A Fresh new Approach that makes developing with spring framework fun , easy and productive.

Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can run.

Spring Boot 2.1

Pre-Requisites

- Java 1.8+
- Spring Fundamentals 5.x
- Mvn
- IDE - STS

Content

- Introduction Spring Boot
- Tools and Setup
- Developing first Spring Boot App
- Application Components
- **Starters**
- **Auto-Configuration**
- Container less Deployment
- **Customizing Configuration**
- Profiles
- Logging
- **Actuator**
- **DevTool**
- **CLI**

Spring Boot Goal

- 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 (such as embedded servers, security, metrics, health checks, and externalized configuration).
- Absolutely no code generation and no requirement for XML configuration.

Spring Boot Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

Without Boot

Problem with Spring

- Huge Framework
- Multiple Setup steps
- Multiple Configuration steps
- Multiple build and deploy steps
  - Spring does a lot for you, but it demands that you do a lot for it in return.
  - Spring Boot Abstract these steps nicely.

Spring Boot

- Spring Boot is Spring's convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications that you can "just run".

- It is preconfigured with the Spring's "opinionated view" of the best configuration and use 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 by using **java -jar** or more traditional war deployments.

- Spring Boot also provide a command line tool that runs “spring scripts”.

Version History

- Spring boot 1.0.0 (April 2014)
- Spring boot 1.1 (June 2014)
- Spring boot 1.2 (March 2015)
- Spring boot 1.3 (December 2016)
- Spring boot 1.4 (January 2017)
- Spring boot 1.5 (February 2017)
- Spring Boot 2.0 (march 2018)
  - https://spring.io/blog/2018/03/01/spring-boot-2-0-goes-ga

Spring Boot 2.0

- What's New
  - infrastructure Upgrade
  - Spring Framework 5
  - Micrometer Support
- What's Changed
  - Configuration Properties
  - Actuator Endpoints
  - Security

Spring Boot 2.0

https://github.com/spring-projects/spring-boot/wiki/spring-boot-2.0-release-notes

Using Spring Boot

- Build systems
- Auto-configuration
- How to run your applications

Installation Instruction

- Use Spring Boot in the same way as any standard Java library.
- Include the appropriate spring-boot-*.jar files on your classpath.
- Spring Boot does not require any special tools integration, so you can use any IDE or text editor.
- Recommended to use with Build tools such as Maven / Gradle.

- With Maven
  - Spring Boot dependencies use the **org.springframework.boot** groupId.
  - Your Maven POM file inherits from the spring-boot-starter-parent project and declares dependencies to one or more "Starters".
  - Spring Boot also provides an optional Maven plugin to create executable jars.

Maven User

- Maven users can inherit from the **spring-boot-starter-parent** project to obtain sensible defaults. The parent project provides the following features:
  - Java 1.8 as the default compiler level.
  - UTF-8 source encoding.
  - A Dependency Management section, inherited from the spring-boot-dependencies POM, that manages the versions of common dependencies.
  - This dependency management lets you omit <version> tags for those dependencies when used in your own POM.

Hello World App

- **Create a maven quickstart project**
- **Inheriting the Starter Parent**
  - <parent>

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.1.2.RELEASE</version>
</parent>
```

- **Add the required starter dependency**

```
<dependencies>
 <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
 </dependency>
</dependencies>
```

Hello World

- **Add the following class**

```
@SpringBootApplication
public class WebApp {
        public static void main(String[] args) {
        SpringApplication.run(WebApp.class, args);
        }
}
```

- **Add a REST Controller class**

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

Running the App

- **Classpath Dependencies**
  - mvn dependency:tree
- **Using Spring Boot Maven Plugin**
  - **Running Application**
    - mvn spring-boot:run
  - **Creating an Executable Jar**
    - mvn package (creates the Fat jar)
    - Uber Jar vs Fat Jar
- **To run the application**
  - java -jar target/myproject-0.0.1-SNAPSHOT.jar

Configuration Classes

- Spring Boot favors Java-based configuration. Although it is possible to use SpringApplication with XML sources, we generally recommend that your primary source be a single @Configuration class.
- Usually the class that defines the main method is a good candidate as the primary @Configuration.

- **Importing Additional Configuration Classes**
  - @Import

- **Importing XML Configuration**
  - @ImportResource

Auto-configuration

- Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added. For example, if HSQLDB is on your classpath, and you have not manually configured any database connection beans, then Spring Boot auto-configures an in-memory database.
- You need to opt-in to auto-configuration by adding the @EnableAutoConfiguration or @SpringBootApplication annotations to one of your @Configuration classes.

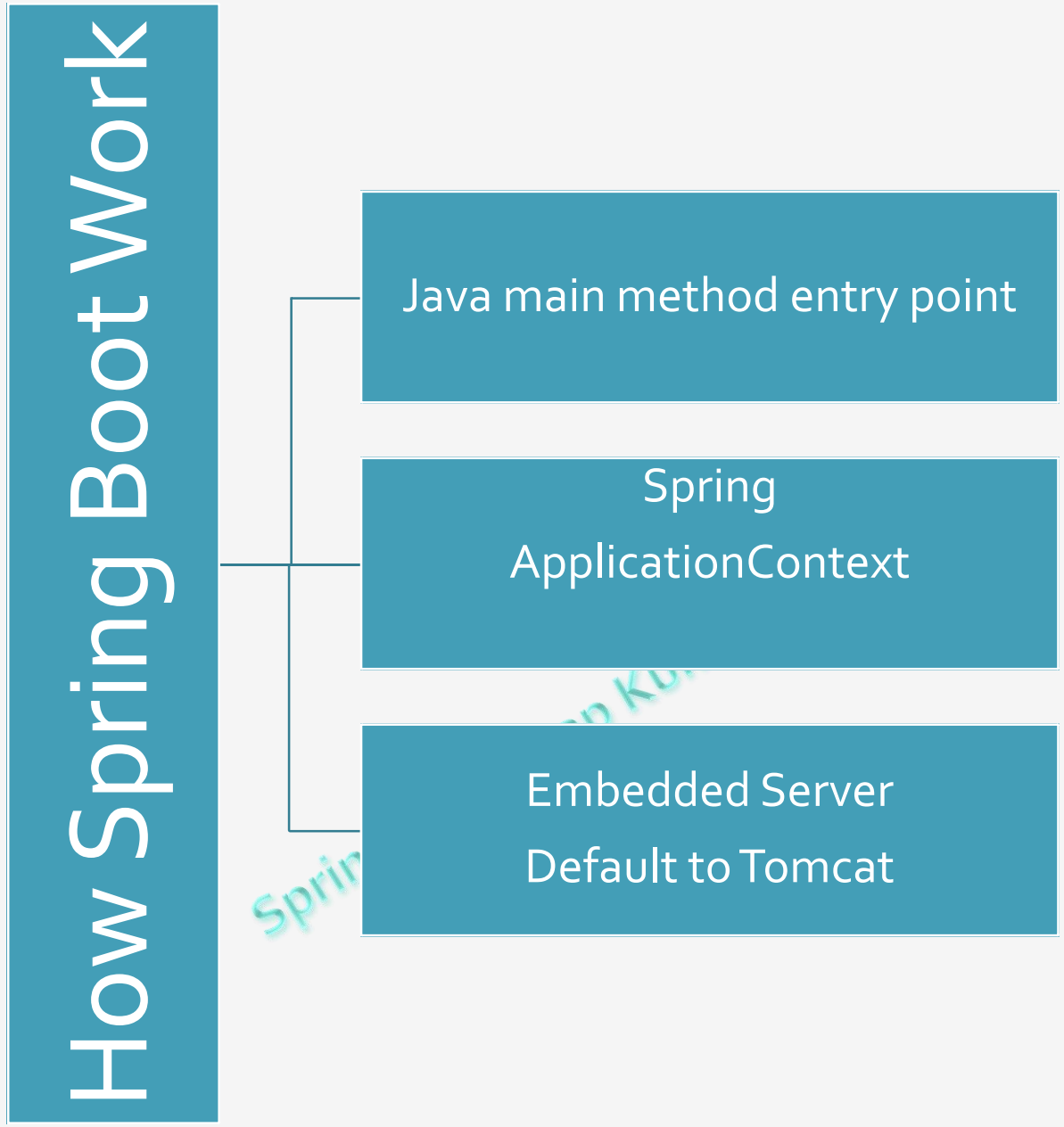
- **Disabling Specific Auto-configuration Classes**

```
@EnableAutoConfiguration(
        exclude={DataSourceAutoConfiguration.class})
```


- **Application.properties**

```
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration
```

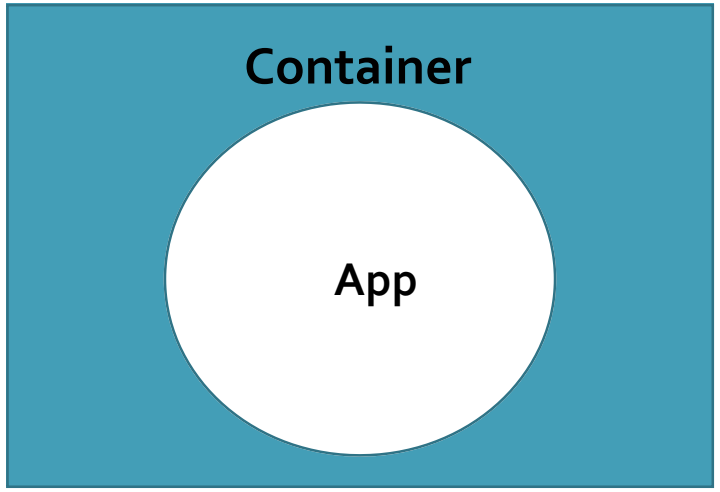
Spring Boot project using Maven



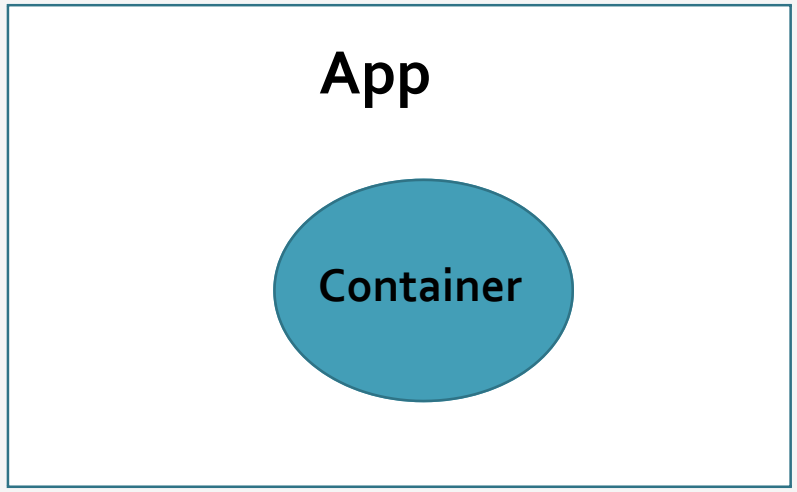
Application.java

- **@SpringBootApplication**
  - A convenience annotation that wraps commonly used annotation with spring Boot
  - **@SringBootConfiguration** Spring configuration on startup
  - **@EnableAutoConfiguration** Auto Configure the framework
  - **@ComponentScan** scans projects for spring components
- **SpringApplication.run(..)** Starts spring , creates spring context, applies annotations and sets up container

Why Move to Container-Less Deployment ?



- Pre-setup and configuration
- Need to use Files like web.xml to tell container how to work
- environment configuration is external to you application



- Runs anywhere java is setup (think cloud deployments)
- container is embedded and the app directs how the container works
- Environment configuration is internal to your application

Tools

- Use The following tool to create spring boot skeleton project .
  - https://start.spring.io/ （***Spring Initializer***）
  - Curl
  - Spring boot CLI
  - IDE support

Spring Initializer

- Visit the Following URL.
  - https://start.spring.io/ (Spring Initializer)

Curl

- https://curl.haxx.se/download.html
- https://skanthak.homepage.t-online.de/curl.html

  - CURL start.spring.io
  - CURL start.spring.io/starter.zip -o demo.zip -d dependencies=web,actuator
  - CURL start.spring.io/starter.zip -o demo.zip --data @options

- options file
  - dependencies=web,actuator&
  - applicationName=MySpringBootApplication&
  - artifactId=boot-one&
  - description=spring boot application&
  - groupid=com.pratap&
  - packageName=com.pratap

Spring Boot CLI

- The Spring Boot CLI is a command line tool that you can use if you want to quickly develop a Spring application (prototype with Spring) .

- Download
  - spring-boot-cli-2.1.2.RELEASE-bin.zip

- You can also bootstrap a new project

- It lets you run Groovy scripts.

Running app with CLI

- You can compile and run Groovy source code by using the run command.
- The Spring Boot CLI is completely self-contained, so you do not need any external Groovy installation.

hello.groovy.

```
@RestController
class WebApplication {
        @RequestMapping("/")
        String home() {    "Hello World!"   }
}
```


- To compile and run the application,
  - > spring run hello.groovy

CLI

- **Packaging Your Application**
  - You can use the jar command to package your application into a self-contained executable jar file.
  - $ spring jar my-app.jar *.groovy
  - java jar my-app.jar
- **Initialize a New Project**
  - The init command lets you create a new project by using start.spring.io without leaving the shell,
  - $ spring init --dependencies=web,data-jpa my-project

Developing Web Applications

- **Create a spring boot web project**
- **Create index.html page under**
  - *src/main/resource/static*
  - Click Here
- **Add one Controller**

```
@RequestMapping("/something")
public String listThings(Model  model){
        SomeBusiness business = new SomeBusiness();
        List<Something> somethings =
                business.getSomethings();
        model.addAttribute("somethings", somethings);
        return "path";
}
```

Web App Using Spring Boot

- **How to use JSP with spring boot App ?**

```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

- By default spring boot look for jsps in src/main/webapp folder

- To protect jsps, Store them in WEB-INF

- The files under WEB-INF are protected by the web server

Tomcat to Jetty

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
<dependency>
   <groupId>org.eclipse.jetty</groupId>
   <artifactId>apache-jsp</artifactId>
</dependency>
```

Spring Boot features

Customizing the Banner

- The banner that is printed on start up can be changed by adding a banner.txt file to your classpath or by setting the spring.banner.location property to the location of such a file.

- spring.main.banner-mode=off

Customizing SpringApplication

- If the SpringApplication defaults are not to your taste, you can instead create a local instance and customize it.


```
SpringApplication app = new
SpringApplication(MySpringConfiguration.class);

app.setBannerMode(Banner.Mode.OFF);

app.run(args);
```

Spring Boot features

Fluent Builder API

- If you need to build an ApplicationContext hierarchy (multiple contexts with a parent/child relationship) or if you prefer using a "fluent" builder API, you can use the SpringApplicationBuilder.

```
new SpringApplicationBuilder()
                .sources(Parent.class)
                .child(Application.class)
                .bannerMode(Banner.Mode.OFF)
                .run(args);
```

Spring Boot features

ApplicationRunner or CommandLineRunner

- If you need to run some specific code once the SpringApplication has started, you can implement the ApplicationRunner or CommandLineRunner interfaces.
- Both interfaces work in the same way and offer a single run method, which is called just before SpringApplication.run(…) completes.

- If several CommandLineRunner or ApplicationRunner beans are defined that must be called in a specific order, you can additionally implement the org.springframework.core.Ordered interface or use the org.springframework.core.annotation.Order annotation.