**Micro Services**

**1.post**

package com.microservice.post.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
  
@Configuration  
public class RestTemplateConfig {  
 @Bean  
 public RestTemplate getRestTemplate(){  
 return new RestTemplate();  
 }  
}

package com.microservice.post.controller;  
  
  
import com.microservice.post.entity.Post;  
import com.microservice.post.payLoad.PostDto;  
import com.microservice.post.repository.PostRepository;  
import com.microservice.post.service.PostService;  
  
  
import io.github.resilience4j.circuitbreaker.annotation.CircuitBreaker;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
  
  
  
@RestController  
@RequestMapping("/api/posts")  
  
public class PostController{  
 @Autowired  
 private PostService postService;  
 @PostMapping  
 public ResponseEntity<Post> savePost(@RequestBody Post post) {  
 Post newPost = postService.savePost(post);  
 return new ResponseEntity<>(newPost, HttpStatus.*CREATED*);  
 }  
  
@GetMapping("/{postId}")  
public Post getPostByPostId(@PathVariable String postId){  
 Post post = postService.findPostById(postId);  
return post;  
  
}  
 @GetMapping("/{postId}/comments")  
 @CircuitBreaker(name = "commentBreaker", fallbackMethod = "commentFallback")  
  
  
 public ResponseEntity<PostDto> getPostWithComments(@PathVariable String postId){  
  
 PostDto postDto = postService.getPostWithComments(postId);  
  
 return new ResponseEntity<>(postDto, HttpStatus.*OK*);  
 }  
  
  
 public ResponseEntity<PostDto> commentFallback(String postId, Exception ex) {  
  
 System.*out*.println("Fallback is executed because service is down : "+ ex.getMessage());  
  
 *// ex.printStackTrace();* PostDto dto = new PostDto();  
 dto.setPostId("1234");  
 dto.setTitle("Service Down");  
 dto.setContent("Service Down");  
 dto.setDescription("Service Down");  
  
 return new ResponseEntity<>(dto, HttpStatus.*BAD\_REQUEST*);  
  
}}

package com.microservice.post.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
  
  
  
@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
@Table(name="posts")  
@Data  
public class Post {  
  
 @Id  
 private String id;  
 private String title;  
 private String description;  
 private String content;  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
}

package com.microservice.post.payLoad;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class Comment {  
 private String CommentId;  
 private String name;  
 private String email;  
 private String body;  
 private String postId;  
  
 public String getCommentId() {  
 return CommentId;  
 }  
  
 public void setCommentId(String commentId) {  
 CommentId = commentId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getBody() {  
 return body;  
 }  
  
 public void setBody(String body) {  
 this.body = body;  
 }  
  
 public String getPostId() {  
 return postId;  
 }  
  
 public void setPostId(String postId) {  
 this.postId = postId;  
 }  
}

package com.microservice.post.payLoad;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import java.util.List;  
  
  
@AllArgsConstructor  
@NoArgsConstructor  
@Data  
public class PostDto {  
 private String PostId;  
 private String title;  
 private String description;  
 private String content;  
  
 public String getPostId() {  
 return PostId;  
 }  
  
 public void setPostId(String postId) {  
 PostId = postId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 public List<Comment> getComments() {  
 return comments;  
 }  
  
 public void setComments(List<Comment> comments) {  
 this.comments = comments;  
 }  
  
 List<Comment> comments;  
}

package com.microservice.post.repository;  
  
import com.microservice.post.entity.Post;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
  
  
public interface PostRepository extends JpaRepository<Post,String> {  
  
}

package com.microservice.post.service;  
  
import com.microservice.post.config.RestTemplateConfig;  
import com.microservice.post.entity.Post;  
import com.microservice.post.payLoad.PostDto;  
import com.microservice.post.repository.PostRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Service;  
  
import java.util.ArrayList;  
  
import java.util.UUID;  
  
  
@Service  
public class PostService {  
  
 @Autowired  
 private PostRepository postRepository;  
  
 @Autowired  
 private RestTemplateConfig restTemplate;  
 public Post savePost(Post post){  
 String postId = UUID.*randomUUID*().toString();  
 post.setId(postId);  
 Post savedPost = postRepository.save(post);  
 return savedPost;  
 }  
  
 public Post findPostById(String postId) {  
 Post post = postRepository.findById(postId).get();  
 return post;  
 }  
  
 public PostDto getPostWithComments(String postId) {  
 Post post = postRepository.findById(postId).get();  
 ArrayList comments = restTemplate.getRestTemplate().getForObject("http://localhost:8082/api/comments/" + postId, ArrayList.class);  
 PostDto postDto=new PostDto();  
 postDto.setPostId(post.getId());  
 postDto.setTitle(post.getTitle());  
 postDto.setDescription(post.getDescription());  
 postDto.setContent(post.getContent());  
 postDto.setComments(comments);  
 return postDto;  
   
  
 }  
}

package com.microservice.post;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class PostApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(PostApplication.class, args);  
 }  
  
}

**2.comment**

package com.microservice.Comment.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
  
@Configuration  
public class RestTemplateConfig {  
 @Bean  
 public RestTemplate getRestTemplate(){  
 return new RestTemplate();  
 }  
}

package com.microservice.Comment.controller;  
  
import com.microservice.Comment.entity.Comment;  
import com.microservice.Comment.service.CommentService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/comments")  
public class CommentController {  
 @Autowired  
 private CommentService commentService;  
 *//http://localhost:8082/api/comments* @PostMapping  
 public ResponseEntity<Comment> saveComment(@RequestBody Comment comment){  
 Comment c = commentService.saveComment(comment);  
 return new ResponseEntity<>(c, HttpStatus.*OK*) ;  
 }  
 @GetMapping("{postId}")  
 public List<Comment> getAllCommentByPostId(@PathVariable String postId) {  
 List<Comment> comments = commentService.getAllCommentByPostId(postId);  
 return comments;  
 }  
  
}

package com.microservice.Comment.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
  
  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
@Table(name="comments")  
public class Comment {  
 @Id  
 private String commentId;  
 private String name;  
  
  
  
 private String email;  
 private String body;  
 private String postId;  
  
  
 public String getCommentId() {  
 return commentId;  
 }  
  
 public void setCommentId(String commentId) {  
 this.commentId = commentId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getBody() {  
 return body;  
 }  
  
 public void setBody(String body) {  
 this.body = body;  
 }  
  
 public String getPostId() {  
 return postId;  
 }  
  
 public void setPostId(String postId) {  
 this.postId = postId;  
 }  
}

package com.microservice.Comment.payload;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@NoArgsConstructor  
@AllArgsConstructor  
@Data  
public class Post {  
 private String PostId;  
 private String title;  
 private String description;  
 private String Content;  
  
  
 public String getPostId() {  
 return PostId;  
 }  
  
 public void setPostId(String postId) {  
 PostId = postId;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public String getContent() {  
 return Content;  
 }  
  
 public void setContent(String content) {  
 Content = content;  
 }  
  
  
}

package com.microservice.Comment.repository;  
  
import com.microservice.Comment.entity.Comment;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.List;  
  
public interface CommentRepository extends JpaRepository<Comment,String> {  
 List<Comment> findByPostId(String postId);  
}

package com.microservice.Comment;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
  
@SpringBootApplication  
  
public class CommentApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CommentApplication.class, args);  
 }  
  
}

spring:  
 application:  
 name: COMMENT-SERVICE  
 datasource:  
 url: jdbc:mysql://localhost:3306/commentdb  
 username: root  
 password: test  
  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true  
 properties:  
 hibernate:  
 dialect: org.hibernate.dialect.MySQL8Dialect  
  
server:  
 port: 8082  
  
eureka:  
 instance:  
 prefer-ip-address: true  
 client:  
 fetch-registry: true  
 register-with-eureka: true  
 service-url:  
 defaultZone: http://localhost:8761/eureka  
  
  
management:  
 health:  
 circuitBreakers:  
 enabled: true  
 endpoints:  
 web:  
 exposure:  
 include: health  
  
 endpoint:  
 health:  
 show-details: always  
  
  
  
resilience4j:  
 circuitBreaker:  
 instances:  
 commentBreaker:  
 registerHealthIndicator: true  
 eventConsumerBufferSize: 10  
 failureRateThreshold: 50  
 minimumNumberOfCalls: 5  
 automaticTransitionFromOpenToHalfOpenEnabled: true  
 waitDurationInOpenState: 6s  
 permittedNumberOfCallsInHalfOpenState: 3  
 slidingWindowSize: 10  
 slidingWindowType: COUNT\_BASED  
  
 retry:  
 instances:  
 commentBreakerService:  
 max-attempts: 3  
 wait-duration: 5s

**3.Gateway**

package com.apigateway.gateway;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
  
@SpringBootApplication  
@EnableEurekaClient  
public class GatewayApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(GatewayApplication.class, args);  
 }  
  
}

server:  
 port: 8083  
  
spring:  
 application:  
 name: API-GATEWAY  
  
  
 cloud:  
 gateway:  
 routes:  
 - id: POST-SERVICE  
 uri: lb://POST-SERVICE  
 predicates:  
 - Path=/api/post/\*\*  
  
 - id: COMMENT-SERVICE  
 uri: lb://COMMENT-SERVICE  
 predicates:  
 - Path=/api/comments/\*\*  
  
  
  
  
eureka:  
 instance:  
 prefer-ip-address: true  
 client:  
 fetch-registry: true  
 register-with-eureka: true  
 service-url:  
 defaultZone: http://localhost:8761/eureka

**4. Gateway**

package com.apigateway.gateway;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;  
  
@SpringBootApplication  
@EnableEurekaClient  
public class GatewayApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(GatewayApplication.class, args);  
 }  
  
}

server:  
 port: 8083  
  
spring:  
 application:  
 name: API-GATEWAY  
  
  
 cloud:  
 gateway:  
 routes:  
 - id: POST-SERVICE  
 uri: lb://POST-SERVICE  
 predicates:  
 - Path=/api/post/\*\*  
  
 - id: COMMENT-SERVICE  
 uri: lb://COMMENT-SERVICE  
 predicates:  
 - Path=/api/comments/\*\*  
  
  
  
  
eureka:  
 instance:  
 prefer-ip-address: true  
 client:  
 fetch-registry: true  
 register-with-eureka: true  
 service-url:  
 defaultZone: http://localhost:8761/eureka