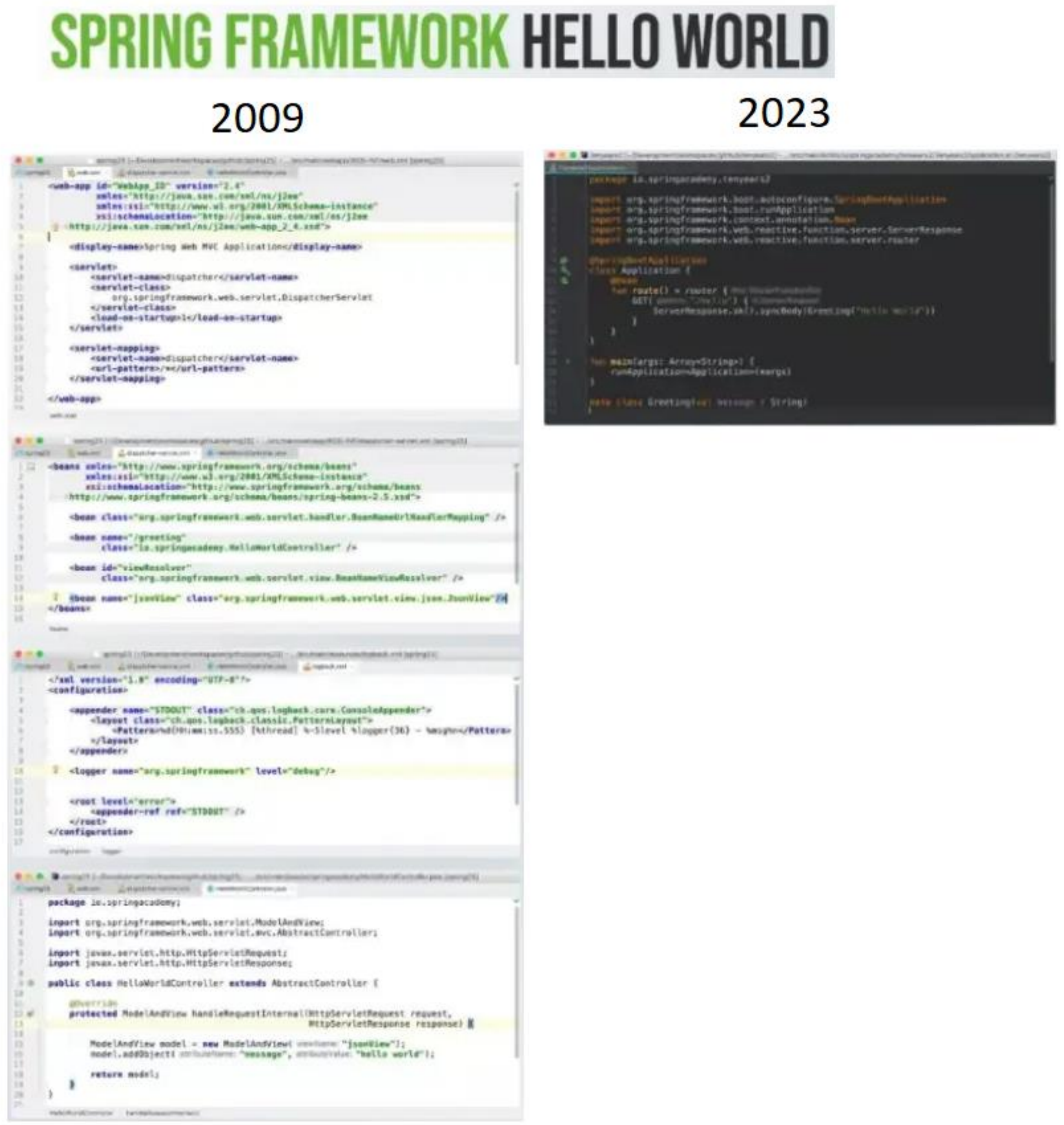
# Spring Boot Intro

RUJUAN XING

## Background

Before spring boot, programmers using the spring framework configured the web application environment, which required a large amount of xml configuration.

The figure shows the configuration difference between `xml` configuration and `SpringBoot`.



```xml
<web-app id="WebApp_ID" version="2.4"
         xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <display-name>Spring Web MVC Application</display-name>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

</web-app>
```

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />

    <bean name="/greeting"
          class="io.springacademy.HelloWorldController" />

    <bean id="viewResolver"
          class="org.springframework.web.servlet.view.BeanNameViewResolver" />

    <bean name="jsonView" class="org.springframework.web.servlet.view.json.JsonView"/>
</beans>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">
            <Pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</Pattern>
        </layout>
    </appender>

    <logger name="org.springframework" level="debug"/>


    <root level="error">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

```java
package io.springacademy;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorldController extends AbstractController {

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest request,
                                                 HttpServletResponse response) {

        ModelAndView model = new ModelAndView( viewName: "jsonView");
        model.addObject( attributeName: "message", attributeValue: "hello world");

        return model;
    }
}
```

```kotlin
package io.springacademy.tenyears2

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.context.annotation.Bean
import org.springframework.web.reactive.function.server.ServerResponse
import org.springframework.web.reactive.function.server.router

@SpringBootApplication
class Application {
    @Bean
    fun route() = router {
        GET( pattern: "/hello") {
            ServerResponse.ok().syncBody(Greeting("Hello World"))
        }
    }
}

fun main(args: Array<String>) {
    runApplication<Application>(*args)
}

data class Greeting(val message : String)
```

## What has spring boot changed?

Spring MVC: the way to configure is like manually connecting wires. Very flexible, but the results of different programmers’ configuration vary widely, and easy to make mistakes.

Spring Boot: the configuration method is like plug and outlet which is a standard. If you want to plug in a third-party open-source library, you can make a starter adaption according to the standard. Although Spring boot is not as flexible as Spring MVC, but it is convenient and easy to use.

The goal of Spring Boot is not to provide new solutions to already solved problem areas, but to bring another new development experience to the platform to simplify the use of these existing technologies.

- Make configuration easy
- Make monitoring simple
- Make deployment easy
- Make development easy

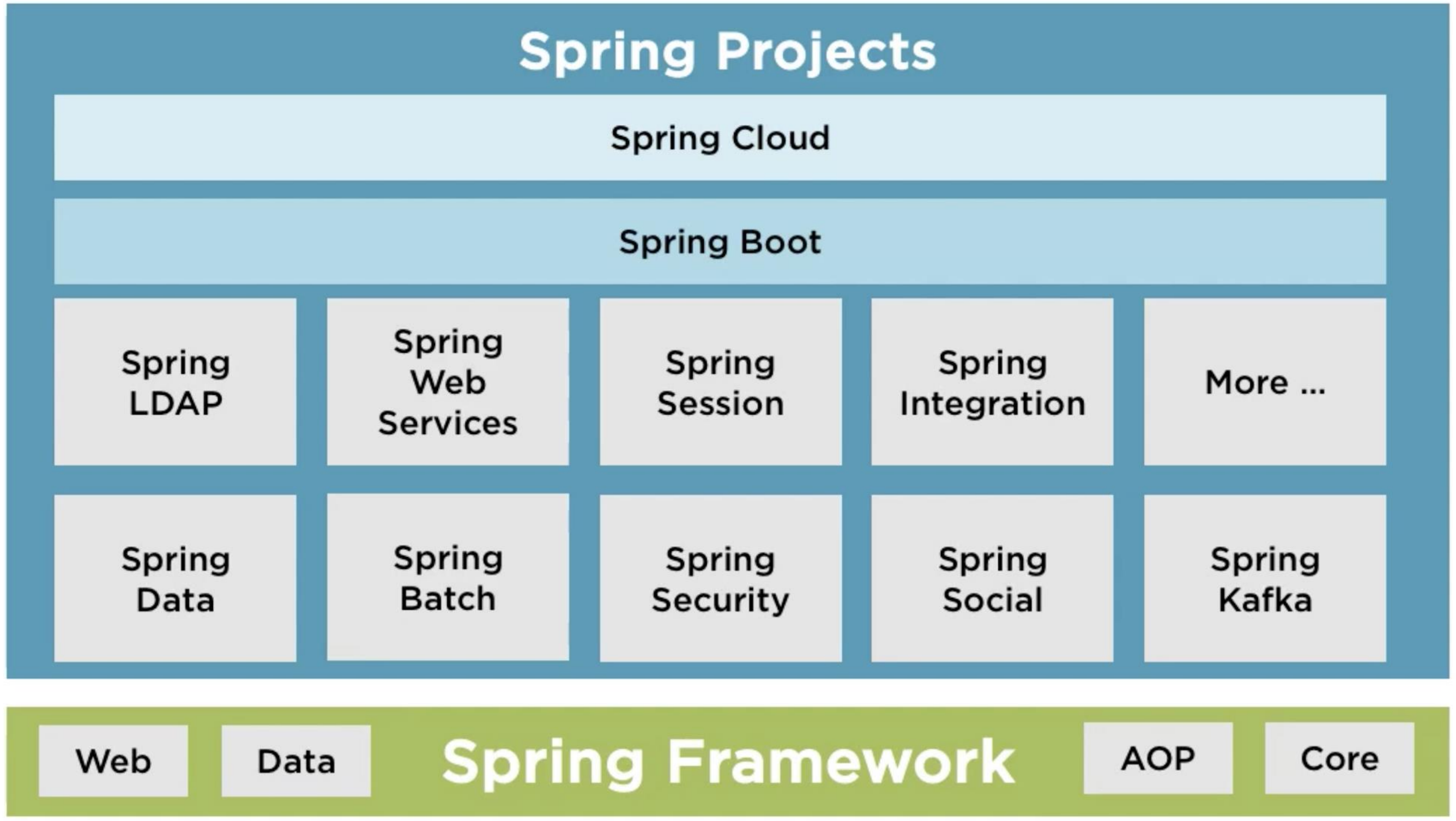
## What is Spring Boot?

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 minimal Spring configuration.

## Spring Boot vs Spring MVC vs Spring

Spring Boot is not to replace Spring and SpringMVC, but to make them easier to use.



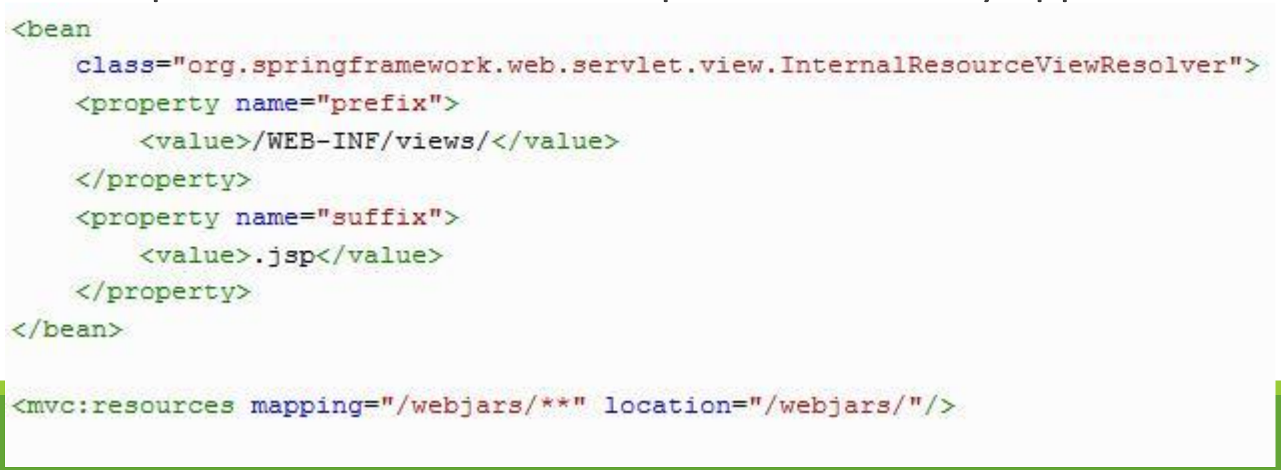
## Spring MVC, Spring Boot

Spring MVC

- provides a friendly way to develop web applications. Web applications can be developed easily by using tools like Dispatcher Servlet, ModelAndView and View Resolver.

Spring Boot

- The biggest drawback of Spring and Spring MVC is that there are a large number of configurations, and these configurations are highly similar in different projects. This leads to repeated configuration, which is cumbersome and messy!
- Spring Boot solves the problem by combining automatic configuration and starters. Additionally, Spring Boot provides features to build production-ready applications faster.



```
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix">
        <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
        <value>.jsp</value>
    </property>
</bean>

<mvc:resources mapping="/webjars/**" location="/webjars/"/>
```

## Spring Framework

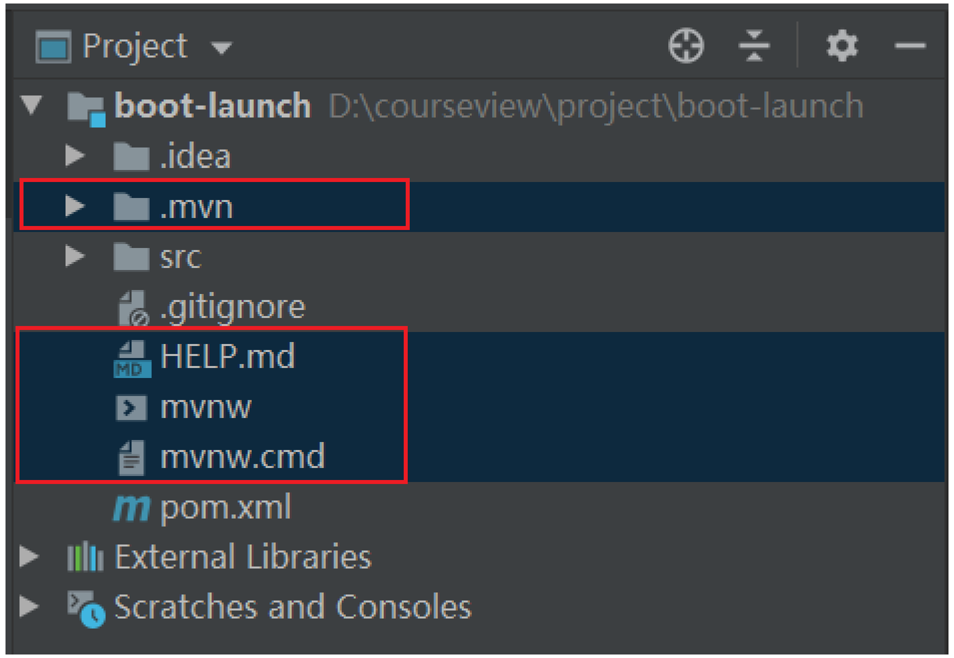
Spring framework is divided into six key areas:

- Spring Core – DI and IOC
- Spring Web – handling web requests, can be done in 2 ways
  - Spring MVC – built on top of Servlet
  - Spring Web flux – react programming, asynchronous
- AOP (Aspect Oriented Programming) - by allowing separation of cross cutting concerns.
- Spring Data Access – reduce a lot of boiler plate code and simpler to write database operations
- Spring Integration – make different systems and applications work together
- Spring Testing

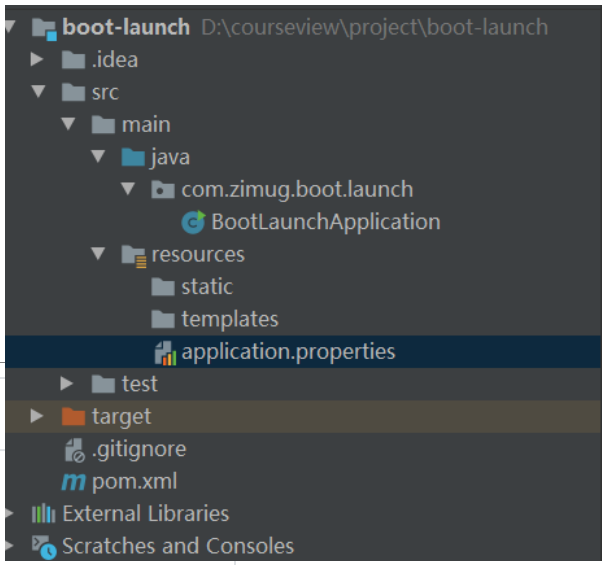
# Hello World

Create a new **Spring Boot project** via the **Spring Initializr** wizard

1. Go to **File | New | Project**.
2. In the left pane of the **New Project** wizard, select **Spring Initializr**.
3. Select **Spring Web,** then click **Create** Button



After the project is built, delete the following files. These files are related to maven version control. We use IDEA to manage maven, these files are not used.



```
boot-launch D:\courseview\project\boot-launch
  .idea
  src
    main
      java
        com.zimug.boot.launch
          BootLaunchApplication
      resources
        static
        templates
        application.properties
    test
  target
  .gitignore
  pom.xml
External Libraries
Scratches and Consoles
```

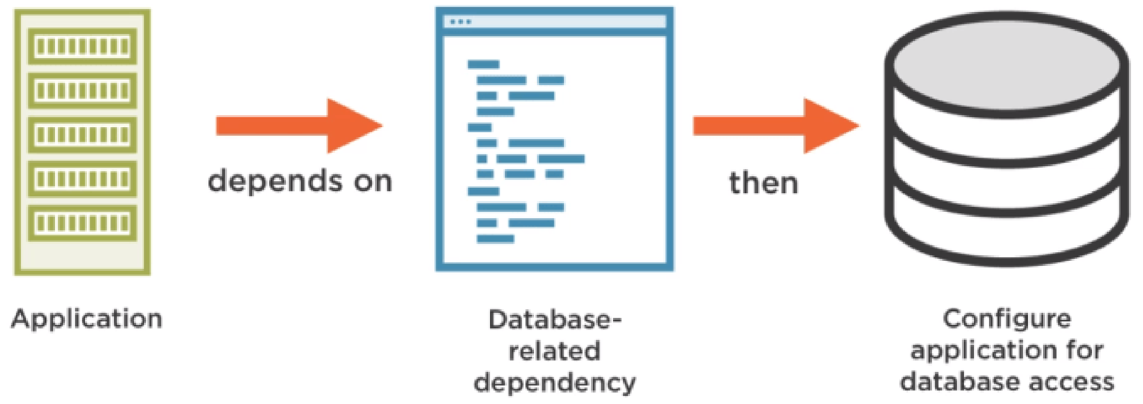
# Project Structure

| Directory | Functionalities |
|---|---|
| src/main/java | The location of all java files.<br>The entry point of the application with file named XxxApplication. |
| src/main/resources | The location of static resources， such as pictures, CSS, JavaScript, Templates |
| src/test | Testing code |
| .gitignore | Telling git which files and directories to ignore when you make a commit. |
| target | IntelliJ will keep the compiled versions of your Java application files. It is automatically managed by IntelliJ and you should not place any of your application files here. |
| pom.xml | Contains information about the project and configuration details used by Maven to build the project |
| application.properties | Used to write the application-related property into that file |

# Spring Boot Features
# - Auto Configuration `@EnableAutoConfiguration`

There is a lot of XML or Java Bean configuration in Spring and Spring MVC applications. Spring Boot provides a new solution to solve this problem.

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. This is one of the most powerful feature of the Spring Boot and most of the work happens silently in the background.

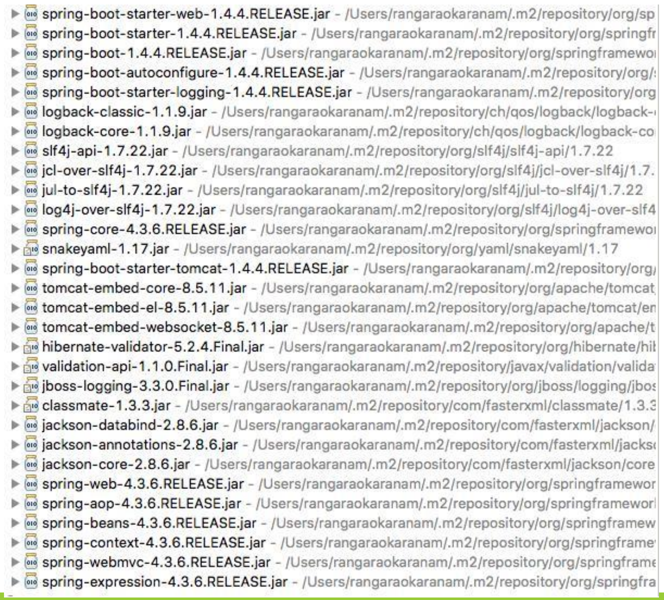

Application
depends on
Database-related dependency
then
Configure application for database access

## Spring Boot Features – Spring Boot Starter

In earlier days, developers used to include all those dependencies. Now Spring Boot Starters provides all those with just a single dependency. The official starters follow a naming convention spring-boot-starter-*, where * denotes application type.

For example, if we want to build web including RESTful applications using Spring MVC we have to use spring-boot-starter-web dependency. It automatically imports dependencies below:

- Spring – Spring Core, beans, context, AOP
- Web MVC – Spring MVC
- Jackson - Translates JSON data to a Java POJO by itself and vice-versa
- Validation – Hibernate Validation
- Embedded Servlet Container – Tomcat
- Logging – logback, slf4j



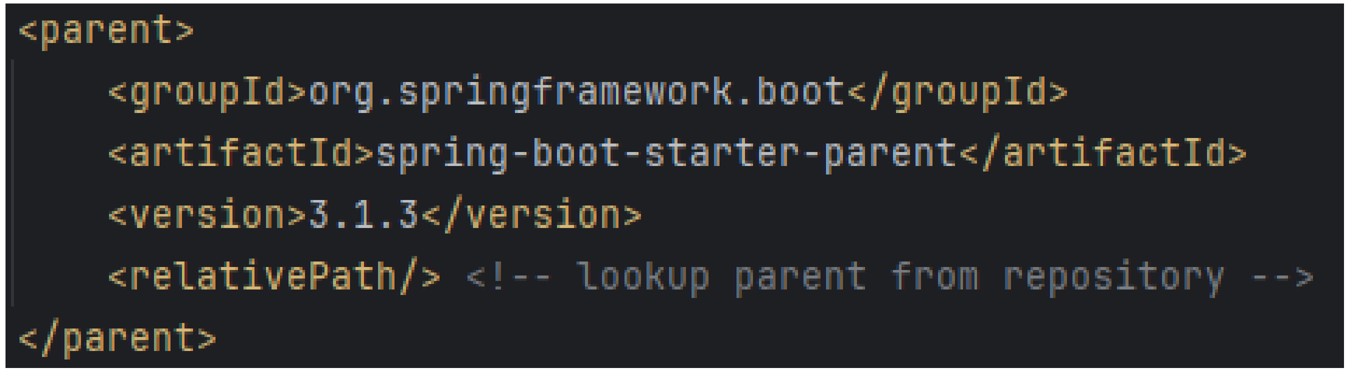
```
spring-boot-starter-web-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/sp
spring-boot-starter-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springfr
spring-boot-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframewo
spring-boot-autoconfigure-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/
spring-boot-starter-logging-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org
logback-classic-1.1.9.jar - /Users/rangaraokaranam/.m2/repository/ch/qos/logback/logback-
logback-core-1.1.9.jar - /Users/rangaraokaranam/.m2/repository/ch/qos/logback/logback-co
slf4j-api-1.7.22.jar - /Users/rangaraokaranam/.m2/repository/org/slf4j/slf4j-api/1.7.22
jcl-over-slf4j-1.7.22.jar - /Users/rangaraokaranam/.m2/repository/org/slf4j/jcl-over-slf4j/1.7.
jul-to-slf4j-1.7.22.jar - /Users/rangaraokaranam/.m2/repository/org/slf4j/jul-to-slf4j/1.7.22
log4j-over-slf4j-1.7.22.jar - /Users/rangaraokaranam/.m2/repository/org/slf4j/log4j-over-slf4
spring-core-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframewo
snakeyaml-1.17.jar - /Users/rangaraokaranam/.m2/repository/org/yaml/snakeyaml/1.17
spring-boot-starter-tomcat-1.4.4.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/
tomcat-embed-core-8.5.11.jar - /Users/rangaraokaranam/.m2/repository/org/apache/tomcat
tomcat-embed-el-8.5.11.jar - /Users/rangaraokaranam/.m2/repository/org/apache/tomcat/en
tomcat-embed-websocket-8.5.11.jar - /Users/rangaraokaranam/.m2/repository/org/apache/t
hibernate-validator-5.2.4.Final.jar - /Users/rangaraokaranam/.m2/repository/org/hibernate/hi
validation-api-1.1.0.Final.jar - /Users/rangaraokaranam/.m2/repository/javax/validation/valida
jboss-logging-3.3.0.Final.jar - /Users/rangaraokaranam/.m2/repository/org/jboss/logging/jbo
classmate-1.3.3.jar - /Users/rangaraokaranam/.m2/repository/com/fasterxml/classmate/1.3.3
jackson-databind-2.8.6.jar - /Users/rangaraokaranam/.m2/repository/com/fasterxml/jackson/
jackson-annotations-2.8.6.jar - /Users/rangaraokaranam/.m2/repository/com/fasterxml/jacks
jackson-core-2.8.6.jar - /Users/rangaraokaranam/.m2/repository/com/fasterxml/jackson/core
spring-web-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframewor
spring-aop-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframewor
spring-beans-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframew
spring-context-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframe
spring-webmvc-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springframe
spring-expression-4.3.6.RELEASE.jar - /Users/rangaraokaranam/.m2/repository/org/springfra
```

## Spring Boot Features
## - Spring Boot Starter Parent

All Spring Boot projects use spring-boot-starter-parent as the parent project of the application by default.

Parent poms let you handle the following for several child projects and modules:

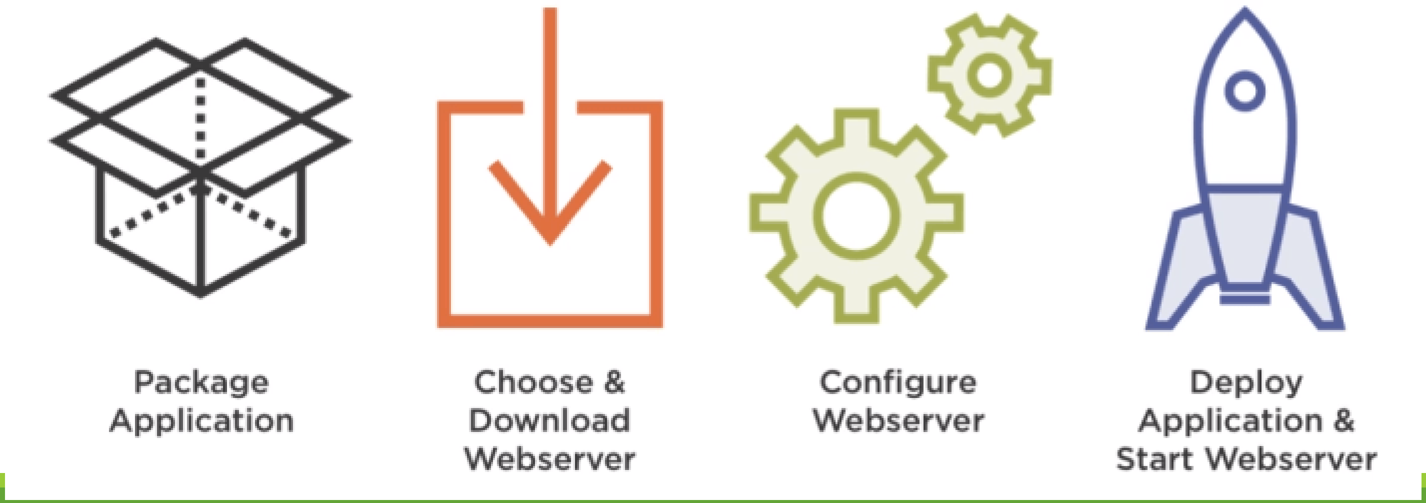
- Configuration - Java Version and Other Properties
- Depedency Management - Version of dependencies, you don't need to specify the version of the dependencies you added, the parent will manage them and it's been tested working.
- Default Plugin Configuration



```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.3</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

# The Process of Starting a Java-Based Web Application

1. **Package** your application.
2. Choose **which type of web servers** you want to use and download it.
3. **Configure** that specific web server.
4. **Oganize the deployment process** and start your web server.


Package Application
Choose & Download Webserver
Configure Webserver
Deploy Application & Start Webserver

## Spring Boot Features
## - Embed Server

Spring boot create stand-alone Spring applications which run web application as standalone application, there is no need to deploy WAR files.

Because Spring boot packages web application as jar file which has embeded Server: Tomcat is default one. It also supports Jetty or Undertow.

`spring-boot-starter-web`: includes a dependency on starter tomcat.

We can start web application using command:

- `java -jar springboot-demo.jar`

# Main Application Class

```
@SpringBootApplication
public class SpringBootWebApplication {

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

- `@SpringBootApplication` annotation is equivalent to using
  - `@Configuration`: is used to indicate that a class declares one or more @Bean methods.
  - `@EnableAutoConfiguration`: is a key annotation in Spring Boot that enables automatic configuration of the Spring application context based on the project's dependencies and the configuration provided by Spring Boot's auto-configuration modules.
  - `@ComponentScan`: used to specify the base package(s) that Spring should scan to discover Spring-managed components such as @Component, @Service, @Repository, and @Controller. This annotation is often used in conjunction with other annotations like @Configuration and @EnableAutoConfiguration to configure and bootstrap a Spring application context.

## Controller

```java
@RestController
public class WelcomeController {

    @GetMapping("/hello")
    public String greeting(String name){
        return "hello, " + name;
    }

    @GetMapping("/")
    public String greeting2(){
        return "Hi from Tina";
    }

}
```

## LiveReload

1. Add dependency: spring-boot-devtools
2. Install browser extension: LiveReload
3. Configure in Intellij, go to: file->settings->build,execution,deployment. Go to ->compiler->build project automatically.

If you don’t want to start the LiveReload server when your application runs you can set the spring.devtools.livereload.enabled property to false.

Two different class loaders:

- One class loader contains all the dependencies that applications needs. These aren’t likely to change often.
- The second class loader contains your actual Java code, property files, etc, which are likely to change frequently. On detecting a change, DevTools reloads only above the class loader with restarting the Spring application context and leaves another class loader, JVM intact.

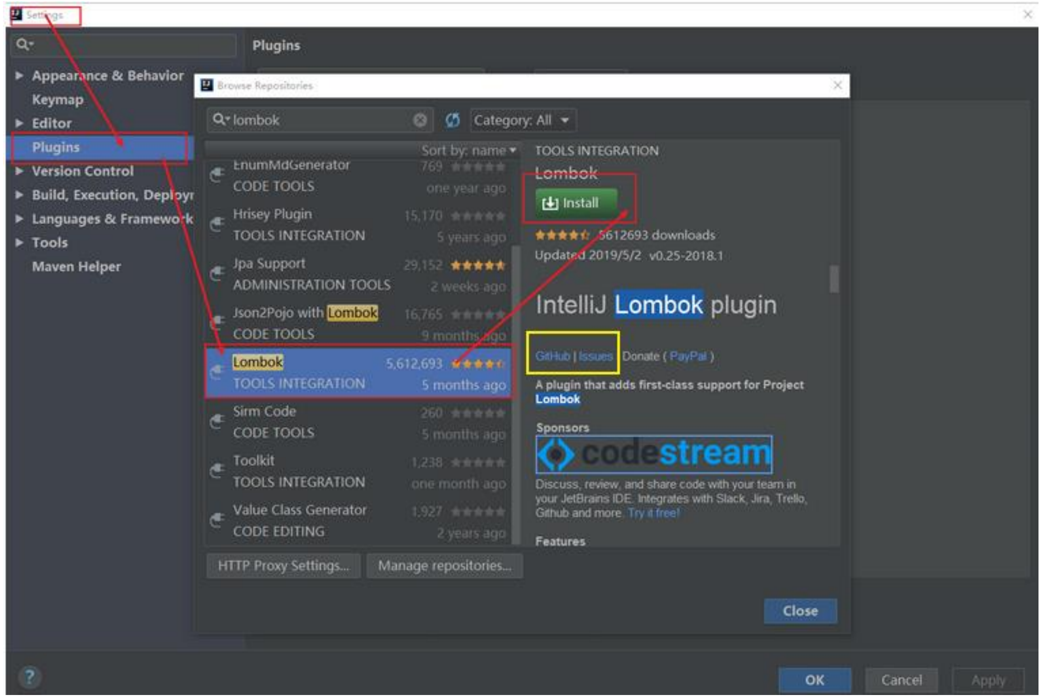
❑ Or run app **debug** mode and press `Ctrl + f9` (short-cut for build)

# Lombok

Project Lombok is a Java library tool that generates code for minimizing boilerplate code. The library replaces boilerplate code with easy-to-use annotations.

For example, by adding a couple of annotations, you can get rid of code clutters, such as getters and setters methods, constructors, hashcode, equals, and toString methods, and so on.

```xml
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```



```
EnumMdGenerator        769
CODE TOOLS             one year ago
Hrisey Plugin          15,170
TOOLS INTEGRATION      5 years ago
Jpa Support            29,152
ADMINISTRATION TOOLS   2 weeks ago
Json2Pojo with Lombok  16,765
CODE TOOLS             9 months ago
Lombok                 5,612,693
TOOLS INTEGRATION      5 months ago
Sirm Code              260
CODE TOOLS             5 months ago
Toolkit                1,238
TOOLS INTEGRATION      one month ago
Value Class Generator  1,927
CODE EDITING           2 years ago
```

TOOLS INTEGRATION
Lombok
5612693 downloads
Updated 2019/5/2 v0.25-2018.1

IntelliJ Lombok plugin

GitHub | Issues Donate ( PayPal )

A plugin that adds first-class support for Project Lombok

Sponsors

codestream

Discuss, review, and share code with your team in your JetBrains IDE. Integrates with Slack, Jira, Trello, Github and more. Try it free!

Features

## Lombok - @Getter and @Setter

You can use the @Getter and @Setter annotations at both the field or class level to generate getters and setters for private fields.

When you use them at the field level, Lombok generates getters and setters only for the decorated fields.

```
@Getter @Setter
public class Product {
    private int id;

    @Getter @Setter
    private String firstname;

    @Getter
    private String lastname;

    private int age;
}
```

# Lombok – @NoArgsConstructor, @AllArgsConstructor

`@NoArgsConstructor` annotation: generate the default constructor that takes no arguments.

`@AllArgsConstructor` annotation: generate a constructor with arguments for all the field

```
@NoArgsConstructor
@AllArgsConstructor
public class Product {

    private int id;

    @Getter @Setter
    private String firstname;

    @Getter
    private String lastname;

    private int age;
}
```

## Lombok - `@Data`

`@Data` is a convenient annotation that combines the features of the following annotations:

- `@ToString`
- `@EqualsAndHashCode`
- `@Getter`
- `@Setter`
- `@RequiredArgsConstructor`

```
@Data
public class Product {
    private int id;
    private String firstname;
    private String lastname;
    private int age;
}
```

## Java record

Java's record keyword is a new semantic feature introduced in Java 14. Records are very useful for creating small immutable objects.

To accomplish the immutability, we create classes with the following:

- `private`, `final` field for each piece of data
- `getter` for each field
- `public constructor` with a corresponding argument for each field
- `equals` method that returns `true` for objects of the same class when all fields match
- `hashCode` method that returns the same value when all fields match
- `toString` method that includes the name of the class and the name of each field and its corresponding value

```
public final class Rectangle {
    private final double length;
    private final double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double length() { return this.length; }
    double width()  { return this.width; }

    // Implementation of equals() and hashCode(), which specify
    // that two record objects are equal if they
    // are of the same type and contain equal field values.
    public boolean equals...
    public int hashCode...

    // An implementation of toString() that returns a string
    // representation of all the record class's fields,
    // including their names.
    public String toString() {...}
}
```

```
public record Rectangle(double length, double width) { }
```

# Lombok vs Java record

1. Transparent Data Carriers
   - Records are classes that act as transparent carriers for immutable data. As a result, we cannot stop a record from exposing its member fields.
   - However, if we want to hide the member fields and only expose some operations performed using them, Lombok will be better suited. - @Value + @Getter
2. Classes With Many Fields
   - records are better for smaller objects. Though, for objects with many fields, the lack of creational patterns will make Lombok's @Builder a better option

```
public record StudentRecord(
        String firstName,
        String lastName,
        Long studentId,
        String email,
        String phoneNumber,
        String address,
        String country,
        int age) {
}
```

```
StudentRecord john = new StudentRecord(
        "John", "Doe", null, "john@doe.com", null, null,
"England", 20);
```

```
@Getter
@Builder
public class Student {
    private String firstName;
    private String lastName;
    private Long studentId;
    private String email;
    private String phoneNumber;
    private String address;
    private String country;
    private int age;
}
```

```
Student john =
Student.builder()
        .firstName("John")
        .lastName("Doe")
        .email("john@doe.com")
        .country("England")
        .age(20)
        .build();
```

## Lombok vs Java record (cont.)

3. Mutable Data
   - We can use java records exclusively for immutable data. If the context requires a mutable java object, we can use Lombok's `@Data` object instead.
   - Some frameworks may require objects with setters or a default constructor. For instance, Hibernate falls into this category. When creating an `@Entity`, we'll have to use Lombok's annotations or plain Java.
4. Inheritance
   - Java records do not support inheritance. Therefore, they cannot be extended or inherit other classes. On the other hand, Lombok's `@Value` objects can extend other classes, but they are final:
   - Besides, `@Data` objects can both extend other classes and be extended. In conclusion, if we need inheritance, we should stick to Lombok's solutions.

## Http Client, Rest Client

- Http Client
  - Intellij Plugin – bundled 232.9921.47
  - Provides the ability to compose and execute HTTP requests from the code editor.
  - HTTP requests are stored in **.http** and **.rest** files.
  - Create HTTP Request
    - Press **Ctrl+Alt+Shift+Insert** and select **HTTP Request**.
    - Or In the **File** menu, point to **New**, and then click **HTTP Request.**

- Rest Client
  - VS Code plugin – install by yourself, if you don’t have Ultimate Version of Intellij.
  - Allows you to send HTTP request and view the response in Visual Studio Code directly.

## Main Points

Understanding the fundamental of Spring Boot features allows us to apply best practices W/R to Enterprise Web Application in Spring Framework.

*Understanding more fundamental aspects of "**any thing"** makes us able to put those principles to proper use. Transcendental Consciousness is the ultimate fundamental aspect of Nature.*