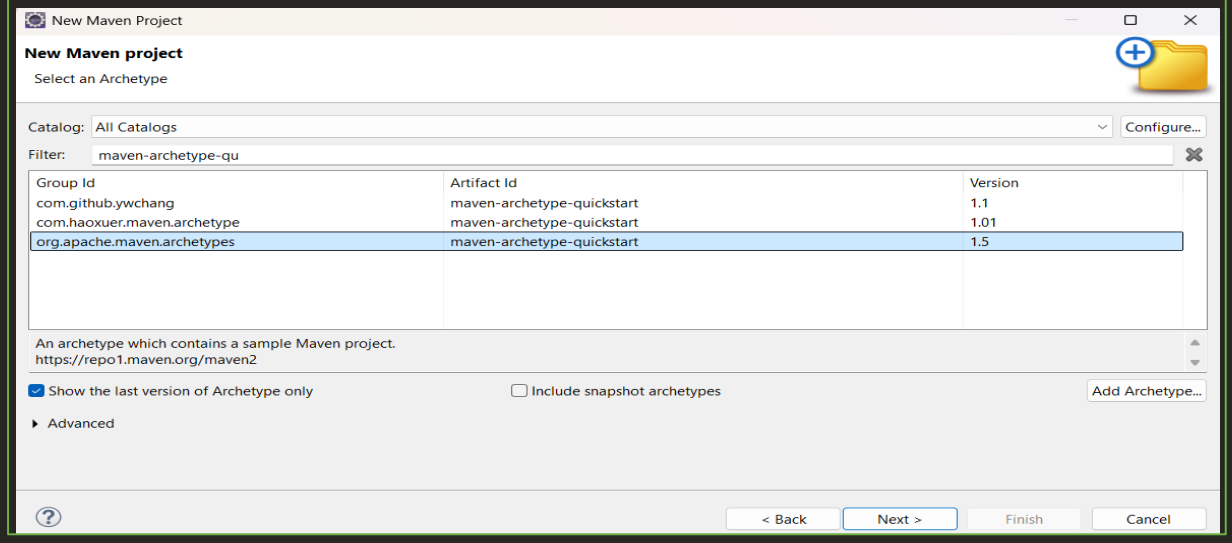
Dependency Injection in Spring

Practicing Spring Core with Beans and ApplicationContext is a solid way to understand how Dependency Injection and Inversion of Control work in Spring.

1- Create maven project -> maven-archetype-quickstart
2- Adding dependencies =>spring core, spring context
3- Creating beans-java pojo
4- Creating configuration file=>config.xml
5- Setter Injection
6- Main class which can pull the object and use

1. Create maven project:



2. Adding Dependency To pom.xml:

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>7.0.0-M7</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>7.0.0-M7</version>
        </dependency>
```

3. Create One package inside the `src/main/java` and inside that package create one Class called `Student`.

```
  package com.bytexl.DependencyInjection;

  public class Student {
       private int rollNumber;
       private String name;
       private float height;
       private int age;

       public int getRollNumber() {
              return rollNumber;
       }
       public void setRollNumber(int rollNumber) {
              this.rollNumber = rollNumber;
       }
       public String getName() {
              return name;
       }
       public void setName(String name) {
              this.name = name;
       }
       public float getHeight() {
              return height;
       }
       public void setHeight(float height) {
              this.height = height;
       }
       public int getAge() {
              return age;
       }
       public void setAge(int age) {
              this.age = age;
       }

       public Student(int rollNumber, String name, float height, int
age) {
              super();
              this.rollNumber = rollNumber;
              this.name = name;
              this.height = height;
              this.age = age;
       }

       public Student() {
              super();
              // TODO Auto-generated constructor stub
       }
```

```
	@Override
	public String toString() {
		return "Student [rollNumber=" + rollNumber + ", name="
+ name + ", height=" + height + ", age=" + age + "]";
	}
}
```

4. Inside the same package i.e. com.bytexl.DependencyInjection create one config.xml file.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <beans xmlns="http://www.springframework.org/schema/beans"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"
         xmlns:p="http://www.springframework.org/schema/p"
         xsi:schemaLocation="
          http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-
beans.xsd
          http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-
context.xsd">

         <bean class="com.bytexl.DependencyInjection.Student"
name="student1">
              <property name="rollNumber">
                    <value>1</value>
              </property>
              <property name="name">
                    <value>Aman Singh</value>
              </property>
              <property name="height">
                    <value>5.10</value>
              </property>
              <property name="age">
                    <value>24</value>
              </property>
         </bean>

         <bean class="com.bytexl.DependencyInjection.Student"
name="student2">
              <property name="rollNumber" value="2" />
              <property name="name" value="Rahul" />
              <property name="height" value="5.5" />
              <property name="age" value="23" />
         </bean>
 </beans>
```

5. Now Create a main class from where all the execution begins: let say `App`.

```
package com.bytexl.DependencyInjection;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        ApplicationContext context = new
ClassPathXmlApplicationContext("com/bytexl/DependencyInjection/confi
g.xml");
        Student student1 = (Student) context.getBean("student1");
        Student student2 = (Student) context.getBean("student2");
        System.out.println(student1);
        System.out.println(student2);
    }
}
```

Additional: These are the different ways to inject values to Class using Bean and property tags.

```
<bean id="student1" class="com.learningSpring.Students">
  <property name="id" value="1" />
  <property name="name" value="Aman" />
  <property name="height" value="5.8" />
</bean>

<bean id="student2" class="com.learningSpring.Students">
  <property name="id" value="2" />
  <property name="name" value="Rishabh" />
  <property name="height" value="5.10" />
</bean>

<bean id="student3" class="com.learningSpring.Students" p:id="3" p:name="Karan" p:height="5.10" />
```

Additionally, by using the context, we can manage the lifecycle methods of the class and retrieve values defined in the <bean> tags as needed. This code belongs to the main Java file, which serves as the entry point for execution.

```java
package com.learningSpring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
        Students student1 = (Students) context.getBean("student1");
        Students student2 = (Students) context.getBean("student2");
        Students student3 = (Students) context.getBean("student3");
        System.out.println(student1);
        System.out.println(student2);
        System.out.println(student3);
    }
}
```

Injecting Values of Primitive Type

```xml
<property name="">
    <value>value</value>
</property>

<property name="" value="" />

<bean p:property:Value />
```

Injecting Values of Collection Type

List

```
<bean>
     <property name="">

      <list>
          <value>10</value>
          <value>1023</value>
          <value>12350</value>
          <value>1052</value>
          <null/>
      </list>

     </property>
<bean>
```

Set

```
<bean>
     <property name="">

      <set>
          <value>10</value>
          <value>1023</value>
          <value>12350</value>
          <value>1052</value>
      </set>

     </property>
</bean>
```

Map

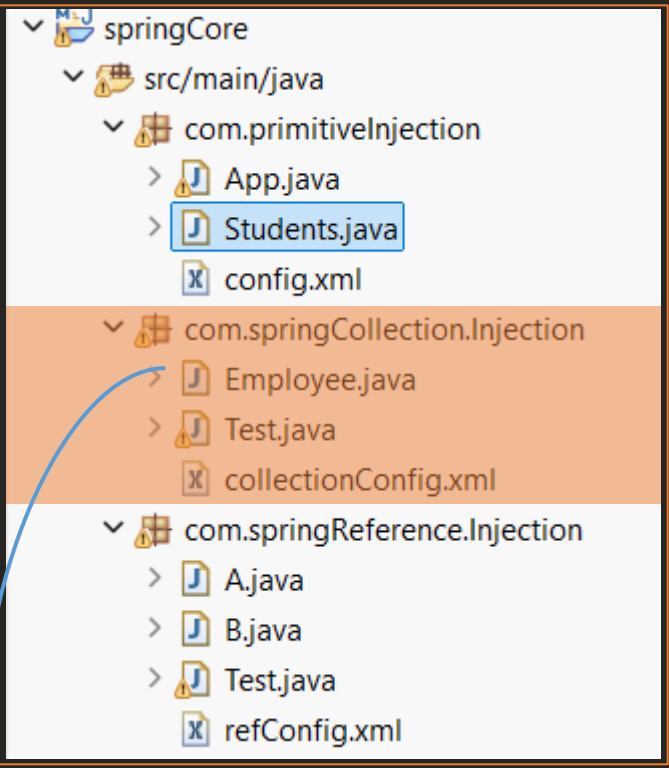
```
<bean>
    <property name="">

     <map>
        <entry key="java" value="2month" />
        <entry key="python" value="1month" />

     </map>

    </property>

</bean>
```

Properties

```
<property name="emp_Prop">
	<props>
		<prop key="name">Rishabh</prop>
		<prop key="age">26</prop>
		<prop key="height">2.7</prop>
	</props>
</property>
```

Practical: Injecting Values Of Collection Type:

Make sure to follow this folder structure when practicing Dependency Injection, as it is essential for running the main Java class without any errors.



Step: 1

```
package com.springCollection.Injection;

import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;

public class Employee {
	private String emp_Code;
	private List<String> subject_Code;
	private Set<String> class_Code;
	private Map<String, String> sub_Duration;
	private Properties emp_Prop;
```

```
	public String getEmp_Code() {
		return emp_Code;
	}
	public void setEmp_Code(String emp_Code) {
		this.emp_Code = emp_Code;
	}
	public List<String> getSubject_Code() {
		return subject_Code;
	}
	public void setSubject_Code(List<String> subject_Code) {
		this.subject_Code = subject_Code;
	}
	public Set<String> getClass_Code() {
		return class_Code;
	}
	public void setClass_Code(Set<String> class_Code) {
		this.class_Code = class_Code;
	}
	public Map<String, String> getSub_Duration() {
		return sub_Duration;
	}
	public void setSub_Duration(Map<String, String> sub_Duration)
{
		this.sub_Duration = sub_Duration;
	}
	public Properties getEmp_Prop() {
		return emp_Prop;
	}
	public void setEmp_Prop(Properties emp_Prop) {
		this.emp_Prop = emp_Prop;
	}

	public Employee(String emp_Code, List<String> subject_Code,
Set<String> class_Code,
			Map<String, String> sub_Duration, Properties
emp_Prop) {
		super();
		this.emp_Code = emp_Code;
		this.subject_Code = subject_Code;
		this.class_Code = class_Code;
		this.sub_Duration = sub_Duration;
		this.emp_Prop = emp_Prop;
	}
	public Employee() {
		super();
	}
}
```

Step: 2

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context.xsd">

       <bean id="emp1"
class="com.springCollection.Injection.Employee">
              <property name="emp_Code" value="1001" />
              <property name="subject_Code">
                     <list>
                            <value>BTCSE101</value>
                            <value>BTCSE102</value>
                            <value>BTCSE103</value>
                     </list>
              </property>

              <property name="class_Code">
                     <set>
                            <value>A401</value>
                            <value>B303</value>
                            <value>C407</value>
                     </set>
              </property>

              <property name="sub_Duration">
                     <map>
                            <entry key="JFS" value="25Days" />
                            <entry key="DBMS" value="15Days" />
                            <entry key="NOS" value="20Days" />
                            <entry key="DSA" value="23Days" />
                     </map>
              </property>

              <property name="emp_Prop">
                     <props>
                            <prop key="name">Rishabh</prop>
                            <prop key="age">26</prop>
                            <prop key="height">2.7</prop>
                     </props>
              </property>
       </bean>
</beans>
```

Step: 3

```
package com.springCollection.Injection;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {
	public static void main(String[] args) {
		ApplicationContext context = new
ClassPathXmlApplicationContext("com/springCollection/Injection/colle
ctionConfig.xml");
		Employee emp1 = (Employee) context.getBean("emp1");

		System.out.println(emp1.getEmp_Code());
		System.out.println(emp1.getSubject_Code());
		System.out.println(emp1.getClass_Code());
		System.out.println(emp1.getSub_Duration());
		System.out.println(emp1.getEmp_Prop());
	}
}
```

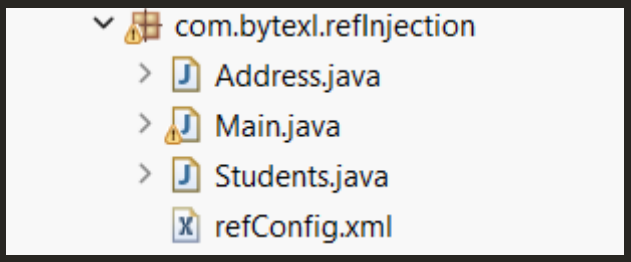
Step: 4

Additionally, by using the context, we can manage the lifecycle methods of the class and retrieve values defined in the <bean> tags as needed. This code belongs to the Test Java file, which serves as the entry point for execution. Run this file to obtain the desired output.

Injecting Values of Reference Type

```
public class A {
    private int x;
    private B obj;
}
```

```
public class B {
    private int y;
}
```



Step: 1

```
package com.bytexl.refInjection;

public class Address {
	private String state;
	private String city;
	private String street;
	private int pinCode;

	public String getState() {
		return state;
	}
	public void setState(String state) {
		this.state = state;
	}
	public String getCity() {
		return city;
	}
	public void setCity(String city) {
		this.city = city;
	}
	public String getStreet() {
		return street;
	}
	public void setStreet(String street) {
		this.street = street;
	}
	public int getPinCode() {
		return pinCode;
	}
	public void setPinCode(int pinCode) {
		this.pinCode = pinCode;
	}

	public Address(String state, String city, String street, int
pinCode) {
		super();
		this.state = state;
		this.city = city;
		this.street = street;
		this.pinCode = pinCode;
	}
```

```
	public Address() {
		super();
		// TODO Auto-generated constructor stub
	}
	@Override
	public String toString() {
		return "Address [state=" + state + ", city=" + city +
", street=" + street + ", pinCode=" + pinCode + "]";
	}
}
```

Step: 2

```
package com.bytexl.refInjection;

public class Students {
	private int rollNo;
	private String name;
	private Address obj;

	public int getRollNo() {
		return rollNo;
	}
	public void setRollNo(int rollNo) {
		this.rollNo = rollNo;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Address getObj() {
		return obj;
	}
	public void setObj(Address obj) {
		this.obj = obj;
	}

	public Students(int rollNo, String name, Address obj) {
		super();
		this.rollNo = rollNo;
		this.name = name;
		this.obj = obj;
	}
	public Students() {
		super();
		// TODO Auto-generated constructor stub
	}
```

```
	@Override
	public String toString() {
		return "Students [rollNo=" + rollNo + ", name=" + name
+ ", obj=" + obj + "]";
	}
}
```

Step: 3

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:p="http://www.springframework.org/schema/p"
	xsi:schemaLocation="
	 http://www.springframework.org/schema/beans
	 http://www.springframework.org/schema/beans/spring-beans.xsd
	 http://www.springframework.org/schema/context
	 http://www.springframework.org/schema/context/spring-
context.xsd">

		<bean name="refAddress"
class="com.bytexl.refInjection.Address">
			<property name="state" value="UP" />
			<property name="city" value="Kanpur" />
			<property name="street" value="Mg Road" />
			<property name="pinCode" value="123456" />
		</bean>

		<bean name="refStudents"
class="com.bytexl.refInjection.Students">
			<property name="rollNo" value="01" />
			<property name="name" value="Aman Singh" />
			<property name="obj">
				<ref bean="refAddress" />
			</property>
		</bean>
</beans>
```

Step: 4

```
package com.bytexl.refInjection;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class Main {
       public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("com/bytexl/refInjection/refconfig.xml");
        Students student1 = (Students)
context.getBean("refStudents");
        System.out.println(student1);
       }
}
```

💬 Let's Connect

I'd love to hear from you, collaborate, or simply connect over tech and ideas. Feel free to explore more or reach out through the links below.

🌐 Portfolio 🐱 GitHub 🔗 Linkedin 👉 Instagram

Keep Learning. Keep Building.

Thanks again for reading — wishing you continued success and growth in your development journey!

— **Rishabh Singh**