LAB 07: INTROCTION TO SPRING BOOT

Goal:

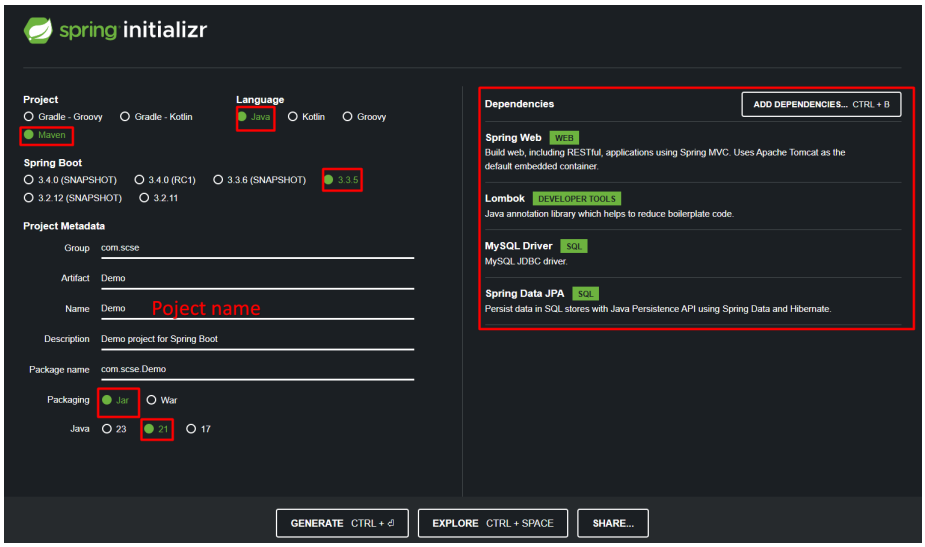
- **Setup Spring Boot project**
- **Undertand Spring Boot architecture**
- **Create a simple register API**
- **Using Postman to test application**
- **Exericse:**
  - **Create CRUD API aplication**

Table of content

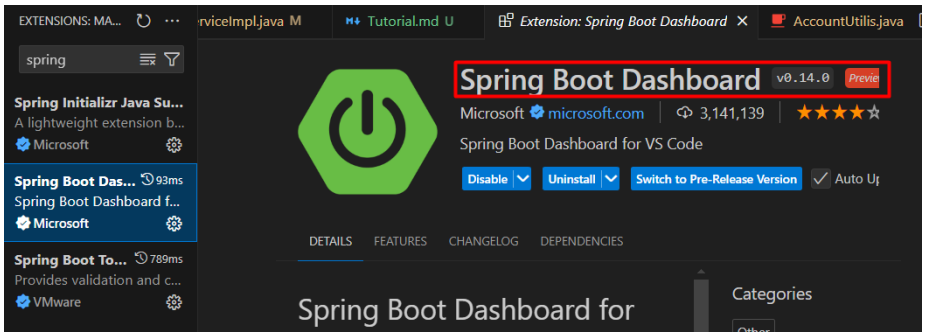
Setup project

Create Spring Boot project

1. go to https://start.spring.io/



2. Install springboot extension for VSCode



3. Install postman extension

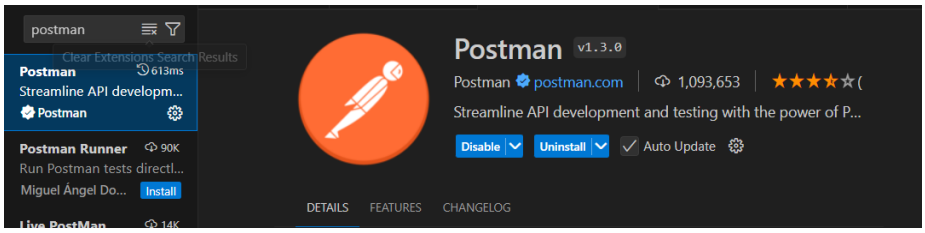


Figure 1: postman-vscode

Create DB using mySQL

create a BD name as `lab07` by using MySQL server

Architecture

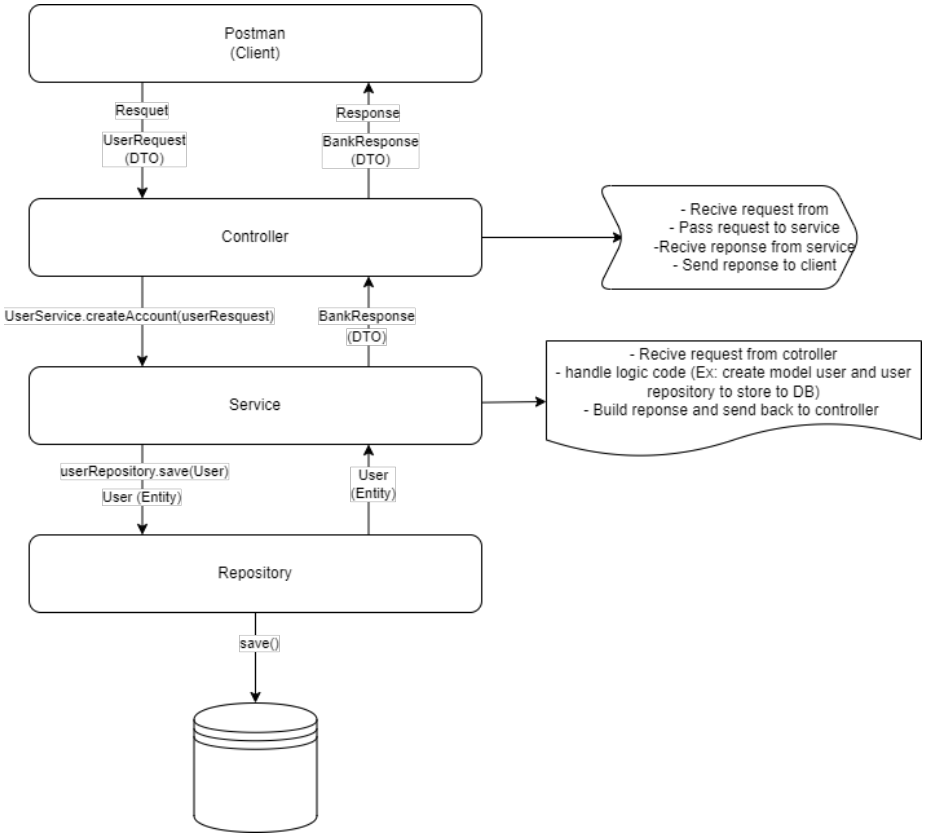


Figure 2: architecture

Project structure and database

- Create a database with the name you like
- Create project structure as follow
- You will create many file in this lab, when you complete this lab your project look like this

Start to code

NOTE: To make the code clear, the import part will not show in some java files

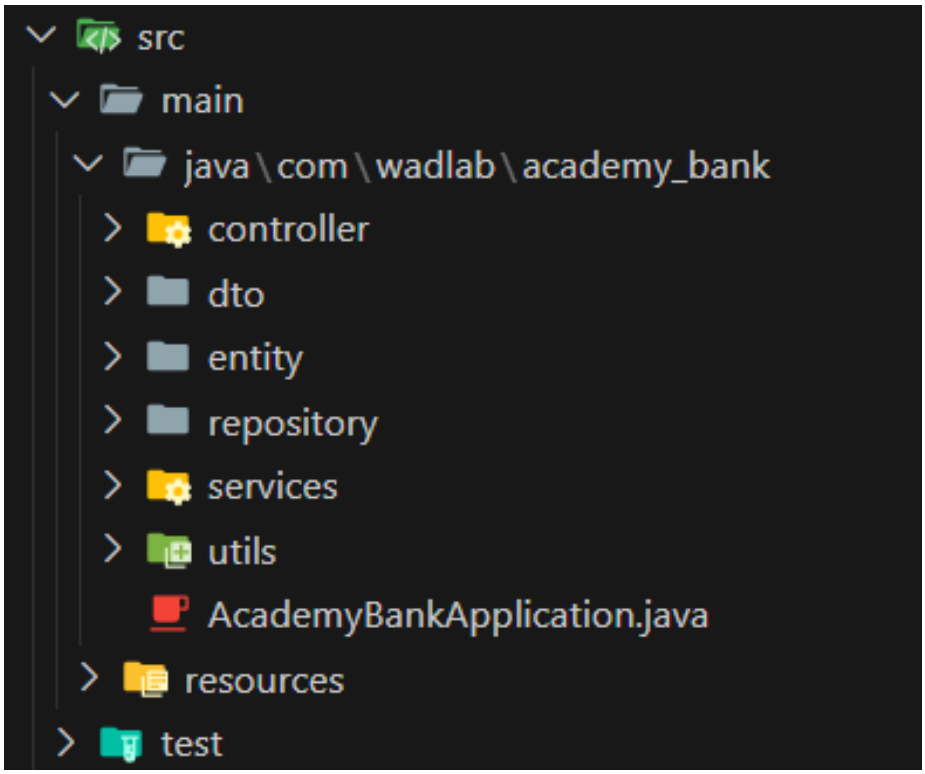


Figure 3: project-structure

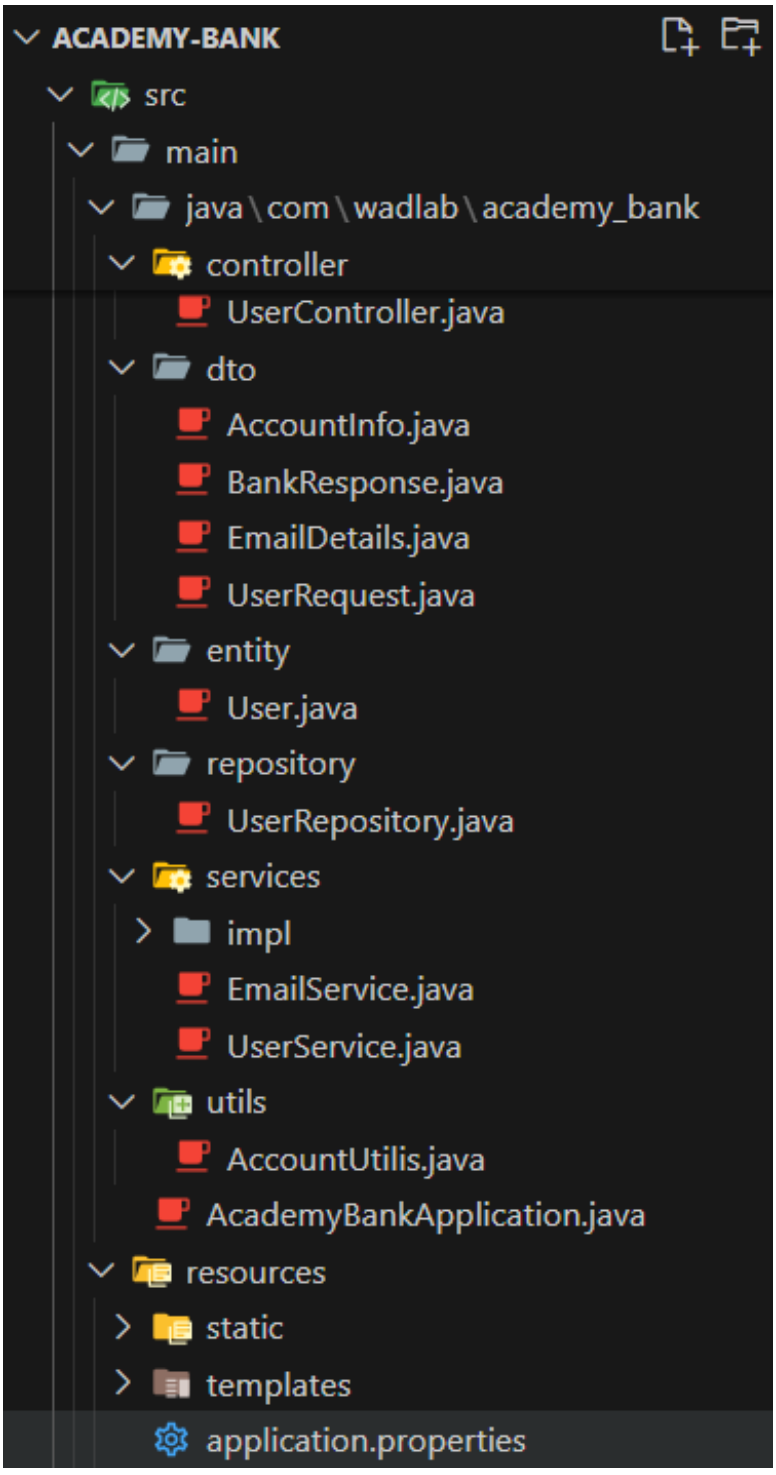


Figure 4: project-structure-completed

Config DB resourses/application.properties

```
spring.application.name=<your application name>

spring.datasource.url=jdbc:mysql://localhost:3306/<your DB Name>
spring.datasource.username=<your DB user name>
spring.datasource.password=<your DB password>
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
```

Create entity/User.java

```
package com.wadlab.academy_bank.entity;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import jakarta.persistence.*;
import lombok.*;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
    private String otherName;
    private String gender;
    private String address;
    private String sateOfOrigin;
    private String accountNumber;
    private BigDecimal accountBalance;
    private String email;
    private String phoneNumber;
    private String alternativePhoneNumber;
```

```
    private String status;

    @CreationTimestamp
    private LocalDateTime createAt;
    @UpdateTimestamp
    private LocalDateTime modifiedAt;
}
```

Create `repository/UserRepository.java`

```
import org.springframework.data.jpa.repository.JpaRepository;
import com.wadlab.academy_bank.entity.User;

public interface UserRepository extends JpaRepository<User, Long>{
    Boolean existsByEmail(String email);
}
```

Create `dto/UserRequest.java`

```
package com.wadlab.academy_bank.dto;

import lombok.*;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class UserRequest {
    private String firstName;
    private String lastName;
    private String otherName;
    private String gender;
    private String address;
    private String sateOfOrigin;
    private String accountNumber;
    private String email;
    private String phoneNumber;
    private String alternativePhoneNumber;
    private String status;
}
```

Create `dto/BankReponse.java`

```
import lombok.*;

@Data
```

```java
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class BankResponse {
    private String responseCode;
    private String responseMessage;
    private AccountInfo AccountInfo;
}
```

Create services/UserService.java

```java
package com.wadlab.academy_bank.services;

import com.wadlab.academy_bank.dto.BankResponse;
import com.wadlab.academy_bank.dto.UserRequest;

public interface UserService {
    public BankResponse createAccount(UserRequest userRequest);
}
```

Create services/imp/UserServiceImpl.java

```java
package com.wadlab.academy_bank.services.impl;

import com.wadlab.academy_bank.dto.*;

import com.wadlab.academy_bank.entity.User;
import com.wadlab.academy_bank.repository.UserRepository;

import com.wadlab.academy_bank.services.EmailService;
import com.wadlab.academy_bank.services.UserService;
import com.wadlab.academy_bank.utils.AccountUtilis;

import java.math.BigDecimal;
import java.math.BigInteger;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    UserRepository userRepository;
    EmailService emailService;
```

```java
@Override
public BankResponse createAccount(UserRequest userRequest) {

    // Check whether email exsit in database
    if(userRepository.existsByEmail(userRequest.getEmail())) {
        // if exsit return BankResponse with response
        return  BankResponse.builder()
                .responseCode(AccountUtilis.ACCOUNT_EXISTS_CODE)
                .responseMessage(AccountUtilis.ACCOUNT_EXISTS_MESSAGE)
                .AccountInfo(null)
                .build();
    }

    // Using builder to create new User you call use contructor instead
    // User newUser = new User("Nghia", "Nguyen", "Trung", "123 Linh Trung"...)
    User newUser = User.builder()
                .firstName(userRequest.getFirstName())
                .lastName(userRequest.getLastName())
                .otherName(userRequest.getOtherName())
                .address(userRequest.getAddress())
                .sateOfOrigin(userRequest.getSateOfOrigin())
                .accountNumber(AccountUtilis.generateAccountNumber())
                .accountBalance(BigDecimal.ZERO)
                .phoneNumber(userRequest.getPhoneNumber())
                .alternativePhoneNumber(userRequest.getAlternativePhoneNumber())
                .email(userRequest.getEmail())
                .status("ACTIVE")
                .build();

    // Save user to database by repository
    User savedUser = userRepository.save(newUser);

    // Create account Info
    AccountInfo accountInfo = AccountInfo.builder()
                .accountBalance(savedUser.getAccountBalance())
                .accountNumber(savedUser.getAccountNumber())
                .accountName(savedUser.getFirstName() + " "
                            + savedUser.getLastName() + " "
                            + savedUser.getOtherName())
                .build();

    return BankResponse.builder()
            .responseCode(AccountUtilis.ACCOUNT_CREATION_SUCCESS_CODE)
            .responseMessage(AccountUtilis.ACCOUNT_CREATION_SUCCESS_MASSAGE)
            .AccountInfo(accountInfo)
            .build();
```

```
    }
}
```

Create utils/AccountUtilis.java

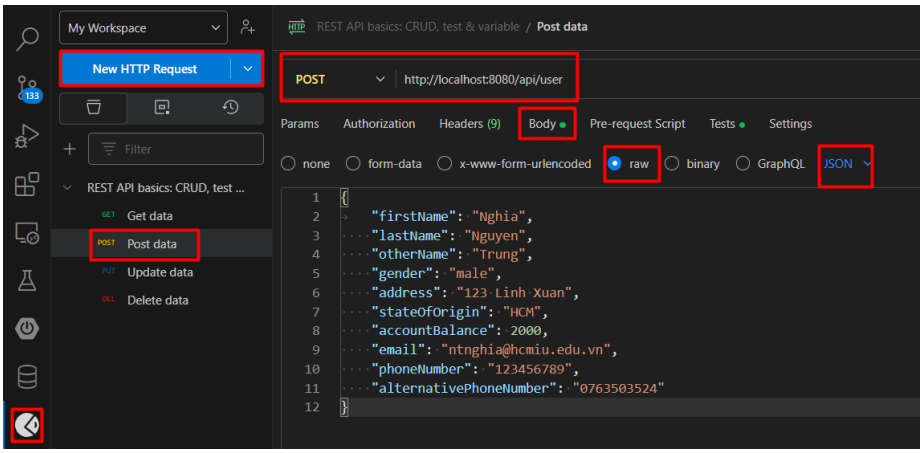
```
package com.wadlab.academy_bank.utils;
import java.time.Year;

public class AccountUtilis {

    public static final String ACCOUNT_EXISTS_CODE = "001";
    public static final String ACCOUNT_EXISTS_MESSAGE =



                                        "Account has been successfully created";
    public static final String ACCOUNT_CREATION_SUCCESS_CODE = "002";
    public static final String ACCOUNT_CREATION_SUCCESS_MASSAGE = "This user already exists


    public static String generateAccountNumber() {

        // 2023 + randonSixDigits
        Year currentYear = Year.now();

        int min = 100000;
        int max = 999999;

        int randNumber = (int) Math.floor(Math.random() * (max - min + 1));

        String year = String.valueOf(currentYear);
        String randomNumber = String.valueOf(randNumber);
        StringBuilder accountNumber = new StringBuilder();

        return accountNumber
                .append(year)
                .append(randomNumber)
                .toString();
    }
}
```

Creart controler/UsersController.java

```
package com.wadlab.academy_bank.controller;

import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.wadlab.academy_bank.dto.BankResponse;
import com.wadlab.academy_bank.services.UserService;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.GetMapping;


@RestController
@RequestMapping("/api/user")
public class UserController {

    @Autowired
    UserService userService;

    @PostMapping
    public BankResponse createAccount(@RequestBody UserRequest userRequest) {
        return userService.createAccount(userRequest);
    }
}
```

Test your API by Postman



Description:
- Using postman to create POST request - Api endpoint POST `localhost:8080/api/user`
- Using JSON fomat to send request - Input the request body:

```
{
    "firstName": "Nghia",
```

```
    "lastName": "Nguyen",
    "otherName": "Trung",
    "gender": "male",
    "address": "123 Linh Xuan",
    "stateOfOrigin": "HCM",
    "accountBalance": 2000,
    "email": "ntnghia@hcmiu.edu.vn",
    "phoneNumber": "123456789",
    "alternativePhoneNumber": "0763503524"
}
```

Result

Sourse code: https://github.com/ntnghia1908/springsboot-academy-bank

Exericse: Create CURD API

Using the database in this lab and follow the project sturcture:

- Create API endpoint GET `/api/users` with that response all users id and name.
- Create API endpoint GET `/api/user/{id}` which get detail of a specific user.
- Create API endpoint PUT `/api/users{id}` which update a specific user
- Create API endpoint DELETE `/api/user/{id}` which delete specific user
- Create API endpoint PUT `/api/user/{id}` which update specific user

hint: search for keyword: curd rest api springboot

Reference

1. https://www.youtube.com/watch?v=OASFA6p0l_Q&list=PLD72JnLc4hpviJusvYgJJBupxRpflOAKc&in
2. https://www.youtube.com/playlist?list=PLD72JnLc4hpviJusvYgJJBupxRpflOAKc
3. https://www.youtube.com/watch?v=H2gquNz1bvs&list=PL2xsxmVse9IaxzE8Mght4CFltGOqcG6FC