**Spring Inner Bean**

Often in Spring application developers has to define a bean which is solely owned by another bean. this type of bean called **inner bean**.

Think about **Gmail account** it is owned by an owner and it is not shareable. Only Owner has access to it, outside world is not permitted to see it. So in Spring If we define Gmail account as a normal bean and associate it with Owner by **ref** property it will be not a perfect solution as

Same bean can be associated with another Owner bean show this will be shared among two owners.

Which breaks the **Encapsulation**. Some people can argue and say **prototype** scope solves the problem but it certainly not, as every request to the account, it will create new bean but

***Gmail Account should be single instance per Owner***

Prototype breaks here.

**So, the ideal savior of this problem is an inner bean which will creates under the Owner context and only visible and accessible via Owner and scope of account is singleton.**

**Example,**

**Java :**

**package** com.example.innerbean;

**public** **interface** Account {

**public** **void** setName(String name);

**public** String getName();

}

**package** com.example.innerbean;

**public** **class** GmailAccount **implements** Account{

**private** String name;

@Override

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String getName() {

// **TODO** Auto-generated method stub

**return** name;

}

@Override

**public** String toString() {

**return** "GmailAccount [name=" + name + "]";

}

}

**package** com.example.innerbean;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.~~XmlBeanFactory~~;

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

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

**import** org.springframework.core.io.ClassPathResource;

**import** com.example.hello.HelloWorld;

**public** **class** EmailOwner {

Account account;

String name;

**public** Account getAccount() {

**return** account;

}

**public** **void** setAccount(Account account) {

**this**.account = account;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "EmailOwner [name=" + name + ", account=" + account + "]";

}

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

ApplicationContext ctx = **new** ClassPathXmlApplicationContext("configFiles/InnerBean.xml");

EmailOwner owner =(EmailOwner) ctx.getBean("owner");

System.***out***.println(owner);

}

}

**InnerBean.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-3.0.xsd"*>

<bean id=*"owner"* class=*"com.example.innerbean.EmailOwner"*>

<property name=*"name"* value=*"Shamik Mitra"*/>

<property name=*"account"*>

<bean id=*"gmail"* class=*"com.example.innerbean.GmailAccount"*>

<property name=*"name"* value=*"Gmail"*/>

</bean>

</property>

</bean>

</beans>

**Output**:

EmailOwner [name=Shamik Mitra, account=GmailAccount [name=Gmail]]

Please look the **InnerBean.xml** , here I declare a Bean under Owner account property,

This bean called as inner bean, When Spring container initialize the Owner bean same time it going to initialize its property and create the **GmailAccount** Object and inject the same to account property.