**2.3  IoC容器的初始化**

IoC容器的初始化包括BeanDefinition的Resouce定位、载入和注册这三个基本的过程。在前面的编程式地使用DefaultListableBeanFactory中，我们可以看到定位和载入过程的接口调用。这里将详细分析这三个过程的实现。值得注意的是，Spring在实现中是把这三个过程分开并使用不同的模块来完成的，这样可以让用户更加灵活地对这三个过程进行剪裁和扩展，定义出最适合自己的IoC容器的初始化过程。

BeanDefinition的资源定位由ResourceLoader通过统一的Resource接口来完成，这个Resource对各种形式的BeanDefinition的使用提供了统一接口。对于这些BeanDefinition的存在形式，相信大家都不会感到陌生。比如说，在文件系统中的Bean定义信息可以使用FileSystemResource来进行抽象；在类路径中可以使用前面提到的ClassPathResource来使用，等等。这个过程类似于容器寻找数据的过程，就像用水桶装水先要把水找到一样。

第二个关键的部分是BeanDefinition的载入，该载入过程把用户定义好的Bean表示成IoC容器内部的数据结构，而这个容器内部的数据结构就是BeanDefinition，下面可以看到这个数据结构的详细定义。总地说来，这个BeanDefinition实际上就是POJO对象在IoC容器中的抽象，这个BeanDefinition定义了一系列的数据来使得IoC容器能够方便地对POJO对象也就是Spring的Bean进行管理。即BeanDefinition就是Spring的领域对象。下面我们会对这个载入的过程进行详细的分析，便于大家对整个过程有比较清楚的了解。

第三个过程是向IoC容器注册这些BeanDefinition的过程。这个过程是通过调用BeanDefinitionRegistry接口的实现来完成的，这个注册过程把载入过程中解析得到的BeanDefinition向IoC容器进行注册。可以看到，在IoC容器内部，是通过使用一个HashMap来持有这些BeanDefinition数据的。

值得注意的是，IoC容器和上下文的初始化一般不包含Bean依赖注入的实现。一般而言，依赖注入发生在应用第一次向容器通过getBean索取Bean时。但有一个例外值得注意，在使用IoC容器时有一个预实例化的配置，这个预实例化是可以配置的，具体来说可以通过在Bean定义信息中的lazyinit属性来设定；有了这个预实例化的特性，用户可以对容器初始化过程作一个微小的控制；从而改变这个被设置了lazyinit属性的Bean的依赖注入的发生，使得这个Bean的依赖注入在IoC容器初始化时就预先完成了。有了以上的一个大概的轮廓，下面就详细地看一看在IoC容器的初始化过程中，BeanDefinition的资源定位、载入和解析过程是怎么实现的。