everis
an NTT DATA Company

BECA Java
Verano 2017

everis
an NTT DATA Company

Agenda

- Spring
  - Introducción
  - Arquitectura
- Spring Boot
  - Introducción
  - Como empezar
  - Pasos con Spring Boot

everis
an NTT DATA Company

01 Spring

everis
an NTT DATA Company

Introducción

- Spring es el framework Java utilizado por excelencia para el desarrollo de aplicaciones empresariales de manera ***simplificada***.
- Un *framework* es un conjunto de clases que nos permiten resolver un problema en específico.
- Una de las mayores ventajas de Spring, es la forma ***modular*** en el que fue creado, permitiendo ***habilitar / deshabilita****r* las características a utilizar según se requiera.
- Spring es utilizado en proyectos muy diversos, como puede ser en Instituciones Bancarias, Aseguradoras, Instituciones Educativas y de Gobierno, entre muchos otros tipos de proyectos y empresas.

everis
an NTT DATA Company

Introducción

- Spring permite desarrollar aplicaciones *flexibles*, altamente *cohesivas* y con un bajo *acoplamiento.*
- Promueve el uso de clases Java Simples (POJO – Plain Old Java Object) para la programación orientada a interfaces y la configuración de servicios (Manejo de Transacciones, Manejo de Excepciones, Parametrización de la aplicación).

everis
an NTT DATA Company

Características Principales

- **DI** (Dependency Inyection): Este patrón de diseño permite suministrar objetos a una clase (POJO) que tiene dependencias, en lugar de ser ella misma quien los proporcione.
- **AOP** (Aspect Oriented Programming): AOP es un paradigma de programación que permite modularizar las aplicaciones y mejorar la separación de responsabilidades entre módulos y/o clases.

everis
an NTT DATA Company

Modulos

everis
an NTT DATA Company

Beans

- Un **Bean** en Spring no es mas que un objeto configurado e instanciado en el ***contenedor de Spring*** usado entre otras cosas para la ***inyección de dependencias***.
- Todos los *beans* permanecen en el contenedor durante toda la vida de la aplicación o hasta que nosotros los destruyamos.
- Tener los beans en el contenedor nos permite ***inyectarlos*** en otros beans, ***reutilizarlos***, o poder ***acceder a ellos*** desde cualquier lugar de la aplicación en el momento que queramos.

everis
an NTT DATA Company

BeanFactory

- **FactoryBean** es un patrón usado para encapsular la ***lógica de construcción*** de objetos en una clase.
- Utilizable para codificar la ***construcción de objetos complejos*** de manera reutilizable.
- Cada bean tiene un **identificador** para poder obtenerlo desde la BeanFactory.

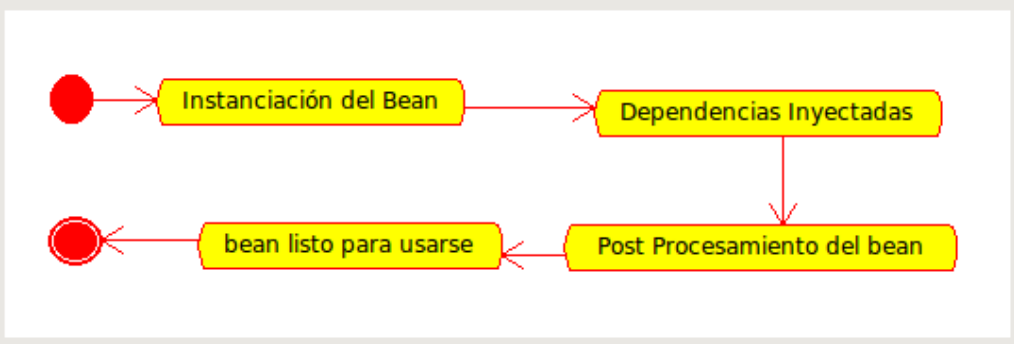
```
<!-- use the DataSource exposed by JNDI -->
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiName" value="java:comp/env/jdbc/mydb"/>
</bean>
<bean id="myDAO" class="com.dao.MyDAO">
    <property name="dataSource" ref="dataSource"/>
</bean>
```

everis
an NTT DATA Company

ApplicationContext

La fase de inicialización esta completa cuando se crea el **Contexto**:

- Parsean archivos XML de configuración.
- Las definiciones de los beans son cargados en el contexto del BeanFactory.
- Se invocan a clases y métodos especiales que nos permiten manipular y transformar grupos de definiciones de beans antes que los objetos sean creados (BeanFactoryPostProcessor como PropertyPlaceholderConfigurer, CustomScopeConfigurer, AspectJWeavingEnabler...).



everis
an NTT DATA Company

Configuración con Anotaciones

A partir de la versión 2 de Spring podemos configurar los beans con Anotaciones. Aunque no definamos los Beans con XML, este fichero siempre tiene que estar presente.

Beans con anotaciones:

- @Service: Para definir las componentes de negocio.
- @Repository: Para definir los DAO.
- @Component: Para componentes mas especificas.

```
package es.everis.spring.negocio;

@Service("miGestor")
public class GestorUsuarios {
  public UsuarioTO login(String login, String password) {...}
}
```

```
<beans>
   <context:component-scan base-package="es.everis.spring"/>
</beans>
```

everis
an NTT DATA Company

Configuración con Anotaciones

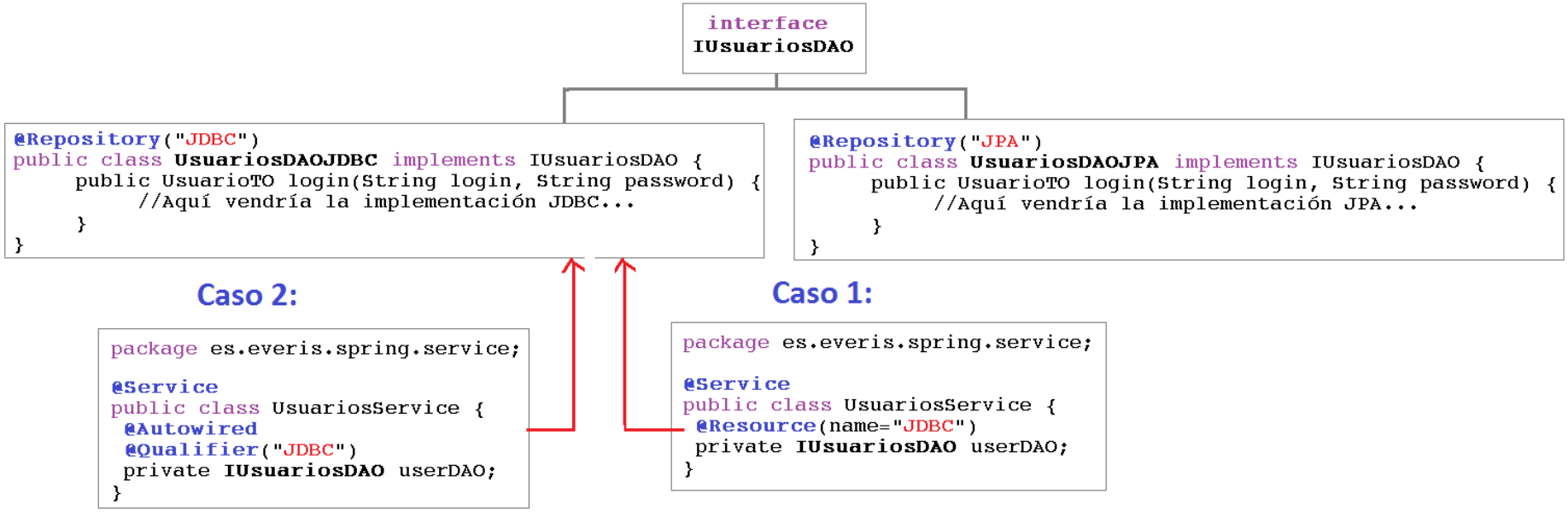
Para acceder a un Bean desde otro Bean se usa la anotación @Autowired.

```
package es.everis.spring.service;

@Service
public class UsuariosServiceImpl implements IUsuariosService{
   @Autowired
   private IUsuariosDAO userDAO;
}
package es.everis.spring.dao;

@Repository
public class UsuariosDAOImpl implements IUsuariosDAO{
   public Usuario getUsuario(String id) {
}
```

Configuración con Anotaciones

@Autowired se puede usar solo si existe una única implementación de un Bean. En caso de múltiples implementaciones de un Bean se puede inyectar de dos maneras:

1. Identificándolo por nombre en la anotación.



everis
an NTT DATA Company

Configuración con Anotaciones

Una vez definidas nuestras clases nuestro applicationContext.xml quedaría así:

```
<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.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:annotation-config />  OR
    <context:component-scan base-package="beans" />
</beans>
```

<context:annotation-config> se usa para activar las anotaciones en los beans registrados dentro del contexto de Spring.

<context:component-scan> hace lo mismo de annotation-config pero escanea los beans al interno del paquete definido para buscar y registrar beans al interno de la aplicación.

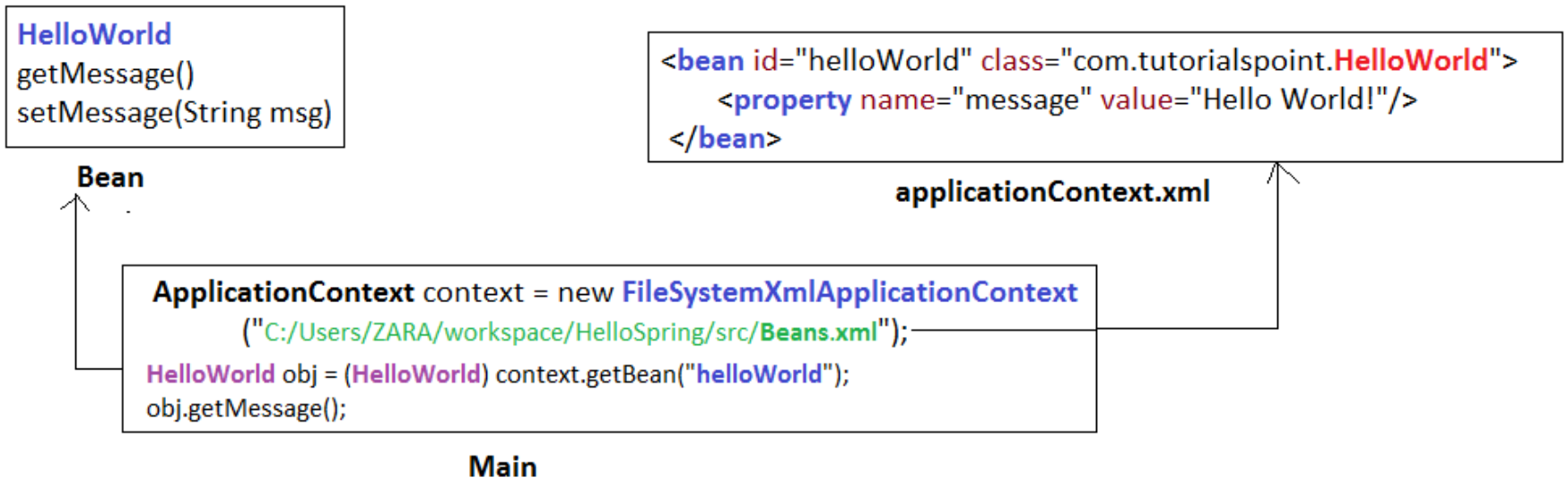
Configuración con Anotaciones

Las 3 maneras mas usadas para cargar el contexto de Spring son:

- FileSystemXmlApplicationContext: Este contenedor carga las definiciones de los beans desde un archivo XML proporcionando la ruta completa.
- ClassPathXmlApplicationContext: Este contenedor también carga las definiciones de los beans desde un archivo XML que debe estar presente en el CLASSPATH.
- WebXmlApplicationContext: Este contenedor carga las definiciones de los beans desde el web application.

everis
an NTT DATA Company

Configuración con Anotaciones



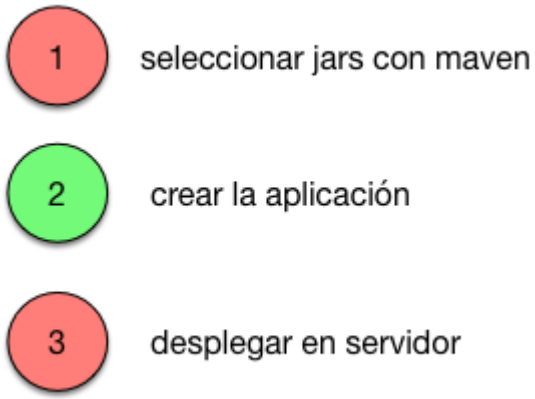
everis
an NTT DATA Company

02 Spring Boot

everis
an NTT DATA Company

Introducción

- Es un nuevo módulo de la plataforma Spring cuyo objetivo es simplificar la creación de aplicaciones y servicios listos para ejecutarse. Si pensamos cómo desarrollamos aplicaciones con spring…



- Spring boot permite resolver los problemas 1 y 3

everis
an NTT DATA Company

Porque usar Spring Boot

- Crea aplicaciones stand-alone de spring
- Los entornos locales
  - Permite embeber muchas otras tecnologías (H2, Mongo, LDAP Server…)
- Tests de integración
- Despliegue en los distintos entornos
- Facilita los entornos de CI
- La gestión de propiedades application.properties/application.yml
- Plugins maven/gradle para el arranque y empaquetado
- Configuración automatizada de spring
- Podemos olvidarnos de la configuración XML
- Gestión de profiles

everis
an NTT DATA Company

Como empezar con Spring Boot

- Opción 1:
  - https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-installing-the-cli
  - Podemos usar SDKMan http://sdkman.io / Windows: https://github.com/flofreud/posh-gvm
  - Creamos nuestro aplicación básica
- Opción 2: http://start.spring.io

Pasos con Spring Boot: Añadir la dependencia

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Pasos con Spring Boot: Creamos la clase con el método main

```java
package demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan
@EnableAutoConfiguration
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Pasos con Spring Boot: Agregamos un Controller

```
@RestController
class GreetingController {

    @RequestMapping("/hello/{name}")
    String hello(@PathVariable String name) {
        return "Hello, " + name + "!";
    }
}
```

Pasos con Spring Boot: Versión de Tomcat (servidor por defecto)

```
<properties>
  <tomcat.version>8.0.3</tomcat.version>
</properties>
```