微服务

(概念、架构与实现)

徐悦甡

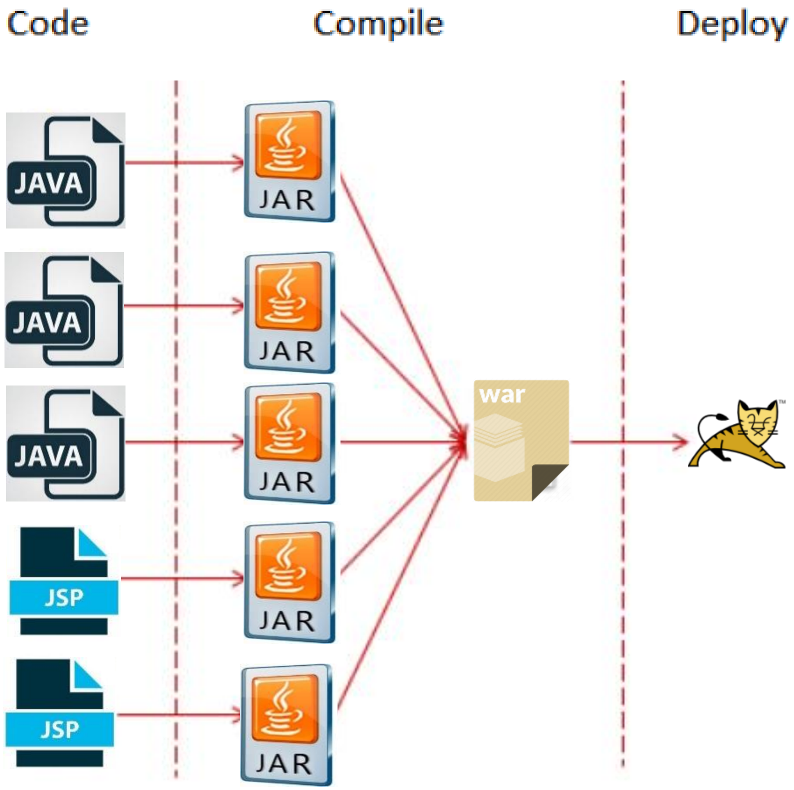
计算机科学与技术学院

西安电子科技大学

目录

- 微服务起源
- 微服务概念
- 架构演化
- 微服务架构
- 微服务实现
- 参考资料

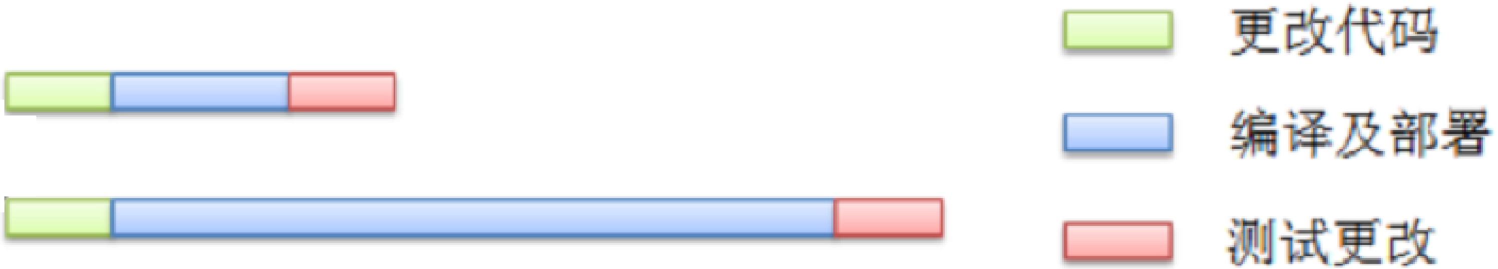
微服务起源


Code
Compile
Deploy
JAVA
JAVA
JAVA
JSP
JSP
JAR
JAR
JAR
JAR
JAR
war

- 传统项目开发，要开发的服务所对应的代码由多个项目所组成，各个项目会根据自身所提供功能的不同具有明确的边界；
- 在编译时，这些项目将被打包成为JAR包，并最终合并在一起形成一个WAR包；
- 接下来，需要将该WAR包上传到Web容器，解压WAR包，并重新启动服务器；
- 在执行完这一系列操作之后，对服务的编译与部署即已完成。

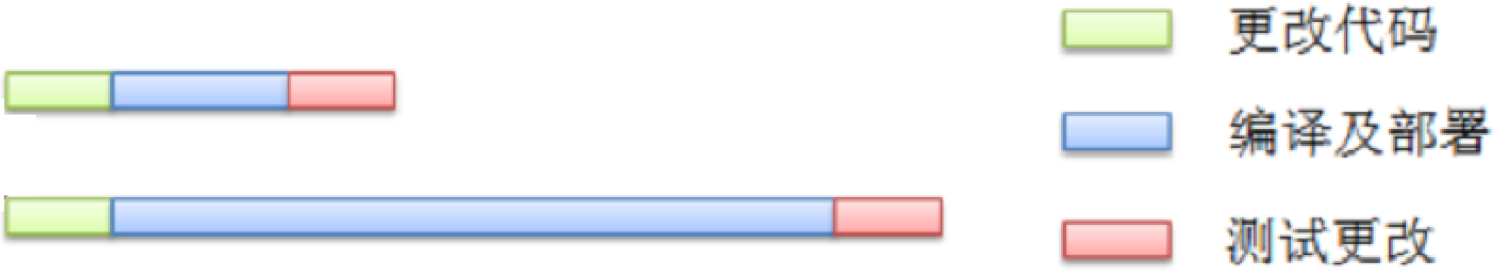
微服务起源

- 这种将所有的代码及功能都包含在一个WAR包中的项目组织方式被称为Monolith
- 在项目较小的情况下，这种代码组织方式可以接受：
  - 更改完代码后，编译器编译代码，然后软件开发人员花费少量时间部署编译出来的WAR包以便测试自己刚刚所做的更改。
- 但随着项目的逐渐变大，整个开发流程的时间也会变得很长：
  - 即使在仅仅更改了一行代码的情况下，软件开发人员需要花费几十分钟甚至超过一个小时的时间对所有代码进行编译，并接下来花费大量的时间重新部署刚刚生成的产品，以验证自己的更改是否正确。



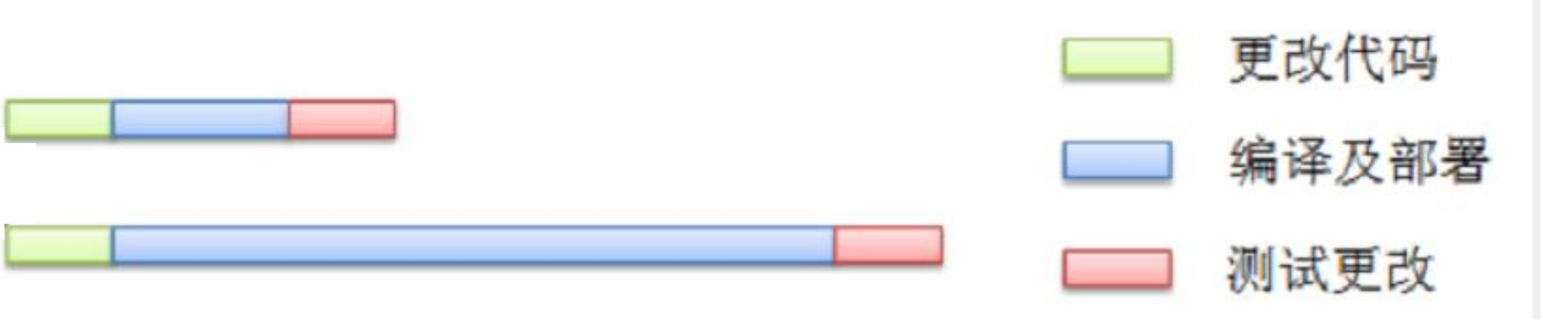
微服务起源

- 如果应用的部署非常麻烦，那么为了对自己的更改进行测试，软件开发人员还需要在部署前进行大量的环境设置，进而使得软件开发人员的工作变得繁杂；
- 在应用变大之后，软件开发人员花在编译及部署的时间明显增多，甚至超过了对代码进行更改并测试的时间，效率低下。

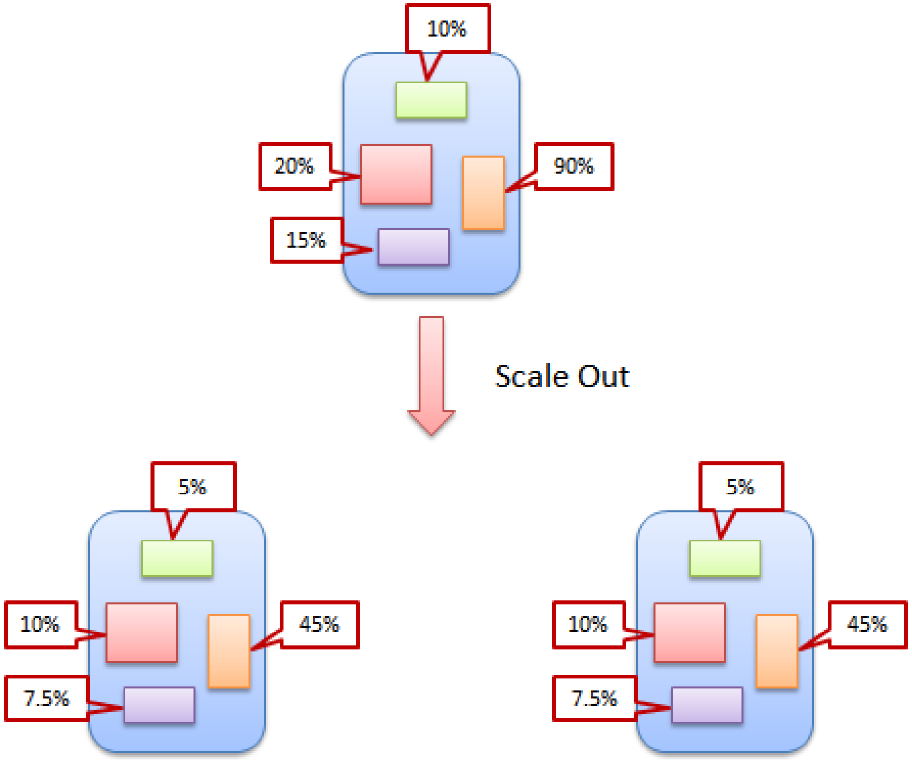


微服务起源

- 在变得越来越大的同时，所使用的技术也会变得越来越多；
- 这些技术有些是不兼容的，比如在一个项目中大范围地混合使用C++和Java，即较为困难；
- 在这种情况下，就需要抛弃对某些不兼容技术的使用，而选择一种不适合的技术来实现特定的功能；
- 除此之外，由于按照Monolith组织的代码将只产生一个包含了所有功能的WAR包，因此在对服务的容量进行扩展的时候，只能选择重复地部署WAR包以扩展服务能力，而不是仅仅扩展出现系统瓶颈的部分。

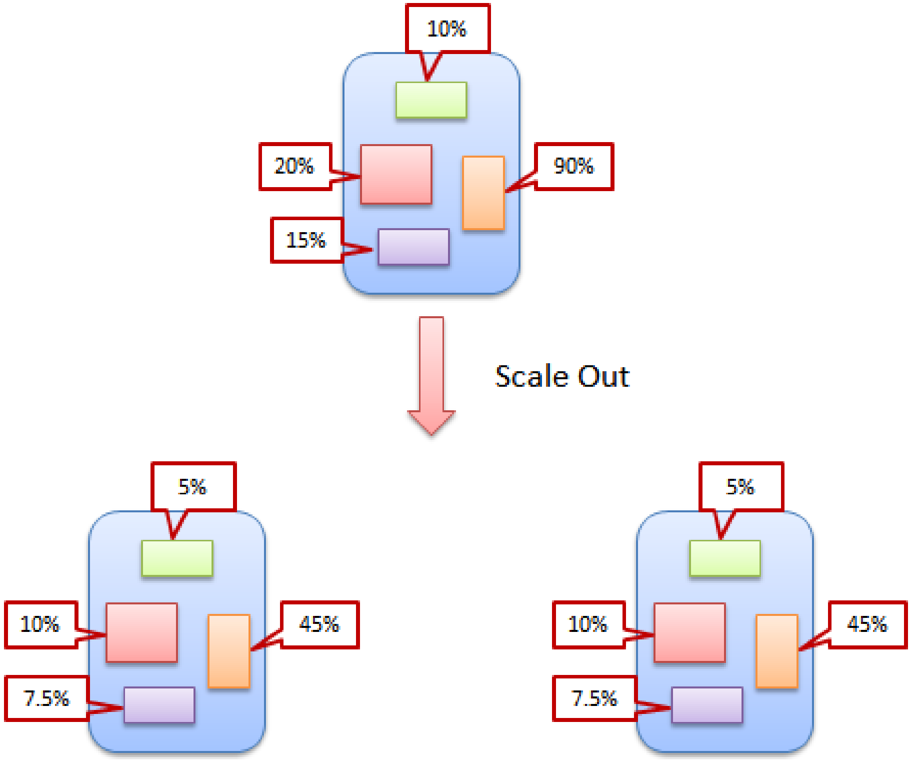


微服务起源


10%
20%
90%
15%
Scale Out
5%
10%
45%
7.5%
5%
10%
45%
7.5%

- 以上这种扩展方式极大地浪费了资源；
- 以上图所展示的情况为例：
  - 在一个服务中，某个组成的负载已经达到了90%，到了不得不对服务能力进行扩容的时候；
  - 而同一服务的其它三个组成的负载还没有到其处理能力的20%。

微服务起源


10%
20%
90%
15%
Scale Out
5%
10%
45%
7.5%
5%
10%
45%
7.5%

- 由于Monolith服务中的各个组成是打包在同一个WAR包中，因此通过添加一个额外的服务实例，虽然可以将需要扩容的组成的负载降低到了45%，
- 但也使得其它各部分的利用率更为低下。

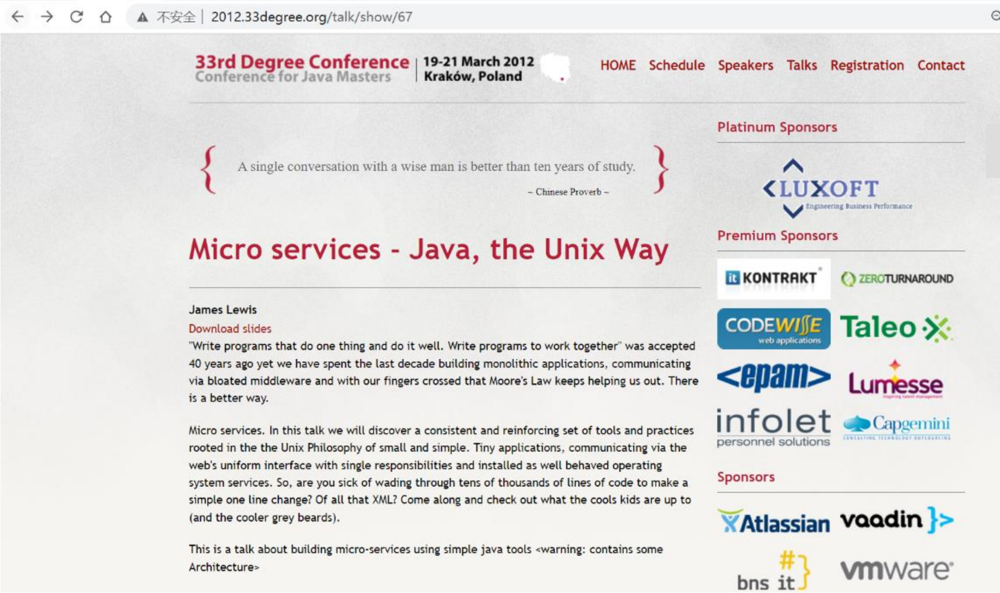
目录

- 微服务起源
- 微服务概念
- 架构演化
- 微服务架构
- 微服务实现
- 参考资料

微服务概念

- James Lewis
  - Micro services - Java, the Unix Way， 2012
  - http://2012.33degree.org/talk/show/67
    - 微服务的代表性特征
    - 单一服务职责
    - 康威Law（Conway's Law）
    - DDD

Principal Consultant at Thoughtworks and member of the Technology Advisory Board


不安全 | 2012.33degree.org/talk/show/67
33rd Degree Conference
Conference for Java Masters
19-21 March 2012
Kraków, Poland
HOME
Schedule
Speakers
Talks
Registration
Contact
Platinum Sponsors
LUXOFT
Engineering Business Performance
Premium Sponsors
KONTRAKT
ZEROTURNAROUND
CODEWISE web applications
Taleo
epam
Lumesse
infolet personnel solutions
Capgemini
Sponsors
Atlassian
vaadin
bns it
vmware

{ A single conversation with a wise man is better than ten years of study. }
~ Chinese Proverb ~

Micro services - Java, the Unix Way

James Lewis

Download slides

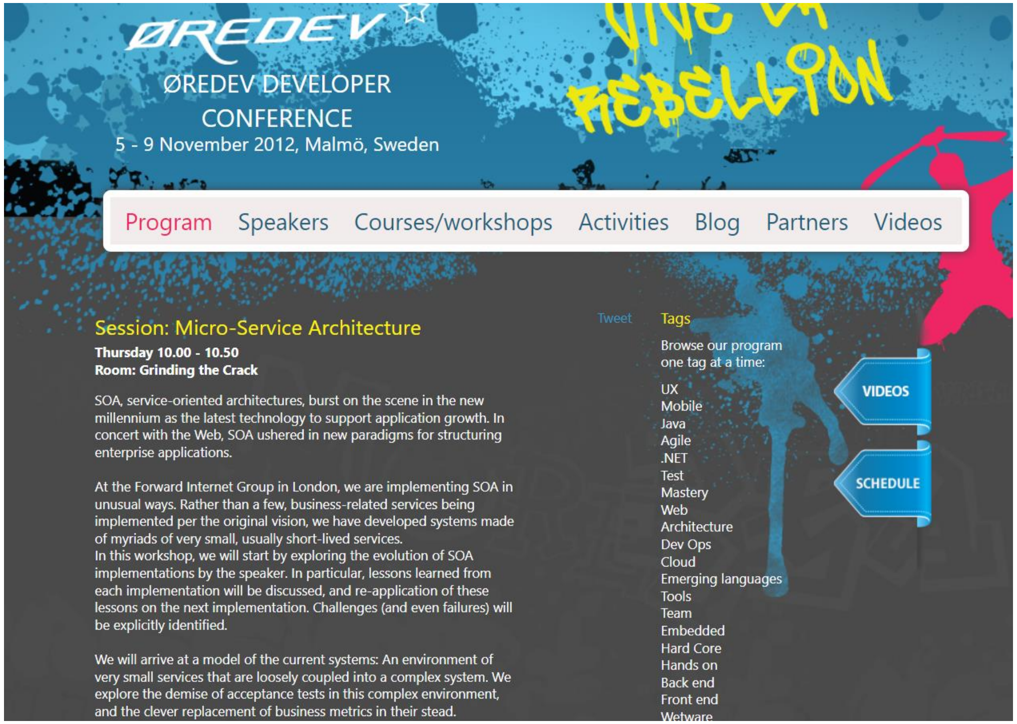
"Write programs that do one thing and do it well. Write programs to work together" was accepted 40 years ago yet we have spent the last decade building monolithic applications, communicating via bloated middleware and with our fingers crossed that Moore's Law keeps helping us out. There is a better way.

Micro services. In this talk we will discover a consistent and reinforcing set of tools and practices rooted in the the Unix Philosophy of small and simple. Tiny applications, communicating via the web's uniform interface with single responsibilities and installed as well behaved operating system services. So, are you sick of wading through tens of thousands of lines of code to make a simple one line change? Of all that XML? Come along and check out what the cools kids are up to (and the cooler grey beards).

This is a talk about building micro-services using simple java tools <warning: contains some Architecture>

微服务概念

- □ 微服务架构雏形
  - ➢ Fred George
  - ➢ http://oredev.org/oredev2012/2012/sessions/micro-service-architecture.html, 2012


ØREDEV
ØREDEV DEVELOPER CONFERENCE
5 - 9 November 2012, Malmö, Sweden
VIVE LA REBELLION
Program
Speakers
Courses/workshops
Activities
Blog
Partners
Videos
Tweet
VIDEOS
SCHEDULE

Session: Micro-Service Architecture

Thursday 10.00 - 10.50
Room: Grinding the Crack

SOA, service-oriented architectures, burst on the scene in the new millennium as the latest technology to support application growth. In concert with the Web, SOA ushered in new paradigms for structuring enterprise applications.

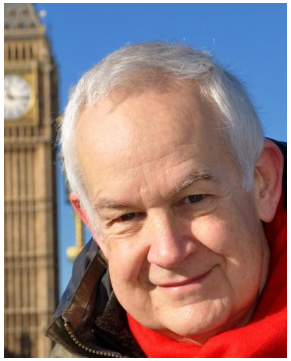
At the Forward Internet Group in London, we are implementing SOA in unusual ways. Rather than a few, business-related services being implemented per the original vision, we have developed systems made of myriads of very small, usually short-lived services.
In this workshop, we will start by exploring the evolution of SOA implementations by the speaker. In particular, lessons learned from each implementation will be discussed, and re-application of these lessons on the next implementation. Challenges (and even failures) will be explicitly identified.

We will arrive at a model of the current systems: An environment of very small services that are loosely coupled into a complex system. We explore the demise of acceptance tests in this complex environment, and the clever replacement of business metrics in their stead.

Tags

Browse our program one tag at a time:

UX
Mobile
Java
Agile
.NET
Test
Mastery
Web
Architecture
Dev Ops
Cloud
Emerging languages
Tools
Team
Embedded
Hard Core
Hands on
Back end
Front end
Wetware



微服务概念

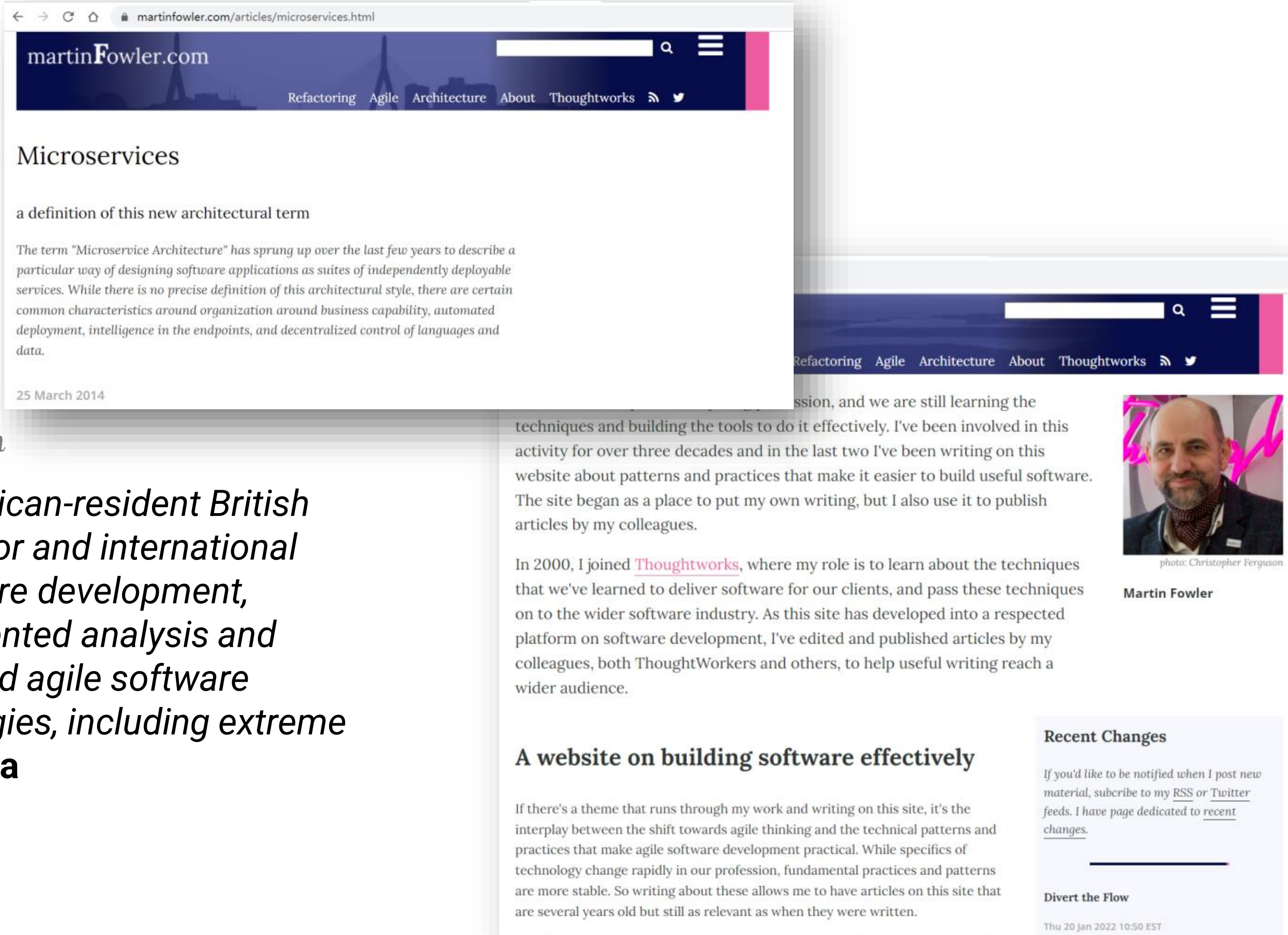
XIDIAN UNIVERSITY

□ 微服务架构正式诞生：

- ➢ https://martinfowler.com/articles/microservices.html, 2014

photo: Christopher Ferguson

Martin Fowler is an American-resident British software developer, author and international public speaker on software development, specialising in object-oriented analysis and design, UML, patterns, and agile software development methodologies, including extreme programming. ~**Wikipedia**



Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

ssion, and we are still learning the techniques and building the tools to do it effectively. I've been involved in this activity for over three decades and in the last two I've been writing on this website about patterns and practices that make it easier to build useful software. The site began as a place to put my own writing, but I also use it to publish articles by my colleagues.

In 2000, I joined Thoughtworks, where my role is to learn about the techniques that we've learned to deliver software for our clients, and pass these techniques on to the wider software industry. As this site has developed into a respected platform on software development, I've edited and published articles by my colleagues, both ThoughtWorkers and others, to help useful writing reach a wider audience.

photo: Christopher Ferguson

Martin Fowler

A website on building software effectively

If there's a theme that runs through my work and writing on this site, it's the interplay between the shift towards agile thinking and the technical patterns and practices that make agile software development practical. While specifics of technology change rapidly in our profession, fundamental practices and patterns are more stable. So writing about these allows me to have articles on this site that are several years old but still as relevant as when they were written.

Recent Changes

If you'd like to be notified when I post new material, subcribe to my RSS or Twitter feeds. I have page dedicated to recent changes.

Divert the Flow

Thu 20 Jan 2022 10:50 EST

微服务概念

- ThoughtWorks的首席科学家，马丁·福勒（Martin Fowler）：
  - *微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于HTTP协议的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。*

目录

- 微服务起源
- 微服务概念
- 架构演化
- 微服务架构
- 微服务实现
- 参考资料

架构演化

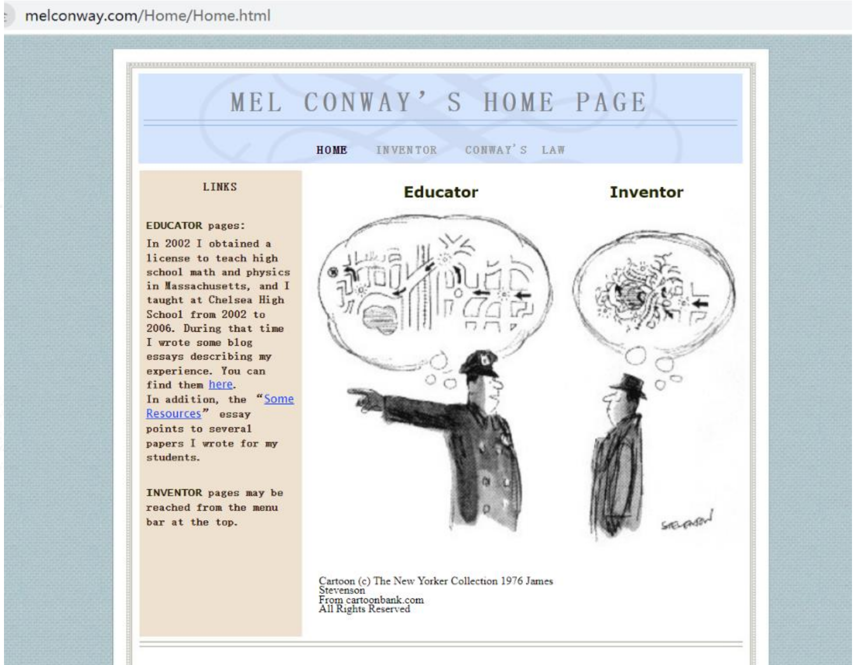
- 康威Law（Conway‘s Law），Melvin Edward Conway
  - Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations. - Melvin Conway(1967)
  - 设计系统的组织，其产生的设计等同于组织之内、组织之间的沟通结构
  - 一种对于系统划分与团队构建之间关系的观点

design organization criteria

HOW DO COMMITTEES INVENT?

by MELVIN E. CONWAY

DATAMATION


melconway.com/Home/Home.html
MEL CONWAY'S HOME PAGE
HOME
INVENTOR
CONWAY'S LAW
LINKS
Educator
Inventor
STEVENSON

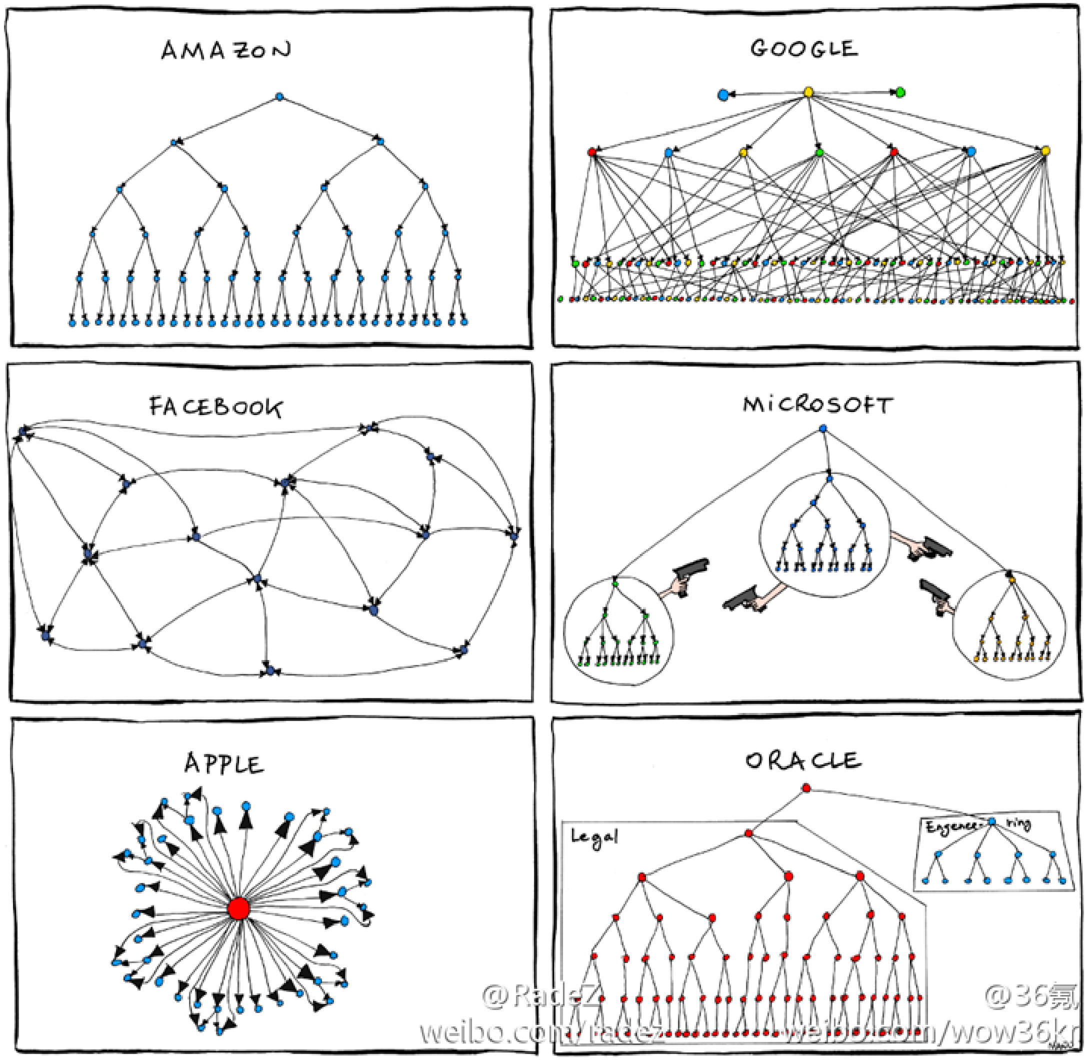
EDUCATOR pages:

In 2002 I obtained a license to teach high school math and physics in Massachusetts, and I taught at Chelsea High School from 2002 to 2006. During that time I wrote some blog essays describing my experience. You can find them here.
In addition, the “Some Resources” essay points to several papers I wrote for my students.

INVENTOR pages may be reached from the menu bar at the top.

Cartoon (c) The New Yorker Collection 1976 James Stevenson
From cartoonbank.com
All Rights Reserved

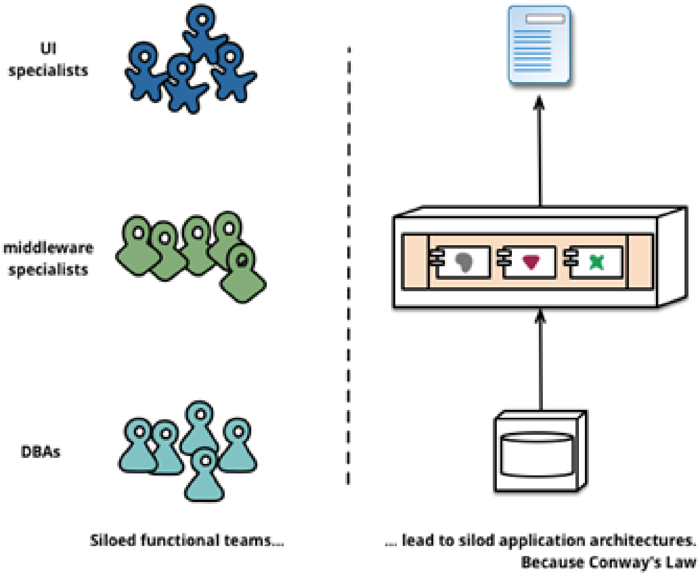
架构演化

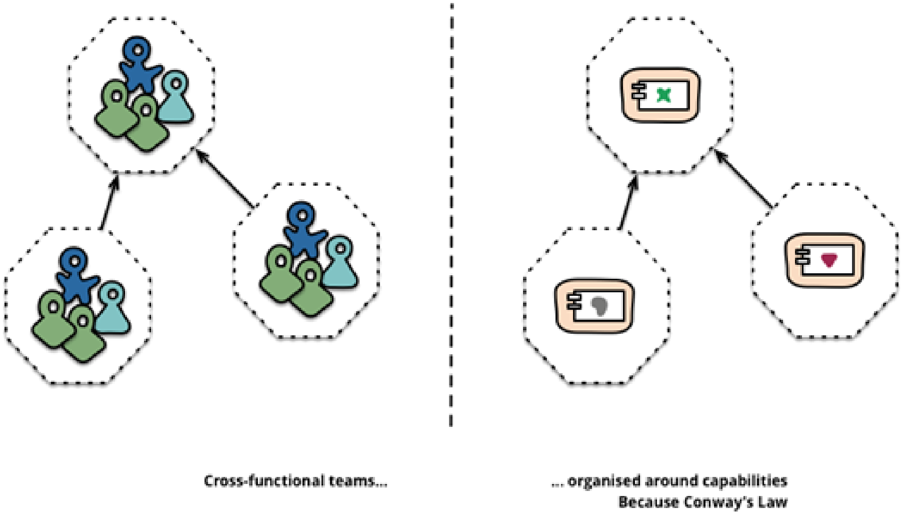

AMAZON
GOOGLE
FACEBOOK
MICROSOFT
APPLE
ORACLE
Legal
Engineering
@RadeZ weibo.com/radez
@36氪 weibo.com/wow36kr

- Communication dictates design（组织沟通方式会通过系统设计表达出来）
- There is a homomorphism from the linear graph of a system to the linear graph of its design organization（线型系统和线型组织架构间有潜在的异质同态特性）
- The structures of large systems tend to disintegrate during development, qualitatively more so than with small systems（大的系统组织总是比小系统更倾向于分解）
- 分而治之

架构演化

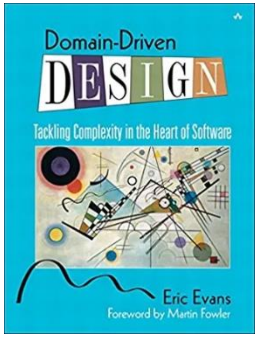
- 想要什么样的系统，就搭建什么样的团队；
- 如果团队分成前端团队，Java后台开发团队，DBA团队，运维团队，系统就会长成下面的样子：
- 如果系统是按照业务边界划分的，大家按照一个业务目标去把自己的模块做出小系统，小产品的话，系统就会长成下面的样子，即微服务架构。

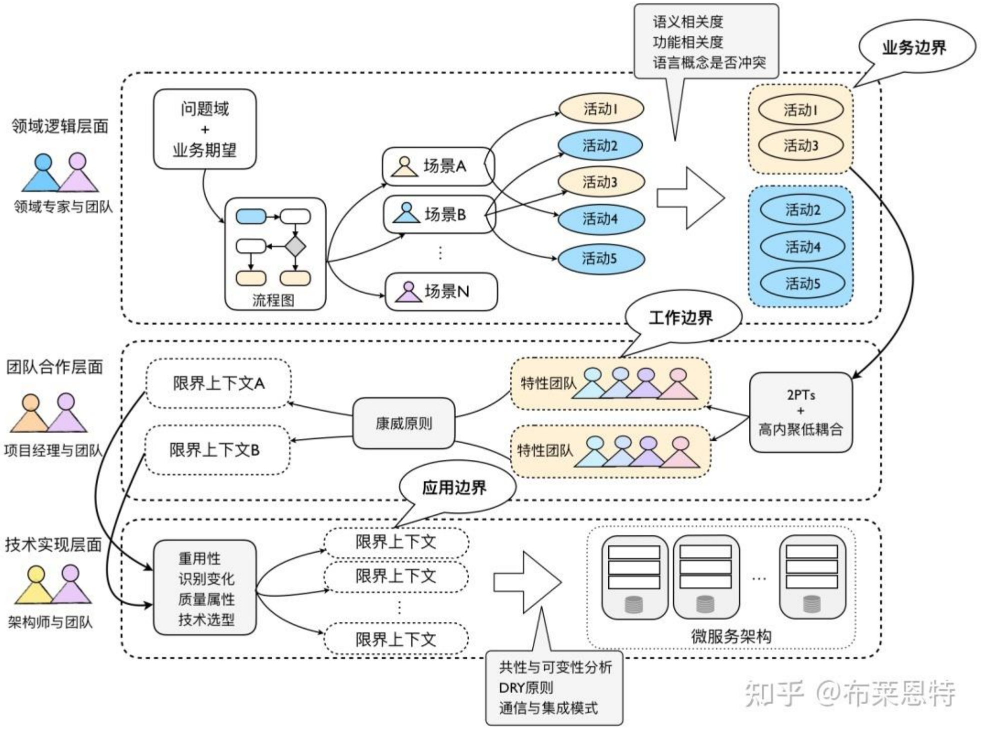

UI specialists
middleware specialists
DBAs
Siloed functional teams...
... lead to silod application architectures.
Because Conway's Law


Cross-functional teams...
... organised around capabilities
Because Conway's Law

架构演化

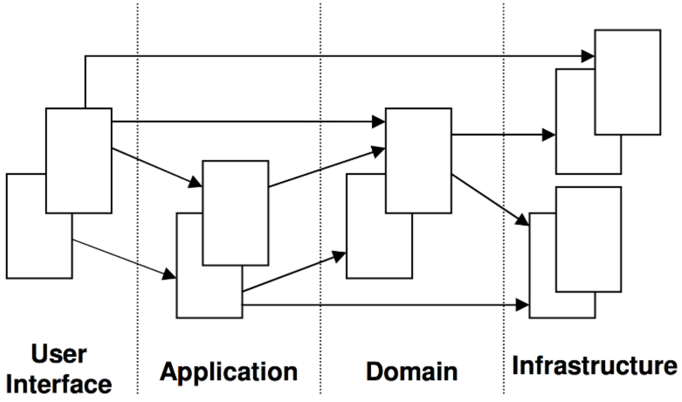
- DDD（Domain-driven Design）领域驱动的设计
  - Eric Evans
  - 《Domain-driven design –tackling complexity in the heart of software》（领域驱动设计—软件核心复杂性应对之道）
  - 瀑布式，基于现有需求整理出一个基本模型，再将结果传递给开发人员
  - 敏捷式，范围会设计到更精细的业务模块，小步迭代，周期性交付
  - DDD: 领域模型(Domain Model),领域模型无关技术，具有高度的业务抽象性，能够精确的描述领域中的知识体系。

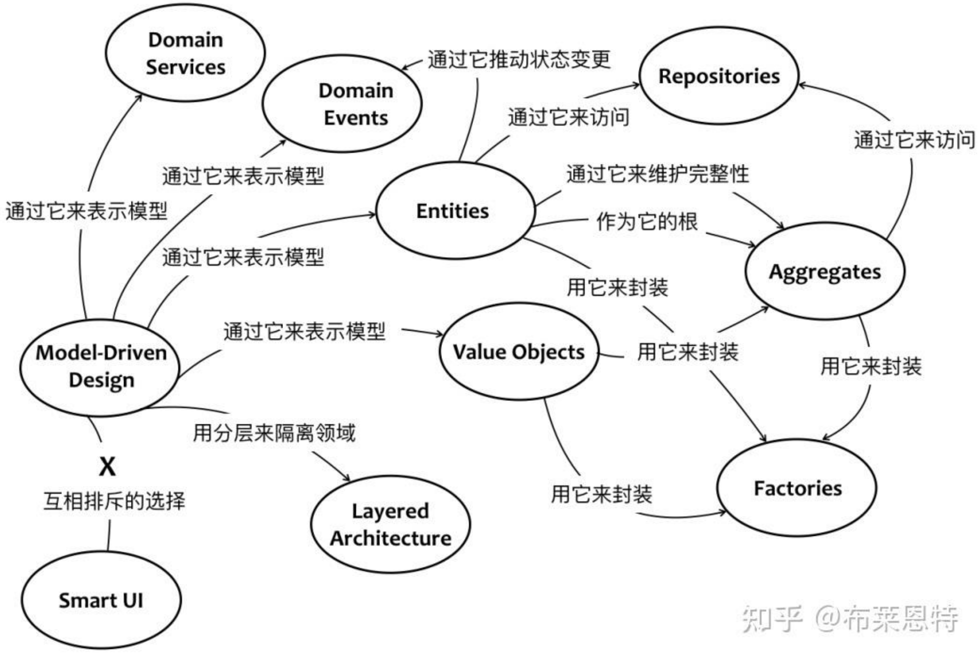

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler


语义相关度
功能相关度
语言概念是否冲突
业务边界
领域逻辑层面
领域专家与团队
问题域
+
业务期望
流程图
场景A
场景B
⋮
场景N
活动1
活动2
活动3
活动4
活动5
活动1
活动3
活动2
活动4
活动5
工作边界
团队合作层面
项目经理与团队
限界上下文A
限界上下文B
康威原则
特性团队
特性团队
2PTs
+
高内聚低耦合
应用边界
技术实现层面
架构师与团队
重用性
识别变化
质量属性
技术选型
限界上下文
限界上下文
⋮
限界上下文
…
微服务架构
共性与可变性分析
DRY原则
通信与集成模式
知乎 @布莱恩特

架构演化

- **第一步**：建立领域知识
- **第二步**：使用通用语言(Ubiquitous Language)描述
- **第三步**：构建模型关系图(Model-Driven Design)
- **第四步**：确定实体(Entity) 、值对象(Value Object)、服务(Services)、模块(Moudles)、聚合(Aggregates)、工厂(Factories)、仓库(Repository)

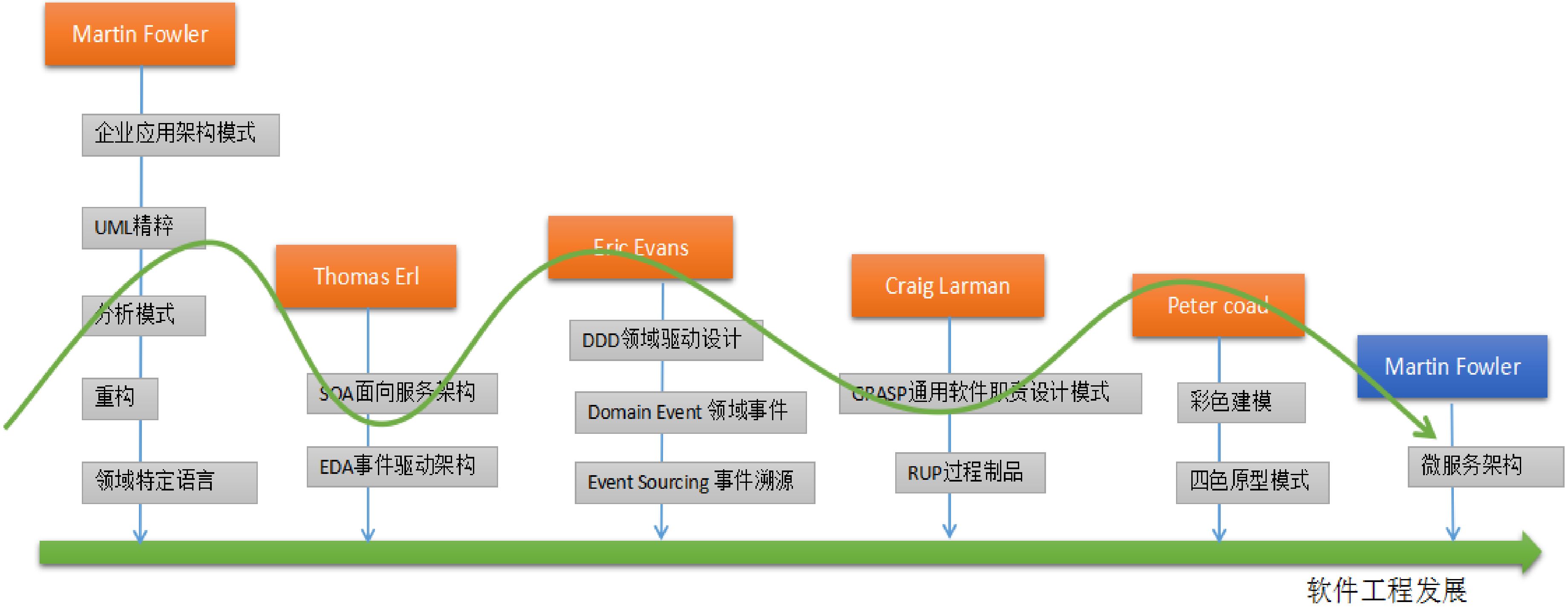

User Interface
Application
Domain
Infrastructure


Domain Services
Domain Events
通过它推动状态变更
Repositories
通过它来访问
通过它来访问
通过它来表示模型
通过它来表示模型
Entities
通过它来维护完整性
作为它的根
通过它来表示模型
用它来封装
Aggregates
通过它来表示模型
Model-Driven Design
Value Objects
用它来封装
用它来封装
用分层来隔离领域
X
互相排斥的选择
Layered Architecture
用它来封装
Factories
Smart UI

知乎 @布莱恩特

架构演化

□ 体系架构演化



Martin Fowler
- 企业应用架构模式
- UML精粹
- 分析模式
- 重构
- 领域特定语言

Thomas Erl
- SOA面向服务架构
- EDA事件驱动架构

Eric Evans
- DDD领域驱动设计
- Domain Event 领域事件
- Event Sourcing 事件溯源

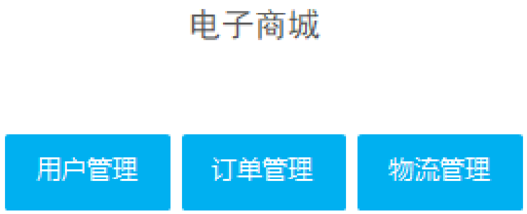
Craig Larman
- GRASP通用软件职责设计模式
- RUP过程制品

Peter coad
- 彩色建模
- 四色原型模式

Martin Fowler
- 微服务架构

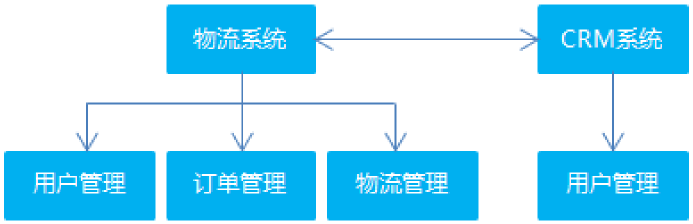
软件工程发展

架构演化


电子商城

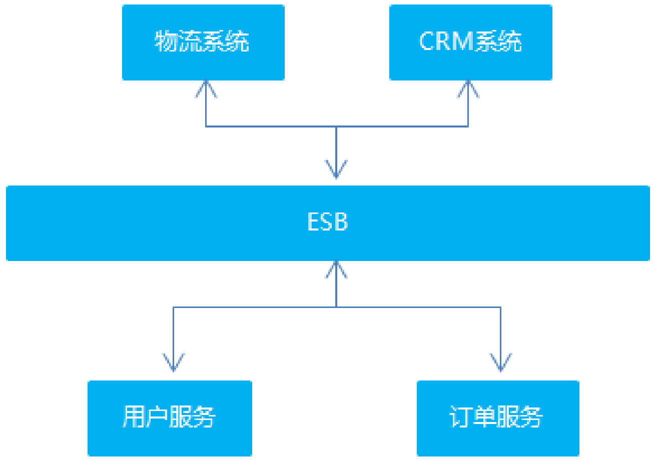
用户管理 订单管理 物流管理

1. 项目不利于扩展、难以维护
2. 技术栈受限：只能使用1种开发语言
3. 系统性能扩展只能通过扩展集群节点，成本高


物流系统 CRM系统

用户管理 订单管理 物流管理 用户管理

1. 项目不利于扩展、难以维护
2. 技术栈受限：只能使用1种开发语言
3. 系统性能扩展只能通过扩展集群节点，成本高
4. 项目之间功能冗余、数据冗余、耦合性强。

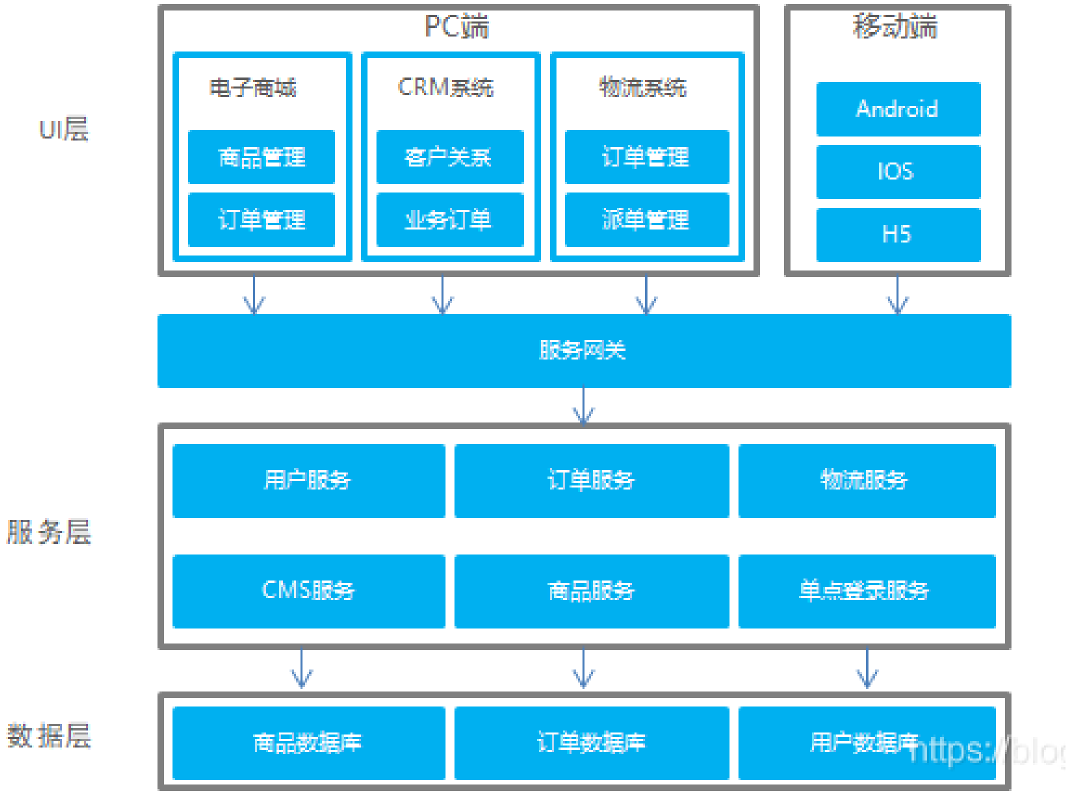

物流系统 CRM系统

ESB

用户服务 订单服务

1. 各系统之间业务不同，很难确认功能或模块是重复的；
2. 抽取服务的粒度大；
3. 系统和服务之间耦合度高。

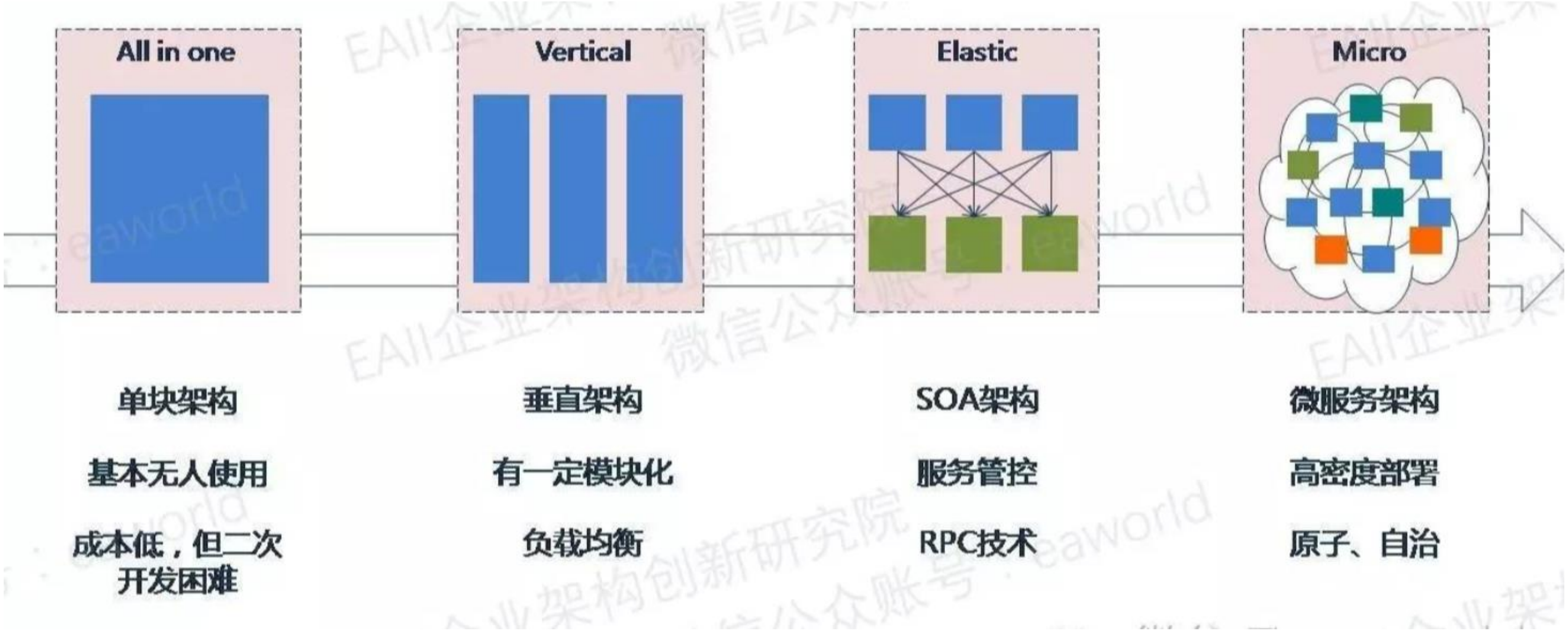
架构演化


PC端
移动端
UI层
电子商城
商品管理
订单管理
CRM系统
客户关系
业务订单
物流系统
订单管理
派单管理
Android
IOS
H5
服务网关
服务层
用户服务
订单服务
物流服务
CMS服务
商品服务
单点登录服务
数据层
商品数据库
订单数据库
用户数据库
https://blog

1. 服务拆分粒度更细，有利于提高开发效率；
2. 可以针对不同服务制定对应的优化方案；
3. 适用于互联网时代，产品迭代周期更短。

架构演化

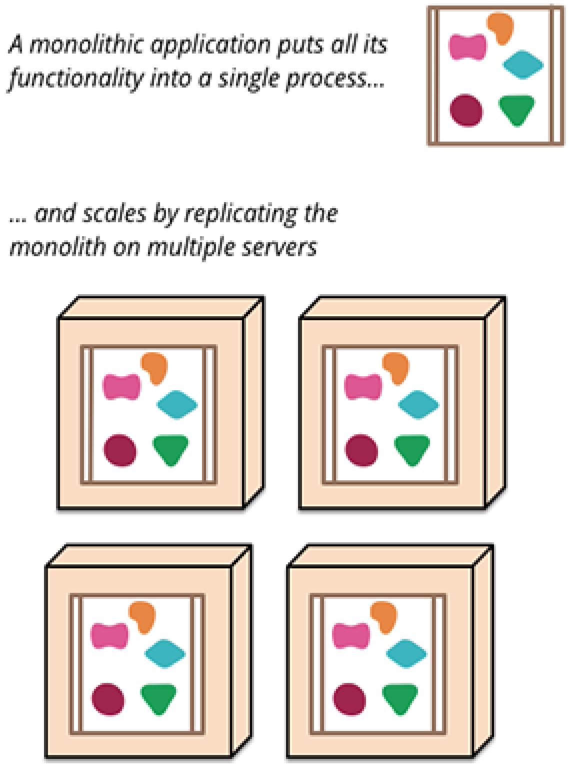
□ 架构演化总结


All in one
Vertical
Elastic
Micro

单块架构	垂直架构	SOA架构	微服务架构
基本无人使用	有一定模块化	服务管控	高密度部署
成本低，但二次开发困难	负载均衡	RPC技术	原子、自治

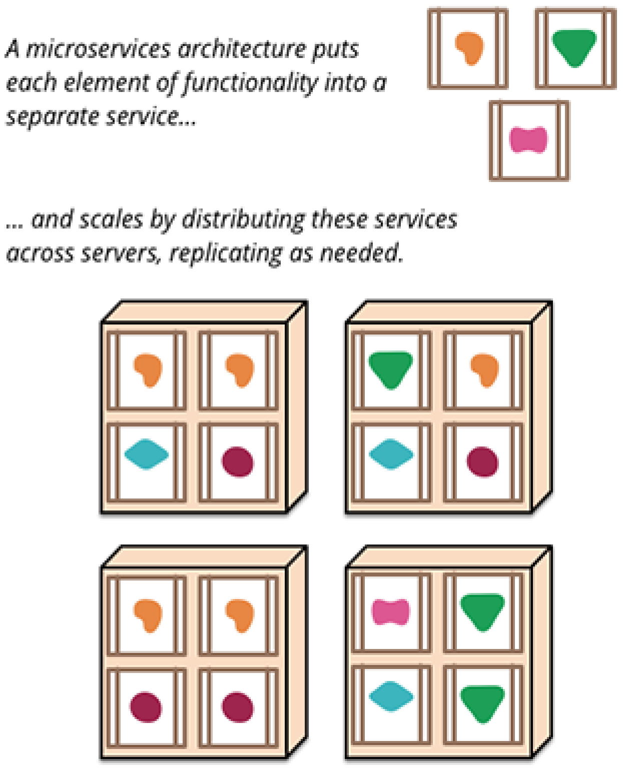
EAII企业架构创新研究院 微信公众账号 eaworld

架构演化



A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

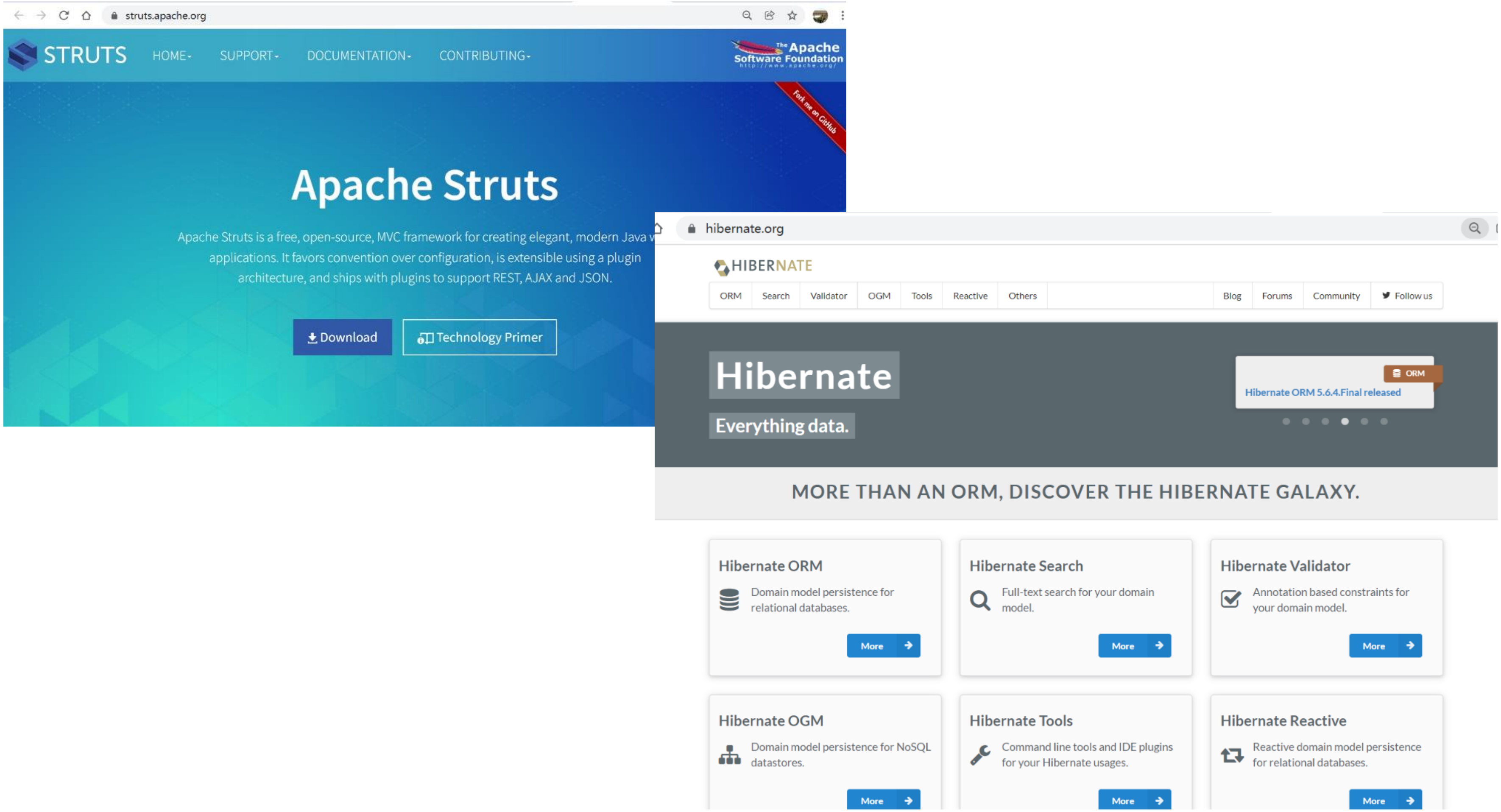
... and scales by distributing these services across servers, replicating as needed.

- 紧耦合，所有功能均在一个进程中，基于整个系统扩展
- 松耦合，功能分散于不同微服务的进程中，独立服务，按需扩展

- 将功能分散到各个离散的服务中然后实现对方案的解耦；
- 服务更原子，自治更小，然后高密度部署服务

架构演化

- SSH框架（Struts, Spring, Hibernate）
  - ➢ 帮助开发人员考虑了模块化、MVC等，但并没有考虑服务


struts.apache.org
STRUTS
HOME
SUPPORT
DOCUMENTATION
CONTRIBUTING
The Apache Software Foundation
Fork me on GitHub

Apache Struts

Apache Struts is a free, open-source, MVC framework for creating elegant, modern Java w applications. It favors convention over configuration, is extensible using a plugin architecture, and ships with plugins to support REST, AJAX and JSON.

Download
Technology Primer
hibernate.org
HIBERNATE
ORM
Search
Validator
OGM
Tools
Reactive
Others
Blog
Forums
Community
Follow us
ORM

Hibernate

Everything data.

Hibernate ORM 5.6.4.Final released

MORE THAN AN ORM, DISCOVER THE HIBERNATE GALAXY.

Hibernate ORM

Domain model persistence for relational databases.

Hibernate Search

Full-text search for your domain model.

Hibernate Validator

Annotation based constraints for your domain model.

Hibernate OGM

Domain model persistence for NoSQL datastores.

Hibernate Tools

Command line tools and IDE plugins for your Hibernate usages.

Hibernate Reactive

Reactive domain model persistence for relational databases.

More
More
More
More
More
More

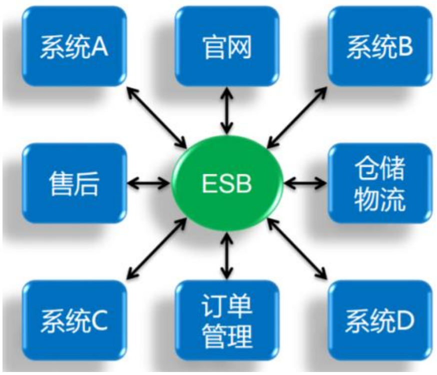
架构演化

□ 微服务与SOA的区别

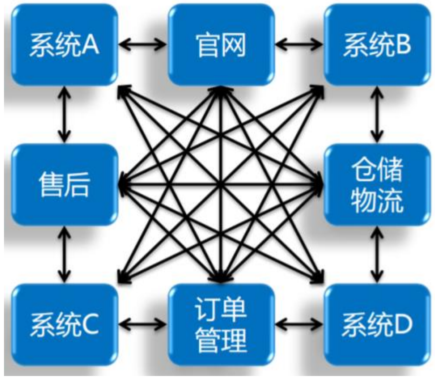
- 传统的面向服务架构（SOA）是集成多个较大组件（一般是应用）的一种机制，将整体构成一个彼此协作的套件。一般来说，每个组件会从始至终执行一块完整的业务逻辑，通常包括完成整体大action所需的各种具体任务与功能；
- 微服务，仍属于SOA 架构，但内涵不同。微服务并不绑定某种特殊的技术，在一个微服务的系统中，可以有 Java 编写的服务，也可以有 Python编写的服务，一般靠Restful架构风格统一系统。
  - ➢ 所以微服务本身与具体技术实现无关，扩展性强。

功能	SOA	微服务
组件大小	大块业务逻辑	单独任务或小块业务逻辑
耦合	通常松耦合	总是松耦合
公司架构	任何类型	小型、专注于功能交叉团队
管理	着重中央管理	着重分散管理
目标	确保应用能够交互操作	执行新功能、快速拓展开发团队

架构演化


系统A
官网
系统B
售后
ESB
仓储物流
系统C
订单管理
系统D

VS

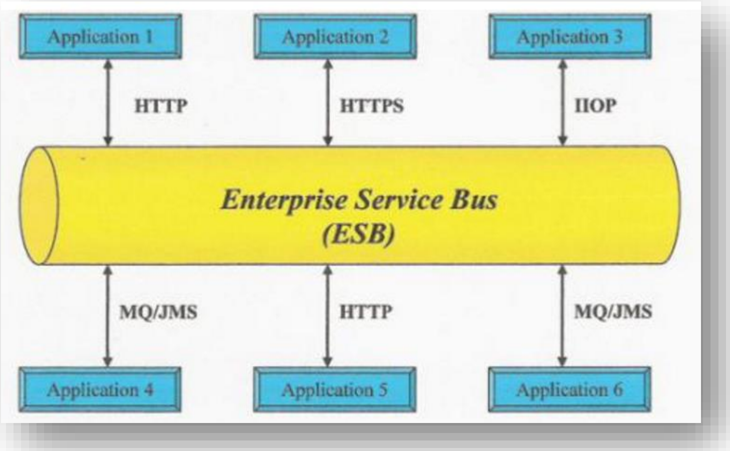

系统A
官网
系统B
售后
仓储物流
系统C
订单管理
系统D

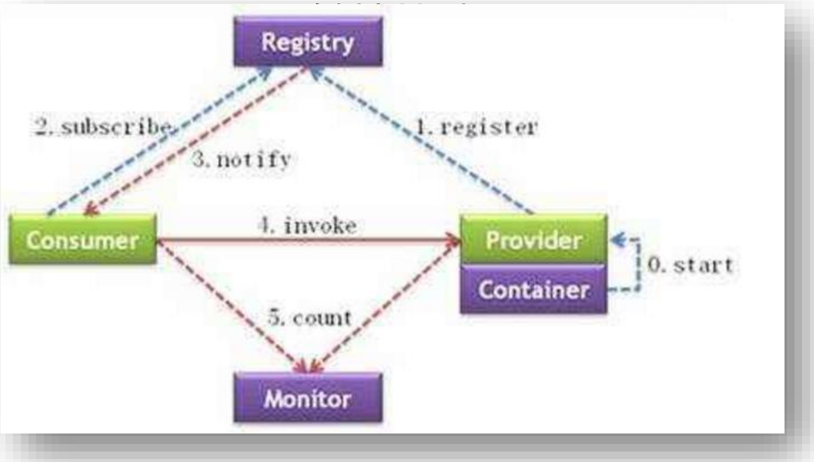
□ SOA和微服务的区别：

- ➢ 微服务不再强调传统SOA架构里面比较重的ESB企业服务总线；
- ➢ SOA的思想进入到单个业务系统内部实现真正的组件化。

□ SOA和微服务的共同点：

- ➢ 服务化


Application 1
Application 2
Application 3
HTTP
HTTPS
IIOP
Enterprise Service Bus
(ESB)
MQ/JMS
HTTP
MQ/JMS
Application 4
Application 5
Application 6


Registry
2. subscribe
3. notify
1. register
Consumer
4. invoke
Provider
Container
0. start
5. count
Monitor

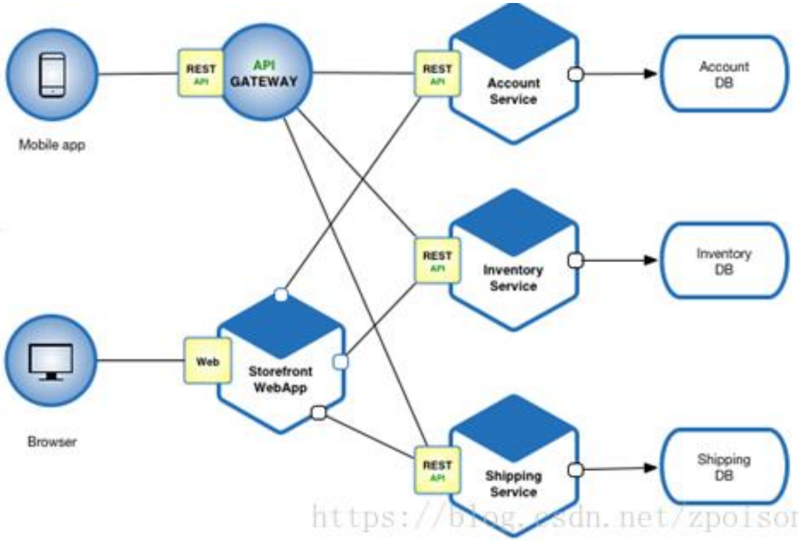
目录

- 微服务起源
- 微服务概念
- 架构演化
- 微服务架构
- 微服务实现
- 参考资料

微服务架构

□ API网关

- API网关是一个服务器，是系统的唯一入口。
- API网关封装了系统内部架构，为每个客户端提供一个定制的API。还可以具有其它职责，如身份验证、监控、负载均衡、缓存、请求分片与管理、静态响应处理。
- API网关方式的核心要点是，所有的客户端和消费端都通过统一的网关接入微服务，在网关层处理所有的非业务功能。
- 通常，网关也是提供REST/HTTP的访问API。服务端通过API网关注册和管理服务。


Mobile app
REST API
API GATEWAY
REST API
Account Service
Account DB
REST API
Inventory Service
Inventory DB
Browser
Web
Storefront WebApp
REST API
Shipping Service
Shipping DB

https://blog.csdn.net/zpoison

微服务架构

- 近年来，软件对**弹性部署**、**高扩展性**、**快速响应需求变更**等特性要求不断增加，使微服务架构得到广泛应用

类别	单体架构	分布式架构	微服务架构
部署方式	应用整体部署	系统部署，机器隔离	独立部署，进程隔离
数据存储	单独数据库	每个机器一个数据库	每个服务独立数据库
耦合程度	高耦合	耦合程度较高	耦合低，去中心化
服务调用	本地内部调用	不重视轻量级通信	重视基于http 轻量通信
可扩展性	扩展性差	扩展性较强	扩展性强
测试	简单，不确定性低	较简单，不确定性高	复杂，不确定性高

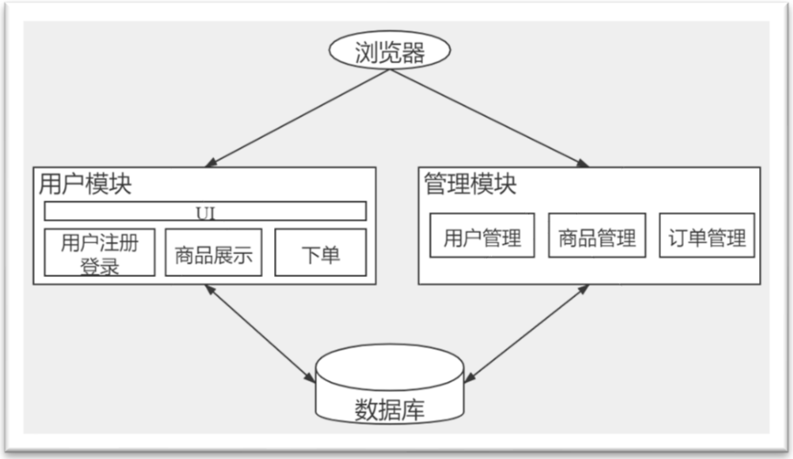
持续交付—容器部署

高扩展性—动态伸缩

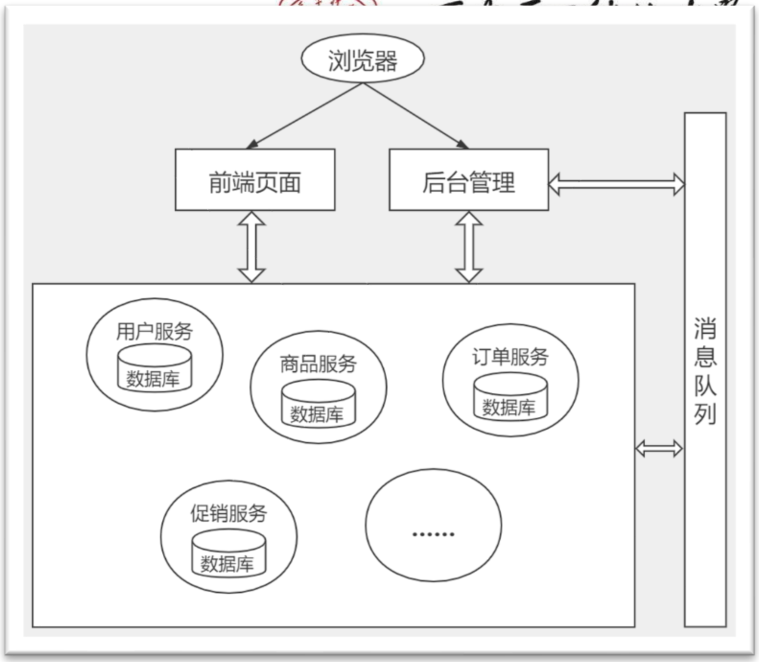
高自治性—职责单一

✓ 微服务架构 = 80%的SOA服务架构思想 + 100%的组件化架构思想 + 80%的领域建模思想

微服务架构


浏览器
用户模块
UI
用户注册登录
商品展示
下单
管理模块
用户管理
商品管理
订单管理
数据库

一个传统单体式架构实现的购物网站


浏览器
前端页面
后台管理
用户服务
数据库
商品服务
数据库
订单服务
数据库
促销服务
数据库
......
消息队列

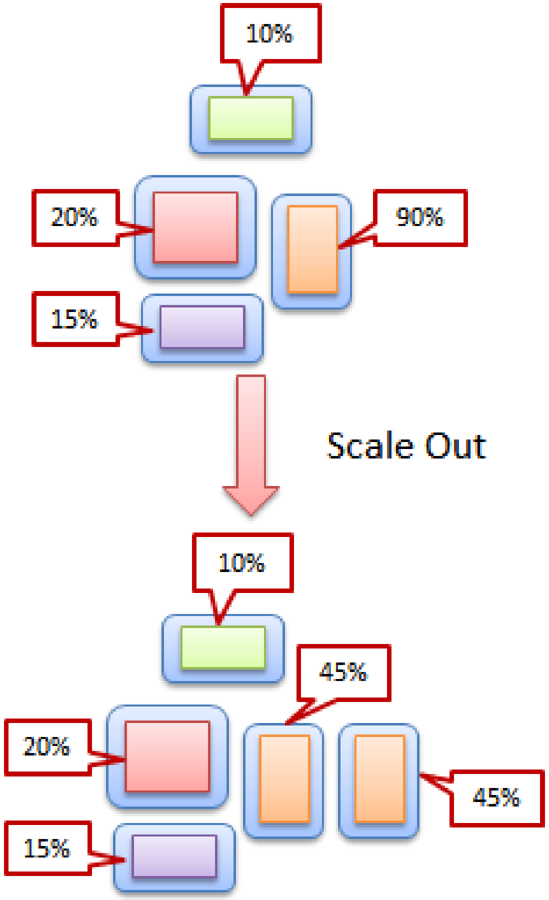
使用微服务架构构建的购物网站

缺点：

- ➢ 效率低：在同一个项目中修改代码，相互等待且冲突不断
- ➢ 维护难：代码功能耦合在一起难以维护
- ➢ 不灵活：构建时间长，任何小修改都要重构整个项目
- ➢ 稳定性差：小问题可能导致整个网站崩溃
- ➢ 扩展性差：无法满足高并发下的业务需求

- ➢ 可快速演化和迭代，可维护性强
- ➢ 容错性强

微服务架构


10%
20%
90%
15%
Scale Out
10%
45%
20%
45%
15%

- □ 在使用微服务架构模式的情况下，软件开发人员可以通过编译并重新部署单个子服务的方式来验证自己的更改，而不再需要重新编译整个应用，节省大量时间；
- □ 同时由于每个子服务是独立的，因此各个服务内部可以自行决定最为合适的实现技术，使得子服务的开发变得更为容易；
- □ 如果当前系统的容量不够了，那么只需要找到成为系统瓶颈的子服务，并扩展该子服务的容量即可。

微服务架构优缺点

□ 优点

- 每个服务足够内聚，足够小，代码容易理解、开发效率提高；
- 服务之间可以独立部署，微服务架构让持续部署成为可能；
- 每个服务可以各自进行x扩展和z扩展，而且，每个服务可以根据自己的需要部署到合适的硬件服务器上。
- 容易扩大开发团队，可以针对每个服务（service）组件开发团队；
- 提高容错性（fault isolation），一个服务的内存泄露并不会让整个系统瘫痪；
- 系统不会被长期限制在某个技术栈上。

微服务架构优缺点

□ 缺点

- 开发人员要处理分布式系统的复杂性
  - 开发人员要设计服务之间的通信机制，对于需要多个后端服务的user case，要在没有分布式事务的情况下实现代码非常困难；
  - 涉及多个服务直接的自动化测试也具备相当的挑战性。
- 服务管理的复杂性，在生产环境中要管理多个不同的服务的实例，这意味着开发团队需要全局统筹，
  - 现在Docker的出现适合解决这个问题
- 应用微服务架构的时机如何把握？
  - 对于业务还没有理清楚、业务数据和处理能力还没有开始爆发式增长之前的创业公司，不需要考虑微服务架构模式，这时候最重要的是快速开发、快速部署、快速试错。

目录

■ 微服务起源

■ 微服务概念

■ 架构演化

■ 微服务架构

■ 微服务实现

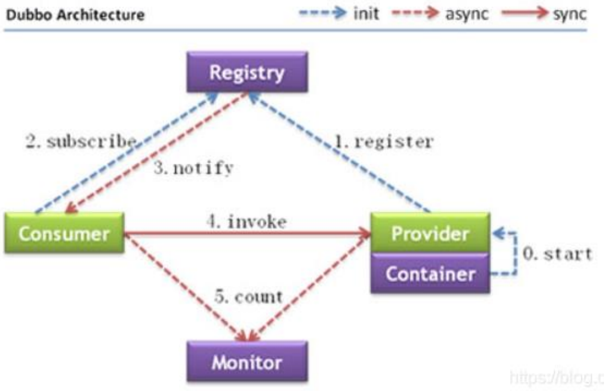
■ 参考资料

微服务实现

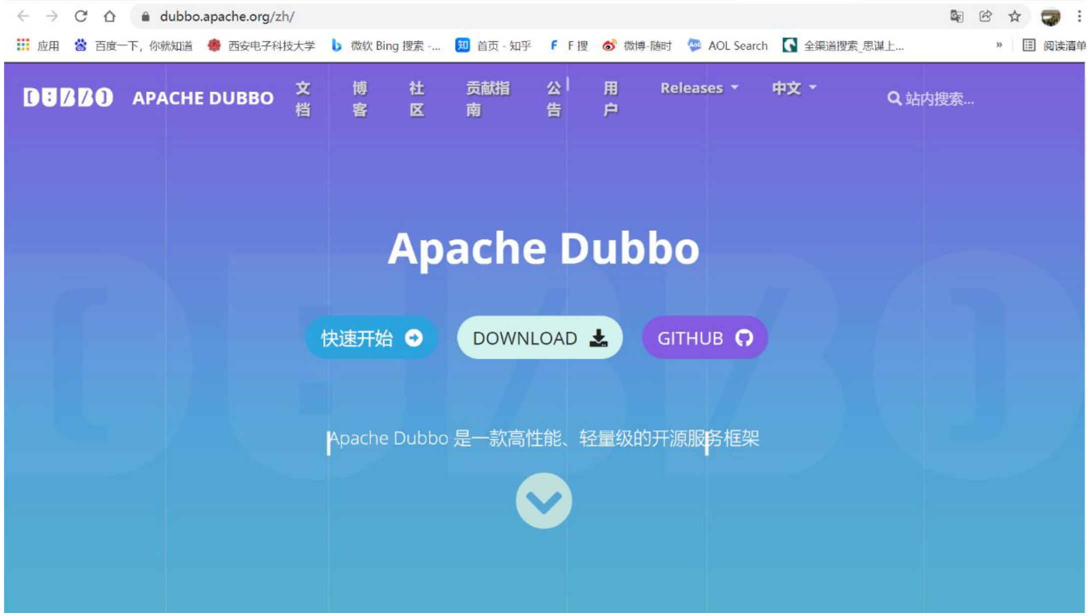
□ Apache Dubbo

- 阿里巴巴公司开源的一个高性能优秀的开源分布式服务框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，
- 主要核心部件：
  - ➢ Remoting:网络通信框架，实现了 sync-over-async 和Logo request-response 消息机制；
  - ➢ RPC：一个远程过程调用的抽象，支持负载均衡、容灾和集群功能；
  - ➢ Registry：服务目录框架用于服务的注册和服务事件发布和订阅。
- 远程方法调用、智能容错 和负载均衡，以及服务自动注册和发现

微服务实现


Dubbo Architecture
init
async
sync
Registry
2. subscribe
3. notify
1. register
Consumer
4. invoke
Provider
Container
0. start
5. count
Monitor

HTTP动作	动作语义
Provider	暴露服务的服务提供方（服务方）
Consumer	调用远程服务的服务消费方 （消费方）
Registry	服务注册与发现的注册中心 （注册中心）
Monitor	统计服务的调用次数和调用时间的监控中心 （监控中心）
Container	服务运行容器


dubbo.apache.org/zh/
应用
百度一下，你就知道
西安电子科技大学
微软 Bing 搜索 -...
首页 - 知乎
F 搜
微博-随时
AOL Search
全渠道搜索_思谋上...
阅读清单
DUBBO
APACHE DUBBO
文档
博客
社区
贡献指南
公告
用户
Releases
中文
站内搜索...
快速开始
DOWNLOAD
GITHUB

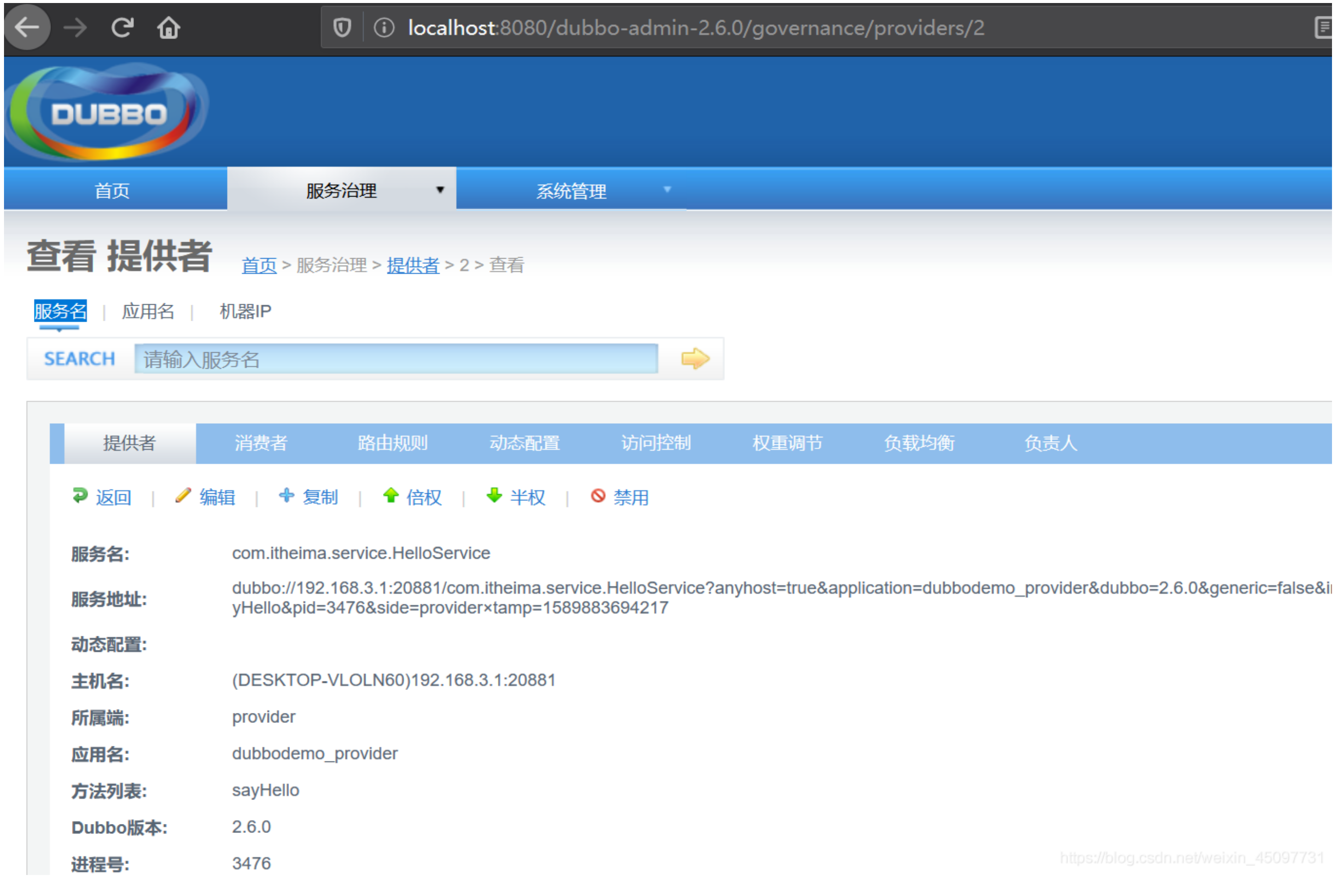
Apache Dubbo

Apache Dubbo 是一款高性能、轻量级的开源服务框架

微服务实现

西安电子科技大学
XIDIAN UNIVERSITY

□ Dubbo系统管理页面



localhost:8080/dubbo-admin-2.6.0/governance/providers/2
DUBBO
首页
服务治理
系统管理
首页 > 服务治理 > 提供者 > 2 > 查看
服务名 | 应用名 | 机器IP
SEARCH
请输入服务名
提供者
消费者
路由规则
动态配置
访问控制
权重调节
负载均衡
负责人
返回 | 编辑 | 复制 | 倍权 | 半权 | 禁用

查看 提供者

服务名:	com.itheima.service.HelloService
服务地址:	dubbo://192.168.3.1:20881/com.itheima.service.HelloService?anyhost=true&application=dubbodemo_provider&dubbo=2.6.0&generic=false&i yHello&pid=3476&side=provider×tamp=1589883694217
动态配置:	
主机名:	(DESKTOP-VLOLN60)192.168.3.1:20881
所属端:	provider
应用名:	dubbodemo_provider
方法列表:	sayHello
Dubbo版本:	2.6.0
进程号:	3476

https://blog.csdn.net/weixin_45097731

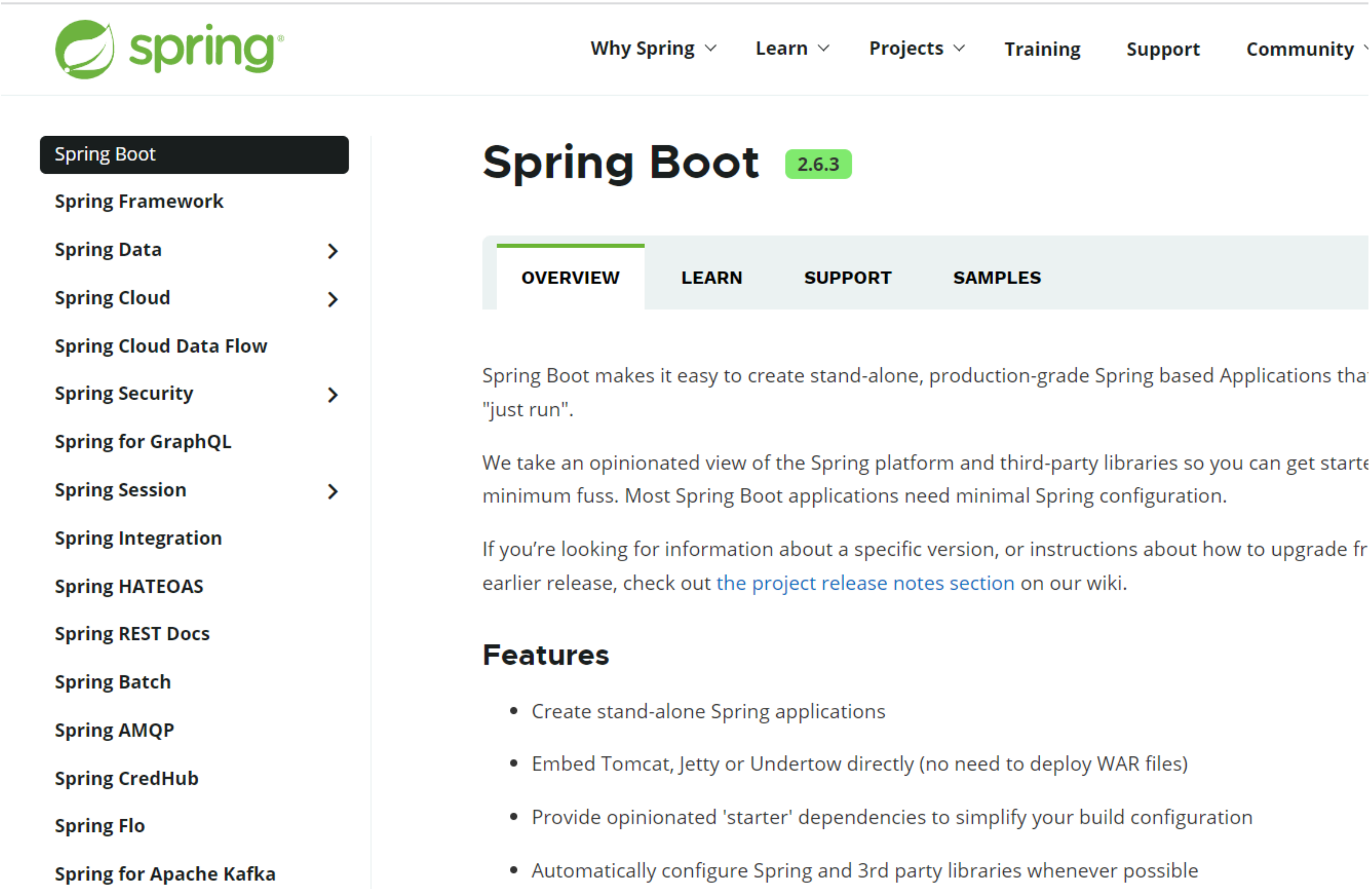
微服务实现

Spring Boot

- Spring Boot 是由 Pivotal 团队提供的全新框架，是所有基于 Spring Framework 5.0 开发的项目的起点。Spring Boot 的设计是为了尽可能快的跑起来 Spring 应用程序并且尽可能减少配置文件；
- Spring Boot 定位于快速构建 Spring 应用；
- Spring Boot 去除了大量的 XML 配置文件，简化了复杂的依赖管理；
- Spring Boot 集成了大量常用的第三方库配置，Spring Boot 应用中这些第三方库几乎可以是零配置的开箱即用（out-of-the-box），
    - 大部分的 Spring Boot 应用都只需要非常少量的配置代码（基于 Java 的配置），开发者能够更加专注于业务逻辑。

Spring Boot

微服务实现



spring
Why Spring
Learn
Projects
Training
Support
Community
Spring Boot
Spring Framework
Spring Data
Spring Cloud
Spring Cloud Data Flow
Spring Security
Spring for GraphQL
Spring Session
Spring Integration
Spring HATEOAS
Spring REST Docs
Spring Batch
Spring AMQP
Spring CredHub
Spring Flo
Spring for Apache Kafka
OVERVIEW
LEARN
SUPPORT
SAMPLES

Spring Boot 2.6.3

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications tha "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get starte minimum fuss. Most Spring Boot applications need minimal Spring configuration.

If you're looking for information about a specific version, or instructions about how to upgrade fr earlier release, check out the project release notes section on our wiki.

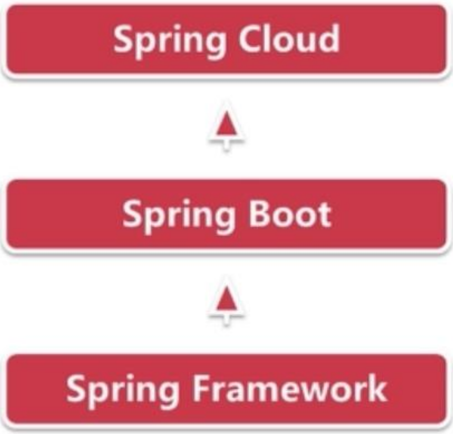
Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible

微服务实现

SpringCloud

- Spring Cloud 是一系列框架的有序集合，利用 Spring Boot 的开发便利性简化了分布式系统的开发，比如服务发现、服务网关、服务路由、链路追踪等；
- 集大成者，Spring Cloud 包含了微服务架构的各个方面；
- 约定优于配置，基于注解，没有配置文件；
- 轻量级组件，Spring Cloud 整合的组件大多比较轻量级，且都是各自领域的佼佼者；
- 开发简便，Spring Cloud 对各个组件进行了大量的封装，从而简化了开发；
- 开发灵活，Spring Cloud 的组件均解耦，开发人员可以灵活按需选择组件。

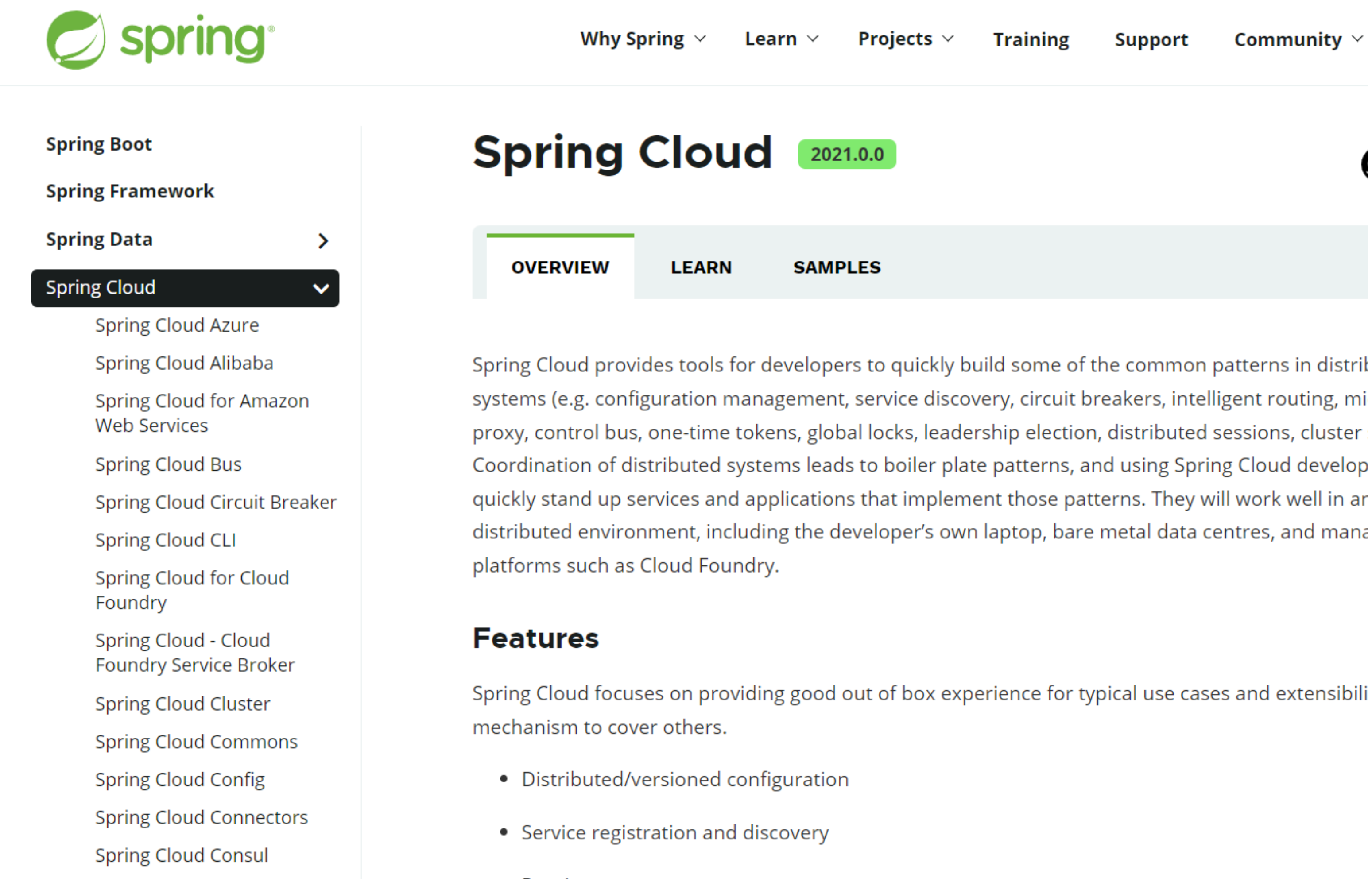


Spring Cloud

Spring Boot

Spring Framework

微服务实现


spring
Why Spring
Learn
Projects
Training
Support
Community
Spring Boot
Spring Framework
Spring Data
Spring Cloud
Spring Cloud Azure
Spring Cloud Alibaba
Spring Cloud for Amazon Web Services
Spring Cloud Bus
Spring Cloud Circuit Breaker
Spring Cloud CLI
Spring Cloud for Cloud Foundry
Spring Cloud - Cloud Foundry Service Broker
Spring Cloud Cluster
Spring Cloud Commons
Spring Cloud Config
Spring Cloud Connectors
Spring Cloud Consul
2021.0.0
OVERVIEW
LEARN
SAMPLES

Spring Cloud

Spring Cloud provides tools for developers to quickly build some of the common patterns in distrit systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, mi proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster Coordination of distributed systems leads to boiler plate patterns, and using Spring Cloud develop quickly stand up services and applications that implement those patterns. They will work well in ar distributed environment, including the developer's own laptop, bare metal data centres, and mana platforms such as Cloud Foundry.

Features

Spring Cloud focuses on providing good out of box experience for typical use cases and extensibili mechanism to cover others.

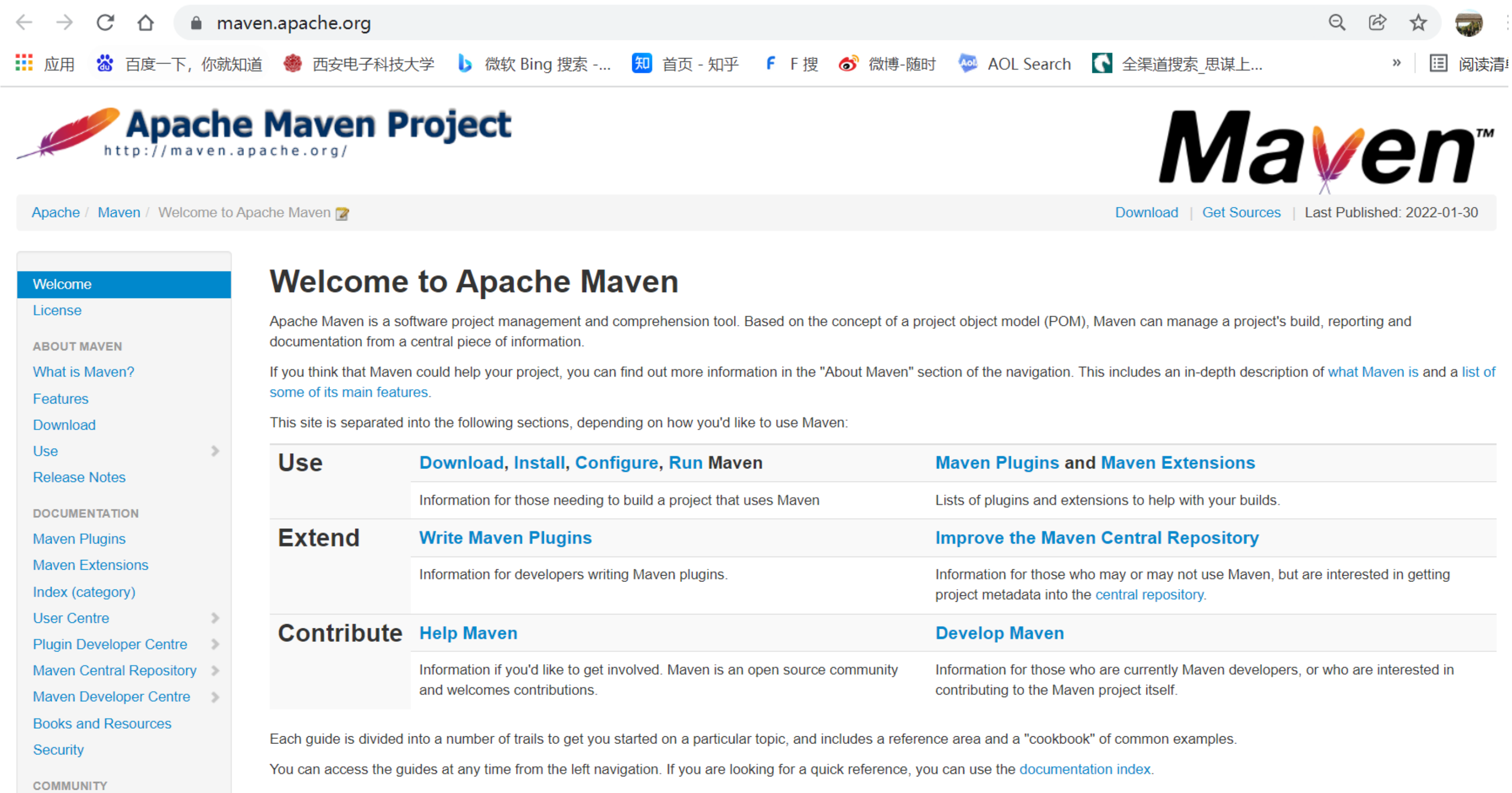
- Distributed/versioned configuration
- Service registration and discovery

微服务实现

□ Maven

- Maven作为一个构建工具，不仅能自动化构建，还能够抽象构建过程，提供构建任务实现；跨平台，对外提供了一致的操作接口，是一种优秀的、流行的构建工具；
- Maven不仅是构建工具，还是一个依赖管理工具和项目管理工具，提供了中央仓库，能够自动下载构件；
- Maven项目的核心是pom.xml文件；
- POM（Project Object Model，项目对象模型）定义了项目的基本信息，用于描述项目如何构建，声明项目依赖等。

微服务实现


maven.apache.org
应用
百度一下，你就知道
西安电子科技大学
微软 Bing 搜索 -...
首页 - 知乎
F 搜
微博-随时
AOL Search
全渠道搜索_思谋上...
阅读清
Apache Maven Project
http://maven.apache.org/
Maven™
Apache / Maven / Welcome to Apache Maven
Download | Get Sources | Last Published: 2022-01-30
Welcome
License
ABOUT MAVEN
What is Maven?
Features
Download
Use
Release Notes
DOCUMENTATION
Maven Plugins
Maven Extensions
Index (category)
User Centre
Plugin Developer Centre
Maven Central Repository
Maven Developer Centre
Books and Resources
Security
COMMUNITY

Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is and a list of some of its main features.

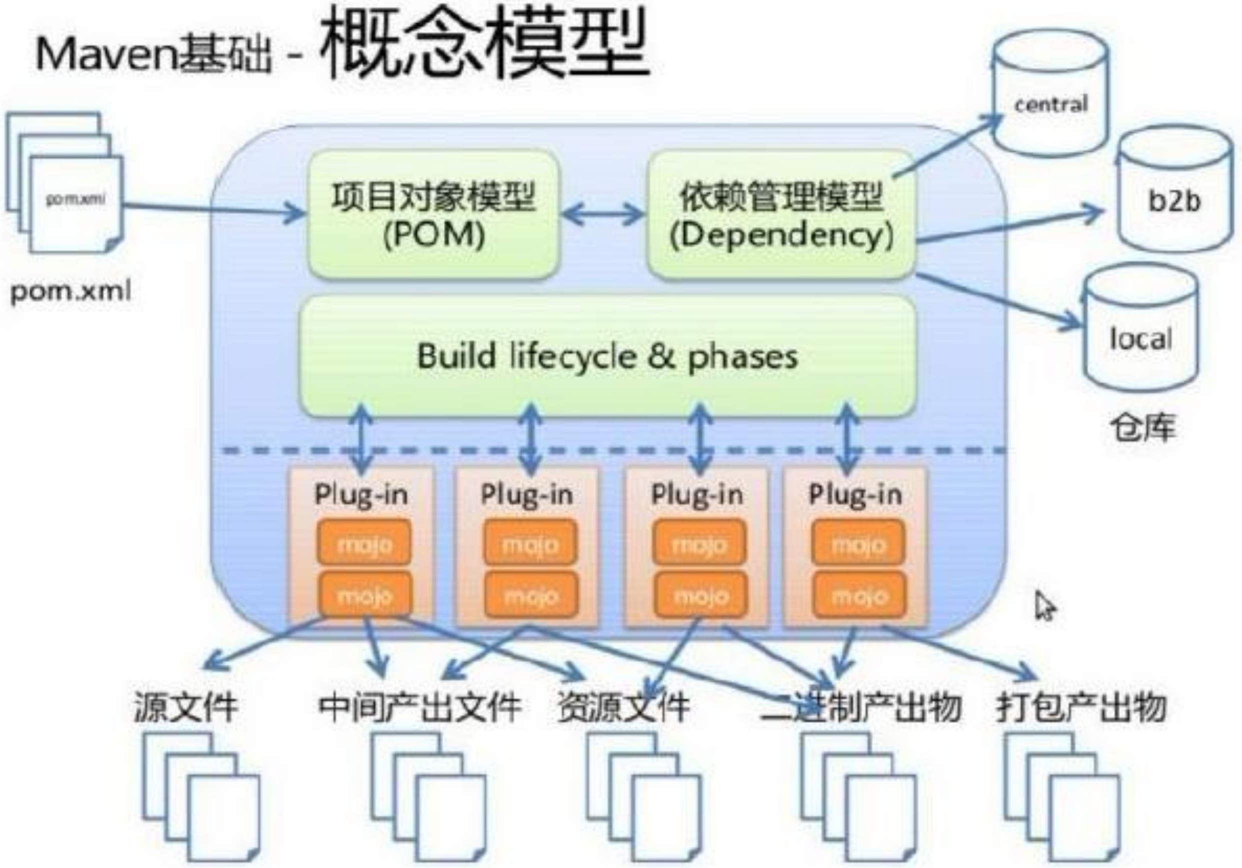
This site is separated into the following sections, depending on how you'd like to use Maven:

Use	**Download, Install, Configure, Run Maven**	**Maven Plugins and Maven Extensions**
	Information for those needing to build a project that uses Maven	Lists of plugins and extensions to help with your builds.
Extend	**Write Maven Plugins**	**Improve the Maven Central Repository**
	Information for developers writing Maven plugins.	Information for those who may or may not use Maven, but are interested in getting project metadata into the central repository.
Contribute	**Help Maven**	**Develop Maven**
	Information if you'd like to get involved. Maven is an open source community and welcomes contributions.	Information for those who are currently Maven developers, or who are interested in contributing to the Maven project itself.

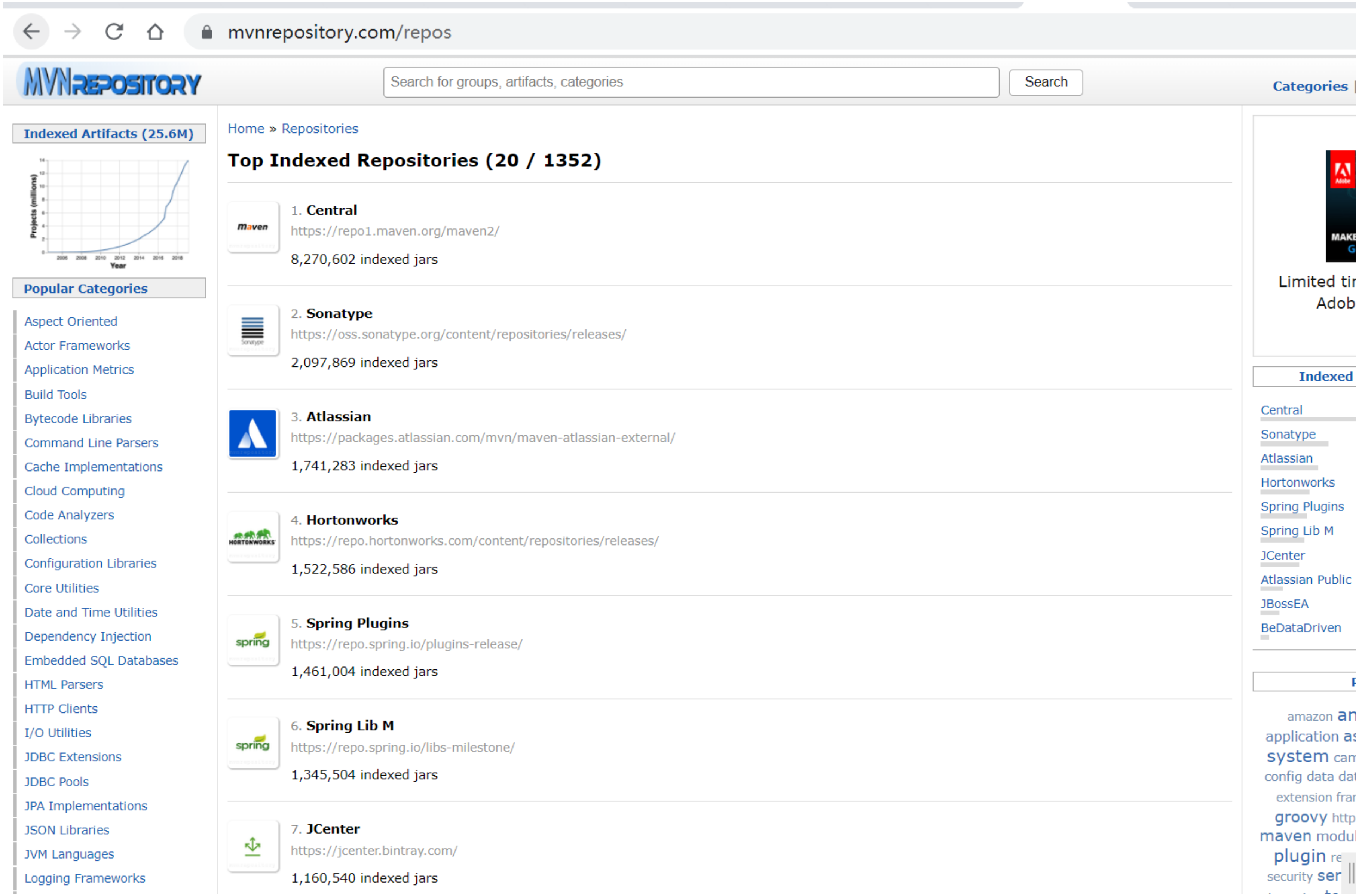
Each guide is divided into a number of trails to get you started on a particular topic, and includes a reference area and a "cookbook" of common examples.

You can access the guides at any time from the left navigation. If you are looking for a quick reference, you can use the documentation index.

微服务实现


Maven基础 - 概念模型
pom.xml
pom.xml
项目对象模型 (POM)
依赖管理模型 (Dependency)
Build lifecycle & phases
Plug-in
mojo
mojo
Plug-in
mojo
mojo
Plug-in
mojo
mojo
Plug-in
mojo
mojo
central
b2b
local
仓库
源文件
中间产出文件
资源文件
二进制产出物
打包产出物

微服务实现



mvnrepository.com/repos

MVNREPOSITORY
Search for groups, artifacts, categories
Search
Categories

Indexed Artifacts (25.6M)

Projects (millions)
Year

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages
- Logging Frameworks

Home » Repositories

Top Indexed Repositories (20 / 1352)

1. **Central**
https://repo1.maven.org/maven2/
8,270,602 indexed jars

2. **Sonatype**
https://oss.sonatype.org/content/repositories/releases/
2,097,869 indexed jars

3. **Atlassian**
https://packages.atlassian.com/mvn/maven-atlassian-external/
1,741,283 indexed jars

4. **Hortonworks**
https://repo.hortonworks.com/content/repositories/releases/
1,522,586 indexed jars

5. **Spring Plugins**
https://repo.spring.io/plugins-release/
1,461,004 indexed jars

6. **Spring Lib M**
https://repo.spring.io/libs-milestone/
1,345,504 indexed jars

7. **JCenter**
https://jcenter.bintray.com/
1,160,540 indexed jars

Indexed

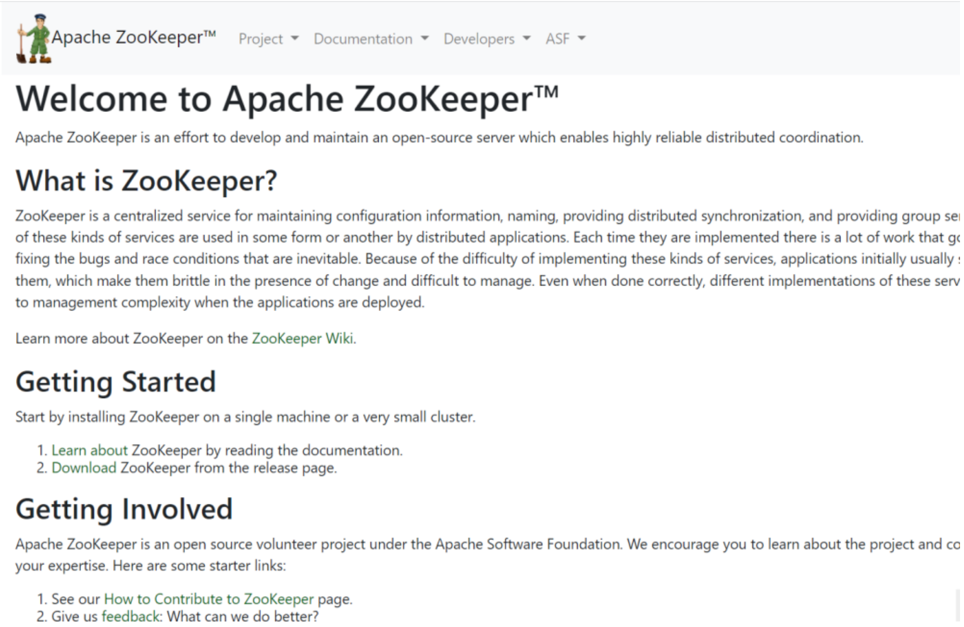
- Central
- Sonatype
- Atlassian
- Hortonworks
- Spring Plugins
- Spring Lib M
- JCenter
- Atlassian Public
- JBossEA
- BeDataDriven

amazon
application
system
config
data
groovy
maven
plugin
security

微服务实现

□ ZooKeeper

- Zookeeper 是 Apache Hadoop 的子项目，是一个树型的**目录服务**，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用 。



Welcome to Apache ZooKeeper™

Apache ZooKeeper is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination.

What is ZooKeeper?

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group se
of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that g
fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually
them, which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these serv
to management complexity when the applications are deployed.

Learn more about ZooKeeper on the ZooKeeper Wiki.

Getting Started

Start by installing ZooKeeper on a single machine or a very small cluster.

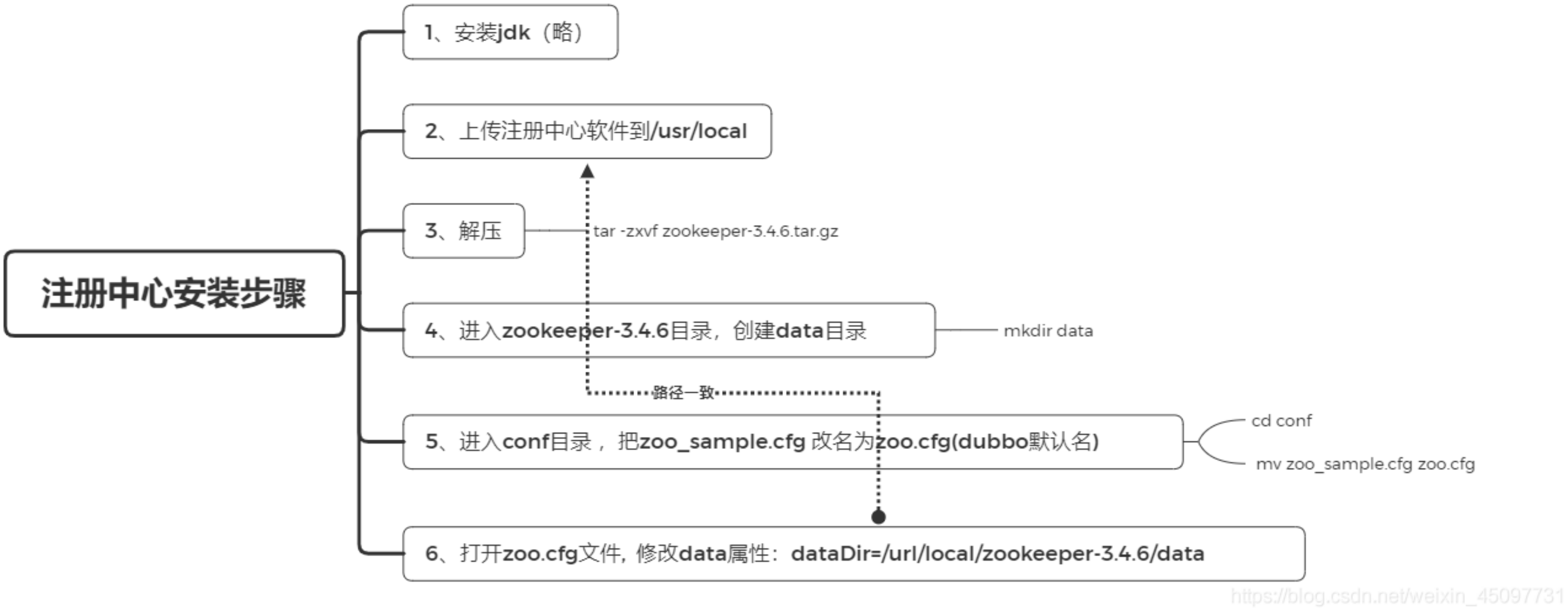
1. Learn about ZooKeeper by reading the documentation.
2. Download ZooKeeper from the release page.

Getting Involved

Apache ZooKeeper is an open source volunteer project under the Apache Software Foundation. We encourage you to learn about the project and co
your expertise. Here are some starter links:

1. See our How to Contribute to ZooKeeper page.
2. Give us feedback: What can we do better?

微服务实现



注册中心安装步骤

1、安装jdk（略）

2、上传注册中心软件到/usr/local

3、解压 — tar -zxvf zookeeper-3.4.6.tar.gz

4、进入zookeeper-3.4.6目录，创建data目录 — mkdir data

5、进入conf目录， 把zoo_sample.cfg 改名为zoo.cfg(dubbo默认名)
- cd conf
- mv zoo_sample.cfg zoo.cfg

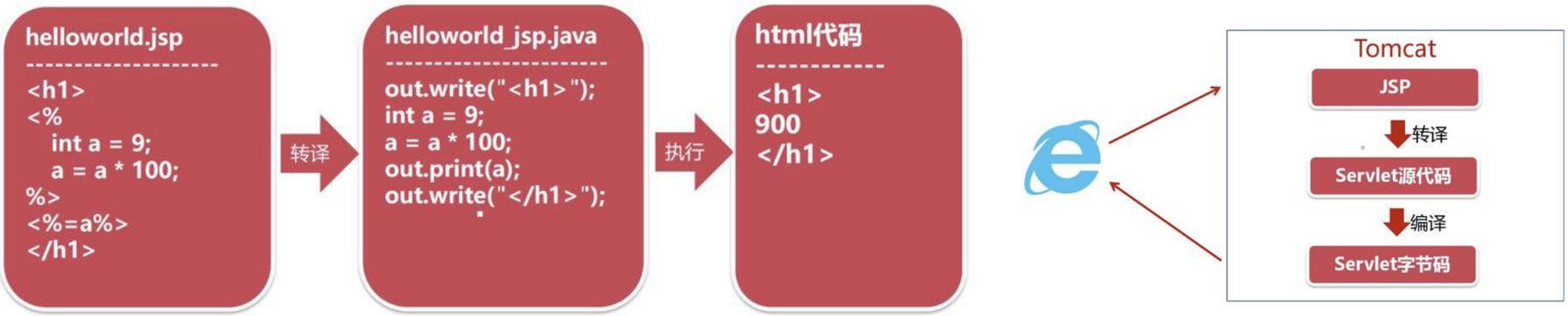
6、打开zoo.cfg文件，修改data属性：dataDir=/url/local/zookeeper-3.4.6/data

https://blog.csdn.net/weixin_45097731

微服务实现

□ Tomcat

- 由 Apache 开发的Web服务器，包含Servlet 容器，实现了对 Servlet 和 JSP 的支持，并提供了作为Web服务器的特有功能，如Tomcat管理和控制平台、安全域管理等。
- 由于 Tomcat 本身也内含了一个 HTTP 服务器，也可以被视作一个单独的 Web 服务器。



helloworld.jsp

```
--------------------
<h1>
<%
   int a = 9;
   a = a * 100;
%>
<%=a%>
</h1>
```

转译

helloworld_jsp.java

```
----------------------
out.write("<h1>");
int a = 9;
a = a * 100;
out.print(a);
out.write("</h1>");
```

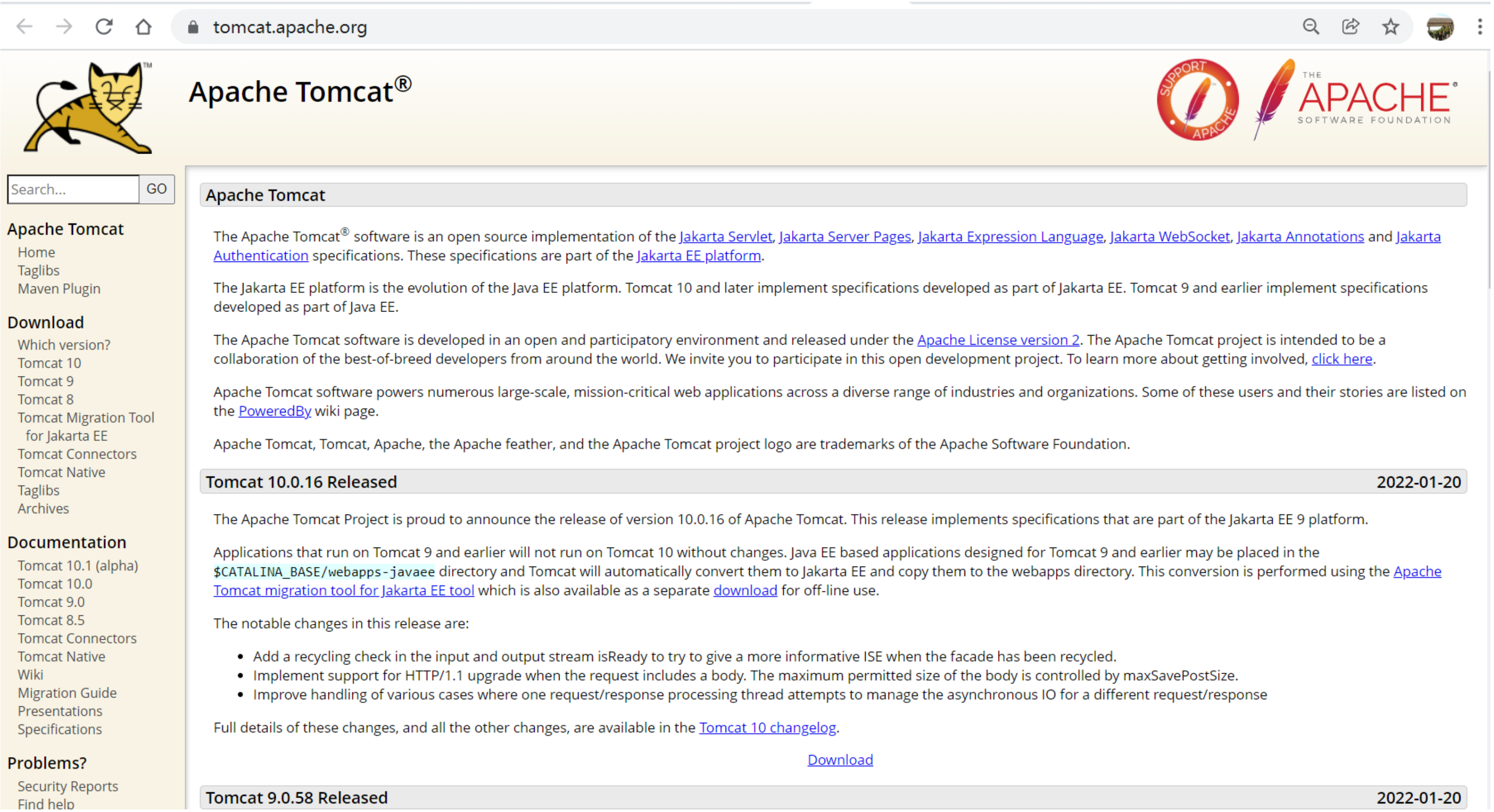
执行

html代码

```
------------
<h1>
900
</h1>
```

Tomcat
JSP
转译
Servlet源代码
编译
Servlet字节码

微服务实现


tomcat.apache.org
Apache Tomcat®
SUPPORT APACHE
THE APACHE SOFTWARE FOUNDATION
Search...
GO
Apache Tomcat
Home
Taglibs
Maven Plugin
Download
Which version?
Tomcat 10
Tomcat 9
Tomcat 8
Tomcat Migration Tool for Jakarta EE
Tomcat Connectors
Tomcat Native
Taglibs
Archives
Documentation
Tomcat 10.1 (alpha)
Tomcat 10.0
Tomcat 9.0
Tomcat 8.5
Tomcat Connectors
Tomcat Native
Wiki
Migration Guide
Presentations
Specifications
Problems?
Security Reports
Find help

Apache Tomcat

The Apache Tomcat® software is an open source implementation of the Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta WebSocket, Jakarta Annotations and Jakarta Authentication specifications. These specifications are part of the Jakarta EE platform.

The Jakarta EE platform is the evolution of the Java EE platform. Tomcat 10 and later implement specifications developed as part of Jakarta EE. Tomcat 9 and earlier implement specifications developed as part of Java EE.

The Apache Tomcat software is developed in an open and participatory environment and released under the Apache License version 2. The Apache Tomcat project is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, click here.

Apache Tomcat software powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the PoweredBy wiki page.

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are trademarks of the Apache Software Foundation.

Tomcat 10.0.16 Released 2022-01-20

The Apache Tomcat Project is proud to announce the release of version 10.0.16 of Apache Tomcat. This release implements specifications that are part of the Jakarta EE 9 platform.

Applications that run on Tomcat 9 and earlier will not run on Tomcat 10 without changes. Java EE based applications designed for Tomcat 9 and earlier may be placed in the `$CATALINA_BASE/webapps-javaee` directory and Tomcat will automatically convert them to Jakarta EE and copy them to the webapps directory. This conversion is performed using the Apache Tomcat migration tool for Jakarta EE tool which is also available as a separate download for off-line use.

The notable changes in this release are:

- Add a recycling check in the input and output stream isReady to try to give a more informative ISE when the facade has been recycled.
- Implement support for HTTP/1.1 upgrade when the request includes a body. The maximum permitted size of the body is controlled by maxSavePostSize.
- Improve handling of various cases where one request/response processing thread attempts to manage the asynchronous IO for a different request/response

Full details of these changes, and all the other changes, are available in the Tomcat 10 changelog.

Download

Tomcat 9.0.58 Released 2022-01-20

参考资料

- 王璐.微服务架构软件的运维治理, 2022
- James Lewis & Martin Fowler. Microservices in a Nutshell. https://www.thoughtworks.com/insights/blog/microservices-nutshell, June 27, 2014
- Wikipedia. Melvin Conway. https://en.wikipedia.org/wiki/Melvin_Conway, 2021
- 孤芳不自賞. https://blog.csdn.net/en_joker/article/details/88306553, 2019
- Pursue. 浅析DDD(领域驱动设计). https://www.jianshu.com/p/b6ec06d6b594, 2017
- 布莱恩特. DDD 领域驱动设计. https://zhuanlan.zhihu.com/p/347603268， 2021
- zpoison. SOA架构和微服务架构的区别. https://blog.csdn.net/zpoison/article/details/80729052?spm=1001.2014.3001.5501, 2018

Q&A