

Servlets in Spring Boot

give me the deep and detailed explanation of Servlet in spring boot mvc 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 explanation

Below is a **deep, structured, no-assumption explanation of Servlets in Spring Boot MVC**, written so that **every layer becomes clear** and you don't carry hidden doubts. I'll start from *zero* and gradually connect everything to **Spring Boot MVC**, exactly matching how you're learning backend development.

Servlet in Spring Boot MVC — Deep & Complete Explanation

1 What is a Servlet? (Absolute Foundation)

A **Servlet** is a **Java class** that runs on a **server** and is responsible for:

- 👉 Receiving HTTP requests
- 👉 Processing those requests
- 👉 Returning HTTP responses

In simple words:

Servlet = Java program that handles web requests

Servlets are part of the **Servlet API**, which defines:

- How requests come in
 - How responses go out
 - How a web application lives on a server
-

2 Why Do We Need Servlets?

Before servlets:

- Servers could only return **static HTML**
- No dynamic logic
- No database interaction

Servlets introduced:

- Dynamic responses
- Java logic on server
- Database + business logic integration

 **Servlets are the foundation of all Java web frameworks**, including:

- Spring MVC
 - Spring Boot
-

3 Who Executes a Servlet?

A Servlet **never runs alone**.

It is executed by a **Servlet Container** (also called **Web Container**).

Examples:

- Apache Tomcat
- Jetty
- Undertow

Responsibilities of a Servlet Container:

- Create servlet objects
- Call lifecycle methods
- Manage threads
- Handle HTTP protocol
- Map URLs to servlets

 In Spring Boot, Tomcat is **embedded by default**.

4 Servlet Lifecycle (VERY IMPORTANT)

A servlet has a **fixed lifecycle** controlled by the container.

Servlet Lifecycle Phases

Loading → Initialization → Request Handling → Destruction

1 Loading

- Happens when application starts OR first request arrives
 - Servlet class is loaded into memory
-

2 Initialization — `init()`

java

```
public void init() throws ServletException
```

- Called **once**
 - Used for:
 - DB connections
 - Resource initialization
 - Configuration loading
-

3 Request Handling — `service()`

java

```
public void service(HttpServletRequest req, HttpServletResponse res)
```

- Called **for every request**
- Delegates to:
 - `doGet()`
 - `doPost()`
 - `doPut()`
 - `doDelete()`

⚠ Servlets are multithreaded

- One servlet instance
- Multiple threads

- Must be thread-safe
-

4 Destruction — `destroy()`

java

```
public void destroy()
```

- Called once before server shutdown
 - Cleanup logic
-

5 Core Servlet Interfaces & Classes

◆ `Servlet Interface`

Root interface (rarely used directly)

◆ `GenericServlet`

Protocol independent

◆ `HttpServlet (MOST IMPORTANT)`

java

```
public class MyServlet extends HttpServlet
```

Provides:

- `doGet()`
- `doPost()`
- `doPut()`
- `doDelete()`

Spring MVC internally builds on `HttpServlet`.

6 HTTP Request & Response Objects

◆ HttpServletRequest

Contains:

- URL
- Headers
- Parameters
- Body
- Session info

Example:

```
java  
  
request.getParameter("username");
```

◆ HttpServletResponse

Used to:

- Set status code
- Write response
- Set headers

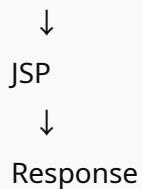
Example:

```
java  
  
response.getWriter().write("Hello");
```

7 Traditional Servlet-Based MVC (Before Spring)

Flow:

```
nginx  
  
Browser  
↓  
Servlet  
↓  
Business Logic
```



Problems:

- ✖ Too much boilerplate
- ✖ Manual URL mapping
- ✖ Hard to maintain
- ✖ No separation of concerns

👉 This is WHY Spring MVC was created

8 Servlet in Spring MVC (Core Concept)

⚠ Spring MVC uses exactly ONE Servlet

That servlet is:

 DispatcherServlet

DispatcherServlet = Front Controller Servlet

📌 It is a **real servlet**, extending HttpServlet .

Entity reference:

DispatcherServlet

9 Role of DispatcherServlet

DispatcherServlet:

- Receives ALL HTTP requests
- Delegates to correct controller
- Handles response rendering

It replaces:

- ✖ Multiple servlets
- ✖ Manual routing

Complete Spring Boot MVC Request Flow

less

Client (Browser / Postman)



Embedded Tomcat



DispatcherServlet ← (Servlet)



Handler Mapping



Controller (@Controller / @RestController)



Service Layer



Repository / Database



Controller



View Resolver / JSON Converter



DispatcherServlet



Response

10 How Spring Boot Auto-Configures Servlet

In **Spring Boot**, you NEVER write:

java

extends HttpServlet

Because:

- Spring Boot auto-registers DispatcherServlet

- Auto-mapped to /

📌 Enabled via:

java

`@SpringBootApplication`

Which internally activates:

- Servlet scanning
 - DispatcherServlet registration
 - Embedded Tomcat
-

1 | 1 Controllers Are NOT Servlets (Important Clarification)

✗ `@RestController` is NOT a servlet

✗ `@Controller` is NOT a servlet

✓ They are POJOs

✓ DispatcherServlet calls them internally

1 | 2 Why Only One Servlet in Spring MVC?

This follows Front Controller Design Pattern

Advantages:

- ✓ Centralized request handling
 - ✓ Better security
 - ✓ Easier logging
 - ✓ Clean architecture
 - ✓ High scalability
-

1 | 3 Servlet vs Spring MVC (Comparison)

Aspect	Servlet	Spring MVC
Level	Low-level	High-level

Aspect	Servlet	Spring MVC
Boilerplate	Very high	Very low
URL Mapping	Manual	Annotation-based
Thread Mgmt	Manual concern	Abstracted
Testability	Hard	Easy

1 4 Real-Life Analogy

-  **Servlet Container** → Building
-  **DispatcherServlet** → Reception Desk
-  **Controllers** → Departments
-  **Service Layer** → Employees
-  **Response** → Final Output

1 5 Key Takeaways (Must Remember)

- ✓ Servlet is the **foundation**
- ✓ Spring MVC is **built on Servlet API**
- ✓ DispatcherServlet is the **heart of Spring MVC**
- ✓ Spring Boot hides servlet complexity
- ✓ You still benefit from servlet power

◀ END Final Summary

Servlets handle web requests at a low level
Spring MVC abstracts servlets using DispatcherServlet
Spring Boot auto-configures everything for you

You are **not avoiding servlets** —
You are **using them intelligently via Spring Boot**.

If you want next:

- Deep dive into **DispatcherServlet internals**
- **Filters vs Interceptors vs Servlet**
- **Thread safety in Servlets**
- **How Spring MVC converts JSON automatically**

Just tell me 

Do you like this personality?