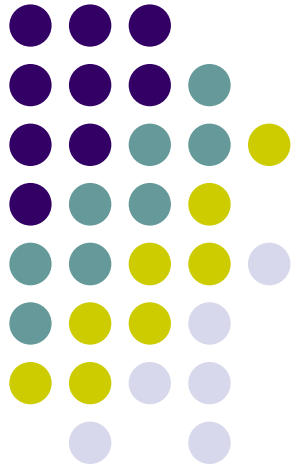
第十章　Web 服务介绍



主要内容

- 面向服务的计算
- Web服务基本概念
- Web服务体系结构
- Web服务协议
- Web服务开发
- Web服务软件产品介绍

面向服务的计算

- 面向服务的计算是一个新的计算规范
  - 服务作为构件→复合应用
  - 支持分布式应用的低成本快速开发
- 服务是自包含的模块
  - 能够在网络上使用基于XML的技术进行描述、定位、编配和编程，体现出“面向服务”的编程方式。
  - 服务的构建方式通常独立于它们的使用方式，服务提供者和服务用户方之间是松耦合的关系。
- 面向服务的计算并不是一个新的技术
  - 分布式系统、软件工程、信息系统、计算机语言
  - 基于Web的计算、XML技术

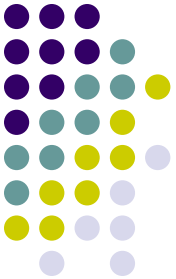
面向服务的计算

- 面向服务的计算模式中，服务需要技术中立、松耦合以及支持位置透明性。
  - 技术中立-广泛地遵循一些公认标准
    - 必须使用要求最低的标准化技术调用服务
    - 所采用的调用技术需要得到绝大多数信息技术环境支持
  - 松耦合
    - 无需了解客户端和服务端的信息
    - 无需了解客户端和服务端的内部结构或内部协议
  - 支持位置透明性
    - 无需考虑服务的具体位置，即可定位以及调用这些服务

面向服务的计算

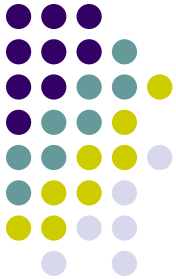
- 当服务使用因特网（Internet）作为通信手段以及使用基于因特网的标准时，即为Web服务
  - 交互使用了公开的、不安全的、低保真度的机制
- Web服务是一个可通过网络使用的自描述、自包含软件模块，这些软件模块可完成任务、解决问题或代表用户、应用程序处理事务
- Web服务建立了一个分布式计算的基础架构。这个基础架构由许多不同的、相互之间进行交互的应用模块组成。这些应用模块通过专用网络或公共网络进行通信，并形成一个虚拟的逻辑系统

面向服务的计算



- Web服务可以是：
  - 自包含的业务任务，如提款或取款服务
  - 成熟的业务流程，如办公用品的自动采购
  - 应用程序，如人寿保险应用程序、需求预测与库存补给应用程序
  - 已启用服务的资源，如访问特定的报春病人病例的后台数据库
- Web服务功能千差万别，既可以是简单请求，如信用卡核对与授权、价格查询、库存状态检查或天气预报，也可以是需要访问和综合多个数据源信息的完整业务应用程序，如保险经纪人系统、保险责任计算，旅行自动规划或者包裹跟踪系统等。

面向服务的计算



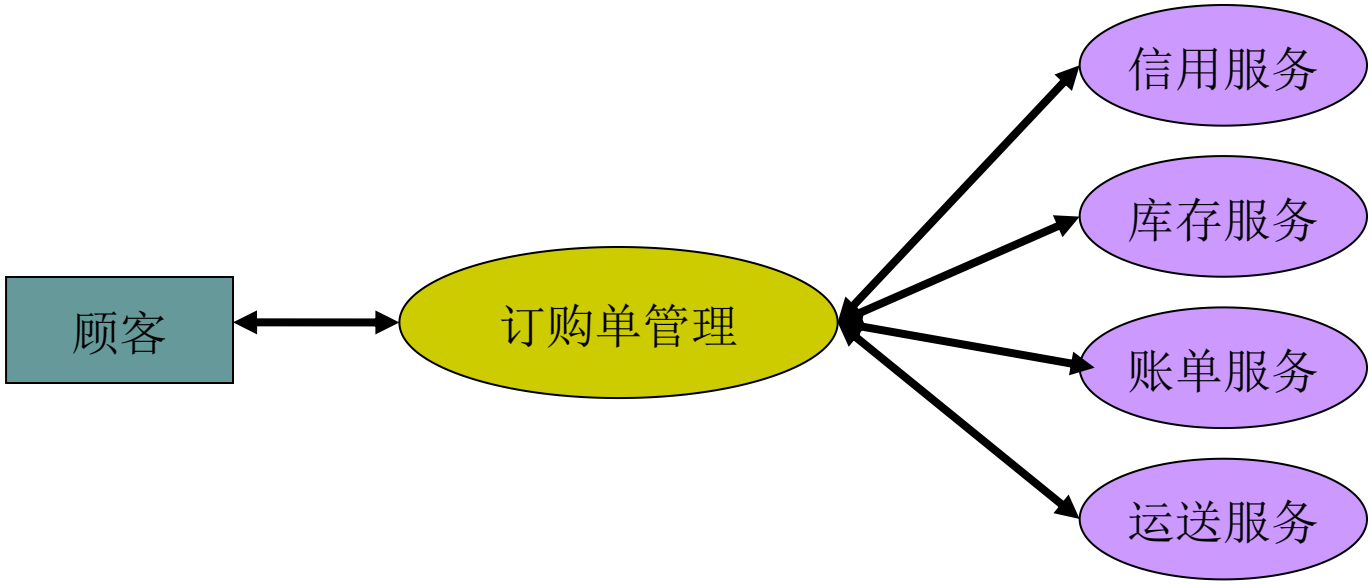
- 远期目标
  - 实现分布式应用，按照不断变化的业务需求动态组配应用程序
  - 根据设备（个人电脑、工作站、便携式计算机、WAP手机、PDA）、网络（有线电视网、移动通信系统、各种数字用户线路、蓝牙）和用户访问的情况定制具体的分布式应用，保证所需之处都可广泛利用任何业务逻辑的具体片段。
  - 一旦部署了一个具体的Web Service，其他的应用和Web Service就能发现和调用这个Web Service。

面向服务的计算

- **案例研究——订单管理流程**
  - **在一个简单的订单管理场景中，需要管理顾客向特定供应商提交的订购单。订单管理解决方案支持端到端的订单处理流程。这个处理流程可以借助复杂的相互交互的Web Service集合来表示，并且这些Web Service之间需要许多同步与协调。**
  - **这些Web Service可配置和订购个性化产品，并向顾客提供供货状况的精确的实时信息，以及一共一些交互的价格选项和实时状态查询，此外还可执行库存和仓库管理等。**

面向服务的计算

- 例如：顾客或采购组织创建一个订购单，并将请求发送给供应商。供应商提供一个订购单Web Service，它可以接受订购单，并给基于一些判断准则，如货物的供货情况及顾客信用情况等，接受或拒绝客户请求。

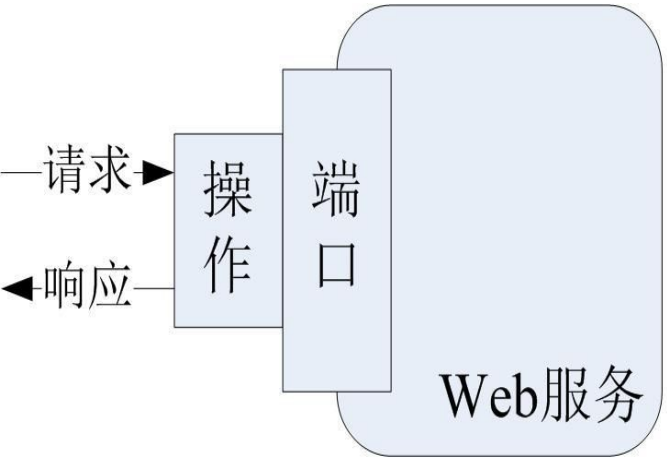


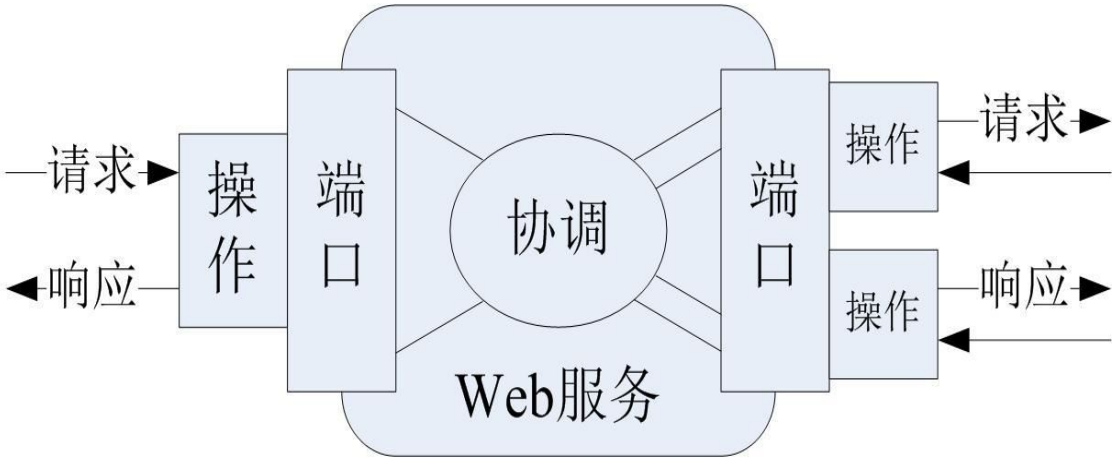
Web服务的定义

- Web服务是一个平台独立的、松耦合的、自包含的、基于可编程的Web的应用程序，可使用开放的XML标准描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序
  - 松耦合：Web服务协议、接口和注册服务可以使用松耦合的方式协同工作
  - Web服务诧义封装各个独立的功能：是一个完成单个任务的自包含的软件模块
  - 编程式访问Web服务：可将Web服务嵌入到进程的应用中
  - 可动态发现Web服务并将其添加到应用中
  - 可使用标准的描述语言来描述Web服务：Web服务描述语言WSDL
  - 可在整个因特网上发布Web服务：使用通用的因特网协议

Web服务的定义

- 按照拓扑结构分类
  - 信息型：经支持简单的请求/响应操作
  - 复合型：在进入操作和离开操作之间进行了一定形式的协调。





Web服务的定义

- 信息型服务：比较简单，可对一些内容进行访问，最终用户通过请求/响应序列与这些内容进行交互
  - 编程式服务：Web服务暴露应用程序的业务功能给其他应用程序
  - 按照所解决的业务类型不同分类
    - 纯内容服务：访问天气预报信息
    - 简单的交易服务
    - 信息联合服务：增值信息Web服务
  - 标准支持：通信协议、服务描述、服务发布和发现

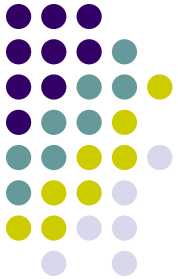
Web服务的定义

- 当企业需要将几个服务组合在一起创建一个业务流程，诸如定制订单、客户支持、采贩和物流支持等，企业则需要使用复合的Web服务
  - 按照组成简单服务的方式分类
    - 构成编程式Web服务，例如：库存检查服务
  - 构成交亏式Web服务
  - 复合服务的功能是粗粒度的，并且复合服务是有状态的

Web服务的定义

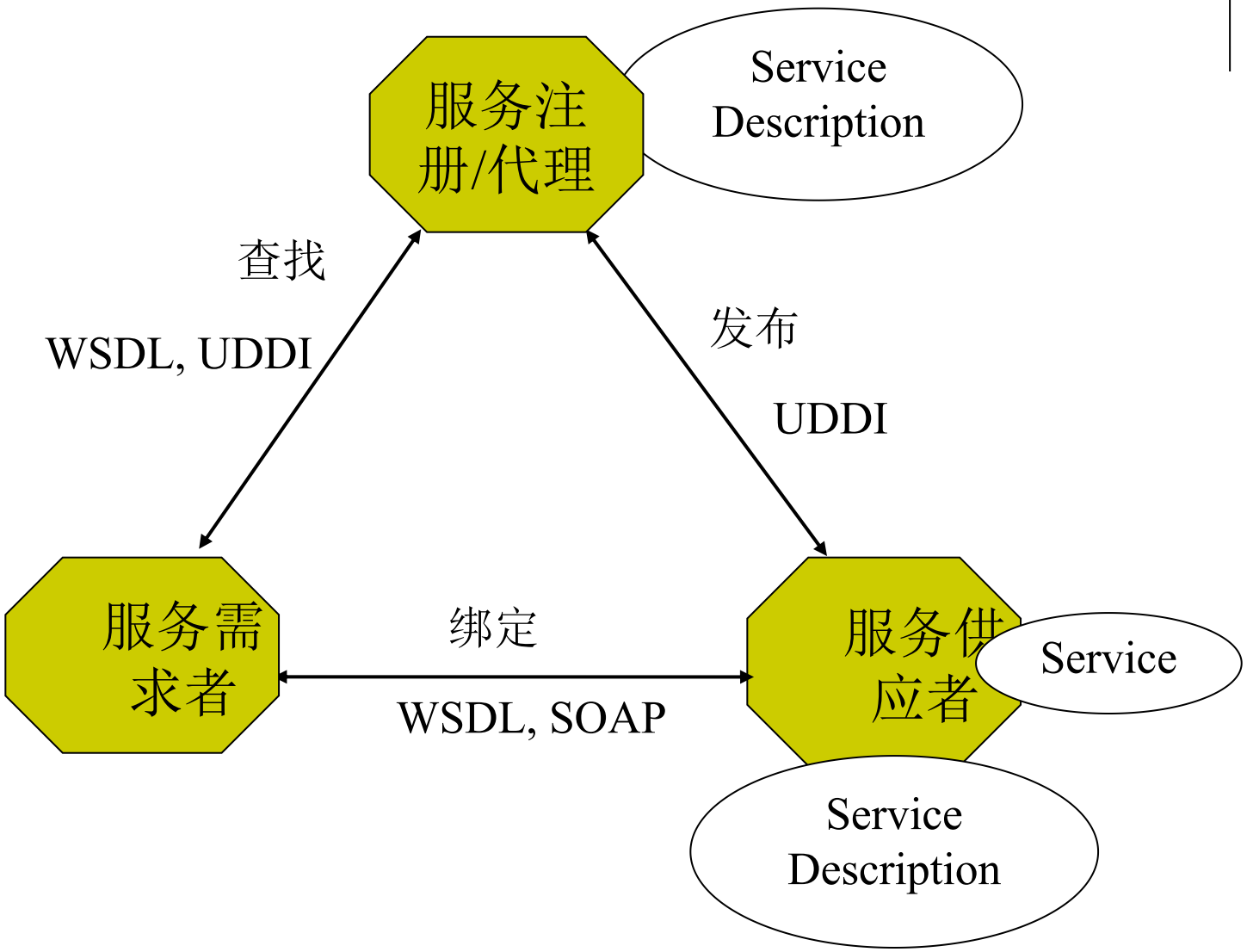
- 功能性描述：详述了操作特性
  - 操作特性定义了服务的整个行为
  - 主要关于消息的语法规则、以及如何配置发送消息的网络协议
- 非功能性描述
  - 主要关于服务质量属性，例如：服务计量和代价、性能度量
  - 主要关于服务请求者的运行环境

Web服务的定义

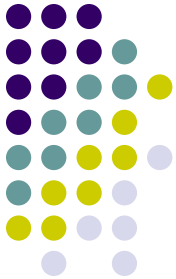


- Web服务既可以是无状态的、也可以是有状态的
  - 无状态：服务可以被重复调用，且无须维持上下文或状态
    - 信息型天气预报服务
  - 有状态：需要维持不同操作调用之间的状态
    - 无论这些操作调用是由Web服务的同一个客户端发出，还是由不同的客户端发出
    - 订单管理应用

Web服务模型


服务注册/代理
Service Description
查找
WSDL, UDDI
发布
UDDI
服务需求者
绑定
WSDL, SOAP
服务供应者
Service
Service Description

Web服务模型——组件

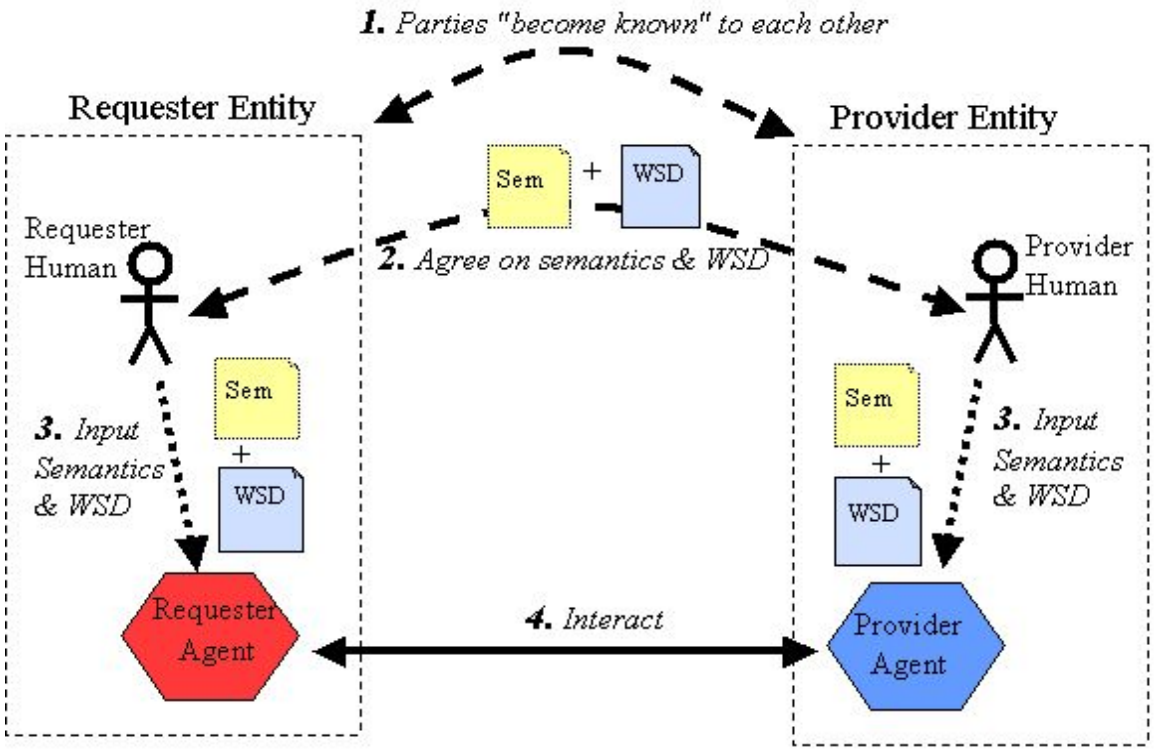


- 三种角色
  - 服务提供者：提供服务及维护注册表以使服务可用；
  - 服务代理：
    - 服务提供者与服务请求者的中介；
    - 传统的代理是UDDI注册中心；
  - 服务请求者：发现 Web 服务，然后调用这些服务以创建应用程序
- 服务
  - 应用程序，通过服务描述语言进行描述，其描述信息通过代理发布

Web服务体系结构 - 操作

- 发布 / 撤除发布（Publish）
  - **发布和撤除发布是指将服务发布至代理处（发布）或除去它们的一些项（撤除发布）。服务提供者通过代理来发布或不发布某个服务。**
- 查找 （Find），
  - **查找操作由服务请求者和服务代理共同完成。服务请求者描述他们正在寻找的服务类型，而服务代理发布与请求最匹配的结果。**
- 绑定 （Bind）
  - **绑定操作发生在服务请求者和服务提供者之间。双方经过适当的商讨之后，请求者就可以访问和调用提供者所提供的服务。**

Web服务实现过程



⇗整个实现过程基本上分为四步

⇗在这个过程中代理发挥着重要的作用

Web Services Architecture

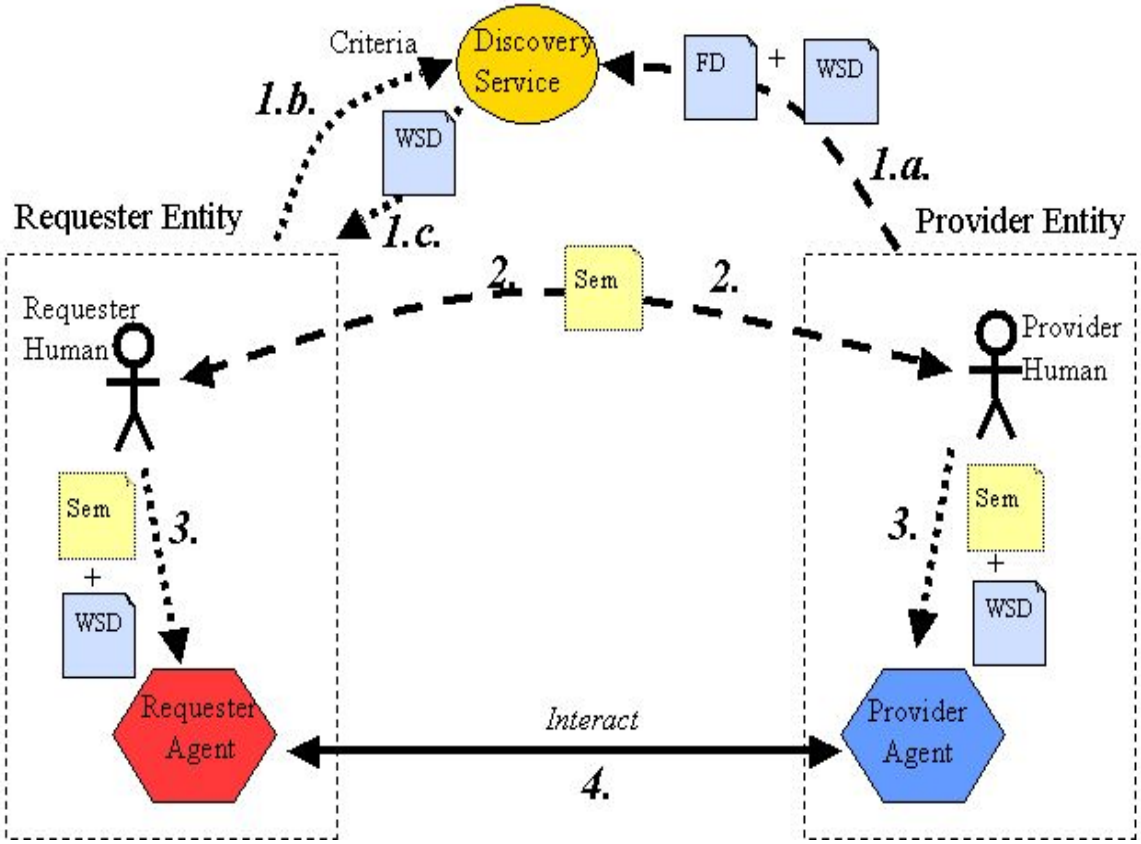
W3C Working Group Note 11 February 2004

Web服务实现过程

- the requester and provider entities become known to each other (or at least one becomes know to the other)
- the requester and provider entities somehow agree on the service description and semantics that will govern the interaction between the requester and provider agents
- the service description and semantics are realized by the requester and provider agents
- the requester and provider agents exchange messages, thus performing some task on behalf of the requester and provider entities. (I.e., the exchange of messages with the provider agent represents the concrete manifestation of interacting with the provider entity's Web service.)

“Known each other”

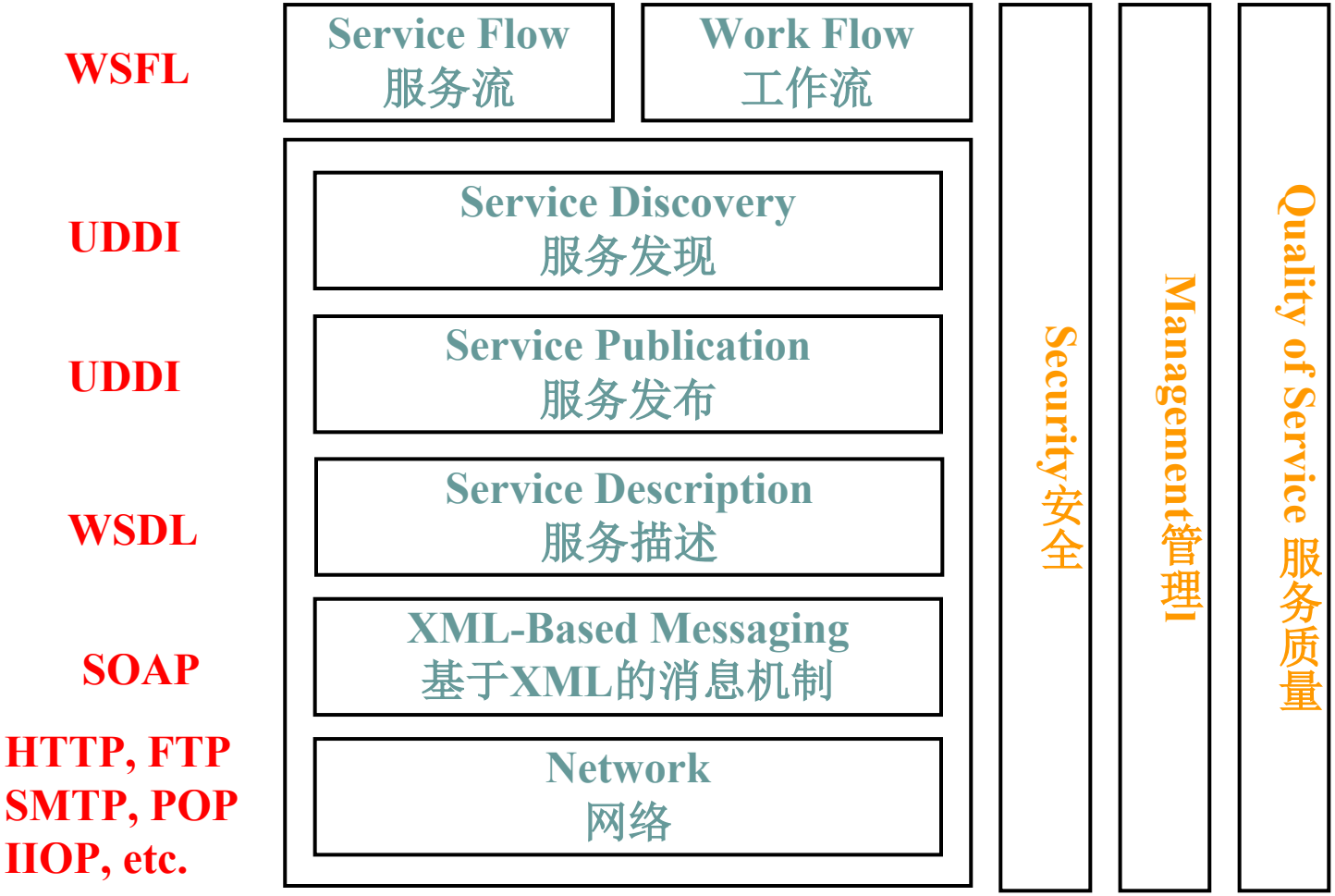
- 发现服务，将发现本身作为一个服务
- 发现服务获取WSD及FD
- 请求者实体向发现服务提供选择WEB服务的信息
- 发现服务返回符合标准的WSD



Web Services Architecture

W3C Working Group Note 11 February 2004

Web服务的体系结构


WSFL
Service Flow
服务流
Work Flow
工作流
UDDI
Service Discovery
服务发现
UDDI
Service Publication
服务发布
WSDL
Service Description
服务描述
SOAP
XML-Based Messaging
基于XML的消息机制
HTTP, FTP
SMTP, POP
IIOP, etc.
Network
网络
Security安全
Management管理
Quality of Service 服务质量

Web服务的体系结构

- **(1)网络层：** Web服务必须是网络可访问的，才能被服务请求者调用。Web服务使用通用Internet协议HTTP、SMTP、POP 和FTP等。
- **(2)XML消息层：** 使用XML作为消息通讯协议的基础。简单对象访问协议(Simple Object Access Protocol, SOAP)定义了信息交换的轻量协议，其中一部分定义了如何用XML表达数据的一套规则，另一部分定义了扩展的消息格式、表达远程过程调用(RPC)的规则和与HTTP协议的绑定机制。
- **(3)服务描述层：** 实际上是一个描述文档栈。Web服务描述语言(WSDL)是Microsoft和IBM共同开发的基于XML的合约语言，用以将Web服务接收和产生的消息(即合约)形成文档。WSDL是基于XML的服务描述的事实标准，是支持Web服务互操作的最小化服务描述。

Web服务的体系结构

(4)服务发布层： 服务发布是服务提供者让服务请求者在其生命周期的任何阶段都能够访问WSDL文档的操作。最简单的静态的例子是，服务提供者直接向服务请求者发送WSDL文档，即直接发布方式。此外，通用描述、发现与集成规范(UDDI)定义了一套机制支持服务提供者发布其开发的Web服务以及服务消费者查找其感兴趣的Web服务，服务提供者还可以将WSDL文档发布到本地WSDL注册库、私有UDDI 注册库或UDDI操作节点。

(5)服务发现层： 服务发现依赖于服务发布，不同的服务发现机制对应于不同的服务发布机制。服务发现是让服务请求者访问某项服务描述并使应用程序在运行时能够使用该项服务。最简单例子是静态发现，即服务请求者从本地文件中获得服务描述。在设计或运行阶段，可以通过本地WSDL注册库、私有UDDI注册库或UDDI操作节点来实现服务发现。

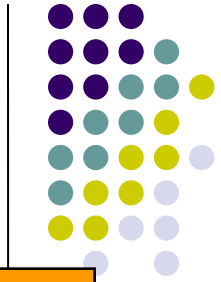
Web服务的体系结构

(6)服务组合层: Web服务的实现是一种软件模块，可以通过Web服务组合来生成新的Web服务。Web服务组合有几种表现形式：企业内的Web服务可以协同作业以向外部提供一个单一接口，或者不同企业的Web服务可以协同作业以实现机器到机器、商业到商业的流程。工作流管理器在业务流程中可以调用每一项Web服务。协议栈的最上层—服务流，描述了服务到服务的通讯、协同和流程的实现。Web服务流语言(WSFL)用来描述这些交互。

(7)配套基础设施: 为了使基于Web服务的应用能够满足电子商务、电子政务等应用需求，还必须提供企业级乃至城市级的基础设施，包括安全、管理、服务质量等。

在Web服务栈中，越是底层的技术，越成熟和标准化，而高层的内容和配套的基础设施还需要进一步开发和标准化。

Web Services协议栈



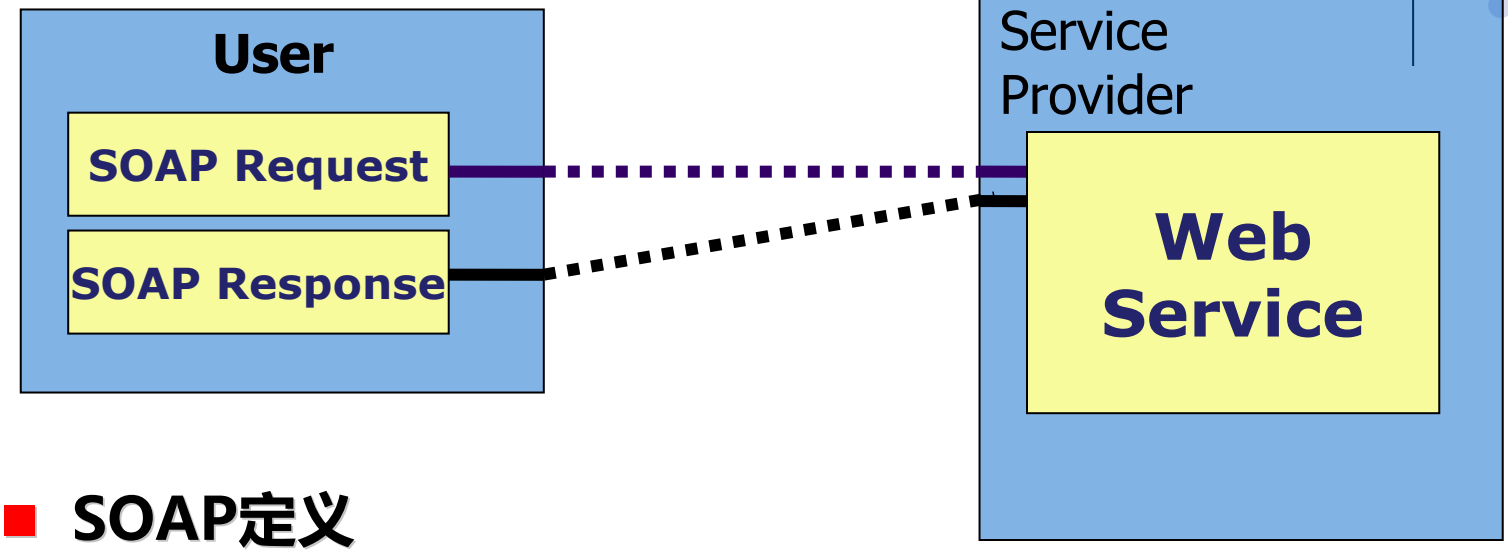
Publish, Find, Use Services:	UDDI
Formal Service Descriptions:	WSDL
Service Interactions:	SOAP
Universal Data Format:	XML
Ubiquitous Communications:	Internet

Simple, Open, Broad Industry Support

Web Services协议

- 交换信息（SOAP）
- 服务描述（WSDL）
- 服务发布和发现（UDDI）

交换消息

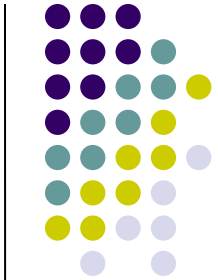


- SOAP定义
- SOAP结构
- SOAP示例
- SOAP优势和不足

Simple Object Access Protocol（SOAP）

- 定义
  - 简单对象访问协议（SOAP）是基于XML实现了一种消息格式以交换请求和使用，使用XML作为SOAP消息的基础使得任何实现基本的INTERNET通信服务的系统都能处理和传送这类消息 。
- 解释
  - SOAP中的“简单”一词是相对术语，在这里，相对于其他用于此目的的协议来说，它是简单的，包括DCOM和CORBA，他们也能实现软件之间的通信，但是也很不友好。
  - 术语“协议”表示双方认可的一个标准，即如何格式化消息以便双方能够通信。SOAP协议只是定义了一种消息格式，它并没有为交换消息而强加某种特定的传送协议，因此可采用HTTP，FTP, SMTP等协议来传送消息。

SOAP结构（1）



- SOAP封装(envelop)，封装定义了一个描述消息中的内容是什么，是谁发送的，谁应当接受并处理它以及如何处理它们的框架；
- SOAP编码规则（encoding rules），用于表示应用程序需要使用的数据类型的实例;
- SOAP RPC表示(RPC representation)，表示远程过程调用和应答的协定;
- SOAP绑定（binding），使用底层协议交换信息。

SOAP结构（2）

- SOAP封装
  - SOAP Header
    - 典型的扩展例子可以是实现一些诸如认证、事务管理以及支付的Header条目。
    - SOAP actor属性: SOAP actor属性的值是一个URI，指明下一个进行消息处理的SOAP中介。若省略SOAP actor属性，则表明接收者是SOAP消息的最终接收者。
  - SOAP Body
    - 这个调用负责指定要执行的方法名以及所有传递给方法的参数。在WEB服务者接受、翻译并处理完了这个方法调用之后，它就会发送一个响应或错误消息。
  - SOAP Fault
    - SOAP Fault元素是用于在SOAP消息中传输错误或状态信息。如果SOAP消息需要包含SOAP Fault元素的话，它必须作为一个Body条目出现，同时在Body元素内它必须不出现多于一次(至多出现一次)。

SOAP结构（2）

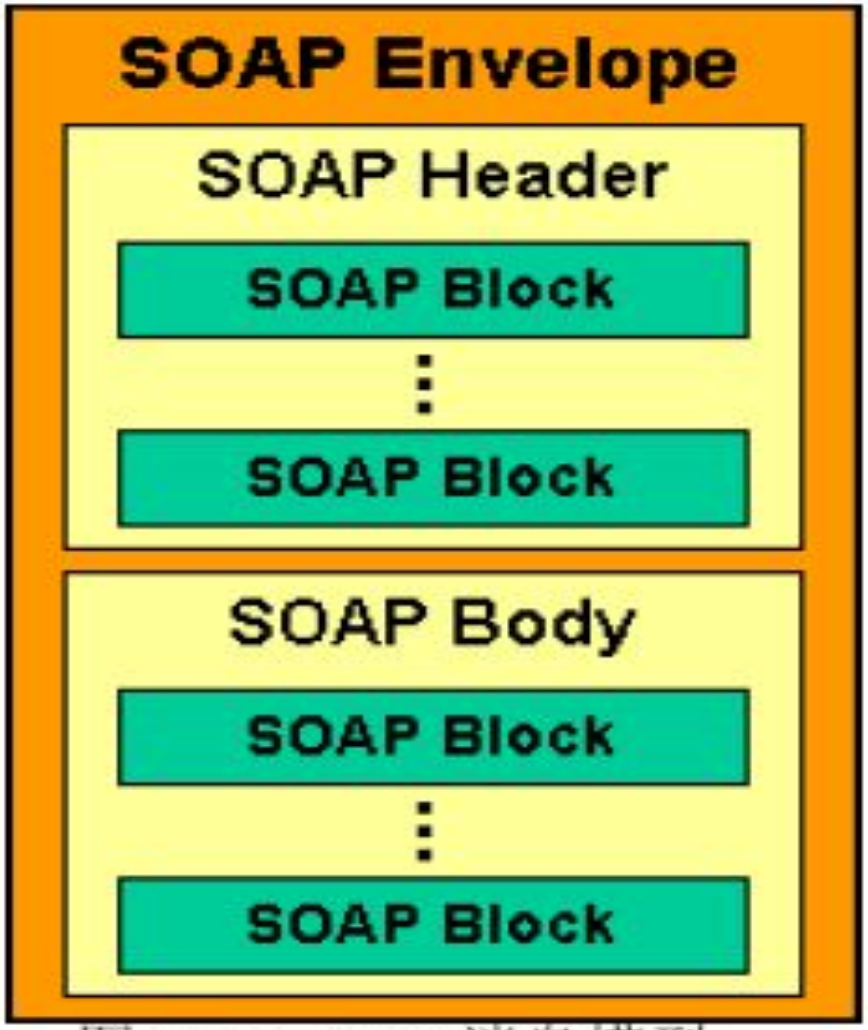
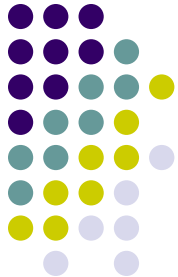


图 2-10 SOAP 消息模型

SOAP结构（3）

- SOAP编码规则(encoding rules)
  - 定义了一个数据的编码机制，通过这样一个编码机制来定义应用程序中需要使用的数据类型，它遵循XML模式规范的结构和数据类型定义，其中包括简单类型（整数，字符串等）和一些复杂类型（structure, Array等）。
- SOAP RPC表示(RPC representation)
  - 定义了一个用于表示远程过程调用和响应的约定，例如如何传输过程调用，在具体传输协议的哪个部分传输过程响应，因为我们可以在HTTP的响应的时候传递过程响应。

SOAP结构（4）



- SOAP绑定
  - 将SOAP绑定在HTTP上可以利用HTTP丰富的特性集，更恰当的描述应当是SOAP的语义通过HTTP的映射而很自然地成为HTTP的语义。同时，SOAP很自然的利用HTTP的请求／响应模型。
  - SOAP Action HTTP请求：可以用于指示SOAP HTTP请求的目的，它的值是一个标识该目的的URI。SOAP对于格式上并没有严格的限制，同时对URI的描述以及是否要是可解析的都没有严格的限制。当发出SOAP HTTP请求时，HTTP客户必须使用该头字段。

SOAP结构（4）

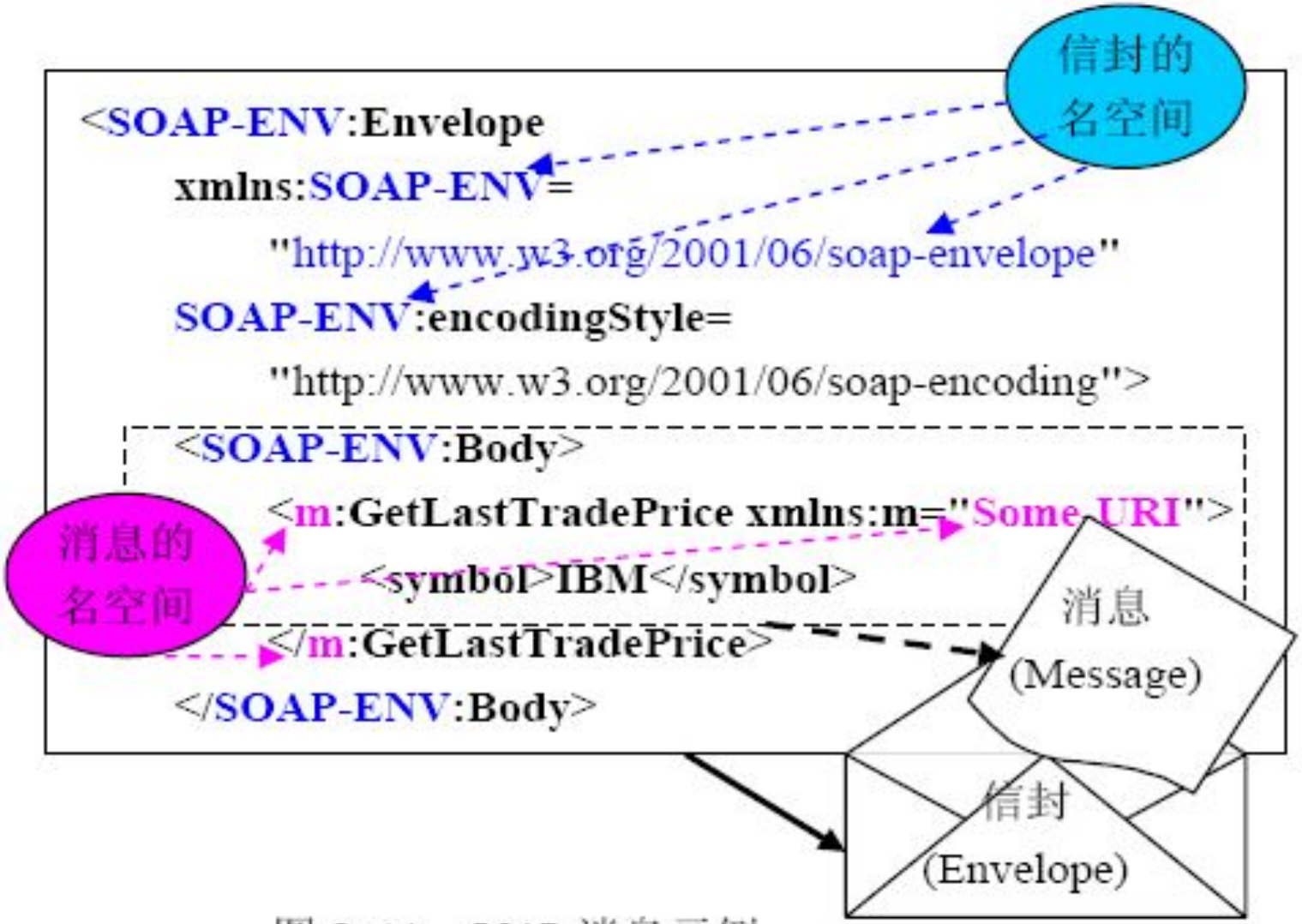


图 2-11　SOAP 消息示例

SOAP示例（1）

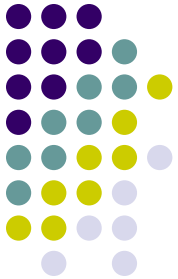
- 为实施一个方法调用，需要以下信息
  - 目标SOAP结点的URI
  - 方法名
  - 可选的方法或过程的特征
  - 方法或过程的参数
  - 可选的头数据
- 示例：股票价格查询
  - 请求消息：请求查询DIS的最新股票价格
  - 应答消息：返回查询结果34.5

SOAP示例（2）

- 请求消息：

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction:
 <env:Envelope xmlns:env="http://...  ..." >
    <env:Body>
      <m:GetLastTradePrice  env:encodingStyle= "..." xmlns:m= "...">
          <symbol>DIS</symbol>
      </m:GetLastTradePrice>
    </env:Body>
 </env:Envelope>
```

SOAP示例（3）



- 应答消息：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8 “
Content-Length: nnnn
<env:Envelope  xmlns:env="http://… …" >
   <env:Body>
      <m:GetLastTradePriceResponse env:encodingStyle=" … “
xmlns:m="http://example.org/2001/06/quotes">
            <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
   </env:Body>
</env:Envelope>
```

SOAP简单的理解，就是这样的一个开放协议SOAP=RPC+HTTP+XML：采用HTTP作为底层通讯协议；RPC作为一致性的调用途径，XML作为数据传送的格式，允许服务提供者和服务客户经过防火墙在INTERNET进行通讯交互。

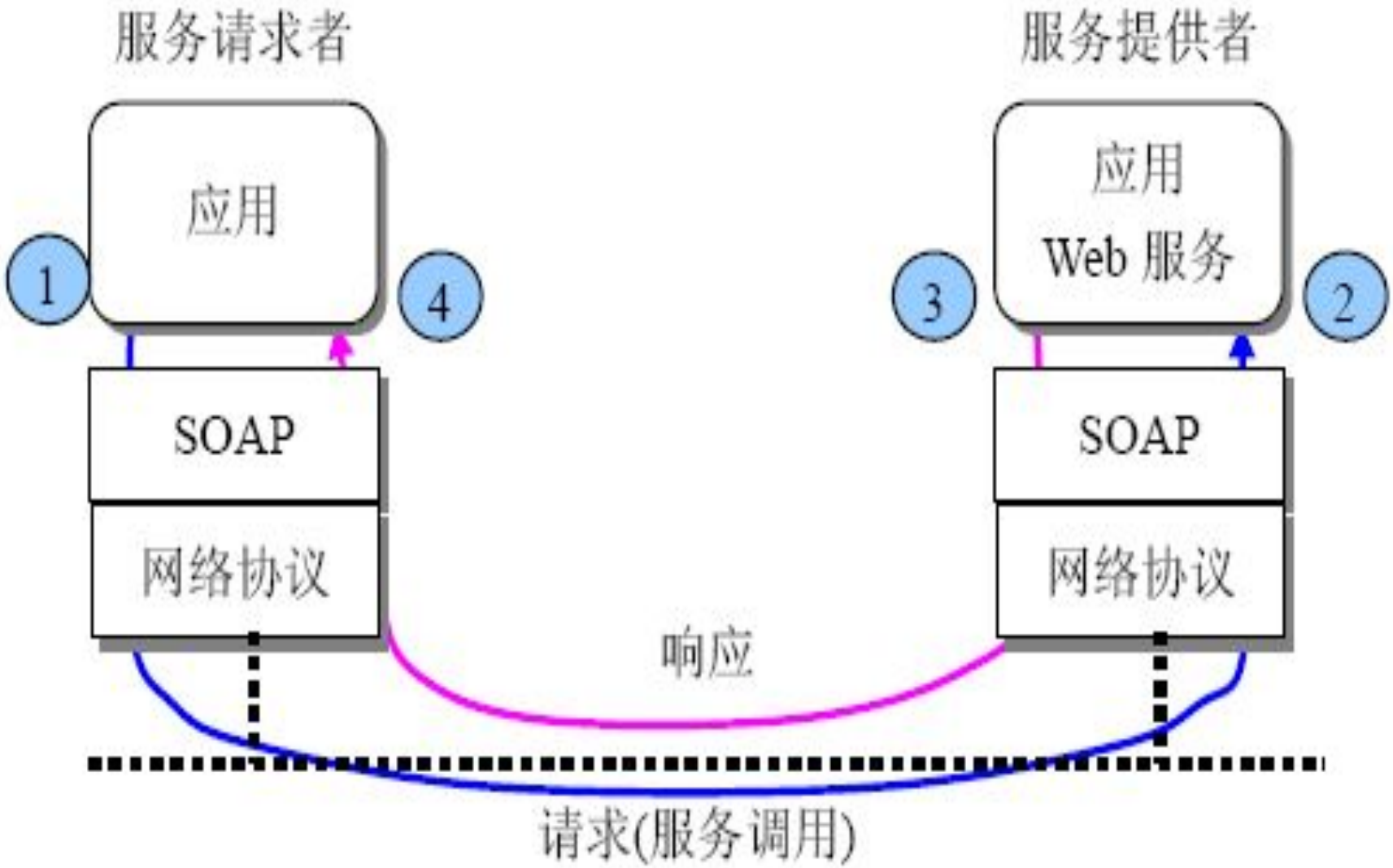
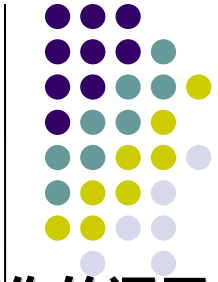


图 2-12　基于 XML 消息机制的分布式计算

基于XML 消息机制的分布式计算



1) 服务请求者的应用创建一个SOAP消息，即对服务提供者的Web服务操作的调用请求，并与服务提供者的网络地址打包后提供给SOAP基础设施，如SOAP客户端运行时进程。SOAP运行时进程与底层的网络协议交互以将SOAP消息发送到网络中。

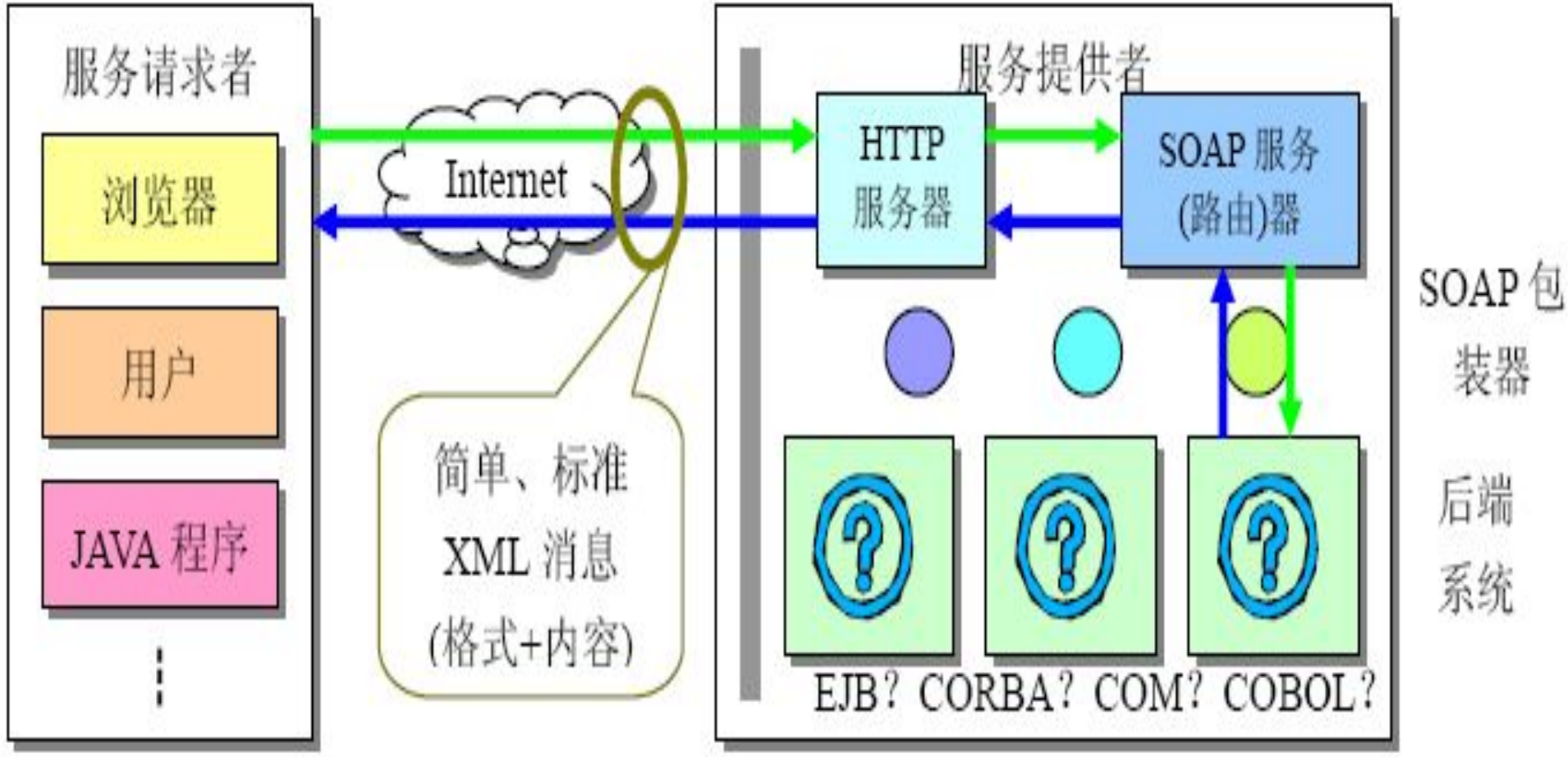
2) 网络基础设施将消息传送到服务提供者的SOAP运行时进程，如SOAP服务器。SOAP服务器按路由将请求送到服务提供者的Web服务

3) Web服务负责处理请求消息并形成一个SOAP响应消息，以服务请求者为目的地提交给SOAP运行时进程。SOAP运行时进程将SOAP消息通过网络送回服务请求者。

4) 服务请求者节点的网络基础设施接收响应消息。该消息通过SOAP基础设施，必要时可转化为某种编程语言实现的对象中的XML消息，然后提交给应用。

SOAP 松耦合

- SOAP 为XML 数据交换提供了任何平台、采用任何编程语言、使用任何网络协议、以及基于任何对象模型之上的中间件或应用系统都支持的标准机制。SOAP 实现了松耦合的软件开发，为应用服务集成提供了有效的解决方案。

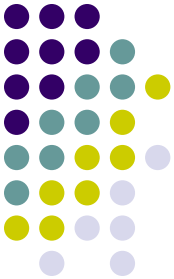


SOAP 松耦合

总之，选择SOAP作为XML消息通讯协议的原因主要有：

1. 它是基于文档消息通讯和XML远程过程调用(XML-RPC)的标准化封装机制，实现了应用服务的互操作机制；
2. 简单化，基本上是以XML信封作为消息体的HTTP POST操作；
3. 由于定义了使用SOAP消息头和操作或功能的标准编码方式实现消息扩展的标准机制而超越了XML简单的HTTP POST操作；
4. SOAP消息支持Web服务体系结构中的发布、查找和绑

SOAP的优势



- SOAP是平台独立的：SOAP是普通的XML，可以运行任何平台。
- SOAP消息描述了消息负载的每个数据元素，这样就可以很容易的诊断可能出现的问题 。
- 自由的传输绑定 (不仅仅是HTTP)；自由的语言绑定 (比如Java, C#)。
- SOAP降低了安全风险：SOAP只是XML，因此它能够使用端口80，而该端口通常配置HTTP，因此SOAP消息无需开放其他任何端口，就能通过防火墙的优点，从而降低了潜在的安全风险。

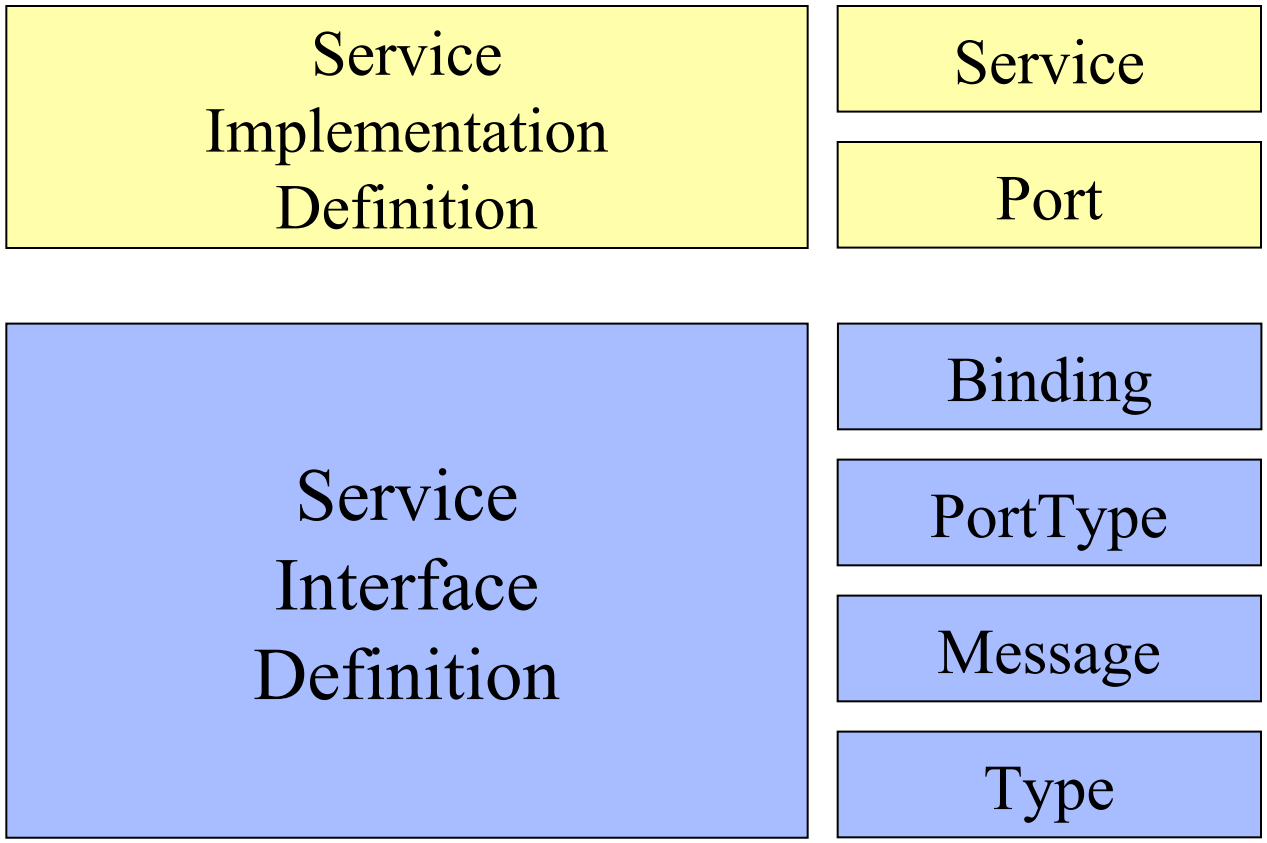
Web Services协议

- 交换信息（SOAP）
- 服务描述（WSDL）
- 服务发布和发现（UDDI）

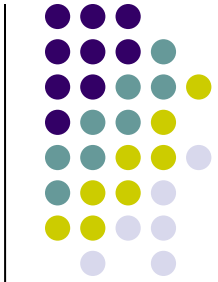
服务描述

XML是一种编码数据的标准方法。SOAP 基于XML定义了一种消息格式以便交换方法请求和响应，并最终完成WEB服务的调用，而WSDL则是用于描述如何使用SOAP来调用WEB服务的。

Web Service Description Language



Web Service Description Language



Web服务描述语言(WSDL)定义了一套XML语法，将Web服务描述为能够交换消息的通讯端点的集合，实现以结构化方式来描述通讯。一个WSDL文档将服务定义为网络端点(Endpoint)或端口(Port)的集合。在WSDL中，端点和消息的抽象定义与其具体的网络部署或数据格式绑定相分离，从而使得消息和端口类型(Port type)的抽象定义可以重用。消息是被交换的数据的抽象描述，端口类型是操作(Operation)的抽象集合。一个特定的端口类型的具体协议和数据格式规格组成了一项可重用的绑定。一个端口由相关的网络地址通过一项可重用的绑定来定义，而一项服务定义为一个端口集合。

Web Service Description Language

- **服务接口定义**
  - **服务接口是一个抽象的、可重用的服务定义，该服务可采用不同的参考实现**
  - **服务接口可能通过工业标准或组织定义，如：RosettaNet（一个电子商务标准）和保健业的HL7（医疗信息系统间电子资料的标准）**
- **服务实现定义**
  - **服务实现说明如何通过Provider来执行Interface，是Web Service描述的核心部分。同时，实现部分包含正确的Web Service端点地址，请求者通过该地址与Web Service交互**

WSDL元素关系

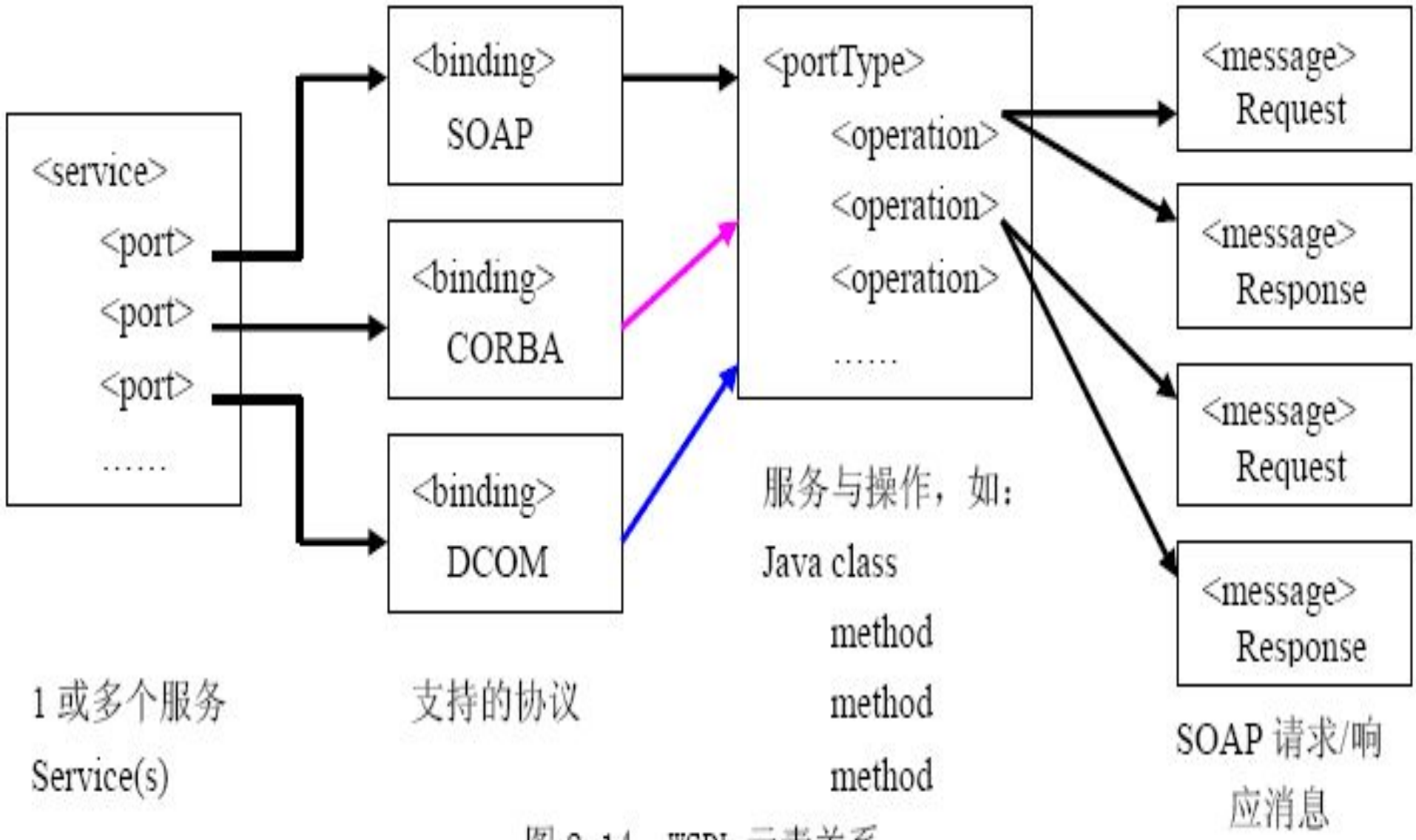
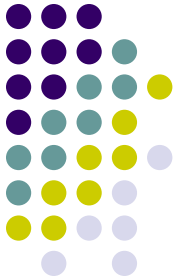


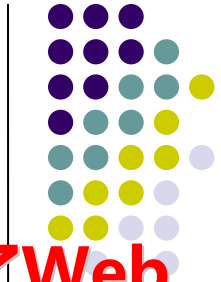
图 2-14　WSDL 元素关系

WSDL元素关系



- WSDL接口元素
  - Types(类型) 采用一些类型系统(如XSD)的数据类型定义的容器；
  - Message(消息) 被交换数据的分类抽象定义；
  - Operation(操作) 服务所支持的操作的抽象描述；
  - Port Type(端口类型) 一个或多个端点所支持的抽象的操作；
  - Binding(绑定) 特定端口类型的具体协议和数据格式规格；
- WSDL实现元素
  - Port(端口) 一个单一的端点，定义为一项绑定和一个网络地址的组合；
  - Service(服务) 相关端点的集合。

WSDL元素关系



在服务接口定义中，WSDL: portType元素中定义了Web服务的操作。这些操作定义了出现在输入、输出数据流中的XML消息，可以将操作理解为编程语言中的方法。WSDL: message元素指定了由那些XML数据类型构成消息的不同部分，用于定义操作的输入、输出参数。而消息中的复杂数据类型的使用则由WSDL:types元素来描述。WSDL:binding元素描述了一个特定服务接口WSDL:portType的协议、数据格式、安全和其它属性。

服务实现定义是一个描述特定的服务接口如何由指定的服务提供者实现的WSDL文档。一项Web服务模型化表达为一个服务元素WSDL:service。一个服务元素是端口元素WSDL:port的集合。端口通过绑定元素WSDL: binding与端点(如网络地址或URL)关联。

WSDL结构（1）

- 类型部分
  - 数据类型的容器，包含了所有在消息中使用的XML元素的类型定义。很多情况下这部分指的是XML Schema定义。

```
- <types>
  - <s:schema elementFormDefault="qualified" targetNamespace="http://localhost/ctem/">
    - <s:element name="Ctemp">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="Temperature"
              type="s:decimal" />
            <s:element minOccurs="0" maxOccurs="1" name="FromUnits" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="ToUnits" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="CtempResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="CtempResult"
              type="s:decimal" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="decimal" type="s:decimal" />
    </s:schema>
  </types>
```

WSDL结构（2）

- **消息部分**
  - **具体定义了通信中使用消息的数据结构，Message元素包含了一组Part元素（相当于函数中的参数）。使用TYPE所定义的类型来定义整个消息的数据结构。**

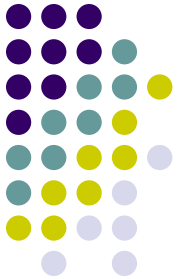
```
- <message name="CtempSoapIn">
    <part name="parameters" element="s0:Ctemp" />
  </message>
- <message name="CtempSoapOut">
    <part name="parameters" element="s0:CtempResponse" />
  </message>
```

WSDL结构（3）

- **端口类型部分**
  - **定义了一种服务访问入口*类型*。所谓访问入口类型就是传入/传出消息*模式及其格式*（WSDL中支持四种访问入口调用的模式：A：单请求；B：单响应；C:请求/响应；D:响应/请求）。PortType的定义会引用消息定义部分的一到两个消息，作为请求或响应消息的格式。**
  - Operation：**对服务中所支持的操作的抽象描述，一般单个Operation描述了一个访问入口的请求/响应消息对。**
  - PortType：**对于某个访问入口点类型所支持的操作的抽象集合**

```
- <portType name="TempConverterSoap">
  - <operation name="Ctemp">
      <input message="s0:CtempSoapIn" />
      <output message="s0:CtempSoapOut" />
    </operation>
  </portType>
```

WSDL结构（4）



- **绑定部分**
  - **定义了某个PortType与具体的网络传输协议或消息交换协议相绑定，以及具体的数据格式规范。**

```
- <binding name="TempConverterSoap" type="s0:TempConverterSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  - <operation name="Ctemp">
      <soap:operation soapAction="http://localhost/ctem/Ctemp" style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
```

WSDL结构（5）

- **服务部分**
  - **描述的是一个具体的被部署的WEB服务所提供的所提供的所有访问入口的部署细节，一个Service往往有多个服务访问入口（Port）。Port描述的是一个服务访问入口的部署细节 。**

```
- <service name="TempConverter">
  - <port name="TempConverterSoap" binding="s0:TempConverterSoap">
      <soap:address location="http://localhost/ctem/ctem.asmx" />
    </port>
```

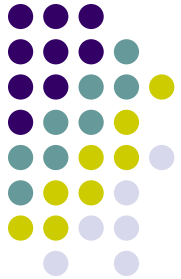
Web Services协议

- 交换信息（SOAP）
- 服务描述（WSDL）
- 服务发布和发现（UDDI）

服务发现与发布

WEB 服务发现是定位或发现一个或多个说明特定的WEB 服务的文档的过程，WEB 服务的客户通过发现来知道某个Web 服务是否存在，以及从哪里功能获取这个WEB 服务的文档。

United Description Discovery and Integration
（统一描述、发现和集成）



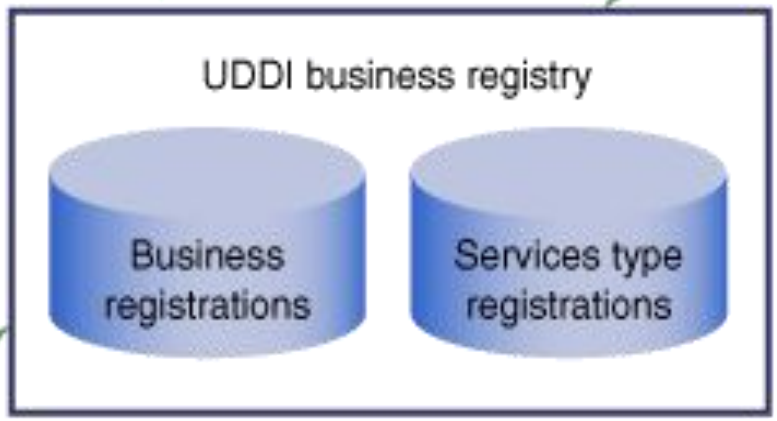
- UDDI是一种使贸易伙伴彼此发现对方和查询对方的规范。它是最终用户通过搜索企业列表、企业分类或者实际WEB服务的可编程描述。使查找产品和服务成为可能。
- UDDI不仅是一个简易的搜索引擎，它也包含了如何通过编程来和这些WEB服务进行交互。

  （对UDDI的使用可以用手动查询或程序查询）

UDDI工作原理

1. Software companies, standards bodies, and programmers populate the registry with descriptions of different tModels

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies



2. Businesses populate the registry with descriptions of the services they support

3. UDDI assigns a programmatically unique identifier (UUID) to each tModel and business registration and stores an Internet registry

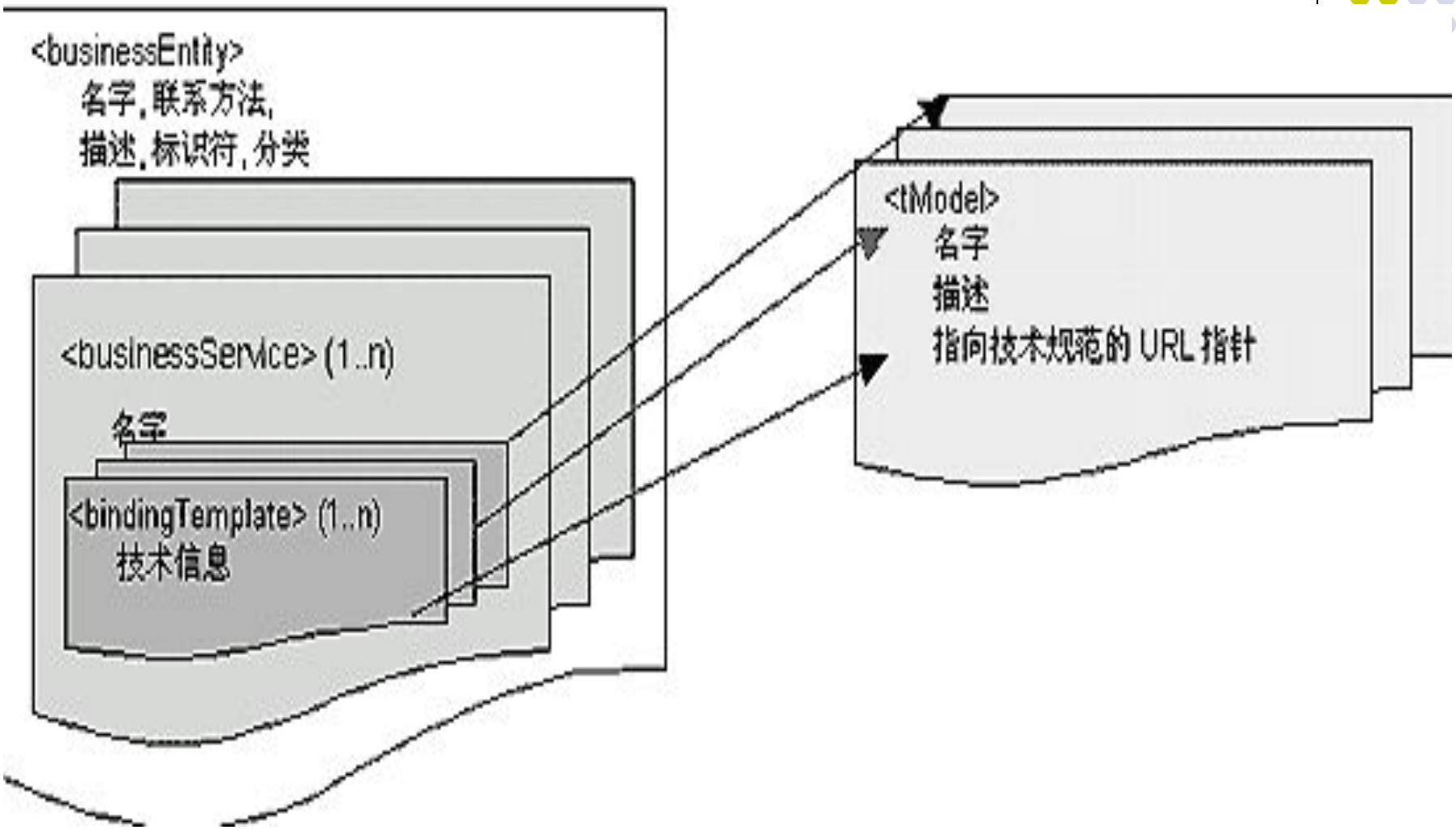
5. Businesses use this data to facilitate easier integration with each other over the Web

Populate 组装

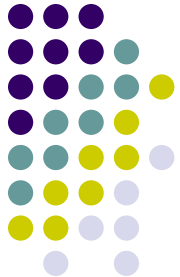
UDDI数据表类型

- **白页**
  - 包含了基本的企业信息，诸如企业名称、文字性介绍（可能是多国语言）以及联系方式，包括名称、电话号码、电子邮件以及属于这些企业的网站。
- **黄页**
  - 按分类法对企业信息进行分类，在UDDI的第一个版本中，这种分类法包括了对行业、产品或服务以及位置的分类。
- **绿页**
  - 包含了如何与企业进行电子交互的信息，包含交易过程（也就是，创建订单和检查存货等多种WEB服务）、服务描述（个人WEB服务和它们的用途）以及解释如何通过调用一个给定的WEB服务的捆绑信息。

UDDI的数据结构(1)


<businessEntity>
名字,联系方法,
描述,标识符,分类
<businessService> (1..n)
名字
<bindingTemplate> (1..n)
技术信息
<tModel>
名字
描述
指向技术规范的 URL 指针

UDDI的数据结构(2)



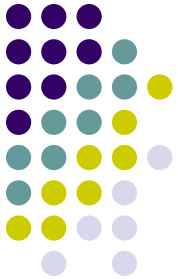
- 商业实体信息：**businessEntity** 元素

  - **许多合作伙伴希望能准确地定位到你提供的服务的相关信息，并把这些信息作为了解你们企业的开始。技术人员、程序员或应用程序希望知道你的企业名称和一些关键性的标识。所有“businessEntity”中的信息支持“黄页”分类法。如：**Business Key, Name，Description，Contacts**等。**

UDDI的数据结构(3)

- 商业服务信息：businessService元素
  - businessService 和下面要提到的bindingTemplate一起构成了“绿页”信息。businessService 结构是一个描述性的容器，它将一系列有关商业流程或分类目录的Web 服务的描述组合到一起。(包含businessKey, serviceKey, name, description).

UDDI的数据结构(4)



- 技术绑定信息：**bindingTemplate**元素

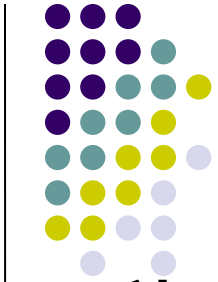
  - **对于每一个**businessService**，存在一个或多个**Web **服务的技术描述**bindingTemplate**。这些技术描述包括应用程序连接远程**Web**服务并与之通讯所必须的信息。这些信息包括**Web**应用服务的地址、应用服务宿主和调用服务前必须调用的附加应用服务等。另外，通过附加的特性还可以实现一些复杂的路由选择，诸如负载平衡等。包括（**bindingKey,businessKey,serviceeKey,accessPoint（指向**WEB**服务入口点的**URL、EMAIL、**电话号码等）等）。

UDDI的数据结构(5)

- 规范描述的指针和技术标识tModel

  - 调用一个服务所需要的信息是在bindingTemplate的结构中定义的。不过一般来说，仅知道Web服务所在的地址是不够的 。因此，每一个bindingTemplate元素都包含一个特殊的元素，该元素包含了一个列表，列表的每个子元素分别是一个调用规范的引用。这些引用作为一个标识符的杂凑集合，组成了类似指纹的技术标识，用来查找、识别实现了给定行为或编程接口的Web服务。（主要包含name,Description,categoryBag等）

UDDI程序员的API规范



- UDDI程序员的API规范是一个文档，概述了调用SOAP接口在UDDI站点上执行的每项操作。它由两部分组成：Inquiry API，用于查询和浏览UDDI注册表来发现最终用户查询的企业和服务：Publisher API，用于添加、更新和删除UDDI注册表中的企业和服务信息。
- 详细信息可查阅：http://www.uddi-china.org/pubs/ProgrammersAPI-V2.00-Open-20010608-CN.pdf

查询API

- Inquiry API
  - Find things
    - find_business
    - find_service
    - find_binding
    - find_tModel
  - Get Details about things
    - get_businessDetail
    - get_serviceDetail
    - get_bindingDetail
    - get_tModelDetail

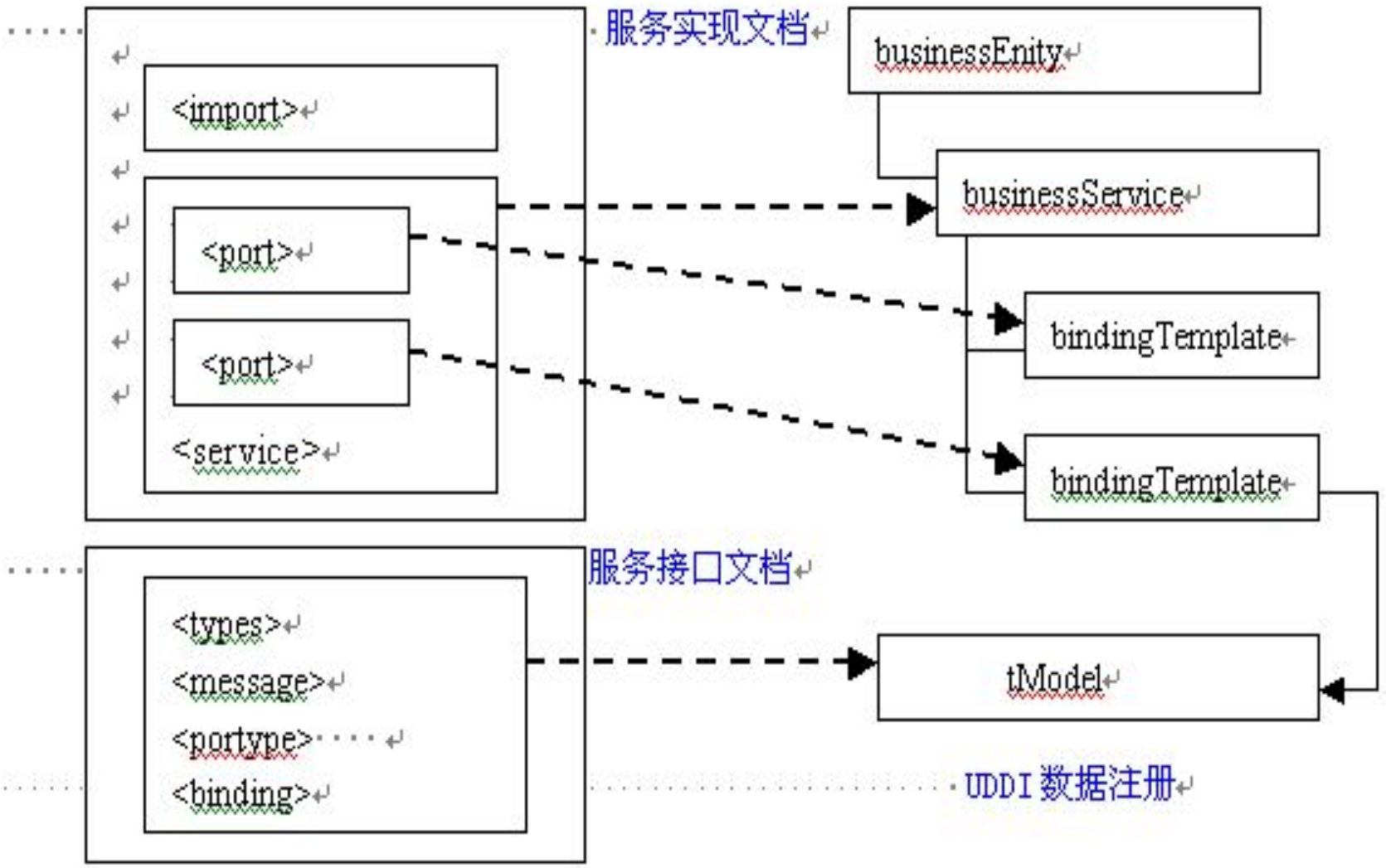
- **每个UDDI数据结构（businessEntity, businessService, bindingTemplate和tModel）都有一个find_xxx和get_xxx函数。这8个函数构成了查询API。它允许用户在数据实体上的注册表中搜索关键词或者值，然后给出所有与这个条目相关的数据。这个API主要作为查找和显示最终用户想查找的企业、服务等的一种方法。**
- **Find_xxx一般是用于定位特定的服务，get_xxx一般是用于得到完整的信息。**

发布API

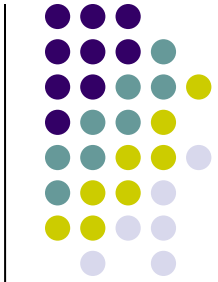
- Publishers API
  - Save things
    - save_business
    - save_service
    - save_binding
    - save_tModel
  - Delete things
    - delete_business
    - delete_service
    - delete_binding
    - delete_tModel
  - security…
    - get_authToken
    - discard_authToken

- 每个UDDI数据结构都有一个save_xxx和delete_xxx函数。加上权限认证函数（get_authToken,discard_authToken)这些函数形成了Pulication(发布)API，它允许用户（经过注册授权的用户）对现有的注册标目进行更新，用save_xxx创建新的条目，用delete_xxx能完全删除给出的数据结构。但是用户必须是已经授权的终端使用者。

UDDI和WSDL的关系



协议制定进程



- UDDI 3.0
  - uddi.org 2002/07/19
  - Ariba、Microsoft、IBM等有了UDDI中心
- WSDL 1.2
  - W3C Working Drafts’ 02/07/09
- SOAP 1.2
  - W3C Working Drafts’ 02/06/26
- XML 1.0
  - W3C Recommendation 00/10/06
- XML Schema
  - W3C Recommendation 01/05/02

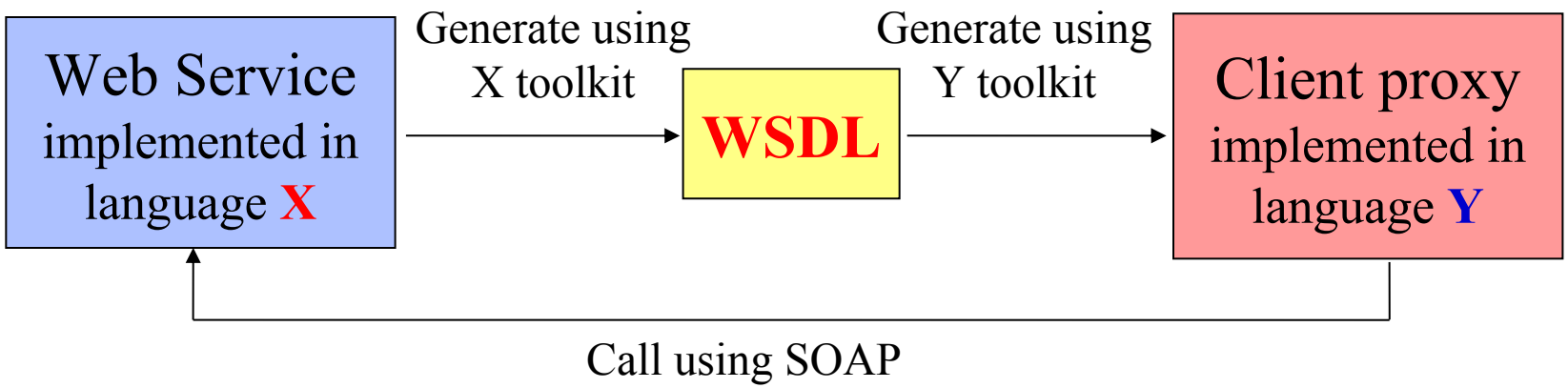
其它协议和标准

- ebXML-UN/CEFACT,OASIS
- XAML-Transaction Authority Markup Language(5 Co. without Mocrosoft)
- SwA-Soap Message with Attachments (HP,Microsoft)
- WSFL,WSCL,WSEL-IBM
- C#-Microsoft
- CLR-Microsoft
- WSCL-W3C Notes 2002/03/14
- SOAP 1.2 Attachment Feature W3C Working Draft 2002/08/14

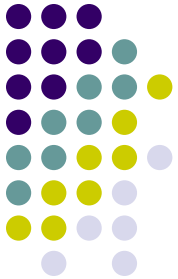
主要内容

- Web Services基本概念介绍
- Web Services体系结构
- Web Services协议
- Web Service开发
- Web Services软件产品介绍

Web Service开发

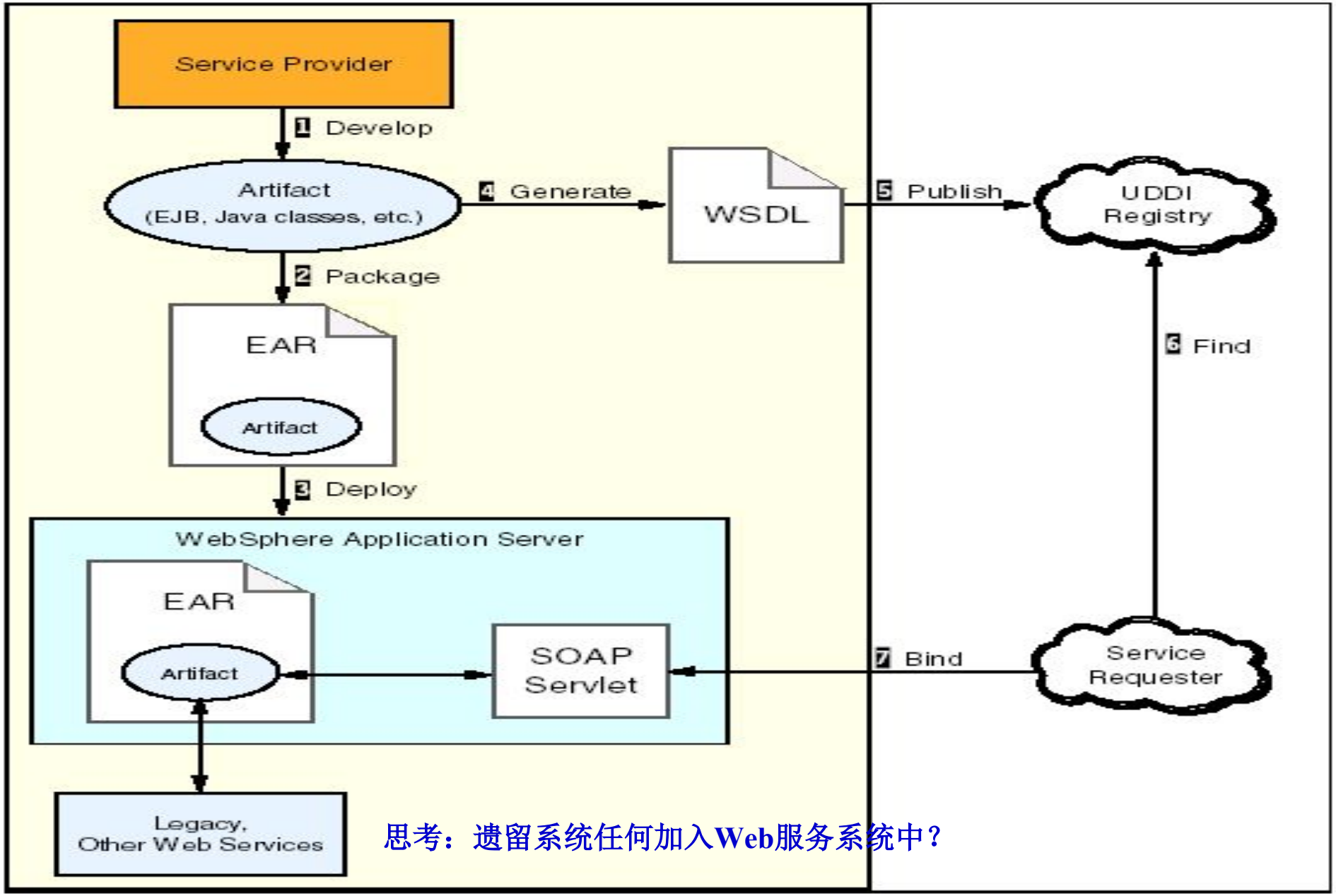


Web Service开发流程



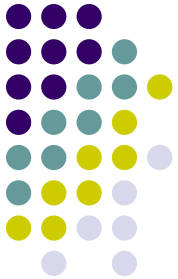
1. 编写代码
2. 打包以便部署
3. 部署到应用服务器
4. 创建WSDL描述
5. 发布服务描述
6. 发现-通过UDDI
7. 绑定、调用

创建Web Service



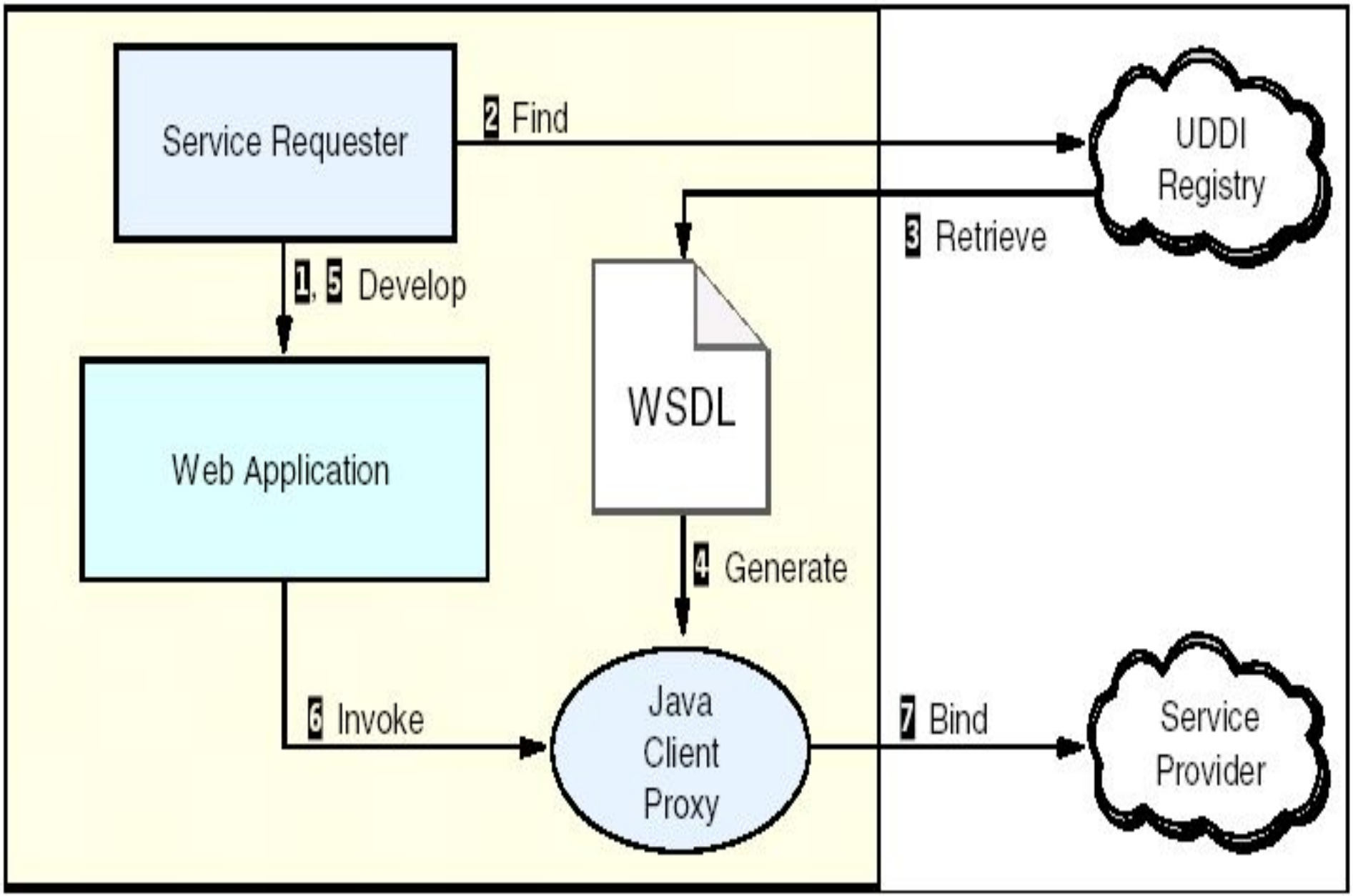
思考：遗留系统任何加入Web服务系统中？

客户端应用开发流程

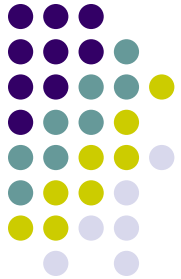


1. 开发新应用
2. 查询UDDI
3. 定位Web服务、下载其描述
4. 根据描述创建Java client proxy
5. 在应用中编写调用Java client proxy methods 代码
6. 测试应用
7. Java client proxy通过SOAP调用Web service methods

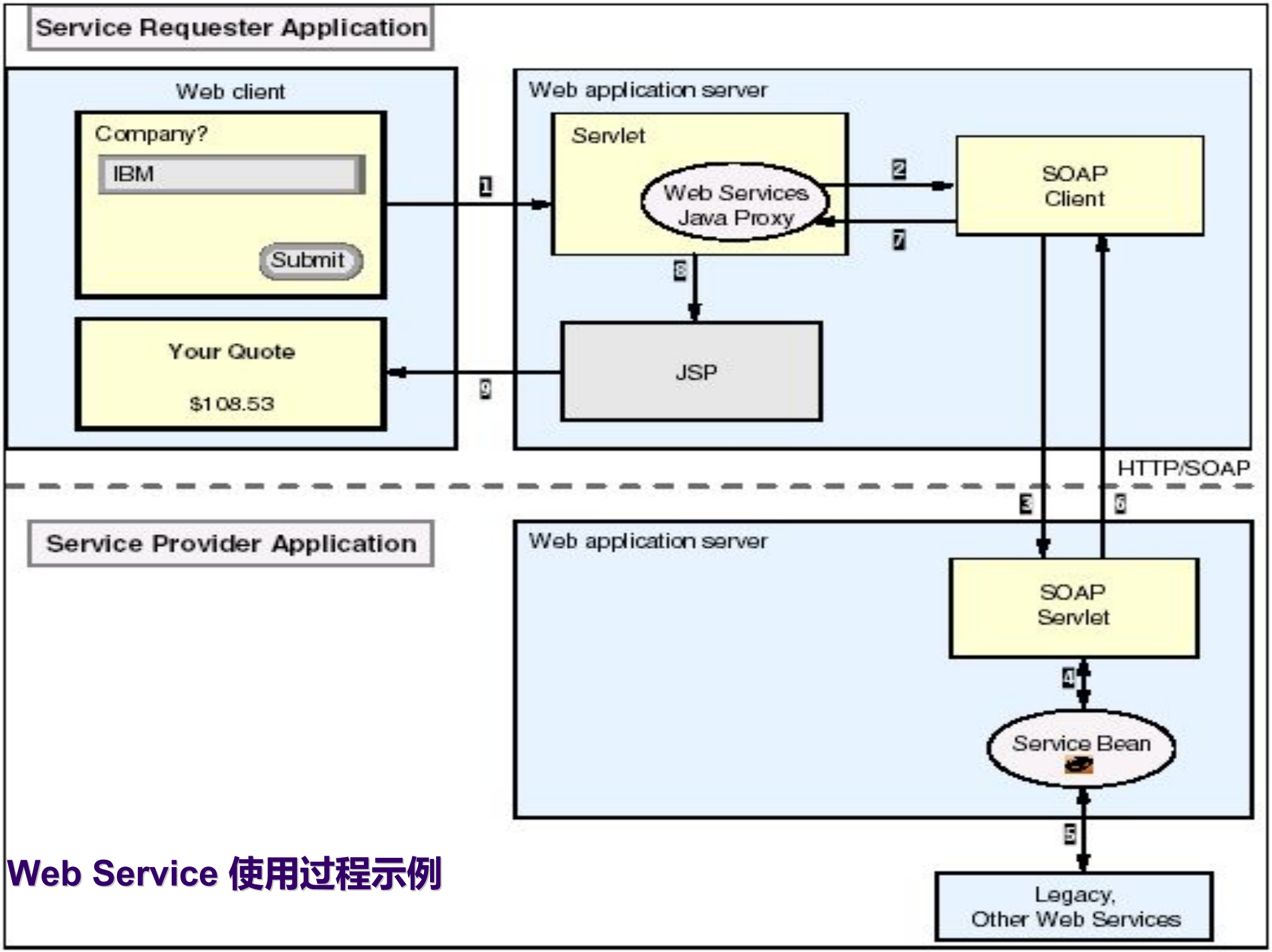
访问Web Service



Web Service 使用过程示例



1. 提交请求(参数)给应用服务器， 服务器调用Java servlet.
2. Servlet调用本地Java proxy股票方法，proxy调用SOAP客户端运行
3. SOAP客户端发送HTTP请求（方法、参数）给提供者.
4. SOAP servlet准备方法和参数, instantiates服务bean, 调用服务方法
5. 访问legacy资源
6. SOAP servlet返回结果给请求者SOAP client.
7. SOAP client返回结果给Java proxy.
8. servlet把结果放在一个data bean中，dispatche一个显示股票查询结果的JSP
9. JSP发送HTML给用户，页面中包含有服务返回的结果

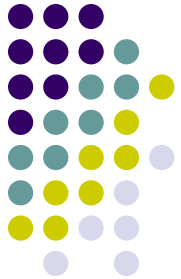


Web Service 使用过程示例

主要内容

- Web Services基本概念介绍
- Web Services体系结构
- Web Services协议
- Web Service开发
- Web Services软件产品介绍

Web Services软件介绍



- Apache Axis
- 其他软件
  - IBM
  - Sun
  - Microsoft
  - Oracle
  - BEA
  - HP

Apache eXtensible Interaction System (Axis)

- 开放源码的Web Service开发工具
  - 基于JAVA
  - 前身是Apache SOAP
  - JAX-RPC的第一个开发源码的实现
  - 下载 http://xml.apache.org/axis ，最新版本Axis 1.1

Axis 特点 (1)

- 对SOAP的支持
  - 完全支持SOAP 1.1
  - 部分支持SOAP 1.2
    - 最终将会完全支持
  - 支持SOAP with Attachments
- 对UDDI的支持
  - 本身不支持
  - 可以用IBM developerWorks 的UDDI4J

Axis 特点 (2)

- SOAP消息监听
  - TCP Monitor工具（tcpmon）监听SOAP请求/响应消息
  - 也可以用于其他SOAP工具包中
- 动态调用
  - 不使用WSDL
  - 使用JAX-RPC Call class
    - 调用web service operation
    - 需要提供SOAP router URL、服务的namespace、operation的名字和参数

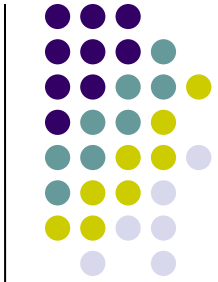
Axis 特点 (3)

- Web service部署
  - 即时部署 (JWS)
    - 把.java文件拷贝到azis的webapp目录
    - 把扩展名改为.jws
  - 定制部署方法
    - 使用Web Service Deployment Descriptor (WSDD)
    - 可以更灵活的控制类型映射
    - 部署不需要源码

Axis 特点 (4)

- 对WSDL的支持
  - Java2WSDL工具
    - 从Java服务接口类生成WSDL
  - WSDL2Java工具
    - 生成客户端stub(桩)，用于客户端应用调用服务
    - 为WSDL描述的服务生成服务skeleton(架)
    - 生成服务端需要的其他文件
  - 由已部署的服务自动生成WSDL
    - 客户端，在web service URL后面加“?wsdl”即可访问WSDL
    - 通常的格式
      - JWS: http://host:port/axis/service-name
      - Non-JWS: http://host:port/axis/services/service-name

其他Web Service软件



- IBM
  - WebSphere应用服务器
  - WebSphere Studio
- SUN
  - Sun ONE Studio
- Microsoft
  - Microsoft .NET
- Oracle
  - Oracle9i Database
  - Oracle9i Application Server
- BEA
  - BEA WebLogic Server
- HP
  - HP Application server