Adloid’s Assignment: **WebApp**

PROJECT DESCRIPTION:

This is web application which performs following basic tasks:

1. Register a user and collects following information:

   1.1) Email ID

   1.2) Name

   1.3) Age

   1.4) Occupation

2. Let a user search for his registered information using his email ID as key.

Web Famework used: Spring Boot (v2.4.3)

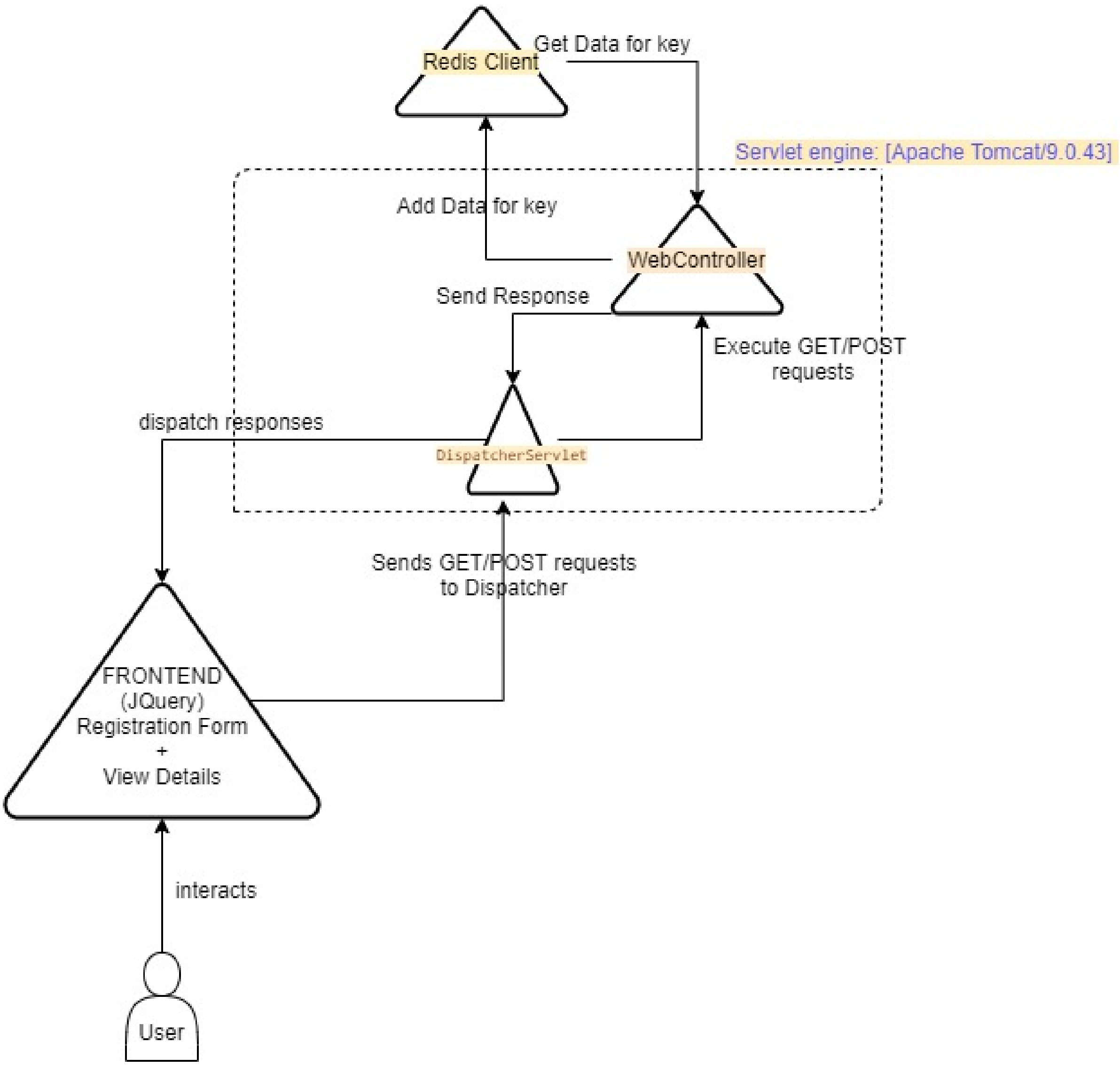
in-memory Data Cache technology used : Redis

Dependencies:

1. spring-boot-starter-web
2. spring-boot-starter-data-redis
3. spring-boot-starter-thymeleaf
4. Java Redis Client Library: Jedis (v3.3.0)

Adloid’s Assignment: **WebApp**

ARCHITECHTURE:


Redis Client
Get Data for key
Servlet engine: [Apache Tomcat/9.0.43]
Add Data for key
WebController
Send Response
Execute GET/POST requests
dispatch responses
DispatcherServlet
Sends GET/POST requests to Dispatcher
FRONTEND (JQuery) Registration Form + View Details
interacts
User

Adloid’s Assignment: **WebApp**

SOURCE CODE DOCUMENTATION:

Class: **WebController**
Access Specifier: public
Annotations: @Controller, @RequestMapping("/")
Desc: This works as Controller (as mentioned in architecture above) . All the http post/get requests are mapped with the methods in this class.

Path: src\main\java\com\adloid\webApp\controller

Methods
1) index()
Params: None
Desc: Mapped with “/ “ directory , loads index.html
2) register()
Params: None
Desc: Mapped with “/register”, loads register.html

3) getDetails()
Params: None
Desc: : Mapped with “/getDetails”, loads getDetails.html

4) getUserDetails(String emailId)
Params: emailID of the user
Desc: This is mapped with “/user” as a GET Method. This takes ‘emailId’ of a user and returns all its data in form of hashMap. RedisRepository is used collect data from redis client.

5) add(String emailID, String name, String age, String occupation)
Param: user details-[user’s emailID, user’s name, user’s age, user’s occupation]
Desc: This is mapped with ‘/add’ as a POST method. This is called when user tries to register its detais. RedisRepository is used to add new users to redis client.

Class: **RedisRepositoryImpl**
Access Specifier: public
Implements: RedisRepository
Annotation: @Repository
Desc: This is a redis repository class , which implements part of CRUD functionality , CREATE and GET

Path: src\main\java\com\adloid\webApp\repository

Methods
1) add(User user)
Params: User object (explained later)

desc: add key value pair to redis client
2) findUser(String emailId)
Params: emailId of the user
desc: Search for values using emailId as Key in redis client.

Class: **User**
Access Specifier: public
implements: Serializable
Fields: String name, String age, String occupation, String emailId

Path: src\main\java\com\adloid\webApp\model

Methods
1) Getters and Setters

Class: **RedisConfig**
Access Specifier: public
Annotation: @Configuration, @ComponentScan("com.adloid.webApp")
Desc: This is configuration class that helps setup connection with redis-server and exposes redisTemplate which is used to interact redis. This helps to facilitate functions for redis cache.

Path: src\main\java\com\adloid\webApp\config

Method
1) jedisConnectionFactory
Params: none
Desc: It returns jedisConnectionFactory object which is used by spring to connect to redis-server. It allows developers to configure various settings for redis-connection, for e.g. hostname , port etc.

2) redisTemplate
Params: none
Desc: this returns redisTemplate, which exposes all the redis functionality required for cache.

Class: ***WebAppApplication***
Desc: This is the exectution point for application. It contains the main method.
Path: src\main\java\com\adloid\webApp

Method
1) Main()
Desc: runs the springApplication.