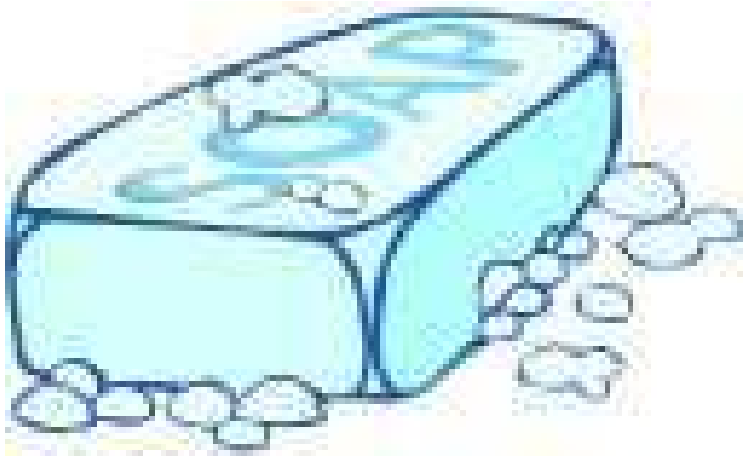
# AASE #11

SOAP dan UDDI

# What is SOAP?



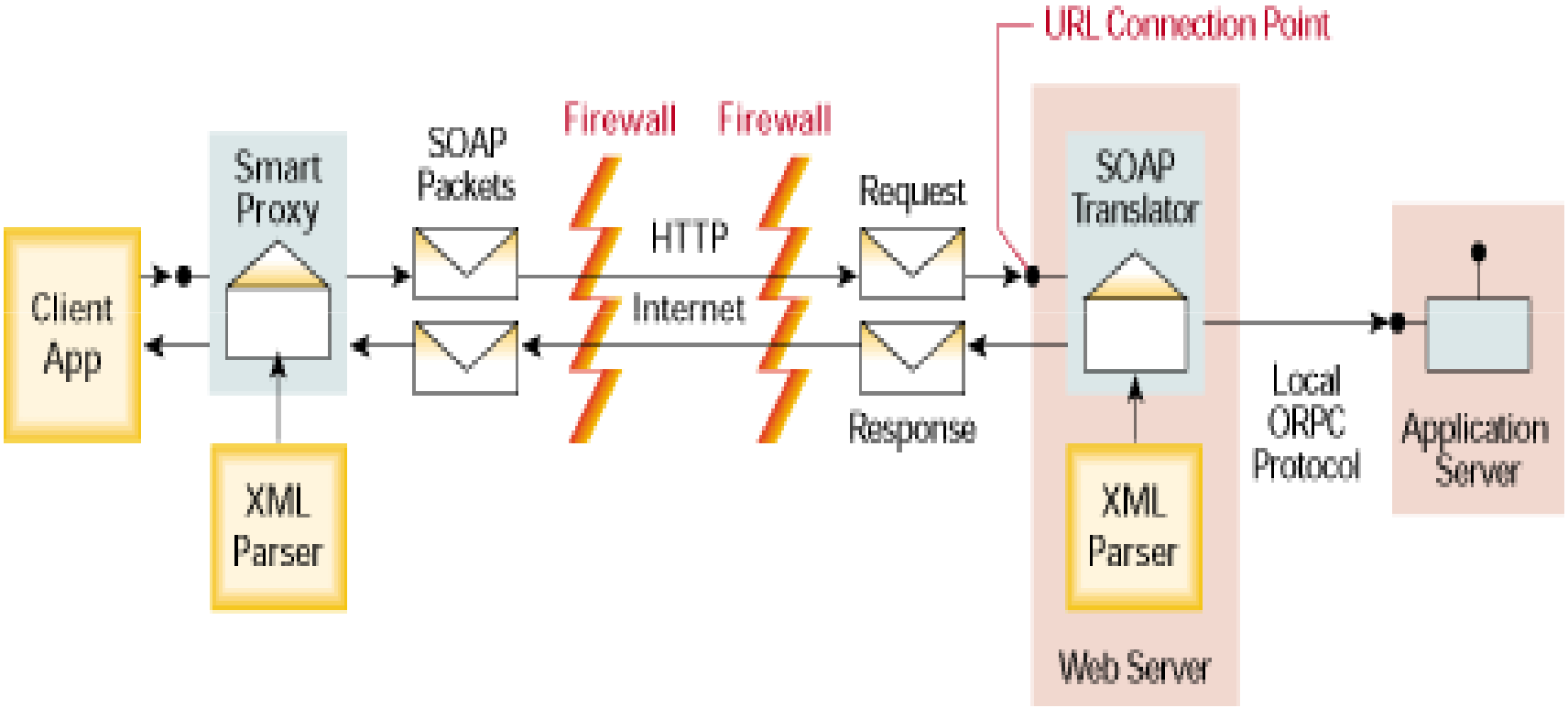
# SOAP Background

- Originally designed by **Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein** in 1998 with backing from Microsoft as an object-access protocol, the SOAP specification is currently maintained by the XML Protocol Working Group of the World Wide Web Consortium.
- Developed by IBM, Microsoft, Lotus, and others
- Submitted to W3C - Became W3C Note in May 1998
- Current Version is 1.2 tahun 2007
  - http://www.w3.org/TR/SOAP

# SOAP (Simple Object Access Protocol)

- SOAP merupakan **protokol komunikasi berbasis** XML yang memperbolehkan aplikasi **saling bertukar informasi** melalui HTTP
- SOAP merupakan protokol yang menangani **web service**
- SOAP merupakan protokol yang berupa **format** untuk mengirimkan message melalui Internet, bersifat *platform independent, language independent, dan merupakan standar W3C*
- SOAP membungkus **request & response XML**

# SOAP call anatomy



# Elemen SOAP

- Elemen **Envelope** yang mengidentifikasi XML dokumen sebagai SOAP message (wajib)
  - Top element of the XML document representing the **message**
- Elemen **Header** yang berisi informasi header (opsional)
  - Determines how a recipient of a SOAP message should process the message
  - Adds features to the SOAP message such as authentication, message routes, additional information, etc…
- Elemen **Body** yang berisi informasi call dan response (wajib)
- Elemen **Fault** yang berisi informasi error yang terjadi (opsional)

# Plus-Minus SOAP

**Plus**

- Uses **HTTP** which is widely used and scalable
- Flexible for growth because of **XML** properties
- Data in **String** message

**Minus**

- Parsing of SOAP packet and mapping to objects **reduces performance**
- If XML data are too long
- **Doesn’t implement security** because it is a wire protocol—relies on HTTP

# Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:testns“
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Body>
            <ns1:getProfessor>
                <course>470</course>
            </ns1:getProf>
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

# Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:testns"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Body>
                <ns1:getProfessorResponse>
                        <Result>Antonie</Result>
                </ns1:getProfResponse>
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

# Another SOAP message

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope”
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <m:MessageInfo xmlns:m="http://www.info.org/soap/message">
            <m:to href="mailto:you@your.com"/>
            <m:from href="mailto:me@my.com"/>
            <m:contact href="mailto:someone@my.com">
      </m:MessageInfo>
   </SOAP:Header>
   <SOAP:Body>
      <msg:Message xmlns:m="http://www.info.org/soap/message">
            <msg:subject>Your house is on fire!</msg:subject>
            <msg:feed href="ram://livenews.com/yourhouse"/>
      </msg:Message>
   </SOAP:Body>
</SOAP:Envelope>
```

# Aturan

- SOAP Envelope menggunakan namespace
  - http://schemas.xmlsoap.org/soap/envelope/
- Default namespace untuk SOAP encoding dan data types adalah
  - http://schemas.xmlsoap.org/soap/encoding/
- SOAP's Syntax Rules:
  - SOAP harus dibuat dengan menggunakan sintaks XML
  - SOAP harus menggunakan SOAP Envelope namespace
  - SOAP harus menggunakan SOAP Encoding namespace
  - SOAP tidak boleh mengandung DTD
- SOAP tidak boleh mengandung **XML Processing Instruction**

# SOAP Skeleton

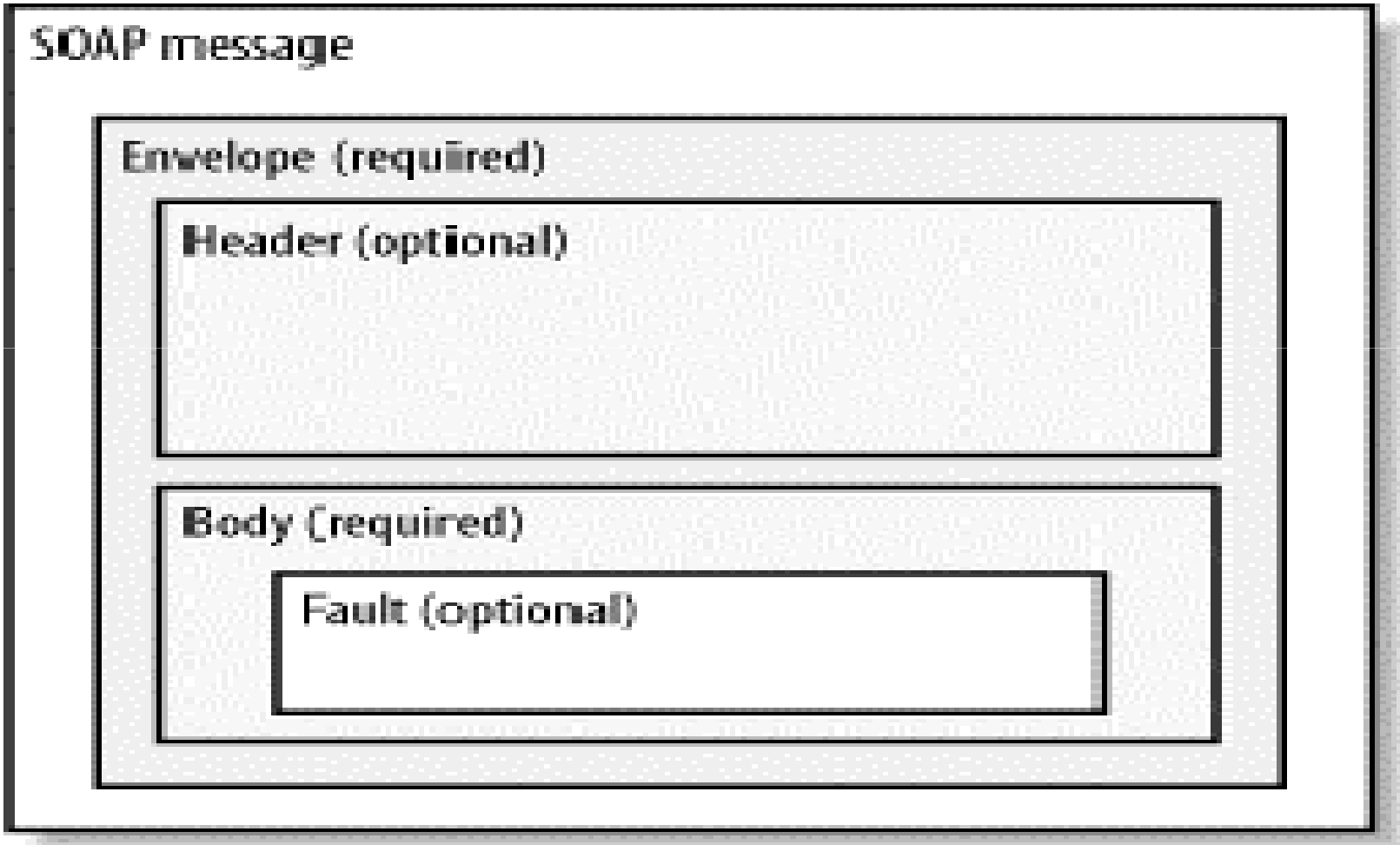
```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

# SOAP Skeleton (2)



# SOAP **Fault** element

- Used to carry **error** and/or **status information** within a SOAP message
- Appears within the SOAP **body**, if appears, it just appears only **once**.

| Sub Element | Description |
|---|---|
| <faultcode> | A code for identifying the fault |
| <faultstring> | A human readable explanation of the fault |
| <faultactor> | Information about who caused the fault to happen |
| <detail> | Holds application specific error information related to the Body element |

# SOAP **Fault Codes** Element

| Error | Description |
|---|---|
| VersionMismatch | Found an invalid namespace for the SOAP Envelope element |
| MustUnderstand | An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood |
| Client | The message was incorrectly formed or contained incorrect information |
| Server | There was a problem with the server so the message could not proceed |

# SOAP Fault Example

- A SOAP message containing an **authentication** service:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope”
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <m:Authentication xmlns:m="http://www.auth.org/simple">
            <m:credentials>Henrik</m:credentials>
      </m:Authentication>
   </SOAP:Header>
   <SOAP:Body>
      … body goes here …
   </SOAP:Body>
</SOAP:Envelope>
```

# SOAP Fault Example… 2

- …results in a fault because the credentials were **bad**:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope”
 SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP:Header>
    <m:Authentication xmlns:m="http://www.auth.org/simple">
          <m:realm>Magic Kindom</m:realm>
    </m:Authentication>
  </SOAP:Header>
  <SOAP:Body>
    <SOAP:Fault>
        <SOAP:faultcode>SOAP:Client</faultcode>
        <SOAP:faultstring>Client Error</faultstring>
     </SOAP:Fault>
  </SOAP:Body>
</SOAP:Envelope>
```

# Another SOAP Fault Example

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV=“http://schemas.xmlsoap.org/soap/envelope/”
   SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”>
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring>Internal Application Error</faultstring>
         <detail xmlns:f=“http://anton.com/CalculatorFault”>
            <f:errorCode>794634</f:errorCode>
            <f:errorMsg>Divide by zero</f:errorMsg>
         </detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# SOAP Header (opsional)

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">234
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

- The **mustUnderstand** attribute is used to indicate whether a header entry is **mandatory** or optional for the recipient to process
- If the receiver does not recognize the element it will **fail** when processing the Header
- The **encodingStyle** attribute is used to define the **data types** used in the document

# The SOAP mustUnderstand Attribute

- Requires that the receiving SOAP processor must **accept, understand** and **obey semantics** of header or fail
  - It is up to the application to define what "understand" means
- This allows old applications to gracefully **fail** on services that they do not understand

# SOAP Body Resquest

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

- the **m:GetPrice** and the Item elements above are application-specific elements.
- They are not a part of the SOAP namespace

# SOAP Body Response

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

- Hasil dari request harga apple

# HTTP Request

- Use HTTP POST request method name
- Use the SOAPAction HTTP header field
  - It cannot be computed – the sender must know

```
POST /Accounts/Henrik HTTP/1.1
Content-Type: text/xml; charset="utf-8“
Content-Length: nnnn
SOAPAction: http://electrocommerce.org/MyMessage
<SOAP:Envelope...
```

# HTTP Response

- Successful responses can 2xx status codes
- All 3xx, 4xx, and 5xx status codes work as normal
- SOAP faults must use 500 status code

```
HTTP/1.1 200 Ok
Content-Type: text/xml; charset="utf-8“
Content-Length: nnnn

<SOAP:Envelope...
```

# Example SOAP Request

```
POST /weather HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-length: XXX
SOAPAction: "http://anton.com/Calculator"
<?xml Version=1.0?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV=“http://schemas.xmlsoap.org/soap/envelope/”
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”>
   <SOAP-ENV:Header>
      <t:transId xmlns:t=“http://anton.com/trans”>345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:Add xmlns:m=“http://anton.com/Calculator”>
         <n1>3</n1>
         <n2>4</n2>
      </m:Add>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Example SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://anton.com/trans">345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:AddResponse xmlns:m="http://anton.com/Calculator">
         <result>7</result>
      </m:AddResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# SOAP Request Example (2)

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

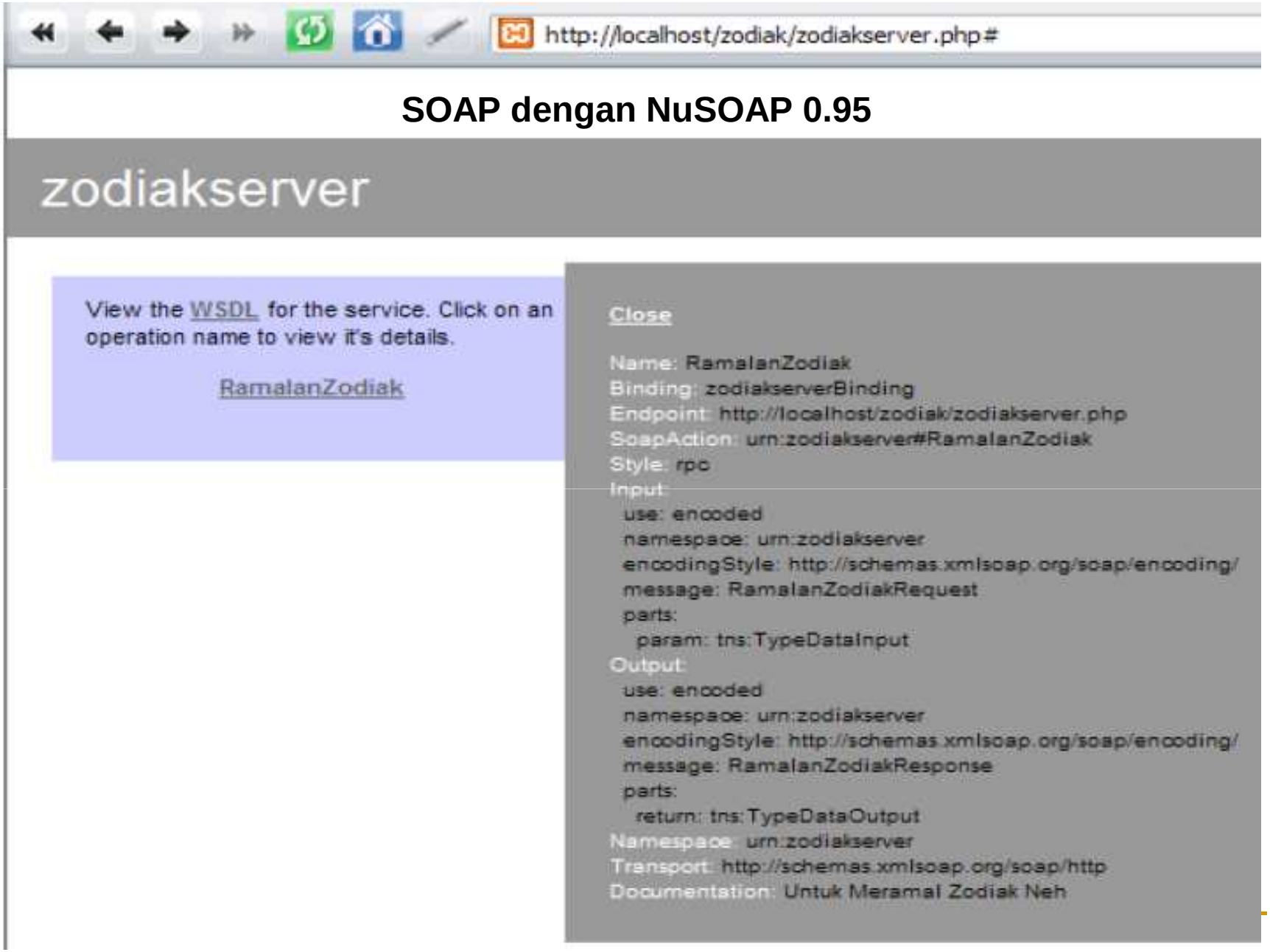
# SOAP Response Example (2)

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```


http://localhost/zodiak/zodiakserver.php#
SOAP dengan NuSOAP 0.95
zodiakserver
View the WSDL for the service. Click on an operation name to view it's details.
RamalanZodiak
Close
Name: RamalanZodiak
Binding: zodiakserverBinding
Endpoint: http://localhost/zodiak/zodiakserver.php
SoapAction: urn:zodiakserver#RamalanZodiak
Style: rpc
Input:
use: encoded
namespace: urn:zodiakserver
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: RamalanZodiakRequest
parts:
param: tns:TypeDataInput
Output:
use: encoded
namespace: urn:zodiakserver
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: RamalanZodiakResponse
parts:
return: tns:TypeDataOutput
Namespace: urn:zodiakserver
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: Untuk Meramal Zodiak Neh

# Fault dan Result

## Fault

```
Array
(
    [faultcode] => client
    [faultactor] =>
    [faultstring] => Parameter harus ada nilainya!
    [detail] =>
)
```

## Result

```
Array
(
    [nama_zodiak] => capricorn
    [tanggal] => januari
    [ramalan] => hem
    [keuangan] => ok
    [kesehatan] => yup
    [angka_keberuntungan] => 18
)
```

# UDDI

- **Universal Description, Discovery and Integration**
- Adalah direktori/registry yang bersifat *platform independent*, untuk mendeskripsikan, menyimpan **service**, pada Internet yang berkomunikasi via **SOAP** dan dideskripsikan dgn **WSDL**.

# What Problems Do UDDI Solve?

Broader B2B

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Smarter Search

A flower shop in Australia wants to be "plugged in" to every marketplace in the world, but doesn't know how

Easier Aggregation

A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

*Describe Services*

*Discover Services*

*Integrate Them Together*

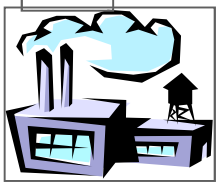
# UDDI History

- **UDDI 1.0** was originally announced by Microsoft, IBM, and Ariba in **September 2000**.
- **May 2001**, Microsoft and IBM launched the first UDDI operator sites and turned the UDDI registry live.
- **June 2001**, UDDI announced Version **2.0**.
- Currently UDDI is sponsored by **OASIS**

# How UDDI Works

1. 

SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2. 

Businesses populate the registry with descriptions of the services they support

UDDI Business Registry

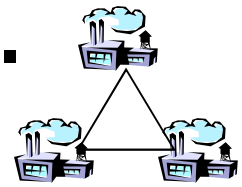
Business Registrations

Segrvice Type Reistrations

3. Programmer assigns a programmatically unique identifier to each service and business registration

4. 

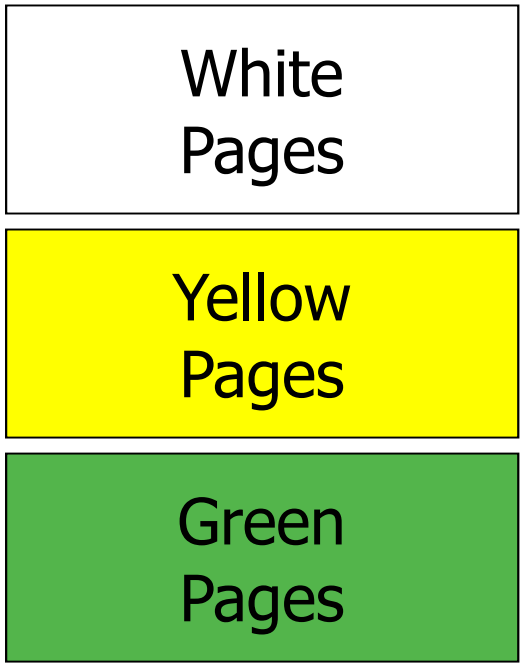
Marketplaces, search engines, and business apps query the registry to discover services at other companies

5. 

Business uses this data to facilitate easier integration with each other over the Web

# Komponen UDDI

- UDDI memiliki 3 komponen:
  - White Pages
  - Yellow Pages
  - Green Pages



# White Pages

- business name
- text description
  - list of multi-language text strings
- contact info
  - names, phone numbers, fax numbers, web sites…
- unique known identifiers
  - list of identifiers that a business may be known

# Yellow Pages

- This has **more details** about the company, and includes **descriptions** of the kind of electronic capabilities the company can offer to anyone who wants to do business with it.
- It includes business categories
  - 3 standard taxonomies:
    - Industry:  NAICS (Industry codes - US Govt.)
    - Product/Services:  UN/SPSC (ECMA)
    - Location: Geographical taxonomy
  - Implemented as **name-value pairs** to allow any valid taxonomy identifier to be attached to the business white page

# Green Pages

- This category contains **technical** information about a web service
- This is what allows someone to **bind** to a Web service after it's been found
- It contains:
  - The various interfaces
  - The URL locations
  - Discovery information and similar data required to find and run the Web service.

# UDDI Data Structure

- The core data structures in a UDDI Registry are:
  - **Business Entity**: A business / the provider of web services
  - **Business Service**: A logical service
  - **Binding Template**: Associates a service with a technical model
  - **Technical Model**: A technical service description
  - **Publisher Assertion :** publisher relation to another parties
- These structures are identified by 128-bit globally unique identifier also known as **UUID**.
  - E.g. **uuid:23453aef-af35-6a3f-c34a-bf798dab965a**

# Business Entity

The business entity structure represents the **provider of web services**

```
<businessEntity businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"
        operator="http://www.ibm.com"
        authorizedName="John Doe">
    <name>Acme Company</name>
    <description>
        We create cool Web services
    </description>
    <contacts>
        <contact useType="general info">
            <description>General Information</description>
            <personName>John Doe</personName>
            <phone>(123) 123-1234</phone>
            <email>jdoe@acme.com</email>
        </contact>
    </contacts>
  <businessServices>
  ...
  </businessServices>
  <identifierBag>
     <keyedReference
        tModelKey="UUID:8609C81E-EE1F-4D5A-B202-3EB13AD01823"
        name="D-U-N-S"
        value="123456789" />
  </identifierBag>
  <categoryBag>
     <keyedReference
        tModelKey="UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2"
        name="NAICS"
        value="111336" />
   </categoryBag>
</businessEntity>
```

# <businessEntity>

- **Primary** information about a business
  - Contact information
  - Categorisation of business in specific taxonomy or classification scheme(s)
  - Identifiers
  - Relationships to other business entities
- Relation is using <publisherAssertion>
  - Descriptions of the business and relation with other parties

# Business Services

- The business service structure represents an **individual web service** provided by the business entity
- An example of a business service structure for the **Hello World web service**

```
<businessService serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
        businessKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
    <name>Hello World Web Service</name>
    <description>A friendly Web service</description>
    <bindingTemplates>
    ...
    </bindingTemplates>
    <categoryBag />
</businessService
```

# <businessService>

- Represents a **service** of a business
  - i.e. “Order a Book”
- Berada di dalam tag <businessEntity>
  - Business can share/reuse services
- The service is represented **logically**
  - how to access the services is not described here

# Binding Template

- Binding templates are the **technical descriptions** of the web services represented by the business service structure
- A single business service may have **multiple binding** templates
- The binding template represents **the actual implementation** of the web service

# Binding Template (2)

```
<bindingTemplate serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
        bindingKey="uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
    <description>Hello World SOAP Binding</description>
    <accessPoint URLType="http">
        http://localhost:8080
    </accessPoint>
    <tModelInstanceDetails>
        <tModelInstanceInfo
              tModelKey="uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
        <instanceDetails>
             <overviewDoc>
                <description>
                    references the description of the
                    WSDL service definition
                </description>
                <overviewURL>
                     http://localhost/helloworld.wsdl
                </overviewURL>
             </overviewDoc>
        </instanceDetails>
        </tModelInstanceInfo>
     </tModelInstanceDetails>
</bindingTemplate>
```

# <bindingTemplate>

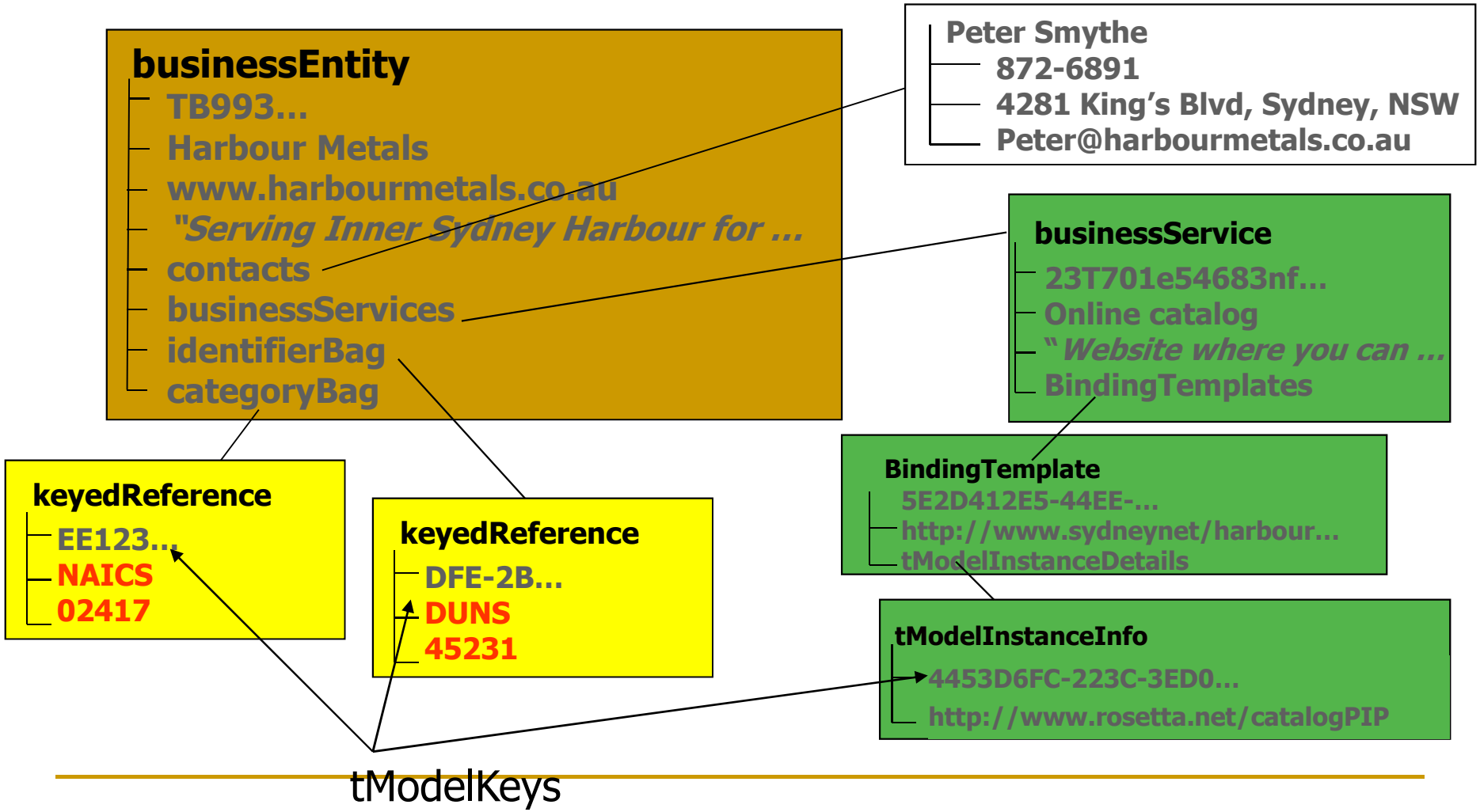
- Berada di dalam tag <businessService>
- Memiliki daftar <accessPoints> ke <businessService>
  - SOAP endpoint URL
  - Email address
  - Telephone/fax number

# tModel

- A tModel is a way of **describing technical data** about the *various business, service, and template structures* stored within the UDDI registry

```
<tModel tModelKey="uuid:xyz987..."
           operator="http://www.ibm.com"
           authorizedName="John Doe">
    <name>HelloWorldInterface Port Type</name>
    <description>
           An interface for a friendly Web service
    </description>
    <overviewDoc>
           <overviewURL>
                http://localhost/helloworld.wsdl
           </overviewURL>
     </overviewDoc>
</tModel>
```

# Example of a Registration



# UDDI Publisher API

- **http://www.tutorialspoint.com/uddi/uddi_api_references.htm**
- Used to publish/change/delete the information in a UDDI registry
  - save_business, delete_business,
  - save_service, delete_service,
  - save_binding, delete_binding,
  - save_tModel, delete_tModel
- To manage relationship assertions
  - get/add/set_publisherAssertions, get_assertionStatusReport
  - get_registeredInfo - get <businessEntity> & <tModels>owned by this publisher

# UDDI Inquiry API

- Used to find matching entities (Browse):
  - find_business,
  - find_relatedBusinesses,
  - find_service, find_binding,
  - find_tModel
- Or to get detailed information (Drill Down):
  - get_businessDetail,
  - get_businessDetailExt,
  - get_serviceDetail,
  - get_bindingDetail, get_tModelDetail

# UDDI creating registry example

```
POST /save_business HTTP/1.1
Host: www.XYZ.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "save_business"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
      <Body>
         <save_business generic="2.0" xmlns="urn:uddi-org:api_v2">
         <businessKey="">
         </businessKey>
         <name>
         XYZ, Pvt Ltd.
         </name>
         <description>
             Company is involved in giving Stat-of-the-art....
         </description>
         <identifierBag> ... </identifierBag>
         ...
         </save_business>
      </Body>
</Envelope>
```

# UDDI retreiving data

```
POST /get_businessDetail HTTP/1.1
Host: www.XYZ.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "get_businessDetail"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
   <Body>
      <get_businessDetail generic="2.0" xmlns="urn:uddi-org:api_v2">
          <businessKey="C90D731D-772HSH-4130-9DE3-5303371170C2">
      </businessKey>
      </get_businessDetail>
   </Body>
</Envelope>
```

# UDDI Programming

- **Java Implementations:**
  - UDDI4J (UDDI for Java): UDDI4J was originally created by IBM. In January 2001, IBM turned over the code to its own open source site. UDDI4J is a Java class library that provides an API to interact with a UDDI.
  - jUDDI: jUDDI is an open source Java implementation of a UDDI registry and a toolkit for accessing UDDI services.
- **Perl Implementation:**
  - UDDI::Lite : provides a basic UDDI client for inquiry and publishing.
- **Ruby Implementation:**
  - UDDI4r: provides a basic UDDI client for inquiry and publishing.
- **Python Implementation:**
  - UDDI4Py: UDDI4Py is a Python package that allows the sending of requests to and processing of responses from the UDDI Version 2 APIs.

# NEXT

- WSDL (Web Service Description Language)