



JAVA 达摩班

Web Service



Service-Oriented Architecture (SOA)

1

传统应用 - 集成



全部重做



模块化应用 - 独立



标准连接



如果能...

- 延伸价值链到供应商和客户
 - 快速和有效的响应变化
 - 减少IT成本
 - 更佳划算的IT投资
 - 集成性 - 所有服务很好的集成在一起
- 
- 重用性
 - 松耦合
 - 可发现性
 - 抽象性
 - 组合性

SOA

SOA是一种软件架构，它用于构建由一组松耦合，黑盒组件组成的商业应用。每个组件代表一个独立的服务。

服务之间松耦合

组织和使用来自不同拥有者的服务的能力

统一的方式去发布，发现，交互和调用服务，而不需要知道底层技术细节

满足不断增长的企业系统规模和网络提供的管理需求

服务就像一个网站，是一个可组合的，可全局访问的功能

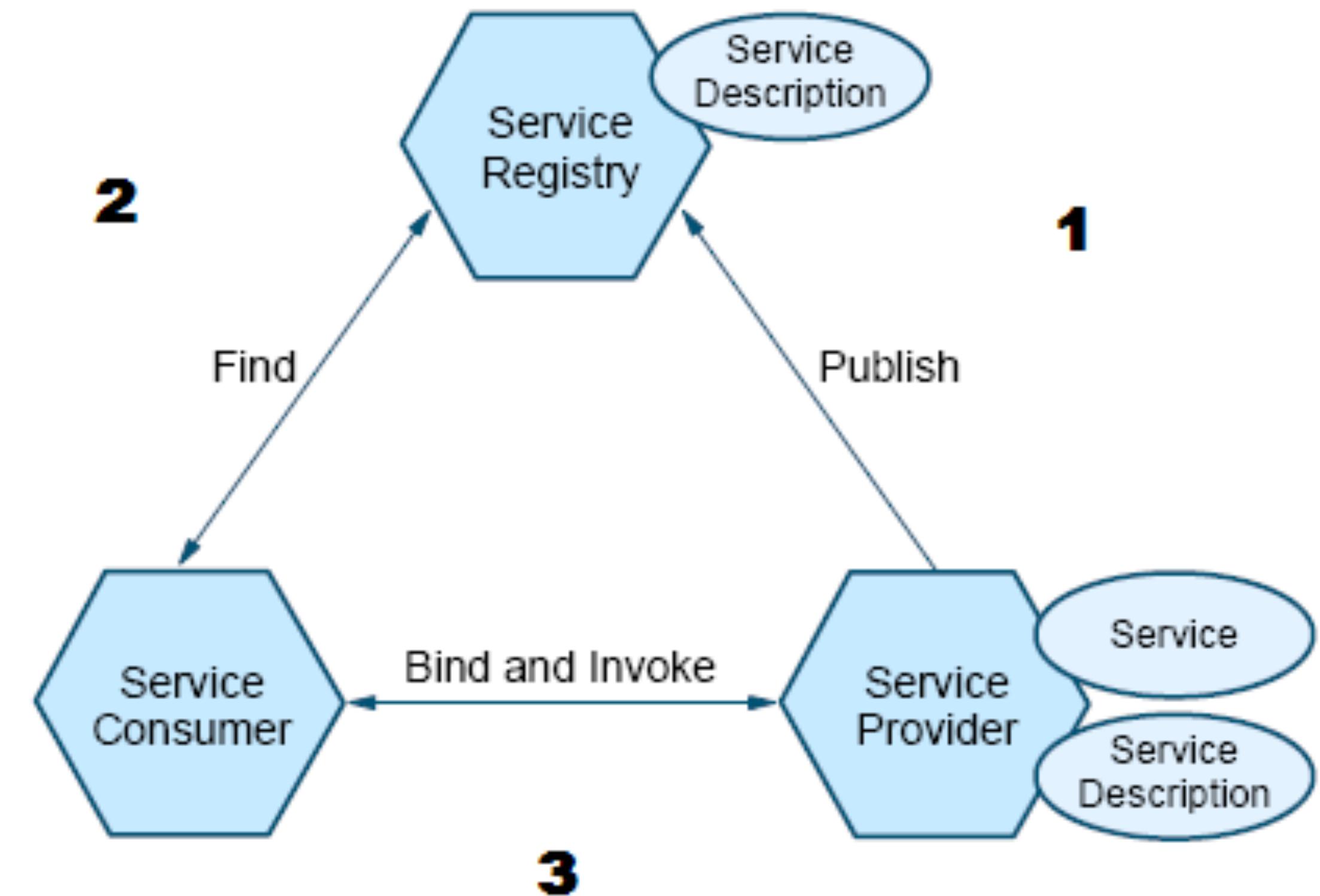
需求和功能独立于SOA存在，服务就是需求和功能组合在一起的机制

e.g. 乘客需求：运输；飞机功能：飞行

SOA

SOA = 服务集合 + 服务连接 (web service)
= 服务注册 + 服务消费者 + 服务提供者

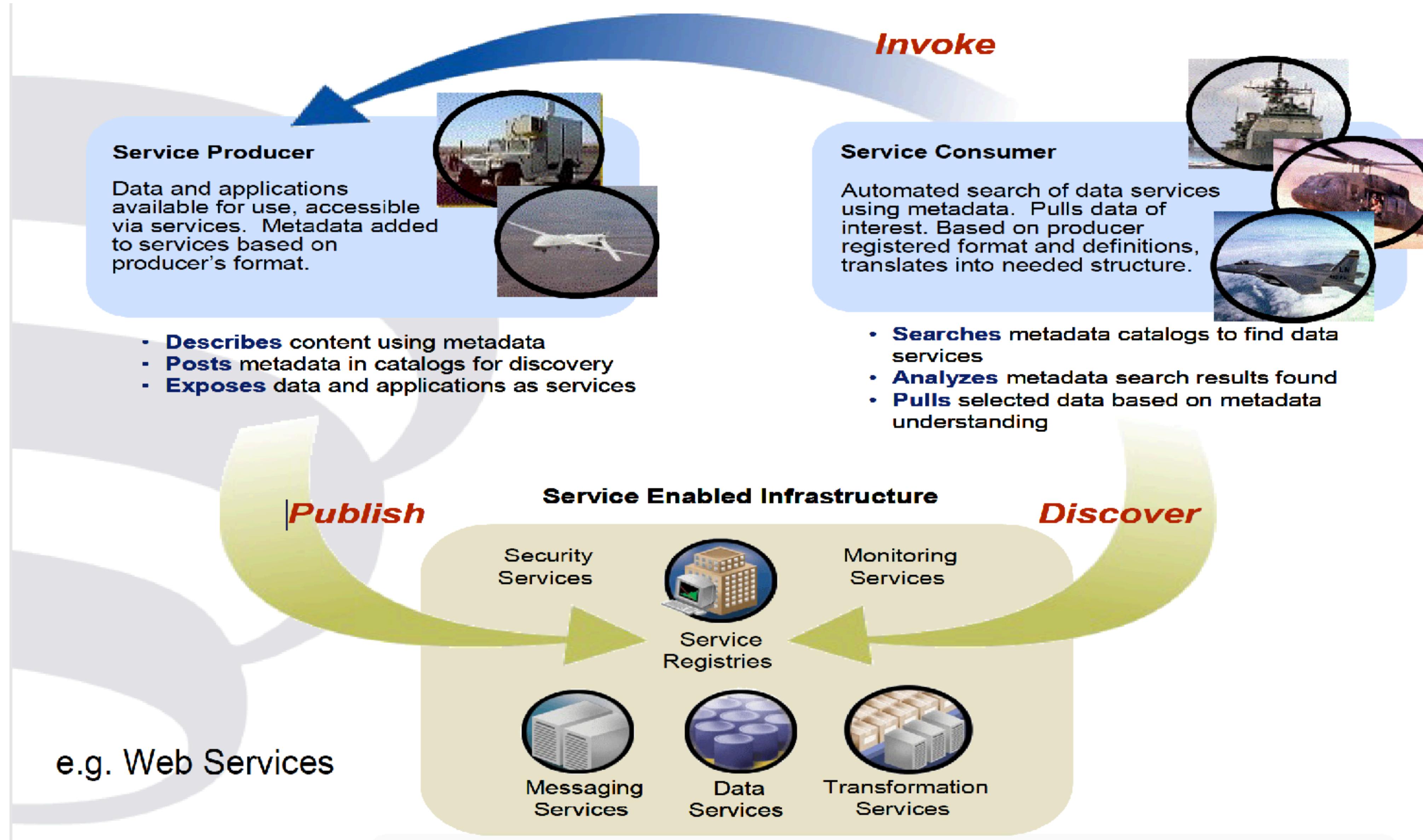
服务 = (良好定义 + 自包含 + 独立) 的功能



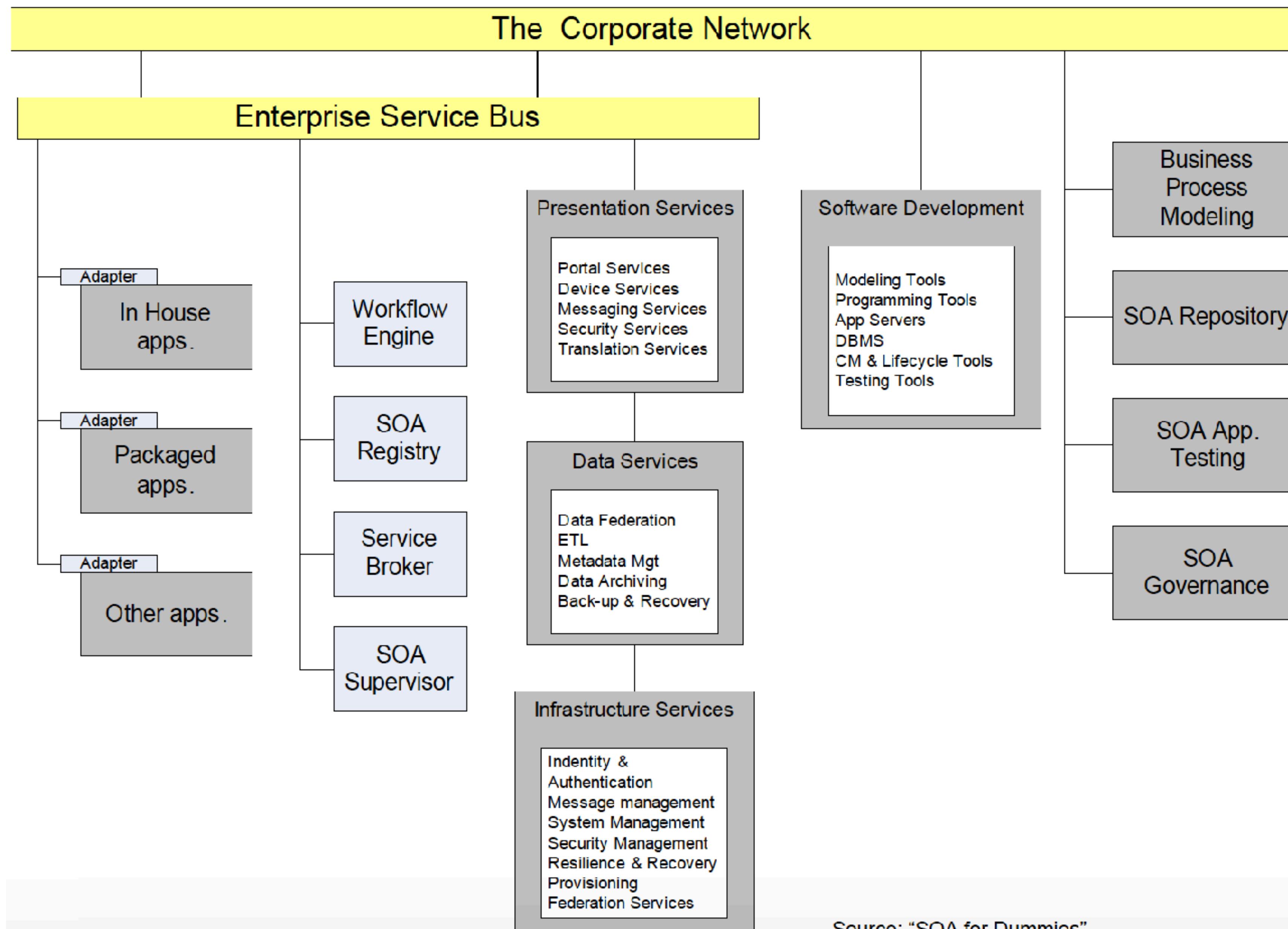
SOA不是...

- 企业级架构
- **web服务**
- 中间件
- C/S架构 - 不满足动态发现
- 面向对象编程
- 分布式计算
- 大型机应用 - 紧耦合，平台依赖

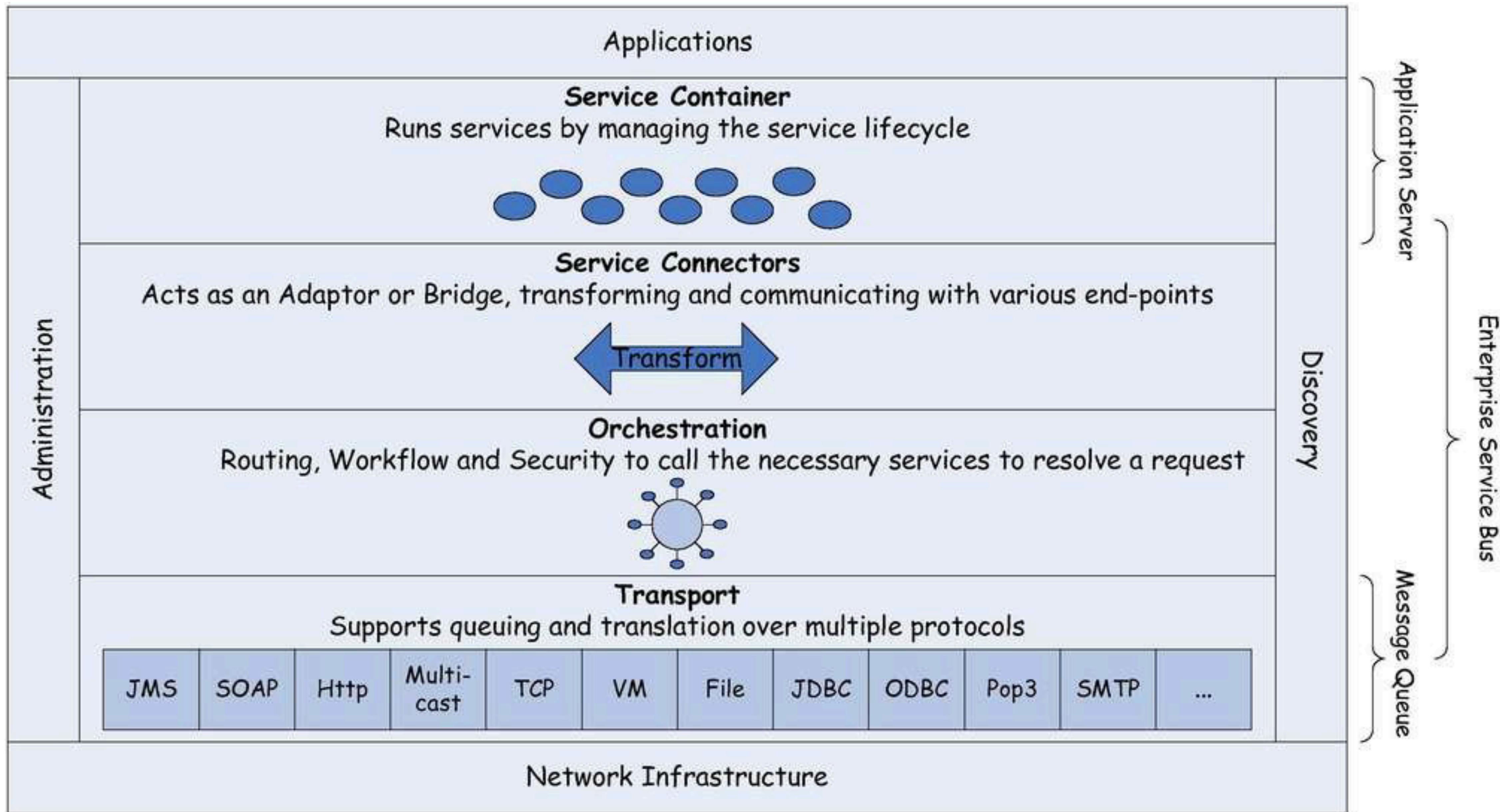
SOA是...



SOA 架构图



SOA 架构图





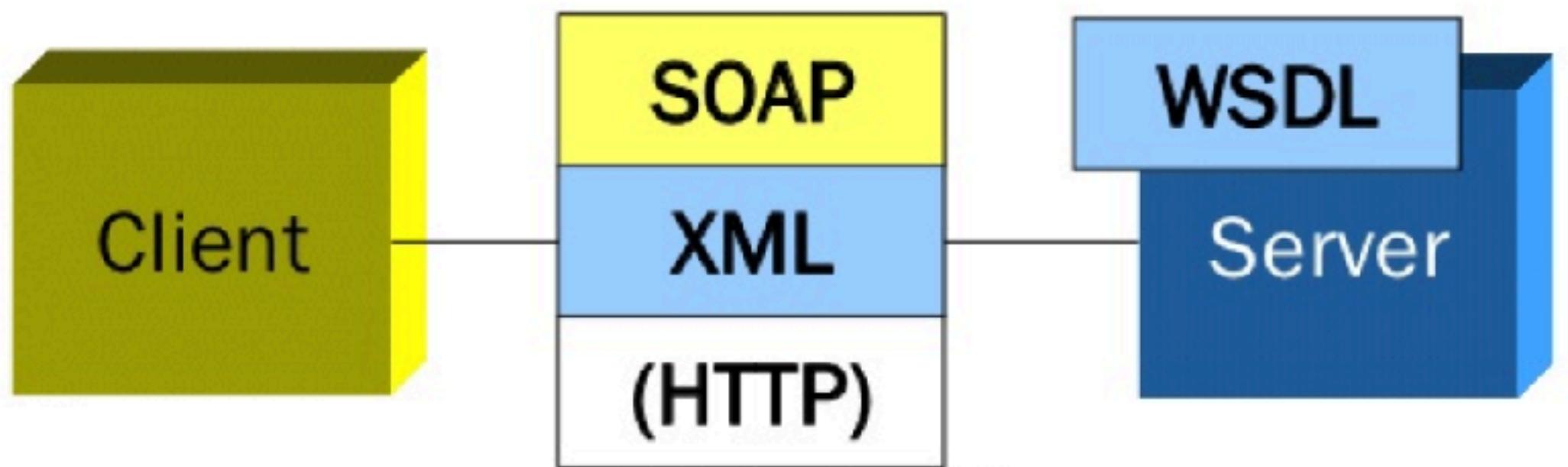
Web Service 2

Web Sites (1992)

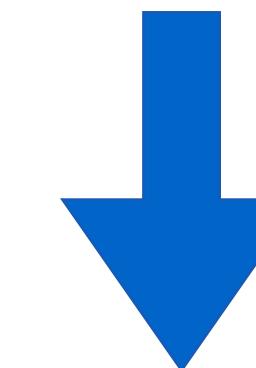


Web服务的发展

WS-* Web Services (2000)

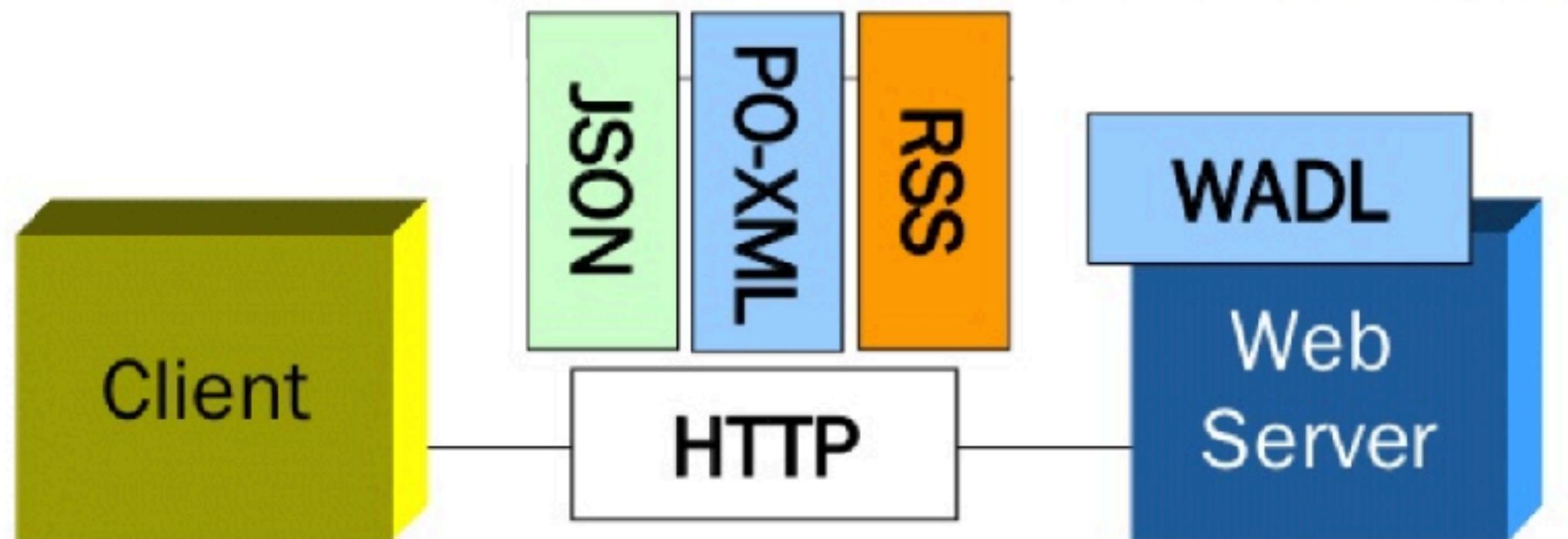


普通 B/S 模型

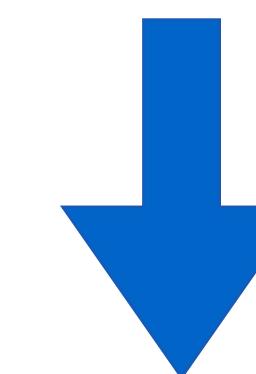


SOAP的W3C标准服务模型

RESTful Web Services (2006)



RESTful基于JSON的无状态Web服务



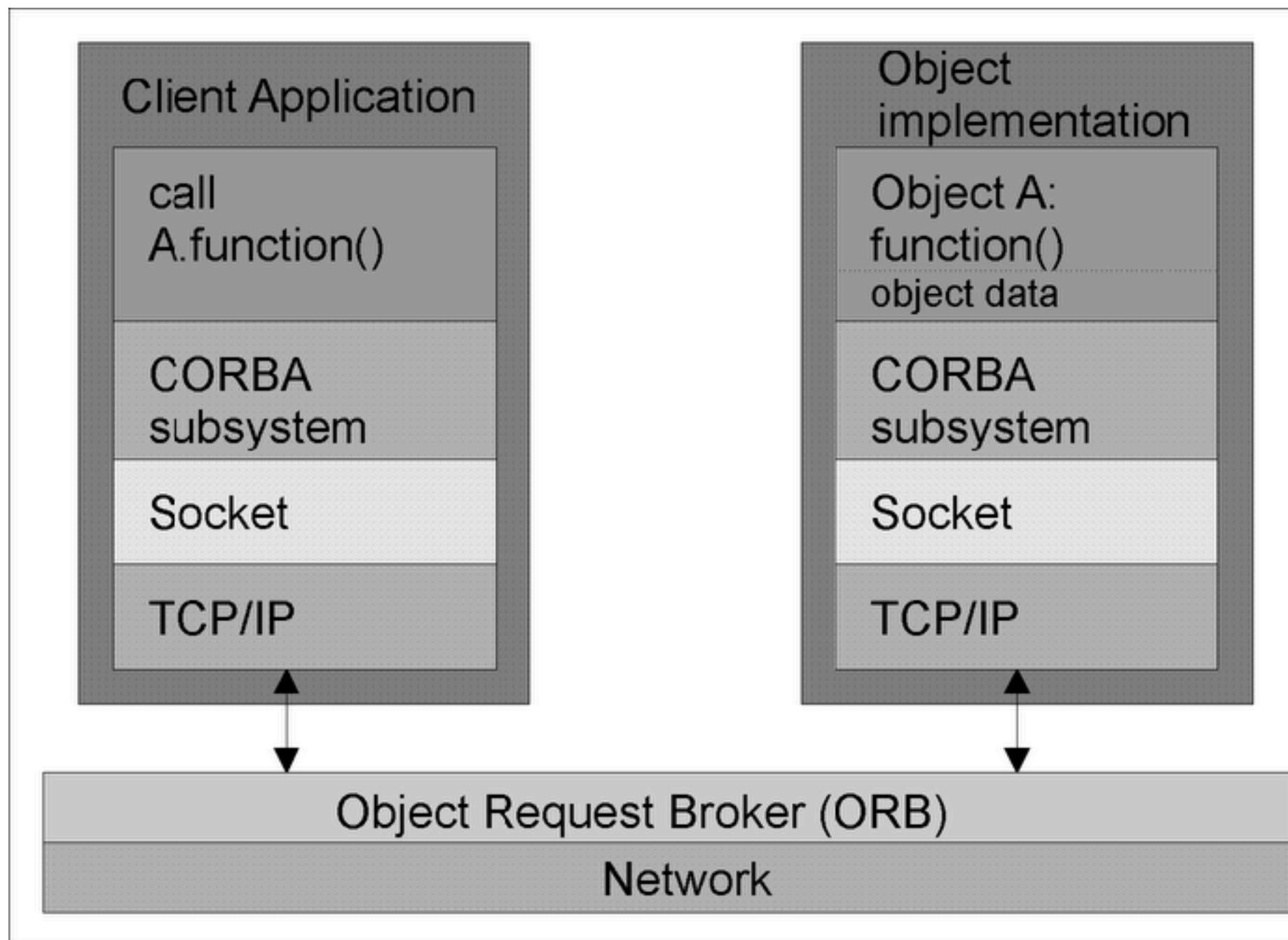
Web服务

Web服务是一种可编程的，基于标准的连接服务的机制

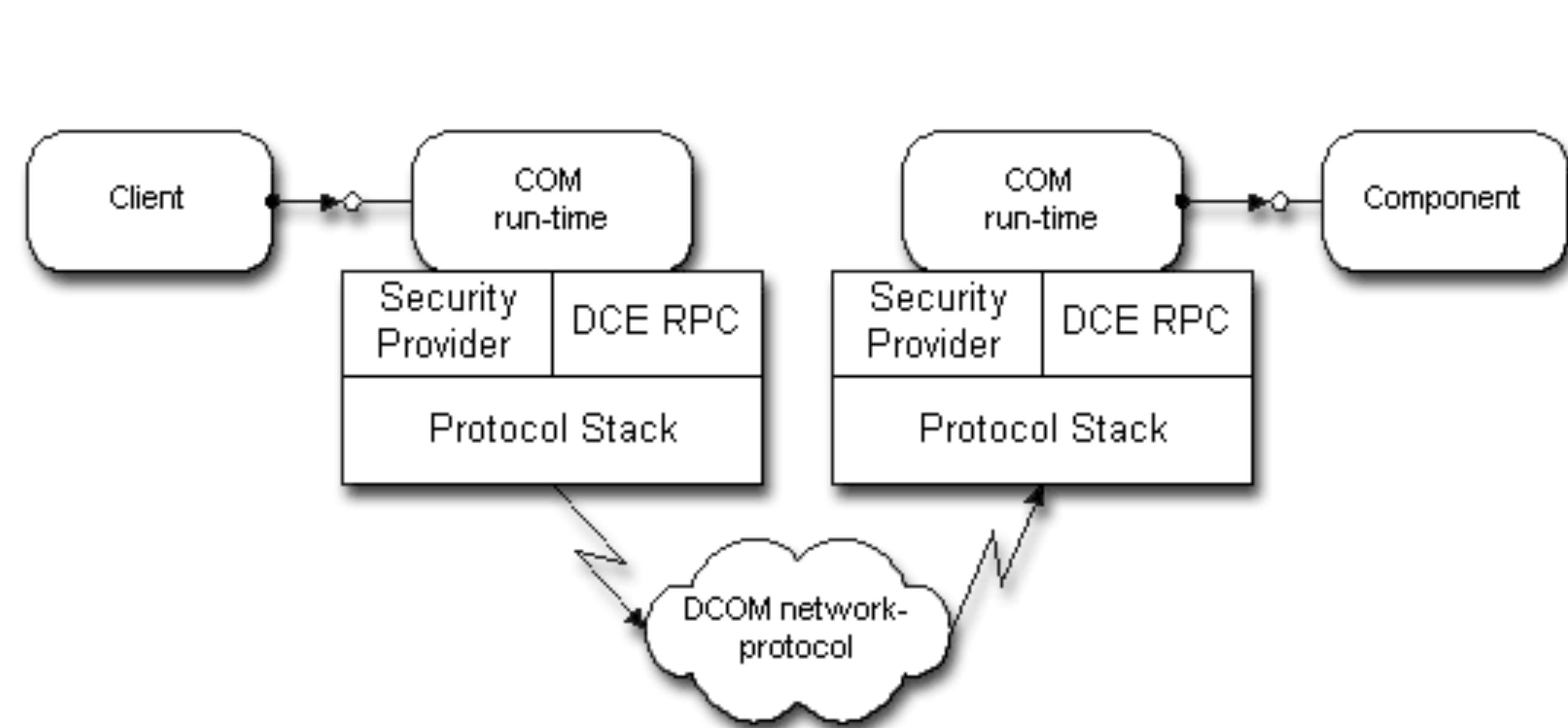
常见的两种Web服务：SOAP 和 REST

非Web服务方式：CORBA, DCOM, EDI等

Common Request Broker Architecture (CORBA)



DCOM



Web服务

Web服务是可发布的，可定位的，可通过网络调用的

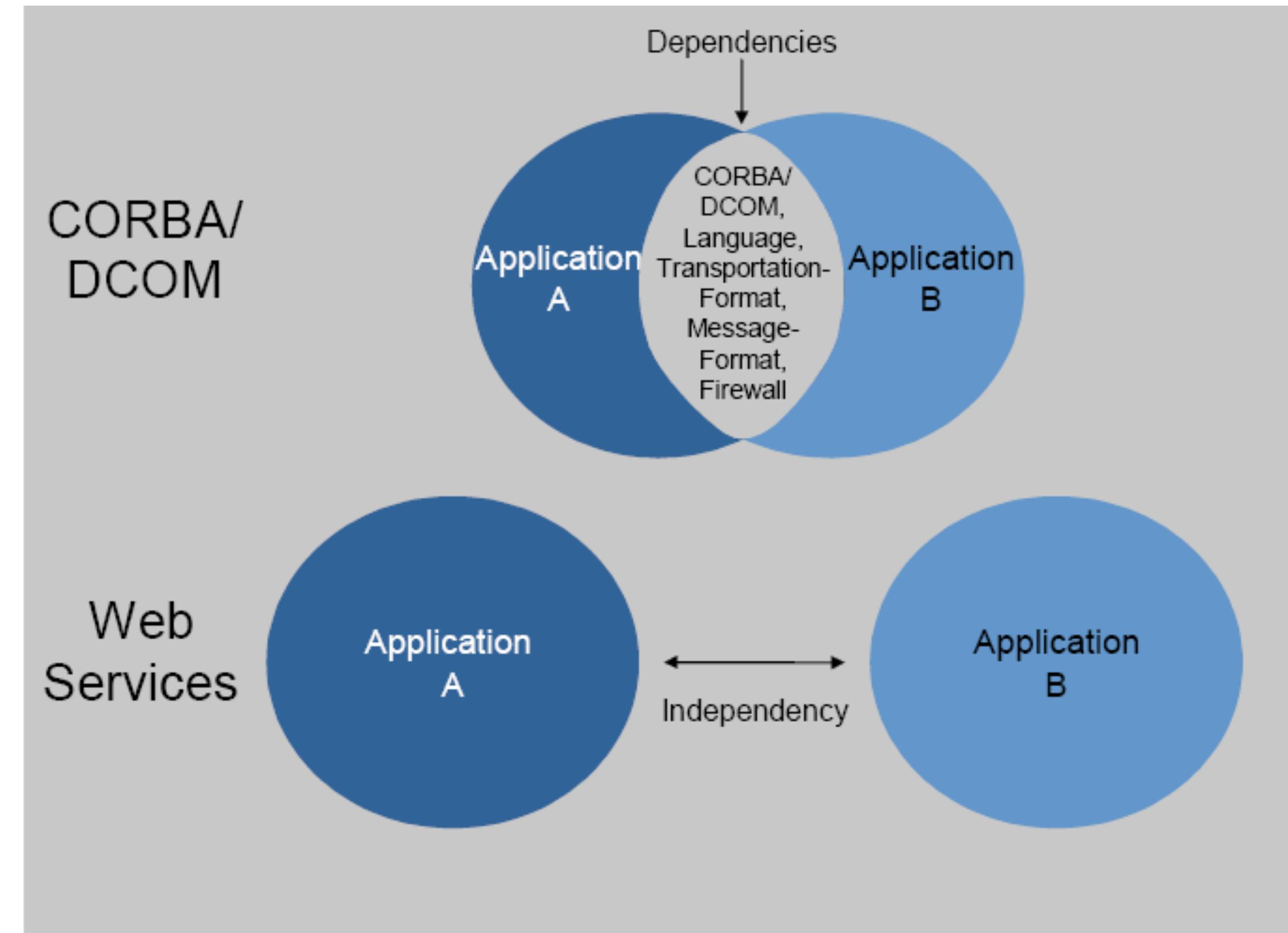
Web服务不绑定用户接口

Web服务不限制开发语言，但遵守一致的通行协议（XML）

Web服务可以来自多个开发者或者组织，不需要同时实现

Web服务特点：

machine-to-machine交互，松耦合，互操作性，基于WWW，
系统独立，平台独立，服务重用，降低成本



SOA vs Web Service

- SOA是架构，属于高层设计概念；Web Service代表一个可重用的业务功能，属于详细设计或者实现概念；
- SOA可以包含多种类型服务，包括web service, window service等；
- Web Service是**SOA**的一部分，可以组装或被组装成其他服务，属于局部概念；



SOAP协议 3

XML

XML代表EXtensible Markup Language，可扩展标记语言

XML是HTML的超集，包含HTML，用于描述数据

XML不是预定义的（HTML是预定义），开发者需要定义自己的标签

XML Schema用来定义XML文档的结构，包括元素，子元素（个数），顺序等

```
<bookshelf>
  <book>
    <title>Microservice</title>
    <author>Ashton</author>
    <price>80</price>
  </book>
</bookshelf>
```

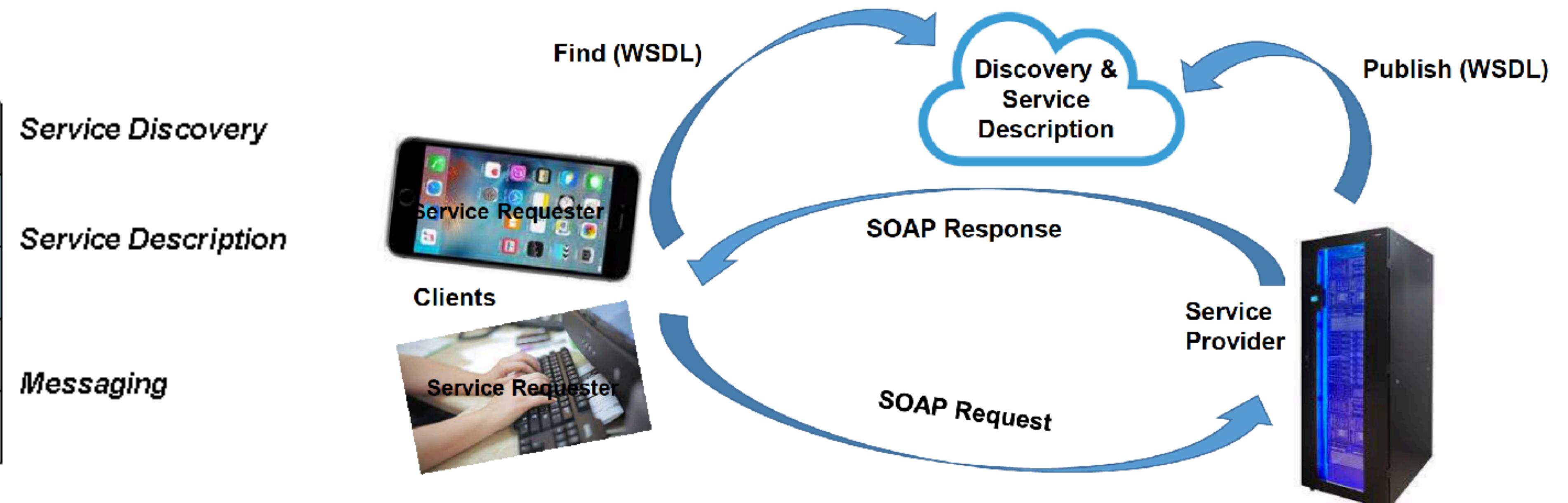
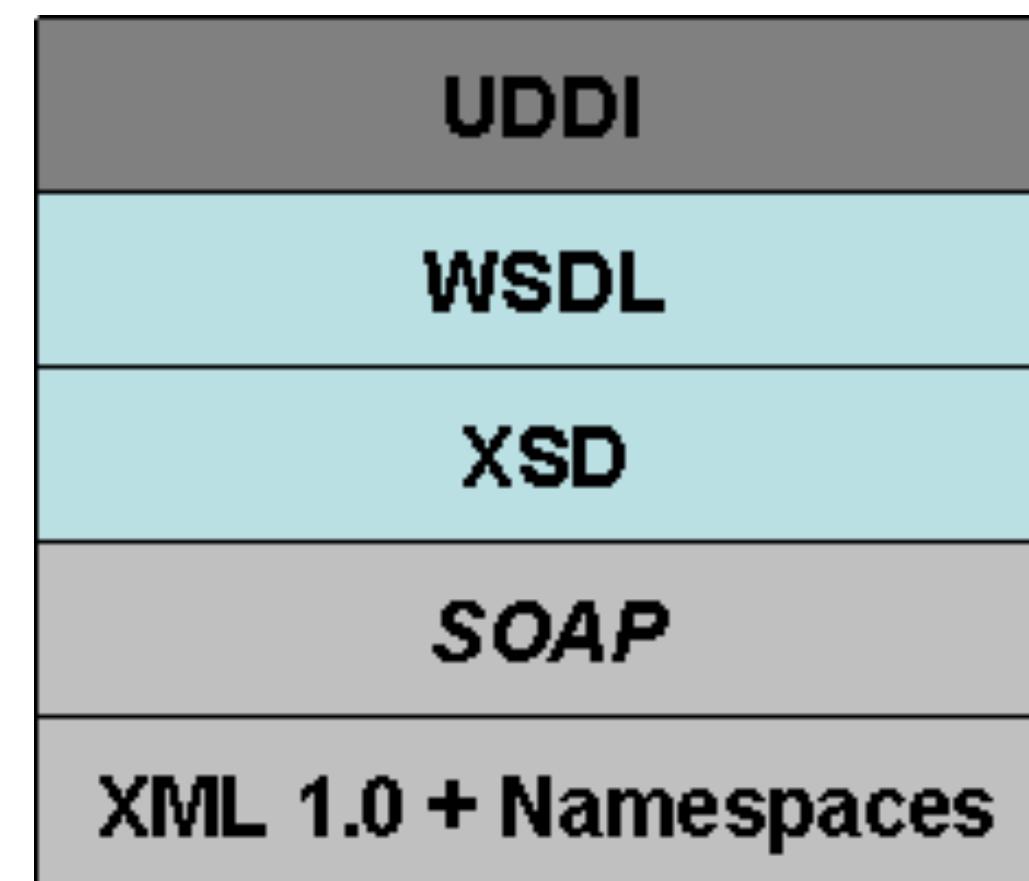
SOAP规范

UDDI (Universal Description, Discovery, and Integration) : 平台独立的描述, 发现Web服务和服务提供商的方式

WSDL (Web Service Description Language) : 用于定义服务为一组网络终端 (接口)

SOAP: 简单, 轻量级的实现结构化信息交换机制

XML: 消息格式



SOAP概念



SOAP代表Simple Object Access Protocol, 简单对象访问协议

误区：SOAP不简单，也不是对象访问协议

SOAP在1998年由微软公司的Dave Winer提出

W3C制订的基于XML，面向消息的Web服务实现

通过网络基于协议传输：HTTP, FTP, SMTP, 甚至TCP

无状态的，单向的消息交换范式

基于WSDL实现服务发现

相比DCOM, CORBA, RMI等机制简单，但学习曲线陡峭

SOAP信封

使用XML交换结构化的，类型定义的消息

消息 = (信封 = 0/1 header + 1 body) + 0/1 附件

信封 (Envelope) 是SOAP消息的根结点

消息头 (Header) 包含元数据，安全信息

消息体 (body) 包含XML数据，异常数据

附件一般用于附带二进制数据，例如图片等

```
<soap:Envelope  
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
    <soap:Header><!-- optional -->  
        <!-- header blocks go here... -->  
    </soap:Header>  
    <soap:Body>  
        <!-- payload or Fault element goes here... -->  
    </soap:Body>  
</soap:Envelope>
```

<http://schemas.xmlsoap.org/soap/envelope/>





REST风格

4

REST概念

REST代表**Representational State Transfer**, 代表性状态转移

首次出现在2000年Roy Fielding的博士论文中，他是HTTP规范的主要编写者之一

是一组架构约束条件和原则，满足这些约束条件和原则的设计就是RESTful

使用简单的HTTP协议和方法

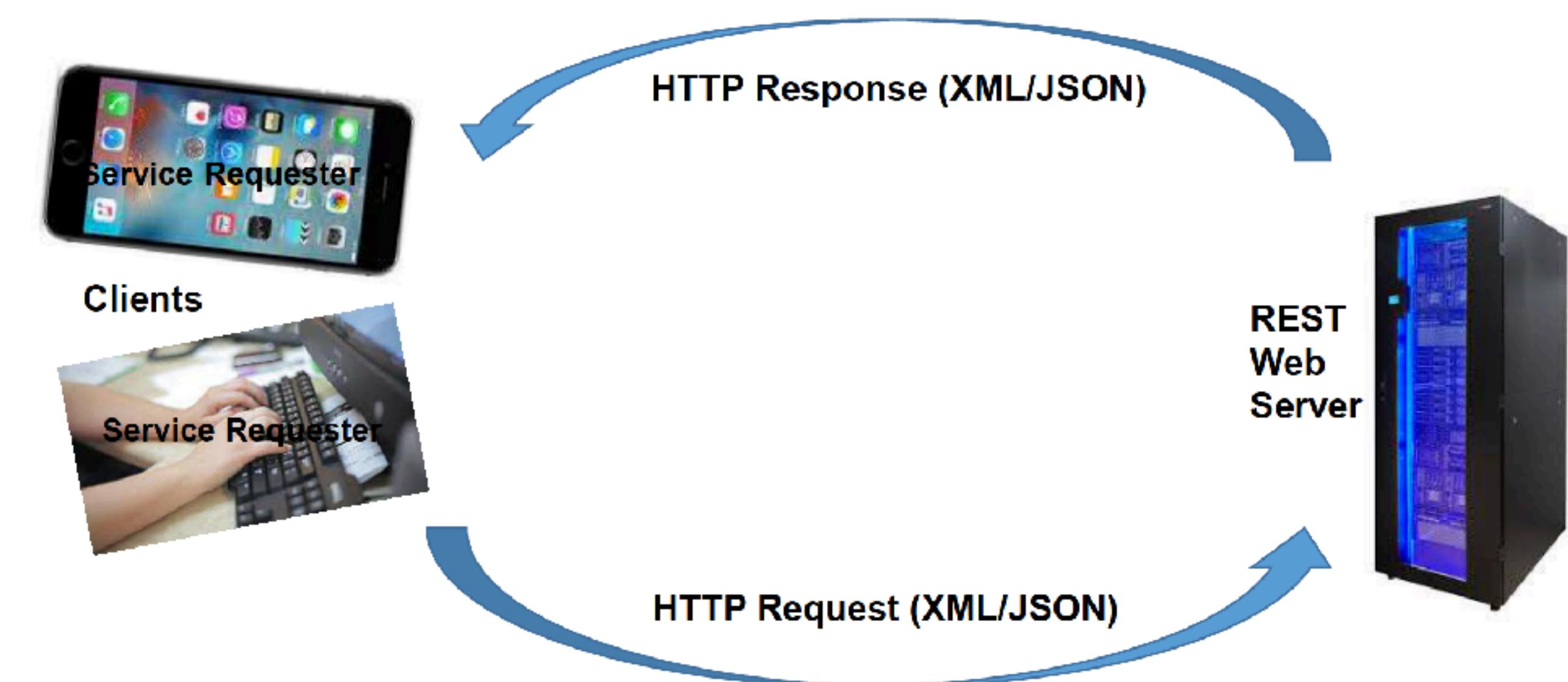
GET: 返回数据, 不改变数据

POST: 创建, 更新, 删除数据

PUT: 替换引用的资源

PATCH: 修改引用的资源

DELETE: 删除引用的资源



REST设计规范（约束）

- **客户端/服务器 (Client/Server)** : 用户接口和数据存储相分离，客户端和服务器相互独立
- **统一接口 (Uniform Interface)** : 所有资源都通过统一的接口（基于HTTP）访问
- **无状态 (Stateless)** : 从客户端到服务器的请求包含所有信息，服务器不存储客户端的session或context数据
- **分层系统 (Layered System)** : 架构分层，每一层都不能“越层访问”
- **缓存 (Cacheable)** : 可以指定数据缓存与否，HTTP响应必须被客户端缓存
- **按需开发 (Code On Demand)** : 客户端扩展性，通过下载并以scripts形式封装

REST 方法

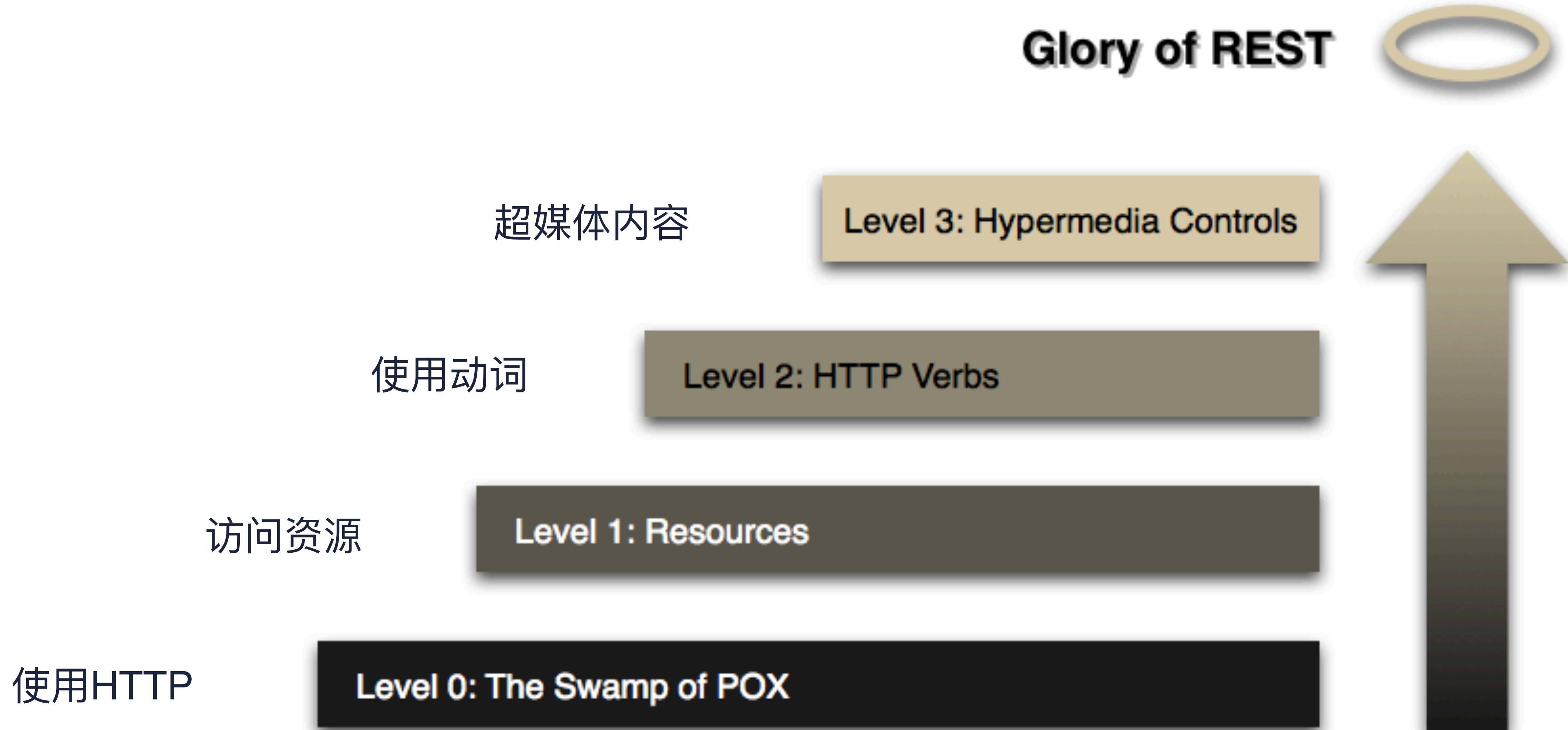
HTTP Method	URI	CRUD	Request Stream	Response Stream	Response Code
POST	/customers	Create	Customer without id	customer	201 / 404 / 409
GET	/customers	Read	n/a	Customers collection	200 / 404
GET	/customers/{id}	Read	n/a	Customer	200 / 404
PUT	/customers/{id}	Update	Customer	n/a	200 / 204 / 404
DELETE	/customers/{id}	Delete	n/a	n/a	200 / 404
OPTIONS	/customers/	Available Methods	n/a	Available Methods	200 / 204

SOAP vs REST

SOAP	RESTful
基于XML的消息协议	一种架构风格
使用WSDL实现消费者和提供者的通信	使用XML或JSON收发数据
面向服务的 - 通过RPC调用服务	面向资源的 - 使用URI和方法 (GET, PUT, POST, DELETE) 暴露资源
支持有状态的实现	完全无状态的
多种协议HTTP, SMTP, FTP等	只支持HTTP
属于分布式计算类型实现	属于Web类型 (C/S) 实现
很难实现前端Javascript调用	容易实现 (更多) 前端Javascript访问
占用更多的资源和带宽	很少的资源和带宽
只支持XML	支持多种格式: 纯文本, HTML, XML, 推荐JSON

理查德森成熟度模型 RMM

- Leonard Richardson

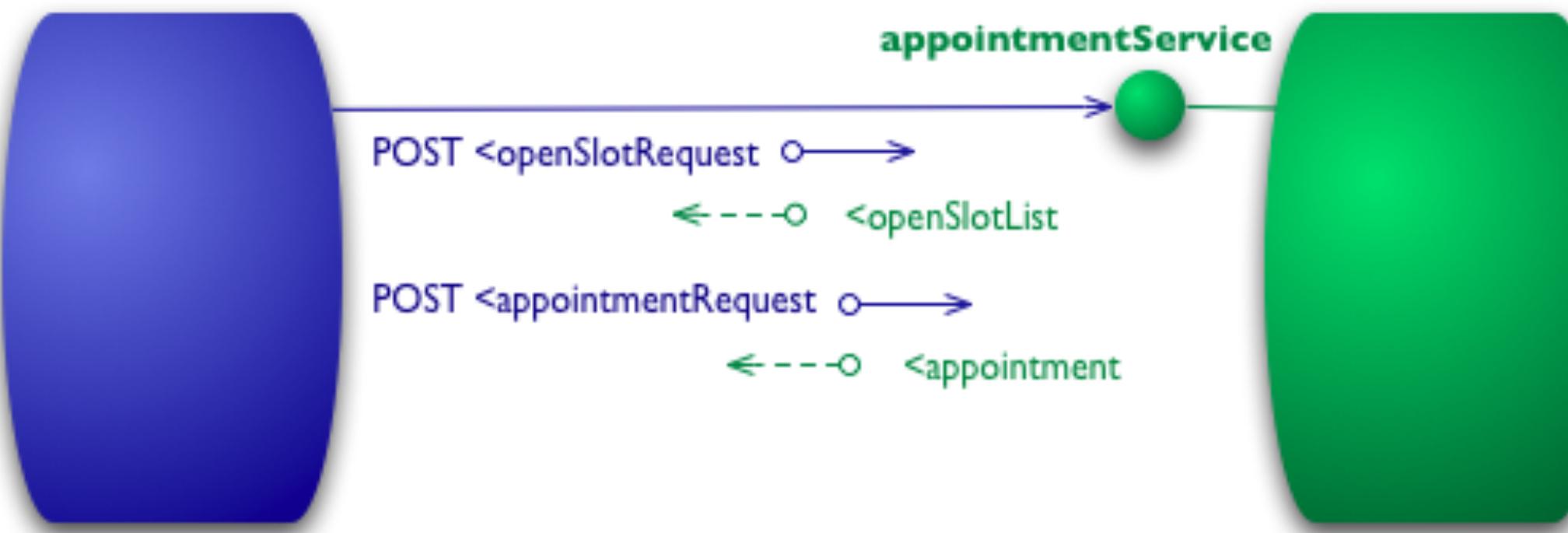


理查德森成熟度模型RMM

- Leonard Richardson

Level 0

- 只要使用HTTP作为远程交互机制
- RPC风格，和SOAP的区别只是消息格式



发送POST请求预约2018-05-01当日名为ashton的医生

POST /appointmentService HTTP/1.1

[various headers]

<openSlotRequest

date = "2018-05-01" doctor = "ashton"/>

返回ashton医生当日可用的预约

HTTP/1.1 200 OK

[various headers]

<openSlotList>

 <slot start = "1400" end = "1450">

 <doctor id = "ashton"/>

 </slot>

 <slot start = "1600" end = "1650">

 <doctor id = "ashton"/>

 </slot>

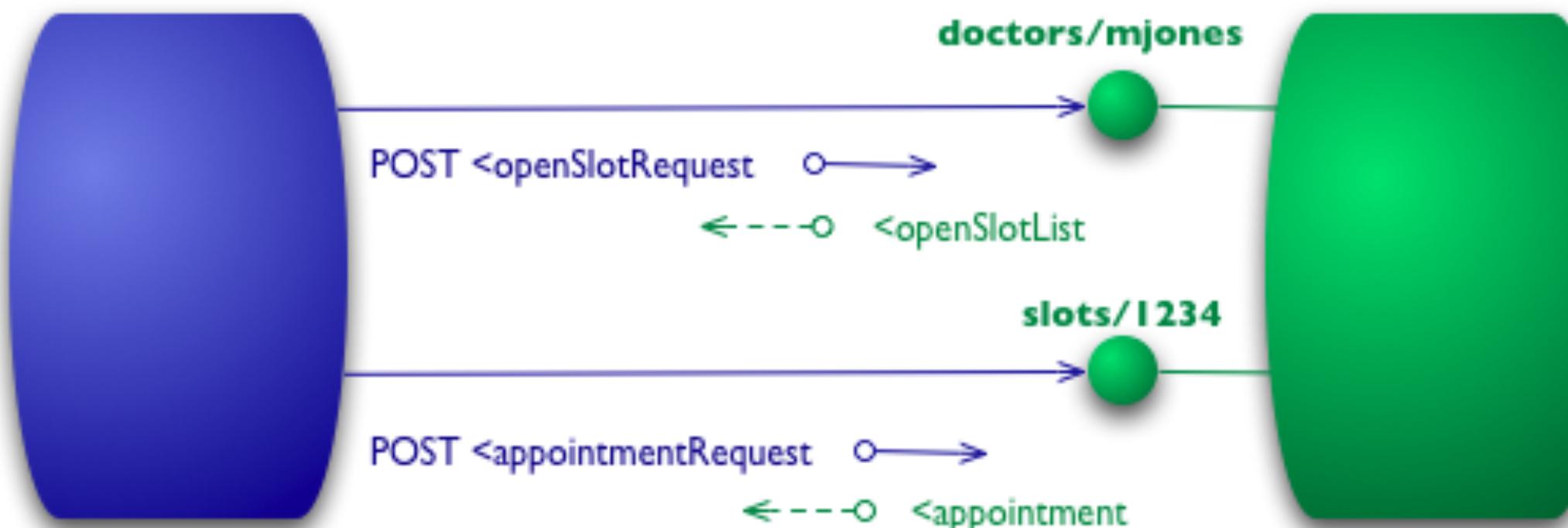
</openSlotList>

理查德森成熟度模型RMM

- Leonard Richardson

Level 1 - 资源

- 直接请求特定资源，而不是发给服务终端
- 资源以URI的形式传递
- 资源可以有多级，即子资源



请求和ashton医生“对话”，查询2018-05-01当天可用的预约

```
POST /doctors/ashton HTTP/1.1
[various other headers]
<openSlotRequest date = "2018-05-01"/>
```

返回结果类似，该医生当天可用预约

```
HTTP/1.1 200 OK
[various headers]
```

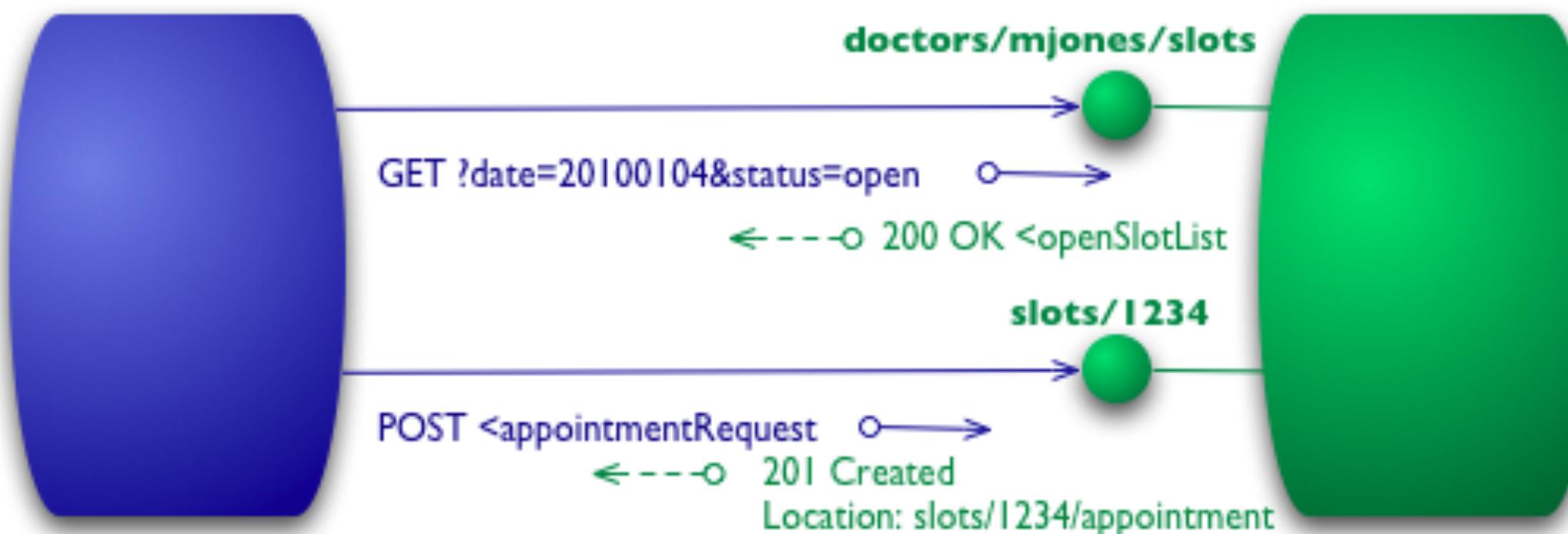
```
<openSlotList>
    <slot id = "1234" doctor = "ashton"
        start = "1400" end = "1450"/>
    <slot id = "5678" doctor = "ashton"
        start = "1600" end = "1650"/>
</openSlotList>
```

理查德森成熟度模型RMM

- Leonard Richardson

Level 2 - 动词

- 使用HTTP所有（合理）的动词发送请求
- 分离安全(get)和非安全(post, put, delete)操作
- 使用合理的状态码表示响应状态



发送GET请求资源为ashton在20180501当前可用预约

GET /doctors/ashton/slots?date=20180501&status=open HTTP/1.1

返回当前可用预约

HTTP/1.1 200 OK

<openSlotList>

<slot id = "1234" doctor = "ashton" start = "1400" end = "1450"/>

<slot id = "5678" doctor = "ashton" start = "1600" end = "1650"/>

</openSlotList>

发送POST请求资源为1234的预约，请求体包含病人信息

POST /slots/1234 HTTP/1.1

<appointmentRequest>

<patient id = "imsick"/>

</appointmentRequest>

返回预约结果

HTTP/1.1 201 Created

Location: slots/1234/appointment

<appointment>

<slot id = "1234" doctor = "ashton" start = "1400" end = "1450"/>

<patient id = "imsick"/>

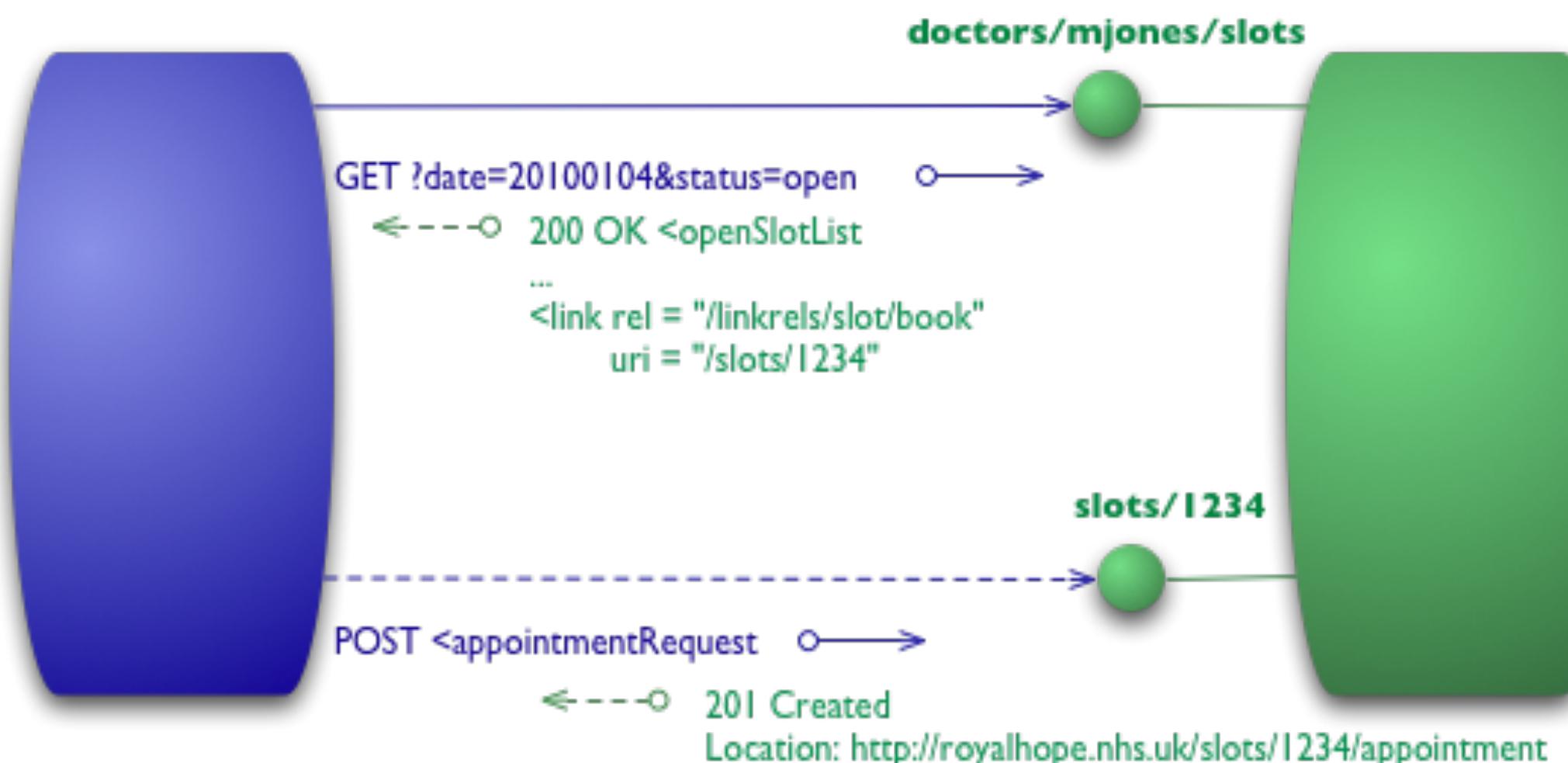
</appointment>

理查德森成熟度模型 RMM

- Leonard Richardson

Level 3 - 超媒体

- 满足 *HATEOAS (Hypertext As The Engine Of Application State)* 约束, RMM 最高级
- 返回结果包含可以发现该资源更多动作的链接
- 用户可以用来“预测”下一步的行为
- 服务器增加或改变链接而不影响用户使用



请求内容和RMM-2相同

GET /doctors/ashton/slots?date=20180501&status=open HTTP/1.1

返回结果中包含代表“下一步行为”的links

HTTP/1.1 200 OK

```
<openSlotList>
  <slot id = "1234" doctor = "ashton"
    start = "1400" end = "1450">
    <link rel = "/linkrels/slot/book"
      uri = "/slots/1234"/>
  </slot>
  <slot id = "5678" doctor = "ashton"
    start = "1600" end = "1650">
    <link rel = "/linkrels/slot/book"
      uri = "/slots/5678"/>
  </slot>
</openSlotList>
```

REST案例分析

为玩具店实现一套基于RESTful API
实现在线购物功能

假设 <https://dharma-mall.cn/api/v1/>

/carts – 购物车

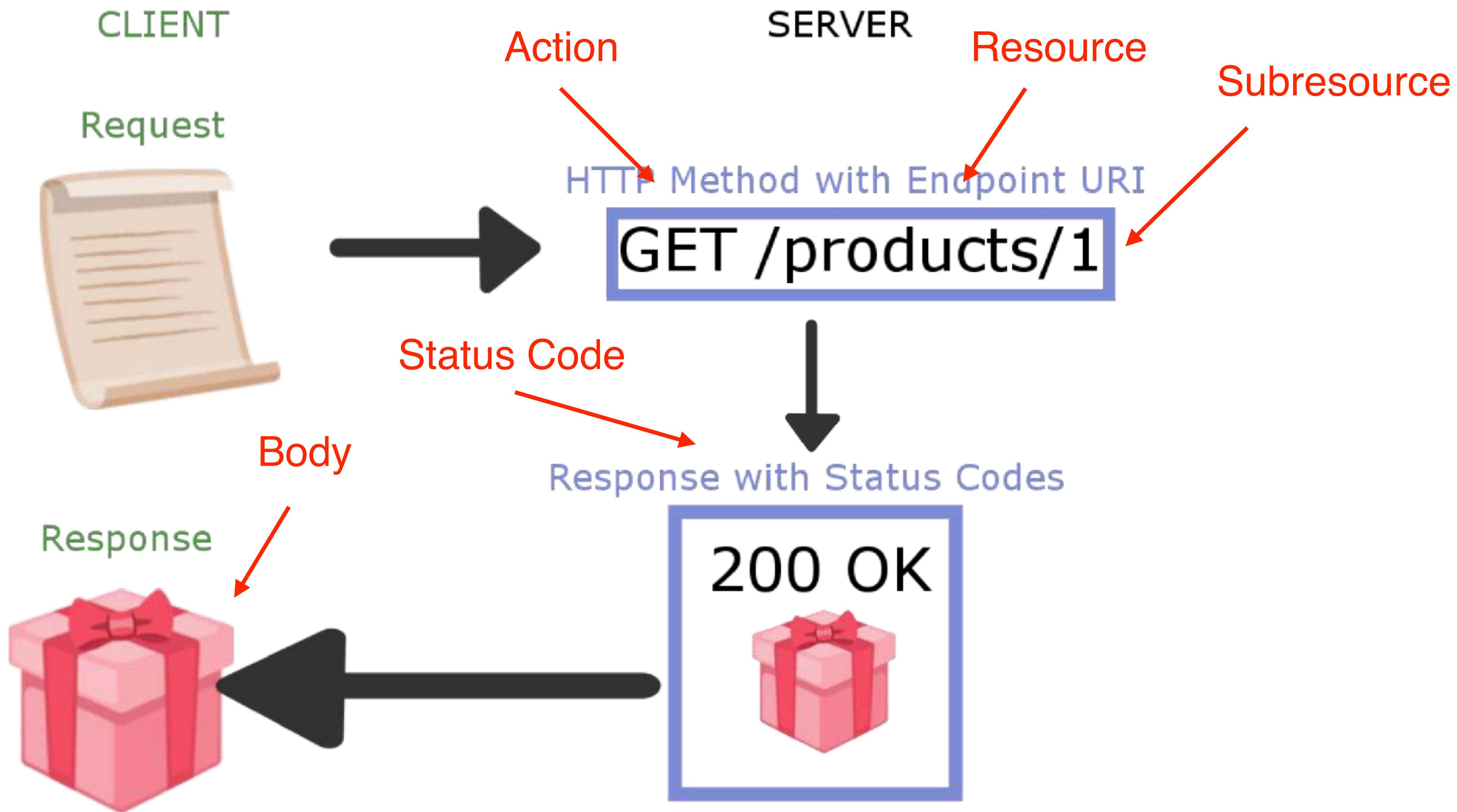
/checkout – 检查订单

/products – 产品列表

/wishlist – 愿望单



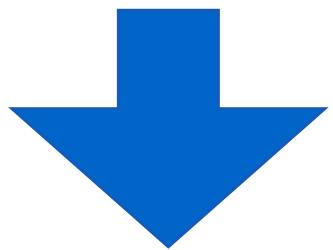
REST案例分析



REST案例分析

请求 = 动词 + 资源 + 子资源

GET /products/bear/teddy



/



/



/



REST案例分析

响应 = 状态码 + 响应体

200 OK

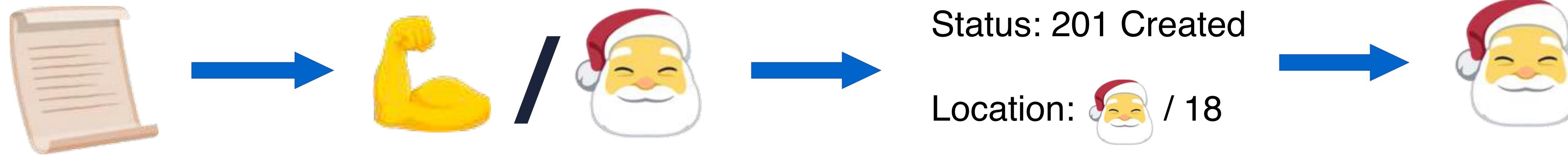
- 1xx *Informational*
- 2xx *Success*
- 3xx *Redirection*
- 4xx *Client Error*
- 5xx *Server Error*

告诉我们请求去往哪里，我们应该怎么办



REST案例分析

创建心愿单 POST



搜索玩具 GET



REST案例分析

添加玩具到心愿单 POST



添加更多玩具到心愿单 POST



REST案例分析

替换心愿单的玩具 PUT

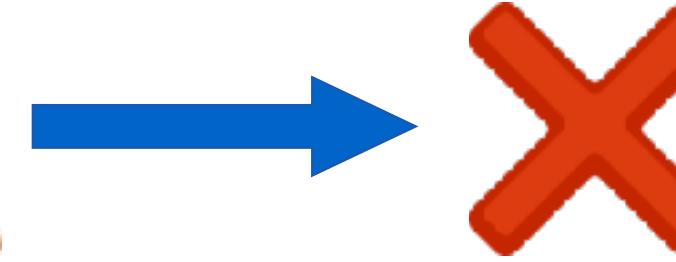


修改心愿单的玩具 PATCH



REST案例分析

删除心愿单的玩具 DELETE



/18/ /15



200 OK

QTY: 2

总结

SOA是一种面向服务的软件架构

Web Service是SOA服务连接中的一种形式

Web Service常见的两种形式是SOAP和RESTful



Thanks!

Any questions?