring 설정 파일에 등록된 Class를 객체 생성할 때 기본 생성자를 호출한다.
519     -따라서 객체를 생성할 때 멤버변수를 초기화하는 작업이 필요하다면, Servlet의 init() 같은 메소드가 필요하다.
520     -이를 위해 Spring 에서는 <bean>에 init-method 속성을 지원한다.
521
522     <bean id="hello" class="com.example.Hello" init-method="initMethod" />
523
```

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.