Introducing to Spring Boot

Spring has been a great framework for years however it had few drawbacks. In this tutorial, An Introduction to Spring Boot we will see how Spring Boot has not only addressed the drawbacks but also supports modern software architecture. Spring Boot is around for sometime now.

Spring Team has released one of major innovation on the top of existing Spring Framework is Spring Boot. It is a completely new project from Pivotal Team (The Spring Team). Spring Boot is their latest innovation to keep up to date with the changing technology needs. The primary motivation behind developing Spring Boot is to simplify the process for configuring and deploying the spring applications.

Spring Boot offers a new paradigm for developing Spring applications with minimal friction. With Spring Boot, you'll be able to develop Spring applications with more agility and be able to focus on addressing your application's functionality needs with minimal (or possibly no) thought of configuring Spring itself. It uses completely new development model to make Java Development very easy by avoiding some tedious development steps and boilerplate code and configuration.

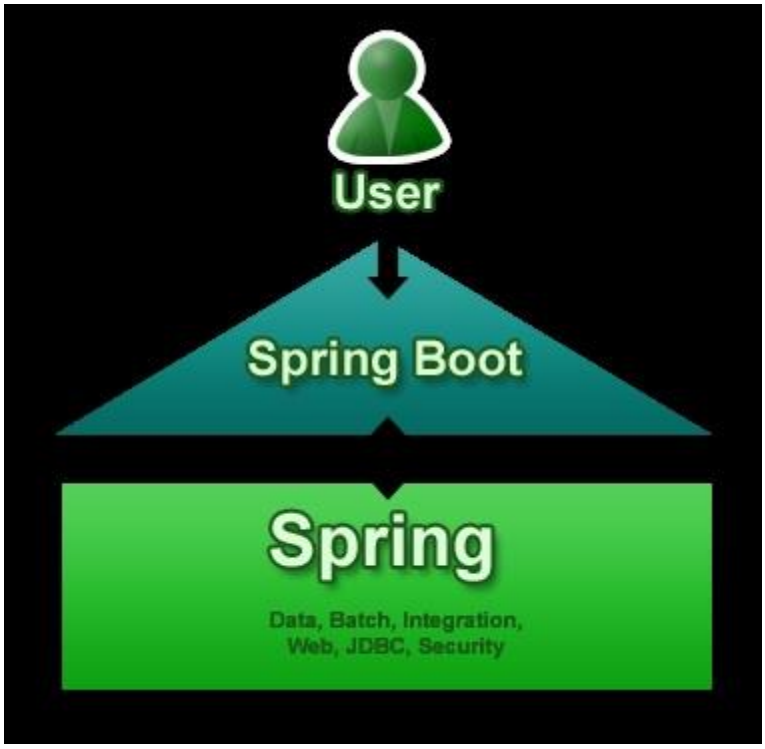
Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

You can use Spring Boot to create Java applications that can be started using java -jar or more traditional war deployments. We also provide a command line tool that runs "spring scripts".

What is spring boot?

First of all Spring Boot is not a framework, it is a way to ease to create enterprise Apps with minimal or zero xml configurations. It is approach to develop spring based application with very less configuration. It provides defaults for code and annotation configuration to quick start new spring projects within no time. It leverages existing spring projects as well as Third party projects to develop production ready applications. It provides a set of Starter pom's or gradle build files which one can use to add required dependencies and also facilitate auto configuration.

Spring Boot automatically configures required classes depending on the libraries on its classpath. Suppose your application want to interact with DB, if there are Spring Data libraries on class path then it automatically sets up connection to DB along with the Data Source class.



Spring Boot Primary Goals

Spring Boot primary goals are:

- Provide a radically faster and widely accessible getting started experience for all Spring development.
- Be opinionated out of the box, but get out of the way quickly as requirements start to diverge from the defaults.
- Provide a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration).
- Absolutely no code generation and no requirement for XML configuration, to avoid XML Configuration completely
- To avoid defining more Annotation Configuration(It combined some existing Spring Framework Annotations to a simple and single Annotation)
- To avoid writing lots of import statements
- To provide some defaults to quick start new projects within no time.

Why New Project Need Spring Boot?

Suppose we are developing one of Hello World application in Spring Framework, for that there is only one item is specific to developing the Hello World functionality: the controller. The rest of it is generic boilerplate that you'd need for any web application developed with Spring. But if all Spring web applications need it, why should you have to provide it? So there are following things any new project need Spring Boot.

- To ease the Java-based applications Development, Unit Test and Integration Test Process.

- To reduce Development, Unit Test and Integration Test time by providing some defaults.
- To increase Productivity.
- When we talk about defaults, Spring Boot has its own opinions. If you are not specifying the details, it will use its own default configurations. If you want persistence, but don't specify anything else in your POM file, then Spring Boot configures Hibernate as a JPA provider with an HSQLDB database.
- To provide bunch of non-functional features/solutions that are very much common to large scale projects (e.g. embedded servers, security, metrics, health checks, externalized configuration).

What Spring Boot Isn't?

As above mention Spring Boot is not a framework to write applications, it helps you to develop and build, package and deploy application with minimal configuration or zero configuration. It is not an application server. But it's the embedded servlet container that provides application server functionality, not Spring Boot itself.

Similarly, Spring Boot doesn't implement any enterprise Java specifications such as JPA or JMS. For example, Spring Boot doesn't implement JPA, but it does support JPA by auto-configuring the appropriate beans for a JPA implementation (such as Hibernate) finally, Spring Boot doesn't employ any form of code generation to accomplish its magic. Instead, it leverages conditional configuration features from Spring 4, along with transitive dependency resolution offered by Maven and Gradle, to automatically configure beans in the Spring application context.

In short, at its heart, Spring Boot is just Spring.Future Spring projects would not have any XML configurations as part of it, everything will be handled by the project Spring Boot.

Pros/Cons of Spring Boot

Pros of Spring Boot:

- It is very easy to develop Spring Based applications with Java or Groovy.
- It reduces lots of development time and increases productivity.
- It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
- It follows "Opinionated Defaults Configuration" Approach to reduce Developer effort
- It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- It provides CLI (Command Line Interface) tool to develop and test Spring Boot (Java or Groovy) Applications from command prompt very easily and quickly.
- It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle.
- It provides lots of plugins to work with embedded and in-memory Databases very easily.

Limitation of Spring Boot:

It is very tough and time consuming process to convert existing or legacy Spring Framework projects into Spring Boot Applications. It is applicable only for brand new/Greenfield Spring Projects.

Spring boot releases

Latest Releases: Spring Boot 1.x and 2.x available now. You require minimum Spring Framework 4.x. or 5.x for this version.

Features Added in SpringBoot 1.4.x are:

- Executable JAR Layout
- Startup error improvements
- Hibernate 5
- Spring Framework 4.3
- Third Party Library
- Custom JSON Serializer and Deserializer
- New auto-configuration support
- Couchbase
- Neo4j
- Narayana transactional manager
- Caffeine Cache
- Actuator improvements
- Testing improvements

Getting started with Spring Boot

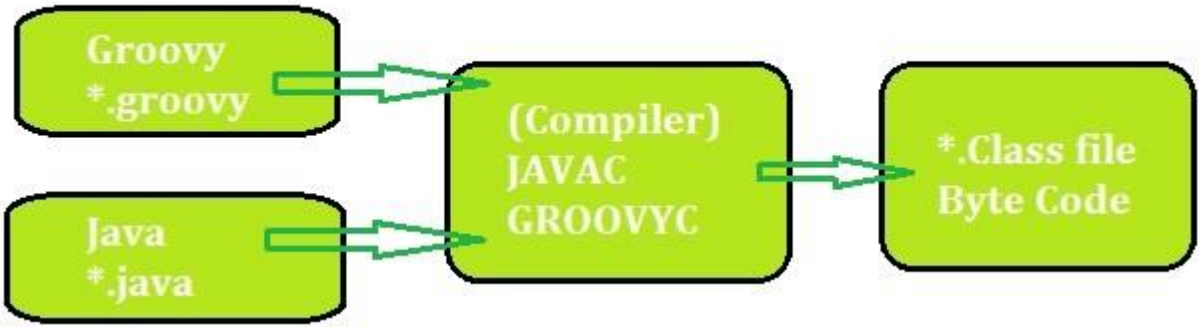
Ultimately, a Spring Boot project is just a regular Spring project that happens to leverage Spring Boot starters and auto-configuration. To Start Opinionated Approach to create Spring Boot Applications, The Spring Team (The Pivotal Team) has provided the following approaches.

- Using Spring Boot CLI Tool
- Using Spring STS IDE
- Using Spring Initializer (Website http://start.spring.io/)
- By configuring STS plugin in Eclipse IDE

We can develop two flavors of Spring-Based Applications using Spring Boot

- Java-Based Applications
- Groovy Applications

We can use Spring Boot CLI or Spring STS IDE or Spring Initializer Website to develop Spring Boot Groovy Applications. However, we can use Spring STS IDE or Spring Initializer Website to develop Spring Boot Java Applications.



Anyhow, Groovy is also JVM language almost similar to Java Language. We can combine both Groovy and Java into one Project. Because like Java files, Groovy files are finally compiled into *.class files only. Both *.groovy and *.java files are converted to *.class file (Same byte code format).

Spring Boot Framework Programming model is inspired by Groovy Programming model. Spring Boot internally uses some Groovy based techniques and tools to provide default imports and configuration.

Spring Boot Framework also combined existing Spring Framework annotations into some simple or single annotations. We will explore those annotations one by one in coming posts with some real-time examples.

Spring Boot Framework drastically changes Spring-Java Based Applications Programming model into new Programming model. As of now, Spring Boot is at initial stage only but future is all about Spring Boot only.

Spring Boot CLI

It is the easiest and quickest way to start using the Spring Boot. It is a command line tool used for executing the groovy scripts. In summary, you can install this tool by following these steps:

1. Download the binary distributions for this project from here. Spring Boot CLI requires Java JDK v1.6 or above in order to run. Groovy v2.1 is packaged as part of this distribution, and therefore does not need to be installed (any existing Groovy installation is ignored)
2. If you unpack the zip file, you will find spring.bat which will check the all the settings. This script can be found under the directory /bin.

Why Spring Boot ?

Even though Spring is a great framework it has few pitfalls. Spring Boot was build not just to address them, it also provides direction to the future of software development. Spring's XML based configuration is a nightmare in the world of annotation. There was no clear leader in the

java framework world to support Microservices. You really don't want different teams building Microservices to adapt different set of Java Libraries and look very different to each other.

What Spring Boot brings to the table ?

Convention over configuration

Spring Boot has taken away all the XML based configurations and provided Annotations for using the Spring Framework. You can start your application with a very minimum annotation in no time. This would be very helpful to the developers, the team productivity would greatly be impacted positively.

Standardization for Microservices

One of the main objective of Spring Boot is to provide a unified ecosystem of libraries & standards to all the developers (teams) utilizing Microservices methodologies. Any project adapting Microservices would have multiple teams and we certainly don't want each of team to build the softwares in very different way. The teams will be also benefit from the all the annotation and tooling Spring Boot brings, however it also provides-

1. A common platform and library support
2. Reduced setup, configuration time in development environment.
3. Cloud Support

Integrated Server for Development

Spring Boot attaches a Tomcat/Jetty server with the compiled Jar using Maven/Gradle. This helps the developers to expedite the development process by using the integrated server. In 2-3 mins you could build & test a RESTful web service from scratch.

Cloud Support

Spring Boot provides cloud support for configuration, tools and clients. It's also compatible with Cloud Native and works seamlessly with Cloud Foundry, Pivotal etc.

Adapt & Support for 3rd Party Library

Spring Boot has taken a significant step and widen support for 3rd Party Open Source Library like Netflix OSS, No-SQL DB, Distributed Cache etc. You will find a full list in the Spring Boot home page, however the seamless integration using Annotation is very powerful.

What is Spring Boot ?

In one sentence, Spring Boot is equal to (Spring Framework – XML Configuration) + Integrated Server.

Spring Boot Components

Spring Boot Auto Configure

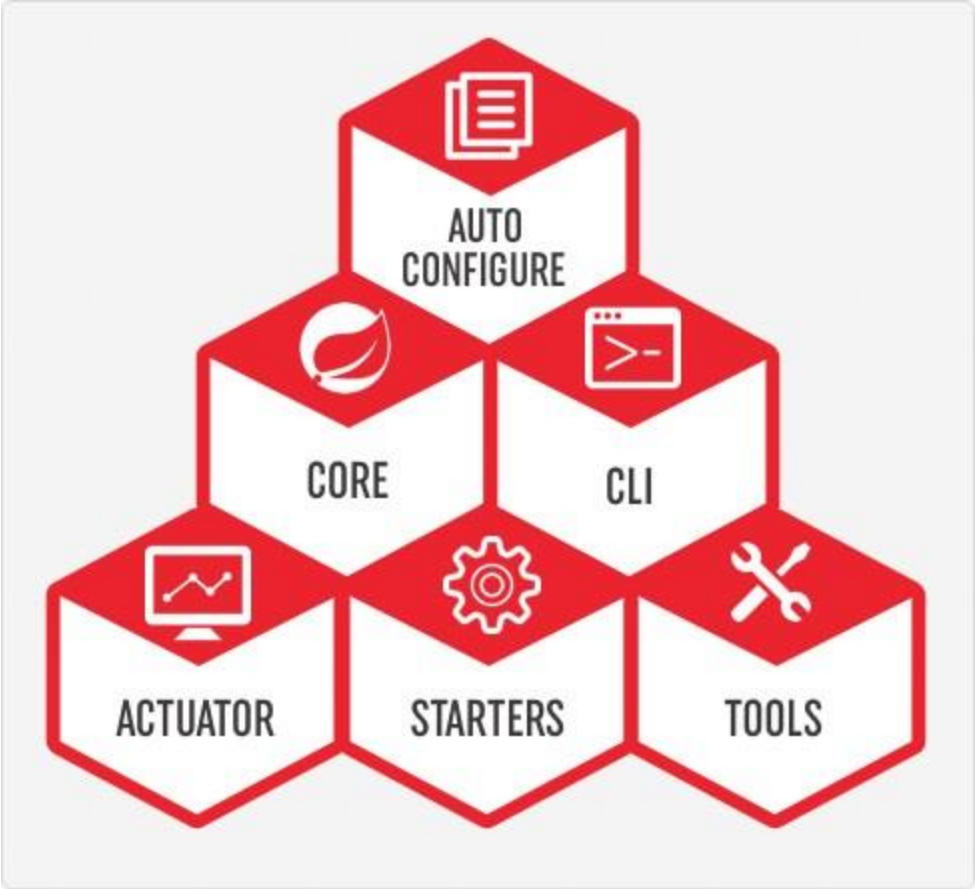
Module to auto configure a wide range of Spring projects. It will detect availability of certain frameworks (Spring Batch, Spring Data JPA, Hibernate, JDBC). When detected it will try to auto configure that framework with some sensible defaults, which in general can be overridden by configuration in an application.properties/yml file.

Spring Boot Core

The base for other modules, but it also provides some functionality that can be used on its own, eg. using command line arguments and YAML files as Spring Environment property sources and automatically binding environment properties to Spring bean properties (with validation).

Spring Boot CLI

A command line interface, based on ruby, to start/stop spring boot created applications.



Spring Boot Actuator

This project, when added, will enable certain enterprise features (Security, Metrics, Default Error pages) to your application. As the auto configure module it uses auto detection to detect certain frameworks/features of your application.For an example, you can see all the REST Services defined in a web application using Actuator.

Spring Boot Starters

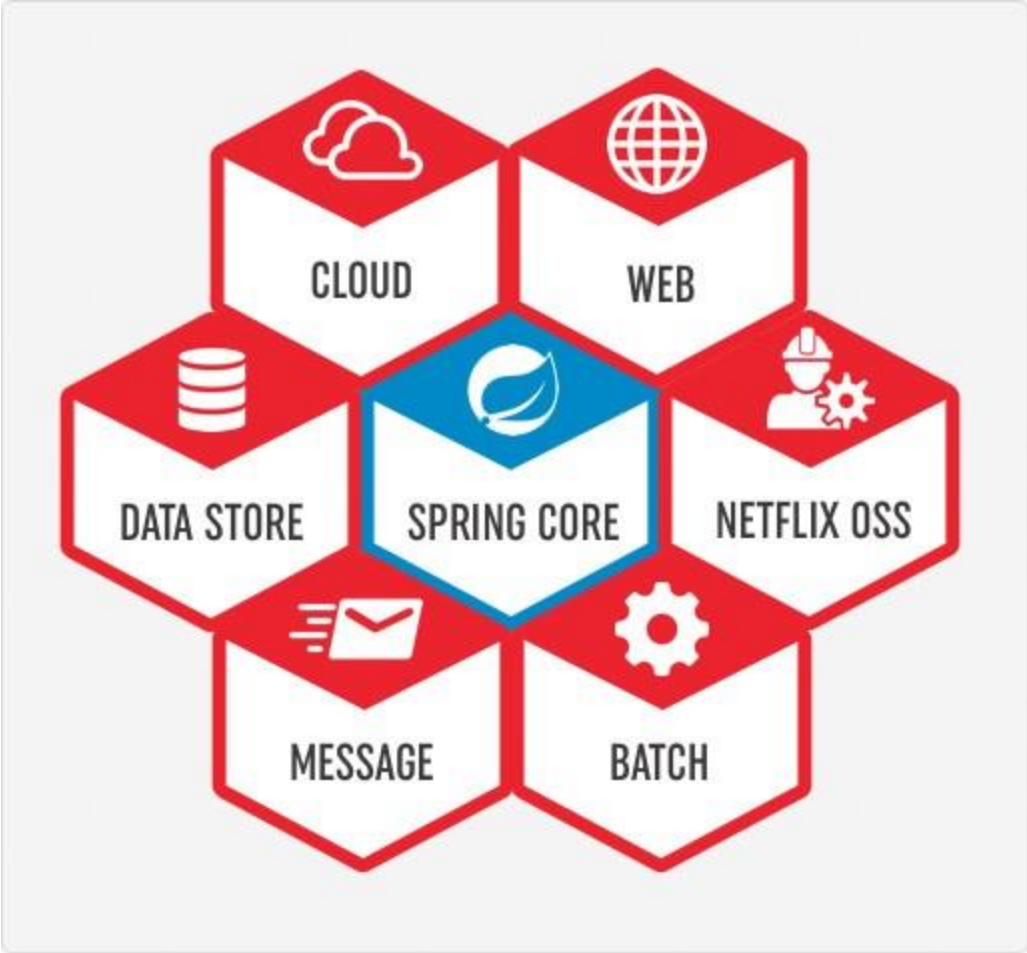
Different quick start projects to include as a dependency in your maven or gradle build file. It will have the needed dependencies for that type of application. Currently there are many starter projects (We will learn about few of them in the next section) and many more are expected to be added.

Spring Boot Tools

The Maven and Gradle build tool as well as the custom Spring Boot Loader (used in the single executable jar/war) is included in this project.

Spring Boot Starters

Spring Boot incorporates many starters packages (in Maven & Gradle) which you can include in order to add appropriate support in your project. At a high-level there are so far 6 types of starters packages available. You can find all of them in Spring Boot official WebSite.

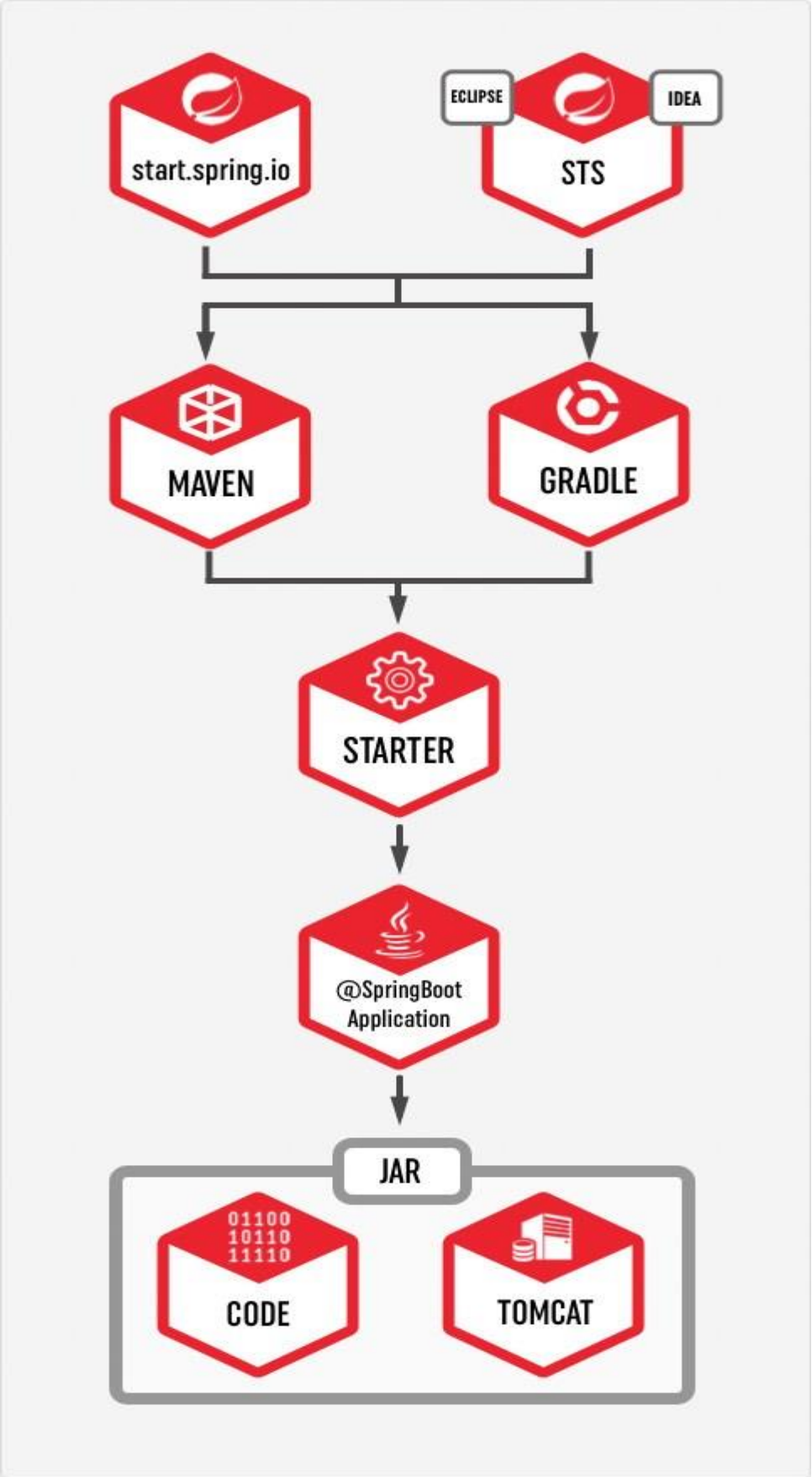


How to use Spring Boot

In another post we will see how to use and run Spring Boot, however here are the steps you need to follow.

- You can use spring initialize to create the initial setup. You can visit either start.spring.io or use STS (Spring Tool Suite) Support available in IDEA or Eclipse to choose all the Spring Boot Starters
- You need to also choose whether to use Maven or Gradle as the build tool.

- If you are using start.spring.io, you need to then download the zip and configure your workspace. Otherwise using your preferred IDE will automatically create the required file in the workspace.
- Add your code as required
- You can either use `mvn clean package` or use IDEA or Eclipse to build and create the jar file.
- By default the JAR would include integrated Tomcat server, so just by executing the JAR you should be able to use your program.


start.spring.io
ECLIPSE
IDEA
STS
MAVEN
GRADLE
STARTER
@SpringBoot
Application
JAR
01100
10110
11110
CODE
TOMCAT

Disadvantages of Spring Boot

Migration Efforts

Migration from already existing spring project to spring boot is not straightforward. Spring Boot is mostly for new development project.

Deployment to WebSphere/WebLogic Servers

Deploying Spring Boot application to WebSphere/WebLogic Servers are also not very simple. You need to make few changes like downgrading JPA Version, remove conflicting Jars etc in order to make it work in WebSphere/WebLogic Application Server.

Microservices & Cloud Native

Spring Boot has been developed keeping Micro services& Cloud Native in mind. You may not see improvements in other areas.

NOTE:

→SpringBoot starter <parent> will do following things finally based on it's version specifies other compatable versions like java,UTF and etc.

→downloads some maven or gradle plugins as discussed above that are required to run the application.

→specifies the versions of the jars that should come based on the <dependency>tags that are added.

In every springBoot project we must import one parent project that is spring-Boot-starter-parent it will give following things from central repository to multiple springBoot projects(child projects)

1. *Configuration-java version and other properties*
2. *Dependency management-version of dependency*
3. *Default plugin configuration*

The above springBoot starter parent automatically configurations the following plugings.

Maven-failsafe- plugin,maven-jar-plugin,maven-surefire-plugin.

→The most commonly used annotations in springBoot application is

@SpringBootApplication it is the combination of three annotations

@SpringBootConfiguration or @Configuration(makes current java class as configuration class.

@@EnableAutoConfiguration(based on jar files and user defined beans that are placed in application it tries configure some predefined bean automatically)