【死磕 Spring】—— IOC 之 BeanDefinition 注册机：BeanDefinitionRegistry (http://cmsblogs.com/?p=4026)

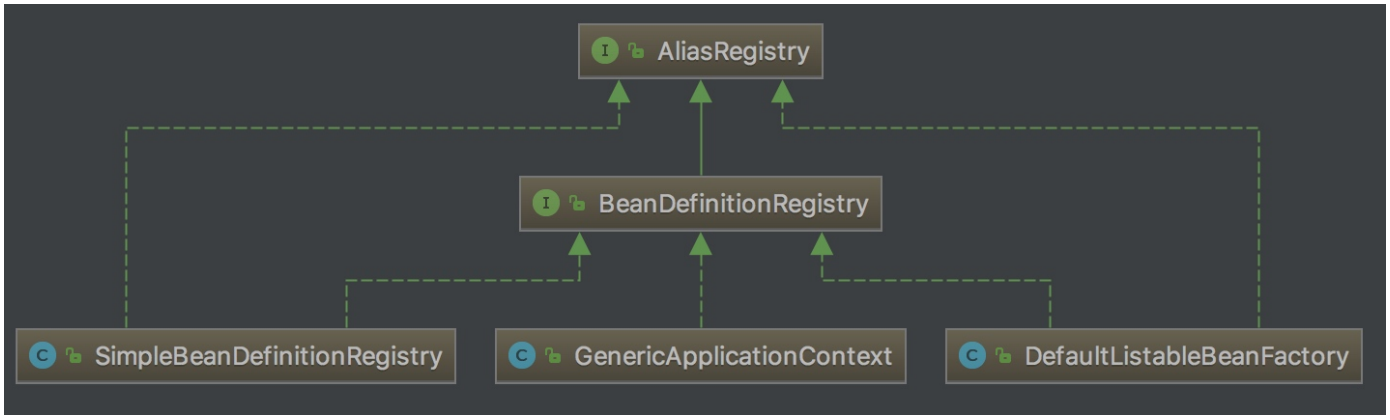
2019-01-28 分类：死磕 Spring (http://cmsblogs.com/?cat=206) 阅读(10381) 评论(0)

> 原文出自：http://cmsblogs.com (http://cmsblogs.com)

将定义 bean 的资源文件解析成 BeanDefinition 后需要将其注入容器中，这个过程由 BeanDefinitionRegistry 来完成。beanDefinitinMap中put("beanName",BeanDefinition)

BeanDefinitionRegistry：向注册表中注册 BeanDefinition 实例，完成注册的过程。

下图是 BeanDefinitionRegistry 类结构图：



(https://gitee.com/chenssy/blog-home/raw/master/image/201811/15397559425630.jpg)

BeanDefinitionRegistry 继承了 AliasRegistry 接口，其核心子类有三个：SimpleBeanDefinitionRegistry、DefaultListableBeanFactory、GenericApplicationContext。

AliasRegistry

用于别名管理的通用型接口，作为 BeanDefinitionRegistry 的顶层接口。 AliasRegistry 定义了一些别名管理的方法。

```
public interface AliasRegistry {
    void registerAlias(String name, String alias);

    void removeAlias(String alias);

    boolean isAlias(String name);

    String[] getAliases(String name);
}
```

BeanDefinitionRegistry

BeanDefinition 的注册接口，如 RootBeanDefinition 和 ChildBeanDefinition。它通常由 BeanFactories 实现，在 Spring 中已知的实现者为：DefaultListableBeanFactory 和 GenericApplicationContext 。BeanDefinitionRegistry 是 Spring 的 Bean 工厂包中唯一封装 BeanDefinition 注册的接口。

BeanDefinitionRegistry 接口定义了关于 BeanDefinition 注册、注销、查询等一系列的操作。

```
public interface BeanDefinitionRegistry extends AliasRegistry {
    // 往注册表中注册一个新的 BeanDefinition 实例
    void registerBeanDefinition(String beanName, BeanDefinition beanDefinition)
            throws BeanDefinitionStoreException;

    // 移除注册表中已注册的 BeanDefinition 实例
    void removeBeanDefinition(String beanName) throws NoSuchBeanDefinitionException;

    // 从注册中取得指定的 BeanDefinition 实例
    BeanDefinition getBeanDefinition(String beanName) throws NoSuchBeanDefinitionException;

    // 判断 BeanDefinition 实例是否在注册表中（是否注册）
    boolean containsBeanDefinition(String beanName);

    // 取得注册表中所有 BeanDefinition 实例的 beanName（标识）
    String[] getBeanDefinitionNames();

    // 返回注册表中 BeanDefinition 实例的数量
    int getBeanDefinitionCount();

    // beanName（标识）是否被占用
    boolean isBeanNameInUse(String beanName);
}
```

SimpleBeanDefinitionRegistry

SimpleBeanDefinitionRegistry 是 BeanDefinitionRegistry 一个简单的实现，它还继承 SimpleAliasRegistry（AliasRegistry 的简单实现），它仅仅只提供注册表功能，无工厂功能。

SimpleBeanDefinitionRegistry 使用 ConcurrentHashMap 来存储注册的 BeanDefinition。

```
private final Map<String, BeanDefinition> beanDefinitionMap = new ConcurrentHashMap<>(64);
```

他对注册其中的 BeanDefinition 都是基于 beanDefinitionMap 这个集合来实现的，如下：

```
@Override
public void registerBeanDefinition(String beanName, BeanDefinition beanDefinition)
  throws BeanDefinitionStoreException {

 Assert.hasText(beanName, "'beanName' must not be empty");
 Assert.notNull(beanDefinition, "BeanDefinition must not be null");
 this.beanDefinitionMap.put(beanName, beanDefinition);
}

@Override
public void removeBeanDefinition(String beanName) throws NoSuchBeanDefinitionException {
 if (this.beanDefinitionMap.remove(beanName) == null) {
  throw new NoSuchBeanDefinitionException(beanName);
 }
}

@Override
public BeanDefinition getBeanDefinition(String beanName) throws NoSuchBeanDefinitionException {
 BeanDefinition bd = this.beanDefinitionMap.get(beanName);
 if (bd == null) {
  throw new NoSuchBeanDefinitionException(beanName);
 }
 return bd;
}
```

实现简单、粗暴。

DefaultListableBeanFactory

DefaultListableBeanFactory ， ConfigurableListableBeanFactory （其实就是 BeanFactory ） 和 BeanDefinitionRegistry 接口的默认实现：一个基于 BeanDefinition 元数据的完整 bean 工厂。所以相对于 SimpleBeanDefinitionRegistry 而言，DefaultListableBeanFactory 则是一个具有注册功能的完整 bean 工厂。它同样是用 ConcurrentHashMap 数据结构来存储注册的 BeanDefinition。

```
// 注册表，由 BeanDefinition 的标识 （beanName） 与其实例组成
private final Map<String, BeanDefinition> beanDefinitionMap = new ConcurrentHashMap<String, bean>(64);

// 标识（beanName）集合
private final List<String> beanDefinitionNames = new ArrayList<String>(64);
```

在看 registerBeanDefinition() ：

```
public void registerBeanDefinition(String beanName, BeanDefinition beanDefinition)
    throws BeanDefinitionStoreException {

        // 省略其他代码

    else {
     if (hasBeanCreationStarted()) {
      // Cannot modify startup-time collection elements anymore (for stable iteration)
      synchronized (this.beanDefinitionMap) {
       this.beanDefinitionMap.put(beanName, beanDefinition);
       List<String> updatedDefinitions = new ArrayList<>(this.beanDefinitionNames.size() + 1);
       updatedDefinitions.addAll(this.beanDefinitionNames);
       updatedDefinitions.add(beanName);
       this.beanDefinitionNames = updatedDefinitions;
       if (this.manualSingletonNames.contains(beanName)) {
        Set<String> updatedSingletons = new LinkedHashSet<>(this.manualSingletonNames);
        updatedSingletons.remove(beanName);
        this.manualSingletonNames = updatedSingletons;
       }
      }
     }
     else {
      // 注册 BeanDefinition
      this.beanDefinitionMap.put(beanName, beanDefinition);
      this.beanDefinitionNames.add(beanName);
      this.manualSingletonNames.remove(beanName);
     }
     this.frozenBeanDefinitionNames = null;
    }

    if (existingDefinition != null || containsSingleton(beanName)) {
     resetBeanDefinition(beanName);
    }
   }
```

其实上面一堆代码最重要就只有一句：

```
this.beanDefinitionMap.put(beanName, beanDefinition);
```

`removeBeanDefinition()` 其实也是调用 `beanDefinitionMap.remove(beanName)`。

对于类 GenericApplicationContext，查看源码你会发现他实现注册、注销功能都是委托 DefaultListableBeanFactory 实现的。

> 所以 BeanDefinition 注册并不是非常高大上的功能，内部就是用一个 Map 实现，并不是多么高大上的骚操作，所以有时候我们会潜意识地认为某些技术很高大上就觉得他很深奥，如果试着去一探究竟你会发现，原来这么简单。虽然 BeanDefinitionRegistry 实现简单，但是它作为 Spring IOC 容器的核心接口，其地位还是很重的。