# 服务注册与发现

作业编号：hw07

## 贯穿项目里程碑

本周任务是在第 06 周的容器化项目基础上,集成 Nacos 实现服务的自动注册与发现,替代硬编码的服务地址。

### 核心任务

- 使用 Docker 启动 Nacos 服务器(单机模式)
- 为 User Service, Catalog Service, 和 Enrollment Service 添加 Nacos Discovery 依赖
- 配置服务自动注册到 Nacos(应用名、服务端口、命名空间等)
- 修改 Enrollment 服务,使用 DiscoveryClient + RestTemplate 通过服务名调用 User Service 和 Catalog Service,不再使用硬编码地址
- 配置健康检查接口,确保 Nacos 能够准确监控服务状态
- 在 Nacos 控制台验证服务注册成功,查看实例列表和健康状态
- 启动多个服务实例,验证负载均衡是否生效
- 提交后执行 `git tag v1.1.0`

### Nacos 配置要求

在 `application.yml` 中添加 Nacos 配置:

```yaml
spring:
  application:
    name: catalog-service  # 服务名: user-service, catalog-service, 或
enrollment-service
  cloud:
    nacos:
      discovery:
        server-addr: nacos:8848
        namespace: dev
        group: COURSEHUB_GROUP
        ephemeral: true
        heart-beat-interval: 5000
        heart-beat-timeout: 15000
```

### Docker Compose 更新要求

更新 `docker-compose.yml` 文件,添加 Nacos 服务:

```yaml
services:
  nacos:
    image: nacos/nacos-server:v3.1.0
    container-name: nacos
    environment:
      - MODE=standalone
    ports:
      - "8848:8848"
      - "9848:9848"
    networks:
      - course-cloud-network
```

同时更新 User Service, Catalog Service 和 Enrollment Service 的配置,使其依赖 Nacos:

```yaml
  user-service:
    depends_on:
      - nacos
      - mysql

  catalog-service:
    depends_on:
      - nacos
      - mysql

  enrollment-service:
    depends_on:
      - nacos
      - mysql
```

### 代码位置建议

- Nacos 配置放在各服务的 `src/main/resources/application.yml`
- 更新后的 `docker-compose.yml` 在项目根目录
- 健康检查接口可使用 Spring Boot Actuator 的 `/actuator/health`

## 课后练习

### 服务发现实验

完成以下实验并记录观察结果:

1. **单实例测试**:启动一个 User 服务实例和一个 Catalog 服务实例,在 Nacos 控制台查看实例信息
2. **多实例测试**:启动 3 个 User 服务实例(端口 8084、8085、8086),观察 Nacos 如何管理多个实例
3. **负载均衡验证**:在 User Service 和 Catalog Service 的 Controller 中返回端口号,通过 Enrollment 服务多次调用这两个服务,验证请求是否均匀分配到 User Service 和 Catalog Service 的不同实例

4. **故障转移测试**:停止一个 User 服务实例,观察 Nacos 多久检测到并将其标记为不健康,验证请求是否自动转移到健康实例
5. **实例上下线**:重新启动被停止的实例,观察其重新注册到 Nacos 的过程

### 架构演进思考

在项目文档中(`docs/week07-reflection.md`)回答以下问题:

- 对比使用 Nacos 前后,服务间调用方式有什么变化?带来了哪些好处?
- Nacos 的临时实例和持久实例有什么区别?当前项目适合使用哪种?
- 如果 Nacos 服务器宕机,已经启动的服务还能正常通信吗?为什么?
- 命名空间(Namespace)和分组(Group)的作用是什么?如何利用它们实现环境隔离?

### 拓展功能

尝试实现以下功能之一(可选):

- 为服务实例添加自定义元数据(如版本号、区域、权重等),并在调用时根据元数据进行路由
- 实现优雅上下线:服务关闭前先从 Nacos 注销,等待正在处理的请求完成后再关闭
- 配置 Nacos 的命名空间和分组,分别部署 `dev` 和 `test` 环境,验证环境隔离效果
- 使用 Nacos 的配置管理功能,将数据库连接信息等配置统一管理在 Nacos 中

## 提交要求

### 代码要求

- User Service, Catalog Service, 和 Enrollment Service 都能够成功注册到 Nacos
- 服务间调用通过服务名而非 IP 地址,特别强调 Enrollment Service 应该通过服务名调用 User Service 和 Catalog Service
- 健康检查配置正确,Nacos 能够准确显示服务状态
- Docker Compose 能够一键启动所有服务(包括 Nacos)
- 项目包名遵循规范:所有代码包名应遵循 `com.zjsu.<姓名缩写>.coursecloud.*` 格式

### 文档要求

- 在 `README.md` 中添加 Nacos 部署和配置章节
- 提供 Nacos 控制台访问地址和默认账号密码(http://localhost:8848/nacos,nacos/nacos)
- 截图展示 Nacos 控制台中的服务列表
- 截图展示多实例负载均衡效果(响应中包含不同的端口号)
- 截图展示故障转移效果(停止一个实例后请求仍然成功)

### 测试脚本

创建 `scripts/nacos-test.sh` 脚本,包含以下步骤:

```
#!/bin/bash

echo "启动所有服务..."
```

```
docker-compose up -d

echo "等待服务启动..."
sleep 30

echo "检查 Nacos 控制台..."
curl http://localhost:8848/nacos/

echo "检查服务注册情况..."
curl -X GET "http://localhost:8848/nacos/v1/ns/instance/list?serviceName=
catalog-service"

echo "测试服务调用..."
for i in {1..10}; do
  echo "第 $i 次请求:"
  curl http://localhost:8082/api/enrollments/test
done

echo "查看容器状态..."
docker-compose ps
```

### 提交信息建议

提交代码后,使用以下 Git 命令标记里程碑:

```
git add .
git commit -m "feat(nacos): integrate nacos for service discovery

- Add nacos-discovery dependency
- Configure service registration with nacos
- Replace hard-coded service urls with service names
- Add nacos server to docker-compose
- Configure health checks for service monitoring
- Test load balancing with multiple instances"

git tag v07
```

## 评分标准

- Nacos 集成配置正确(30%)
- 服务能够成功注册和发现(30%)
- 负载均衡和故障转移验证通过(20%)
- 健康检查配置正确(10%)
- 文档和测试脚本完整(10%)

# 常见问题

## 1. 服务无法注册到 Nacos

**可能原因**:

- Nacos 服务器未启动或网络不通
- 配置文件中的 `server-addr` 地址错误
- 命名空间 ID 不存在

**解决方法**:

- 检查 Nacos 容器是否正常运行:`docker logs nacos`
- 在 Docker 网络中使用容器名 `nacos` 而非 `localhost`
- 在 Nacos 控制台创建对应的命名空间

## 2. 服务间调用失败

**可能原因**:

- 服务名配置错误(大小写敏感)
- 目标服务未启动或未注册成功
- 网络配置问题

**解决方法**:

- 在 Nacos 控制台确认目标服务已注册且状态健康
- 检查 DiscoveryClient 调用的服务名是否与 Nacos 中注册的服务名一致(大小写敏感)
- 确保所有服务在同一 Docker 网络中

## 3. 健康检查一直显示不健康

**可能原因**:

- 健康检查接口返回非 200 状态码
- 应用未完全启动
- Actuator 端点未暴露

**解决方法**:

- 添加 Actuator 依赖并配置:`management.endpoints.web.exposure.include=health`
- 确保 `/actuator/health` 端点可访问
- 增加服务启动等待时间