Spring Framework Core

Diego Armando Gómez M
dgomez@vortexbird.com

2019

VORTEXBIRD
Information Technology becomes easy

Temas

- Que es Spring
- Arquitectura de Spring
- Principales módulos de Spring
  - Inyección de dependencias
  - Programación orientada aspectos.
- Configurando aplicación usando Spring
  - Instalar el contenedor de Inversión de Control de Spring (IoC)
  - Crear y configurar un Bean en el contenedor de Inversión de Control .
  - Usando el Auto-Writing Bean con XML
  - Usando el Auto-Writing Bean con Anotaciones.

VORTEXBIRD
Information Technology becomes easy

Que es Spring?

- Un **framework** contenedor liviano basado en la técnica **Inversión de Control (IoC)** y una implementación de desarrollo según el paradigma **Orientado a Aspectos (AOP)**
- Es el mas usado en el desarrollo de aplicaciones de Back End en Java.

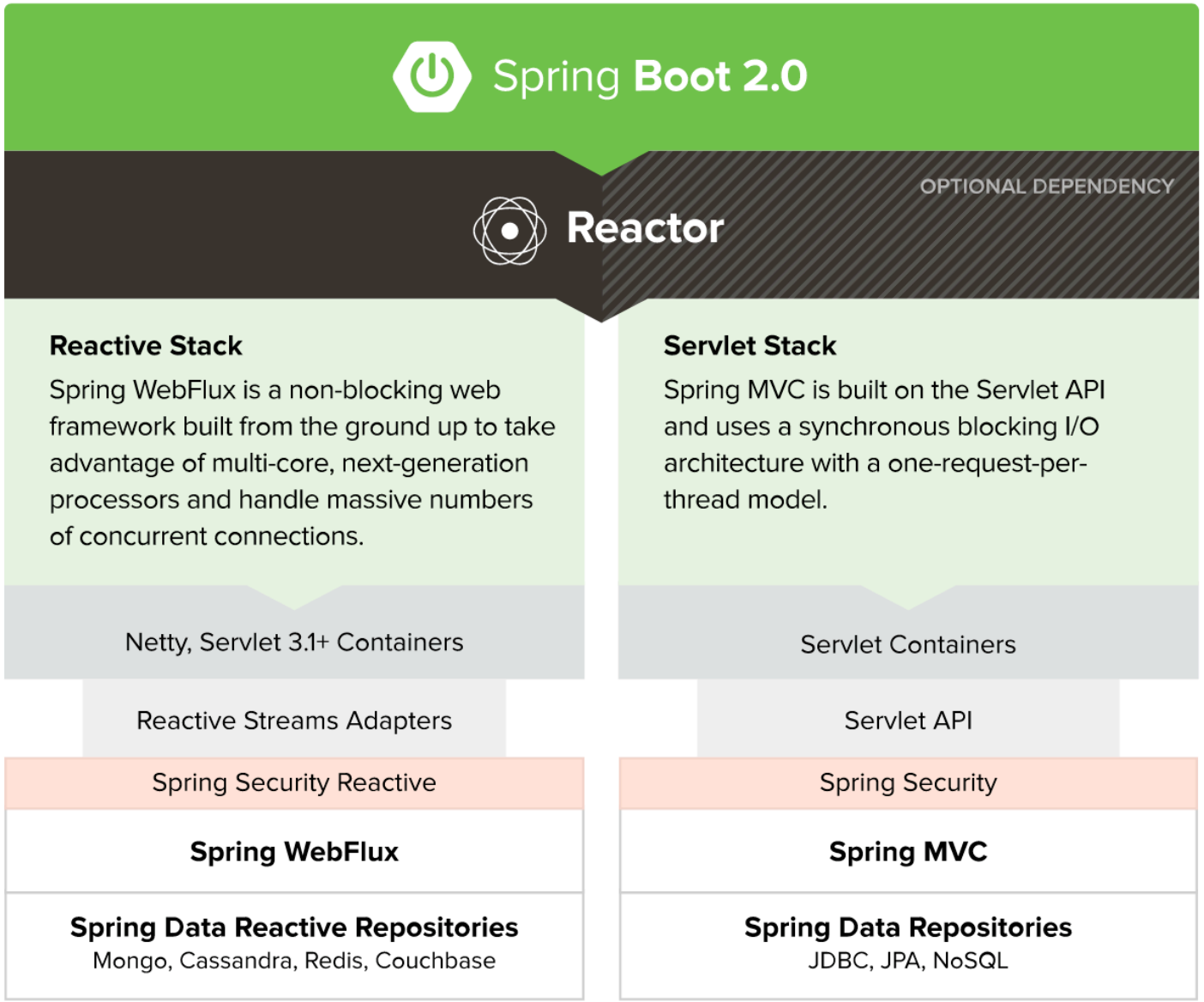
VORTEXBIRD
Information Technology becomes easy

Que es Spring?

- Framework: porque define la forma de desarrollar aplicaciones JavaEE, dando soporte y simplificando complejidad propia del software empresarial.
- Inversión de Control (IoC): promueve el bajo acoplamiento a partir de la inyección de dependencias (DI) entre los objetos (relaciones).
- Orientación a Aspectos (AOP): presenta una estructura simplificada para el desarrollo y utilización de aspectos (módulos multiple object crosscutting).

VORTEXBIRD
Information Technology becomes easy

Arquitectura de Spring



VORTEXBIRD
Information Technology becomes easy

Spring Framework Artifacts

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
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/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
org.springframework	spring-web	Foundational web support, including web client and web-based remoting
org.springframework	spring-webmvc	HTTP-based Model-View-Controller and REST endpoints for Servlet stacks
org.springframework	spring-webmvc-portlet	MVC implementation to be used in a Portlet environment
org.springframework	spring-websocket	WebSocket and SockJS infrastructure, including STOMP messaging support

VORTEXBIRD
Information Technology becomes easy

Inyección de dependencias

- 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 quien cree el objeto.
- El término fue acuñado por Martin Fowler.

VORTEXBIRD
Information Technology becomes easy

Sin Inyección de dependencias

```
public class Vehiculo {

   private Motor motor = new Motor();

   /** @retorna la velocidad del vehículo*/
   public Double enAceleracionDePedal(int presionDePedal) {
      motor.setPresionDePedal(presionDePedal);
      int torque = motor.getTorque();
      Double velocidad = ... //realiza el cálculo
      return velocidad;
   }

}
```

VORTEXBIRD
Information Technology becomes easy

Con Inyección de dependencias sin contenedor o motor

```
public class Vehiculo {

   private Motor motor = null;

   public setMotor(Motor motor){
            this.motor = motor;
   }

   /** @retorna la velocidad del vehículo*/
   public Double enAceleracionDePedal(int presionDePedal) {
      Double velocidad = null;
      if (null != motor){
            motor.setPresionDePedal(presionDePedal);
            int torque = motor.getTorque();
            velocidad = ... //realiza el cálculo
      }

      return velocidad;
   }
}
```

VORTEXBIRD
Information Technology becomes easy

Con Inyección de dependencias sin contenedor o motor

```
//se omite la clase Motor ya que no es relevante para este ejemplo
public class VehiculoFactory {

    public Vehiculo construyeVehiculoElectrico() {
        Vehiculo vehiculo = new Vehiculo();
        Motor motorElectrico = new MotorElectrico ();
        vehiculo.setMotor(motorElectrico);
        return vehiculo;
    }

    public Vehiculo construyeVehiculoGasolina() {
        Vehiculo vehiculo = new Vehiculo();
        Motor motorGasolina = new MotorGasolina();
        vehiculo.setMotor(motorGasolina);
        return vehiculo;
    }

}
```

VORTEXBIRD
Information Technology becomes easy

Con Inyección de dependencias usando Spring

```
public class Vehiculo {

   @Autowired
   private Motor motor;

  public Double enAceleracionDePedal(int presionDePedal) {
        Double velocidad = null;
        motor.setPresionDePedal(presionDePedal);
        int torque = motor.getTorque();
        velocidad = ... //realiza el cálculo
         return velocidad;
    }
}
```

VORTEXBIRD
Information Technology becomes easy

Con Inyección de dependencias con EJB

```
public class Vehiculo {

   @EJB
   private Motor motor;

  public Double enAceleracionDePedal(int presionDePedal) {
        Double velocidad = null;
        motor.setPresionDePedal(presionDePedal);
        int torque = motor.getTorque();
        velocidad = ... //realiza el cálculo
         return velocidad;
    }
}
```

VORTEXBIRD
Information Technology becomes easy

Con Inyección de dependencias con EJB

```
public class Vehiculo {

 @Inject
 private Motor motor;

  public Double enAceleracionDePedal(int presionDePedal) {
        Double velocidad = null;
        motor.setPresionDePedal(presionDePedal);
        int torque = motor.getTorque();
        velocidad = ... //realiza el cálculo
         return velocidad;
    }
}
```

VORTEXBIRD
Information Technology becomes easy

Inversión de control

- Inversion of Control (IoC)
- Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedure calls) o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

VORTEXBIRD
Information Technology becomes easy

TInversión de control

- Principio de Hollywood
  - “no nos llames; nosotros te llamaremos”
- Las implementaciones mas comunes:
  - Callback
  - Listener

Laboratorio inyección de dependencias

VORTEXBIRD
Information Technology becomes easy

Configurando aplicación usando Spring

- Agregar depencias de Spring
- Instalar el contenedor de Inversión de Control de Spring (IoC)
- Crear y configurar un Bean en el contenedor de Inversión de Control .
- Usando el Auto-Writing Bean con XML
- Usando el Auto-Writing Bean con Anotaciones.

Spring con Maven

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.1.10.RELEASE</version>
    </dependency>
</dependencies>
```

Spring con Gradle

```
compile 'org.springframework:spring-context:5.1.10.RELEASE'
```

applicationContext.xml

- Este archivo permite hacer las configuraciones de las capacidades soportadas por 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"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">


</beans>
```

VORTEXBIRD
Information Technology becomes easy

Configurando un Bean

```
<?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"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-4.3.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

    <bean id="motor" class="co.edu.usbcali.demo.autos.Motor”/>

</beans>
```

VORTEXBIRD
Information Technology becomes easy

Configurando el Bean con atributos

```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"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

    <bean id="automovil" class="co.edu.usbcali.demo.autos.Automovil">
        <property name="color" value="Azul"/>
        <property name="marca" value="BMW"/>
        <property name="modelo" value="2017"/>
        <property name="motor" ref="motor"/>
    </bean>

</beans>
```

VORTEXBIRD
Information Technology becomes easy

Usando el Auto-Writing Bean con XML

```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:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

        <bean name="vehiculo" class="com.vortexbird.demo.modelo.Vehiculo" autowire="byType"/>
        <bean name="motor" class="com.vortexbird.demo.modelo.Motor"/>

</beans>
```

VORTEXBIRD
Information Technology becomes easy

Usando el Auto-Writing Bean con Anotaciones

```
<?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:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <context:annotation-config/>
    <context:component-scan base-package="com.vortexbird.demo" />

</beans>
```

Junit y Spring

Test

```
package co.edu.usbcali.demo.autos;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit.jupiter.SpringExtension;

@ExtendWith(SpringExtension.class)
@ContextConfiguration("/applicationContext.xml")
class TestAutos {

	@Autowired
	ApplicationContext applicationContext;


@Test
void test() {
		assertNotNull(applicationContext);
}


}
```

VORTEXBIRD
Information Technology becomes easy

Configurando aplicación usando Spring

- Factory Method

```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"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">


    <bean id="clienteService"
          class="co.edu.usbcali.demo.service.ClienteService"
          factory-method="createInstance"/>

</beans>
```

```java
package co.edu.usbcali.demo.service;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ClienteService {

    private final static Logger log=LoggerFactory.getLogger(ClienteService.class);

    private final static ClienteService clienteService=new ClienteService();

    private ClienteService() {}

    public static ClienteService createInstance() {
        log.info("Se creo el clienteService");
        return clienteService;
    }

}
```

Configurando aplicación usando Spring

- Factory Bean

```
public class DefaultServiceLocator {

    private static ClientService clientService = new ClientServiceImpl();

    private static AccountService accountService = new AccountServiceImpl();

    public ClientService createClientServiceInstance() {
        return clientService;
    }

    public AccountService createAccountServiceInstance() {
        return accountService;
    }
}
```

Configurando aplicación usando Spring

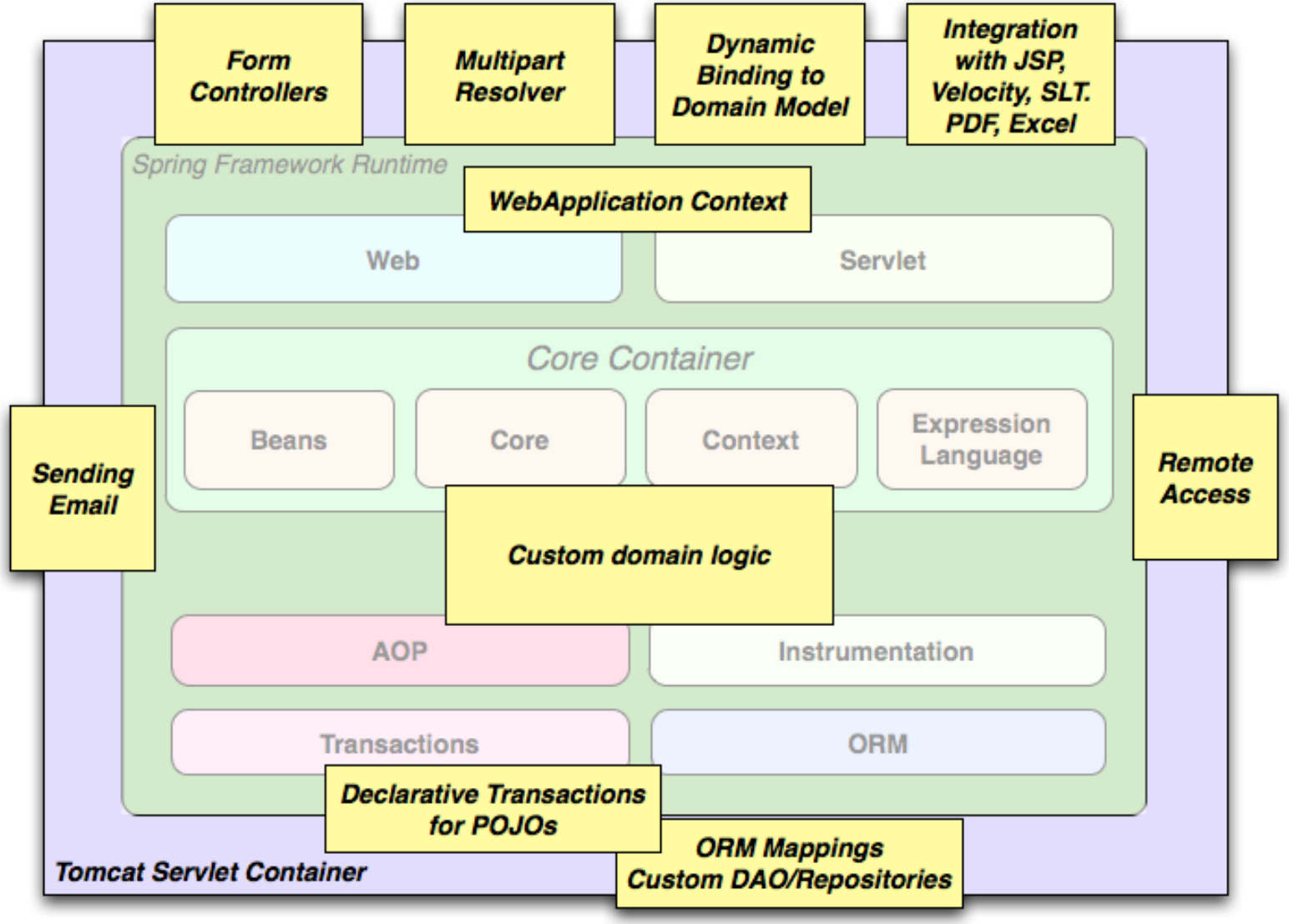
- Factory Bean

```xml
<bean id="serviceLocator" class="examples.DefaultServiceLocator">
    <!-- inject any dependencies required by this locator bean -->
</bean>

<bean id="clientService"
    factory-bean="serviceLocator"
    factory-method="createClientServiceInstance"/>

<bean id="accountService"
    factory-bean="serviceLocator"
    factory-method="createAccountServiceInstance"/>
```

Escenarios de Uso

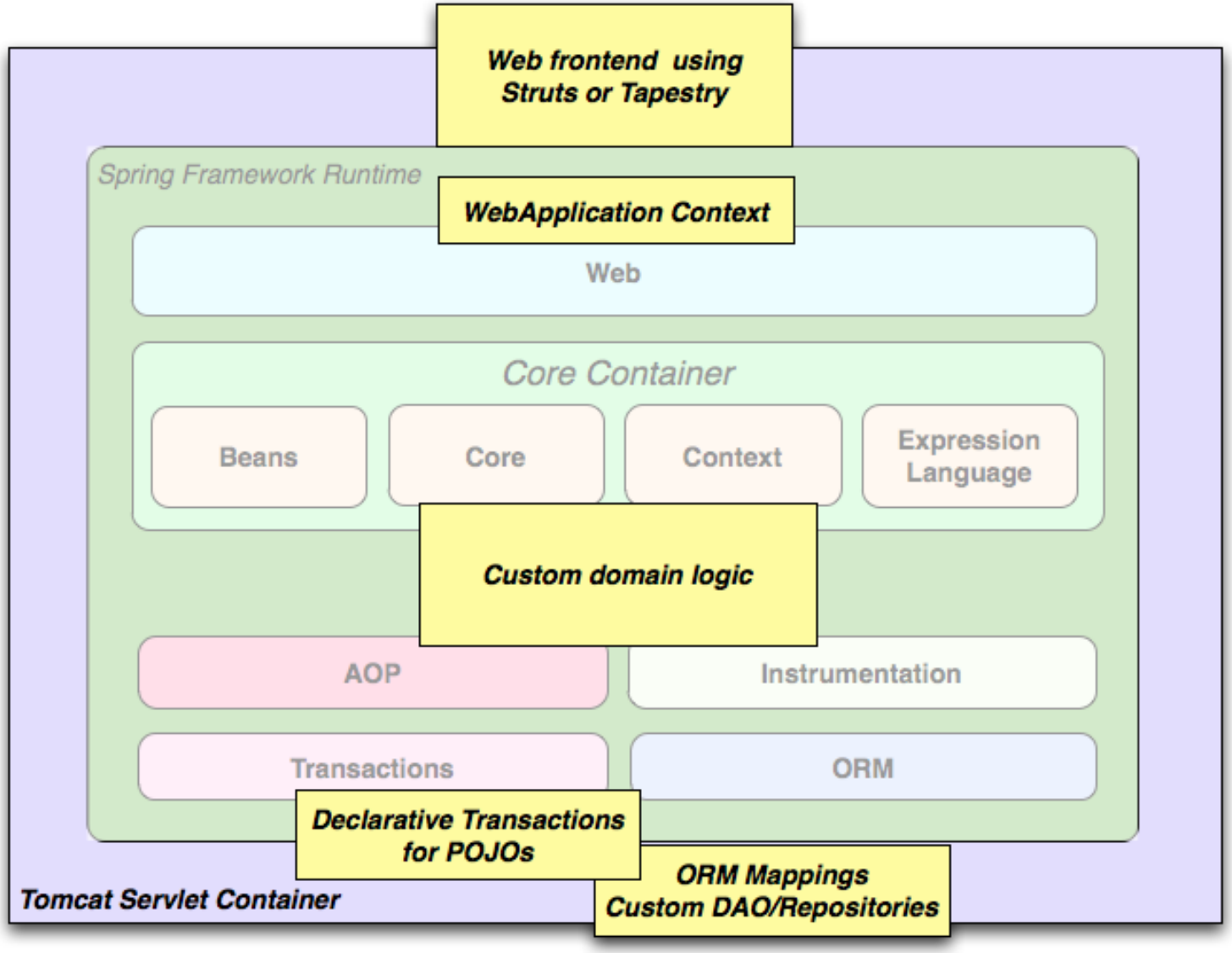
VORTEXBIRD
Information Technology becomes easy

Typical full-fledged Spring web application



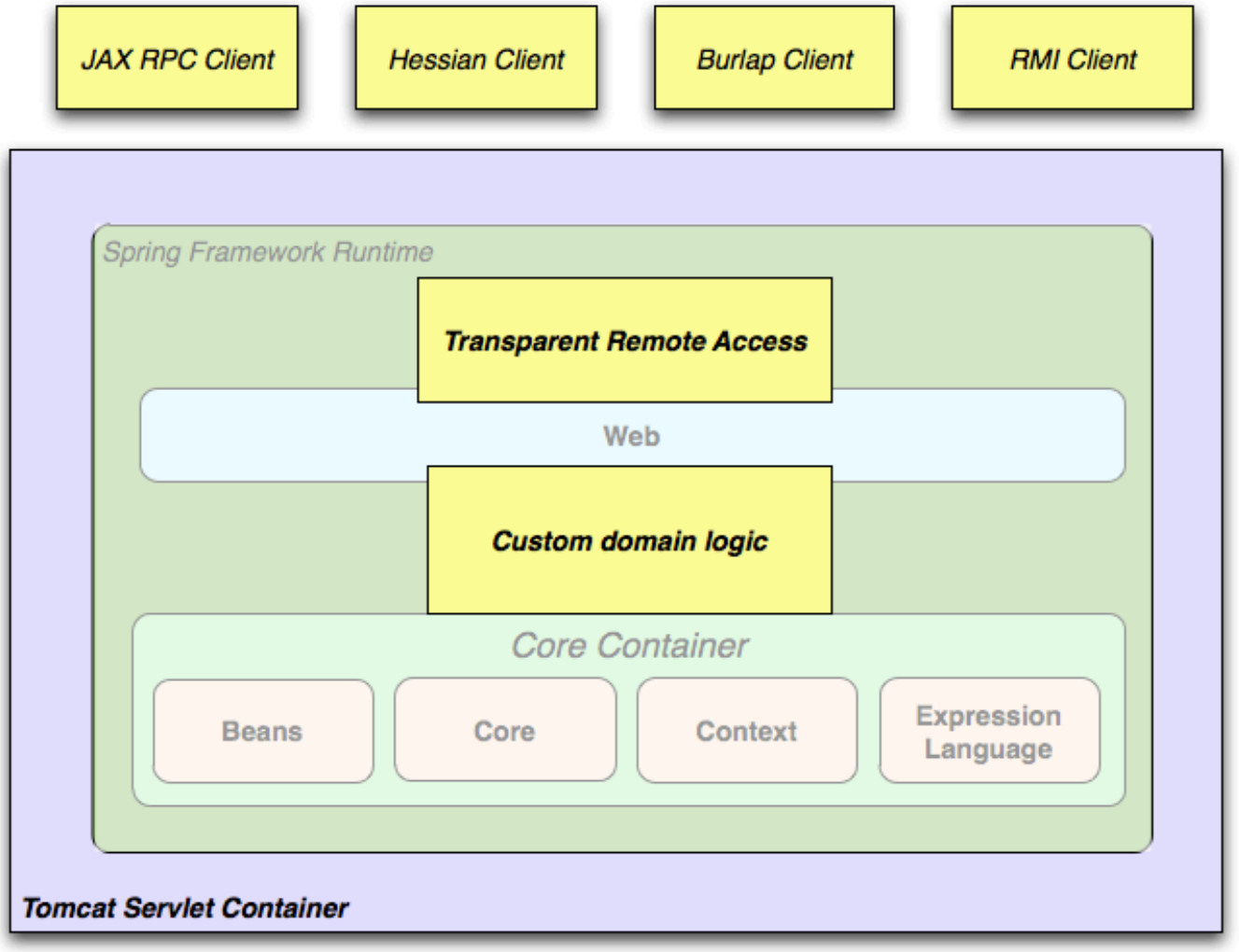
VORTEXBIRD
Information Technology becomes easy

Spring middle-tier using a third-Web

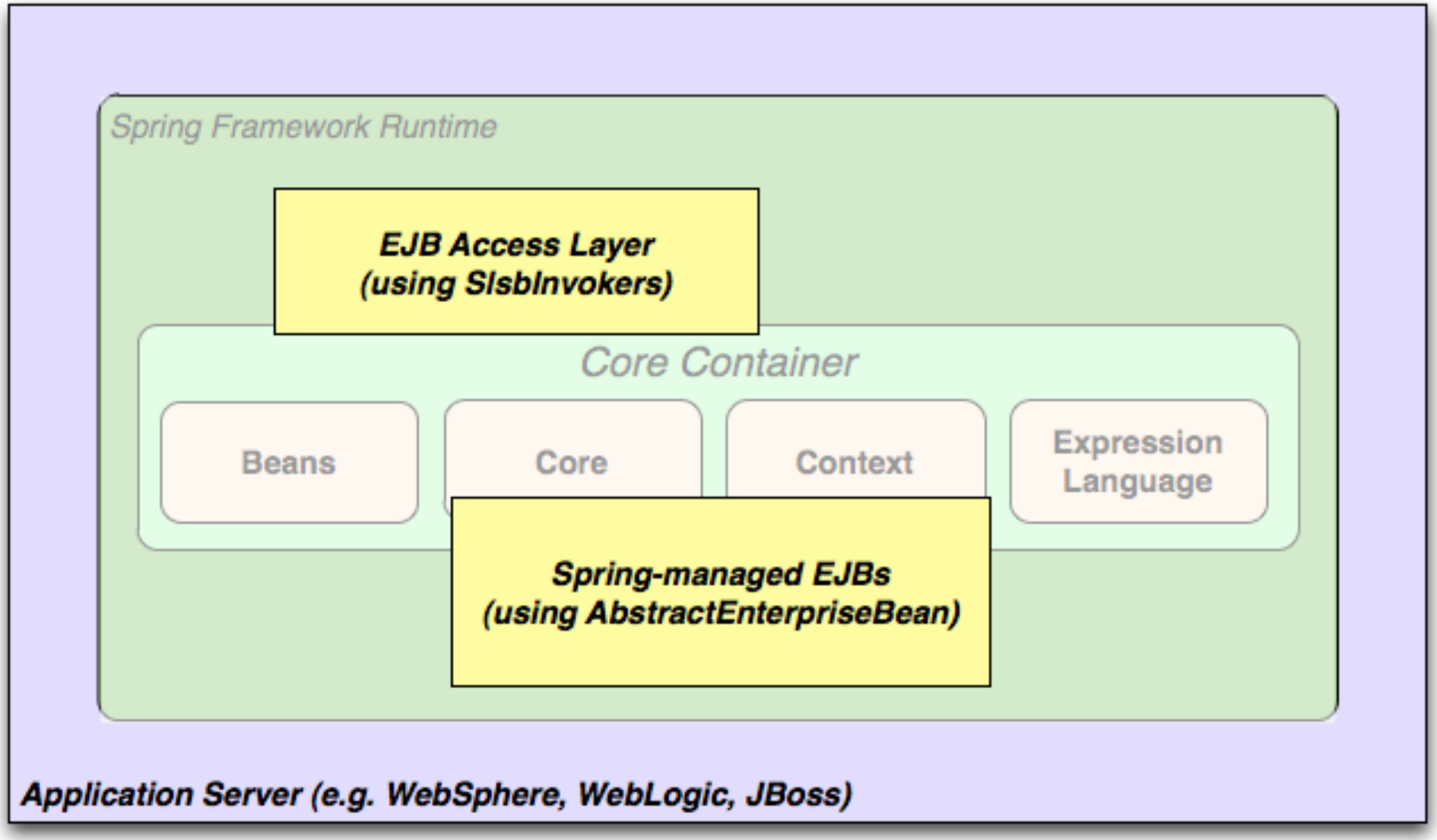


VORTEXBIRD
Information Technology becomes easy

Remoting usage scenario



EJBs - Wrapping existing POJOs



VORTEXBIRD

Information Technology becomes easy

Gracias

info@vortexbird.com

(057-2) 3798187 / (057) 316 482 4629

Calle 18 No 118 – 241 Oficina 21 Parque Cañas Gordas (Cali - Colombia)