Dependency management -> maven, gradle, ant

Usually the class that defines the main method is a good candidate as the primary @Configuration.

Spring beans and dependency injection

----------------------------------------------------

Add @ComponentScan without any arguments or use the @SpringBootApplication annotation which implicitly includes it. All of your application components

All of your application components (@Component, @Service, @Repository, @Controller, and others) are automatically registered as Spring Beans

Folder Structure

--------------------

com

+- example

+- myapplication

+- MyApplication.java

|

+- customer

| +- Customer.java

| +- CustomerController.java

| +- CustomerService.java

| +- CustomerRepository.java

|

+- order

+- Order.java

+- OrderController.java

+- OrderService.java

+- OrderRepository.java

SpringBootApplication

-----------------------------

@SpringBootApplication same as @SpringBootConfiguration @EnableAutoConfiguration @ComponentScan

* @EnableAutoConfiguration: enable [Spring Boot’s auto-configuration mechanism](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using.auto-configuration)
* @ComponentScan: enable @Component scan on the package where the application is located
* @SpringBootConfiguration: enable registration of extra beans in the context or the import of additional configuration classes. An alternative to Spring’s standard @Configuration that aids [configuration detection](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#features.testing.spring-boot-applications.detecting-configuration) in your integration tests.

Running as package application

-----------------------------------------

$ java -jar target/myapplication-0.0.1-SNAPSHOT.jar

$ mvn spring-boot:run

$ export MAVEN\_OPTS=-Xmx1024m (operating system environment variable)