What is Spring Boot

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

It is a Spring module that provides the **RAD (*Rapid Application Development*)** feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.



In short, Spring Boot is the combination of **Spring Framework** and **Embedded Servers**.

In Spring Boot, there is no requirement for XML configuration (deployment descriptor). It uses convention over configuration software design paradigm that means it decreases the effort of the developer.

We can use Spring **STS IDE** or **Spring Initializr** to develop Spring Boot Java applications.

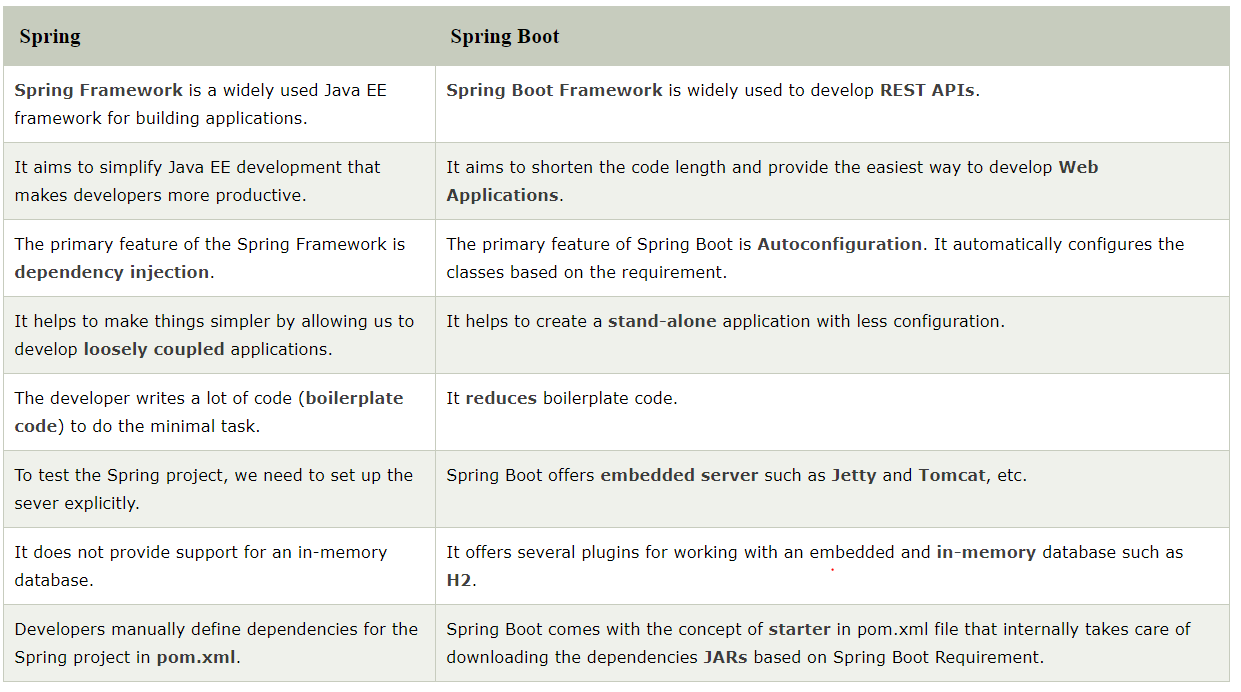
**Why should we use Spring Boot Framework?**

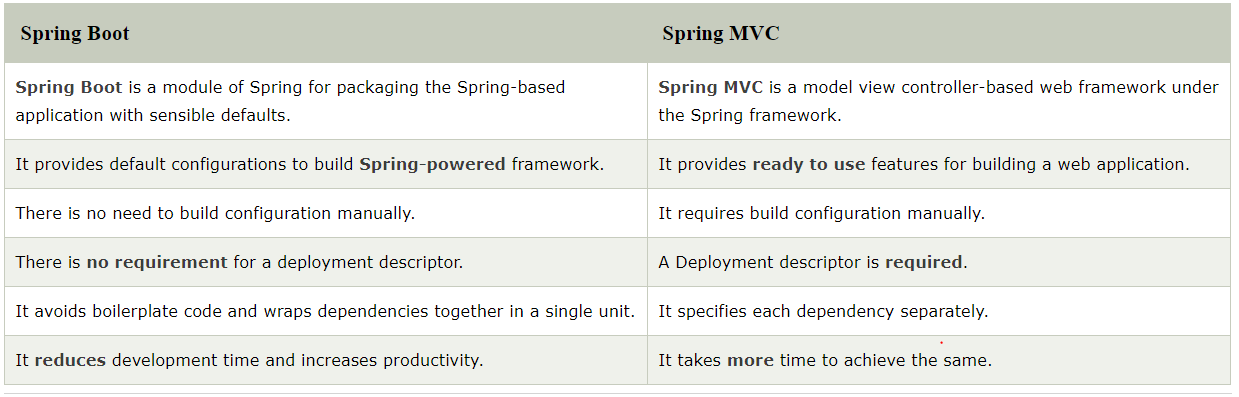
We should use Spring Boot Framework because:

* The dependency injection approach is used in Spring Boot.
* It contains powerful database transaction management capabilities.
* It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.
* It reduces the cost and development time of the application.

Along with the Spring Boot Framework, many other Spring sister projects help to build applications addressing modern business needs. There are the following Spring sister projects are as follows:

* **Spring Data:** It simplifies data access from the relational and **NoSQL** databases.
* **Spring Batch:** It provides powerful **batch** processing.
* **Spring Security:** It is a security framework that provides robust **security** to applications.
* **Spring Social:** It supports integration with **social networking** like LinkedIn.
* **Spring Integration:** It is an implementation of Enterprise Integration Patterns. It facilitates integration with other **enterprise applications** using lightweight messaging and declarative adapters.





Spring Boot is a module of the Spring Framework. It is used to create stand-alone, production-grade Spring Based Applications with minimum efforts. It is developed on top of the core Spring Framework.

Spring Boot follows a layered architecture in which each layer communicates with the layer directly below or above (hierarchical structure) it.

Before understanding the **Spring Boot Architecture**, we must know the different layers and classes present in it. There are **four** layers in Spring Boot are as follows:

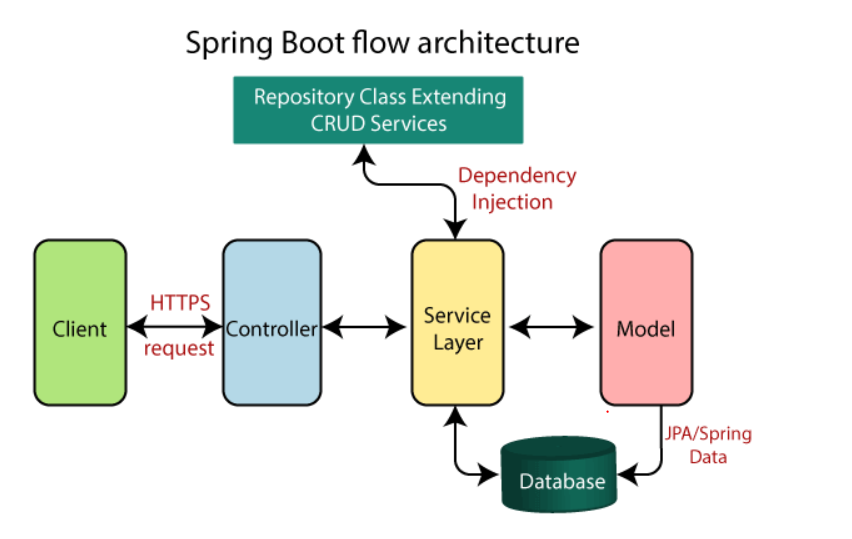
* **Presentation Layer**
* **Business Layer**
* **Persistence Layer**
* **Database Layer**

**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.



* Now we have 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. It performs the 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 Hibernate

## Create Repository class to Read Student information

/src/main/java/com/in28minutes/springboot/jpa/hibernate/h2/example/student/StudentRepository.java

We create a simple interface StudentRepository extending JpaRepository.

package com.in28minutes.springboot.jpa.hibernate.h2.example.student;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface StudentRepository extends JpaRepository<Student, Long>{

}

public interface StudentRepository extends JpaRepository<Student, Long> - We are extending JpaRepository using two generics - Student & Long. Student is the entity that is being managed and the primary key of Student is Long.

### JpaRepository

JpaRepository (Defined in Spring Data JPA) is the JPA specific Repository interface.

public interface JpaRepository<T, ID extends Serializable>

extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {

*JpaRepository extends PagingAndSortingRepository which in turn extends CrudRepository interface. So, JpaRepository inherits all the methods from the two interfaces shown below.*

PagingAndSortingRepository

public abstract interface PagingAndSortingRepository extends CrudRepository {

public abstract Iterable findAll(Sort arg0);

public abstract Page findAll(Pageable arg0);

}

CrudRepository

public interface CrudRepository<T, ID extends Serializable>

extends Repository<T, ID> {

<S extends T> S save(S entity);

T findOne(ID primaryKey);

Iterable<T> findAll();

Long count();

void delete(T entity);

boolean exists(ID primaryKey);

*// … more functionality omitted.*

}