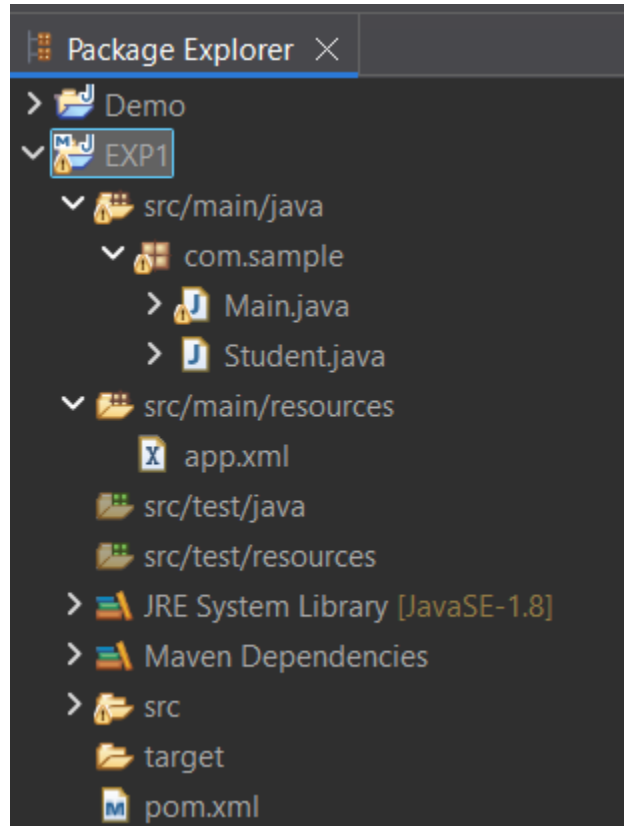


1. Basic Spring Application with Constructor Injection.

Aim : Implementing a Basic Spring Application with Constructor Injection.

Description: Constructor injection in Spring is a way to inject dependencies through a class constructor. It ensures that all required dependencies are provided at the time of object creation. In XML-based configuration, this is done using the <constructor-arg> tag inside the <bean> definition. Constructor injection promotes immutability and is ideal for mandatory dependencies. It is one of the core concepts of Spring's Inversion of Control (IoC) container.

Package Explorer:



Pojo class(Student.java):

```
package com.sample;

public class Student {

    int sid;

    String Sname;

    public Student(int sid, String sname) {

        super();

        this.sid = sid;

        Sname = sname;    }

    public Student(int sid) {

        super();

        this.sid = sid;  }

    @Override

    public String toString() {

        return "Student [sid=" + sid + ", Sname=" + Sname + "];    }    }
```

Main Class(Main.java):

```
package com.sample;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("app.xml");

        Student s1 = context.getBean("x",Student.class);

        System.out.println(s1); } }
```

XML File(app.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
```

```
    https://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
    <bean id="x" class="com.sample.Student">
```

```
        <constructor-arg index = "0" value="142"></constructor-arg>
```

```
        <constructor-arg index = "1" value="Charan"></constructor-arg>
```

```
    </bean>
```

```
</beans>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>ABC</groupId>

    <artifactId>EXP1</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-core</artifactId>

            <version>7.0.0-M7</version>

        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

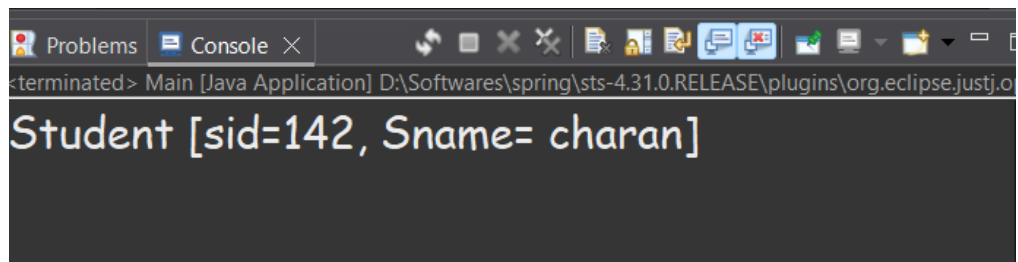
            <version>7.0.0-M7</version>

        </dependency>

    </dependencies>

</project>
```

Output:



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems' and 'Console'. The console text shows the application has terminated and displays the output: 'Student [sid=142, Sname= charan]'.

```
<terminated> Main [Java Application] D:\Softwares\spring\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.o  
Student [sid=142, Sname= charan]
```

2.Spring Application with Setter Injection.

Aim : Implementation of Spring Application with Setter Injection.

Description: Setter injection in Spring is used to inject dependencies through JavaBean-style setter methods. In XML configuration, the <property> tag is used within the <bean> definition to set values or references. This method allows optional dependencies to be set after object creation. Setter injection provides flexibility and is easy to modify without changing the constructor. It is a fundamental feature of Spring's Dependency Injection.

PojoClass(Student.java):

```
package com.sample;

public class Student {

    int Sid;

    String Sname;

    public int getSid() {

        return Sid;    }

    public void setSid(int sid) {

        Sid = sid;    }

    public String getSname() {

        return Sname;    }

    public void setSname(String sname) {

        Sname = sname;    }

    @Override

    public String toString() {

        return "Student [Sid=" + Sid + ", Sname=" + Sname + "];    }

}
```

Main Class(Main.java):

```
package com.sample;

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

public class Main {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("app.xml");

        Student s = context.getBean("x",Student.class);

        System.out.println(s);

    }

}
```

app.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

        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="x" class = "com.sample.Student">

        <property name = "Sid" value = "142"></property>

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

    </bean>

</beans>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>sdf</groupId>

    <artifactId>EXP2</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-core</artifactId>

            <version>7.0.0-M7</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>

            <groupId>org.springframework</groupId>

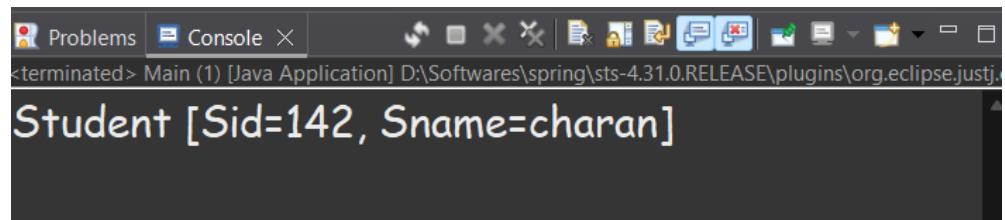
            <artifactId>spring-context</artifactId>

            <version>7.0.0-M7</version>
        </dependency>

    </dependencies>

</project>
```


Output:



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems' and 'Console'. The console text area displays the output: 'Student [Sid=142, Sname=charan]'. Above the text area, the status bar shows '<terminated> Main (1) [Java Application] D:\Softwares\spring\sts-4.31.0.RELEASE\plugins\org.eclipse.justj'.

```
<terminated> Main (1) [Java Application] D:\Softwares\spring\sts-4.31.0.RELEASE\plugins\org.eclipse.justj,  
Student [Sid=142, Sname=charan]
```

3. Spring Auto Scanning Example.

Aim: Implementation of Spring Auto Scanning Example.

Description: Spring Auto Scanning allows the framework to automatically detect and register beans using annotations like `@Component`, `@Service`, `@Repository`, and `@Controller`. It eliminates the need for explicit bean definitions in XML. The `context:component-scan` tag in the configuration file enables this feature by specifying the base package to scan. This approach simplifies configuration and supports cleaner, annotation-driven development.

Pojoclass1(Student1):

```
package com.sample;

import org.springframework.stereotype.Component;

@Component

public class Student1 {

    @Override

    public String toString() {

        return "Student Name: Charan";

    }

}
```

Pojoclass2(Student2):

```
package com.sample;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Component;
```

```
@Component
```

```
public class Student2 {
```

```
    @Autowired
```

```
    Student1 f;
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Student Name: Praveen "+f;
```

```
    }
```

```
}
```

Mainclass(Main.java):

```
package com.sample;

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

public class Main {

    public static void main(String[] args) {

        ApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");

        Student2 s = (Student2)context.getBean("student2");

        System.out.println(s);

    }
}
```

config.xml:

```
<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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">

    <context:component-scan base-package = "com.sample"></context:component-scan>

</beans>
```

Pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>sa</groupId>

    <artifactId>EXP3</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-core</artifactId>

        <version>7.0.0-M7</version>
    </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>

        <groupId>org.springframework</groupId>

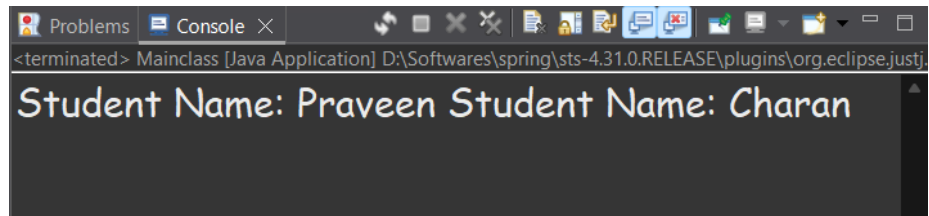
        <artifactId>spring-context</artifactId>

        <version>7.0.0-M7</version>
    </dependency>

    </dependencies>

</project>
```

Output:



The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems' and 'Console'. The console text area displays the output of a Java application, which has terminated. The output text is 'Student Name: Praveen Student Name: Charan'.

```
<terminated> Mainclass [Java Application] D:\Softwares\spring\sts-4.31.0.RELEASE\plugins\org.eclipse.justj  
Student Name: Praveen Student Name: Charan
```