**Dependency checking-**

It is used to check whether the required properties have been set or injected called as dependency checking.

**Why?**

It is not mandatory to pass the value through setter dependency injection but if you want to make setter base injection compulsory. There is one concepts called as “Dependency checking”.

**Note-**

1. Constructor dependency injection is mandatory because it is compulsory to pass the value into parameterized constructor then it will find the parameterized constructor in your class and it will create the objects.
2. Default scope of dependency checking is “None”, It is not mandatory to call setter methods.

There are following types of dependency checking.

* simple
* objects
* all
* none

**1. Simple-**

If you set dependency checking to simple then it is mandatory to call primitives types setter methods.

Program for dependency checking using simple.

**Employee.java**

**package** com.demo;

**public** **class** Employee {

**private** **int** balance

**private** Address address;

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

**public** **int** getBalance() {

**return** balance;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** getAllEmployee() {

System.***out***.println("balance" + balance);

System.***out***.println("Address>>" + address.getPincode());

}

}

**Address.java**

**package** com.demo;

**public** **class** Address {

**private** **int** pincode;

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

}

**Spring.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-2.5.xsd"*>

<bean id=*"a"* class=*"com.velocity.Address"*>

<property name=*"pincode"* value=*"123456"*></property>

</bean>

<bean id=*"e"* class=*"com.velocity.Employee"*

dependency-check=*"simple"*>

<property name=*"balance"* value=*"8000"*></property>

<property name=*"address"* ref=*"a"*></property>

</bean>

</beans>

</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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring.test</groupId>

<artifactId>PropertyFileDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<properties>

<spring.version>4.2.3.RELEASE</spring.version>

</properties>

</project>

**TestMain.java**

**package** com.demo;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");

Employee employee = (Employee) context.getBean("e");

employee.getAllEmployee();

}

}

**Output-**

balance8000

Address>>123456

**2. Objects-**

If you set dependency checking to object then it is mandatory to call secondary types setter methods.

**Employee.java**

**package** com.demo;

**public** **class** Employee {

**private** **int** balance

**private** Address address;

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

**public** **int** getBalance() {

**return** balance;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** getAllEmployee() {

System.***out***.println("balance" + balance);

System.***out***.println("Address>>" + address.getPincode());

}

}

**Address.java**

**package** com.demo;

**public** **class** Address {

**private** **int** pincode;

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

}

**Spring.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-2.5.xsd"*>

<bean id=*"a"* class=*"com.velocity.Address"*>

<property name=*"pincode"* value=*"123456"*></property>

</bean>

<bean id=*"e"* class=*"com.velocity.Employee"*

dependency-check=*"objects"*>

<property name=*"balance"* value=*"8000"*></property>

<property name=*"address"* ref=*"a"*></property>

</bean>

</beans>

</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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring.test</groupId>

<artifactId>PropertyFileDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<properties>

<spring.version>4.2.3.RELEASE</spring.version>

</properties>

</project>

**TestMain.java**

**package** com.demo;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");

Employee employee = (Employee) context.getBean("e");

employee.getAllEmployee();

}

}

**Output-**

balance8000

Address>>123456

**3. All**

If you want to make the primitive’s types as well as secondary types setter method compulsory then you should go for this type.

Program for dependency checking using all.

**Employee.java**

**package** com.demo;

**public** **class** Employee {

**private** **int** balance

**private** Address address;

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

**public** **int** getBalance() {

**return** balance;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** getAllEmployee() {

System.***out***.println("balance" + balance);

System.***out***.println("Address>>" + address.getPincode());

}

}

**Address.java**

**package** com.demo;

**public** **class** Address {

**private** **int** pincode;

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

}

**Spring.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-2.5.xsd"*>

<bean id=*"a"* class=*"com.velocity.Address"*>

<property name=*"pincode"* value=*"123456"*></property>

</bean>

<bean id=*"e"* class=*"com.velocity.Employee"*

dependency-check=*"all"*>

<property name=*"balance"* value=*"8000"*></property>

<property name=*"address"* ref=*"a"*></property>

</bean>

</beans>

</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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring.test</groupId>

<artifactId>PropertyFileDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<properties>

<spring.version>4.2.3.RELEASE</spring.version>

</properties>

</project>

**TestMain.java**

**package** com.demo;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");

Employee employee = (Employee) context.getBean("e");

employee.getAllEmployee();

}

}

**Output-**

balance8000

Address>>123456

**4. None**

If you set dependency checking to none then it is not mandatory to call setter methods without calling setter methods, you can create the object of your bean class.

Program for dependency checking using none.

**Employee.java**

**package** com.demo;

**public** **class** Employee {

**private** **int** balance

**private** Address address;

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

**public** **int** getBalance() {

**return** balance;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** getAllEmployee() {

System.***out***.println("balance" + balance);

System.***out***.println("Address>>" + address.getPincode());

}

}

**Address.java**

**package** com.demo;

**public** **class** Address {

**private** **int** pincode;

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

}

**Spring.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-2.5.xsd"*>

<bean id=*"a"* class=*"com.velocity.Address"*>

<property name=*"pincode"* value=*"123456"*></property>

</bean>

<bean id=*"e"* class=*"com.velocity.Employee"*

dependency-check=*"none"*>

<property name=*"balance"* value=*"8000"*></property>

<property name=*"address"* ref=*"a"*></property>

</bean>

</beans>

</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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring.test</groupId>

<artifactId>PropertyFileDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<properties>

<spring.version>4.2.3.RELEASE</spring.version>

</properties>

</project>

**TestMain.java**

**package** com.demo;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");

Employee employee = (Employee) context.getBean("e");

employee.getAllEmployee();

}

}

**Output-**

balance8000

Address>>123456

**@Required Annotation-**

It is used to set the particular property of class called as @Required annotation.

This is the method level annotation.

**Why?**

If there are 5 setter method or dependency in class and I want to make only one setter base injection mandatory or setter method at time. Simple dependency checking not worked, to overcome above problem we should go for @Required annotation.

Program for using @Required annotation.

**Student.java**

**package** com.velocity;

**import** org.springframework.beans.factory.annotation.Required;

**public** **class** Student {

**private** String firstName;

**private** String lastName;

**private** String mobileNumber;

**private** String city;

@Required

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

@Required

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** **void** setMobileNumber(String mobileNumber) {

**this**.mobileNumber = mobileNumber;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** **void** getStudentData() {

System.***out***.println("First Name>>" + firstName);

System.***out***.println("Last Name>>" + lastName);

System.***out***.println("Mobile Number>>" + mobileNumber);

System.***out***.println("City>>" + city);

}

}

**TestMain.java**

**package** com.demo;

**import** java.sql.SQLException;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** TestMain {

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, SQLException {

ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("spring.xml");

Student student = (Student) applicationContext.getBean("s");

student.getStudentData();

}

}

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring.test</groupId>

<artifactId>PropertyFileDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.21</version>

</dependency>

</dependencies>

<properties>

<spring.version>4.2.3.RELEASE</spring.version>

</properties>

</project>

**Spring.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"*

xmlns:context = *"http://www.springframework.org/schema/context"*

xsi:schemaLocation = *"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config/>

<bean id=*"s"* class=*"com.velocity.Student"*>

<property name=*"firstName"* value=*"jay"*></property>

<property name=*"lastName"* value=*"pawar"*></property>

<property name=*"mobileNumber"* value=*"8585859595"*></property>

<property name=*"city"* value=*"pune"*></property>

</bean>

</beans>

Output-

First Name>>jay

Last Name>>pawar

Mobile Number>>8585859595

City>>pune

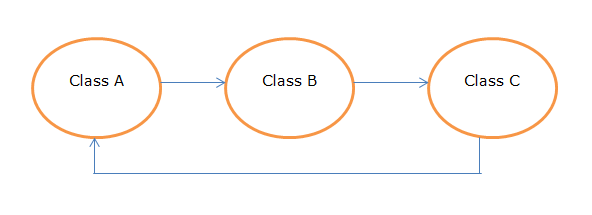
**Dependency-on**

If one bean has a dependency on another bean called as depends-on concepts.

**Why?**

If class A is depends on B, class B depends on class c then we should go for dependency on concepts.

Note- Mutual dependency is not allowed here as shown in below fig as-

Program-

**A.java**

**package** com.test;

**public** **class** A {

**public** A() {

System.***out***.println("A object");

}

}

**B.java**

**package** com.test;

**public** **class** B {

**public** B() {

System.***out***.println("B object");

}

}

**C.java**

**package** com.test;

**public** **class** C {

**public** C() {

System.***out***.println("C object");

}

}

**Spring.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-2.5.xsd"*>

<bean id=*"a"* class=*"com.demo.A"* depends-on=*"b"*></bean>

<bean id=*"b"* class=*"com.demo.B"* depends-on=*"c"*></bean>

<bean id=*"c"* class=*"com.demo.C"*></bean>

</beans>

**Test.java**

**package** com.test;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Test {

**public** **static** **void** main(String[] args) {

ApplicationContext ap=**new** ClassPathXmlApplicationContext("spring.xml");

}

}

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>spring</groupId>

<artifactId>SpringDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<properties>

<spring.version>3.2.3.RELEASE</spring.version>

</properties>

</project>

**Output-**

A object

C object

B object