
Spring
Sebas Navarrete Molina

Objetivos

Spring

- ¿Qué es Spring?
- Módulos que componen Spring
- Requisitos para trabajar con Spring

Conceptos fundamentales

- Características de IoC y DI
- Contenedor Beans
- Patron Singleton- prototype
- Estereotipos
- Patrones

Que es Spring .-

- Spring es un framework (marco de trabajo) de **desarrollo** y **contenedor de inversión de control** de código abierto.
- Fue creado para simplificar el desarrollo de aplicaciones empresariales usando POJO's para conseguir objetivos que antes sólo eran posibles con EJB.
- Es un framework **no intrusivo**, lo que genera componentes **no dependientes** del propio marco de trabajo.
- Ayuda a mejorar el **desacoplamiento** ("loose coupling") en los desarrollos modulares.

Rod Johnson
.-

Revisión histórica

- Primera versión en Junio 2003 fue lanzado bajo la licencia de Apache 2.0
- Spring 1.0 en Marzo 2004 (con la versión 1.2.6 obtuvo el reconocimiento Jolt Awards y Jax Innovation Awards en 2006).
- Spring 2.0 en Octubre 2006
- Spring 2.5 en Noviembre 2007
- Spring 3.0 en Diciembre de 2009
- Spring 4.0 en 2013
- Spring 5.1 abril de 2019
- Spring 6.0 marzo 2022

¿Por qué se utiliza Spring?

J2EE

- EJB – Dolor de cabeza

1999–2003

JEE

- EJB

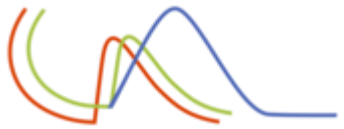
2005–2017

2004–2019

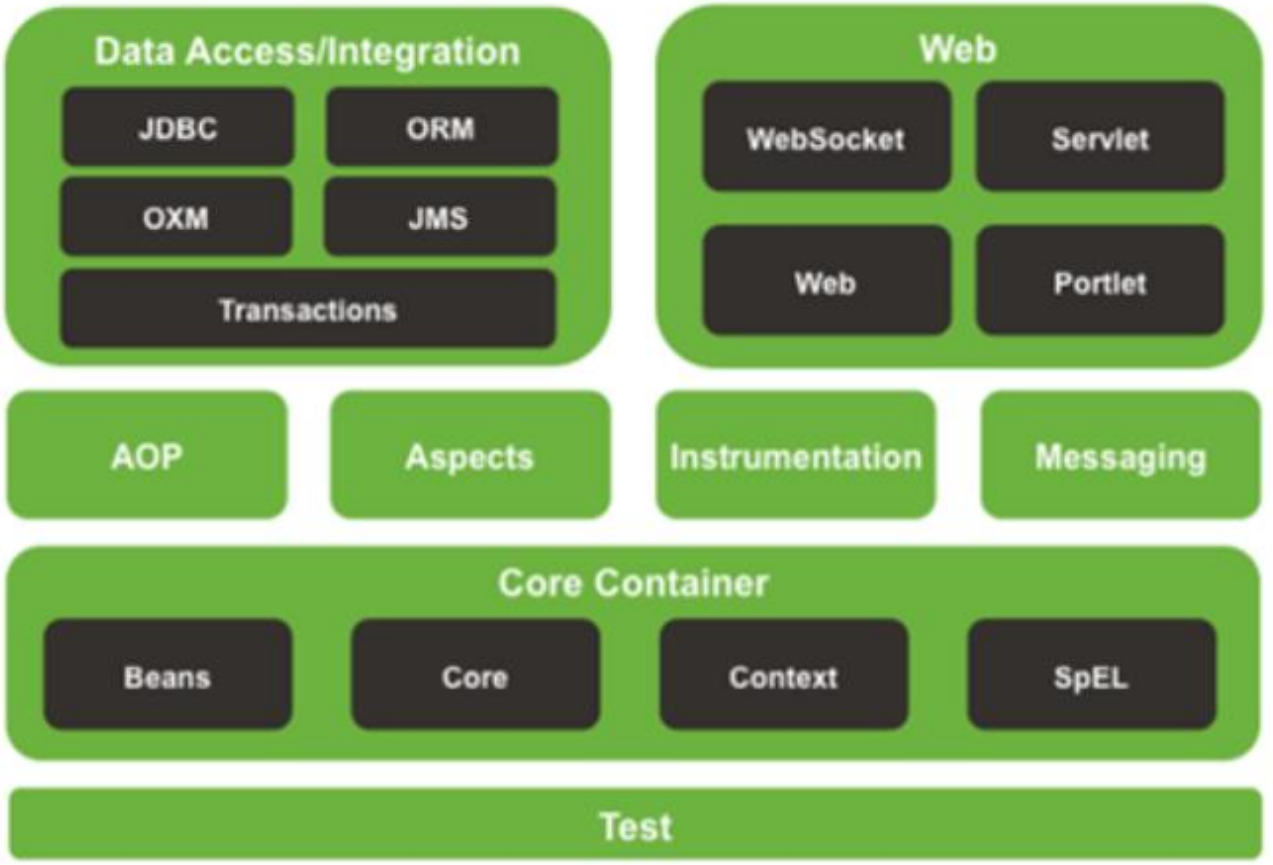
Rod Johnson

Ventajas

- Inyección de dependencias (favorece el “loose coupling”)
- Desarrollo sencillo con POJOS (Plain Old Java Objects).
- Minimiza el código repetitivo (boilerplate code).
- Simplifica el acceso a datos. JDBC
- Programación Orientada a Aspecto (AOP) favorece el desarrollo modular.

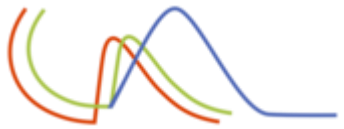


Módulos de Spring



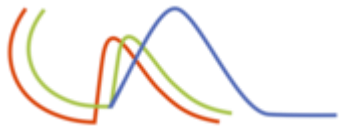
Core Container

- Modulo Core; es el núcleo de Spring
- Modulo Beans; junto al modulo Core recoge todos los fundamentos de Spring, incluyendo IoC (Inversión de control) y DI (Inyección de dependencias)
- Modulo Context; añade soporte para la internacionalización, propagación de eventos, carga de recursos
- Modulo Expression Language; lenguaje de expresiones de Spring.



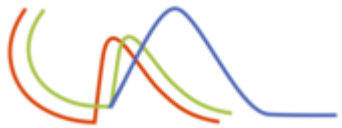
Data Access/Integration

- Modulo JDBC; Acceso a base de datos mediante JDBC
- Modulo ORM; Acceso a base de datos a través JPA, Hibernate, Ibatis,..etc.
- Modulo OXM; soporta el mapeo Objeto/XML como por ejemplo JAXB.
- Modulo JMS; Producir y consumir mensajes asíncronos.
- Modulo Transaction; Para manejar transacciones programáticas o declarativas.



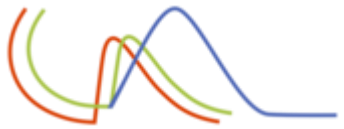
Web

- Modulo Web; Funcionalidades básicas de aplicaciones web.
- Modulo Web-Servlet; más conocido como Spring MVC
- Modulo Web-Struts; permite integrar Struts 2.0 con Spring 3.0.
- Modulo Web-Portlets; integración de Portlets.



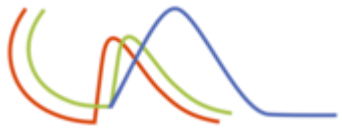
AOP and Instrumentation

- Modulo AOP; Programación Orientada a Aspectos.
- Modulo Instrumentation; implementación del cargador de clases para ser utilizado en determinados servidores de aplicaciones.



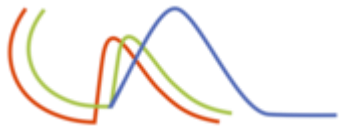
Test

- Modulo Test; permite probar componentes Spring con JUnit.



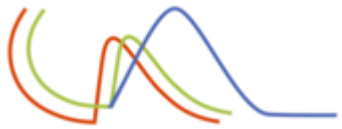
CONTENEDORES DE BEANS

- La filosofía de Spring es la reutilización de objetos (en adelante beans). Se trata de crear un objeto y almacenarlo en un contenedor para su posterior uso.
- En una aplicación desarrollada con Spring, los beans (objetos de aplicación) van a residir dentro del contenedor de beans.
- Dicho contenedor se encuentra en el núcleo (Core).
- El contenedor utiliza DI (Inyección de Dependencias) para crear los beans, conectarlos con otros, configurarlos y administrar su ciclo de vida.



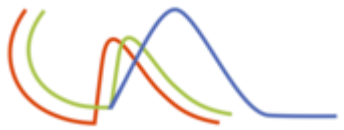
Patrones Singleton y Factory

- Singleton
  - ¿Qué es?
    - Patrón de diseño que tiene como objetivo asegurar que solo hay una instancia u objeto por clase y un punto de acceso global a ella.
  - ¿Por qué es necesario?
    - En algunos escenarios es necesario asegurarse de que las clases controlan y gestionan el acceso a un único recurso (un fichero abierto, una conexión a BBDD). Es frecuente necesitar un punto de acceso único a algún recurso del sistema compartido y un único objeto que centralice la administración del recurso.
  - Spring trabaja por defecto el patrón Singleton



AMBITOS DE UN BEAN @Scope

- A la hora de crear el bean se le puede definir un ámbito. Esto nos limitará el uso que le podemos dar. Los ámbitos definidos en Spring son:
- Singleton; Por defecto todos los beans son instancias únicas.
- Prototype; Permite que un bean se instancie tantas veces se quiera.
- Request; El bean pertenece al ámbito de petición. Solo para aplicaciones Web.
- Session; El bean pertenece al ámbito de sesión. Solo para aplicaciones Web.
- Global-session; El bean pertenece al ámbito de sesión global. Solo es válido para el contexto de un portlets.



Inversión de Control (IoC)

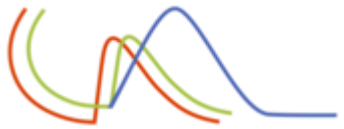
- Principio de Hollywood (Martin Fowler)

Después de la entrevista ¿Cuántas llamadas recibirán ?

Inversión de control – Nosotros le llamamos

IoC (Inversión de control)

- Invierte el flujo de control del programa, externalizando (agente externo: framework) la construcción y manejo de objetos.
- Ventajas
  - Proporciona modularidad
  - Permite ampliar la funcionalidad de nuestras aplicaciones sin modificar las clases
  - Evita la dependencia entre las clases
  - Flexibiliza nuestras aplicaciones haciéndolas más configurables.



Dependency Injection (DI)

- Patrón de diseño orientado a objetos en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos

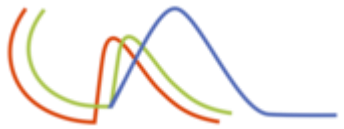
Formas de crear un bean .-

- Podemos usar diferentes alternativas a la hora de crear un bean :

- A través de constructor
- Mediante una factoría estática
- Utilizando una factoría de instancia

Java annotations

- ¿Qué son las java annotations?
  - Son etiquetas que se añaden a las clases, ya se a métodos, campos, variables en un programa Java.
- ¿Para qué sirven ?
  - Añadir metadatos a nuestros programas (clases) de Java
  - Metadatos : conjunto de datos que describen el contenido y/o propósito de un objeto
  - Las annotations son procesada en tiempo de compilación y en tiempo de ejecución.
  - Se escanea en segundo plano las anotaciones. Cuando escanea una anotación ya registra el bean



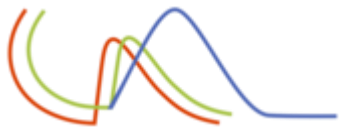
¿Qué se tiene que hacer?

1. ~~Preparar el XML para que Spring busque en el código (clases)~~
2. Agregar annotations a nuestras clases de Java
3. Pedir el bean al contenedor

```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
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd">

<context:component-scan base-package="modelo"></context:component-scan>
```

```java
@Component("comercial")
public class Comercial implements Empleado {
```



Estereotipos

1. Todas las clases que estén anotadas con:
   - @Component; componente de Spring
   - @Controler; controlador MVC
   - @Repository; repositorio de datos
   - @Service; servicio
   - @Configuration; clase que define beans

- Pueden registarse de forma automática como beans.

@Autowired .-

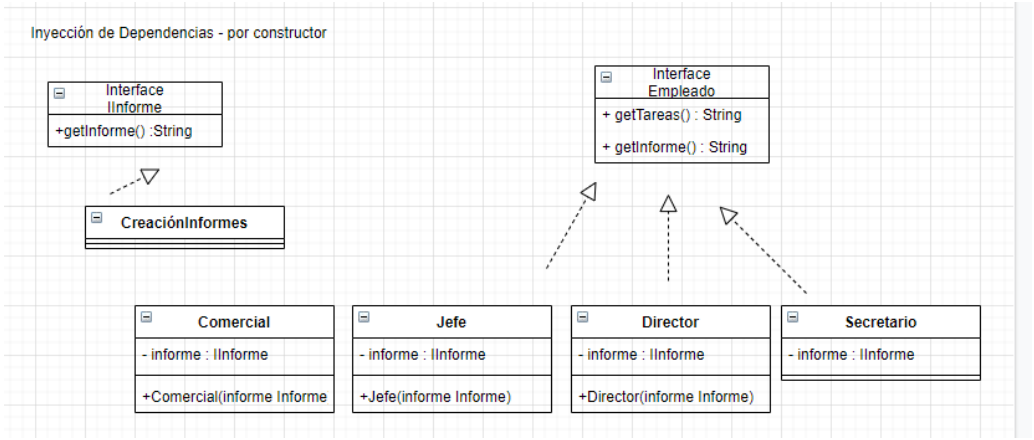
- En el siguiente método Spring intenta la conexión automática por tipo, esto es, si encuentra un bean de tipo Instrument lo conecta.

```
@Autowired
public Instrumentalist(Instrument instrument) {
  this.instrument = instrument;
}
```

- Podemos utilizar esta anotación en constructores, propiedades y cualquier método aunque sean privados.

- Si no existe un bean que poder conectar o hay varios, Spring genera NoSuchBeanDefinitionException.

Anotación @Autowired



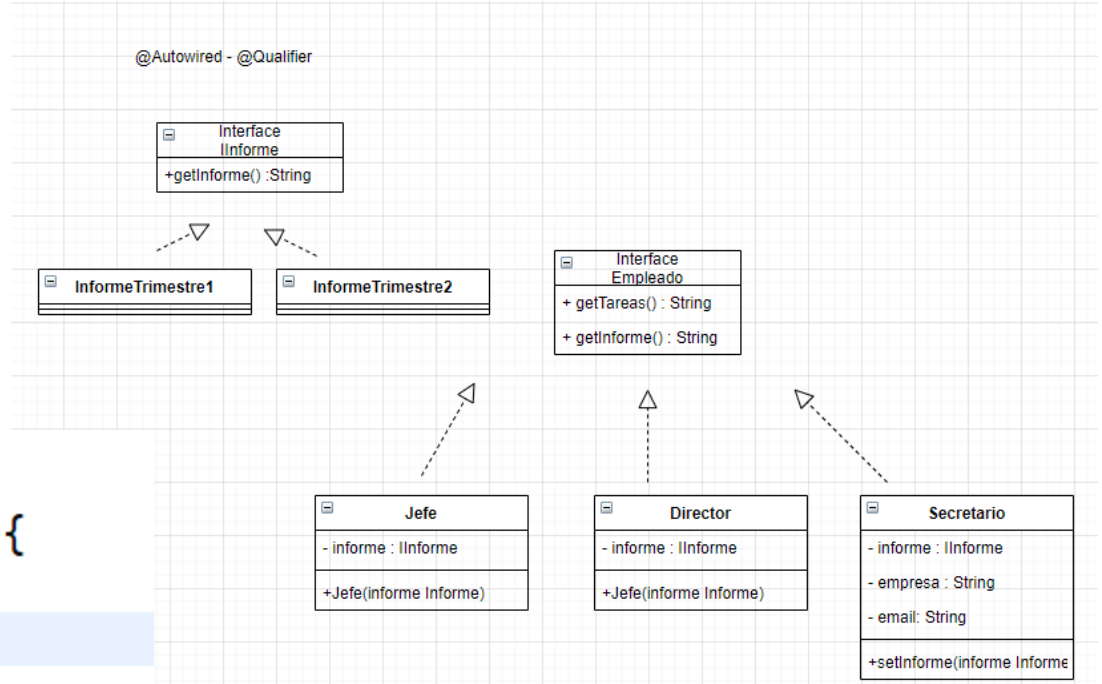

```java
@Component
public class Informe implements IInforme {
```

```java
@Autowired
public Comercial(IInforme iinforme) {
    this.iinforme = iinforme;
}
```

Anotación @Qualifier

- Cuando hay un tipo que puede inyectar una dependencia y hay varias clases (opciones)
- Se pone enseguida del Autowired

```
@Component("comercial")
public class Comercial implements Empleado {
    @Autowired
    @Qualifier("informeTrimestre2")
    private IInforme iinforme;
```



Anotación @Primary

- Cuando hay un tipo que puede inyectar una dependencia y hay varias clases (opciones)
- En el vean se puede pone @Primary

```java
package com.softtek.ej2.persistencia.impl;

import org.springframework.context.annotation.Primary;

@Component("P")
@Primary
public class AccesoProduccion implements IDAO {

    @Override
    public String insertar(Cliente c) {
        // TODO Auto-generated method stub
        return "Desde produccion " + c.toString();
    }

}
```