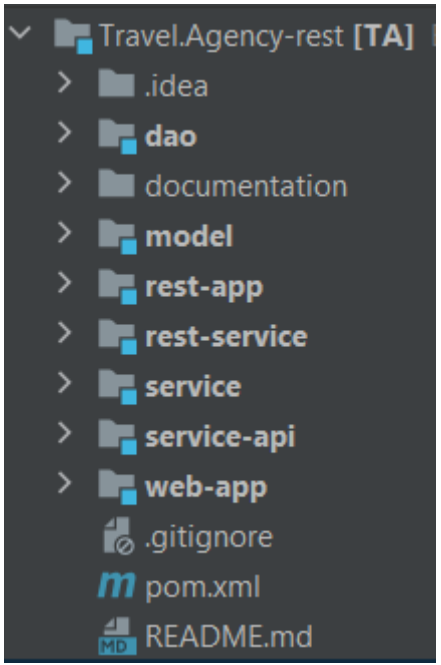
Travel agency

Web-application using Microservice Architecture

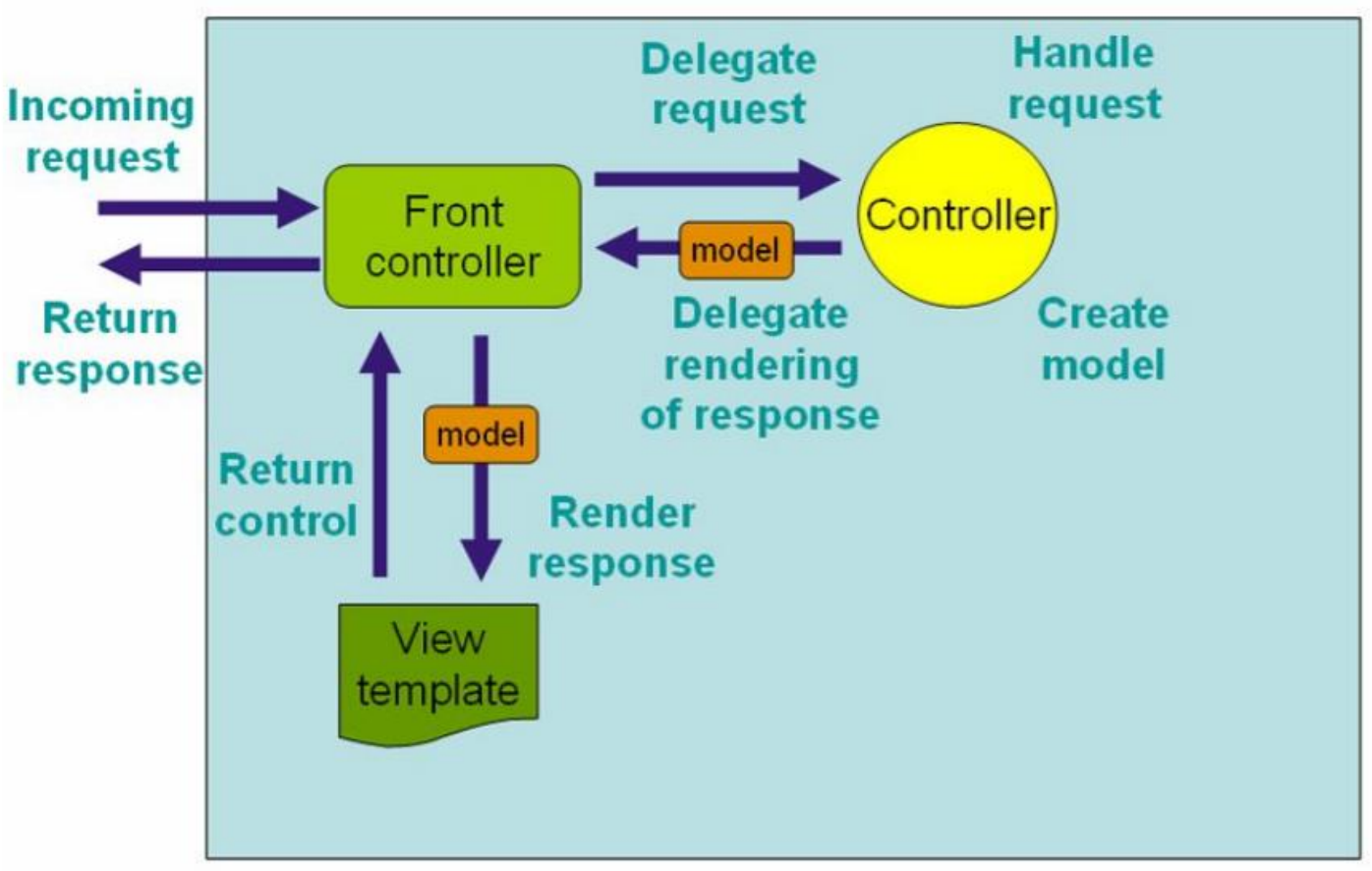
Creator: Siarhei Kakichau

Project structure

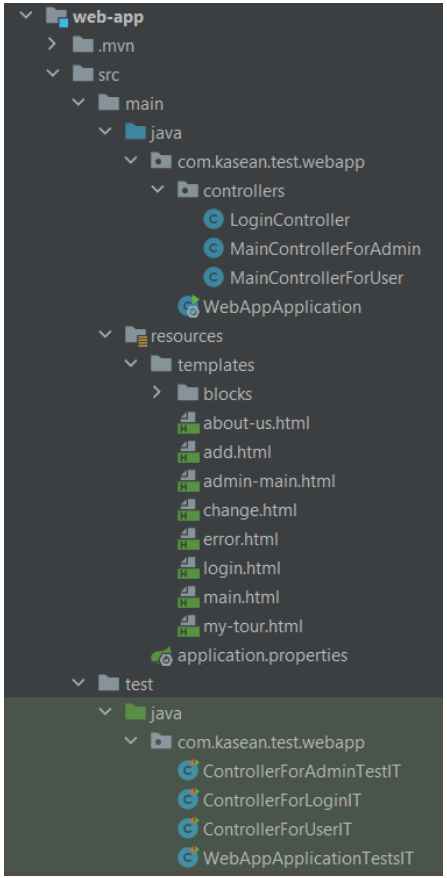


Data access object (DAO) part. A pattern that provides an abstract Interface to some type of database or other persistence mechanism.
Database part included in the rest-app module . Includes liquibase scripts for initialization a configuration
class that connect DB with Spring Boot
Model part. Contains a description of the main entities of the application.
“Server” module handle part. Consists of configuration class that is used to start REST-application (this is where “void main” is placed) and REST-controllers.
Service part. This module is made for feng shui application. Those modules are capable of coordinating web-app/rest-service modules and rest-app controllers/DAO modules interaction.
“Client” - web part. This module is responsible for deploying operations results through user-friendly UI. It refers directly to rest-service module to perform any kind of transactions, as it lacks business logics itself. This module contains configuration class.

Project architecture



Web-app



The controllers package contains the controllers of the web pages

```
@GetMapping("/Add")
public String showAddPage(Model model){
    return "add";
}
```

Class WebAppApplication launches the client side of the application

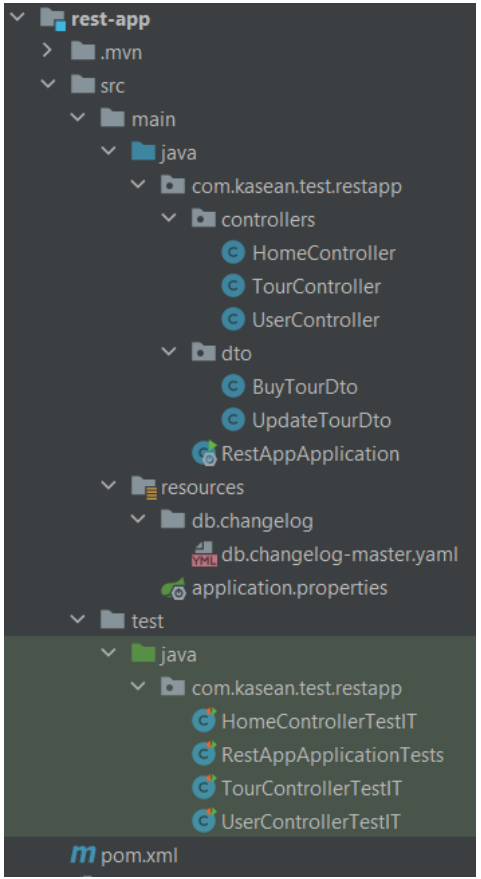
```
@SpringBootApplication
@ComponentScan(basePackages = "com.kasean.test")
public class WebAppApplication{

    public static void main(String[] args) { SpringApplication.run(WebAppApplication.class, args); }

}
```

The templates package contains static HTML pages

application.properties contains the basic settings of the clients side of the application

The test package contains the tests of the web controllers

Rest-app



Rest-app entry point.

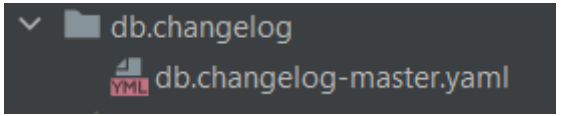
```
@SpringBootApplication
@ComponentScan({"com.kasean.test"})
@EntityScan("com.kasean.test.model")
@EnableJpaRepositories("com.kasean.test.dao")
@RequestMapping(value = "", produces = "application/json")
public class RestAppApplication {

    public static void main(String[] args) { SpringApplication.run(RestAppApplication.class, args); }

}
```

Back-end controllers. Just like web-app ones, those are used to define which services to trigger to get/send data. Response entities are deployed on defined mapping.

```
@PostMapping(path = "/createTour", consumes = "application/json", produces = "application/json")
public ResponseEntity<Long> createTour(@RequestBody Tour tour) {
    //LocalDate tourDate = LocalDate.parse(date);
    Long newTourId = tourService.createTour(tour);
    return new ResponseEntity<>(newTourId, HttpStatus.OK);
}
```

Property file with REST-server configuration (server port).

Database



the database configuration and filling it with test data is in the file db.changelog-master.yaml