Date : 05-Jul-21
Spring Boot 9AM
Mr. RAGHU

Spring Boot + MS Mini Project

Setup
Eureka Server : Regsiter & Discover (MS#, Zuul)
Config Server :
  Common key-val for all MS# , provide here.

Q) Does Every MS# need to be register with Eureka or not?
A) YES. using key=val

```
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

Q) Does Every MS# needs Actuator and Admin Server setup?
A) YES. Recomanded even

```
management.endpoints.web.exposure.include=*
spring.boot.admin.client.url=http://localhost:9999
```

+---+
| Stage#2 Externalize Common Properties in MS# |
| Config Server |
+---+

---Config Server - Common key:vals-----------------------------
S#1. Goto Gitlab
https://gitlab.com/raghunit123/my-mini-project-test.git

S#2 Enter un/pwd and login

S#3 Open application.properties

S#4 Edit file and add below keys

```
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
management.endpoints.web.exposure.include=*
spring.boot.admin.client.url=http://localhost:9999
```

S#5 Commit file

*) Run Application in order
a. Eureka Server | Config Server | Admin Server
b. MS# apps
c. Zuul Server
d. Enter this URL:
   http://localhost:9696/cust-api/customers/msg

+---+

```
-----+
|               Stage#3     MS#  Coding Part
|
+-------------------------------------------------------------------
-----+
1. Model class
2. Repository
3. Service
4. RestController

http://localhost:8081/swagger-ui.html

Task: Hibernate Validator API  # Validation API
Download Link:
https://www.mediafire.com/file/w5x9w5vcmkwkkdv/RaghuSirNareshITJavaPdf
s.zip/file

*) Extract and Open : HibernateRaghuSirNareshIT.pdf
       Goto Concept :  Validation API in Hibernate
       Page         : 212


==Coding Part : MS# Customer=========================
1. Model
package in.nareshit.raghu.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;

@Data
@Entity
@Table(name="custtb")
public class Customer {
        @Id
        @Column(name="cid")
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Integer custId;

        @Column(name="cname")
        private String custName;

        @Column(name="cmail")
        private String custMail;

        @Column(name="cmob")
        private String custMobile;

        @Column(name="caddr")
        private String custAddr;

        @Column(name="cstatus")
```

```
	private String status;

}
```

2. Repository

```
package in.nareshit.raghu.repo;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;

import in.nareshit.raghu.model.Customer;

public interface CustomerRepository
	extends JpaRepository<Customer, Integer> {

	Optional<Customer> findByCustMobile(String custMobile);
	//task : define load customer by email

	@Modifying
	@Query("UPDATE Customer SET status=:status WHERE custId=:id")
	Integer updateCustomerStatus(Integer id,String status);

}
```

3. Service

```
package in.nareshit.raghu.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import in.nareshit.raghu.consts.CustomerStatus;
import in.nareshit.raghu.exception.CustomerNotFoundException;
import in.nareshit.raghu.model.Customer;
import in.nareshit.raghu.repo.CustomerRepository;

@Service
public class CustomerService {

	@Autowired
	private CustomerRepository repo;

	public Integer saveCustomer(Customer cust) {
		cust.setStatus(CustomerStatus.ACTIVE.name());
		return repo.save(cust).getCustId();
	}

	public Customer getCustomer(Integer id) {
		return repo.findById(id)
				.orElseThrow(
					()->new
CustomerNotFoundException(

"Cusomer Not Exist")
```

```
                                                );
        }

        public Customer getCustomerByMobile(String custMobile) {
                return  repo.findByCustMobile(custMobile).orElseThrow(
                                ()->new CustomerNotFoundException(
                                                "Cusomer Not Exist")
                                );
        }

        //task : define load customer by email
        @Transactional
        public Integer updateStatus(Integer id,String status) {
                return repo.updateCustomerStatus(id, status);
        }
}
```

4. RestController

```
package in.nareshit.raghu.rest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import in.nareshit.raghu.consts.CustomerStatus;
import in.nareshit.raghu.model.Customer;
import in.nareshit.raghu.service.CustomerService;

@RestController
@RequestMapping("/rest/customers")
public class CustomerRestController {

        @Autowired
        private CustomerService service;

        @PostMapping("/create")
        public ResponseEntity<String> saveCustomer(
                        @RequestBody Customer customer
                        )
        {
                Integer id  = service.saveCustomer(customer);
                return ResponseEntity.ok("Customer '"+id+"' is
created");
        }

        @GetMapping("/find/id/{id}")
        public ResponseEntity<Customer> getCustomerById(
                        @PathVariable Integer id
                        )
        {
                return ResponseEntity.ok(
```

```
                                service.getCustomer(id)
                                );
        }

        @GetMapping("/find/contact/{contact}")
        public ResponseEntity<Customer> getCustomerByContact(
                        @PathVariable String contact
                        )
        {
                return ResponseEntity.ok(
                                service.getCustomerByMobile(contact)
                                );
        }

        @PatchMapping("/activate/{id}")
        public ResponseEntity<String> activateCustomer(
                        @PathVariable Integer id
                        )
        {
                service.updateStatus(id,
CustomerStatus.ACTIVE.name());
                return ResponseEntity.ok("Customer '"+id+"' is
Activated!!");
        }

        @PatchMapping("/inactivate/{id}")
        public ResponseEntity<String> inActivateCustomer(
                        @PathVariable Integer id
                        )
        {
                service.updateStatus(id,
CustomerStatus.INACTIVE.name());
                return ResponseEntity.ok("Customer '"+id+"' is
InActivated!!");
        }
}
```

5. Custom Exception

```
package in.nareshit.raghu.exception;

public class CustomerNotFoundException extends RuntimeException {

        private static final long serialVersionUID = 1L;

        public CustomerNotFoundException() {
                super();
        }

        public CustomerNotFoundException(String message) {
                super(message);
        }
}
```

6. Enum for Status

```
package in.nareshit.raghu.consts;
```

```
public enum CustomerStatus {

	ACTIVE, INACTIVE

}
```

7. Swagger Config

```
package in.nareshit.raghu.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

	@Bean
	public Docket createDocket() {
		return new Docket(DocumentationType.SWAGGER_2)
				.select()

.apis(RequestHandlerSelectors.basePackage("in.nareshit.raghu.rest"))
				.paths(PathSelectors.regex("/rest.*"))
				.build()
				;

	}
}
```

8. pom.xml for swagger

```
		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger-ui</artifactId>
			<version>2.9.2</version>
		</dependency>

		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger2</artifactId>
			<version>2.9.2</version>
		</dependency>
```

Task#2 Email Util

On register, send email to customer -- welcome message

```
package in.nareshit.raghu.util;

import java.io.InputStream;
```

```
import javax.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Component;
import org.springframework.web.multipart.MultipartFile;

import in.nareshit.raghu.model.WhUserType;

@Component
public class MailUtil {

	@Autowired
	private JavaMailSender mailSender;


	public boolean sendEmail(
				String to,
				String[] cc,
				String[] bcc,
				String subject,
				String text,
				MultipartFile file
				)
	{
		boolean flag = false;
		try {
				//1. create one empty email (mime message)
				MimeMessage message =
mailSender.createMimeMessage();

				//2. fill details
				MimeMessageHelper helper = new
MimeMessageHelper(message, file!=null);

				helper.setTo(to);
				if(cc!=null && cc.length>0)
					helper.setCc(cc);
				if(bcc!=null && bcc.length>0)
					helper.setBcc(bcc);

				helper.setSubject(subject);
				helper.setText(text,true);

				// add attachment
				if(file!=null)

helper.addAttachment(file.getOriginalFilename(), file);

				//3. send email
				mailSender.send(message);
				flag = true;
		} catch (Exception e) {
				e.printStackTrace();
		}
```

```
                return flag;
        }


}
```