

Arsitektur Aplikasi Perangkat Enterprise #10



Antonius Rachmat C, S.Kom, M.Cs



"Today, the principal use of the World Wide Web is for interactive access to documents and applications.

In almost all cases, such access is by human users, typically working through Web browsers, audio players, or other interactive front-end systems.

The Web can grow significantly in power and scope if it is extended to support communication between applications, from one program to another."

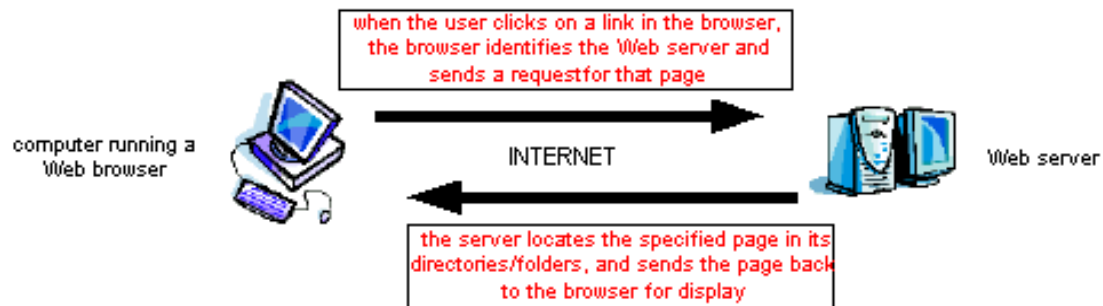
From the W3C XML Protocol Working Group Charter

World Wide Web

□ the Web is the world's largest client/server system

communication occurs via message passing

- within browser, select URL of desired page
- browser requests page from server
- server responds with message containing
 - type of page (HTML, gif, pdf, zip, ...)
 - page contents
- browser uses type info to correctly display page
- if page contains other items (images, applets, ...), browser must request each separately



Yesterday...

1st & 2nd Generation Web Apps



Web Server



Browser

1-1 correspondence
of page to file



Web Server



Browser

“Dynamic Pages”

Now, Web Service...

- ❑ **Web Services can convert your applications into Web-applications.**
 - By using Web services, your application can publish its function or message to the rest of the world.
- ❑ **Web Services can be used by other applications.**
 - Ex: with Web services your accounting department's Win 2k servers can connect with your IT supplier's UNIX server.
- ❑ **The basic Web Services platform is XML + HTTP.**
 - Web services uses XML to encode and decode your data and SOAP (based on HTTP) to transport it.

Pengertian-pengertian

- ❑ **Generic:** Any application accessible to other applications over the Web
 - Very open definition, almost any URL becomes a “Web service” under this definition
- ❑ **UDDI consortium:** Web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces
 - Better, but still not specific enough
- ❑ **W3C:** A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols
 - Slightly better, we now know that URIs and XML are involved (somehow) and that services are similar to components with interfaces that can be defined, described, and discovered

Pengertian-pengertian

- **Wikipedia:** According to the W3C, a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format such as WSDL. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.
 - Much more specific

Pengertian-pengertian

- ❑ "Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. ... Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service." (**from IBM web service tutorial**)
- ❑ A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP. (**from Java Web Service**)
- ❑ A Web service is a new technology that you can use to create platform-independent applications. You can develop a Web service by using languages and platforms that adhere to a standard set of technologies. (**from Web Service Professional Projects**)

WS Example

- ❑ <http://www.google.com/apis/>
- ❑ <http://teraserver.microsoft.net/TerraService.asmx>
- ❑ <http://www.xmethods.net>
- ❑ <http://soap.amazon.com/schemas2/AmazonWebServices.wsdl>
- ❑ <http://api.google.com/GoogleSearch.wsdl>

Example: A Google Client

- ❑ Create a local proxy class, instantiate, and invoke
- ❑ The proxy class **MyService** generated from a WSDL file, an XML-encoded service description

```
Dim MyLicenseKey As String ' Variable to Store the License Key
' Declare variable for the Google search service
Dim MyService As com.google.api.GoogleSearchService = New _
    com.google.api.GoogleSearchService
' Declare variable for the Google Search Result
Dim MyResult As com.google.api.GoogleSearchResult
' Please Type your license key here
MyLicenseKey = "tGCTJkYos3YItLYzI9Hg5quBRY8bGqiM"
' Execute Google search on the text enter and license key
MyResult = MyService.doGoogleSearch(MyLicenseKey, _
    TextBox1.Text, 0, 1, False, "", False, "", "", "")
' output the total Results found
Label2.Text = "Total Found : " & _
    CStr(MyResult.estimatedTotalResultsCount)
```

Outline Architecture

My Desktop
Windows

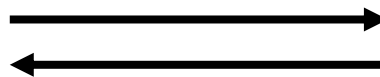
```
MyService s =  
new  
com.google.api.GoogleSearchService();
```



Implementation via
proxy class and
HTTP transport

The Internet
TCP/IP

SOAP
Request



SOAP
Response

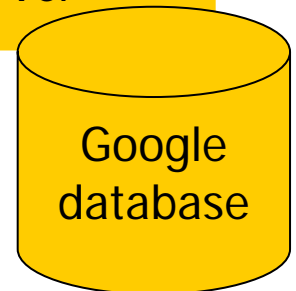
Vendor-neutral
XML-encoding
over HTTP

Google.com/apis
Unix/Linux?

```
[WebMethod]  
... doGoogleSearch(myKey, q) ...
```



Implementation via
WebService classes
in Web Server



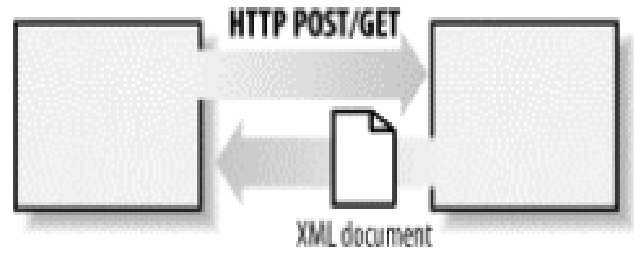
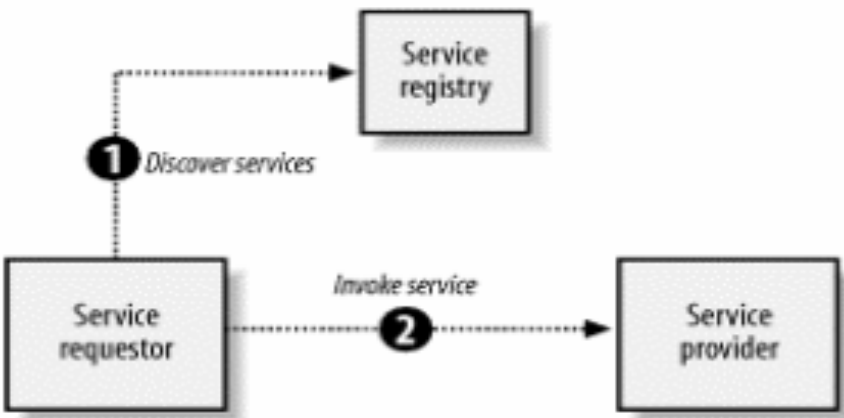
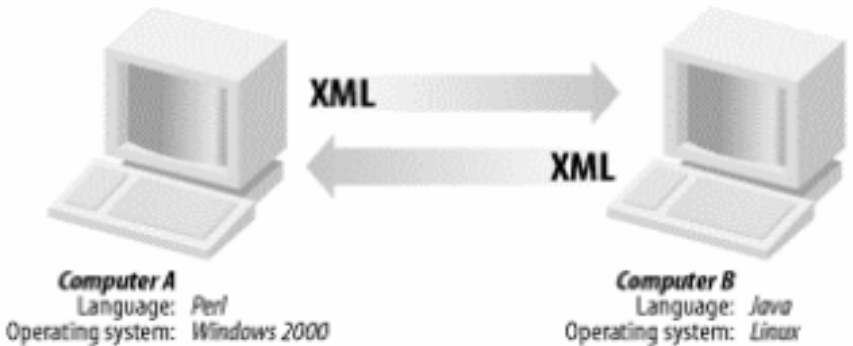
Web Service

- ❑ Mempertukarkan data dalam format XML.
- ❑ Tersedia dan dikomunikasikan melalui Internet atau intranet.
- ❑ Bersifat operating system/programming language independent.
- ❑ Berupa web application yang tidak memiliki web interface.
- ❑ Web service mempertukarkan data antara service requestor (aplikasi yang menggunakan data/service) dan service provider (server penyedia data/service) menggunakan service registry (yang berisi kumpulan service-service), dengan salah satu teknologi:
 - XML-RPC
 - SOAP

Web Service

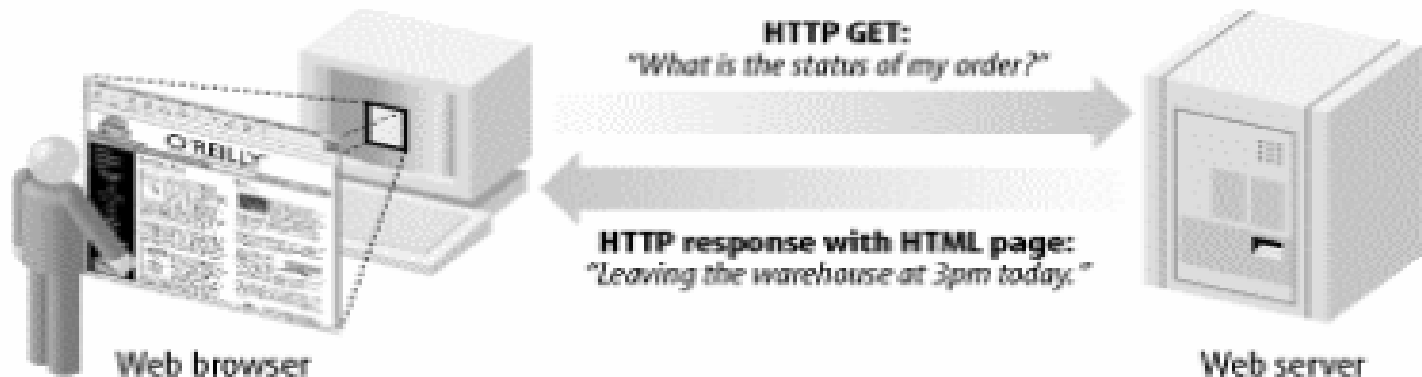
- ❑ Web service berbeda dengan model aplikasi terdistribusi tradisional seperti CORBA dan RMI dimana method yang dipanggil dieksekusi melalui jaringan dan menghasilkan **result** berupa **value** method tersebut, sedangkan web service menyediakan method yang akan menghasilkan **result** berupa **data XML** (bukan data biner)
- ❑ Web service dapat dipanggil/digunakan melalui web, aplikasi desktop, ataupun aplikasi mobile
- ❑ Kelebihan:
 - Interoperability (platform dan aplikasi)
 - Dapat mempublikasikan service dan method sehingga mudah digunakan
 - Mendukung reusable-components

Figure 1-1. A basic web service



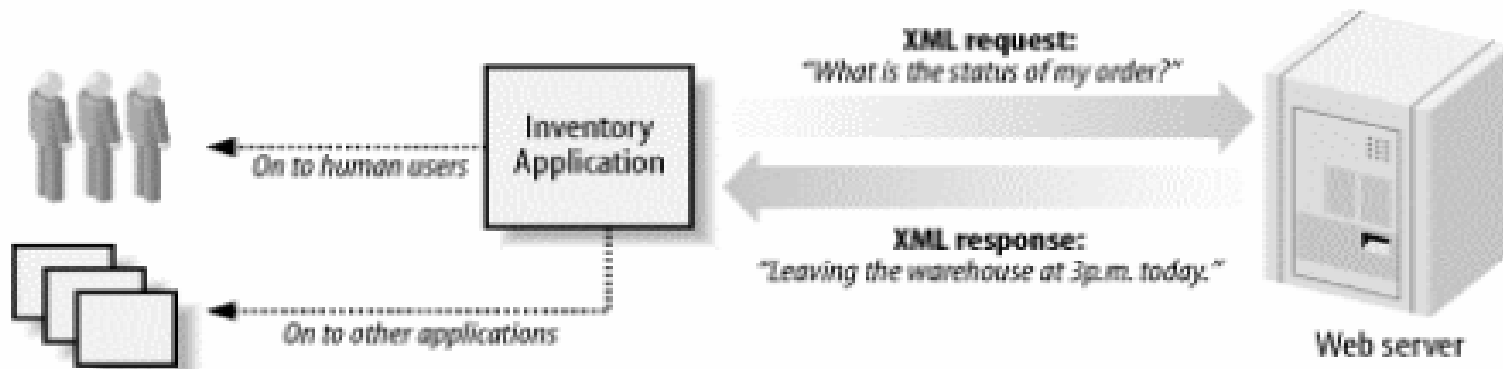
Website vs Web service

- ❑ Memiliki web interface
- ❑ Dibuat untuk berinteraksi langsung dengan user
- ❑ Dibuat untuk bekerja pada web browser
- ❑ Bersifat front-end
- ❑ Bersifat human-centric: manusia menjadi aktor utama



Website vs Web Service

- ❑ Tidak memiliki interface yang baik
- ❑ Dibuat untuk berinteraksi langsung dengan aplikasi yang lain, baik berbeda OS sekalipun, bukan dengan user.
- ❑ Dibuat untuk bekerja pada semua tipe client aplikasi / perangkat device
- ❑ Bersifat behind the scene.
- ❑ Bersifat application-centric: komunikasi terjadi antar aplikasi



HTTP Methods

- ❑ GET (retrieve a resource)
- ❑ HEAD (retrieve metadata about a resource)
- ❑ POST (add information to a resource)
- ❑ PUT (create a new resource / upload)
- ❑ DELETE (delete an existing resource)

WebDAV

- ❑ **Web-based Distributed Authoring and Versioning** is a set of extensions to the Hypertext Transfer Protocol (HTTP) which allows users to collaboratively edit and manage files on remote World Wide Web servers
- ❑ The protocol facilitates "Intercreativity", making the Web a readable and writable medium
- ❑ The original vision of the Web as expounded by Tim Berners-Lee was a both readable and writable medium.
- ❑ In fact Berners-Lee's first web browser, called WorldWideWeb, was able to both view and edit web pages; but, as the Web grew, it became, for most users, a read-only medium. Jim Whitehead and other like-minded people wanted to fix that limitation

WebDAV

- ❑ PROPFIND — Used to retrieve properties, stored as XML, from a resource. It is also overloaded to allow one to retrieve the collection structure (a.k.a. directory hierarchy) of a remote system.
- ❑ PROPPATCH — Used to change and delete multiple properties on a resource in a single atomic act.
- ❑ MKCOL — Used to create collections (a.k.a. directory).
- ❑ COPY — Used to copy a resource from one URI to another.
- ❑ MOVE — Used to move a resource from one URI to another.
- ❑ LOCK — Used to put a lock on a resource. WebDAV supports both shared and exclusive locks.
- ❑ UNLOCK — To remove a lock from a resource.

WebDAV alternatives

- ❑ CMS (Content Management System)
- ❑ FTP with CVS (Control Versioning System) / SVN (Subversion)
- ❑ SMB Protocol (Samba and Windows)
- ❑ Wiki System
- ❑ Web Service

Web Service properties

- ❑ Web service harus "**self-describing**": jika kita membuat web service, kita harus juga mempublikasikan public interface yang dapat dimengerti dan dipakai.
 - Minimalnya, service harus memiliki human-readable documentation sehingga developer lain dapat mengintegrasikan aplikasinya dengan web service yang kita buat.
 - Kita wajib membuat public interface sehingga dapat digunakan untuk mengidentifikasi semua method yang public, argumen-argumennya dan juga return valuenya.
- ❑ Web service harus "**discoverable**": jika kita membuat web service, harus ada mekanisme sederhana agar service dan public method yang kita buat dapat dikenal dan ditemukan oleh aplikasi lain.

HTTP GET dan POST

- GET: parameter beserta nilai yang dikirimkan ke server merupakan bagian dari URL yang panjangnya terbatas hanya 4096 bytes saja sehingga parameter dan nilai dari metode GET juga terbatas.
- Contoh PHP:

```
<HTML>
```

```
<HEAD><TITLE>Coba GET</HEAD>
```

```
</HTML>
```

```
<BODY>
```

```
Percobaan nilai NIM adalah = <? echo $nim; ?>
```

```
Percobaan nilai nama adalah = <? echo $nama; ?>
```

```
</BODY>
```

HTTP GET dan POST

- Disimpan di c:\apache\htdocs dengan nama cobaget.php
- TELNET:

```
GET cobaget.php?nim=22002529&nama=anton HTTP/1.1
<enter>
Host: localhost <enter>
Connection: close <enter>
<enter>
<enter>
```

HTTP GET dan POST

```
[[antonie@localhost ~]$ telnet 192.168.1.2 80
Trying 192.168.1.2...
Connected to 192.168.1.2 (192.168.1.2).
Escape character is '^]'.
GET /cobaget.php?nim=22002529&nama=anton HTTP/1.1
Host: 192.168.1.2
Connection: close
```

```
HTTP/1.1 200 OK
Date: Mon, 12 Sep 2005 14:28:57 GMT
Server: Apache/2.0.54 (Fedora)
X-Powered-By: PHP/5.0.4
Content-Length: 120
Connection: close
Content-Type: text/html; charset=UTF-8
```

```
<HTML>
<HEAD><TITLE>Coba GET</HEAD>
</HTML>
<BODY>
```

```
Percobaan nilai NIM adalah = 2202529Percobaan nilai nama adalah
= anton</BODY>
```

```
Connection closed by foreign host.
```

HTTP GET dan POST

- ❑ POST: parameter dan nilai yang dikirimkan ke server adalah merupakan bagian dari message body dokumen, karena itu ukurannya jauh lebih besar ($\leq 2\text{GB}$). Biasanya digunakan dalam sebuah form.
- ❑ Dengan file php yang sama:

```
POST cobapost.php HTTP/1.1 <enter>
Host: localhost<enter>
Content-Type: application/x-www-form-urlencoded <enter>
Content-Length: 16 <enter>
<enter>
nim=22002529&nama=anton <enter>
<enter>
<enter>
```

HTTP GET dan POST

```
[antonie@localhost ~]$ telnet 192.168.1.2 80
Trying 192.168.1.2...
Connected to 192.168.1.2 (192.168.1.2).
Escape character is '^]'.
POST /cobapost.php HTTP/1.1
Host: 192.168.1.2
Content-Type: application/x-www-form-urlencoded
Content-Length: 16

nim=22002529&nama=anton

HTTP/1.1 200 OK
Date: Mon, 12 Sep 2005 14:30:28 GMT
Server: Apache/2.0.54 (Fedora)
X-Powered-By: PHP/5.0.4
Content-Length: 124
Connection: close
Content-Type: text/html; charset=UTF-8

<HTML>
<HEAD><TITLE>Coba POST</HEAD>
</HTML>
<BODY>
Percobaan nilai NIM adalah = 22002529Percobaan nilai nama adalah
= anton

</BODY>
Connection closed by foreign host.
```

Pondasi Web Service

Service Directory:

UDDI

Service Description:

WSDL

Service Interaction:

SOAP

Format Description:

XML Schema

Data Format:

XML

Communication Protocol:

HTTP

Communication Network:

Internet

WS Layer

□ Service transport

- Bagian ini bertanggung jawab untuk **mengirim message** antar aplikasi.
- Internet Protokol yang ada di bagian ini: Hyper Text Transfer Protocol (HTTP)

□ XML messaging dan encoding/decoding

- Bagian ini bertanggung jawab untuk **mengencode/mendecode message** dalam format XML sehingga message dapat dimengerti dan dipertukarkan.
- Protokol/Service Interaction yang ada di bagian ini: XML-RPC dan SOAP.

WS Layer

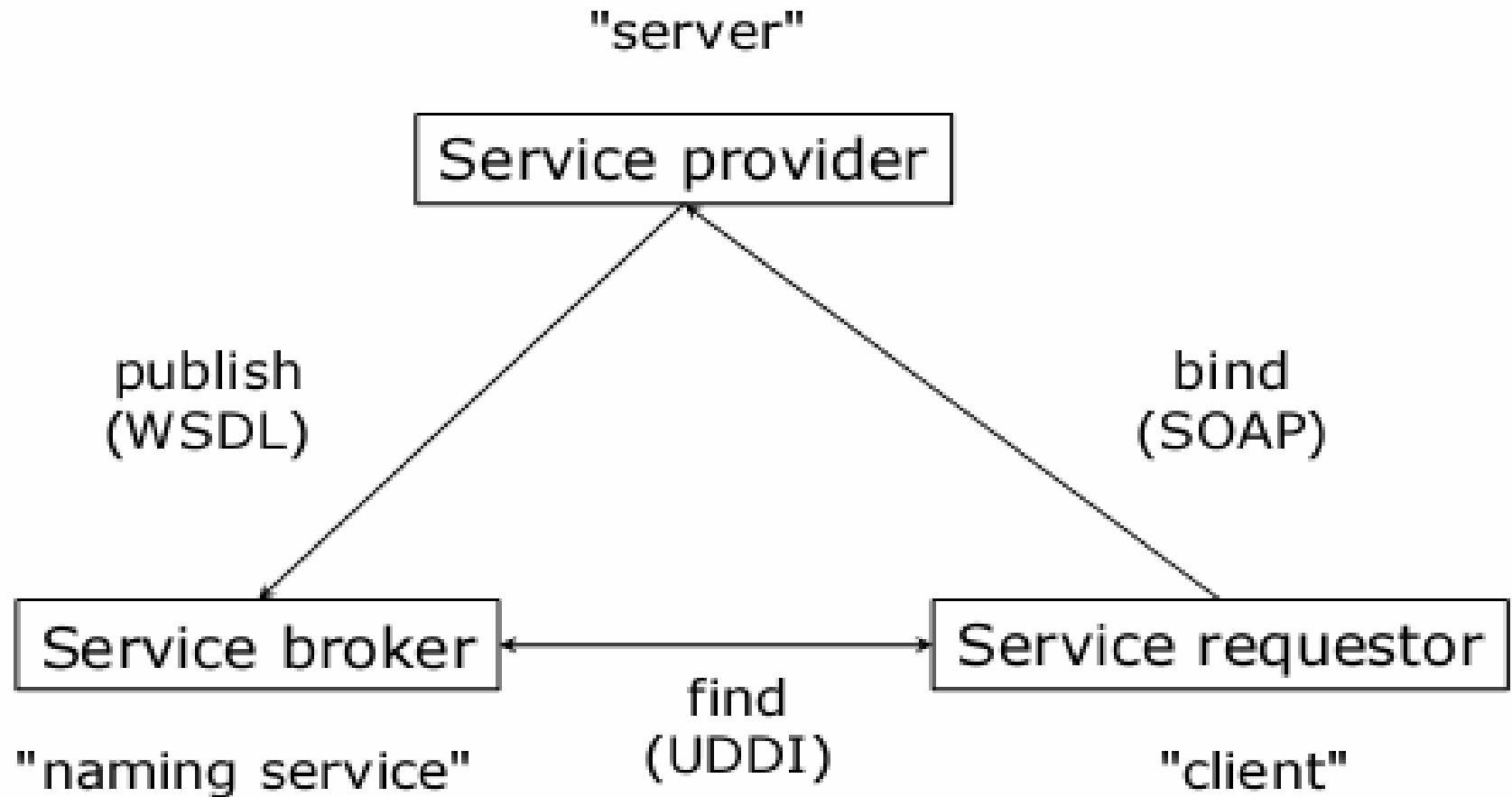
□ Service description

- Bagian ini bertanggung jawab untuk **mendesripsikan public interface** sesuai dengan web service yang spesifik.
- Bagian ini dihandle melalui Service Description: Web Service Description Language (WSDL) dan XML Schema

□ Service discovery

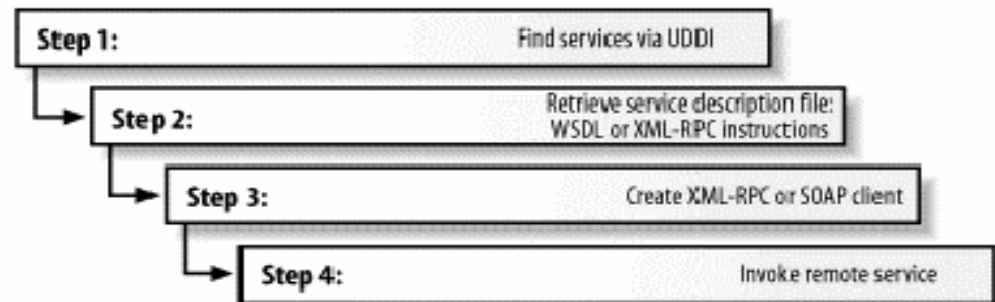
- Bagian ini bertanggung jawab untuk **mengumpulkan services ke dalam common registry dan menyediakan kemudahan untuk mempublikasikan interface dan kemudahan dalam pencarian method.**
- Bagian ini dihandle oleh Service Directory: Universal Description, Discovery, and Integration (UDDI).

WS Architecture



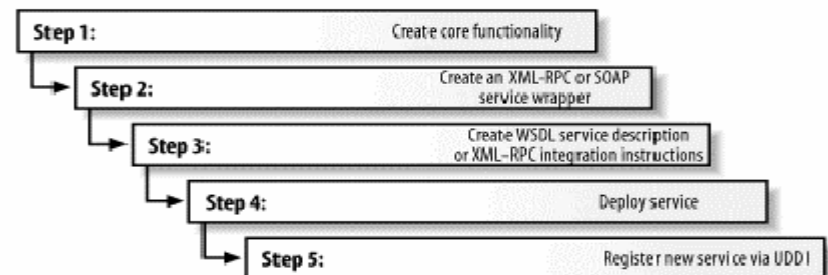
Service Requestor

- ❑ **Mengidentifikasi dan menemukan service** yang relevan sesuai dengan aplikasi yang ada. Biasanya dilakukan dengan pencarian terhadap **UDDI** Business Directory yang berisi service-service yang ada.
- ❑ Setelah mendapatkan service yang diinginkan, langkah selanjutnya adalah **mencari service description**. Bisa berupa WSDL ataupun XML-RPC instruction
- ❑ **Membuat client application**. Kita dapat membuat XML-RPC atau SOAP client sesuai dengan bahasa pemrograman yang kita sukai. Jika service ini memiliki file WSDL, kita juga memiliki opsi untuk secara otomatis membuat program client menggunakan WSDL invocation tool.
- ❑ **Jalankan client application** yang kita buat dan **invoke** the web service



Service Provider

- ❑ Kembangkan **fungsi-fungsi utama** dari service. Biasanya ini merupakan bagian yang paling sulit, seperti misalnya koneksi database, EJB, COM, dan lain-lain
- ❑ Kembangkan sebuah **service wrapper** untuk fungsi-fungsi utama tersebut dalam bentuk XML-RPC atau SOAP service wrapper/function/method.
- ❑ Kita harus menyediakan **service description**. Jika kita membuat SOAP service kita harus membuat file WSDL. Jika kita membuat XML-RPC service, kita harus membuat human-readable instructions untuk integrasi.
- ❑ **Deploy the service**. Kita dapat menyesuaikan dengan kebutuhan, apakah diinstal atau berjalan dalam server standalone atau diintegrasikan dengan web server yang telah ada.
- ❑ Kita harus **mempublikasikan** keberadaan dan spesifikasi service yang telah kita buat. Biasanya dapat dilakukan dengan cara mempublikasikan ke global UDDI directory atau private UDDI directory.

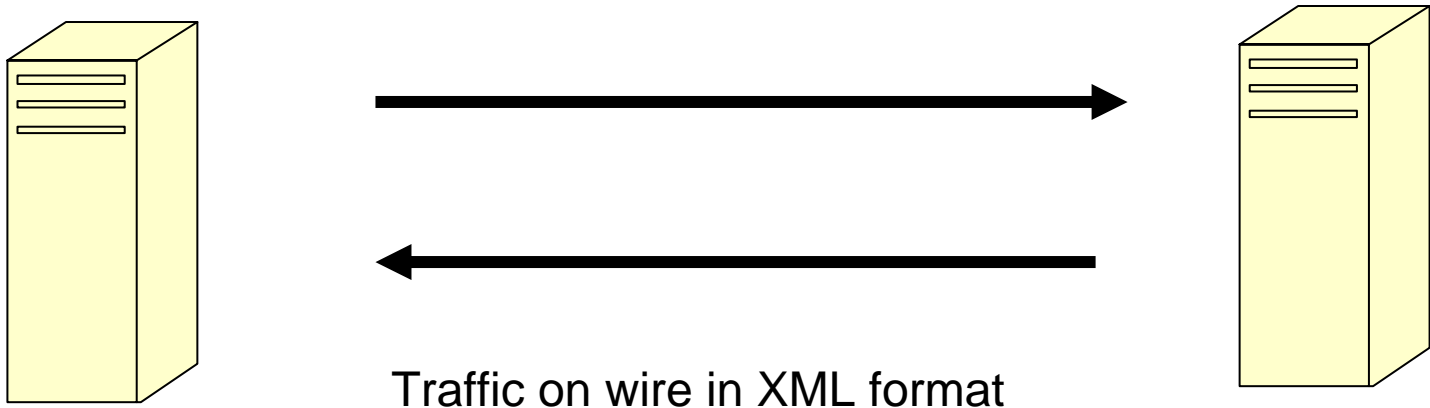


Service Broker / Service Registry

- The service brokers allow service providers to publish their services (register and categorize). They provide also mechanisms to locate services and their providers

XML-RPC

- XML-RPC is an attempt to implement conventional Remote Procedure Call (RPC) concepts using XML to transmit the RPC information



XML-RPC

- Client menentukan prosedur dan parameter yang akan diinvoke ke dalam XML request, sedangkan server akan meresponse
 - entah itu fault atau success di dalam XML response
- Dikembangkan tahun 1998 oleh UserLand Software (<http://www.xmlrpc.com>)
- Diikuti oleh Apache dan Sun

Request example

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: cis69.dyndns.com
Content-Type: text/xml
Content-length: 144
```

The Header

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.name</methodName>
  <params>
    <param>
      <value><int>41</int></value>
    </param>
  </params>
</methodCall>
```

examples.name(41)



Request Example (again)

POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

```
<?xml version="1.0"?>
  <methodCall>
    <methodName>Calculator.add</methodName>
    <params>
      <param> <value><int>41</int></value> </param>
      <param> <value><int>17</int></value> </param>
    </params>
  </methodCall>
```

- A User-Agent and Host must be specified.
- The Content-Type is text/xml.
- The Content-Length must be specified and must be correct.

XML-RPC data model

Default type: string

Table 2-1. Basic data types in XML-RPC

Type	Value	Examples
int or i4	32-bit integers between -2,147,483,648 and 2,147,483,647.	<code><int>27</int></code> <code><i4>27</i4></code>
double	64-bit floating-point numbers	<code><double>27.31415</double></code> <code><double>-1.1465</double></code>
Boolean	true (1) or false (0)	<code><boolean>1</boolean></code> <code><boolean>0</boolean></code>
string	ASCII text, though many implementations support Unicode	<code><string>Hello</string></code> <code><string>bonkers! @</string></code>
dateTime.iso8601	Dates in ISO8601 format: <i>CCYYMMDDTHH:MM:SS</i>	<code><dateTime.iso8601>20021125T02:20:04</dateTime.iso8601></code> <code><dateTime.iso8601>20020104T17:27:30</dateTime.iso8601></code>
base64	Binary information encoded as Base 64, as defined in RFC 2045	<code><base64>SGVsbG8sIFdvcmxkIQ==</base64></code>

<Value>

- Semua tipe data tersebut harus diapit oleh element `<value> ... </value>`

Untuk data **array string**:

```
<value>
  <array>
    <data>
      <value><string>This </string></value>
      <value><string>is </string></value>
      <value><string>an </string></value>
      <value><string>array.</string></value>
    </data>
  </array>
</value>
```

Untuk **array integer**

```
<value>
  <array>
    <data>
      <value><int>7</int></value>
      <value><int>1247</int></value>
      <value><int>-91</int></value>
      <value><int>42</int></value>
    </data>
  </array>
</value>
```

Array (2)

- Untuk array yang terdiri dari berbagai macam tipe data

```
<value>
  <array>
    <data>
      <value><boolean>1</boolean></value>
      <value><string>Chaotic collection, eh?</string></value>
      <value><int>-91</int></value>
      <value><double>42.14159265</double></value>
    </data>
  </array>
</value>
```

Untuk array multidimensi

```
<value>
  <array>
    <data>
      <value>
        <array>
          <data>
            <value><int>10</int></value>
            <value><int>20</int></value>
            <value><int>30</int></value>
          </data>
        </array>
      </value>
      <value>
        <array>
          <data>
            <value><int>15</int></value>
            <value><int>25</int></value>
            <value><int>35</int></value>
          </data>
        </array>
      </value>
    </data>
  </array>
</value>
```

Struct

Untuk simple struct

```
<value>
  <struct>
    <member>
      <name>givenName</name>
      <value><string>Joseph</string></value>
    </member>
    <member>
      <name>familyName</name>
      <value><string>DiNardo</string></value>
    </member>
    <member>
      <name>age</name>
      <value><int>27</int></value>
    </member>
  </struct>
</value>
```

Untuk struct yang terdiri dari struct lain atau array

```
<value>
  <struct>
    <member>
      <name>name</name>
      <value><string>a</string></value>
    </member>
    <member>
      <name>attributes</name>
      <value><struct>
        <member>
          <name>href</name>
          <value><string>http://example.com</string></value>
        </member>
        <member>
          <name>target</name>
          <value><string>_top</string></value>
        </member>
      </struct></value>
    </member>
    <member>
      <name>contents</name>
      <value><array>
        <data>
          <value><string>This </string></value>
          <value><string>is </string></value>
          <value><string>an example.</string></value>
        </data>
      </array></value>
    </member>
  </struct>
</value>
```

XML-RPC Request Structure

- Setiap request terdiri XML, dimana root elemennya adalah `<methodCall>`.
 - Setiap elemen `<methodCall>` terdiri dari elemen `<methodName>` yang dapat berisi elemen `<params>`.
- Elemen `<methodName>` mendefinisikan nama prosedur yang dipanggil sementara `<params>` berisi daftar parameter dan nilainya, setiap `params` terdiri dari elemen-elemen `<param>` yang juga terdiri dari elemen `<value>`

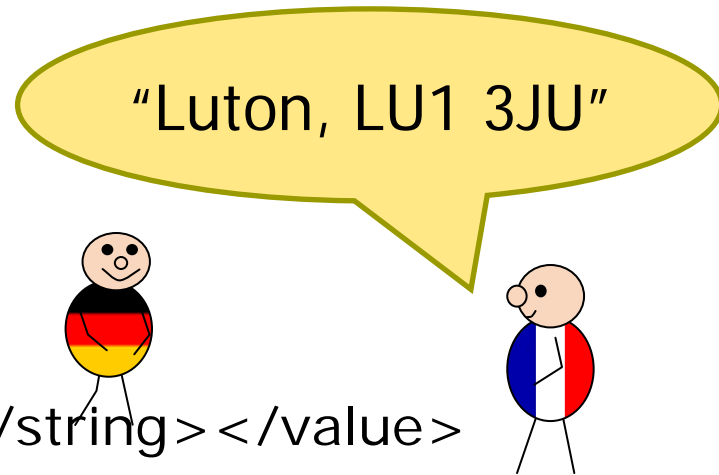
Pengiriman Parameter Array

```
<?xml version="1.0"?>
<methodCall>
  <methodName>sortArray</methodName>
  <params>
    <param>
      <value>
        <array>
          <data>
            <value><int>10</int></value>
            <value><int>20</int></value>
            <value><int>30</int></value>
          </data>
        </array>
      </value>
    </param>
    <param>
      <value>
        <array>
          <data>
            <value><string>A</string></value>
            <value><string>C</string></value>
            <value><string>B</string></value>
          </data>
        </array>
      </value>
    </param>
  </params>
</methodCall>
```

Here's an example of a response to an XML-RPC request:

HTTP/1.1 200 OK
Connection: close
Content-Length: 134
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

```
<?xml version="1.0"?>  
<methodResponse>  
  <params>  
    <param>  
      <value><string>Luton, LU1 3JU</string></value>  
    </param>  
  </params>  
</methodResponse>
```



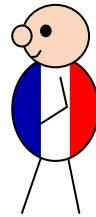
Response format

- ❑ Unless there's a lower-level error, always return 200 OK.
- ❑ The Content-Type is text/xml. Content-Length must be present and correct.
- ❑ The body of the response is a single XML structure, a `<methodResponse>`, which can contain a single `<params>` which contains a single `<param>` which contains a single `<value>`.
- ❑ The `<methodResponse>` could also contain a `<fault>` which contains a `<value>` which is a `<struct>` containing two elements, one named *faultCode*, an `<int>` and one named *faultString*, a `<string>`.

Fault example

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>4</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Overflow</string> </value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

faultCode: 4
faultString: Overflow



NEXT

- SOAP dan UDDI