

SOAP e i Web Services



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

Ringraziamenti

- Corso di Tecnologie Web, prof. Fabio Vitali, Università di Bologna

Introduzione ai Web Service / 1

- Il Web ha fornito il suo primo supporto alle attività umane attraverso un meccanismo di scambio di dati testuali e grafici.
 - Questo livello di interazione risulta soddisfacente in molti casi per l'uomo (ad esempio per leggere e cercare informazioni), ma non consente una reale interazione fra software.
 - Risulta necessario un metodo che consenta alle applicazioni di interagire tra di loro, e di eseguire automaticamente operazioni che dovrebbero altrimenti essere impostate a mano.
- 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.

Introduzione ai Web Service / 2

- I Web Service consentono una comunicazione fra programmi sfruttando XML come "lingua franca":
 - utilizzando un documento XML come messaggio, un programma può spedire una richiesta ad un altro Web Service sulla rete (ed eventualmente ottenere risposta).
 - Lo standard dei Web Service definisce il formato del messaggio, specifica l'interfaccia verso la quale il messaggio deve essere spedito, definisce meccanismi per pubblicare e rintracciare le interfacce Web Service.
- I campi di applicazione dei WS sono molteplici:
 - facilitare l'accesso ad applicazioni Internet per la prenotazione di servizi,
 - possono essere usati nell'ambito del B2B,
 - aiutare nel risolvere problemi di integrazione, fornire una piattaforma di collegamento fra applicazioni diverse.

Introduzione ai Web Service / 3

- I Web Service forniscono un meccanismo standard per interfacciare la rete con sistemi software di back-end (ad esempio database management systems, J2EE, .NET, CORBA, Enterprise Resource Planning—ERP).
- Le interfacce Web Service ricevono messaggi XML dalla rete, trasformano i dati secondo le esigenze delle applicazioni back-end, e eventualmente forniscono dei messaggi di risposta.
- L'implementazione software dei Web Service può essere fatta utilizzando un qualunque linguaggio di programmazione.
- I Web Service non si propongono solo come interfaccia verso servizi, oggetti, applicazioni. Attraverso la composizione di Web Service si può infatti giungere a creare nuovi più complessi servizi Web.

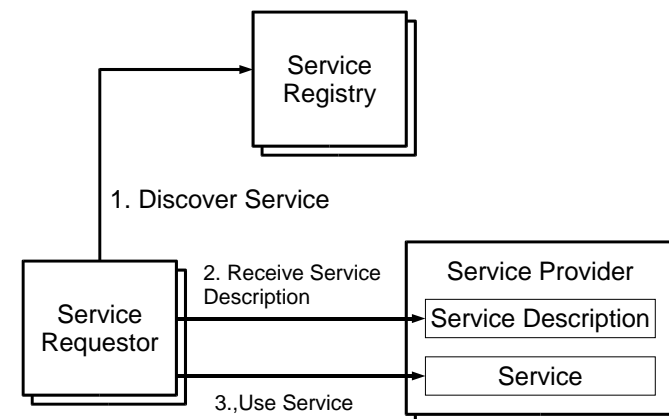
Requisiti dei Web Service

- Programmi che interagiscono fra di loro sul Web devono:
 - Essere in grado di “trovarsi” sul Web.
 - Capire quali informazioni si possono scambiare per poter comunicare.
 - Negoziare il livello della qualità del servizio, i meccanismi di sicurezza o di affidabilità, ecc.
- Alcuni di questi aspetti vengono presi in considerazione da tecnologie già esistenti e consolidate, mentre altre no.
 - La definizione della struttura dei WS è un processo tutt'ora in evoluzione. Questa infrastruttura viene pensata per essere estensibile, in maniera tale che nuovi standard possano essere introdotti facilmente.

Ruoli nella architettura WS

- All'interno dell'architettura WS possono essere individuati tre ruoli principali:
 - Il fornitore di un servizio, che espone un'interfaccia e mette a disposizione un insieme di operazioni che possono essere invocate da client.
 - Il consumatore del servizio, che inoltra delle richieste ad un fornitore per avere un determinato servizio.
 - Un servizio di registrazione che mette a disposizione un meccanismo per consentire al fornitore e al consumatore di trovarsi e comunicare.

Il WEB “automatizzato”



Interazione con Web Service / 1

- Lo standard e la tecnologia WS in generale forniscono due modalità di interazione fra applicazioni:
 - Remote Procedure Call (*online*). In questo caso, una richiesta verso un WS ha la forma di chiamata a metodo o a procedura remota. La richiesta e la risposta sono disposte come scambio di messaggi in maniera sincrona: l'applicazione che invia il messaggio rimane in attesa della risposta
 - Document Oriented (*batch*). In questo caso la richiesta verso un WS ha la forma di un documento XML completo che deve essere processato. Il messaggio inviato viene accodato e processato in maniera asincrona. Successivamente chi fornisce il servizio comunica una risposta (ad esempio via email)

Interazione con Web Service / 2

- L'interazione di tipo Document Oriented tipicamente richiede un agreement fra le parti nella definizione dei documenti che devono essere processati (ad esempio ordini, fatture ecc.).
- Tali parti si accordano anche sui processi di elaborazione dei dati per lo scambio dei documenti (es. uso degli acknowledgement, o invio di email di avviso).

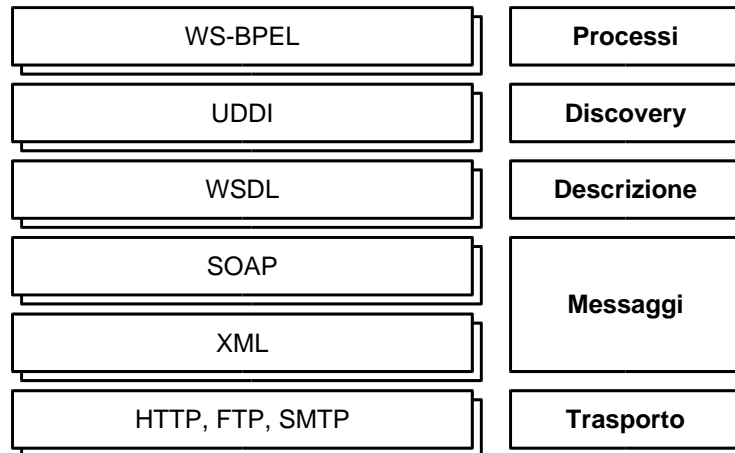
Tecnologie dei Web Service / 1

- 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*), la base sulla quale sono fondati i Web Service, fornisce un linguaggio per definire i dati e processare i dati.
 - **WSDL** (*Web Service Description Language*), un formato XML per definire le interfacce dei WS.
 - **SOAP** (*Simple Object Access Protocol*), un formato XML che fornisce un meccanismo di packaging dei messaggi, attraverso la definizione di una "busta" per la creazione e trasmissione dei messaggi XML.
 - **UDDI** (*Universal Description, Discovery and Integration*), un meccanismo per registrare, categorizzare e rintracciare le interfacce WS.

Tecnologie dei Web Service / 2

- Tali tecnologie vengono utilizzate insieme: una volta che la specifica WSDL di un WS è stata ottenuta o rintracciata grazie a un server UDDI, un messaggio SOAP viene utilizzato per la comunicazione con il WS stesso. Altre tecnologie nascono per espandere le funzionalità relativamente ad altri aspetti.

Stack dei WS



Scenario d'uso

- Un programma che deve inviare un documento a un Web Service:
 - Trasforma i suoi dati in input (esempio un file strutturato) in un documento XML, secondo le esigenze del WS.
 - Invia il documento in una busta SOAP ad un URL utilizzato come indirizzo della risorsa WS sul Web. Tale WS è descritto tramite un documento WSDL accessibile al computer che fa richiesta del servizio e utilizzato per creare il messaggio di richiesta.
 - Quando il WS riceve un documento, lo parserizza e lo valida, verifica aspetti relativi alla sicurezza, e processa il documento immettendolo in un flusso di elaborazione.

WS e XML

- XML è una famiglia di tecnologie: XML 1.0, XML Schema, XML namespace, Xpointer, Xpath, Xlink, XSLT, DOM
- Nel contesto WS la famiglia di tecnologie XML viene utilizzata:
 - per specificare il formato dei messaggi da scambiare
 - per consentire la validazione stessa dei dati scambiati
 - per definire gli stessi WS
- Si richiede sempre la definizione di un agreement sul significato degli elementi XML utilizzati.

SOAP / Introduzione

- Il Simple Object Access Protocol (SOAP) è una delle specifiche più importanti nelle tecnologie WS
 - 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.
- Per gestire correttamente tali messaggi, un server HTTP (Apache o IIS) deve disporre di un SOAP processor.

SOAP / Introduzione

- La specifica del W3C definisce:
 - Come deve essere strutturato il messaggio SOAP
 - Cosa rappresentano i dati contenuti nel messaggio
 - Le regole di Binding verso un protocollo di trasporto (ad esempio HTTP)
 - Come eseguire attraverso SOAP una chiamata a procedura remota (RPC)
- I messaggi SOAP utilizzano una sintassi XML,
 - I SOAP Processor deve qualificare elementi e attributi
 - I messaggi SOAP non devono contenere né DTD, né Processing Instruction

SOAP per scambiare messaggi

- Lo scambio di messaggi tramite SOAP è una trasmissione da un mittente ad un destinatario attraverso zero o più intermediari, anche se, dipendentemente dalla natura dell'operazione che si vuole effettuare, spesso è necessaria anche una risposta da parte del destinatario.
- Un intermediario SOAP è un'applicazione capace sia di ricevere che di inoltrare messaggi SOAP

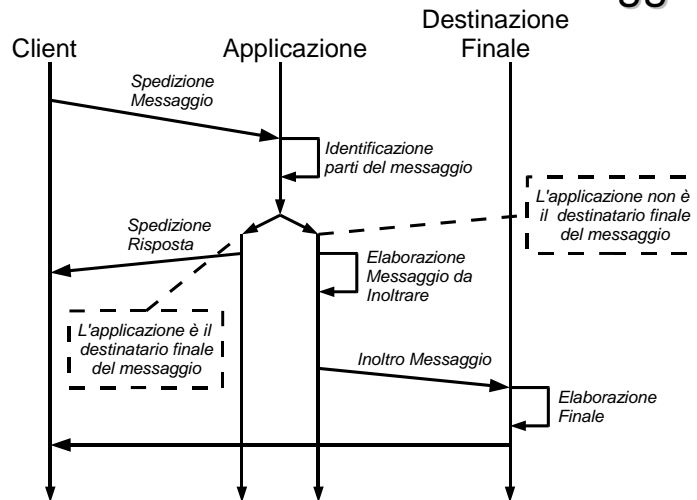
SOAP e protocolli di trasporto

- SOAP abilita una connessione application-to-application e può potenzialmente essere utilizzato con una vasta gamma di protocolli di trasporto;
- Per ovviare a problemi con firewall o proxy, si è comunque soliti utilizzare SOAP mediante HTTP (nelle specifiche viene descritto proprio come combinarlo con HTTP e SMTP).
- Questo significa che un messaggio SOAP viene spedito come parte di una richiesta (o risposta) HTTP. Un'altra ragione che porta a questa scelta nell'utilizzo di SOAP è che praticamente ogni computer connesso ad una rete supporta traffico HTTP.
- Utilizzando HTTP e XML, SOAP cerca di garantire una comunicazione semplice e leggera, in termini computazionali, tra due applicazioni eseguite indipendentemente dalla piattaforma e connesse mediante una infrastruttura già esistente.

SOAP – elaborazione dei messaggi

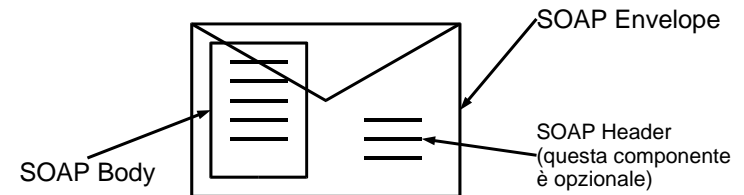
- Un'applicazione SOAP che riceve un messaggio deve elaborare il messaggio compiendo in ordine le seguenti azioni:
 - Identificare tutte le parti del messaggio;
 - Verificare che tutte le parti identificate nel passo precedente siano supportate dall'applicazione alla quale è rivolto il messaggio ed elaborarle. L'applicazione può ignorare le parti identificate come opzionali nel primo passo senza compromettere l'elaborazione del resto del messaggio;
 - Se l'applicazione SOAP alla quale è arrivato il messaggio non è la destinazione finale del messaggio stesso, elabora il messaggio aggiungendo od eliminando informazioni in modo da inoltrare soltanto i dati significativi per il destinatario successivo.

SOAP – elaborazione dei messaggi



SOAP – formato dei messaggi

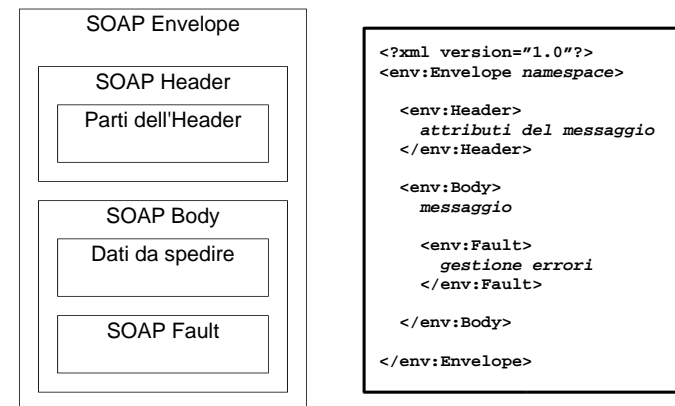
- Un messaggio SOAP somiglia ad una lettera. Si può quindi schematizzare un documento SOAP come una lettera composta da:
 - Una busta (*Envelope*);
 - Informazioni varie per la spedizione del messaggio (*Header*);
 - Un documento da spedire (*Body*).



SOAP – formato dei messaggi

- Più formalmente 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; è il primo figlio dell'envlope; può avere elementi figli.
 - 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



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:093a2dal-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 Jögvan Öyvind</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.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next" env:mustUnderstand="true">
      <m:reference>uuid:093a2dal-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 Jögvan Öyvind</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 Envelope

SOAP Header

Header Block
reservation

Header Block
passenger

SOAP Body

Body sub-element
itinerary

Body sub-element
lodging

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 / 1

- Nel messaggio SOAP può nascere l'esigenza di inserire informazioni non facenti parte del messaggio ma fondamentali per la spedizione del messaggio stesso (legati ad esempi a aspetti di sicurezza, transazioni, ecc.).
- SOAP consente inoltre l'estensione delle informazioni del messaggio: è possibile definire, oltre agli attributi standard, dei propri attributi.
- Gli elementi figli dell'header sono definiti da un nome composto da URI di namespace e nome locale.

SOAP Header / 2

- 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.

SOAP Header / 3

- L'attributo `role` (il cui valore è un URI) indica il ruolo del destinatario dell'elemento dell'header con tale attributo (se l'elemento non è presente il destinatario è il destinatario finale del messaggio).
- SOAP fornisce tre predefiniti ruoli per un nodo:
 - `next`: identifica tutti i nodi intermediari e il destinatario finale.
 - `none`: dichiara che il contenuto dell'elemento non può essere processato da nessun nodo.
 - `ultimateReceiver`: identifica il destinatario finale (di default se non è specificato nessun ruolo).
- SOAP non specifica come un nodo può assumere un particolare ruolo.

SOAP Header / 4

- L'attributo `mustUnderstand` è usato per indicare se una voce dell'Header debba essere elaborata dal destinatario o meno.
 - Il valore dell'attributo può essere "true" o "false". L'assenza di questo attributo è semanticamente equivalente alla sua presenza con valore "false".
- Se un elemento dell'Header possiede questo attributo con valore "true", il destinatario *deve* elaborarlo correttamente oppure restituire un errore

```
<env:Header>
  <t:transaction
    xmlns:t="http://thirdparty.example.org/transaction"
    env:encodingStyle="http://example.com/encoding"
    env:mustUnderstand="true" >5</t:transaction>
</env:Header>
<!-- ... -->
</env:Header>
```

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.
- Gli elementi figli del body sono definiti da un nome composto da URI di namespace e nome locale.
- L'elemento *fault* del body trasporta informazioni riguardanti gli errori occorsi durante la spedizione del messaggio.
 - Nel caso sia presente, fault deve apparire come una voce del Body e non può comparire più di una volta.

SOAP Body. Elemento fault

- Sono definiti cinque sottoelementi:
 - **env:Code**: elemento obbligatorio che definisce un codice per l'identificazione automatica dell'errore. Ha due elementi figli: `env:Value` obbligatorio e `env:SubCode` opzionale.
 - **env:Reason**: fornisce una descrizione, comprensibile da un utente, del tipo di errore. Anche questo elemento deve essere presente all'interno di `Fault`. Ha uno o più elementi figli chiamati `Text` ognuno dei quali deve contenere un valore diverso nell'attributo obbligatorio `xml:Lang`.
 - **env:Node**: fornisce informazioni sul nodo che ha causato l'errore che viene identificato attraverso un URI.
 - **env:Detail**: fornisce informazioni specifiche riguardanti gli errori. Può avere un numero qualsiasi di figli.
 - **env:Role**: è opzionale ed indica il ruolo che ricopriva il nodo che ha causato l'errore

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 2 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**; il parametro `charset` è opzionale e può assumere i valori `utf-8` o `utf-16`.

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.ovvind@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>uid:093a2dal-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>like J6gván @yvind</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:reference>none</q:reference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

SOAP via mail—Risposta

```
From: reservations@travelcompany.example.org
To: a.ovvind@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>uid:093a2dal-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>like J6gván @yvind</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>
```

SOAP e RPC

- Come nei casi precedenti, la rappresentazione delle RPC (*Remote Procedure Call*) di SOAP non è legata ad un unico protocollo, ma generalmente viene utilizzato HTTP.
- Per invocare una RPC sono necessari:
 - L'URL del nodo al quale verrà effettuata la chiamata;
 - Il nome del metodo;
 - I parametri in input da spedire e quelli in output che si aspetta come risposta;
 - Il modello dello scambio di messaggi ed un identificatore del metodo web (per es. "GET" o "POST"), che verranno utilizzati;
 - Eventuali informazioni aggiuntive (come, ad esempio, un check di controllo per la sicurezza) contenute negli header.

WSDL

- Il Web Service Description Language (WSDL) è un formato XML che definisce il meccanismo, platform-independent, per descrivere le interfacce dei Web Service.
 - Un documento WSDL contiene sostanzialmente delle definizioni.
- WSDL si propone per vari utilizzi base:
 - 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 è stato sviluppato originariamente da Microsoft e IBM e sottomesso all'approvazione del W3C da 25 compagnie. WSDL fa parte del cuore dell'architettura WS
- WSDL fornisce :
 - un meccanismo comune per rappresentare le operazioni supportate 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.

Caratteristiche di WSDL / 1

- Ciascun dei tre componenti di WSDL può essere specificato in file separati e combinato in vari modi per generare una descrizione di un WS.
 - La descrizione WSDL deve essere disponibile a entrambi gli attori (chi richiede il servizio e chi lo fornisce) di una transizione WS.
- Il binding consente di associare a messaggi o operazioni uno o più meccanismi di trasporto.
 - WSDL è progettato come una piattaforma XML estensibile per gestire diversi tipi di dato, operazioni, protocolli di trasporto e definizione di messaggi.
 - WSDL "nasconde" dietro ad un formato comune la effettiva implementazione del servizio (che può adottare, ASP, servlet per l'interfaccia per il Web, .NET, CORBA o altro per il sistema di back.end)

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:wssoap= "http://www.w3.org/2006/01/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsi="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>
```

More

49

WSDL—elemento radice

- WSDL fornisce definizioni di servizi:
 - l'elemento radice del documento è <definitions>.
- Tale elemento
 - ha un attributo **name** che contiene il nome del documento, come forma soprattutto di documentazione.
 - definisce i namespace utilizzati per consentire di fare riferimento alle estensioni di WSDL o ad altre specifiche.
 - può avere come figli 5 elementi: *types*, *message*, *portType* (definizioni astratte), *bindings*, *service*, (definizioni concrete)

Moreno Marzolla

Tecnologie Web

50

WSDL passo passo / description

```
<?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"
  . . . >
  . . .
</description>
```

Moreno Marzolla

Tecnologie Web

51

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>
```

Moreno Marzolla

Tecnologie Web

52

<types>

- Mediante il tag <types> è possibile definire i tipi di dato utilizzati negli scambi di messaggi
- Tutti i tipi di dato vengono dichiarati mediante l'attributo `type` del tag <element>
 - Tipi di dato complessi devono essere dichiarati separatamente con un proprio nome, e tale nome può quindi venire impiegato come valore dell'attributo `type`
- WSDL utilizza in genere XML Schema per la dichiarazione dei tipi, anche se in teoria è possibile utilizzare qualunque altra notazione

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"
    style="http://www.w3.org/2006/01/wsdl/style/iri"
    wsdlx:safe = "true">
    <input messageLabel="In"
      element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>
</interface>
```

<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 / binding

```
<binding name="reservationSOAPBinding"
  interface="tns:reservationInterface"
  type="http://www.w3.org/2006/01/wsdl/soap"
  wssoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
  <operation ref="tns:opCheckAvailability"
    wssoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
  <fault ref="tns:invalidDataFault"
    wssoap:code="soap:Sender"/>
</binding>
```

<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 / service

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

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

</service>
```

<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.
- Each endpoint must also reference a previously defined binding to indicate what protocols and transmission formats are to be used at that endpoint. A service is only permitted to have one interface.

WSDL passo passo / documentation

```
<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>
```

<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