08 青出于蓝而胜于蓝之揭秘Spring容器ApplicationContext

更新时间：2020-08-10 14:44:01

> 才能一旦让懒惰支配，它就一无可为。——克雷洛夫

> 青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳，輮以为轮，其曲中规。虽有槁暴，不复挺者，輮使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎已，则知明而行无过矣。-----《荀子·劝学》

背景

用过 Spring 的童鞋都知道，BeanFactory 是 Spring 最重要的最简单的 Bean 容器，但是由于 BeanFactory 的功能有些简单，所以它并不适合实际应用的企业级开发，因此，Spring 提供了另外的一套 Bean 容器管理的体系-ApplicationContext 体系。

ApplicationContext 简介

BeanFactory 和 ApplicationContext 都是 spring 的 IoC 容器，BeanFactory 提供了一种可以管理任意类型对象的高级配置机制，ApplicationContext 继承自 BeanFactory，并新增了以下功能：

- spring aop 特性的早期集成；
- 国际化的 messageSource ；
- 事件发布；
- 应用层特定的上下文，如 Web 应用层的 WebApplicationContext。

简单的说，BeanFactory 提供了配置框架和基本功能，ApplicationContext 增加了更多特定的企业功能。

为了更好的理解，我们看看 ApplicationContext 的定义吧：

```
public interface ApplicationContext extends EnvironmentCapable, ListableBeanFactory, HierarchicalBeanFactory,
  MessageSource, ApplicationEventPublisher, ResourcePatternResolver {
 ..........
}
```

ApplicationContext 接口继承众多接口，集众多接口功能与一身，为 Spring 的运行提供基本的功能支撑。根据程序设计的“单一职责原则”，其实每个较顶层接口都是“单一职责的”，只提供某一方面的功能，而 ApplicationContext 接口继承了众多接口，相当于拥有了众多接口的功能。

下面看看它的主要功能：

它继承了 EnvironmentCapable 的功能，可获取环境相关信息：是开发环境、测试环境、沙箱环境还是生产环境；

它继承了 ListableBeanFactory 的功能，可以管理、装配 Bean，可以有父级 BeanFactory 实现 Bean 的层级管理（具体到这里来说它可以有父级的 ApplicationContext，因为 ApplicationContext 本身就是一个 BeanFactory。这在 Web 项目中很有用，可以使每个 Servlet 具有其独立的 context， 所有 Servlet 共享一个父级的 context），它还是 Listable 的，可以枚举出所管理的 Bean 对象；

它继承了 HierarchicalBeanFactory 的功能，可以获取父 BeanFactory，也可以根据名称判断 bean 是否在此 beanFactory 中；

它继承了 MessageSource 的功能，可以管理一些 Message 实现国际化等功能；

它继承了 ApplicationEventPublisher 的功能，可以发布事件给注册的 Listener，实现监听机制；

它继承了 ResourcePatternResolver 的功能，可以加载资源不同样式的资源文件。

ApplicationContext 实例

都说 Talk is cheap. Show me the code，那就来啦。

javaBean类

```
package com.davidwang456.test;
public class Employee {
    private int age;
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

配置文件

SpringBeans.xml 位于resouces目录下面

```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" xmlns:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans
  <bean id="bean1" class="com.davidwang456.test.Employee">
    <property name="name" value="Rakesh" />
    <property name="age" value="20" />
  </bean>
</beans>
```

测试类

```java
package com.davidwang456.test;

import org.springframework.context.ApplicationContext;

/**
 *
 * @since 5.2
 */
public class ApplicationContextTest {
    /**
     * @param args
     */
    public static void main(String[] args) {
        @SuppressWarnings("resource")
        ApplicationContext context = new ClassPathXmlApplicationContext("SpringBeans.xml");
        Employee emp = (Employee) context.getBean("bean1");
        System.out.println(emp.getAge());
    }
}
```

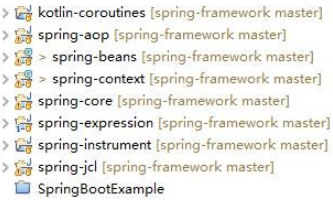
代码使用代码框展示(注意缩进及格式)，如下

```
public class ApplicationContextTest{
   public static void main(String[] args){
      @SuppressWarnings("resource")
ApplicationContext context =
        new ClassPathXmApplicationContext("SpringBeans.xml");
 Employee emp = (Employee) context.getBean("bean1");
 System.out.print(emp.getAge());
 }
}
```

上一篇《07：别整虚的！揭开 spring IOC、DI 的神秘面纱》，有一个从 XmlBeanFactory 获取 bean 的实例，和本节的 ClassPathXmlApplicationContext 是不是很相似，但后者的程序数量远小于前者哦，ApplicationContext 的功能强大可见一斑。

深入 ApplicationContext 原理

将 Spring-context 源码导入到 Eclipse（或者 IDEA），并创建自己的 package 和 class，如下图：



- 容器的启动

使用给定的父容器，创建容器 ClassPathXmlApplicationContext ，从给定的 xml 文件中加载 Bean 定义，如下图：

```java
/**
 * Create a new ClassPathXmlApplicationContext with the given parent,
 * loading the definitions from the given XML files.
 * @param configLocations array of resource locations
 * @param refresh whether to automatically refresh the context,
 * loading all bean definitions and creating all singletons.
 * Alternatively, call refresh manually after further configuring the context.
 * @param parent the parent context
 * @throws BeansException if context creation failed
 * @see #refresh()
 */
public ClassPathXmlApplicationContext(
        String[] configLocations, boolean refresh, @Nullable ApplicationContext parent)
        throws BeansException {

    super(parent);
    setConfigLocations(configLocations);
    if (refresh) {
        refresh();
    }
}
```

子调用 AbstractApplicationContext.java，如下图：

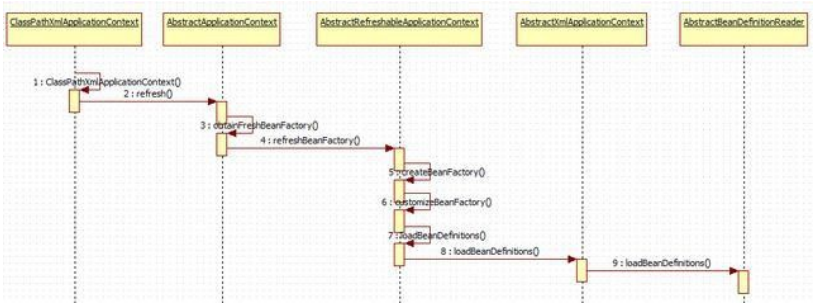
```java
@Override
public void refresh() throws BeansException, IllegalStateException {
    synchronized (this.startupShutdownMonitor) {
        // Prepare this context for refreshing.
        prepareRefresh();

        // Tell the subclass to refresh the internal bean factory.
        ConfigurableListableBeanFactory beanFactory = obtainFreshBeanFactory();

        // Prepare the bean factory for use in this context.
        prepareBeanFactory(beanFactory);

        try {
            // Allows post-processing of the bean factory in context subclasses.
            postProcessBeanFactory(beanFactory);

            // Invoke factory processors registered as beans in the context.
            invokeBeanFactoryPostProcessors(beanFactory);

            // Register bean processors that intercept bean creation.
            registerBeanPostProcessors(beanFactory);

            // Initialize message source for this context.
            initMessageSource();

            // Initialize event multicaster for this context.
            initApplicationEventMulticaster();

            // Initialize other special beans in specific context subclasses.
            onRefresh();

            // Check for listener beans and register them.
            registerListeners();

            // Instantiate all remaining (non-lazy-init) singletons.
            finishBeanFactoryInitialization(beanFactory);

            // Last step: publish corresponding event.
            finishRefresh();
        }
```

如果一一深入代码，本篇文章的长度会超出想象，故本次抽取关键步骤：获取 BeanFactory 加以深入。

获取 BeanFactory

// Tell the subclass to refresh the internal bean factory.

ConfigurableListableBeanFactory beanFactory = obtainFreshBeanFactory();



其中涉及到 **BeanFactory** 的产生，读取配置文件 **springbeans.xml** 中，使用了 **ResourceLoader** 和 **ResourcePatternResolver** 将文件转换为 **Resource** 资源。

另外：

初始化MessageSource
// Initialize message source for this context.
initMessageSource();

初始化事件发射器
// Initialize event multicaster for this context.
initApplicationEventMulticaster();

总结

罗马不是一天建成的，ApplicationContext 也是站在巨人的肩头的。它的主要功能主要来自于继承：

它继承了 EnvironmentCapable 的功能，可获取环境相关信息：是开发环境、测试环境、沙箱环境还是生产环境；

它继承了 ListableBeanFactory 的功能，可以管理、装配 Bean，可以有父级 BeanFactory 实现 Bean 的层级管理（具体到这里来说它可以有父级的 ApplicationContext，因为 ApplicationContext 本身就是一个 BeanFactory。这在 Web 项目中很有用，可以使每个 Servlet 具有其独立的 context， 所有 Servlet 共享一个父级的 context），它还是 Listable的，可以枚举出所管理的 Bean 对象；

它继承了 HierarchicalBeanFactory 的功能，可以获取父 BeanFactory，也可以根据名称判断 bean 是否在此 BeanFactory 中；

它继承了 MessageSource 的功能，可以管理一些 Message 实现国际化等功能；

它继承了 ApplicationEventPublisher 的功能，可以发布事件给注册的 Listener，实现监听机制；

它继承了 ResourcePatternResolver 的功能，可以加载资源不同样式的资源文件。

}

07 别整虚的！揭开Spring IoC、DI的神秘面纱

09 Spring IoC容器ApplicationContext如何实现国际化