


# Deploy a Spring Boot REST App as a WAR to Tomcat 10

 [appsdeveloperblog.com/deploy-a-spring-boot-rest-app-as-a-war-to-tomcat-10](https://appsdeveloperblog.com/deploy-a-spring-boot-rest-app-as-a-war-to-tomcat-10)

David Mbochi

May 22, 2021

In this tutorial, the reader will learn how to deploy a Spring Boot REST app to Tomcat 10.

According to [Apache](#), the Jakarta EE platform is the evolution of the Java EE platform. Tomcat 10 and later implement specifications developed as part of Jakarta EE. Tomcat 9 and earlier implement specifications developed as part of Java EE. Due to this reason applications developed for Tomcat 9 and earlier will not run on Tomcat 10. However, there is still a way to do it.

Java EE-based applications designed for Tomcat 9 and earlier may be placed in the \$CATALINA\_BASE/webapps-javaee directory and Tomcat will automatically convert them to Jakarta EE and copy them to the webapps directory. This conversion is performed using the [Apache Tomcat migration tool for the Jakarta EE tool](#) which is also available as a separate download for [off-line](#) use.

To demonstrate how to deploy a Spring Boot REST application to a standalone Tomcat 10 server, in this tutorial, we will do the following:

1. Download and install Apache Tomcat 10,
2. Create and build a very simple Spring Boot REST application,
3. Deploy WAR file to a standalone Tomcat 10.

## Tomcat 10 Installation

Use the following link to download Tomcat 10.

### [Tomcat 10 Download](#)

For windows users, add the following to environment variables with the name CATALINA\_HOME.

```
C:\Program Files\Apache Software Foundation\Tomcat 10.0
```

To start Tomcat issue the following command on windows command prompt.

```
C:\Program Files\Apache Software Foundation\Tomcat 10.0\bin>startup
```

If the configurations are correct the following will be the output after starting Tomcat.

```
C:\Program Files\Apache Software Foundation\Tomcat 10.0\bin>startup
Using CATALINA_BASE: "C:\Program Files\Apache Software Foundation\Tomcat 10.0"
Using CATALINA_HOME: "C:\Program Files\Apache Software Foundation\Tomcat 10.0"
```

Using CATALINA\_TMPDIR: "C:\Program Files\Apache Software Foundation\Tomcat 10.0\temp"

Using JRE\_HOME: "C:\Program Files\Java\jdk-11.0.10"

Using CLASSPATH: "C:\Program Files\Apache Software Foundation\Tomcat 10.0\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat 10.0\bin\tomcat-juli.jar"

Using CATALINA\_OPTS: ""

C:\Program Files\Apache Software Foundation\Tomcat 10.0\bin>

## Creating a Spring Boot Application

Navigate to [Spring Initializr](#) and create a new project with the dependencies shown below. Extract the generated zip folder and import the project to IntelliJ or your preferred IDE.

The screenshot shows the Spring Initializr web application interface. It includes a sidebar with a hamburger menu and a search icon. The main content area is divided into several sections: 'Project' with radio buttons for 'Maven Project' (selected) and 'Gradle Project'; 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for versions 2.5.1 (SNAPSHOT), 2.5.0 (selected), 2.4.7 (SNAPSHOT), 2.4.6, 2.3.12 (SNAPSHOT), and 2.3.11; 'Project Metadata' with input fields for 'Group' (com.example.javadev), 'Artifact' (tomcat10), 'Name' (tomcat10), 'Description' (Demo project for Spring Boot), and 'Package name' (com.example.javadev.tomcat10); 'Packaging' with radio buttons for 'Jar' and 'War' (selected); and 'Java' with radio buttons for versions 16, 11 (selected), and 8. On the right, the 'Dependencies' section shows 'Spring Web' with a 'WEB' tag and a description. At the bottom, there are buttons for 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'. A social media share icon is also visible on the left.

Set the packaging type to WAR.

```
<packaging>war</packaging>
```

Since we are going to use an external Tomcat, add provided tag to tomcat dependency in pom.xml.

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
<scope>provided</scope>
</dependency>
```

In the main application extend `SpringBootServletInitializer` and `@Override` configure method to enable us to run the application from Tomcat the classical way.

```
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 Tomcat10Application extends SpringBootServletInitializer {
    public static void main(String[] args) {
        SpringApplication.run(Tomcat10Application.class, args);
    }
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
        return builder.sources(Tomcat10Application.class);
    }
}

```

### **Create a User model**

Create a class named User in a package called model with the following fields.

- firstName
- lastName
- email

```

public class User {
    private String firstName;
    private String lastName;
    private String email;
    public User(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }
    // Generate toString method
}

```

### **Create a User Controller**

Create a class named UserController in a package called the controller and create a user object which will be returned when a user issues the following request.

`/user/read/one`

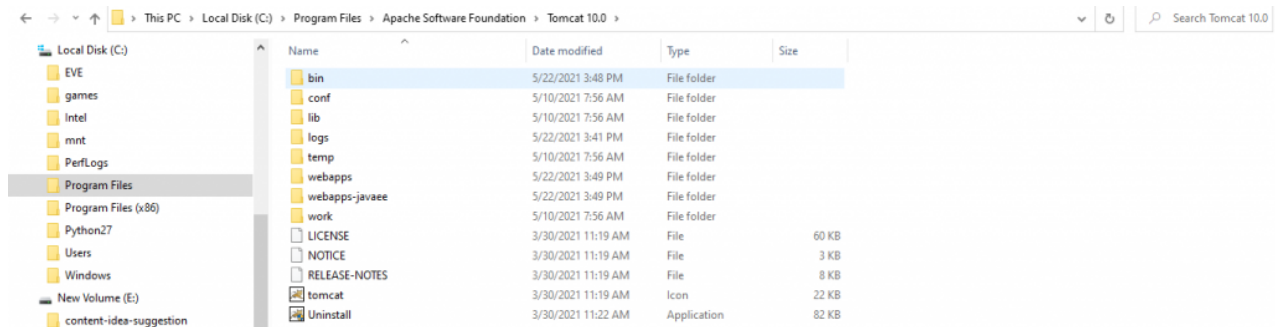
```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/user")
public class UserController {
    @GetMapping("/read/one")
    public String readUser(){
        User user = new User("John","doe","John@javadev.com");
        return user.toString();
    }
}

```

## Deploying the application on Tomcat 10

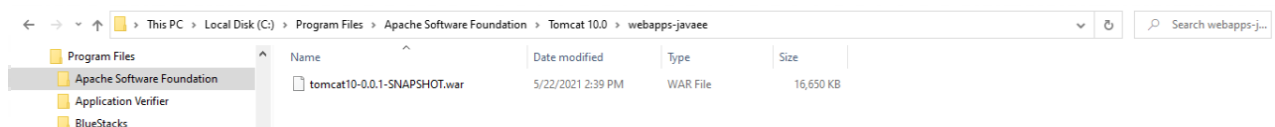
Navigate to the Tomcat installation directory mostly found under program files and create a folder named webapps-javaee.



On a command prompt, navigate to your project source, and generate a WAR file using the following command.

```
mvnw verify
```

Add the generated WAR file to webapps-javaee folder where it will be automatically migrated by Tomcat and added to webapps folder where it can then be deployed to the client.



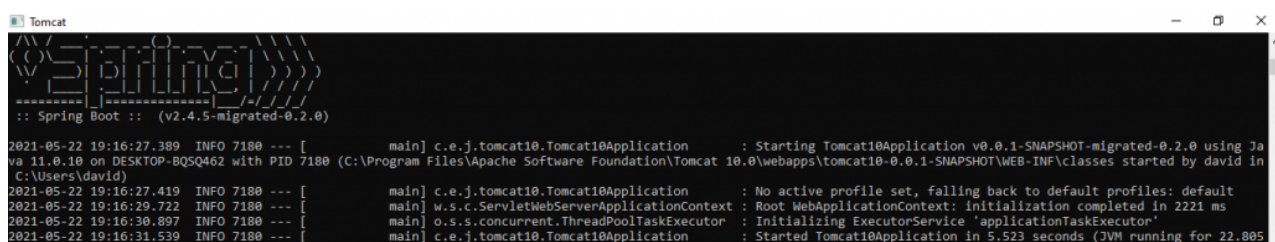
Restart Tomcat and note the following logs on server startup.

## Using jakartaee-migration Tool

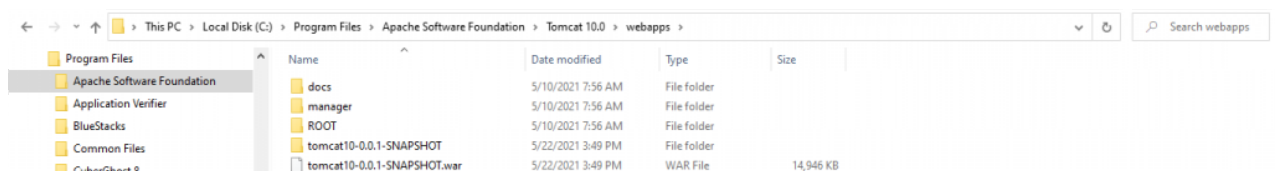
Alternatively, the application can be migrated using the following command.

```
java -jar jakartaee-migration-*-shaded.jar <source> <destination>
```

Once the command has executed successfully, restart Tomcat.

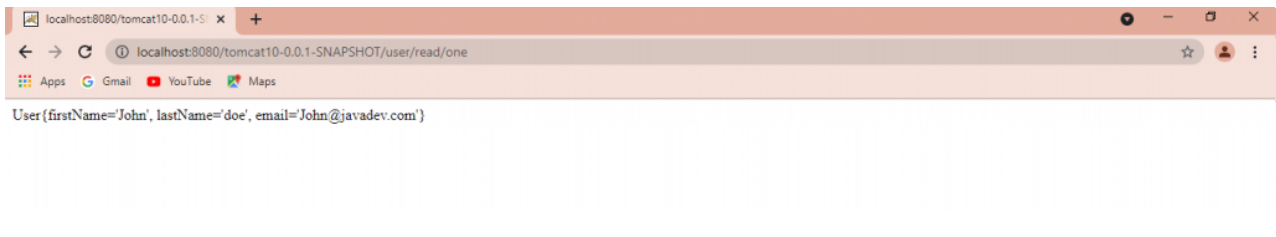


Once you restart the server, Tomcat will migrate the application automatically and add it to webapps folder under Tomcat installation.



On your preferred browser issue the following link which will return a user object that was created in the controller.

[HTTP://localhost:8080/tomcat10-0.0.1-SNAPSHOT/user/read/one](http://localhost:8080/tomcat10-0.0.1-SNAPSHOT/user/read/one)



## Conclusion

---

In this tutorial, you have learned how to deploy a Spring Boot REST application to Tomcat 10 by leveraging the default functionality provided by Tomcat 10 to migrate Java EE applications. You have observed how Tomcat 10 migrates the application behind the scene and generates an application that can be run on Tomcat 10.

Happy coding!