# Introduzione ai Web Services


Moreno Marzolla
INFN Sezione di Padova
moreno.marzolla@pd.infn.it
http://www.dsi.unive.it/~marzolla

Incontro CCR—LNL, 10 dicembre 2008

# Service Oriented Architecture (SOA)

*"Service Oriented Architecture is a paradigm for organizing and utilizing* ***distributed capabilities*** *that may be under the control of* ***different ownership domains****. It provides a uniform means to* ***offer****,* ***discover****,* ***interact*** *with and use capabilities to produce desired effects consistent with measurable preconditions and expectations"*

http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

# A cosa servono i Web Services

- Il Web ha fornito il suo primo supporto alle attività umane attraverso un meccanismo di scambio di dati testuali e grafici.
  - Originariamente tale scambio era iniziato da utenti “umani”
- Applicazioni Internet-based devono essere  in grado di trovare, accedere, e interagire con altre applicazioni Internet-based.
- L’architettura dei Web Service mira a sfruttare le potenzialità di Internet fornendo un meccanismo di comunicazione application-to-application.

# Web Service

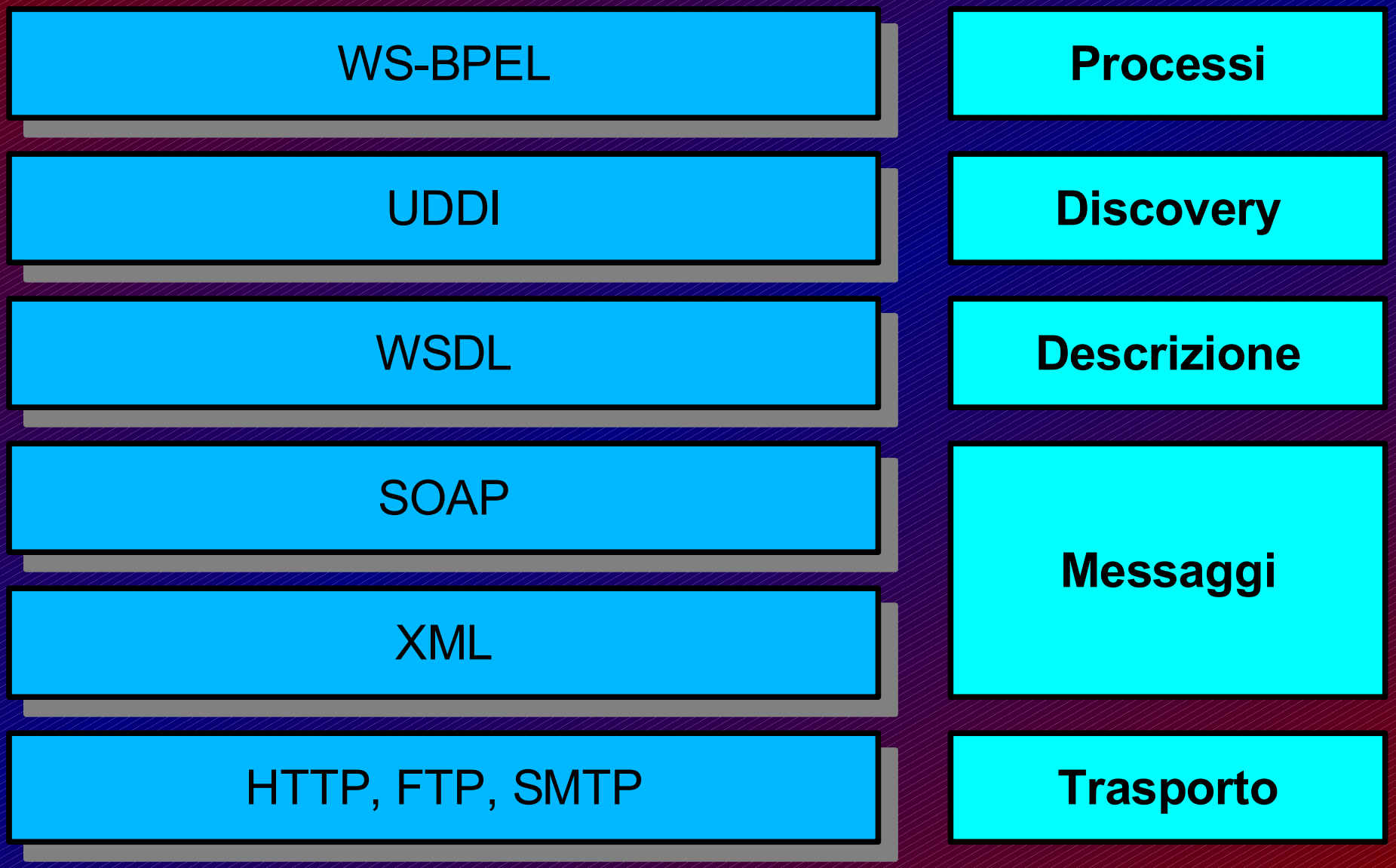
*"A Web service is a **software system** designed to support **interoperable machine-to-machine interaction** over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."*

http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

# Tecnologie dei Web Services

- L'infrastruttura dei WS si basa su diverse tecnologie XML per il trasporto, lo scambio e la trasformazione dei dati tra programmi e applicazioni. In particolare:
  - **XML** (*Extensible Markup Language*), fornisce un linguaggio per definire i dati e processare i dati.
  - **WSDL** (*Web Service Description Language*), una notazione XML per definire le interfacce dei WS.
  - **SOAP** (*Simple Object Access Protocol*), una notazione XML che fornisce un meccanismo di packaging dei messaggi
  - **UDDI** (*Universal Description, Discovery and Integration*), un meccanismo per registrare, categorizzare e rintracciare le interfacce WS.

# Stack dei WS

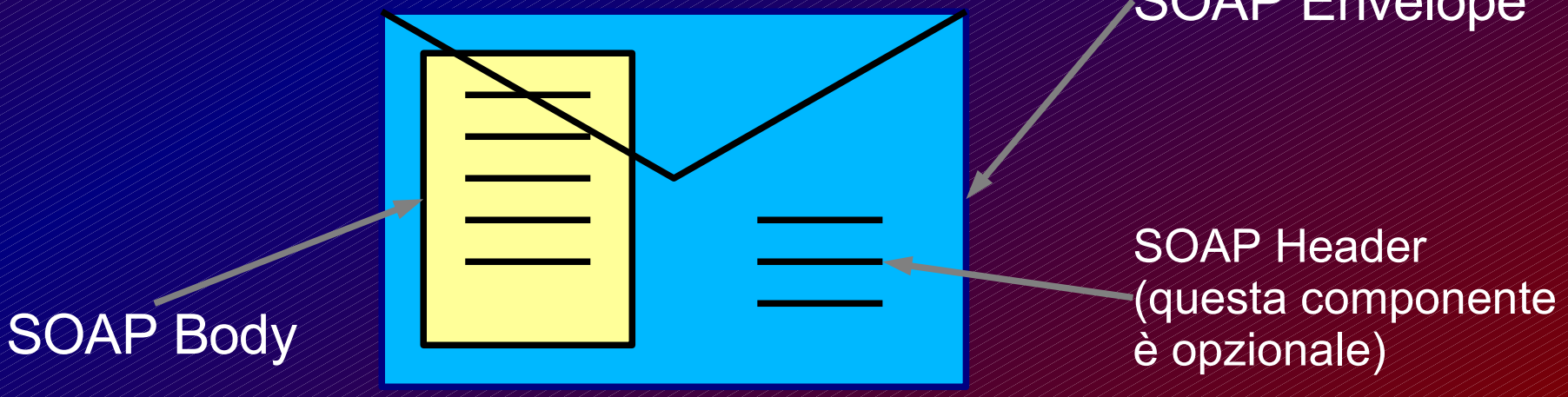

WS-BPEL
UDDI
WSDL
SOAP
XML
HTTP, FTP, SMTP
Processi
Discovery
Descrizione
Messaggi
Trasporto

# SOAP

- Il Simple Object Access Protocol (SOAP):
  - Definisce un meccanismo per il trasporto dei dati da un punto all'altro della rete.
  - Consente al mittente e al destinatario di un documento XML di disporre di un protocollo comune.
  - Consente la spedizione di messaggi XML su HTTP e la relativa ricezione di una risposta.

# SOAP—Formato dei messaggi

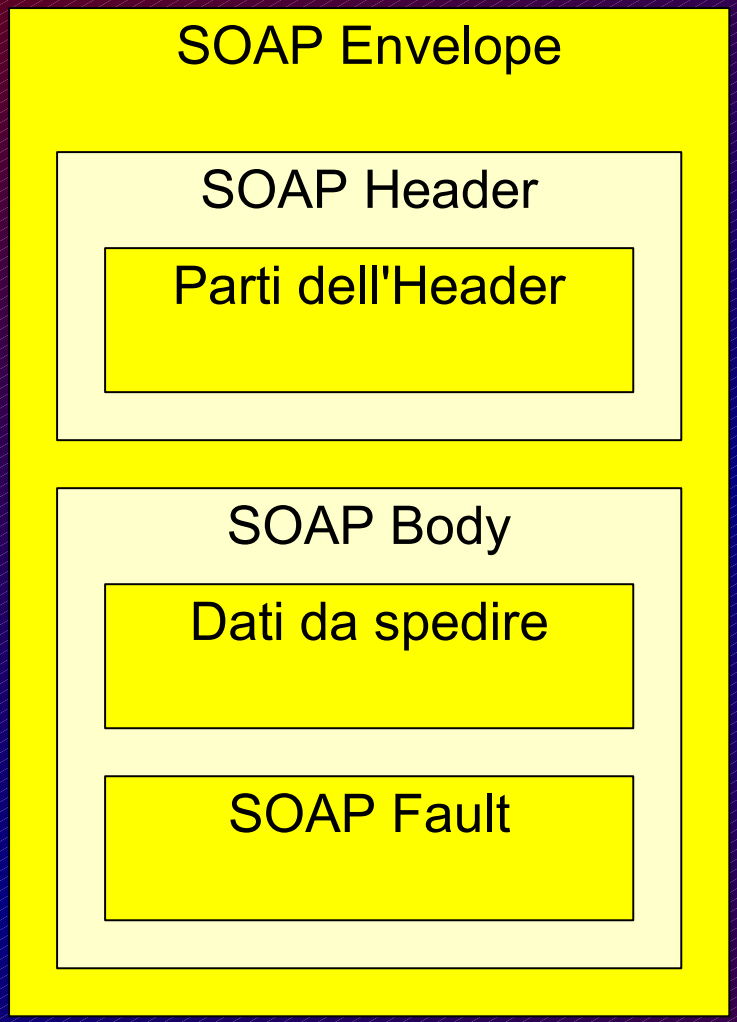
- Un messaggio SOAP somiglia ad una lettera. Si può quindi schematizzare un documento SOAP come una lettera composta da:
  - Una busta (*Envelope*);
  - Informazioni varie per la spedizione del messaggio (*Header*);
  - Un documento da spedire (*Body*).



# SOAP—Formato dei messaggi

- Un messaggio SOAP è un documento XML consistente in:
  - Un elemento SOAP Envelope (*obbligatorio*) che rappresenta la radice del documento XML; tale elemento DEVE essere presente nel documento, PUO' contenere un Header, DEVE contenere un BODY.
  - Un elemento SOAP Header (*opzionale*) contenente informazioni aggiuntive per la comprensione e la spedizione del messaggio.
  - Un elemento SOAP Body (*obbligatorio*) contenente le informazioni che si intendono far giungere al destinatario. Al suo interno viene inoltre definito l'elemento Fault usato per la gestione degli errori; viene come primo figlio dopo Header.

# SOAP—Formato dei messaggi



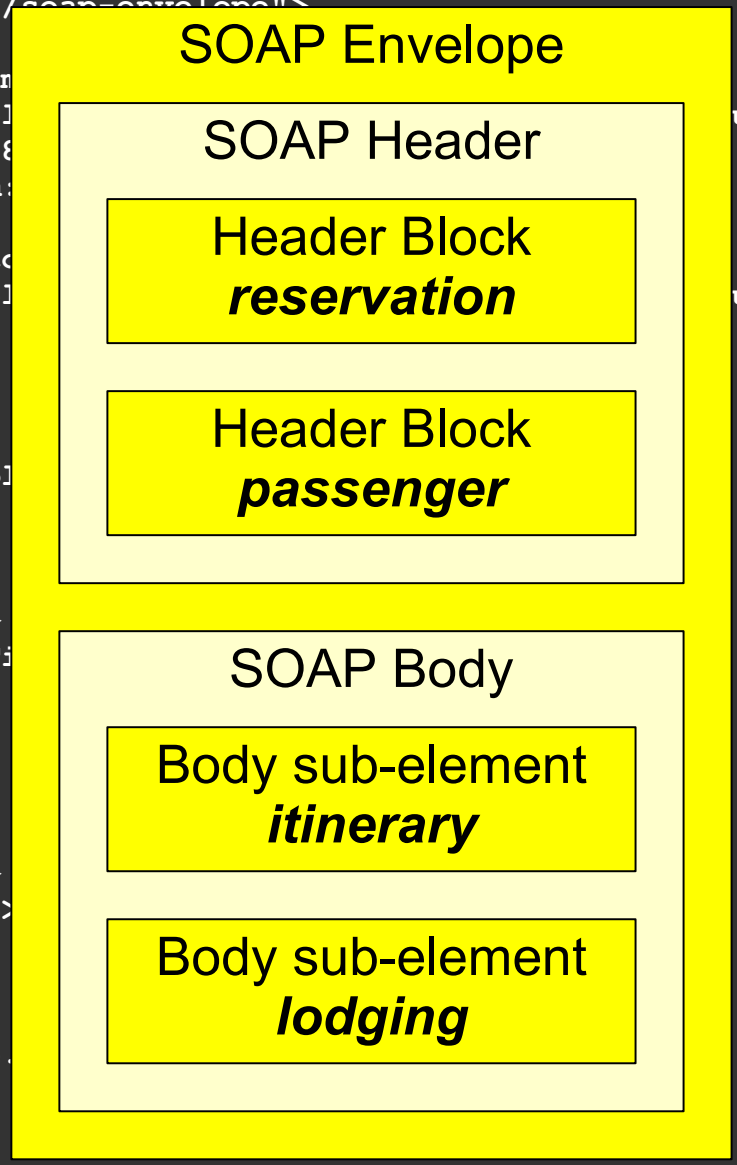

```
<?xml version="1.0"?>
<env:Envelope namespace>

  <env:Header>
    attributi del messaggio
  </env:Header>

  <env:Body>
    messaggio

    <env:Fault>
      gestione errori
    </env:Fault>

  </env:Body>

</env:Envelope>
```

# Esempio

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

# Esempio

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.exam
     env:role="http://www.w3.org/2003/05/soap-envel                    ue">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.co
     env:role="http://www.w3.org/2003/05/soap-envel                    ue">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.exampl
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTi
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travelcompany.example.
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```



# SOAP Envelope

- L'elemento **`Envelope`** identifica l'inizio e la fine del messaggio, in maniera tale che il ricevente sappia quando l'intero messaggio è stato ricevuto.
- Si risolve in questo modo il problema di sapere quando la ricezione è completata e si può iniziare a processare il messaggio.
  - Agisce come meccanismo di packaging.

# SOAP Header

- Le informazioni contenute nell'header:
  - possono essere utilizzate da intermediari SOAP identificati da URI in un qualunque punto del cammino seguito dal messaggio.
  - non devono essere indirizzate necessariamente al destinatario finale.
- L'attributo **`mustUnderstand`** è usato per indicare se una voce dell'Header debba essere elaborata dal destinatario o meno.
  - Se un elemento dell'Header possiede questo attributo con valore "true", il destinatario *deve* elaborarlo correttamente oppure restituire un errore

# SOAP Body

- Il *Body* di un messaggio SOAP può essere visto come il documento vero e proprio che si vuole spedire: questa parte contiene i dati che si intendono mandare al destinatario.
- L'elemento *fault* del body trasporta informazioni riguardanti gli errori occorsi durante la spedizione del messaggio.

# SOAP Body. Elemento fault

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.example.org/timeouts”
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
<env:Body>
 <env:Fault>
   <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
       <env:Value>m:MessageTimeout</env:Value>
     </env:Subcode>
   </env:Code>
   <env:Reason>
     <env:Text xml:lang="en">Sender Timeout</env:Text>
   </env:Reason>
   <env:Detail><m:MaxTime>P5M</m:MaxTime></env:Detail>
  </env:Fault>
</env:Body>
</env:Envelope>
```

# SOAP con HTTP

- Benché SOAP possa utilizzare diversi protocolli di trasporto, HTTP è quello più comunemente usato.
- Le richieste HTTP consistono in un comando HTTP, come **`POST`** o **`GET`**, seguito dall'URL richiesto e dalla versione del protocollo (es. HTTP 1.1).
- Le risposte seguono la semantica di HTTP per fornire il codice dello stato della risposta;
  - Ad esempio, uno status code del tipo 2xx indica che la richiesta è stata ricevuta, capita, accettata, ecc.

# SOAP con HTTP

- Esistono due modelli per lo scambio di messaggi SOAP via HTTP:
  - il modello “SOAP request-response” in cui il metodo POST viene utilizzato per portare i messaggi SOAP nel Body delle richieste/risposte http;
  - il modello “SOAP response” in cui nelle richieste http viene usato il metodo GET per ottenere il messaggio SOAP ed inserirlo nel Body della risposta.
- Utilizzando SOAP con HTTP si devono ricordare di settare l’header **`Content-Type`** come **`application/SOAP+xml`**

# SOAP con HTTP GET

Richiesta HTTP GET

```
GET /travel.example.org/reservations?code=F3T  HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
```

Risposta HTTP GET

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: 200
<?xml version='1.0' ?>
<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>...</env:Header>
  <env:Body>...</env:Body>
</env:Envelope>
```

# SOAP con HTTP POST

Richiesta HTTP POST

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: 230
<?xml version='1.0' ?>
 <env:Envelope
      xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header> ... </env:Header>
 <env:Body> ... </env:Body>
</env:Envelope>
```

Risposta HTTP POST simile al caso della GET

# SOAP via mail—Richiesta

```
From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id: <EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

# SOAP via mail—Risposta

```
From: reservations@travelcompany.example.org
To: a.oyvind@mycompany.example.com
Subject: Which NY airport?
Date: Thu, 29 Nov 2001 13:35:11 EST
Message-Id: <200109251753.NAA10655@travelcompany.example.org>
In-reply-to:<EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:itineraryClarifications>
      ...
      ...
   </p:itineraryClarifications>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

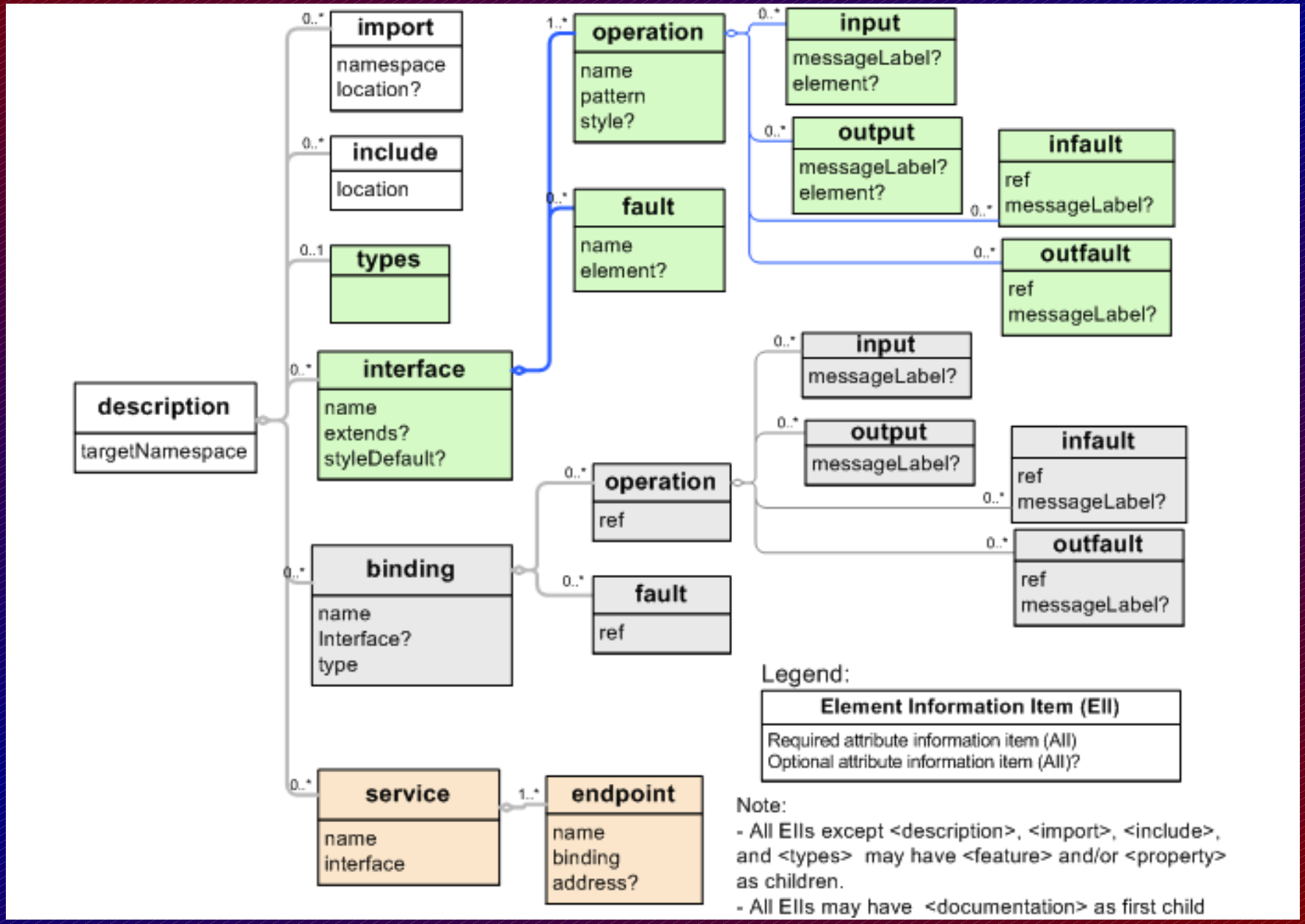
# WSDL

- Il Web Service Description Language (WSDL) è una notazione XML con cui descrivere le interfacce dei Web Services.
  - un documento WSDL fornisce ad un programmatore una descrizione di come può essere instaurata una comunicazione con un determinato servizio; un sistema automatico potrebbe utilizzare questa descrizione per invocare dinamicamente un servizio Web.
  - generare automaticamente, a partire da una descrizione WSDL, il servizio stesso o il client per inoltrare delle richieste al servizio descritto

# WSDL

- WSDL fornisce :
  - un meccanismo comune per rappresentare le operazioni supportare dai WS.
  - la definizione dei tipi di dato da utilizzare nei messaggi.
  - il meccanismo il binding dei messaggi con i protocolli di trasporto da adottare.
- WSDL è progettato per poter essere utilizzato sia nella modalità procedure-oriented (RPC) che document-oriented.

# WSDL


import
namespace
location?
include
location
types
description
targetNamespace
interface
name
extends?
styleDefault?
operation
name
pattern
style?
fault
name
element?
input
messageLabel?
element?
output
messageLabel?
element?
infault
ref
messageLabel?
outfault
ref
messageLabel?
input
messageLabel?
output
messageLabel?
infault
ref
messageLabel?
outfault
ref
messageLabel?
operation
ref
fault
ref
binding
name
Interface?
type
service
name
interface
endpoint
name
binding
address?
Legend:
Element Information Item (EII)
Required attribute information item (AII)
Optional attribute information item (AII)?
Note:
- All EIIs except <description>, <import>, <include>, and <types> may have <feature> and/or <property> as children.
- All EIIs may have <documentation> as first child

# Elementi principali di WSDL

- Un documento WSDL utilizza i seguenti elementi per descrivere un servizio:
  - L'elemento **`types`** descrive i tipi di messaggi che il servizio invia e riceve
  - L'elemento **`interface`** descrive le funzionalità astratte fornite dal WS
  - L'elemento **`binding`** descrive *come* accedere al servizio
  - L'elemento **`service`** descrive *dove* è possibile accedere al servizio

# WSDL by example

- Consideriamo un ipotetico hotel, che voglia fornire una interfaccia WS per implementare le due seguenti funzionalità
  - **`CheckAvailability`**: controlla la disponibilità di una camera. Ovvorre specificare: data di arrivo (check-in date), data di partenza (check-out date), tipo di stanza. Questo metodo ritorna un numero reale (il costo della stanza al giorno). se c'è una stanza disponibile, zero altrimenti.
  - **`MakeReservation`**: Riceve in input anche cognome, nome e numero di carta di credito.

# WSDL by example

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/2006/01/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/2006/01/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx= "http://www.w3.org/2006/01/wsdl-extensions">

  <documentation>
    <!-- Documentazione -->
  </documentation>

  <types>
    <!-- Dichiarazione tipi -->
  </types>

  <interface>
    <!-- Dichiarazione interfacce -->
  </interface>

  <binding>
    <!-- Binding -->
  </binding>

  <service>
    <!-- Dichiarazione servizi -->
  </service>

</description>
```

# <types>

- Mediante il tag **<types>** è possibile definire i tipi di dato utilizzati negli scambi di messaggi
- WSDL utilizza in genere **XML Schema** per la dichiarazione dei tipi

# WSDL passo passo / types

```
<types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="checkAvailability" type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>

    </xs:schema>
  </types>
```

# <interface>

- A WSDL 2.0 interface defines the *abstract interface* of a Web service as a set of abstract operations, each operation representing a simple interaction between the client and the service.
  - Each operation specifies the types of *messages* that the service can send or receive as part of that operation.
  - Each operation also specifies a *message exchange pattern* that indicates the sequence in which the associated messages are to be transmitted between the parties.
  - For example, the **`in-out`** pattern indicates that if the client sends a message in to the service, the service will either send a reply message back out to the client (in the normal case) or it will send a fault message back to the client (in the case of an error).

# WSDL passo passo / interface

```
<interface  name = "reservationInterface" >

    <fault name = "invalidDataFault"
            element = "ghns:invalidDataError"/>

    <operation name="opCheckAvailability"
            pattern="http://www.w3.org/2006/01/wsdl/in-out">
        <input messageLabel="In"
              element="ghns:checkAvailability" />
        <output messageLabel="Out"
              element="ghns:checkAvailabilityResponse" />
        <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>

  </interface>
```

# <binding>

- Although we have specified *what* abstract messages can be exchanged, we have not yet specified *how* those messages can be exchanged. This is the purpose of a **binding**.
  - A binding specifies concrete message format and transmission protocol details for an interface, and must supply such details for every operation and fault in the interface.
  - In the general case, binding details for each operation and fault are specified using operation and fault elements inside a binding element.

# WSDL passo passo / binding

```
<binding name="reservationSOAPBinding"
  interface="tns:reservationInterface"
  type="http://www.w3.org/2006/01/wsdl/soap"
  wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

  <operation ref="tns:opCheckAvailability"
    wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

  <fault ref="tns:invalidDataFault"
    wsoap:code="soap:Sender"/>

</binding>
```

# <service>

- Now that our binding has specified *how* messages will be transmitted, we are ready to specify *where* the service can be accessed
- A WSDL 2.0 **service** specifies a single interface that the service will support, and a list of *endpoint locations* where that service can be accessed.
- A service is only permitted to have one interface.

```
<service name="reservationService"
     interface="tns:reservationInterface">

 <endpoint name="reservationEndpoint"
   binding="tns:reservationSOAPBinding"
   address="http://greath.example.com/2004/reservation"/>

</service>
```

# <documentation>

- Documentation should explain the purpose and use of the service, the meanings of all messages, constraints on their use, and the sequence in which operations should be invoked.
- The **documentation** element allows the WSDL 2.0 author to include some human-readable documentation inside a WSDL 2.0 document

```
<documentation>
  This document describes the GreatH Web service.  Additional
  application-level requirements for use of this service --
  beyond what WSDL 2.0 is able to describe -- are available
  at http://greath.example.com/2004/reservation-documentation.html
</documentation>
```

# REST

- REST: REpresentational State Transfer
  - Utilizza le caratteristiche e i protocolli del World Wide Web come modelli architetturali
  - http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Non è un protocollo
  - E' possibile costruire servizi RESTful mediante HTTP oppure SOAP

# REST: Principi fondamentali

- Ogni risorsa è rappresentata da un URI

```
Un job Grid in esecuzione sull'host cream-01.pd.infn.it
http://cream-01.pd.infn.it/activities/CREAM01324

Lo stato del job di cui sopra
http://cream-01.pd.infn.it/activities/CREAM01324/status

La descrizione del job di cui sopra
http://cream-01.pd.infn.it/activities/CREAM01324/submitted
```

# REST: Principi fondamentali

- E' possibile avere rappresentazioni multiple della stessa risorsa

```
GET /activities/CREAM01324/submitted HTTP/1.1
Host: cream-01.pd.infn.it
Accept: text/xml

HTTP/1.1 200 OK
Content-Type: text/xml
...
Rappresentazione JSDL del job
...
```

```
GET /activities/CREAM01324/submitted HTTP/1.1
Host: cream-01.pd.infn.it
Accept: text/plain

HTTP/1.1 200 OK
Content-Type: text/plain
...
Rappresentazione Classad(JDL) del job
...
```

# REST: Principi fondamentali

- Interfaccia uniforme

```
POST /activities/CREAM01324/status HTTP/1.1
Host: cream-01.pd.infn.it
Content-length: XXX
Cotnent-type: text/plain

CANCELLED
```

# Riferimenti


- *Web Services Glossary*, http://www.w3.org/TR/ws-gloss/
- *Web Services Description Working Group,* http://www.w3.org/2002/ws/desc/
- *SOAP Version 1.2 Part 0: Primer* (Second Edition), W3C Recommendation 27 April 2007, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/
- *Web Services Description Language (WSDL)* Version 2.0 Part 0: Primer, W3C Recommendation 26 June 2007, http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/
- S. Andreozzi, M. Marzolla, *A RESTful Approach to the OGSA Basic Execution Service Specification*, INFN Technical Note INFN/TC_08/4, september 22, 2008 http://www.dsi.unive.it/~marzolla/papers/infn_tc_08_4_besrest.pdf