# 引言

## 编写目的

根据《支持微服务架构的工业应用开发框架技术要求书》，此文档的主要目的是明确并完整记录支持微服务架构的工业应用开发框架的详细设计方案。

## 预期读者

本文档针对的读者主要是开发工程师、架构工程师以及运维工程师，包括对微服务感兴趣者也可以参考此文档了解微服务的设计原理。

## 阅读建议

阅读此文档的读者需要掌握Java编程语言，了解Vue、SpringBoot、SpringCloud、Jenkins、Docker和Kubernets等知识点。

## 参考资料

《深入理解Spring Cloud与微服务构建（第二版）》 --方志朋著

《微服务设计》 --Sam Newman著

《Jenkins权威指南》 --John Ferguson Smart著

《Kubernets权威指南（第四版）》 --龚正著

《疯狂流程引擎讲义（基于Activiti6.x的应用开发）》 --杨恩雄著

# 课题概述

## 微服务概述

本小节主要介绍微服务的由来，微服务能够解决什么问题，又会引入什么问题，什么是CAP理论，使用微服务应该遵循哪些原则。

### **什么是微服务**

微服务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的 API（例如 REST）集相互通讯，且每个服务可以被单独部署，简单举例：看军事新闻的读者应该都知道，一艘航空母舰作战能力虽然很强，但是弱点太明显，就是防御能力太差，单艘的航空母舰很少单独行动，通常航空母舰战斗群才是主要军事力量，你可以把单艘航母理解为的单体应用（防御差，机动性不好），把航母战斗群（调度复杂，维护费用高）理解为微服务。

它具备以下三个核心特点：

微服务为大型系统而生。随着业务的快速增长，会带来系统流量压力和复杂度的上升，系统的可维护性和可扩展性成为架构设计的主要考虑因素，微服务架构设计理念通过小而美的业务拆分，通过分而自治来实现复杂系统的优雅设计实现。

微服务架构是面向结果的。微服务架构设计风格的产生并非是出于学术或为标准而标准的设计，而是在软件架构设计领域不断演进过程中，面对实际工业界所遇到问题，而出现的面向解决实际问题的架构设计风格。

专注于服务的可替代性来设计。微服务架构设计风格核心要解决的问题之一便是如何便利地在大型系统中进行系统组件的维护和替换，且不影响整体系统稳定性。

### **微服务能够解决什么问题**

大部分的开发者经历和开发过单体应用，无论是传统的 Servlet + JSP，还是 SSM，还是现在的 SpringBoot，它们都是单体应用，那么单体应用有什么弊端？个人总结主要问题如下：

* 重复建设严重，日积月累会衍生出大量的独立应用系统
* 部署成本高（无论是修改1行代码，还是10行代码，都要全量替换）
* 改动影响大，风险高（不论代码改动多小，成本都相同）
* 因为成本高，风险高，所以导致部署频率低（无法快速交付客户需求）
* 无法满足快速扩容，弹性伸缩，无法适应云环境特性等问题

以上问题，都是微服务架构要解决的问题，至于微服务是如何解决的，在下一章节中会做详细介绍。