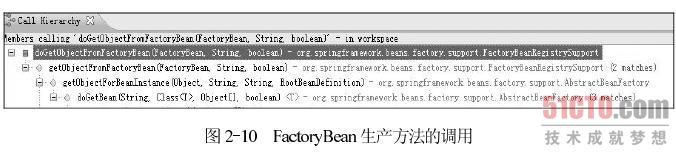
**2.5.2  FactoryBean的实现**

下面看看常见的工厂Bean是怎样实现的，这些FactoryBean为应用生成需要的对象，这些对象往往是经过特殊处理的，比如像 ProxyFactoryBean 这样的特殊 Bean。FactoryBean 的生产特性是在getBean中起作用的，我们看到下面的调用：

1. bean = getObjectForBeanInstance(sharedInstance, name, beanName, mbd);

我们看看这个 getObjectForBeanInstance 做了哪些处理，如图 2-10 所示，描述的是整个调用过程中涉及的方法。对 getObjectForBeanInstance 的实现，在这个方法里可以看到和我们在FactoryBean中常见的getObject方法的接口，详细的实现过程如代码清单2-32所示。

[](http://images.51cto.com/files/uploadimg/20091222/0938260.jpg)

代码清单2-32  FactoryBean特性的实现

1. **protected** Object getObjectForBeanInstance(   Object
2. beanInstance, String name, String beanName,
3. RootBeanDefinition mbd) {  /\*\*  \* Don't let calling code try
4. to dereference the factory if  \* the bean isn't a factory.
5. \*/  // 如果这里不是对FactoryBean的调用,那么结束处理。  if
6. (BeanFactoryUtils.isFactoryDereference(name) &&
7. !(beanInstance **instanceof** FactoryBean)) {    **throw** **new**
8. BeanIsNotAFactoryException(transformedBeanName(name),beanIns
9. tance. getClass());  }  /\*\*  \* Now we have the bean
10. instance, which may be a normal bean or a FactoryBean.  \* If
11. it's a FactoryBean, we use it to create a bean instance,
12. unless the  \* caller actually wants a reference to the
13. factory.  \*/  **if** (!(beanInstance **instanceof** FactoryBean)
14. ||BeanFactoryUtils.isFactory Dereference(name)) {   **return**
15. beanInstance;  }  Object object = **null**;  **if** (mbd == **null**) {
16. object = getCachedObjectForFactoryBean(beanName);  }  **if**
17. (object == **null**) {   // Return bean instance from factory.
18. FactoryBean factory = (FactoryBean) beanInstance;   //
19. Caches object obtained from FactoryBean **if** it is a
20. singleton.   **if** (mbd == **null** &&
21. containsBeanDefinition(beanName)) {    mbd =
22. getMergedLocalBeanDefinition(be}   **boolean** synthetic = (mbd
23. != **null** && mbd.isSynthetic());
24. //这里从FactoryBean中得到bean。   object =
25. getObjectFromFactoryBean(factory, beanName, !synthetic);  }
26. **return** object; } **protected** Object
27. getObjectFromFactoryBean(FactoryBean factory, String
28. beanName, **boolean** shouldPostProcess) {  **if**
29. (factory.isSingleton() && containsSingleton(beanName)) {
30. **synchronized** (getSingletonMutex()) {    Object object =
31. **this**.factoryBeanObjectCache.get(beanName);    **if** (object ==
32. **null**) {     object = doGetObjectFromFactoryBean(factory,
33. beanName, shouldPostProcess);
34. **this**.factoryBeanObjectCache.put(beanName, (object != **null** ?
35. object : NULL\_OBJECT));    }    **return** (object !=
36. NULL\_OBJECT ? object : **null**);   }  }  **else** {   **return**
37. doGetObjectFromFactoryBean(factory, beanName,
38. shouldPostProcess);  } } **private** Object
39. doGetObjectFromFactoryBean(   **final** FactoryBean factory,
40. **final** String beanName, **final** Boolean    shouldPostProcess)
41. **throws** BeanCreationException {  AccessControlContext acc =
42. AccessController.getContext();  **return**
43. AccessController.doPrivileged(**new** PrivilegedAction<Object>()
44. { **public** Object run() {    Object object;
45. //这里调用factory的getObject方法来从FactoryBean中得到bean。
46. **try** {     object = factory.getObject();    }    **catch**
47. (FactoryBeanNotInitializedException ex) {     **throw** **new**
48. BeanCurrentlyInCreationException(beanName, ex.toString());
49. }    **catch** (Throwable ex) {     **throw** **new**
50. BeanCreationException(beanName, "FactoryBean threw
51. exception on object creation", ex);    }     /\*\*    \* Do not
52. accept a null value for a FactoryBean that's not fully    \*
53. initialized yet: Many FactoryBeans just return null then.
54. \*/    **if** (object == **null** &&
55. isSingletonCurrentlyInCreation(beanName)) {     **throw** **new**
56. BeanCurrentlyInCreationException(       beanName,
57. "FactoryBean which is currently in creation        returned
58. **null** from getObject");    }    **if** (object != **null** &&
59. shouldPostProcess) {     **try** {      object =
60. postProcessObjectFromFactoryBean(object, beanName);     }
61. **catch** (Throwable ex) { ean's object failed", ex);     }    }
62. **return** object;   }  }, acc);}

这里返回的已经是作为工厂的 FactoryBean 生产的产品，并不是 FactoryBean 本身。这种FactoryBean的机制可以为我们提供一个很好的封装机制，比如封装Proxy、RMI、JNDI等。经过对FactoryBean实现过程的原理分析，相信读者会对getObject方法有很深刻的印象。这个方法就是主要的FactoryBean 的接口，需要实现特定的工厂的生产过程，至于这个生产过程是怎样和IoC容器整合的，就是我们在上面分析的内容。