Spring 中工厂模式简介

工厂模式作用：

核心思想：实例化对象

需求：

1.多例对象的创建

2.有些对象 例如接口/抽象类不能直接实例化 接口–代理 抽象类–创建子类

3.需要对对象进行二次封装

静态工厂：

说明： 静态工厂必须有静态方法。

调用方式：类名.静态方法

例：

（1）建立工厂类

```
package com.jt.manage.factory;

import java.util.Calendar;

public class StaticFactory {
	public static Calendar getInstance(){
		return Calendar.getInstance();
	}
}
```

```
1
2
3
4
5
```

```
6
7
8
9
```

（2）交给 Spring 管理

配置 xml 文件

```
<!--配置静态工厂    -->
        <bean id="calendarA" class="com.jt.manage.factory.StaticFactory" factory-
method="getInstance" />
```

```
1
2
```

静态工厂可以通过类名.静态方法的形式获取对象，因此只需要将静态工厂对象交给 Spring 管理,并提供静态工厂方法即可。

实例工厂：

说明:实例化工厂要求必须先创建工厂对象,之后通过工厂方法调用获取对象

（1）建立工厂类

```
package com.jt.manage.factory;

import java.util.Calendar;

public class InstanceFactory {
        public    Calendar getInstance(){
                return Calendar.getInstance();
        }
}
```

```
1
2
3
```

```
4
5
6
7
8
9
```

（2）交给 Spring 管理

配置 xml 文件

```
        <!--配置实例化工厂  -->
        <bean id="instanceFactory" class="com.jt.manage.factory.InstanceFactory"></bean>
        <bean id="calendarB" class="com.jt.manage.factory.InstanceFactory" factory-
bean="instanceFactory" factory-method="getInstance"></bean>
```

```
1
2
3
```

实例工厂需要将工厂对象交给 Spring 管理，然后建立 bean 对象，将工厂对象和工厂对象的获取对象方法交给其即可。

Spring 工厂：

说明:该模式有 spring 内部调用,不需要做多余的配置但是需要实现特定的接口 FactoryBean。

（1）建立工厂类

```
public class CalendarFactory implements FactoryBean<Calendar> {

        @Override
        public Calendar getObject() throws Exception {
                // TODO Auto-generated method stub
                return Calendar.getInstance();
```

```
	}

	@Override
	public Class<?> getObjectType() {
		// TODO Auto-generated method stub
		return Calendar.class;
	}

	@Override
	public boolean isSingleton() {
		// TODO Auto-generated method stub
		return false;
	}
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
16
17
18
19
```

（2）交给 Spring 管理

```
<!--配置 spring 工厂    -->
        <bean id="calendarC" class="com.jt.manage.factory.CalendarFactory"></bean>
```

```
1
2
```

FactoryBean 接口需要实现三个方法并制定工厂管理的对象泛型。

isSingleton()方法：用于指定对象是否为单例，单例对象由 Spring 创建、存储并管理；多例对象 Spring 只负责创建，不负责销毁。

getObjectType() 方法：若对象为单例，则在 Spring 容器关闭时调用此方法，获得该对象的 class 对象并通过 class 对象寻找所有该类对象进行删除。

getObject()方法：获取对象。

工具类测试：

```
@Test
        public void testFactory(){
                //1.实例化 Spring 容器
                ApplicationContext context =
                                        new ClassPathXmlApplicationContext("spring/factory.xml");
                Calendar calendarA = (Calendar) context.getBean("calendarA");
                System.out.println(calendarA.getTime());

                Calendar calendarB = (Calendar) context.getBean("calendarB");
                System.out.println(calendarB.getTime());

                Calendar calendarC = (Calendar) context.getBean("calendarB");
                System.out.println(calendarC.getTime());
```

```
    }
```