## **Description** :

**Relation Ship :**

This is E-commerce system as my part implementing Payment Microservice. Payment-service include an ID, buyerID, stockID, sellerID, Price, salles Amount. Therefor I creating payment-service have to integrate and create tree microservice . those are

* Stock-service – {ID, Name, Price, Available Amount }
* Buyer-service – {ID, Name, Phone}
* Seller-service – {ID, Name, Phone}

**Technology :**

* To register a Microservice Swagger 2 Rest API,
* Routing a Microservice use Zuul proxy
* Database use Hibernate ORM.

**Post Method Scenario** :

The payment-service and others service are integrate using post method(use create method to integrate payment, stock, seller, buyer services). The creating a payment is getting value :

* ID : user input
* stockID : stock-service
* buyerID : buyer-service
* sellerID : seller-service
* Price : seller-service
* salles Amount : user input

**A picture containing screenshot

Description automatically generated**

**Controller Class**

4). Payment-Server

package com.mtit.assignment.paymentservice.controller.impl;  
import com.fasterxml.jackson.databind.exc.MismatchedInputException;  
import com.mtit.assignment.paymentservice.entity.\*;  
import com.mtit.assignment.paymentservice.Service.impl.PaymentServiceImpl;  
import com.mtit.assignment.paymentservice.Service.impl.massageImpl;  
import com.mtit.assignment.paymentservice.controller.paymentController;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.web.client.RestTemplateBuilder;  
import org.springframework.context.annotation.Bean;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.MediaType;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.client.RestTemplate;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.UUID;  
  
@CrossOrigin  
@RestController  
@RequestMapping("api/v1/payment")  
public class PaymentControllerImpl implements paymentController {  
  
 @Autowired  
 private PaymentServiceImpl paymentService;  
  
 @Autowired  
 RestTemplate restTemplate;  
  
 @PostMapping(consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ResponseStatus(HttpStatus.*CREATED*)  
 @Override  
 public List<CreateReturnValue> Create(@RequestBody ConnectionEntity conE) throws MismatchedInputException {  
  
 massageImpl ms = new massageImpl();  
 List returnList = new ArrayList();

// Stock Create post method   
 ResponseEntity<Stock> t = restTemplate.postForEntity(

"http://localhost:8000/stockService/api/v1/stock"

,new Stock( conE.getStockId(),

conE.getPrice(),

conE.getAvailable())

, Stock.class );

// Buyer Create post method  
 ResponseEntity<Buyer> s = restTemplate.postForEntity(

"http://localhost:8000/buyerService/api/v1/buyer"

,new Buyer(conE.getBuyerId(),

conE.getBuyerName(),

conE.getBuyerPhone())

,Buyer.class );

// Seller Create post method  
 ResponseEntity<Seller> b = restTemplate.postForEntity(

"http://localhost:8000/sellerService/api/v1/seller",

new Seller(conE.getSellerId(),

conE.getSellerName(),

conE.getSellerPhone())

, Seller.class );

Payment payment = new Payment( conE.getId()

,s.getBody().getId()

,b.getBody().getId()

,t.getBody().getId()

,t.getBody().getPrice()

,t.getBody().getAvailable()

);  
 paymentService.create(payment);  
  
 returnList.add( NewCreateReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("Create"),

payment,

conE)

);  
  
 return returnList;  
 }

@GetMapping  
 @Override  
 public ResponseEntity<List<ReturnValue>> findAll() {  
 massageImpl ms = new massageImpl();  
 List list = paymentService.findAll();  
 System.*out*.println(list.size());  
 HttpHeaders httpHeaders = new HttpHeaders();  
 httpHeaders.add("X-Count",list.size() + "");  
 List returnList = new ArrayList();  
 returnList.add(new ReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("FindAll"),

list)

);  
 return new ResponseEntity<List<ReturnValue>>(returnList,httpHeaders,

HttpStatus.*OK*);  
 }  
  
 @Override  
 @PutMapping(value = "/{id}",consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 public List<ReturnValue> update(@PathVariable("id") String id, @RequestBody

Payment payment) {

System.*out*.println(id);  
 massageImpl ms = new massageImpl();  
 List returnList = new ArrayList();  
 List list = new ArrayList();  
 if(payment.getId().equals(id)){  
 paymentService.update(payment);  
 list.add(payment);  
 returnList.add(new returnList.add(new ReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("Update"),

l list)

);

}else{  
 returnList.add(new returnList.add(new ReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("Error"),

l list)

);  
 }  
 return returnList;  
 }  
  
 @Override  
 @DeleteMapping("/{id}")  
 @ResponseStatus(HttpStatus.*OK*)  
 public List<ReturnValue> delete(@PathVariable("id") String id) {  
 System.*out*.println(id);  
 massageImpl ms = new massageImpl();  
 paymentService.delete(id);  
 List returnList = new ArrayList();  
 returnList.add(new ReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("Delete"))

);  
 return returnList;  
 }  
  
 @Override  
 @GetMapping("/{id}")  
 public List<ReturnValue> find(@PathVariable("id") String id) {  
 massageImpl ms = new massageImpl();  
 System.*out*.println(id);  
 List list = paymentService.find(id);  
 List returnList = new ArrayList();  
 returnList.add(new returnList.add(new ReturnValue(

UUID.*randomUUID*().toString(),

ms.successfulGeneratedMessage("Update"),

l list)

);

return returnList;  
 }  
  
 @Bean  
 public RestTemplate restTemplate(RestTemplateBuilder builder){  
 return builder.build();  
 }  
}

## **Project Folder Structure**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Postman, Console Test Out Put:**

1. Payment

**A screenshot of a computer

Description automatically generated**

1. Buyer

A screenshot of a computer

Description automatically generated

1. Seller

A screenshot of a computer

Description automatically generated

1. Stock

A screenshot of a computer

Description automatically generated

1. Zuul proxy

A screenshot of a computer screen

Description automatically generated

1. Postman

A screenshot of a computer screen

Description automatically generated

A screenshot of a social media post

Description automatically generated

1. XML Out Put:
   1. Buyer

A screenshot of a cell phone

Description automatically generated

* 1. Seller

A screenshot of a cell phone

Description automatically generated

* 1. Stock

A screenshot of a cell phone

Description automatically generated

* 1. Payment

A screenshot of a cell phone

Description automatically generated