DAW
Desarrollo de Aplicaciones Web
2º Curso

# DWES
# Desarrollo Web Entorno Servidor

UD 5 Spring Boot

1. Introducción a Spring

IES BALMIS
Dpto Informática
Curso 2025-2026
Versión 1 (10/2025)

# UD6.1 – Introducción a Spring

## ÍNDICE

## 1.1. Introducción

**Spring o Spring Framework** es el entorno de desarrollo de aplicaciones más popular de Java.

- La característica principal de Spring Framework es la inyección de dependencia o inversión de control (IoC).
- Con la ayuda de Spring Framework, podemos desarrollar una aplicación débilmente acoplada.

Este Framework se agrupa en dos grandes estructuras de librerías, Spring Boot y Spring MVC, dependiendo del producto final que deseamos crear.

**Spring Boot** es un módulo de Spring Framework que nos permite construir una aplicación independiente con configuraciones mínimas o nulas.

- Es ideal para desarrollar una aplicación simple con servicios **RESTful**.

**Spring MVC** es un entorno desarrollo Web basado en Modelo Vista Controlador (MVC) para crear aplicaciones web.

- Contiene una gran cantidad de archivos de configuración para varias capacidades.
- Es un entorno de desarrollo de aplicaciones web orientado a **HTTP**.

Spring contiene diferentes proyectos y módulos que proporcionan funcionalidades muy diversas.

| ***Módulos de Spring*** |
| --- |
| https://spring.io/projects |

La estructura modular de Spring se refiere a cómo el framework está organizado internamente en diferentes módulos que se pueden usar de forma independiente para construir aplicaciones Java.

Spring Framework no es solo un contenedor de inversión de control, sino que se compone de una familia de proyectos que cubren diversas áreas como desarrollo web, seguridad, microservicios, etc.

Estos proyectos, a su vez, se dividen en módulos que agrupan funcionalidades específicas, permitiendo a los desarrolladores elegir qué módulos utilizar en sus aplicaciones

**Spring** es un framework de desarrollo para aplicaciones Java que simplifica la creación de software empresarial.

Proporciona herramientas para:

- **Inversión de Control (IoC)**:
  - Gestiona objetos y dependencias.
  - Spring gestiona los objetos (beans) en lugar de que el código los cree directamente
- **Inyección de Dependencias (DI)**:
  - Asigna automáticamente las dependencias.
  - Spring inyecta las dependencias automáticamente (evita new manual).
- **Programación orientada a aspectos (AOP)**:
  - Maneja transacciones, seguridad, etc.
- **Acceso a datos**:
  - Integración con JDBC, JPA, Hibernate, etc.
- **Spring MVC**:
  - Para desarrollo web.
  - Incluye Thymeleaf entre otros módulos
    - Thymeleaf es un motor de plantillas para aplicaciones web en Spring.
    - Permite crear vistas HTML dinámicas integradas con Spring MVC.
    - Se usa comúnmente en aplicaciones web tradicionales (no solo APIs de tipo REST).

Los componentes principales de Spring son:

| Módulo | Descripción |
|---|---|
| **Spring Core** | Contiene IoC y DI (base del framework). |
| **Spring Boot** | Autoconfiguración para proyectos rápidos. |
| **Spring MVC** | Para aplicaciones web (controladores, vistas). |
| **Spring Data** | Simplifica el acceso a bases de datos. |
| **Spring Security** | Manejo de autenticación y autorización. |

## 1.2. Spring Starter y dependencias

**Spring Starter** es un asistente que genera la estructura de un proyecto Spring inclueyendo dependencias preconfiguradas que simplifican enormemente la configuración de proyectos Spring Boot.

Spring Starter nos permite:

- Eliminar la necesidad de buscar y configurar dependencias manualmente
- Garantizar la compatibilidad entre versiones
- Proporcionar configuraciones automáticas inteligentes
- Ahorrar tiempo en la configuración inicial de proyectos

Spring Starter (https://start.spring.io/) dispone de varias dependencias que añadidas al proyecto proveen de funcionalidades avanzadas.

Las que usaremos en nuestros proyectos son:

- Spring Developer Tools
  - **Spring Boot DevTools**: módulo diseñado para agilizar el desarrollo, proporcionando herramientas que mejoran la productividad durante la fase de codificación
  - **Lombok**: biblioteca de anotaciones de Java que ayuda a reducir el código repetitivo
- Spring Web
  - **Spring Web**: módulo diseñado para crear aplicaciones web, ya sea APIs REST (servicios backend) o aplicaciones MVC tradicionales (con vistas HTML).
- Sring Template Engines
  - **Thymeleaf**: es un motor de plantillas moderno para aplicaciones web Java, especialmente diseñado para integrarse perfectamente con Spring
- Spring Security
  - **Spring Security**: framework de control de acceso y autenticación altamente personalizable para aplicaciones Spring
- Spring SQL
  - **Spring Data JDBC:** gestionar datos en BD SQL con JDBC simple usando Spring Data
  - **Spring Data JPA:** gestionar datos en BD SQL con Java Persistence API usando Hibernate y Spring Data
  - **H2 Database**: provee de una BD SQL en memoria que soporta JDBC API y JPA
  - **MySQL Driver**: MySQL JDBC driver

Las **dependencias** de Spring son bibliotecas (librerías) que agregas a tu proyecto para incluir funcionalidades del ecosistema Spring, como web, seguridad, persistencia, pruebas, etc.

En un proyecto Maven (como en Spring Boot), se declaran en el archivo **pom.xml**, y permiten que tu aplicación use módulos específicos de Spring.

Los **starters de Spring Boot** son agrupaciones de dependencias listas para usar. Te evitan tener que añadir manualmente muchas bibliotecas.

A continuación se muestran algunos:

| Starter | Descripción |
|---|---|
| **spring-boot-starter-web** | Permite crear aplicaciones web REST o MVC.<br>Incluye Tomcat, Jackson, Spring MVC, etc. |
| **spring-boot-starter-thymeleaf** | Motor de plantillas HTML para aplicaciones web con vistas. |
| **spring-boot-starter-data-jpa** | Soporte para JPA/Hibernate.<br>Permite usar @Entity, JpaRepository, etc. |
| **spring-boot-starter-validation** | Añade soporte para validaciones con anotaciones como @NotNull, @Email, etc. |
| **spring-boot-starter-security** | Añade autenticación y autorización<br>(Spring Security). |
| **spring-boot-starter-test** | Añade librerías para pruebas unitarias e integración: JUnit, Mockito, AssertJ, etc. |

Para acceder a la información de cada librería y ver las dependencias que incluye, disponemos de Maven Repository:

| | Dependencias |
|---|---|
| **spring-boot-starter-web** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web/3.5.3 |
| **spring-boot-starter-thymeleaf** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf/3.5.3 |
| **spring-boot-starter-data-jpa** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa/3.5.3 |
| **spring-boot-starter-validation** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation/3.5.3 |
| **spring-boot-starter-security** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-security/3.5.3 |
| **spring-boot-starter-test** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-test/3.5.3 |
| **spring-boot-devtools** | https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools/3.5.3 |

Los árboles de dependencias de cada librería se pueden obtener fácilmente en cualquier portal de IA, pero como ejemplo mostraremos algunas:

```
spring-boot-starter-web
├── spring-boot-starter
│   ├── spring-boot
│   ├── spring-boot-autoconfigure
│   ├── spring-boot-starter-logging
│   │   ├── logback-classic
│   │   └── slf4j-api
├── spring-web
├── spring-webmvc
├── spring-boot-starter-json
│   ├── jackson-databind
│   ├── jackson-core
│   └── jackson-annotations
├── jakarta.servlet-api
└── tomcat-embed-core (Tomcat embebido)
```

```
spring-boot-starter-thymeleaf
├── spring-boot-starter
├── thymeleaf
├── thymeleaf-spring6 (o spring5, según versión)
├── thymeleaf-extras-java8time (para fechas con Java 8+)
```

```
spring-boot-starter-data-jpa
├── spring-boot-starter
├── spring-boot-starter-aop
├── spring-boot-starter-jdbc
│   ├── HikariCP (por defecto)
│   └── spring-jdbc
├── spring-data-jpa
│   ├── spring-data-commons
│   ├── spring-data-repository
├── jakarta.persistence-api
├── hibernate-core (JPA provider)
└── hibernate-entitymanager
```

```
spring-boot-starter-security
├── spring-boot-starter
├── spring-security-core
├── spring-security-config
├── spring-security-web
├── spring-security-crypto
```

```
spring-boot-devtools
├── spring-boot
├── spring-boot-autoconfigure
├── spring-boot-devtools (código interno)
│
│ (Incluye funcionalidades como:)
│  ├── reinicio automático (classloader reiniciado al detectar cambios)
│  ├── live reload (con LiveReload server)
│  ├── configuración para desarrollo (p.e.cache desactivado en templates)
│  └── deshabilitación automática en producción
```

## 1.3. Tecnologías

Los proyectos de Spring utilizan dos tecnologías:

- Java (21-LTS este curso)
- Maven

**JAVA** es el lenguaje de programación usado en los proyectos Spring.

Para que funcione un proyecto Spring debemos tener instalado el JDK (Kit de Desarrollo de Java) en nuestro sistema.

La versión a instalar debe ser la última estable y con soporte a largo plazo (LTS), que en estos momentos es la 21.

**Maven** es una herramienta de gestión y automatización de proyectos utilizada principalmente en proyectos de Java. Fue desarrollada por la Apache Software Foundation.

Maven proporciona:

- **Gestión de dependencias**: Descarga automáticamente bibliotecas y frameworks externos que tu proyecto necesita (por ejemplo, JUnit, Spring, Hibernate) desde un repositorio central.
- **Compilación y empaquetado**: Compila el código, lo empaqueta (por ejemplo, en un .jar o .war) y puede incluso desplegarlo.
- **Estandarización**: Define una estructura estándar de proyecto, lo cual facilita el mantenimiento y colaboración.
- **Plugins**: Utiliza plugins para tareas comunes como ejecución de pruebas, generación de documentación, análisis de código, etc.
- **Portabilidad y consistencia**: Como usa un archivo de configuración central (llamado pom.xml), cualquier desarrollador puede descargar el proyecto y tenerlo funcionando rápidamente.

El archivo de configuración es **pom.xml**, donde se define:

- Las dependencias de tu proyecto.
- Información del proyecto (nombre, versión, desarrollador).
- Plugins.
- Fases del ciclo de vida del build.

## 1.4. IDE recomendado

Como ya se comentó al principio del módulo, los IDE recomendados son:

NetBeans 27 con plugins:

- **NB Springboot**

Visual Studio Code Community con plugins:

- Generalistas
  - **Prettier – Code Formatter**
  - **XML**
  - **Spanish Language Pack for Visual Studio Code**
- **Extension Pack for Java**, que incluye
  - Project Manager for Java
  - IntelliCode
  - IntelliCode API Usage Examples
  - Debugger for Java
  - Language Support for Java(TM) by Red Hat
  - Maven for Java
  - Gradle for Java (deshabilitar)
  - Test Runner for Java
- **Spring Boot Extension Pack**, que incluye
  - Spring Initilizr Java Support
  - Spring Boot Dashboard
  - Spring Boot Tools
- **Community Server Connectors**, que incluye
  - Runtime Server Protocol UI
    - Visitar (https://code.visualstudio.com/docs/java/java-tomcat-jetty)
    - (No instalar) Tomcat
    - (No instalar) Local Tomcat
- **Thunder CLient**
  - (No instalar) Postman

## 1.5. Formación

| ***Spring*** |
| --- |
| https://spring.io/<br>https://spring.io/projects → Proyectos |

### *Tutoriales*

| ***Tutorialspoint*** |
| --- |
| https://www.tutorialspoint.com/java_technology_tutorials.htm<br>→ Learn Spring<br>→ Learn Spring Boot<br>→ Learn Spring Boot JPA<br>→ Learn Spring Security |

| ***Baeldung*** |
| --- |
| https://www.baeldung.com/start-here |

### *Cursos de Spring*

| ***Openwebinars*** |
| --- |
| https://academia.openwebinars.net/catalogo/ → Catálogo de cursos con filtrado<br><br>https://openwebinars.net/rutas/programador-java/ → Programador Java Web<br>https://openwebinars.net/rutas/spring-y-spring-boot-junior-a-experto/ → Spring |

| ***Openwebinars – Curso de Spring con Spring Tool Suite Eclipse*** |
| --- |
| https://academia.openwebinars.net/aprende/spring-boot |

| ***Youtube – Curso de Spring con NetBeans*** |
| --- |
| https://www.youtube.com/watch?v=cCcItxk4uWs&list=PLCIjncxyvEHbSAhlMhSrMROJtg1s_tlG8&index=1 |

| ***Youtube – Curso de Spring con VSCode*** |
| --- |
| https://www.youtube.com/watch?v=5M39KMhVOWA&list=PL2Z95CSZ1N4EWw14HZ0NFD3woFcn6uiCm&index=1 |

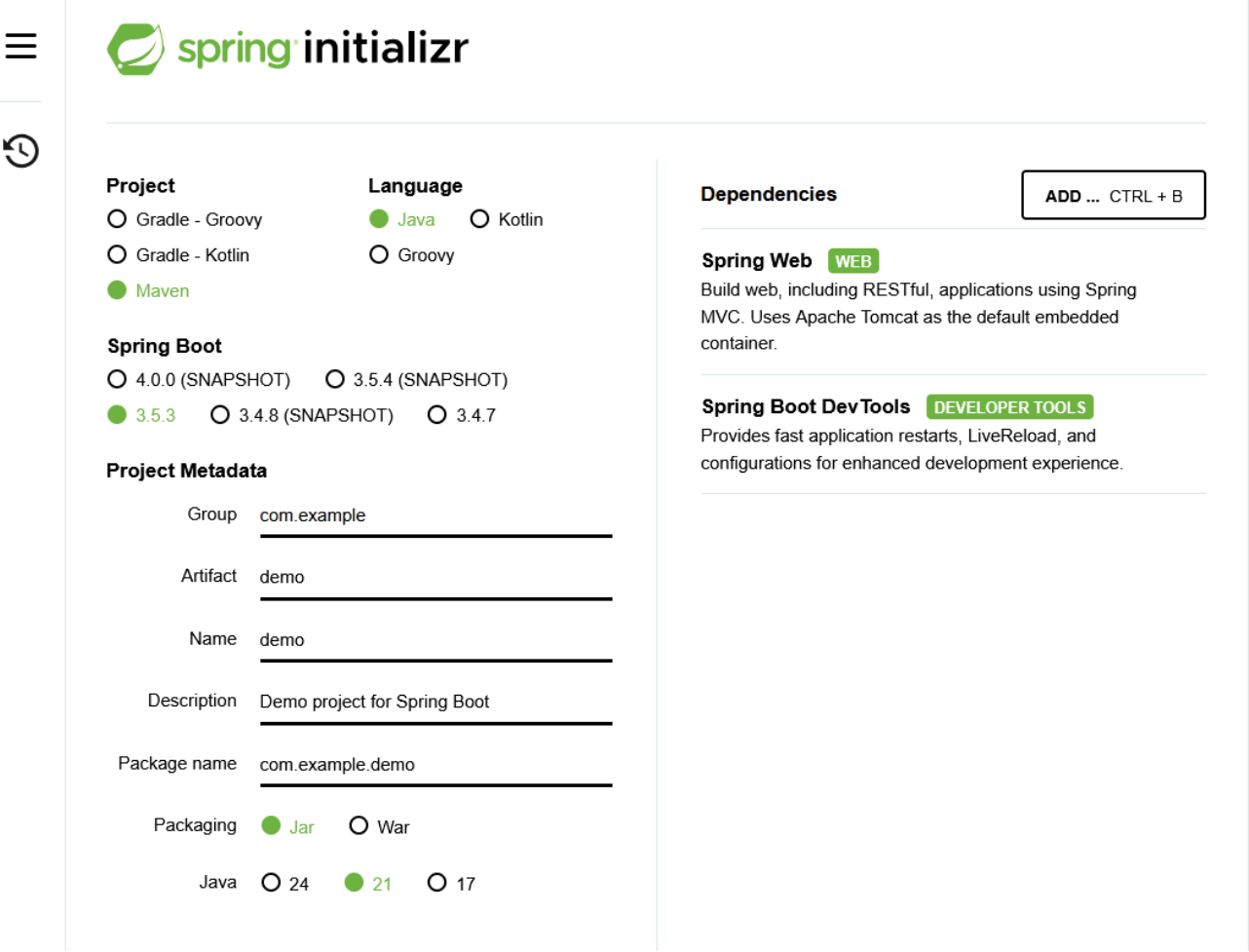
| ***Youtube – Curso de Spring con IntelliJ*** |
| --- |
| https://www.youtube.com/watch?v=C_6DqxKbGM8&list=PLU8oAlHdN5BnVCDXq-B2j3-9QXE8LpSxI → Hasta el vídeo 8 incluido |

A partir de ahora nos referiremos a **Spring Boot** como **SB**.

# 1.6. Generar un proyecto Spring con Spring Initializr

***Spring Initializr***

***http://start.spring.io***




Dejaremos la versión propuesta **Spring Boot 3.5.X** porque es la estable. No seleccionar las superiores son SNAPSHOT ya que son versiones en continuo desarrollo y pueden ser inestables.

En desarrollo usaremos **JAR** al principio para disponer de un Apache Tomcat embebido. Más adelante cambiaremos de JAR a WAR para desplegar estos proyectos en un Apache Tomcat de producción.

Seleccionaremos **Java 21** porque es la versión LTS con la que iniciamos el curso actual.

Para nuestro primer proyecto, añadiremos las dependencias:

- **Spring Web**
- **Spring Boot Dev Tools**

# 1.7. Generar un proyecto base Spring con NetBeans

NetBeans permite generar un proyecto de Spring Boot usando el "**Spring Initializr**" teniendo el **Plugin NB SpringBoot** instalado, pero necesita conexión a **Internet**.

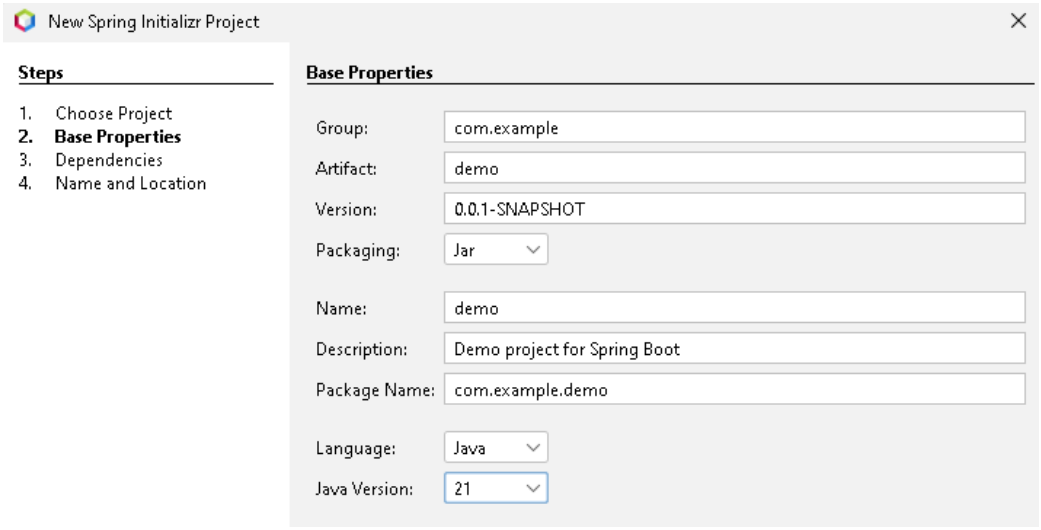
Para ello usaremos:

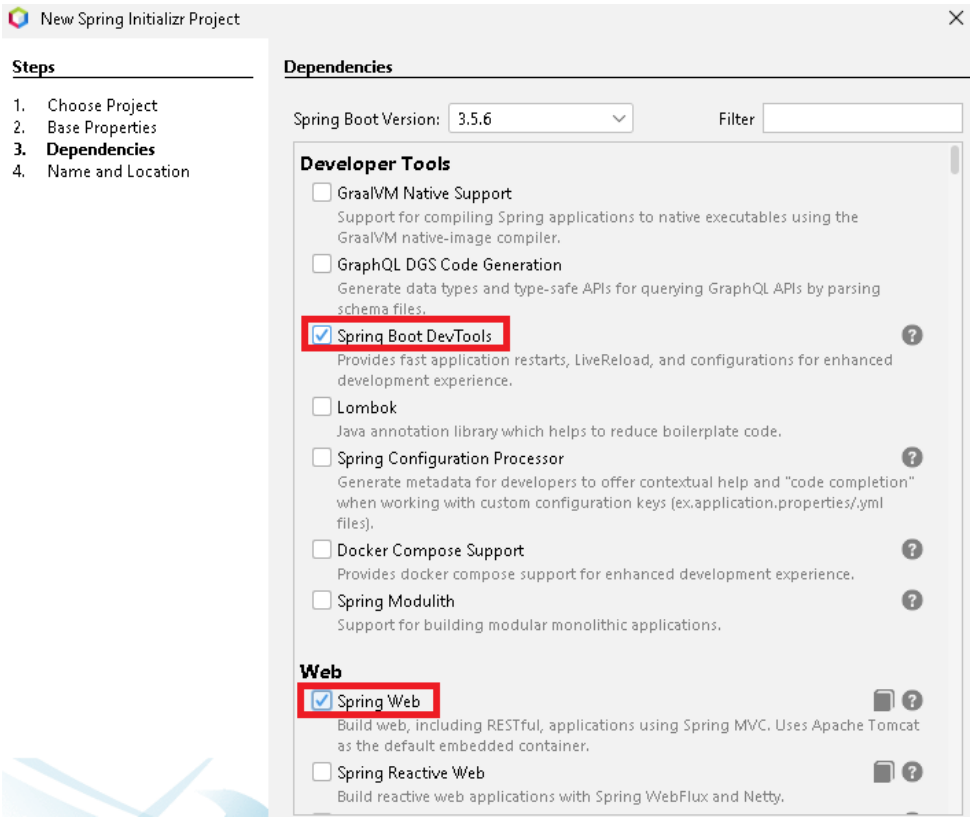
**New Project**

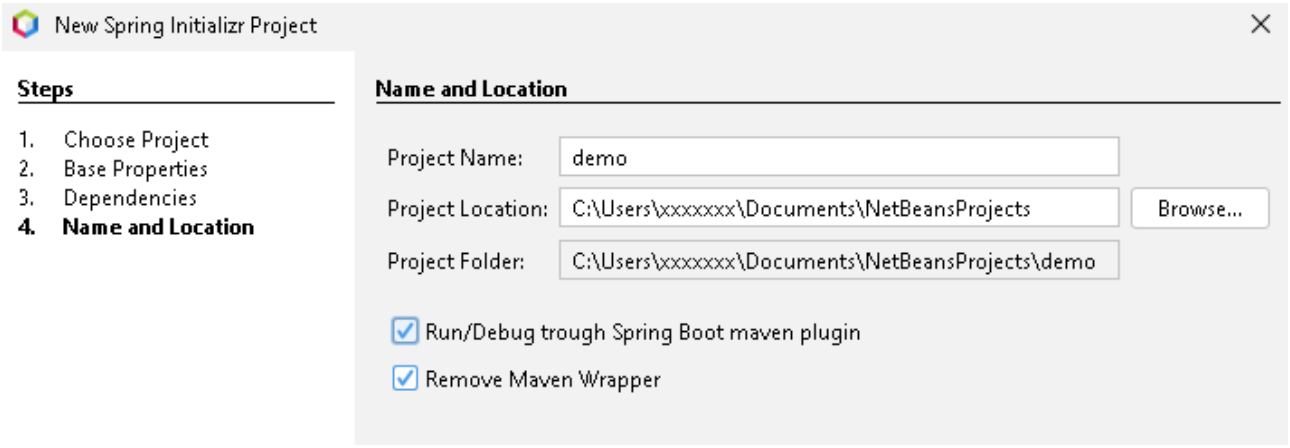
→ **Java with Maven**

→ **Spring Boot Initializr project**

y seguir las indicaciones del asistente

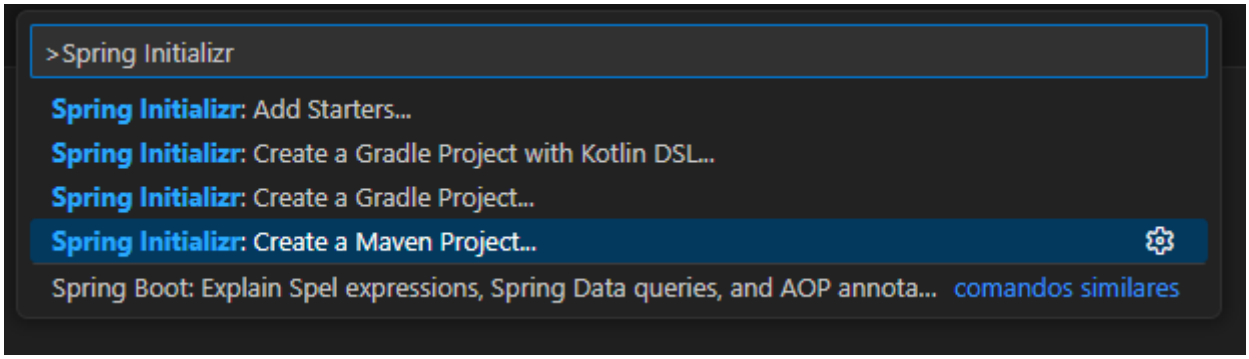




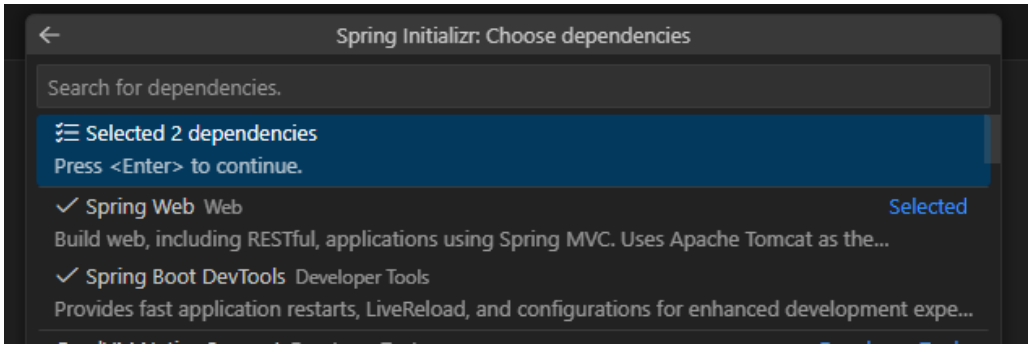

## 1.8. Generar un proyecto base Spring con VSCode

para crear un proyecto denominado **SpringEjemplo** de **Spring Boot,** abriremos VSCode con **vscode-spring.bat** y pulsaremos Ctrl+Shift+P y tecleamos Spring Initializr para seleccionar **Spring Initializr: Create a Maven Project**:




A continuación iremos seleccionando opciones:

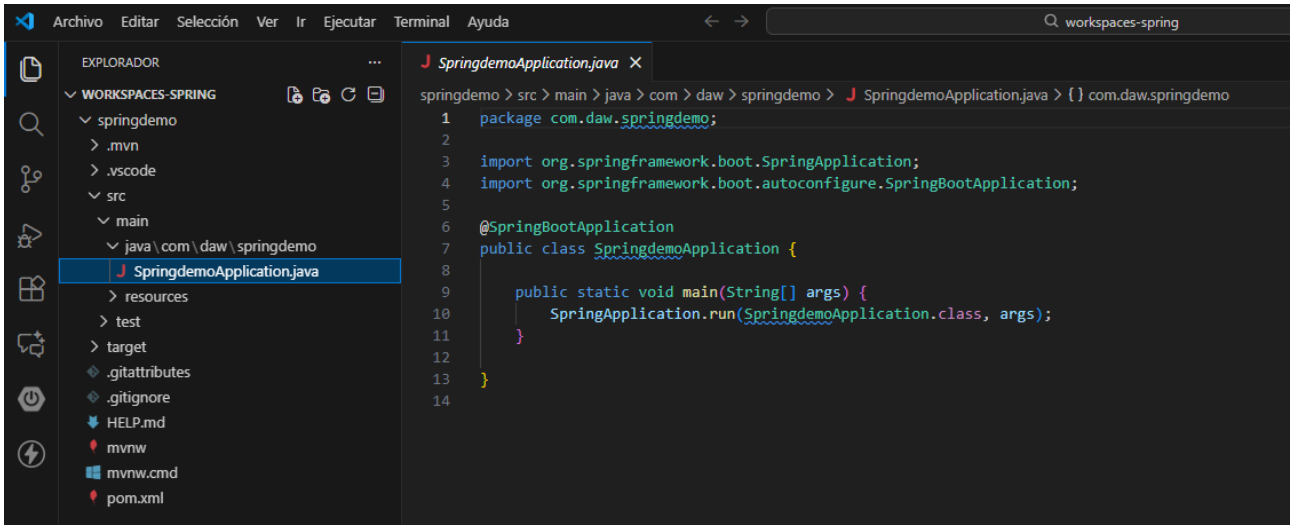
- Specify Spring Boot version 4.0.0
- Specify project language Java
- Input Group Id com.daw
- Input artifact springdemo
- Input Package Name com.daw.springdemo
- Specify packaging type Jar
- Specify Java version 21
- Search for dependencias Spring Web y Spring Boot DevTools



Y por último seleccionaremos la carpeta donde alojar nuestro proyecto, que en nuestro ejemplo es:

**C:\IDE-Programacion\VSCodePortable-1.106\workspaces-spring**

El proyecto generado tendrá la siguiente estructura:



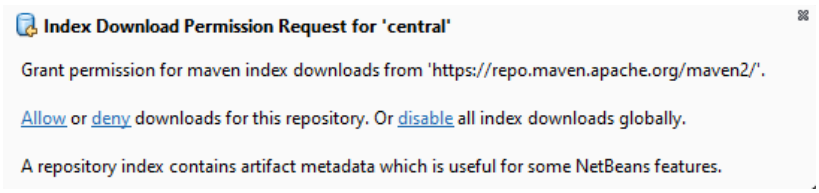

A lo largo de las próximas unidades didácticas, veremos las posibilidades de trabajar con Spring Boot.

## 1.9. Spring Web con HTML File

Abriremos el proyecto generado en http://start.spring.io o crearemos uno nuevo mediante el asistente.
Debemos abrir el proyecto con **NetBeans 27** con el plugin **NB SpringBoot** instalado o con **VSCode** y sus **plugins para Spring**.

***Configuración Maven Index en NetBeans***

Al abrir la primera vez un proyecto de tipo Maven se nos mostrará este mensaje pidiendo permiso par descargar las librerías/dependencias indicadas en el archivo **pom.xml**:




Una vez termine de transferir el index y desempaquetarlo, lo deberemos compilar para que descargue las dependencias y cree la estructura del proyecto.

**Este proceso ya está realizado en la versión portable configurada facilitada por el profesor.**

## *Configuración del Context Path*

La dependencia **Spring Web**, crea un Apache Tomcat embebido en el proyecto que se ofrece por defecto en el puerto **8080**.

Además instalará nuestro proyecto web en el raíz, es decir, en el **Context Path = "/",** teniendo que acceder a él mediante **http://localhost:8080/** pudiendo interferir en otros proyectos.

En Spring Boot podemos cambiar el valor por defecto de variables de configuración en el archivo **src/main/resources/application.properties.**
Si queremos cambiar el **context path** de este proyecto de SB (Spring Boot):

**(Other sources) src/main/resources/application.properties**

```
spring.application.name=demo
server.servlet.context-path=/demo
```

## *Contenido a mostrar HTML+CSS+JS*

La carpeta de archivos HTML+CSS+JS del proyecto para el servidor web (lo equivalente a un Servidor Web como Apache) es **src/main/resources/.**

Si creamos/copiamos un archivo **index.html** en **src/main/resources/static** se mostrará directamente en http://localhost:8080/demo ya que el valor de **context path** del proyecto. Como se ha explicado antes, si no cambiamos esta variable en el **aplication.properties** el valor por defecto es /.

Si nuestra web contiene varios archivos html o archivos .css / .js, deberemos copiarlos también.

## *Compilar el proyecto la primera vez en NetBeans*

Run Maven → Clean Installl

- De esta forma generamos los archivos **build** necesarios para NetBeans.
- Recomendamos cerrar el proyecto y volverlo a abrir para cargar toda la configuración completa.

## *Ejecutar y probar desde el navegador*

Para probar, ejecutar el proyecto y abrir el navegador en la web:

http://localhost:8080/demo