

# 分布计算环境

北京邮电大学计算机学院

- ◆ Web技术的发展历程
- ◆ Web 1.x
- ◆ XML技术
- ◆ 语义WEB
- ◆ Web 2.0
- ◆ Web Service
- ◆ 小结

- ◆ **什么是Web 2.0**
- ◆ **Web 2.0的典型应用和技术**
- ◆ **Web 2.0的设计模式**

- ◆ “Web 2.0” 的概念2004年开始于O'Reilly公司和MediaLive公司之间的头脑风暴
  - “同所谓的“崩溃”迥然不同，互联网比其他任何时候都更重要，令人激动的新应用程序和网站正在以令人惊讶的规律性涌现出来”
  - 那些幸免于当初网络泡沫的公司看起来有一些共同的特点。那么会不会是互联网公司那场泡沫的破灭标志了互联网的一种转折？
- ◆ 来源：
  - <http://www.oreilly.com.cn/news/whatisweb20.php?c=>
  - 作者：Tim O'Reilly

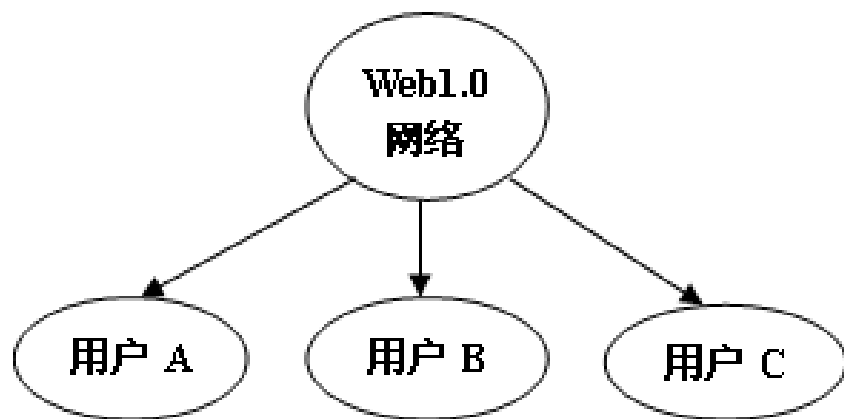
- ◆ 无统一定义
- ◆ 定义1：Web 2.0是一个作为平台的网络，跨越了所有连接的设备；Web 2.0的这些应用构成了这个平台的优势所在：发布软件成为一个持续更新的服务，并使更多的人更好的使用这种服务；获取并重组那些包括其他个人用户在内的各种来源的数据，并对其他人提供自己的数据与服务以便他们以同样的方式使用；通过一种‘共享网络架构’的方式高效的创建网络，并提供比Web 1.0更丰富的用户体验”
  - 《What is Web 2.0》，Tim O'Reilly，2005年10月

- ◆ Web 2.0是以 Flickr、Craigslist、Linkedin、Tribes、Ryze、Friendster、Del.icio.us、43Things.com等网站为代表，以Blog、TAG、SNS、RSS、Wiki等社会软件的**应用为核心**，依据六度分隔、XML、AJAX等新理论和技术实现的互联网新一代模式
  - Web 2.0 概念诠释， Blogger Don
- ◆ Web 2.0是互联网的一次理念和思想体系的升级换代，由原来自上而下的由少数资源控制者集中控制主导的互联网体系转变为自下而上的由**广大用户集体智慧和力量主导**的互联网体系。其内在的动力来源是将互联网的主导权交还个人从而充分发掘了个人的积极性参与到体系中来，广大个人所贡献的影响和智慧和个人联系形成的社群的影响取代了原来少数人所控制和制造的影响，从而极大的解放了个人的创作和贡献的潜能，使得互联网的创造力上升到了新的量级
  - 互联网协会

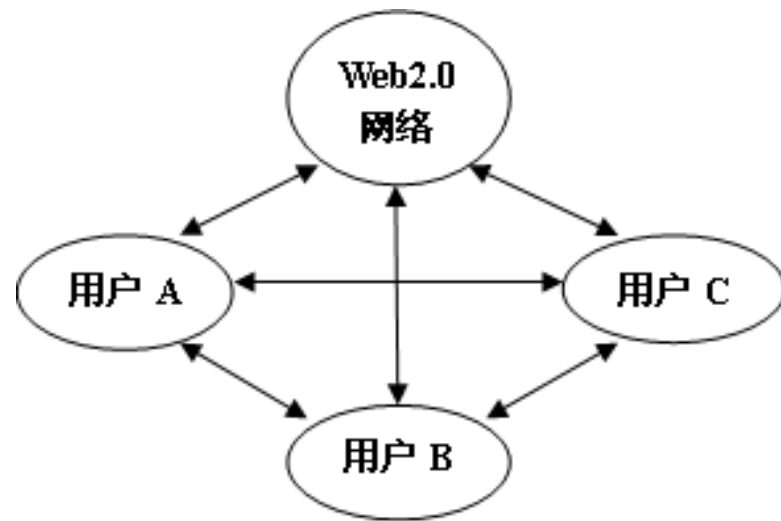
# Web1.0、2.0的主要区别

- ◆ Web1.0的主要特点在于用户通过浏览器获取信息，Web2.0则更注重**用户的交互作用**，用户既是网站内容的消费者（浏览者），也是网站内容的制造者
- ◆ Web1.0到Web2.0的转变，具体的说，从模式上是单纯的“读”向“写”、“共同建设”发展。所以互联网下一步，是要让所有的人都忙起来，用全民力量共同织出贴近生活的网
  - 一种以用户为中心的网络技术与服务
  - 以用户参与、用户互动为典型特征的万维网
- ◆ 与其说web2.0是互联网技术的创新，不如说是互联网应用指导思想的革命

# 用户与网络的关系



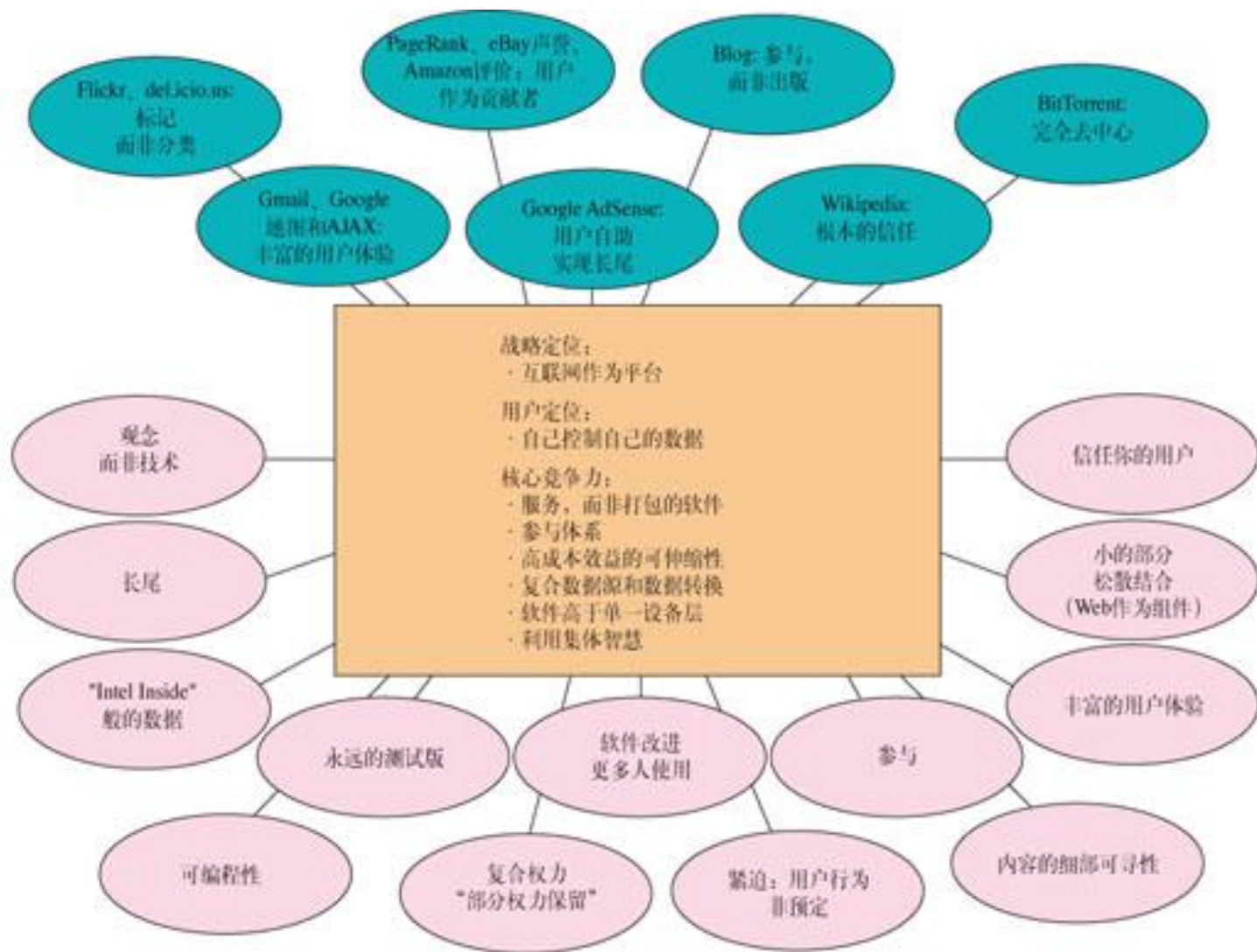
Web 1.0时代



Web 2.0时代



- ◆ Tim O'Reilly在《What is Web 2.0》一文中，给出了Web 2.0的大体框图
  - 展示了从Web 2.0核心理念中衍生出的许多概念。他指出，Web 2.0没有一个明确的界限，而是一个重力核心。不妨将Web 2.0视作一组原则和实践，由此来把距离核心或远或近的网站组成为一个类似太阳系的网络系统。距离这个核心越近，就越具有Web 2.0的特征，距离核心越远，Web 2.0的特征就越模糊
- ◆ 该图基本上仍处于演化阶段，但已经描绘出了从Web 2.0核心理念中衍生出的许多概念



- ◆ 什么是Web 2.0
- ◆ Web 2.0的典型应用和技术
- ◆ Web 2.0的设计模式

# Web 2.0的典型应用和技术

- ◆ BLOG、微博
- ◆ SNS
- ◆ 维基百科Wiki
- ◆ 内容聚合RSS
- ◆ Mash up
- ◆ Ajax
- ◆ 社会书签
- ◆ 微信、QQ
- ◆ 今日头条
- ◆ 抖音、快手
- ◆ .....

web2.0的核心不是技术而在于指导思想。web2.0有一些典型的技术，但技术是为了达到某种目的所采取的手段

- ◆ 微博是一种通过关注机制分享简短实时信息的广播式的社交网络平台。其中有五方面的理解：
  - 1、关注机制：可单向可双向两种
  - 2、简短内容：通常为140字
  - 3、实时信息：最新实时信息
  - 4、广播式：公开的信息，谁都可以浏览
  - 5、社交网络平台：把微博归为社交网络
- ◆ 微博提供了这样一个平台，你既可以作为观众，在微博上浏览你感兴趣的信息；也可以作为发布者，在微博上发布内容供别人浏览。发布的内容一般较短，例如140字的限制，微博由此得名。也可以发布图片，分享视频等。微博最大的特点就是：发布信息快速，信息传播的速度快

- ◆ 新浪微博开放平台（Weibo Open Platform）是基于新浪微博海量用户和强大的传播能力，接入第三方合作伙伴服务，向用户提供丰富应用和完善服务的开放平台。将第三方服务接入微博平台，有助于推广产品，增加网站/应用的流量、拓展新用户，获得收益



# 平台核心能力（截止13年底）

## 用户多、活跃度高

- 5.56亿用户资源
- 日活跃用户超过5千万
- 信息传播快、程度深

## 丰富的接口资源

- 200+ Open API,  
日均调用量超过330亿/天
- 各主流语言的SDK

## 完善的服务支持

- 丰富的推广渠道
- 2亿开发者基金
- 便捷的微支付系统

## 未来潜力无限

- 应用频道日活跃人数持续增长,  
无线应用半年内提升200%
- 移动端用户增长迅速

2018年第四季度末，微博月活跃用户达到4.62亿  
2019年第四季度末，微博月活跃用户达到5.16亿  
2020年第二季度末，微博月活跃用户达到5.23亿



- ◆ 超过200个数据接口，包括微博内容、评论、用户、关系、话题等信息，API日均调用量超过330亿次。不限语言、不限平台的自由接入，不收取任何费用
- ◆ 多种SDK，包括C++、PHP、JAVA、Action Script、Python、JS、iOS、Android、WP7等流行语言的软件开发工具包
- ◆ 提供发微博、读取微博等功能实例代码，可以帮助开发者快速掌握调用API方法，降低开发门槛



## 接口例：微博接口

微博		
读取接口	statuses/home_timeline	获取当前登录用户及其所关注用户的最新微博
	statuses/user_timeline	获取用户发布的微博
	statuses/repost_timeline	返回一条原创微博的最新转发微博
	statuses/mentions	获取@当前用户的最新微博
	statuses/show	根据ID获取单条微博信息
	statuses/count	批量获取指定微博的转发数评论数
	statuses/go	根据ID跳转到单条微博页
	emotions	获取官方表情
写入接口	statuses/share	第三方分享链接到微博 <span>新</span>

- ◆ 什么是Web 2.0
- ◆ Web 2.0的典型应用和技术
- ◆ Web 2.0的设计模式

## ◆ 长尾

- 小型网站构成了互联网内容的大部分内容；细分市场构成了互联网的大部分可能的应用程序。所以，利用客户的自服务和算法上的数据管理来延伸到整个互联网，到达边缘而不仅仅是中心，到达长尾而不仅仅是头部

## ◆ 数据是下一个Intel Inside

- 应用程序越来越多地由数据驱动。因此：为获得竞争优势，应设法拥有一个独特的难于再造的数据资源

## ◆ 用户增添价值

- 对互联网程序来说，竞争优势的关键在于用户多大程度上会在你提供的数据中添加他们自己的数据。因而，不要将你的“参与的体系”局限于软件开发。要让你的用户们隐式和显式地为你的程序增添价值

## ◆ 默认的网络效应

- 只有很小一部分用户会不嫌麻烦地为你的程序增添价值。因此：要将默认设置得使聚合用户的数据成为用户使用程序的副产品

## ◆ 一些权力保留

- 知识产权保护限制了重用也阻碍了实验。因而，在好处来自于集体智慧而不是私有约束的时候，应确认采用的门槛要低。遵循现存准则，并以尽可能少的限制来授权。设计程序使之具备可编程性和可混合性

## ◆ 永远的测试版

- 当设备和程序连接到互联网时，程序已经不是软件作品了，它们是正在展开的服务。因此，不要将各种新特性都打包到集大成的发布版本中，而应作为普通用户体验的一部分来经常添加这些特性。吸引你的用户来充当实时的测试者，并且记录这些服务以便了解人们是如何使用这些新特性的

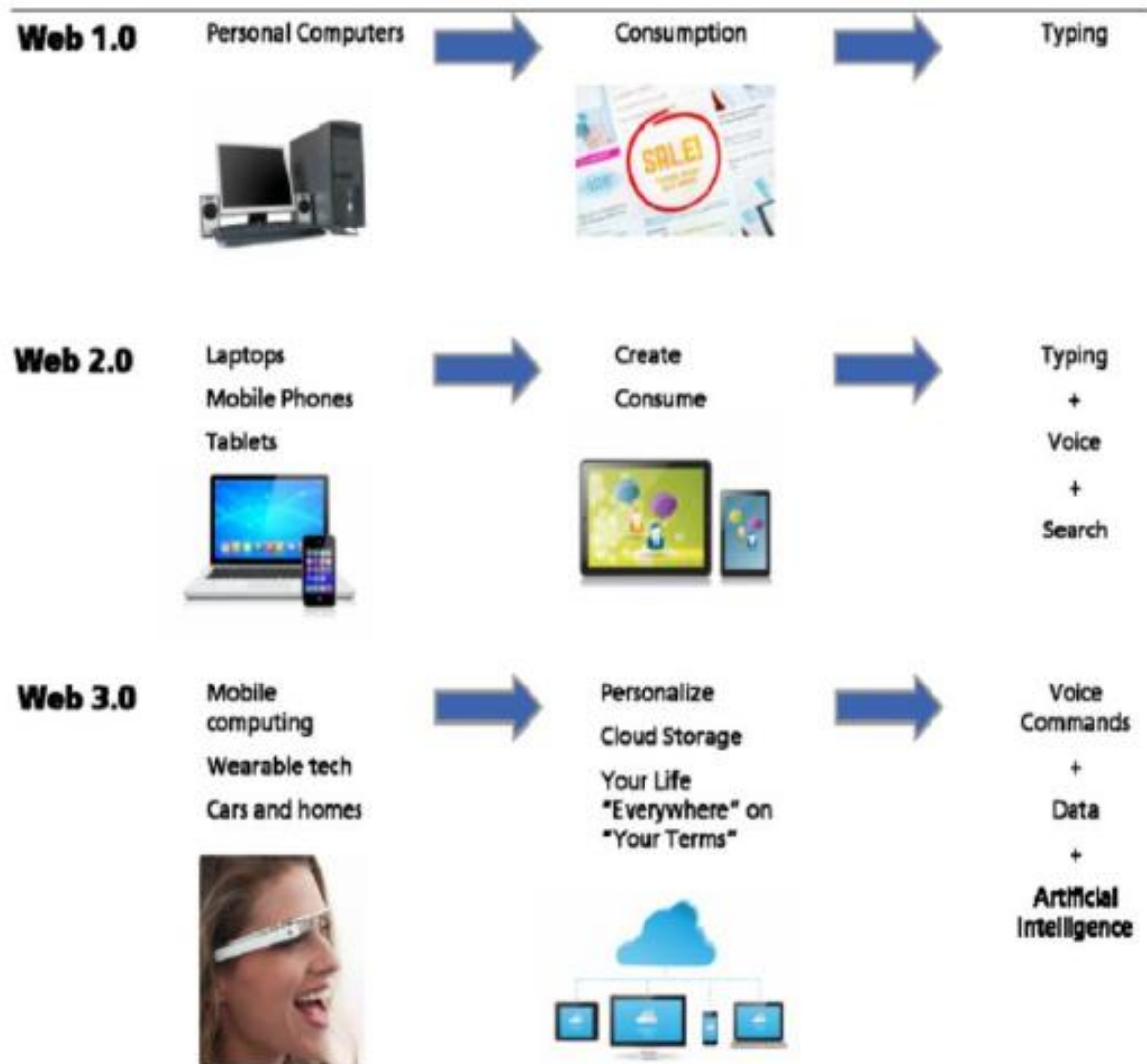
## ◆ 合作，而非控制

- Web 2.0的程序是建立在合作性的数据服务网络之上的。因此：提供网络服务界面和内容聚合，并重用其它人的数据服务。支持允许松散结合系统的轻量型编程模型

## ◆ 软件超越单一设备

- PC不再是互联网应用程序的唯一访问设备，而且局限于单一设备的程序的价值小于那些相连接的程序。因此：从一开始就设计你的应用程序，使其集成跨越手持设备，PC机，和互联网服务器的多种服务

# Web 1.0、Web2.0、 Web3.0?



## ◆ Web技术的发展历程

### ◆ Web 1.x

### ◆ XML技术

### ◆ 语义WEB

### ◆ Web 2.0

### ◆ Web Service

### ◆ 小结



- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service

- ◆ 在现有的各种异构平台的基础上，构筑一个通用的，与应用无关、语言无关的技术层，各种不同平台之上的应用依靠这个技术层来实施彼此的连接和集成
- ◆ 即：传统的Web技术解决的问题是如何让人来使用Web应用所提供的服务，而Web Service则要解决如何让计算机系统来使用Web应用所提供的服务

## ◆ 人使用Web的模式

- 浏览互相链接的文档
- 通过手工操作处理采购等商业事务
- 下载文件



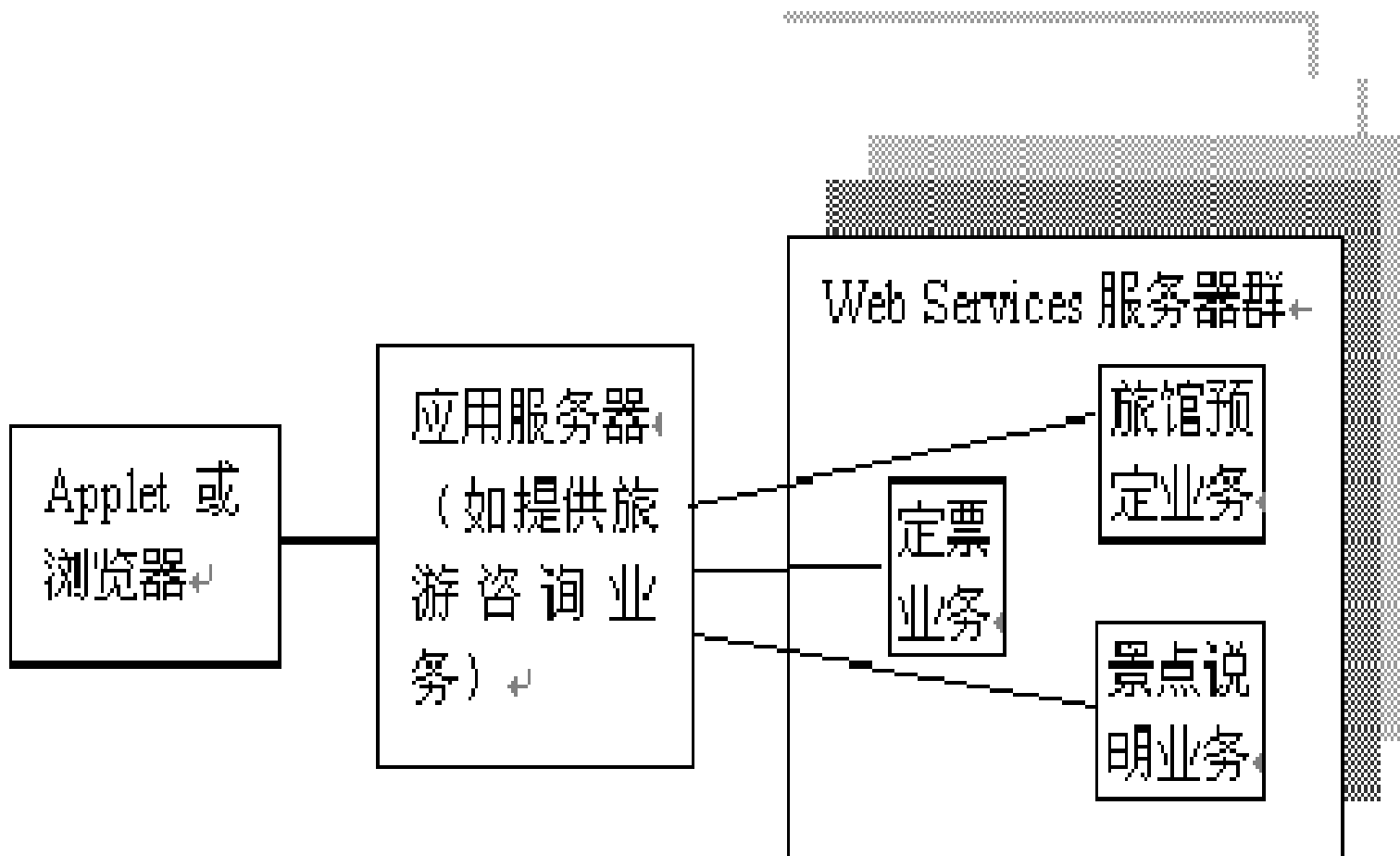
通过浏览器  
手工操作

- ◆ Web Service使得程序可以使用Web
  - 通过程序自动启动和处理商务事务，而并非使用浏览器
  - 能够在一个分布式的计算环境中动态地描述、发布、发现和调用
  - 支持基于Web Service的新型应用



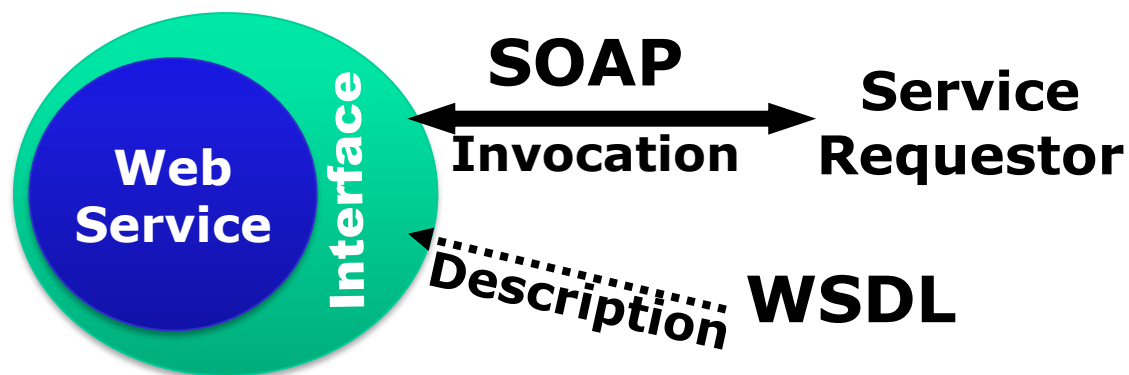
完全基于XML以及其他相关的Internet标准

## 例：旅游咨询业务系统



## 什么是 XML (SOAP) Web 服务?

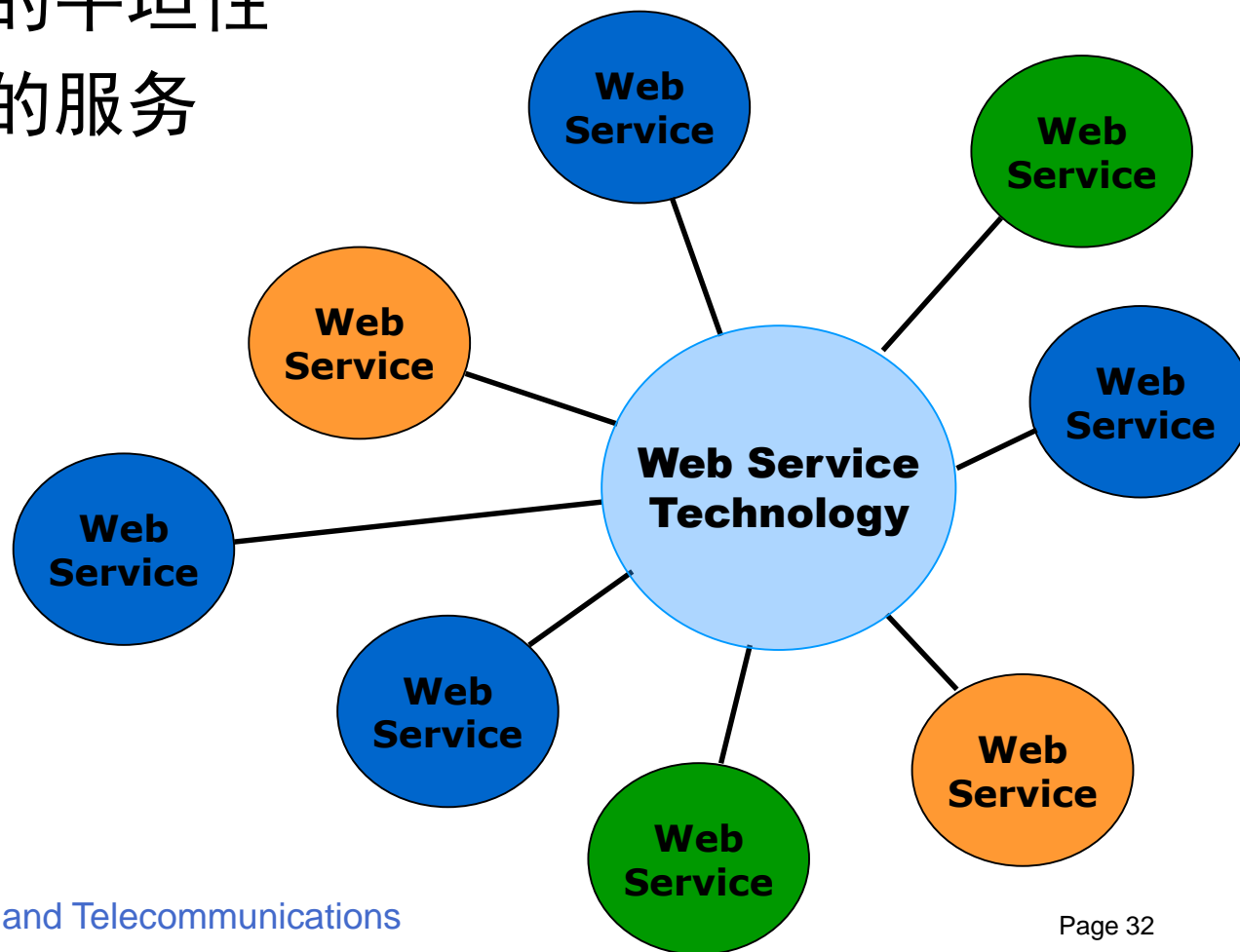
- ◆ 一个能够使用XML消息 (SOAP) 通过网络来访问的服务, 这个服务通过Interface描述了一组可访问的操作
  - 由SOAP+WSDL包装的 “Object”
  - 服务的行为、输入/输出都可使用WSDL描述
  - 适应松散耦合的网络环境, 可通过Web访问, 手段是 SOAP Message



- ◆ HTTP+XML, 最通用的访问方式
- ◆ 完好的封装性
  - 使用者仅看到Web Service提供的功能列表
- ◆ 松散耦合
  - 只要接口不变, 其使用方法就不会改变
- ◆ 使用标准协议规范
  - 使用开放的标准协议进行描述、传输和交换
- ◆ 高度可互操作性
  - 可以跨越平台、语言进行调用
- ◆ 高度可集成能力

# Web Service实现了松散耦合的连接

- ◆ 连接技术的单一性
- ◆ 连接架构的平坦性
- ◆ 基于对象的服务





- ◆ 对象接口描述: WSDL
- ◆ 对象访问: SOAP
- ◆ 对象接口发现: UDDI
- ◆ 对象实现: 任何可用于对象实现的技术



- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service

- ◆ WSDL (Web Service Description Language) 是采用XML语言来描述Web Service的属性的语言, WSDL文档可以包含以下内容:
  - What: Web Service做什么
  - Where: Web Service位于哪里
  - How: 怎样调用
- ◆ 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。

## ◆ Service

- 可以通过一个或多个ports调用

## ◆ Port

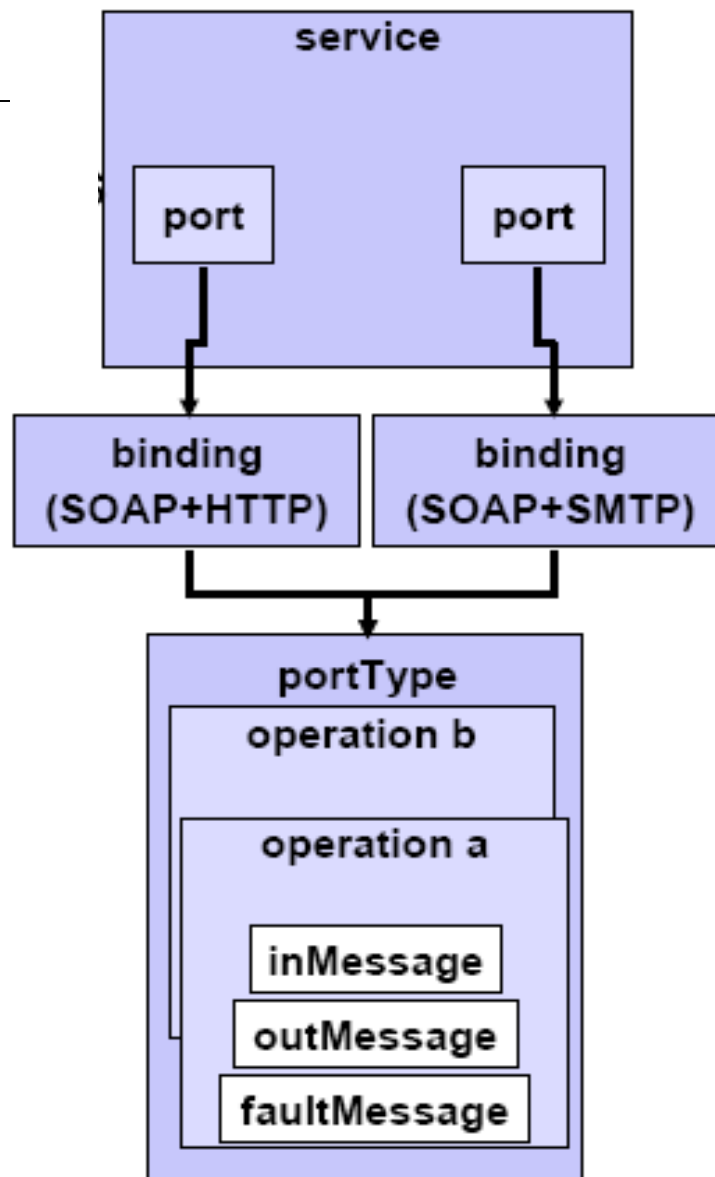
- 获取服务的方法，绑定到 portType

## ◆ Binding

- 如何获取 portType，每个 portType可以有多个绑定，如 HTTP, JMS, SMTP 等等。

## ◆ portType

- 服务的抽象定义，即接口的思想



◆ `<?xml version="1.0">`

`<types>`

//定义服务使用的任何复杂数据类型

`</types>`

`<message name=" GetLastTradePriceInput ">`

//一个message对应应在调用者和服务之间传递的一条消息，要用到前面定义的数据类型

`</message>`

`<portType name=" StockQuotePortType ">`

//定义服务提供什么操作，要用到前面定义的消息：单请求、单响应、  
//请求/响应、 响应/请求

`<binding name="StockQuoteSoapBinding "`

//定义服务如何被调用，使用何种传输协议

`</binding>`

`<service name="StockQuoteService ">`

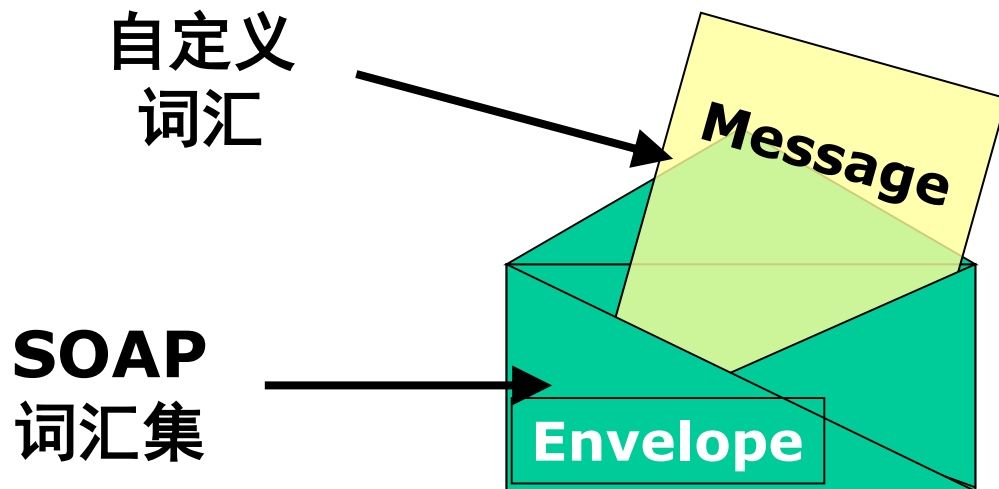
//描述服务位于哪里，一组Port组成一个服务

`</service>`

- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service

- ◆ SOAP 是一个简单的用于在Web上交换结构信息的XML协议： Simple Object Access Protocol
- ◆ 设计目标：
  - 只提供最必要的功能
  - 可以在一个周末实现
- ◆ 模块化和可扩展
  - 没有应用语义和传输语义
    - ➔ 自由的传输绑定 (不仅仅是HTTP)
    - ➔ 自由的语言绑定 (比如Java, C#)
    - ➔ 可插入的数据格式 (当然必须基于XML)

- ◆ SOAP 定义了一个 “envelope”对象
  - 使用 “envelope”包装消息自身
  - 消息可以采用自身特定的XML词汇
  - 使用namespace来区分彼此





POST /StockQuote HTTP/1.1

Host: example.com

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

SOAPAction: "http://example.com/GetLastTradePrice"

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:TradePriceRequest xmlns:m="http://example.com/stockquote.xsd">
      <tickerSymbol>MSFT</tickerSymbol>
    </m:TradePriceRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[https://blog.csdn.net/starnight\\_cbj/article/details/4986857](https://blog.csdn.net/starnight_cbj/article/details/4986857)

HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

<SOAP-ENV:Envelope xmlns:

SOAP- ENV=<http://schemas.xmlsoap.org/soap/envelope/>

SOAP-ENV:

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />

<SOAP-ENV:Body>

<m:TradePriceResult xmlns:m="

http://example.com/stockquote.xsd ">

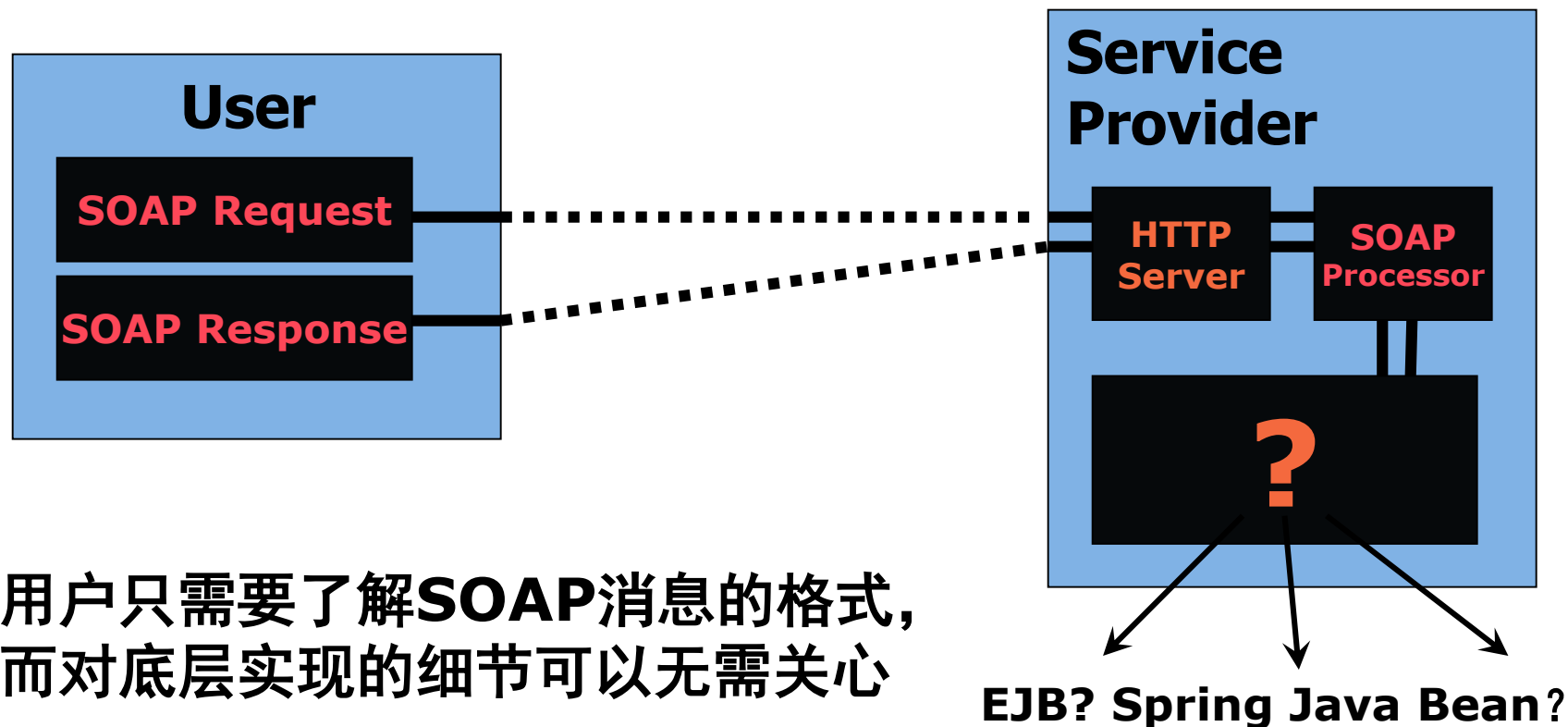
<price>74.5</price>

</m:TradePriceResult >

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

# SOAP Hides the Implementation



# 为什么SOAP在当时能成功

**Internet环境下实现技术的多样性使得早期的分布式技术无法实现普遍的互相连接**

**DCOM – 需要每个连接点都使用Windows**

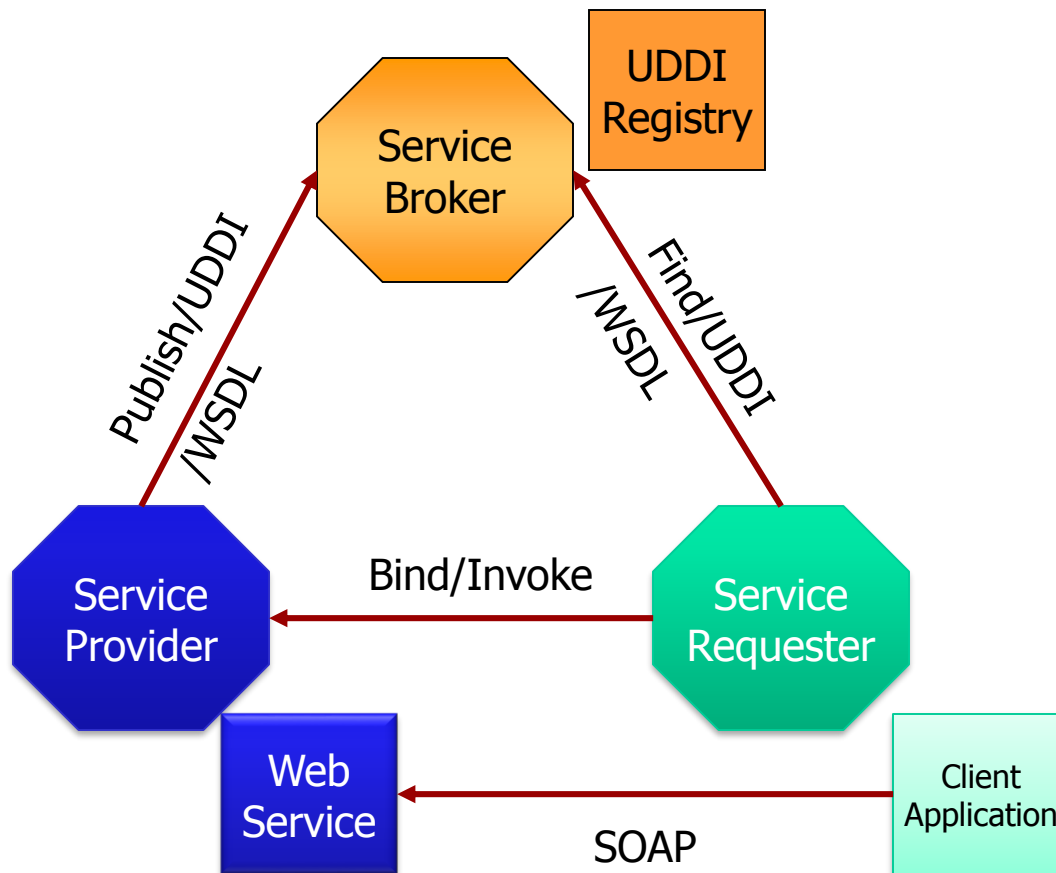
**CORBA – 需要每个连接点都有ORB**

**RMI – 需要每个连接点都使用Java**

**SOAP是基于平台独立的选择**

- 简单的XML格式
- 可以在任意平台采用任意技术
- 可以使用开放源代码资源

- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service



## ◆ Web Service的架构

### ■ 三个参与者：

- 服务提供者（Service Provider）
- 服务请求者（Service Requester）
- 服务代理（Service Broker）

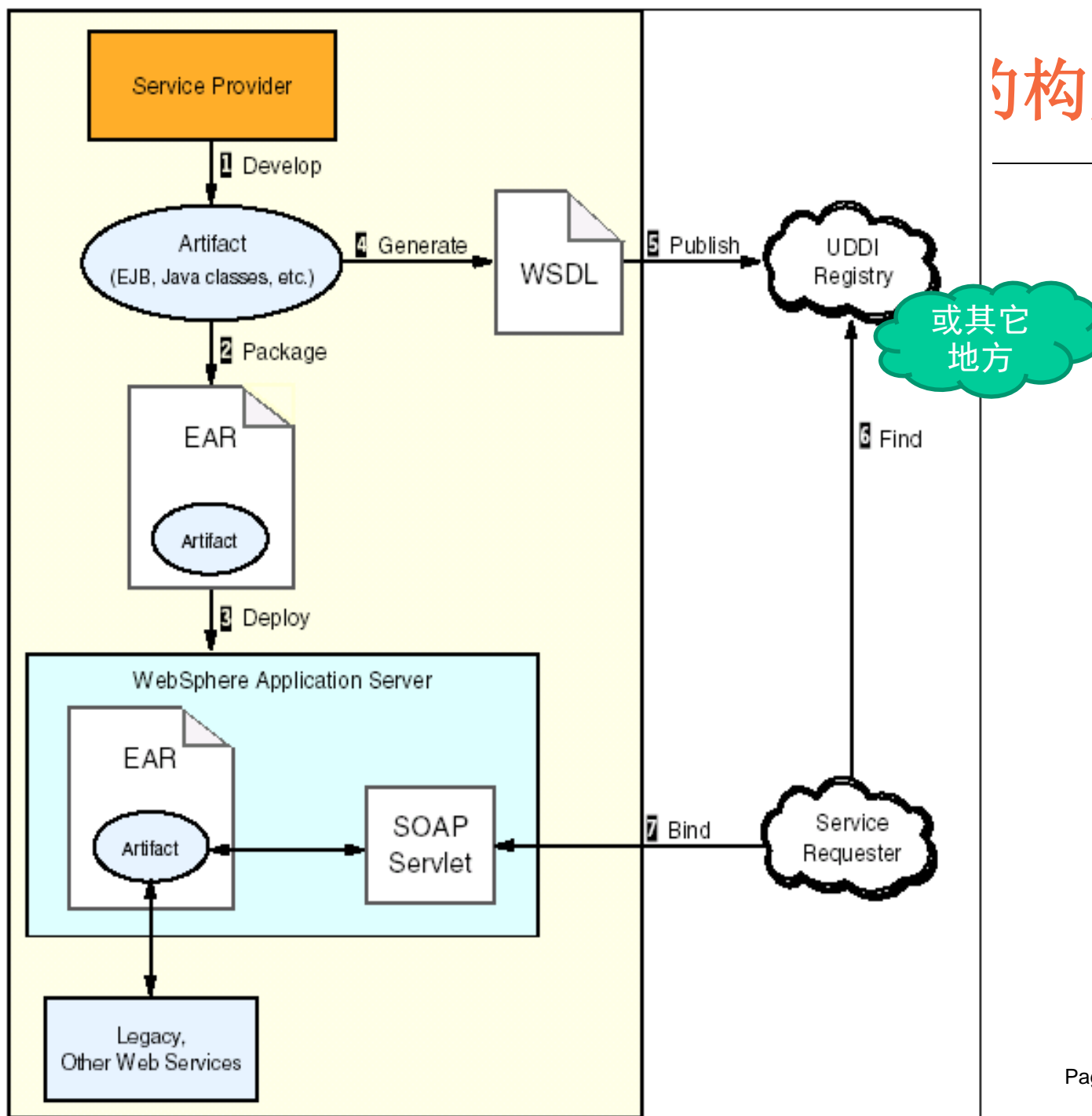
### ■ 三个基本操作

- 发布（Publish）
- 查找（Find）
- 绑定/调用（Bind/Invoke）

- ◆ 服务提供者将所提供的服务发布到服务代理的一个目录上
- ◆ 服务请求者首先到服务代理提供的目录上搜索服务，得到如何调用该服务的信息
- ◆ 根据得到的信息调用服务提供者提供的服务



- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service



```
package com.yjpeng.hello;

import javax.ws.WebService;
import javax.ws.WebMethod;
import javax.xml.ws.Endpoint;

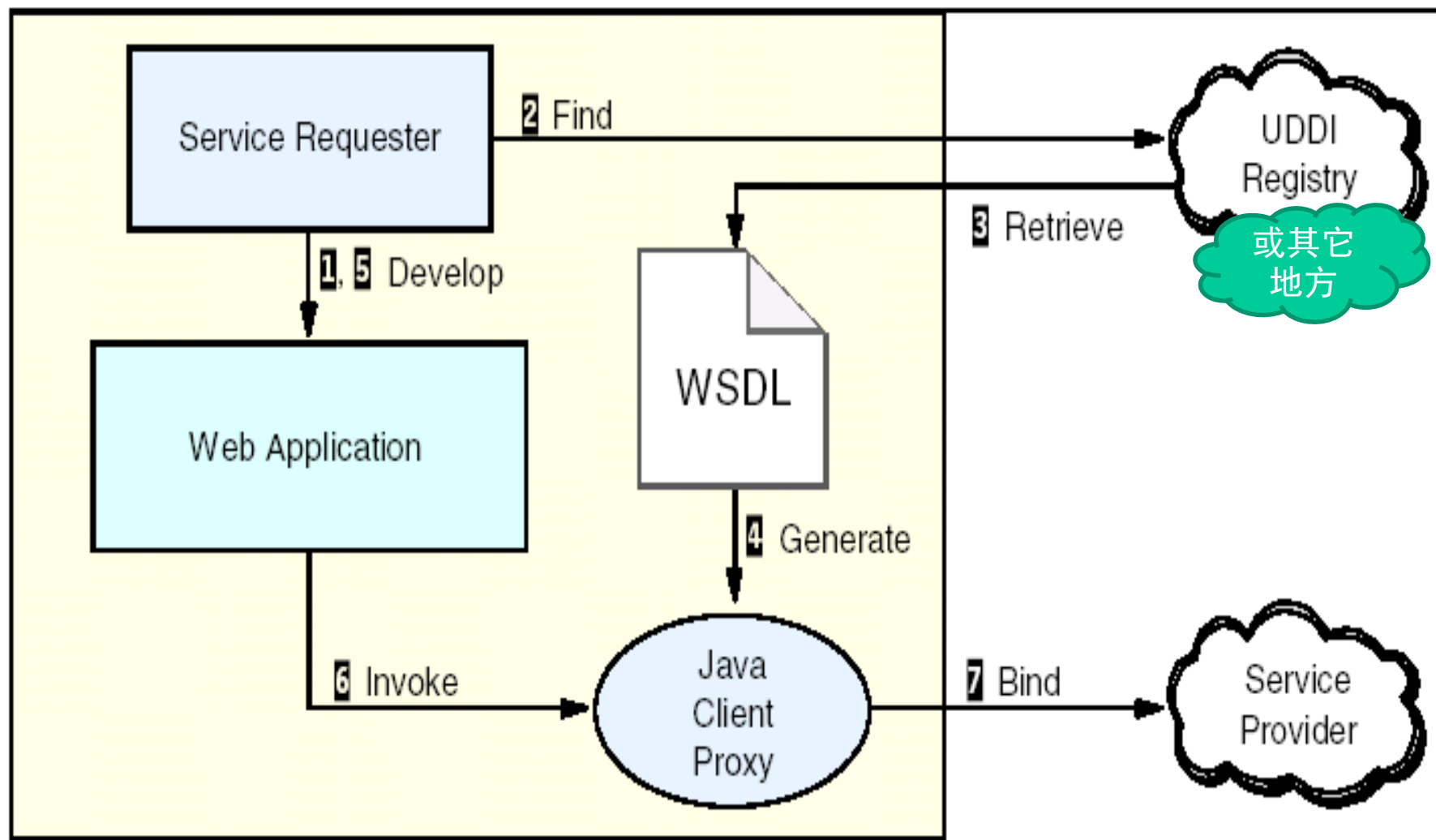
@WebService
public class HelloService {

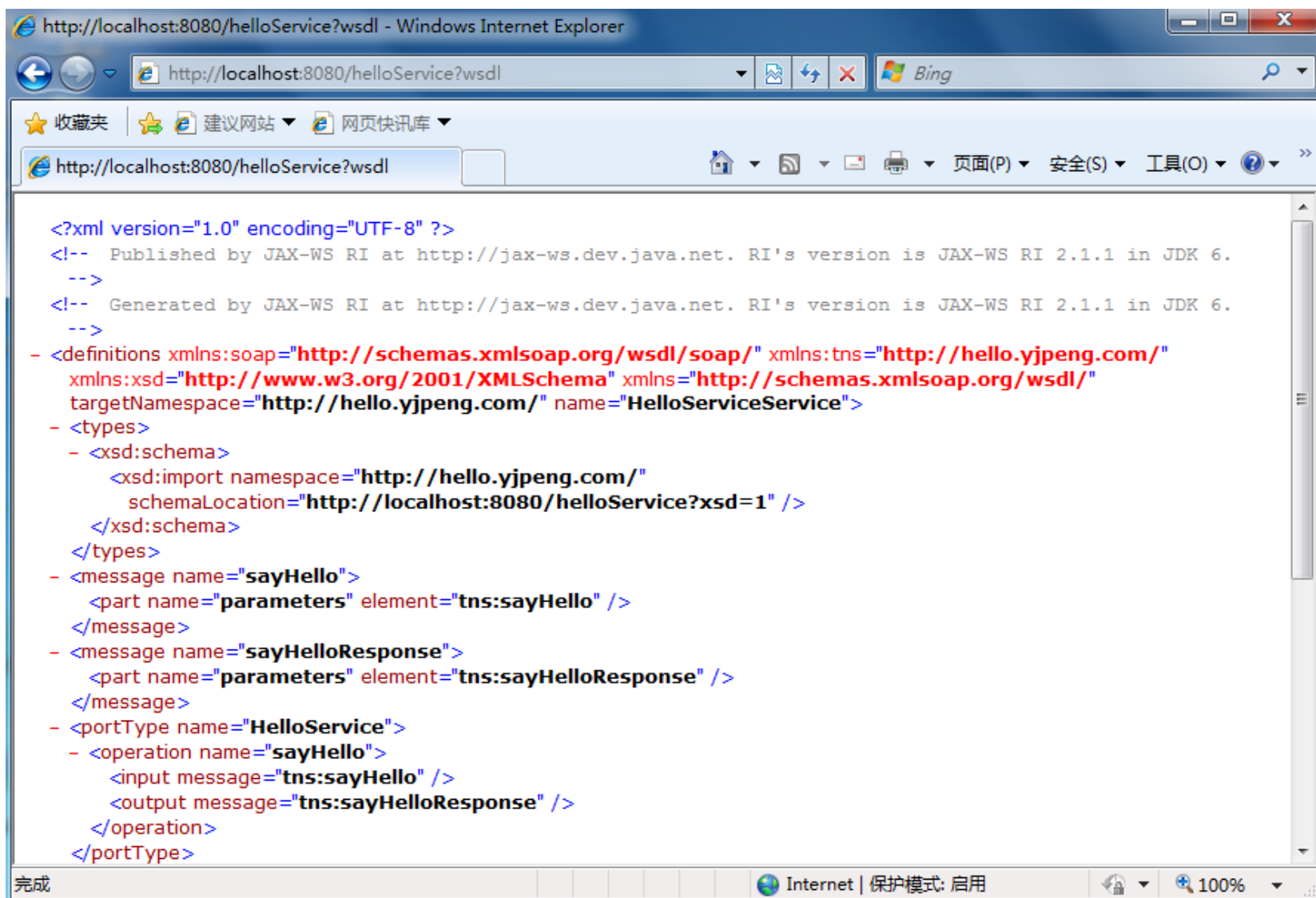
    @WebMethod
    public String sayHello(String message){
        return "Hello ," + message;
    }

    public static void main(String[] args) {
        //create and publish an endPoint
        HelloService hello = new HelloService();
        Endpoint endPoint = Endpoint.publish("http://localhost:8080/helloService", hello);
    }
}
```

<https://www.cnblogs.com/ruiati/p/6635453.html>

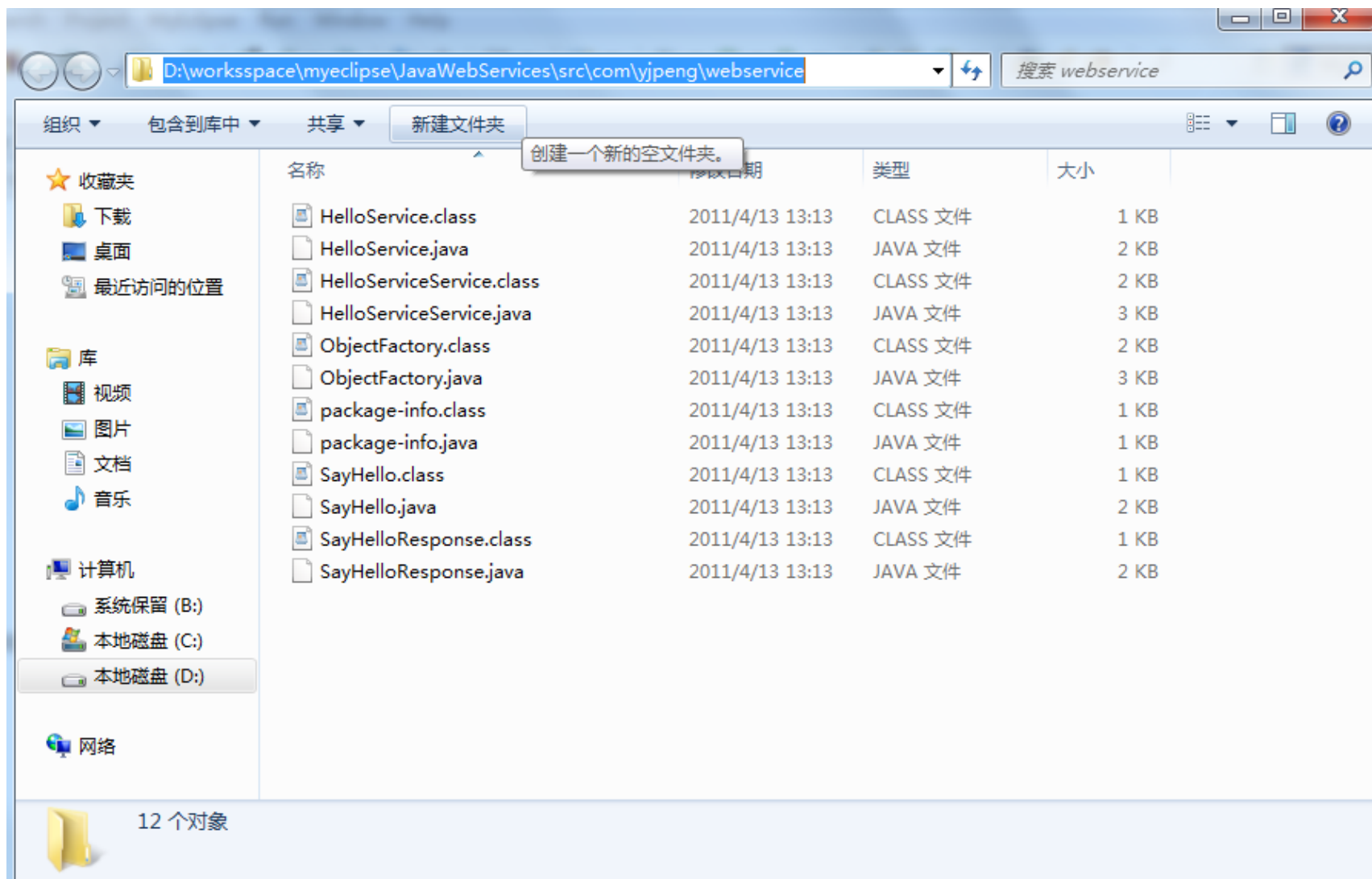
# 使用Web Service过程





```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.1 in JDK 6. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.1 in JDK 6. -->
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://hello.yypeng.com/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://hello.yypeng.com/" name="HelloServiceService">
- <types>
- <xsd:schema>
  <xsd:import namespace="http://hello.yypeng.com/"
    schemaLocation="http://localhost:8080/helloService?xsd=1" />
</xsd:schema>
</types>
- <message name="sayHello">
  <part name="parameters" element="tns:sayHello" />
</message>
- <message name="sayHelloResponse">
  <part name="parameters" element="tns:sayHelloResponse" />
</message>
- <portType name="HelloService">
- <operation name="sayHello">
  <input message="tns:sayHello" />
  <output message="tns:sayHelloResponse" />
</operation>
</portType>
```

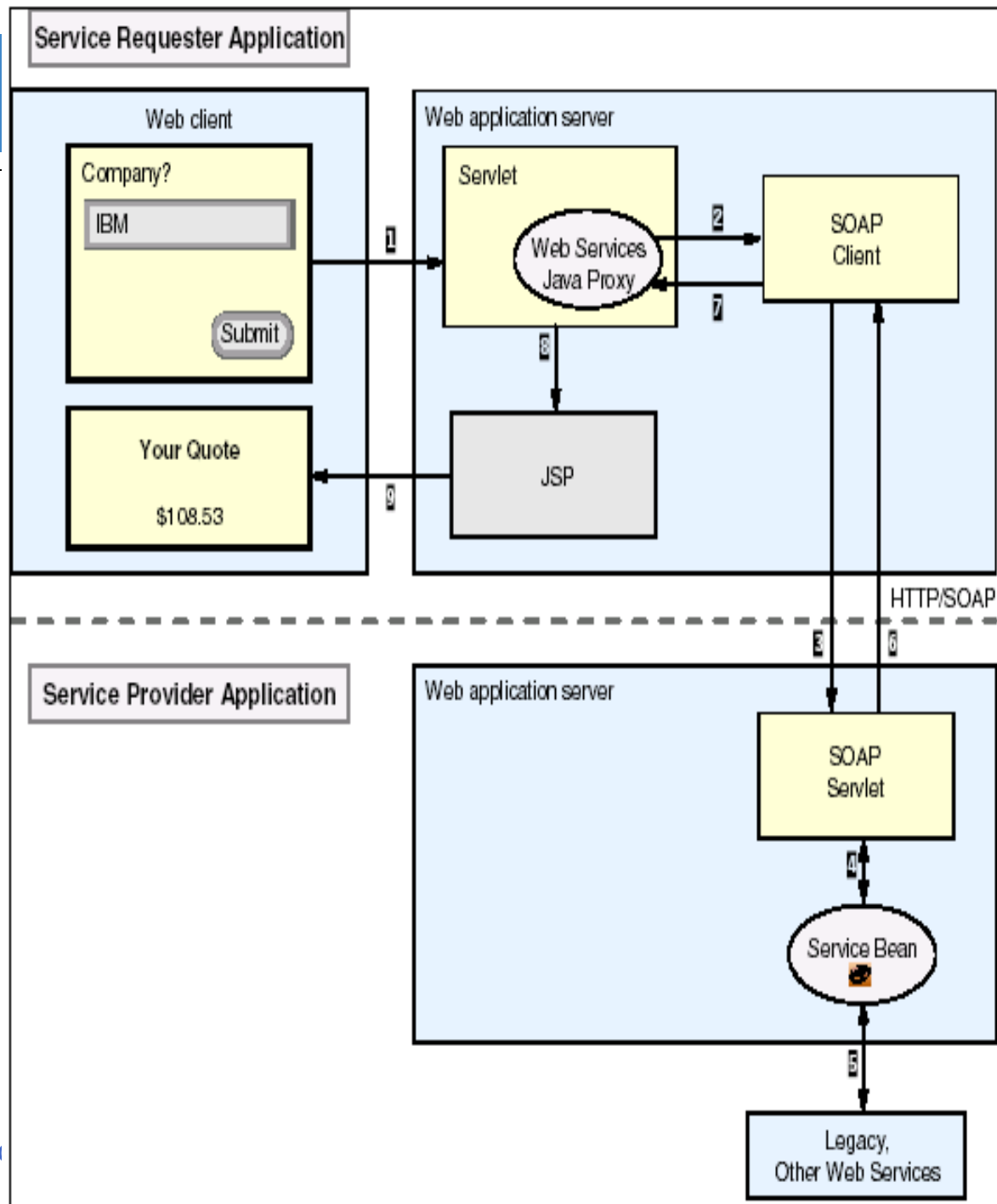
# 生成的Java Client Proxy（客户端Stub）例



```
package com.yjpeng.hello;

import com.yjpeng.webservice.HelloServiceService;

public class HelloClient {
    public static void main(String[] args) {
        HelloServiceService helloServiceService = new HelloServiceService();
        com.yjpeng.webservice.HelloService helloService = helloServiceService.getHelloServicePort();
        System.out.println(helloService.sayHello("你好"));
    }
}
```





- ◆ Web Service
  - 什么是Web Service
  - WSDL概述
  - SOAP概述
  - Web Service 架构
  - Web Service构建与使用
- ◆ RESTful Web Service

- ◆ REpresentation State Transfer, 表现层（表示/表征/表述/具象)状态转移（传输）
- ◆ 加上主语，是 Resource Representational State Transfer
  - Resource：资源，使用URI标识，如 <https://github.com/git>
    - ➔ 可以是网页、服务等
  - Representation：资源的某种表现形式。在客户端和服务端之间传送的是资源的表述，而不是资源本身。如文本资源可以采用html、xml、json等格式，图片资源可以使用PNG或JPG展现出来
  - State Transfer：状态转移
    - ➔ 表述常常是超媒体，不仅包含数据，还包含指向其他资源的链接。这样，客户端的应用状态（该应用在某时刻呈现出来的样子（各个页面））在“链接”的指引下发生变迁：连通
    - ➔ 通过HTTP动词实现：如GET、PUT、POST、DELETE
- ◆ 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

- ◆ 服务器可以生成包含状态转移的表述数据，用来响应客户端对于一个资源的请求
- ◆ 客户端借助这份表述数据，得到了当前的状态以及对应可转移状态的方式

```
HTTP/1.1 200 OK
Content-Type: application/atom+xml

<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Posts</title>
  <link href="http://example.org/posts" rel="self" />
  <link href="http://example.org/posts?pn=2" rel="next" />
```

- ◆ REST是一种面向资源的软件架构风格，包含一组架构约束条件和原则。如：
  - **客户-服务器（Client-Server）**，提供服务的服务器和使用服务的客户需要被隔离对待
  - **无状态（Stateless）**，来自客户的每一个请求必须包含服务器处理该请求所需的所有信息。换句话说，服务器端不存储来自某个客户的某个请求中的信息，并在该客户的其他请求中使用
  - **统一接口（Uniform Interface）**，客户和服务器之间通信的方法必须是统一化的（如：GET, POST, PUT, DELETE, etc）
- ◆ 满足这些约束条件和原则的应用程序或设计就是RESTful的

- ◆ 是指使用HTTP的语法和语义将其功能作为一组可寻址资源提供的Web服务
  - 使用URI来标识唯一对应的资源
  - 使用HTTP作为其应用协议
    - ➔ 使用HTTP的统一方法作为操作资源的抽象方法
- ◆ 方法信息（method information）都在HTTP方法中；作用域信息（scoping information）都在URI中
- ◆ 它能很好地利用Web简单、松耦合的特点
- ◆ 具有可寻址性、无状态性、连通性、接口统一性等特点

## ◆ 在 REST 架构中，用不同的 HTTP 请求方法来处理对资源的 CRUD（创建、读取、更新和删除）操作

### ■ POST: 创建

➔ [POST] <http://www.example.com/photo/logo>

### ■ GET: 读取

➔ [GET] <http://www.example.com/photo/logo>

### ■ PUT: 更新

➔ [PUT] <http://www.example.com/photo/logo>

### ■ DELETE: 删除

➔ [DELETE] <http://www.example.com/photo/logo>

```
package com.waylau.rest.resources;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/hello")
public class HelloResource{
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello() {return "Hello world from tsh";}
}
```

<https://blog.csdn.net/u011645059/article/details/53260534>

## ◆ 状态：

- 应用状态：客户端自己管理
- 资源状态：服务器端管理，对每个客户端相同

## ◆ Web的一个重要REST原则：客户端和服务端之间的交互在请求之间是无状态的。

- 客户端发起的每一次请求必须是独立的，不会依赖于以前的某一次请求，即服务器对该请求的处理应该不依赖于之前的任何请求
  - 如果该请求的处理需要之前请求中的一些信息，则客户端在发起请求时，应该携带此次和上次请求的全部信息
- ## ◆ 如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能



## 例：无状态请求

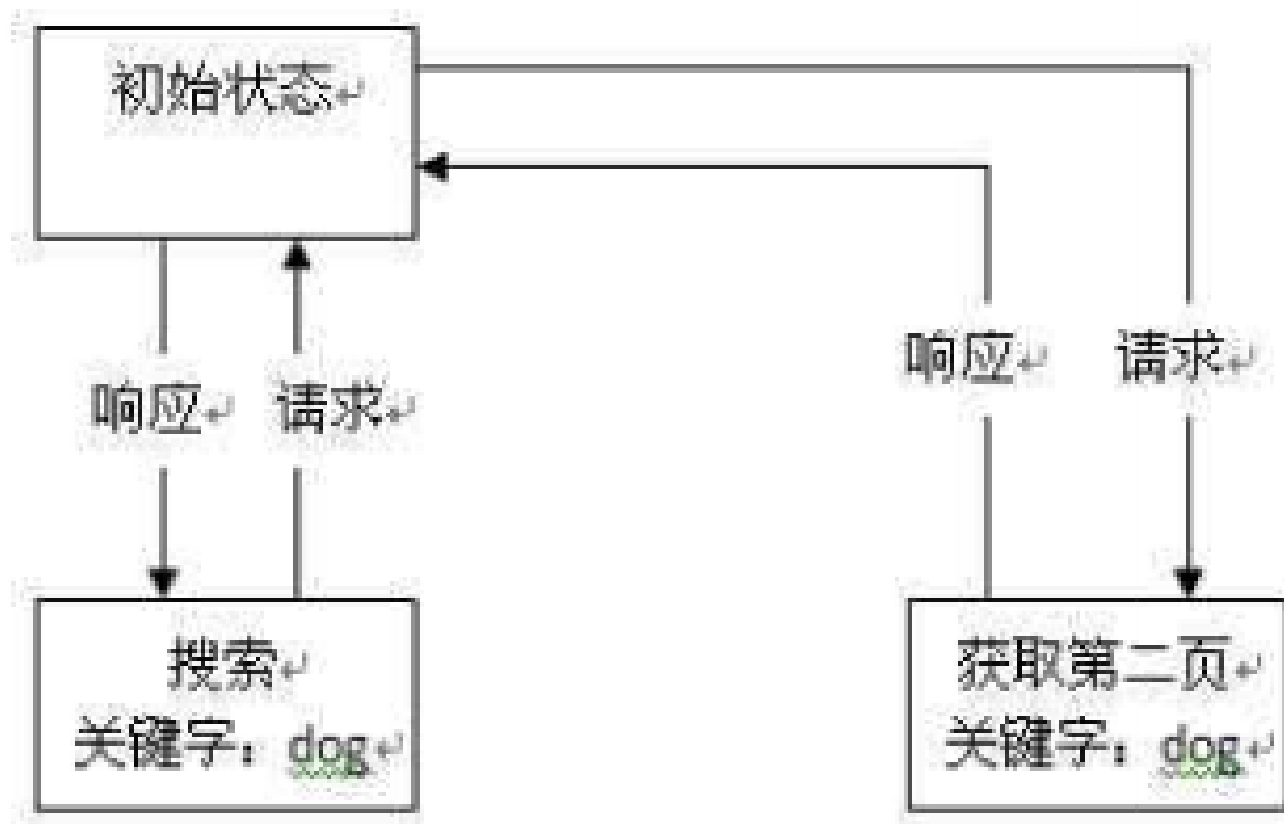
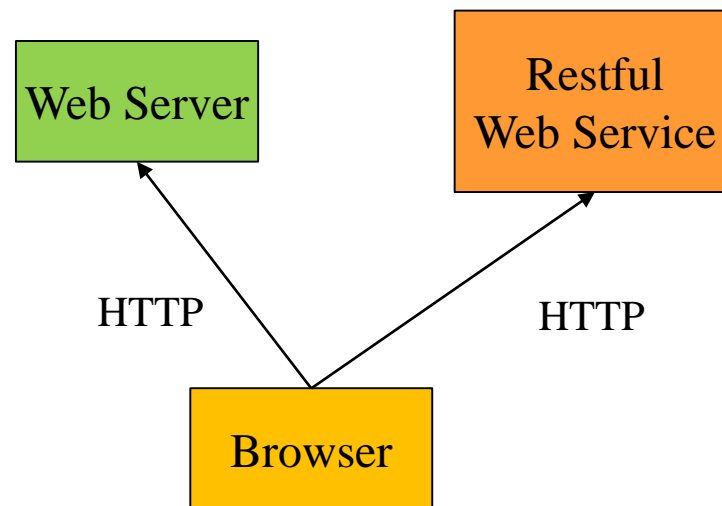
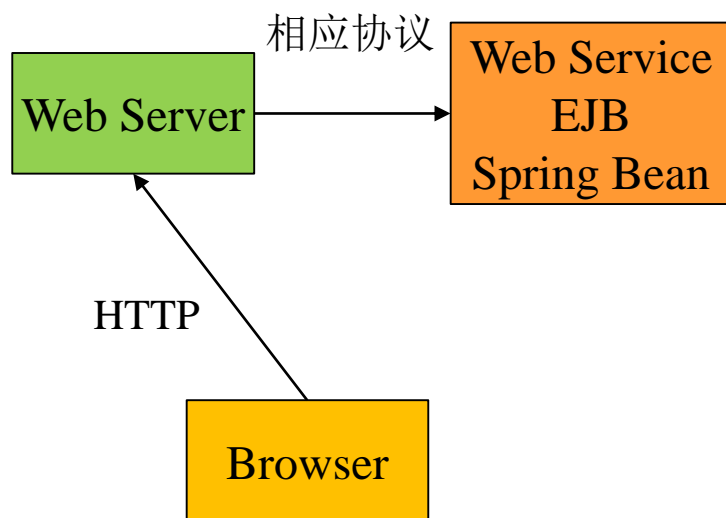


图 3.1 无状态请求原理图

<http://www.examplesearch.cn/search?keyword=dog&startindex=10>

## 两种Web应用架构



- ◆ 基于SOAP的Web 服务，也称作RPC SOAP Web 服务，RPC是面向过程的，服务请求者通过调用过程来执行指定的方法
  - 面向过程，其关注点是动作本身
  - HTTP成为一种用于传输庞大XML负载（payload）的协议，而“真正的”描述信息在XML里。最终的服务变得太过复杂、难于调试，而且要求客户端必须与服务端具备同样的配置环境
- ◆ REST架构是面向资源的，把服务器端看成是若干资源的大集合，服务器端把所有的方法、数据都暴露为资源供客户端调用
  - 面向资源，关注的是资源（名词）
  - HTTP是其应用协议，继承了HTTP带来的简单性、松耦合性

- ◆ RESTful的推崇者认为：
  - 实现Web服务并不一定非得用RPC Web服务，相反，用Web就足够了，Web基础技术足以成为默认的分布式服务平台
- ◆ 当然，有时，有比Web的优点更为重要的需求得考虑，这时也许采用RPC式架构就是合适的了
- ◆ 不少人认为：对于开放的API，如豆瓣、新浪微博等，REST好用，非常合适；对于内部开发，需要更细粒度的接口，REST不太好
  - API设计非常关键

- ◆ 由Lenard Richardson在《Restful Web Services》一书中提出
- ◆ 用于描述介于REST架构和纯RPC架构之间的Web Services
- ◆ 这些服务表面看起来是REST风格，但实际是RPC实现，只是暴露的接口类似REST

向flickr发出GET HTTP请求

```
GET services/rest?api_key=*&method=flickr.photos.search&tags=penguin HTTP/1.1  
Host:www.flickr.com
```

搜索一个penguin标签的  
照片结果列表

当需要实现的是删除操作时.....

```
GET services/rest?api_key=*&method=flickr.photos.delete&tags=penguin HTTP/1.1  
Host:www.flickr.com
```

发出GET HTTP请求

尽量避免这种设计

- ◆ Web技术的发展历程
- ◆ Web 1.x
- ◆ XML技术
- ◆ 语义WEB
- ◆ Web 2.0
- ◆ Web Service
- ◆ 小结

- ◆ Web的首要任务就是向人们提供信息和信息服务
- ◆ 侧重于人与人之间信息交流的方便
  - Web 1.0, Web 2.0
  - 语义Web
- ◆ 侧重于机器之间信息交流的方便
  - Web Service
  - 语义Web Service



## ◆ Web 1.X 的主要技术是Web领域的基础支撑技术

- Web网页、Web 服务器、浏览器

- 主要技术例

  - HTML、CSS

  - JavaScript、DOM

  - HTTP

  - JSP/Servlets

- ◆ XML：利用数据标识表示数据的含义，利用简单的嵌套和引用来实现数据元素之间的关系。
  - 信息提供者能够根据需要，自行定义新的标识及属性名
  - 数据结构的嵌套可以复杂到任意程度
  - XML文件可以包括一个语法结构描述，使应用程序可以对此文件进行结构确认
- ◆ 语义WEB：旨在赋予万维网上所有资源唯一的标识，并在资源之间建立起机器可处理的各类语义联系。元数据是语义描述的基础。
  - XML
  - RDF
  - 本体

- ◆ Web 2.0
  - 一种以用户为中心的网络技术与服务
  - 以用户参与、用户互动为典型特征的万维网
  - 典型应用和技术：BLOG、微博、社会书签、维基百科 Wiki、内容聚合RSS、SNS、Mash up、Ajax
- ◆ Web Service：在现有的各种异构平台的基础上，构筑一个通用的，与应用无关、语言无关的技术层，各种不同平台之上的应用依靠这个技术层来实施彼此的连接和集成
  - 一个能够使用XML消息通过网络来访问的服务, 这个服务描述了一组可访问的操作
    - SOAP、WSDL、UDDI
  - Restful Web Service

## ◆ 可编程的Web

### All Categories



Mapping (4,538)

Social (3,555)

Tools (2,391)

Search (2,339)

eCommerce (2,338)

Mobile (2,290)

API (2,208)

Video (1,720)

Messaging (1,661)

Enterprise (1,596)

Photos (1,573)

Financial (1,565)

Application Development (1,417)

Reference (1,388)

News Services (1,316)

Cloud (1,302)

Telephony (1,232)

Music (1,158)

Travel (1,152)

Government (1,127)

Analytics (1,040)

Payments (1,024)

Marketing (995)

Data (980)

Events (842)

Security (831)

Business (795)

# 想学习Web开发?例



此 HTML 文件名为 "te" X 建站手册 X w3school 在线教程 X W3School 在线测试工具 X

← → C ① www.w3school.com.cn ☆

← → C ① www.w3school.com.cn/tiy/t.asp?f=ajax\_async\_true ☆

← → C ① runoob.com/html/html-tutorial.html

HTML 元素  
HTML 属性  
HTML 标题  
HTML 段落  
HTML 文本格式化  
HTML 链接  
HTML 头部  
HTML CSS  
HTML 图像  
HTML 表格  
HTML 列表  
HTML 区块  
HTML 布局  
HTML 表单  
HTML 框架  
HTML 颜色  
HTML 颜色名  
HTML 颜色值  
HTML 脚本  
HTML 字符实体  
HTML URL  
HTML 搜索引擎

在本教程中，您将学习如何使用 HTML 来创建站点。  
HTML 很容易学习！相信您能很快学会它！

## HTML 实例

本教程包含了数百个 HTML 实例。  
使用本站的编辑器，您可以轻松实现在线修改 HTML，并查看实例运行结果。

**注意：**对于中文网页需要使用 `<meta charset="utf-8">` 声明编码，否则会出现乱码。有些浏览器(如 360 浏览器)会设置 GBK 为默认编码，则需要设置为 `<meta charset="gbk">`。

### 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>
  <h1>我的第一个标题</h1>
  <p>我的第一个段落。</p>
</body>
</html>
```

[尝试一下 »](#)

点击 "尝试一下" 按钮查看在线实例

## HTML文档的后缀名