Spring - Setter Injection with Non-String Map

Last Updated : 23 Jul, 2025

In Spring Framework, Dependency Injection (DI) is a core concept that allows objects to be injected into one another, reducing tight coupling. **Setter-based Dependency Injection (SDI)** is a technique where dependencies are injected through setter methods. In this article, we will explore how to perform Setter Injection with a Non-String Map in Spring.

Setter Dependency Injection (SDI)

Setter Dependency Injection is a method of injecting dependencies into a Spring bean using setter methods. In the Spring configuration file (applicationContext.xml), we use the <property> tag to define which value is assigned to the bean's variable.

Using Collections in Spring Dependency Injection

Spring provides support for injecting Java Collections such as List, Map, and Set into beans. The example below demonstrates how to inject a Map into a Spring bean using Setter injection.

The Map will have:

- **Key**: An Employee object with the following fields (Name, EmployeeID, Department).
- **Value**: An Address object with the following fields (House No, Pincode, State, Country).

Step 1: Create Employee.java

This class represents an Employee with fields:

- name, employeeId, and department.
- It includes getter and setter methods for these fields.
- The toString() method formats the Employee details for display.

```
public class Employee {
    private String name;
    private int employeeId;
    private String department;

    // Getters and Setters
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    @Override
    public String toString() {
        return "[" + name + ", " + employeeId + ", " + department + "]";
    }
}
```

Step 2: Create Address.java

This class represents an Address with fields:

- houseNo, pincode, state, and country.
- It provides getter and setter methods.

- The toString() method returns a formatted string of the Address.

```
public class Address {
    private String houseNo;
    private String pincode;
    private String state;
    private String country;

    // Getters and Setters
    public String getHouseNo() {
        return houseNo;
    }

    public void setHouseNo(String houseNo) {
        this.houseNo = houseNo;
    }

    public String getPincode() {
        return pincode;
    }

    public void setPincode(String pincode) {
        this.pincode = pincode;
    }

    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;
    }

    @Override
    public String toString() {
        return "[" + houseNo + ", " + pincode + ", " + state + ", " + country
+ "]";
    }
}
```

Step 3: Company.java

This class contains a Map<Employee, Address>, representing a mapping of Employees to their respective Addresses.

- A setter method is provided to inject the Map into the bean.

```
import java.util.Map;

public class Company {
    private Map<Employee, Address> employeeAddressMap;

    // Getter and Setter for employeeAddressMap
    public Map<Employee, Address> getEmployeeAddressMap() {
        return employeeAddressMap;
    }

    public void setEmployeeAddressMap(Map<Employee, Address>
employeeAddressMap) {
        this.employeeAddressMap = employeeAddressMap;
    }
}
```

Step 4: applicationContext.xml

This is the Spring configuration file where we define the beans and inject the Map using Setter Injection.

This is the Spring configuration file where:

- We define Employee and Address beans.
- The Company bean receives a Map, where:
  - Key: Employee bean (Sahil, 101, Game Development).
  - Value: Address bean (House No: 2, Pincode: 110111, Bihar, India).

```
<beans xmlns="http://www.springframework.org/schema/beans
       xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/bea
           http://www.springframework.org/schema/beans//spring-
beans.xsd">

    <!-- Define Employee Bean -->
    <bean id="employee" class="com.example.Employee">
```

```
        <property name="name" value="Sahil" />
        <property name="employeeId" value="101" />
        <property name="department" value="Game development" />
    </bean>

    <!-- Define Address Bean -->
    <bean id="address" class="com.example.Address">
        <property name="houseNo" value="2" />
        <property name="pincode" value="110111" />
        <property name="state" value="Bihar" />
        <property name="country" value="India" />
    </bean>

    <!-- Define Company Bean -->
    <bean id="company" class="com.example.Company">
        <property name="employeeAddressMap">
            <map>
                <entry>
                    <key>
                        <ref bean="employee" />
                    </key>
                    <ref bean="address" />
                </entry>
            </map>
        </property>
    </bean>
</beans>
```

Step 5: Test.java

This is the main class to run the application and display the output.

This is the main class that:

- Loads applicationContext.xml.
- Retrieves the Company bean.
- Iterates over the employeeAddressMap and prints the Employee with their Address.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.Map;

public class Test {
    public static void main(String[] args) {
        // Load the Spring configuration file
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve the Company bean
        Company company = (Company) context.getBean("company");

        // Display Employee and Address data
        for (Map.Entry<Employee, Address> entry :
company.getEmployeeAddressMap().entrySet()) {
            Employee employee = entry.getKey();
            Address address = entry.getValue();
            System.out.println("Employee Data -> " + employee + ", Address ->
" + address);
        }
    }
}
```

Output:

Employee Data -> [Sahil, 101, Game development], Address -> [2, 110111, Bihar, India]