

Setter Dependency Injection

Setter injection is a dependency injection in which the spring framework injects the dependency object using the setter method. The call first goes to no argument constructor and then to the setter method. It does not create any new bean instance. Let's see an example to inject dependency by the setter method.

Address.java

```
package com.example.demo;

public class Address {
    private String city;
    private String state;
    private String country;
    public String getCity()
    {
        return city;
    }
    public void setCity(String city)
    {
        this.city = city;
    }
    public String getState()
    {
        return state;
    }
    public void setState(String state)
    {
        this.state = state;
    }
    public String getCountry()
    {
```

```
        return country;
    }
    public void setCountry(String country)
    {
        this.country = country;
    }
    public String toString()
    {
        return city+" "+state+" "+country;
    }
}
```

Employee.java

```
package com.example.demo;
public class Employee {
    private int id;
    private String name;
    private Address address;
    public int getId()
    {
        return id;
    }
    public void setId(int id)
    {
        this.id = id;
    }
    public String getName()
    {
        return name;
    }
}
```

```
    }  
    public void setName(String name)  
    {  
        this.name = name;  
    }  
    public Address getAddress()  
    {  
        return address;  
    }  
    public void setAddress(Address address)  
    {  
        this.address = address;  
    }  
    public String toString()  
    {  
        return id+" "+name+" "+address;  
    }  
}
```

PdApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication
public class PdApplication
{
    public static void main(String[] args)
    {
        //SpringApplication.run(Pd1Application.class, args);
        ClassPathXmlApplicationContext context=new
ClassPathXmlApplicationContext("emp.xml");
        System.out.println(context.getBean("e1"));
    }
}
```

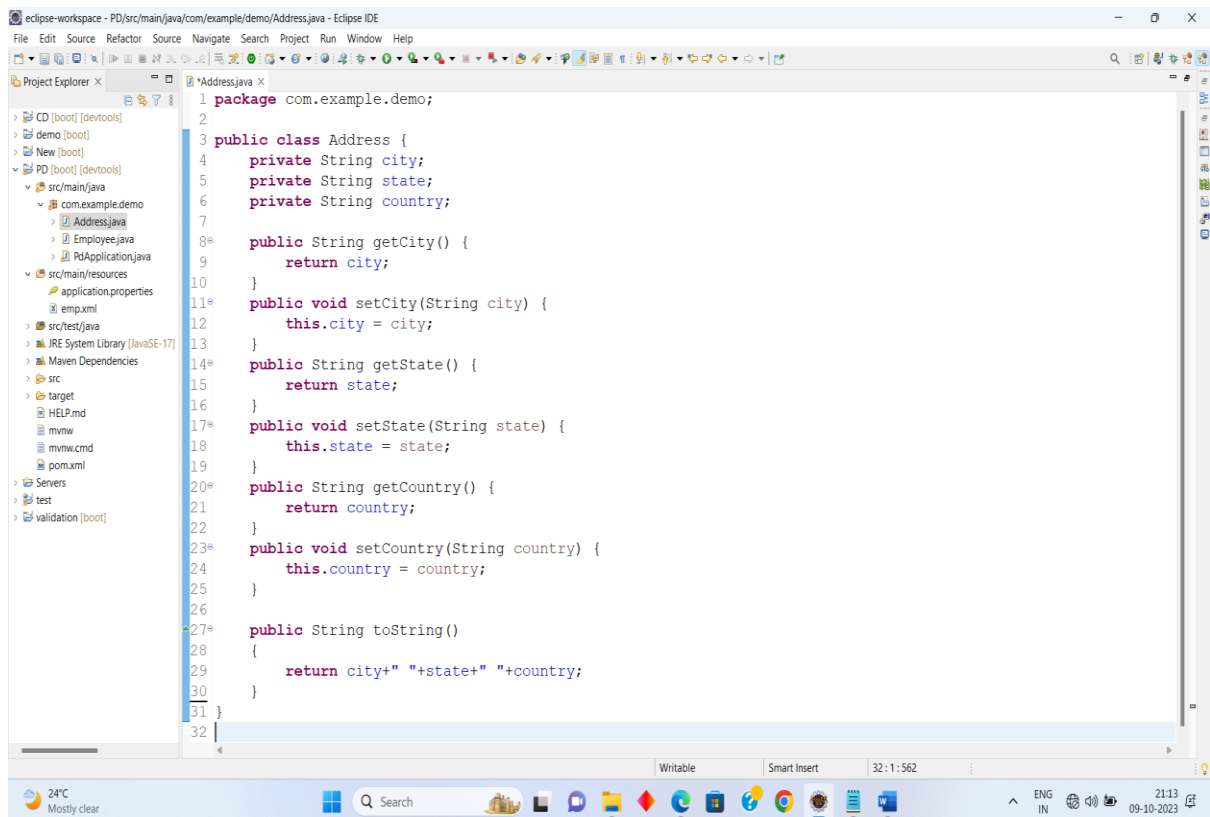
Emp.xml

```
<?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="a1" class="com.example.demo.Address">
    <property name="city" value="Bengaluru"/>
    <property name="state" value="Karnataka"/>
    <property name="country" value="INDIA"/>
  </bean>
  <bean id="e1" class="com.example.demo.Employee">
    <property name="id" value="30"/>
    <property name="name" value="Tom"/>
    <property name="address" ref="a1"/>
  </bean>
</beans>
```

The screenshot displays the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Design, Window, and Help. The Project Explorer on the left shows the project structure for 'eclipse-workspace - CD/src/main/resources/e.xml - Eclipse IDE'. The project 'CD [boot] [devtools]' is expanded, showing sub-projects 'demo [boot]', 'New [boot]', and 'PD [boot] [devtools]'. The 'PD [boot] [devtools]' project is further expanded, showing the 'src/main/java' package with 'com.example.demo' containing 'Address.java', 'Employee.java', and 'PdApplication.java'. The 'src/main/resources' package contains 'application.properties' and 'emp.xml'. Other packages include 'src/test/java', 'JRE System Library [JavaSE-17]', 'Maven Dependencies', 'src', 'target', 'HELP.md', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The 'Servers' tab is also visible.

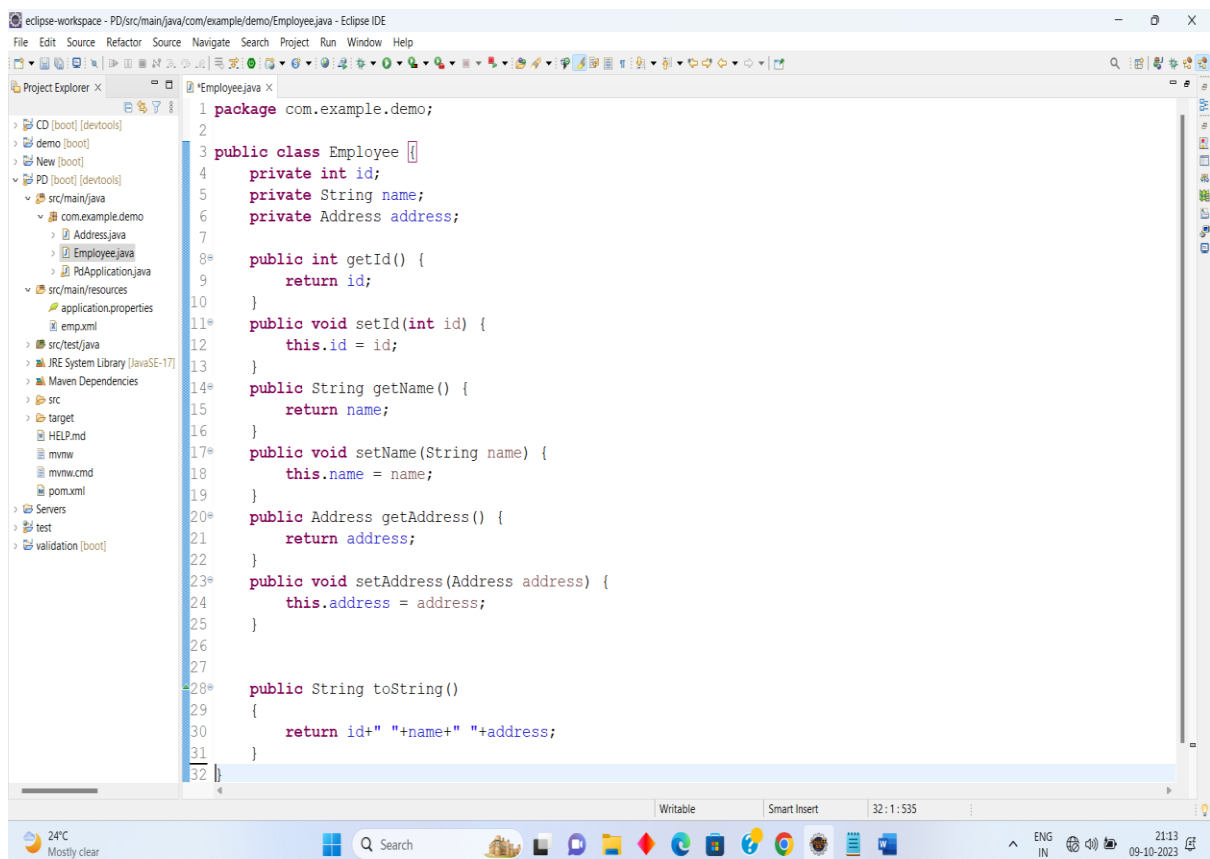
The 'New Spring Starter Project Dependencies' dialog is open, showing the 'Spring Boot Version' as 3.1.4. The 'Frequently Used' section includes 'Spring Boot DevTools' (checked), 'Spring Web' (checked), and 'Spring Web Services' (unchecked). The 'Available' section lists various dependencies, including 'Developer Tools', 'Google Cloud Platform', 'I/O', and 'Messaging'. The 'Selected' section shows 'Spring Boot DevTools' and 'Spring Web' selected. The 'Make Default' and 'Clear Selection' buttons are at the bottom right. The dialog has a 'Back' button and a 'Next >' button. The 'Finish' and 'Cancel' buttons are at the bottom right.

The bottom status bar shows the system tray with a search bar, a weather widget (23°C Rain showers), and a clock (20:10 09-10-2023).



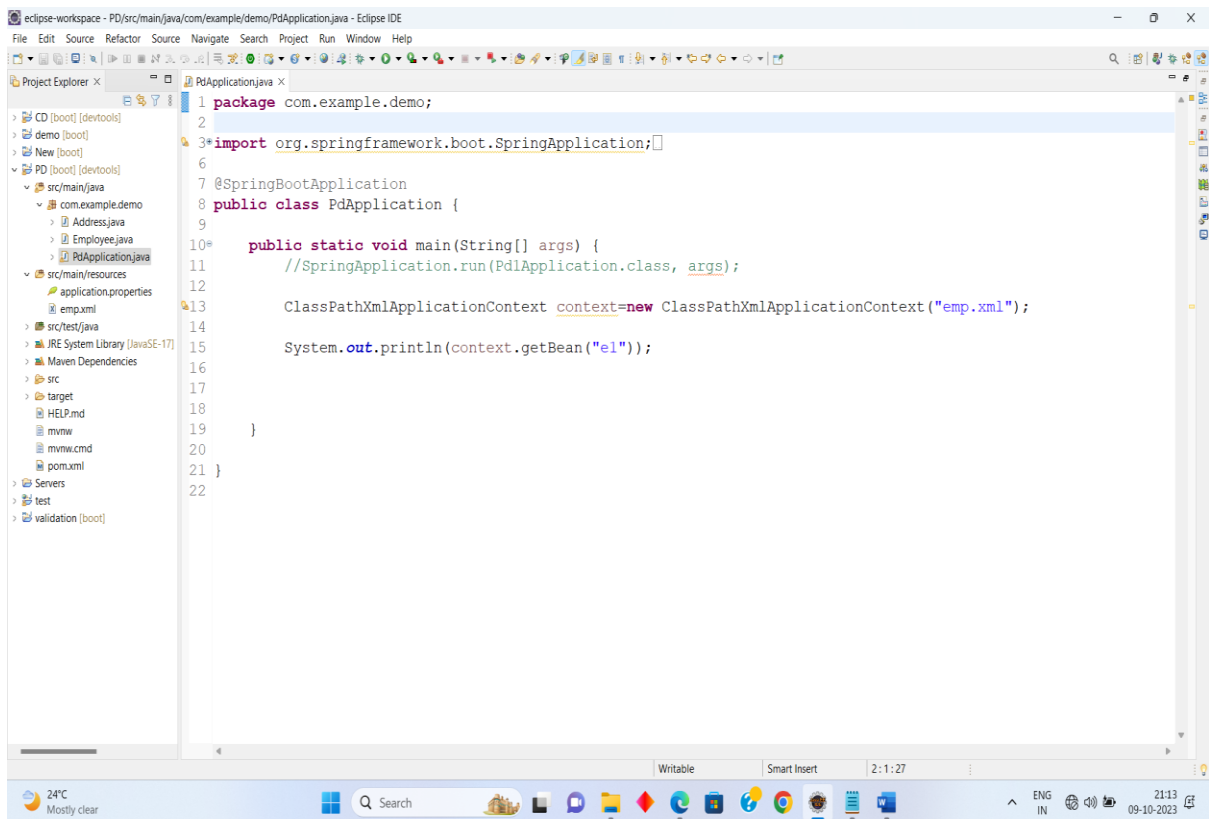
The screenshot shows the Eclipse IDE with the file Address.java open. The code defines a package com.example.demo and a class Address. The class has three private attributes: city, state, and country. It includes getter and setter methods for each attribute and a toString method that concatenates the values.

```
1 package com.example.demo;
2
3 public class Address {
4     private String city;
5     private String state;
6     private String country;
7
8     public String getCity() {
9         return city;
10    }
11    public void setCity(String city) {
12        this.city = city;
13    }
14    public String getState() {
15        return state;
16    }
17    public void setState(String state) {
18        this.state = state;
19    }
20    public String getCountry() {
21        return country;
22    }
23    public void setCountry(String country) {
24        this.country = country;
25    }
26
27    public String toString()
28    {
29        return city+" "+state+" "+country;
30    }
31 }
32
```

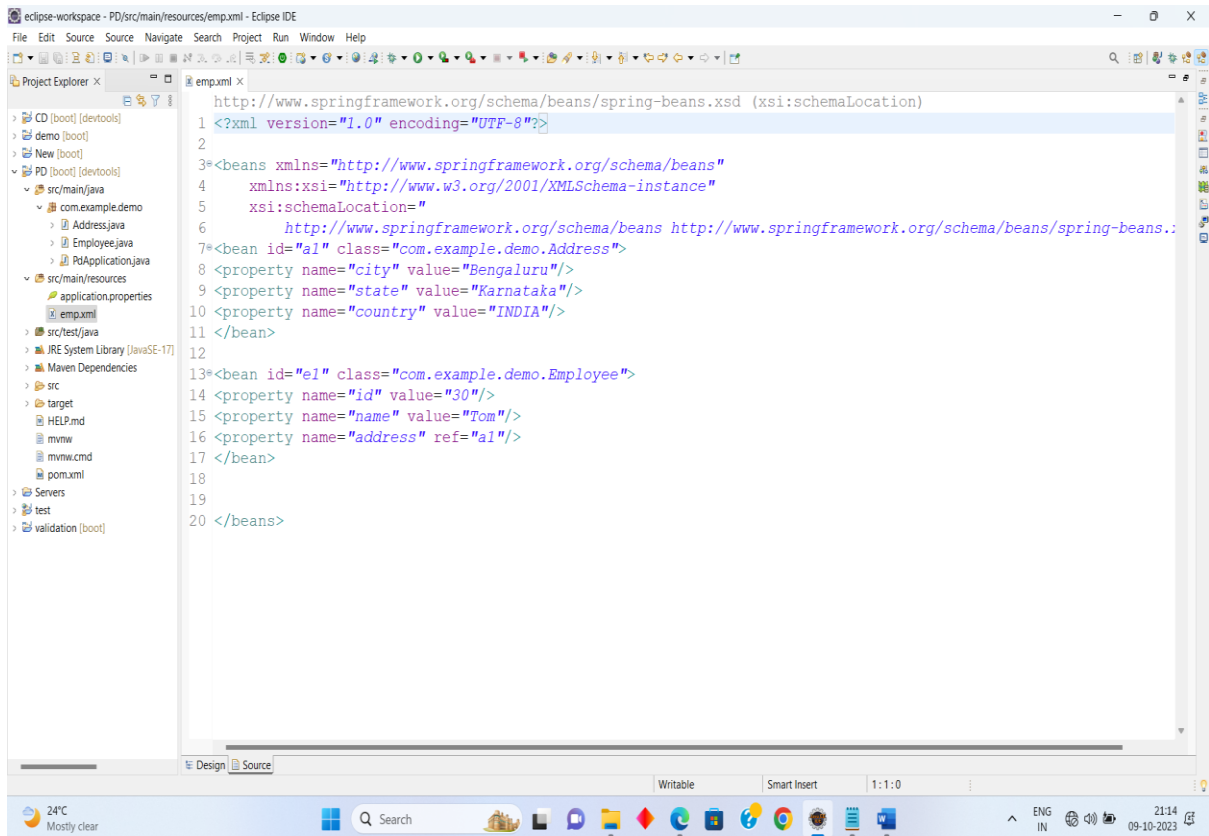


The screenshot shows the Eclipse IDE with the file Employee.java open. The code defines a package com.example.demo and a class Employee. The class has three private attributes: id, name, and address. It includes getter and setter methods for each attribute and a toString method that concatenates the values.

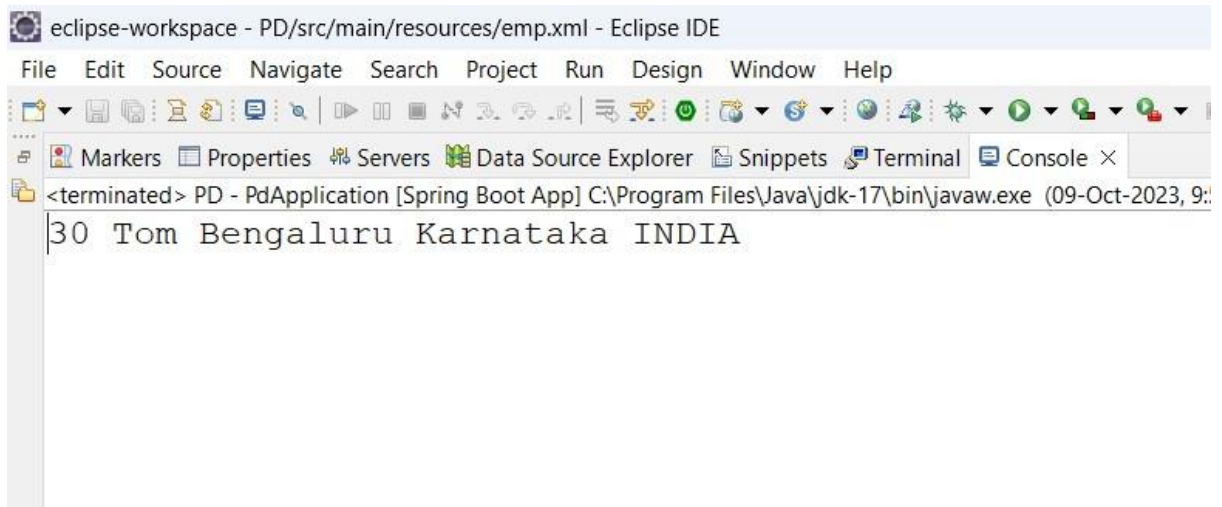
```
1 package com.example.demo;
2
3 public class Employee {
4     private int id;
5     private String name;
6     private Address address;
7
8     public int getId() {
9         return id;
10    }
11    public void setId(int id) {
12        this.id = id;
13    }
14    public String getName() {
15        return name;
16    }
17    public void setName(String name) {
18        this.name = name;
19    }
20    public Address getAddress() {
21        return address;
22    }
23    public void setAddress(Address address) {
24        this.address = address;
25    }
26
27
28    public String toString()
29    {
30        return id+" "+name+" "+address;
31    }
32 }
```



```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class PdApplication {
7
8     public static void main(String[] args) {
9         //SpringApplication.run(PdApplication.class, args);
10
11         ClassPathXmlApplicationContext context=new ClassPathXmlApplicationContext("emp.xml");
12
13         System.out.println(context.getBean("e1"));
14
15     }
16
17 }
18
19
20
21
22
```



```
1 http://www.springframework.org/schema/beans/spring-beans.xsd (xsi:schemaLocation)
2
3 <?xml version="1.0" encoding="UTF-8"?>
4
5 <beans xmlns="http://www.springframework.org/schema/beans"
6       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7       xsi:schemaLocation="
8         http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
9     <bean id="a1" class="com.example.demo.Address">
10         <property name="city" value="Bengaluru"/>
11         <property name="state" value="Karnataka"/>
12         <property name="country" value="INDIA"/>
13     </bean>
14     <bean id="e1" class="com.example.demo.Employee">
15         <property name="id" value="30"/>
16         <property name="name" value="Tom"/>
17         <property name="address" ref="a1"/>
18     </bean>
19 </beans>
20
```


Output:-A screenshot of the Eclipse IDE interface. The title bar reads 'eclipse-workspace - PD/src/main/resources/emp.xml - Eclipse IDE'. The menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Design, Window, and Help. The toolbar contains various icons for file operations and development. The 'Console' tab is active, showing the output of a Java application. The output text is: '<terminated> PD - PdApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (09-Oct-2023, 9:30 Tom Bengaluru Karnataka INDIA'.

```
<terminated> PD - PdApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (09-Oct-2023, 9:30 Tom Bengaluru Karnataka INDIA
```