微服务核心由三部分组成：

- 注册中心：注册各个微服务的（服务提供者、服务消费者），所有的微服务必须在注册中心完成注册才可以调用。
- 服务提供者：提供某一项服务，具体是和业务关联。
- 服务消费者：调用服务提供者提供的服务。

注册中心相当于电商平台，服务提供者相当于卖家开的店铺，把店铺的相关信息在平台进行注册，服务消费者相当于普通用户在平台注册账户，用户就可以在平台上的店铺中进行购物，相当于服务消费者可以调用注册中心的各个服务提供者。

注册中心是通过 Eureka Server 来实现的，服务提供者是通过 Eureka Client 来实现的。

- 在父工程下创建 Module，实现 Eureka Client，pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
</dependencies>
```

- 在 resources 路径下创建配置文件 application.yml，添加 Eureka Client 相关配置，此时的 Eureka Client 是服务提供者 provider。

```
server:
  port: 8010
spring:
  application:
    name: provider
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true
```

> 属性说明：

`server.port`：当前 Eureka Client 服务端口。

`spring.application.name`：当前服务注册在 Eureka Server 上的名称。

`eureka.client.service-url.defaultZone`：注册中心的访问地址。

`eureka.instance.prefer-ip-address`：是否将当前的服务 IP 注册到 Eureka Server。

- 在 java 路径下创建启动类。

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProviderApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication.class,args);
    }
}
```

- 创建 StudentRepository、StudentRepositoryImpl。

```
package com.southwind.repository;

import com.southwind.entity.Student;

import java.util.Collection;

public interface StudentRepository {
    public Collection<Student> findAll();
    public Student findById(long id);
    public void saveOrUpdate(Student student);
    public void deleteById(long id);
}
```

```
package com.southwind.repository.impl;

import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

public class StudentRepositoryImpl implements StudentRepository {

    private Map<Long,Student> studentMap;

    public StudentRepositoryImpl(){
        studentMap = new HashMap<>();
        studentMap.put(1L,new Student(1L,"张三",22));
        studentMap.put(2L,new Student(2L,"李四",23));
        studentMap.put(3L,new Student(3L,"王五",24));
    }

    @Override
```

```
    public Collection<Student> findAll() {
        return studentMap.values();
    }

    @Override
    public Student findById(long id) {
        return studentMap.get(id);
    }

    @Override
    public void saveOrUpdate(Student student) {
        studentMap.put(student.getId(),student);
    }

    @Override
    public void deleteById(long id) {
        studentMap.remove(id);
    }
}
```

- 创建 StudentHandler，通过 @Autowired 注解将 StudentRepository 的实例对象注入 StudentHandler。

```
package com.southwind.controller;

import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Collection;

@RestController
@RequestMapping("/student")
public class StudentHandler {

    @Autowired
    private StudentRepository studentRepository;

    @GetMapping("/findAll")
    public Collection<Student> findAll(){
        return studentRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public Student findById(@PathVariable("id") long id){
        return studentRepository.findById(id);
    }
```

```
    @PostMapping("/save")
    public void save(@RequestBody Student student){
        studentRepository.saveOrUpdate(student);
    }

    @PutMapping("/update")
    public void update(@RequestBody Student student){
        studentRepository.saveOrUpdate(student);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long id){
        studentRepository.deleteById(id);
    }
}
```

服务消费者

不同服务之间的调用时通过 RestTemplate 来实现的。

什么是 RestTemplate?

RestTemplate 是 Spring 框架提供的一个用于访问 REST 服务的组件，底层对 HTTP 请求及响应进行了封装，提供了很多访问远程 REST 服务的方法，可简化代码的开发。

如何使用 RestTemplate

- 通过 Spring Boot 快速搭建一个 REST 服务，新建 Maven 工程，pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

REST 服务搭建成功之后，使用 RestTemplate 来访问该服务。

- 实例化 RestTemplate 对象并且通过 @Bean 注入 IoC，在启动类添加代码。

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
```

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

    @Bean
    public RestTemplate restTemplate(){
        return new RestTemplate();
    }
}
```

- 在父工程下创建 Module，实现 Eureka Client，pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
</dependencies>
```

- 创建配置文件 application.yml

```
server:
  port: 8020
spring:
  application:
    name: consumer
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true
```

- 创建启动类

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class ConsumerApplication {
    public static void main(String[] args) {
```

```
        SpringApplication.run(ConsumerApplication.class,args);
    }

    @Bean
    public RestTemplate restTemplate(){
        return new RestTemplate();
    }
}
```

- 创建 Handler

```
package com.southwind.controller;

import com.southwind.entity.Student;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

import java.util.Collection;

@RestController
@RequestMapping("/consumer")
public class RestHandler {

    @Autowired
    private RestTemplate restTemplate;

    @GetMapping("/findAll")
    public Collection<Student> findAll(){
        return
restTemplate.getForEntity("http://localhost:8010/student/findAll",Collection
.class).getBody();
    }

    @GetMapping("/findAll2")
    public Collection<Student> findAll2(){
        return
restTemplate.getForObject("http://localhost:8010/student/findAll",Collection
.class);
    }

    @GetMapping("/findById/{id}")
    public Student findById(@PathVariable("id") long id){
        return
restTemplate.getForEntity("http://localhost:8010/student/findById/{id}",Stud
ent.class,id).getBody();
    }

    @GetMapping("/findById2/{id}")
```

```
    public Student findById2(@PathVariable("id") long id){
        return
restTemplate.getForObject("http://localhost:8010/student/findById/{id}",Stud
ent.class,id);

    }

    @PostMapping("/save")
    public Collection<Student> save(@RequestBody Student student){
        return
restTemplate.postForEntity("http://localhost:8010/student/save",student,Coll
ection.class).getBody();
    }

    @PostMapping("/save2")
    public Collection<Student> save2(@RequestBody Student student){
        return
restTemplate.postForObject("http://localhost:8010/student/save",student,Coll
ection.class);
    }

    @PutMapping("/update")
    public void update(@RequestBody Student student){
        restTemplate.put("http://localhost:8010/student/update",student);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long id){

 restTemplate.delete("http://localhost:8010/student/deleteById/{id}",id);
    }

}
```