Embedded Server in Spring Boot

give me the deep and detailed explaination of Embedded Server
in spring boot so that i can able to understand it well and do not have any type of doubt in this.

so give me the deep and detailed explaination

Below is a **very deep, step-by-step, no-assumptions explanation** of **Embedded Server in Spring Boot**, written in **simple language**, keeping a **backend learner like you (Java + Spring Boot focus)** in mind.
By the end, you should have **zero confusion** about *what it is, why it exists, how it works internally, and how requests flow*.

🌱 What Is an Embedded Server in Spring Boot?

An **Embedded Server** is a **web server that is packaged inside your Spring Boot application itself**.

This means:

> Your Spring Boot application **contains the server**, instead of being deployed **on an external server**.

Common embedded servers used by Spring Boot:

- Apache Tomcat (default)
- Jetty
- Undertow

🧠 Before Spring Boot: Traditional Way (Very Important to Understand)

Let's first understand **what problem Spring Boot solved**.

🔴 Traditional Spring / Java Web Apps

Earlier, the flow looked like this:

1. You write a **Spring MVC application**
2. You generate a **WAR file**
3. You install a server separately:
    - Apache Tomcat
    - WebLogic
    - JBoss
4. You deploy the WAR file into the server
5. Then the server starts your app

❌ Problems with this approach

- Manual server installation
- Server configuration headaches
- Environment mismatch (works on my machine 😤)
- Deployment is slow
- Not cloud-friendly

🚀 Spring Boot Solution: Embedded Server

Spring Boot said:

> "Why not **ship the server with the application itself**?"

So Spring Boot introduced **Embedded Servers**.

✅ What Changes with Embedded Server?

Traditional Way	Spring Boot Way
External Tomcat	Embedded Tomcat
WAR deployment	JAR execution
Manual setup	Auto-configured
Heavy config	Zero / minimal config

You simply run:

```bash
java -jar myapp.jar
```

And 💥 **server starts automatically**.

🏗️ What Does “Embedded” Actually Mean?

It means:

- The **server libraries (Tomcat/Jetty/etc.)** are added as **dependencies**
- The server is started **inside the same JVM**
- Spring Boot controls:
  - Server startup
  - Port binding
  - Lifecycle management

No separate server process ❌

No deployment folder ❌

🔍 Where Does the Embedded Server Come From?

Spring Boot Starter

When you add:

```xml
spring-boot-starter-web
```

Spring Boot automatically adds:

- Spring MVC
- Jackson
- Validation
- **Embedded Apache Tomcat**

This happens via **dependency management** using:

- Maven or

- Gradle

⚙️ How Embedded Server Starts (Internal Flow)

This is **VERY IMPORTANT**. Let's go step by step.

🧩 Step 1: `main()` Method Starts

```java
public static void main(String[] args) {
    SpringApplication.run(MyApp.class, args);
}
```

This does **NOT** start Tomcat immediately.

🧩 Step 2: SpringApplication Bootstraps Context

- Creates **ApplicationContext**
- Scans beans
- Applies auto-configuration

🧩 Step 3: Detects Web Application

Spring Boot checks:

> "Is this a web app?"

It detects:

- `DispatcherServlet`
- `spring-web`
- `spring-webmvc`

So it decides:

👉 **I need a web server**

🧩 Step 4: Auto-Configuration Kicks In

Spring Boot uses:

- `ServletWebServerFactoryAutoConfiguration`

This:

- Chooses **Tomcat** by default
- Creates `TomcatServletWebServerFactory`

🧩 Step 5: Embedded Server Is Created

Spring Boot:

- Creates Tomcat instance
- Configures:
    - Port (default: 8080)
    - Context path
    - Thread pool
    - Error pages

🧩 Step 6: Server Starts Automatically

Tomcat starts **inside JVM**:

```scss
Tomcat started on port(s): 8080 (http)
```

Your app is now live 🚀

🔄 Request Flow with Embedded Server

Let's trace **one HTTP request**.

Example:

```bash
GET http://localhost:8080/api/users
```

Flow:

1. **Browser** sends request
2. **Embedded Tomcat** receives it
3. Tomcat forwards request to:
    - `DispatcherServlet`
4. `DispatcherServlet`:
    - Finds matching controller
5. Controller method executes
6. Response returned to:
    - Tomcat
7. Tomcat sends response to browser

🧠 **Same flow as traditional server**, just embedded.

🧠 DispatcherServlet Role (Quick Clarity)

- `DispatcherServlet` is **NOT the server**
- It is a **front controller**
- Server (Tomcat) only:
    - Accepts request
    - Forwards to servlet

🔧 Can We Change Embedded Server?

Yes 💯

Switch Tomcat → Jetty

Exclude Tomcat and add Jetty:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

No code change required ✅

⚖️ Tomcat vs Jetty vs Undertow (Simple View)

Server	Best For
Tomcat	Default, stable, most used
Jetty	Lightweight, async-heavy
Undertow	High performance, low memory

🔐 Embedded Server & Spring Security

Embedded server:

- Handles HTTP connection
- Spring Security:
    - Adds filters
    - Intercepts requests
    - Authenticates users

Security works **on top of embedded server**, not instead of it.

☁️ Why Embedded Server Is Cloud & Microservice Friendly

Embedded servers enable:

- Docker containers 🐳
- Kubernetes ☸️
- Microservices architecture
- CI/CD pipelines

Each service:

- Runs independently
- Has its own server
- Can be scaled individually

❗ Common Doubts (Clearing Confusion)

❓ Is Embedded Server Less Powerful?

❌ No

Same Tomcat, same performance.

❓ Can I still deploy WAR?

✅ Yes

But JAR + embedded server is preferred.

❓ Is Embedded Server Mandatory?

❌ No

But Spring Boot **strongly encourages it**.

🧠 Final Mental Model (Remember This)

> **Spring Boot Application = Application Code + Embedded Server + Auto Configuration**

Everything runs **together**, **automatically**, **inside one JVM**.

If you want, next I can explain:

- Embedded Server **vs External Server**
- How Spring Boot creates **Servlet Container**
- How to configure thread pool & ports deeply
- How embedded server works with **Spring Security filters**

Just tell me 👍