Deploying Spring Boot WAR file with JSP to Tomcat

Deploy Spring Boot apps with JSP to Tomcat

You can deploy a Spring Boot application using JSP to Tomcat. In this scenario, we will create a WAR file and deploy the WAR to the Tomcat server running externally. This is known as a traditional deployment.

High-level steps

1. Update main Spring Boot application

2. Update Maven POM file

3. Update application.properties

4. Move JSP view files to WEB-INF/view

5. Create WAR file

6. Deploy to Tomcat

Spring Boot Reference Manual

For full details on this process, see the Spring Boot Reference Manual: Section 92.1 Creating a Deployable WAR file

Working Example

I have a full working project. You can download this app and perform test deployments to Tomcat

Download: deploy-spring-boot-and-jsp-on-tomcat.zip

This app is a very simple helloworld example that exposes a "/test" request mapping

```java
package org.demo.bootjsp.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HelloWorldController {

    @RequestMapping("/test")
    public String sayHello() {
        return "hello";
    }

}
```

and a simple JSP page: hello.jsp

```jsp
    <html><body>

    <p>
    Hello World! Time is <%= new java.util.Date() %>
    </p>

    <p>
    We are running on  <%= application.getServerInfo() %>!!!
    </p>

    </body></html>
```

Detailed steps

1. Update main Spring Boot application

In your main Spring Boot application, you need to

a. extend the SpringBootServletInitializer

b. override the configure(...) method

Your code should look like this

```
package org.demo.bootjsp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class DemowebApplication extends SpringBootServletInitializer {

         @Override
         protected SpringApplicationBuilder configure(SpringApplicationBuilder applicat
ion) {
                  return application.sources(DemowebApplication.class);
         }

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

}
```

2. Update Maven POM file

Update your POM.xml to use WAR packaging

`<packaging>war</packaging>`

In POM.xml, add dependency to be able to compile JSPs

```
<dependency>
         <groupId>org.apache.tomcat.embed</groupId>
         <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

Make sure the Tomcat embedded does not interfere with external Tomcat server

```
<dependency>
         <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
```

3. Update application.properties

In your application.properties file, you should have

```
spring.mvc.view.prefix=/WEB-INF/view/
spring.mvc.view.suffix=.jsp
```

4. Move JSP view files to WEB-INF/view

Move your JSP view pages should to `src/main/webapp/WEB-INF/view`

5. Create WAR file

Create the WAR file with the command: `mvn clean package`

This will generate a WAR file in your project directory: **target/bootjspdemo.war**

6. In Eclipse, stop all servers you may have running

7. Outside of Eclipse, run your Tomcat server

8. Copy your WAR file to the **<<tomcat-install-dir>>/webapps**directory

Wait for about 15-30 seconds for Tomcat to deploy your app. You will know your app is deployed when you see a new folder created based on your WAR file name. In our example, you will see a new directory named: **bootjspdemo**

9. In a web browser, access your app at: `http://localhost:8080/bootjspdemo/test`

Replace <<bootjspdemo>> with the name of your WAR file if you are using a different app

If everything is successful, you will see your application's web page.

Congratulations! You deployed a Spring Boot WAR file with JSP on to a Tomcat server :-)