I can see how referring to a bean by calling its method can be confusing. There’s

another way that might be easier to digest:

@Bean

public CDPlayer cdPlayer(CompactDisc compactDisc) {

return new CDPlayer(compactDisc);

}

Here, the cdPlayer() method asks for a CompactDisc as a parameter. When Spring

calls cdPlayer() to create the CDPlayer bean, it autowires a CompactDisc into the

configuration method. Then the body of the method can use it however it sees fit.

With this technique, the cdPlayer() method can still inject the CompactDisc into the

CDPlayer’s constructor without explicitly referring to the CompactDisc’s @Bean

method.

This approach to referring to other beans is usually the best choice because it

doesn’t depend on the CompactDisc bean being declared in the same configuration

class. In fact, there’s nothing that says the CompactDisc bean even needs to be

declared in JavaConfig; it could have been discovered by component scanning or

declared in XML. You could break up your configuration into a healthy mix of configuration

classes, XML files, and automatically scanned and wired beans. No matter how

the CompactDisc was created, Spring will be happy to hand it to this configuration

method to create the CDPlayer bean.

Spring In Action 4th Edition(2.3.3)