**CHAPTER- I**

**https://www.youtube.com/watch?v=OCWTzBlZiPE&list=PLqwmiTs6Z6PHb9tFkKPwlSALNLWzocd-B**

**1. Introduction to Spring Framework**

* Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application.
* It was **developed by Rod Johnson in 2003**. Spring framework makes the easy development of JavaEE application.
* Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs.

### **Advantages of Spring Framework**

There are many advantages of Spring Framework. They are as follows:

#### **1) Predefined Templates**

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

#### **2) Loose Coupling**

The Spring applications are loosely coupled because of dependency injection.

#### **3) Easy to test**

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

#### **4) Lightweight**

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

#### **5) Fast Development**

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

#### **6) Powerful abstraction**

It provides powerful abstraction to JavaEE specifications such as [JMS](https://www.javatpoint.com/jms-tutorial), [JDBC](https://www.javatpoint.com/java-jdbc), JPA and JTA.

#### **7) Declarative support**

It provides declarative support for caching, validation, transactions and formatting.

<https://techwithmaddy.com/spring-boot-architecture#heading-what-is-spring-boot>

<https://www.javaguides.net/2020/07/spring-boot-project-architecture.html>

<https://bushansirgur.in/java-3-tier-architecture/>

<https://www.interviewbit.com/blog/spring-vs-spring-boot/>

<https://www.simplilearn.com/tutorials/spring-boot-tutorial/what-is-spring-framework-and-its-advantages#:~:text=Spring%20is%20a%20lightweight%20framework,AOP%2C%20Context%2C%20and%20ORM>.

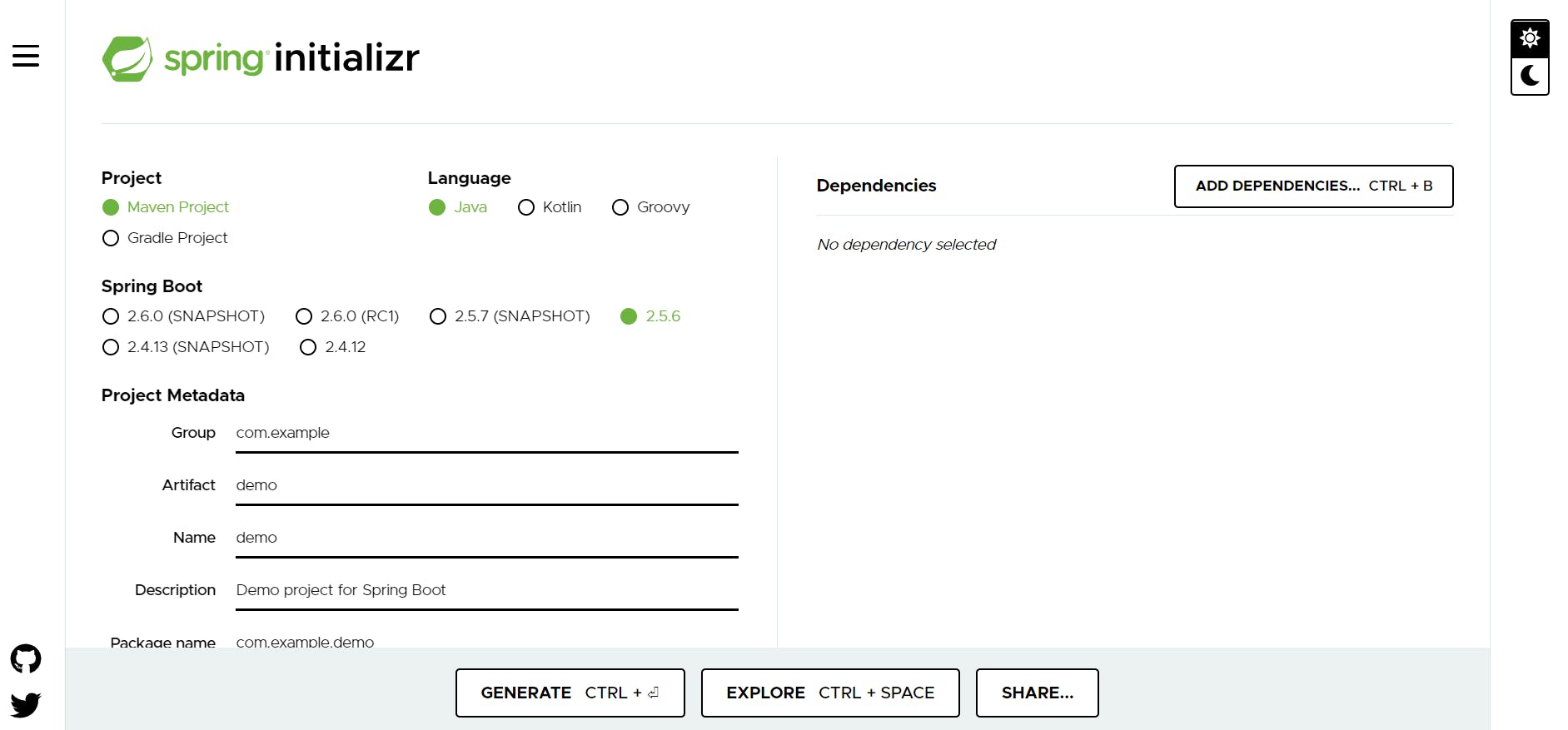
https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework#:~:text=The%20Spring%20Framework%20(Spring)%20is,old%20Java%20objects%20(POJOs).

# **What Is Spring Boot?**

Spring Boot is built on top of the Spring Framework. It's a more automated and simplified version of Spring.

Spring Boot reduces manual configuration, and through Maven (or Gradle), it makes it easier to resolve dependency conflicts. In addition, Spring Boot has a built-in HTTP server that helps engineers start the application quickly.

Spring Boot makes it easy to create an up and running application in a few minutes. You can go to [Spring Initializr](https://start.spring.io/)



And the website will generate the application for you in an instant.

If you're undecided on whether to learn Spring or Spring Boot, I'd say to at least know the basics of the Spring Framework and then jump onto Spring Boot.

### **What Is the Difference Between Spring and Spring Boot?**

If you've ever tried to write a Java application time ago, you would have noticed how much work you had to do to perform database operations. It's a tiring process!

The Spring Framework simplifies these operations, thanks to the Spring JDBC module.

Spring is the combination of multiple sub-frameworks. It has many modules, for example, Spring MVC, Spring JDBC, Spring AOP, Spring ORM, and Spring Test. The core feature of the Spring Framework is Dependency Injection.

Even though Spring has made the development of Java applications easier, I think that it is vast and complicated to master. In the past, I tried to learn the Spring Framework from scratch, and I found it a long experience. This is where **Spring Boot** comes in handy, though. 😁

Spring Boot is an extension of Spring, which eliminates the boilerplate configurations required for setting up a Spring application. Featuring default codes and annotation based configuration, Spring Boot enables a faster and more efficient development ecosystem.

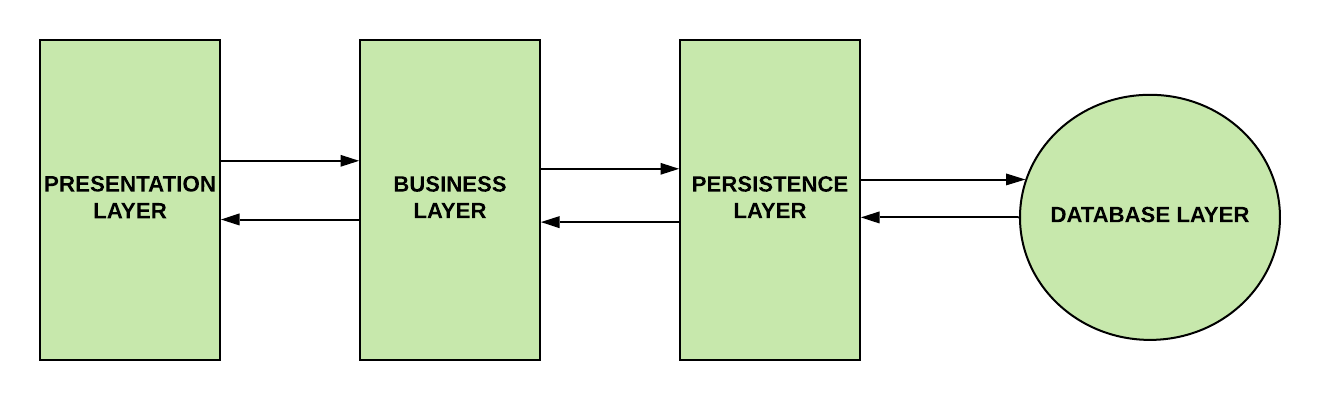
Loose coupling in Java means that the classes are independent of each other.

Now let's talk about Spring Boot Architecture.

### **Spring Boot Architecture**

Spring Boot Architecture has four layers:

* **Presentation Layer**
* **Business Layer**
* **Persistence Layer**
* **Database Layer**



### **Presentation Layer**

This layer is at the top of the architecture. This tier is responsible for:

✔️ Performing authentication.

✔️ Converting JSON data into an object (and vice versa).

✔️ Handling HTTP requests.

✔️ Transfering authentication to the business layer.

The presentation layer is the equivalent of the Controller class. The Controller class handles all the incoming REST API requests (GET, POST, PUT, DELETE, PATCH) from the Client.

### **Business Layer**

The business layer is responsible for:

✔️ Performing validation.

✔️ Performing authorization.

✔️ Handling the business logic and rules.

This layer is the equivalent to the Service class. It's where we handle the business logic. If you're wondering what do we mean by "business logic", I found an interesting discussion on [StackExchange](https://softwareengineering.stackexchange.com/questions/234251/what-really-is-the-business-logic" \t "_blank). In short, the business logic in software engineering is where we decide what the software needs to do. An example of this is validation. If you are ever requested to validate something, this needs to happen inside the Service class.

The Business layer communicates with both the Presentation layer and the Persistence Layer.

### **Persistence Layer**

This layer is responsible for:

✔️ Containing storage logic.

✔️ Fetching objects and translating them into database rows (and vice versa).

This layer is the equivalent of the Repository interface. We write database queries inside this interface.

The Persistence layer is the only layer that communicates with the Business layer and the Database layer.

### **Database Layer**

This layer is responsible for:

✔️ Performing database operations (mainly CRUD operations).

This layer is simply the actual database that you decide to use to build your application.

### **Spring Boot Workflow**



Spring Boot workflow acts like this:

1. The Client makes an HTTP request.
2. The Controller class receives the HTTP request.
3. The Controller understands what type of request will process, and then it deals with it.
4. If it is needed, it calls the service class.
5. The Service Class is going to handle the business logic. It does this on the data from the database.
6. If everything goes well, we return a JSP page.

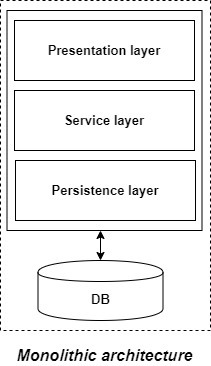
* Presentation Layer - **controller**package
* Business Logic Layer - **service**package
* Data Access Layer - **repository**package
* Persistence layer -use the **JPARepository** Interface

**MONOLITHIC ARECHITECTURE AND MICROSERVICES ARCHITECTURE**

**Monolithic Application:**

**If all the functionalities of a project exist in a single codebase, then that application is known as a monolithic application.**

**We design our application in various layers like presentation, service, and persistence and then deploy that codebase as a single jar/war file. This is nothing but a monolithic application, where “mono” represents the single codebase containing all the required functionalities.**

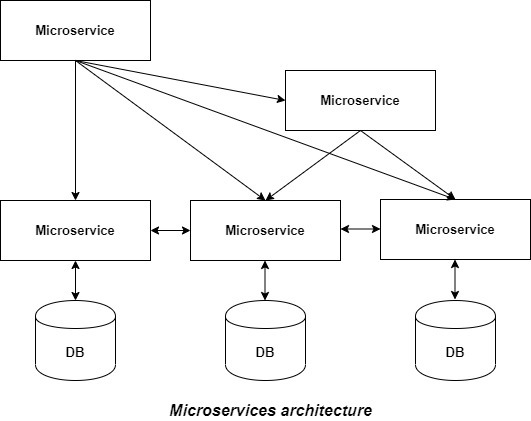


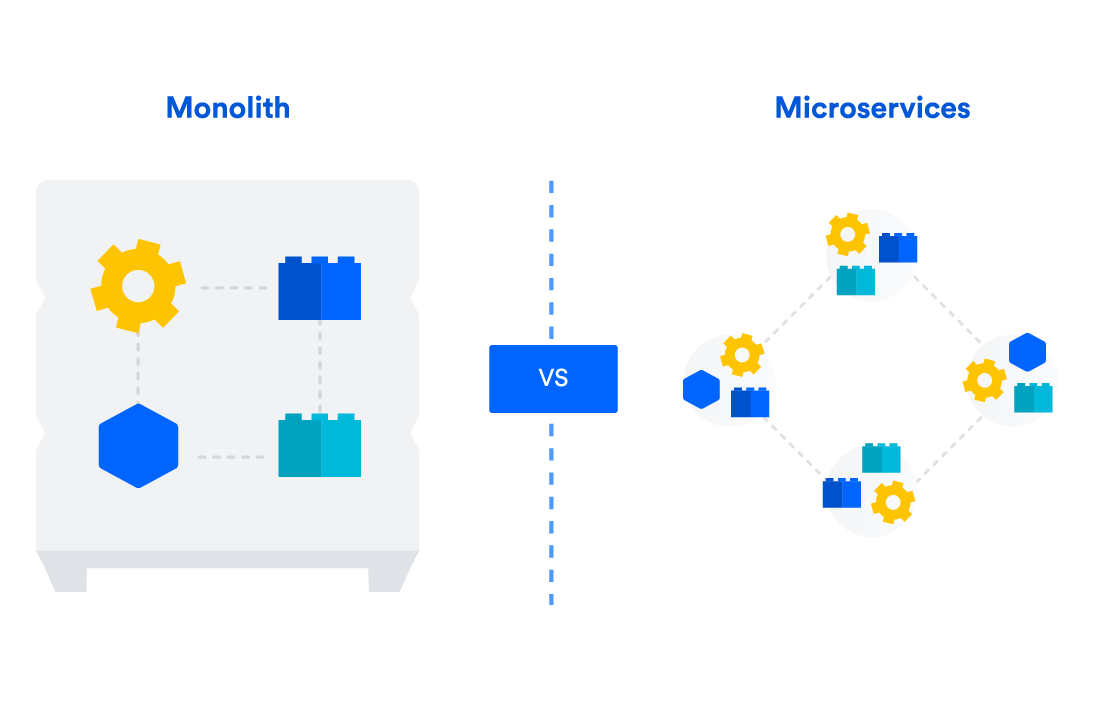
Disadvantages of Monolithic applications:

* It becomes too large with time and hence, difficult to manage.
* We need to redeploy the whole application, even for a small change.
* As the size of the application increases, its start-up and deployment time also increases.
* For any new developer joining the project, it is very difficult to understand the logic of a large Monolithic application even if his responsibility is related to a single functionality.
* It is not very reliable, as a single bug in any module can bring down the entire monolithic application.

**Microservice:**

It is an architectural development style in which the application is made up of smaller services that handle a small portion of the functionality and data by communicating with each other directly using lightweight protocols like HTTP.





Advantage of Microservice:

* Microservices are self-contained, independent deployment module.
* The cost of scaling is comparatively less than the monolithic architecture.
* Microservices are independently manageable services. It can enable more and more services as the need arises. It minimizes the impact on existing service.
* Less dependency and easy to test.

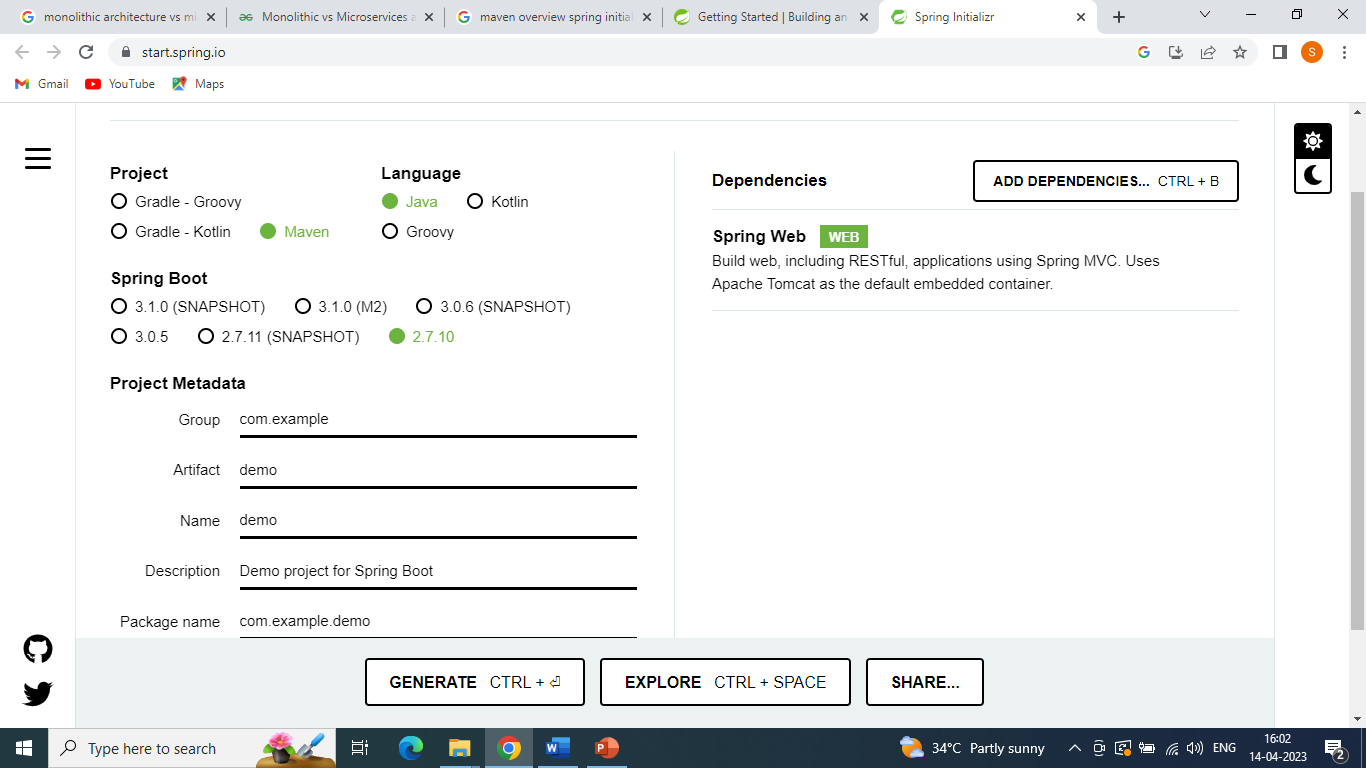
**Difference between Spring and springBoot:**

**(Refer student)**

**CHAPTER -2**

**Building Spring Boot Application**

Spring Intializer:



Maven:

Maven is used to download required dependencies(jar file).Based on mentioned starters name.