Debugging Spring Boot Application in IntelliJ IDEA

Debugging :

Debugging is the process of **finding and fixing errors** (bugs) in your code. It helps you analyze the flow and state of the application at various execution points.

Importance of Debugging :

- To **inspect variables and objects** at runtime.
- To **check flow of execution**.
- To fix **NullPointerExceptions**, logic errors, or configuration issues.
- To understand how **Beans, Controllers, Services** are working internally.

3. Prerequisites

- Spring Boot project imported in **IntelliJ IDEA**.
- Use **Maven** build system.
- Ensure the project runs with the `main()` method from `@SpringBootApplication`.

4. Running in Debug Mode

Option 1: Using IntelliJ Debug Button

1. Open your main Spring Boot class (with `@SpringBootApplication`).
2. Click the **green bug icon** near the `main()` method.
3. Or click the **debug icon** (a bug) in the top-right toolbar.
4. This will **start your application in debug mode**.

Option 2: Set Breakpoint and Debug

1. Open any class (e.g., `Controller`, `Service`).
2. Click on the left margin (gutter) next to the line number – this sets a **breakpoint** (red dot).
3. Run the app in **Debug mode**.
4. When the app hits the breakpoint, it will pause.

5. Breakpoints

- **Breakpoint**: A point in the code where execution will pause.
- You can **inspect values**, change variables, step over/into code.
- Right-click on breakpoint → Add conditions or hit count.

6. Debugger Tools in IntelliJ

Tool	Description
▶ Resume Program	Continue execution till next breakpoint
⏭ Step Over	Execute the current line and go to the next
🔽 Step Into	Go inside the method being called
⏏ Step Out	Exit the current method
⬇ Variables Pane	See current variable values
⛓ Frames	Check call stack (method chain)
📌 Watches	Monitor expressions/variables

7. Debugging Web Requests (REST API)

1. Set a breakpoint inside your **Controller** or **Service**.
2. Run the app in Debug mode.
3. Hit the endpoint via **Postman**, **curl**, or browser.
4. IntelliJ will pause the request where the breakpoint is set.
5. Use debug panel to inspect request/response data.

8. Common Use-Cases to Debug

- `@Autowired` dependency not injected properly.
- Service method not called.
- JPA Repository query not returning expected result.
- Unexpected null/empty value.
- API returning 500 status.