Spring Microservices

Microservices with Spring

- Spring's Microservice tools are contained in Spring Cloud
- Spring Cloud is an extension to Spring Boot
- Microservice Tools
  - Service Discovery - Eureka
  - Load Balancing - Ribbon
  - Centralized Configuration - Configuration Server

The Breakup - We have to talk

Separations of Concerns

- What are the core pieces of our App?
- What can operate independently?
- How do we break things up?

Demo 12: The New Plan

Create two microservices. One Services indicators and the other Customers.

1. Go back to the initializer and create a new Spring Boot Application
2. Download and import into the IDE
3. Move the Customer, CustomerRepository, CustomerServiceImpl, and CustomerController into the new project.
4. Annotate with Eureka and Ribbon
5. Centralize configuration
6. Create plain old java Library for the Exception handling, AoP, and Custom Annotations.

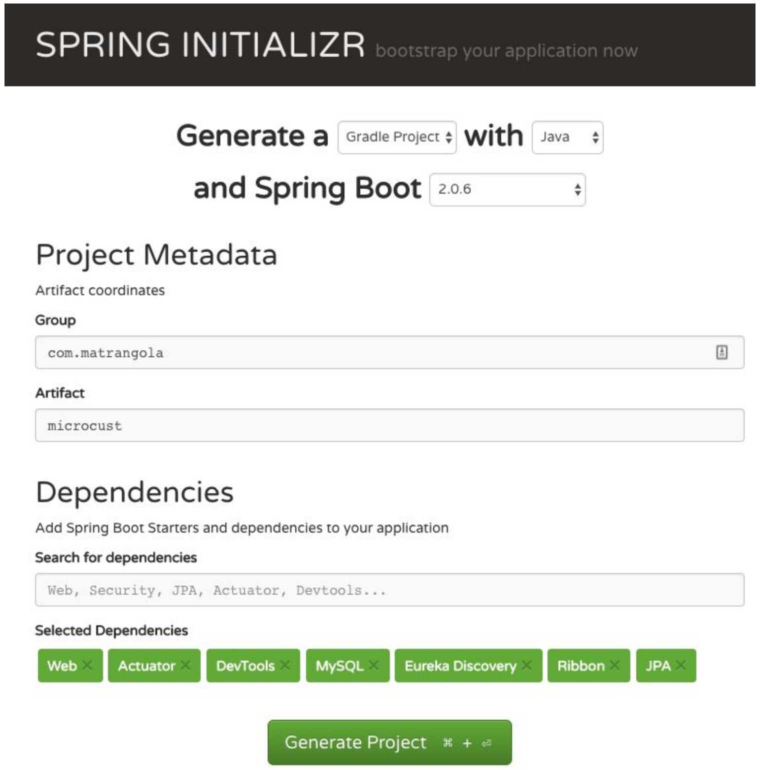
Lab 12: Break it Up

- Create a new microcustomer app using http://start.spring.io
- Download and open in your IDE (See Demo/Lab 1)
- Create new Customer Database
- Move Customer Related files from Indicators to Microcustomer
- Set new port number and application name for in application.properties

Lab 12a: Rebound

https://start.spring.io/

- Gradle Project
- Java
- 2.0.5
- Group: com.whatever
- Artifact: customers
- Dependencies: Web, Actuator, DevTools, MySQL, JPA, Ribbon, Eureka Discovery
- Download
- Unzip



SPRING INITIALIZR bootstrap your application now

Generate a Gradle Project with Java and Spring Boot 2.0.6

Project Metadata

Artifact coordinates

Group: com.matrangola

Artifact: microcust

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies: Web, Actuator, DevTools, MySQL, Eureka Discovery, Ribbon, JPA

Demo 12b: The new MicroCustomer App

After downloading and importing the template from the Initializer move the files from the Indicators App to the new MicroCustomer App.

1. Customer.java
    a. Remove References to Indicators in Customer.java
    b. Remove Reference to Customer in Indicator.java
2. CustomerRepository.java
3. Copy the REST Annotations from CustomerController to the new MicroCustomerApplication Class
4. Copy the @RequestMapping methods from CustomerController to MicroCustomerApplication Class

Finding your way

Problem: We want to list all indicators referenced by a particular customer.

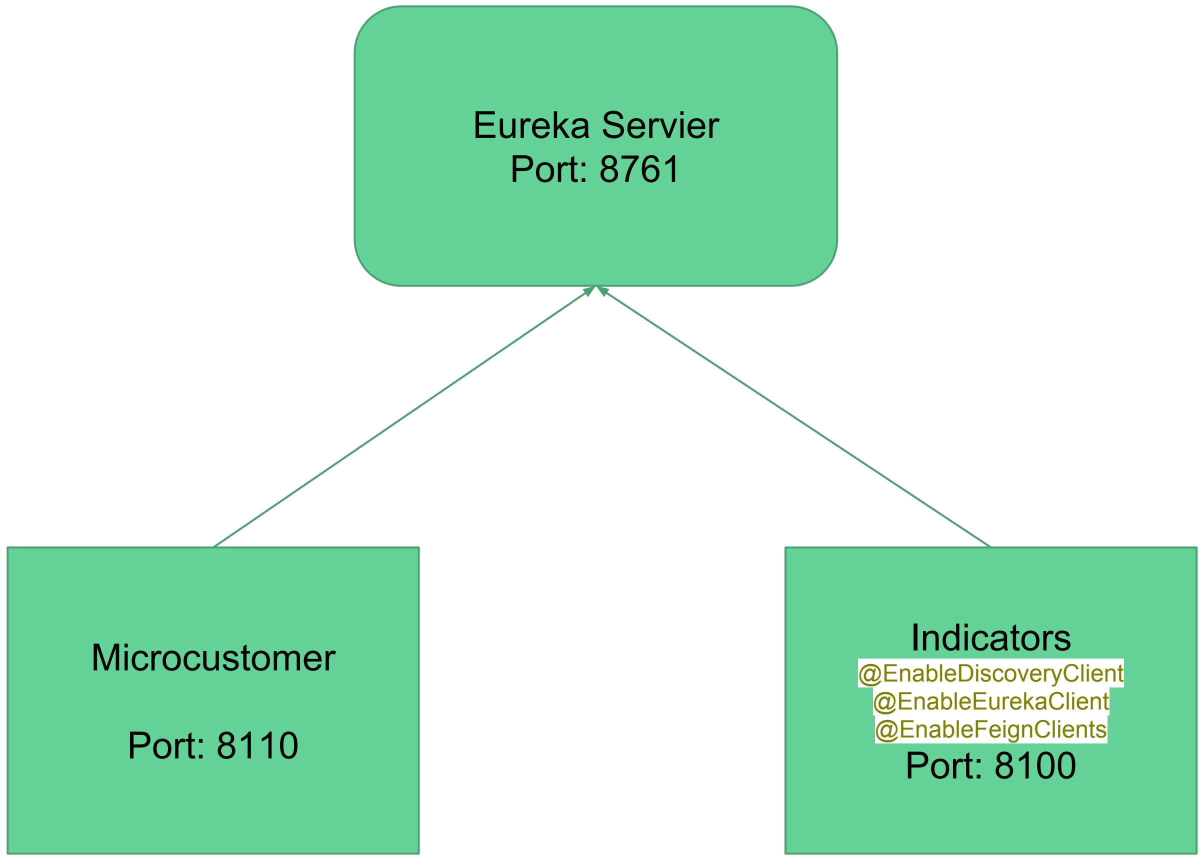
Q: How do we execute REST across microservices?

A: One microservice can be a client of another.

Q: How does one microservice find another?

A: Use Eureka to register in the service directory.

Our Eureka Setup



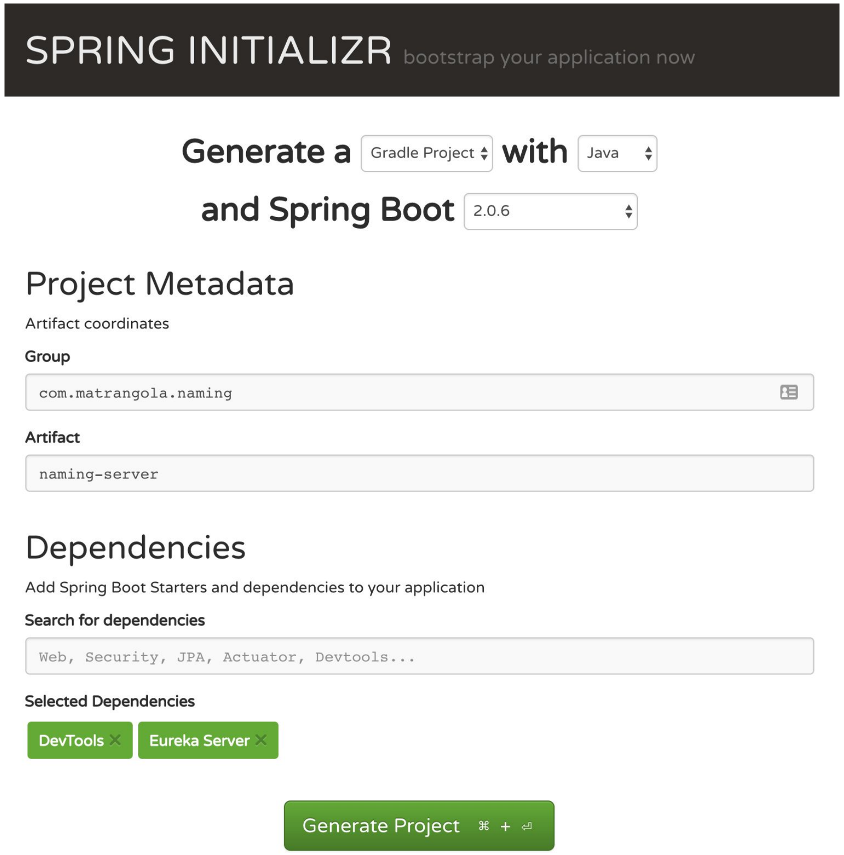
Demo 13b: Making Yourself Available

- Setup Naming Server
    - Initializer
    - Download
    - Setup Application Class
    - Setup application.properties
- Setup microcusotmer and gadgets
    - Setup application.properties restart
- http://localhost:8761/

Lab 13a: Making Yourself Available

https://start.spring.io/

- Gradle Project
- Java
- 2.0.5
- Group: com.whatever
- Artifact: naming-server
- Dependencies: Eureka Server, DevTools
- Download
- Unzip


SPRING INITIALIZR bootstrap your application now

Generate a Gradle Project with Java

and Spring Boot 2.0.6

Project Metadata

Artifact coordinates

Group

com.matrangola.naming

Artifact

naming-server

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

DevTools ✕ Eureka Server ✕

Lab 13b: Making Yourself Available

- Initializer from demo slide
- Setup Naming Server
  - Open the naming-service in your IDE
  - Add @EnableEurekaServer to the NamingServerApplication Class
  - Start Naming Server
  - Browse to: http://localhost:8761/
- Setup microcusotmer and indicators
  - Add build.gradle: `implementation('org.springframework.cloud:spring-cloud-starter-netflix-eureka-client')`
  - Update applicaton.properties

```
spring.application.name=microcustomer

server.port=8110

eureka.client.service-url.default-zone=http://localhost:8761/eureka

eureka.instance.preferIpAddress=true
```

  - Restart Microservices
- Refresh http://localhost:8761/

Lab 13c - Naming Service Configuration

naming-service/main/resources/application.properties

```
pring.application.name=naming-server
server.port=8761

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```