# Gateway Aggregator pattern

Gateway Aggregator pattern intro

Use case

<http://localhost:7070/swagger-ui/>

As part of the aggregator pattern, we must call below APIs simultaneously. Product page has all the necessary information ready

1. <http://localhost:7070/sec01/product/1> Provides the product details for the given product id (up to 50)
2. <http://localhost:7070/sec01/promotion/1> Product id 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 are not present in this service
3. <http://localhost:7070/sec01/review/1> Product id 10, 20, 30, 40, 50 are not present in this service. Product Id 7 has some weird issues

We use ProductAggregatorService to call all the API using ZIP function

public Mono<ProductAggregate> aggregate(Integer id){  
 return Mono.zip(  
 this.productClient.getProduct(id),  
 this.promotionClient.getPromotion(id),  
 this.reviewClient.getReviews(id)  
 )  
 .map(t -> toDto(t.getT1(), t.getT2(), t.getT3()));  
}

Error handling

Since the /product/id API is the main API and if fails there is nothing to send to user hence we return Mono.empty()

public Mono<ProductResponse> getProduct(Integer id){  
 return this.client  
 .get()  
 .uri("{id}", id)  
 .retrieve()  
 .bodyToMono(ProductResponse.class)  
 .onErrorResume(ex -> Mono.empty());  
}

Then we handle the empty in controller

@RestController  
@RequestMapping("sec01")  
public class ProductAggregateController {  
  
 @Autowired  
 private ProductAggregatorService service;  
  
 @GetMapping("product/{id}")  
 public Mono<ResponseEntity<ProductAggregate>> getProductAggregate(@PathVariable Integer id){  
 return this.service.aggregate(id)  
 .map(ResponseEntity::ok)  
 .defaultIfEmpty(ResponseEntity.notFound().build());  
 }  
  
}

For promotion if there is no promotion is present for Product id 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 and it will return 404, so we need to handle it as Zip will expect tuple. So we need to handle it

public Mono<PromotionResponse> getPromotion(Integer id){  
 return this.client  
 .get()  
 .uri("{id}", id)  
 .retrieve()  
 .bodyToMono(PromotionResponse.class)  
 .onErrorReturn(noPromotion);  
}

For review also we have similar issues where Product id 10, 20, 30, 40, 50 are not present in this service. Product Id 7 has some weird issues. So we need to handle error.

public Mono<List<Review>> getReviews(Integer id){  
 return this.client  
 .get()  
 .uri("{id}", id)  
 .retrieve()  
 .bodyToFlux(Review.class)  
 .collectList() // Will collect to List in non blocking way  
 .onErrorReturn(Collections.emptyList());  
}

# Scatter gather pattern

# Orchestrator Pattern

27 Orchestrator pattern intro

External services

1. <http://localhost:7070/sec03/product/1> return product result with price
2. <http://localhost:7070/sec03/user/1> return user details with balance
3. <http://localhost:7070/sec03/user/deduct> POST API where we deduct the amount from user balance