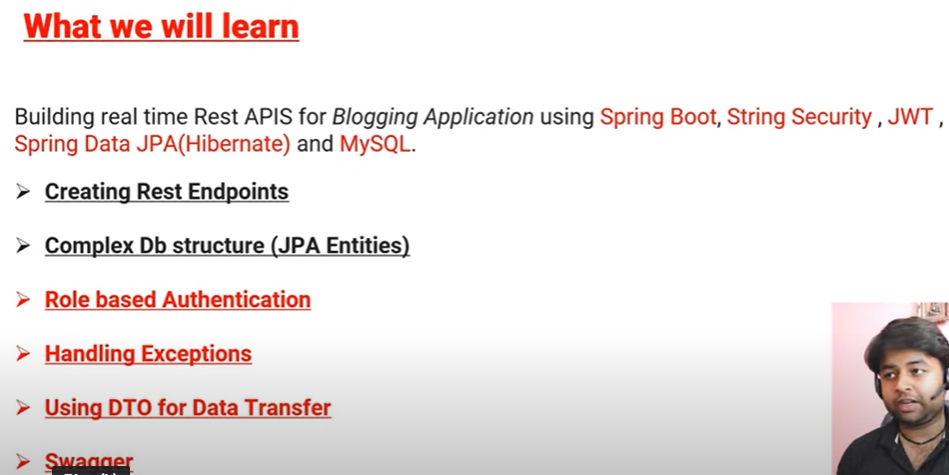
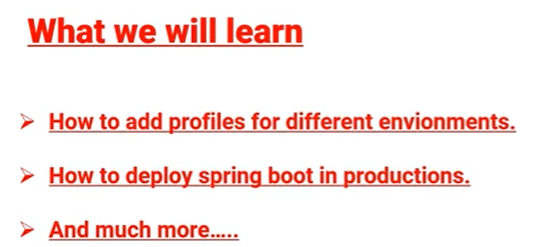
**BLOG APPLICATION SPRING BOOT**

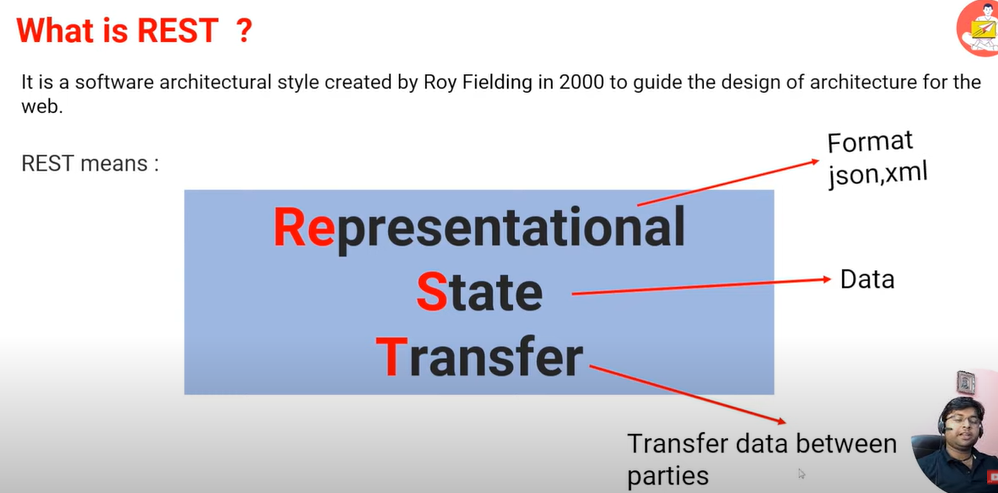
****

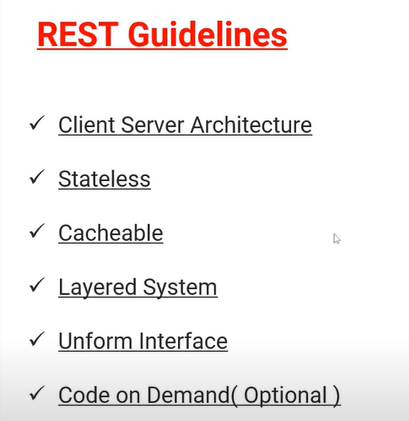
****

**REST Architecture | What is REST ? | REST Constraints in Detail**

****Client - Consumer

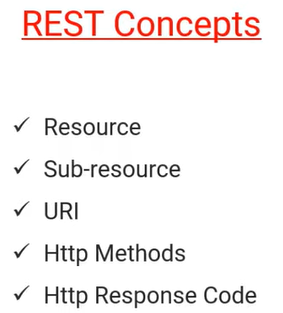
Server – Provider

****

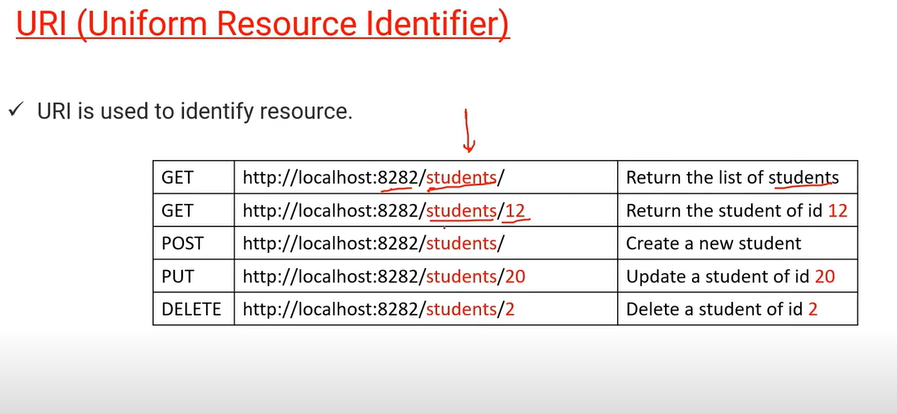
****

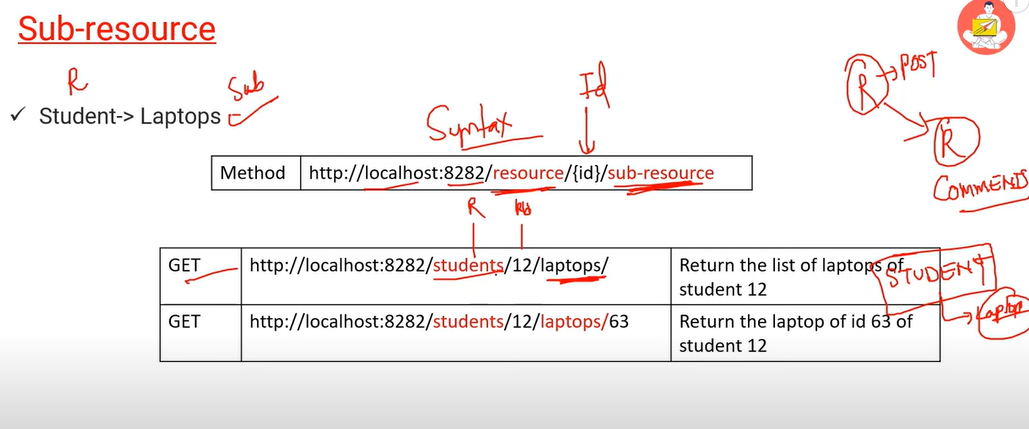
* **Stateless** - Don't manage your state in the server.
* **Cacheable** - It can improve your App performance for example user, again and again, request the same data then you can save that data in the cache for improve the performance of your app.
* **Layered** **System**: A layered system means three-tier or two-tier architecture. With the help of layered architecture, you can improve your app scalability and load-balancing.
* **Uniform** **interface**: Jitnay bhi interaction b/w client server sb unified honay chaiye takay decouple kar sakay architecture ko.

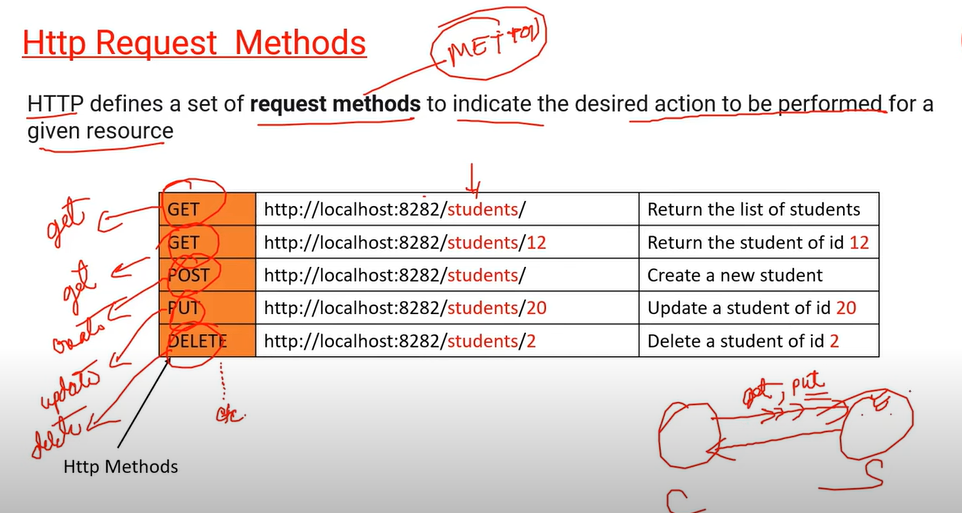
# REST Key Concepts ?

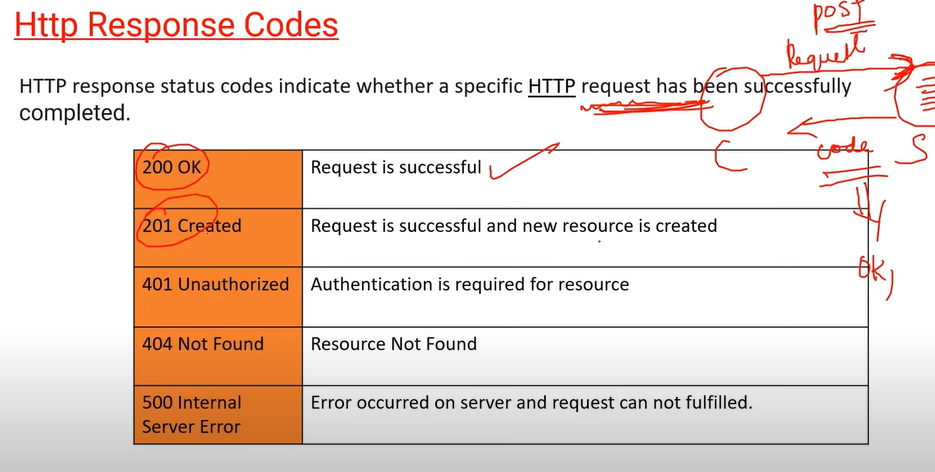




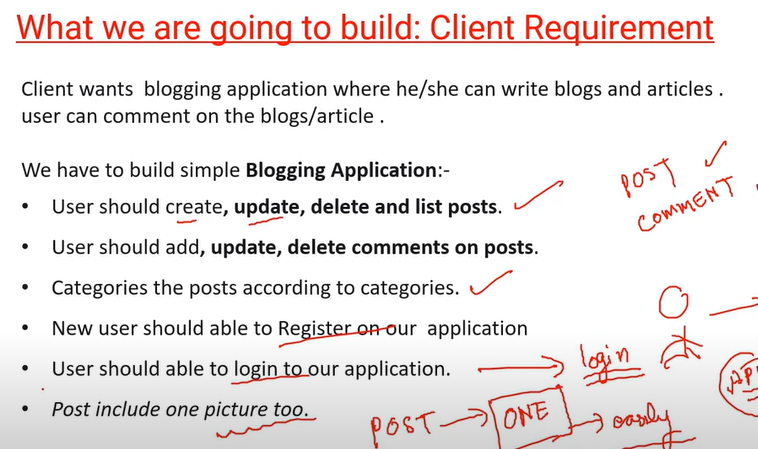


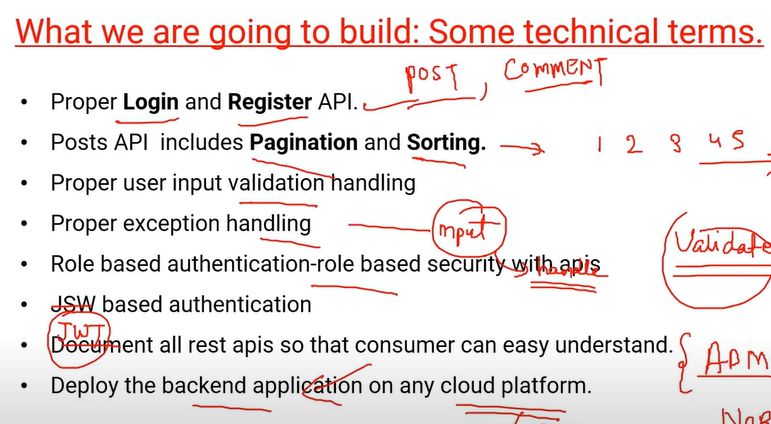




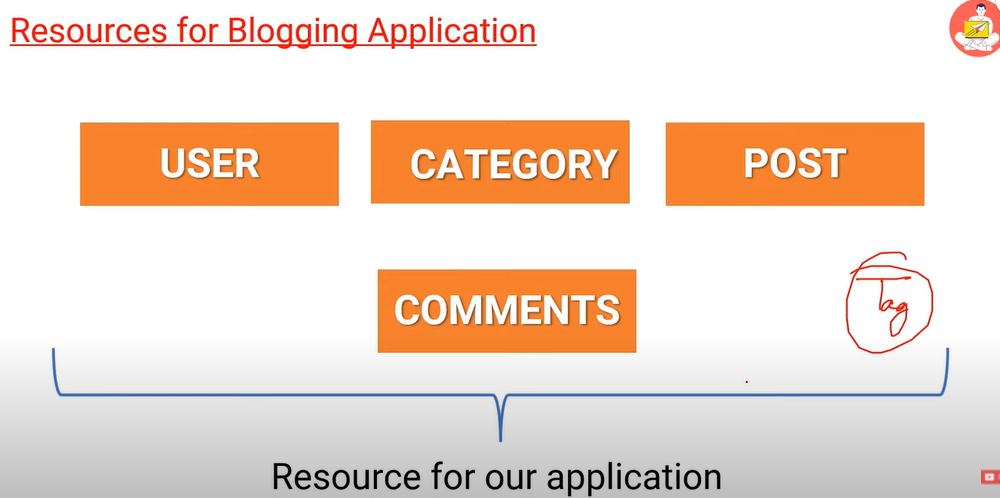


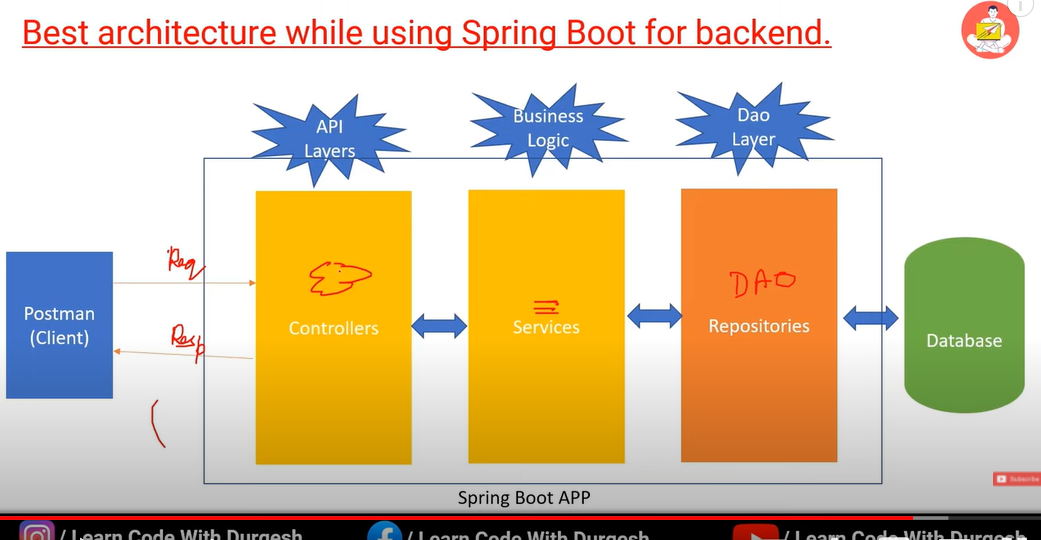
# Client Requirements for Project



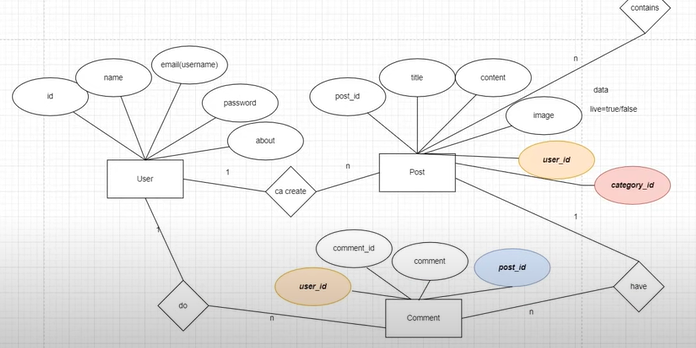


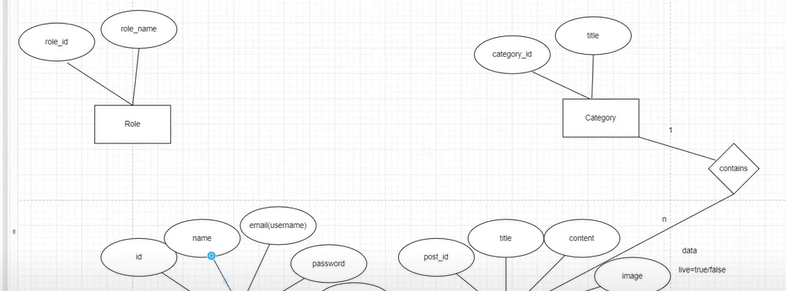
# Finding Resource for Project





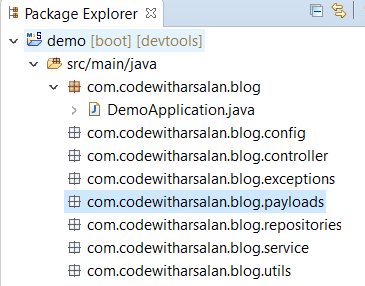
# How Database Structure look like for Blogging Project





# Project Setup from Scratch | Database setup full video

First I will create all the important packages for our project



**Utils Package:** In this package we will add helper classes like Date helper, Image file uploader helper or App constants etc.

After creating schema in MySQL workbench we will do database configuration in application.properties file.

You can use ctrl+shift+t for search

**#db configuration**

**spring.datasource.url=jdbc:mysql://localhost:3306/blog\_app\_apis**

**spring.datasource.username=root**

**spring.datasource.password=root**

**spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver**

**spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect**

**#create, update, create-drop, validate**

**spring.jpa.hibernate.ddl-auto=update**

if you forget your mysql port number:

open mysql terminal > type **\s** then enter

# Start Creating User API | Installing Lombok

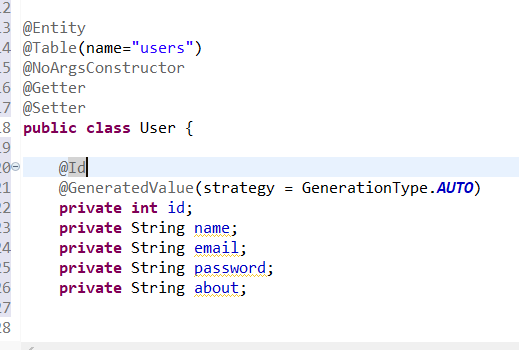
Using @Entity annotation to create this class is entity

Using @Table entity you can change the name of table in database.

Using @Column entity you can change the name of column in database.



You can use Lombok for getter and setter



After creating user entity we will create repository class with the name of UserRepo and change this class into interface and then extend this interface with JpaRepository.

In JpaRepository we have multiple operations. You can use that operation to implement database.

Hum entity class ko direct service main nahi dengay hum DTO use karengay us k liye hum payload package mai userDto k name sai class banakar same entity waali cheezain add kardengay.

**What is the use of DTO instead of entity?**

Short answer: Entities may be part of a business domain. Thus, they can implement behavior and be applied to different use cases within the domain. DTOs are used only **to transfer data from one process or context to another**.

DTO is an abbreviation for **Data Transfer Object**, so it is used to transfer the data between classes and modules of your application.

* DTO should only contain private fields for your data, getters, setters, and constructors.
* DTO is not recommended to add business logic methods to such classes, but it is OK to add some util methods.

DAO is an abbreviation for **Data Access Object**, so it should encapsulate the logic for retrieving, saving and updating data in your data storage (a database, a file-system, whatever).

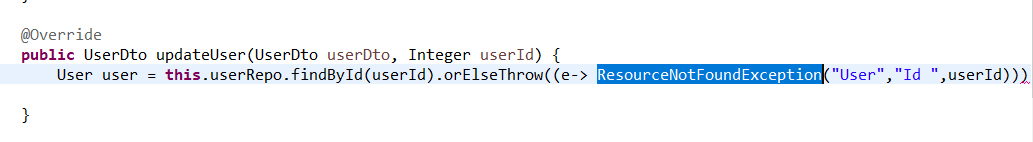
Lombok install karna paray ga for getter and setter usko phr sts walay folder mai paste kar k cmd open karengay.

OPEN CMD > Type java -jar lombok.jar > Lombok open > Click on specify location > select sts.exe file > install > quit installer > again open sts.

Now create Service impl package and add class of UserService impl and add user class automatically it can implement all the methods.

Now in this implement class we will autowired userRepo class.

Now we have UserDto for passing data but we need user entity to store data in database. We will user model Mapper for this conversion but for now we will create method for userDto to User conversion.



In updateUser Api we will implement exceptional handling. For exceptional handling we will create ResourceNotFoundException handling class in exceptions package and we will extends this class with RuntimeException.

# How We Autowired Interface ? | Interview Question

For ex jab humne repository ka interface banaya or usko JpaRepository sai extend kardia or uski koe implementation class banaye hi nahi tw usko Autowired kaise karsaktay dosri class mai

Reflection Api ki madad sai hum runtime par bannay waali class ko dekh saktay

<https://www.javatpoint.com/java-reflection#:~:text=Java%20Reflection%20is%20a%20process,time%20behavior%20of%20a%20class>.

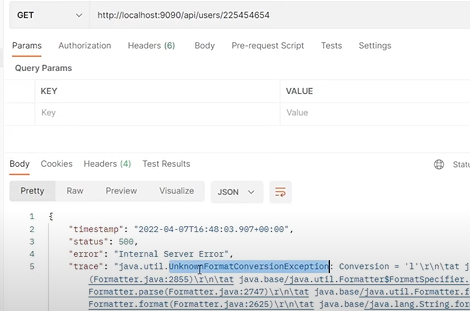
Hamesha autowired kartay hoa class ka hi object ayega tw jab spring boot ki application bootstrap horahi hoti hai tw us waqt repository walay interface ko dynamically runtime par class di jaati hai or un classes ko proxy class kaha jaata hai.



Is cheez ko hum pathUri Variable kehtay hai spring boot mai

# How to handle Request with user id not found | Handling Exception in Spring Boot

If I enter different id which is not exist in our database then we will get unknownFormaterrorexcepion



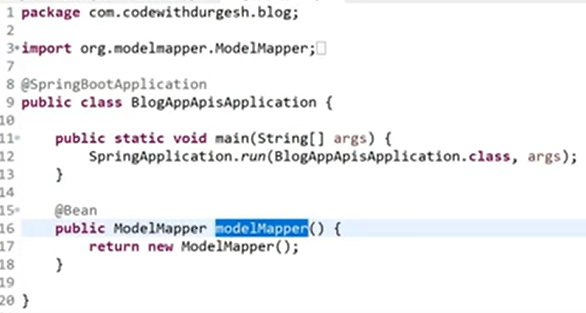
We will create one class with the name of Global exception handler

And then we will annotate the class with **RestControllerAdvice** if we are creating Rest Api otherwise we will use **ControllerAdvice.**

# How to use ModelMapper to map our models in Spring boot

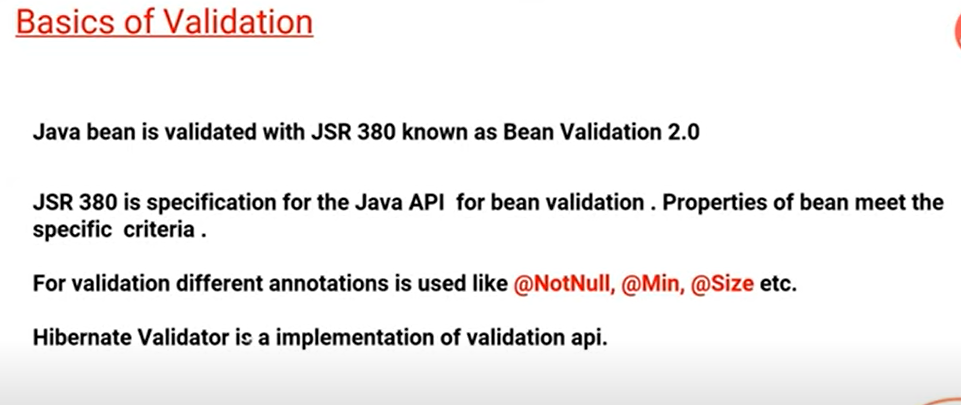
ModelMapper, is an object-to-object framework that converts Java Beans (Pojos) from one representation to another. It automates different object mappings with a "convention follows configuration" approach allowing at the same time advanced functionality for cases with special needs

We need to create model mapper method tw yai hum ya tw new class bean banayengay ya main class mai bhi karsaktay means jaha @SpringBootApplication ki annotation q k wo bhi 1 bean hai q k wo bhi 1 configuration class hai jaha configuration ki annotation hai usai bean keh saktay.

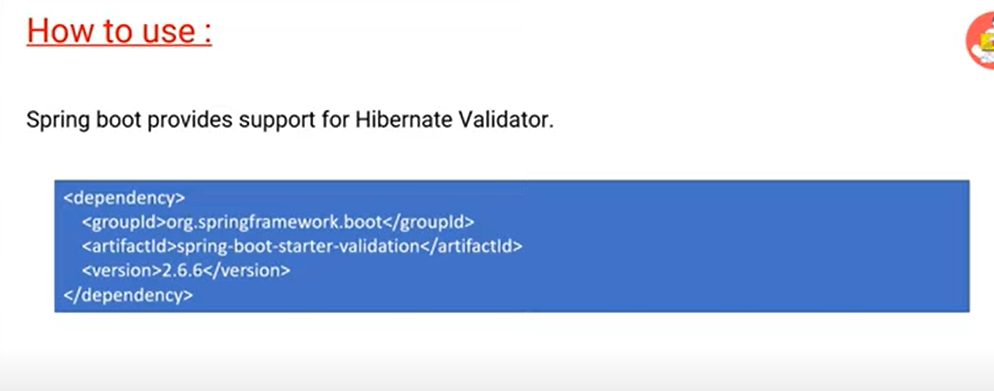


Ab yai har jaga autowired karsaktay

# Simplest way of Validating Data using Bean Validator in Spring Boot







Ctrl+shift+f sai format karsaktay pom.xml file ko

Simple 3 steps hai validation k:

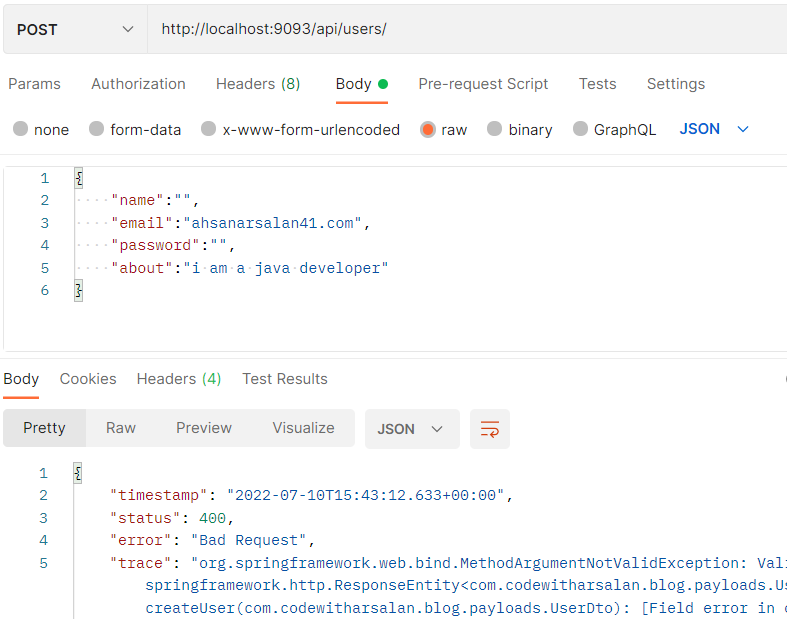
Pom.xml file mai dependency add karni hai

Entity ya dto mai field k uppar annotation use karni hai for example @NotNull.

Controller mai @valid use karna hai

Ab humay jo error msg araha usko customize karna hai

After adding validation we will get that type of error when we want to add new data.



So now we need to create MethodArgumentNotValidationException method in my GlobalExceptionHandler Class.

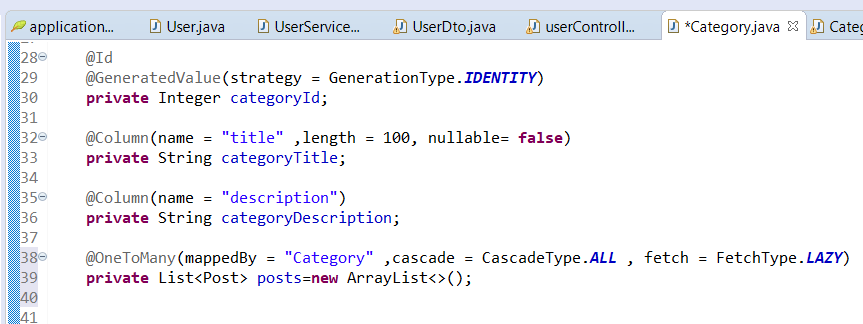
# Creating Complete Category API

Ctrl+2 is the shortcut key to assign a value

# Creating Post and Relationship in one go

Post ki entity mai relationship banay ga is liye us mai User or Category dono add hongi.

Phr category ki entity mai list banayengay q k 1 category ki multiple post hosakti



Cascading is a feature in Hibernate, which is **used to manage the state of the mapped entity whenever the state of its relationship owner (superclass) affected**. When the relationship owner (superclass) is saved/ deleted, then the mapped entity associated with it should also be saved/ deleted automatically

What is FetchType lazy in Hibernate?

The FetchType. LAZY **tells Hibernate to only fetch the related entities from the database when you use the relationship**. This is a good idea in general because there's no reason to select entities you don't need for your uses case. You can see an example of a lazily fetched relationship in the following code snippets.

What is the difference between FetchType eager and FetchType lazy?

Pros and Cons of these two fetch types. Lazy initialization improves performance by avoiding unnecessary computation and reduce memory requirements. Eager initialization takes more memory consumption and processing speed is slow. Having said that, depends on the situation either one of these initialization can be used.

Jab hum bidirectional mapping karengay to 1 agar one to many hai tw dosra manytoone hoga

# Creating POST API in simple ways with having 2 Parent Resource

If you need users all post then you need to create method in PostRepo Interface

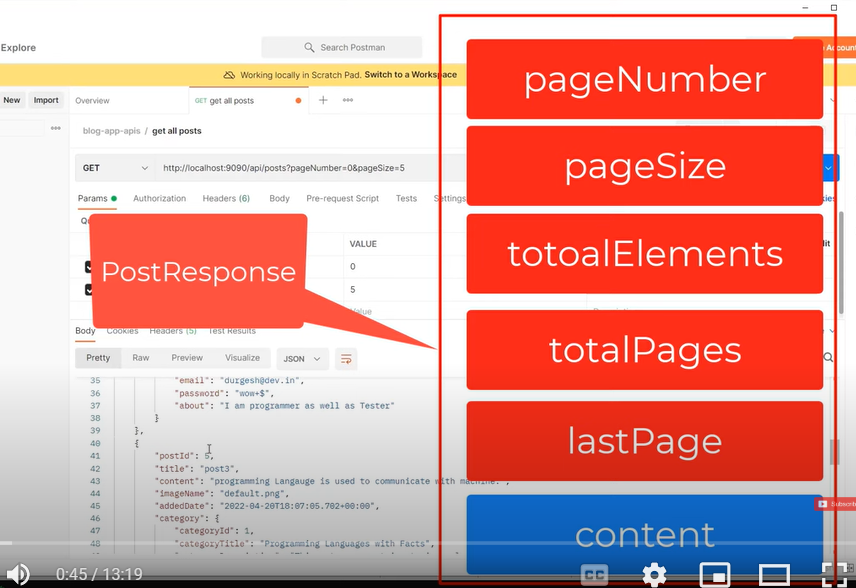
# Implement Pagination in very simple ways in AP



Springboot mai JPA ki madad sai pagination and sorting usi k method ko use kar k kar saktay

# Modifying Post Response in POST API | Its very important for creating API

Abhi jo hum pagination implement kar rahay tw us mai yai sb karne k liye humay response ko modify karna hoga



Post response k naam sa 1 class banakar us kuch field define kar k getter setter banayengay

Phr usko change kardengay jaha get all post mai List<postDto> hoga waha PageRespone likh dengay us k baad 1 PostResponse ka constructor banakar value set kardengay.

Yahi kaam getby post and getbycategory par bhi karna hai.

# Implementing Sorting in Blogging Application in Spring Boot

Controller mai requestparam add kardengay phr wohi field getallpost ki declaration and implementation mai add karengay

Us k baad pagerequest mai Sort.by use kar k us mai apni field add kardengay

# Implementing Searching in our Backend Application

1 method banayengay PostRepo mai phr usko implement kardengay

Abhi kuch search kartay hoa 1 error araha because of hibernate version issue is liye hum query use karengay.

# Securing Rest APIs in Backend Application

When we add spring security rest api it will secure all the url of our website.