# App for fictious Car Showroom

## Base params of application

* Spring Boot 2.7.6
* Java 17
* DB: Oracle 19
* Template engine: Thymeleaf

## Description

This document describes code of sample application in Spring Boot on which I have been working for last days. At this moment the applications is consist of about 30 entities and similar number of tables in database. For now is in progress stuff around car configuration. Entities - CarBrand, CarModel, CarEquipmentPack, CarEngine, etc. And the other branch where I am working on is around human resources - entity Employee, and many relational entities. I was created 3 ER diagrams

* ER-human-resources
* ER-car-variant
* ER-rest-of

## Documentation

ER diagrams: https://github.com/mirosimo/car\_showroom/tree/main/xxx-documentation/ER-diagrams

Some View screenshots Examples: https://github.com/mirosimo/car\_showroom/tree/main/xxx-documentation/app-screenshots

## Implemented Features

For now there are implemented things like: Inserting, Updating, Deleting records, Relation between entities, One To Many, Many To Many, Many To Many with extra columns in connection table, Exception management, Multilanguage, Views Validation, Authentification, Authorization - Users, Roles management Images are saved as blob in db, Thymeleaf, Css.

## Spring Security

Spring security contains bunch of stuff - to comprehend all the main parts and principles how everithing works together is quite challenging - it takes some time. E.g. Things like:

* SecurityFilterChain
* ProviderManager
* AuthenticationProvider
* SecurityContextHolder
* UserDetailsService
* etc...

Just now is in development state and during the time are implementing new things and features related to Authentification and Authorization. Detail of actual state are described below.

### Authentication

User details are fetched from database – table **Users** - username, password, email, ... . Password is hasshed by BCryptPasswordEncoder. For authentification is used username and password. In a few days will be add authentication possibility using SSO - OpenID (Using credentials from google account). And I also would like to implement **two step login** --> for practicing work with more AuthenticationProviders.

### Authorization

For roles was created entity Role. Relation between **User** and **Role** is M : N. Through the entity **UserRole** we can finde roles assigned to particular User. Relation **End Point** -**assigned ROLES** is hardcoded in config file (HttpSecurity). Later will be done more sophisticated role management - Roles hierarchy, priviledges.

## Multilanguage

* messages.properties
* messages\_en.properties

## Error, Exception management

When occur an exception, than is user informed ( In concise form via Error page ).

When will be done Priviledge, Role management, than the users with needed priviledges could also display print stack Trace of an exception.

## Thymeleaf

Template files are places in folder: **Resources/templates**

Templates are divided into 4 base groups. Postfix

* -new - view for adding new entity into system
* -list - view where are displayed data in table form
* -detail - view where is displayed detail for particular entity
* -update - view for updating entity

Attribute menuItem (used for menu generating) is pass into templates through MenuInterceptor - Not from controllers. Defining in each controller had been little bit annoying.

## Fragments

Fragment files are placed in folder: **Resources/templates/fragments**

Just now are using 3 fragments.

* **top\_part** - the very top part of application - contains logo and two design images
* **navbar** - contains manu
* **input\_warning\_board** - contains alert board, which is displayed, when user inserts and confirm wrong data.

## CSS

* Top Strip – logo, two images – used FLEX.
* Views for adding new records (label, input box) - used Grid.
* Views for table form data output – used Table
* Top Menu – resources/static/css/top\_navbar.css

Main CSS file – resources/static/css/app\_style.css

## TODO

In the near future will be done :

* User management - Privileges, Roles
* Error logging
* Fundamental Change - When I'll go through the Spring Security than will come New Version using client side rendering - probably React - Instead of server side rendering - Thymeleaf (Why I used Thymeleaf at the beginning ? - it just happend) . I would like practice using JSON object, using restController instead of Controller and creating Views on client side, authentication using JWT, etc.