

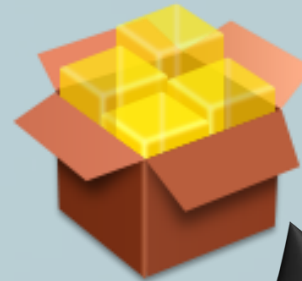
*¿Qué es Spring
Framework?*



¿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



Características: CDI



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 por sobre el API JDBC

Integración con frameworks de persistencia como Hibernate, JPA etc

Soporte de la lógica de negocio, específicamente en clases de acceso a datos (DAO Support)

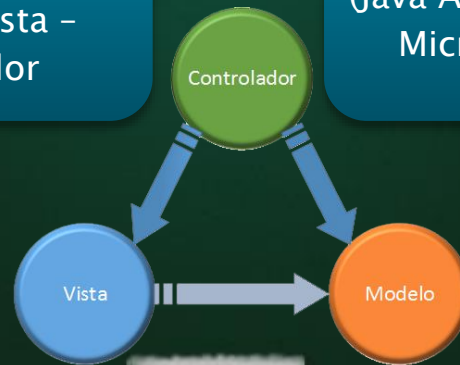
Componentes encargados de la gestión de transacciones de base de datos



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

- 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
Spring?*



¿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 mantienen su código limpio, simple y modular, bajo acoplamiento y alta cohesión
- *Simplicidad*
 - ✓ Las aplicaciones con Spring son simples y requieren mucho menos código (Java y XML) para la misma funcionalidad



¿Por qué usar Spring Framework?

- *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

- *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

¿Por qué usar Spring Framework?

- AOP (Aspect 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 basadas 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 3.2 (Lógica 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 Spring

DAO

Spring JDBC
Transaction
management

ORM

Hibernate
JPA
TopLink
JDO
OJB
iBatis

JEE

JMX
JMS
JCA
Remoting
EJBs
Email

Web

Spring Web MVC
Framework Integration
Struts
WebWork
Tapestry
JSF
Rich View Support
JSPs
Velocity
FreeMarker
PDF
Jasper Reports
Excel
Spring Portlet MVC

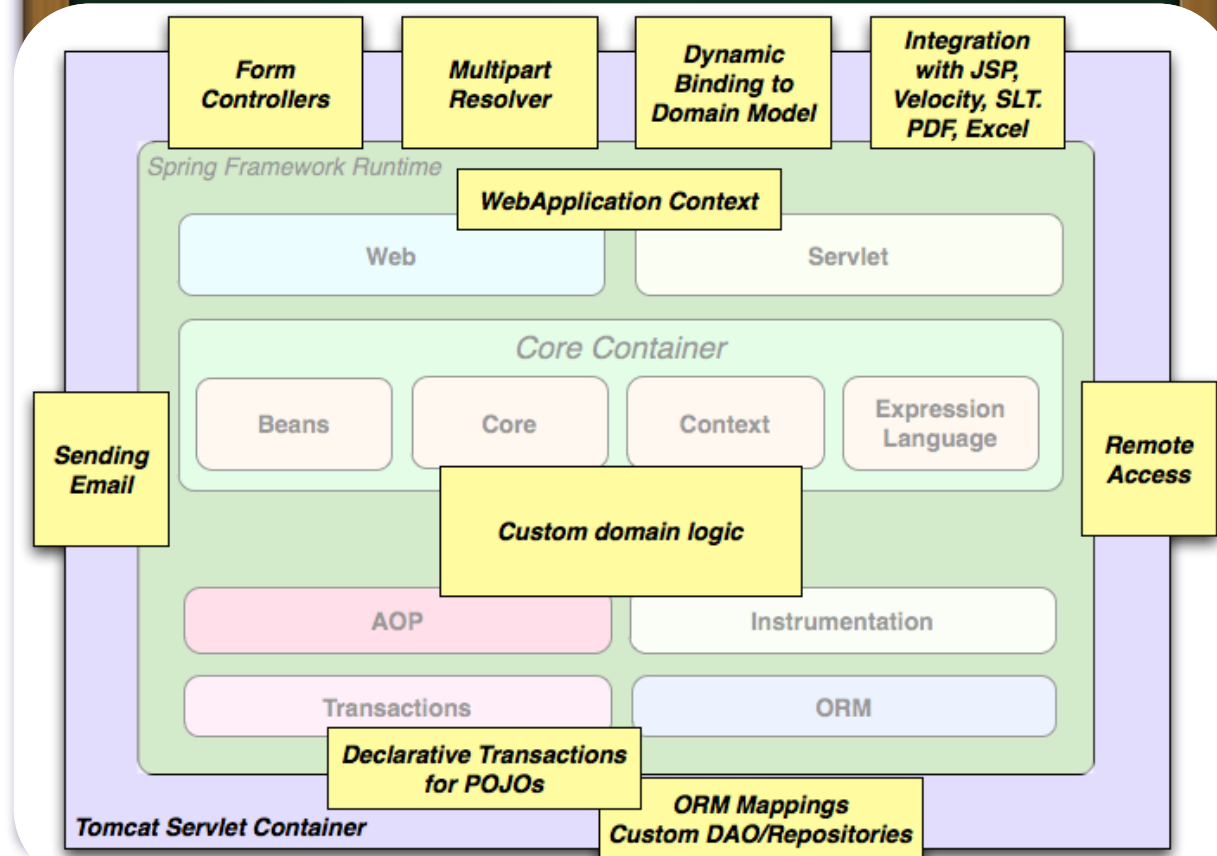
AOP

Spring AOP
AspectJ integration

Core

The IoC container

Arquitectura 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*



Arquitectura Spring

- *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



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 otros Frameworks



Spring vs Struts2

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, IbTIS 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.

Spring vs EJB3

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 trabajar la persistencia con spring data access o ejb persistencia, dos caminos y alternativas, incluso en un proyecto podría ser ambas con ejb que accede a datos desde otro server y locamente 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*





Herramientas

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
 - ✓ Tc vFabric Server es un Tomcat que está optimizado para Spring
- También incluye plugins específicos para trabajar con spring y plantillas para generación de proyectos spring



*¿Qué es la Inyección
de Dependencia?*



Presentación

JSP- JavaServe Page



Inyección de dependencia

DI Spring



Componentes Manejados por
Spring

Contenedor DI
Spring



DAOs



Java Beans



Entities



Contexto de persistencia

Hibernate

*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 contextos 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)*



- 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)



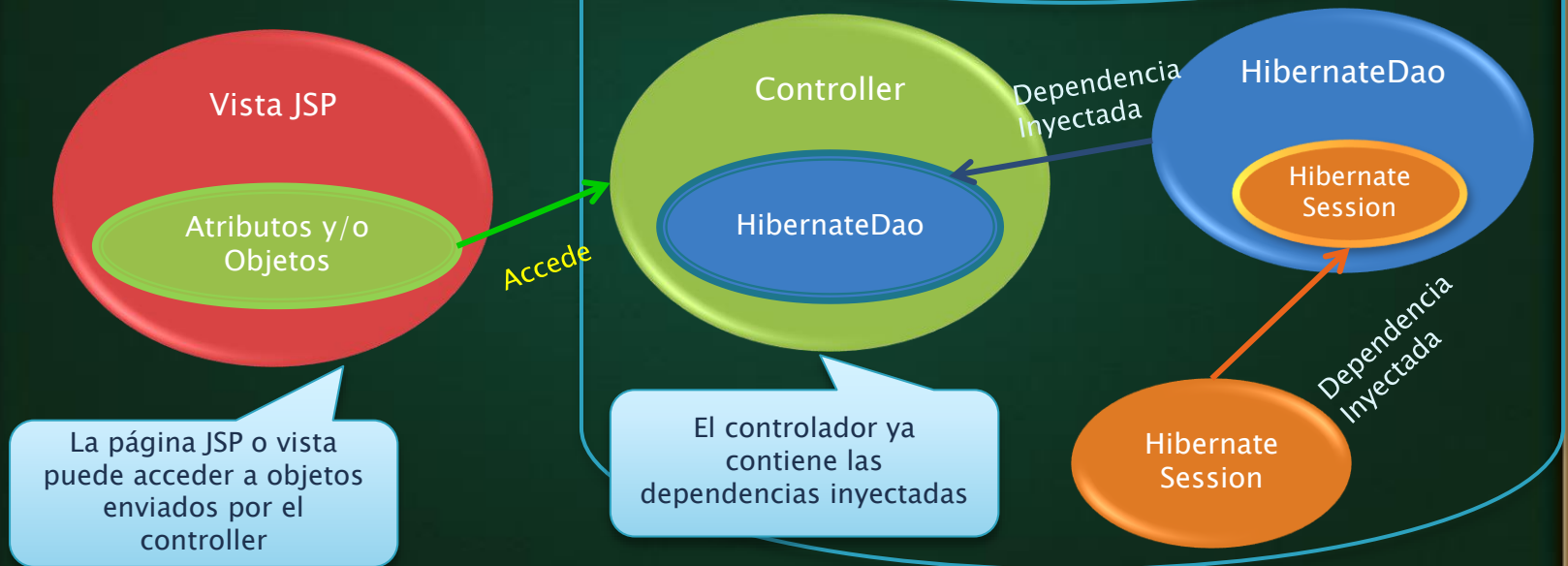
El mecanismo de inyección (injection) permite a un componente A obtener de un contexto una referencia de una instancia de un componente B



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

Presentación

Contexto de Spring



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 del 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 "bean" son clases en forma de JavaBeans
 - ✓ Sin args en el constructor.
 - ✓ Métodos 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*



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 = "Andrés Guzmán";  
    private int edad = 35;  
    private float altura = 1.78;  
    private boolean esProgramador = true;  
  
    @Autowired  
    private Direccion direccion;  
  
    public Direccion getDireccion() {  
        return direccion;  
    }  
  
}
```

Auto-scanning



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;  
    }  
  
    ... etc ...  
}
```