Hi folks,  
[#Spring](https://www.facebook.com/hashtag/spring?source=feed_text&epa=HASHTAG) [#boot](https://www.facebook.com/hashtag/boot?source=feed_text&epa=HASHTAG) [#architecture](https://www.facebook.com/hashtag/architecture?source=feed_text&epa=HASHTAG) diagram [#real\_time](https://www.facebook.com/hashtag/real_time?source=feed_text&epa=HASHTAG)  
Spring boot is a module of spring framework which is used to create stand-alone, production-grade Spring based Applications with minimum programmer’s efforts. It is developed on top of core spring framework. The main concept behind spring boot is to avoid lot of boilerplate code and configuration to improve development, unit test etc. As in case of creating a new spring application, we have write many XML configurations, server setting, adding dependencies etc. These configuration files are one of the example of boilerplate code. Spring boot avoids all these boilerplate codes.

[#Understanding](https://www.facebook.com/hashtag/understanding?source=feed_text&epa=HASHTAG) Spring Boot's Architecture  
The building blocks of Spring Boot are:  
1.Spring Boot Starter Projects  
2.Spring Boot Starter Parent  
3.Auto Configuration

#.Spring Boot Starter Projects

--Spring Boot provides a wide range of starter projects. Spring Initializr suppports all of them and more. Among the varied range of starter projects and options supported are:

spring-boot-starter-web-services: Build applications exposing SOAP web services  
spring-boot-starter-web: Build Web applications and RESTful applications  
spring-boot-starter-test: Write great unit and integration tests  
spring-boot-starter-jdbc: Traditional JDBC applications  
spring-boot-starter-hateoas: Make your services more RESTful by adding HATEOAS features  
spring-boot-starter-security: Authentication and authorization using Spring Security  
spring-boot-starter-data-jpa: Spring Data JPA with Hibernate  
spring-boot-starter-cache: Enabling Spring Framework’s caching support  
spring-boot-starter-data-rest: Expose simple REST services using Spring Data REST

2.Spring Boot Starter Parent  
All Spring Boot projects typically use spring-boot-starter-parent as the parent in the pom.xml.

<parent>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-parent</artifactId>  
<version>2.0.0.RELEASE</version>  
</parent>

Parent POMs allow you to manage the following things for multiple child projects and modules:

Configuration: Java version and other properties  
Depedency Management: Version of dependencies  
Default plugin configuration

3. Auto Configuration

Spring based applications have a lot of configuration.

When we use Spring MVC, we need to configure a component scan, the dispatcher servlet, a view resolver, web JARs (for delivering static content), among other things.

<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/"/>

The following code snippet shows a typical configuration of a dispatcher servlet in a web application.

<servlet>  
<servlet-name>dispatcher</servlet-name>  
<servlet-class>  
org.springframework.web.servlet.DispatcherServlet  
</servlet-class>  
<init-param>  
<param-name>contextConfigLocation</param-name>  
<param-value>/WEB-INF/todo-servlet.xml</param-value>  
</init-param>  
<load-on-startup>1</load-on-startup>  
</servlet>  
<servlet-mapping>  
<servlet-name>dispatcher</servlet-name>  
<url-pattern>/</url-pattern>  
</servlet-mapping>

When we use Hibernate/JPA, we would need to configure a datasource, an entity manager factory, a transaction manager, among a host of other things.

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"  
destroy-method="close">  
<property name="driverClass" value="${db.driver}" />  
<property name="jdbcUrl" value="${db.url}" />  
<property name="user" value="${db.username}" />  
<property name="password" value="${db.password}" />  
</bean>  
<jdbc:initialize-database data-source="dataSource">  
<jdbc:script location="classpath:config/schema.sql" />  
<jdbc:script location="classpath:config/data.sql" />  
</jdbc:initialize-database>  
<bean  
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"  
id="entityManagerFactory">  
<property name="persistenceUnitName" value="hsql\_pu" />  
<property name="dataSource" ref="dataSource" />  
</bean>  
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">  
<property name="entityManagerFactory" ref="entityManagerFactory" />  
<property name="dataSource" ref="dataSource" />  
</bean>  
<tx:annotation-driven transaction-manager="transactionManager"/>