**Spring Boot:**

* **Goals:**

Enable building production ready applications quickly.

Provide common non-functional features.

Embedded servers.

Metrics.

Health checks.

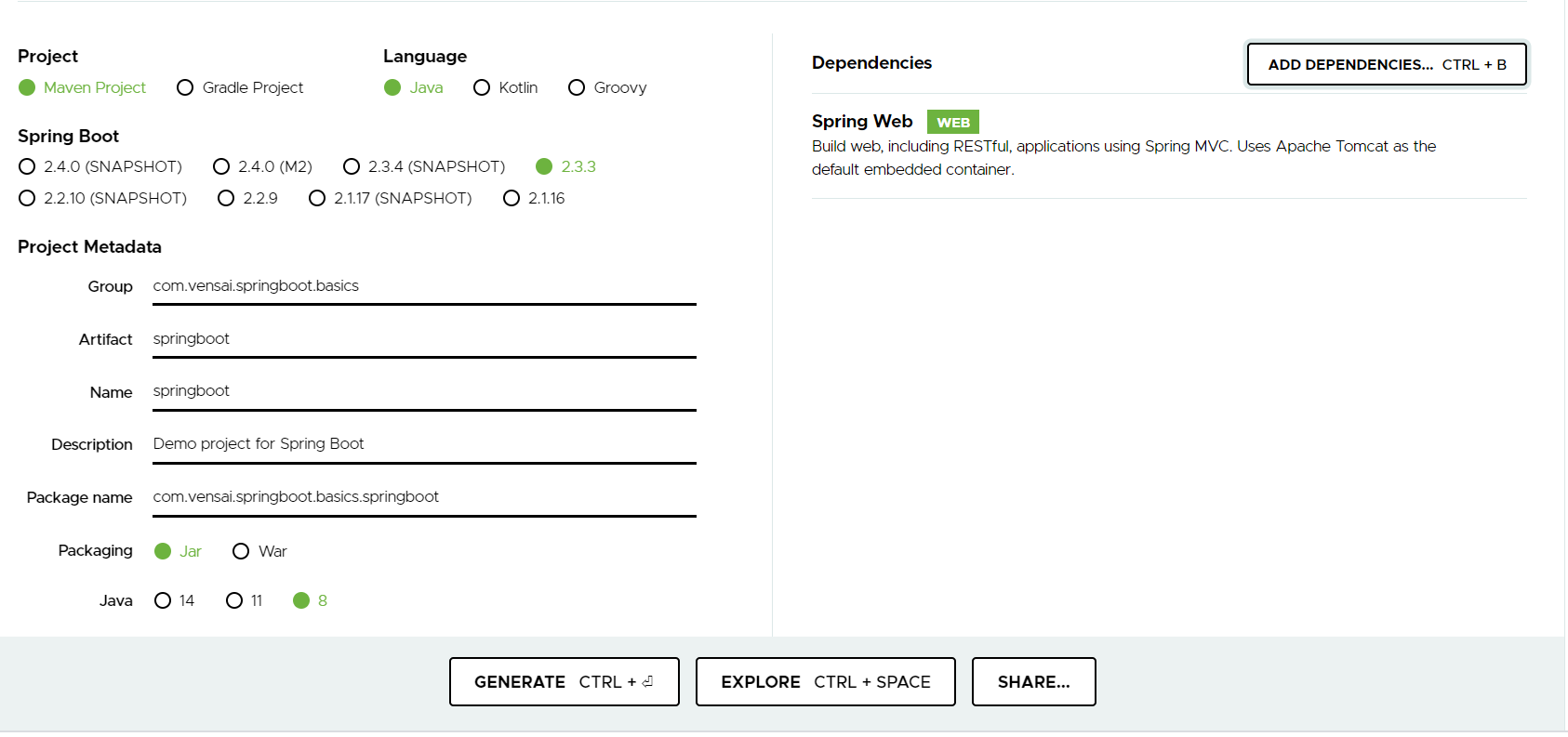
Externalized configuration

Spring boot provides great integration with Tomcat and JT and undertow

No code generation.

**Using Spring Initializr to create a project:**

The way we can crate a spring boot project is 🡪 Go to start.spring.io in your browser And select the build tool and language and spring boot version.



Once you click on generate it will create a Zip file unzip it to any folder on your hard disc.

* Go to eclipse and import the above file as MAVEN project.

Once the project imported you can see lot of Maven dependency’s which are coming in .

So if I open that up and go to the pom dot xml, you’d see that Spring Boot starter web is configured in here.

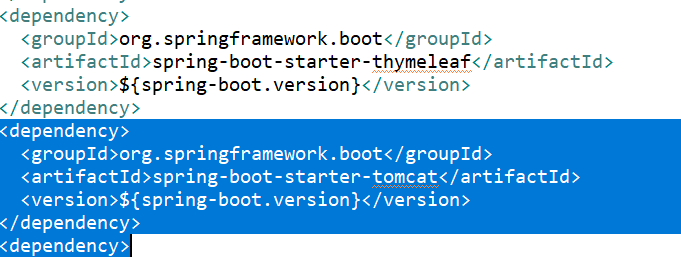
So this Spring Boot starter web is the preferred stater for Spring Boot to develope web applications as well as RESTful web services.

So if I go control and hover over it or command and hover over it and click open pom dot xml for Spring Boot

starter web you would see that it defines varied kind of dependencies.

So what does it really define?

Let's look at it. When we run the application, we were automatically using Tomcat. That’s because of this Spring Boot



If you look at Spring Boot starter web, it contains a dependency on Spring Boot starter Tomcat and that's how the application is automatically running in Tomcat.

The other things which you would seen here are Spring web and Spring web MVC. Spring MVC

as you would know is this Spring MVC framework. If I add in a Spring Boot starter web, I automatically get Spring

MVC, I automatically get validation.

**Validation** -

the default implementation for Java validation API is hibernate validator. So I get that as well and I get Tomcat. And, also I get the starter JSON. Starter JSON is more for RESTful web services.

We saw that when we actually invoke the RESTful web service, the conversion to JSON automatically happen. If you actually looked at our code, We just returned the bean back. So in the booksController, what we're doing is we are just returning a list of beans back.

How does it automatically convert it to JSON? That's done through this starter. The starter JSON brings in the JSON into the picture. Basically when you look at all starter projects, they have a set of dependencies that are already defined. And these dependencies are automatically provided to our project.

So when I add in Spring Boot starter web, I get all these dependencies. So you'd see that I'm getting

Jackson for handling all the JSON stuff. You'd see that I'm getting Tomcat. I'm getting hibernate validator and validation API for doing the validation. And, also I would get the logging frameworks as well because the logging frameworks are by default defined in the Spring Boot starter. So Spring Boot starter is one of the dependencies in Spring Boot starter web.

This has a dependency on Spring Boot auto configure. That's how we get auto configuration.

This has a dependency on Spring Boot starter logging. That's how we would get logging.

And we have a dependency on Spring core as well.

YML is one of the configuration languages that Spring Boot supports. So we also have snake YML in here.

Other than that, another important starter which is frequently used as a **Spring Boot starter JPA.**

If you know JPA, JPA is kind of the interface for hibernate. So JPA defines how ORM applications or ORM frameworks

should work. ORM is object relational mapping. Let's now add in a dependency on Spring Boot starter JPA. You

would see now that there would be a lot more Maven dependence which would come in.

Actually, it's Spring Boot starter data JPA.

So now we have hibernate core, hiberante JPA. We have javax transaction api, hibernate common annotations. You have spring data, spring data JPA, spring ORM, spring tx and a lot of such dependencies come in. That's because of the Spring Boot starter JPA. So if you open up the pom dot xml for spring boot starter JPA then you would see all of the dependencies that are listed down. So this has a dependency on spring boot starter AOP, starter JDBC, hiberante transaction API and spring data JPA.

So all this jars you'd get. As soon as I add in a spring starter data JPA, I would get all these things for free.

**Different Spring Boot Starter Projects :**

As we see from Spring Boot Starter Web, starter projects help us in quickly getting started with developing specific types of applications.

* spring-boot-starter-web-services - SOAP Web Services
* spring-boot-starter-web - Web & RESTful applications
* spring-boot-starter-test - Unit testing and Integration Testing
* spring-boot-starter-jdbc - Traditional JDBC
* spring-boot-starter-hateoas - Add HATEOAS features to your services
* spring-boot-starter-security - Authentication and Authorization using Spring Security
* spring-boot-starter-data-jpa - Spring Data JPA with Hibernate
* spring-boot-starter-cache - Enabling Spring Framework’s caching support
* spring-boot-starter-data-rest - Expose Simple REST Services using Spring Data REST

There are a few starters for technical stuff as well

* spring-boot-starter-actuator - To use advanced features like monitoring & tracing to your application out of the box
* spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat - To pick your specific choice of Embedded Servlet Container
* spring-boot-starter-logging - For Logging using logback
* spring-boot-starter-log4j2 - Logging using Log4j2