



spring®

[HTTPS://SPRING.IO/](https://spring.io/)

# Qué es Spring Framework?

**Es un robusto Framework para el Desarrollo de Aplicaciones empresariales en el lenguaje JAVA**

- ▶ Aplicaciones Web MVC
- ▶ Aplicaciones Empresariales
- ▶ Aplicaciones de escritorio
- ▶ Aplicaciones Batch
- ▶ Integración con REST/SOA
- ▶ Spring Data
- ▶ Spring Security

# SPRING FRAMEWORK SE UTILIZA PARA CONSTRUIR APLICACIONES EMPRESARIALES

Una aplicación empresarial es...

- ▶ Gran aplicación comercial, industrial.
- ▶ Compleja, escalable, distribuida, crítica
- ▶ Despliegue en redes corporativas o internet
- ▶ Centrada en los datos
- ▶ Intuitiva, de uso fácil
- ▶ Requisitos de seguridad y mantenibilidad.

# Características: CDI(Contextos e inyección de dependencia)

- ▶ Gestión de configuración basada en componentes JavaBeans y aplica el principio Inversión de control, específicamente utilizando la inyección de dependencias (DI) para manejar relaciones entre los objetos, evitando relaciones manuales y creaciones de instancias explícitas con operador new, esto hace un bajo acoplamiento y alta cohesión, mejorando la reutilización y mantención de los componentes
- ▶ Spring en su CORE está basado en un contenedor liviano y es usado globalmente dentro de nuestra aplicación

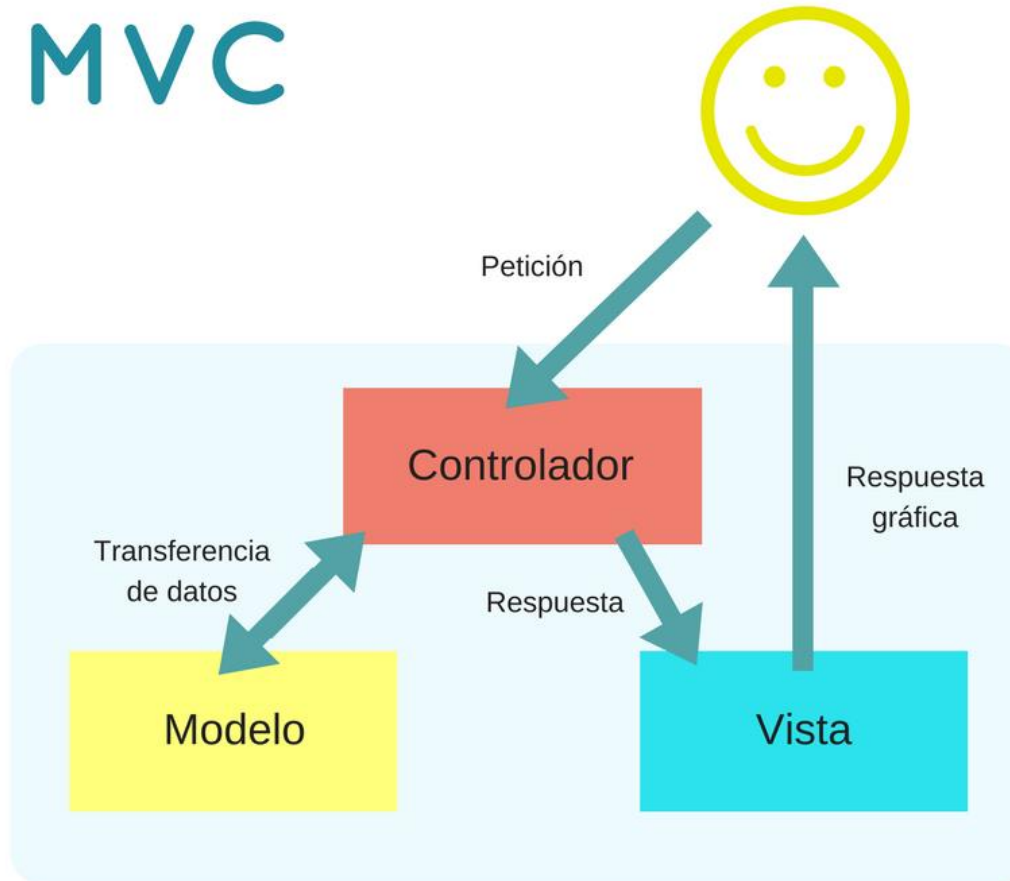
# Características: ORM y Persistencia

- ▶ Alta abstracción sobre el API JDBC
- ▶ Integración con Frameworks de persistencia como Hibernate, JPA (Java Persistence API) etc.
- ▶ Soporte de la lógica de negocio, específicamente en clases de acceso a datos (DAO Support)
- ▶ Componentes encargados de la gestión de transacciones de base de datos.



# Características: MVC

La arquitectura MVC es uno de los principales componentes y tecnologías, y como su propio nombre nos indica implementa una arquitectura Modelo-Vista-Controlador



Soporta varias tecnologías para generación de las vistas, entre ellas JSP, Thymeleaf, FreeMarker, Velocity, Tiles, iText, y POI (Java API para archivos Microsoft Office)

# Características: AOP (Programación orientada a Aspectos)

- ▶ AOP es un paradigma de programación que permite modularizar las aplicaciones y mejorar la separación de responsabilidades entre componentes y/o clases
- ▶ Similar a los componentes de Inyección de Dependencia, AOP tiene como objetivo mejorar la modularidad de nuestra aplicación
- ▶ Spring amplía la programación orientada a aspectos AOP para incluir servicios tales como manejo de transacciones, seguridad, logger etc.

# ¿Por qué usar Spring Framework?

**Modularidad de Componentes:** a través del patrón Inyección de Dependencia(CDI).

- ▶ Promueve la composición y modularidad entre las partes que componen una aplicación.
- ▶ Plain Old Java Objects(POJO) mantienen su código limpio, simple y modular, bajo acoplamiento y alta cohesión.



# ¿Por qué usar Spring Framework?

## **Simplicidad**

- ▶ Las aplicaciones con Spring son simples y requieren mucho menos código (JAVA y XML) para la misma funcionalidad

## **Capacidad de pruebas unitarias**

- ▶ Dependencias limpias, actualizadas y lo justo y necesario, aseguran que la integración con unit testing sea muy simple
- ▶ Clases POJO se pueden testear sin estar atado al framework

# ¿Por qué usar Spring Framework?

## **Facilidad de configuración**

- ▶ Se elimina la mayor parte del código repetitivo y la configuración de XML a partir de sus aplicaciones y mayor uso de anotaciones

## **AOP(Asspect Oriented Programming)**

- ▶ Programación declarativa AOP, paradigma de programación que permite modularizar las aplicaciones y mejorar la separación de responsabilidades entre módulos y/o clases aspectos
- ▶ Facilidad de configurar aspectos, soporte de transacciones, seguridad.

## **Diseño orientado a interfaces**

- ▶ Programación basada en contratos de implementación, permitiendo al usuario centrarse en la funcionalidad, ocultando el detalle de implementación.

# ¿Por qué usar Spring Framework?

## **Plenamente probado, seguro y confiable**

- ▶ Spring ha sido probado y utilizado en diversos proyectos alrededor del mundo, como en Instituciones Bancarias, Aseguradoras, Instituciones Educativas y de Gobierno, entre muchos otros tipos de proyectos y empresas

## **Productividad**

- ▶ Ganancias de productividad y una reducción en el tiempo de desarrollo e implementación utilizando Spring

# ¿Por qué usar Spring Framework?

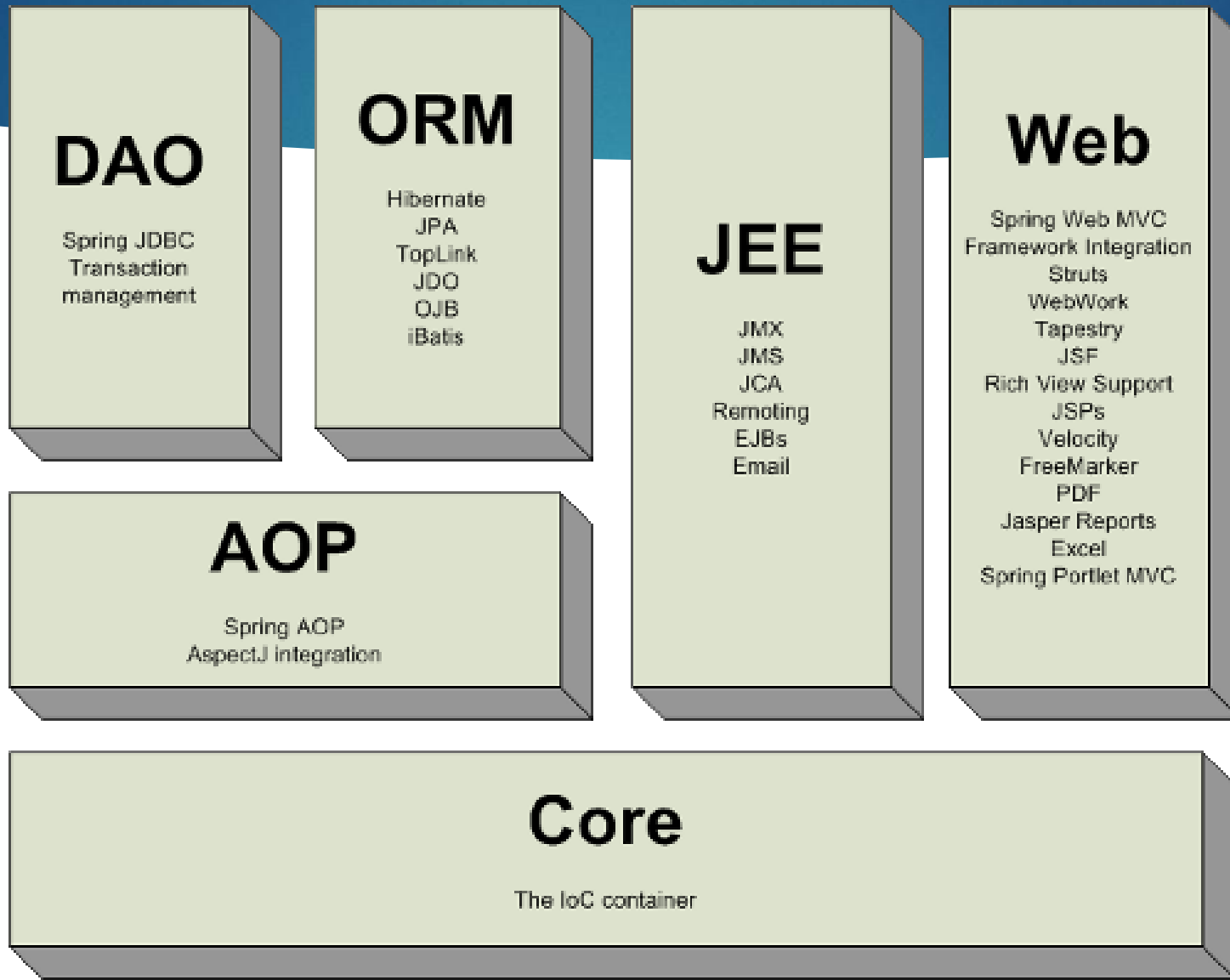
## **Integración con otras Tecnologías**

- ▶ EJB (Logica de negocio)
- ▶ JPA, Hibernate, iBates, JDBC(Persistencia)
- ▶ Velocity, etc(Vista)
- ▶ JSF2, Struts, etc(Capa Web)

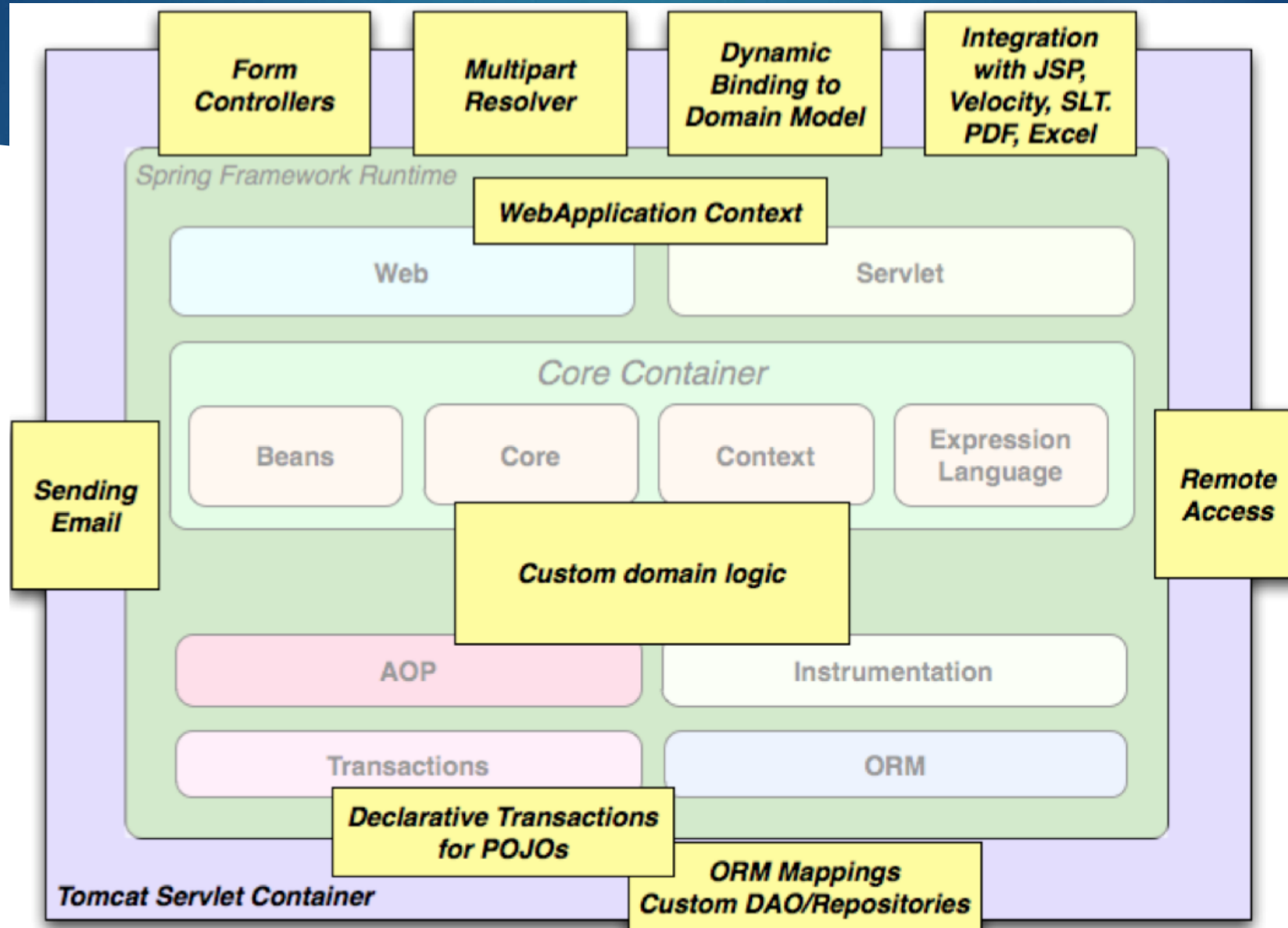
## **Otras Razones**

- ▶ Bien diseñado
- ▶ Abstracciones aíslan detalles de la aplicación, eliminando código repetitivo
- ▶ Fácil de extender
- ▶ Muchas clases reutilizables

# Arquitectura de Spring



# Arquitectura de Spring



# Arquitectura Spring

**La arquitectura se compone en distintas capas, cada una tiene su función específica:**

- ▶ **Capa Web:** Spring simplifica el desarrollo de interfaces de usuario en aplicaciones Web MVC mediante el soporte de varias tecnologías para generación de contenido, entre ellas JSP, Thymeleaf, FreeMarker, Velocity, Tiles etc.
- ▶ **Capa Lógica de Negocio:** en esta capa podemos encontrar tecnología como los Java Beans (POJOs), Dao Support, Services, EJBs etc. Y clases Entities.
- ▶ **Capa de Datos:** aquí vamos a encontrar tecnologías JDBC, ORM (JPA ,Hibernate, etc.), Datasource y conexiones a bases de datos.

# Escenarios de uso

- ▶ Podemos usar Spring en todo tipo de escenarios, desde pequeñas app o páginas web hasta grandes aplicaciones empresariales implementando Spring Web MVC, control de transacciones, remoting, web services e integración con otros framework como struts
- ▶ Spring es utilizado en diversos proyectos alrededor del mundo, como en instituciones bancarias, aseguradoras, instituciones educativas y de gobierno, entre muchos otros tipos de proyectos y empresas



# Spring vs Struts 2

- ▶ Hay un punto bien importante que los diferencia enormemente, y es que Struts2 es sólo un Framework Web MVC mientras que Spring además de tener un componente Web MVC tiene varios componentes más por ejemplo para la persistencia que integra diferentes Framework de persistencia y ORM como Hibernate JPA, iBatis JDO etc.. Además de los componentes IoC para trabajar con inyección de dependencia, diferentes resoluciones de vista hasta integra componentes como Jasper, EJB, WS, AOP etc, es decir es un mundo mucho más amplio que struts, por lo tanto lo hace mucho más grande, completo y robusto.
- ▶ Además ambos se puede integrar, por ejemplo usar el MVC de struts y todo lo que es persistencia e inyección se hace con spring.

# Spring vs EJB3

- ▶ Comparando Spring Framework y la plataforma EJB3, podríamos decir que no son tan comparables en muchos aspectos, por ejemplo spring es un Framework Java EE (como tal con todas sus letras) que tiene componentes web con mvc, persistencia, ioc, aop, forms, layout, pdf, rest, validaciones etc, no sólo está relacionado a la lógica de negocio y acceso a datos si no también a la capa web e incluso aplicaciones standard alone, mientras que EJB es sólo persistencia, transacciones, seguridad, aop y lógica de negocio (no web), solo podríamos hacer un comparativo sobre el acceso a datos de spring con el uso de ejb y persistencia básicamente.
- ▶ Por otro lado los componentes de spring, los beans no se despliegan de forma remota, sólo local dentro de un proyecto, mientras que los EJB3 está basado en CORBA y los beans se pueden desplegar en un servidor de aplicaciones y acceder de forma local y remota.
- ▶ Por lo tanto podemos decir que son tecnologías complementarias, ya que podríamos tener un spring web mvc que trabaja la lógica de negocio y persistencia a través de los ejb y no con su propio componente Hibernate dao support u otro, pero pueden ser sustitutivas en el lado de la persistencia, dos caminos alternativos, incluso en un proyecto podría ser ambas con ejb que accede a datos desde otro server y localmente accedemos a los datos con spring, las variaciones son infinitas.
- ▶ Sin duda Spring es un Framework complejo, pero como todo en la vida es tire y floja, practica y práctica.

# Herramientas necesarias

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

- ▶ SpringSource Tool Suite (STS)
- ▶ Es un IDE(entorno de desarrollo basado en Eclipse) para crear aplicaciones empresariales de Spring.
- ▶ Soporta Java, Spring, Groovy y Grails.
- ▶ Viene incluido el servidor Tc vFabric, que es un tomcat que está optimizado para Spring
- ▶ También incluye plugins específicos para trabajar con spring y plantillas para generación de proyectos spring

# Eclipse

<https://www.eclipse.org/downloads/>

- ▶ Eclipse es un entorno de desarrollo software multi-lenguaje construido alrededor de un workspace al que pueden incluirse un gran número de plug-ins que proporcionan funcionalidades concretas relacionadas con lenguajes específicos o con la interacción con otras herramientas implicadas en el desarrollo de una aplicación. Pese a ser un entorno multi-lenguaje, está desarrollado en Java, siendo el desarrollo en este lenguaje su aplicación principal.
- ▶ Eclipse en su versión **JEE**, es el IDE más utilizado para desarrollar aplicaciones empresariales en JAVA, para la utilización de Spring framework se debe agregar ciertos plugins.

# ¿Que es la Inyección de dependencia?



# Resuelve el problema de reutilización y modularidad entre componentes

- Inyectar es justamente suministrar a un objeto una referencia de otros que necesite según la relación, tiene que plasmarse mediante configuración XML o la anotación `@Autowired`

# Principio Hollywood

- ▶ No nos llames, nosotros te llamaremos

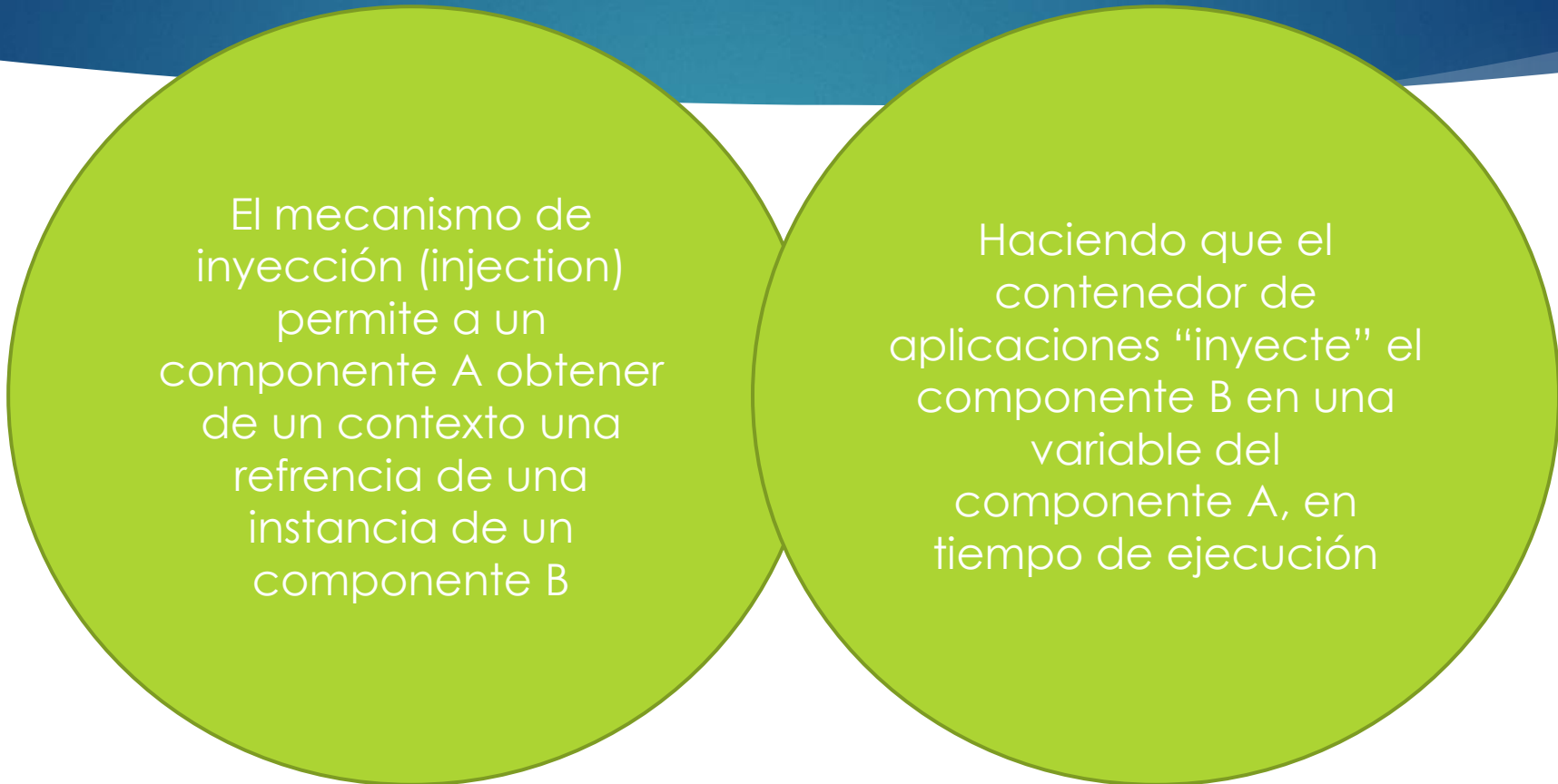
## También es un tipo de Inversión de Control (IoC):

- ▶ “CDI Container” Maneja los contexto y resuelve dependencias de componentes mediante la asociación e inyección de objetos (push)
- ▶ En contra- oposición de la creación explícita (operador new) de objetos (pull)



# También es un tipo de Inversión de Control (IoC):

- ▶ El “Contenedor” se encarga de gestionar las instancias y relaciones (así como sus creaciones y destrucciones) de los objetos
- ▶ El objetivo es lograr un bajo acoplamiento entre los objetos de nuestra aplicación
- ▶ Martin Fowler lo llama inyección de dependencias (DI)



The diagram consists of two overlapping light green circles. The left circle contains text describing the mechanism of injection, and the right circle contains text describing the action of the container. The background features a dark blue header with a light green vertical bar on the right side.

El mecanismo de  
inyección (injection)  
permite a un  
componente A obtener  
de un contexto una  
referencia de una  
instancia de un  
componente B

Haciendo que el  
contenedor de  
aplicaciones “inyecte” el  
componente B en una  
variable del  
componente A, en  
tiempo de ejecución

## Presentación

Vista JSP

Atributo y/o  
Objetos

La página JSP o vista puede  
acceder a objetos enviados  
por el controller

## Contexto de Spring

Controller

HibernateDao

El controlador ya contiene  
las dependencias  
inyectadas

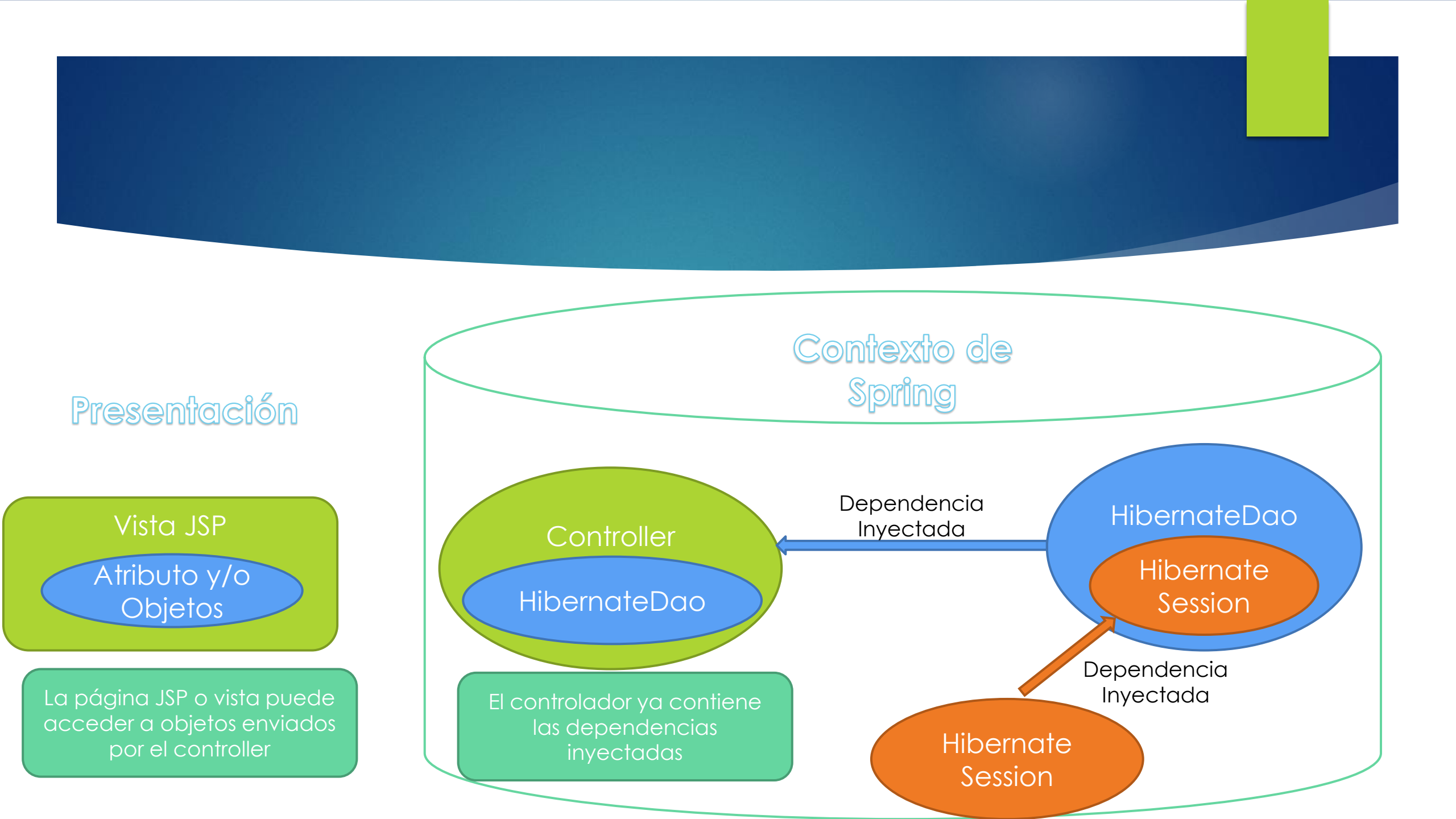
Dependencia  
Inyectada

HibernateDao

Hibernate  
Session

Dependencia  
Inyectada

Hibernate  
Session



# Por qué Inyección de dependencia

- ▶ Flexible: No hay necesidad de tener código lookup en la lógica de negocio
- ▶ Testable con clases POJO: Pruebas unitarias automáticas (como parte del proceso de compilación)- Maven o Ant

# Modular y Extensible

- ▶ Permite reutilizar en diferentes contextos y entornos de aplicación, mediante configuración de anotaciones en lugar de código.
- ▶ Promueve una forma de trabajo consistente, que alinea a todos nuestros proyectos y al equipo de desarrollo bajo un mismo estandar.

# Tres variantes para implementar Inyección de dependencia

## **Inyección de dependencia via constructor:**

- ▶ las dependencias se proporcionan a través de los constructores de una clase o componente
- ▶ Pasándose como argumento

## **Mediante método Setter**

- ▶ Las dependencias se establecen mediante métodos setter de un componente(estilo JavaBean)

## **Mediante atributo**

- ▶ Las dependencias se establecen directamente sobre el atributo de la clase componente o beans
- ▶ Es la forma más típica y recomendada

# Inyección de dependencia vía atributo

- ▶ En general, las dependencias se establecen a través de los atributos de un componente Spring usando la anotación `@Autowired`

```
public class InyeccionSetter {  
    @Autowired  
    private Dependencia miDependencia;  
}
```

# Inyección de dependencia vía Setter

- Las dependencias se establecen a través de los métodos setter de un componente Spring usando la anotación @Autowired

```
public class InyeccionSetter {  
    private Dependencia miDependencia;  
  
    @Autowired  
    public void setMiDependencia(Dependencia dep){  
        this.miDependencia = dep;  
    }  
}
```



# Inyección de dependencia Constructor

```
public class InyeccionConstructor{  
    private Dependencia miDependencia;  
  
    @Autowired  
    public InyeccionConstructor(Dependencia dep){  
        this.miDependencia= dep;  
    }  
}
```

# Beans

- ▶ El término “bean” (o componente) se utiliza para referirse a cualquier componente manejado por Spring
- ▶ Los “beans” son clases en forma de JavaBeans ,Sin args en el constructor, Método getter y setter para los atributos
- ▶ Atributos de los “beans” pueden ser valores simples o probablemente referencias a otros “beans”
- ▶ Los “beans” pueden tener varios nombres

# Anotación @Autowired

- ▶ Se utiliza en el código java, en clases sobre atributos, métodos, setter, constructor para especificar requerimiento DI( en vez de archivo XML)
- ▶ Necesita JDK 1.5 o superior

# Ejemplo @Autowired Clase Target

```
public class Persona{  
    private String nombre = "Rolando Lopez";  
    private int edad = 35;  
    private float altura = 1.78;  
    private boolean esProgramador = true;
```

```
@Autowired
```

```
private Direccion direccion;
```

```
    public Direccion getDireccion(){  
        return direccion;  
    }  
}
```

# Auto-scanning

- ▶ Puede ser usado para crear instancias de los objetos beans en lugar de declararlos en clases de configuración (anotación @Configuration)
- ▶ Spring Boot lo resuelve de forma automática
- ▶ Los beans deben ser anotado con la anotación @Component
- ▶ Cualquier beans anotado con @Component bajo el package base serán instanciados y manejados por el contenedor DI de Spring

# Bean anotado con @Component

```
package com.formacionbdi.dominio;  
import org.springframework.stereotype.Component;
```

```
@Component
```

```
public class Direccion{  
    private int numeroCalle =1234;  
    private String nombreCalle = "Av. Kennedy";  
    private String ciudad = "Santiago";  
    private String pais= "Chile";  
  
    public int getNumeroCalle(){  
        return numeroCalle;  
    }  
    ....codigo...  
}
```