微服务架构设计与实践

开发框架篇

孙玄@58集团

关于我

- 58集团技术主席
- 58集团高级架构师
- 转转架构负责人
- 百度高级工程师
- 毕业于浙江大学
- 代表公司多次对外分享
- 企业内训&公开课

关于我

企业内训

- ✓ 华为
- ✓ 中航信
- ✓ 平安
- ✓ 银联
- ✓ 华泰证券
- ✓ 思科
- ✓ 云南电力
- ✓ 深信服
- ✓ 新华社
- ✓ 民生银行
- ✓ 招商银行
- ✓ ……

公开课

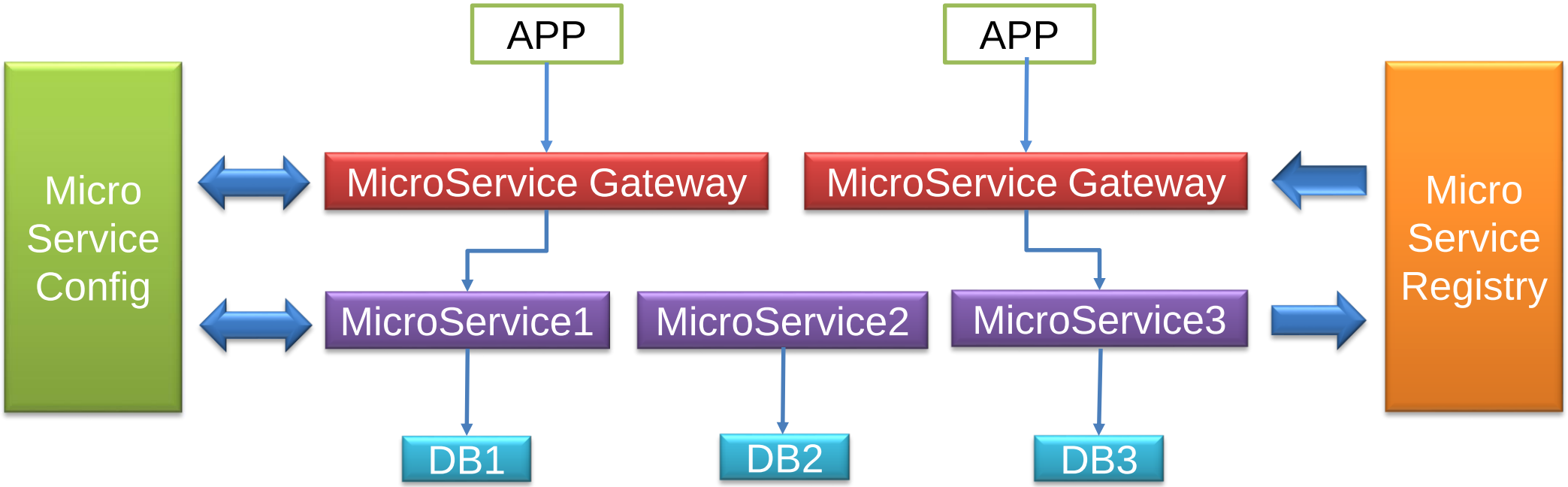
- ✓ 北京
- ✓ 上海
- ✓ 深圳
- ✓ 广州
- ✓ 成都
- ✓ ……

分享要点

微服务架构

1. 开发框架选择
2. 选择Spring Boot原因
3. 特点
4. 如何使用
5. 生产级特性
6. RPC

开发框架选择


APP
APP
Micro
Service
Config
MicroService Gateway
MicroService Gateway
Micro
Service
Registry
MicroService1
MicroService2
MicroService3
DB1
DB2
DB3

开发框架选择

- 微服务网关层
  - Web应用
    - HTTP API
    - REST API
- 微服务层
  - RPC应用
    - 开发更优化
    - 长连接（TCP）
    - 天然的注册中心

开发框架选择

- 微服务网关层

开发框架选择

- 微服务层

选择Spring Boot原因

- 传统开发框架（Spring）

大量XML配置信息
打包War
部署War到Web Server

选择Spring Boot原因

- Spring Boot

解决传统开发框架问题

轻量级

Spring Boot 特性

基于 Spring 轻量级

微内核 插件化 架构

JAR 独立 运行

注解 不需 配置

开箱 即用

生产 特性

Spring Boot插件化

- **RDBMS**
  - MySQL、PostgreSQL
- **NoSQL**
  - MongoDB、Redis
- **SQL API**
  - JDBC、JPA
- **MQ**
  - RabbitMQ
- **Transaction**
  - Atomikos
- **Template**
  - Velocity

Spring Boot使用场景

Spring Boot如何使用

- **搭建开发框架**
  - **Spring Boot**
    - http://projects.spring.io/spring-boot/
    - Version 1.4.3
  - **Maven**
    - http://maven.apache.org/
    - Version 3.2.1 +
  - **IDE**
    - Eclipse
    - IDEA

Spring Boot如何使用

- **开发步骤**
  - 创建一个Maven项目
  - 添加Spring Boot的Maven依赖配置
  - 创建一个Controller类
  - 创建一个Application类
  - Spring Boot编译打包
  - 运行Spring Boot应用程序

Spring Boot如何使用

- **Maven配置（pom.xml）**
  - 继承Spring Boot Parent Starter

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.3.RELEASE</version>
</parent>
```

  - 启用Web插件

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

Spring Boot如何使用

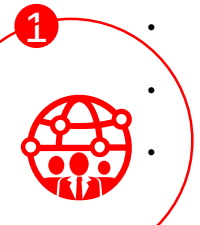
- **hello/HelloController.java（Controller类）**

```
package hello;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

 //@RestController注解，表示具备发布REST API能力，并将返回值自动序列化JSON
@RestController
public class HelloController {

    //@RequestMapping注解，表示方法和请求路径
    @RequestMapping("method=RequestMethod.GET, path=/hello")
   //Request: GET/hello；Response:  “Hello World!”
    String hello() {
        return "Hello World!";
    }
}
```



Spring Boot如何使用

- **hello/HelloApplication.java（Application类）**

```
package hello;

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

//定义为Spring Boot应用入口
@SpringBootApplication
public class HelloApplication {

public static void main(String[] args) throws Exception {
        //运行Spring Boot程序
        SpringApplication.run(HelloApplication.class, args);
    }
}
```

Spring Boot如何使用

- **Spring Boot编译打包**
  - **可部署的war包**
    - 可以直接部署到tomcat中
  - **可运行的jar包**
    - 直接使用Java命令运行

Spring Boot如何使用

- **生成Spring Boot可直接运行jar包**
  - **Maven插件依赖配置**

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

  - **运行java命令启动Spring Boot应用**
    - java -jar xxx.jar

Spring Boot生产级特性

- **开启Actuater插件**
  - **Maven依赖**

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

  - **端点（Endpoint）**
    - 发送相应请求到Actuator获取Spring Boot应用进程信息
    - GET方法类型，不带任何请求参数
    - www.zhuanzhuan.com:8080/metrics

Spring Boot生产级特性

- **端点**

端点	描述
autoconfig	获取自动配置信息
health	获取健康检查信息
metrics	获取性能指标信息
info	获取应用基本信息
trace	获取请求调用信息
dump	获取当前进程基本信息
……	……

Spring Boot生产级特性

- **端点设置**

  //关闭metrics端点

  - endpoints.metrics.enabled=false

  //修改metrics端点名称

  - endpoints.metrics.id=zzmetrics

  //关闭所有端点

  - endpoints.enable=false

Spring Boot生产级特性

- **查看健康检查**
  - **Request**
    - http://www.zhuanzhuan.com:8080/health
  - **Response**
    - {

      Status : “up”,

      - diskSpace: {

        status: “up”,

        total: 589357219003,

        free: 235457125902,

        threshold: 1048575 }

        }

RPC

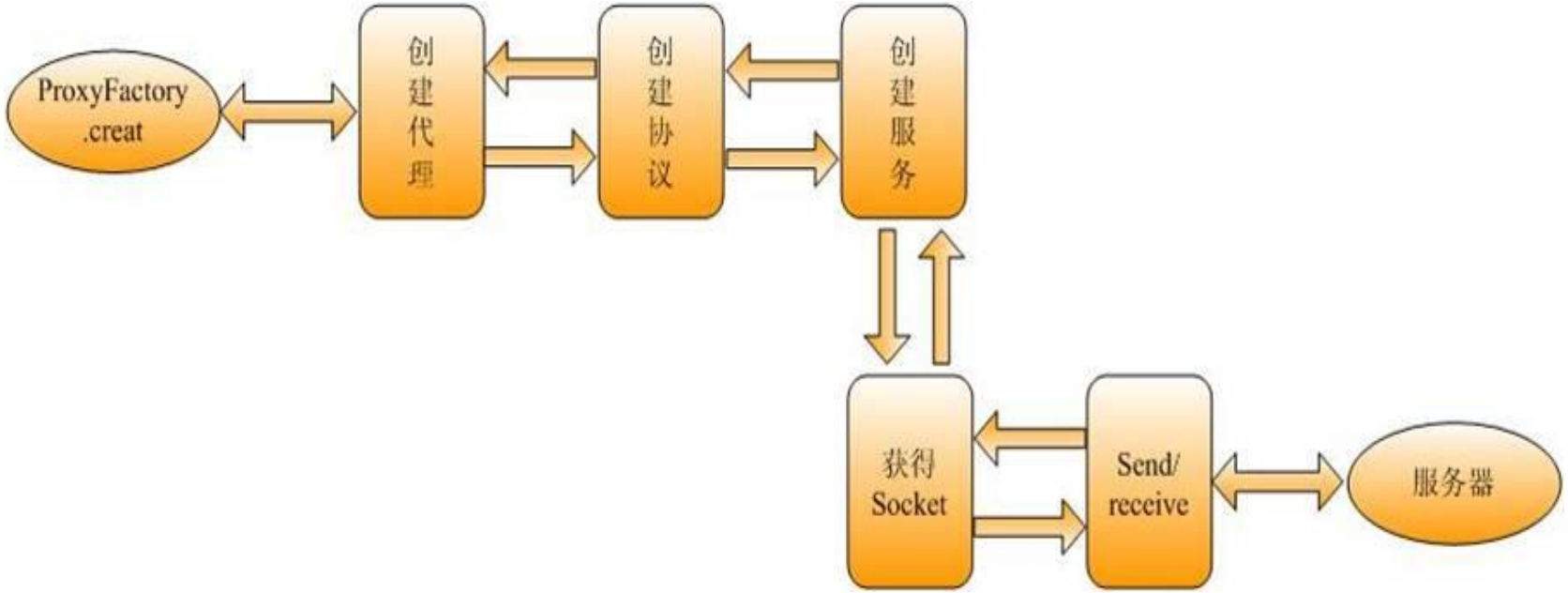
- **微服务**
  - **RPC**
    - Apache Thrift
    - Twitter Finagle
    - Google gRPC
    - Ali Dubbo
    - 自主研发
      - zzRPC

RPC

- **zzPRC构成**
  - **zzPRC Server**
    - 服务容器，宿主开发人员所开发SCF服务，接收和处理来自客户端请求。
  - **zzPRC Client**
    - 多种客户端，调用者就像在调用本地接口一样，提供负载均衡，容错等机制。
  - **zzPRC Serialize**
    - 提供了跨平台的二进制序列化解决方案。
  - **zzPRC Protocol**
    - 分为传输协议和数据协议。传输协议用于进行数据传输，数据协议用于请求和响应结果的内容数据。

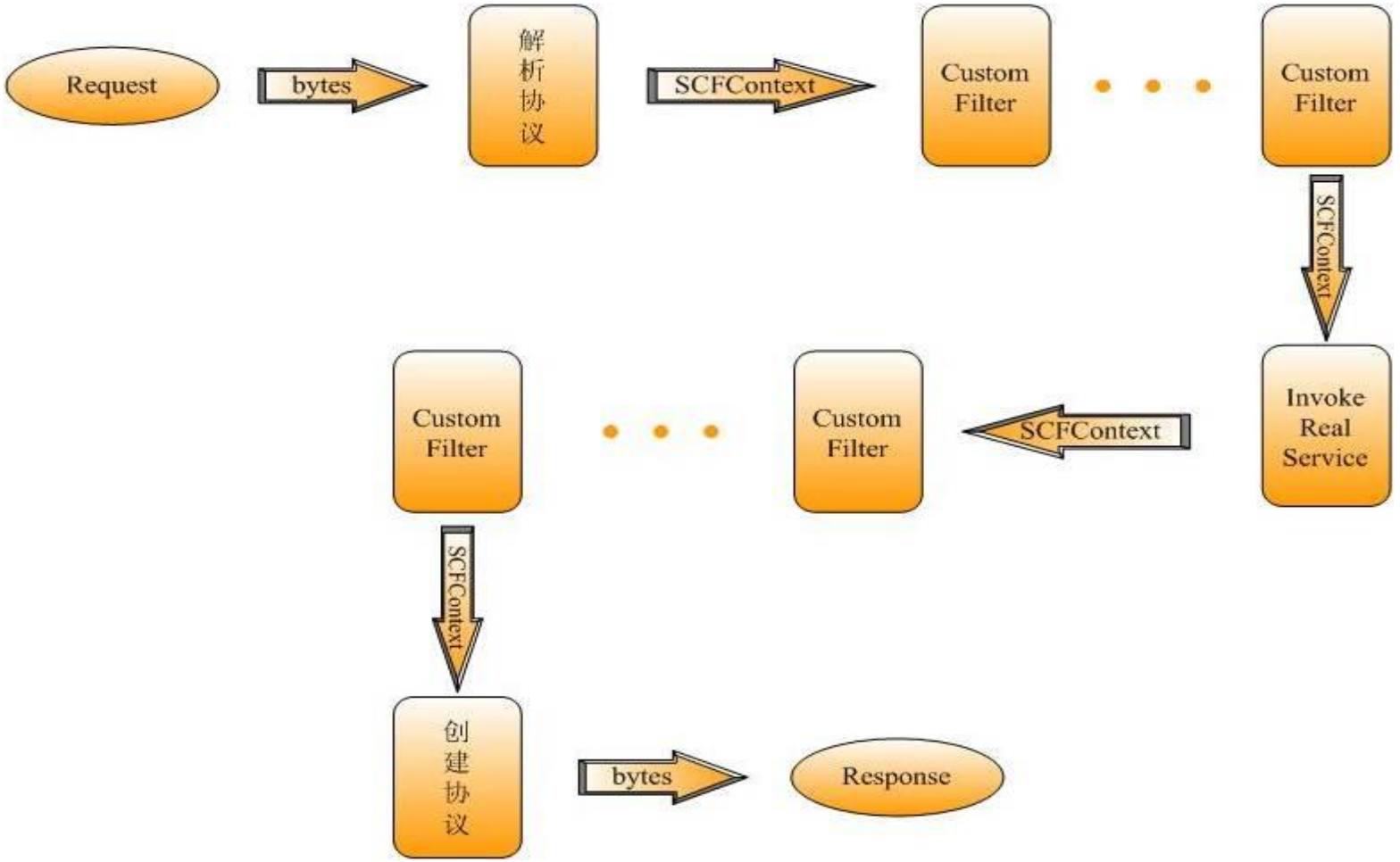
RPC

- **zzPRC Client流程图**



RPC

- **zzPRC Server流程图**



分享要点

微服务架构

1. 开发框架选择
2. 选择Spring Boot原因
3. 特点
4. 如何使用
5. 生产级特性
6. RPC

欢迎关注本人公众号 “架构之美”

Thanks!

让 生 活 更 简 单