Spring Boot

Introduction

- Spring Boot is a Framework from "The Spring Team" to ease the bootstrapping and development of new Spring Applications.
- It provides defaults for code and annotation configuration to quick start new Spring projects within no time. Uses opinionated approach, simplifying the bootstrapping and development of a new **Spring** application. Developers don't need to define boilerplate configuration and hence improve –
  - ✓ Development,
  - ✓ Unit Testing and
  - ✓ Integration Test Process.
- Spring Boot Framework is not implemented from the scratch but on top of existing Spring Framework (Spring IO Platform). It is not used for solving any new problems. It is used to solve same problems like Spring Framework.

contd..

- Goals of Spring Boot –

- ✓ Spring Boot makes it easy to create stand-alone, production – grade Spring – based Applications. It takes an opinionated view of the Spring platform and third – party libraries,
- ✓ 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.

contd..

- With Spring Boot –

- ✓ its very easy to develop Spring Based applications with Java or Groovy as it avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- ✓ It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
- ✓ it provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- ✓ It provides CLI (Command Line Interface) tool to develop and test Spring Boot(Java or Groovy) applications from command prompt very easily and quickly.
- ✓ It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle

Opinionated Software

- A framework is opinionated, if it locks or guides the developer into their way of doing things.
- Opinionated software means that there is basically one way (the **right way**™) to do things and trying to do it differently will be difficult and frustrating. Doing things the **right way**™ can make it very easy to develop with the software as the number of decisions user has to make is reduced and the ability of the software designers to concentrate on making the software work is increased.
- Non-opinionated software, on the other hand, leaves lots of flexibility to the user (developer). It doesn't proscribe one method of solving a problem, but provides flexible tools that can be used to solve the problem in many ways.

contd..

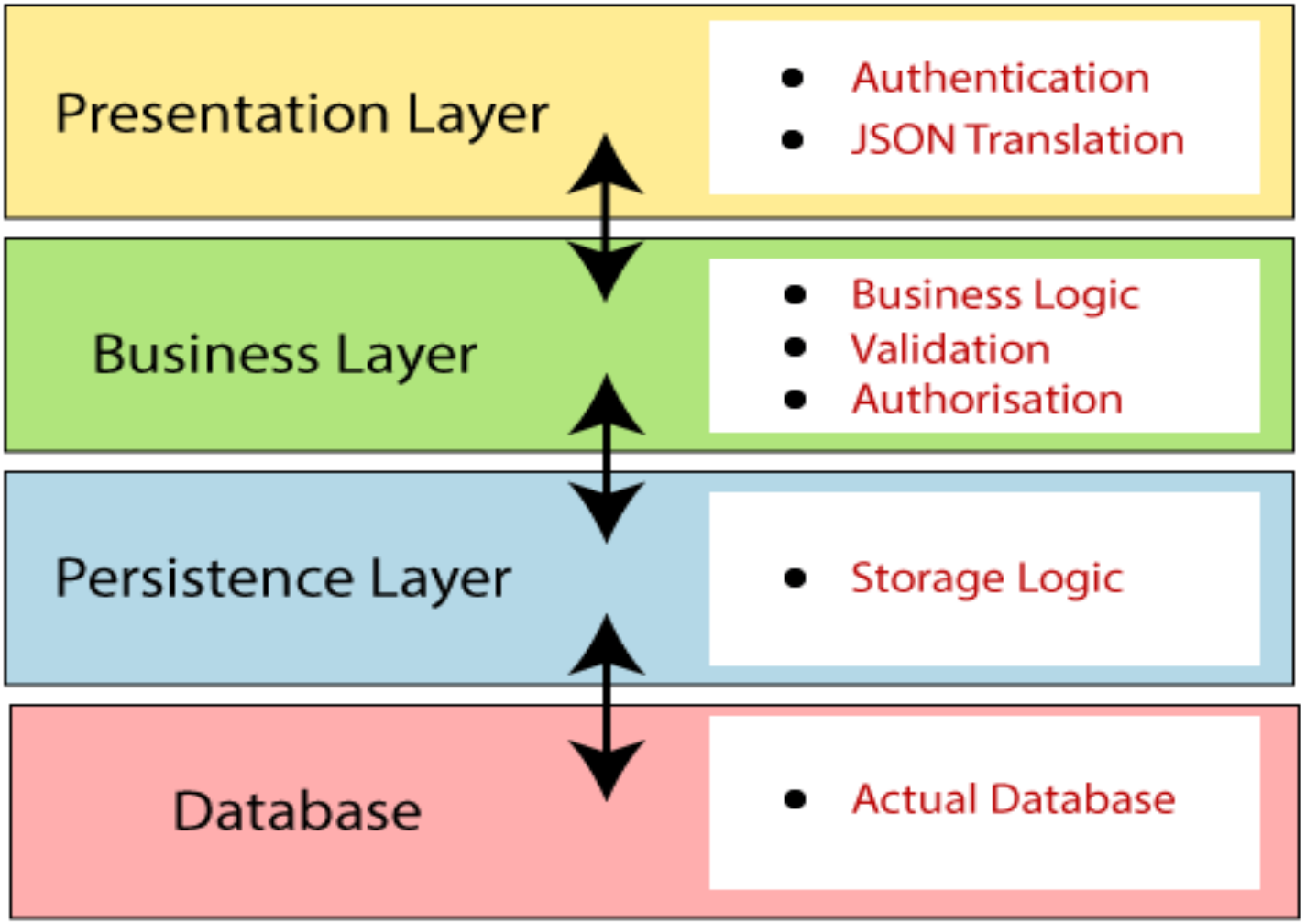
- To Start Opinionated Approach to create Spring Boot Applications, The Spring Team (The Pivotal Team) has provided the following three approaches.
  - ✓ Using Spring Boot CLI Tool
  - ✓ Using Spring STS IDE
  - ✓ Using Spring Initializr Website
- All three are used to develop Spring Boot Groovy Applications.
- Spring Initializr Website is at: http://start.spring.io/
  Two flavors of Spring-Based Applications can be developed using Spring Boot
  - ✓ Java-Based Applications with last two.
  - ✓ Groovy Applications with all three approaches

Components of Spring Boot

- Key Components of Spring Boot Framework
  - ✓ Spring Boot Starters
  - ✓ Spring Boot AutoConfigurator
  - ✓ Spring Boot CLI
  - ✓ Spring Boot Actuator
  - ✓ Spring Initilizr
  - ✓ Spring Boot IDEs
- To quick start new Spring Boot projects, use "Spring Initializr" web interface, URL: http://start.spring.io.
- Spring Boot IDEs are Eclipse IDE, IntelliJ IDEA, Spring STS Suite etc.

Spring Boot Layers

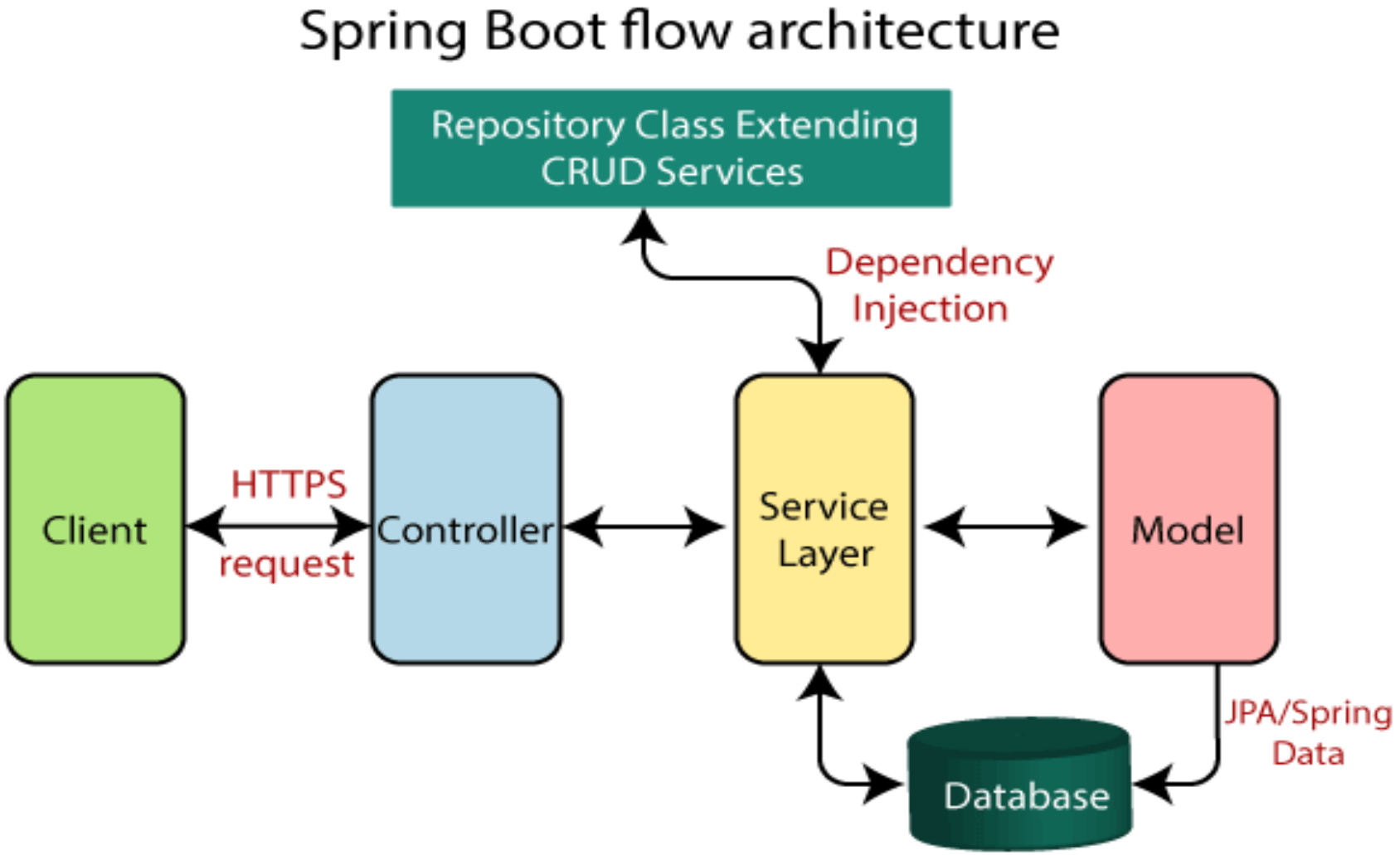
- There are **four** layers in Spring Boot are as follows:



contd..

- **Presentation Layer:** The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfer it to the business layer. In short, it consists of **views** i.e., frontend part.
- **Business Layer:** The business layer handles all the **business logic**. It consists of service classes and uses services provided by data access layers. It also performs **authorization** and **validation**.
- **Persistence Layer:** The persistence layer contains all the **storage logic** and translates business objects from and to database rows.
- **Database Layer:** In the database layer, **CRUD** (create, retrieve, update, delete) operations are performed.

Spring Boot Flow Architecture



contd..

- The components are validator classes, view classes, and utility classes.
- Spring Boot uses all the modules of Spring-like Spring MVC, Spring Data, etc.
- The architecture of Spring Boot is the same as the architecture of Spring MVC, except one thing: there is no need for **DAO** and **DAOImpl** classes in Spring boot.
- Creates a data access layer and performs CRUD operation.
- The client makes the HTTP requests (PUT or GET).
- The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.
- In the service layer, all the business logic performs logic on the data that is mapped to JPA with model classes.
- A JSP page is returned to the user if no error occurred.

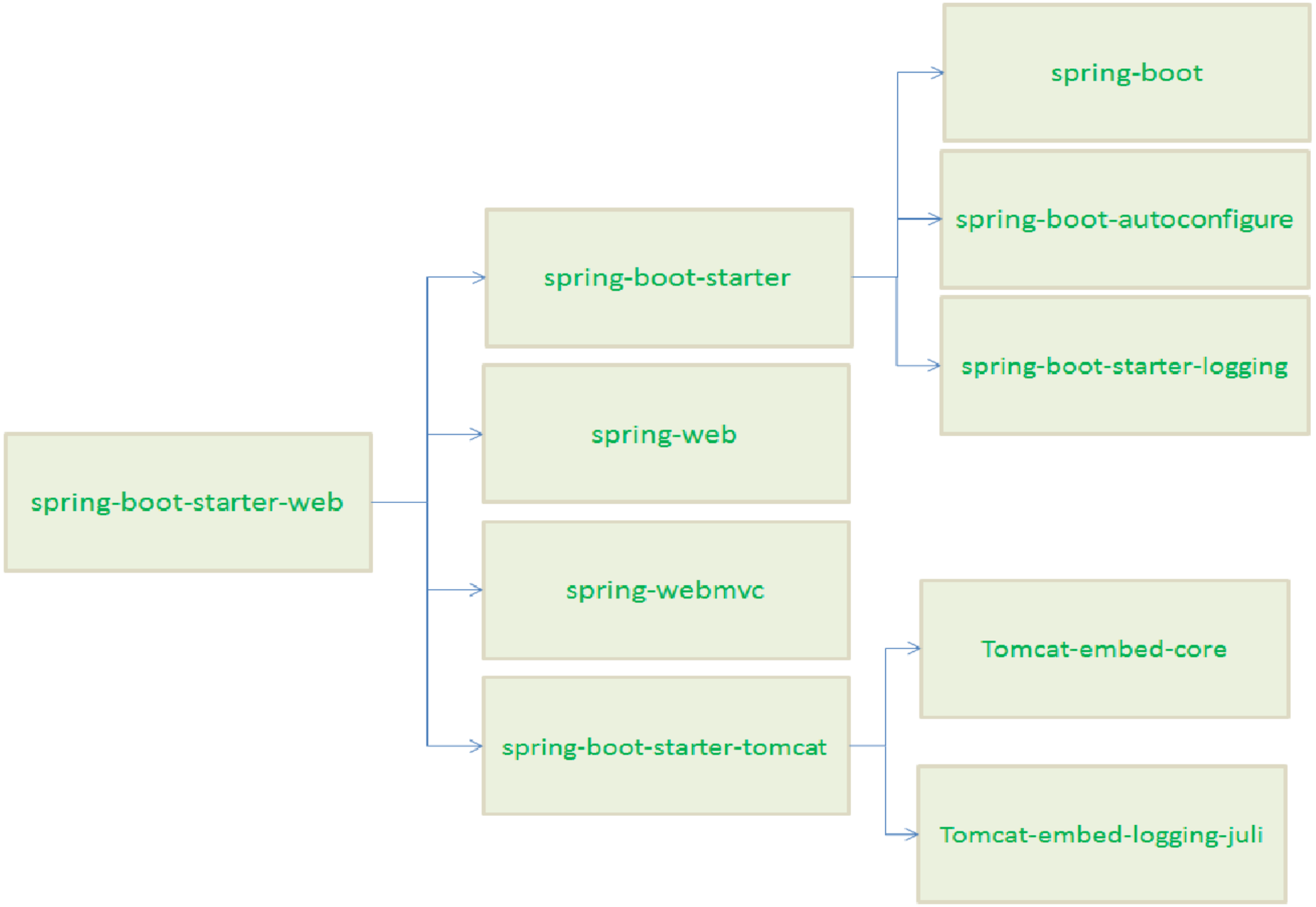
Spring Boot Starter

- Spring Boot Starter –
  - ✓ is one of the major key features or components of Spring Boot Framework.
  - ✓ the main responsibility of Spring Boot Starter is to combine a group of common or related dependencies into single dependencies.

contd..

- On adding “spring-boot-starter-web” jar file dependency to the build file, the Spring Boot Framework will automatically download all required jars and add to our project classpath.

contd..


spring-boot-starter-web
spring-boot-starter
spring-web
spring-webmvc
spring-boot-starter-tomcat
spring-boot
spring-boot-autoconfigure
spring-boot-starter-logging
Tomcat-embed-core
Tomcat-embed-logging-juli

contd..

- In the same way, “spring-boot-starter-logging” jar file loads all it’s dependency jars like “jcl-over-slf4j, jul-to-slf4j,log4j-over-slf4j, logback-classic” to the project classpath.
- Advantages of Spring Boot Starter
  - ✓Spring Boot Starter reduces defining many dependencies
  - ✓Spring Boot Starter simplifies project build dependencies.

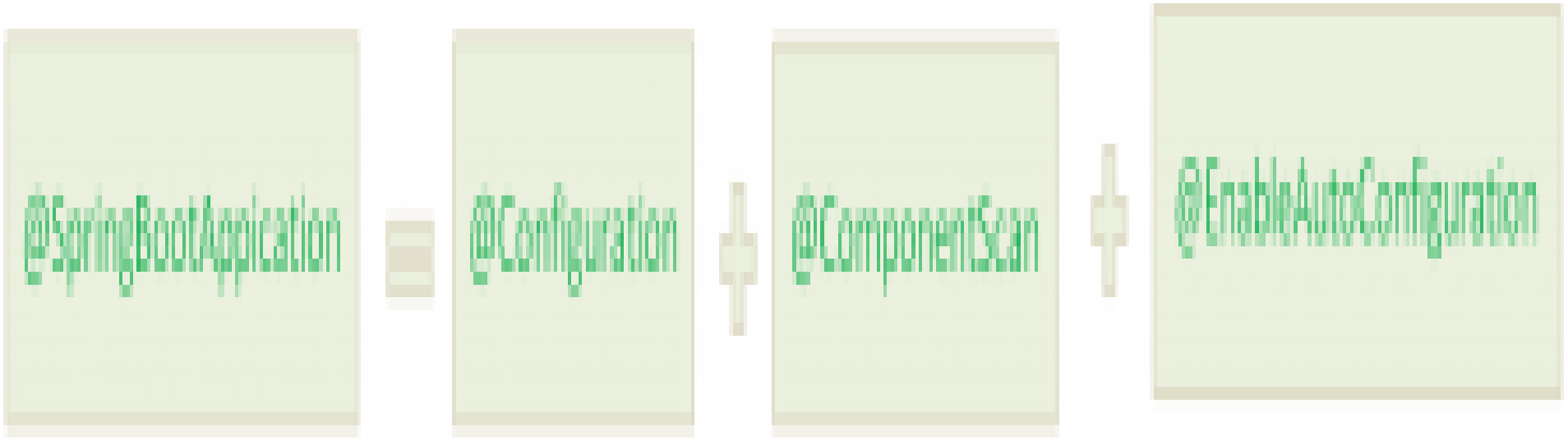
Spring Boot AutoConfigurator

- Another important key component of Spring Boot Framework is Spring Boot AutoConfigurator.
- The solution to the problem of writing lot of configuration is Spring Boot AutoConfigurator.
- The main responsibility of Spring Boot AutoConfigurator is to reduce the Spring Configuration.
- With Spring Boot development environment, then there is no need to define single XML configuration and almost no or minimal Annotation configuration. Spring Boot AutoConfigurator component takes care of providing those information.

contd..

- The “spring-boot-starter-web” jar file in the project build file, will resolve views, view resolvers etc. automatically.
- Spring Boot also reduces defining of Annotation configuration. On using @SpringBootApplication annotation at class level, the Spring Boot AutoConfigurator will automatically add all required annotations to Java Class ByteCode.

contd..



@SpringBootApplication = @Configuration + @ComponentScan + @EnableAutoConfiguration.

contd..

- Annotations –

- ✓ @RestController also known as a *stereotype* annotation. It provides hints for people reading the code and for Spring that the class plays a specific role. @RestController annotation tells Spring to render the resulting string directly back to the caller.
- ✓ @Controller – Spring considers it when handling incoming web requests.
- ✓ @RequestMapping annotation provides "routing" information. It tells Spring that any HTTP request with the / path should be mapped to the home method.

contd..

- ✓ @EnableAutoConfiguration – This annotation tells Spring Boot to "guess" i.e. how user wants to configure Spring, based on the jar dependencies that have been added. E.g. if spring-boot-starter-web added Tomcat and Spring MVC, the auto-configuration assumes that user is developing a web application and sets up Spring accordingly.
- The "main" Method – the final part of application is the main method, a standard method that follows the Java convention for an application entry point.

1. The main method delegates to Spring Boot's SpringApplication class by calling run.
2. SpringApplication bootstraps our application, starting Spring, which, in turn, starts the auto-configured Tomcat web server.

contd..

Note : The class is passed (E.g. TestMain.class) as an argument to the run method to tell SpringApplication which is the primary Spring component.

- ✓ The args array is also passed through to expose any command-line arguments.

Dependency Management

- **Dependency Management**

Each release of Spring Boot provides a curated list of dependencies that it supports. In practice, user does not need to provide a version for any of these dependencies in the build configuration, as Spring Boot manages that. When there is of Spring Boot itself, these dependencies are upgraded as well in a consistent way.

- **Note** a version can be specified and override the Spring Boot's recommendations if need be.

The curated list contains all the spring modules that can be use with Spring Boot as well as a refined list of third party libraries. The list is available as a standard Bills of Materials (spring-bootdependencies) that can be used with both Maven and Gradle.

contd..

- Starters are a set of convenient dependency descriptors that user can include in the application. Developer gets a one-stop shop for all the Spring and related technologies that are needed without having to hunt through sample code and copy-paste loads of dependency descriptors.

  E.g. if need to start using Spring and JPA for database access, include the spring-boot-starter-data-jpa dependency in the project.

The starters contain a lot of the dependencies that user needs to get a project up and running quickly and with a **consistent, supported set of managed transitive dependencies**.

contd..

- ✓ Spring Boot **auto-configuration** attempts to automatically configure Spring application based on the jar dependencies mentioned. E.g. if HSQLDB is on the classpath, and is not manually configured any database connection beans, then Spring Boot auto-configures an in-memory database.

Spring Initializr

- **Spring Initializr** is a **web-based tool** provided by the Pivotal Web Service.
- **Spring Initializr**, easily generate the structure of the **Spring Boot Project**. It offers extensible API for creating JVM-based projects.
- It also provides various options for the project that are expressed in a metadata model.
- The metadata model allows us to configure the list of dependencies supported by JVM and platform versions, etc.
- It serves its metadata in a well-known that provides necessary assistance to third-party clients.

Spring Initializr Modules

- Spring Initializr has the following module :

- **initializr–actuator:** It provides additional information and statistics on project generation. It is an optional module.

- **initializr–bom:** In this module, **BOM** stands for **Bill Of Materials**. In Spring Boot, BOM is a special kind of **POM** that is used to control the **versions** of a project's **dependencies**.

- It provides a central place to define and update those versions. It provides flexibility to add a dependency in our module without worrying about the versions.

- Outside the software world, the **BOM** is a list of parts, items, assemblies, and other materials required to create products. It explains **what, how,** and **where** to collect required materials.

contd..

- **initializr-docs:** It provides documentation.
- **initializr-generator:** It is a core project generation library.
- **initializr-generator-spring:**
- **initializr-generator-test:** It provides a test infrastructure for project generation.
- **initializr-metadata:** It provides metadata infrastructure for various aspects of the projects.
- **initializr-service-example:** It provides custom instances.
- **initializr-version-resolver:** It is an optional module to extract version numbers from an arbitrary POM.
- **initializr-web:** It provides web endpoints for third party clients.

Getting Started

- **SpringApplication** SpringApplication class provides a convenient way to bootstrap a Spring application that is started from a main() method. It can also be delegated to the static SpringApplication.run method –

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

- **Startup Failure** if the application fails to start, registered FailureAnalyzers provide a dedicated error message and a concrete action to fix the problem. E.g. if you start a web application on port 8080 and that port is already in use, should see something similar to the following message:
- Spring Boot provides numerous FailureAnalyzer implementations, and can be added upon.

contd..

- **Web Environment** SpringApplication attempts to create the right type of ApplicationContext for user. The algorithm used to determine a WebApplicationType –

1. If Spring MVC is present, an AnnotationConfigServletWebServerApplicationContext is used

2. If Spring MVC is not present and Spring WebFlux is present, an AnnotationConfigReactiveWebServerApplicationContext is used

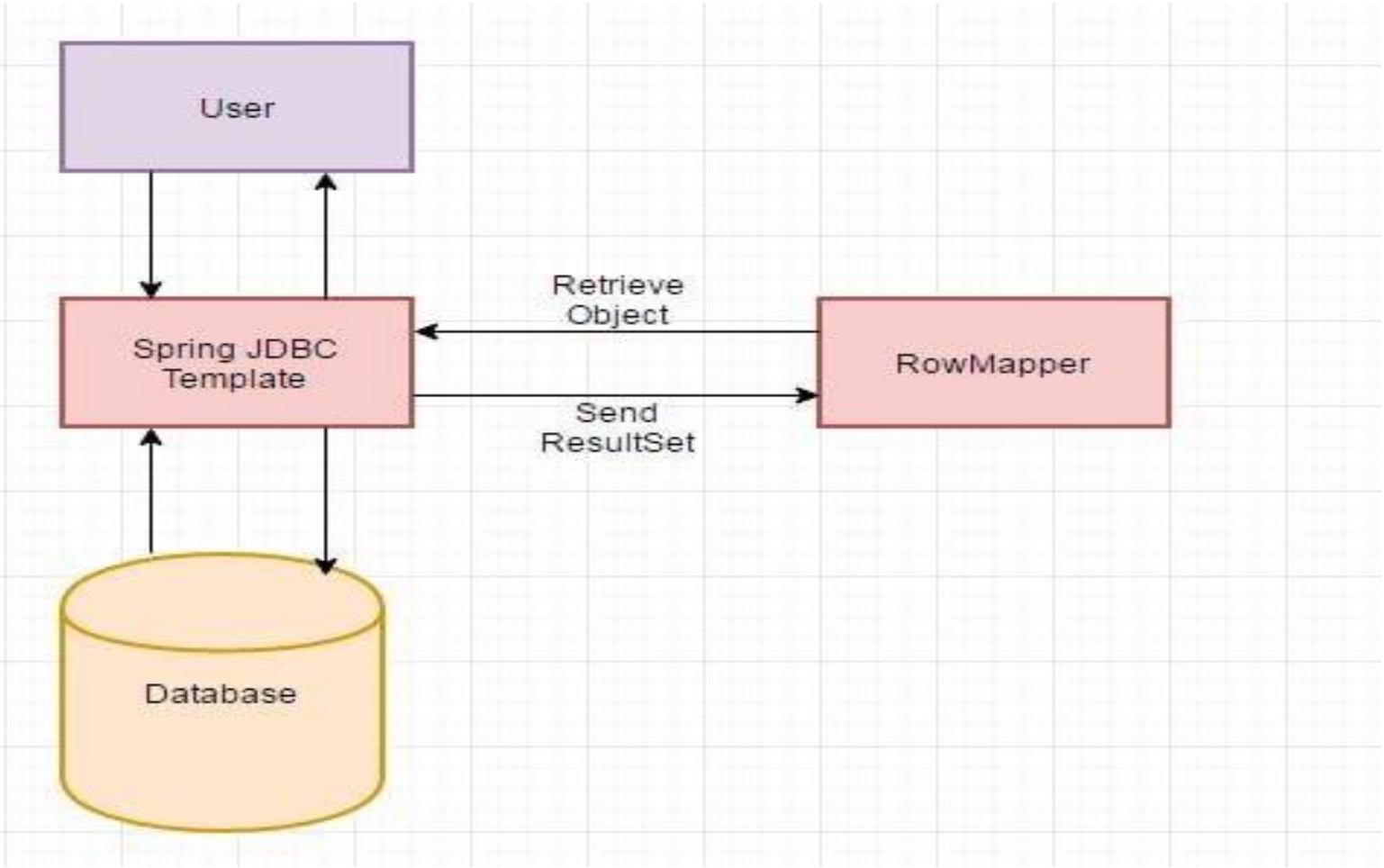
3. Otherwise, AnnotationConfigApplicationContext is used

Note – This means that if you are using Spring MVC and the new WebClient from Spring WebFlux in the same application, Spring MVC will be used by default. Can override that easily by calling setWebApplicationType(WebApplication Type).

Spring Boot CLI

- Spring Boot CLI(Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt.
- On executing Spring Boot applications using CLI, it internally uses Spring Boot Starter and Spring Boot AutoConfigurate components to resolve all dependencies and execute the application.
- Simple Spring Boot CLI Commands can be used to execute Spring Web Applications.
- Spring Boot CLI has introduced a new “spring” command to execute Groovy Scripts from command line.

Spring Boot with JDBC



Spring Boot Actuator

- Spring Boot Actuator components gives many features, but two major features are
  - ✓ Providing Management EndPoints to Spring Boot Applications.
  - ✓ Spring Boot Applications Metrics.
- When Spring Boot Web Application is run using CLI, Spring Boot Actuator automatically provides hostname as "localhost" and default port number as "8080".
- This application can be accessed using "http://localhost:8080/" end point.
- HTTP Request methods like GET and POST are used to represent Management EndPoints using Spring Boot Actuator.

Internals of Spring Boot Framework

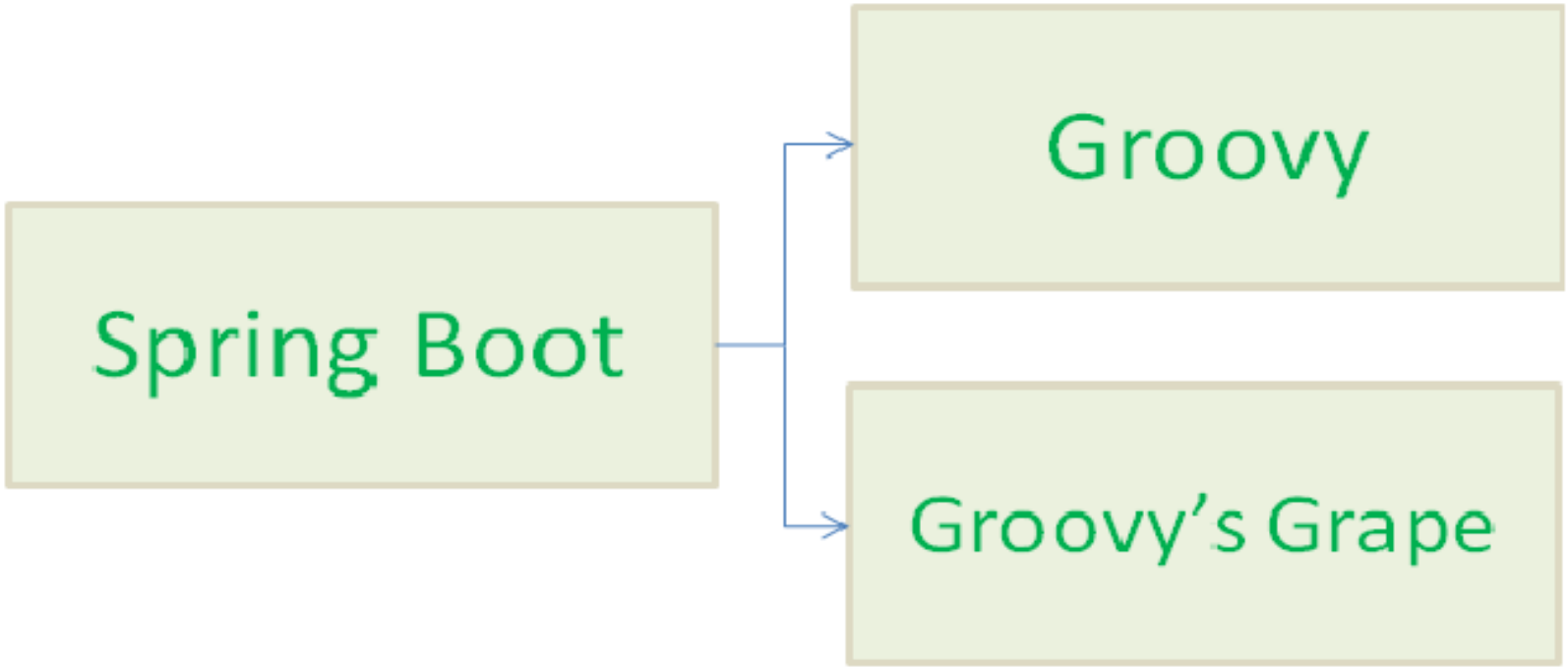
- In Groovy Programming language, there is no need to add some imports and no need to add some dependencies to Groovy project.
- When compiled Groovy scripts using Groovy Compiler(groovyc), it automatically adds all default import statements then compile it.
- In the same way, Groovy Programming language contains a JAR Dependency Resolver to resolve and add all required jar files to Groovy Project classpath.
- Spring Boot Framework internally uses Groovy to add some defaults like Default import statements, Application main() method etc. To run Groovy Scripts from CLI Command prompt, it uses this main() method to run the Spring Boot Application.

contd..

- **Grape**

- ✓ Grape is an Embedded Dependency Resolution engine. Its a JAR Dependency Manager embedded into Groovy.
- ✓ Grape lets user quickly add maven repository dependencies to the project classpath to reduce build file definitions.
- ✓ Spring Boot Framework internally depends on these two major components: Groovy and Grape.

contd..



Groovy

- Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release.
- Features of Groovy

1. Support for both static and dynamic typing.
2. Support for operator overloading.
3. Native syntax for lists and associative arrays and native support for regular expressions.
4. Native support for various markup languages such as XML and HTML.
5. Groovy is simple for Java developers since the syntax for Java and Groovy are very similar.
6. Can be used with existing Java libraries. Groovy extends the java.lang.Object.

Thank You