
FULLSTACK
KNIGHTS
Spring Boot on FIRE
Tribu Fullstack Knights
spring boot
Iniciativa del Gremio Java
Francisco Seguel A.
Julio 2023

¿Qué aprendemos hoy?

AGENDA

Introducción

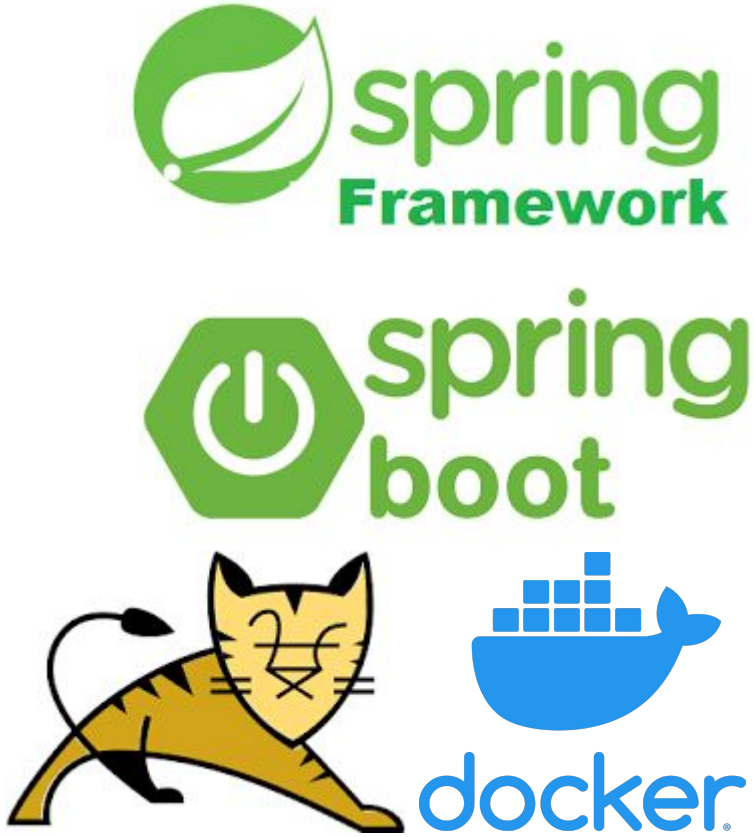
- Framework Spring
- ¿Por qué Spring?
- Spring Framework - CORE
- Spring - Todos los Módulos
- What Spring can do?
- De Spring a Spring Boot
- Spring Boot
- Beneficios de usar Framework Spring
- Desventajas de usar Spring-boot

Conceptos Básicos

- Instalación Spring-Boot
  - Clases de Arranque de la aplicación
  - Compilación
  - Ejecutar
- Principio de Dependencia Injection
- Principio de Inversion of Control (IoC)
- Programación orientada a aspecto
- Servicios REST | Web RESTful | APIs REST

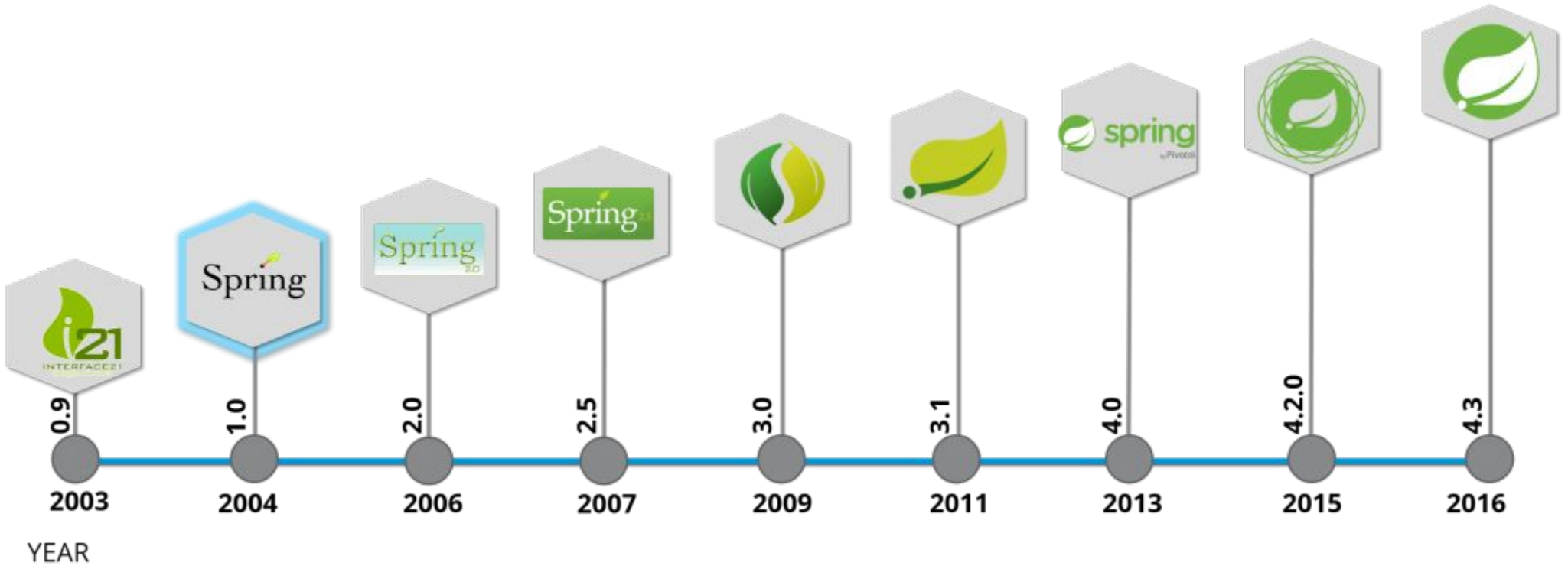
Práctico

- Crear nuestros primeros servicios web RESTfull.
- Spring con JPA
  - Entidades
  - Repositorios
- Pruebas Automatizadas
  - Pruebas Funcionales
  - Pruebas de Carga
- Docker es el aliado perfecto.



Introducción

Spring Framework



La primera versión de Spring se lanzó en junio de 2003, aunque el gran lanzamiento se hizo en Marzo de 2004, con la versión 1.0. Meses mas tarde, en concreto el 21 de Junio de 2004, Rod Johnson, creador de Spring, publicó el libro: "J2EE development without EJB". 15-06-2011

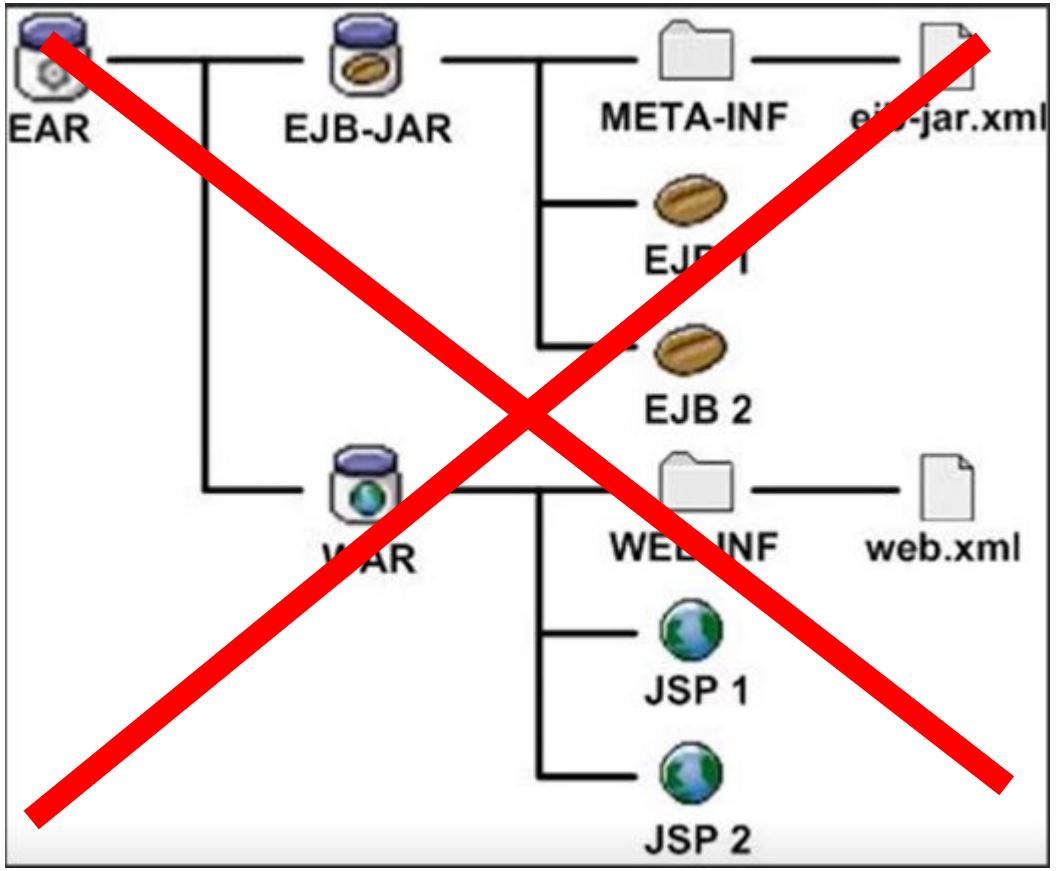
https://javadesde0.com/un-poco-de-historia-sobre-spring/

¿Por qué Spring?

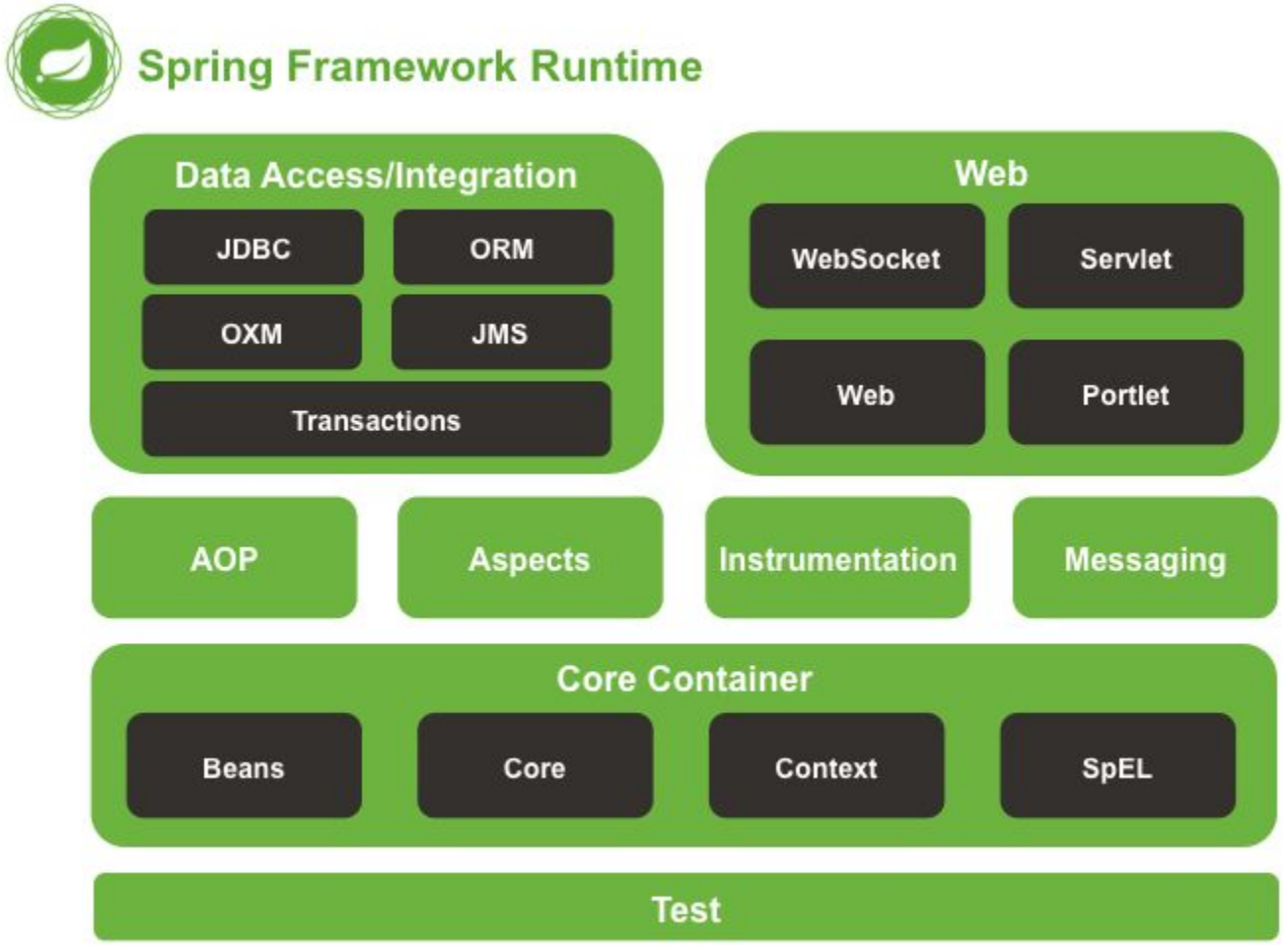
spring
Framework

spring
boot

Menos configuraciones



Spring Framework - CORE



https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/overview.html

Spring - Todos los Módulos

Spring Boot
Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

Spring Framework
Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

Spring Data
Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.

Spring Cloud
Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

Spring Cloud Data Flow
Provides an orchestration service for composable data microservice applications on modern runtimes.

Spring Security
Protects your application with comprehensive and extensible authentication and authorization support.

Spring Statemachine
Provides a framework for application developers to use state machine concepts with Spring applications.

Spring Web Flow
Supports building web applications that feature controlled navigation, such as checking in for a flight or applying for a loan.

Spring Web Services
Facilitates the development of contract-first SOAP web services.

Spring Authorization Server
Provides a secure, light-weight, and customizable foundation for building OpenID Connect 1.0 Identity Providers and OAuth2 Authorization Server products.

Spring for GraphQL
Spring for GraphQL provides support for Spring applications built on GraphQL Java.

Spring Session
Provides an API and implementations for managing a user's session information.

Spring Integration
Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.

Spring HATEOAS
Simplifies creating REST representations that follow the HATEOAS principle.

Spring REST Docs
Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.

Spring Batch
Simplifies and optimizes the work of processing high-volume batch operations.

Spring AMQP
Applies core Spring concepts to the development of AMQP-based messaging solutions.

Spring Flo
Provides a JavaScript library that offers a basic embeddable HTML5 visual builder for pipelines and simple graphs.

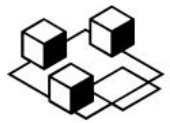
Spring for Apache Kafka
Provides Familiar Spring Abstractions for Apache Kafka.

Spring LDAP
Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.

Spring Shell
Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.

https://spring.io/projects/

What Spring can do?



Microservices

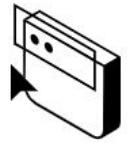
Quickly deliver production-grade features with independently evolvable microservices.

Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.

Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.

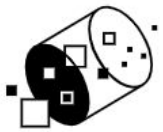


Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.

Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



Event Driven

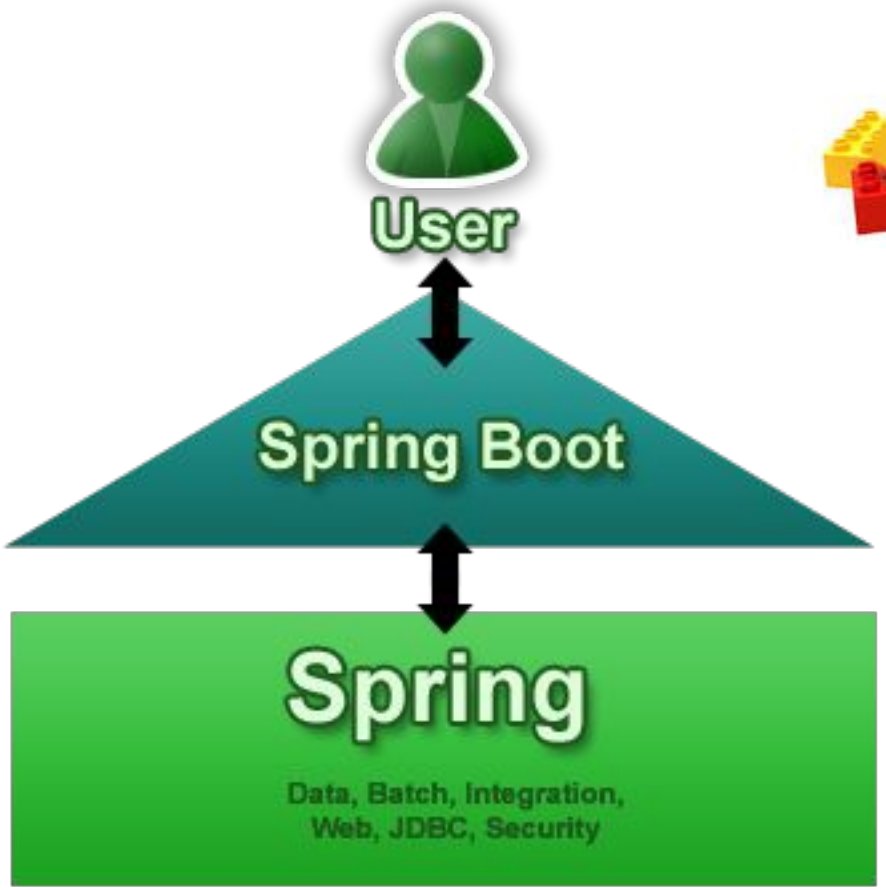
Integrate with your enterprise. React to business events. Act on your streaming data in realtime.

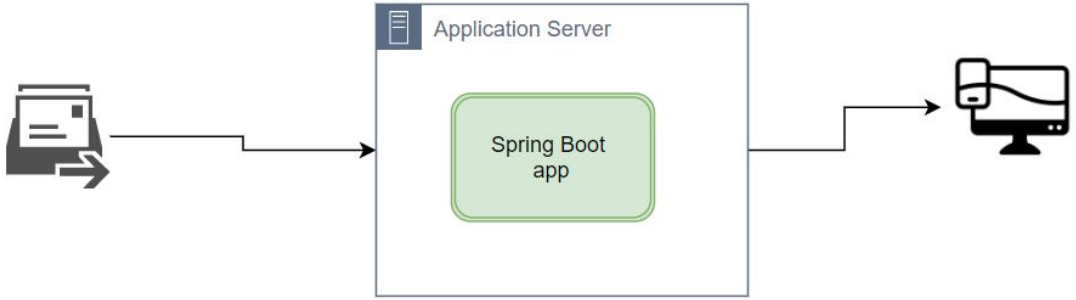
Batch

Automated tasks. Offline processing of data at a time to suit you.

https://spring.io/why-spring

De Spring a Spring Boot





Spring Boot

Preparing for Spring Boot 3.0

ENGINEERING | PHIL WEBB | MAY 24, 2022 | 51 COMMENTS

Spring Boot 2.0 was the first release in the 2.x line and was published on Feburary 28th 2018. We've just released Spring Boot 2.7 which means that, so far, we've been maintaining the 2.x line for just over 4 years. In total we've published 95 distinct releases over that timeframe!

The entire Spring team, and many in our community of contributors, are now preparing for the next generation of Spring. We are planning to release Spring Boot 3.0 in November 2022. This next major revision will be based on Spring Framework 6.0 and will require Java 17 or above. It will also be the first version of Spring Boot that makes use of Jakarta EE 9 APIs (`jakarta.*`) instead of EE 8 (`javax.*`).

The next six months offer an ideal opportunity to prepare your own projects for this major release. In this blog post we'll cover some of the things that you can do *today* to make any future migration as painless as possible.

Upgrade to Java 17

Spring Boot 3.0 will require Java 17, but you don't need to wait until that release to upgrade to the latest LTS Java version. Any recent Spring Boot 2.x release will work really well with Java 17. You can also make use of Java 17 features (such as records) in your own codebases.

We highly recommend that you upgrade your JDK today if at all possible.

https://spring.io/blog/2022/05/24/preparing-for-spring-boot-3-0

Spring Boot 2.x versions - Java version compatability

Based on the reference documenation from https://docs.spring.io/spring-boot/docs/.

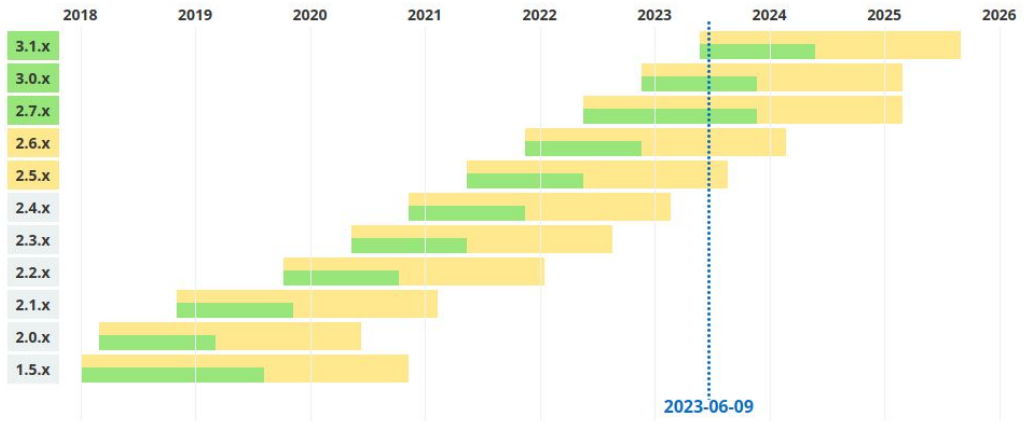
```
2.0.9.RELEASE  - Java 8 or 9
2.1.18.RELEASE - Java 8 to 12
2.2.13.RELEASE - Java 8 to 15
2.3.9.RELEASE  - Java 8 to 15
2.4.13.RELEASE - Java 8 to 16
2.5.14.RELEASE - Java 8 to 18
2.6.14.RELEASE - Java 8 to 19
2.7.7.RELEASE  - Java 8 to 19
```

https://gist.github.com/jameskennard/a421e09e08e9a5e973ce58cd6f174075

Spring Boot 3.1.0

OVERVIEW | LEARN | SUPPORT | SAMPLES

Branch	Initial Release	End of Support	End Commercial Support *
3.1.x	2023-05-18	2024-05-18	2025-08-18
3.0.x	2022-11-24	2023-11-24	2025-02-24
2.7.x	2022-05-19	2023-11-18	2025-02-18
2.6.x	2021-11-17	2022-11-24	2024-02-24
2.5.x	2021-05-20	2022-05-19	2023-08-24
2.4.x	2020-11-12	2021-11-18	2023-02-23
2.3.x	2020-05-15	2021-05-20	2022-08-20
2.2.x	2019-10-16	2020-10-16	2022-01-16
2.1.x	2018-10-30	2019-10-30	2021-01-30
2.0.x	2018-03-01	2019-03-01	2020-06-01
1.5.x	2017-01-30	2019-08-06	2020-11-06



https://spring.io/projects/spring-framework#support

Beneficios de usar Framework Spring

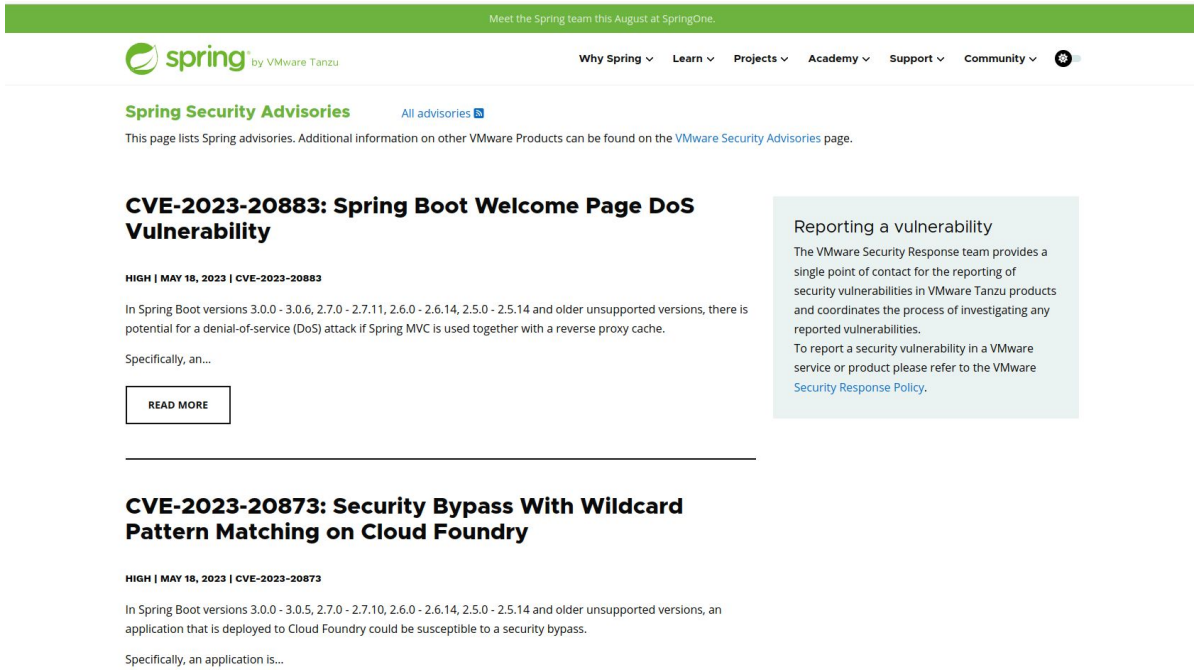
- **Spring está organizado de forma modular.** A pesar de la cantidad de paquetes y clases que tiene, solo debemos ocuparnos de aquellos que necesitemos para nuestro desarrollo e ignorar el resto.
- **Utiliza algunas de las tecnologías existentes**, como varios frameworks ORM, JEE, temporizadores Quartz y JDK, frameworks de registro y otras tecnologías de visualización.
- **Probar una aplicación escrita con Spring es un proceso simple,** porque el código dependiente del entorno se traslada a este framework. se vuelve más fácil utilizar la inyección de dependencia para hacer pruebas, para ello podemos hacer uso de datos dummies o mocks, para ver las respuestas.
- **El framework web de Spring es un framework MVC web bien diseñado,** que proporciona una excelente alternativa a los frameworks web como Struts u otros frameworks web sobre diseñados o menos populares.
- Spring proporciona una API para traducir excepciones específicas de la tecnología (como por ejemplo las generadas por JDBC, Hibernate o JDO) en excepciones consistentes y no verificadas.
- **Los contenedores de IoC (Inversion of Control) tienden a ser livianos**, especialmente cuando se comparan con los Enterprise JavaBeans (EJB). Esto es ideal para desarrollar y desplegar aplicaciones en máquinas con memoria y recursos limitados.
- **Spring proporciona una interfaz de gestión de transacciones coherente que puede reducirse a una transacción local (utilizando una única base de datos) y ampliarse a transacciones globales.**
- **Permite separar el registro, la auditoría, las transacciones declarativas, la seguridad, el almacenamiento en caché, de la lógica comercial a través de la AOP (Programación Orientada a Aspectos).**
- **Cuenta con plantillas para diversas tecnologías** entre la cuales podemos destacar las siguientes: JDBC, Hibernate y JPA, de forma tal que no hay necesidad de escribir un código extenso, ya que con estas plantillas simplifica el trabajo en cuanto a los pasos básicos a implementar de estas tecnologías.

https://openwebinars.net/blog/conoce-que-es-spring-framework-y-por-que-usarlo/#beneficios-de-utilizar-spring-framework

Desventajas de usar Spring-boot

Spring-boot **es complejo** - al tener gran variedad de componentes e integración con otros lo vuelve complejo. Al tener una historia de años, conocer cómo funciona se vuelve complejo.
Las recetas funcionan pero no explica el porqué.

La curva de aprendizaje es engañosa.
Existen más cosas por detrás



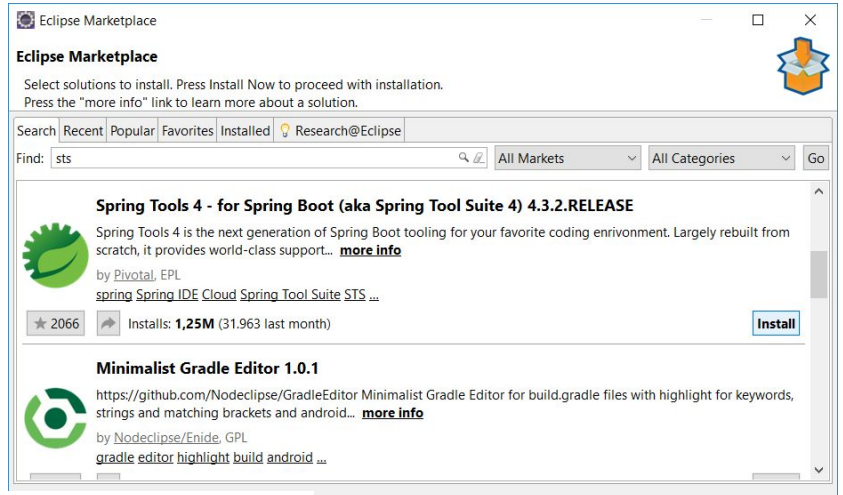
https://spring.io/security/

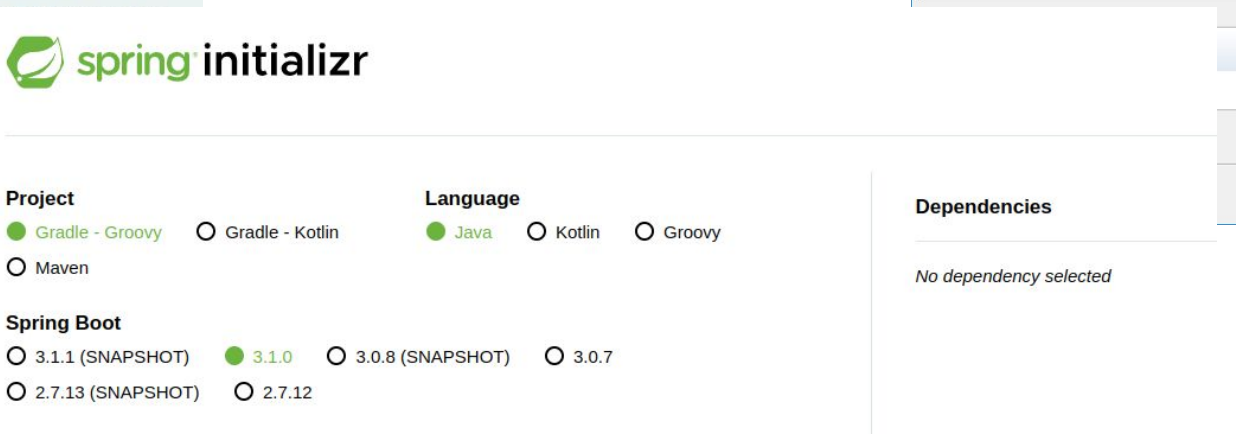
https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Spring%20Framework

Conceptos Básicos

Instalación Spring-Boot

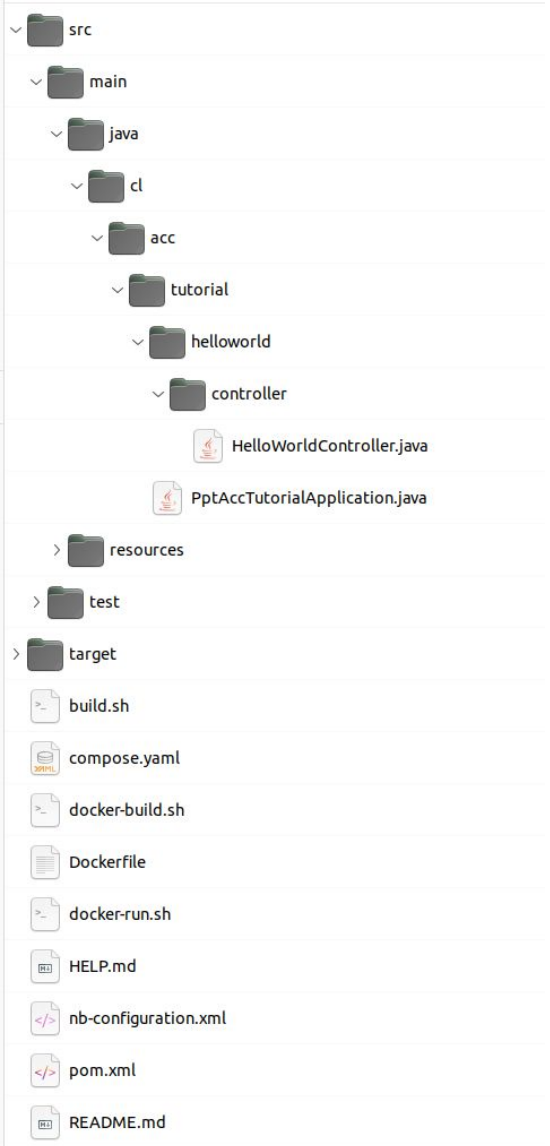
- Instalado *Spring Boot Starter* (que es un Eclipse con Spring Boot integrado). Se puede descargar desde su página oficial: https://spring.io/tools.
- **Accediendo a la página https://start.spring.io y generando el proyecto para descargarlo en tu PC.**
- Instalando *Spring Tools Suite* en Eclipse (Se instala desde Eclipse, como un plugin).

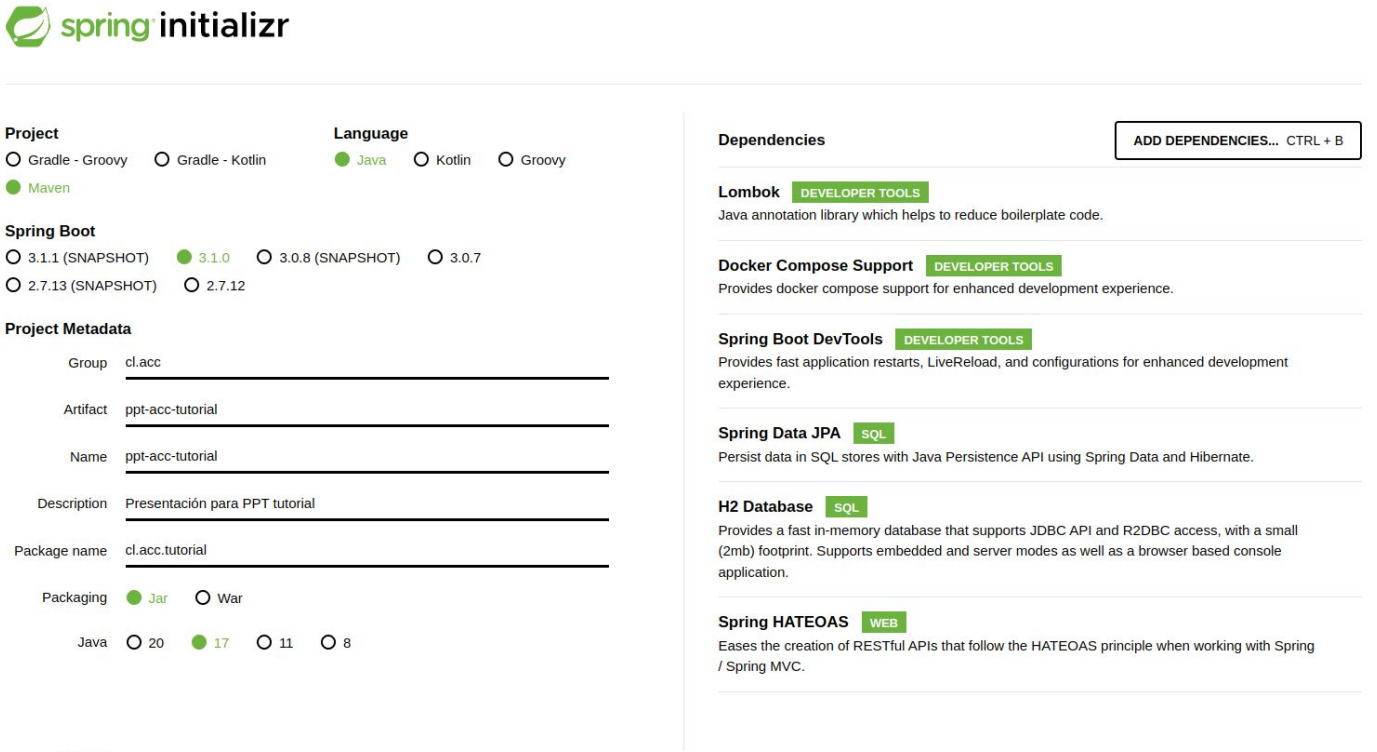




> https://rosamarfil.es/tutoriales/programacion/spring-boot-api-rest-2-crear-una-aplicacion-web-con-spring-boot/#:~:text=Instalar%20Spring%20Boot,-Existen%20dos%20maneras&text=Se%20puede%20descargar%20desde%20su,Eclipse%2C%20como%20un%20plugin).

Clases de Arranque de la aplicación





GitHub

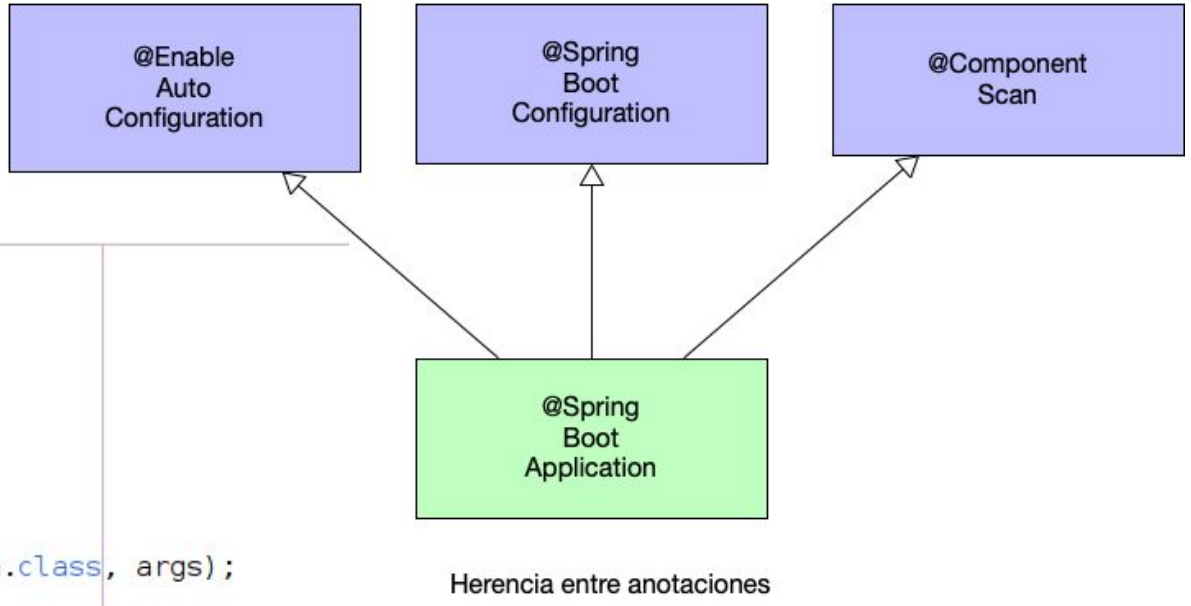
https://github.com/FrankSeguel/ppt-acc-tutorial

git clone git@github.com:FrankSeguel/ppt-acc-tutorial.git
git fetch
git checkout feature/HelloWorld!

@SpringBootApplication anotación clave

@SpringBootApplication

Cuando usamos la anotación @SpringBootApplication esta anotación herede el comportamiento de un conjunto de anotaciones:




```java
package cl.acc.tutorial;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PptAccTutorialApplication {

	public static void main(String[] args) {
		SpringApplication.run(primarySource:PptAccTutorialApplication.class, args);
	}

}
```

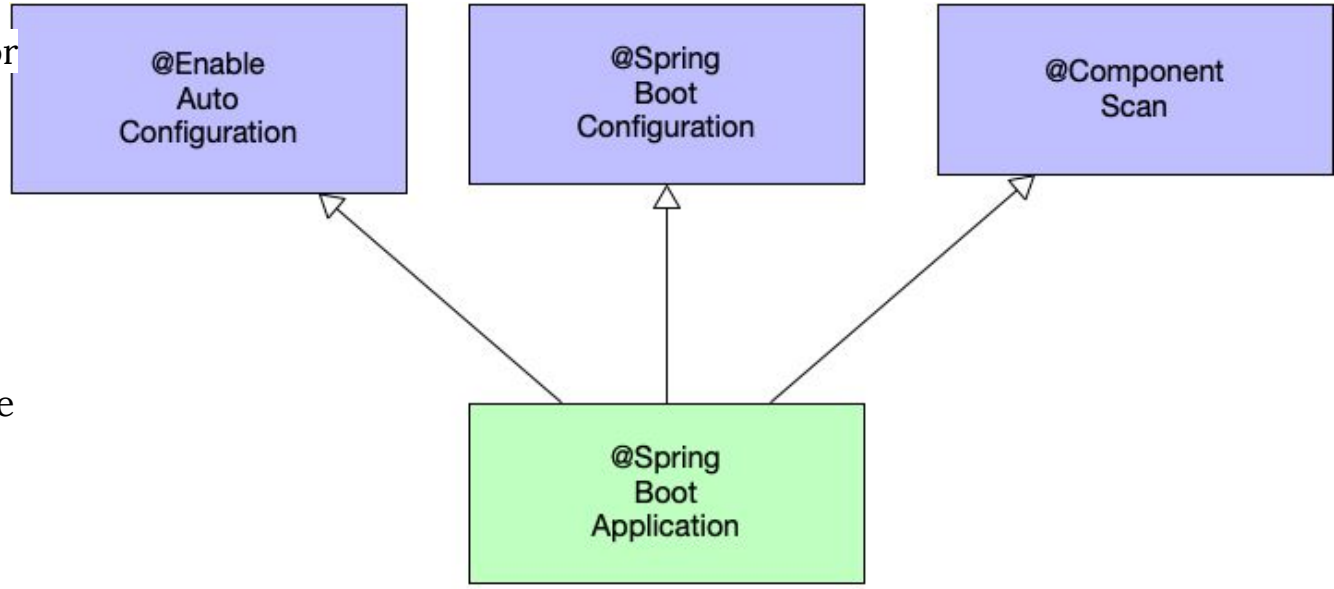
https://www.arquitecturajava.com/springbootapplication-una-anotacion-clave/

Anotación clave

@EnableAutoConfiguration : Esta es una anotación clásica de Spring que se encarga de forma inteligente de intentar configurar Spring de forma automática . Es la anotación encargada de buscar en el Classpath todas las clases con @Entity y registrarlas con el proveedor de persistencia que tengamos. Por lo tanto por eso con Spring Boot es suficiente configurar simplemente el dataSource a nivel application.properties ya que Spring buscará todos las clases.

@SpringConfiguration : Es la anotación que define que el fichero es un fichero de Configuración de Spring .Normalmente esto se solía hacer antiguamente con @Configuratión . La particularidad que tiene @SpringConfiguration es que solo puede haber una en la aplicación

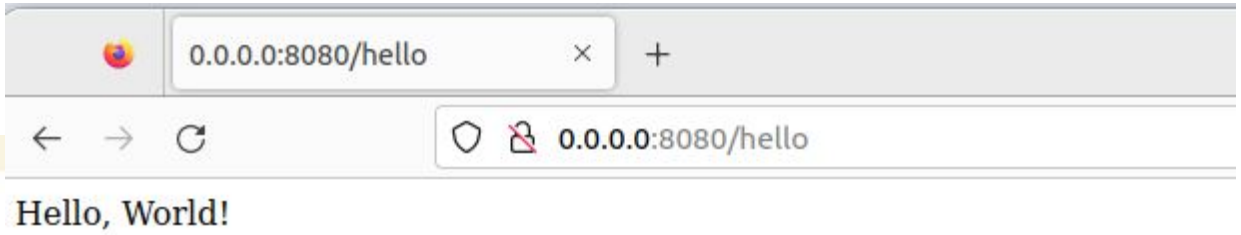
@ConponentScan : Se encarga de revisar los paquetes actuales y registrar de forma automática cualquier @Servie @Repository @Controller etc que la aplicación tenga de forma totalmente transparente para Spring Framework.



Herencia entre anotaciones

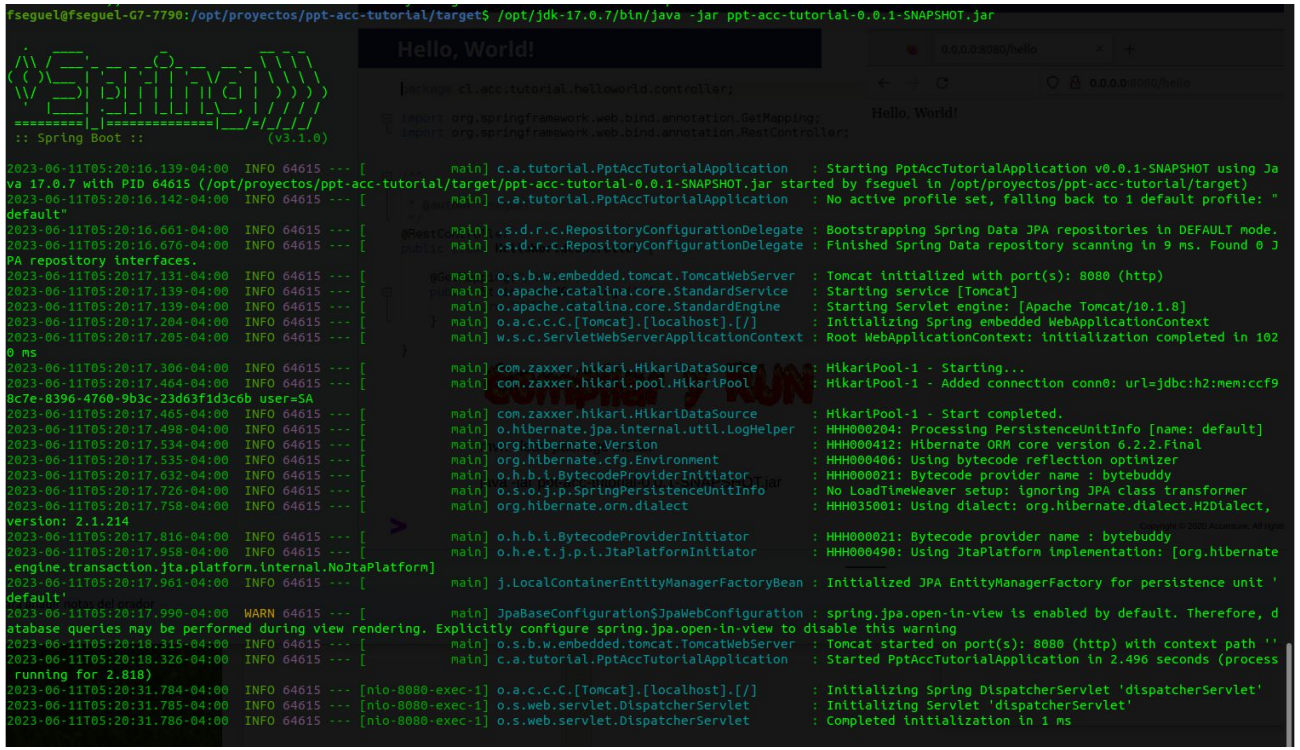
Hello, World!

```java
package cl.acc.tutorial.helloworld.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;


/**
 *
 * @author fseguel
 */
@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String sendGreetings() {
        return "Hello, World!";
    }

}
```



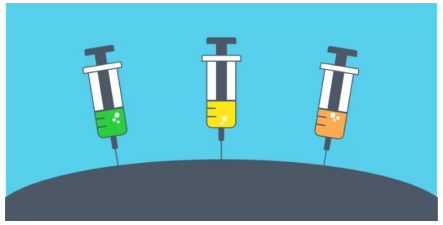
mvn clean package install

java -jar ppt-acc-tutorial-0.0.1-SNAPSHOT.jar



Patrones

Principio de Dependencia Injection



```
@RequestMapping("/usuario")
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;
```

```
 */
public interface UsuarioService {

    public UsuarioDto crearUsuario(UsuarioDto
```

```
@Service
public class UsuarioServiceImpl implements UsuarioService {

    @Autowired
    private UsuarioRepository UsuarioRepository;

    @Autowired
    private Mapper mapper;

    @Override
    public UsuarioDto crearUsuario(UsuarioDto usuario) {
        log.debug("UsuarioServiceImpl - crearUsuario : " + usuario.toString());
        Usuario userEntity = new Usuario();
```

inyección de dependencias (en inglés *Dependency Injection*, DI) es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos. Esos objetos cumplen contratos que necesitan nuestras clases para poder funcionar (de ahí el concepto de *dependencia*). **Nuestras clases no crean los objetos que necesitan, sino que se los suministra otra clase 'contenedora' que inyectará la implementación deseada a nuestro contrato.**[1]

En otras palabras, se trata de un patrón de diseño que se encarga de extraer la responsabilidad de la creación de instancias de un componente para delegarla en otro.

El término fue acuñado por primera vez por Martin Fowler.

dia.org/wiki/Inyecci%C3%B3n_de_dependencias

Principio de Inversión de Dependencias

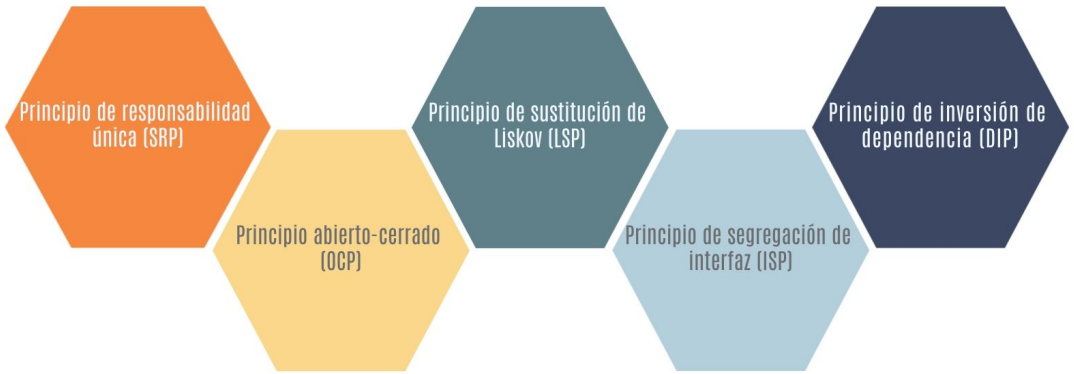
Los módulos de alto nivel no deben depender de los módulos de bajo nivel, sino que ambos deben depender de abstracciones.

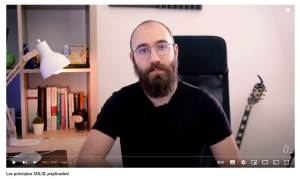
Las implementaciones concretas no deben de depender de otras implementaciones concretas.

A. Las clases de alto nivel no deberían depender de las clases de bajo nivel. Ambas deberían depender de las abstracciones.

B. Las abstracciones no deberían depender de los detalles. Los detalles deberían depender de las abstracciones.

Reducir el acople en capas de software





click-it

https://www.youtube.com/watch?v=2X50sKeBAcQ&t=333s

https://www.youtube.com/watch?v=90uW1_l0Y3I&t=365s

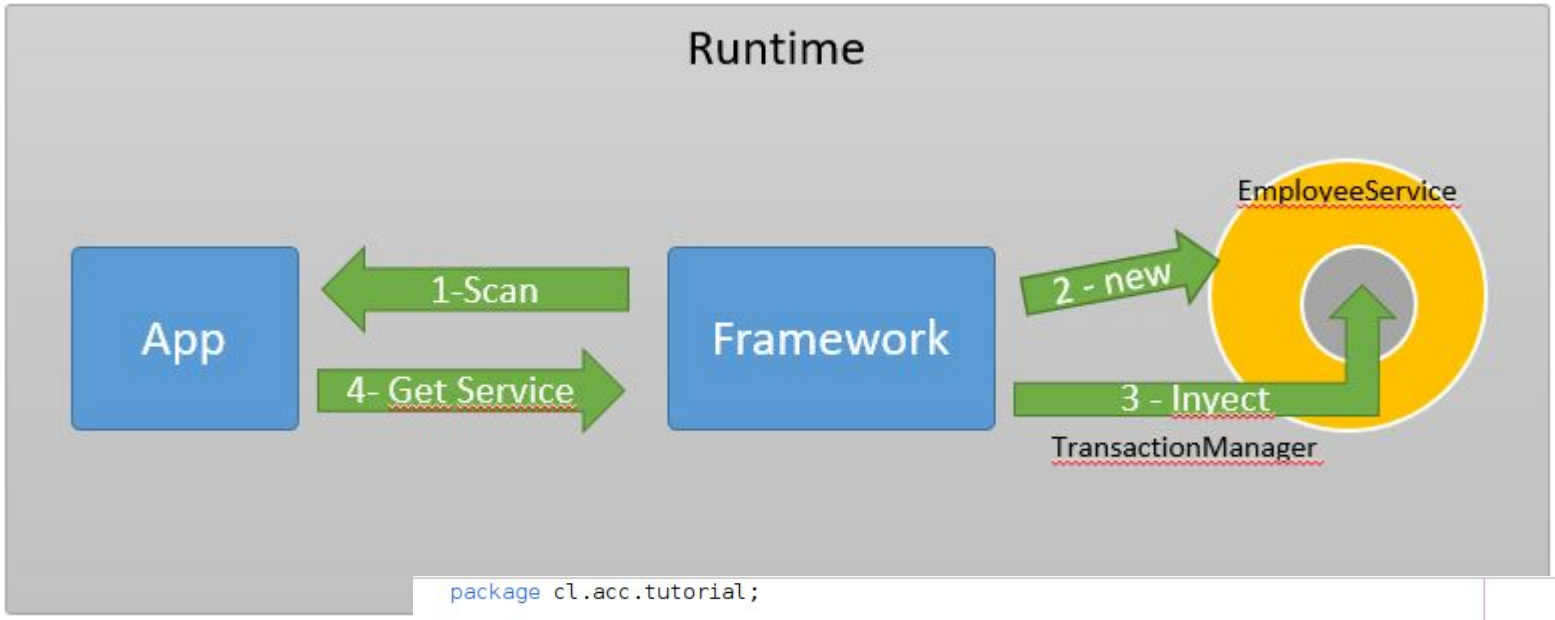
https://es.wikipedia.org/wiki/Principio_de_inversi%C3%B3n_de_la_dependencia

https://devexperto.com/principio-de-inversion-de-dependencias/

Principio de Inversion of Control (IoC)

Inversión de control (Inversion of Control en inglés, IoC) es un principio de diseño de software en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales.

No nos llames; nosotros te llamaremos

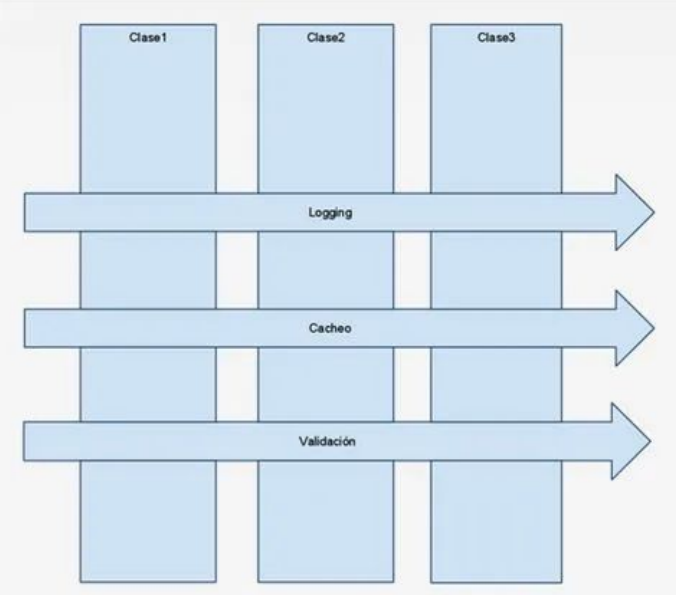



```java
package cl.acc.tutorial;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PptAccTutorialApplication {

	public static void main(String[] args) {
		SpringApplication.run(primarySource:PptAccTutorialApplication.class, args);
	}

}
```

```java
package cl.acc.tutorial.helloworld.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;


/**
 *
 * @author fseguel
 */
@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String sendGreetings() {
        return "Hello, World!";
    }

}
```

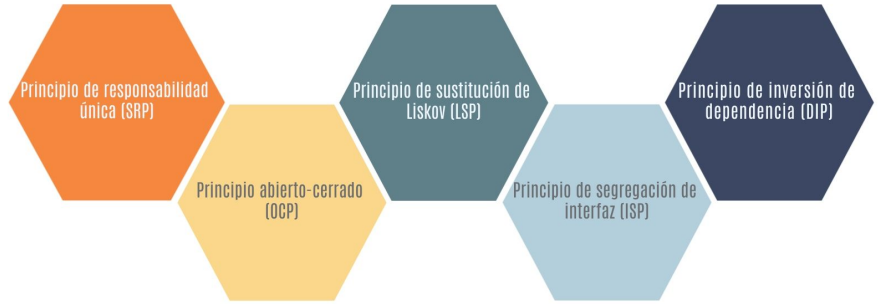
Programación orientada a aspecto

La Programación Orientada a Aspectos o POA (en inglés: aspect-oriented programming) es un paradigma de programación que permite una adecuada modularización de las aplicaciones y posibilita una mejor separación de responsabilidades (Obligación o correspondencia de hacer algo).




```
7    void showUserName(temp) {
8      if (temp.length > 0) {
9        print(temp);
1      } else {
0        throw Exception('User Name is
1    Empty.');
1      }
1    }
2    void main() {
1      var userName = '';
3      try {
1        showUserName(userName);
4      } catch (e) {
1        print('Exception is
5    Handled...');
1      }
6    }
```

S.O.L.I.D.



click-it

```
public class MyHtmlPresentationHandler extends HtmlPresentationHandler {

    @Override
    public void handleException(ChannelContext channelContext, Exception e) {
        if (e instanceof ChannelPolicyException) {
            // See if we have a response to use.
            HttpServletResponse res = getOrgResponse(channelContext);
            if (res != null) {
                try {
                    PrintWriter htmlOut = res.getWriter();
                    htmlOut.println("Policy Check result:");
                    htmlOut.println(e.getMessage());
                } catch (IOException exc) {
                    // do nothing
                }
            }
        } else {
            super.handleException(channelContext, e);
        }
    }

}
```

https://medium.com/@PabloLeonPsi/una-aproximaci%C3%B3n-a-la-programaci%C3%B3n-orientada-a-aspectos-a62d377ebe79

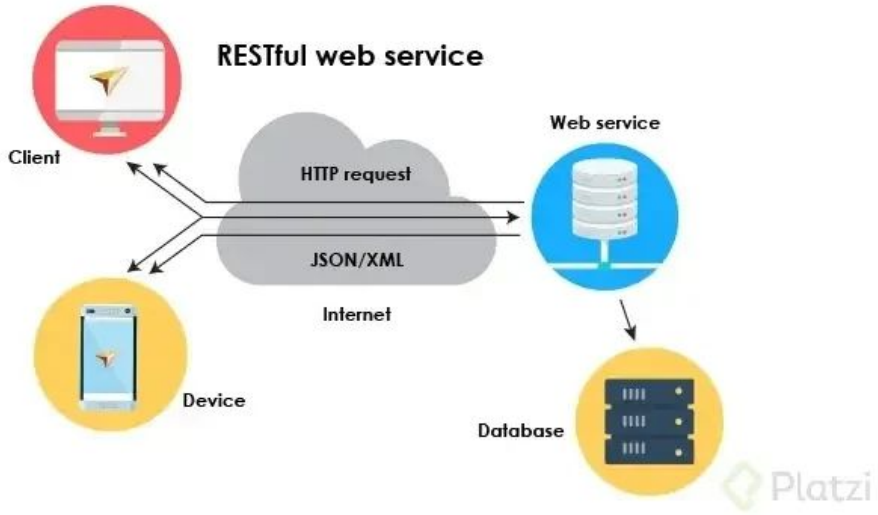
Servicios REST | Web RESTful | APIs REST

Diferencias entre REST y RESTful

REST es una arquitectura para aplicaciones en redes (REpresentational State Transfer). RESTful por otro lado, son programas (a modo de web service o API), basados en REST. Muchas veces se usan ambos términos como sinónimos.

Qué es REST

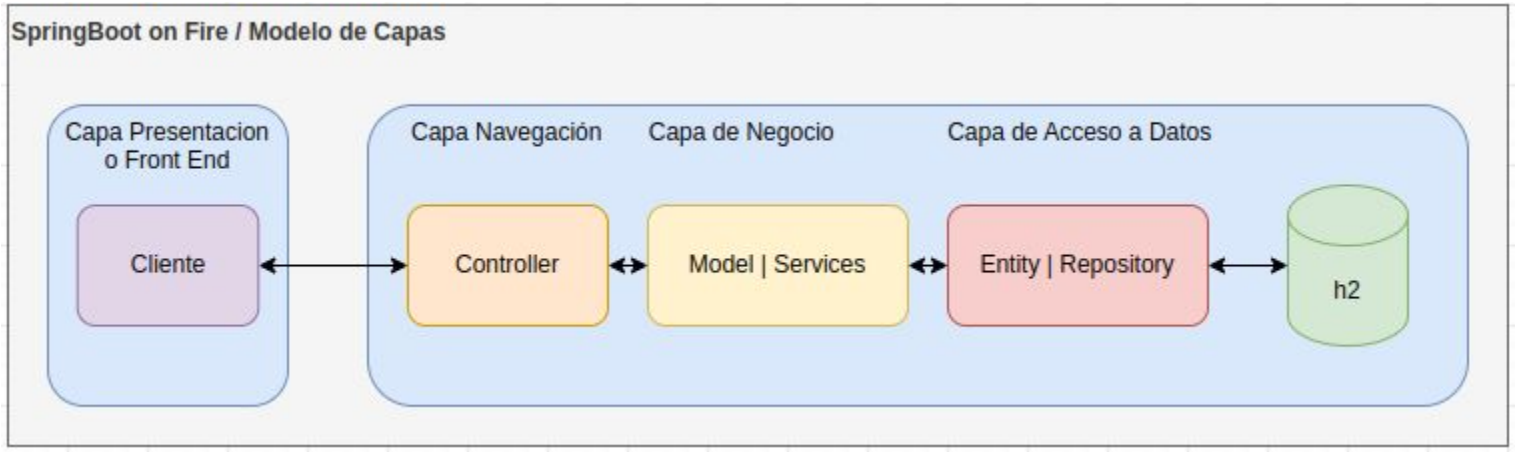
REST es un estilo de arquitectura de software enfocado en el intercambio de recursos y basado en HTTP. Le agrega una capa muy delgada de complejidad y abstracción a HTTP. Mientras que HTTP es transferencia de archivos, REST se basa en la transferencia de recursos.



https://platzi.com/clases/1638-api-rest/21611-que-significa-rest-y-que-es-una-api-restful/#:~:text=Diferencias%20entre%20REST%20y%20RESTful,usan%20ambos%20terminos%20como%20sinonimos.

Práctico

Crear nuestros primeros servicios web RESTfull.



https://github.com/FrankSeguel/ppt-acc-tutorial

git clone git@github.com:FrankSeguel/ppt-acc-tutorial.git
git fetch
git checkout feature/HelloWorld!
git checkout feature/PrimerRestFull
git checkout feature/Aop

EJERCICIO PRÁCTICO

Link de Intereses

- https://www.facebook.com/watch/live/?ref=watch_permalink&v=400583604476228
- https://docs.spring.io/spring-boot/docs/1.4.x/reference/html/common-application-properties.html

Encuesta

https://forms.office.com/pages/responsepage.aspx?id=OT154DkJbUmxKRmO3ZFv65AJF2RCV3RBiXG6gaXulBpUOFhLSE1EUFdZQjJQWVNSRE1HMkFDUVYwRCQlQCN0PWcu