Programación Distribuida

Spring Framework - Beans

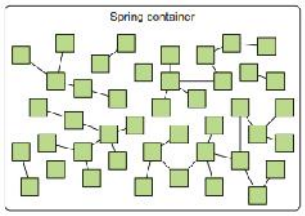
JAIME SALVADOR

AGENDA

- **Configuración**
- Web
- Ejemplo

Configuración

- En Spring, los objetos no son responsables de crear o buscar dependencias (otros objetos)
- El contenedor es el encargado de crear y proporcionar "objetos"
- *Wiring:* creación de asociaciones entre objetos (DI)



Configuración

- Configuración explícita en XML
- Configuración explícita en Java
- *Wiring* automático

No son excluyentes

Configuración

***Wiring* automático**

El soporte para wiring automático está dado por:

- Component scanning: Spring configure automáticamente los components y los asocial a un context
- Autowiring: Spring se encarga de proporcionar las dependencias de forma automática

Configuración

***Wiring* automático**

- `@Component`
- `<context:component-scan base-package="nombre.paquete" />`

Configuración

Wiring automático

```
package com.distribuida.ejemplo01.servicios;

public interface ServicioOperaciones {

    public int sumar( int n1, int n2 );
}
```

```
package com.distribuida.ejemplo01.servicios;

import org.springframework.stereotype.Component;

@Component(value="servicio01")
public class ServicioOperacionesImpl implements ServicioOperaciones {

    public int sumar( int n1, int n2 ) {
        return n1 + n2;
    }
}
```

Configuración

Wiring automático: test

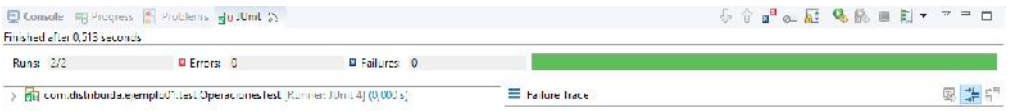
- `junit-4.12.jar`
- `hamcrest-core-1.3.jar`
- `spring-test-5.0.5.RELEASE.jar`

```
package com.distribuida.ejemplo01.test;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.distribuida.ejemplo01.servicios.ServicioOperaciones;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:app.xml")
public class OperacionesTest {

    @Autowired private ServicioOperaciones servicio;

    @Test
    public void testServicioNull( ) {
        assertNotNull( servicio );
    }
}
```

Configuración

***Wiring* automático: test**

- `junit-4.12.jar`
- `hamcrest-core-1.3.jar`
- `spring-test-5.0.5.RELEASE.jar`



Configuración

***Wiring* automático**

- `@Autowired`

```
package com.distribuida.ejemplo01.servicios;

public interface LogService {
    public void log( String msg );
}
```

```
package com.distribuida.ejemplo01.servicios;

import org.springframework.stereotype.Component;

@Component(value="logservice")
public class LogServiceImpl implements LogService {

    public void log( String msg ) {

        System.out.println( "[LOG] - " + msg );
    }
}
```

Configuración

***Wiring* automático**

- `@Autowired`

```
@Component(value="servicioOp")
public class ServicioOperacionesImpl implements ServicioOperaciones {

    @Autowired private LogService servicioLog;
```

Configuración

***Wiring* automático**

- `@Component` → `@Stateless`
- `@Autowired` → `@EJB`

Configuración

***Wiring* Java**

- Spring permite configurar components proporcionados por terceros (en muchos casos proporcionados como archivos JAR)
- Es possible anotarlos con **@Component**?

Configuración

***Wiring* Java**

- Spring permite configurar components proporcionados por terceros (en muchos casos proporcionados como archivos JAR)
- Es possible anotarlos con **@Component**? **NO**

Configuración

***Wiring* Java**

- Es necesario una configuración explícita:
  - Java: JavaConfig
  - XML: XML Config

Configuración

***Wiring* JavaConfig**

- Type safe
- Refactor-friendly
- Opción más utilizada

Configuración

***Wiring* JavaConfig**

- `@Configuration`
- Refactor-friendly
- No es necesario anotar los components con `@Component`
- Opción más utilizada

Configuración

***Wiring* JavaConfig**

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

    <context:component-scan base-package="com.distribuida.ejemplo01.servicios"/>

</beans>
```

Configuración

***Wiring* JavaConfig**

```
package com.distribuida.ejemplo02.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.distribuida.ejemplo02.servicios.LogService;
import com.distribuida.ejemplo02.servicios.LogServiceImpl;
import com.distribuida.ejemplo02.servicios.ServicioOperaciones;
import com.distribuida.ejemplo02.servicios.servicioOperacionesImpl;

@Configuration
public class Ejemplo02Config {

    @Bean
    public ServicioOperaciones servicioOp( LogService servicioLog ) {

        ServicioOperacionesImpl servicio = new ServicioOperacionesImpl( );

        servicio.setServicioLog( servicioLog );

        return servicio;
    }

    @Bean
    public LogService servicioLog( ) {
        return new LogServiceImpl( );
    }
}
```

Configuración

***Wiring* XML**

- Primera forma para configurar un contenedor Spring

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


</beans>
```

Configuración

***Wiring* XML**

- Ejemplo (presentación anterior)

```
<?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:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.1.xsd">

    <aop:aspectj-autoproxy />

    <bean name="servicioOperaciones" class="spring.ejemplo01.servicios.OperacionesImpl">
    </bean>

    <bean name="logAspect" class="spring.ejemplo01.aspectos.LogAspect">
    </bean>
</beans>
```

AGENDA

- Configuración
- **Web**
- Ejemplo

Web

- Cómo configurar un contenedor Spring en una aplicación web?

Servlet Listener + XML Config

Web

Servlet Listener

- http://docs.oracle.com/cd/B14099_19/web.1012/b14017/filters.htm#i1000654
- Spring incluye un *Servlet Listener* encarcado de inicializar un contenedor
- `spring-web-5.0.5.RELEASE.jar`

`web.xml`

```
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Web

Context Param

- `ContextLoaderListener` necesita de la ruta al archivo XML de configuración
- El Servelt Listener crea un contenedor utilizando el archive de configuración dado por *contextConfigLocation*

`web.xml`

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:app.xml</param-value>
</context-param>
```

Web

- Cómo configurar un contenedor Spring en una aplicación web?

Servlet Listener + JavaConfig

Web

Servlet Listener

- http://docs.oracle.com/cd/B14099_19/web.1012/b14017/filters.htm#i1000654
- Spring incluye un *Servlet Listener* encarcado de inicializar un contenedor

`web.xml`

```
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Web

Context Param

- `ContextLoaderListener` necesita dos parámetros de configuración
- *contextClass*
- *contextConfigLocation*

`web.xml`

```
<context-param>
    <param-name>contextClass</param-name>
    <param-value>org.springframework.web.context.support.AnnotationConfigWebApplicationContext</param-value>
</context-param>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>com.distribuida.ejemplo02.config.Ejemplo02Config</param-value>
</context-param>
```

Web

Servlet 3.0+

- `WebApplicationInitializer`

```
public void onStartup(ServletContext container) throws ServletException {
    XmlWebApplicationContext appContext = new XmlWebApplicationContext( );
    appContext.setConfigLocation("classpath:app.xml");
    ContextLoaderListener listener = new ContextLoaderListener( appContext );
    container.addListener( listener );
}
```

Web

Servlet 3.0+

- `WebApplicationInitializer`

```
package com.distribuida.ejemplo02.config.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;

import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

import com.distribuida.ejemplo02.config.Ejemplo02Config;

public class MyWebAppInitializer implements WebApplicationInitializer {

    public void onStartup(ServletContext container) throws ServletException {

        AnnotationConfigWebApplicationContext appContext = new AnnotationConfigWebApplicationContext( );

        appContext.register( Ejemplo02Config.class );

        ContextLoaderListener listener = new ContextLoaderListener( appContext );

        container.addListener( listener );
    }
]
```

PREGUNTAS?

Bibliografía

- Craig Walls. Spring in Action, Fourth Edition. Manning, 2015.
- Luliana Comina et al. Pro Spring 5. Apress, 2018.
- https://docs.spring.io/spring/docs/5.0.5.RELEASE/spring-framework-reference/ consultada el 2018-04-15,