UNDERSTANDING SPRING BOOT AUTO CONFIGURATION

1. **We need lot of configurations to build Spring app**:
    a. Component Scan, DispatcherServlet, Data Sources, JSON Conversion, ...
2. **How can we simplify this?**
    a. Auto Configuration: Automated configuration for your app
        i. Decided based on:
            1. Which frameworks are in the Class Path?
            2. What is the existing configuration (Annotations etc)?
3. **Example**: Spring Boot Starter Web
    a. Dispatcher Servlet (DispatcherServletAutoConfiguration)
    b. Embedded Servlet Container - Tomcat is the default.
    c. (EmbeddedWebServerFactoryCustomizerAutoConfiguration)
    d. Default Error Pages (ErrorMvcAutoConfiguration)
    e. Bean<->JSON
    f. (JacksonHttpMessageConvertersConfiguration)

In Java, the classpath is a system variable that specifies the location of the Java class files and other resources that are required for a Java application to run.

When a Java application is launched, the JVM searches for the required class files and resources on the classpath. The classpath can include directories, JAR files, or ZIP archives that contain Java class files, as well as other resources such as configuration files, properties files, and images.

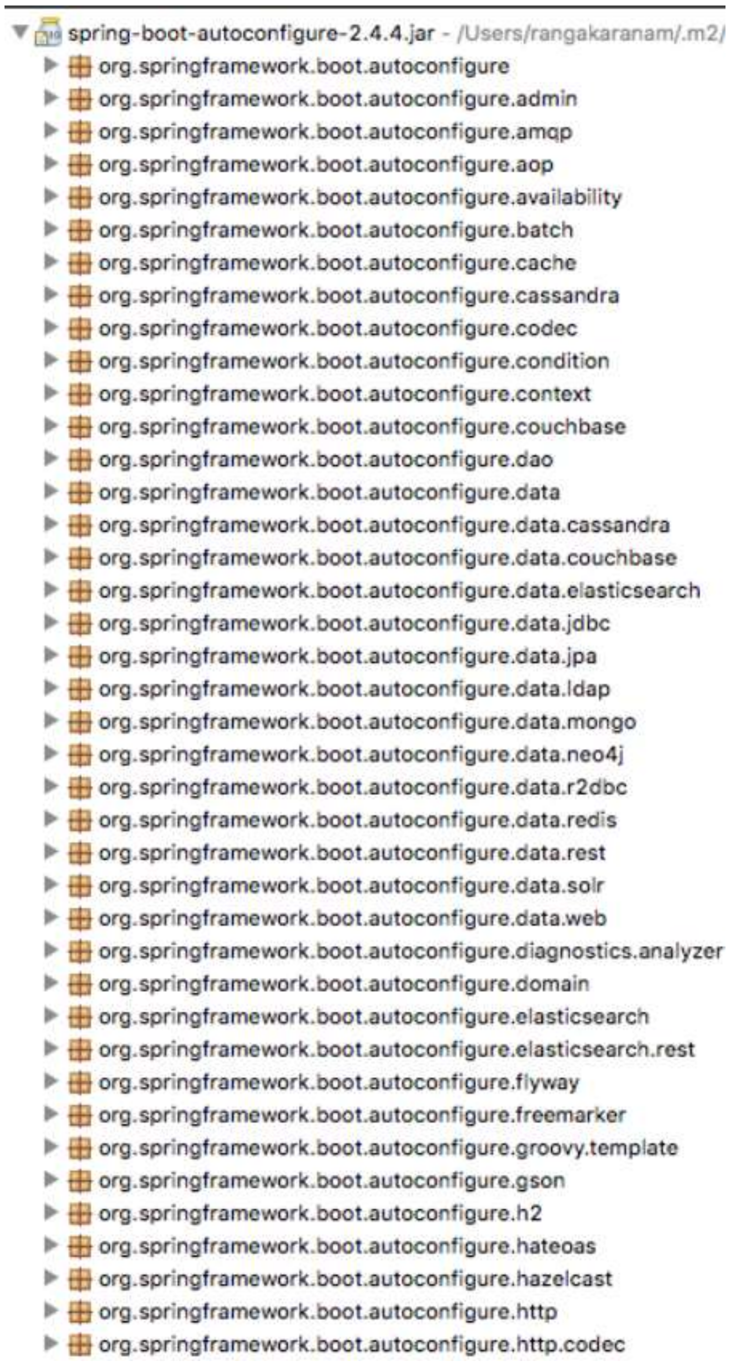
Auto-configuration is a powerful feature in Spring Boot that simplifies the process of configuring a Spring application by automatically configuring many components based on classpath settings, dependency management, and other sensible defaults.

In other words, Spring Boot's auto-configuration automatically configures beans that are typically required in a Spring application, such as databases, web servers, and messaging queues, based on the available dependencies and application settings.

When a Spring Boot application starts, it scans the classpath for any libraries and frameworks that are available, and based on this information, Spring Boot auto-configures the necessary components. This means that developers do not need to spend time and effort on configuring the application and can instead focus on writing business logic.

For example, if a developer includes the Spring Data JPA library in their project, Spring Boot will automatically configure a data source, a transaction manager, and a JPA entity manager based on sensible defaults. This makes it easy to start using Spring Data JPA without having to write a lot of configuration code.

Auto-configuration in Spring Boot is based on a set of conventions and conditions that determine which components should be configured based on the available dependencies and settings. Developers can customize the auto-configuration behaviour by providing their own configuration classes, properties, or by using conditional annotations to enable or disable certain auto-configuration features.



```
spring-boot-autoconfigure-2.4.4.jar - /Users/rangakaranam/.m2/
    org.springframework.boot.autoconfigure
    org.springframework.boot.autoconfigure.admin
    org.springframework.boot.autoconfigure.amqp
    org.springframework.boot.autoconfigure.aop
    org.springframework.boot.autoconfigure.availability
    org.springframework.boot.autoconfigure.batch
    org.springframework.boot.autoconfigure.cache
    org.springframework.boot.autoconfigure.cassandra
    org.springframework.boot.autoconfigure.codec
    org.springframework.boot.autoconfigure.condition
    org.springframework.boot.autoconfigure.context
    org.springframework.boot.autoconfigure.couchbase
    org.springframework.boot.autoconfigure.dao
    org.springframework.boot.autoconfigure.data
    org.springframework.boot.autoconfigure.data.cassandra
    org.springframework.boot.autoconfigure.data.couchbase
    org.springframework.boot.autoconfigure.data.elasticsearch
    org.springframework.boot.autoconfigure.data.jdbc
    org.springframework.boot.autoconfigure.data.jpa
    org.springframework.boot.autoconfigure.data.ldap
    org.springframework.boot.autoconfigure.data.mongo
    org.springframework.boot.autoconfigure.data.neo4j
    org.springframework.boot.autoconfigure.data.r2dbc
    org.springframework.boot.autoconfigure.data.redis
    org.springframework.boot.autoconfigure.data.rest
    org.springframework.boot.autoconfigure.data.solr
    org.springframework.boot.autoconfigure.data.web
    org.springframework.boot.autoconfigure.diagnostics.analyzer
    org.springframework.boot.autoconfigure.domain
    org.springframework.boot.autoconfigure.elasticsearch
    org.springframework.boot.autoconfigure.elasticsearch.rest
    org.springframework.boot.autoconfigure.flyway
    org.springframework.boot.autoconfigure.freemarker
    org.springframework.boot.autoconfigure.groovy.template
    org.springframework.boot.autoconfigure.gson
    org.springframework.boot.autoconfigure.h2
    org.springframework.boot.autoconfigure.hateoas
    org.springframework.boot.autoconfigure.hazelcast
    org.springframework.boot.autoconfigure.http
    org.springframework.boot.autoconfigure.http.codec
```