Deploy Spring Boot WAR file with Thymeleaf to Tomcat

Deploy Spring Boot apps with Thymeleaf to Tomcat

You can deploy a Spring Boot application as a WAR file to Tomcat. In this scenario, we will use Thymeleaf as the view template.

We will create a WAR file and deploy the WAR to the Tomcat server. This is known as a traditional deployment.

High-level steps

1. Update main Spring Boot application

2. Update Maven POM file

3. Create WAR file

4. 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-war-with-thymeleaf-on-tomcat.zip

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

```java
package com.luv2code.deploydemo.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 Thymeleaf page: hello.html

```html
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<body>

<h3>Hello World from Thymeleaf!</h3>

<p>
We are running on <span th:text="${#servletContext.getServerInfo()}"></span>!!!
</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 com.luv2code.deploydemo;

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 DeploydemoApplication extends SpringBootServletInitializer {

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

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

}
```

2. Update Maven POM file

Update your POM.xml to use WAR packaging

`<packaging>war</packaging>`

The WAR packaging should appear just after your Maven coordinates (group, artifact, version)

```
	<groupId>com.luv2code</groupId>
	<artifactId>deploydemo</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>war</packaging>
```

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. Create WAR file

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

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

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

5. Outside of Eclipse, run your Tomcat server

6. 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: **deploydemo**

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

Replace <<deploydemo>> 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 Thymeleaf on a Tomcat server :-)