Surging

刘旭东

[公司名称]

Surging

使用 .Net Core构建微服务



目录

[1介绍微服务 2](#_Toc503209681)

[1.1它是啥玩意？ 2](#_Toc503209682)

[1.2深刻理解它 2](#_Toc503209683)

[1.3为啥使用它 2](#_Toc503209684)

[1.4牢记它的工作原理 2](#_Toc503209685)

[1.5它的优势 2](#_Toc503209686)

[1.6不是你想就能用——先决条件？ 3](#_Toc503209687)

[1.7 3](#_Toc503209688)

[2实现微服务架构 3](#_Toc503209689)

[2.1一个服务有多大？ 3](#_Toc503209690)

[2.2好的服务的标准是什么 3](#_Toc503209691)

[2.3我的最爱—DDD和它对于微服务的重要性 3](#_Toc503209692)

[2.4The concept of seam 3](#_Toc503209693)

[2.5服务之间的通信 3](#_Toc503209694)

[2.6Revisiting the Flix One case study 3](#_Toc503209695)

[3服务之间的技术集成 3](#_Toc503209696)

[3.1服务之间的通信 3](#_Toc503209697)

[3.2套路之——集成模式 3](#_Toc503209698)

[The API gateway 3](#_Toc503209699)

[The event-driven pattern 4](#_Toc503209700)

[Event sourcing 4](#_Toc503209701)

[Eventual consistency 4](#_Toc503209702)

[Compensating transactions 4](#_Toc503209703)

[Competing consumers 4](#_Toc503209704)

[Azure Service Bus 4](#_Toc503209705)

[Azure queues 4](#_Toc503209706)

[4服务的测试 4](#_Toc503209707)

[4.1如何进行 4](#_Toc503209708)

[4.2测试方法 4](#_Toc503209709)

[4.3Testing pyramid 4](#_Toc503209710)

[4.3测试的分类 4](#_Toc503209711)

[4.3.1单元测试 4](#_Toc503209712)

[4.3.2组件(服务)测试 4](#_Toc503209713)

[4.3.3集成测试 4](#_Toc503209714)

[4.3.4契约测试 5](#_Toc503209715)

[4.3.5性能测试 5](#_Toc503209716)

[4.3.6端对端(前后端)测试 5](#_Toc503209717)

[Sociable versus isolated unit tests 5](#_Toc503209718)

[约定与模拟 5](#_Toc503209719)

[4.4实操演练 5](#_Toc503209720)

[5微服务的部署 5](#_Toc503209721)

[5.1部署实施层面的术语 5](#_Toc503209722)

[5.2成功部署的先决条件 5](#_Toc503209723)

[5.3部署的隔离需求 5](#_Toc503209724)

[5.4部署范例 5](#_Toc503209725)

[5.5容器技术 5](#_Toc503209726)

[5.6Docker介绍 5](#_Toc503209727)

[6系统安全性 5](#_Toc503209728)

[7运行监控 5](#_Toc503209729)

[8系统扩展性 6](#_Toc503209730)

[9Introduction to Reactive Microservices 6](#_Toc503209731)

[10Surging框架的实现解说 6](#_Toc503209732)

# 1介绍微服务

## 1.1它是啥玩意？

微服务 (Microservices) 是一种软件架构风格，它是以专注于单一责任与功能的小型功能区块 (Small Building Blocks) 为基础，利用模组化的方式组合出复杂的大型应用程序，各功能区块使用与语言无关 (Language-Independent/Language agnostic) 的 API 集相互通讯

## 1.2深刻理解它

## 1.3为啥使用它

## 1.4牢记它的工作原理

## 1.5它的优势

## 1.6不是你想就能用——先决条件？

## 1.7

# 2实现微服务架构

## 2.1一个服务有多大？

## 2.2好的服务的标准是什么

## 2.3我的最爱—DDD和它对于微服务的重要性

## 2.4The concept of seam

## 2.5服务之间的通信

## 2.6Revisiting the Flix One case study

# 3服务之间的技术集成

## 3.1服务之间的通信

## 3.2套路之——集成模式

### The API gateway

### The event-driven pattern

### Event sourcing

### Eventual consistency

### Compensating transactions

### Competing consumers

### Azure Service Bus

### Azure queues

# 4服务的测试

## 4.1如何进行

## 4.2测试方法

## 4.3Testing pyramid

## 4.3测试的分类

### 4.3.1单元测试

### 4.3.2组件(服务)测试

### 4.3.3集成测试

### 4.3.4契约测试

### 4.3.5性能测试

### 4.3.6端对端(前后端)测试

### Sociable versus isolated unit tests

### 约定与模拟

## 4.4实操演练

# 5微服务的部署

## 5.1部署实施层面的术语

## 5.2成功部署的先决条件

## 5.3部署的隔离需求

## 5.4部署范例

## 5.5容器技术

## 5.6Docker介绍

# 6系统安全性

# 7运行监控

# 8系统扩展性

# 9Introduction to Reactive Microservices

# 10Surging框架的实现解说