Spring Framework

Introduction

- The Spring Framework is an open source application framework and inversion of control container for the java platform.
- The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.
- Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBean (EJB) model.

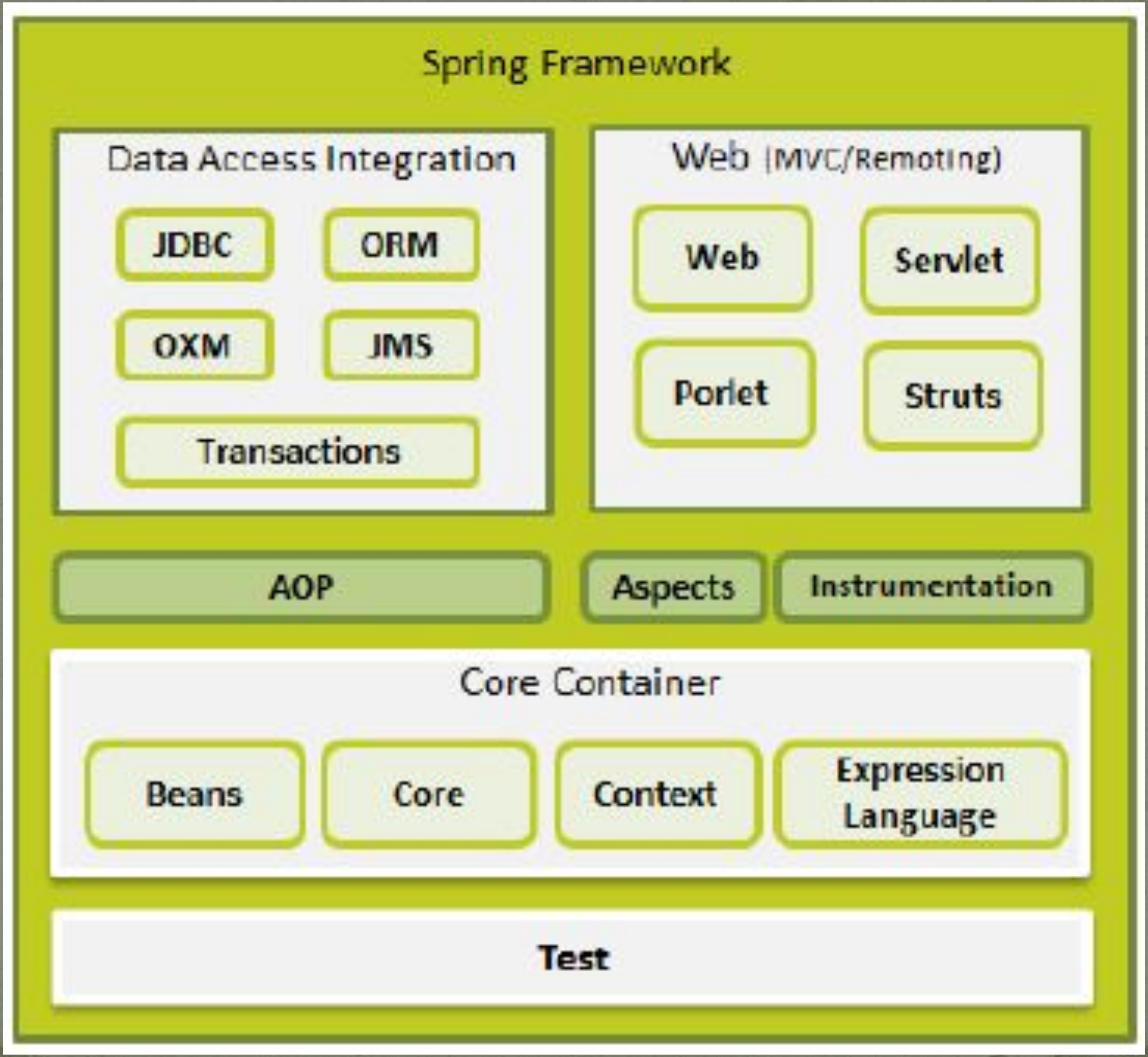
Continue…..

- Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, reusable code.
- Spring framework is an open source Java platform and it was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.
- Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.

- The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promote good programming practice by enabling a POJO-based programming model.

Spring Architecture

- Spring could potentially be a one-stop shop for all your enterprise applications, however, Spring is modular, allowing you to pick and choose which modules are applicable to you, without having to bring in the rest.
- The Spring Framework provides about 20 modules which can be used based on an application requirement.


Spring Framework
Data Access Integration
JDBC
ORM
OXM
JMS
Transactions
Web (MVC/Remoting)
Web
Servlet
Porlet
Struts
AOP
Aspects
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

Core Container

- The Core Container consists of the Core, Beans, Context, and Expression Language modules whose detail is as follows:
- The Core module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
- The Bean module provides BeanFactory which is a sophisticated implementation of the factory pattern.

Continue…

- The Context module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.
- The Expression Language module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access Integration

- The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows:
- The JDBC module provides a JDBC-abstraction layer that removes the need to do tedious JDBC related coding.
- The ORM module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
- The OXM module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.

- The Java Messaging Service JMS module contains features for producing and consuming messages.
- The Transaction module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web

- The Web layer consists of the Web, Web-Servlet, Web-Struts, and Web-Portlet modules whose detail is as follows:
- The 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.
- The Web-Servlet module contains Spring's model-view-controller (MVC) implementation for web applications.
- The Web-Struts module contains the support classes for integrating a classic Struts web tier within a Spring application.

- The Web-Portlet module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous

- There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules whose detail is as follows:
- The AOP module provides aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
- The Aspects module provides integration with AspectJ which is again a powerful and mature aspect oriented programming (AOP) framework.

- The Instrumentation module provides class instrumentation support and class loader implementations to be used in certain application servers.
- The Test module supports the testing of Spring components with JUnit or TestNG frameworks.

Spring Framework Definition

- The Spring framework is a comprehensive tool for supporting applications using Java programming language.
- Many Web applications and other technologies use Java, and the Spring framework provides an array of resources for integrating applications, streamlining code and generally promoting more efficient development strategies.

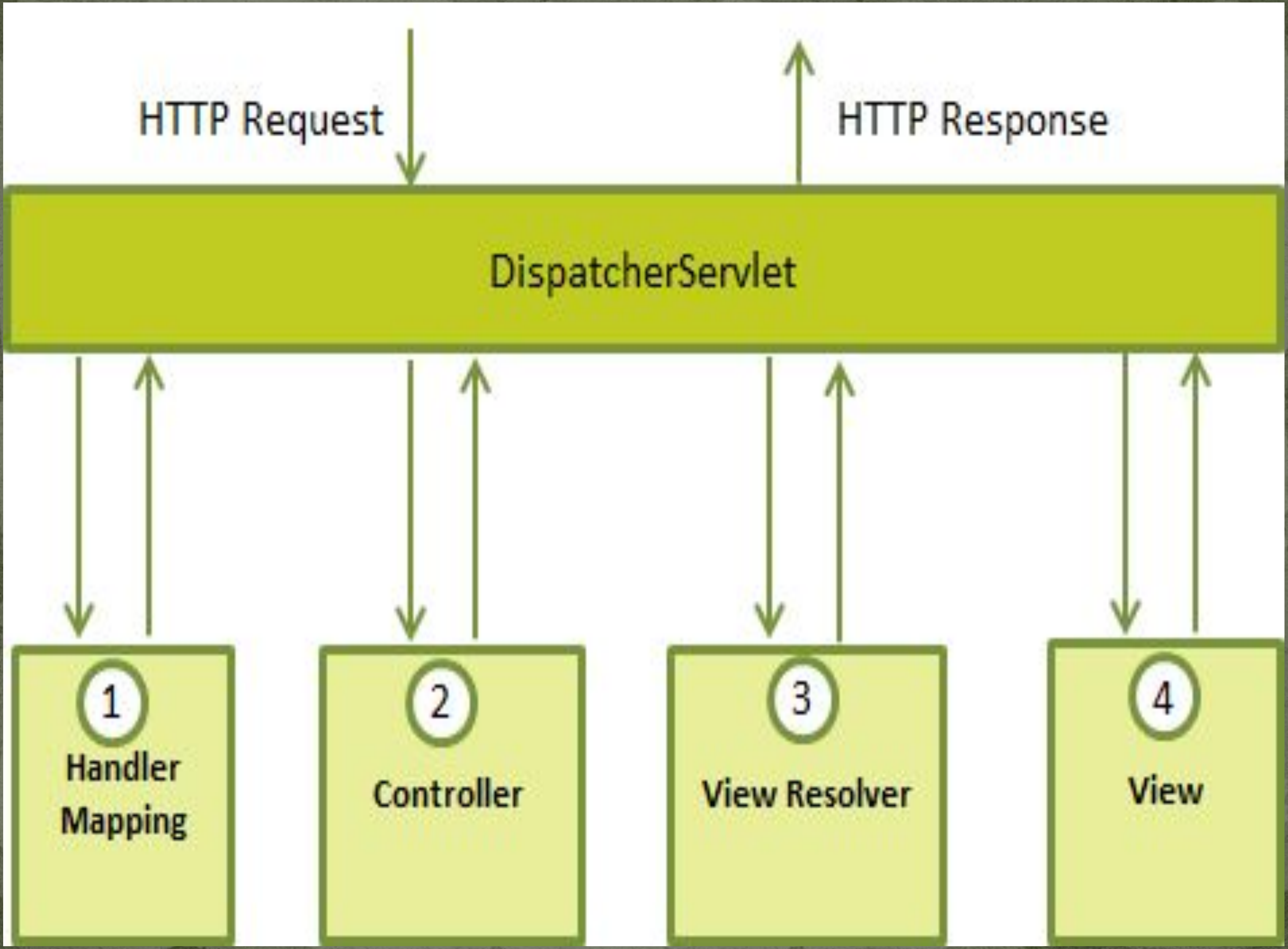
Spring & MVC

- The Spring web MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications.
- The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

- The **Model** encapsulates the application data and in general they will consist of POJO.
- The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
- The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

The DispatcherServlet

- The Spring Web model-view-controller (MVC) framework is designed around a *DispatcherServlet* that handles all the HTTP requests and responses. The request processing workflow of the Spring Web MVC *DispatcherServlet* is illustrated in the following diagram:


HTTP Request
HTTP Response
DispatcherServlet
1
Handler Mapping
2
Controller
3
View Resolver
4
View

- Following is the sequence of events corresponding to an incoming HTTP request to *DispatcherServlet*:
- After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate *Controller*.
- The *Controller* takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.
- The *DispatcherServlet* will take help from *ViewResolver* to pickup the defined view for the request.
- Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser.

- All the above mentioned components ie. HandlerMapping, Controller and ViewResolver are parts of *WebApplicationContext* which is an extension of the plain *ApplicationContext* with some extra features necessary for web applications.

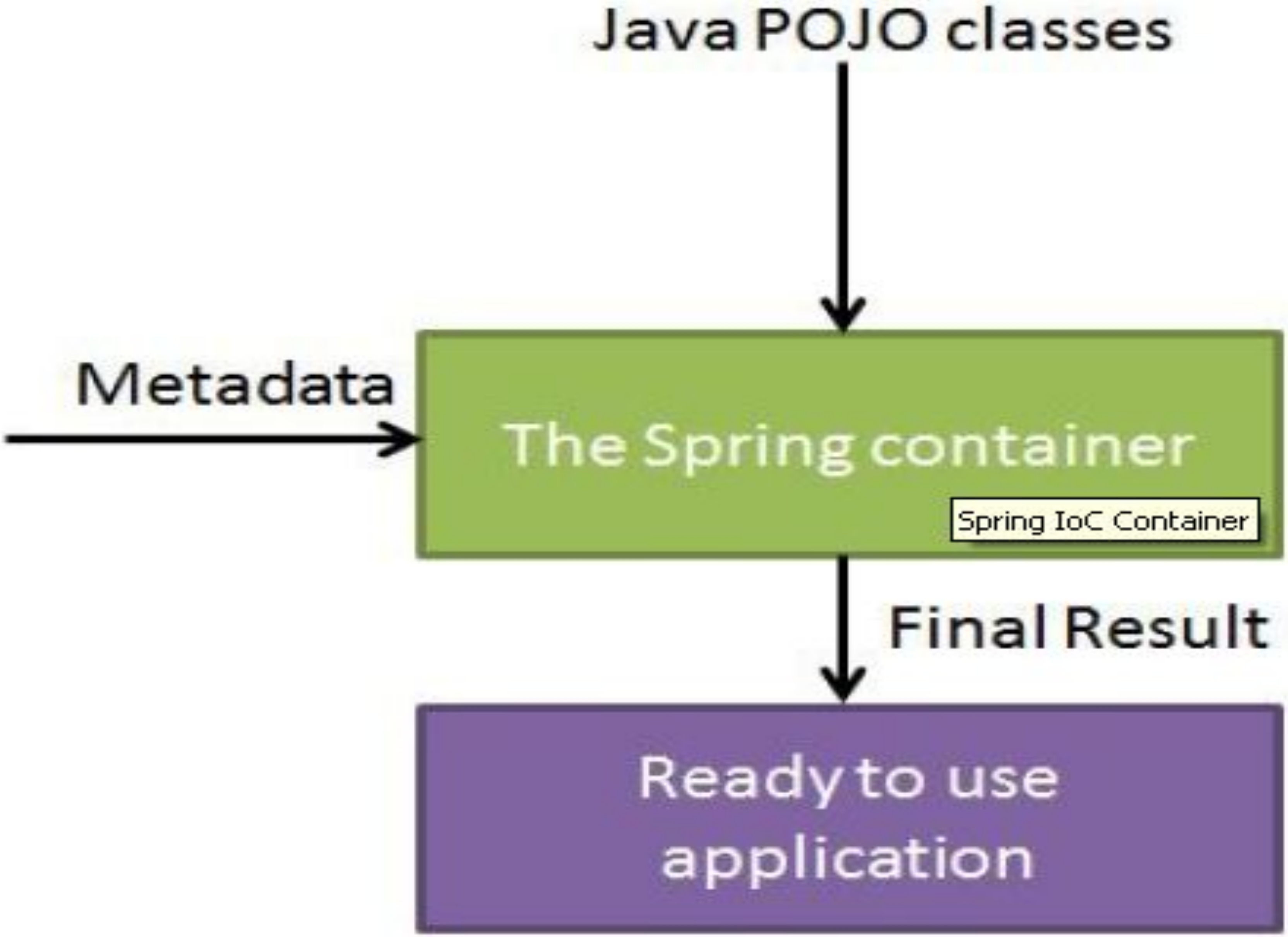
Spring Context Definition

- Inside a Spring application are beans, and these beans interact with each other to provide the application services.
- The interacting beans are also called *collaborators*, and you can define them using application contexts. This is the process of "wiring" your beans together. Listing 2 shows an example of an application context.

IOC (Inversion Of Control)

- The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction. The Spring container uses dependency injection (DI) to manage the components that make up an application. These objects are called Spring Beans.
- The container gets its instructions on what objects to instantiate, configure, and assemble by reading configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code.

- The following diagram is a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.


Java POJO classes
Metadata
The Spring container
Spring IoC Container
Final Result
Ready to use application

Spring provides following two distinct types of containers.

- **Spring BeanFactory Container :**
- This is the simplest container providing basic support for DI and defined by the*org.springframework.beans.factory.BeanFactory interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purposes of backward compatibility with the large number of third-party frameworks that integrate with Spring.*

- **Spring ApplicationContext Container**:
- This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the *org.springframework.context.ApplicationContextinterface.*

Aspect oriented programming

- One of the key components of Spring Framework is the **Aspect oriented programming (AOP)framework. Aspect Oriented Programming entails breaking down program logic into distinct parts called so-called concerns. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects like logging, auditing, declarative transactions, security, and caching etc.**

- The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Dependency Injection helps you decouple your application objects from each other and AOP helps you decouple cross-cutting concerns from the objects that they affect. AOP is like triggers in programming languages such as Perl, .NET, Java and others.
- Spring AOP module provides interceptors to intercept an application, for example, when a method is executed, you can add extra functionality before or after the method execution.

AOP Terminologies

Terms	Description
Aspect	A module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging. An application can have any number of aspects depending on the requirement.
Join point	This represents a point in your application where you can plug-in AOP aspect. You can also say, it is the actual place in the application where an action will be taken using Spring AOP framework.
Advice	This is the actual action to be taken either before or after the method execution. This is actual piece of code that is invoked during program execution by Spring AOP framework.
Pointcut	This is a set of one or more joinpoints where an advice should be executed. You can specify pointcuts using expressions or patterns.

AOP Terminologies

Terms	Description
Introduction	An introduction allows you to add new methods or attributes to existing classes.
Target object	The object being advised by one or more aspects, this object will always be a proxied object. Also referred to as the advised object.
Weaving	Weaving is the process of linking aspects with other application types or objects to create an advised object. This can be done at compile time, load time, or at runtime.