服务间通信与负载均衡

作业编号：`hw08`

贯穿项目里程碑

本周任务是在 hw07 的服务注册与发现基础上，使用 OpenFeign 实现服务间通信，并通过多实例部署验证负载均衡效果，同时添加基本的容错保护机制。

版本信息：

- 项目名称：`course-cloud`
- 版本号：`v1.2.0`（服务间通信与负载均衡）
- 基于版本：`v1.1.0`

系统架构说明

当前系统已集成 Nacos 作为服务注册中心。本周将在 Enrollment Service 中使用 OpenFeign 调用 User Service 和 Catalog Service，替代之前的 RestTemplate 方式，并配置基本的熔断降级机制。

核心任务

- 在 Enrollment Service 中集成 OpenFeign，替换现有的 RestTemplate 调用方式
- 创建 UserClient 和 CatalogClient 接口
- 实现 Fallback 降级处理
- 配置 Resilience4j 熔断器（失败率阈值 50%，滑动窗口 10 次）
- 使用 Docker Compose 启动 User Service 和 Catalog Service 的多个实例（每个服务 3 个实例）
- 验证负载均衡效果
- 完成后执行 `git tag v1.2.0`

任务详解

1. OpenFeign 依赖配置

在 Enrollment Service 的 `pom.xml` 中添加：

```
<!-- OpenFeign -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>

<!-- Resilience4j 熔断器 -->
```

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>
</dependency>
```

2. 启用 Feign 客户端

在主应用类上添加 `@EnableFeignClients` 注解（参考课件示例）。

3. 创建 Feign Client 接口

创建 `UserClient.java` 和 `CatalogClient.java`，以下为 UserClient 示例：

```java
@FeignClient(
    name = "user-service",
    fallback = UserClientFallback.class
)
public interface UserClient {
    @GetMapping("/api/users/students/{id}")
    StudentDto getStudent(@PathVariable Long id);
}
```

请参考此示例自行实现 `CatalogClient`，调用 `/api/courses/{id}` 获取课程信息。

4. 实现 Fallback 类

为每个 Feign Client 实现 Fallback 降级处理。以下为 `UserClientFallback` 示例：

```java
@Component
@Slf4j
public class UserClientFallback implements UserClient {
    @Override
    public StudentDto getStudent(Long id) {
        log.warn("UserClient fallback triggered for student: {}", id);
        throw new ServiceUnavailableException("用户服务暂时不可用，请稍后再试");
    }
}
```

请自行实现 `CatalogClientFallback`。

5. 配置文件

在 `application.yml` 中添加 Feign 和 Resilience4j 配置（参考课件中的配置示例）：

核心配置项：

- `feign.circuitbreaker.enabled: true` - 启用熔断器
- `feign.client.config.default.connectTimeout` - 连接超时（建议 3 秒）
- `feign.client.config.default.readTimeout` - 读取超时（建议 5 秒）

- `resilience4j.circuitbreaker.instances.{service-name}` - 为每个服务配置熔断器参数

请根据作业要求（失败率阈值 50%，滑动窗口 10 次）自行完成配置。

6. 使用 Feign Client

在 `EnrollmentService` 中注入并使用 `UserClient` 和 `CatalogClient`，参考课件中的使用示例。

7. 多实例部署配置

更新 `docker-compose.yml`，为 User Service 和 Catalog Service 各配置 3 个实例。

关键配置要点：

- 每个服务实例使用不同的 `container_name`（如 `user-service-1`、`user-service-2`、`user-service-3`）
- 所有实例使用相同的 `image` 和 `SERVER_PORT`
- 配置 `SPRING_CLOUD_NACOS_DISCOVERY_SERVER_ADDR=nacos:8848`
- 确保所有服务在同一 Docker 网络中
- Enrollment Service 只需 1 个实例，并暴露端口 8083

请参考课件和之前作业的 docker-compose.yml 格式自行完成配置。

8. 验证负载均衡

在各服务的 Controller 中添加日志，输出实例标识（如端口号或容器名），以便观察负载均衡效果。

提示：使用 `@Value("${server.port}")` 注入端口号，在处理请求时打印日志。

测试要求

负载均衡测试

1. 启动所有服务：`docker-compose up -d`
2. 访问 Nacos 控制台确认所有服务实例已注册
3. 通过 Enrollment Service 连续发送多次选课请求
4. 查看各服务日志，确认请求分配到不同实例

熔断降级测试

1. 停止所有 User Service 实例
2. 发送选课请求，观察 fallback 是否触发
3. 查看日志确认降级处理被调用
4. 重启服务，验证恢复正常

代码位置建议

```
course-cloud/
├── services/
```

```
│   ├── enrollment-service/
│   │   ├── src/main/java/com/zjgsu/coursecloud/enrollment/
│   │   │   ├── client/
│   │   │   │   ├── UserClient.java
│   │   │   │   ├── UserClientFallback.java
│   │   │   │   ├── CatalogClient.java
│   │   │   │   └── CatalogClientFallback.java
│   │   │   ├── dto/
│   │   │   │   ├── StudentDto.java
│   │   │   │   └── CourseDto.java
│   │   │   └── service/
│   │   │       └── EnrollmentService.java
│   │   └── src/main/resources/
│   │       └── application.yml
│   ├── user-service/
│   └── catalog-service/
├── docker-compose.yml
└── scripts/
    └── test-load-balance.sh
```

提交要求

代码要求

- ☐ OpenFeign 集成正确，能够成功调用其他服务
- ☐ Fallback 降级处理实现完整
- ☐ 多实例部署配置正确，能够启动并注册到 Nacos
- ☐ 熔断器配置生效，能够在服务故障时触发

文档要求

在 `docs/week08-notes.md` 中记录：

- ☐ OpenFeign 配置说明
- ☐ 负载均衡测试结果（包含日志截图，显示不同实例处理请求）
- ☐ 熔断降级测试结果（包含日志截图，显示 fallback 触发）
- ☐ OpenFeign vs RestTemplate 对比分析

测试证明

- ☐ 完整的负载均衡测试日志
- ☐ 熔断降级测试截图
- ☐ Nacos 控制台显示多实例注册的截图

评分标准

- OpenFeign 集成正确 (30%)
- Feign Client 实现 (25%)

- 熔断降级配置 (30%)
- 多实例部署 (10%)
- 文档和测试 (5%)