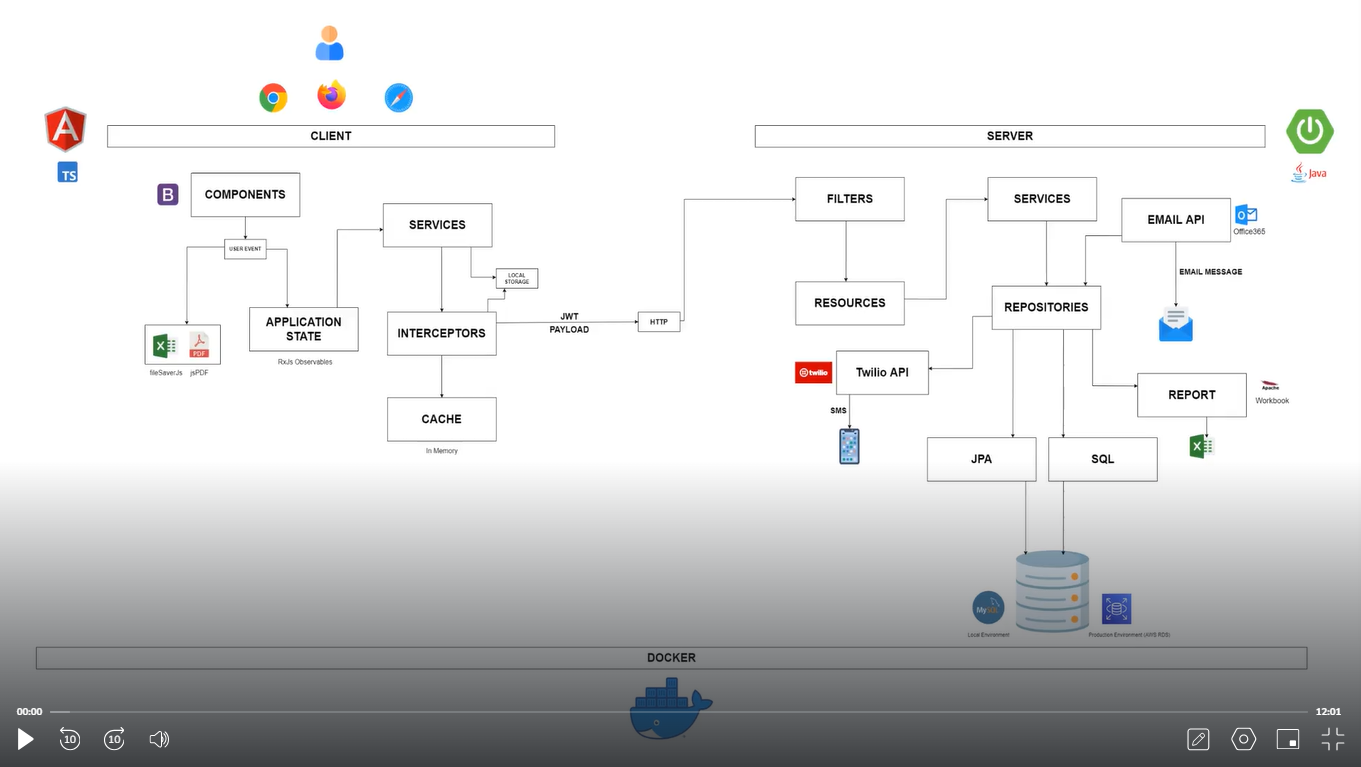
# Backend Application

## Backend Application

### Introduction

### Architecture Design



### Software Requirements

JAVA JDK

SPRING FRAMEWORK

MYSQL

INTELLIJ IDEA

### Creating Spring Boot Application

SETUP SPRING BOOT USING SPRING INITIALIZR (https://start.spring.io/)

* Project Choose Maven
* Languange Choose JAVA
* Version Choose 3.2.2
* Projects Metadata
  + Group : cc.kfy
  + Artifact : blitzmart
* DEPENDENCIES:
  + Spring Web
  + MySql Driver
  + Spring Security
  + Spring Data JDBC
  + Lombok
  + Validation I/O

Buka file blitzmart/pom.xml untuk pastikan semua dependencies terakomodasi.

### Run Spring Boot Application

buat database blitzmart pada mysql, pada project ini menggunakan provider mysql mariadb yang ada pada xampp.

Rename file blitzmart/src/main/resources/application.properties jadi application.yml

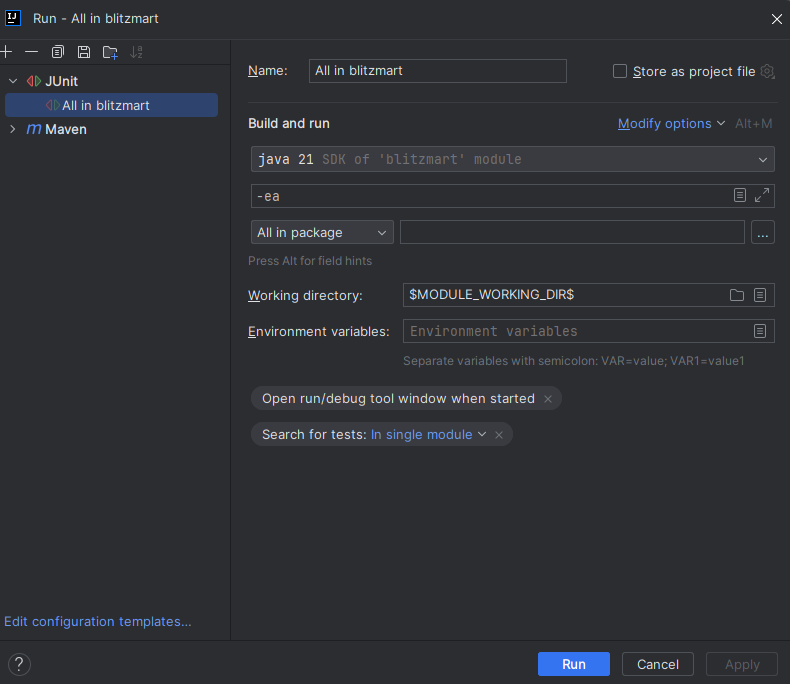
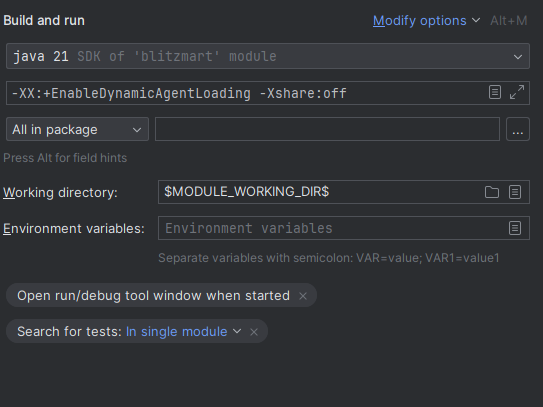
Lalu Buka file application.yml tersebut dan ketikan kode konfigurasi berikut :

*#blitzmart/src/main/resources/application.yml*

spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/blitzmart  
 password: admin123  
 username: root  
 jpa:  
 database-platform: org.hibernate.dialect.MariaDBDialect  
 generate-ddl: true  
 show-sql: true  
 hibernate:  
 ddl-auto: update  
 properties:  
 hibernate:  
 globally\_quoted\_identifiers: true  
 dialect: org.hibernate.dialect.MariaDBDialect  
 format\_sql: true  
 sql:  
 init:  
 mode: never

Untuk memastikan aplikasi pada menu bar , pilih Run > Run, lalu pilih Edit Configurations…

Pada JUnit, namakan perintah tersebut, lalu pada build and run atur pengaturan seperti berikut :

*-XX:+EnableDynamicAgentLoading -Xshare:off*

Klik apply dan run.

Jika pilih run langsung tanpa konfigurasi argumen run maka akan muncul pesan peringatan seperti berikut :

WARNING: A Java agent has been loaded dynamically (C:\Users\hwang\.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.11\byte-buddy-agent-1.14.11.jar)

WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning

WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information

WARNING: Dynamic loading of agents will be disallowed by default in a future release

Untuk warning ini bisa dihilangkan dengan argument *-XX:+EnableDynamicAgentLoading*

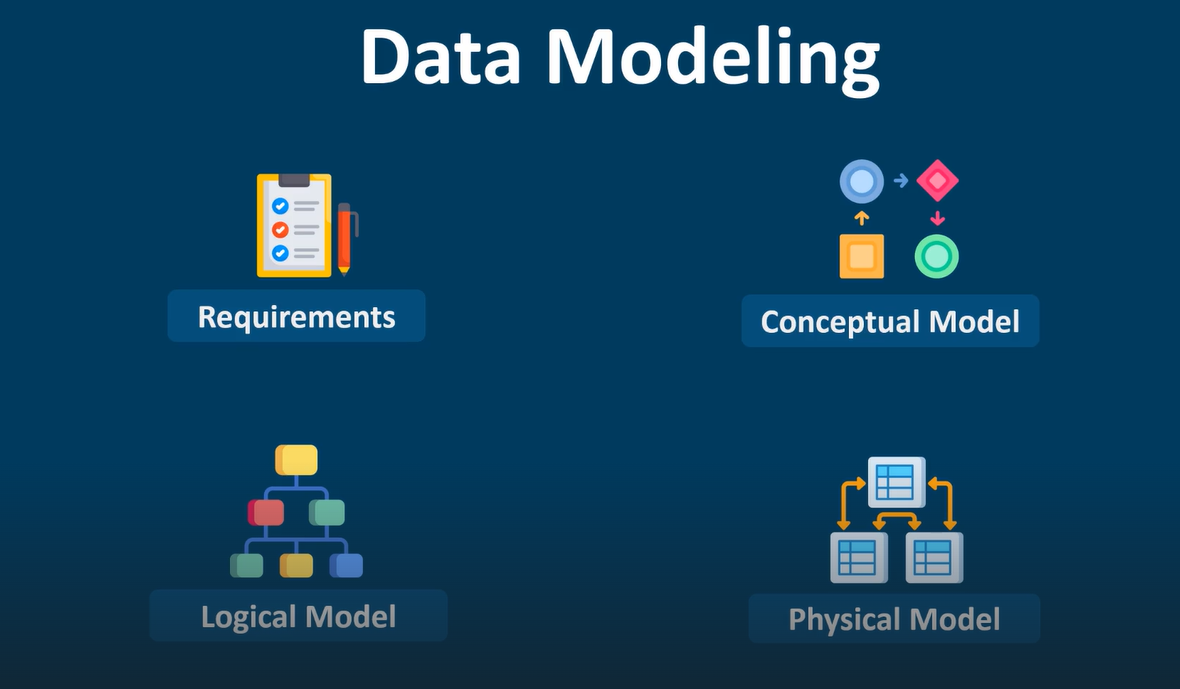
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended

Untuk warning ini bisa dihilangkan dengan argument *-Xshare:off*

Jika hasil run adalah Tests passed maka aplikasi berhasil dijalankan.

## Data Modeling- Forward Engineering

### Introduction



### Modelling Tools

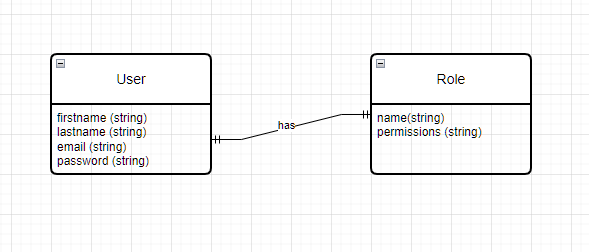
Gunakan draw io

### Application Requirements

Baca file blitzmart/requirements.md

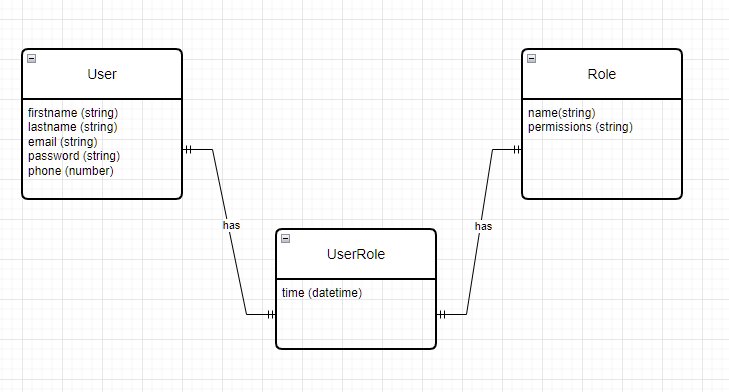
### Conceptual Model

Guanakan <https://app.diagrams.net/>



### Logical Model

Gunakan <https://app.diagrams.net/>



Src: [Blitzmart Data Modelling - Google Drive](https://drive.google.com/drive/folders/1HKSjspKyo4UR_5i7U3PHwOccGob232mF)

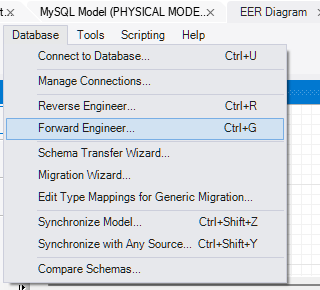
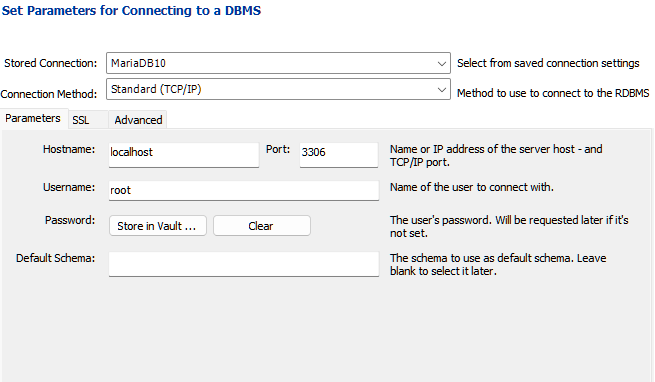
### Physical Model

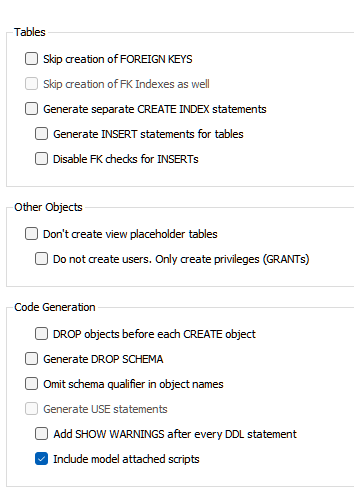
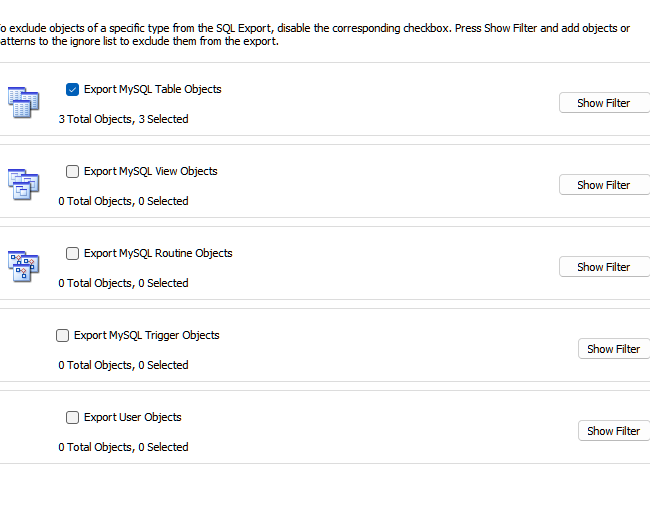
Gunakan MySQL Workbench

### Foreign Key Constraints

Gunakan MySQL Workbench > Foreign Key > Cascade

### Forward Engineering

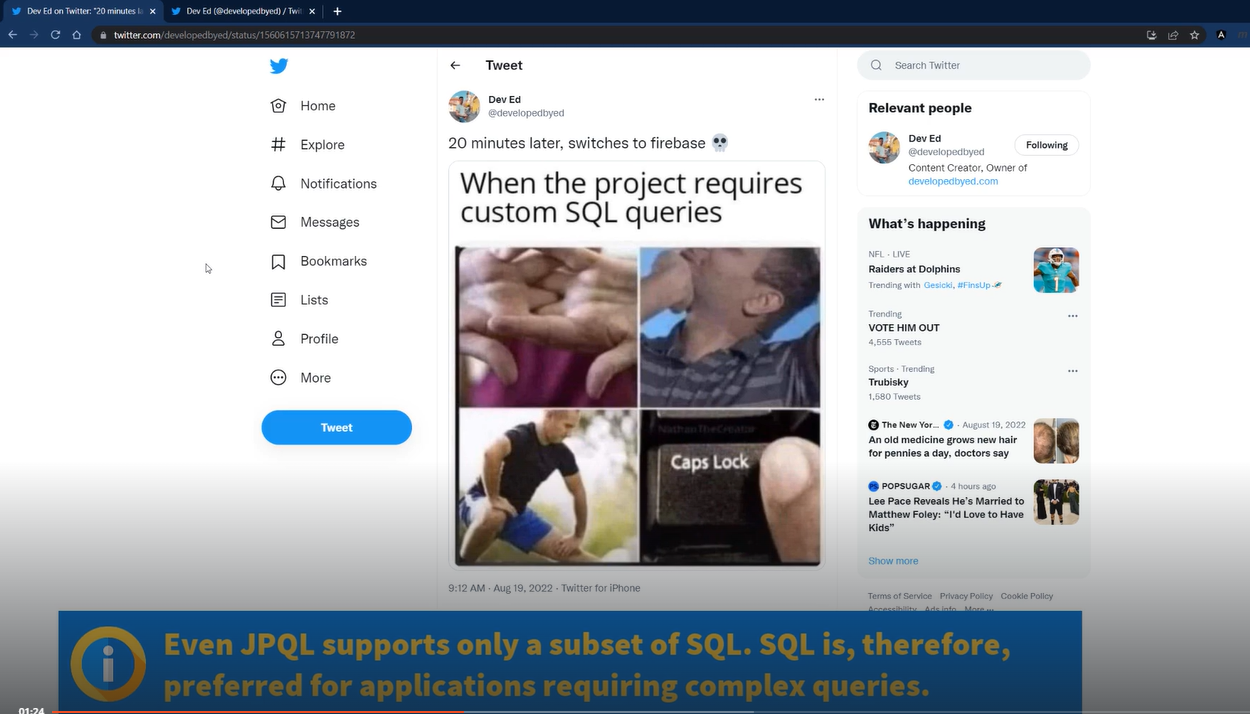
 

Hapus `VISIBLE` pada query disebelah karena tidak compatible dengan mariadb.

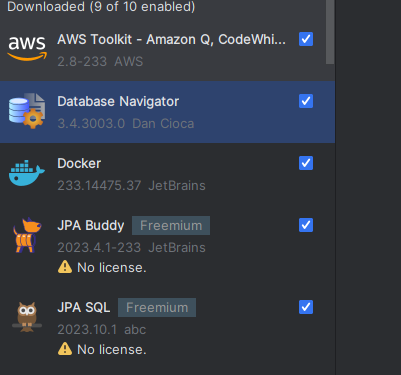
## Data Modelling Reverse Engineering

### Introduction



### SQL General Rules

Install plugins Database Navigator pada intellij idea community edition.



Buat file blitzmart/main/resources/schema.sql

### User Tables

### User Role Tables

### User Event Tables

### Account Verification Table

### Reset Password Verification Table

### Two Factor Verification Table

### Run Application Schema

Pertama buka file blitzmart/src/main/resources/application.yml

Lalu rubah dan tambahkan kode berikut :

mode: always

continue-on-error: false

Tepat dibawah settingan

sql :

init:

mode: always

continue-on-error: false

mode : always agar script dapat tergenerate ke database.

Setelah script sql selesai dibuat, perlu menambah timezone pada sql agar perintah set time\_zone pada script schema.sql dapat dijalankan.

Download file timezone mysql di <https://dev.mysql.com/downloads/timezones.html>

simpan kedua file tersebut ke lokasi folder zoneinfo.

Buka mysql workbench, pada connection, pilih edit connection, dan pastikan schema yang terpilih adalah mysql.

Selanjutnya buka query window, ketikan kode berikut :

Use mysql;

Selanjutnya copy paste isi script sql timezone\_posix.sql dan timezone\_leaps.sql ke query editor.

Lalu jalankan untuk mengimport data script ke table mysql.

Setelah itu kembali ke intellij idea dan jalankan run.

Setelah selesai buka workbench kembali, maka table-table pada script schema.sql akan terdaftar pada database blitzmart.

### Reverse Engineering With MYSQL Workbench

Buka mysql workbench

Buka menu database > Reverse Engineer

Lalu next dan pilih database blitzmart.

Next setelah itu workbench akan mengenerate table secara otomatis.

## User Registration Features

### User Repository Interface

* Create new packages named “**domain**”
* Create new packages named “**repository**”
* Then create new java class named ‘**User’** in domain packages
* Then create new java interface named **'UserRepository’**, this repository inherith from User, then the interface should code like public interface UserRepository<T extends User> { }

### User Class

Open user class in domain packages and create some property with the annotations for validation.

### User Repository Implementation

* Inside “repository” packages create new packages named “**implementation**”
* Create new class in implementation packages, named ‘**UserRepositoryImpl’** that implemented the UserRepository Interface where created in “repository” packages, like public class UserRepositoryImpl implements UserRepository<User> { }
* In ‘UserRepositoryImpl’ class create some method that used for CRUD and another operations
* Create new packages named “**exception”** to create custom application exception, create new class named ‘**ApiException’**
* Create new packages named “**query**” to hold class for manage sql query,and then create the new class named as ‘**UserQuery’** for handle user database query
* Create new packages named “**enumeration”** for holding custom type information, create new class named as ‘**RoleType’**,and create another new class named as ‘**VerificationType’**
* Open ‘RoleRepository’ interface in “repository” packages and add some method for the repository operation
* Create new class in “implementaion” packages inside the “repository” packages to implement the role repository named as ‘**RoleRepositoryImpl’**
* Create new class named ‘**RoleQuery’** in “Query” Packages to write sql query to handle role database query
* Create new packages named as “**rowmapper**”, and create new class to map role java object named as ‘**RoleRowMapper’**
* Update query packages for ‘UserQuery’ class and ‘RoleQuery’ class

### User Service

* Create new packages named as “**service**”, and create new class named as ‘**UserService’**
* Create new package named as “**dto**”, and inside the package create new java class named as `**UserDTO’** base on `User` class in domain packages, the dto is an object sent to frontend that some field where no need at frontend
* To make domain class can switch between backend and frontend, we need to create mapper, create new package named as “**dtomapper**”, inside the packages create a new class named as ‘**UserDTOMapper’** that used to map between dto object and domain object class vice versa
* Open back the `UserService` to continue writing the operation of the service feature
* Inside the “service” package, create new packages named as “**implementation**” package for holding implementation of service, then inside the “**implementation**” package create new class named as ‘**UserServiceImpl’** to create implementation of ‘UserService’ class

### User Resource

* Create the controller or Resources, then create new package named as “**resource**”, and inside the package create new class named as ‘**UserResource’** as User Controller, this controller is RestController for mapping api address
* Create new helper class inside “domain” package named as ‘**HttpResponse’**, this class is used as custom http response
* Back to ‘UserResource’ class to consume the custom httpresponse come from ‘HttpResponse’ class inside of “domain” package

### Populate Roles Table

Before the application can be test and run, we need the Role value that manually insert into sql database, the query for all this section is above :

use blitzmart;

select \* from users;

select \* from roles;

select \* from accountverifications;

insert into roles(name, permission)

values

('ROLE\_USER','READ:USER,READ:CUSTOMER'),

('ROLE\_MANAGER','READ:USER,READ:CUSTOMER,UPDATE:USER,UPDATE:CUSTOMER'),

('ROLE\_ADMIN','READ:USER,READ:CUSTOMER,CREATE:USER,CREATE:CUSTOMER,UPDATE:USER,UPDATE:CUSTOMER'),

('ROLE\_SYSADMIN','READ:USER,READ:CUSTOMER,CREATE:USER,CREATE:CUSTOMER,UPDATE:USER,UPDATE:CUSTOMER,DELETE:USER,DELETE:CUSTOMER');

### Debugging and Testing

* To run and debugging application, just go to intellijidea and run the project as application
* When an error occurred then fixed it,

Example this error :

Consider defining a bean of type 'org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate' in your configuration.

Then doing below configuration in development process : exclude some package in **BlitzmartApplication**

@SpringBootApplication(  
 exclude = { SecurityAutoConfiguration.class}  
)  
public class BlitzmartApplication {

in **application.yml**

*#blitzmart/src/main/resources/application.yml*spring:  
 autoconfigure:  
 exclude: org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration

* Then use POSTMan to test the api router

## Application Profiles

### Profiles Setup

* Change the configuration from development to production
* Copy the file `**application.yml’** come from `**blitzmart/resources/**` directory and paste it there, named new pasted file as ‘**application-dev.yml**`, doing the same thing once again and named as `**application-prod.yml**`
* Delete all the configuration that write in `application.yml` and write again new code like above :

spring:  
 profiles:  
 active: prod

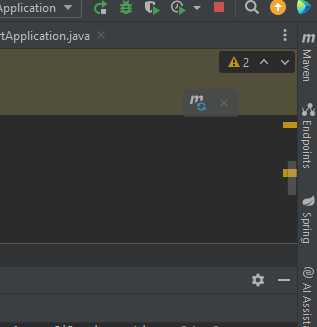
* the active property is used to point out which configuration file that we want to use in spring boot. If we write `**prod**` then the configuration that we use is `application-prod.yml` and when we write `**dev**` the application-dev.yml` configuration file that we used as our application configuration
* or we can simplify using above syntax in `application.yml` to configure our application configuration file

spring:  
 profiles:  
 active: @spring.profiles.active@

* And then Open the `**pom.xml**` file than put above code, put this code above <build></build> section

<profiles>  
 <profile>  
 <id>dev</id>  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 </activation>  
 <properties>  
 <spring.profiles.active>dev</spring.profiles.active>  
 </properties>  
 </profile>  
 <profile>  
 <id>prod</id>  
 <properties>  
 <spring.profiles.active>prod</spring.profiles.active>  
 </properties>  
 </profile>  
</profiles>

* Delete the `blitzmart/**target**` folder and re run again
* Before run, make sure to reload the pom file using maven (click the circle)



* To change the using of configuration mode just move above code to the section we want to use

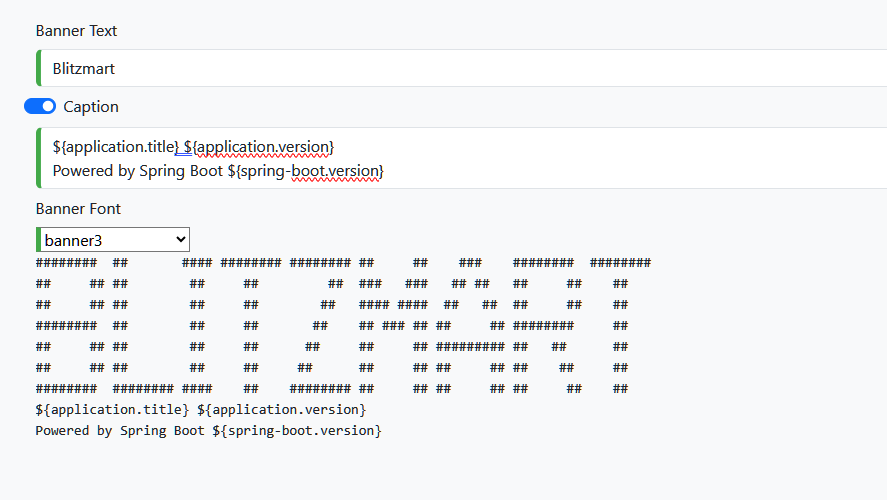
<activation>  
 <activeByDefault>true</activeByDefault>  
</activation>

* After that, reload the pom file using maven

### Customizing Banner

* Untuk membuat banner pada saat running aplikasi, maka buatlah sebuah file dengan nama `**banner.txt**` di folder ‘**blitzmart/resources/**’
* Lalu buka situs dibawah :

<https://devops.datenkollektiv.de/banner.txt/index.html>



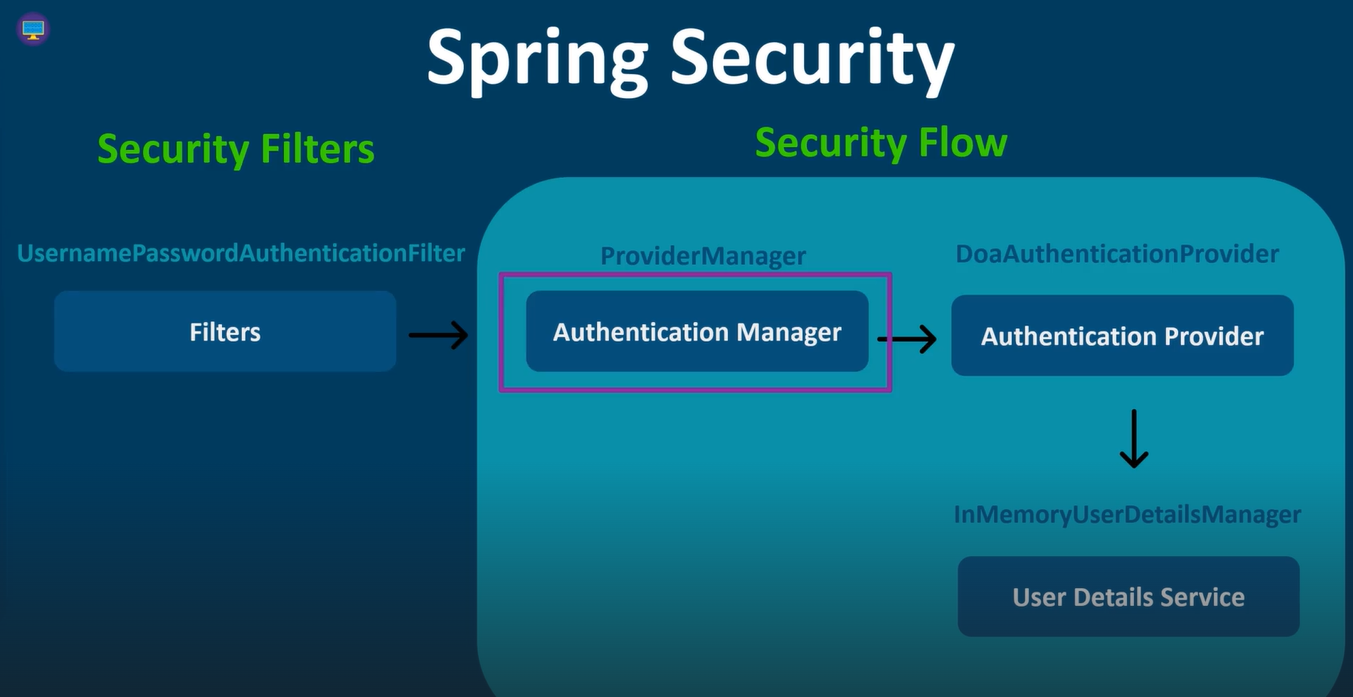
* Copy text tersebut dan simpan ke file ‘**bliztmart/resources/banner.txt**’
* Buka file `**apllication.yml**` yang ada didirektori ‘**blitzmart/resources/**’ lalu tambahkan kode berikut :

*#blitzmart/src/main/resources/application.yml*spring:  
 profiles:  
 active: '@spring.profiles.active@'  
  
application:  
 title: Blitzmart  
 version: 1.0

## Spring Security

### Introduction





### Security Configuration

* Create new package named as “**configuration**”, and then create a java class called `**SecurityConfig**`
* Open site [Username/Password Authentication :: Spring Security](https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/index.html#servlet-authentication-unpwd-input) to understand more about spring-security Username/Password Authentication
* To understand about SecurityFilterChain read <https://docs.spring.io/spring-security/reference/servlet/architecture.html#servlet-securityfilterchain>
* To understand about customize AuthenticationManager read <https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/index.html#customize-global-authentication-manager>
* Next open ‘**UserResource’** class from “**resource**” Package, then add `**login**` method to that class
* Create bean of authenticationManager in ‘**SecurityConfig’** class that’s come from “**configuration**” package

private final BCryptPasswordEncoder encoder;

@Bean  
public AuthenticationManager authenticationManager(){  
 DaoAuthenticationProvider authenticationProvider = new DaoAuthenticationProvider();  
 authenticationProvider.setUserDetailsService(null);  
 authenticationProvider.setPasswordEncoder(encoder);  
 return new ProviderManager(authenticationProvider);  
}

* The encoder that we use is come from BCryptPasswordEncoder
* Add below annotations to the class

@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(prePostEnabled = false, securedEnabled = true)  
@RequiredArgsConstructor

public class SecurityConfig {

...

}

* Create new package named as “**handler**”, and then create a class called ‘**CustomAccessDeniedHandler’** and implement the AccessDeniedHandler interface
* Back to ‘**SecurityConfig’** class, and modify the exceptionHandling section, in accessDeniedHandler use ‘**CustomAccessDeniedHandler’** class as parameter
* To used CustomAccessDeniedHandler class need to define a bean of CustomAccessDeniedHandler class

private final CustomAccessDeniedHandler customAccessDeniedHandler;

* Create new class named ‘**CustomAuthenticationEntryPoint’** in “**handler**” package and implement the AuthenticationEntryPoint interface
* Back to ‘**SecurityConfig’** class again, in authenticationEntryPoint part of exceptionHandling section

add the ‘**CustomAuthenticationEntryPoint’** as parameter

* Create class named as ‘**UserPrincipal’** in “**domain**” package and implemented the ‘**UserDetails’** interface comes from springframework.security.core.userdetails.UserDetail
* Open ‘**UserRepositoryImpl’** class from “**repository.implementation**” package and then add the implementation of ‘**UserDetailsService’** interface and implement the loadUserByUsername method that’s come from the interface
* And then create a new class into “**rowmapper**” package, named as ‘**UserRowMapper’**
* And then open ‘**SecurityConfig’** class back and replace the part of authenticationProvider.setUserDetailsService of authenticationManager method to use userDetailsService which define as

private final UserDetailsService userDetailsService;

### Testing Login Method

### Spring Security Update

## Login Functionality

### Review Achitecture Design

### Verification Code

### Login Flow Review

### Token Provider

### Login Test with Token

### Verify Code

### Login Test with Code

### Code Refactoring

### Login Test after Refactoring

## Authorization Filter

### Introduction

### Authorization Filter

## Exception Handling

### Exception Handling

### Testing Exception Handling

### Overiding White Label Error

### Code Refactoring

### Testing Refactored Code

## Reset Password Feature

### Introduction

### Reset Password

## Account Verification

### Account Verification

## Refresh Token

### Refresh Token

# Front End Application

## Front End Application

### Introduction

### Software Requirements

### Creating Angular App

### Main Components

### Routes Configurations

### Bootstrap Setup

### Login Page HTML Markup

### Login State

### Mapping HTTP Response

### User Service Login Function

### Login Form Template

### User Service Verify Code

### Test Login

### Test Verify Code

### Source Code