VoloCalendar

VoloCalendar is web project for user management, order forecasting, working time scheduling, reporting. The project is developed for Volo company. App based on the example app from the book: Mastering Web Application Development with AngularJS. We changed backend from nodejs to spring mvc, replaced angularjs, angular-ui, bootstrap.css with last variant then developed current app.

Build:

First if you have proxy in your network please change your network to other one without proxy. You need to install Node.js and then the development tools. Node.js comes with a package manager called [npm](http://npmjs.org/) for installing NodeJS applications and libraries.

* Install Intellij IDE 13.1.2+ and open project(if dependencies broken, fix them by adding all jar files in VoloCalendar/lib folder)
* [Install node.js](http://nodejs.org/download/)
* Install Grunt-CLI and Karma as global npm modules from command line:

npm install -g grunt-cli karma

* Install local dependencies (from the project root folder) from command line:

cd VoloCalendar\angular

npm install

(This will install the dependencies declared in the client/package.json file)

* Build client application from command line:

cd VoloCalendar\angular

grunt build --force

* Continuous Building from command line

The watch grunt task will monitor the source files and run the default build task every time a file changes:

cd VoloCalendar\angular

grunt watch --force

* Install postgresql and GUI for it(pgAdminIII)
* Run database script to create schema : Volocalendar/documentation/db.sql
* Set database parameters in VoloCalendar\web\WEB-INF\config\ asanRepositoryContext.xml file
* In VoloCalendar\src\volo\voloCalendar\util\Settings.java set path variables properly(you can take testData.json from VoloCalendar\documentation folder)
* Build project
* Build Artifact(Build/Build Artifacts…/VoloCalendar:war/Clean, Build/Build Artifacts…/VoloCalendar:war/Rebuild)
* Configure one Tomcat server inside Intellij IDE and run project

TODO for future:

All todo tasks in java code, nothing in javascript/html/css. After completion Volo service you can switch to rest api call scenario:

* TODO 1.1: uncomment after adding mail\_location\_id to Driver class by Stefan.
* TODO 1.2: uncomment after adding mail\_location\_id to Driver class by Stefan
* TODO 2: DE is ok?
* TODO 3: set next 2 values accordingly to your file system, do not forget to use proper file separator
* TODO 4: delete these two lines and usages
* TODO 5: comment next uncomment after next
* TODO 6: Order filtering, get only orders up to lowerBound
* TODO 7:comment next line
* TODO 8:uncomment next line
* TODO 9: implement it, we need to get user role on authentication, for example by "isAdmin" boolean header as shown
* TODO 10.1: filtering, filtering should be done by email, name and phone fields, fix after and according server side is done
* TODO 10.2: filtering, filtering should be done by email
* TODO 10.3: filtering, filtering should be done by status
* TODO 11: implement it, mapping between local field name and server side sort parameter desired value
* TODO 12: implement, somehow you should provide api to change password of driver, or you can add password directly to Driver object in put request(update of whole driver)

Contents:

# Documentation folder:

db.sql – script to create database

Backup.postman\_dump – postman dump for testing Volo side rest api

testData.json – lines of date strings that is the dispatched\_at fields of real order data of Volo until 2015 February

documentation.docx – documentation file for developers

manual.docx – user manual and overview for developers

# Lib folder:

All jar dependencies for project

# Web folder:

## Resources folder:

angular.js – standard angular library last version

bootstrap.js – standard angular-ui library last version

bootstrap.css – standard bootstrap style library, but custom changes added to the end of file

jquery.js – standard jquery library last version

mongolab.js – custom library for enlarging JavaScript objects that converted from json, rest methods added to make coding easy as : obj.save/obj.update/obj.remove

angular-app.js - grunt compiled file, that holds all javascript files in angular folder and all html files as javascript strings

## WEB-INF folder:

### Config folder:

log4j.properties – log4j properties file, but currently custom logging does not exist

applicationContext.xml – application context, just container that links other specific configuration file

asanRepositoryContext.xml – spring-data configuration

main-servlet.xml – common configuration for spring controllers

security-app-context.xml – spring security configuration

### views folder:

main.jsp – main page that is returned every time when user write any bookmark able url in address bar of browser and press enter

# angular folder: - holds all javascript/html custom files

gruntFile.js – grunt build file as “maven” for javascript, defines javascript source file strunture and all commands like “build, watch, etc.”

## vendor folder:

holds all standard libraries as angular, angular-ui, jquery and custom library mongolab-resource.js all of them will be copied to web/resources folder on build or watch grunt command execution

## src folder:

### common folder:

#### resources folder:

users.js – defines user javascript class as mongolab-resource(for definition look on the top) class + one additional getFullName method

#### directives folder:

##### crud folder:

edit.js – defines crud-edit directive(CRUD operations for any mangolab-resource object, in our case user object), for usage look at angular/src/app/admin/users/ users-create.tpl.html or users-edit.tpl.html

crudButtons.js – defines crud-buttons directive( just save and revert changes buttons html content) ), for usage look at angular/src/app/admin/users/ users-create.tpl.html or users-edit.tpl.html

crud.js – container for above two

#### security folder:

for explanations please read “Mastering Web Application Development with AngularJS” book , chapter 7: Securing Your Application. All code taken from source code of book and changed little bit according to app logic

#### services folder:

breadcrumbs.js – breadcrumbs implementation

crud.js – some additional functions, for usage look at angular\src\app\admin\users\admin-users-list.js

crudRouteProvider.js – for creating CRUD routes for mongolab-resource class, for usage loof at angular\src\app\admin\users\admin-users.js

exceptionHandler.js – global exception handler

httpRequestTracker.js – defines if there is ongoing request/reply at current moment

notifications.js – notifications to user(for example , Driver deleted/added/updated successfully/unsuccessfully) for current or next page

localizedMessages.js – for localization aims, after translation parametres inserted into proper positions((hello.string, [‘Emin’]) 🡪 (Servus, {name}!, [‘Emin’]) 🡪 Servus, Emin!), key-value pairs saved in angular\src\app\app.js

i18nNotifications.js – enlarges notification.js using localizedMessages.js to create localized notifications, used everywhere in project

utilMethods.js – utility methods for file🡪base64String conversion, week day/month names exchange, object get/set(just save object and get it from memory when you need), colour shading operations

### app folder:

app.js – container module that links all other modules, defines header of page activities and notification activities

header.tpl.html – referenced from main.jsp, controller inside app.js

notifications.tpl.html – referenced from main.jsp, activities inside app.js

#### home folder:

home.js – if current user admin home page is admin/calendar , if driver home page is /calendar , if not authenticated user home page is /info

#### info folder:

just static information page, not required authentication

#### calendar: only for drivers

driver calendar page view/controller/directive

#### admin folder: only for admins

admin.js – container for all admin pages

##### calendar folder:

admin calendar page view/controller/directive

##### forecasting folder:

forecasting page view/controller/directive

##### reporting folder:

reporting page view/controller

##### users folder: taken from “Mastering Web Application Development with AngularJS” book source code and changed according to app logic

user management(user CRUD) pages(all users sorting/filtering/paging, edit user, add user ) view/controller/directive

# src folder:

## META-INF folder:

persistence.xml – entity classes list and some jpa configuration

## volo folder:

### voloCalendar folder:

#### controller folder: all spring controllers

just taking http request, extracting controller parameters and forwarding request to services(\*Logic.java classes)

AdminCalendarController.java – admin calendar activities

DriverCalendarController.java – driver calendar activities

ForecastingController.java – manual and statistical forecasting activities

LoginController.java – get current user, login, logout, update profile(only for drivers) activities

MainController.java – return main.jsp for all bookmark able url entries typed on browser address bar or selected from saved bookmarks, forwards urls starting with “static” to resources folder to get images, fonts and other html resources

ReportsController.java – active drivers by month(monthly) and all users(overview) reports

UserManagementController.java – user sorting, filtering by name/email/phone number, paging, marking as deleted, restoring, adding(with unique email address), editing, viewing activities

#### Dao folder:

All spring data interfaces having methods that either is implemented dynamically by spring or acting as proxy to named queries of jpa level

#### Entity folder:

Persistent objects and enums belonging to them

#### externalModel folder:

All Volo server side objects that is receiving/sending in rest api by Volo backend, used only for converting to/from local objects

#### Security folder:

CustomEntryPoint.java – do not redirect 401 Unauthenticated exceptions to login page as default spring mvc platform does, but just send this exception to client side (angular gear in browser)

#### Service folder: logical operations implemented here

CalendarLogic.java – common logic for driver and admin calendar activities, parent class for DriverCalendarLogic.java and AdminCalendarLogic.java, used in any case if current user is driver or admin

DriverCalendarLogic.java – if current user is driver

AdminCalendarLogic.java – if current user is admin

ForecastingLogic.java – forecasting logic

LoginLogic.java – authenticates current user, used by LoginController.java

ReportLogic.java – reporting logic

UserManagement.java – common interface of two user management services(UserManagementLogic.java and UserManagementLocalLogic.java)

UserManagementLogic.java – implements rest api calls to Volo service

UserManagementLocalLogic.java – default UserManagement implementation, work with local postgresql table User

#### Util folder: helper classes

CalendarUtilMethods.java – utility methods for calendar activities, not connects to database

UserGrantedAuthority.java – holder class for role information of user

UtilMethods.java – json to/from object conversions, authenticating rest api calls, etc.

Settings.java – static values for future pages that will change this values and save them in database not in code

#### viewModel folder:

All models that are not persisted to database

##### Admin folder:

View model classes used in only admin calendar activities

##### Driver folder:

View model classes used in only driver calendar activities

##### Common folder:

View model classes used in both admin and driver calendar activities

##### Forecasting folder:

Forecasting view models

##### User folder:

UserTable.java – holds user action parameters for html table that holds user list in admin/user page, see comments in file for explanations of fields

UserTableItems.java – holds drivers for only one proper page and count of all drivers