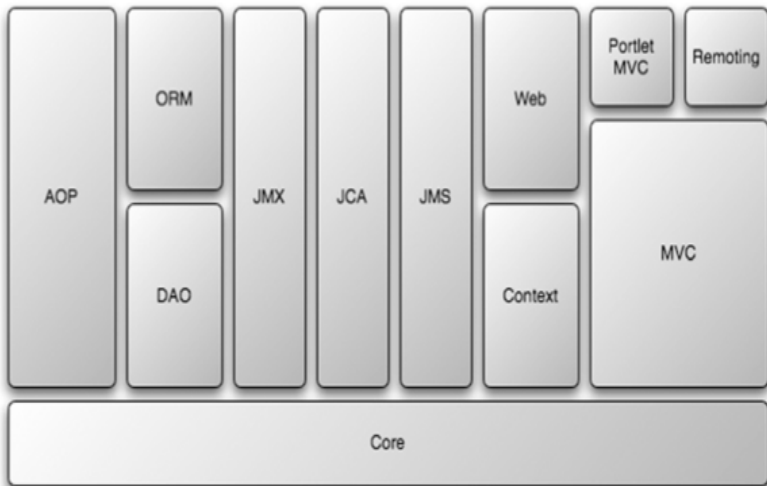


SPRING IN ACTION

Carmine Dodaro

- Open source framework
- Created to address the complexity of enterprise application development
- Any Java application can benefit from Spring in terms of simplicity and testability



- The Spring Framework is made up of several modules
- Do I need to use all of the modules? No, it is possible to use a fragment of Spring Framework
- Spring offers integration points with several other frameworks and libraries

Important Features

- Is **Lightweight**, in terms of both size (more or less 2.5 MB) and overhead (negligible)
- Supports **Dependency Injection (DI)**, objects are passively given their dependencies instead of creating or looking for dependent objects for themselves
- Supports **Aspect Oriented Programming (AOP)**, i.e. application objects are not responsible for other system aspects, such as logging
- Is a **Container**, in the sense that it contains and manages the lifecycle and configuration of application objects
- Is a **Framework**, since it allows to configure and compose complex applications from simpler components

DEPENDENCY INJECTION

- All real-world applications are made up of two or more classes that collaborate with each other
- In general, each object is responsible for obtaining its own references to the objects it collaborates with
- Using Dependency Injection objects are given their dependencies at creation time by some external entity that coordinates each object in the system
- Objects are not responsible for finding or creating the other objects that they need
- Instead, they are given references to the objects that they collaborate with by the container
- The act of creating these associations between application objects is referred to as **wiring**

APPLICATION CONTEXT

- An **ApplicationContext** is responsible
 - to load bean definitions
 - to wire beans together
 - to dispense beans upon request
 - ... and much more
- Many implementations of **ApplicationContext**
 - **ClassPathXmlApplicationContext**: loads a context definition from an XML file located in the classpath
 - **FileSystemXmlApplicationContext**: loads a context definition from an XML file in the file system
 - **XmlWebApplicationContext**: loads context definitions from an XML file contained within a web application

AOP: BASIC CONCEPTS

- **Aspect**: a modularization of a concern that cuts across multiple classes
- **Join point**: a point during the execution of a program (e.g. the execution of a method)
- **Advice**: action taken by an aspect at a particular join point
 - **Before**: Advice that executes before a join point
 - **After returning**: Advice to be executed after a join point completes normally
 - **After throwing**: Advice to be executed if a method exits by throwing an exception
 - **After (finally)**: Advice to be executed regardless of the means by which a join point exits
 - **Around**: Advice that surrounds a join point such as a method invocation
- **Pointcut**: a predicate that matches join points (e.g. the execution of a method with a certain name)