🌐Spring MVC With External Tomcat Server — Complete Beginner-Friendly Guide

This note explains how to build a **Spring MVC application** that runs on an **external Apache Tomcat server** (not embedded).

It includes:

✅What is Spring MVC (classic, non-boot)
✅What is Apache Tomcat (external)
✓ How to install + configure Tomcat in IntelliJ / Eclipse
✓ How to create a Spring MVC project (WAR)
✓ How to configure `web.xml`
✓ How DispatcherServlet works
✓ How to set up JSP views
✓ How to deploy the WAR to Tomcat

🧰1. What is Spring MVC?

Spring MVC is a Web Framework from the Spring ecosystem that follows the **Model–View–Controller** architecture.

Before Spring Boot existed, Spring MVC used:

- `web.xml` deployment descriptors
- External Tomcat server
- WAR packaging
- Manual view resolver configuration

Spring Boot automated all of this, but classic Spring MVC still works exactly the same.

🧰2. What Is an External Tomcat Server?

External Tomcat = you download Tomcat separately and run it as a standalone server.

External Tomcat responsibilities:

- Starts on its own via `startup.sh` / `startup.bat`
- Hosts Java web applications (WAR files)
- Loads servlets via `web.xml`
- Manages HTTP requests
- Compiles JSPs via Jasper

You **deploy** your application into Tomcat's `webapps` folder.

🧰 3. Setting Up External Tomcat

✔️ Step 1 — Download Tomcat

From official site: https://tomcat.apache.org

Choose version 10 or 9 (depending on Jakarta vs javax).

✔️ Step 2 — Extract ZIP

Unzip anywhere.

✔️ Step 3 — Configure in IntelliJ

```
File → Settings → Build Tools → Application Servers → Add Tomcat
```

Or in **Eclipse**:

```
Servers tab → New → Apache Tomcat
```

✔️ Step 4 — Add Tomcat as Server in Project

IntelliJ → Add configuration → Tomcat Server → Local

🧰 4. Creating a Spring MVC Web Application (WAR project)

Use Maven archetype or IDE template:

Folder structure:

```
src/main/java
src/main/resources
src/main/webapp
    └── WEB-INF
          └── web.xml
          └── views
                └── home.jsp
```

🛅5. pom.xml Dependencies (Spring MVC + JSP)

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

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

    <!-- JSP Support -->
    <dependency>
        <groupId>jakarta.servlet.jsp.jstl</groupId>
        <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
        <version>3.0.0</version>
    </dependency>
</dependencies>
```

🛅6. Configure web.xml (Very Important)

This is how you register `DispatcherServlet`.

```xml
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         version="6.0">

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</
servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

```

```
</web-app>
```

This tells Tomcat: - Create a servlet named **dispatcher** - Map all URLs `/*` to it - Let Spring MVC handle routing

---

💼7. Spring MVC Java Configuration (dispatcher-servlet.xml)

Spring loads this file automatically because its name matches `dispatcher`.

Location: `WEB-INF/dispatcher-servlet.xml`.

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans https://
www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context https://
www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/mvc https://
www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!-- Enable Spring MVC -->
    <mvc:annotation-driven />

    <!-- Scan @Controller classes -->
    <context:component-scan base-package="com.example" />

    <!-- JSP View Resolver -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

💼8. Controller Example

```
@Controller
public class HomeController {

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

💼9. JSP View (WEB-INF/views/home.jsp)

```
<html>
<body>
    <h1>${msg}</h1>
</body>
</html>
```

💼10. Build WAR and Deploy to Tomcat

Run:

```
mvn clean package
```

This generates:

```
target/myapp.war
```

Copy it to:

```
apache-tomcat/webapps/
```

Start Tomcat:

```
bin/startup.sh   (Linux/Mac)
bin/startup.bat  (Windows)
```

Open:

```
http://localhost:8080/myapp/
```

🧳11. Spring MVC Request Flow (External Tomcat)

```
Browser
  ↓
External Tomcat
  ↓
DispatcherServlet (from web.xml)
  ↓
HandlerMapping
  ↓
Controller Method
  ↓
InternalResourceViewResolver
  ↓
JSP
  ↓
HTML Response
```

🟩Summary — Spring MVC With External Tomcat

Feature	Description
Server	External Tomcat installation
Packaging	WAR
DispatcherServlet	Defined in web.xml
View Rendering	JSP (via Tomcat Jasper)
Autoconfiguration	❌No
You must configure	Controller scanning, ViewResolver, DispatcherServlet

‼️ Want More?

I can also add: - ☸️Form submission example - ☸️Session management - ☸️JDBC + DAO layer - ☸️ Filters & Interceptors - ☸️JSP → Servlet conversion diagram

Just tell me! 🚯