**MEMBUAT PROJECT RESTFULL API DENGAN SPRING BOOT**

**By : Abdul Wahid**

* Aplikasi yang dibutuhkan



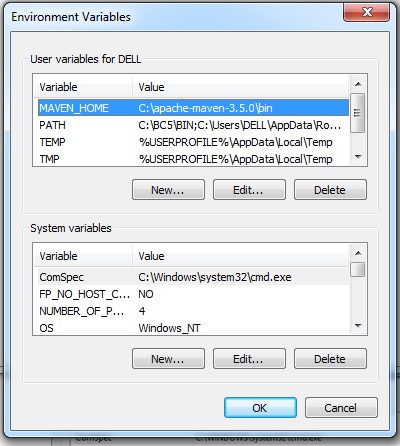
1. Eclipse : digunakan untuk pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web.
2. XAMPP : adalah sebagai server yang berdiri sendiri (localhost)
3. SQLyog(bila perlu) : yang memudahkan pengguna melakukan administrasi maupun melakukan pengolahan data
4. Postman : fungsinya adalah sebagai REST Client atau istilahnya adalah aplikasi yang digunakan untuk melakukan uji coba REST API yang telah kita buat.

Catatan : Lakukan Start pada XAMPP sebelum memulai projek (MySql)

* Install Maven

Extract maven pada data C:

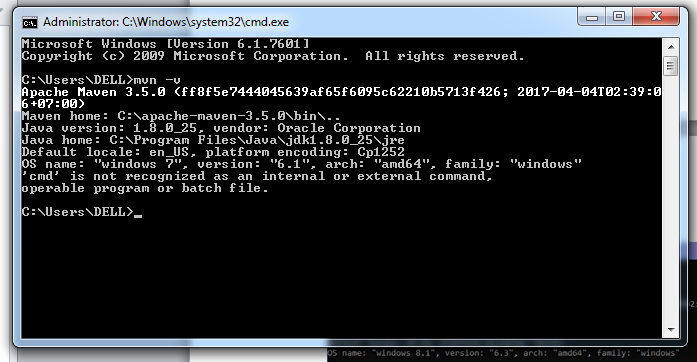
Lalu lakukan setting path seperti berikut



1. Buka Control Panel
2. Pilik System & Security
3. Pilih System
4. Pilih Advanced System Settings
5. Pilih Environment Variables (maka akan tampil seperti gambar disamping)
6. Kemudian buatlah variabel baru MAVEN\_HOME dengan value adalah lokasi penyimpanan file maven
7. Kemudian edit pada variabel path dengan menambahkan /bin pada akhir value
8. Klik OK

Kemudian untuk mengecek apakah maven sudah terinstall, buka cmd dan ketik **mvn –v**

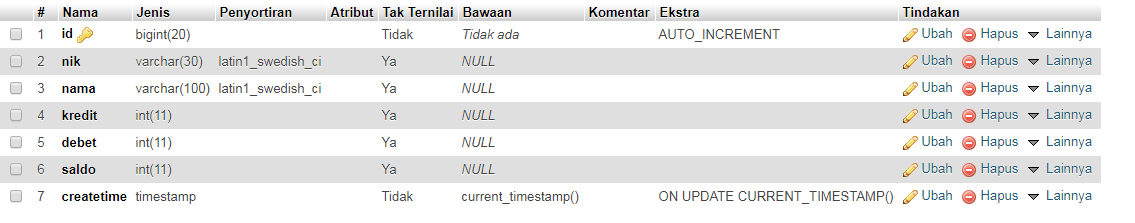
Seperti gambar di bawah ini :



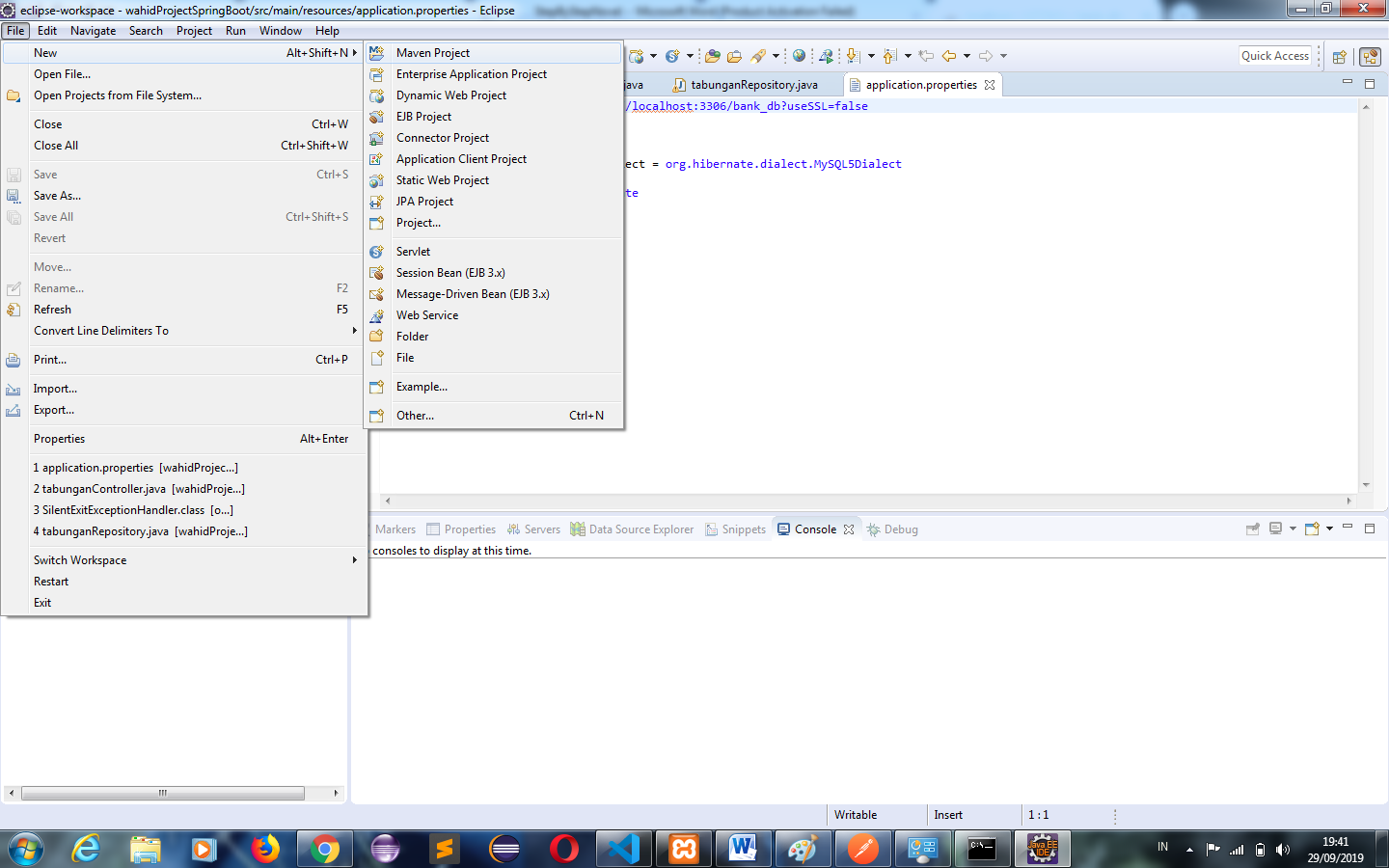
Apabila sudah menampilkan hasil seperti gambar di samping, maka artinya maven sudah ter install

* Create Projek Maven

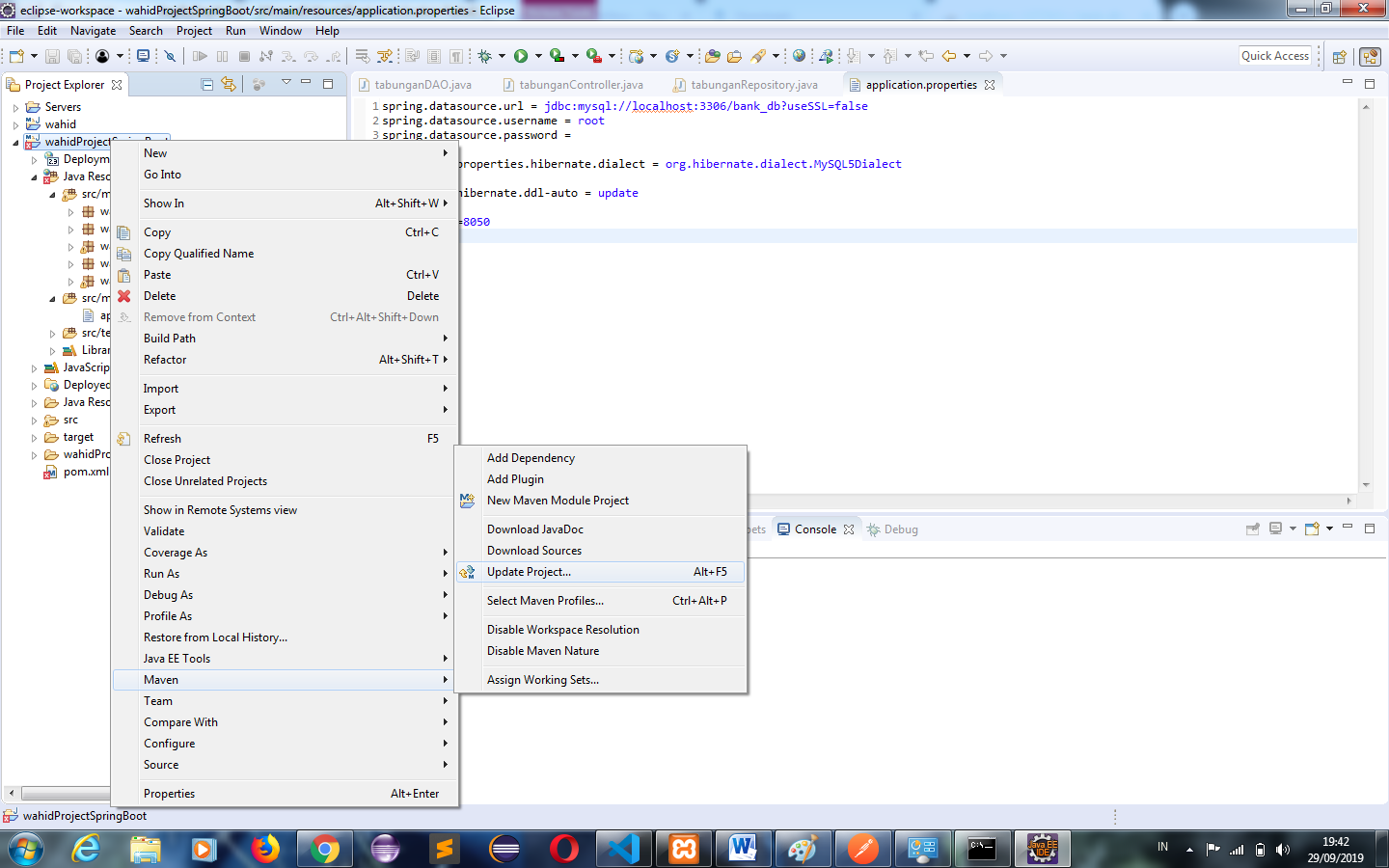
1. Sebelumnya buatlah database seperti di bawah ini pada MySql



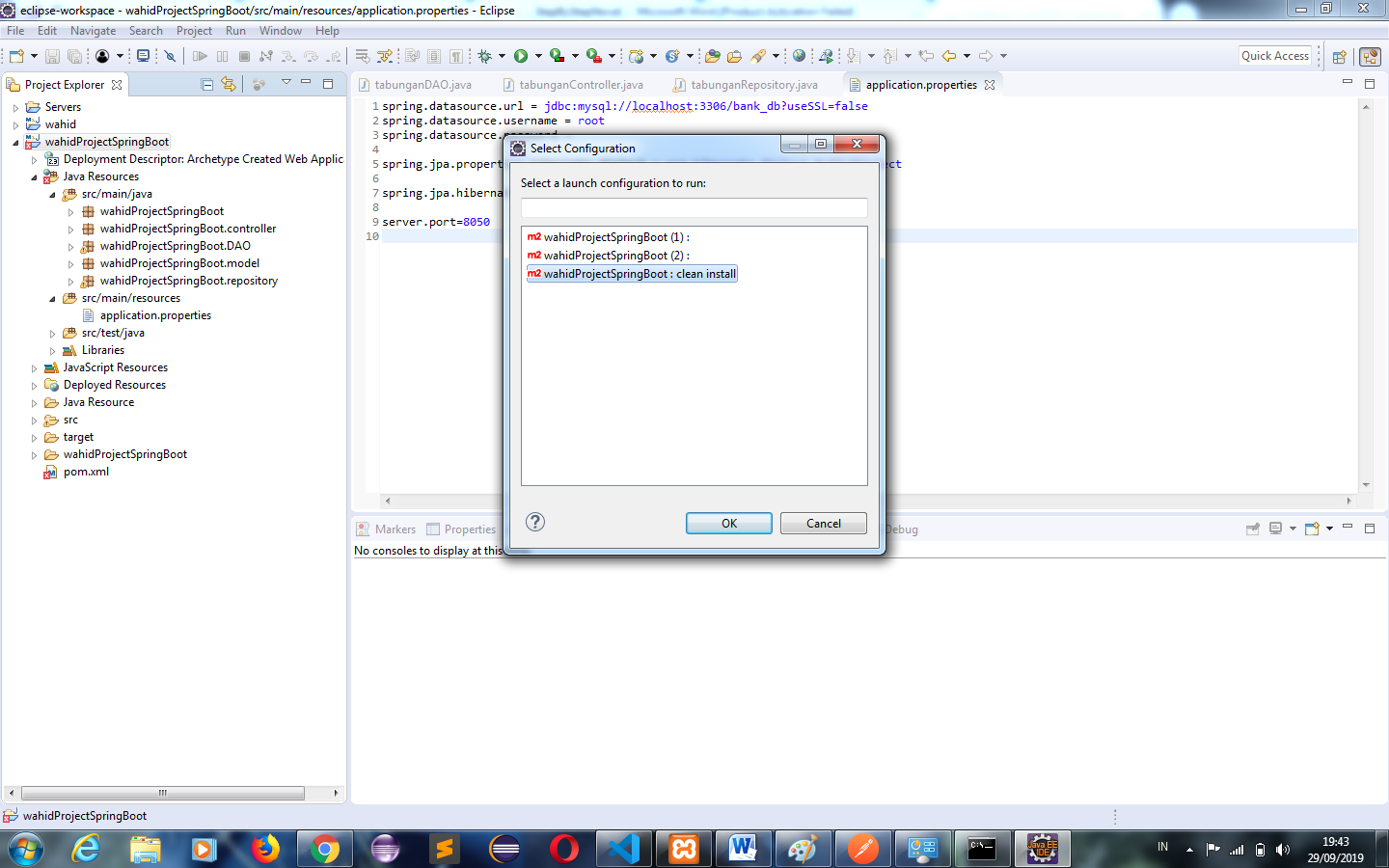
1. Untuk membuat projek maven pada eclipse **File -> New –> Project Maven –> Plih Lokasi Penyimpanan –> Kemudian pada select an Archetype pilih webapp -> Beri nama projek pada Group Id & nama pada Artifact Id -> Finish**



1. Setelah create maka muncul tanda error pada projek, untuk menghilangkannya **Klik kanan projek -> maven -> update project**



1. Run projek dengan cara **Klik kanan projek -> pilih Run As -> Maven Build -> isi Goals : clean install** seperti gambar di bawah ini :



1. Buka pom.xml, kemudian isi dengan ssource seperti di bawah ini :

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>novelProjectSpringBoot</groupId>

<artifactId>novelProjectSpringBoot</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>novelProjectSpringBoot Maven Webapp</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.4.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>novelProjectSpringBoot</finalName>

</build>

</project>

1. Untuk konfigurasi dengan MySql maka **klik kanan pada src/main resource -> new -> file -> application.properties**

spring.datasource.url = jdbc:mysql://localhost:3306/bank\_db?useSSL=false

spring.datasource.username = root

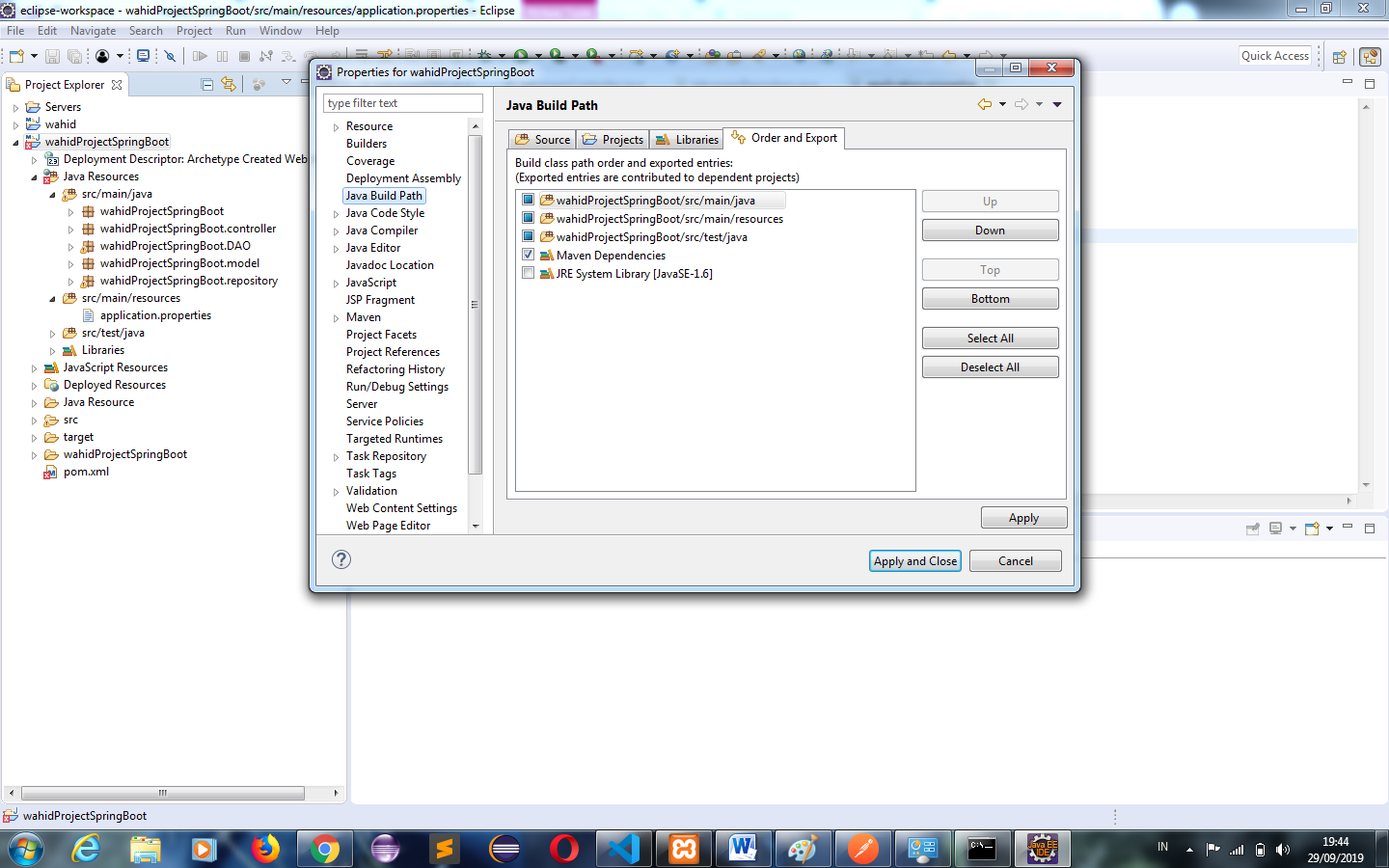
spring.datasource.password =

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

spring.jpa.hibernate.ddl-auto = update

server.port=8050

1. Untuk membuat pemodelan dan DAO **Klik kanan pada projek -> properties -> Java Build Path -> ceklist pada (Maven Dependencies & JRE) -> Apply -> Apply and close**



1. Membuat model **klik kanan pada src/main -> new -> class -> beri nama class “tabungan.java” dan package “**wahidProjectSpringBoot.model**” -> Finish**

Lalu isi dengan source berikut

package wahidProjectSpringBoot.model;

import java.util.Date;

import javax.persistence.Entity;

import javax.persistence.EntityListeners;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

import org.hibernate.validator.constraints.NotBlank;

import org.springframework.data.annotation.LastModifiedDate;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

@Entity

@Table(name="tabungan\_tbl")

@EntityListeners(AuditingEntityListener.class)

public class tabungan {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private long id;

@NotBlank

private String nik;

@NotBlank

private String nama;

private int kredit;

private int debet;

private int saldo;

@Temporal(TemporalType.TIMESTAMP)

@LastModifiedDate

private Date createtime;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getNik() {

return nik;

}

public void setNik(String nik) {

this.nik = nik;

}

public String getNama() {

return nama;

}

public void setNama(String nama) {

this.nama = nama;

}

public int getKredit() {

return kredit;

}

public void setKredit(int kredit) {

this.kredit = kredit;

}

public int getDebet() {

return debet;

}

public void setDebet(int debet) {

this.debet = debet;

}

public int getSaldo() {

return saldo;

}

public void setSaldo(int saldo) {

this.saldo = saldo;

}

public Date getCreatetime() {

return createtime;

}

public void setCreatetime(Date createtime) {

this.createtime = createtime;

}

}

1. Membuat DAO **klik kanan pada src/main -> new -> class -> beri nama class “tabunganDAO” dan package “**wahidProjectSpringBoot.DAO**” -> Finish**

Lalu isi dengan source code berikut

package wahidProjectSpringBoot.DAO;

import java.util.ArrayList;

import java.util.List;

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

import org.springframework.stereotype.Service;

import wahidProjectSpringBoot.model.tabungan;

import wahidProjectSpringBoot.repository.tabunganRepository;

@Service

public class tabunganDAO {

@Autowired

tabunganRepository tabunganrepository;

//AMBIL SEMUA DATA

public List<tabungan> getALL(){

return tabunganrepository.findAll();

}

//DATA BERDASARKAN NIK

public List<tabungan> getByNIK(String nik){

return tabunganrepository.findTabunganByNIK(nik);

}

//UNTUK SIMPAN TABUNGAN

public void save(tabungan tabungan) {

tabunganrepository.save(tabungan);

}

//UNTUK AMBIL DATA DENGAN ID

public tabungan getfindOne(Long id) {

return tabunganrepository.findOne(id);

}

//UNTUK HAPUS DATA

public void delete(Long id) {

tabunganrepository.delete(id);

}

}

1. Membuat Controller **klik kanan pada src/main -> new -> class -> beri nama class “tabunganController” dan package “**wahidProjectSpringBoot.controller**” -> Finish**

Lalu isi dengan source berikut

**package** wahidProjectSpringBoot.controller;

**import** java.util.List;

**import** javax.validation.Valid;

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

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

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

**import** org.springframework.web.bind.annotation.PathVariable;

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

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** wahidProjectSpringBoot.DAO.tabunganDAO;

**import** wahidProjectSpringBoot.model.tabungan;

@RestController

@RequestMapping("/bank")

**public** **class** tabunganController {

@Autowired

tabunganDAO tabungandao;

//UNTUK UPDATE SALDO

**public** **void** updeteSaldo(String nik) {

List<tabungan> data=tabungandao.getByNIK(nik);

**int** a=0;

**int** saldo=0;

**while**(a<data.size()) {

saldo=(saldo+data.get(a).getDebet())-data.get(a).getKredit();

data.get(a).setSaldo(saldo);

tabungandao.save(data.get(a));

a++;

}

}

//UNTUK AMBIL SEMUA DATA

@GetMapping("/tabungan")

**public** List<tabungan> getALL(){

**return** tabungandao.getALL();

}

//UNTUK SIMPAN TABUNGAN

@PostMapping("/tabungan")

**public** ResponseEntity<tabungan> saveTabungan(@Valid @RequestBody tabungan tabunganprm) {

tabungandao.save(tabunganprm);

updeteSaldo(tabunganprm.getNik());

**return** ResponseEntity.*ok*().body(tabunganprm);

}

//UNTUK AMBIL DATA DENGAN ID

@GetMapping("/tabungan/{id}")

**public** ResponseEntity<tabungan> getByID(@PathVariable(value="id") Long id) {

tabungan b=tabungandao.getfindOne(id);

**if**(b==**null**) {

**return** ResponseEntity.*notFound*().build();

}

**else** {

**return** ResponseEntity.*ok*().body(b);

}

}

//UNTUK HAPUS DATA DENGAN ID

@DeleteMapping("/tabungan/{id}")

**public** ResponseEntity<tabungan> delete(@PathVariable(value="id") Long id) {

tabungan b=tabungandao.getfindOne(id);

**if**(b==**null**) {

**return** ResponseEntity.*notFound*().build();

}

**else** {

tabungandao.delete(id);

updeteSaldo(b.getNik());

**return** ResponseEntity.*ok*().build();

}

}

//UNTUK UBAH DATA DENGAN ID

@PutMapping("/tabungan/{id}")

**public** **void** update(@PathVariable(value="id") Long id,@Valid @RequestBody tabungan tabungan){

tabungan b=tabungandao.getfindOne(id);

**if**(b==**null**) {

ResponseEntity.*notFound*().build();

}

**else** {

b.setNama(tabungan.getNama());

b.setKredit(tabungan.getKredit());

b.setDebet(tabungan.getDebet());

b.setSaldo(tabungan.getSaldo());

tabungandao.save(b);

updeteSaldo(b.getNik());

ResponseEntity.*ok*().body(b);

}

}

}

1. Membuat Repository **klik kanan pada src/main -> new -> class -> beri nama class “tabunganRepository” dan package “**wahidProjectSpringBoot.repository**” -> Finish**

Lalu isi dengan source berikut

package wahidProjectSpringBoot.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.jpa.repository.cdi.JpaRepositoryExtension;

import org.springframework.data.repository.query.Param;

import wahidProjectSpringBoot.model.tabungan;

public interface tabunganRepository extends JpaRepository<tabungan, Long>{

@Query(value="SELECT \*FROM tabungan\_tbl WHERE nik=:nik ",nativeQuery=true)

List<tabungan> findTabunganByNIK(@Param("nik") String nik);

}

1. Membuat class main **klik kanan pada src/main -> new -> class -> beri nama class “bankApplication” dan package “**wahidProjectSpringBoot**” -> Finish**

Lalu isi dengan source berikut

package wahidProjectSpringBoot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication

@EnableJpaAuditing

public class bankAplication {

public static void main(String[] x) {

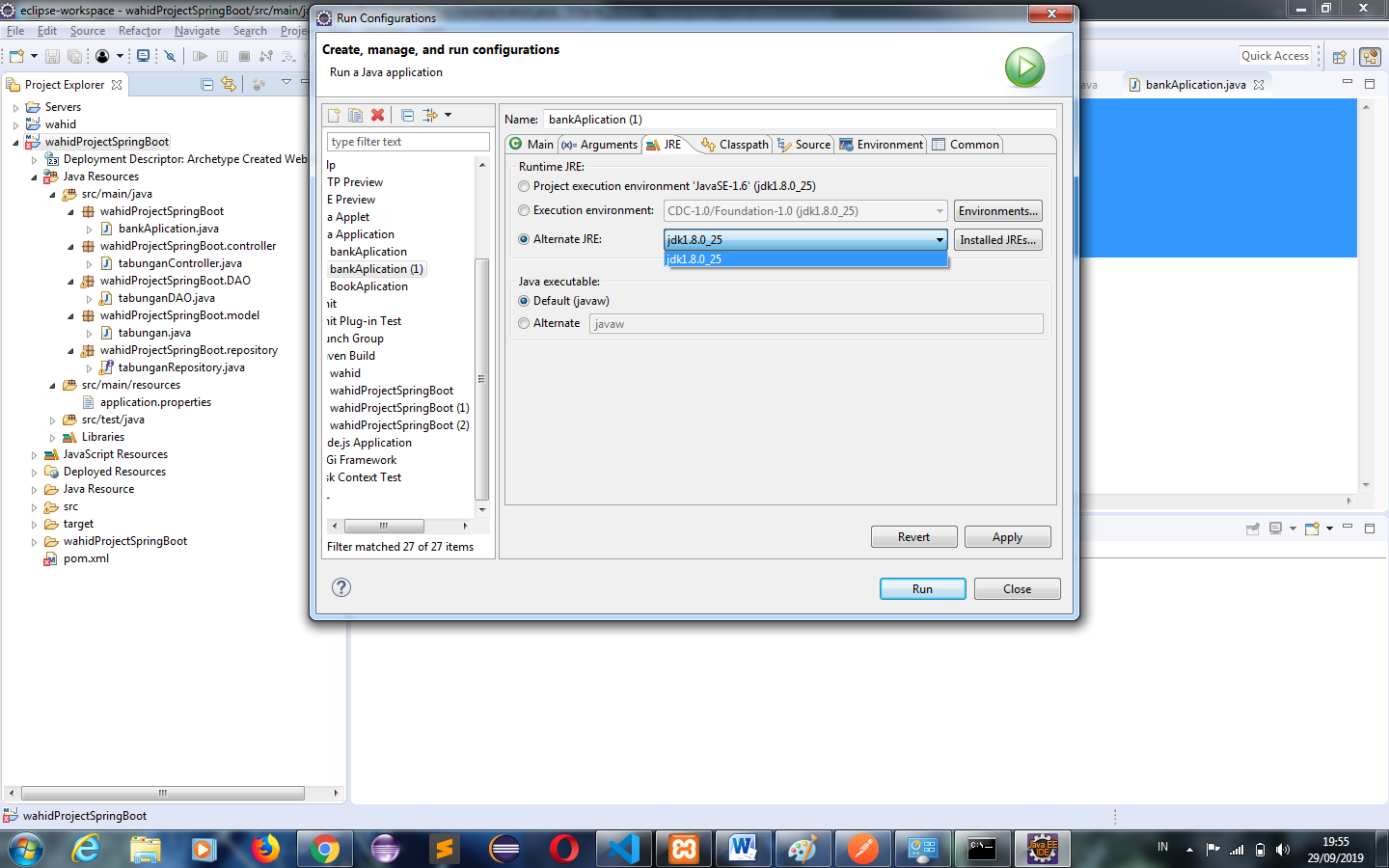
SpringApplication.run(bankAplication.class, x);

}

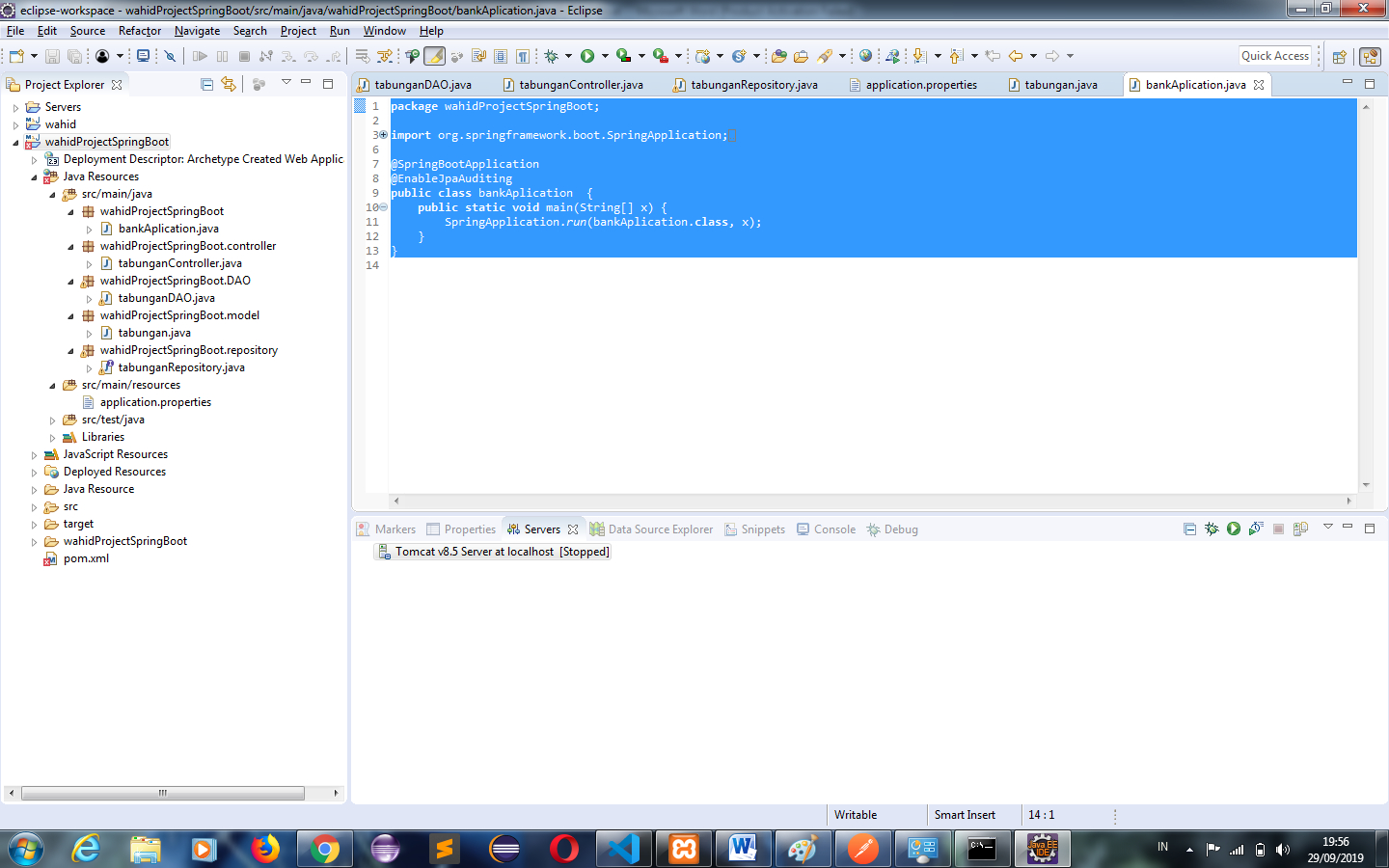
}

1. Mengganti JRE ke JDK

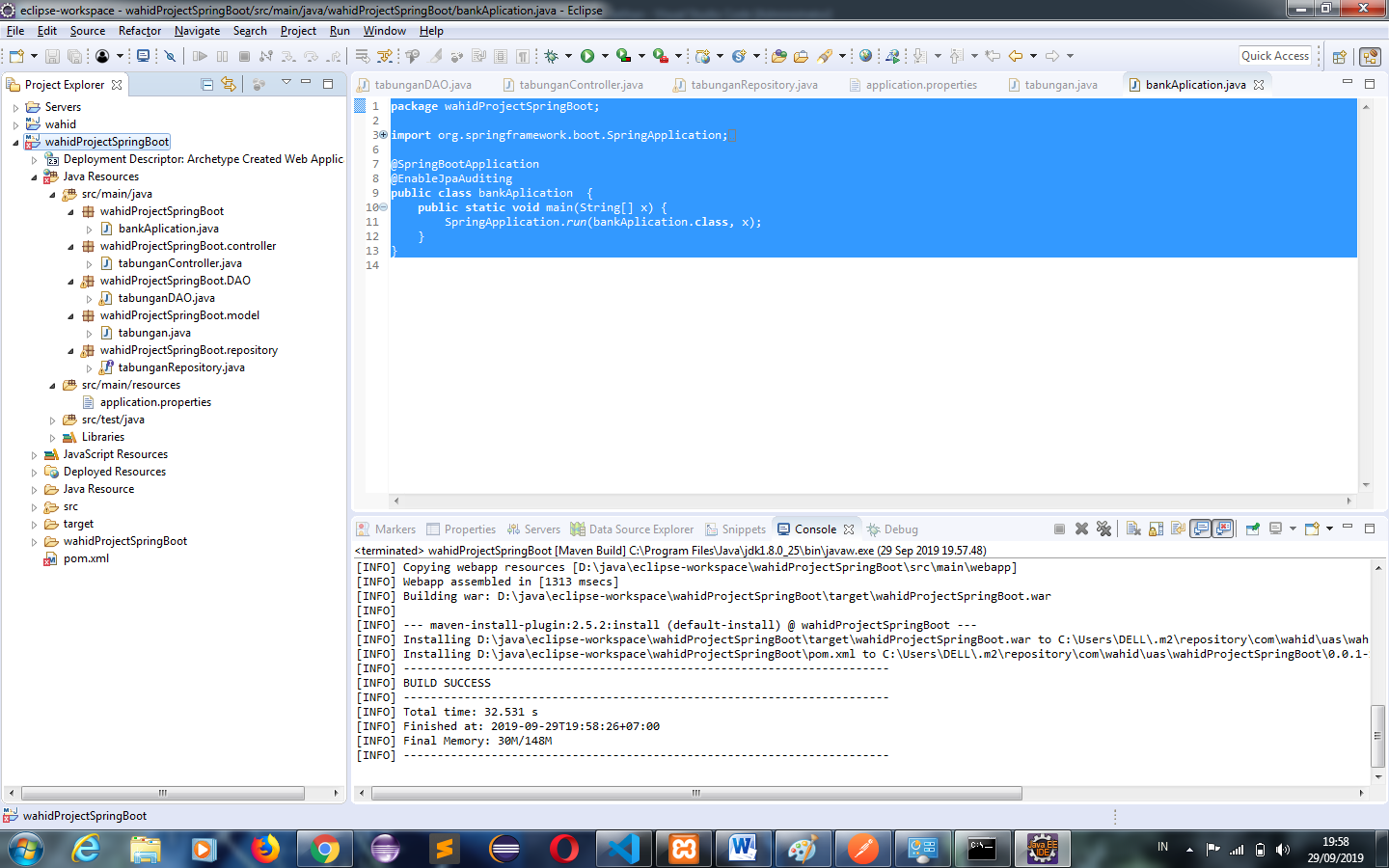
**Klik kanan -> Run As -> Run Configuration -> JRE -> Installed JREs.. -> add -> cari tempat penyimpanan file jdk -> finish**



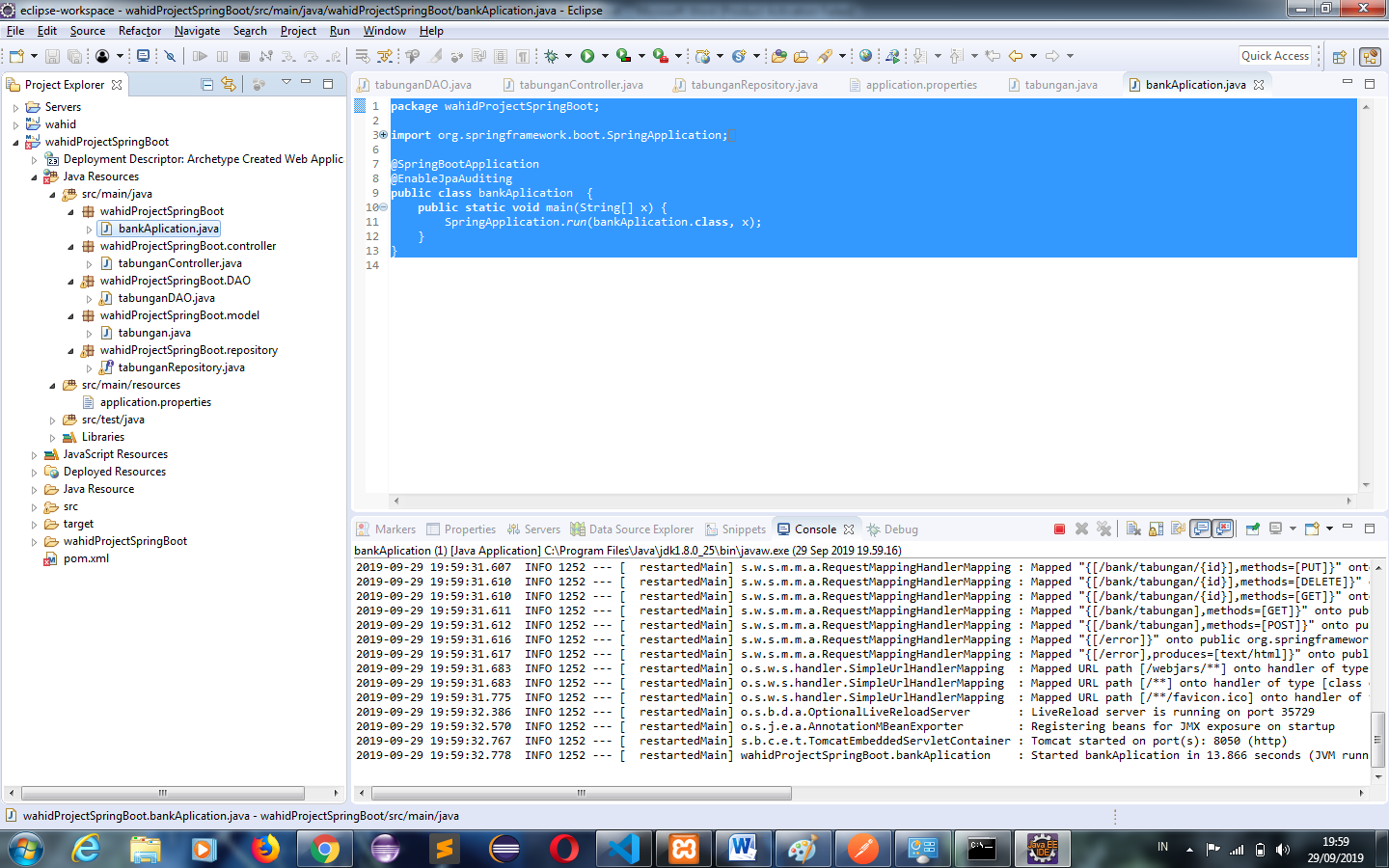
1. Membuat Server Tomcat



1. Untuk mengecek projek kita apakah sukses atau tidak maka **klik kanan pada projek -> Run As -> Maven Build**

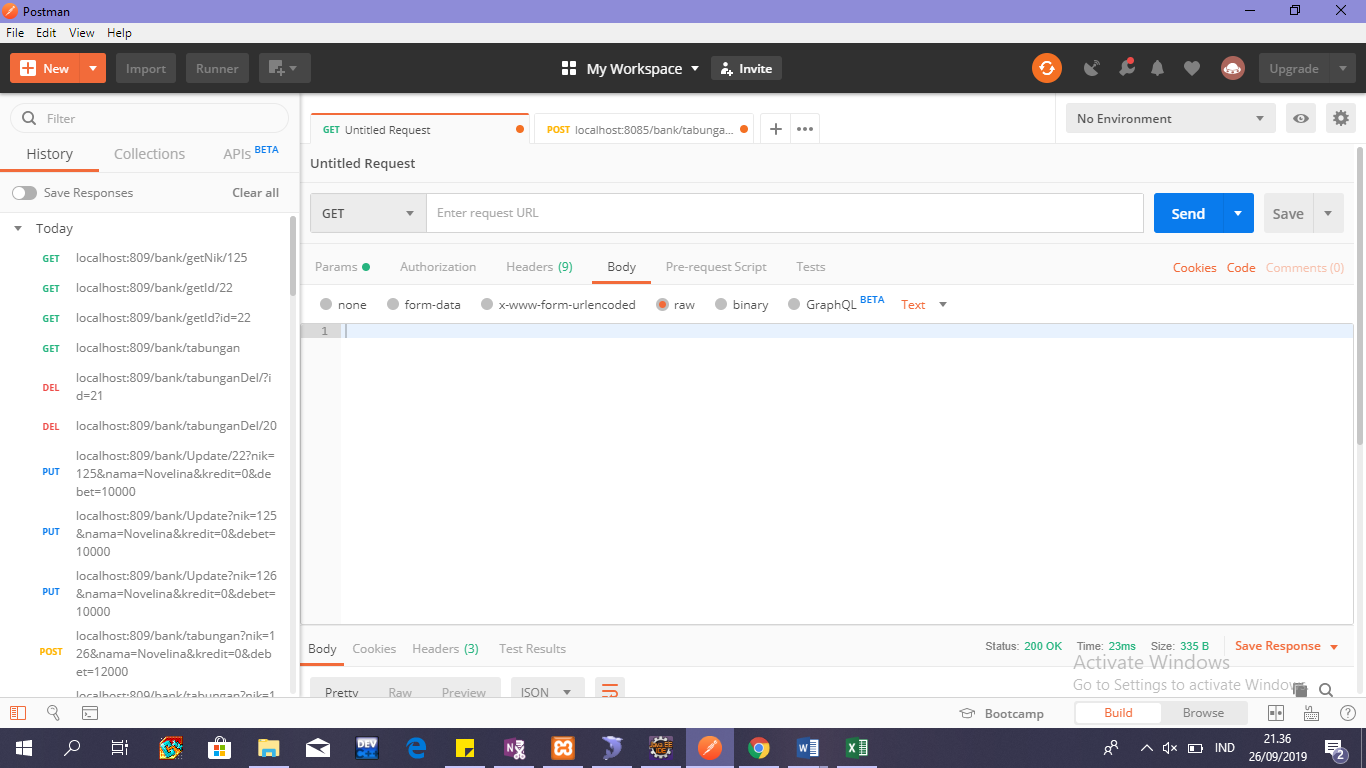


1. Kemudian **Klik kanan pada class TabunganApplication -> Run -> Java Application**, apabila berhasil maka akan menampilkan :



* Menjalankan Project dengan Postman

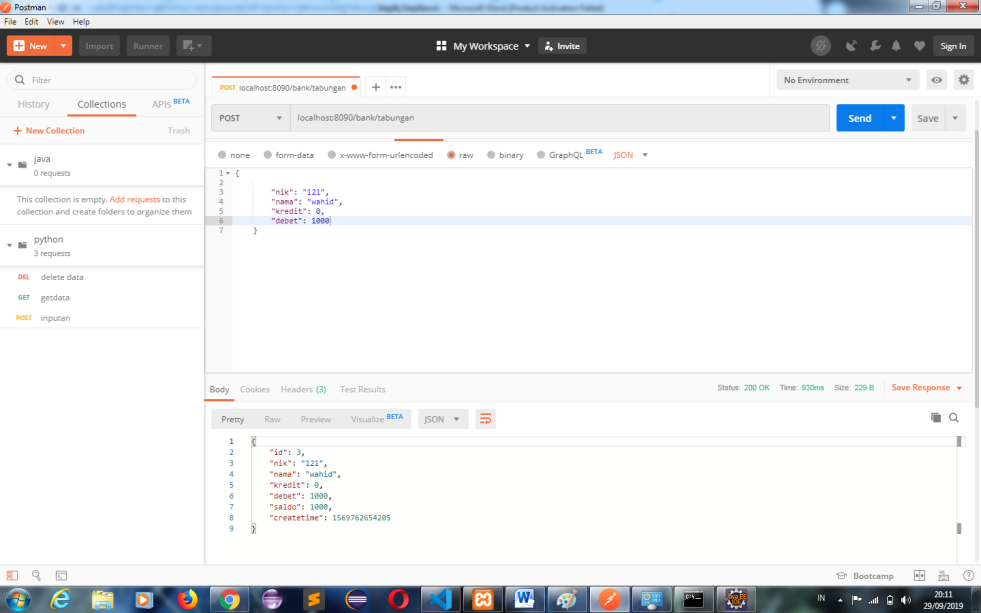
1. Buka Aplikasi Postman
2. Sign in dengan menggunakan akun email
3. Maka akan menampilkan tampilan postman seperti dibawah ini :



1. Untuk mencoba Rest API yang telah dibuat maka tuliskan alamat URL nya kemudian send maka akan tampil hasilnya.
2. Adapun method yang terdapat dalam postman (yang sering digunakan) antara lain :
   * GET digunakan untuk memanggil data yang telah ada
   * POST digunakan untuk menambahkan data ke dalam server
   * PUT digunakan untuk mengupdate data yang telah ada
   * DELETE digunakan untuk menghapus data yang telah ada
3. Contoh cek Rest API POST (untuk menambah data/insert data)

* Masukkan URL (sesuaikan port, kata kunci dan parameter)

localhost:8090/bank/tabungan?nik=121&nama=wahid&kredit=0&debet=1000



* Maka akan menampilkan hasil dari proses POST

