

东软睿道内部公开

文件编号：D000-

Spring Cloud微服务架构

版本：1.0.0

第2章 Spring Cloud入门

东软睿道教育信息技术有限公司
(版权所有，翻版必究)

Copyright © Neusoft Educational Information Technology Co., Ltd
All Rights Reserved



本章教学目标

- ✓ 了解什么是Spring Cloud;
- ✓ 了解Spring Cloud特征;
- ✓ 了解Spring Cloud版本;
- ✓ 掌握Spring Cloud技术栈;
- ✓ 掌握服务发现模块编写
- ✓ 掌握微服务提供者模块编写;
- ✓ 掌握微服务消费者模块编写;



本章教学内容

节	知识点	掌握程度	难易程度	教学形式	对应在线微课
Spring Cloud简介	Spring Cloud是什么	了解		线下	
	Spring Cloud技术栈	掌握		线下	
	Spring Cloud特征	了解		线下	
	Spring Cloud版本	了解		线下	
使用Spring Cloud实战微服务	开发环境需求	掌握		线下	
	编写整体父工程	掌握		线下	
	编写注册发现功能	掌握	难	线下	
	编写微服务提供者模块	掌握		线下	
	编写微服务消费者模块	掌握	难	线下	

CONTENTS

目录

01

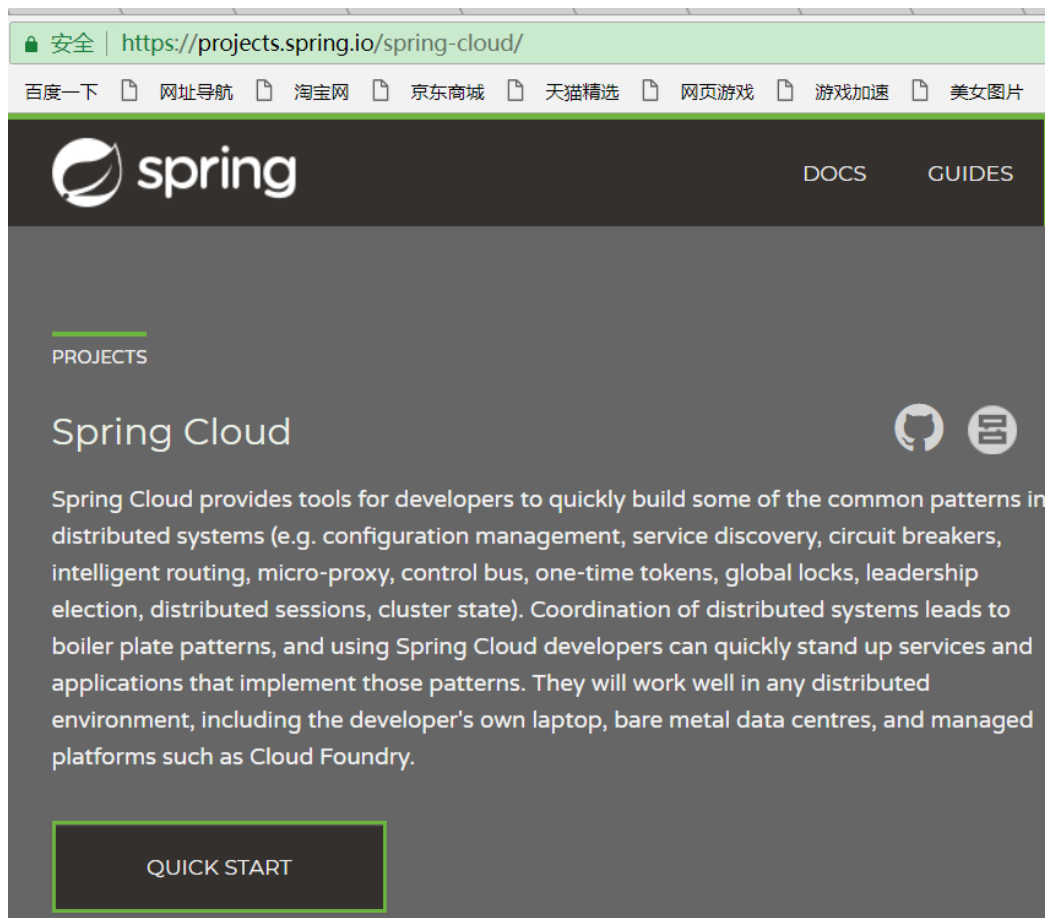
Spring Cloud简介

02

使用Spring Cloud实战微服务

Spring Cloud是什么

- ❖ Spring Cloud项目的官方网址：
<https://projects.spring.io/spring-cloud/>



Spring Cloud是什么

- ❖ Spring Cloud提供了一系列工具，可以帮助开发人员迅速搭建分布式系统中的公共组件（比如：配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性令牌、全局锁、主节点选举、分布式session、集群状态）。协调分布式环境中各个系统，为各类服务提供模板性配置。使用Spring Cloud，开发人员可以搭建实现了这些模板的应用，并且在任何分布式环境下都能工作得非常好，小到笔记本电脑，大到数据中心和云平台。

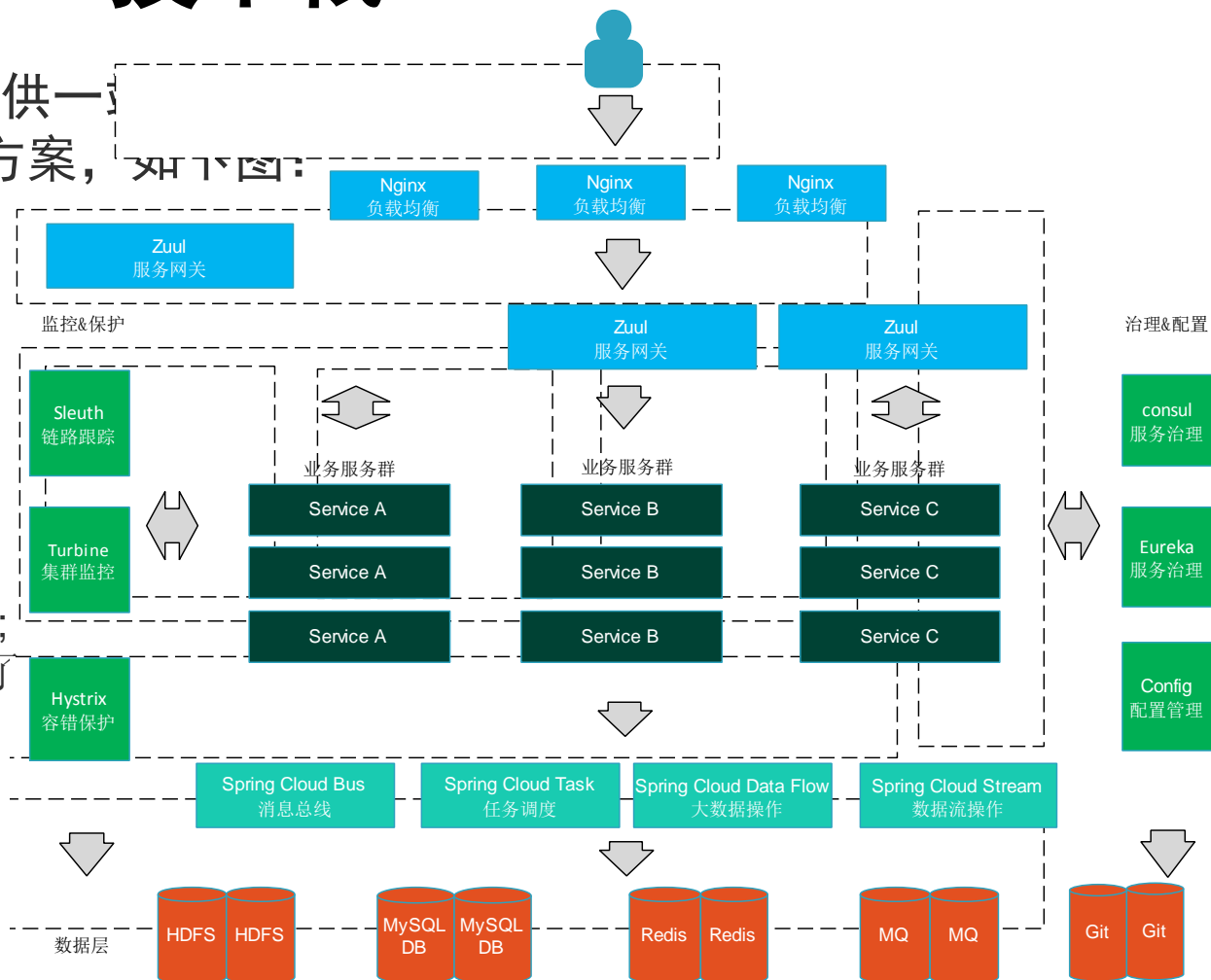
Spring Cloud是什么

- ❖ Spring Cloud并不是一个项目，而是一组项目的集合。在Spring Cloud中包含了很多的子项目，每一个子项目都是一种微服务开发过程中遇到的问题的一种解决方案。它利用Spring Boot的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。
- ❖ Spring Cloud并没有重复制造轮子，它只是将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包。

Spring Cloud技术栈

❖ Spring Cloud 提供一套微服务架构解决方案，如下图所示：

Zuul 祖鲁;
consul 英['kɒnsəl] 领事;
sleuth 英[slu:θ] 侦探;
turbine 英['tɜ:bain] 涡轮机;
eureka 英[ju:'ri:kə] 我发现了



Spring Cloud技术栈

- ❖ Spring Cloud为开发人员构建微服务架构提供了完整的解决方案，SpringCloud是若干个框架的集合，它包括spring-cloud-config、spring-cloud-bus等近20个子项目，它提供了服务治理、服务网关、智能路由、负载均衡、断路器、监控跟踪、分布式消息队列、配置管理等领域的解决方案。

Spring Cloud技术栈

❖ 子项目介绍

子项目名称	描述
Spring Cloud Aws	用于简化整合Amazon Web Service的组件
Spring Cloud Bus	事件、消息总线，用于在集群（例如，配置变化事件）中传播状态变化，可与Spring Cloud Config联合实现热部署。
Spring Cloud Cli	基于 Spring Boot CLI，可以让你以命令行方式快速建立云组件。
Spring Cloud Commons	服务发现、负载均衡、熔断机制这种模式为Spring Cloud客户端提供了一个通用的抽象层。
Spring Cloud Config	配置管理开发工具包，可以让你把配置放到远程服务器，目前支持本地存储、Git以及Subversion。
Spring Cloud Contract	
Spring Cloud Netflix	针对多种Netflix组件提供的开发工具包，其中包括Eureka、Hystrix、Zuul、Archaius等。
Spring Cloud Security	安全工具包
Spring Cloud Cloudfoundry	通过Oauth2协议绑定服务到CloudFoundry，CloudFoundry是VMware推出的开源PaaS云平台
Spring Cloud Consul	封装了Consul操作，consul是一个服务发现与配置工具，与Docker容器可以无缝集成。
Spring Cloud Sleuth	日志收集工具包，封装了Dapper,Zipkin和HTrace操作.Spring Cloud应用的分布式跟踪实现
Spring Cloud Stream	数据流操作开发包，封装了与Redis,Rabbit、Kafka等发送接收消息,实现的消息微服务。
Spring Cloud Zookeeper	基于ZooKeeper的服务发现与配置管理组件
Spring Boot	
Spring Cloud Task	用于快速构建数据处理的应用
Spring Cloud Gateway	Spring Cloud网关相关的整合实现

Spring Cloud特征

- ❖ Spring Cloud专注于为经典用例和扩展机制提供良好的开箱即用
 - ▶ 分布式/版本化配置
 - ▶ 服务注册和发现
 - ▶ 路由选择
 - ▶ Service-to-service调用
 - ▶ 负载均衡
 - ▶ 断路器
 - ▶ 分布式消息传递

Spring Cloud版本

❖ 下载官网

▶ <http://projects.spring.io/spring-cloud/>

❖ Spring Cloud 项目目前仍然是快速迭代期，版本变化很快。

❖ 大部分Spring软件的版本是以：主版本. 次版本. 增量版本. 里程碑版本的形式命名，例如Spring Framework的里程碑版本5.0.0.M4、稳定版本 5.1.0.RELEASE等

❖ Spring Cloud的最新版本如图：

Spring Cloud	
RELEASE	DOCUMENTATION
Finchley SR1 GA	Reference
Finchley SNAPSHOT	
Edgware SR4 GA	Reference
Edgware SNAPSHOT	
Dalston SR5 GA	Reference

❖ 帮助文档

▶ <https://cloud.spring.io/spring-cloud-static/Finchley.SR1/>

Spring Cloud版本

- ❖ Spring Cloud是一个由众多独立子项目组成的大型综合项目，每个子项目有不同的发行节奏，都维护着自己的发布版本号。Spring Cloud通过一个资源清单BOM (Bill of Materials) 来管理每个版本的子项目清单。为避免与子项目的发布号混淆，所以没有采用版本号的方式，而是通过命名的方式。
- ❖ 这些版本名称的命名方式采用了伦敦地铁站的名称，同时根据字母表的顺序来对应版本时间顺序，比如：最早的Release版本：Angel，第二个Release版本：Brixton，然后是Camden、Dalston、Edgware，目前最新的是Finchley版本。
- ❖ 当一个版本的Spring Cloud项目的发布内容积累到临界点或者解决了一个严重bug后，就会发布一个“Service Releases”版本，简称SRX版本，其中X是一个递增数字。

Spring Cloud版本

❖ 最新版本对应的子项目版本

Component	Edgware.SR4	Finchley.SR1	Finchley.BUILD-SNAPSHOT
spring-cloud-aws	1.2.3.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-bus	1.3.3.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-cli	1.4.1.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-commons	1.3.4.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-contract	1.2.5.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-config	1.4.4.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-netflix	1.4.5.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-security	1.2.3.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-cloudfoundry	1.1.2.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-consul	1.3.4.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-sleuth	1.3.4.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-stream	Ditmars.SR4	Elmhurst.SR1	Elmhurst.BUILD-SNAPSHOT
spring-cloud-zookeeper	1.2.2.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-boot	1.5.14.RELEASE	2.0.4.RELEASE	2.0.4.BUILD-SNAPSHOT

Spring Cloud版本

❖ 最新版本对应的子项目版本

spring-cloud-task	1.2.3.RELEASE	2.0.0.RELEASE	2.0.1.BUILD-SNAPSHOT
spring-cloud-vault	1.1.1.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-gateway	1.0.2.RELEASE	2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-openfeign		2.0.1.RELEASE	2.0.2.BUILD-SNAPSHOT
spring-cloud-function	1.0.0.RELEASE	1.0.0.RELEASE	1.0.1.BUILD-SNAPSHOT

Spring Cloud版本

❖ Spring Cloud与Spring Boot版本匹配关系

Spring Cloud	Spring Boot
Finchley	兼容Spring Boot 2.0.x, 不兼容Spring Boot 1.5.x
Dalston和Edgware	兼容Spring Boot 1.5.x, 不兼容Spring Boot 2.0.x
Camden	兼容Spring Boot 1.4.x, 也兼容Spring Boot 1.5.x
Brixton	兼容Spring Boot 1.3.x, 也兼容Spring Boot 1.4.x
Angel	兼容Spring Boot 1.2.x

CONTENTS

目录

01

Spring Cloud简介

02

使用Spring Cloud实战微服务

服务提供者与服务消费者

- ❖ 使用微服务构建的是分布式系统，微服务之间通过网络进行通信。我们使用服务提供者与服务消费者来描述微服务直接的调用关系。

服务提供者与服务消费者

名词	定义
服务提供者	服务的被调用方（即：为其他服务提供服务的服务）
服务消费者	服务的调用方（即：依赖其他服务的服务）

使用Spring Cloud实战微服务

- ❖ 示例：实现用户微服务提供者和服务消费者
 - ▶ 启动注册中心
 - ▶ 启动微服务提供者
 - ▶ 启动服务消费者
 - ▶ 浏览器输入 <http://localhost/consumer/user/findUserById/1>

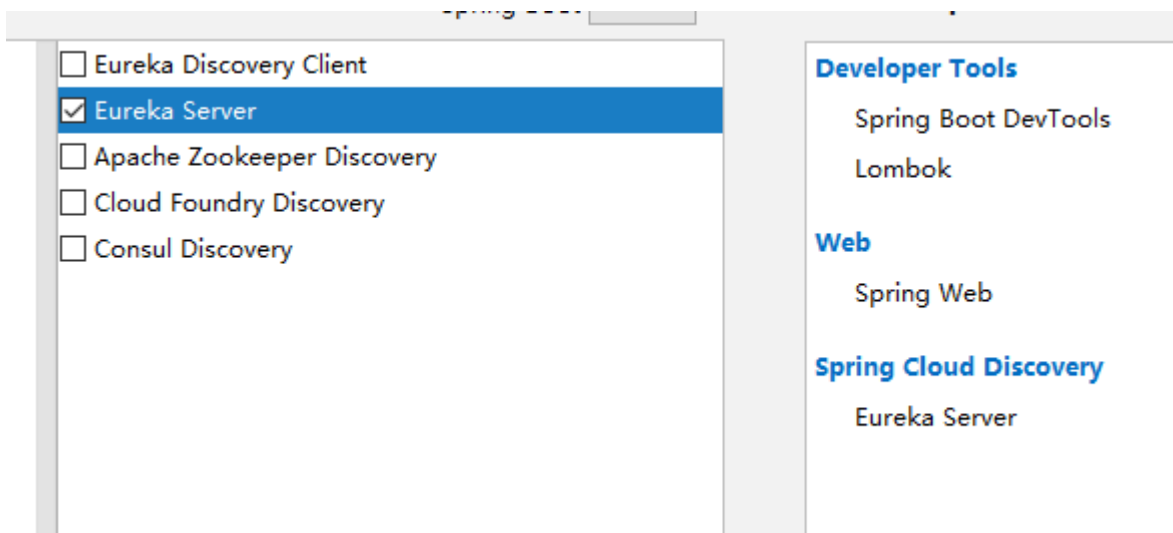
开发环境需求

- ❖ java环境:
 - ▶ Java1.8
- ❖ IDEA
- ❖ 数据库
 - ▶ Mysql5.7

Spring Cloud项目构建方式

❖ 使用SpringBoot构建父工程

- ▶ 先创建一个父工程
- ▶ 其他的模块都作为父工程的module来创建



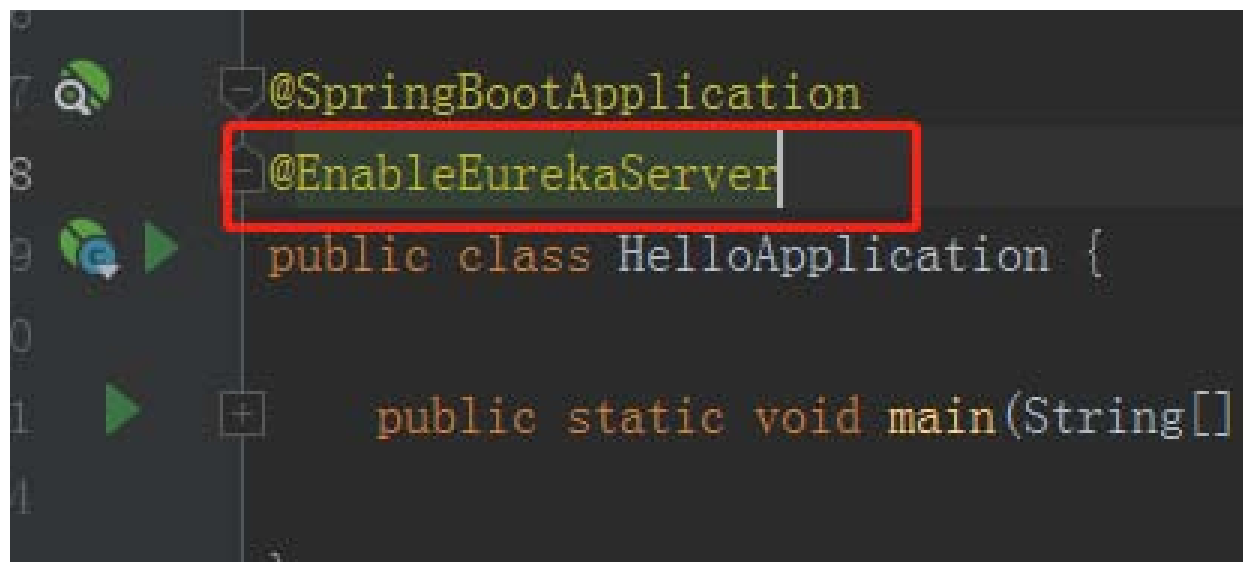
使用Spring Cloud实战微服务

❖ 编写子模块

- ▶ 编写整体父工程（服务注册发现），端口7001
- ▶ 编写Provider模块，端口8001 用户微服务提供者Module
- ▶ 编写Comsumor模块，端口80 用户微服务消费者Module

编写整体父工程

- ❖ 编写整体父工程（服务注册发现）
 - ▶ 修改启动类



```
7  @SpringBootApplication
8  @EnableEurekaServer
9  public class HelloApplication {
10
11      public static void main(String[]
```

编写整体父工程

❖ 修改核心配置文件（application.yml）

▶ spring:

application:

name: register #注册中心的名字

server:

port: 7001 #服务端口

eureka: #注册中心相关配置

server: # 配置关闭自我保护，并按需配置Eureka Server清理无效节点的时间间隔（5000ms）。

enable-self-preservation: false

eviction-interval-timer-in-ms: 5000

client:

register-with-eureka: false # 不将自己注册到注册中心

fetch-registry: false # 因为自己是注册中心，因此不用检索服务信息

service-url: # 注册中心的地址

defaultZone: http://localhost:7001/eureka/

instance:

prefer-ip-address: true #用机器名标识服务名

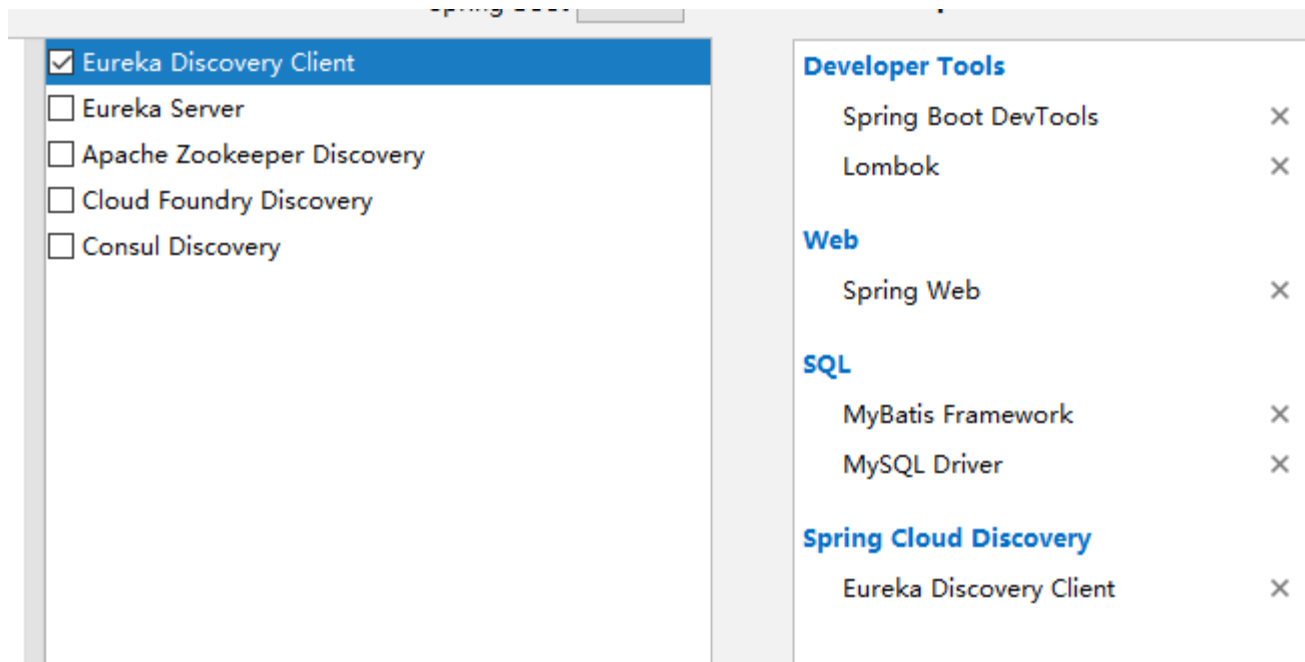
编写微服务提供者模块

❖ 步骤

- ▶ 新建Module Provider，端口8001
- ▶ 编写Provider业务代码
- ▶ 编写核心配置文件application.yml
- ▶ 修改启动类

编写微服务提供者模块

❖ 新建Module Provider




编写微服务提供者模块

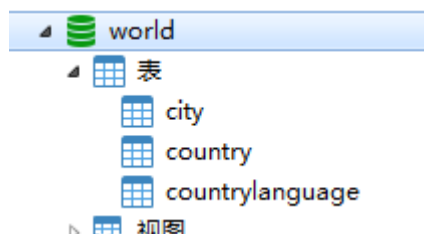
❖ 编写核心配置文件application.yml

```
spring:
  application:
    name: provider
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/world?characterEncoding=utf-8&serverTimezone=GMT%2B8
    username: root
    password: 123456
  server:
    port: 8001
  eureka:
    client:
      service-url:
        defaultZone: http://localhost:7001/eureka/
    instance:
      prefer-ip-address: true
      instance-id: city-provider
  info.app.name: provider
  info.company.name: example
  info.author.name: demo
  logging:
    level:
      root: INFO
```

编写微服务提供者模块

❖ 示例用数据库world

字段	索引	外键	触发器	选项	注释	SQL 预览					
名					类型	长度	小数点	不是 null	虚拟	键	注释
ID					int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
Name					char	35	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
CountryCode					char	3	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
District					char	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Population					int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		



ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020
19	Zaanstad	NLD	Noord-Holland	135621
20	's-Hertogenbosch	NLD	Noord-Brabant	129170
21	Amersfoort	NLD	Utrecht	126270
22	Maastricht	NLD	Limburg	122097

编写微服务提供者模块

❖ 修改启动类

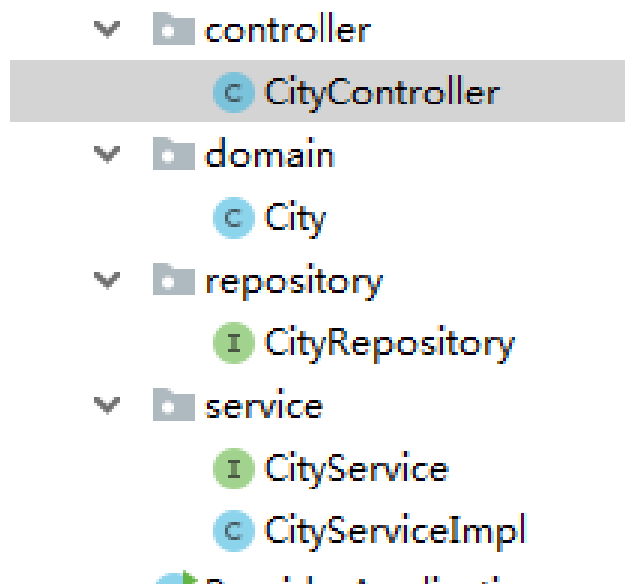
@SpringBootApplication

@EnableEurekaClient

```
public class Application {  
    public static void main(String[] args) {  
        .....  
    }  
}
```

编写微服务提供者模块

❖ 增加服务提供者代码

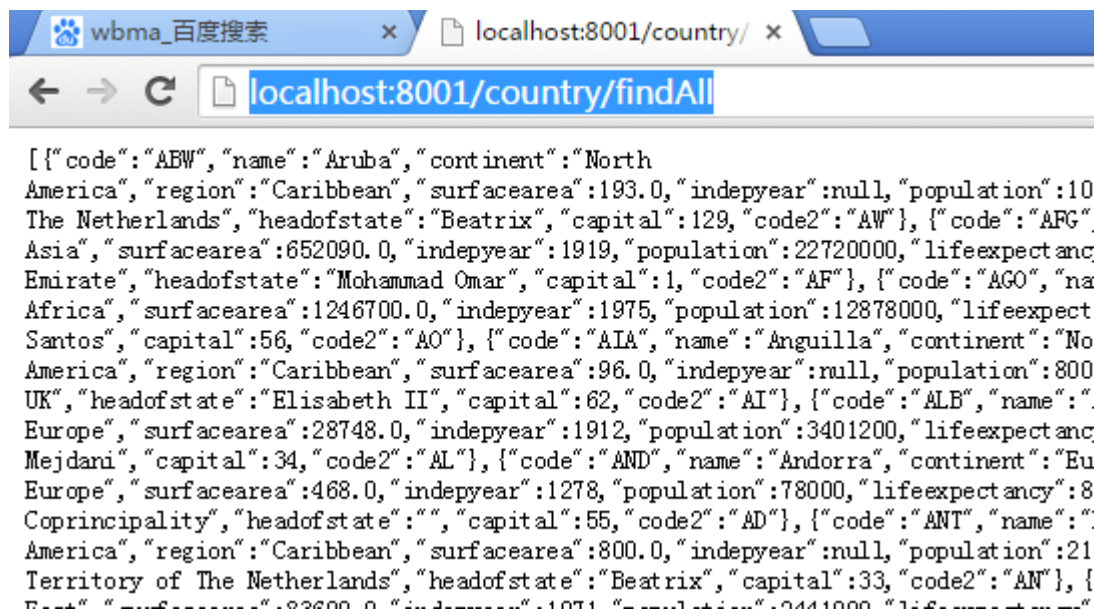


编写微服务提供者模块

❖ 测试微服务提供者

- ▶ 启动provider主启动类
- ▶ 浏览器访问

★ <http://localhost:8001/country/findAll>



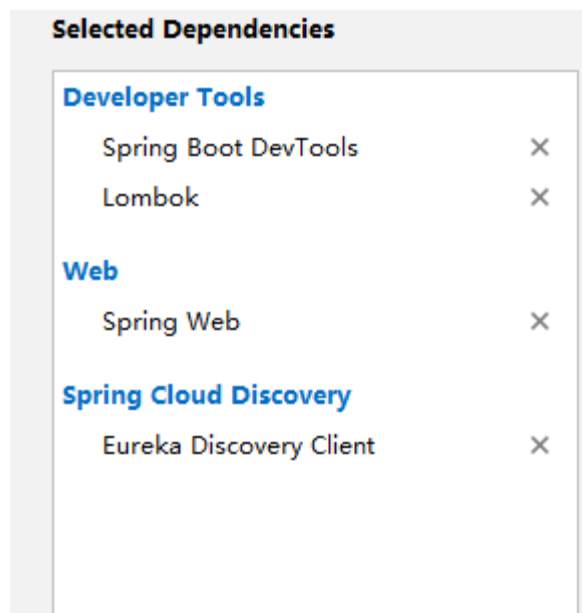
编写微服务消费者模块

❖ 步骤

- ▶ 新建Module Comsumor, 端口80
- ▶ 修改全局配置文件
- ▶ 编写ConfigBean加载RestTemplate
- ▶ 编写消费者代码
- ▶ 修改主启动类
- ▶ 测试

编写微服务消费者模块

❖ 新建Module Comsumor, 端口80



编写微服务消费者模块-方法2

- ❖ 修改全局配置文件 (application.yml)

```
spring:
```

```
  application:
```

```
    name: comsumor
```

```
server:
```

```
  port: 80
```

```
eureka:
```

```
  client:
```

```
    service-url:
```

```
      defaultZone: http://localhost:7001/eureka/
```

编写微服务消费者模块

❖ 编写ConfigBean加载RestTemplate

- ▶ 传统情况下在java代码里访问restful服务，一般使用Apache的HttpClient。不过此种方法使用起来太过繁琐。spring提供了一种简单便捷的模板类来进行操作，这就是RestTemplate。

@Configuration

```
public class ConfigBean {  
    @Bean  
    public RestTemplate getRestTemplate() {  
        return new RestTemplate();  
    }  
}
```

编写微服务消费者模块

❖ 编写消费者代码

```
@RestController
@RequestMapping("consumer")
public class CallController {
    private final static String PROVIDER_URL = "http://localhost:8001/";

    @Autowired
    RestTemplate restTemplate;

    @RequestMapping("/findAll")
    public Object findAll() {
        return restTemplate.getForObject(PROVIDER_URL + "country/findAll",
Object.class);
    }
    @RequestMapping("/findByCode/{code}")
    public Object findByCode(@PathVariable String code) {
        return restTemplate.getForObject(PROVIDER_URL + "country/find/" +
code, Object.class);
    }
}
```

编写微服务消费者模块

❖ 修改主启动类

```
@SpringBootApplication  
@EnableEurekaClient  
public class HelloApplication {  
    .....  
}
```

编写微服务消费者模块

❖ 测试

← → ↺ ⓘ localhost:7001

应用 梯度下降法的推导... 6.5 矩阵的运算及... 降维算法----PCA... SpringCloud微服... MNIST数据集介绍... vn.py量化社区 - B... 知 vn

Application	AMIs	Availability Zones	Status
CONSUMOR	n/a (1)	(1)	UP (1) - PC-20200320LFAL:comsumor:80
PROVIDER	n/a (1)	(1)	UP (1) - city-provider

← → ↺ ⓘ localhost/consumer/findAll

应用 梯度下降法的推导... 6.5 矩阵的运算及... 降维算法----PCA... SpringCloud微服... MNIST数据集介绍... vn.py量化社区 - B... 知 vn

```
[{"id":1,"name":"Kabul","countryCode":"AFG","district":"Kabul","population":1780000}, {"id":2,"name":"Qandahar","countryCode":"AFG","district":"Qandahar","population":186800}, {"id":3,"name":"Herat","countryCode":"AFG","district":"Herat","population":186800}, {"id":4,"name":"Mazar-e-Sharif","countryCode":"AFG","district":"Mazar-e-Sharif","population":186800}, {"id":5,"name":"Amsterdam","countryCode":"NLD","district":"Noord-Holland","population":731200}, {"id":6,"name":"Rotterdam","countryCode":"NLD","district":"Zuid-Holland","population":440900}, {"id":7,"name":"Haag","countryCode":"NLD","district":"Zuid-Holland","population":440900}, {"id":8,"name":"Utrecht","countryCode":"NLD","district":"Utrecht","population":201843}, {"id":9,"name":"Eindhoven","countryCode":"NLD","district":"Noord-Brabant","population":201843}, {"id":10,"name":"Tilburg","countryCode":"NLD","district":"Noord-Brabant","population":201843}, {"id":11,"name":"Groningen","countryCode":"NLD","district":"Groningen","population":172701}, {"id":12,"name":"Breda","countryCode":"NLD","district":"Gelderland","population":153491}, {"id":13,"name":"Apeldoorn","countryCode":"NLD","district":"Gelderland","population":153491}, {"id":14,"name":"Nijmegen","countryCode":"NLD","district":"Gelderland","population":153491}, {"id":15,"name":"Enschede","countryCode":"NLD","district":"Overijssel","population":149544}, {"id":16,"name":"Haarlem","countryCode":"NLD","district":"Zuid-Holland","population":142465}, {"id":17,"name":"Almere","countryCode":"NLD","district":"Flevoland","population":142465}, {"id":18,"name":"Arnhem","countryCode":"NLD","district":"Gelderland","population":135621}, {"id":19,"name":"Zaanstad","countryCode":"NLD","district":"Noord-Holland","population":135621}, {"id":20,"name":"Hertogenbosch","countryCode":"NLD","district":"Gelderland","population":129170}, {"id":21,"name":"Amersfoort","countryCode":"NLD","district":"Utrecht","population":126270}, {"id":22,"name":"Maastricht","countryCode":"NLD","district":"Limburg","population":122087}, {"id":23,"name":"Dordrecht","countryCode":"NLD","district":"Zuid-Holland","population":117196}, {"id":24,"name":"Leiden","countryCode":"NLD","district":"Zuid-Holland","population":117196}, {"id":25,"name":"Haarlemmermeer","countryCode":"NLD","district":"Zuid-Holland","population":11021}, {"id":26,"name":"Zoetermeer","countryCode":"NLD","district":"Zuid-Holland","population":11021}, {"id":27,"name":"Emmen","countryCode":"NLD","district":"Drenthe","population":105853}, {"id":28,"name":"Zwolle","countryCode":"NLD","district":"Gelderland","population":101574}, {"id":29,"name":"Ede","countryCode":"NLD","district":"Gelderland","population":101574}, {"id":30,"name":"Delft","countryCode":"NLD","district":"Zuid-Holland","population":95052}, {"id":31,"name":"Heerlen","countryCode":"NLD","district":"Limburg","population":95052}, {"id":32,"name":"Alkmaar","countryCode":"NLD","district":"Zuid-Holland","population":2345}, {"id":33,"name":"Willemstad","countryCode":"ANT","district":"Curaçao","population":2345}, {"id":34,"name":"Tirana","countryCode":"ALB","district":"Tirana","population":2168000}, {"id":35,"name":"Algier","countryCode":"DZA","district":"Algier","population":2168000}, {"id":36,"name":"Oran","countryCode":"DZA","district":"Oran","population":443727}, {"id":37,"name":"Constantine","countryCode":"DZA","district":"Constantine","population":443727}, {"id":38,"name":"Annaba","countryCode":"DZA","district":"Annaba","population":183377}, {"id":39,"name":"Batna","countryCode":"DZA","district":"Batna","population":183377}, {"id":40,"name":"Sétif","countryCode":"DZA","district":"Sétif","population":153106}, {"id":41,"name":"Sidi Bel Abbès","countryCode":"DZA","district":"Sidi Bel Abbès","population":153106}, {"id":42,"name":"Skikda","countryCode":"DZA","district":"Skikda","population":128281}, {"id":43,"name":"Biskra","countryCode":"DZA","district":"Biskra","population":128281}, {"id":44,"name":"Blida (el-Boulaida)","countryCode":"DZA","district":"Blida","population":117162}, {"id":45,"name":"Béjaïa","countryCode":"DZA","district":"Béjaïa","population":117162}, {"id":46,"name":"Mostaganem","countryCode":"DZA","district":"Mostaganem","population":112007}, {"id":47,"name":"Tébessa","countryCode":"DZA","district":"Tébessa","population":112007}, {"id":48,"name":"Tlemcen (Tilimsen)","countryCode":"DZA","district":"Tlemcen","population":107311}, {"id":49,"name":"Béchar","countryCode":"DZA","district":"Béchar","population":107311}, {"id":50,"name":"Tiaret","countryCode":"DZA","district":"Tiaret","population":96794}, {"id":51,"name":"Ech-Chleff (el-Asnam)","countryCode":"DZA","district":"Chlef","population":96794}, {"id":52,"name":"Ghardaïa","countryCode":"DZA","district":"Ghardaïa","population":96794}
```

编写微服务消费者模块

- ❖ RestTemplate提供了多种便捷访问远程Http服务的方法，是一种简单便捷的访问Restful服务模板类，是Spring提供的用于访问Rest服务的客户端模板工具集
- ❖ 接口的参数：
 - ▶ url, REST请求地址
 - ▶ requestMap, 请求参数
 - ▶ ResponseBean.class, HTTP响应转换被转换成的对象类型。

本章重点总结

- ❖ 了解什么是Spring Cloud;
- ❖ 了解Spring Cloud特征;
- ❖ 了解Spring Cloud版本;
- ❖ 掌握Spring Cloud技术栈;
- ❖ 掌握服务发现模块编写
- ❖ 掌握微服务提供者模块编写;
- ❖ 掌握微服务消费者模块编写;

课后作业【必做任务】

- ❖ 1、独立完成课件中的示例

