🚀Spring MVC (Without Spring Boot) — Using Embedded Tomcat, Jakarta Servlet API, and Tomcat Jasper for JSP Parsing

This is a **fully rewritten** version focusing ONLY on:

✅**Pure Spring MVC** (no Spring Boot)
✅**Embedded Tomcat server** (no external Tomcat)
✅**Jakarta Servlet API**
✅**Tomcat Jasper** to compile JSP → Servlet
✅**DispatcherServlet configuration**
✅**InternalResourceViewResolver** setup

This note gives you everything needed to build a **complete Spring MVC project** using embedded Tomcat, JSP views, controllers, and the Spring Web MVC framework.

🛖1. What is Spring MVC (Without Spring Boot)?

Spring MVC is the older version of Spring Web MVC that requires **manual configuration**.

You must manually configure:

- DispatcherServlet
- HandlerMapping / HandlerAdapter (Spring does this internally once enabled)
- ViewResolver
- Embedded Tomcat (manually added)
- Component scanning
- JSP rendering via Jasper

Spring Boot normally auto-configures all these — but WITHOUT Boot, **you configure everything yourself**.

🛖2. Project Dependencies (pom.xml)

Use **Jakarta Servlet API**, **Spring MVC**, **Embedded Tomcat**, and **Tomcat Jasper**.

```
<dependencies>

    <!-- Spring MVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>6.1.1</version>
    </dependency>

    <!-- Jakarta Servlet API -->
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>6.0.0</version>
        <scope>provided</scope>
    </dependency>

    <!-- Embedded Tomcat -->
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-core</artifactId>
        <version>11.0.0</version>
    </dependency>

    <!-- Jasper (JSP parser) → Required for JSP rendering -->
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <version>11.0.0</version>
    </dependency>

    <!-- JSP Standard Tag Library -->
    <dependency>
        <groupId>jakarta.servlet.jsp.jstl</groupId>
        <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
        <version>3.0.0</version>
    </dependency>

</dependencies>
```

🛖3. Directory Structure

```
src/main/java
   └── com.example.mvc
          ├── App.java  (starts embedded Tomcat)
          ├── WebConfig.java (Spring MVC config)
          └── controllers
                └── HomeController.java

src/main/webapp
   └── WEB-INF
         └── views
               └── home.jsp
```

🛖4. Embedded Tomcat Configuration (App.java)

This class: - Starts Tomcat - Creates a webapp root - Registers DispatcherServlet manually

```
public class App {
    public static void main(String[] args) throws Exception {

        Tomcat tomcat = new Tomcat();
        tomcat.setPort(8080);

        // Root context directory
        String webAppDir = new File("src/main/webapp").getAbsolutePath();

        // Create web application context
        Context context = tomcat.addWebapp("", webAppDir);

        // Register Spring's DispatcherServlet
        AnnotationConfigWebApplicationContext appContext = new
AnnotationConfigWebApplicationContext();
        appContext.register(WebConfig.class);

        DispatcherServlet dispatcherServlet = new
DispatcherServlet(appContext);

        Tomcat.addServlet(context, "dispatcher", dispatcherServlet);
        context.addServletMappingDecoded("/", "dispatcher");

        tomcat.start();
        tomcat.getServer().await();
    }
}
```

🛖5. Spring MVC Configuration (WebConfig.java)

This config replaces **web.xml**. It declares:

- Component Scanning
- Enabling Spring MVC
- View Resolver (JSP folder + .jsp suffix)

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.example.mvc")
public class WebConfig implements WebMvcConfigurer {
    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver vr = new InternalResourceViewResolver();
        vr.setPrefix("/WEB-INF/views/");
        vr.setSuffix(".jsp");
        return vr;
    }
}
```

🛖6. Controller Example

```
@Controller
public class HomeController {

    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute("message", "Welcome to Embedded Tomcat Spring
MVC!");
        return "home";
    }
}
```

🛖7. JSP View (home.jsp)

Located at:

```
src/main/webapp/WEB-INF/views/home.jsp
```

```
<html>
<head><title>Home</title></head>
<body>
    <h1>${message}</h1>
</body>
</html>
```

🛖8. How JSP is Parsed Using Tomcat Jasper

Normally JSP → Servlet conversion happens **inside Tomcat**. But with embedded Tomcat, we must explicitly include **jasper**.

Why Jasper?

Because **JSP must be compiled into servlet Java code**.

Jasper handles:

✔️Parsing the JSP file
✔️Converting it to Java servlet code
✔️Compiling it into a `.class` file
✔️Executing it on request

Without Jasper **your JSP will not work** in embedded Tomcat.

🛖9. How DispatcherServlet Works Internally

```
Browser → Embedded Tomcat (HTTP Server) → DispatcherServlet
           ↓
        HandlerMapping
           ↓
        HomeController
           ↓
          ViewResolver
           ↓
         home.jsp
           ↓
       HTML Response
```

`DispatcherServlet` is the **front controller**. It routes incoming requests to the correct controller.

🛖10. Summary

Feature	Spring MVC (This Note)
Embedded Tomcat	✔️Manual setup
JSP Support	✔️via Jasper
DispatcherServlet	✔️Manually registered
web.xml	❌No (Java config used)
Autoconfiguration	❌No
Controller Mapping	✔️Manual with @GetMapping
ViewResolver	✔️Manual setup

🌆Final Result

You now have a complete **Spring MVC application using Embedded Tomcat + JSP** — WITHOUT Spring Boot.

If you want, I can also add:

✅Form submission example
✅Session handling
✅DAO + JDBC integration
✅Filters + Interceptors

Just say **"add form example"**, **"add JDBC"**, etc.