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)New > Spring Legacy Project > Simple Projects > Simple Spring Maven

Project Name : SpringLifecycle

2)Create Package : src/main/java/info.javaexpert

3)info.javaexpert.Student.java

```
package info.javaexpert;

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;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

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

```
    }

4)src/main/resources/ApplicationContext.xml 생성
  -/src/main/resources > right-click > New > Spring Bean Configuration File
    File name : ApplicationContext.xml

  <?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="student1" class="info.javaexpert.Student">
      <constructor-arg value="한지민" />
      <constructor-arg value="25" />
      <constructor-arg>
        <list>
          <value>독서</value>
          <value>영화감상</value>
          <value>요리</value>
        </list>
      </constructor-arg>
      <property name="height" value="165" />
      <property name="weight">
        <value>45</value>
      </property>
    </bean>
  </beans>

5)info.javaexpert.MainClass.java
  package info.javaexpert;

  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();
    }
  }

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


3. Spring Bean Lifecycle
  1)context.refresh() 할 때 bean 초기화 과정에서는 InitializingBean interface의 afterPropertiesSet() 또는 @PostConstruct annotation 을 호출
    -ctx.load("classpath:applicationContext.xml"); 에서 Bean이 생성되는 것이고,
    -ctx.refresh() 할 때 Bean이 생성되는 것이다.
    -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)New > Spring Legacy Project > Simple Projects > Simple Spring Maven
Project Name : SpringLifecycle1
2)Create Package : src/main/java/info.javaexpert
3)InitializingBean, DisposableBean interface 이용하기
-info.javaexpert.Student.java

```
package info.javaexpert;

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("Called destroy()");
   }
   @Override
   public void afterPropertiesSet() throws Exception {
      System.out.println("Called afterPropertiesSet()");
   }
}
```

-src/main/resources/ApplicationContext.xml

```
<?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="info.javaexpert.Student">
      <constructor-arg value="한지민" />
      <constructor-arg value="25" />
   </bean>

</beans>
```

-info.javaexpert.MainClass.java

```
package info.javaexpert;
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();
   }
}
```

7)@PostConstruct, @PreDestroy 이용하기

-info.javaexpert.Student2.java

```
package info.javaexpert;

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

public class Student2 {
   private String name;
   private int age;

   public Student2(String name, int age) {
      this.name = name;
      this.age = age;
   }

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

   @PostConstruct  <--bean이 생성단계에서 해야할 일 기술
   public void initMethod(){
      System.out.println("Called initMethod()");
   }

   @PreDestroy     <--bean이 소멸할 때 해야할 일 기술
   public void destroyMethod(){
      System.out.println("Called destoryMethod()");
   }
}
```

-src/main/resources/ApplicationContext.xml

```
<?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
   class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor" />
   <--필수

   <bean id="student2" class="info.javaexpert.Student2">
      <constructor-arg value="설운도" />
      <constructor-arg value="50" />
   </bean>
</beans>
```

-info.javaexpert.MainClass.java

```
package info.javaexpert;

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.
      -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)New > Spring Legacy Project > Simple Projects > Simple Spring Maven
      Project Name : SpringScopeDemo

   2)Create Package : src/main/java/info.javaexpert

   3)info.javaexpert.Student.java

      package info.javaexpert;

      import java.util.ArrayList;
      import org.springframework.beans.factory.DisposableBean;
      import org.springframework.beans.factory.InitializingBean;

      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)src/main/resources/ApplicationContext.xml
      <?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="info.javaexpert.Student" scope="singleton">
            <constructor-arg value="한지민" />
            <constructor-arg value="25" />
         </bean>

      </beans>

   6)info.javaexpert.MainClass.java

      package info.javaexpert;
```

```
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
   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();
   }
}
```