Curso: BackEnd

Tema: SprigBoot

Al finalizar esta sesion, el participante será capaz de:

- Crear un proyecto con Spring Initializr
- Conocer las anotaciones principales de Spring Boot
- Crear un API REST con Spring Boot y mysql
- Realizar pruebas del Rest con Postman.

Temario:

1. Qué es Spring Boot?
2. Anotaciones principales.
3. Crear proyecto con Initializr
4. Configuracion de Entorno & BD
5. Desarrollo de API
6. Pruebas con Postman

Qué es Spring Boot?

Qué es Spring Boot?

Spring Boot es un marco de trabajo (framework) de desarrollo de aplicaciones en Java que simplifica y agiliza la creación y configuración de aplicaciones basadas en Spring. Se enfoca en simplificar la configuración y el proceso de inicio de una aplicación Spring, eliminando gran parte de la complejidad que a menudo está asociada con la configuración manual de una aplicación Spring.

CODIGO
powered by Tecsup

spring-boot-starter-web

SPRING-BOOT-STARTER-WEB

¿Qué es spring-boot-starter-web?

- Es un **starter de Spring Boot** para crear **aplicaciones web y APIs REST**
- Habilita el desarrollo con **Spring MVC**
- Incluye un **servidor web embebido**
- Permite exponer endpoints HTTP fácilmente

Importante!! Es la base para construir APIs REST en Spring Boot

¿Para qué sirve?

- Crear **APIs REST**
- Manejar solicitudes HTTP
- Procesar JSON (request / response)
- Construir aplicaciones web backend
- Exponer servicios para frontend y apps móviles

SPRING-BOOT-STARTER-WEB

Problema que soluciona

Sin el starter tendrías que:

- Configurar servidor web (Tomcat)
- Configurar Spring MVC manualmente
- Configurar serialización JSON
- Manejar request/response a bajo nivel

Con el starter:

- Todo queda **auto-configurado**
- Sigues convenciones
- Menos configuración manual

SPRING-BOOT-STARTER-WEB

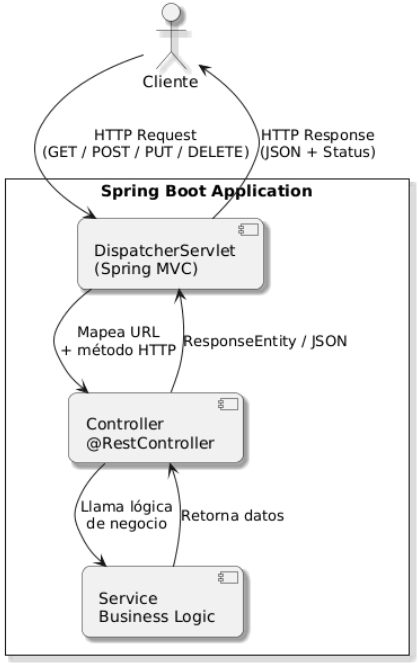
¿Qué incluye internamente?

spring-boot-starter-web incluye:

- **Spring MVC**
- **Servidor embebido Tomcat**
- **Jackson (JSON)**
- **Spring Web**
- **Validaciones HTTP**
- **Auto-configuración Spring Boot**

SPRING-BOOT-STARTER-WEB

Flujo de una request HTTP



SPRING-BOOT-STARTER-WEB

¿Qué es Spring MVC?

- Framework web de Spring
- Modelo **MVC** (Model – View – Controller)
- En APIs REST:
  - Se usa solo **Controller**
  - Se retorna **JSON**, no vistas

Servidor web embebido

Incluye por defecto:
- **Tomcat embebido**
- No necesitas instalar servidor externo
- La app se ejecuta con java -jar

Opcionales:
- Jetty
- Undertow

SPRING-BOOT-STARTER-WEB

Serialización JSON (Jackson)

- Convierte objetos Java ↔ JSON
- Automática al usar @RestController
- Maneja:
  - @RequestBody
  - respuestas JSON

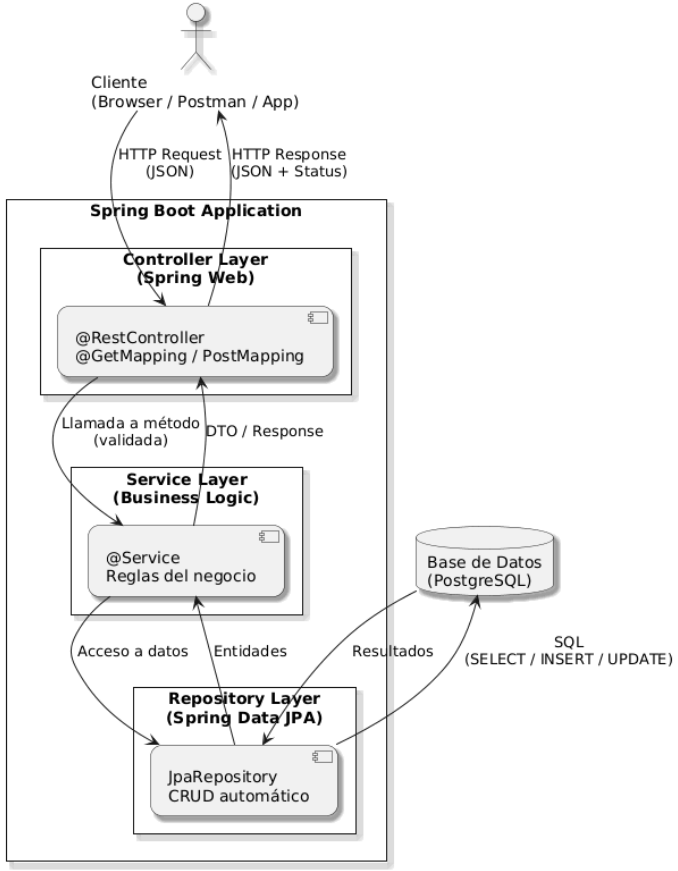
¿Qué habilita automáticamente?

Con el starter se configura:

- DispatcherServlet
- Mapeo de rutas HTTP
- Conversión JSON
- Manejo de errores HTTP
- Validaciones (@Valid)
- Soporte REST

SPRING-BOOT-STARTER-WEB

Arquitectura típica con Spring Web



SPRING-BOOT-STARTER-WEB

¿Qué puedes construir con este starter?

- APIs REST
- Backends para frontend web
- Servicios para apps móviles
- Microservicios HTTP
- Gateways simples

Ventajas principales

- Curva de aprendizaje baja
- Integración perfecta con Spring Boot
- Convención sobre configuración
- Servidor embebido
- Amplia adopción en la industria

SPRING-BOOT-STARTER-WEB

Consideraciones

- Modelo **bloqueante** (thread-per-request)
- No ideal para miles de conexiones concurrentes
- Para alta concurrencia → Spring WebFlux

CODIGO
powered by Tecsup

spring-boot-starter-data-jpa

SPRING-BOOT-STARTER-DATA-JPA

¿Qué es spring-boot-starter-data-jpa?

- Es una **dependencia starter de Spring Boot**
- Integra **Spring Data + JPA + Hibernate**
- Configura automáticamente el acceso a **bases de datos relacionales**
- Elimina configuraciones manuales complejas

Importante!! Un solo starter habilita persistencia completa con JPA

SPRING-BOOT-STARTER-DATA-JPA

¿Para qué sirve?

- Conectar aplicaciones Spring Boot a bases de datos relacionales
- Persistir entidades Java como tablas
- Realizar CRUD sin escribir SQL manual
- Manejar transacciones automáticamente
- Reducir código boilerplate de acceso a datos

SPRING-BOOT-STARTER-DATA-JPA

Problema que soluciona

Sin el starter tendrías que:

- Configurar JDBC manualmente
- Crear DataSource
- Configurar EntityManager
- Configurar TransactionManager
- Manejar conexiones y transacciones

Con el starter:

- Todo queda **auto-configurado**
- Sigues **convenciones de Spring Boot**
- Te enfocas en la lógica de negocio

SPRING-BOOT-STARTER-DATA-JPA

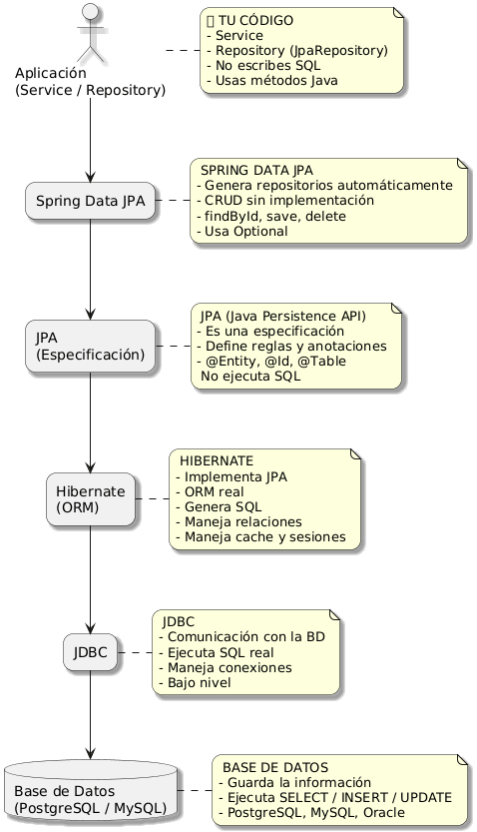
¿Qué incluye internamente?

`spring-boot-starter-data-jpa` NO es una sola librería, es un **conjunto**:

- **Spring Data JPA**
- **Hibernate (ORM)**
- **Jakarta Persistence API (JPA)**
- **Spring ORM**
- **Soporte de transacciones**
- **Auto-configuración Spring Boot**

SPRING-BOOT-STARTER-DATA-JPA

Relación entre tecnologías



SPRING-BOOT-STARTER-DATA-JPA

¿Qué es Spring Data JPA?

- Parte de Spring Data
- Permite crear **repositorios automáticos**
- Elimina la implementación manual de DAOs
- Usa interfaces como `JpaRepository`

¿Qué es JPA?

- **Java Persistence API**
- Es una **especificación**, no una implementación
- Define:
  - `@Entity`
  - `@Id`
  - `@Table`
  - Relaciones (`@OneToMany`, etc.)
- Permite independencia del proveedor ORM

SPRING-BOOT-STARTER-DATA-JPA

¿Qué es Hibernate?

- Es la **implementación más usada de JPA**
- ORM (Object Relational Mapping)
- Traduce:
    - Objetos Java → SQL
    - Filas → Objetos

Maneja:

- Caché
- Transacciones
- Ciclo de vida de entidades

SPRING-BOOT-STARTER-DATA-JPA

¿Qué tipo de bases de datos soporta?

Bases de datos **relacionales**:

- PostgreSQL
- MySQL / MariaDB
- Oracle
- SQL Server
- H2 (en memoria)

OJO!! El starter NO incluye el driver de la BD

SPRING-BOOT-STARTER-DATA-JPA

¿Qué habilita automáticamente?

Al usar el starter, Spring Boot configura:

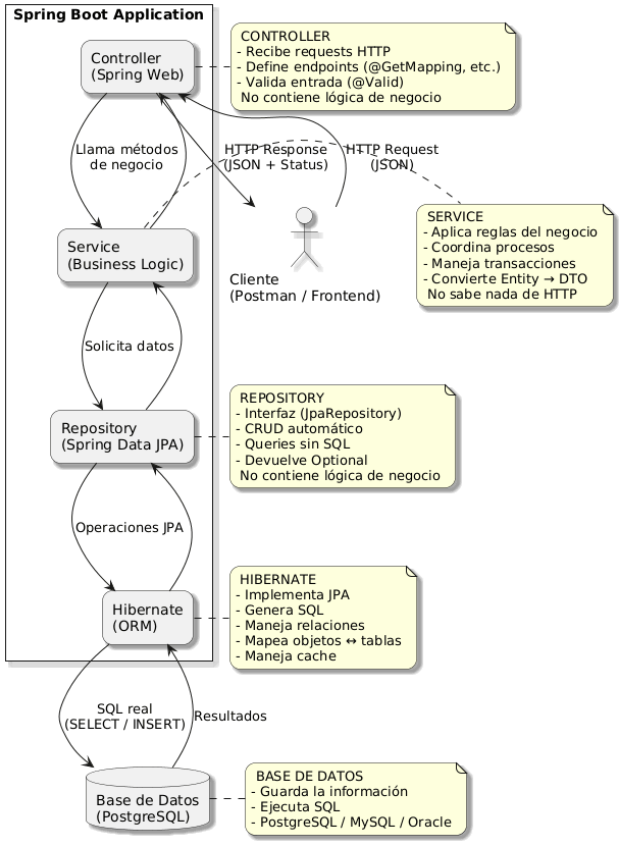
- DataSource
- EntityManager
- TransactionManager
- Repositorios Spring Data
- Escaneo de entidades (@Entity)
- Manejo de transacciones con @Transactional

¿Qué puedes hacer con este starter?

- Crear entidades JPA
- Guardar, actualizar, eliminar datos
- Consultar usando métodos de repositorio
- Crear queries JPQL o SQL nativo
- Usar paginación y sorting
- Manejar transacciones automáticamente

SPRING-BOOT-STARTER-DATA-JPA

Arquitectura típica usando el starter



SPRING-BOOT-STARTER-DATA-JPA

Ventajas principales

- Alta productividad
- Menos código repetitivo
- Integración nativa con Spring Boot
- Transacciones automáticas
- Abstracción de la base de datos
- Código más limpio y mantenible

Ventajas principales

- Alta productividad
- Menos código repetitivo
- Integración nativa con Spring Boot
- Transacciones automáticas
- Abstracción de la base de datos
- Código más limpio y mantenible

spring-boot-starter-data-jpa integra Spring Data, JPA y Hibernate para facilitar el acceso a bases de datos relacionales en Spring Boot.

Anotaciones

SPRING WEB (Spring MVC / REST)

Controladores y rutas:

- **@RestController** = Marca una clase como controlador REST (retorna JSON, no vistas).
- **@Controller** = Controlador MVC tradicional (HTML, vistas).
- **@RequestMapping** = Define la ruta base de un controlador o endpoint.
- **@GetMapping** = Endpoint HTTP GET.
- **@PostMapping** = Endpoint HTTP POST.
- **@PutMapping** = Endpoint HTTP PUT (update completo).
- **@PatchMapping** = Endpoint HTTP PATCH (update parcial).
- **@DeleteMapping** = Endpoint HTTP DELETE.

SPRING WEB (Spring MVC / REST)

Parámetros de request

- **@PathVariable** = Obtiene valores desde la URL (/alumnos/{id}).
- **@RequestParam** = Obtiene parámetros query (?page=1).
- **@RequestBody** = Mapea el body JSON a un objeto Java.
- **@RequestHeader** = Lee headers HTTP.
- **@CookieValue** = Lee cookies.

SPRING WEB (Spring MVC / REST)

Parámetros de request

- **@PathVariable** = Obtiene valores desde la URL (/alumnos/{id}).
- **@RequestParam** = Obtiene parámetros query (?page=1).
- **@RequestBody** = Mapea el body JSON a un objeto Java.
- **@RequestHeader** = Lee headers HTTP.
- **@CookieValue** = Lee cookies.

Responses y status

- **@ResponseBody** = Retorna directamente el cuerpo de la respuesta (JSON).
- **@ResponseStatus** = Define el status HTTP de la respuesta.
- **ResponseEntity<T>** = Control total de status, headers y body.

SPRING WEB (Spring MVC / REST)

Validación / errores

- **@Valid** = Activa validaciones del objeto recibido.
- **@RestControllerAdvice** = Manejo global de errores REST.
- **@ExceptionHandler** = Captura excepciones específicas.

SPRING DATA (CORE)

Repositorios

- **@Repository** = Marca la clase como repositorio (traduce errores de BD).
- **CrudRepository** = CRUD básico.
- **JpaRepository** = CRUD + paginación + sorting.
- **PagingAndSortingRepository** = CRUD con paginación.

Queries

- **@Query** = Define consultas JPQL o SQL nativo.
- **@Param** = Asocia parámetros en queries.
- **nativeQuery = true** = Indica SQL nativo.

JPA (Jakarta Persistence API)

¡¡IMPORTANTE! : Estas anotaciones las usas SIEMPRE con ***spring-boot-starter-data-jpa***

Entidades

- **@Entity** = Marca la clase como entidad JPA.
- **@Table** = Define nombre y propiedades de la tabla.
- **@Id** = Clave primaria.
- **@GeneratedValue** = Generación automática del ID.
- **@Column** = Configura columnas (nullable, length, unique).
- **@Transient** = Campo NO persistente.

JPA (Jakarta Persistence API)

¡¡IMPORTANTE! : Estas anotaciones las usas SIEMPRE con ***spring-boot-starter-data-jpa***

Relaciones

- **@OneToOne** = Relación uno a uno.
- **@OneToMany** = Relación uno a muchos.
- **@ManyToOne** = Muchos a uno.
- **@ManyToMany** = Muchos a muchos.
- **@JoinColumn** = Columna FK.
- **@JoinTable** = Tabla intermedia (ManyToMany).
- **fetch = FetchType.LAZY** = Carga diferida.
- **fetch = FetchType.EAGER** = Carga inmediata.

JPA (Jakarta Persistence API)

¡¡IMPORTANTE! : Estas anotaciones las usas SIEMPRE con ***spring-boot-starter-data-jpa***

Ciclo de vida

- **@PrePersist** = Antes de INSERT.
- **@PostPersist** = Después de INSERT.
- **@PreUpdate** = Antes de UPDATE.
- **@PostUpdate** = Después de UPDATE.
- **@PreRemove** = Antes de DELETE.
- **@PostLoad** = Después de SELECT.

HIBERNATE (anotaciones específicas)

Optimizaciones y comportamiento

- **@CreationTimestamp** = Fecha automática al crear.
- **@UpdateTimestamp** = Fecha automática al actualizar.
- **@DynamicInsert** = INSERT solo con campos no nulos.
- **@DynamicUpdate** = UPDATE solo de campos modificados.
- **@Where** = Filtro automático en queries (soft delete).
- **@SQLDelete** = Override del DELETE físico.

Cache

- **@Cacheable** = Cachea entidades.
- **@Cache** = Configuración de cache Hibernate.

VALIDATIONS (Jakarta Validation)

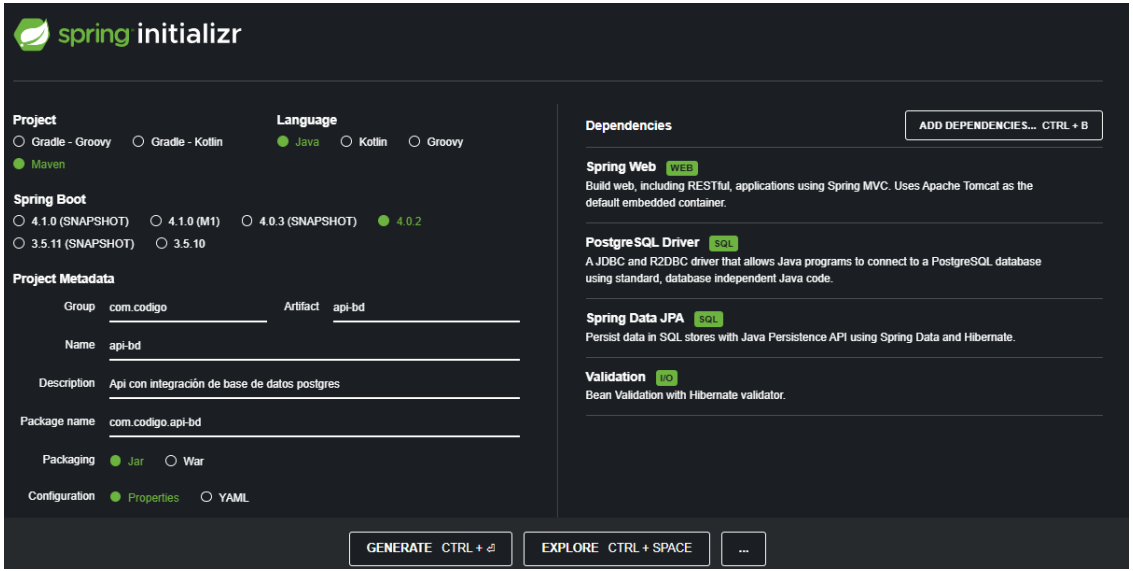
Las mas usadas junto con Spring Web + JPA.

- **@NotNull** = No permite null.
- **@NotBlank** = String no vacío.
- **@NotEmpty** = Colección no vacía.
- **@Size** = Tamaño mínimo/máximo.
- **@Min** = Valor mínimo.
- **@Max** = Valor máximo.
- **@Email** = Email válido.
- **@Pattern** = Expresión regular.

Crear proyecto con Initializr

Crear proyecto con Initializr

https://start.spring.io/



Desarrollo de API

Pruebas en Postman