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.

- none
- simple
- object
- all

1. 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.

2. Simple

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

3. Object

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

4. 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.

Employee.java

```
package com.test;

import java.util.*;

public class Employee {

	private String firstName;

	private Address address;

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public void setAddress(Address address) {
		this.address = address;
	}

	public Address getAddress() {
		return address;
	}


	public void getAllEmployee() {
		System.out.println("first name>>"+firstName);
```

```
	System.out.println("Address>>"+address.getAddressLine())
;
	}

}
```

Address.java

```
package com.test;

public class Address {

	private String addressLine;

	public String getAddressLine() {
		return addressLine;
	}

	public void setAddressLine(String addressLine) {
		this.addressLine = addressLine;
	}

}
```

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/schem
a/beans
    http://www.springframework.org/schema/beans/spring-
beans-2.5.xsd">

	<bean id="a" class="com.test.Address" dependency-
check="all">
		<property name="addressLine" value="M G
Road"></property>
```

```
    </bean>

    <bean id="e" class="com.test.Employee" dependency-
check="none">
        <property name="firstName" value="ram"></property>
        <property name="address" ref="a"></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
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>3.2.3.RELEASE</spring.version>
    </properties>

</project>
```

TestMain.java

```
package com.test;

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");
		System.out.println(employee);
	}
}
```

Output-
first name>>ram
Address>>M G Road