

Programación Distribuida

 Spring Framework

JAIME SALVADOR



AGENDA

- **Requisitos**
- Introducción
- Escenarios
- Contenedor
- Ejemplo

Requisitos

- JDK

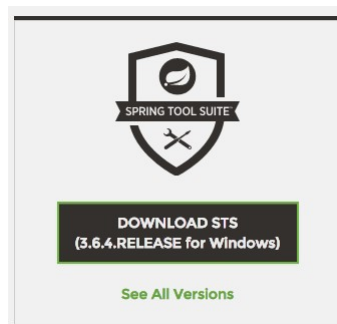
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



Requisitos

- Spring STS




<https://spring.io/tools/sts>



Requisitos

- Spring 4.1.6

<http://projects.spring.io/spring-framework/>

Spring Framework	
RELEASE	DOCUMENTATION
4.2.0 <small>SNAPSHOT</small>	Reference API
4.1.6 <small>CURRENT</small> 	Reference API
4.0.9 	Reference API
3.2.13 	Reference API

AGENDA

- Requisitos
- **Introducción**
- Escenarios
- Contenedor
- Ejemplo

Introducción

QUE ES SPRING?

Introducción

- Spring es un framework lightweight para la construcción de aplicaciones Java.
- "The Spring Framework is a lightweight solution and a potential one-stop-shop for building your enterprise-ready applications."
- Spring fue creado como una alternative a la tecnología J2EE, específicamente la tecnología EJB
- *Lightweight: ligero, liviano*

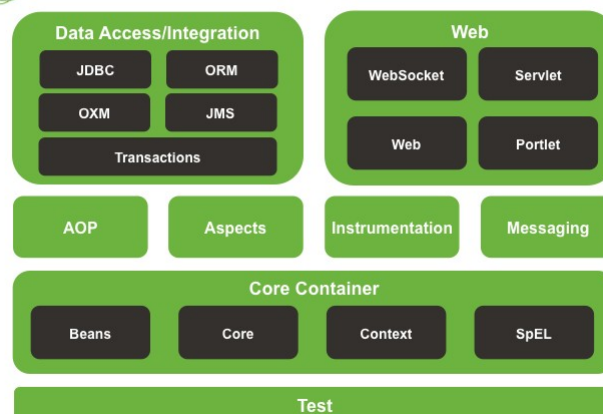
Introducción

Spring = DI + AOP

Introducción

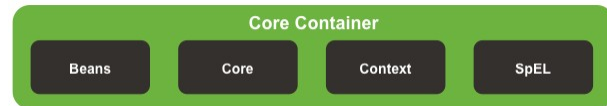


Spring Framework Runtime



Introducción

Core Container



- Core, Beans: Proporciona la parte fundamental de Spring: IoC y DI
- Factory Pattern
- http://es.wikipedia.org/wiki/Factory_Method_%28patr%C3%B3n_de_dise%C3%B1o%29
- Context: Similar a JNDI
- SpEL: Expression Language para la manipulación de un grafo de objetos

Introducción

AOP



- Proporciona una implementación de Programación Orientada a Aspectos
- Permite definir interceptores
- @Anotaciones
- Aspects: Integración con AspectJ
- Instrumentation: soporte para **classloading**

Introducción

Messaging

Messaging

- Abstracción para Spring Integration
- Aplicaciones basadas en mensajes

Introducción

Data Access

Data Access/Integration

JDBC

ORM

OXM

JMS

Transactions

- Abstracción para el acceso a datos
- Transacciones declarativas
- Mapeo Object-XML
- JMS

Introducción

Web

- Soporte para pruebas unitarias y de integración
- Junit
- TestNG



Introducción

Test

- Abstracción para Spring Integration
- Aplicaciones basadas en mensajes



Introducción

Módulos

GroupId	ArtifactId	Description
org.springframework	spring-aop	Proxy-based AOP support
org.springframework	spring-aspects	AspectJ based aspects
org.springframework	spring-beans	Beans support, including Groovy
org.springframework	spring-context	Application context runtime, including scheduling and remoting abstractions
org.springframework	spring-context-support	Support classes for integrating common third-party libraries into a Spring application context
org.springframework	spring-core	Core utilities, used by many other Spring modules
org.springframework	spring-expression	Spring Expression Language (SpEL)
org.springframework	spring-instrument	Instrumentation agent for JVM bootstrapping

Introducción

Módulos

org.springframework	spring-instrument-tomcat	Instrumentation agent for Tomcat
org.springframework	spring-jdbc	JDBC support package, including DataSource setup and JDBC access support
org.springframework	spring-jms	JMS support package, including helper classes to send and receive JMS messages
org.springframework	spring-messaging	Support for messaging architectures and protocols
org.springframework	spring-orm	Object/Relational Mapping, including JPA and Hibernate support
org.springframework	spring-oxm	Object/XML Mapping
org.springframework	spring-test	Support for unit testing and integration testing Spring components
org.springframework	spring-tx	Transaction infrastructure, including DAO support and JCA integration

Introducción

Módulos

org.springframework	spring-web	Web support packages, including client and web remoting
org.springframework	spring-webmvc	REST Web Services and model-view-controller implementation for web applications
org.springframework	spring-webmvc-portlet	MVC implementation to be used in a Portlet environment
org.springframework	spring-websocket	WebSocket and SockJS implementations, including STOMP support

Introducción

Módulos

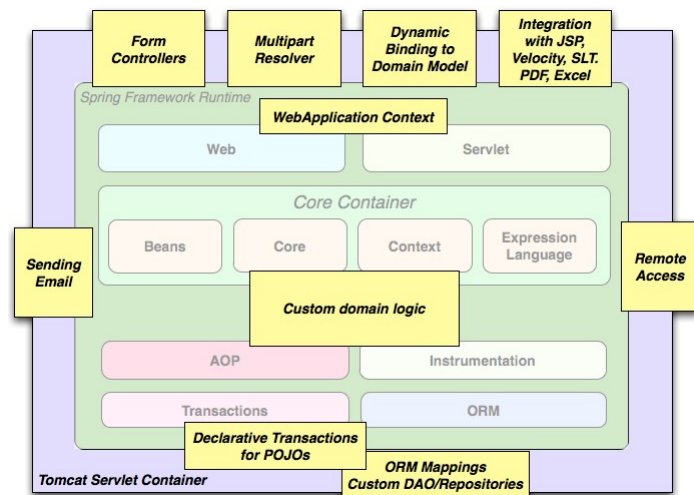


AGENDA

- Requisitos
- Introducción
- **Escenarios**
- Contenedor
- Ejemplo

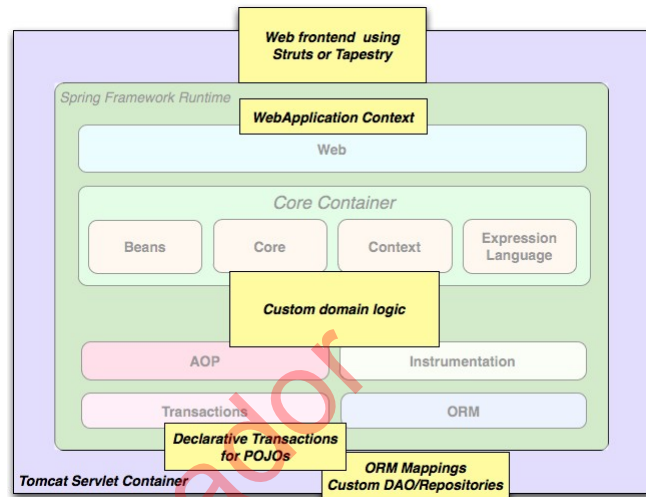
Escenarios

Aplicación web completa



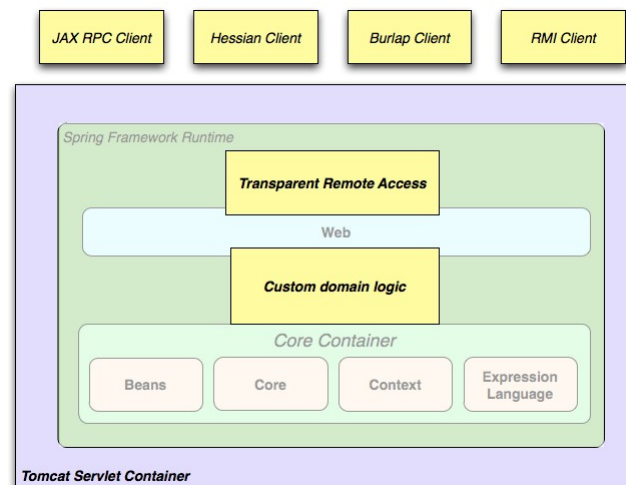
Escenarios

Spring: Capa de negocio
+
Framework web



Escenarios

Cientes remotos



AGENDA

- Requisitos
- Introducción
- Escenarios
- **Contenedor**
- Ejemplo

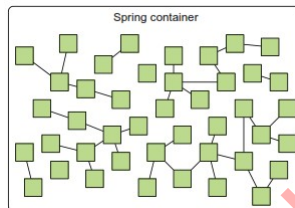
Contenedor

- En una aplicación SPRING los objetos viven en?

Contenedor

- En una aplicación SPRING los objetos viven en?

CONTENEDOR



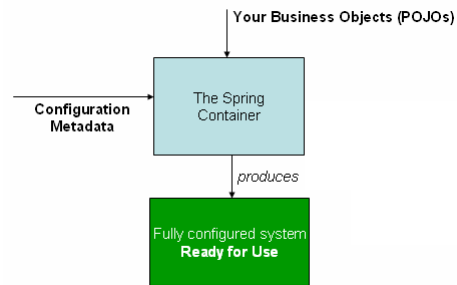
Contenedor

El contenedor:

- Crea objetos
- Los conecta
- Los configura
- Maneja el ciclo de vida

Contenedor

El contenedor:



Contenedor

Tipos de contenedores:

- *BeanFactory*: soporte básico para DI
- *ApplicationContext*: proporciona servicios adicionales

Contenedor

Tipos de contenedores: *ApplicationContext*

- `AnnotationConfigApplicationContext`
- `AnnotationConfigWebApplicationContext`
- **`ClassPathXmlApplicationContext`**
- **`FileSystemXmlApplicationContext`**
- `XmlWebApplicationContext`

Contenedor

`ClassPathXmlApplicationContext`

```
ApplicationContext context = new  
    ClassPathXmlApplicationContext( "app.xml" );
```


Contenedor

ClassPathXmlApplicationContext

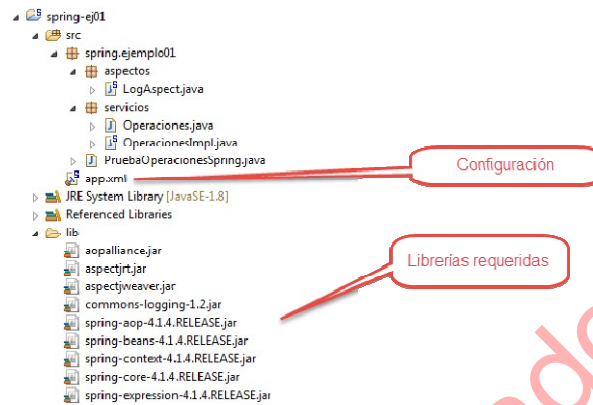
```
ApplicationContext context = new  
    FileSystemXmlApplicationContext( "c:/app.xml" );
```

AGENDA

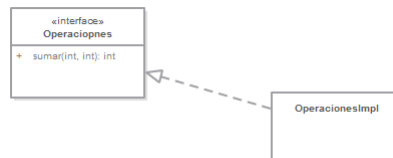
- Requisitos
- Introducción
- Escenarios
- Contenedor
- Ejemplo

Ejemplo

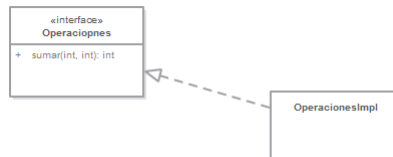
- Estructura del proyecto



Ejemplo



Ejemplo



```
import org.aspectj.lang.JoinPoint;
```

```
@Aspect
public class LogAspect {
    @After("execution(* spring.ejemplo01.servicios.OperacionesImpl.*(..))")
    public void logAfter(JoinPoint joinPoint) {

        long t1 = System.currentTimeMillis();

        System.out.println("***Ejecutando método : " + joinPoint.getSignature().getName());

        long t2 = System.currentTimeMillis();

        System.out.println( "**Tiempo: " + (t2-t1) + " ms" );

    }
}
```

```
public interface Operaciones {

    public int sumar( int n1, int n2 );

}

public class OperacionesImpl implements Operaciones {

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

}
```

Ejemplo

app.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: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>
```

Ejemplo

app.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: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>
```

```
public class OperacionesImpl implements Operaciones {
    public int sumar( int n1, int n2 )
    {
        return n1 + n2;
    }
}
```

Ejemplo

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import spring.ejemplo01.servicios.Operaciones;

public class PruebaOperacionesSpring {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext( "app.xml" );
        Operaciones servicio = context.getBean( "servicioOperaciones", Operaciones.class );

        int respuesta = servicio.sumar( 7, 5 );

        System.out.println( "Respuesta: " + respuesta );
    }
}
```

Ejemplo

```
may 17, 2015 8:18:26 PM org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh  
INFORMACIÓN: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@117ae12: startup date [Sun May 17 20:18:26  
may 17, 2015 8:18:26 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  
INFORMACIÓN: Loading XML bean definitions from class path resource [app.xml]  
**Ejecutando método : sumar  
**Tiempo: 2 ms  
Respuesta: 12
```

PREGUNTAS?



Bibliografía

- Craig Walls. Spring in Action, Fourth Edition. Manning, 2015.
- Chros Schaefer, Clarence Ho, and Rob Harrop. Pro Spring. Apress, 2014.
- <http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/>, consultada el 2015-05-17,

Jaime Salvador