Atitit 2016 技术趋势与没落技术

目录

[1.1. 流水线 即代码通过编码而非配置CI/CD运行工具的方式，来定义部署 流水线 1](#_Toc19324)

[1.2. 将APIs当作产品 1](#_Toc22738)

[1.3. 无服务器架构是一种架构方法 2](#_Toc29984)

[1.4. 随着BFF - 后端服务前端模式和单向数据绑定框架， 2](#_Toc29685)

[1.5. 谷歌的TensorFlow是一个可用于各种从研究到生产的开源机 器学习平台， 2](#_Toc23083)

[1.6. Webpack已经证明了自己是值得选择的JavaScript模块打包工 具 3](#_Toc9630)

[1.7. Springboot 3](#_Toc5751)

[1.8. 比REST更合适的方法， 那就是Facebook的GraphQL 3](#_Toc8708)

[2. 没落技术 3](#_Toc7680)

[2.1. 贫血 REST这种反模式，使用GraphQL这类dsl 4](#_Toc25427)

[2.2. CMS作为平台来使用 4](#_Toc26618)

[2.3. angulr 4](#_Toc6869)

## 流水线 即代码通过编码而非配置CI/CD运行工具的方式，来定义部署 流水线

团队在推进跨环境的自动化，包括开发的基础设施。 流水线 即代码通过编码而非配置CI/CD运行工具的方式，来定义部署 流水线。LambdaCD ，Drone，GoCD和Concourse等都是 这一技术的应用案例。此外，像 GoMatic这样的CI/CD系统自 动化配置工具，可以对部署流水线即代码进行版本化和测试。

## 将APIs当作产品

企业已经全力拥抱通过APIs将业务能力暴露给内外部开发者。 基于APIs可将现有核心能力进行整合，快速验证新的业务创 意。但APIs跟传统的企业集成服务有什么区别呢？一个区别在 于将APIs当作产品，即使消费者是一个内部系统。APIs团队 应该理解客户需求，为他们提供有吸引力的产品。从长期看， 产品应该得到改进、维护和支持；指定专门的负责人为客户发 声，并努力持续改进。产品还应该被积极地维护和支持，易于 找到和易于使用。根据我们的经验，企业集成服务缺乏产品导 向，是与基于APIs的敏捷业务最大的区

## 无服务器架构是一种架构方法

，使用即时请求、用后即销毁的 短暂计算能力来取代长期运行的虚拟机。从上一期技术雷达开 始，我们已经有若干个团队在正式产品中应用了“无服务器架 构”风格。我们的团队喜欢并适应这种方式，我们认为这是一 种有效的架构选择。需要指出的是，无服务器架构并非一种绝 对的架构风格：我们某些团队将系统中的一部分采用无服务器 架构，而其它部分继续采用传统架构。

我们从微服务架构的引入中获益匪浅，它允许团队规模化地交 付那些能够独立部署和维护的服务。然而，在前端，团队经常 做了很多努力来避免创建一个大型的单体和庞大的浏览器应用 程序，就如我们已经放弃的服务器端单体应用一样，这些应用 程序难以维护和演化。我们看到一种方法正在浮现，我们的团 队将其称之为微前端。在这种方法中，一个Web应用程序可 以分解为页面和特性，每个特性由一个单独的团队从端到端对 其负责。现存的很多种技术可以将这些应用特性组织在一起， 从而提供一个统一内聚的用户体验。但是微前端的目标仍然是 允许特性之间彼此独立，每个特性可以独立地开发、测试和部 署。BFF - 后端服务前端的方法也可以很好地应用于这个场景 下，每个团队可以开发一个BFF以支持其一系列的应用程序特 性。

## 随着BFF - 后端服务前端模式和单向数据绑定框架，

如React. js的日益普及，我们注意到很多人对REST风格架构表现出反 感。批评者们指责REST导致了系统之间繁琐低效的交互， 且无法适应客户端需求的变化。他们提出了一些框架，如 GraphQL或Falcor作为数据获取机制的替代方法，它们可以让 客户端指定返回的数据格式。但基于我们的经验，我们认为这 些问题并不是REST引起的。相反，它们源于未能将领域作为 一组资源来正确地建模。通过模板化的URL、简单地暴露静态 分层数据模型来开发

## 谷歌的TensorFlow是一个可用于各种从研究到生产的开源机 器学习平台，

它可在小到一部智能手机、大到大规模图形处理 单元（GPU）集群上运行。其重要性表现在，它让实施深度学 习算法变得更容易和便捷。抛开那些炒作，其实TensorFlow并 不是什么新算法：所有这些技术都已经通过学术界在公开领域 存在一段时间了。需要注意的是，对于大多数连最基本的预测 分析都还没开始的企业而言，直接进入深度学习其大多数的数 据集并没有太大帮助。但是，对于那些有明确问题和数据集的 企业，TensorFlow将会是一个有用的工具。

## Webpack已经证明了自己是值得选择的JavaScript模块打包工 具

。伴随着这份加载器列表的不断增长，Webpack可以为你所 有的静态资源提供单一的依赖树，对JavaScript、CSS等资源进 行灵活的操作，并将向浏览器发送内容的数量和次数最小化。 特别重要的是，它可以平滑的在AMD、CommonJS以及ES6模 块间进行集成，从而让团队能够使用ES6，并在有浏览器兼容 性需要的时候自动无缝转译（通过Babel）为更早版本。我们 的很多团队也觉得Browserify值得一用，它做了类似的工作， 但是更专注于如何让Node.js模块在客户端变得可用

## Springboot

## 比REST更合适的方法， 那就是Facebook的GraphQL

，有一种比REST更合适的方法， 那就是Facebook的GraphQL，它是一个很有趣的替代方 案。GraphQL做为一种远程接收对象图网络的协议，在近期 获得了非常高的关注。GraphQL最有趣的特性之一，是它本 质上面向消费者，响应体的结构不取决于服务端，而是完全 语言和框架 接上页 © April 2016, ThoughtWorks, Inc. All Rights Reserved. TECHNOLOGY RADAR APRIL 2016 | 15 由客户端驱动。这样它就将消费者解耦，并且强迫服务端遵守 Postel法则。目前客户端支持多种编程语言，我们看到人们对 Facebook的Relay表现出强烈的兴趣。它是一个支持到React.js 无状态组件模型的Javascript框架。

# 没落技术

## 贫血 REST这种反模式，使用GraphQL这类dsl

设计RESTful风格APIs遇到的诸多问题，都可以归咎于贫血 REST这种反模式，一些场景也证明了需有更多解决方法。特别 是，当组织必须支持一些长尾客户应用时（大量增长的API版 本，即使已经使用了消费者驱动的契约测试——大部分的APIs 都要支持无尽的活跃订阅端数目 —— 也许触及到了RESTful 架构的极限。这些问题可以使用客户直接查询来解决。我们已 经在GraphQL和Falcor中看到了成功案例：这项技术可以让客 户在内容和反馈数据的粒度两个方面掌握更多的控制权。这将 更多的职责推到了服务层，并且会导致数据模型的强耦合，但 当设计良好的RESTful APIs不工作的时候，还是值得尝试。

通过模板化的URL、简单地暴露静态 分层数据模型来开发一个服务，会导致实现出贫血REST。在 一个建模良好的领域中，REST应该支持的不仅仅是简单重复 的数据获取。在一个经过了充分演进的RESTful架构中，业务 事件和抽象概念同样能被建模为资源，并且它的实现应当能有 效地使用超文本、链接关系和媒体类型，从而最大程度地将各 个服务解耦。贫血REST是一个反模式，它与贫血领域模型模 式密切相关，根据它所设计出来的服务，在Richardson成熟度 模型中，会处于成熟度较低的层次。在我们的洞见文章《REST API设计资源模型》中提供

## CMS作为平台来使用

我们看到太多组织为了交付大型和复杂的数据应用，试图将 他们的CMS作为平台来使用，从而陷入困境。这常常是由供 应商的“好意”驱动的，想要帮助业务部门绕过那些响应缓慢 的IT组织，从而可以在生产环境上直接对业务通过拖拽的方式 进行变更。当然我们非常支持给内容生产者提供正确的工具和 工作流，但是对于拥有复杂业务逻辑的应用，我们还是倾向于 把CMS只作为平台的一个组件（通常用hybrid或headless模我们看到太多组织为了交付大型和复杂的数据应用，试图将 他们的CMS作为平台来使用，从而陷入困境。这常常是由供 应商的“好意”驱动的，想要帮助业务部门绕过那些响应缓慢 的IT组织，从而可以在生产环境上直接对业务通过拖拽的方式 进行变更。当然我们非常支持给内容生产者提供正确的工具和 工作流，但是对于拥有复杂业务逻辑的应用，我们还是倾向于 把CMS只作为平台的一个组件（通常用hybrid或headless模

## angulr