Learning Consolidation

Develop Backend Application by Using Spring Framework

Learning Objectives

- Explore the Spring Framework
- Configure the Spring Core Container
- Define Beans in the Spring Core Container

Slide Note

The Spring Framework

- Spring is a powerful, flexible, fast, secure, and lightweight framework used for building Java web applications.
- The Spring Framework is a well-defined tool that supports several web applications using Java as a programming language.
- Spring is an open source project and has a large and active community.
- The Spring framework is divided into modules.
- Modules are a set or package of classes that provide functionality for the Spring framework.
- Applications can choose which modules they need.

Menu

00:03:00:00:10 03/ 10

Slide Note

● Core Container

The *Core Container* consists of the Core, Beans, Context, and Expression Language modules.

The *Core and Beans* modules provide the fundamental parts of the framework, including the including the Inversion of Control and Dependency Injection features.

● Data Access/Integration

The *Data Access/Integration* layer consists of the JDBC, ORM, OXM, JMS, and Transaction modules.

The JDBC module provides a JDBC-abstraction layer that eliminates the need to do tedious JDBC coding and parsing of database-vendor- specific error codes.

● Web

The *Web* layer consists of the Web, Web-Servlet, Web-Struts, and Web-Portlet modules.

Spring's *Web* module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.

● AOP (Aspect-Oriented Programming)

Spring's *AOP* module provides an *AOP Alliance*-compliant, aspect-oriented programming implementation.

Instrumentation

The *Instrumentation* module provides class instrumentation support and classloader implementations to be used in certain application servers.

Test

The *Test* module supports the testing of Spring components with JUnit or TestNG.

Components of Spring

- The Spring framework consists of features organized into about 20 modules, or independent functionalities.
- These modules are grouped into:
  - Core Container
  - Data Access/Integration
  - Web
  - AOP (Aspect-Oriented Programming)
  - Instrumentation
  - Test

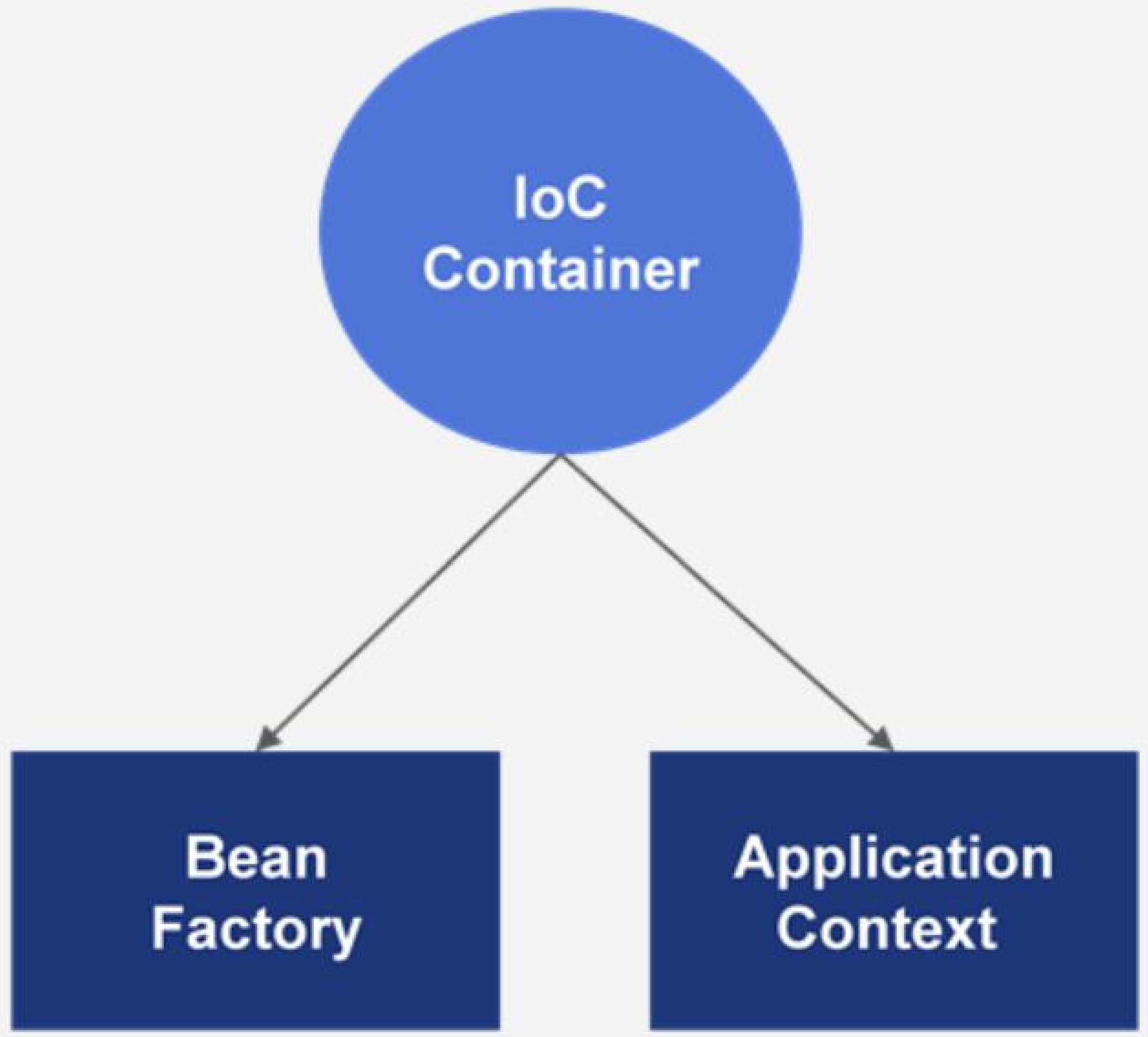
Slide Note

IoC and DI

- The two most important aspects of the core container are:
  - Inversion of Control (IoC)
  - Dependency Injection (DI)
- LoC is a mechanism by which the control of objects is transferred to a container or framework.
- It is used in the context of object-oriented programming.
- DI is a pattern that can be used to implement IoC, where the control is inverted and given to the framework.
- Connecting objects with other objects, or injecting objects into other objects, is done by the framework itself.

Menu

Slide Note



IoC Container

- There are two types of IoC containers:
  - `BeanFactory` – Provides a configuration framework and basic functionality to manage objects.
  - `ApplicationContext` – A sub-interface of the `BeanFactory` that provides more enterprise-specific functionality.

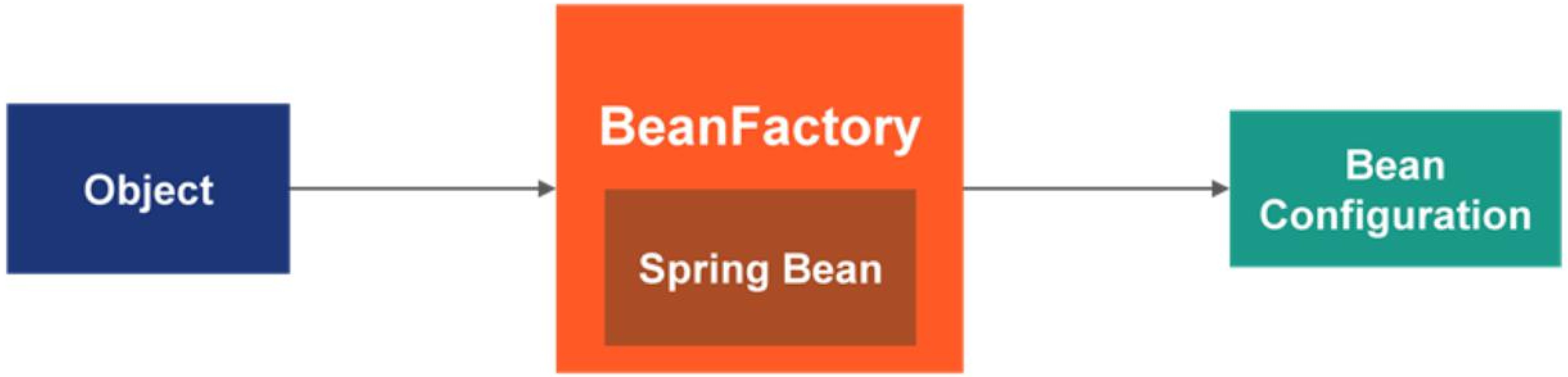
Menu

00:06:00:00:10 06/ 10

Slide Note

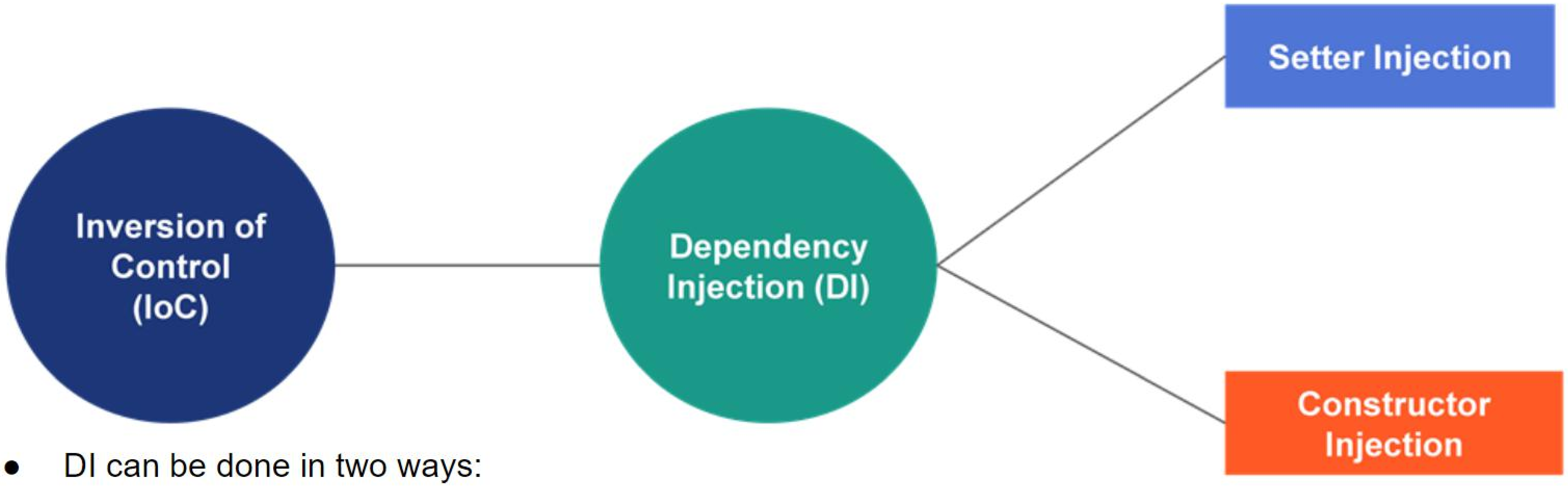
Spring-Managed Objects:

- Spring has a `BeanFactory` to create new objects.
- Instead of a hard-coded new object, Spring `BeanFactory` creates an object.
- To create objects, or beans, `BeanFactory` reads from the configuration file that contains the bean definition.
- Since the new bean is created in the `BeanFactory`, Spring knows and manages the lifecycle of this bean. Spring acts as a container for this new bean or object created.



Slide Note

Dependency Injection (DI) – Injecting Objects



- DI can be done in two ways:
  - Setter – Setter injection is accomplished by the container calling the setter methods on the beans after invoking a no-argument constructor to instantiate the bean.
  - Constructor – Constructor-based DI is accomplished when the container invokes a class constructor with the specified number of arguments, each representing a dependency on the other class.

Steps for Configuration

1. Create a Java Maven project with `archetype` as `quickstart` and add the dependency below in `pom.xml;` this will set up the `BeanFactory` and `ApplicationContext`.

```xml
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.22</version>
</dependency>
```

2. Create the domain classes whose objects will be handled within the Spring container.
3. Define a configuration class that will have all bean definitions for the domain class objects. The `@Bean` will be used here.
4. Use the `AnnotationConfigApplicationContext` class to access the beans declared within the container.