

## Spring boot: Project Setup and Layered Architecture (Concept ...

"Concept && Coding" YT Video Notes

### Setting up the Project

1. Go to Spring Initializr i.e. "start.spring.io"

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

Language

☒ Java

☐ Kotlin

☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT)

☐ 3.3.0 (M1)

☐ 3.2.4 (SNAPSHOT)

☒ 3.2.3

☐ 3.1.10 (SNAPSHOT)

☐ 3.1.9

Project Metadata

Group

com.conceptandcoding

Artifact

learningspringboot

Name

springboot application

Description

project for learning spring boot

Package name

com.conceptandcoding.learningspringboot

Packaging

☒ Jar

☐ War

Java

☐ 21

☒ 17

Dependencies

ADD DEPENDENCIES... 36 + 8

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Report Abuse

### LAYERED ARCHITECTURE

```
graph LR
    Clients[Clients] <--> CL[Controller Layer]
    CL <--> SL[Service Layer]
    SL <--> RL[Repository Layer]
    RL <--> DB[(DB)]
    CL <--> DTO[DTO]
    CL <--> Utility[Utility]
    CL <--> Entity[Entity]
    CL <--> Configuration[Configuration]
```

The diagram illustrates a layered architecture. At the top, four boxes represent cross-cutting concerns: DTO, Utility, Entity, and Configuration. Below these is a large container box. Inside this container, three layers are shown: Controller Layer, Service Layer, and Repository Layer, connected by bidirectional arrows. To the left of the Controller Layer is a box for Clients, also connected by a bidirectional arrow. Below the Repository Layer is a cylinder representing the Database (DB), connected by a bidirectional arrow.

```
@RestController
@RequestMapping("/payments")
public class PaymentController {

    @Autowired
    PaymentService paymentService;

    @GetMapping("/{id}")
    public ResponseEntity<PaymentResponse> getPaymentById(@PathVariable Long id) {

        //Map incoming data to internal request DTO
        PaymentRequest internalRequestObj = new PaymentRequest();
        internalRequestObj.setPaymentId(id);

        //pass this internalRequestObj to further layer for processing
        PaymentResponse payment = paymentService.getPaymentDetailsById(internalRequestObj);

        //return the Response DTO
        return ResponseEntity.ok(payment);
    }
}
```

```
@Service
public class PaymentService {

    @Autowired
    PaymentRepository paymentRepository;

    public PaymentResponse getPaymentDetailsById(PaymentRequest internalRequestObj) {

        PaymentEntity paymentModel = paymentRepository.getPaymentById(internalRequestObj);

        //Map it to response obj
        PaymentResponse paymentResponse = mapPaymentToResponse(paymentModel);
        return paymentResponse;
    }

    private PaymentResponse mapPaymentToResponse(PaymentEntity paymentEntity){

        PaymentResponse response = new PaymentResponse();
        response.setPaymentId(paymentEntity.getId());
        response.setAmount(paymentEntity.getAmount());
        response.setCurrency(paymentEntity.getCurrency());
        return response;
    }
}
```

```
@Repository
public class PaymentRepository {

    public PaymentEntity getPaymentById(PaymentRequest request){
        PaymentEntity paymentModel = executeQuery(request);
        return paymentModel;
    }

    private PaymentEntity executeQuery(PaymentRequest request){
        //connect with DB and fetch the data
        PaymentEntity payment = new PaymentEntity();
        payment.setId(request.getPaymentId());
        payment.setPaymentCurrency("INR");
        payment.setPaymentAmount(100.00);
        return payment;
    }
}
```