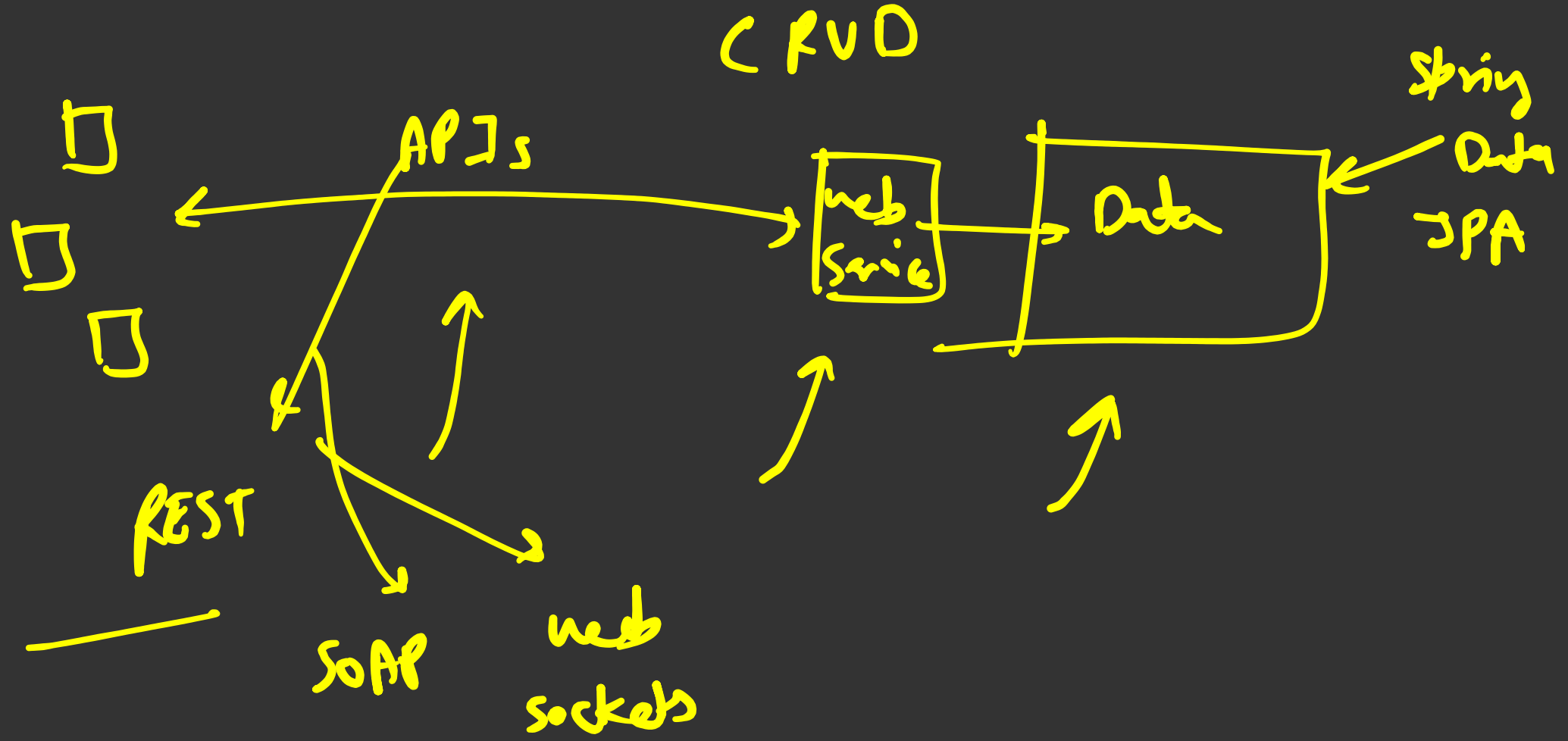
2.1

Spring Boot Web



REST APIs

REST (Representational State Transfer) APIs (Application Programming Interfaces) are a set of rules and conventions for building and interacting with web services.

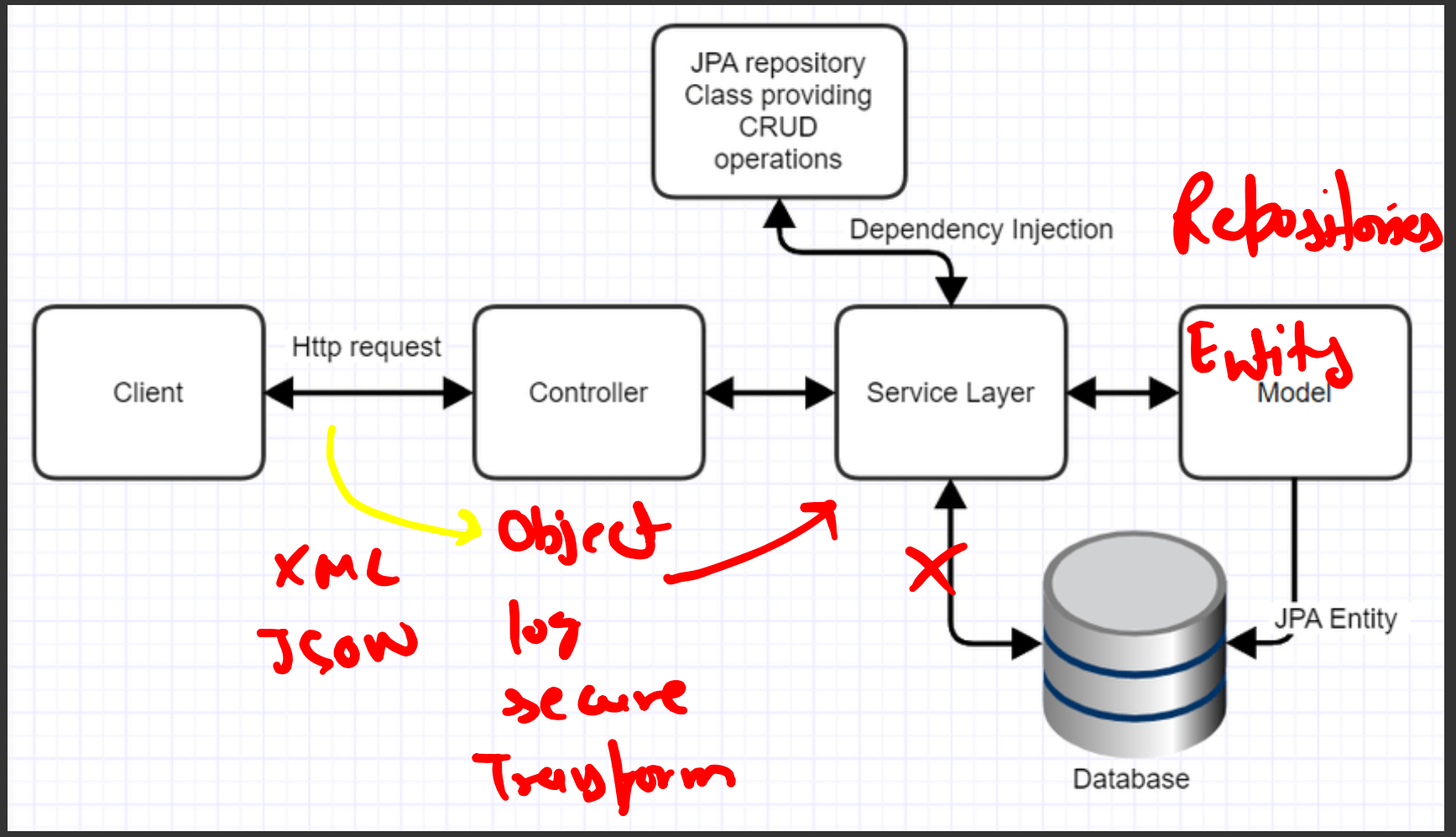
- **GET /users**: Retrieve a list of all users.
- **GET /users/{id}**: Retrieve a specific user by ID.
- **POST /users**: Create a new user.
- **PUT /users/{id}**: Update an existing user by ID.
- **PATCH /users/{id}**: Partially update an existing user by ID.
- **DELETE /users/{id}**: Delete a user by ID.

spring-boot-starter-web

spring-boot-starter-web contains the following dependencies:

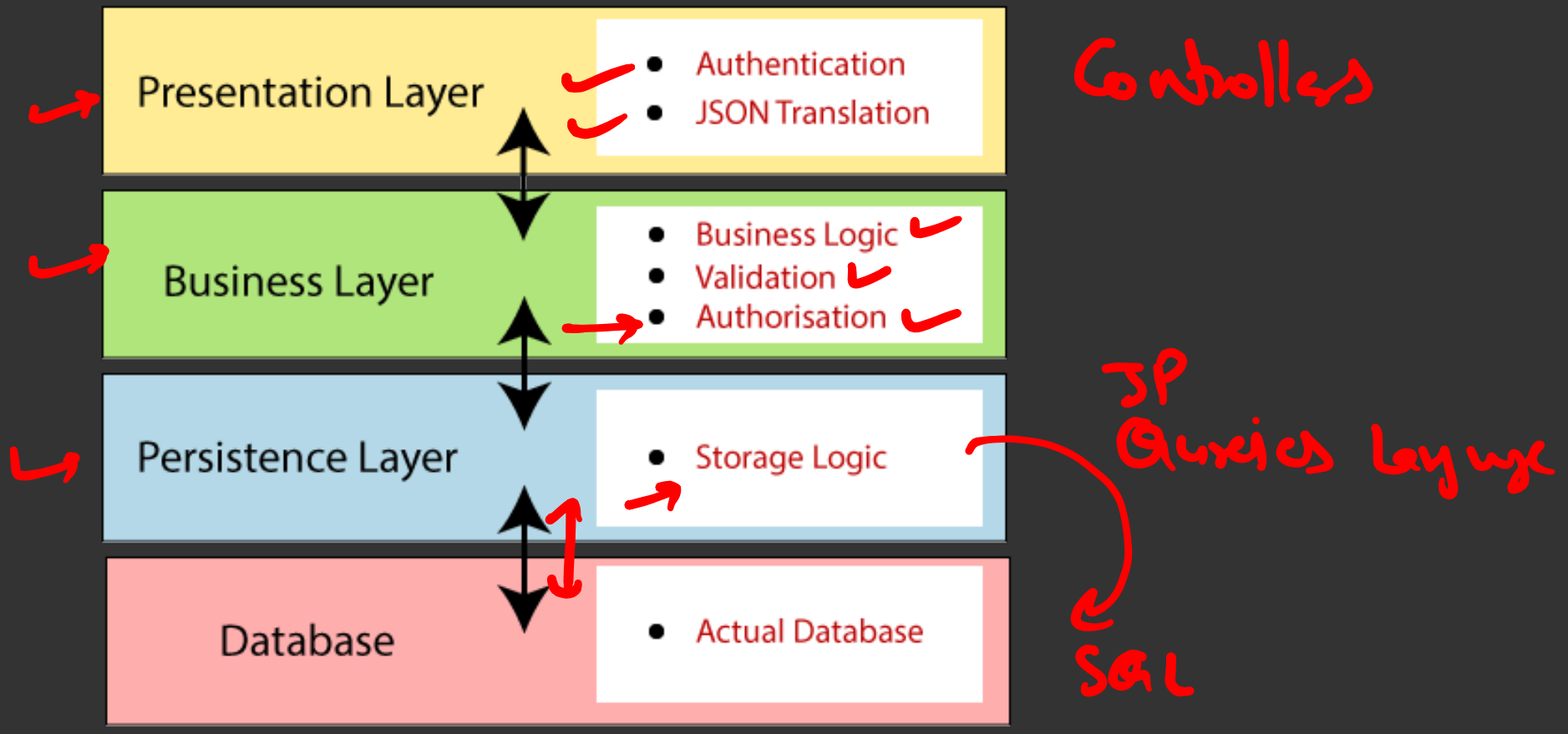
- spring-boot-starter
- jackson
- spring-core
- spring-mvc
- spring-boot-starter-tomcat

Spring Boot MVC Architecture



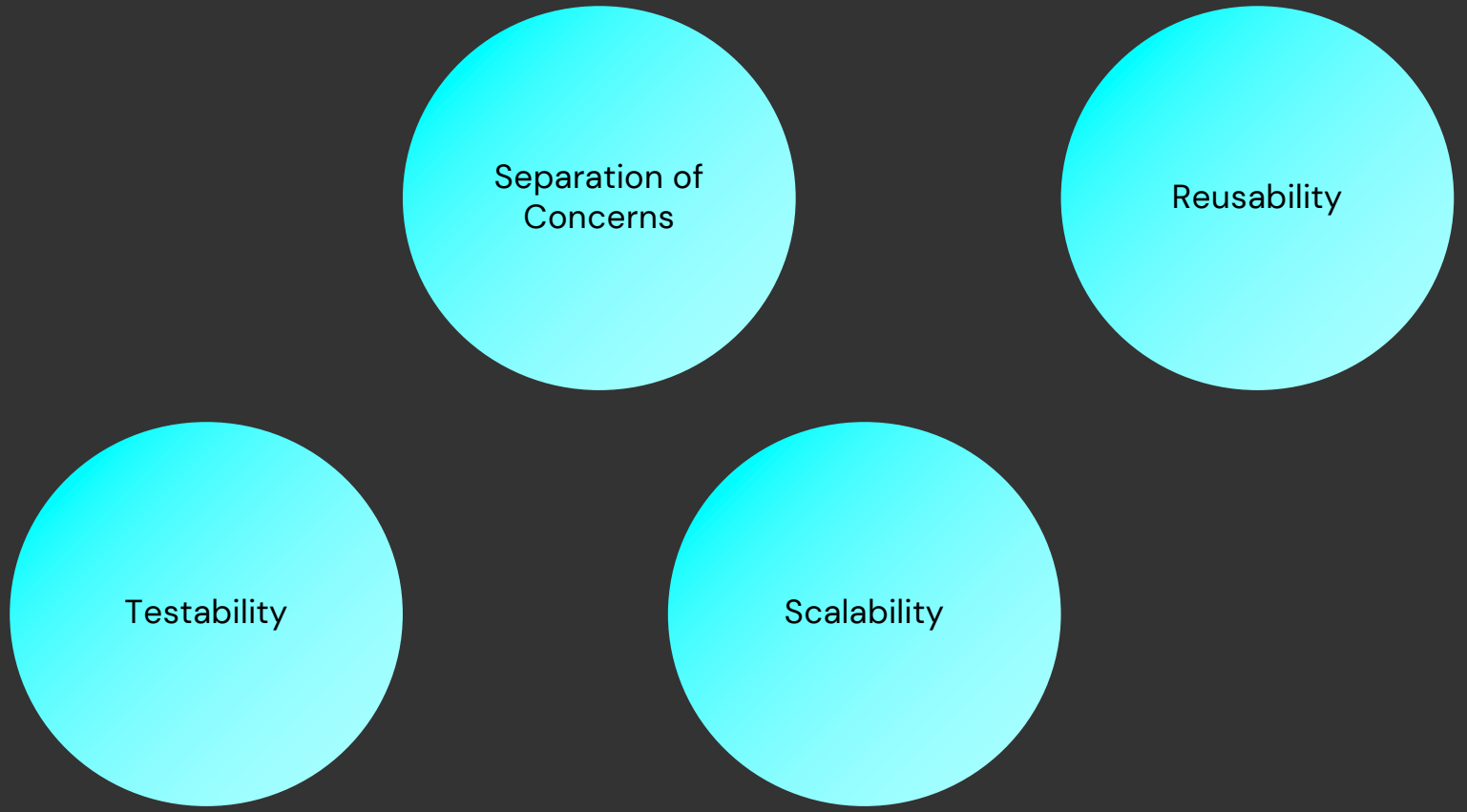
Model
View
Controller

Layered Architecture

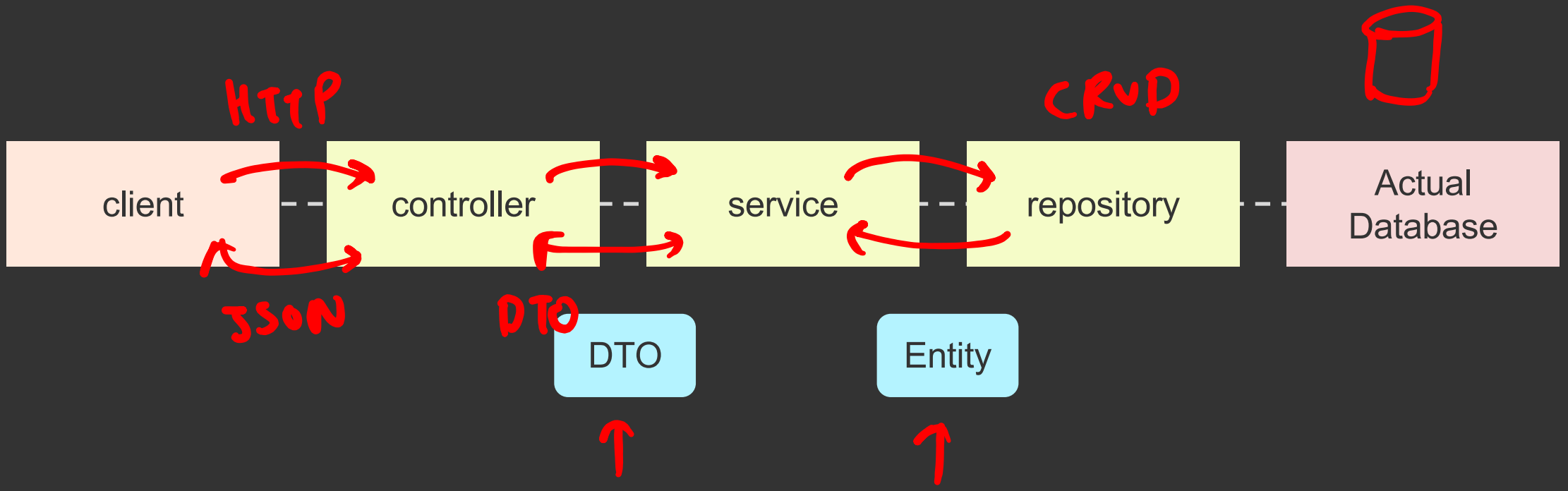


CODING SHUTTLE

Why to use MVC Architecture



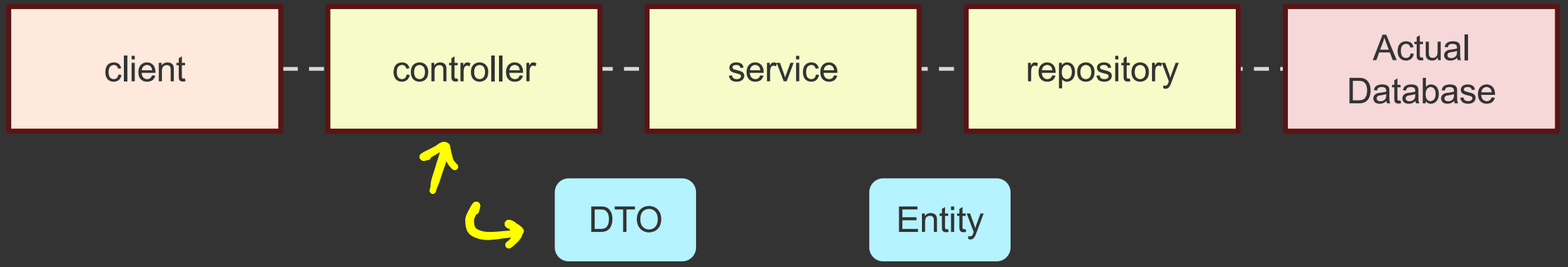
Spring Boot Web Project Structure



2.2

Presentation Layer

Spring Boot Web Project Structure



Annotated Controllers

Spring MVC provides an annotation-based programming model where @Controller and @RestController components use annotations to express request mappings, request input, exception handling, and more.

The @RestController annotation is a shorthand for @Controller and @ResponseBody, meaning all methods in the controller will return JSON/XML directly to the response body.

Request Mappings

You can use the @RequestMapping annotation to map requests to controllers methods. It has various attributes to match by URL, HTTP method, request parameters, headers, and media types.

There are also HTTP method specific shortcut variants of @RequestMapping:

- @GetMapping
- @PostMapping
- @PutMapping
- @DeleteMapping
- @PatchMapping

Dynamic URLs Paths

@PathVariable

/employees/123

Use path variables when the parameter is an essential part of the URL path that identifies a resource.

@RequestParam

/employees?id=123

Use query parameters when the parameter is optional and used for filtering, sorting, or other modifications to the request.

RequestBody

@RequestBody is used to bind the HTTP request body to a Java object. When a client sends data in the body of a request (e.g., JSON or XML), @RequestBody maps this data to a Java object.

Use Case:

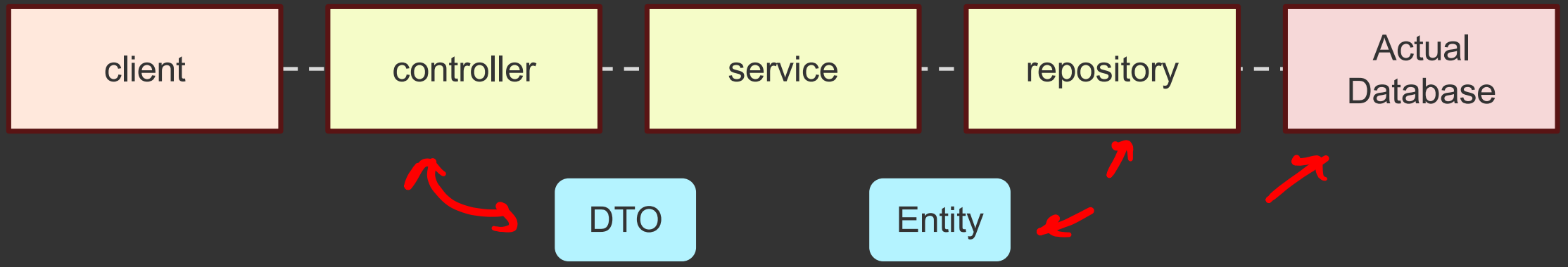
- Typically used in POST, PUT, and PATCH methods where the client sends data that needs to be processed by the server.
- Converts JSON or XML data from the request body into a Java object using a message converter (e.g., Jackson for JSON).

CODING SHUTTLE

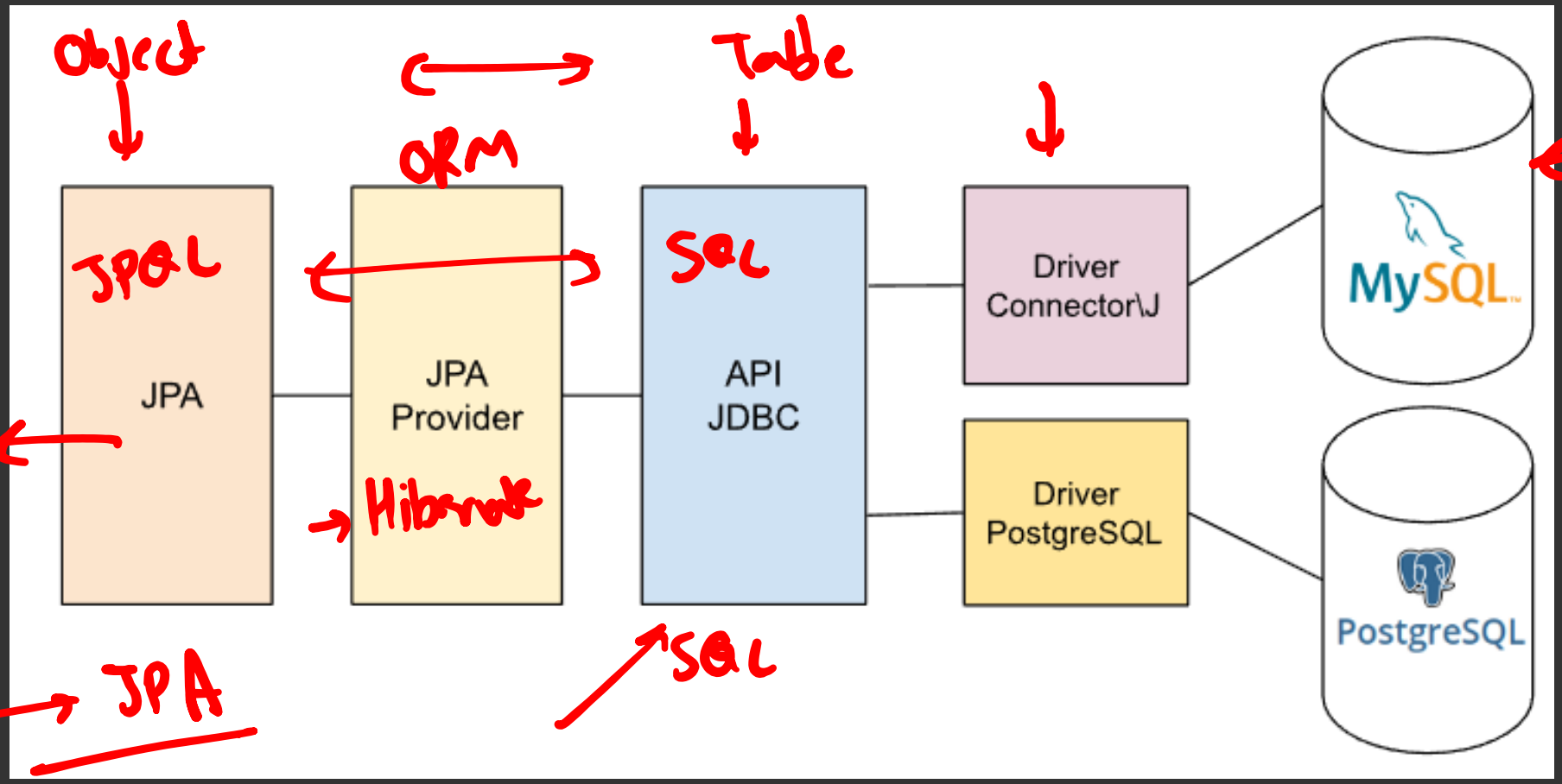
2.3

Persistence Layer & JPA

Spring Boot Web Project Structure



Java Persistence API - JPA



Spring H2 Database

Spring Boot makes it very easy to set up an in-memory H2 database for development and testing purposes. H2 is a lightweight, fast, and open-source relational database engine that can run in both in-memory and persistent modes.

Dependency:

```
<groupId>com.h2database</groupId>
```

@Entity Annotation

The @Entity annotation in Spring and Java Persistence API (JPA) is used to mark a class as a persistent entity, meaning it represents a table in a relational database. This is a fundamental part of the ORM (Object-Relational Mapping) paradigm, where Java objects are mapped to database tables.

Key Points of @Entity:

- Class-Level Annotation
- Primary Key
- Automatic Table Mapping

JPARepository Interface

The JPARepository interface in Spring Data JPA provides a set of CRUD (Create, Read, Update, Delete) operations and query methods for interacting with the database.

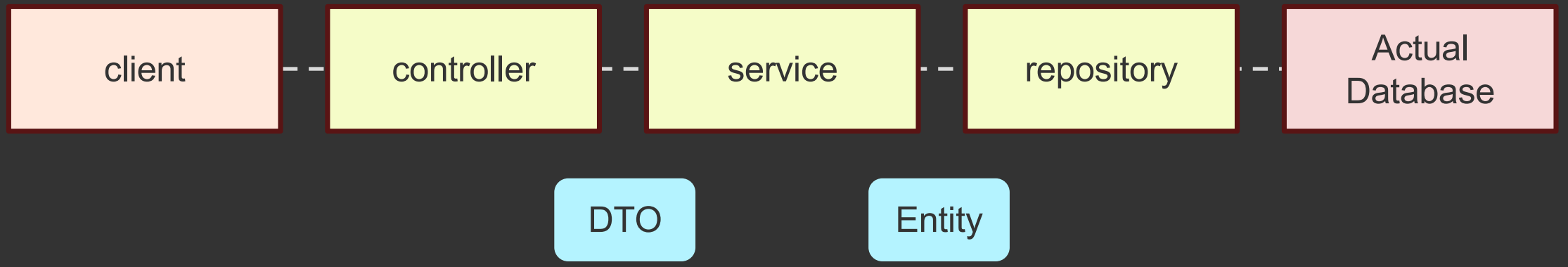
Key Points of CrudRepository :

- Generic Interface
- Predefined Methods
- Custom Queries

2.4

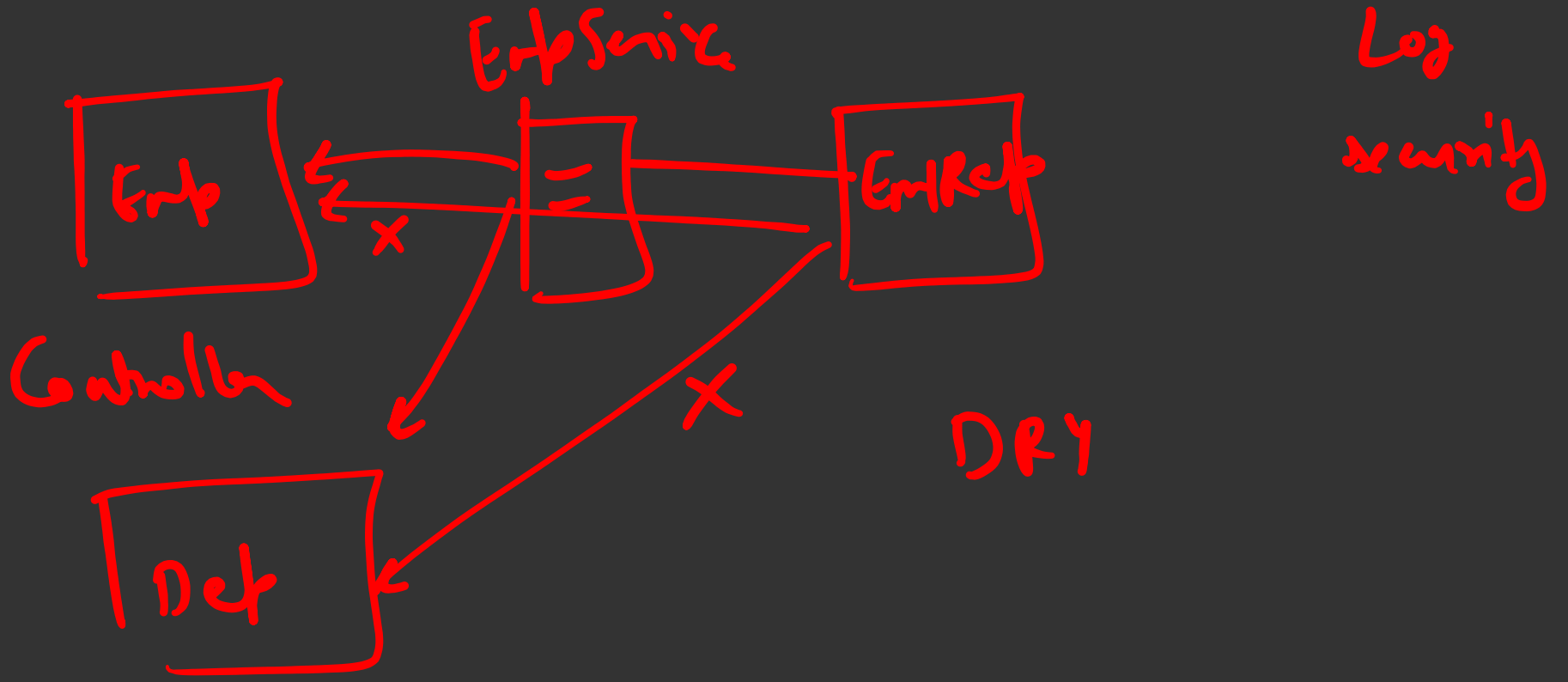
Service Layer

Spring Boot Web Project Structure



Key roles of Service Layer

- The service layer acts as a bridge between the persistence layer (responsible for data access) and the presentation layer (handling user interaction)
- It encapsulates the business logic of the application, orchestrates interactions between different components, and provides a clean interface for external clients to interact with the system.
- By abstracting away the complexities of data access and business operations, the service layer promotes modularity, maintainability, and scalability.


EmpService
Emp
EmpRepo
Controller
Dept
Log
security
DRY

2.5

PUT, PATCH and DELETE Mappings

CODING SHUTTLE

2.6

Input Validation

Annotations for Validation

S.No.	Annotation	Description
1	@NotNull	Ensures that the annotated field is not null.
2	@NotEmpty	Ensures that the annotated field is not null and its size/length is greater than zero. (For collections, arrays, and strings)
3	@NotBlank	Ensures that the annotated string is not null and its trimmed length is greater than zero.
4	@Size	Validates that the annotated element's size falls within the specified range.

Annotations for Validation

S.No.	Annotation	Description
5	@Max	Ensures that the annotated element is a number with a value no greater than the specified maximum.
6	@Email	Validates that the annotated string is a valid email address.
7	@Pattern	Validates that the annotated string matches the specified regular expression.
8	@Positive	Ensures that the annotated element is a positive number (greater than zero).
9	@PositiveOrZero	Ensures that the annotated element is a positive number or zero.

Annotations for Validation

S.No.	Annotation	Description
10	@Negative	Ensures that the annotated element is a negative number (less than zero).
11	@NegativeOrZero	Ensures that the annotated element is a negative number or zero.
12	@Past	Ensures that the annotated date or calendar value is in the past.
13	@PastOrPresent	Ensures that the annotated date or calendar value is in the past or present.
14	@Future	Ensures that the annotated date or calendar value is in the future.

Annotations for Validation

S.No.	Annotation	Description
15	@FutureOrPresent	Ensures that the annotated date or calendar value is in the future or present.
16	@Digits	Ensures that the annotated number has up to a specified number of integer and fraction digits.
17	@DecimalMin	Ensures that the annotated element is a number with a value no less than the specified minimum, allowing for decimal points.
18	@DecimalMax	Ensures that the annotated element is a number with a value no greater than the specified maximum, allowing for decimal points.

Annotations for Validation

S.No.	Annotation	Description
19	@AssertTrue	Ensures that the annotated boolean field is true.
20	@AssertFalse	Ensures that the annotated boolean field is false.
21	@Valid	Validates the associated object recursively (applies bean validation to nested objects).

Handling Validation Exceptions

- MethodArgumentNotValidException is thrown by the @Valid validation
- You can get a list of all the errors from the bindingResult of this exception.
- You can use this list to return some useful error messages as the API response.

CODING SHUTTLE

2.7

Exception Handling In Spring Boot MVC

Benefits of Exception Handling

- Prevent application crashes.
- Provide user-friendly error responses.
- Facilitate debugging and maintenance.
- Ensure consistent error handling across the application.

Handling Exceptions

- Use @ExceptionHandler to handle specific exceptions in controllers.
- Use @RestControllerAdvice for global exception handling.
- Return appropriate HTTP status codes and error messages.
- Use Custom error response class to provide structured error details.

2.8

Transforming API Response

Transforming API Response

- Extend your class with @ResponseBodyAdvice<Object> to define the custom return type.
- Use @RestControllerAdvice for global API Response transformation.
- Return appropriate HTTP status codes and error messages.
- You can also return the timestamp of the API response.

Spring Boot Web and MVC

Homework

Recap

1. Spring Boot Web and MVC Architecture
2. Presentation Layer
3. Persistence Layer
4. Service Layer
5. Exception Handling in Spring Boot MVC
6. Input Validation Annotations

Homework

1. Make a list of all the annotations we have seen so far in the Spring Web Framework.
2. Create the following REST endpoints for the following Entity

Department
- id
- title
- isActive
- createdAt

REST APIs:

GET: /departments
POST: /departments
PUT: /departments
DELETE: /departments
GET: /departments/{id}

Homework

3. Write Exception Handling logic for the Department Entity.

4. Add the appropriate fields in the Department and Employee Entities so that the following validations can be added:
@Null, @NotNull, @AssertTrue, @AssertFalse, @Min, @Max, @DecimalMin, @DecimalMax, @Negative, @NegativeOrZero, @Positive, @PositiveOrZero, @Size, @Digits, @Past, @PastOrPresent, @Future, @FutureOrPresent, @Pattern, @Email, @NotEmpty, @NotBlank, @Length, @Range, @CreditCardNumber, @URL

Homework

5. Create custom Annotation to handle a Prime number input
6. Create custom Annotation to check if the Password String field satisfies the following criteria:
    a. contains one uppercase letter
    b. contains one lowercase letter
    c. contains one special character
    d. minimum length is 10 characters