



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

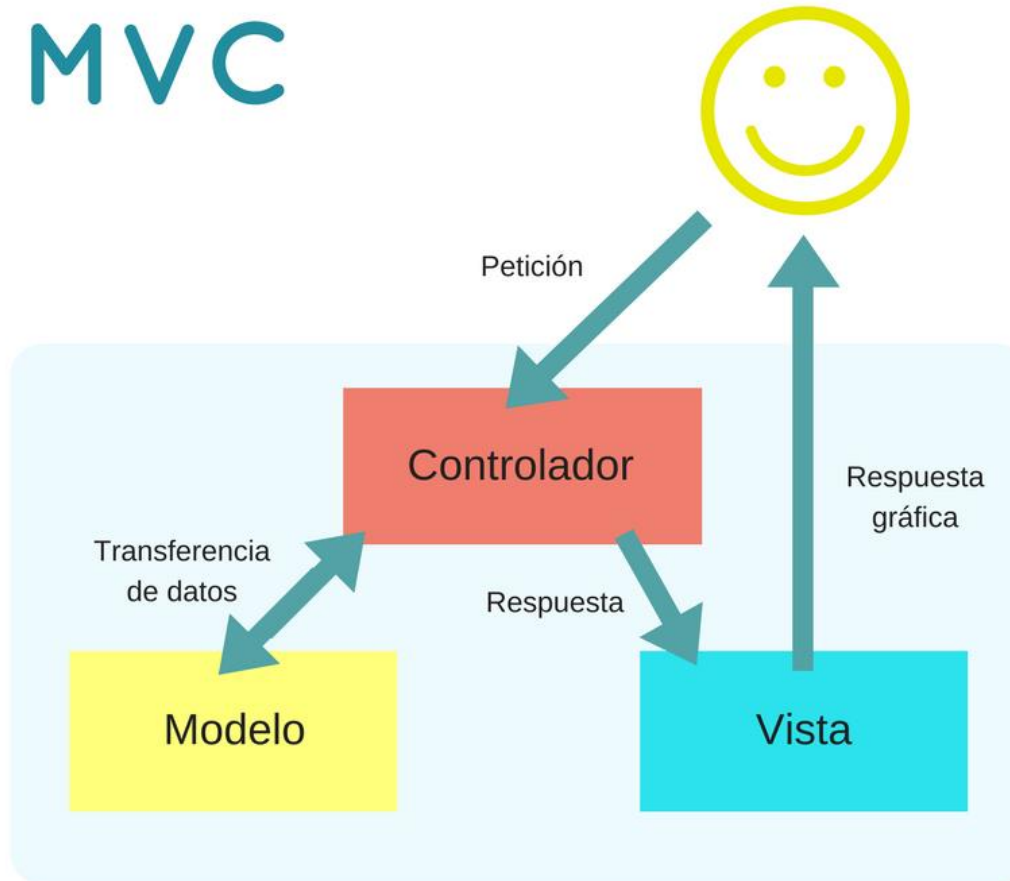
Características: ORM y Persistencia

- ▶ 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?

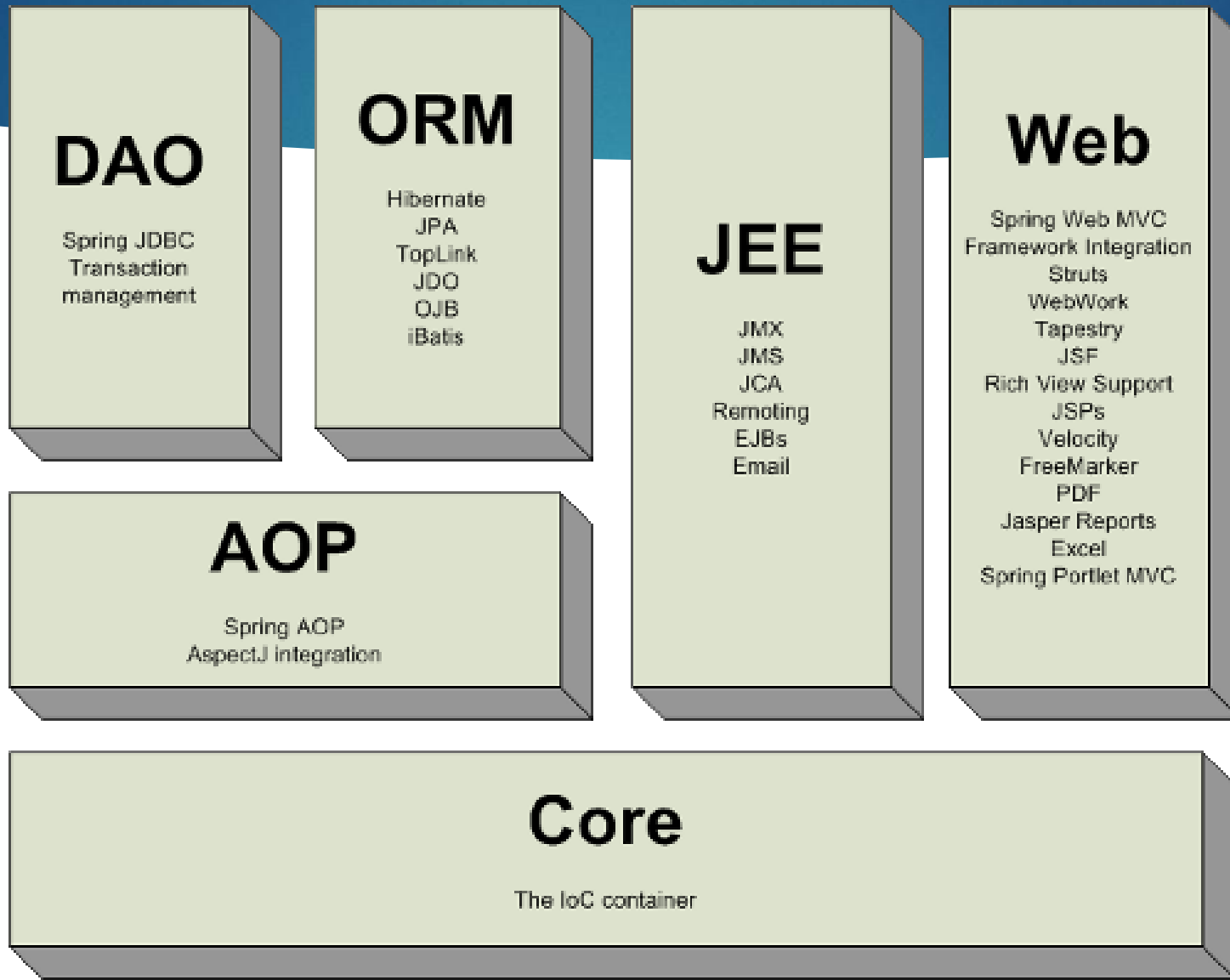
Integración con otras Tecnologías

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

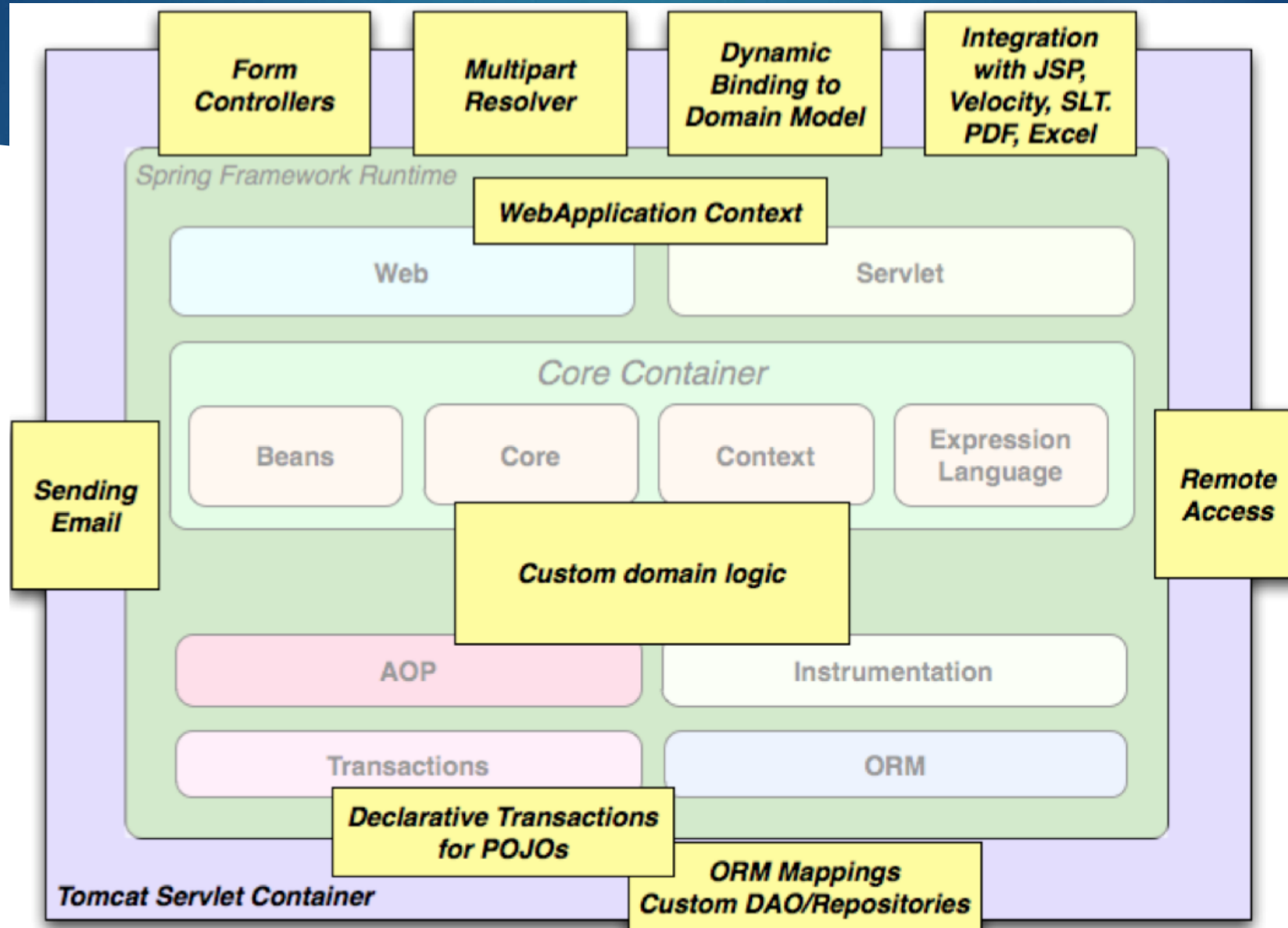
Otras Razones

- ▶ 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?



Principio Hollywood

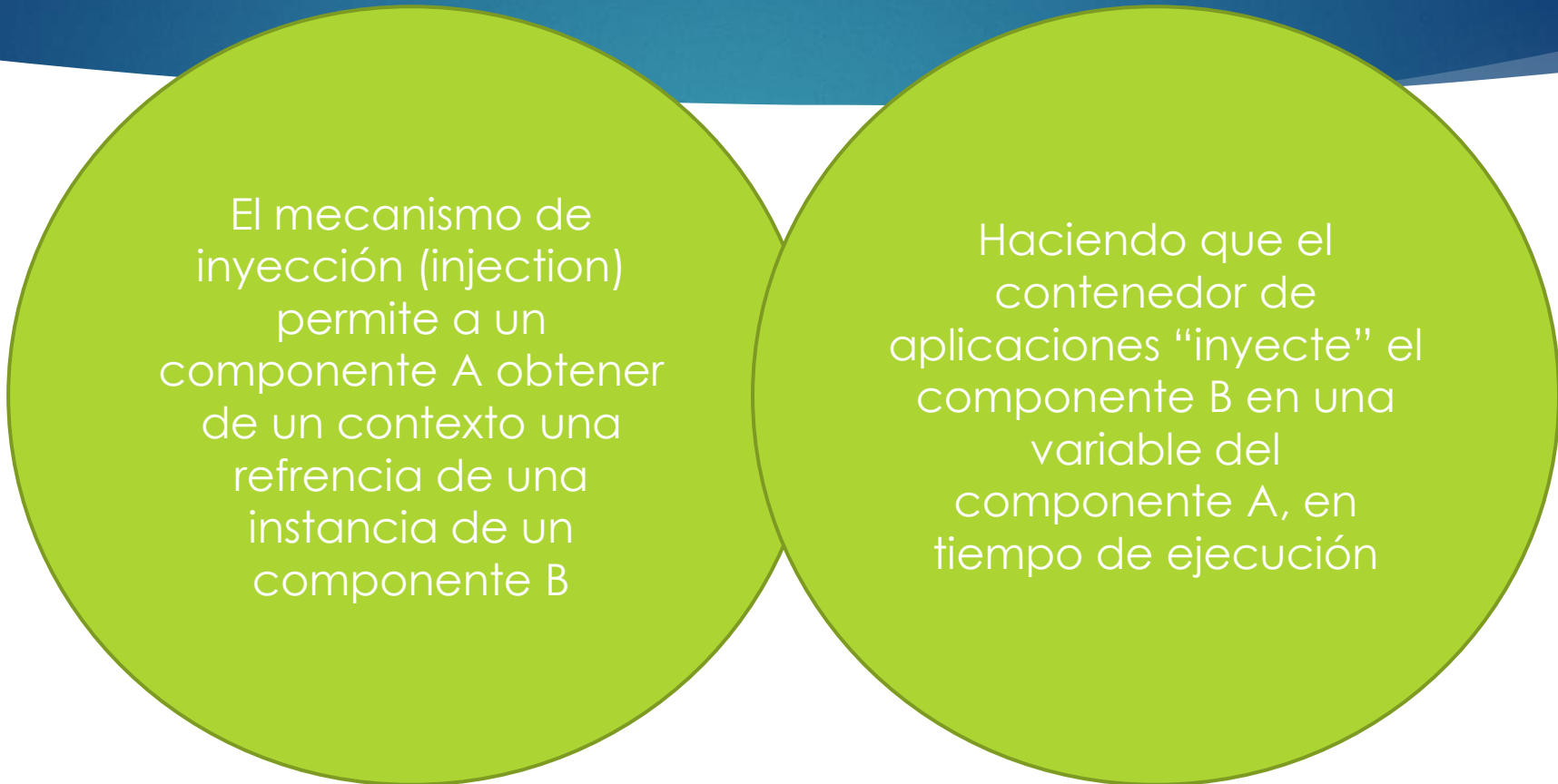
- ▶ No nos llames, nosotros te llamaremos

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

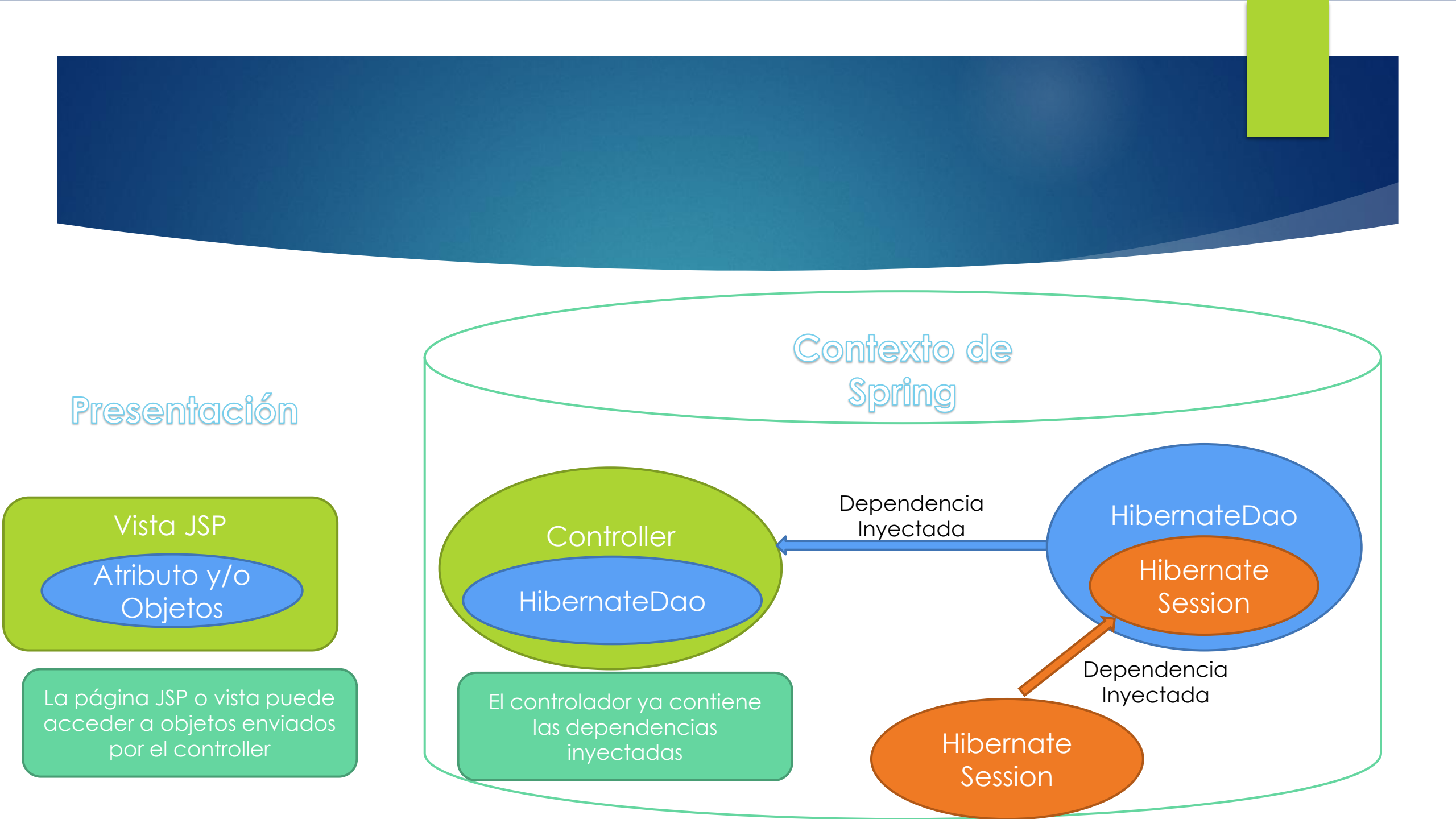
Dependencia
Inyectada

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...  
}
```



*Maven*TM



[HTTPS://MAVEN.APACHE.ORG/](https://maven.apache.org/)

Que es?

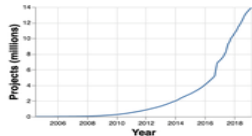
- ▶ Apache Maven es un software de gestión de proyectos y podemos decir que es una herramienta de comprensión. Se basa en el concepto del modelo de objetos del proyecto (POM), Maven puede administrar la construcción, informes y documentación de un proyecto a partir de una pieza central de información.

Para que lo utilizamos?

- ▶ Maven simplifica y estandariza el proceso de construcción del proyecto. Maneja la colaboración en equipo, la compilación, la distribución, la documentación y las tareas independientes sin problemas. Maven aumenta la reutilización y también se encarga de la mayoría de las tareas relacionadas con la construcción. Funciona en muchos pasos, como agregar archivos jar a la biblioteca del proyecto, crear informes y ejecutar casos de prueba de JUnits, crear archivos Jar, War, Ear para el proyecto e incluso muchas más cosas.
- ▶ Un aspecto muy importante de Maven es el propósito de los repositorios para administrar archivos jar.
- ▶ El Maven se puede utilizar además en la construcción y gestión de proyectos escritos en lenguajes como C #, ruby y otros lenguajes de programación.

https://mvnrepository.com/

Indexed Artifacts (24.3M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages
- Logging Frameworks
- Logging Bridges
- Mail Clients

What's New in Maven

256

ORMLite Android

[com.j256.ormlite](#) » [ormlite-android](#) » 5.7

Lightweight Object Relational Model (ORM) Android classes

Last Release on Nov 8, 2021

92 usages

ISC

256

ORMLite Core

[com.j256.ormlite](#) » [ormlite-core](#) » 5.7

Lightweight Object Relational Model (ORM) for persisting objects to SQL databases.

Last Release on Nov 8, 2021

92 usages

ISC

256

ORMLite JDBC

[com.j256.ormlite](#) » [ormlite-jdbc](#) » 5.7

Lightweight Object Relational Model (ORM) JDBC classes

Last Release on Nov 8, 2021

37 usages

ISC

{i}

Injekt Core Library

[com.ivianuu.injekt](#) » [injekt-core-jvm](#) » 0.0.1-dev654

Next gen DI for kotlin

Last Release on Nov 7, 2021

13 usages

Apache

{i}

Injekt Core Library

[com.ivianuu.injekt](#) » [injekt-core](#) » 0.0.1-dev654

Next gen DI for kotlin

Last Release on Nov 7, 2021

12 usages

Apache

{i}

Injekt Common Library

[com.ivianuu.injekt](#) » [injekt-common](#) » 0.0.1-dev654

Next gen DI for kotlin

Last Release on Nov 7, 2021

10 usages

Apache



Your new development career awaits. Check out the latest listings.

ADS VIA CARBON

Indexed Repositories (1351)

- Central
- Sonatype
- Atlassian
- Spring Plugins
- Hortonworks
- Spring Lib M
- JCenter
- Atlassian Public
- JBossEA
- BeDataDriven

Popular Tags

amazon android apache api application assets aws build build-system camel client clojure cloud config data database eclipse example extension framework github gradle groovy http io jboss library logging maven module osgi persistence plugin repository rest slang scala sdk security server service spring starter streaming testing tools ui web webapp

Maven Repository: servlet

mvnrepository.com/search?q=servlet

MVNREPOSITORY

servlet

Search

Categories | Popular | Contact Us

Repository

Central 1.9k

Sonatype 706

Spring Plugins 539

Spring Lib M 534

JBoss Releases 197

Geomajas 115

IBiblio 101

Spring Lib Release 80

Group

org.mobicents 283

org.apache 178

com.github 85

org.jboss 81

org.eclipse 54

org.glassfish 47

com.aoapps 38


org.wso2 32

Category

Web App 350

Found 2593 results

Sort: relevance | popular | newest



1. **Java Servlet API**


[javax.servlet](#) » [javax.servlet-api](#)

15,296 usages

GPL CDDL

Java Servlet API

Last Release on Apr 20, 2018



2. **JavaServlet(TM) Specification**


[javax.servlet](#) » [servlet-api](#)

11,453 usages

GPL GPL CDDL

JavaServlet(TM) Specification

Last Release on Apr 17, 2008



3. **Jetty :: Servlet Handling**


[org.eclipse.jetty](#) » [jetty-servlet](#)

2,157 usages

EPL Apache

Jetty Servlet Container

Last Release on Oct 12, 2021




4. **Java Servlet 3.1 API**

[org.jboss.spec.javax.servlet](#) » [jboss-servlet-api-3.1-spec](#)

494 usages

GPL



You deserve the cloud database that's built to adapt. Try MongoDB Atlas.

ADS VIA CARBON

Indexed Repositories (1351)

Central

Sonatype

Atlassian

Spring Plugins

Hortonworks

Spring Lib M

JCenter

Atlassian Public

JBossEA

BeDataDriven

Gradle

- ▶ Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando **Groovy o Kotlin DSL (Domain Specific Language)**.
- ▶ **Gradle tiene una gran flexibilidad y nos deja hacer usos otros lenguajes y no solo de Java**, también cuenta con un sistema de gestión de dependencias muy estable. Gradle es altamente personalizable y rápido ya que completa las tareas de forma rápida y precisa reutilizando las salidas de las ejecuciones anteriores, sólo procesar las entradas que presentan cambios en paralelo.
- ▶ Además es el **sistema de compilación oficial para Android** y cuenta con soporte para diversas tecnologías y lenguajes.
- ▶ **Gradle** cuenta con paquetes para ser implementado en cualquier plataforma su gran versatilidad permite trabajar con monorepos o multirepositorios, modelando, sistematizando y construyendo soluciones exitosas, de forma rápida y precisa.

Características de Gradle

- **Depuración colaborativa:** Permite compartir los resultados de la compilación para resolver en equipo de forma eficiente posibles problemas que aparezcan.
- **Construcción incremental:** Valida en el proceso de compilación si la entrada, salida o implementación de una tarea ha cambiado, en caso de no existir algún cambio la considera actualizada y no se ejecuta.
- **Diseño de repositorio personalizado:** Podremos tratar prácticamente cualquier estructura de directorios del sistema de archivos como un repositorio de Artifacts.
- **Dependencias transitivas:** Es uno de los principales beneficios que ofrece al utilizar la gestión de dependencias ya que se encarga de descargar y administrar las dependencias transitivas.
- **Soporte a Groovy y Scala incorporado:** Compatibilidad con los proyectos de Groovy, permitiendo trabajar con código Groovy o código Scala e inclusive desarrollar código mixto Java y Groovy o Java y Scala.
- **Compilación incremental para Java:** En caso de que el código fuente o la ruta de clase cambien, Gradle cuenta con la capacidad para detectar todas las clases que se vean afectadas por dicho cambio y procederá a recompilarlas.
- **Embalaje y distribución de JAR, WAR y EAR:** Cuenta con herramientas para empaquetar el código basado en JVM (Java Virtual Machine) en archivos de archivo comunes.
- **Integración con Android Studio:** Android Studio no cuenta con un generador interno, sino que delega todas las tareas de compilación en Gradle, garantizando la corrección en todas las construcciones, ya sea que se ejecuten desde Android Studio, la línea de comandos o un servidor de construcción de integración continua.

Características de Gradle

- **Soporte de MS Visual C ++ y GoogleTest:** Gradle acepta la construcción con el compilador de Visual C de Microsoft en Windows. (VS 2010, VS 2013 y VS 2015 compatibles), así como también realizar pruebas de aplicaciones C con GoogleTest.
- **Publicar en repositorios Ivy y Maven:** Permite publicar Artifacts en repositorios Ivy con diseños de directorios completamente personalizables. De igual modo, sucede con Maven en Bintray o Maven Central.
- **TestKit para pruebas funcionales:** Permite la ejecución programática de builds inspeccionando los resultados de compilación, ésta es una prueba de compatibilidad entre versiones.
- **Distribuciones personalizadas:** En Gradle cada distribución cuenta con un directorio init.d en el que se pueden colocar scripts personalizados que pre configuran su entorno de compilación.
- **Lee el formato POM:** Es compatible con el formato de metadatos POM, por lo que es posible recuperar dependencias de cualquier repositorio compatible con Maven.
- **Compara builds:** Resalta de forma rápida las diferencias entre compilaciones, lo que hace que el análisis de la causa raíz sea mucho más rápido y eficaz.
- **Compilador daemon:** Gradle crea un proceso de daemon que se reutiliza dentro de una compilación de múltiples proyectos, cuando necesita bifurcar el proceso de compilación, mejorando la velocidad de compilación.
- **Personalizar y extender escaneos:** Ofrece la opción de agregar sus propios datos para construir escaneos como etiquetas, valores y enlaces, integrando escaneos de compilación en la cadena de herramientas.
- **Caché de dependencia de terceros:** Las dependencias de repositorios remotos se descargan y almacenan en caché localmente, las compilaciones posteriores utilizan los artifacts almacenados en caché para evitar el tráfico de red innecesario.

¿Cuál debo elegir Maven o Gradle?

- La clave está en la flexibilidad, si el proyecto se puede gestionar con metas standard o con poca personalización entonces Maven es suficiente, por otro lado si el proyecto requiere diferentes comportamientos en función de múltiples variables en tiempo de construcción quizá Gradle responda más eficazmente a esos requerimientos.

Que es una Api Rest

- ▶ Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. El informático Roy Fielding es el creador de la transferencia de estado representacional (REST).

Que es una Api Rest

- ▶ Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta). Por ejemplo, el diseño de una API de servicio meteorológico podría requerir que el usuario escribiera un código postal y que el productor diera una respuesta en dos partes: la primera sería la temperatura máxima y la segunda, la mínima.

REST

- ▶ REST (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones que tener en cuenta en la arquitectura software que usaremos para crear aplicaciones web respetando HTTP.

JSON

- ▶ JSON es el lenguaje de programación más popular, ya que tanto las máquinas como las personas lo pueden comprender y no depende de ningún lenguaje, a pesar de que su nombre indique lo contrario.

```
{
  "business_id": "PK6aSizckHFWk8i0xt5DA",
  "full_address": "400 Waterfront Dr E\nHomestead\nHomestead, PA 15120",
  "hours": {},
  "open": true,
  "categories": [
    "Burgers",
    "Fast Food",
    "Restaurants"
  ],
  "city": "Homestead",
  "review_count": 5,
  "name": "McDonald's",
  "neighborhoods": [
    "Homestead"
  ],
  "longitude": -79.910032,
  "state": "PA",
  "stars": 2,
  ...
}
```

Para que una API se considere de RESTful, debe cumplir los siguientes criterios:

- ▶ Arquitectura cliente-servidor compuesta de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP.
- ▶ Comunicación entre el cliente y el servidor sin estado, lo cual implica que no se almacena la información del cliente entre las solicitudes de GET y que cada una de ellas es independiente y está desconectada del resto.
- ▶ Datos que pueden almacenarse en caché y optimizan las interacciones entre el cliente y el servidor.

Para que una API se considere de RESTful, debe cumplir los siguientes criterios:

Una interfaz uniforme entre los elementos, para que la información se transfiera de forma estandarizada. Para ello deben cumplirse las siguientes condiciones:

- ▶ Los recursos solicitados deben ser identificables e independientes de las representaciones enviadas al cliente.
- ▶ El cliente debe poder manipular los recursos a través de la representación que recibe, ya que esta contiene suficiente información para permitirlo.
- ▶ Los mensajes autodescriptivos que se envíen al cliente deben contener la información necesaria para describir cómo debe procesarla.
- ▶ Debe contener hipertexto o hipermedios, lo cual significa que cuando el cliente acceda a algún recurso, debe poder utilizar hipervínculos para buscar las demás acciones que se encuentren disponibles en ese momento.

Para que una API se considere de RESTful, debe cumplir los siguientes criterios:

- ▶ Un sistema en capas que organiza en jerarquías invisibles para el cliente cada uno de los servidores (los encargados de la seguridad, del equilibrio de carga, etc.) que participan en la recuperación de la información solicitada.
- ▶ Código disponible según se solicite (opcional), es decir, la capacidad para enviar códigos ejecutables del servidor al cliente cuando se requiera, lo cual amplía las funciones del cliente.

Métodos en una API REST

- ▶ **GET** es usado para recuperar un recurso.
- ▶ **POST** se usa la mayoría de las veces para crear un nuevo recurso. También puede usarse para enviar datos a un recurso que ya existe para su procesamiento. En este segundo caso, no se crearía ningún recurso nuevo.
- ▶ **PUT** es útil para crear o editar un recurso. En el cuerpo de la petición irá la representación completa del recurso. En caso de existir, se reemplaza, de lo contrario se crea el nuevo recurso.
- ▶ **PATCH** realiza actualizaciones parciales. En el cuerpo de la petición se incluirán los cambios a realizar en el recurso. Puede ser más eficiente en el uso de la red que PUT ya que no envía el recurso completo.
- ▶ **DELETE** se usa para eliminar un recurso.

Otras operaciones menos comunes pero también destacables son

- ▶ **HEAD** funciona igual que GET pero no recupera el recurso. Se usa sobre todo para testear si existe el recurso antes de hacer la petición GET para obtenerlo (un ejemplo de su utilidad sería comprobar si existe un fichero o recurso de gran tamaño y saber la respuesta que obtendríamos de la API REST antes de proceder a la descarga del recurso).
- ▶ **OPTIONS** permite al cliente conocer las opciones o requerimientos asociados a un recurso antes de iniciar cualquier petición sobre el mismo.

¿Cuáles son las ventajas de utilizar una API Rest?

► Separación entre cliente y servidor

Una de las ventajas de utilizar el modelo API Rest es la separación entre las aplicaciones de front-end y back-end.

Esto es importante para proteger el almacenamiento de datos, ya que no existe un tratamiento de las reglas comerciales, es decir, **solo se intercambia informaciones** sea para recuperar datos, o para insertar o eliminar nuevos registros.

Más visibilidad, confiabilidad y escalabilidad

Debido a la separación cliente / servidor, hay mucha más facilidad durante el desarrollo de la aplicación. Esto se debe a que se puede escalar fácilmente, ya que no hay dificultad para vincular recursos.

Como cada solicitud se realiza de forma única e independiente, es posible cambiar una solicitud a otro DNS, sin interferir con la aplicación.

En otras palabras, **una API Rest permite que la aplicación acceda a bases de datos desde diferentes servidores**, lo que a menudo es importante para el desarrollo en aplicaciones grandes. Por lo tanto, su uso garantiza una mayor visibilidad y credibilidad a la hora de utilizar estos recursos.

Multiplataforma

- ▶ Las requisiciones HTTP realizadas en API Rest devuelven datos en formato JSON. Cabe mencionar que existen otros posibles formatos de retorno, como XML, sin embargo, JSON es el más utilizado. Por lo tanto, la mayoría de los sitios que trabajan con este modelo reciben este formato de datos.
- ▶ **Esta característica es fundamental para el desarrollo de aplicaciones multiplataforma.** Eso se debe a que, al recibir los datos en este formato, la camada front-end de la aplicación es capaz de realizar el tratamiento adecuado para mostrar los resultados según el tipo de dispositivo utilizado.
- ▶ El uso de **Rest API es importante para agregar varias funciones al sitio.** Sus características permiten la integración con diferentes aplicaciones; entre ellos, redes sociales y sistemas de pago.
- ▶ Por eso, es una tecnología que garantiza una mayor fiabilidad y escalabilidad, además de facilitar el desarrollo de aplicaciones multiplataforma.



Client sends a **request**



HTTP methods



Server sends a **response**

