

Spring boot: Project Setup and Layered Archit...

"Concept && Coding" YT Video Notes

Setting up the Project

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

The screenshot shows the Spring Initializr interface. Under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', version '3.2.4 (SNAPSHOT)' is selected. In the 'Dependencies' section, 'Spring Web' is selected. The 'Project Metadata' section includes fields for Group (com.conceptandcoding), Artifact (learningspringboot), Name (springboot application), Description (project for learning spring boot), and Package name (com.conceptandcoding.learningspringboot). The 'Packaging' field is set to 'Jar'. The Java version dropdown shows '21' is selected.

Last Edited : Aug 28, 2024

Spring boot: Project Setup and Layered Archit...

The diagram illustrates a layered architecture. At the top level is the 'Clients' box, which has a bidirectional arrow pointing to the 'Controller Layer' box. The 'Controller Layer' box has a bidirectional arrow pointing to the 'Service Layer' box. The 'Service Layer' box has a bidirectional arrow pointing to the 'Repository Layer' box. The 'Repository Layer' box has a vertical arrow pointing down to a cylinder labeled 'DB'.

```
@PostMapping("/payments")
public class PaymentController {
    @Autowired
    PaymentService paymentService;

    @PostMapping("/v1")
    public ResponseEntity<PaymentResponse> getPaymentById(@PathVariable Long id) {
        //map incoming data to internal request
        PaymentRequest internalRequest = new PaymentRequest();
        internalRequest.setId(id);
        internalRequest.setAmount(100);

        //map internal request to further layer for processing
        PaymentResponse payment = paymentService.getPaymentById(id, internalRequest);
        //return the response
        return ResponseEntity.ok(payment);
    }
}
```

```
public class PaymentService {
    @Autowired
    PaymentRepository paymentRepository;

    @Override
    public PaymentResponse getPaymentById(PaymentRequest internalRequest) {
        PaymentEntity payment = paymentRepository.getPaymentBy(id, internalRequest);
        PaymentResponse paymentResponse = mapEntityToResponse(payment);
        return paymentResponse;
    }
}

private PaymentResponse mapEntityToResponse(PaymentEntity paymentEntity) {
    PaymentResponse response = new PaymentResponse();
    response.setPaymentId(paymentEntity.getId());
    response.setAmount(paymentEntity.getAmount());
    response.setCurrency(paymentEntity.getCurrency());
    response.setServerTimestamp(paymentEntity.getServerTimestamp());
    return response;
}
```

```
@Repository
public class PaymentRepository {
    public PaymentEntity getPaymentById(PaymentRequest request) {
        PaymentEntity payment = executeQuery(request);
        return payment;
    }

    private PaymentEntity executeQuery(PaymentRequest request) {
        //connect with DB and fetch the data
        PaymentEntity payment = new PaymentEntity();
        payment.setInternalRequest(getInternalRequest());
        payment.setPaymentId("1");
        payment.setPaymentCurrency("INR");
        payment.setPaymentAmount(100);
        return payment;
    }
}
```