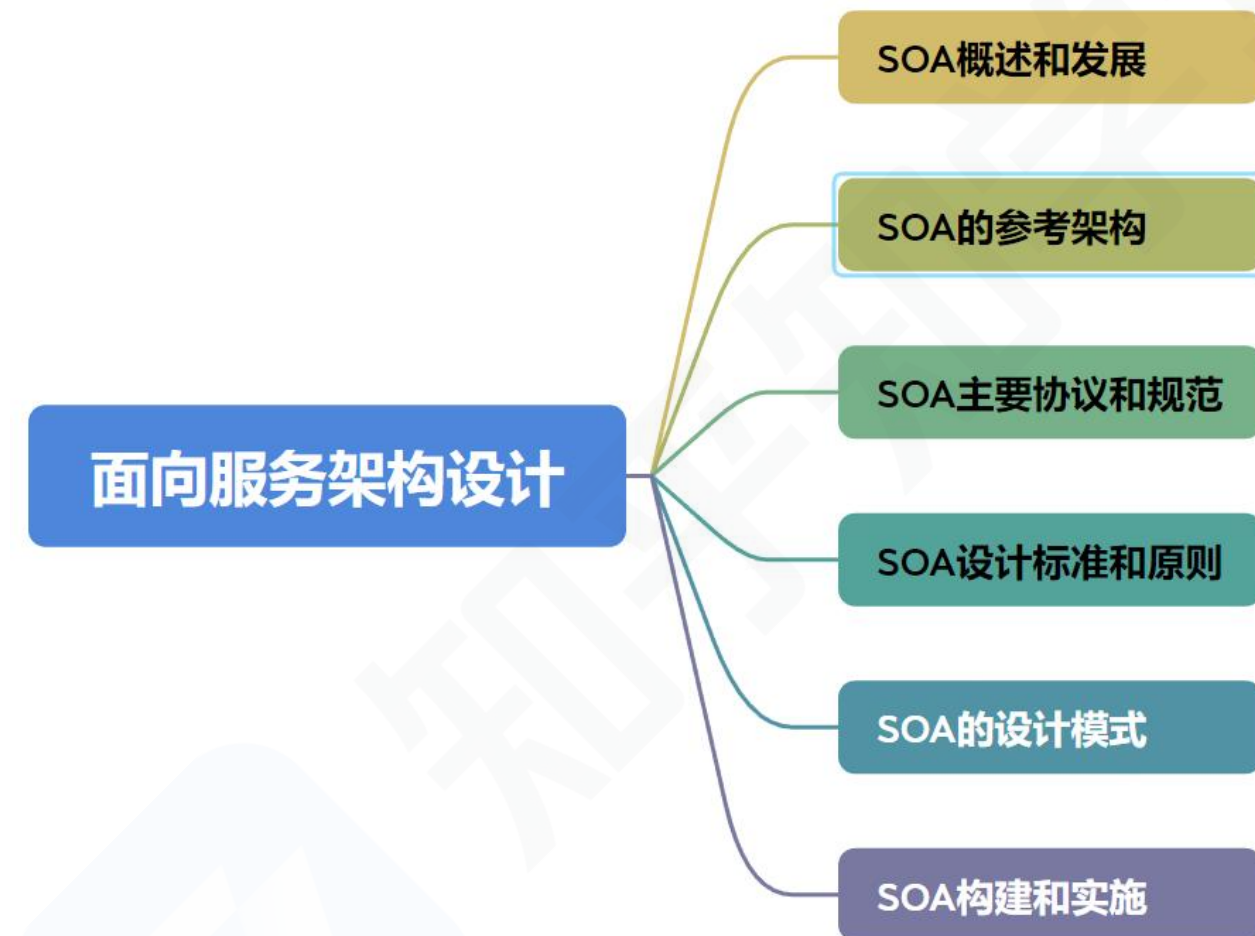


NEW PLAN







SOA概述和发展

- 在面向服务的体系结构(SOA)中，服务的概念有了延伸，泛指**系统对外提供的功能集**。
- 从应用的角度定义，可以认为**SOA是一种应用框架**，它着眼于日常的业务应用，并将它们**划分为单独的业务功能和流程**，即所谓的**服务**。SOA使用户可以构建、部署和整合这些服务，且无需依赖应用程序及其运行平台，从而提高业务流程的灵活性。
- 从软件的基本原理定义，可以认为**SOA是一个组件模型**，它将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。
- 业务流程是指为了**实现某种业务目的行为所进行的流程或一系列动作**。
- BPEL：面向Web 服务的业务流程执行语言，是一种**使用Web服务定义和执行业务流程的语言**。使用BPEL，用户可以通过组合、编排和协调Web服务自上而下地实现面向服务的体系结构。BPEL目前用于整合现有的Web Services，将现有的Web Services按照要求的业务流程整理成为一个新的WebServices，在这个基础上，形成一个从外界看来和单个Service一样的Service。



SOA概述和发展-面向服务和微服务区别

■ 粒度不同：

- 整体上来说，SOA 的服务粒度要粗一些，而微服务的服务粒度要细一些。例如，对一个大型企业来说，“员工管理系统”就是一个 SOA 架构中的服务；而如果采用微服务架构，则“员工管理系统”会被拆分为更多的服务，比如“员工信息管理”、“员工考勤管理”、“员工假期管理”和“员工福利管理”等更多服务。

■ 通信模式：

- SOA 采用了 ESB 作为服务间通信的关键组件，负责服务定义、服务路由、消息转换、消息传递，总体上是重量级的实现。
- 微服务推荐使用统一的协议和格式，例如，RESTful 协议、RPC 协议，无须 ESB 这样的重量级实现。

■ 服务交付：

- SOA 对服务的交付并没有特殊要求，因为 SOA 更多考虑的是兼容已有的系统；
- 微服务的架构理念要求“快速交付”，相应地要求采取自动化测试、持续集成、自动化部署等敏捷开发相关的最佳实践。如果没有这些基础能力支撑，微服务规模一旦变大（例如，超过 20 个微服务），整体就难以达到快速交付的要求，这也是很多企业在实行微服务时踩过的一个明显的坑，就是系统拆分为微服务后，部署的成本呈指数上升。

■ 应用场景：

- SOA 更加适合于庞大、复杂、异构的企业级系统，这也是 SOA 诞生的背景。这类系统的典型特征就是很多系统已经发展多年，采用不同的企业级技术，有的是内部开发的，有的是外部购买的，无法完全推倒重来或者进行大规模的优化和重构。因为成本和影响太大，只能采用兼容的方式进行处理，而承担兼容任务的就是 ESB。
- 微服务更加适合于快速、轻量级、基于 Web 的互联网系统，这类系统业务变化快，需要快速尝试、快速交付；同时基本都是基于 Web，虽然开发技术可能差异很大（例如，Java、C++、.NET 等），但对外接口基本都是提供 HTTP RESTful 风格的接口，无须考虑在接口层进行类似 SOA 的 ESB 那样的处理。

■ 因此，我们可以看到，SOA 和微服务本质上是两种不同的架构设计理念，只是在“服务”这个点上有交集而已。

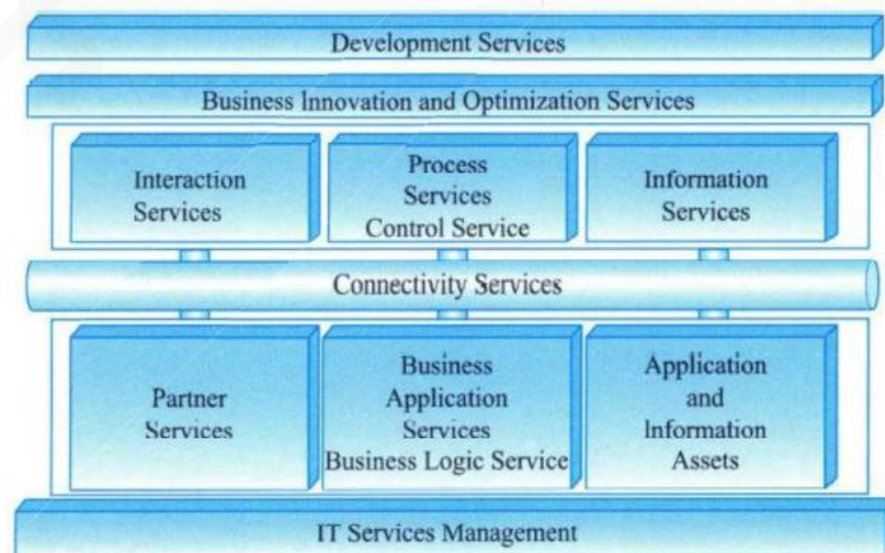
■ SOA 和微服务是两种不同理念的架构模式，并不存在孰优孰劣，只是应用场景不同而已。我们介绍 SOA 时候提到其产生历史背景是因为企业的 IT 服务系统庞大而又复杂，改造成本很高，但业务上又要求其互通，因此才会提出 SOA 这种解决方案。如果我们将微服务的架构模式生搬硬套到企业级 IT 服务系统中，这些 IT 服务系统的改造成本可能远远超出实施 SOA 的成本。

对比维度	SOA	微服务
服务粒度	粗	细
服务通信	重量级，ESB	轻量级，例如 HTTP RESTful
服务交付	慢	快
应用场景	企业级	互联网



SOA的参考架构

- 典型的以服务为中心的企业集成架构如下图所示，采用“**关注点分离**”的方法规划企业集成中的各种架构元素，同时从服务视角规划每种架构元素提供的服务，以及服务如何被组合在一起完成某种类型的集成。可划分为六大类：
- **业务逻辑服务 (Business Logic Service)**：包括用于实现业务逻辑的服务和执行业务逻辑的能力，其中包括业务应用服务 (Business Application Service)、业务伙伴服务 (PartnerService) 以及应用和信息资产 (Application and Information asset)。
- **控制服务 (Control Service)**：包括实现人 (People)、流程 (Process) 和信息 (Information) 集成的服务，以及执行这些集成逻辑的能力。
- **连接服务 (Connectivity Service)**：通过提供企业服务总线提供分布在各种架构元素中服务间的连接性。
- **业务创新和优化服务 (Business Innovation and Optimization Service)**：用于监控业务系统运行时服务的业务性能，并通过及时了解到的业务性能和变化，采取措施适应变化的市场。
- **开发服务 (Development Service)**：贯彻整个软件开发生命周期的开发平台，从需求分析，到建模、设计、开发、测试和维护等全面的工具支持。
- **IT服务管理 (IT Service Management)**：支持业务系统运行的各种基础设施管理能力或服务，如安全服务、目录服务、系统管理和资源虚拟化。





SOA的参考架构

■ 连接服务：通过企业服务总线(ESB)来实现，它的基本特征和能力包括：

- 描述服务的元数据和服务注册管理；
- 在服务请求者和提供者之间传递数据，以及对这些数据进行转换的能力，并支持由实践中总结出来的一些模式如同步模式、异步模式等；
- 发现、路由、匹配和选择的能力，以支持服务之间的动态交互，解耦服务请求者和服务提供者。
- 高级一些的能力，包括对安全的支持、服务质量保证、可管理性和负载平衡

■ 业务逻辑服务

- 整合已有应用(应用和信息访问服务)：实现对已有应用和信息的集成，主要有两类访问服务：可接入服务、事件发现服务。
- 整合新开发的应用(业务应用服务)：实现新应用集成，主要有三类业务应用服务：组件服务(可重用)、核心服务（运行时）、接口服务。
- 整合客户和业务伙伴(B2C/B2B，伙伴服务)：提供与企业外部的B2B的集成能力，包括：社区服务、文档服务、协议服务。

SOA的参考架构

■ 控制服务：

- **数据整合**(信息服务)：提供集成数据的能力，目前主要包括如下集中信息服务：联邦服务(不同类型数据聚合)、复制服务(远程数据本地访问)、转换服务(格式转换)、搜索服务。
- **流程整合**(流程服务)：完成业务流程集成，包括：编排服务(预定义流程顺序)、事务服务(保证ACID)、人工服务(人工活动集成到流程中)。
- **用户访问整合**(交互服务)：实现用户访问集成，包括：交付服务(运行时交互框架)、体验服务、资源服务(运行时交互组件的管理)。

■ 开发服务：开发环境和工具中为不同开发者的角色提供的功能被称为开发服务。根据开发过程中开发者角色和职责的不同，有如下4类服务：建模服务、设计服务、实现服务、测试服务。

■ 业务创新和优化：以业务性能管理(BPM)技术为核心提供业务事件发布、收集和关键业务指标监控能力。包括以下服务：

- 公共事件框架服务：通过一个公共事件框架提供IT和业务事件的激发、存储和分类等。
- 采集服务：通过基于策略的过滤和相关性分析检测感兴趣的服务。
- 监控服务：通过事件与监控上下文间的映射，计算和管理业务流程的关键性能指标。

■ IT服务管理：为业务流程和服务提供安全、高效和健康的运行环境，包括：安全和目录服务、系统管理和虚拟化服务。



SOA主要协议和规范

- Web服务最基本的协议包括**UDDI**、**WSDL**和**SOAP**，通过它们，可以提供直接而又简单的Web Service支持,如图所示。
- UDDI: UDDI 是用于服务的注册和发现。它允许提供者将其 Web 服务描述注册到 UDDI 注册中心，以便其他用户可以发现并访问这些服务。
- 示例：假设有一个电子商务网站，它提供了一组 Web 服务，包括产品搜索、购物车管理和订单处理。这些服务的描述信息可以被注册到 UDDI 注册中心，以便其他电子商务网站或应用程序可以查找并使用这些服务。
- WSDL: WSDL 是用于用于描述服务的接口和操作。它提供了一种标准方式来定义服务的输入、输出以及如何与服务进行交互。
- 示例：假设一个天气预报 Web 服务，它提供了获取特定城市天气信息的功能。WSDL 文件将定义该服务的接口，包括输入参数（城市名称）和输出（天气信息）。其他开发人员可以使用这个 WSDL 文件来了解如何调用该服务。
- SOAP: SOAP 是一种用于在不同计算机之间进行通信的协议(在不同服务之间进行消息交换)。它允许 Web 服务之间的消息交换，并提供了一种标准的、跨平台的通信机制。一般来说是一个XML文件
- 示例：考虑一个在线支付系统，它需要与信用卡验证 Web 服务进行通信以验证用户的支付信息。在此过程中，SOAP 协议可以用于创建包含用户支付信息的信息，并将其发送到信用卡验证服务。该服务将使用相同的 SOAP 协议返回验证结果。





SOA主要协议和规范

■ REST (Representational State Transfer) 是一种用于构建分布式系统的架构风格和设计原则，它强调使用现有的 Web 标准和协议，并通过资源的状态表示来实现通信。REST 常用于构建 Web 服务和 API。以下是 REST 规范的核心原则：

- 资源：在 REST 中，一切都是资源，如网页、图片、文件等。每个资源都有一个唯一的地址 (URL) 来标识它。
- 表现层：资源的状态以某种格式表示，通常是 JSON 或 XML。例如，一篇博客文章可以以 JSON 形式表示其标题和内容。
- HTTP 方法：HTTP 方法 (如 GET、POST、PUT、DELETE) 用于对资源进行操作。GET 用于获取资源，POST 用于创建资源，PUT 用于更新资源，DELETE 用于删除资源。
- 无状态：REST 是无状态的，每个请求都包含足够的信息，服务器不需要保存客户端的状态。
- 客户端-服务器：客户端负责请求资源，服务器负责提供资源。这种分离性使系统更容易扩展和维护。
- 统一接口：REST 使用统一的接口规则，使客户端和服务端能够彼此理解。

■ 举例：假设你使用 REST 架构构建一个博客系统：

- 每篇博客文章是一个资源，每篇文章都有一个唯一的 URL。
- 博客文章的信息以 JSON 形式表示，包括标题、内容、作者等。
- 你可以使用 HTTP GET 请求来获取文章，HTTP POST 请求来创建新文章，HTTP PUT 请求来更新文章，HTTP DELETE 请求来删除文章。
- 服务器不需要保存客户端的状态，每个请求都是独立的。
- 客户端通过 HTTP 请求获取博客文章，服务器通过 HTTP 响应提供博客文章。

■ 只要遵循 REST 设计思想同时满足设计约束的一类架构设计或者应用程序的统称，这一类都可以称之为 RESTful



SOA设计标准和原则

■ SOA设计的要求

- 文档标准化：XML文档，Web描述语言。
- 通信协议标准：用消息进行通信，消息使用XML Schema来定义。
- 应用程序统一登记与集成：通过扮演目录列表角色的登记处来维护，UDDI是标准。

■ 服务质量(QoS)：每项SOA服务都有一个与之相关的服务质量。QoS的一些关键元素有安全需求(例如认证和授权)、可靠通信以及谁能调用服务的策略。其服务和标准包括：

- 可靠性：“仅且仅仅传送一次”“最多传送一次”“重复消息过滤”和“保证消息传送”等特性消息的发送和确认。
- 安全性：主要包括认证交换、消息完整性和消息保密。
- 策略：服务提供者有时候会要求服务消费者与某种策略通信。
- 控制：在SOA中，进程是使用一组离散的服务创建的。
- 管理：让系统管理员管理所有，运行在多种环境下的服务的管理系统。



SOA设计标准和原则

■ SOA的设计原则

- **无状态**。调用服务的时候不用考虑到它还需要其他的状态及数据，以避免服务请求者依赖于服务提供者的状态。
- **单一实例**。每个服务都只提供单一的功能，避免功能冗余。
- **明确定义的接口**。使用者依赖服务规约调用服务，所以服务定义必须长时间稳定，一旦公布，不能随意更改;服务的定义应尽可能明确，减少使用者的不适当使用;不要让使用者看到服务内部的私有数据。
- **自包含和模块化**。服务封装了那些在业务上稳定、重复出现的活动和组件，实现服务的功能实体是完全独立自主的，独立进行部署、版本控制、自我管理和恢复。
- **粗粒度**。服务数量不应该太大，依靠消息交互而不是远程过程调用(RPC)，通常消息量比较大，但是服务之间的交互频度较低。
- **服务之间的松耦合性**。服务使用者看到的是服务的接口，其位置、实现技术和当前状态等对使用者是不可见的，服务私有数据对服务使用者是不可见的。
- **重用能力**。服务应该是可以重用的。
- **互操作性、兼容和策略声明**。为了确保服务规约的全面和明确，策略成为一个越来越重要的方面。这可以是技术相关的内容，例如一个服务对安全性方面的要求;也可以是跟业务有关的语义方面的内容，例如需要满足的费用或者服务级别方面的要求，这些策略对于服务在交互时是非常重要的。

SOA设计模式

■ 服务注册表模式，支持如下SOA治理功能：

- 服务注册：应用开发者，也叫服务提供者，向注册表公布他们的功能。
- 服务位置：也就是服务应用开发者，帮助他们查询注册服务，寻找符合自身要求的服务。
- 服务绑定：服务的消费者利用检索到的服务合同来开发代码，开发的代码将与注册的服务绑定、调用注册的服务以及与他们实现互动。

■ 企业服务总线模式，由中间件技术实现的支持面向服务架构的基础软件平台，支持异构环境中的服务以基于消息和事件驱动模式的交互，并且具有适当的服务质量和可管理性。

■ 一个典型的在ESB环境中组件之间的交互过程是：首先由服务请求者触发一次交互过程，产生一个服务请求消息，并将该消息按照ESB的要求标准化，然后标准化的消息被发送给服务总线。ESB根据请求消息中的服务名或者接口名进行目的组件查找，将消息转发至目的组件，并最终将处理结果逆向返回给服务请求者。这种交互过程不再是点对点的直接交互模式，而是由事件驱动的消息交互模式。

■ ESB的核心功能如下。

- 提供位置透明性的消息路由和寻址服务。
- 提供服务注册和命名的管理功能。
- 支持多种消息传递范型(如请求/响应、发布/订阅等)。
- 支持多种可以广泛使用的传输协议。
- 支持多种数据格式及其相互转换。
- 提供日志和监控功能。



SOA的设计模式

- **微服务模式**，不再强调传统SOA架构里面比较重的ESB企业服务总线，同时**SOA 的思想进入到单个业务系统内部实现真正的组件化。**
- **微服务模式特点：**复杂应用解耦、独立、技术选型灵活、容错、松耦合易扩展。
- **常见的微服务设计模式：**
 - **聚合器微服务：**聚合器调用**多个微服务实现系统应用程序所需功能**，具体有两种形式，一种是将检索到的**数据信息进行处理**并直接展示;另一种是对获取到的数据信息增加业务逻辑处理后，再进一步发布成一个新的微服务作为一个更高层次的组合微服务，相当于从服务消费者转换成服务提供者。
 - **链式微服务：**客户端或服务在收到请求后，会返回一个经过合并处理的响应，服务之间形成一条调用链。
 - **数据共享微服务：**当服务之间存在强耦合关系时，可能存在多个微服务共享缓存与数据库存储的现象。
 - **异步消息传递微服务：**消息队列将消息写入一个消息队列中，实现业务逻辑以异步方式运行，从而加快系统响应速度。
- **微服务架构的问题与挑战：**
 - 微服务架构分布式特点带来的复杂性;
 - 微服务架构的**分区数据库体系**，不同服务拥有不同数据库;
 - 增加了**测试的复杂性**;
 - 在**大规模应用部署中**，在监控、管理、分发及扩容等方面,也带来了不少复杂性



SOA构建和实施

■ SOA构建注意问题

- **原有系统架构中的集成需求**：当SOA架构师遇到一个十分复杂的企业系统时，首先考虑的应该是如何重用已有的投资而不是替换遗留系统。集成类型包括：应用程序集成的需求，终端用户界面集成的需求，流程集成的需求以及已有系统信息集成的需求。
- **服务粒度的控制以及无状态服务的设计**：SOA系统中服务的构建有两点需要特别注意的地方；首先是对于服务粒度的控制，另外就是对于无状态服务的设计。

- SOA的实施过程：从以下三个方面选择SOA最佳的解决方案：**尽量选择能进行全局规划的方案、选择时充分考虑企业自身的需求、从平台实施等技术方面进行考察。**

■ 业务流程分析过程：

1. **建立服务模型**

- 自顶向下分解法：从业务着手进行分析，选择端到端的业务流程进行逐层分解至业务活动。
- 业务目标分析法：通过关键性能指标分析来验证已有服务候选者以及发现遗漏的服务候选者。
- 自底向上分析法：利用已有资产来实现服务。

2. **建立业务流程**

- 建立业务对象：业务对象是对数据进行检索和处理的组件，是简单的真实世界的软件抽象。
- 建立服务接口。
- 建立业务流程：流程是指定的活动顺序，包含明确确定的用于提供业务值的输入和输出。

THE END

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

