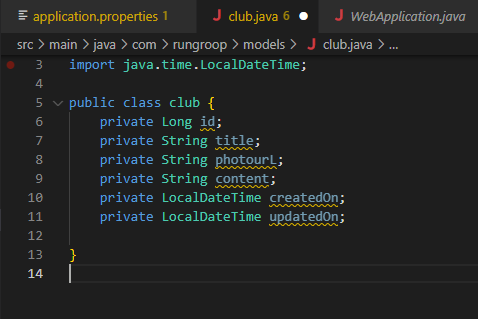
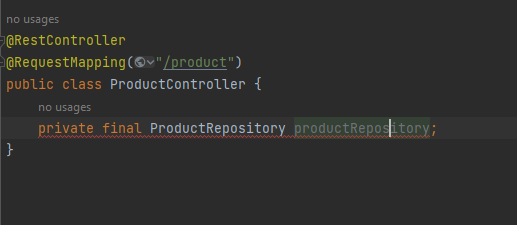
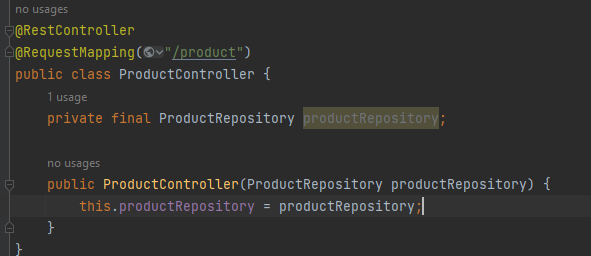
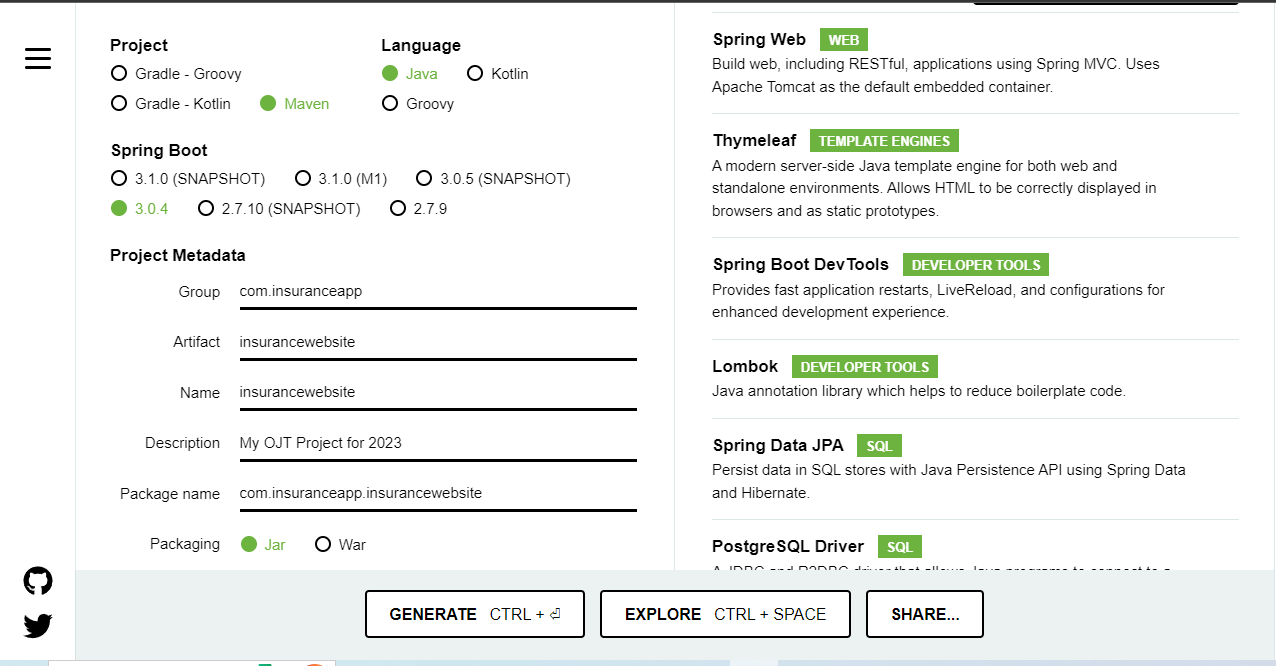
Crating model in spring boot

1. Create folders name it Models inside main and java project.
2. Create file with extension of java
3. Input the all field in code
4. 
5. To view another command pres alt+enter





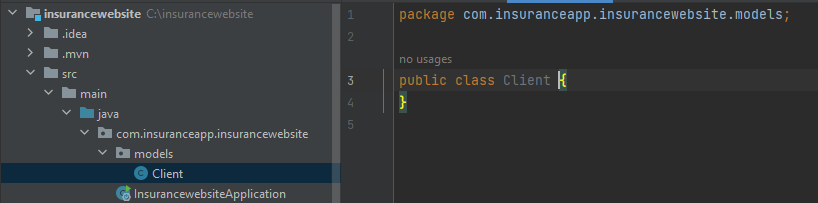
Start project insurance app



1. Create databae connection in applictaion.properties under src folder.

spring.datasource.url=jdbc:postgresql://localhost:5432/insurance\_claim\_app   
spring.datasource.username=postgres  
spring.datasource.password=1234  
  
spring.datasource.hikari.connectionTimeout=20000  
spring.datasource.hikari.maximum-pool-size=5  
  
spring.datasource.driver-class-name=org.postgresql.Driver  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation=true  
  
spring.jpa.hibernate.ddl-auto=update  
# spring.jpa.hibernate.ddl-auto=create  
  
  
spring.jpa.show-sql=true

1. Creating model creating fiel one by one in java class same as table.
2. Create package name is models and inside of this package create model or a java class name client.



1. Creating field of the table and run the project and they automatically create ah table.

package com.insuranceapp.insurancewebsite.models;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.hibernate.annotations.CreationTimestamp;  
import org.hibernate.annotations.UpdateTimestamp;  
  
import java.time.LocalDateTime;  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
@Entity  
@Table(name = "clients")  
public class Client {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String title;  
 private String insurance\_category;  
 private String dependent;  
  
 private String beneficiary;  
  
 @CreationTimestamp  
 private LocalDateTime createdOn;  
 @UpdateTimestamp  
 private LocalDateTime updatedOn;  
}

1. Creating repository-itong yung malapit sa database or model itong ay para makapagcreate bg CRUD method para makuha natin ang specific table galing sayong database para maturn into object so they can travel papuntang services at pauntang controller and display on your website.

Dito pwd ka din gumawa na sarili mong method at ilagay ito sa DTO at automatically create automatically create code for you.

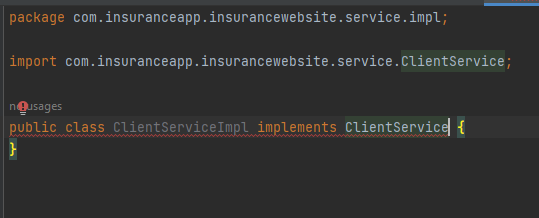
1. Create package name repository and crete java interface name ClientRepository

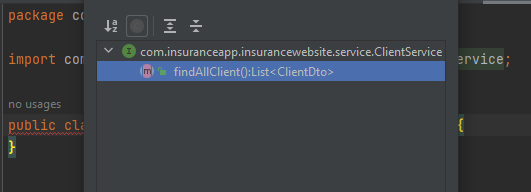
package com.insuranceapp.insurancewebsite.repository;  
  
import com.insuranceapp.insurancewebsite.models.Client;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.Optional;  
  
public interface ClientRepository extends JpaRepository<Client, Long> {  
 Optional<Client> findByTitle(String url);  
}

1. Creating DTO para ito lahat ng information para hindi lahat maipakita sa mga user or sa admin. Example meron tayong field sa model na password dito sa dto ay pws nating hindi isama un.para ireturn specific field into user. Create package name DTO in create java class name ClientDto.

package com.insuranceapp.insurancewebsite.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
import java.time.LocalDateTime;  
@Data  
@Builder  
public class ClientDto {  
 private Long id;  
 private String name;  
 private String insurance\_category;  
 private String dependent;  
 private String beneficiary;  
 private LocalDateTime createdOn;  
 private LocalDateTime updatedOn;  
}

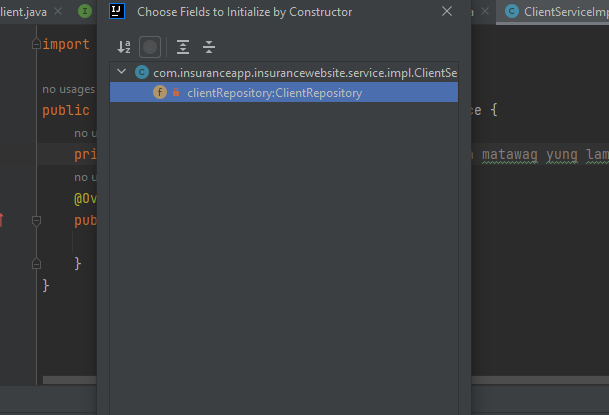
1. Creating service repository in regular repository you have regular CRUD method, dito sa services nandito yung layer of abstraction meaning gagamit ka dito in services that look specific what type of data that you want need at ito ay binubuo ng repository method.
2. Create package bame services and java interface name clientservices and create simple method name.
3. Create impl package inside package service and ClientServiceImpl

click the red bulb

implement the method.

Pagkatapos kailangan I bring natin yung repository dahil dahil ito ay gagamitin ng service para sa abstraction.

Para makapag create ng constructor ay right clivk the repo



Dockerizing a spring boot app

1. Click maven and fing mvc package.

Link for

0:41 / 5:10

# Hide or Show a div using HTML Select Tag & JS

1. <https://www.youtube.com/watch?v=SV-6PSOC97I>

# JavaScript Confirmation box | How to Delete Record from database | Employee management system

1. <https://www.youtube.com/watch?v=fAhrlsKnzUo>

### [https://www.javaguides.net/2018/09/spring-boot-spring-mvc-role-based-spring-security-jpa-thymeleaf-mysql-tutorial.html](https://www.javaguides.net/2018/09/spring-boot-spring-mvc-role-based-spring-security-jpa-thymeleaf-mysql-tutorial.html-for) -- for basis **Spring Boot + Spring MVC + Role Based Spring Security + JPA + Thymeleaf + MySQL Tutorial**

1. [https://www.javaguides.net/2018/10/user-registration-module-using-springboot-springmvc-springsecurity-hibernate5-thymeleaf-mysql.html](https://www.javaguides.net/2018/10/user-registration-module-using-springboot-springmvc-springsecurity-hibernate5-thymeleaf-mysql.html-)  -- original recommended

# <https://www.youtube.com/watch?v=xibt9JG1i54&t=23s> **Handling Date and Time in Spring Boot with Thymeleaf**

# **Image Upload Spring boot MVC , Thymeleaf | File Upload using Spring boot tutorials**

1. <https://www.youtube.com/watch?v=oCtxqzD4PhY>

# **Spring Boot Update User Details and Refresh Username in Menu**

<https://www.youtube.com/watch?v=aFx2SfG5fmY&t=19s>

@PostMapping("/register/save")  
public String registration(@Valid @ModelAttribute("user") UserDto userDto,  
 @RequestParam("avatar")MultipartFile multipartFile,  
 BindingResult result,  
 Model model) throws IOException {  
 User existingUser = userService.findUserByEmail(userDto.getEmail());  
  
 if(existingUser != null && existingUser.getEmail() != null && !existingUser.getEmail().isEmpty()){  
 result.rejectValue("email", null,  
 "There is already an account registered with the same email");  
 }  
  
  
 if (!multipartFile.isEmpty()){  
 String fileName = StringUtils.*cleanPath*(multipartFile.getOriginalFilename());  
 userDto.setAvatar(fileName);  
 UserDto registration = userService.saveUser(userDto);  
 String upload = "images/"+ userDto.getId();  
  
 FileUploadUtil.*savefile*(upload,fileName, multipartFile);  
 }else {  
 if (userDto.getAvatar().isEmpty()){  
 userDto.setAvatar(null);  
 userService.saveUser(userDto);  
 }  
 }  
 if(result.hasErrors()){  
 model.addAttribute("user", userDto);  
 return "/register";  
 }  
  
 userService.saveUser(userDto);  
 return "redirect:/register?success";  
}

# **Spring Boot File Upload Tutorial (Upload and Display Images)**

1. <https://www.codejava.net/frameworks/spring-boot/spring-boot-file-upload-tutorial#:~:text=Spring%20Boot%20File%20Upload%20Tutorial%20%28Upload%20and%20Display,...%204%204.%20Display%20uploaded%20images%20in%20browser>