Spring Boot [Build Status] [Chat]

Spring Boot makes it easy to create Spring-powered, production-grade applications and services with absolute minimum fuss. It takes an opinionated view of the Spring platform so that new and existing users can quickly get to the bits they need.

You can use Spring Boot to create stand-alone 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.

Our 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

Installation and Getting Started

The reference documentation includes detailed installation instructions as well as a comprehensive `getting started` guide.

Here is a quick teaser of a complete Spring Boot application in Java:

```
import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.web.bind.annotation.*;

@RestController
@SpringBootApplication
public class Example {

    @RequestMapping("/")
    String home() {
        return "Hello World!";
    }

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

}
```

Getting help

Having trouble with Spring Boot? We'd like to help!

- Check the reference documentation, especially the How-to's — they provide solutions to the most common questions.
- Learn the Spring basics — Spring Boot builds on many other Spring projects, check the spring.io web-site for a wealth of reference documentation. If you are just starting out with Spring, try one of the guides.
- If you are upgrading, read the release notes for upgrade instructions and "new and noteworthy" features.
- Ask a question - we monitor stackoverflow.com for questions tagged with `spring-boot`. You can also chat with the community on Gitter.
- Report bugs with Spring Boot at github.com/spring-projects/spring-boot/issues.

Reporting Issues

Spring Boot uses GitHub's integrated issue tracking system to record bugs and feature requests. If you want to raise an issue, please follow the recommendations below:

- Before you log a bug, please search the issue tracker to see if someone has already reported the problem.
- If the issue doesn't already exist, create a new issue.
- Please provide as much information as possible with the issue report, we like to know the version of Spring Boot that you are using, as well as your Operating System and JVM version.

- If you need to paste code, or include a stack trace use Markdown ``` escapes before and after your text.
- If possible try to create a test-case or project that replicates the problem and attach it to the issue.

Building from Source

You don't need to build from source to use Spring Boot (binaries in repo.spring.io), but if you want to try out the latest and greatest, Spring Boot can be easily built with the maven wrapper. You also need JDK 1.8.

```
$ ./mvnw clean install
```

If you want to build with the regular `mvn` command, you will need Maven v3.5.0 or above.

NOTE	You may need to increase the amount of memory available to Maven by setting a `MAVEN_OPTS` environment variable with the value `-Xmx512m`. Remember to set the corresponding property in your IDE as well if you are building and running tests there (e.g. in Eclipse go to `Preferences→Java→Installed JREs` and edit the JRE definition so that all processes are launched with those arguments). This property is automatically set if you use the maven wrapper.

Also see CONTRIBUTING.adoc if you wish to submit pull requests, and in particular please fill out the Contributor's Agreement before your first change, however trivial.

Building reference documentation

First of all, make sure you have built the project:

```
$ ./mvnw clean install
```

The reference documentation requires the documentation of the Maven plugin to be available so you need to build that first since it's not generated by default.

```
$ ./mvnw clean install -pl spring-boot-project/spring-boot-tools/spring-boot-maven-
plugin -Pdefault,full
```

The documentation also includes auto-generated information about the starters. You might have that in your local repository already (per the first step) but if you want to refresh it:

```
$ ./mvnw clean install -f spring-boot-project/spring-boot-starters
```

Once this is done, you can build the reference documentation with the command below:

```
$ ./mvnw clean prepare-package -pl spring-boot-project/spring-boot-docs -Pdefault,full
```

TIP The generated documentation is available from `spring-boot-project/spring-boot-docs/target/generated-docs/reference/html`

Modules

There are a number of modules in Spring Boot, here is a quick overview:

spring-boot

The main library providing features that support the other parts of Spring Boot, these include:

- The `SpringApplication` class, providing static convenience methods that make it easy to write a stand-alone Spring Application. Its sole job is to create and refresh an appropriate Spring `ApplicationContext`
- Embedded web applications with a choice of container (Tomcat, Jetty or Undertow)
- First class externalized configuration support
- Convenience `ApplicationContext` initializers, including support for sensible logging defaults

spring-boot-autoconfigure

Spring Boot can configure large parts of common applications based on the content of their classpath. A single `@EnableAutoConfiguration` annotation triggers auto-configuration of the Spring context.

Auto-configuration attempts to deduce which beans a user might need. For example, if `HSQLDB` is on the classpath, and the user has not configured any database connections, then they probably want an in-memory database to be defined. Auto-configuration will always back away as the user starts to define their own beans.

spring-boot-starters

Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop-shop for all the Spring and related technology that you need without having to hunt through sample code and copy paste loads of dependency descriptors. For example, if you want to get started using Spring and JPA for database access just include the `spring-boot-starter-data-jpa` dependency in your project, and you are good to go.

spring-boot-cli

The Spring command line application compiles and runs Groovy source, making it super easy to write the absolute minimum of code to get an application running. Spring CLI can also watch files, automatically recompiling and restarting when they change.

spring-boot-actuator

Actuator endpoints let you monitor and interact with your application. Spring Boot Actuator provides the infrastructure required for actuator endpoints. It contains annotation support for actuator endpoints. Out of the box, this module provides a number of endpoints including the `HealthEndpoint`, `EnvironmentEndpoint`, `BeansEndpoint` and many more.

spring-boot-actuator-autoconfigure

This provides auto-configuration for actuator endpoints based on the content of the classpath and a set of properties. For instance, if Micrometer is on the classpath, it will auto-configure the `MetricsEndpoint`. It contains configuration to expose endpoints over HTTP or JMX. Just like Spring Boot AutoConfigure, this will back away as the user starts to define their own beans.

spring-boot-test

This module contains core items and annotations that can be helpful when testing your application.

spring-boot-test-autoconfigure

Like other Spring Boot auto-configuration modules, spring-boot-test-autoconfigure, provides auto-configuration for tests based on the classpath. It includes a number of annotations that can be used to automatically configure a slice of your application that needs to be tested.

spring-boot-loader

Spring Boot Loader provides the secret sauce that allows you to build a single jar file that can be launched using `java -jar`. Generally you will not need to use `spring-boot-loader` directly, but instead work with the Gradle or Maven plugin.

spring-boot-devtools

The spring-boot-devtools module provides additional development-time features such as automatic restarts, for a smoother application development experience. Developer tools are automatically disabled when running a fully packaged application.

Samples

Groovy samples for use with the command line application are available in spring-boot-cli/samples. To run the CLI samples type `spring run <sample>.groovy` from samples directory.

Guides

The spring.io site contains several guides that show how to use Spring Boot step-by-step:

- Building an Application with Spring Boot is a very basic guide that shows you how to create a simple application, run it and add some management services.
- Building a RESTful Web Service with Spring Boot Actuator is a guide to creating a REST web service and also shows how the server can be configured.
- Converting a Spring Boot JAR Application to a WAR shows you how to run applications in a web server as a WAR file.

License

Spring Boot is Open Source software released under the Apache 2.0 license.