



SAPIENZA  
UNIVERSITÀ DI ROMA

**Ph.D. Giovanni Stilo**



# Spring is Coming

Very fast introduction and basic principles to Spring 3

26 mar 2014

# Requirements

- Polymorphism (General)
  - Inheritance
  - Overloading
  - Overriding
  - Interfaces
  
- Reflection API (Java)
  - <http://docs.oracle.com/javase/tutorial/reflect/TOC.html>
  - Extensibility Features: An application may make use of external, user-defined classes by creating instances of extensibility objects using their fully-qualified names.
  - Class Browsers and Visual Development Environments : A class browser needs to be able to enumerate the members of classes.
  - Debuggers and Test Tools Debuggers need to be able to examine private members on classes.
  
- Maven 2 ( Optional )



# IoC – Inversion of Control

- In software engineering, **inversion of control (IoC)** is a programming technique, expressed in terms of object-oriented programming, in which object coupling is bound at run time by an assembler object and is typically not known at compile time using static analysis.
  - Main benefit is that it offers configuration flexibility because alternative implementations of a given service can be used without recompiling code.
  - Another benefit of using the dependency injection approach is the reduction of boilerplate code in the application objects since all work to initialize or set up dependencies is handled by a **provider component**.
  - Furthermore, dependency injection facilitates the writing of testable code.

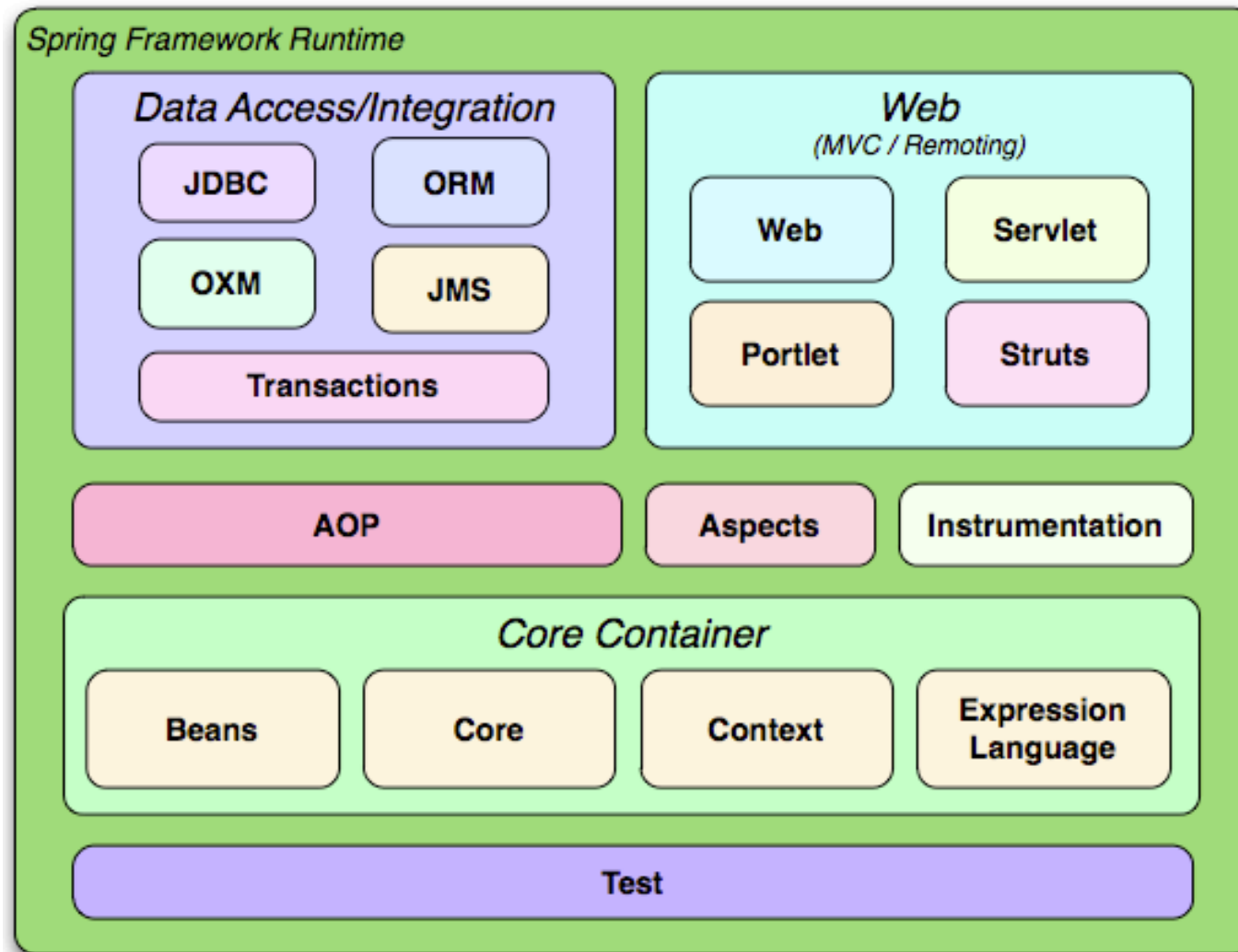


# Spring Framework

- Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application.
  - Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. (J2SE - J2EE)
- The Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test, as shown in the following diagram.



# Spring Modules

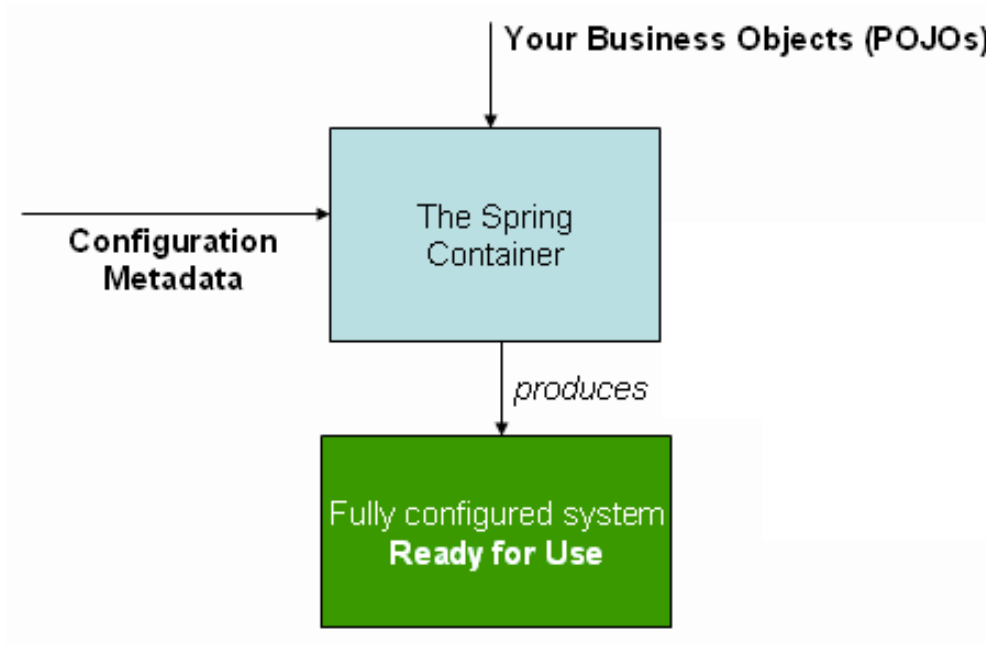


# Core Container

- The *Core and Beans* modules provide the fundamental parts of the framework, including the IoC and Dependency Injection features.
- The *Context* module builds on the solid base provided by the *Core and Beans* modules: it is a means to access objects in a framework-style manner that is similar to a JNDI registry. 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. It is an extension of the unified expression language (unified EL) as specified in the JSP 2.1 specification.



# Application Context



- Several implementations of the **ApplicationContext interface** are supplied out-of-the-box with Spring. In standalone applications it is common to create an instance of [ClassPathXmlApplicationContext](#) or [FileSystemXmlApplicationContext](#).

# XML Metadata

```
<?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 http://  
www.springframework.org/schema/beans/spring-beans-3.0.xsd">
```

```
  <bean id="myservice" class="it.example.MyServiceImpl">  
    <property name="text" value="Hello Worlds!"/>  
    <property name="refProp" >  
      <ref bean="referencedBeanId"/>  
    </property>
```

```
  </bean>
```

```
  <bean id="..." class="...">
```

```
    <!-- collaborators and configuration for this bean -->
```

```
  </bean>
```

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

```
</beans>
```

- Bean are composed of:
  - class
  - id
  - properties





# Minimal Code

## Maven:

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>3.2.2.RELEASE</version>
</dependency>
```

## Java:

```
// create and configure beans
```

```
ApplicationContext context = new FileSystemXmlApplicationContext
    (new String[] {"main.xml", "mods-one.xml"});
```

```
// retrieve configured instance
```

```
MyServiceIF service= (MyServiceIF) context.getBean("myservice");
```

```
// use configured instance how do you prefer
```

```
Object produced= service.doSomething();
```

```
System.out.println(service.getText());
```



# Excercise

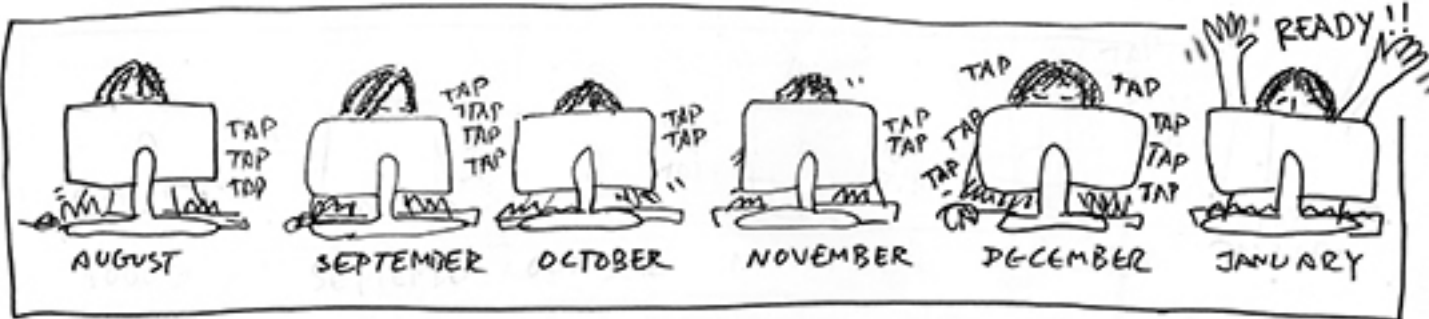
Do three Class as Bean (setter / getter):

- **Person:**
  - Name
  - Surname
  - Birthdate
  - Address
  
- **Address**
  - City
  - Street
  - Number
  
- **PersonPrinter (based on Spring)**
  - Using Spring retrieve instances and print their Information Including Address.
  
- Library also at <http://repo1.maven.org/maven2/org/springframework/spring-core/3.2.8.RELEASE/spring-core-3.2.8.RELEASE.jar>





# Let's Try?!?!



# Remarks

Maybe the example point your attention in the wrong direction. So some clarifications are needed:

- Spring is not a database engine
- Spring is not intended to manage data.
- Spring is not SOAP or any others access protocol or data related access protocol.
  
- Spring is basically a flexible factory of instances.
- Spring help to implement IoC applications.
- Spring help to write loosely coupled Classes.
- Spring help but do not do all by itself.

