1. Mainly configuration changes
2. No change – controllers and components
3. Starters – let starters manage transient dependencies

See link - <https://github.com/spring-projects/spring-boot/tree/main/spring-boot-project/spring-boot-starters>

1. Entry point – each application will have an entry point with main method and @SpringBootApplication
2. Question convert xml configuration to annotation based
3. Migrate resources in files :
   1. /resuources , /public, static /META-INF/resources

We have to move all our files at above locations , we can customize resource location by customizing below

spring.resources.static-locations=classpath:/images/,classpath:/jsp/

**Migrating Spring Web Applicaion**

1. spring-boot-starter-web – it will automatically get all the web related dependencies
2. No need of – WebInitializer class which do the setup of DispatcherServlet , then no need, no need of @EnableWebMVC
3. Auto configuration will happen
   1. Support for static content in classpath /static,/public, /resources, /META-INF/resources
   2. HTTPMessageConverter – Seriliation and Deserialization – JSON-OBJ and visaversa
   3. /error mapping

Note – Its already done in eclipse , but with Springboot no explicit definition is needed

**View Technologies**

1. Template files should be placed in template folder /template/resources

**Embeded WebServer**

1. Apache ,jettey and underdow

**Migrate Spring Data Application**

Not so useful - Next Site -https://dzone.com/articles/migrating-a-legacy-spring-application-to-a-spring

1. Need to find out how the application launches in old application - SpringApplicationLauncher.launch(Constants.CONTEXT\_XML);

**Useful -** [**https://www.javadevjournal.com/spring-boot/migrating-from-spring-to-spring-boot/**](https://www.javadevjournal.com/spring-boot/migrating-from-spring-to-spring-boot/)

Almost same as first website in this document

1. Customize static resources as follows :

spring.resources.static-locations=classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/ # Locations of static resources.

Spring-mvc-starter – will remove all the explicit dependencies like

Jacson Mapper, jsp prefixes will persist

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

1. Deploy a war file – Use SpringBootServletInitializer

**Spring Integration** > Kunal - <https://spring.io/projects/spring-integration>

**https://stefan.kapferer.ch/2015/04/25/convert-spring-web-application-to-spring-boot/**

1. Add spring-boot-starter dependency
2. Remove traditional webapp folder

Webapp

{

Configuration files , web.xml and property files

}

SpringBoot this is not needed – move the files to src/main/resources/META-INF

renamed: src/main/webapp/WEB-INF/config/database.properties -> src/main/resources/META-INF/database.properties

renamed: src/main/webapp/WEB-INF/config/repository-context.xml -> src/main/resources/META-INF/repository-context.xml

renamed: src/main/webapp/WEB-INF/config/rest-services-config.xml -> src/main/resources/META-INF/rest-services-config.xml

renamed: src/main/webapp/WEB-INF/config/security-config.xml -> src/main/resources/META-INF/security-config.xml

deleted: src/main/webapp/WEB-INF/web.xml

1. Add Maven plugin

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

<version>${spring-boot-version}</version>

</plugin>

</plugins>

</build>

1. We will remove the xml configuration of dispatcher servlet -> web.xml will be removed and java will be used by implementing WebApplicationInitializer @override – onStartup()

**https://www.baeldung.com/spring-xml-vs-java-config**

<https://medium.com/gumgum-tech/migration-from-vanilla-spring-mvc-to-spring-boot-cba2aa824e>

1. WebMvcConfigurer – no need in boot

Issues :

1. For customer validations need to put hibernate validator

Explicitly this has to be added

<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>2.0.0.Final</version>

</dependency>

1. Now migrated the resource.properties in default SpringBoot folder – Or we can also change the location : spring.resources.static-locations=classpath:/images/,classpath:/jsp/
2. Standard properties names : <https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#application-properties.core>
3. Spring-boot-starter-web > Everything needed for spring web MVC.

Good article –

1. <https://www.javatpoint.com/spring-boot-starter-web> - General

<https://stackoverflow.com/questions/43225549/dispatcherservlet-and-web-xml-in-spring-boot/55701504> - web.xml related e.g if we want to keep web.xml then we can use : <listener>

<listener-class>org.springframework.boot.legacy.context.web.SpringBootContextLoaderListener</listener-class>

</listener>

**Note :**  **Spring-boot prefer annotations over xml**

Steps :

1. convert xml to annotation

2. make beans @bean if external bean is to be used

5. <https://www.javatpoint.com/restful-web-services-spring-boot-auto-configure>

// imp. Jar starter-web, Lombok, hibernate, validation,

1. Webapplicationinitializer – if we use default .jar , we do not need it. But if we want to build and deploy war file we need it
2. Maven plug in :

***spring-boot-maven-plugin***

- f you intend to deploy your Spring Boot application in an existing Servlet container

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

1. <https://stackoverflow.com/questions/44131612/spring-boot-where-to-place-the-jsp-files> - I think v useful kunal