REST的主要优势到底是什么？ PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/245/2021_2022_REST_E7_9A_84_E4_B8_BB_c104_245779.htm 在JavaEye论坛上回答网友joyjiang的疑问：“ REST的优势到底是什么?开发效率?文档的管理?url的直观?还是其它的什么优势呢? ”REST的主要优势在我看来其实在于它是一种对于服务器的更加有效的抽象方式。 对于基于网络的应用来说，你怎么样看待服务器，就会产生什么样的架构风格，随之产生与该架构风格相关的交互模式。 RPC架构风格将服务器看作是由一些过程组成，客户端调用这些过程来执行特定的任务。SOAP就是RPC风格的一种架构。过程是动词性的（做某件事），因此RPC建模是以动词为中心的。 分布式对象架构风格认为服务器是由一些对象和对象上的方法组成，客户端通过调用这些对象上的方法来执行特定的任务。并且客户端调用这些对象上的方法应该就像是调用本地对象上的方法一样，这样开发就可以完全按照统一的面向对象方法来做。但是很可惜，这样的抽象并不是很有效，因为分布式对象与本地对象存在着巨大的本质差别，想要掩盖这些差别很多时候甚至是有害无益的。 REST架构风格并没有试图掩盖这些差别，而是将服务器抽象为一组离散资源的集合。资源是一个抽象的概念，而不是代表某个具体的东西。注意：要真正理解REST，就一定要增强自己的抽象思维能力，充分理解到资源是抽象的。如果完全不具有抽象思维的能力，一定要将资源与数据库中的一张表或服务器端的一个文件（HTML、Servlet、JSP、etc.）一一挂起钩来，就无法真正理解REST了。资源是名词性的，因此REST建模

是以名词为中心的。 上述是目前基于网络的应用的主要的三种抽象方式。这三种不同的抽象方式会严重影响客户端与服务器的交互模式，而不同交互模式的交互效率差别相当大。分布式对象的交互模式很多时候效率很低，因为掩盖了分布式对象与本地对象的差别，很多时候都会导致细粒度的API（需要一再强调才能让一些不明就里的架构初哥按照正确的方式来做设计）。实践已经证明，与RPC和分布式对象相比，REST是一种对于服务器更加有效的抽象方式，将会带来粒度更大和更有效率的交互模式。这样的效果与Fielding设计REST的初衷是吻合的，REST就是专门为交互的性能和可伸缩性进行过优化的一种架构风格。而SOAP在设计的时候优先考虑的从来不是性能和可伸缩性，而是互操作性。除非出现奇迹，否则你种什么，就应该长出来什么。你种的是瓜，长出来的就是瓜；你种的是豆，长出来的就是豆。 Fielding写到：“ REST提供了一组架构约束，当作为一个整体来应用时，强调组件交互的可伸缩性、接口的通用性、组件的独立部署、以及用来减少交互延迟、增强安全性、封装遗留系统的中间组件。 ”有人认为REST不是面向对象的，其实REST虽然没有分布式对象那么面向对象，在我看来至少比RPC更加面向对象。按照《企业应用架构模式》，以动词为中心建模是什么？是不是就是事务脚本？以名词为中心建模是什么？是不是就是领域模型？这就扯远了，网络通信是否一定需要实现为面向对象的形式，我认为是不需要的。“ REST的主要优势在我看来其实在于它是一种对于服务器的更加有效的抽象方式。 ”这句话等于是，我先把一个骨架放在这里，还没有用血肉来充实它，也就是还没有举出具体的实例来。具体的

实例以后我们还需要来详细讨论。REST是非常简练的，同时又是一种非常强大的抽象方式，在我看来就是从根本上简化Web开发的一味良药。