

Spring framework  
Define bean in xml  
Load beans to container  
getBean()

in xml      how to define bean by injecting value type dependencies thru setters

```
17      <!-- define mytime bean and inject values thru setters -->
18      <bean class="org.example.model.MyTime" id="time2">
19          <property name="hours" value="10"/>
20          <property name="minutes" value="25"/>
21          <property name="seconds" value="30"/>
22      </bean>
```

how to define bean by injecting value type dependencies thru constructor

```
24      <!-- define bean by injecting dependencies thru constructor -->
25      <bean class="org.example.model.MyTime" id="time3">
26          <constructor-arg value="11"/> <!-- 11 passed 1st arg for constructor -->
27          <constructor-arg value="22"/> <!-- 22 passed 2nd arg for constructor -->
28          <constructor-arg value="33"/> <!-- 33 passed 3rd arg for constructor -->
29      </bean>
```

how to define bean by injecting ref type dependencies

```
9      <!--
10      Book b = new Book();
11      b.setBookName("Network protocols");
12      b.setBookPrice(2314);
13      b.setAuthor(new Author("Wen", "US"));
14      -->
15      <!-- define book bean, inject dependencies thru setters -->
16      <bean class="org.example.model.Book" id="book1">
17          <property name="bookName" value="Network protocols"/>
18          <property name="bookPrice" value="2314"/>
19          <property name="author">
20              <!-- define author bean -->
21              <bean class="org.example.model.Author">
22                  <constructor-arg value="Wen"/>
23                  <constructor-arg value="US"/>
24              </bean>
25          </property>
26      </bean>
```

```
28      <!--
29      Author a1 = new Author("xyz", "delhi");
30      Book b = new Book();
31      b.setBookName("abcd");
32      b.setBookPrice(345);
33      b.setAuthor(a1);
34      -->
35      <bean class="org.example.model.Author" id="a1">
36          <constructor-arg value="xyz"/>
37          <constructor-arg value="delhi"/>
38      </bean>
39      <bean class="org.example.model.Book" id="book2">
40          <property name="bookName" value="abcd"/>
41          <property name="bookPrice" value="345"/>
42          <property name="author" ref="a1"/>
43      </bean>
```

How to define bean if dependency value is collection type

```
class Student{  
    String name  
    int marks[ ]  
}
```

How to define bean as singleton / prototype

How to implement autowiring

```
class Face  
{  
    Eye e1;  
}
```

```
Face f = new Face();  
//f.setEye( e);
```

```
class Eye {  
}
```

```
Eye e = new Eye(.....);
```

### Wiring

Attaching dependency object to dependent object

### Manual wiring (explicit)

```
Face f = new Face();  
f.setEy( e);
```

```
Eye e = new Eye();
```



### Auto wiring (byType)

Need to inform spring container about dependencies to be automatically wired to dependents by type matching

Informed container face object needs Eye type object as dependency

```
Face f = new Face();
```

```
Eye e = new Eye();
```



Qualifiers are used to identify unique dependency objects in autowire scenario

## Bean scope

Life time of bean in application

range of bean where it is accessible in application

## Singleton\*

Single copy of object existing in application

Single copy of object is referred by every instance request

```
10      <!-- define mytime bean and inject values thru setters -->
11      <bean class="org.example.model.MyTime" id="time1" scope="singleton">
12          <property name="hours" value="30"/> <!-- calls setHours(30) -->
13          <property name="minutes" value="45"/> <!-- calls setMinutes(45) -->
14          <property name="seconds" value="56"/> <!-- calls setSecond(56) -->
15      </bean>
```

```
9      ApplicationContext container = new ClassPathXmlApplicationContext( configLocation: "beans3.xml");
```

```
11      MyTime a = container.getBean( s: "time1", MyTime.class);
12      System.out.println(a.hashCode());
13      System.out.println(a);
```

```
15      MyTime b = container.getBean( s: "time1", MyTime.class);
16      System.out.println(b.hashCode());
17      System.out.println(b);
```

```
19      MyTime c = container.getBean( s: "time1", MyTime.class);
20      System.out.println(c.hashCode());
21      System.out.println(c);
```

```
MyTimeBeanApp2 x
"C:\Program Files\Java\jdk-14.0.1\bin\ja
MyTime:Default constructor executed...
70807318
30 : 45 : 56
70807318
30 : 45 : 56
70807318
30 : 45 : 56
```

## Prototype

Creates new bean for every instance request

```
10      <!-- define mytime bean and inject values thru setters -->
11      <bean class="org.example.model.MyTime" id="time1" scope="prototype">
12          <property name="hours" value="30"/> <!-- calls setHours(30) -->
13          <property name="minutes" value="45"/> <!-- calls setMinutes(45) -->
14          <property name="seconds" value="56"/> <!-- calls setSecond(56) -->
15      </bean>
```

9      ApplicationContext container = new ClassPathXmlApplicationContext( configLocation: "beans3.xml"); 10 11      MyTime a = container.getBean( s: "time1", MyTime.class); 12      System.out.println(a.hashCode()); 13      System.out.println(a); 14 15      MyTime b = container.getBean( s: "time1", MyTime.class); 16      System.out.println(b.hashCode()); 17      System.out.println(b); 18 19      MyTime c = container.getBean( s: "time1", MyTime.class); 20      System.out.println(c.hashCode()); 21      System.out.println(c);	MyTime:Default constructor executed.... 1128096251 30 : 45 : 56 MyTime:Default constructor executed.... 2136288211 30 : 45 : 56 MyTime:Default constructor executed.... 1008925772 30 : 45 : 56
---	---

## Explicit / manual wiring

```
17 <bean class="org.example.model.Book" id="book1" >
18     <property name="bookName" value="abcd"/>
19     <property name="bookPrice" value="345"/>
20     <property name="author" ref="a1"/>
21 </bean>
22
23 <bean class="org.example.model.Author" id="a1">
24     <constructor-arg value="xyz"/>
25     <constructor-arg value="delhi"/>
26 </bean>
```

ApplicationContext container = new ClassPathXmlApplicationContext( configLocation: "beans4.xml");

```
10 Book b = container.getBean( s: "book1", Book.class);
11 System.out.println(b.getBookName());      abcd
12 System.out.println(b.getBookPrice());      345
13 System.out.println(b.getAuthor());         xyz, delhi
```

## what if manually wiring not done

```
17 <bean class="org.example.model.Book" id="book1" >
18     <property name="bookName" value="abcd"/>
19     <property name="bookPrice" value="345"/>
20     <!-- <property name="author" ref="a1"/> -->
21 </bean>
22
23 <bean class="org.example.model.Author" id="a1">
24     <constructor-arg value="xyz"/>
25     <constructor-arg value="delhi"/>
26 </bean>
```


abcd  
345  
null

## Auto wiring / implicit wiring

Dependency object wired/injected/attached to dependent object automatically by container

```
17 <bean class="org.example.model.Book" id="book1" autowire="byType" >
18     <property name="bookName" value="abcd"/>
19     <property name="bookPrice" value="345"/>
20 </bean>
21
22 <bean class="org.example.model.Author" id="a1">
23     <constructor-arg value="xyz"/>
24     <constructor-arg value="delhi"/>
25 </bean>
```

abcd  
345  
xyz, delhi

A blue arrow points from the `id="a1"` attribute of the `Author` bean to the `autowire="byType"` attribute of the `Book` bean, indicating that the container automatically injects the `Author` bean into the `Book` bean based on their types.