**Basic Spring Boot Project (For Spring Boot and PostgreSQL Connectivity)**

1. Generated Sprint Boot project through start.spring.io with required dependencies

2. Extracted and opened in IntelliJ IDEA

3. Required packages:

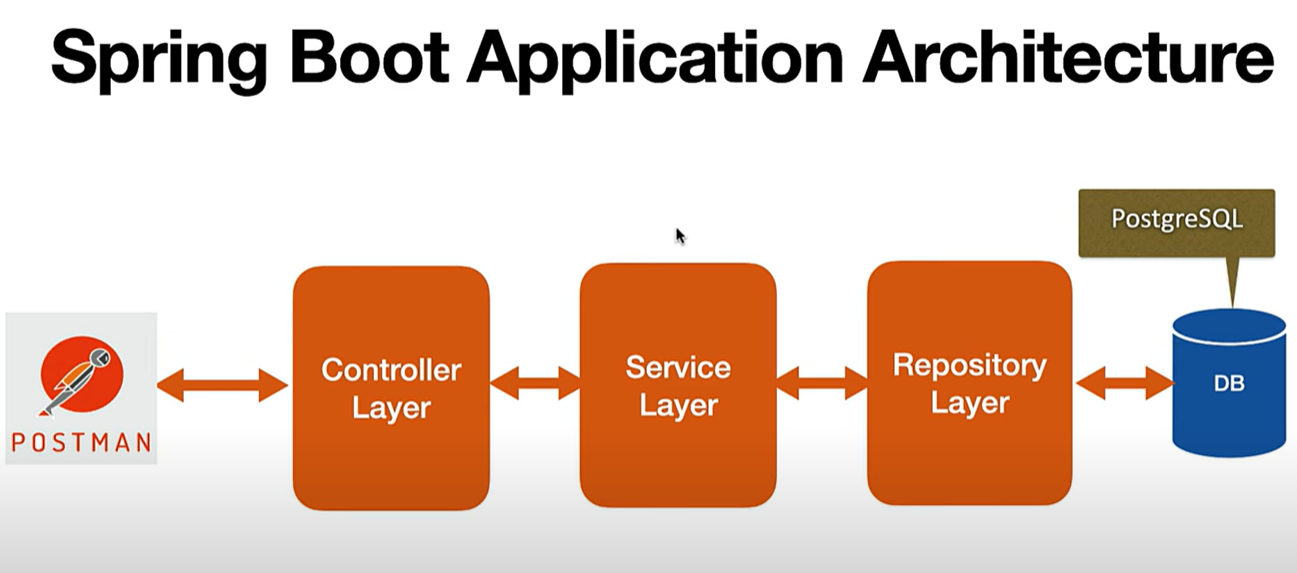
a) service

b) controller

c) repository

d) domain

e) model (entity)



> Controller Layer (@RestController, @Controller) – I. Use to handle HTTP request from the client.

II. It takes data and send it to the service layer

III. It processes the response and sends back to the user.

> Service Layer (@Service) – I. It is used for handling the business logic

II. It is the main brain of the application.

III. All the rules and calculations are done in this.

IV. It takes data from repository and give to the controller layer and also processes the data.

V. It is a reusable and testable component.

> Repository Layer (@Repository) – I. It works directly with the database.

II. It performs the CRUD Operations.

III. It performs query on the data and give it to the service layer.

> Domain Layer – I. It represents those objects which are important for the which are important for the application’s business rules.

> Model/Entity Layer (@Entity, @Table) – I. It represents the database table in the form of Java Class.

II. It basically represents the structure of the database.

III. To work with the Hibernate/JPA we use @Entity annotation

> DTO Layer (Data Transfer Object Layer) – I. It simplifies the transfer of data between client and server.

II. It contains only those fields which clients required.

III. It act as an intermediary between Model (Entity) and client.

Client (Browser/Postman)

↓

[Controller] – Handles the HTTP Request

↓

[Service] – Processes the business Logic

↓

[Repository] – Interacts with the database

↓

Database (PostgreSQL/MySQL)

By following the following steps the application becomes modular, scalable and testable.

4. Spring Boot uses for JPA (Java Persistence API) Implementation

5. For the connection with PostgreSQL go to the resources->application.properties and write the following things

##connecting to the database

logging.level.org.springframework=debug

spring.datasource.dbcp2.driver-class-name=org.postgresql.Driver

spring.datasource.url=jdbc:postgresql://localhost:5432/newappdb

spring.datasource.username=postgres

spring.datasource.password=sameer

##jpa config

spring.jpa.properties..hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

6. Create entity package in which we will create a model:

Under this package we will create a model using Java Class named Person:

package com.example.myapp.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import lombok.Data;  
  
@Data  
@Entity  
@Table(name="person")  
public class Person {  
 @Id  
 private long id;  
 private String name;  
}

7. Create repository package under which we have to create Java Class PersonRepository:

package com.example.myapp.repository;  
  
import com.example.myapp.entity.Person;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.rest.core.annotation.RepositoryRestResource;  
  
@RepositoryRestResource  
public interface PersonRepo extends JpaRepository<Person,Long> {  
  
}

8. Create controller package under which we have to create Java Class PersonController.

package com.example.myapp.controller;  
  
import com.example.myapp.entity.Person;  
import com.example.myapp.repository.PersonRepo;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class PersonController {  
  
 @Autowired  
 PersonRepo repo;  
  
 @PostMapping("/addPerson")  
 public void addPerson(@RequestBody Person person)  
 {  
 repo.save(person);  
 }  
}