**Spring Boot**

Pom Entry

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.3.1.RELEASE</version>

</parent>

pring-boot-starter-web – is going to integrate spring MVC and auto configure

<dependencies>

<dependency>

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

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

* Spring Boot runs just like regular java main
* To enable spring boot we need to add an annotation @SpringBootApplication
* @RestController is nothing but @controller to make class as controller to application
* Spring boot initialisers
  + <https://start.spring.io/> - Web
  + Spring Boot CLI – Command line
* Spring boot does not have web.xml
* Since no deployment descriptor can run anywhere no need configuration its very useful for cloud deployments
* The application along with spring boot, direct the embedded servlet
* you can still deploy spring boot into containers

|  |  |
| --- | --- |
| public static void main( ... ) | Starts Java and then the  application |
| @SpringBootApplication | A convenience annotation that  wraps commonly used  annotations with Spring Boot |
| @Configuration  @EnableAutoConfiguration  @ComponentScan | Spring configuration on startup  Auto configures frameworks  Scans project for Spring  components |
| SpringApplication.run( ... ); | Starts Spring, creates spring  context, applies annotations  and sets up container |

Spring Boot Static Serving – means apply the updated changes without restarting application

for simple text no need extra set-up but to if you write any logic need setup Grunt or gulp to continuously update the UI Content

Customise the auto configuration integration points of spring boot using the application properties

* application.properties // you can use YAML instead of properties
* logging.level.***org.springframework.web***= DUBUG

//it means set the logging level for the package

* Depend on environment profile configuration
* Spring Boot Profile arguments

-Dspring.profiles.active=test

* for more application properties search for **common-application-properties**

**DBA Boot**

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jpa -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

<version>2.0.0.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.h2database/h2 -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.196</version>

<scope>test</scope>

</dependency>

DB Pooling Settings

spring.datasource.**max-active=10**

spring.datasource.max-idle=8

spring.datasource.max-wait=10000

spring.datasource.min-evictable-idle-time-millis=1000

spring.datasource.min-idle=8

spring.datasource.time-between-eviction-runs-millis=1

FLYWAY-DB concepts wes higbee

Spring Boot also provides other starter projects including the typical dependencies to develop specific type 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-data-rest - Expose Simple REST Services using Spring Data REST

Spring Boot aims to enable production ready applications in quick time. Spring Boot provides a few non functional features out of the box like caching, logging, monitoring and embedded servers.

* 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-cache - Enabling Spring Framework’s caching support

spring-boot-maven-plugin provides a few commands which enable you to package the code as a jar or run the application

* spring-boot:run runs your Spring Boot application.
* spring-boot:repackage repackages your jar/war to be executable.
* spring-boot:start and spring-boot:stop to manage the lifecycle of your Spring Boot application (i.e. for integration tests).
* spring-boot:build-info generates build information that can be used by the Actuator.

### enable auto reload of my application with Spring Boot

Use Spring Boot Developer Tools.

Adding Spring Boot Developer Tools to your project is very simple.

Add this dependency to your Spring Boot Project pom.xml

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

Restart the application.

You are all Set.

If you would want to auto load the page as well, you can look at LiveReload

* http://www.logicbig.com/tutorials/spring-framework/spring-boot/boot-live-reload/.

In my trials, we found LiveReload buggy. Do let us know if you have a better experience with it.

### Q : What and Why Embedded Servers?

Think about what you would need to be able to deploy your application (typically) on a virtual machine.

* Step 1 : Install Java
* Step 2 : Install the Web/Application Server (Tomcat/Websphere/Weblogic etc)
* Step 3 : Deploy the application war

What if we want to simplify this?

How about making the server a part of the application?

*You would just need a virtual machine with Java installed and you would be able to directly deploy the application on the virtual machine. Isn’t it cool?*

This idea is the genesis for Embedded Servers.

When we create an application deployable, we would embed the server (for example, tomcat) inside the deployable.

*For example, for a Spring Boot Application, you can generate an application jar which contains Embedded Tomcat. You can run a web application as a normal Java application!*

Embedded server is when our deployable unit contains the binaries for the server (example, tomcat.jar).