

huxin1的专栏

□□

空间 博客 好友 相册 留言

用户操作

[留言] [发消息] [加为好友]

订阅我的博客

37 位读者
POWERED BY FEEDSKY

订阅

订阅到 鲜果

订阅到 Google

订阅到 抓虾

huxin1的公告

文章分类

APACHE 与T

OMCATE整合

C++

extjs

FLASH

hibernate使

用及源码学习

J2EE(其他)

javascript基

础

java基础知识

MySql

Oracle

spring securi

ty

原 spring入门 - Aware相关接口 收藏

Spring中提供一些Aware相关接口，像是BeanFactoryAware、ApplicationContextAware、ResourceLoaderAware、ServletContextAware等等，实作这些 Aware接口的Bean在被初始之后，可以取得一些相对应的资源，例如实作BeanFactoryAware的Bean在初始后，Spring容器将会注入BeanFactory的实例，而实作ApplicationContextAware的Bean，在Bean被初始后，将会被注入 ApplicationContext的实例等等。

Bean取得BeanFactory、ApplicationContextAware的实例目的是什么，一般的目就是要取得一些档案资源的存取、相 关讯息资源或是那些被注入的实例所提供的机制，例如ApplicationContextAware提供了publishEvent()方法，可以支持基于Observer模式的事件传播机制。

ApplicationContextAware接口的定义如下：

ApplicationContextAware.java

```
public interface ApplicationContextAware {  
    void setApplicationContext(ApplicationContext context);  
}
```

我们这边示范如何透过实作ApplicationContextAware注入ApplicationContext来实现事件传播，首先我们的HelloBean如下：

HelloBean.java

```
package onlyfun.caterpillar;
```


```
import org.springframework.context.*;
```


```
public class HelloBean implements ApplicationContextAware {  
    private ApplicationContext applicationContext;  
    private String helloWord = "Hello!World!";
```

```
    public void setApplicationContext(ApplicationContext context) {  
        this.applicationContext = context;
```

 spring源码学

习

 sql server

 Tomcat使用

 Tomcat源码


学习

 UML


 Web Service


s

 佛法

 其他知识


 问题列表


 项目测试

 项目管理

 心情写照

 需求分析

 英语

 正方工作日志

存档

2010年05月(3)

2010年04月(2)

2010年02月(1)

2010年01月(6)

2009年12月(4)

2009年11月(5)

2009年10月(1)

2009年09月(5)

2009年08月(12)

2009年07月(4)

2009年06月(2)

2009年03月(3)

2009年02月(5)

2009年01月(13)

2008年12月(13)

2008年11月(31)

2008年10月(3)

2008年09月(8)

2008年08月(1)

```
}
```

```
public void setHelloWord(String helloWord) {  
    this.helloWord = helloWord;  
}
```

```
public String getHelloWord() {  
    applicationContext.publishEvent(  
        new PropertyGettedEvent "[" + helloWord + "] is getted");  
    return helloWord;  
}  
}
```

ApplicationContext会由Spring容器注入，publishEvent()方法需要一个继承ApplicationEvent的对象，我们的PropertyGettedEvent继承了ApplicationEvent，如下：

PropertyGettedEvent.java

package onlyfun.caterpillar;

```
import org.springframework.context.*;
```

```
public class PropertyGettedEvent extends ApplicationEvent {  
    public PropertyGettedEvent(Object source) {  
        super(source);  
    }  
}
```

当ApplicationContext执行publishEvent()后，会自动寻找实作ApplicationListener接口的对象并通知其发生对应事件，我们实作了PropertyGettedListener如下：

PrppertyGettedListener.java

package onlyfun.caterpillar;

```
import org.springframework.context.*;
```

```
public class PropertyGettedListener implements ApplicationListener {  
    public void onApplicationEvent(ApplicationEvent event) {  
        System.out.println(event.getSource().toString());  
    }  
}
```

Listener必须被实例化，这我们可以在Bean定义档中加以定义：

2008年05月(4)
2008年04月(9)
2008年03月(2)
2008年01月(1)
2007年11月(1)
2007年08月(2)
2007年07月(1)
2006年12月(1)
2006年10月(4)
2006年09月(1)
2006年01月(1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING/DTD BEAN/EN" "http://www.springfram
ework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="propertyGetterListener" class="onlyfun.caterpillar.PropertyGettedLi
stener"/>

    <bean id="helloBean" class="onlyfun.caterpillar.HelloBean">
        <property name="helloWord"><value>Hello!Justin!</value></property>
    </bean>
</beans>
```

我们写一个测试程序来测测事件传播的运行：

Test.java

```
package onlyfun.caterpillar;
```

```
import org.springframework.context.*;
```

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

```
public class Test {
```

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

```
        ApplicationContext context = new ClassPathXmlApplicationContext("bean.x
ml");
```

```
        HelloBean hello = (HelloBean) context.getBean("helloBean");
```

```
        System.out.println(hello.getHelloWord());
```

```
    }
```

```
}
```

执行结果会如下所示：

```
log4j:WARN No appenders could be found for logger
```

```
(org.springframework.beans.factory.xml.XmlBeanDefinitionReader).
```

```
log4j:WARN Please initialize the log4j system properly.
```

```
org.springframework.context.support.ClassPathXmlApplicationContext:
```

```
displayName=[org.springframework.context.support.ClassPathXmlApplicationCo
ntext;
```

```
hashCode=33219526]; startup date=[Fri Oct 29 10:56:35 CST 2004];
```

```
root of ApplicationContext hierarchy
```

```
[Hello!Justin!] is getted
```

```
Hello!Justin!
```

以上是以实作事件传播来看看实作**Aware**接口取得对应对象后，可以进行的动作，同样的，您也可以实作**ResourceLoaderAware**接口：

ResourceLoaderAware.java

```
public interface ResourceLoaderAware {  
    void setResourceLoader(ResourceLoader loader);  
}
```

实作**ResourceLoader**的Bean就可以取得**ResourceLoader**的实例，如此就可以使用它的**getResource()**方法，这对于必须存取档案资源的Bean相当有用。

基本上，**Spring**虽然提供了这些**Aware**相关接口，然而Bean上若实现了这些界面，就算是与**Spring**发生了依赖，从另一个角度来看，虽然您可以直接在Bean上实现这些接口，但您也可以透过**setter**来完成依赖注入，例如：

HelloBean.java

```
package onlyfun.caterpillar;
```

```
import org.springframework.context.*;
```

```
public class HelloBean {  
    private ApplicationContext applicationContext;  
    private String helloWord = "Hello!World!";  
  
    public void setApplicationContext(ApplicationContext context) {  
        this.applicationContext = context;  
    }  
  
    public void setHelloWord(String helloWord) {  
        this.helloWord = helloWord;  
    }  
  
    public String getHelloWord() {  
        applicationContext.publishEvent(new PropertyGettedEvent "[" + helloWord  
+ "]" is getted));  
        return helloWord;  
    }  
}
```

注意这次我们并没有实作**ApplicationContextAware**，我们在程序中可以自行注入**ApplicationContext**实例：

```
ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");
```

```
HelloBean hello = (HelloBean) context.getBean("helloBean");  
hello.setApplicationContext(context);  
System.out.println(hello.getHelloWord());
```

就Bean而言，降低了对Spring的依赖，可以比较容易从现有的框架中脱离。

发表于 @ 2009年02月16日 15:46:00 | [评论\(0\)](#) | [举报](#) | [收藏](#)

旧一篇:c++关键字详解(volatile, mutable, explicit, dynamic_cast(expression))等 | 新一篇:spring构造注入



18个月，自考本科助学

发表评论

表情:



评论内容:

用户名: 匿名用户

[登录](#) [注册](#)

验证码:




[重新获得验证码](#)

[公司简介](#) | [广告服务](#) | [银行汇款帐号](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#)

北京创新乐知广告有限公司 版权所有, 京 ICP 证 070598 号

世纪乐知(北京)网络技术有限公司 提供技术支持

 Email:webmaster@csdn.net

Copyright © 1999-2010, CSDN.NET, All Rights Reserved

