[云计算导论2. 服务器](#_Toc31689)

[1. 服务计算原理和基础 2](#_Toc17065)

1. 服务定义 2

2. 服务模型 3

3. 面向服务 4

4. SOA 5

5. Web服务 7

***laaS、PaaS、SaaS、面向服务、关注点分离SOC、面向服务架构SOA、消除依赖、Web、  
分布式、W3C、自包含、自描述、URI、AXIS(Apache的SOAP服务器)***

[2. XML 9](#_Toc22926)

1. 简介 9

2. 整体结构 9

3. XML有效的书写 10

***标记语言、传输\交换\存储数据、自描述、W3C、well-formed***

[3. SOAP 12](#_Toc27201)

1. Intro WSDL 12

2. SOAP简介 13

3. Why SOAP 13

***WSDL、接口描述、XML、HTTP、相对低速***

[4. REST服务 13](#_Toc12343)

1. REST简介 13

2. 什么是资源Resource 14

3. 资源表述Representation 15

4. 超链：将资源链接到一起Hateoas 16

5. 统一资源访问接口(基于HTTP) 17

6. 设计及开发REST服务 18

7.联系：学生信息管理系统 18

***表述性状态转移、架构风格、URI(URL,URN)、资源的表述、JSON、相对快速、  
HTTP对应的CRUD方法、幂等性***

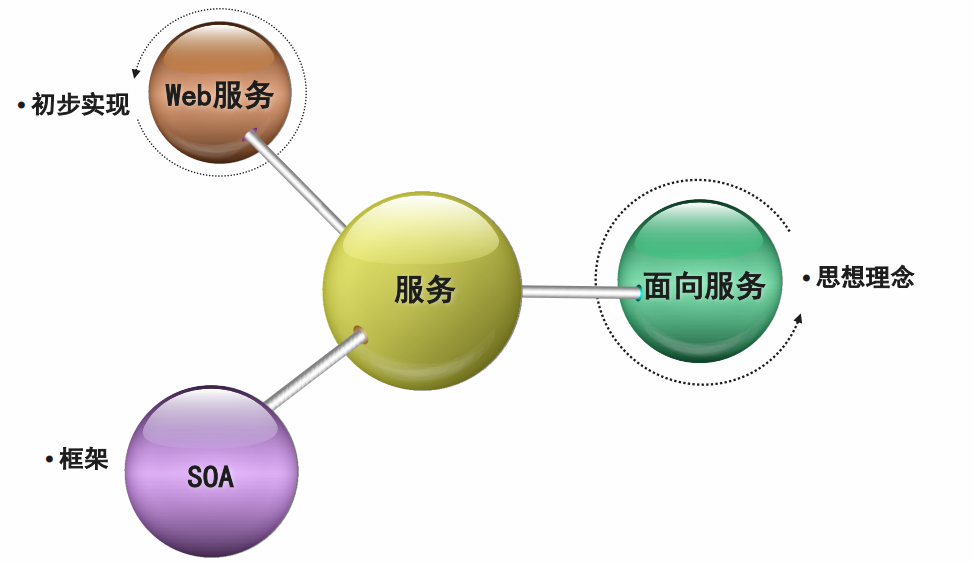
# 云计算导论2. 服务器

### 服务计算原理和基础

###### 一般的服务特性

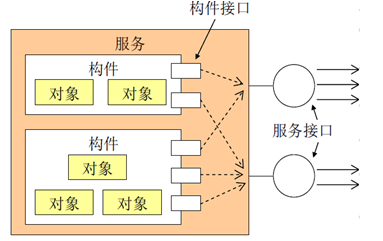
* Intangibility无形性：服务是一种活动，不是实物，看不见摸不着
* Inseparability不可分离性：服务提供者和使用者都包含在服务的生产过程中

###### 服务计算的四个核心概念

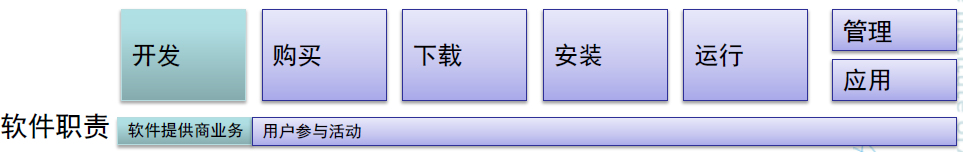


###### 服务定义

* 服务是使用产品和资源创造价值的活动，它没有固定的表现形态，使用方法与具体的实现技术无关，其使用过程通常表现为一个多角色参与的、具有时效性的用户交互过程服务组件：服务通常对应于一个**定义良好的、可寻址以及可组合的软件组件**在软件系统的构建过程中，服务组件又常被狭义称为“服务
* 含义
  + 外特性看：**一个服务被定义为显式的、独立于服务具体实现细节的接口**
  + 内特性看：**服务封装了可复用的业务功能**，这些功能通常是大粒度业务，如业务过程、业务活动等。服务的实现可采用任何技术平台，如J2EE、.Net等



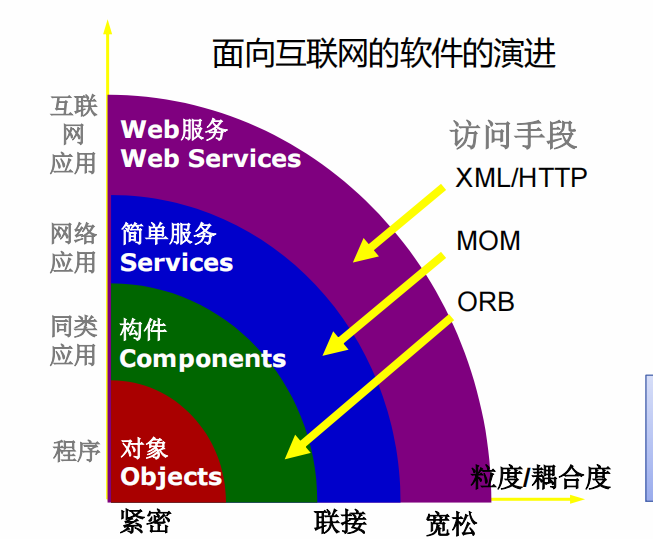
* 传统与软件服务

* 服务模型
  + 基础设施即服务laaS：为用户提供按需付费的**基础设施服务**，例如虚拟服务器、存储等
  + 平台即服务PaaS：为用户提供基于可扩展的大规模基础设施的**平台环境服务**，例如云应用开发与运行环境、用户数据、信息资源、公共服务能力
  + 软件即服务SaaS：为用户提供基于云基础架构的**应用软件服务**，例如CRM、ERP、eHR



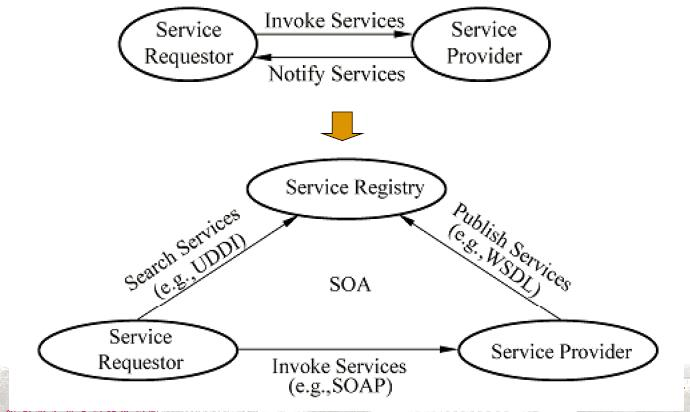
###### 面向服务



* **代表了一种关注点分离(Separation of concerns, SOC)的复杂软件开发方法**
  + **将一个复杂问题分解为更小的、相关的部分，每个部分都能被更好的创建、执行和管理**
  + **每个部分都对应一个关注点或者问题的一个特定部分**
* 面向服务的软件开发方法
  + 组成复杂软件的基本逻辑称为“单元”
  + **服务自治、但并非孤立的**
  + **服务具有一定的通用性和标准化程度，但可以独立演化**

###### SOA(Service-Oriented Architecture)

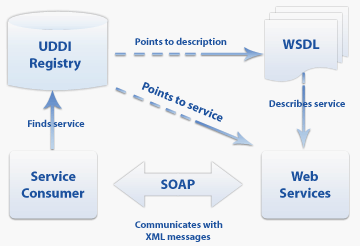
* **定义：SOA是一种基于服务组织计算资源，具有松耦合和间接服务寻址能力的软件体系结构风格**
* **多角度诠释**
  + **体系结构：SOA是一种软件体系结构风格**
    - 软件体系结构：如何将一个分布式系统从逻辑上分解成相关组件，并且定义这些软件组件之间所具有的各种关系，探索如何以最佳方式划分一个系统、如何标识组件、组件之间如何通信、信息如何沟通，怎么能够有序地进化，以及上述所有东西如何能够使用形式化的和非形式化符号加以描述
    - 软件体系结构风格：人们在设计软件体系结构时，对经常出现的问题（如性能问题、安全问题等）进行描述并给出的成熟可靠的解决方案，是人们经验和知识的总结。例如常见的分层体系结构风格就有助于不同组件之间实现信息隐藏。不同的软件体系结构风格是可以被组合的，从而实现更为复杂的设计目标
  + 更广泛的技术角度：SOA还是**包含了编程模型和方法学**等在内的一整套系统构造方法和环境
  + 软件开发商：SOA作为其中间件产品的**软件架构理念**(有时成为产品推广“口号”)
  + 业务角度：SOA尤指**以松散耦合方式**实现业务层面各种服务能力的共享、供给和重学得到方法学
* **SOA基本原理：体系结构**



* **消除依赖**
  + 消除语言、平台和厂商依赖：服务描述对消费者隐藏了服务的实现细节，消除了传统构件所具有的语言、平台和厂商相关性等假依赖关系
  + 消除时间依赖：对服务异步交互方式的支持是消除时间依赖的关键所在
  + 消除访问地址依赖：通过把服务访问地址的相关信息注册到服务注册中心，消费者可以通过检索来发现需要的服务。这使得消费者在使用服务前不再一定需要知道服务的访问地址，从而消除了对访问地址的依赖
  + 消除访问协议依赖：消费者可以不用关心服务的具体实现细节，并以统一的方式和服务进行交互
* SOA相关技术
  + 与SOA系统的基本构造元素——**Web服务**相关的一组技术。具体包括：Web服务建模、Web服务的描述、发布与发现技术、Web服务通信技术、Web服务交互技术、Web服务组合技术、Web服务安全和QoS保障技术、Web服务的事务管理技术等
  + **SOA工程**相关的一组技术。具体包括SOA方法学、领域建模、需求分析、服务编程与开发技术，以及基于服务的信息集成、应用集成等服务集成技术等
  + 与**分布式系统**共性问题相关的一组基础技术，包括SOA系统的体系结构、可伸缩性、可靠性及可用性保障技术等
* 标准化或业界协作组织
  + W3C(World Wide Web Consortium)万维网联盟
  + OASIS(Organization for Advancement of Structured Information Standards)结构化信息标准促进组织
  + WS-I(Web Services Interoperability)Web服务互操作组织
  + OMG(Object Management Group)对象管理组织
  + OSOA(Open Service Oriented Architecture)开放面向服务架构协作组织

###### Web服务

1. 背景
   * Web服务及其协议栈是对SOA理念的一种最初尝试。它将服务实现为一种可以**自包含、自描述以及模块化**的Web组件，并可以通过Web进行发布、查找和调用
   * Web服务：是由**URI标识**的软件应用，其接口和绑定可以用**XML**来定义和描述并且可以被发现。Web服务可以和其他软件代理（agent）之间通过**基于Internet的协议以XML消息交换**的方式直接交互 ——W3C
2. WEB服务技术的基本协议
   * SOAP((Simple Object Access Protocol)
     + 怎样调用并触发一个Web服务
   * **WSDL(Web Service Description Language)**
     + **如何描述一个Web服务**
   * UDDI(Universal Description Discovery and Integration)
     + 如何发布、管理及查找Web服务的描述信息

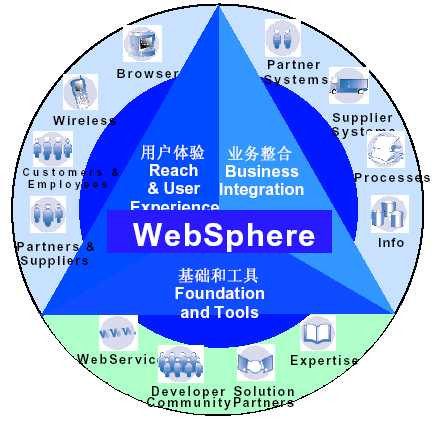


1. WEB服务的应用场景和局限

**Web Services在需要通过Web进行互操作或远程调用的情况下最适用**

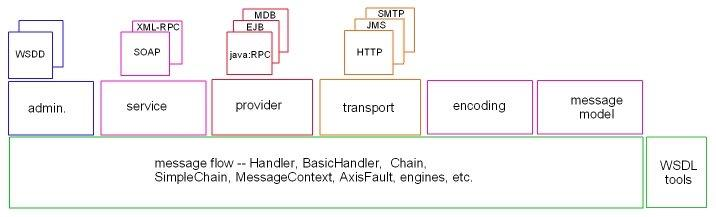
* + 适合的应用场合：跨防火墙通信，应用程序集成，B2B集成，软件和数据重用
  + 不适用的场合
    - 1. 单机应用程序（使用本地API更高效；不便于对本地系统资源进行操作
    - 2.局域网的同构应用程序（Web Services大多基于HTTP协议，在同构局域网环境下不如应用协议效率高（如DCOM,RMI,.NETRemoting等））

1. WEB服务开发过程
   * 开发生命周期
     + 构建：开发和测试服务实现、定义服务接口描述和定义Web Services实现描述
     + 部署：向注册中心发布服务接口和服务实现的定义，把Wen Services部署到执行环境中
     + 运行：调用Web Services
     + 管理：持续管理和经营Web Services应用程序
   * IBM的Web Services框架
     + 基于XML的Web Services协议栈，如SOAP/UDDI/WFML,及基于websphere应用服务器的软件整合



*IBM Web Service框架*

* + Apache的SOAP服务器：AXIS
    - 为Web服务应用程序的创建和发布提供了平台



### **XML**

###### 简介

1. 标记语言(置标语言)Markup Language：用一系列约定好的标记来对电子文档进行标记，以实现对电子文档的语义、结构、及格式的定义。这些标记必须很容易的和内容区分，并且易于识别
2. XML**(eXtensible Markup Language)**可扩展标记语言
3. •XML 是一种**标记语言**，可以标记任何一种事物，很类似 HTML

•XML 的设计宗旨是**传输数据**，而非显示数据。为数据交换和数据共享提供平台无关的数据格式，可以提供对数据的简单存储。

•XML同时也是一组规范，需遵守这一规范来开发自己的软件或信息•XML 被设计为具有**自我描述性**

•XML 是 **W3C 的推荐标准**（**W3C**，万维网联盟（World Wide WebConsortium）， **1998年2月正式推出**

###### 整体结构



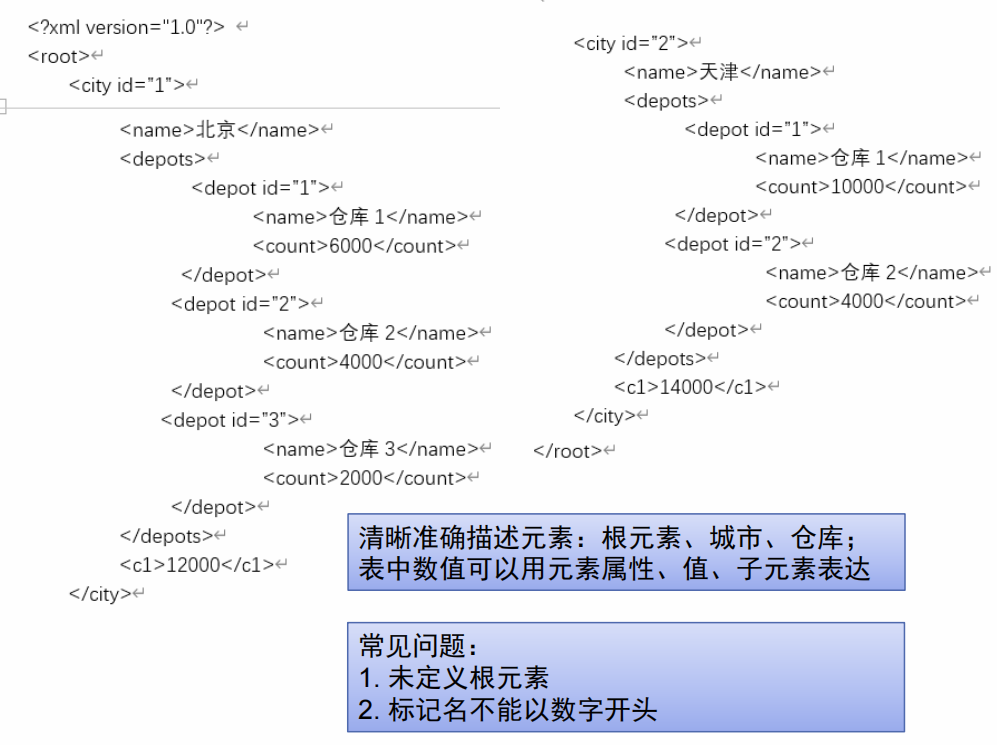
###### 有效的(valid)

1. Well-Formed XML
   1. 在XML中，“形式良好”(**well-formed**)有着明确的标准，即要**遵守XML1.0规范中的语法规则**
   2. 形式良好是一篇XML文档在语法上的最基本的要求，只有形式良好的XML文档才能被正确地分析和处理，形式良好的文档易于浏览器浏览，简化浏览器的设计
   * 确定且唯一的根元素
   * 元素的标记区分大小写，开始标记和结束标记匹配
   * 正确的元素标记嵌套
   * 为属性赋值时必须将属性值用引号括起来
   * 同一个元素属性不能重复
   1. 练习题



1. 文件的序文必须包含合适的文件形态宣告，该宣告中包括了文件模式定义，如DTD(Document Type Definition),XML Schema，负责定义文件结构
2. 文件其余部分必须符合模式定义中所定义的结构
3. 练习题





### **SOAP**

###### Intro:WSDL

###### **WSDL(Web Service Desciption Language)：是采用XML语言来描述Web Service的属性的语言**

###### What:Web Service做什么

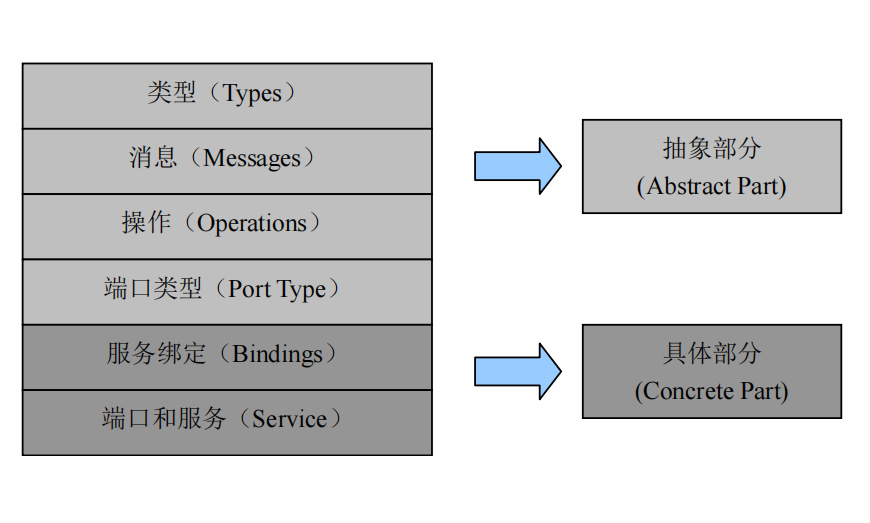
###### Where:Web Service位于哪里

###### How:怎么调用

###### 如果将Web Service作为一个分布式对象来看，WSDL就是Web Service的**接口描述**语言（IDL）

###### WSDL定义了一套基于XML的语法，将Web Service描述为能够进行消息交换的服务访问点的集合

###### WSDL结构



###### SOAP性质

###### SAOP是一种WS(Web Service)实现方式

###### **简单对象访问协议 SOAP（Simple Object Access Protocol）是在松散的、分布的环境中使用XML交换结构化的和类型化的信息的一种简单协议**

###### XML的标准化为跨平台之间的交流带来了极大的方便，而SOAP的实质就是把这种交流的方式标准化了

###### SOAP本身并不定义任何应用语义，如编程模型或特定语义实现，它只定义了一种简单的以模块化的方式包装数据的机制

###### 将数据/对象打包成XML格式的数据

###### Why SOAP

* SOAP是一个轻型的分布式计算协议，它允许在一个分布式的环境中交换信息
* SOAP用基于文本的XML协议与分布式系统通信，而不是用其他分布式计算协议（例如CORBA、RMI和DCOM）使用的二进制格式。这使得SOAP具有跨硬件平台、操作系统、编程语言和网络硬件平台的高度互操作性
* **SOAP可以在HTTP上传输**，HTTP允许它利用已有的基础设施投资，例如Web服务器、代理服务器和防火墙。SOAP也可以用其他的协议（如SMTP和JMS）进行传输
* **SOAP的调用效率比较低**
  + **HTTP不是有效率的通信协议**
  + **XML需要额外的文件解析**

### **REST服务**

###### 简介

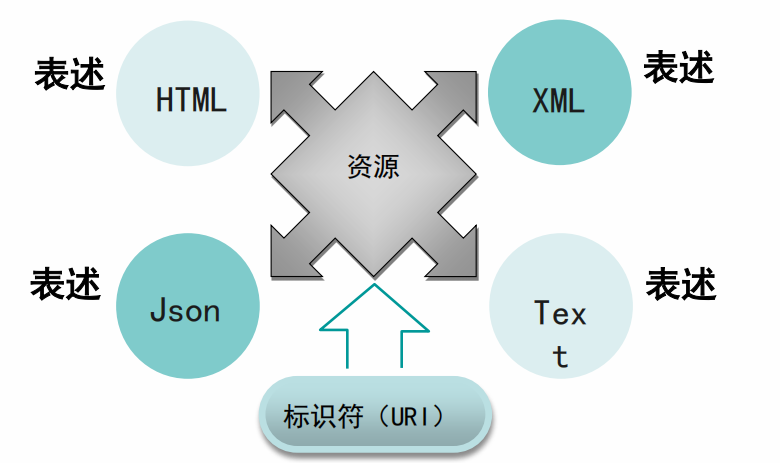
* **REST(Representational State Transfer) 表述性状态转移，一种架构风格**
* 不是一个具体的标准或者架构，只是一套设计原则和一种架构风格
  + REST 从**资源**的角度来观察整个网络
  + 分布在各处的资源由**URI** 确定
  + 客户端的应用通过URI来获取资源的**表述**
  + 获得这些表述致使这些应用程序转变了其状态。随着不断获取资源的表述，客户端应用不断地在转变着其状态，所谓**表述性的状态转移**

###### 什么是资源Resource

* A resource is anything that’s important enough to be referenced as a thing in itself [Rich+07] ；任何事物，只要有被引用的必要，就是一个资源(resource)。它可以是一个实物，也可以是一个抽象的概念。
* 通常一个资源是某个可以存放在计算机上并体现为比特流的事物。在Web中，可以这样认为——资源是URI标示的东西
* 分类
  + 文档、图片
  + 服务
  + 资源的集合
  + 不是网络上的对象也可以被看作资源(eg人)
* URI(Uniform Resources Indentifier)统一资源标志符号
  + 作用：是为互联网上所涉及到的所有资源提供一种统一的唯一标识
  + 组成：  
     ***<schme>:<hier-part>[?<query>][#<fragment>]***
    - <schme>是命名方案名称，例如http、mailto等。后面必须是一个冒号。
    - <hier-part>是一个表示层次结构的成分，例如域名加上存储路径和文件名。
    - 可选部分<query>表示某种查询字符串，这种字符串开头必须有一个问号。
    - 另一个可选部分<fragment>是片断标识，以符号＃开头。URL和URN都是URI的子集

###### 资源表述Representation

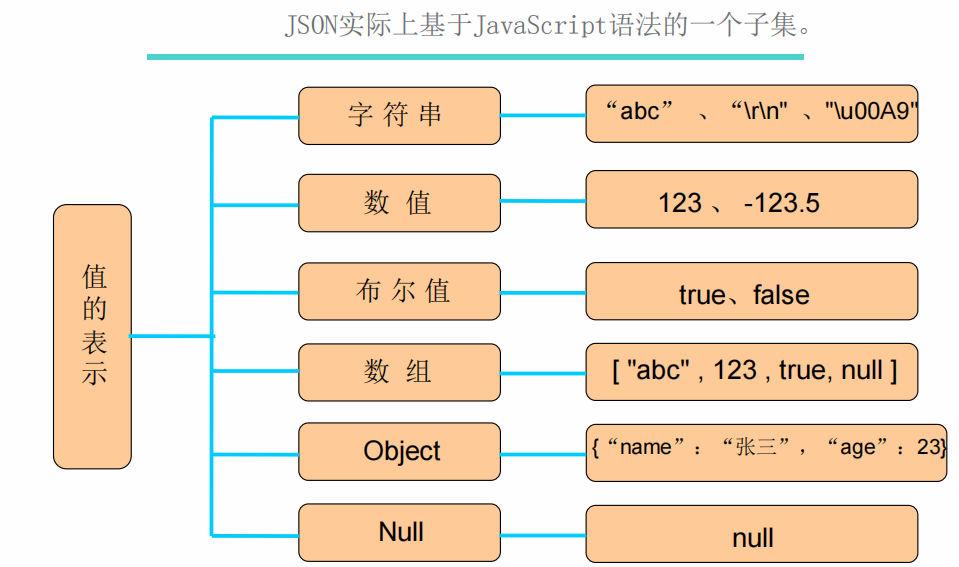
* **资源和表述不是一码事。Web上获取的不是资源，而是资源的表述**
* a representation is just some data about the current state of aresource [Rich+07]如：以特定语言特定文件格式表示的字节序列
* 对于给定的资源，可以有很多不同的表述



* + JSON
    - JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式
    - JSON 语法是 JavaScript 语法的子集
    - 由名称、值对构成
    - JSON 可以将 JavaScript 对象中表示的一组数据转换为字符串，然后就可以在函数之间轻松地传递这个字符串，或者在异步应用程序中将字符串从 Web 客户机传递给服务器端程序



* + - 基本语法



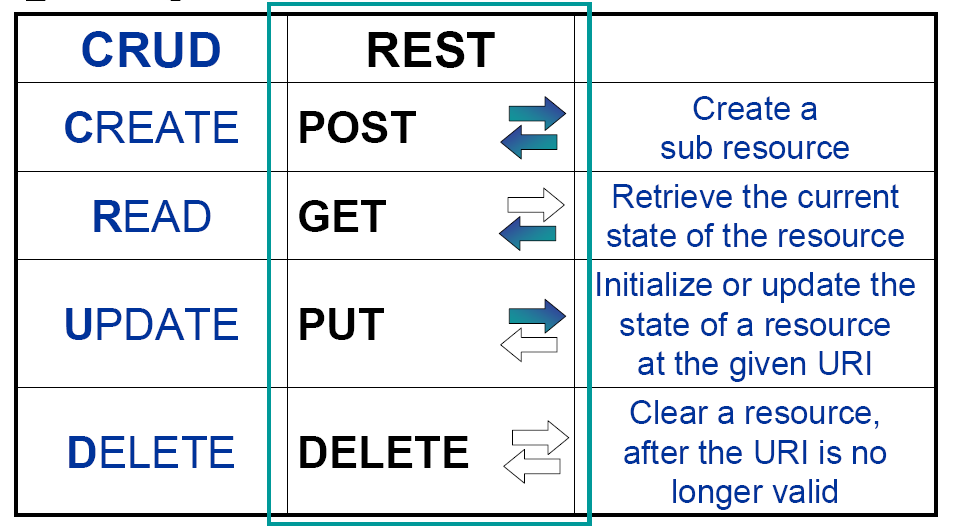
* + - * 优点
        + 相对于XML，它**更加易读、更便于肉眼检查**
        + 在语法的层面上，JSON中的分隔符限于引号、中括号、大括号、冒号和逗号,和**XML相比体积变小,有利于网络传输**
        + JavaScript中解析JSON是其原生支持的功能,所以在JS中**使用起来极为简单**,JSON配合AJAX使用可以很大的提高开发效率,而操作XML的DOM语法就不那么简洁了
      * 缺点
        + 在JSON中不支持命名空间的概念
        + JSON验证过程比一般的XML复杂 ,XML有DTD和schema的通用的验证方式，JSON Schema尚在发展之中 （http://json-schema.org/）

###### 超链:将资源链接到一起Hateoas

* 任何可能的情况下，使用链接指引可以被标识的资源
* 大多数表述是超媒体(hypermedia)的，它不仅包含数据，还包含指向其它资源的链接
* 使用URI而不是id？链接可以指向由不同应用、不同服务器甚至不同公司提供的资源——因为URI命名规范是全球标准，构成Web的所有资源都可以互联互通

###### 统一资源访问接口(基于HTTP)

* REST 要求开发人员显式地使用 HTTP 方法，并且使用方式与协议定义一致。 这个基本 REST 设计原则建立了创建、读取、更新和删除（create, read, update, and delete，CRUD）操作与 HTTP 方法之间的一对一映射。 根据此映射
* 若要在服务器上创建资源，应该使用 POST 方法。选择合适的动词！
* 若要检索某个资源，应该使用 GET 方法。
* 若要更改资源状态或对其进行更新，应该使用 PUT 方法。
* 若要删除某个资源，应该使用 DELETE 方法。



* 幂等性(IDEMPOTENCE)
  + HTTP方法的幂等性是指一次和多次请求某一个资源应该具有同样的副作用。
  + HTTP方法的幂等性
    - HTTP GET方法用于获取资源，不应有副作用，所以是幂等的
    - HTTP DELETE方法用于删除资源，有副作用，但它应该满足幂等性
    - 两次相同的POST请求会在服务器端创建两份资源，它们具有不同的URI，**POST方法不具备幂等性**
    - PUT所对应的URI是要创建或更新的资源本身，因此，PUT方法具有幂等性

###### 设计及开发REST服务

1. 分析应用需求中的数据集，映射数据集到Rest服务的资源。

2. 对于每一资源，命名它的 URI。

3. 为每一资源设计其 Representations

4. 用 hypermedia links 表述资源间的联系

5. 为每一资源设计对其的操作。

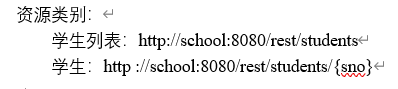
6. 实现Rest服务部署到Web服务器上

7. 使用浏览器进行测试

###### **练习：学生信息管理系统**

题目：负责学生信息的创建，修改，删除，查询。学生信息主要包括：学号,姓名,性别,班级,电话,EMAIL

1. 列举所涉及的相关资源，并为每种资源定义URI来对其进行唯一标识

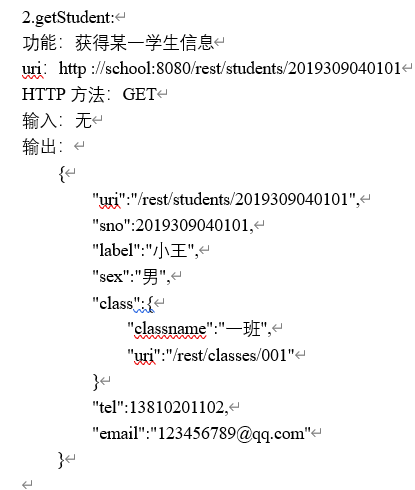


1. 为每种资源设计其表述形式，并利用超链建立各类资源之间的关联关系

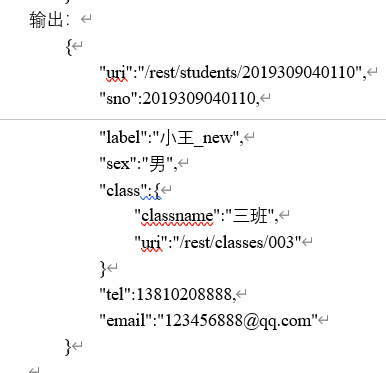
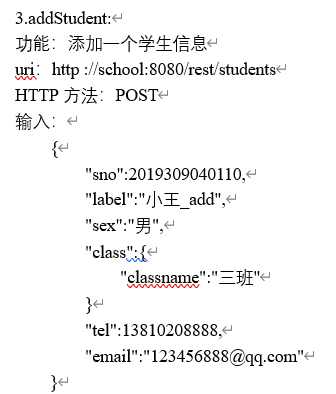


1. 设计REST服务接口，实现对资源的增、删、查、改功能。仅需给出服务的名称、HTTP方法、URI、输入参数、输出参数即可，无需给出服务的内部实现代码

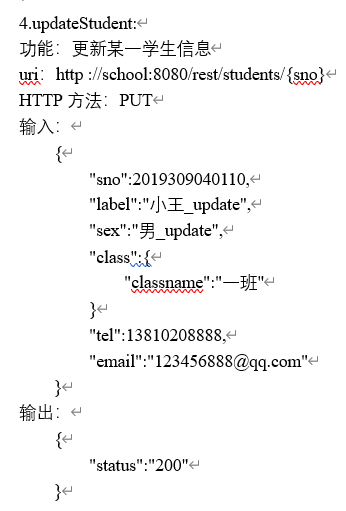
获取学生信息



新建学生信息



修改学生信息



删除学生信息

