**Spring Bean Scope:**

1) singleton : This scope specifies , for every spring bean and for one configuration spring container creates one object. This is the default scope of all spring beans.

2) prototype: This scope indicates, creating a spring bean object on every read in spring container.

3) request: It is applicable to Web applications only. On every request sent by client it creates object.

4) session: it is also related to web applications. On every session, creates new object in container.

5)global session:- it is related to portlets concept. It creates a spring bean for the global session of portlet.

To specify the spring bean scope use bean tag scope attribute.

ex:

**package** com.app;

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

**private** **int** empId;

**private** String empName;

**public** **int** getEmpId() {

**return** empId;

}

**public** **void** setEmpId(**int** empId) {

**this**.empId = empId;

}

**public** String getEmpName() {

**return** empName;

}

**public** **void** setEmpName(String empName) {

**this**.empName = empName;

}

@Override

**public** String toString() {

**return** "Employee [empId=" + empId + ", empName=" + empName + "]";

}

}

<bean class=*"com.app.Employee"* name=*"empObj"* scope=*"singleton"*>

<property name=*"empId"* value=*"885"*/>

<property name=*"empName"* value=*"ABCD"*/>

</bean>

**package** com.app;

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

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

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

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

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("config.xml");

Employee obj = (Employee)context.getBean("empObj");

System.*out*.println(obj.hashCode());

Employee obj2 = (Employee)context.getBean("empObj");

System.*out*.println(obj2.hashCode());

Employee obj3 = (Employee)context.getBean("empObj");

System.*out*.println(obj3.hashCode());

}

}

retusn output of same hash code for all above objects.

========

to specify other scopes:

<bean class="..." name="..." scope="prototype">... </bean>

<bean class="..." name="..." scope="request">... </bean>

<bean class="..." name="..." scope="session">... </bean>

============

Lookup method Injection:-

If a independent class is prototype scope and dependent class is single tone scope then always spring container returns first time banded object with singleton class , even though it creates new object for prototype.

To change this functionality use, look up method injection, for this we need to add CGLIB Jar.

download link : (control button +mouse click to download)

<http://central.maven.org/maven2/cglib/cglib/3.1/cglib-3.1.jar>

Specify the abstract method that should return the Prototype class and call that method somewhere in the java code. make class as even abstract, that class will be implemented as proxy by CGLIB jar with same name.

ex:

java code:

**package** com.app;

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

**private** **int** addrId;

**private** String loc;

**public** **int** getAddrId() {

**return** addrId;

}

**public** **void** setAddrId(**int** addrId) {

**this**.addrId = addrId;

}

**public** String getLoc() {

**return** loc;

}

**public** **void** setLoc(String loc) {

**this**.loc = loc;

}

@Override

**public** String toString() {

**return** "Address [addrId=" + addrId + ", loc=" + loc + "]";

}

}

=====

**package** com.app;

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

**private** Address addr;

**public** Address getAddr() {

addr=createObject();

**return** addr;

}

@Override

**public** String toString() {

**return** "Employee [addr=" + addr + "]";

}

**protected** **abstract** Address createObject();

}

====

config.xml

*"*>

<bean class=*"com.app.Address"* name=*"addrObj"* scope=*"prototype"*>

<property name=*"addrId"* value=*"152"*/>

<property name=*"loc"* value=*"HYD"*/>

</bean>

<bean class=*"com.app.Employee"* name=*"empObj"* scope=*"singleton"*>

<lookup-method bean=*"addrObj"* name=*"createObject"*/>

</bean>

=======  
Test.java

**package** com.app;

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

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

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

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

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("config.xml");

Employee obj = (Employee)context.getBean("empObj");

System.*out*.println("emp:"+obj.hashCode()+",addr:"+obj.getAddr().hashCode());

Employee obj2 = (Employee)context.getBean("empObj");

System.*out*.println("emp:"+obj2.hashCode()+",addr:"+obj2.getAddr().hashCode());

Employee obj3 = (Employee)context.getBean("empObj");

System.*out*.println("emp:"+obj3.hashCode()+",addr:"+obj3.getAddr().hashCode());

Employee obj4 = (Employee)context.getBean("empObj");

System.*out*.println("emp:"+obj4.hashCode()+",addr:"+obj4.getAddr().hashCode());

}

}

output:

emp:5122060,addr:13177628

emp:5122060,addr:17152415

emp:5122060,addr:13508999

emp:5122060,addr:16471729