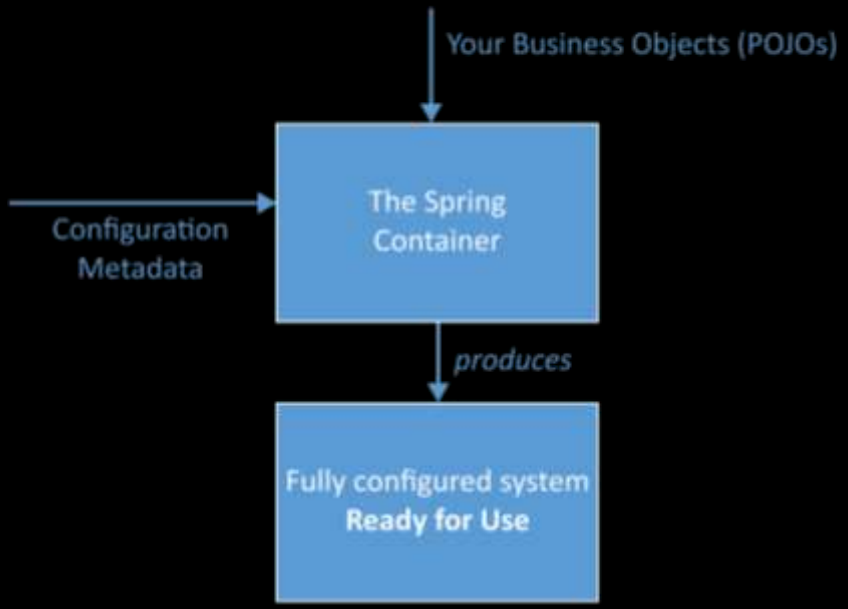
Dependency configuration

Configuration Metadata


Your Business Objects (POJOs)
Configuration Metadata
The Spring Container
produces
Fully configured system Ready for Use

As shown on diagram, the Spring IoC container consumes a form of configuration metadata. This configuration metadata represents how you, as an application developer, tell the Spring container to instantiate, configure, and assemble the objects in your application.

Configuration metadata can be supplied by:

- XML-based configuration
- Annotation-based configuration
- Java-based configuration

XML-based configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="..." class="..."> ① ②
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <bean id="..." class="...">
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

① The **id** attribute is a string that identifies the individual bean definition.

② The **class** attribute defines the type of the bean and uses the fully qualified classname.

Composing XML-based Configuration Metadata

```
<beans>
    <import resource="services.xml"/>
    <import resource="resources/messageSource.xml"/>
    <import resource="/resources/themeSource.xml"/>

    <bean id="bean1" class="..."/>
    <bean id="bean2" class="..."/>
</beans>
```

It can be useful to have bean definitions span multiple XML files. Often, each individual XML configuration file represents a logical layer or module in your architecture.
You can use the application context constructor to load bean definitions from all these XML fragments or use one or more occurrences of the *<import/>* element to load bean definitions from another file or files as shown above.

Annotation-based Configuration

Annotation-based configuration is an alternative to XML setup. It relies on the bytecode metadata for wiring up components instead of angle-bracket declarations.
Annotation injection is performed before XML injection. Thus, the XML configuration overrides the annotations for properties wired through both approaches.
By default, Spring annotation wiring is not turned on in Spring Framework. Therefore, you need to enable it before you can use the Spring annotation-based wiring in the Spring Configuration file.

```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:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-
context.xsd">

    <context:annotation-config/>

</beans>
```

Once the tag *<context:annotation-config/>* is configured you have the authority to start annotating your code. It will indicate that the Spring should automatically wire the values into properties methods and constructors.

Annotations for annotation-based Configuration

- @Required: It is applicable to bean property setter methods.
- @Autowired: It is only applied to the bean property setter methods, constructors, non-setter methods and properties.
- @Qualifier: This Spring Framework annotation along with the @Autowired is used for removing the confusion by specifying the exact bean to wire.
- JSR-250 Annotations: These Spring annotations are supported by Spring Framework including @Resource, @PreDestroy and @PostConstruct annotations. These annotations are not really required as you already have the alternatives.

Java-based Container Configuration

The central artifacts in Spring's new Java-configuration support are **@Configuration**-annotated classes and **@Bean**-annotated methods.
The **@Bean** annotation is used to indicate that a method instantiates, configures, and initializes a new object to be managed by the Spring IoC container.

```
@Configuration
public class AppConfig {

    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
}
```

The preceding *AppConfig* class is equivalent to the following Spring *<beans/>* XML

```
<beans>
    <bean id="myService" class="com.acme.servic
es.MyServiceImpl"/>
</beans>
```

You can use **@Bean**-annotated methods with any Spring **@Component**. However, they are most often used with **@Configuration** beans. Annotating a class with **@Configuration** indicates that its primary purpose is as a source of bean definitions. Furthermore, **@Configuration** classes let inter-bean dependencies be defined by calling other **@Bean** methods in the same class.