



## 第2章补充内容3-- WebService

# Web服务体系结构 (WebService)

-----基于服务的体系结构



# 目标

---

- 理解Web Service (Web服务)的概念与原理
- 能够搭建Web Service 系统并编写简单的Web服务



---

## 目录

- 3. 1 Web服务概述
- 3. 2 Web服务体系结构
- 3. 3 Web服务的核心技术
- 3. 4 例子

## 3.1 Web服务概述

### 3.1.1 问题的引入

- 同一JVM虚拟机环境下的方法调用（功能调用）。

```
public class MyClient {  
  
    public static void main(String[] args) {  
        Calculator        calculator =new Calculator();  
        int result=calculator.add(1,2);  
        System.out.println(result);  
    }  
}
```

```
public class Calculator {  
  
    public int add(int x,int y){  
        return x+y;  
    }  
}
```

### 3.1.1 问题的引入

- 不同JVM虚拟机环境下的方法调用

```
public class MyClient {  
  
    public static void main(String[] args) {  
        Calculator      calculator =new Calculator();  
        int result=calculator.add(1,2);  
        System.out.println(result);  
    }  
  
}
```

A机  
器

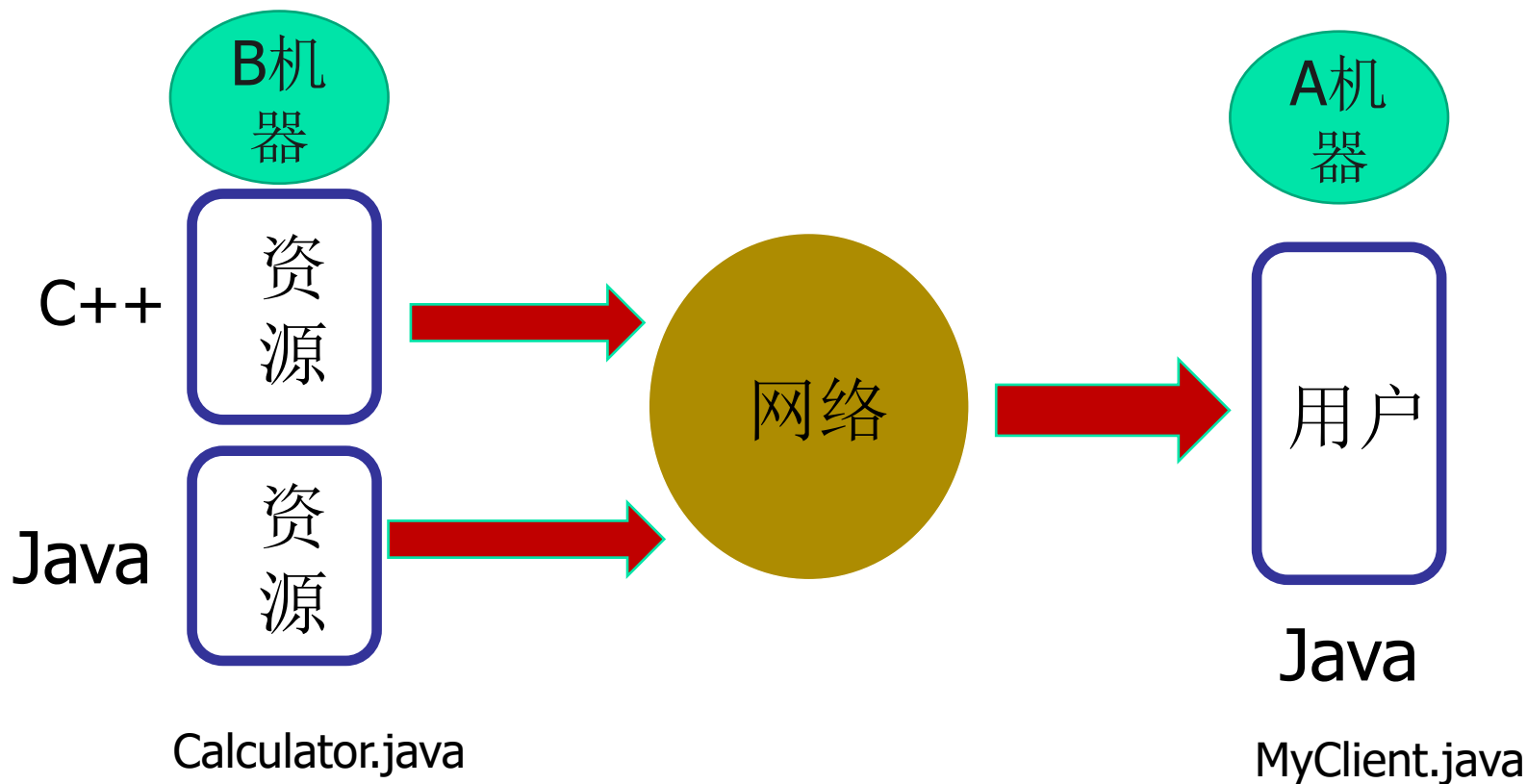
网络

```
public class Calculator {  
  
    public int add(int x,int y){  
        return x+y;  
    }  
  
}
```

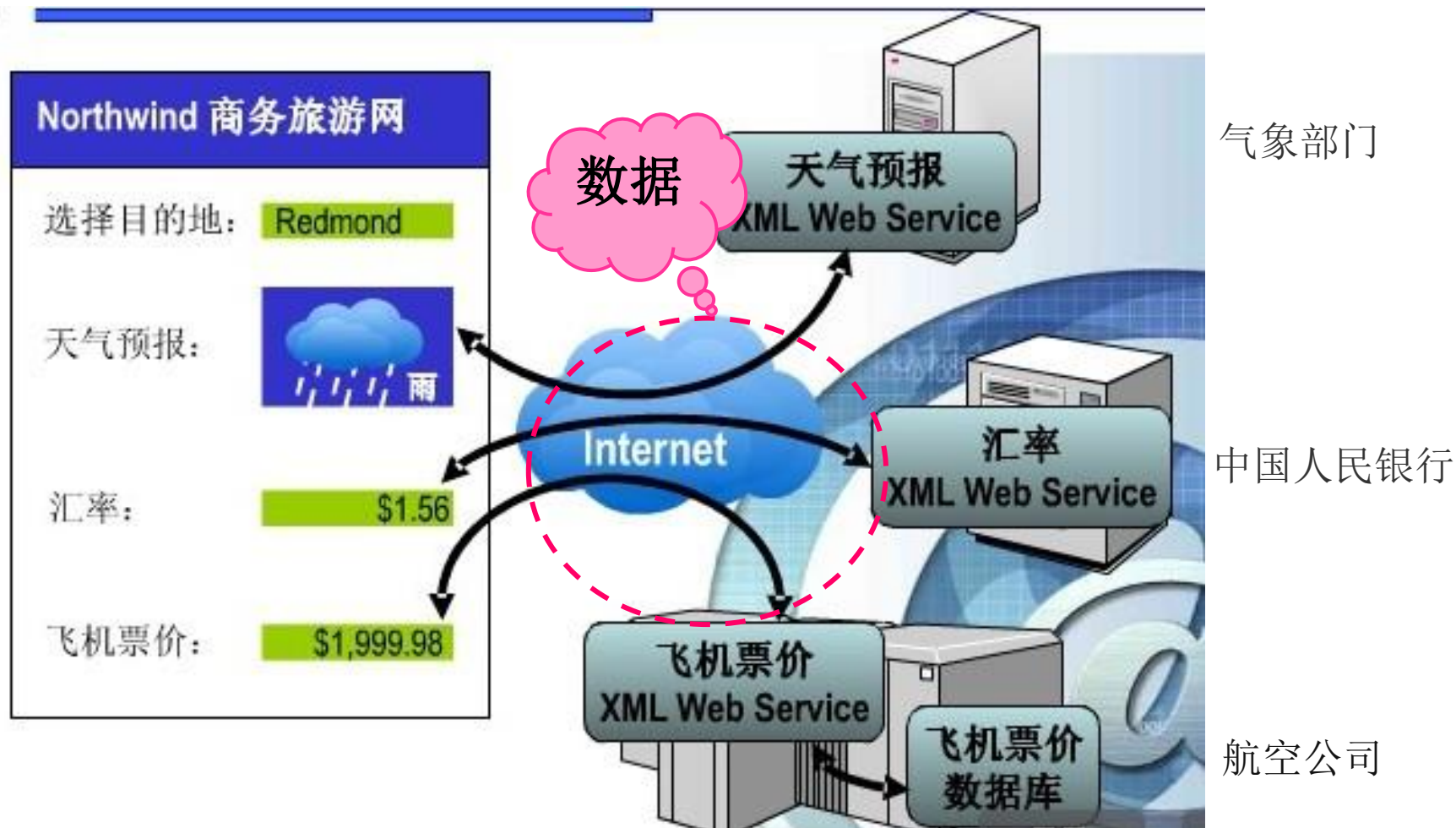
B机  
器

## 3.1 Web服务概述

### 3.1.2 如何访问异构资源（计算资源）



### 3.1.2 如何访问异构资源（计算资源）



“一切都是服务” -- 美国Microsoft(微软)公司

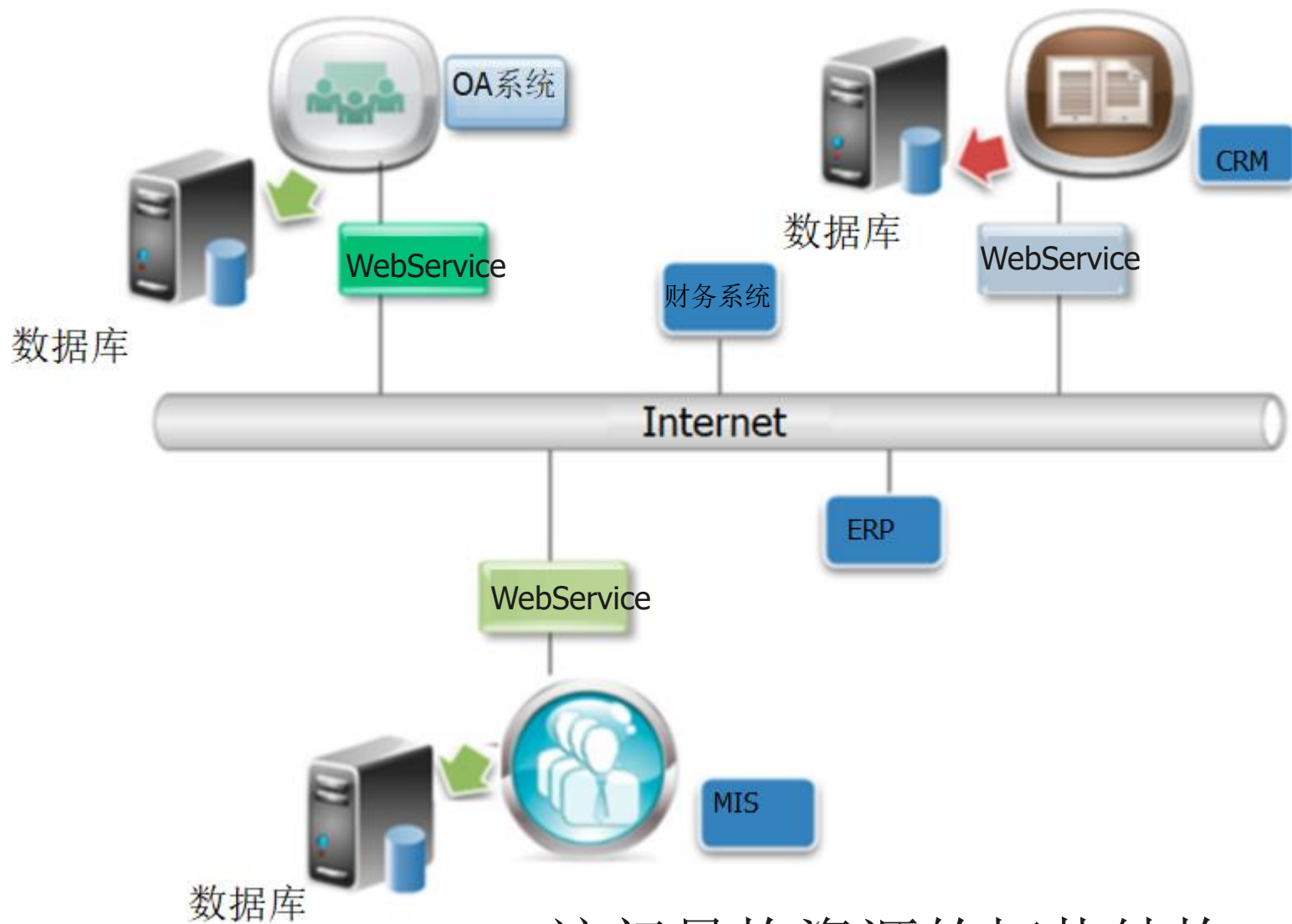
### 3.1.2 如何访问异构资源（计算资源）



分布式计算



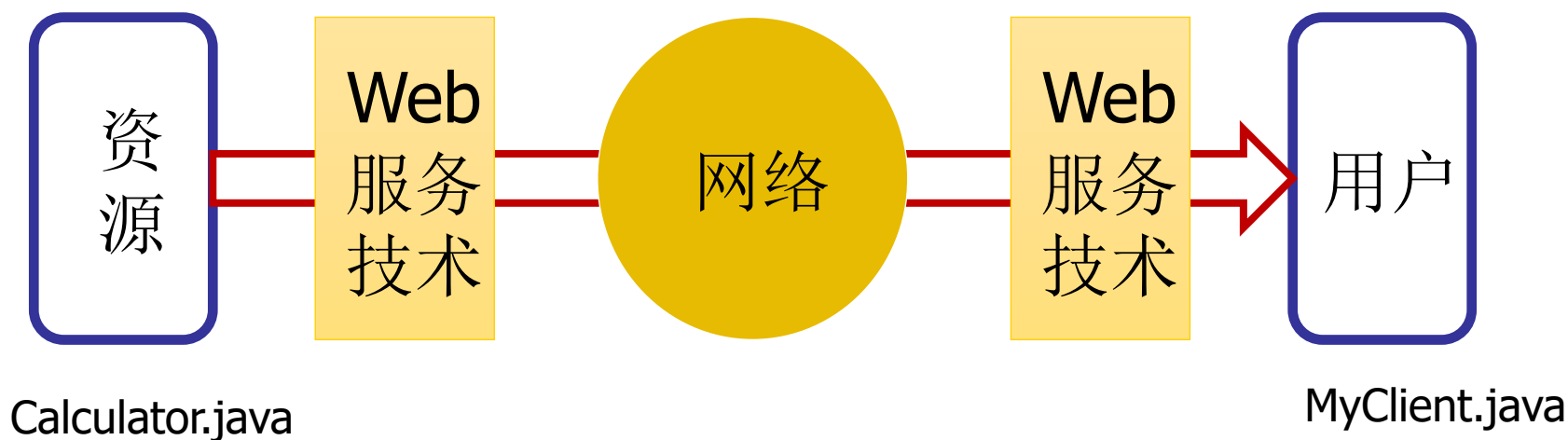
### 3.1.2 如何访问异构资源（计算资源）



访问异构资源的拓扑结构

### 3.1.2 如何访问异构资源（计算资源）

问题：资源如何表示、如何描述、如何发现、如何访问？



**Web服务（WebService）** 能够将异构的资源以一种标准的形式提供给用户访问。



### 3.1.2 如何访问异构资源（计算资源）

- Web服务的核心问题

如何表示数据类型

如何描述Web服务

如何发现Web服务

如何访问Web服务

- Web服务是基于服务的体系结构（SOA）的一种实现。



## 3.1.2 SOA概述

---

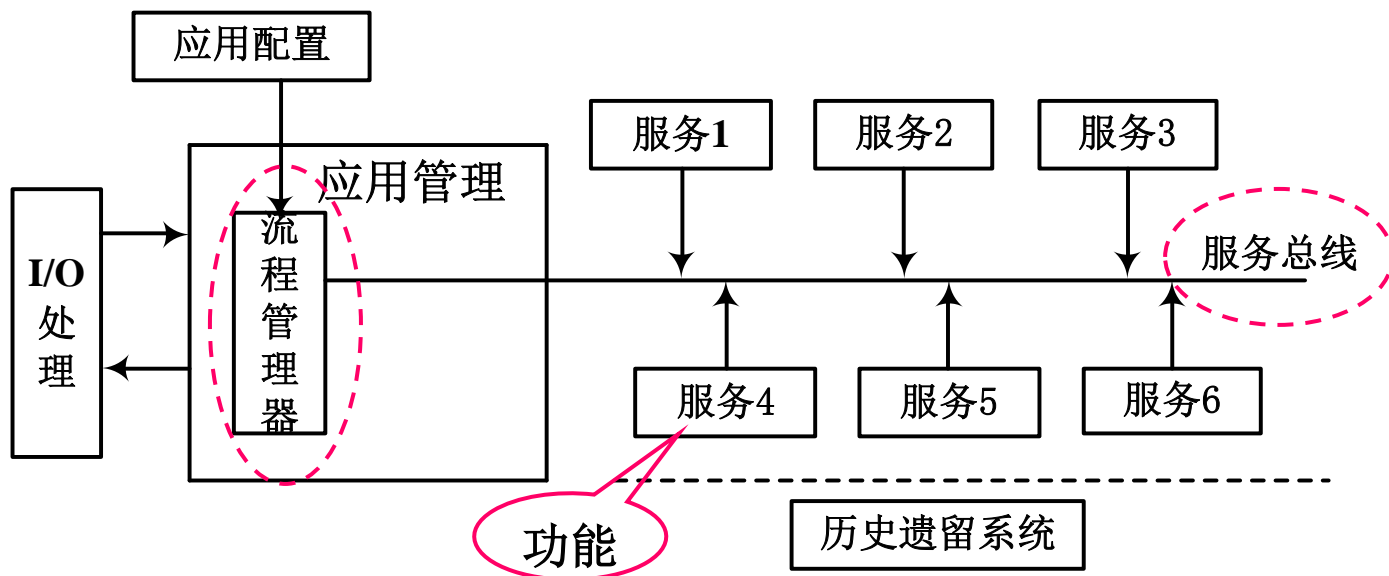
- SOA: service-oriented architecture

### ➤ W3C定义

SOA为一种应用程序体系结构风格，在这种体系结构风格中，**所有功能都定义为独立的服务**，这些服务带有定义明确的可调用接口，可以调用这些服务形成业务流程。

## 3.1.2 SOA概述

### ■ 面向服务的体系结构模型



- 在SOA模型中，所有的**功能**都定义成了独立的**服务**。
- 所有的服务通过服务总线或流程管理器来连接。
- 这种**松散耦合的体系结构**使得各服务在交互过程中无须考虑双方的内部实现细节，以及部署在什么平台上。
  - ✓ 基于Tcp Socket通信的C/S体系结构需要知道双方的实现细节



## 3.1.2 SOA概述

---

### ■ SOA模型的特征

SOA是一种**粗粒度**、**松耦合**的服务体系结构，其服务之间通过简单、精确定义**接口**进行通信，不涉及底层编程接口和通信模型。

具有以下特征：

- 松散耦合
- 粗粒度服务
- 标准化接口



### 3.1.3 Web服务概述

---

#### ■ Web服务

Web服务 (Web service) 作为一种新兴的Web应用模式，是一种崭新的分布式计算模型，是Web上数据和信息集成的有效机制。  
是SOA的一种实现。

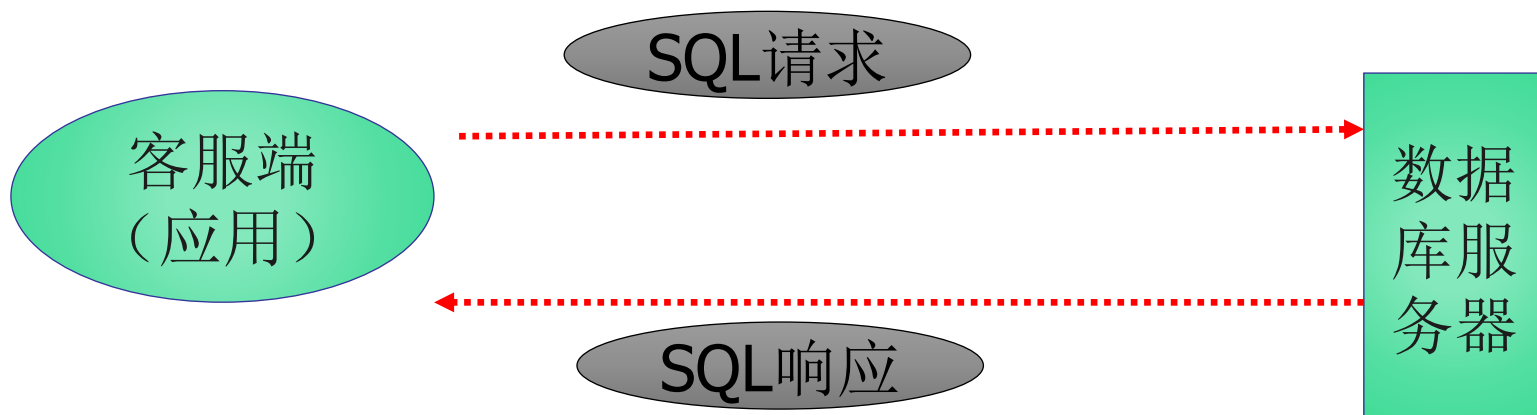
目的：

成群的个人电脑、服务器、智能设备都可以基于因特网服务无缝协同作业。



## ■ C/S、B/S、WebService

### C/S



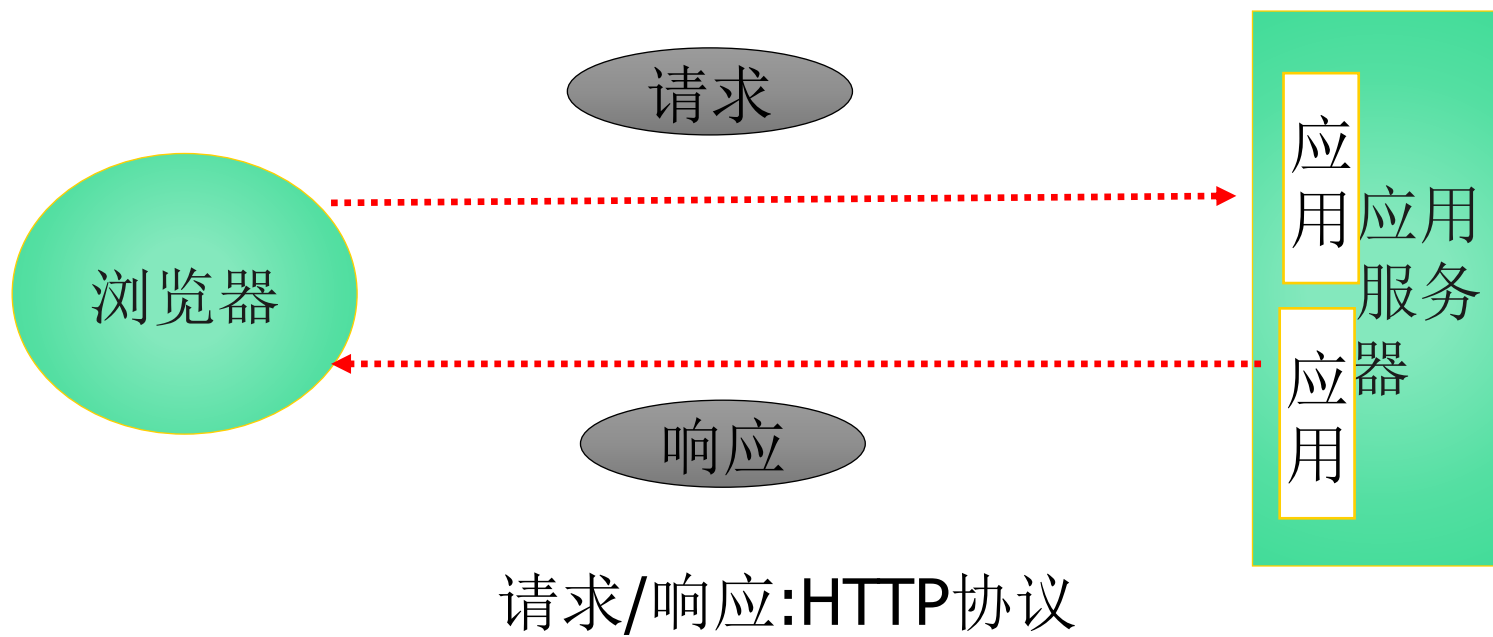
SQL请求/响应: 协议(JDBC、ODBC)





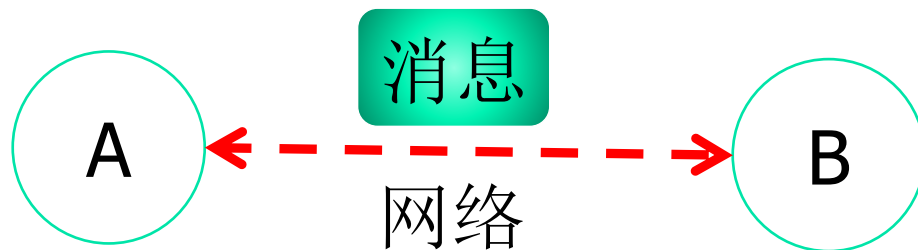
---

B/S

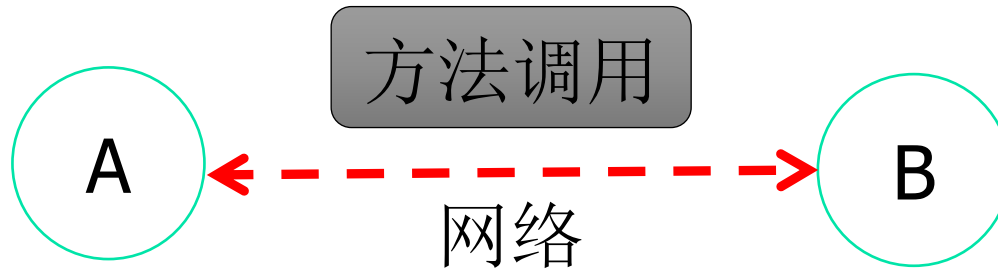




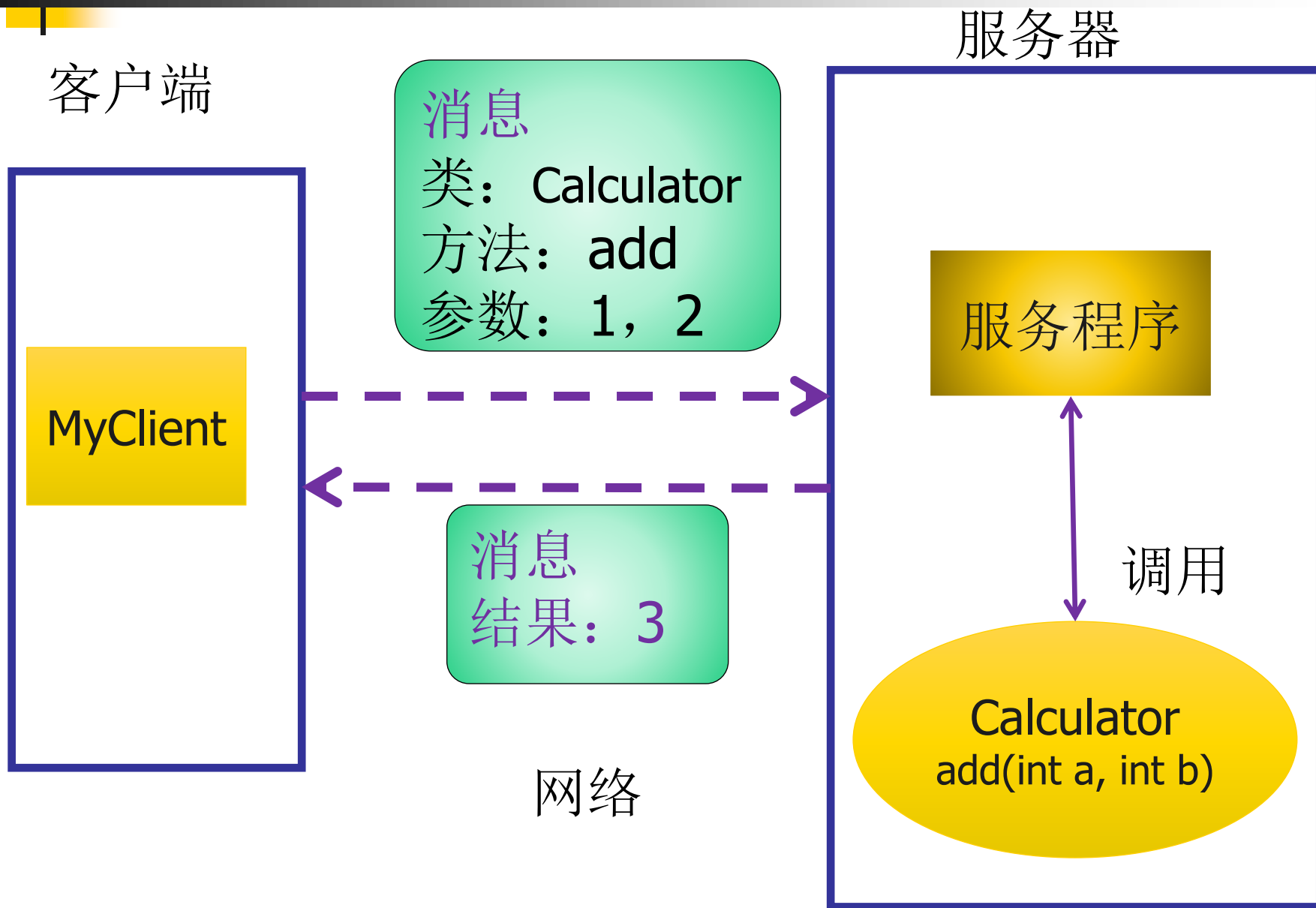
# WebService



类似于



### 3.1.4 Webservice基本原理

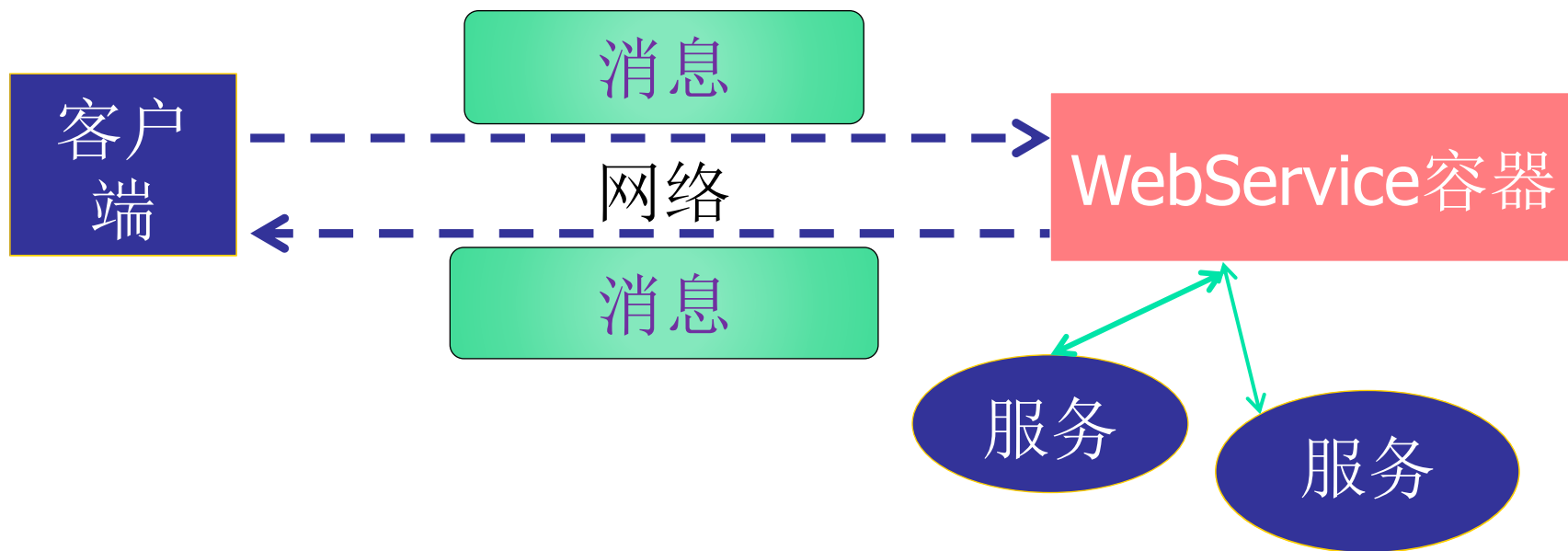


### 3.1.4 WebService基本原理

**客户端(Client):** 调用发起者，是一个程序

**服务(Service):** 被调用的程序

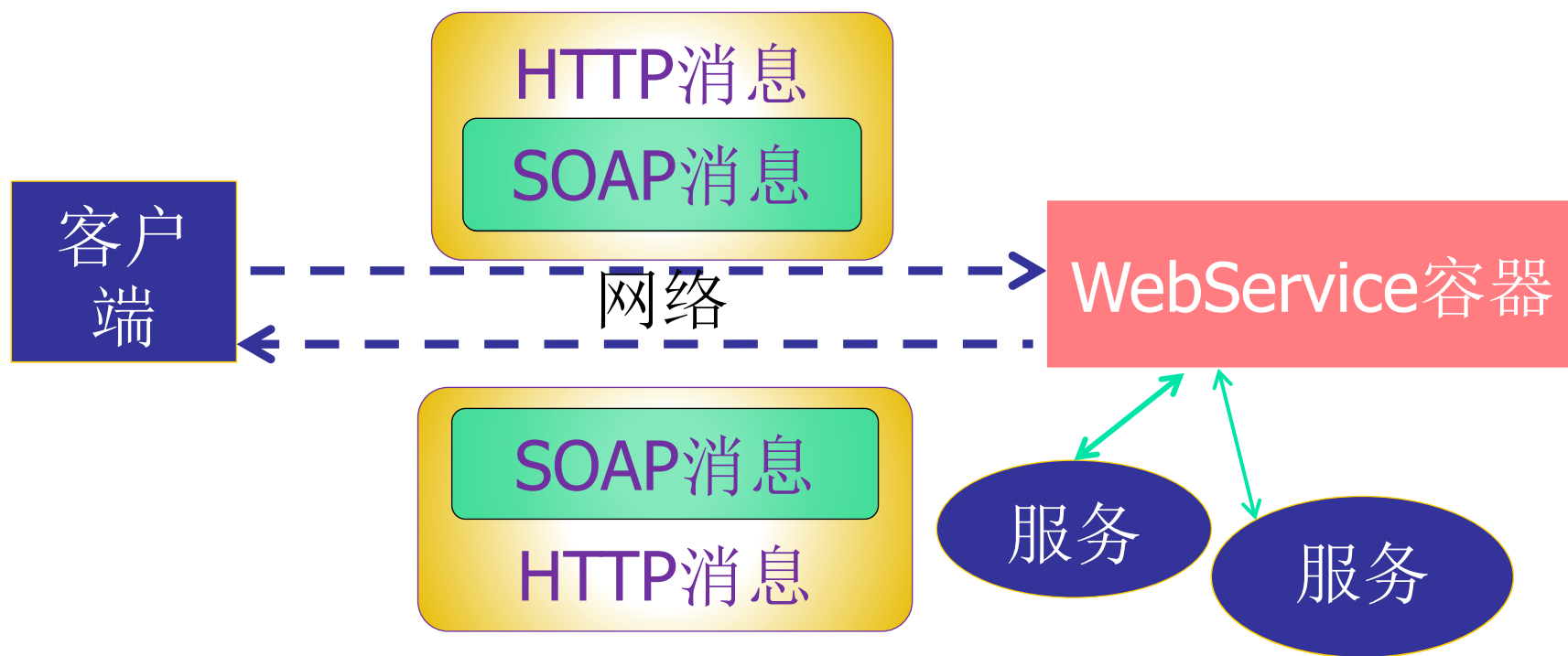
**WebService容器(Container):** 管理服务并代理调用的程序



### 3.1.4 WebService基本原理

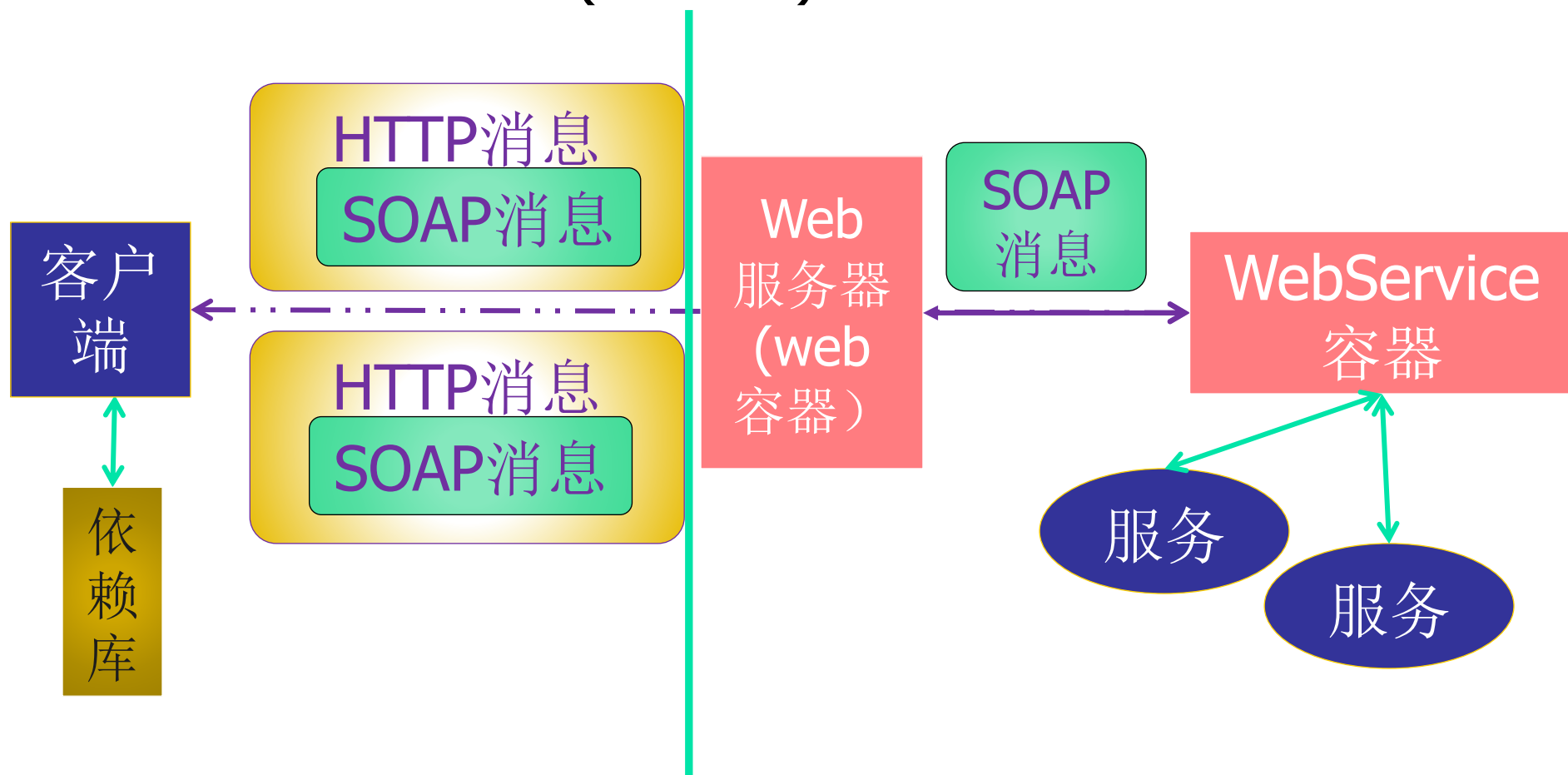
**SOAP消息：** 特定格式（包含调用所需信息或返回结果的字符串）

**如何传递消息：** 基于已有通用协议传递SOAP。（http）



### 3.1.4 WebService基本原理

通过Web服务器(Tomcat)提供WebService。





### 3.1.4 WebService基本原理

---

#### ■ Web服务的核心问题

如何表示数据类型 ——XML Schema

如何描述Web服务——WSDL

如何发现Web服务——UDDI

如何访问Web服务——SOAP

## 3.2 Web服务体系结构

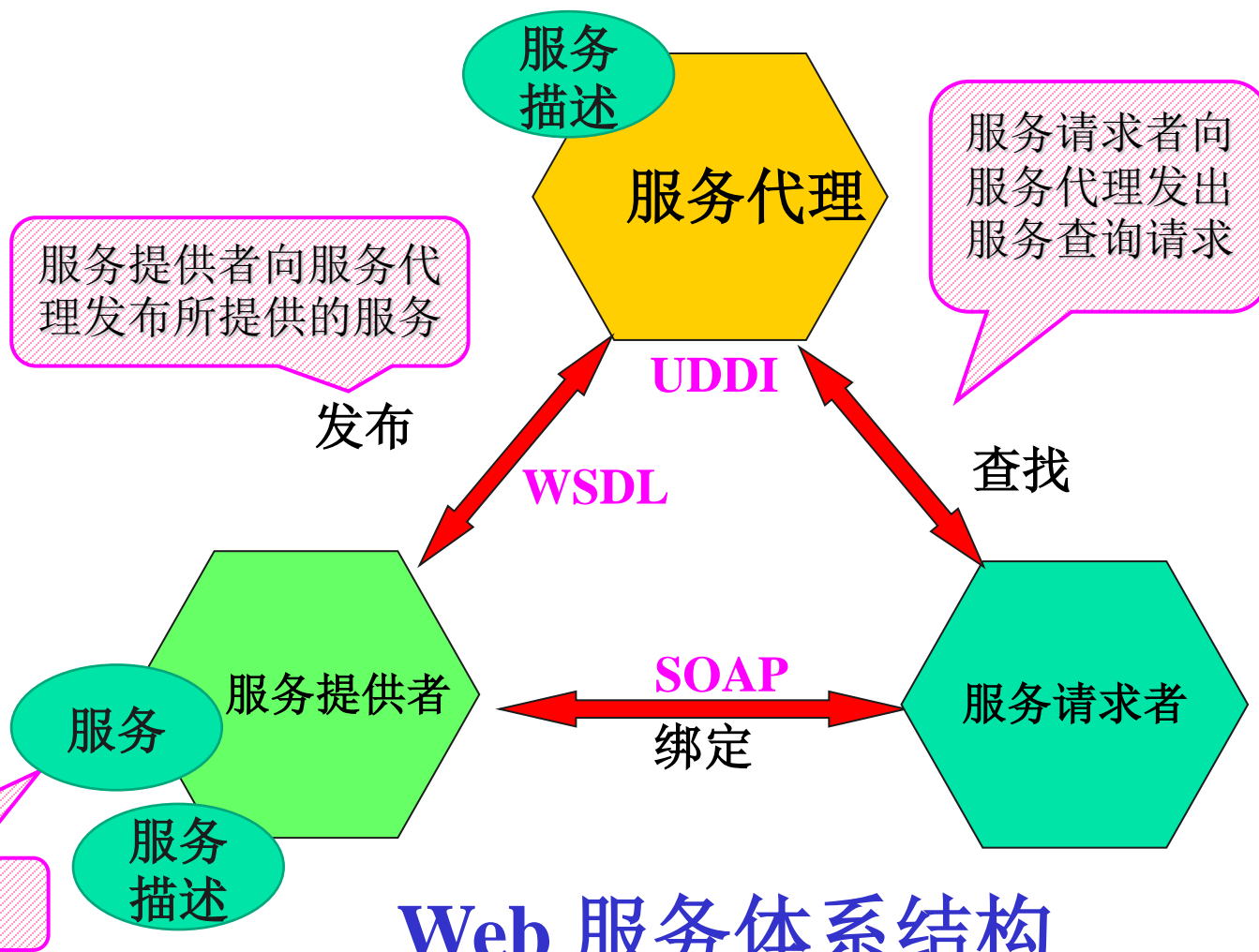
### 3.2.1 Web服务体系结构

三个Web服务组件:

- 服务提供者
- 服务代理
- 服务请求者

三个Web服务操作:

- 发布操作
- 查找操作
- 绑定操作



Web 服务体系结构



## 3.2 Web服务体系结构

### 3.2.2 Web服务体系结构—组件

#### ✚ 服务提供者

- 提供服务及维护注册表以使服务可用；

#### ✚ 代理

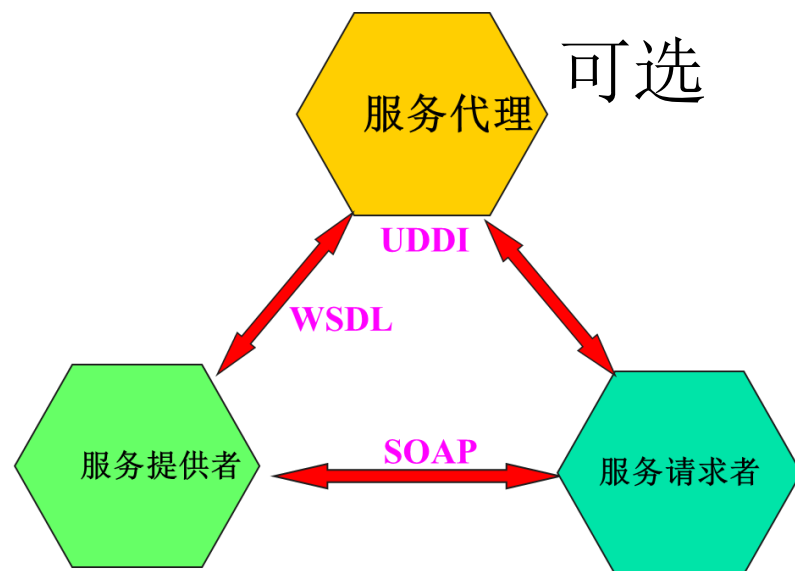
- 服务提供者与服务请求者的中介；
- 传统的代理是UDDI注册中心；

#### ✚ 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



## 3.2 Web服务体系结构

### 3.2.3 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。

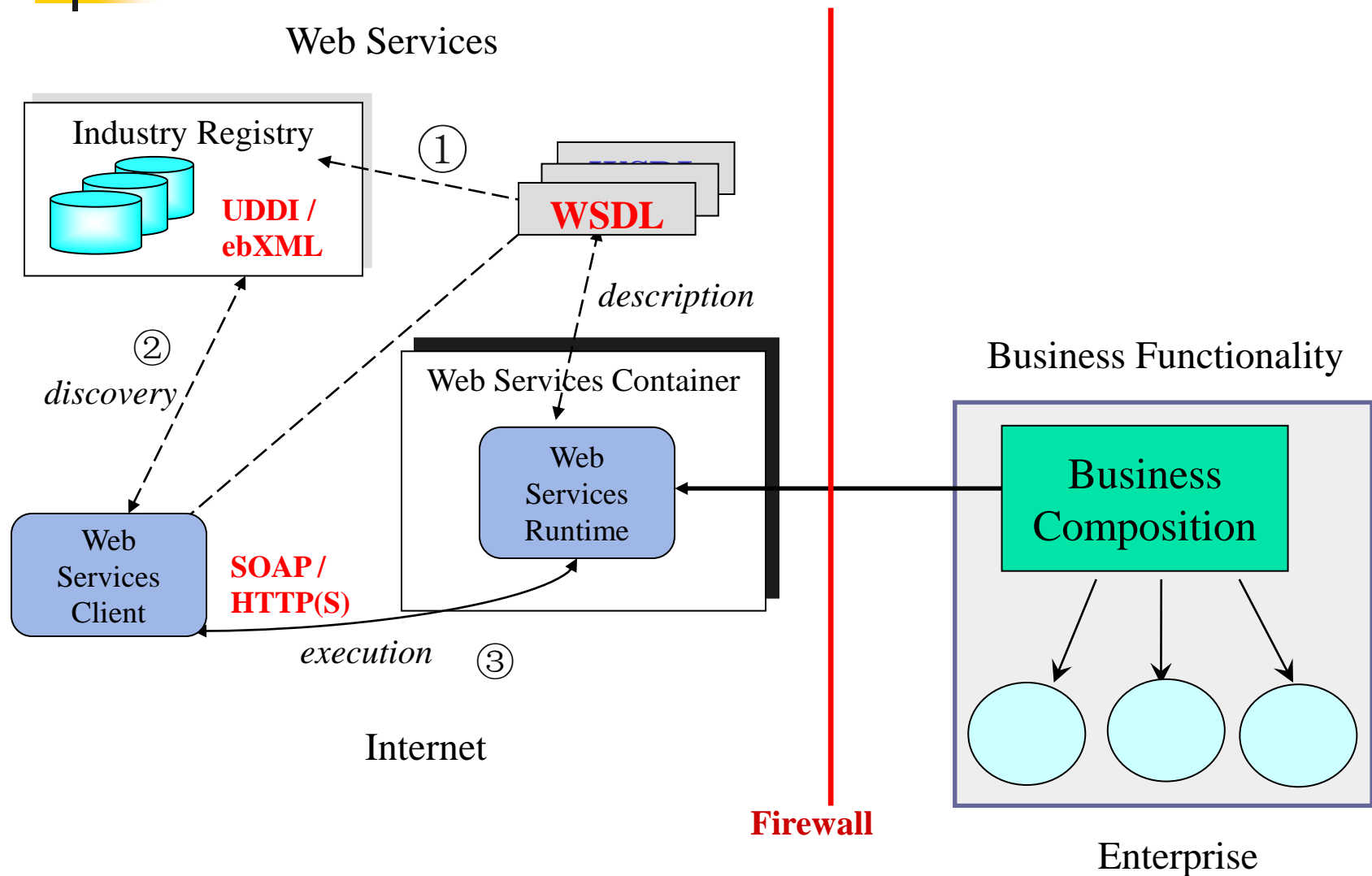
#### 查找 (**Find**) ,

- 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**)

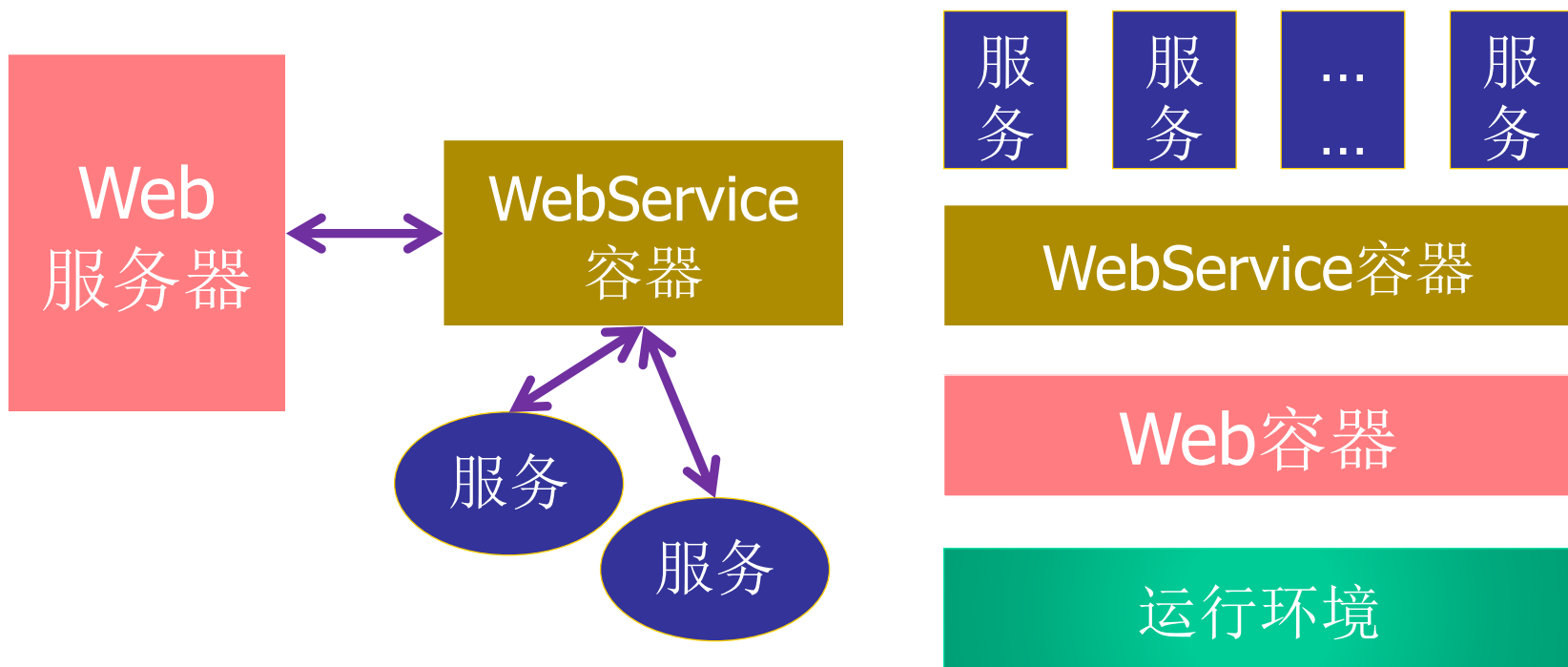
- 绑定操作发生在服务请求者和服务提供者之间。双方经过绑定后，请求者就可以访问和调用提供者所提供的服务。

## 3. 2Web服务体系结构



## 3. 2Web服务体系结构

### 3.2.4 WebService 实践



## 3. 2Web服务体系结构

### 3.2.5 环境搭建





## Tomcat

- Apache组织提供的开源web服务器，支持jsp、servlet
- 在WebService体系中的作用

在WebService体系中，tomcat负责接受http消息请求，根据地址将消息内容转发至相应的应用处理，将处理结果以http消息的形式返回至客户端



## AXIS

- Apache组织提供的开源服务容器，由多种语言版本
- AXIS在WebService体系中的作用

Axis接收到请求消息后，根据服务名查找配置文件得到实现类，再使用消息中提供的方法名、参数产生调用，调用完成得到结果，将结果封装成SOAP消息返回给tomcat



## 3. 2Web服务体系结构

---

### 3. 2. 6 例子

服务

```
package test;  
public class Test{  
  
    public String hello(String name){  
        return "Hello:" + name;  
    }  
  
}
```



### 3. 2. 6 例子

POST **/axis/services/ABC** HTTP/1.0

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

Accept: application/soap+xml, application/dime, multipart/related, text/\*

User-Agent: Axis/1.2RC2

Host: 127.0.0.1:8081

Cache-Control: no-cache

Pragma: no-cache

SOAPAction: ""

Content-Length: 406

```
<soapenv:Envelope >
```

```
  <soapenv:Body>
```

```
    <bello>
```

```
      <arg0 type="soapenc:string" >张三</arg0>
```

```
    </bello>
```

```
  </soapenv:Body>
```

```
</soapenv:Envelope>
```

### 3.2.6 例子

POST /axis/services/**ABC**

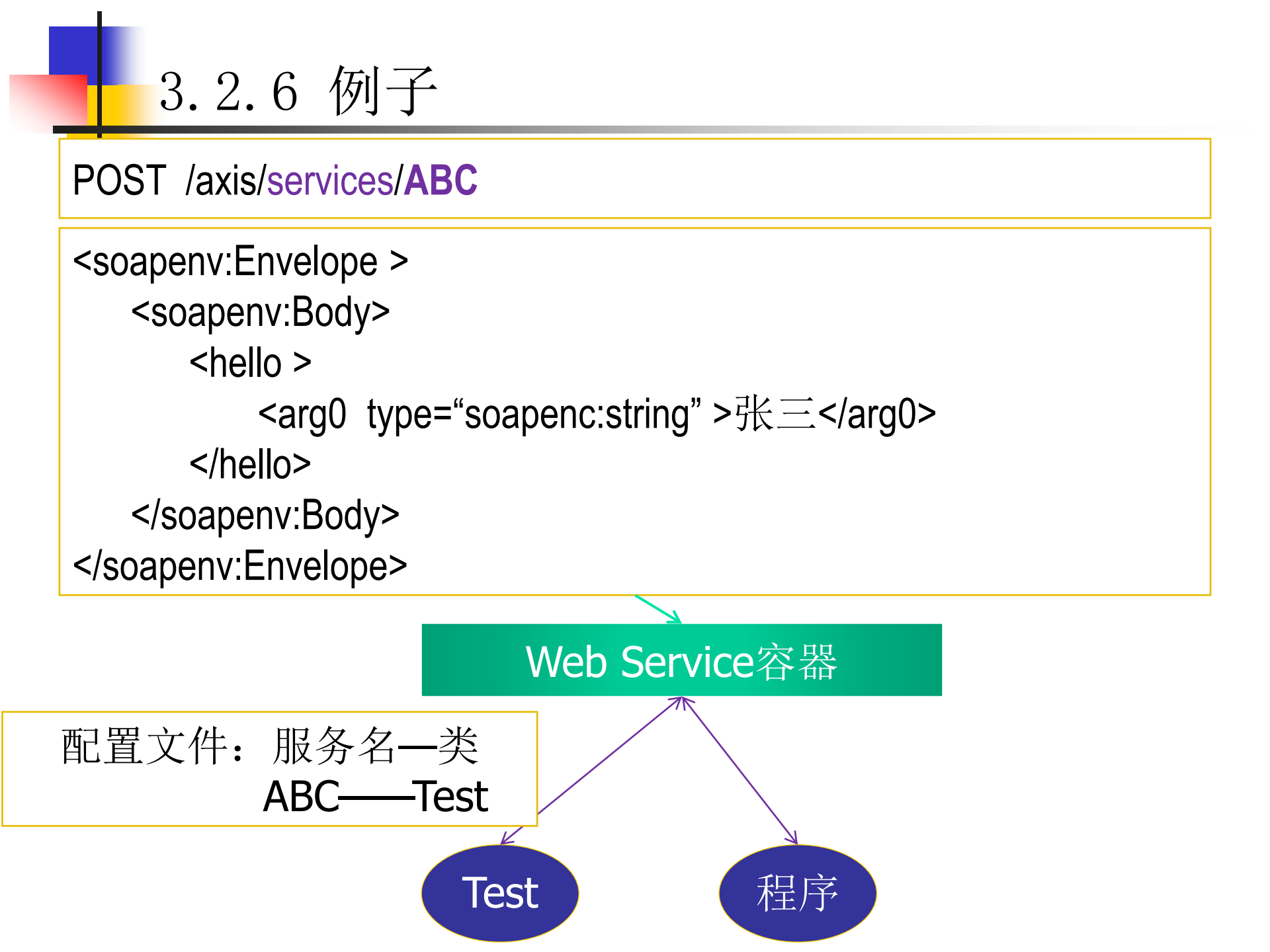
```
<soapenv:Envelope >  
  <soapenv:Body>  
    <hello >  
      <arg0 type="soapenc:string" >张三</arg0>  
    </hello>  
  </soapenv:Body>  
</soapenv:Envelope>
```

Web Service容器

配置文件：服务名—类  
ABC—Test

Test

程序





### 3. 2. 6 例子

---

HTTP/1.1 200 OK

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

Date: Thu, 22 May 2008 02:19:06 GMT

Server: Apache-Coyote/1.1

Connection: close

<soapenv:Envelope >

  <soapenv:Body>

    <helloResponse >

      <helloReturn xsi:type="soapenc:string" >**Hello:** 张三

      </helloReturn>

    </helloResponse>

  </soapenv:Body>

</soapenv:Envelope>



### 3.2.7 特点及优点


---

#### WebService技术的特点

- 基于标准、通用协议、语言不相关
- 广域、跨平台、屏蔽异构性
- 使用简单、易于扩展

#### WebService的优点

- 跨越防火墙的通讯、
- 应用程序集成、
- B2B的集成、软件
- 数据重用



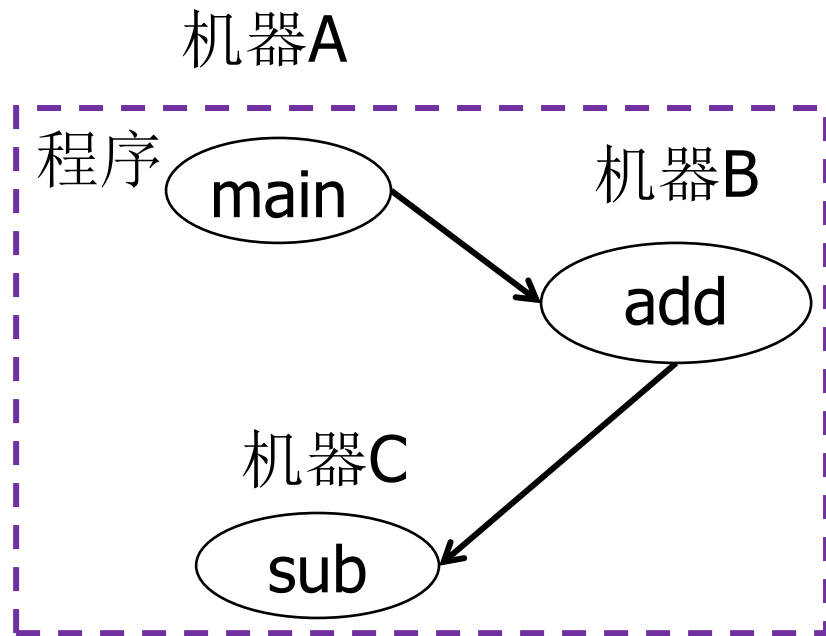
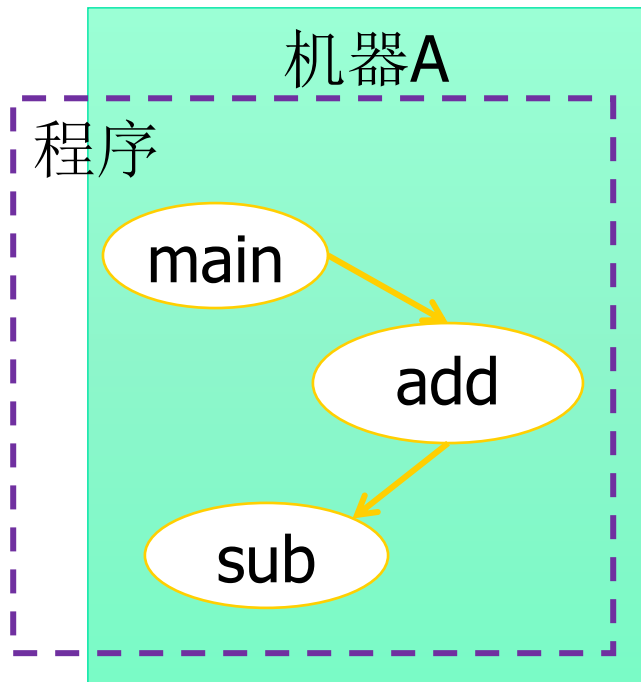
### 3.2.8 意义

---

- 信息化：异构资源接入
- 系统：面向服务的体系结构
- 编程：面向服务的编程模型
- 商业：面向服务的商业应用模式

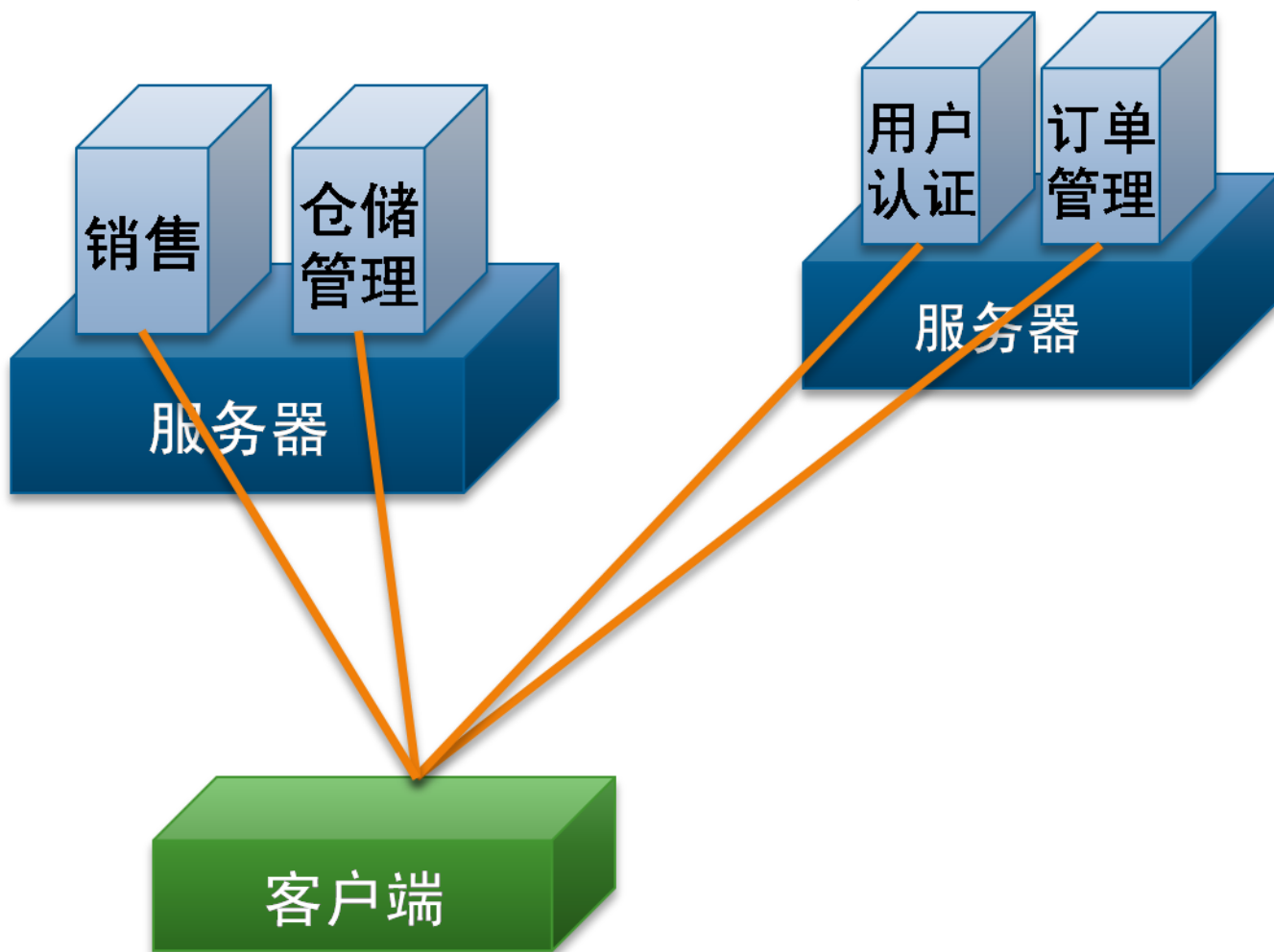
### 3.2.8 意义

## 面向服务的编程(SOA)



### 3.2.8 意义

## 新型的商业应用模式





## 3.3 Web服务的核心技术

---

### 3.3.1 Web服务的核心技术

- XML 可扩展标记语言
- WSDL Web服务描述语言
- SOAP 简单对象访问协议
- UDDI 统一描述、发现和集成协议





帮助客户端应用程序解析远  
程服务的位置

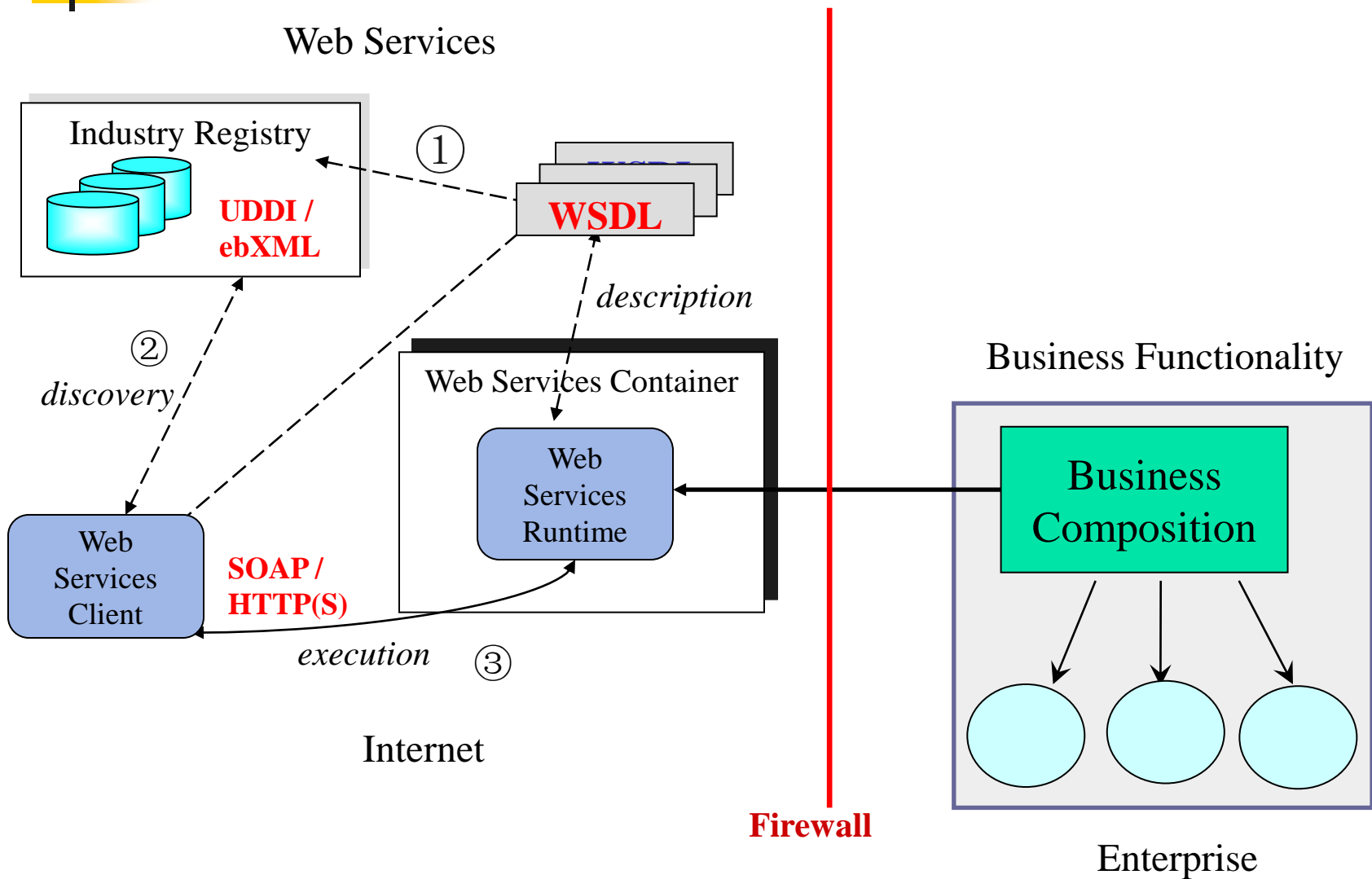
为客户端应用程序提供正确地与  
远程服务交互的  
描述信息

发现服务	UDDI
描述服务	WSDL
消息格式层	SOAP
编码格式层	XML: Schema
传输协议层	HTTP、TCP/IP、SMTP等

保证客户端与服务  
器在格式设置  
上的一致性

为客户端和服务端之间提供  
交互的网络通信协议

为客户端和服务端之间提供  
一个标准的、独立于平台的  
数据交换编码格式





### 3.3.2 XML

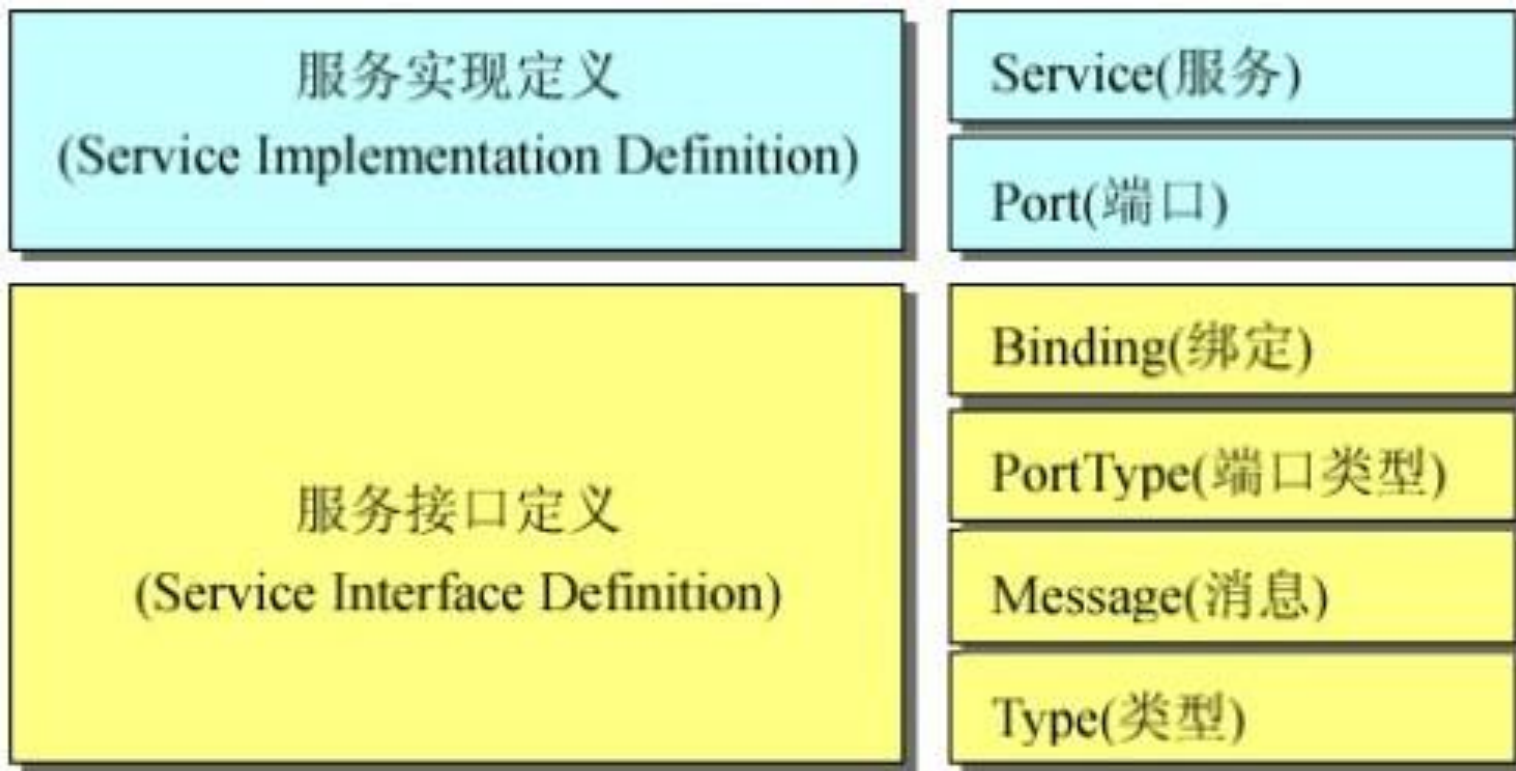
- XML(Extensible Markup Language)是 W3C(World Wide Web Consortium)的一个标准，它允许人们定制自己需要的标记。
- XML是提供与平台无关的数据类型和数据结构的描述，不用知道对方的系统是什么，只需要遵循在 XML Schema 中定义规范即可。
- XML内容与展现是分开的。通过对同一个文档采用不同的样式表 (stylesheet)，一个 XML 文档可以被展现成不同的形式。
- 2001年5月, W3C 确定了基于XML的数据类型系统标准，XML Schema。
- XML Schema 标准定义了一套标准的数据类型，并给出了一种语言扩展这套数据类型和描述XML文件中类型的结构。
- XSD (XML Schema Definition Language ) 是定义XML Schema的一种语法。
- Web Service把XSD作为其数据类型系统。

# 例子

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2006/06/addressing/wsam"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://service/" name="CalculatorService">
  <types>
    ...
  </types>
  <message name="add">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="addResponse">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <portType name="Calculator">
    <operation name="add">
      <input wsam:Action="http://service/Calculator/addRequest" message="tns:add"/>
      <output wsam:Action="http://service/Calculator/addResponse" message="tns:addResponse"/>
    </operation>
  </portType>
  <binding name="CalculatorPortBinding" type="tns:Calculator">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="add">
      ...
    </operation>
  </binding>
  <service name="CalculatorService">
    <port name="CalculatorPort" binding="tns:CalculatorPortBinding">
      <soap:address location="http://127.0.0.1:456/calculator"/>
    </port>
  </service>
</definitions>
```

### 3.3.3 WSDL– Web服务描述语言

#### ■ 基于WSDL的Web服务描述





### 3.3.3 WSDL– Web服务描述语言

---

- WSDL (Web Service Description Language)是一个基于XML语法的描述语言，用于描述：
  - 类型 (Type)、
  - 消息 (Message)、
  - 接口 (称为 端口类型: PortTypes )
    - ✓ 操作 (Operation)
  - 绑定
    - ✓ 协议
  - Web Service
    - ✓ 服务名
    - ✓ 所处的位置

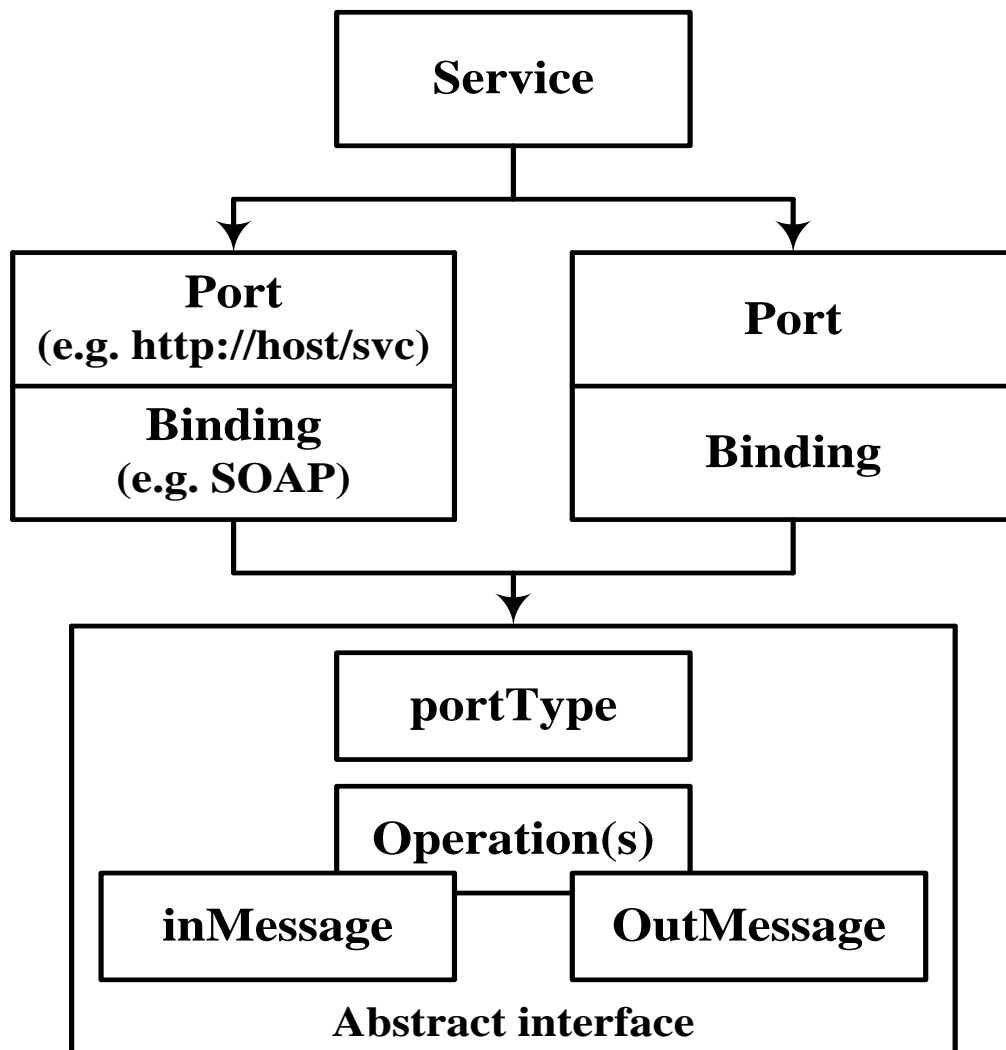


### 3.3.3 WSDL– Web服务描述语言

- 把 Web 服务描述成一组运行在消息上的端点集。
- 消息描述了客户端和服务之间的通信（通过交换的数据类型来描述）。
- 操作包括输入和输出消息。
- PortTypes 包括一组操作。而且，PortTypes 被约束在某些协议上，这称为绑定（binding）。
- 一些最新的开发工具：
  - 既能根据你的Web Service生成WSDL文档；
  - 又能导入WSDL文档，生成调用相应Web Service的客户代码（帮我们生成本地代理，再通过本地代理来访问 **webservice**： java 自带的wsimport 命令）。

### 3.3.3 WSDL– Web服务描述语言

#### ■ WSDL模型





```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.1"
xmlns:wspl_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2006/06/addressing/wsam"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://service/" name="CalculatorService">
```

```
<types>
...
</types>
```

} 类型

```
<message name="add">
  <part name="parameters" element="tns:add"/>
</message>
<message name="addResponse">
  <part name="parameters" element="tns:addResponse"/>
</message>
```

} 消息

```
<portType name="Calculator">
  <operation name="add">
    <input wsam:Action="http://service/Calculator/addRequest" message="tns:add"/>
    <output wsam:Action="http://service/Calculator/addResponse" message="tns:addResponse"/>
  </operation>
</portType>
```

} 接口

```
<binding name="CalculatorPortBinding" type="tns:Calculator">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="add">
    ...
  </operation>
</binding>
```

} 绑定的协议

```
<service name="CalculatorService">
  <port name="CalculatorPort" binding="tns:CalculatorPortBinding">
    <soap:address location="http://127.0.0.1:456/calculator"/>
  </port>
</service>
</definitions>
```

} 服务

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.1"
xmlns:wspl_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2004/08/wsaddressing"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://service/" name="CalculatorService">
```

WSDL一说明服务在哪里，如何调用，其实就是一个使用说明书

```
<types>
...
</types>
<message name="add">
  <part name="parameters" element="tns:add"/>
</message>
<message name="addResponse">
  <part name="parameters" element="tns:addResponse"/>
</message>
<portType name="Calculator">
  <operation name="add">
    <input wsam:Action="http://service/Calculator/addRequest" message="tns:add"/>
    <output wsam:Action="http://service/Calculator/addResponse" message="tns:addResponse"/>
  </operation>
</portType>
<binding name="CalculatorPortBinding" type="tns:Calculator">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="add">
    ...
  </operation>
</binding>
<service name="CalculatorService">
  <port name="CalculatorPort" binding="tns:CalculatorPortBinding">
    <soap:address location="http://127.0.0.1:456/calculator"/>
  </port>
</service>
</definitions>
```

3、web服务所提供的方法一即服务

2、web服务的名称

1、web服务的地址



### 3.3.3 WSDL– Web服务描述语言

---

```
package test;
import service.*;

public class MyClient {

    public static void main(String[] args) {
        CalculatorService service = new CalculatorService();
        Calculator calculator =service.getCalculatorPort();

        int result=calculator.add(1,2);//注意服务器: public int add(int x,int y);
        System.out.println(result);

    }
}
```



### 3.3.4 : SOAP

- SOAP (Simple Object Access Protocol)是一个基于XML、用于分散或分布式的环境中，进行消息传递和远程方法调用的简单协议。
- SOAP = 在HTTP的基础上+XML数据。
- SOAP主要由以下三部分组成：
  - **SOAP信封(envelop)**，定义了一个整体的SOAP消息表示框架，描述交互消息中的内容是什么，是谁发送的，谁应当接受并处理它，以及这些处理操作是可选的还是必须的，等等。
    - ▣ **Body** – 必须的。在body部分，包含要执行的服务器的方法，和发送到服务器的数据。
  - **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实现数据串行化。
  - **SOAP RPC的表示(RPC representation)**，定义了一个用于表示远端方法调用和响应的约定，例如如何使用HTTP或SMTP协议与SOAP**绑定 (binding)**。

## SOAP协议的范本：—请求示例：

以下发出HTTP请求，向服务器发送的是XML数据！

```
POST /WebServices/MobileCodeWS.asmx HTTP/1.1
Host: webservice.webxml.com.cn
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://WebXml.com.cn/getMobileCodeInfo"
```

1、因为是在HTTP上发数据，  
所以必须先遵循HTTP协议

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getMobileCodeInfo xmlns="http://WebXml.com.cn/">
      <mobileCode>string</mobileCode>
      <userID>string</userID>
    </getMobileCodeInfo>
  </soap:Body>
</soap:Envelope>
```

2、XML部分即SOAP协议，必须包含  
Envelope元素和Body元素。



## SOAP协议：一响应示例：

响应的信息，同发送信息一样，先必须是HTTP协议，然后再遵循SOAP协议。

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

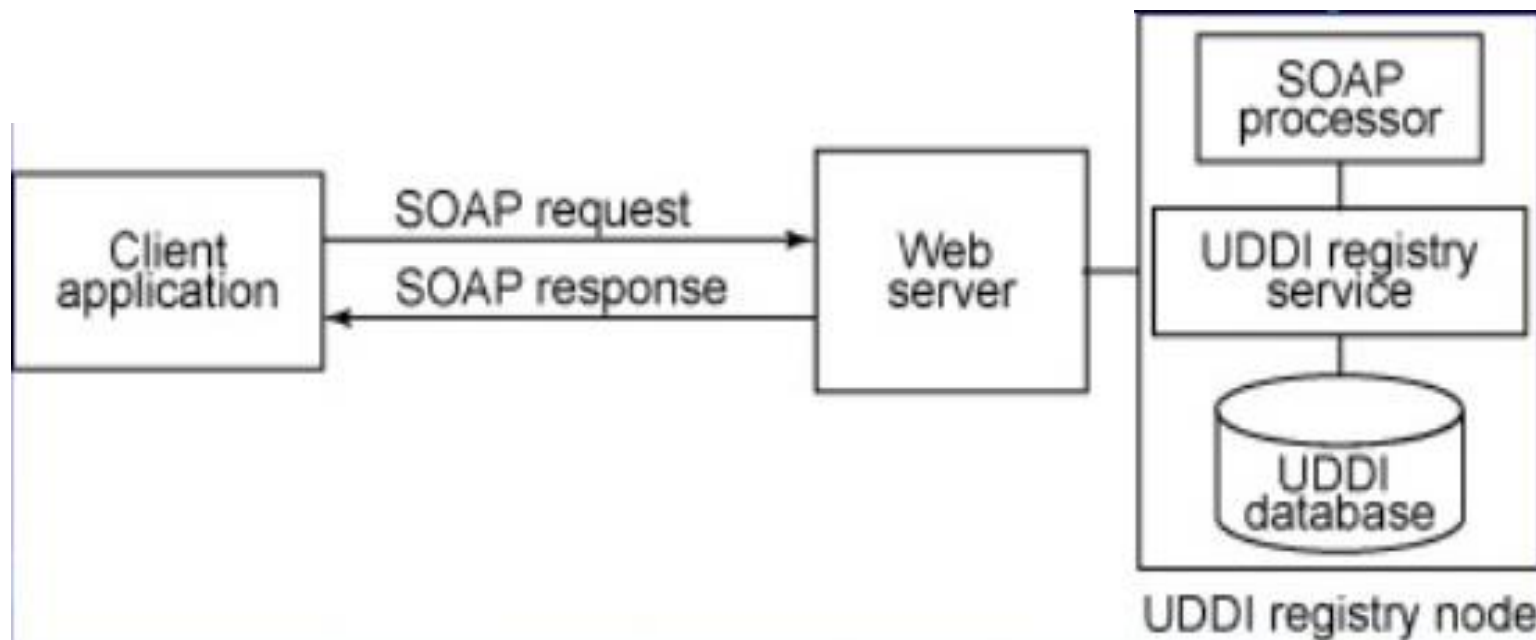
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getMobileCodeInfoResponse xmlns="http://WebXml.com.cn/">
      <getMobileCodeInfoResult>string</getMobileCodeInfoResult>
    </getMobileCodeInfoResponse>
  </soap:Body>
</soap:Envelope>
```

### 3.3.5 统一描述、发现和集成协议

#### UDDI, Universal Description, Discovery and Integration

■ 是一种目录服务，企业可以使用它对Web服务进行注册和搜索。

■ UDDI注册原理





### 3.3.6 小结

---

■在Web服务中：

XML是数据的格式

SOAP是调用Web服务的协议

WSDL是描述Web服务

UDDI是Web服务登记、查找和利用的组合

四个方面组成了整个Web服务的架构。





## 3.4 例子

---

### 3.4.1 发布WebService服务

#### Calculator.java

注：

- (1) @WebService: 注解，指定将类发布成一个WebService.
- (2) Endpoint: 此类为**端点服务类**，它的publish方法用于将一个经添加了@WebService注解的对象绑定到一个地址的端口上。
- (3) 运行以上程序进行发布。

### 3.4.2 查看wsdl地址: <http://127.0.0.1:456/calculator?wsdl>

只要在客户端浏览器能看到此WSDL文档, 说明服务发布成功

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata
xmlns:tns="http://service/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" tar
  <types>
    <xsd:schema>
      <xsd:import namespace="http://service/" schemaLocation="http://127.0.0.1:456/calculator?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="add">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="addResponse">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <portType name="Calculator">
    <operation name="add">
      <input wsam:Action="http://service/Calculator/addRequest" message="tns:add"/>
      <output wsam:Action="http://service/Calculator/addResponse" message="tns:addResponse"/>
    </operation>
  </portType>
  <binding name="CalculatorPortBinding" type="tns:Calculator">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="add">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="CalculatorService">
    <port name="CalculatorPort" binding="tns:CalculatorPortBinding">
      <soap:address location="http://127.0.0.1:456/calculator"/>
    </port>
  </service>
</definitions>
```

### 3.4.3 客户端访问发布的服务

(1) 根据WSDL文档生成客户端访问服务器端服务所需的代码:

**wsimport**: 是JDK自带的, 可以根据WSDL文档生成客户端调用代码的工具。无论服务器端WebService使用什么语言编写的, 都将在客户端生成Java代码。

解析WSDL地址生成源码到文件夹中:

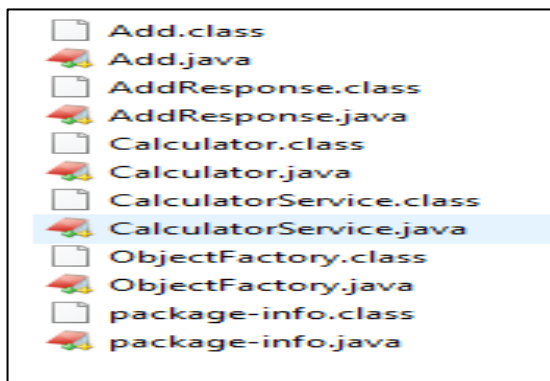
**wsimport -s . http://127.0.0.1:456/calculator?wsdl**

```
D:\save\java\webservice\client>wsimport -s . http://127.0.0.1:456/calculator?wsdl
parsing WSDL...

Generating code...

Compiling code...
```

在文件夹下生成如下包及类:



## (2)客户端借助生成的代码访问远程服务器上提供的WebService

### ■ 创建访问webservice服务的类：MyClient.java

```
package test;
import service.*;

public class MyClient {

    public static void main(String[] args) {
        CalculatorService service = new CalculatorService();
        Calculator calculator =service.getCalculatorPort();

        int result=calculator.add(1,2);//注意服务器:  public int add(int x,int y);
        System.out.println(result);

    }
}

//注意: 编译client端代码应该在client中, 不能在test目录中编译, 否则找不到CalculatorService类
```

说明: 此程序需要在test目录的上级目录编译

```
D:\save\java\webbservice\client>javac test/*.java
```

### ■ 运行结果:

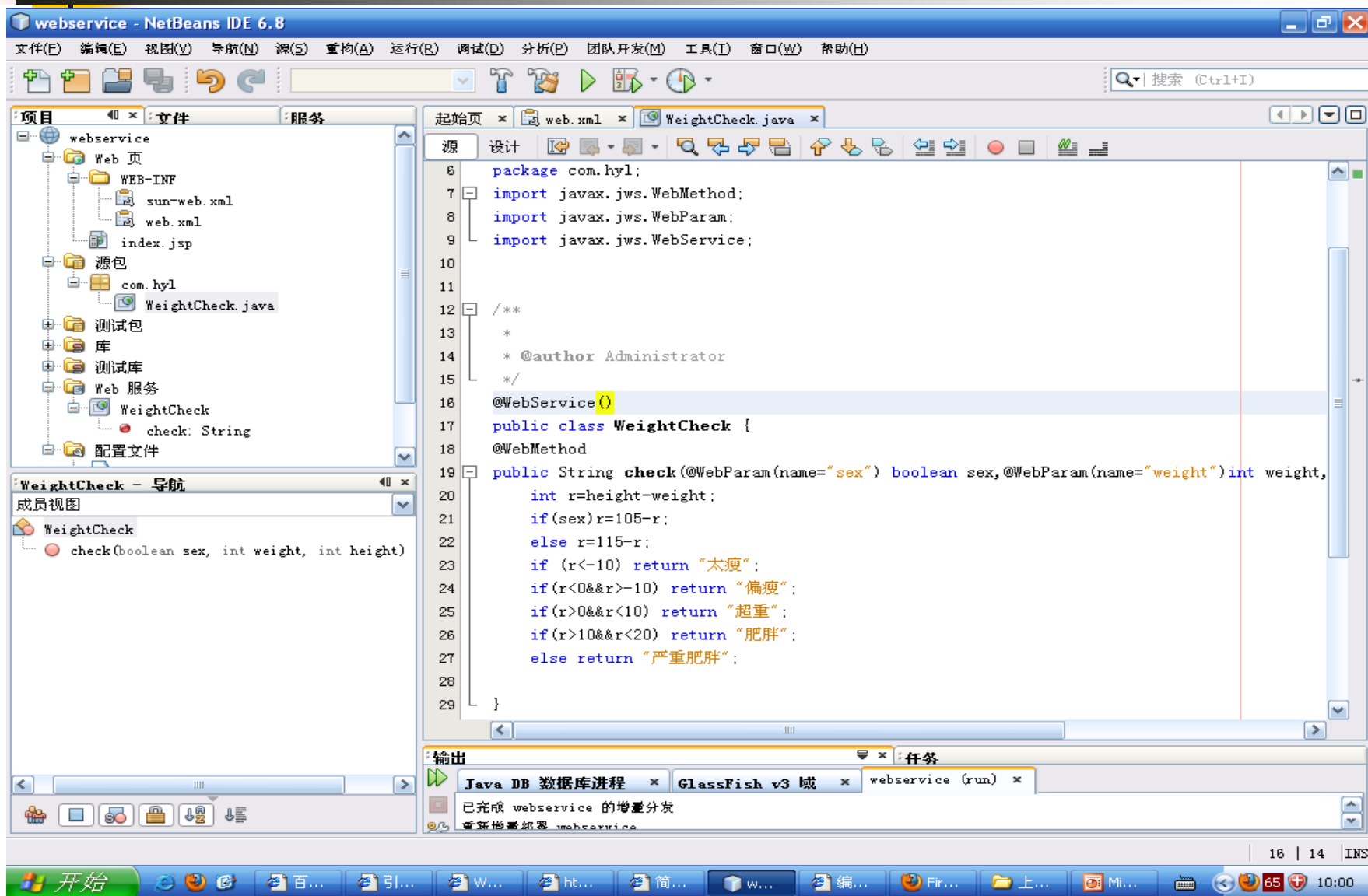
```
D:\save\java\webbservice\client>java test.MyClient
3
```



---

## Web service例2: (jsp访问webservice)

# Web service服务端的编程实现



# Web Service服务端在glassfish平台的部署

编辑应用程序 - Microsoft Internet Explorer

文件(E) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)

地址(D) http://localhost:4848/web/apps/editWar.jsf?appName=webservice 转到

用户: admin 域: domain1 服务器: localhost

Sun GlassFish™ Enterprise Server v3

有 56 个更新可用。

树

- 日常任务
  - 注册
  - GlassFish 新闻
  - Enterprise Server
  - 应用程序
    - webservice
  - 生命周期模块
  - 资源
    - JDBC
    - 连接器
    - 资源适配器配置
    - JMS 资源
    - JavaMail 会话
    - JNDI
  - 配置
    - JVM 设置
    - 日志程序设置
    - Web 容器
    - EJB 容器
    - Ruby 容器

名称: webservice

状态: ☒ 已启用

虚拟服务器: server

将 Internet 域名与物理服务器关联

上下文根: /webservice

相对于服务器的基本 URL 的路径

描述:

位置: file:/C:/Documents%20and%20Settings/Administrator/桌面/webservice/build/web/

库:

模块和组件 (4)

模块名称	引擎	组件名称	类型	操作
webservice	[web, webservicess]	-----	-----	启动
webservice		jsp	Servlet	
webservice		default	Servlet	
webservice		WeightCheck	Servlet	查看端点

完毕

本地 Intranet

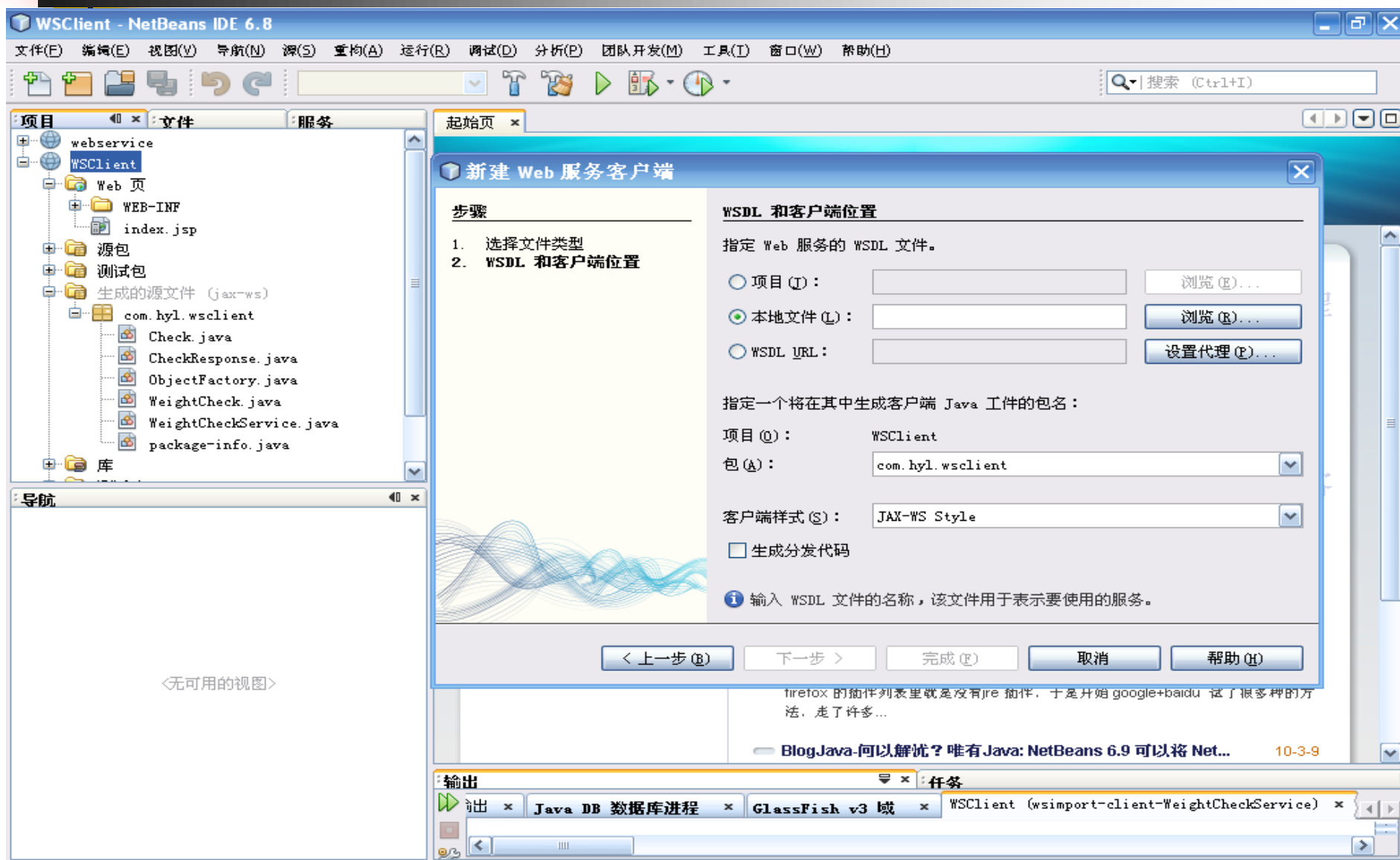
开始 百... 引... W... ht... 简... w... 编... Fir... 上... Mi... 64 10:01

WSDL文件的生成(注意地址栏地址，服务端和客户端处于两个不同项目,还可以是分布式的两个项目，当然也可处同一个项目中，那样的话做成web service意义不大)





# Web Service客户端的编程(需指出web服务在哪)



# 客户端的invoke.jsp对服务端的引用

WSClient - NetBeans IDE 6.8

文件(F) 编辑(E) 视图(V) 导航(N) 源(S) 重构(A) 运行(R) 调试(D) 分析(P) 团队开发(M) 工具(T) 窗口(W) 帮助(H)

搜索 (Ctrl+I)

项目: webservice, WSClient, Web 页, WEB-INF, index.jsp, invoke.jsp, 源包, 测试包, 生成的源文件 (jax-ws), com.hyl.wsclient, Check.java, CheckResponse.java, ObjectFactory.java, WeightCheck.java, WeightCheckService.java, package-info.java

invoke.jsp - 导航: html, head, meta, title, body, hr, hr

起始页 x invoke.jsp x index.jsp x

```
18 // TODO initialize WS operation arguments here
19 boolean sex = false;
20 int weight = 48;
21 int height = 165;
22 String xb=request.getParameter("sex");
23 if(xb.equals("true")) sex=true;
24 weight=Integer.parseInt(request.getParameter("weight"));
25 height=Integer.parseInt(request.getParameter("height"));
26 %>
27 <!-- start web service invocation --><hr/>
28 <%
29 try {
30     com.hyl.wsclient.WeightCheckService service = new com.hyl.wsclient.WeightCheckService();
31     WeightCheckPort port = service.getWeightCheckPort();
32
33     // TODO call web service operation here
34     java.io.PrintWriter out = response.getWriter();
35     out.println("你的身高为"+height+"<br>");
36     out.println(port.check(sex, weight, height));
37 } catch (Exception e) {
38     // TODO handle exception here
39 }
40 %>
```

格式化代码 (Alt+Shift+F)  
运行文件(E) (Shift+F6)  
新建监视(W)...  
开启/关闭行断点(I) (Ctrl+F8)  
剪切(I)  
复制(Y)  
粘贴(P) (Ctrl+V)  
属性(P)  
代码折叠(C)  
选择范围

Web 服务客户端资源 调用 Web 服务操作...

输出: Java DB 数据库进程, GlassFish v3 域, WSClient (wsimport-client-WeightCheckService)

过滤器: [ ] [ ] [ ]

# 客户端的invoke.jsp对服务端的引用

WSClient - NetBeans IDE 6.8

文件(F) 编辑(E) 视图(V) 导航(N) 源(S) 重构(A) 运行(R) 调试(D) 分析(P) 团队开发(M) 工具(T) 窗口(W) 帮助(H)

搜索 (Ctrl+I)

项目 文件 服务

webservice  
WSClient  
Web 页  
WEB-INF  
index.jsp  
invoke.jsp  
源包

选择要调用的操作

可用的 Web 服务引用:

WSClient  
WeightCheckService  
WeightCheckService  
WeightCheckPort  
check

起始页 x invoke.jsp x index.jsp x

```
18 // TODO initialize WS operation arguments here
19 boolean sex = false;
20 int weight = 48;
21 int height = 165;
String xb=request.getParameter("sex");
if(xb.equals("true")) sex=true;
weight=Integer.parseInt(request.getParameter("weight"));
height=Integer.parseInt(request.getParameter("height"));

-- start web service invocation --%><hr/>

{
com.hyl.wsclient.WeightCheckService service = new com.hyl.wsclient.WeightCheckSer
com.hyl.wsclient.WeightCheck port = service.getWeightCheckPort();

// TODO process result here
java.lang.String result = port.check(sex, weight, height);
out.println("你的体重为"+ weight+",你的身高为"+height+"<br>");
out.println("测试结果为: "+result);
} catch (Exception ex) {
// TODO handle custom exceptions here
```

确定 取消

DB 数据库进程 x GlassFish v3 域 x WSClient (wsimport-client-WeightCheckService) x

开始 Macromedia Dr... WSClient - Net... JSP Page - Mic... 上课课件 Microsoft Powe... 13:56

项目 文件 服务

webservice  
WSClient  
Web 页  
WEB-INF  
index.jsp  
invoke.jsp  
源包  
测试包  
生成的源文件 (jax-ws)  
com.hyl.wsclient  
Check.java  
CheckResponse.java  
ObjectFactory.java  
WeightCheck.java  
WeightCheckService.java  
package-info.java

index.jsp - 导航

html  
head  
meta  
title  
body  
h1  
form id=info  
label  
select id=sex  
option  
option  
label  
input id=weight  
label

起始页 x invoke.jsp x index.jsp x

```
html body form
3      Created on : 2010-5-4, 13:34:47
4      Author    : Administrator
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9        "http://www.w3.org/TR/html4/loose.dtd">
10
11      <html>
12      <head>
13        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14        <title>体重超标测试</title>
15      </head>
16      <body>
17        <h1>体重超标测试</h1>
18        <form action="invoke.jsp" method="post" name="info" id="info">
19          性别:
20          <label>
21            <select name="sex" id="sex">
22              <option>男</option>
23              <option>女</option>
24            </select>
25          </label>
26          体重:
```

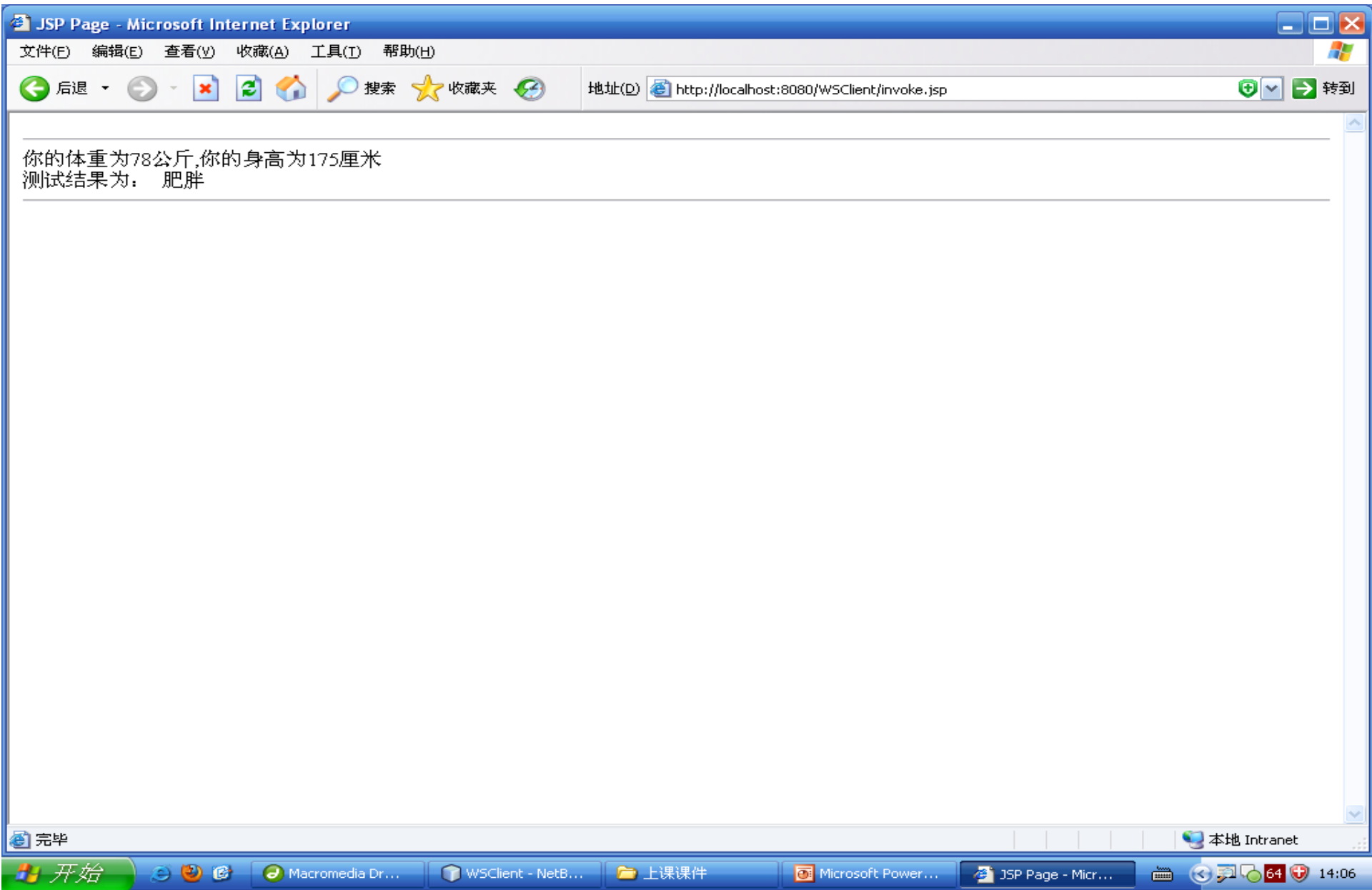
输出 任务

检索器输出 x Java DB 数据库进程 x GlassFish v3 域 x WSClient (run) x

# 体重超标测试

性别: 男 体重: 78 公斤 身高: 175 厘米 测试

# 调用的结果（试想一下网页上的天气预报就是web service实现）





---

thanks