Tecniche di Programmazione avanzata

Corso di Laurea Specialistica in Ingegneria Telematica
Università Kore – Enna – A.A. 2009-2010

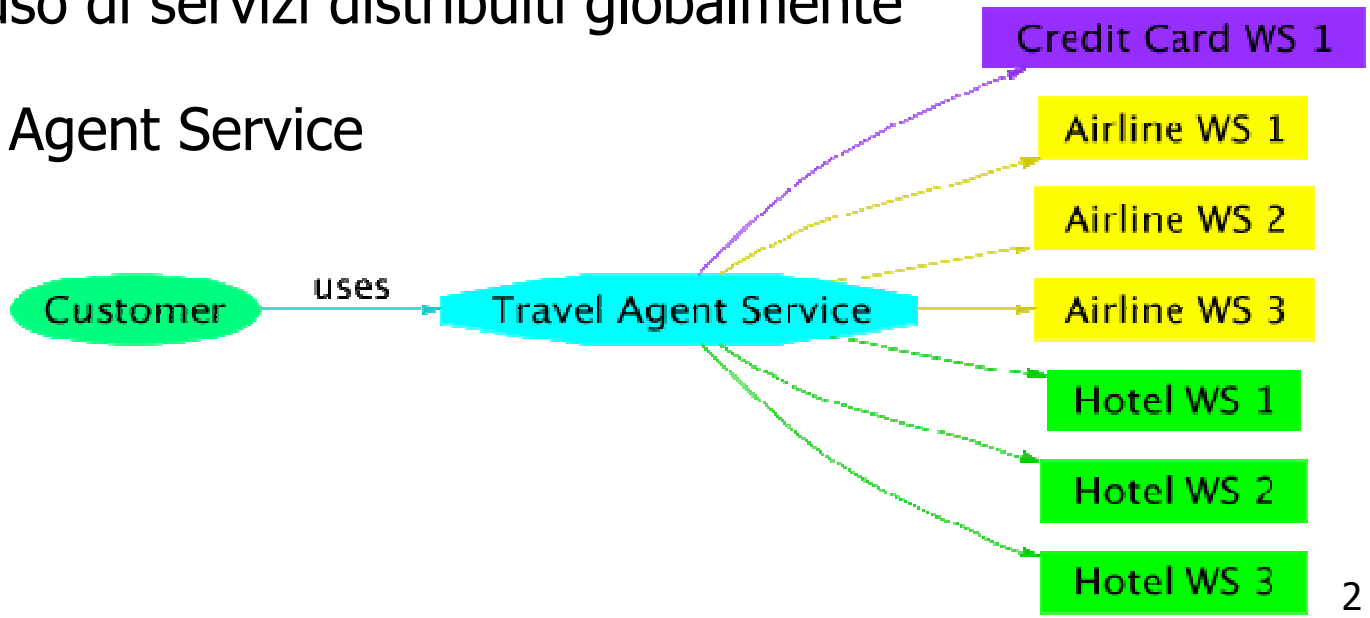
Alessandro Longheu
http://www.diit.unict.it/users/alongheu
alessandro.longheu@diit.unict.it

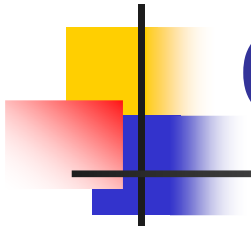
Web Services

Perchè

- Integrazione di sistemi distribuiti ed eterogenei
  - Distribuiti globalmente
  - Diversi linguaggi di programmazione
  - Differenti librerie di sviluppo (API)
- Supporto sia per applicazioni B2C e B2B
- Permettere l'uso di servizi distribuiti globalmente

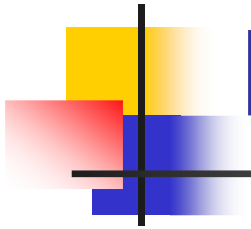
Esempio: Travel Agent Service





Che cos'è un Web Service ?

- Un middleware per applicazioni distribuite
- Uno strumento per le chiamate remote di procedura e lo scambio di dati
- Uno Standard aperto basato su XML
- Uno strumento per "loosely coupled software services"
- Un approccio indipendente dal linguaggio di programmazione e dal sistema operativo
- Uno strumento basato su architetture e protocolli già esistenti e collaudati.



Definizione data dal W3C

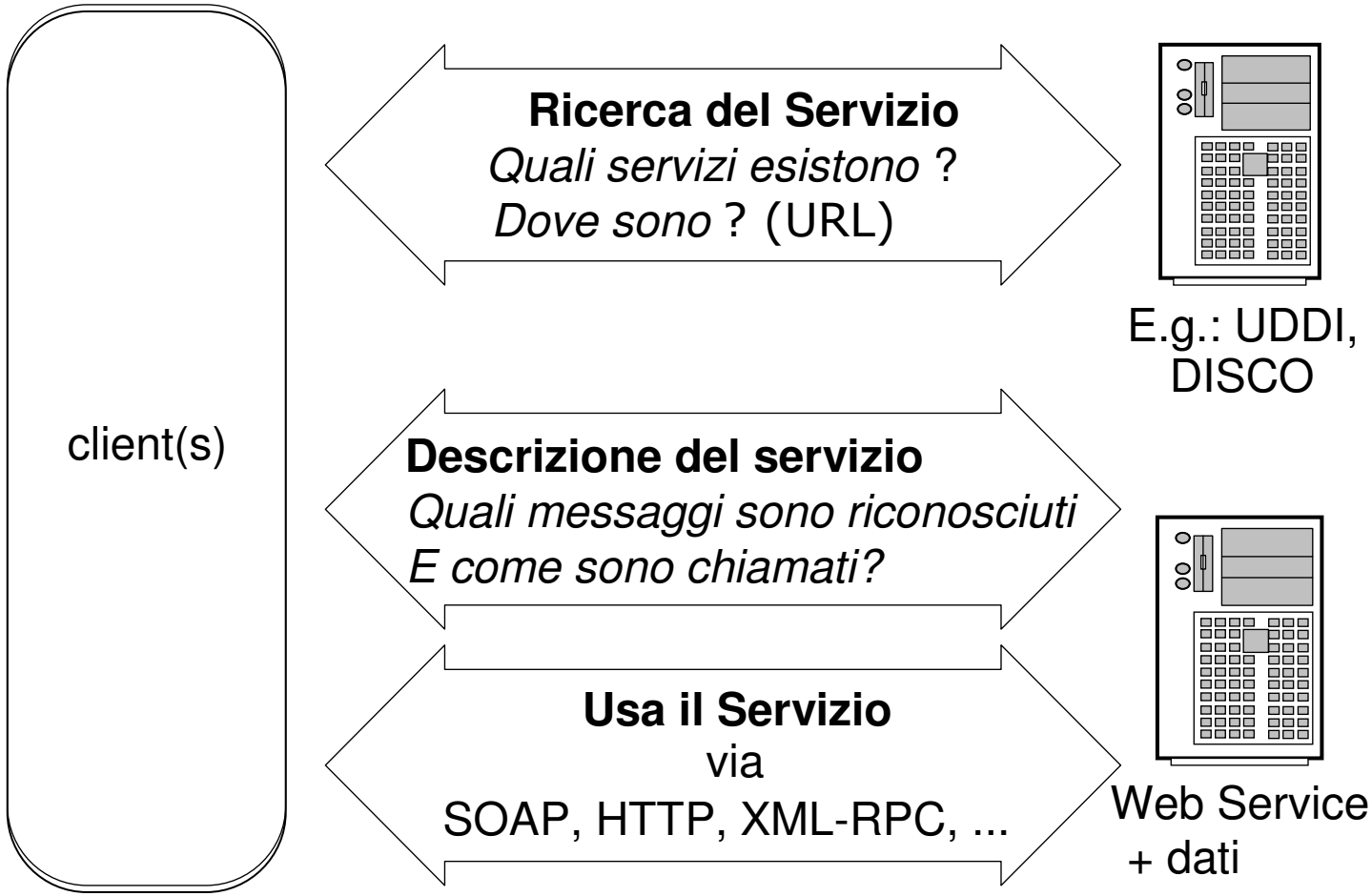
- **Una applicazione software identificata da una URI**
- Descrizione dell'interfaccia in XML e codifica dei messaggi in XML
- Scambio dei messaggi basato sui protocolli Internet
- http://www.w3.org/standards/webofservices/:
  "Web of Services refers to message-based design frequently found on the Web and in enterprise software. The Web of Services is based on technologies such as HTTP, XML, SOAP, WSDL, SPARQL, and others."
- Il **consorzio OASIS** (Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/) ed il **W3C** sono i principali responsabili dell'architettura e della standardizzazione dei Web Service.

Pila Protocollare dei Web Services

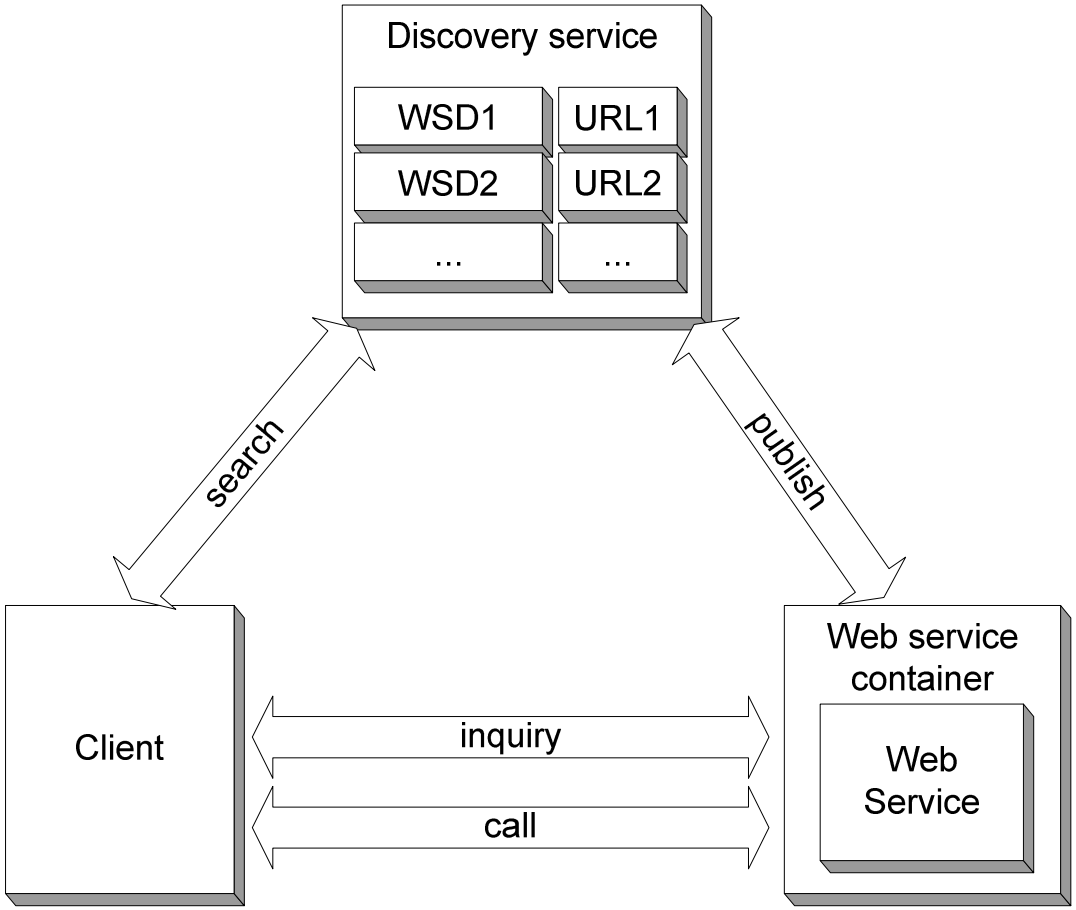
- La pila protocollare dei Web Service è composta da quattro aree:
- **Trasporto del servizio**: che sfrutta i protocolli esistenti: HTTP, SMTP, FTP, ed i più recenti e XMPP (messaggistica istantanea XML based) e BEEP (che permette di crearsi un proprio protocollo).
- **XML Messaging**: il messaggio viene codificato in accordo allo standard SOAP, ma anche utilizzare JAX-RPC, XML-RPC o REST.

- **Descrizione del servizio**: l'interfaccia pubblica di un Web Service viene descritta tramite WSDL (Web Services Description Language) un linguaggio basato su XML usato per la creazione di "documenti" descrittivi delle modalità di interfacciamento ed utilizzo del Web Service.
- **Elencazione dei servizi**: la centralizzazione della descrizione dei Web Service in un "registro" comune permette la ricerca dei Web Service disponibili; a tale scopo viene attualmente utilizzato il protocollo UDDI.

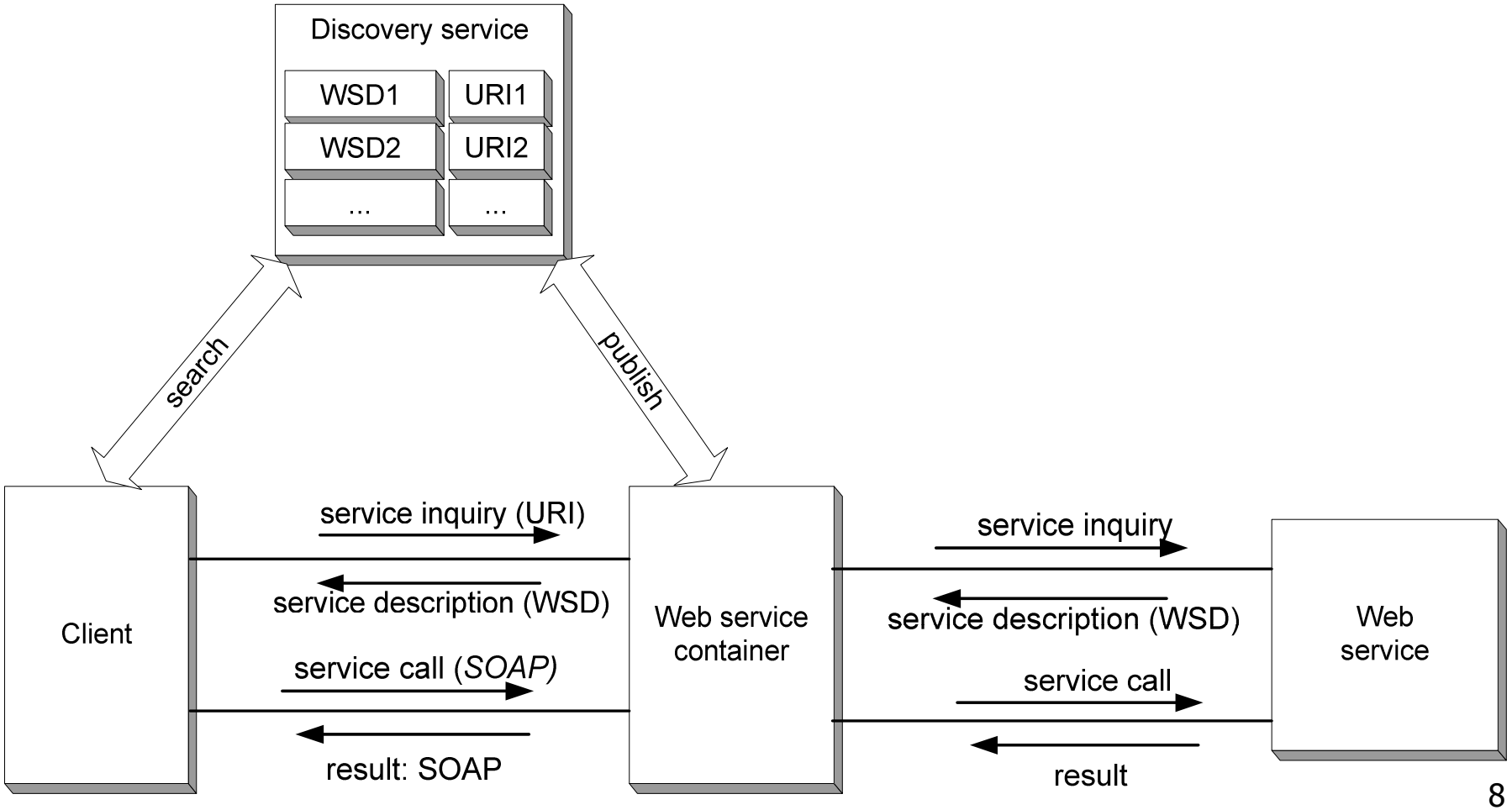
Architettura



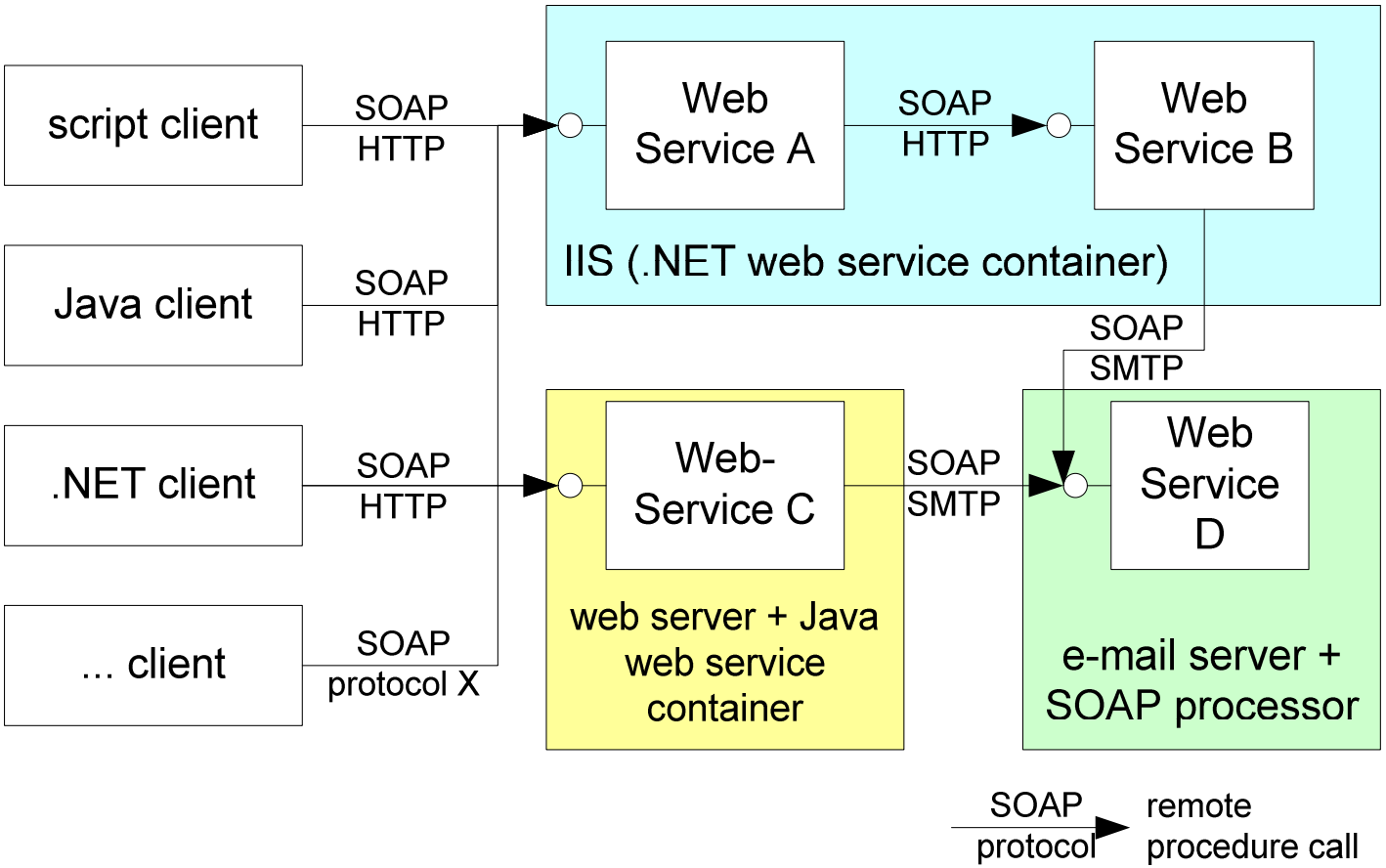
Architettura

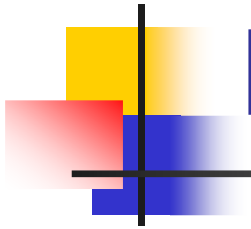


Architettura



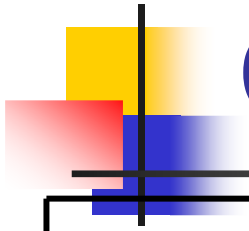
Architettura





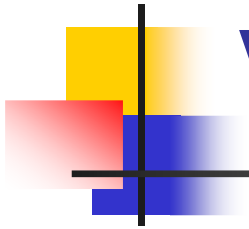
Estensioni ai protocolli

- Oltre i protocolli della pila principale, per i web services sono stati definiti anche:
- **WS-Security**: che permette l'autenticazione degli utenti e la confidenzialità dei messaggi scambiati con l'interfaccia del Web Service
- **WS-Reliability:** che soddisfa la richiesta di messaggi affidabili, critica per alcune delle applicazioni che utilizzano i Web Service (transazioni bancarie o applicazioni di E-commerce).



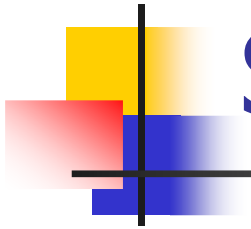
Confronti con altri approcci

	Java RMI	.NET Remoting	CORBA	Web Services
Linguaggi	Java	Linguaggi supportati	Indipendente	Indipendente
Def. Interfaccia	Java Interfacc.	C# Interfacc.	CORBA IDL	WSDL XML-based
Struttura dati	Oggetti Java	Oggetti .NET	Oggetti IDL	XML data
Protocollo trasporto	RMI-IIOP	Binario o OAP	GIOP/IIOP	HTTP, HTTPS, …
Packaging	Serializz. Java	Serializz. .NET	ORB/CDR	SOAP
Infrastrut.	Java RMI	.NET remoting	ORBs	Web, mail, ftp, …



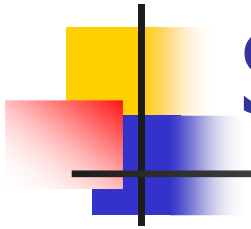
Vantaggi e svantaggi

- Vantaggi
  - Indipendenza dal linguaggio di programmazione, dall'ambiente di esecuzione e dal sistema operativo
  - Costruito sulla infrastruttura internet esistente
  - Standard
  - Sponsorizzato dalle maggiore aziende informatiche (Microsoft, IBM, SAP, Sun)
- Svantaggi
  - Prestazioni (uso dell'XML)
  - Mancanza di standard per applicazioni mission critical (ad esempio transazioni distribuite)
  - Problema di sicurezza (viaggiando su HTTP, i web services passano attraverso i firewall)



SOAP - Caratteristiche

- Un semplice protocollo per lo scambio di messaggi basato su XML
  - Permette di "impacchettare" i dati provenienti da qualsiasi applicazione
  - Definisce il formato di un singolo messaggio (one-way protocol)
  - Asincrono
- Indipendente dal protocollo di trasporto
- SOAP:
  - Non è un modello ad oggetti distribuito
  - Non definisce un protocollo di comunicazione
  - Non impone nè esprime alcuna semantica dei messaggi

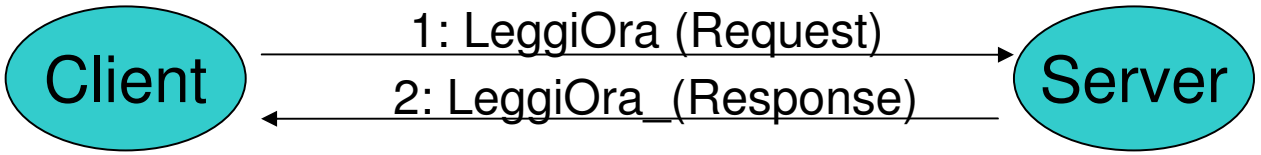


SOAP - Caratteristiche

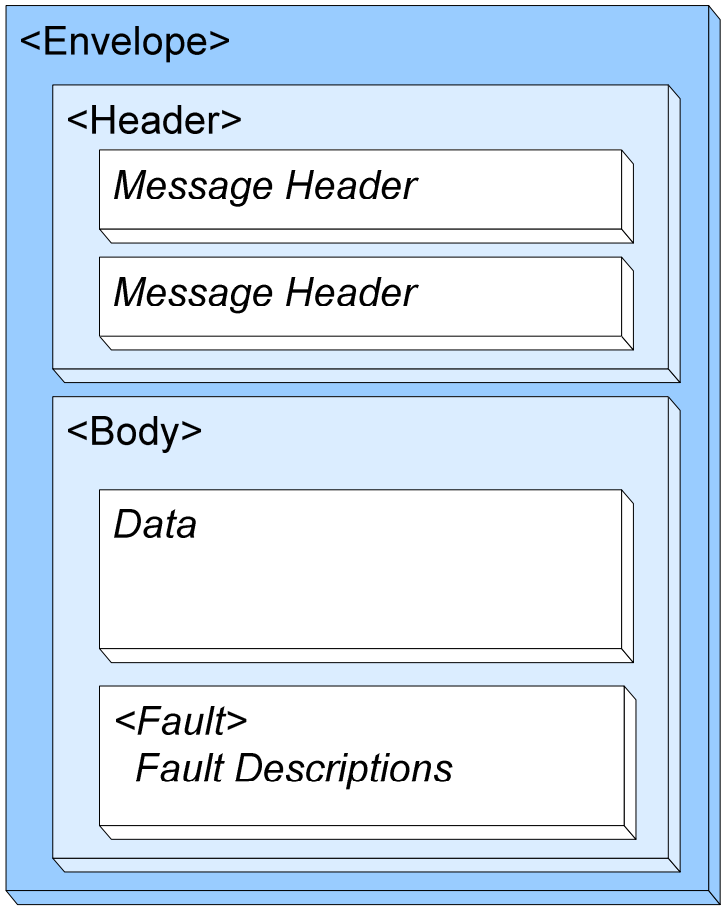
- SOAP non definisce semantiche per i dati e le chiamate, ma fornisce agli sviluppatori i mezzi per farlo.
- Con un intenso uso dei *Namespaces* XML (un messaggio SOAP è composto da elementi provenienti da almeno tre namespaces diversi), SOAP permette agli autori dei messaggi di dichiararne la semantica usando grammatiche XML definite per lo scopo in particolari namespaces.

Applicazioni di SOAP

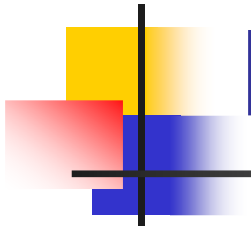
- SOAP è estendibile
  - Per implementare tecniche per la chiamata remota di procedura (RPC)
  - Per implementare modelli di sicurezza
  - Per implementare tecniche di autenticazione
- I protocolli vengono costruiti come una combinazione di messaggi (“message exchanging pattern”)
  - Modelli one-way, request-response, multicast



Formato dei messaggi SOAP



- La busta <envelope> contiene
  - L'intestazione del messaggio <header> che contiene meta-informazioni
  - Il messaggio <body>, che contiene dati in formato XML
  - Messaggi di errore <fault>
- the default namespace for the SOAP envelope is:
- http://schemas.xmlsoap.org/soap/envelope/
- and the default namespace for SOAP encoding and data types is:
- http://schemas.xmlsoap.org/soap/encoding/



Formato dei messaggi SOAP

- Un messaggio SOAP è composto da:
  - Un elemento radice, *envelope*, obbligatorio. Il namespace di SOAP viene dichiarato all'interno di questo elemento
  - Un elemento *header* opzionale. Il suo scopo è quello di trasportare informazioni non facenti parte del messaggio, destinate agli "attori", cioè alle varie parti che il messaggio attraverserà per arrivare al suo destinatario finale.
  - Un elemento *body* obbligatorio. Questo elemento contiene il messaggio vero e proprio.

Formato dei messaggi SOAP

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="http://www.trading.org/ex">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

L'elemento *Envelope* è la radice dei messaggi SOAP.

In namespace di SOAP,**http://schemas.xmlsoap.org/soap/envelope/**, è dichiarato nell'elemento radice *Envelope*.

L'attributo *encodingStyle* punta al sistema si serializzazione standard di SOAP per i tipi, che ha URI http://schemas.xmlsoap.org/soap/encoding/

L'elemento *Body* contiene il messaggio vero e proprio. Al suo interno non ci sono elementi definiti da SOAP, ma importati dal namespace del mittente del messaggio.

Formato dei messaggi SOAP

- Gli headers forniscono un modo **per estendere un messaggio in maniera modulare** e senza la necessità di rinegoziare l'intero formato tra tutte le parti in causa.
- L'elemento *header* deve essere il primo figlio di *envelope*.
- Tutti gli elementi figli di *header* vanno considerati singole **header entries**.
  - Come per il *Body*, non esiste un formato per le header entries: la grammatica va importata da altri namespaces.

Formato dei messaggi SOAP

```
<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:Transaction xmlns:t="http://www.trading.org/tr">15_USER_72</t:Transaction >
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <m:GetLastTradePrice xmlns:m="http://www.trading.org/ex">
   <symbol>DIS</symbol>
  </m:GetLastTradePrice>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un messaggio SOAP "completo".

In questo caso, compare anche l'elemento opzionale *Header*. All'interno dell'header, ogni figlio rappresenta una **header entry**, cioè un blocco di dati distinto destinato all'elaborazione da parte dei destinatari del messaggio.

I figli dell'header sono costruiti usando grammatiche specificate dell'utente. In questo caso, l'unica header entry è un elemento chiamato *Transaction* definito nel namespace http://www.trading.org/tr. Il significato del contenuto di questo elemento sarà noto solo a chi conosce questo namespace.

Formato dei messaggi SOAP

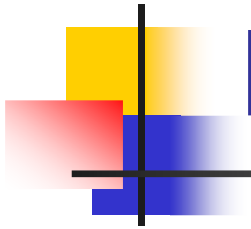
- Un messaggio SOAP può **attraversare varie applicazioni prima di raggiungere la sua destinazione.**
  - Le header entry, oltre che al destinatario finale, possono essere destinate a queste applicazioni "intermedie".
- L'attributo globale *actor* può essere applicato a ogni header entry per specificare, con una URI identificativa, l'applicazione destinataria della entry.
  - Se non viene specificato, l'*actor* predefinito è sempre il destinatario del messaggio.

Formato dei messaggi SOAP

```
<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:Transaction xmlns:t=http://www.trading.org/tr
     SOAP-ENV:actor= "http: //www.diit.unict.it / alongheu" >
    15_USER_72
  </t:Transaction >
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <m:GetLastTradePrice xmlns:m="http://www.trading.org/ex">
    <symbol>DIS</symbol>
  </m:GetLastTradePrice>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

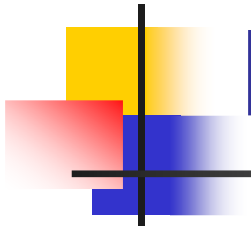
Headers con destinatari.

Mentre nell'esempio precedente i dati nella entry *Transaction* erano da considerarsi indirizzati al destinatario di tutto il messaggio, qui si specifica che andranno elaborati dall'attore rispondente all'URI **"http: //www.diit.unict.it / alongheu"**



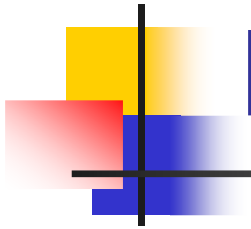
Formato dei messaggi SOAP

- L'attore destinatario di una header entry **deve sempre rimuoverla** prima di inoltrare il messaggio all'attore successivo lungo il path che conduce al destinatario.
  - Un attore può, se vuole, inserire altre header entries nel messaggio, ma non toccarne il corpo.
- La URI speciale http://schemas.xmlsoap.org/soap/actor/next indica che un header dovrà essere elaborato dal primo (o prossimo) attore che incontrerà sul path.



Formato dei messaggi SOAP

- **Il corpo SOAP contiene le informazioni per il destinatario finale del messaggio.**
  - Il contenuto di *Body* è definito dall'implementatore del messaggio tramite l'uso di un particolare namespace.
  - Il destinatario dovrà elaborare i dati e obbedire alla loro semantica, specificata dal namespace di appartenenza.
  - L'elemento *Body* è semanticamente equivalente a una header entry con *mustUnderstand*="1" e *actor* non specificato.



Formato dei messaggi SOAP

- Esiste un solo elemento predefinito in SOAP che può apparire nel *Body* del messaggio: l'elemento *Fault*.
- *Fault* compare nei messaggi SOAP di risposta se il destinatario o un intermediario non sono stati in grado di elaborare la loro parte del messaggio di richiesta.
- Se presente, l'elemento *Fault* deve essere uno dei figli di *Body* e può comparire solo una volta.
- L'elemento *Fault* serve a **fornire informazioni su errori derivanti dall'elaborazione del messaggio**.

Formato dei messaggi SOAP

- Il contenuto dell'elemento *Fault* è:
  - Un elemento *faultcode*, obbligatorio. Fornisce un **codice di identificazione per l'errore**, ad uso del software. SOAP definisce alcuni codici standard per questo elemento.
  - Un elemento *faultstring*, obbligatorio. Fornisce una **descrizione testuale dell'errore**.
  - Un elemento *faultactor*, obbligatorio. Specifica **l'attore che ha generato l'errore**, se non si tratta del destinatario (cioè se l'errore è stato provocato dalla mancata elaborazione di una header entry).

Formato dei messaggi SOAP

- Il messaggio 1 richiede che l'attore identificato dall'URI http://www.aaa.it/test elabori (mustUnderstand=1) il messaggio
- L'attore non è tuttavia riuscito ad elaborare il messaggio, quindi risponde con il messaggio 2 dove indica che si è verificato l'errore

```
<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:Transaction xmlns:t=http://www.trading.org/tr
   SOAP-ENV:actor=  http://www.aaa.it/test
   SOAP-ENV:mustUnderstand="1">
   15_USER_72
  </t:Transaction >
 </SOAP-ENV:Header>
  …
</SOAP-ENV:Envelope>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
  <SOAP-ENV:Fault>
   <faultcode>SOAP-ENV:MustUnderstand</faultcode>
   <faultstring>SOAP Must Understand Error</faultstring>
   <faultactor>  http://www.aaa.it/test  </faultstring>
  </SOAP-ENV:Fault>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Formato dei messaggi SOAP

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <m:GetLastTradePrice xmlns:m="http://www.trading.org/ex">
     <symbol>DIS</symbol>
   </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

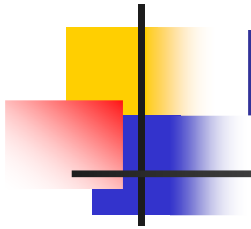
```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        < e:myfaultdetails xmlns:e="http: //www.aaa.it/errornamespace">
              <message>Unknown code</message>
              <errorcode>1001</errorcode>
        </e:myfaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Errore nell'elaborazione del corpo del messaggio.

In questo esempio, il destinatario del messaggio in alto non è riuscito ad elaborarne il corpo.

Il destinatario ha quindi spedito al mittente del messaggio la risposta in basso, indicando l'errore.

Visto che l'errore è generato dal destinatario, l'elemento *faultactor* non è necessario, mentre l'elemento *detail* è obbligatorio e contiene elementi specifici importati da un namespace opportuno per descrivere l'errore.



Formato dei messaggi SOAP

- La codifica dei tipi è utilizzata per la **serializzazione dei valori** nei messaggi.
- La **codifica standard** dei tipi di SOAP è indicata ponendo l'attributo *encodingStyle* di un elemento al namespace http://schemas.xmlsoap.org/soap/encoding/.
- Tuttavia, gli sviluppatori possono definire ed utilizzare anche altre codifiche dei tipi.

Formato dei messaggi SOAP

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <m:SayHelloTo xmlns:m="http://www.aaa.it/test">
      <name xsi:type="xsd:string">Giuseppe</name>
    </m:SayHelloTo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Serializzazione di un valore semplice in un messaggio SOAP.

In questo caso, il corpo del messaggio contiene un valore, accessibile tramite l'accessor *name*, dichiarato di tipo *xsd:string* all'interno del documento.

L'attributo *xsi:type* è importato dal namespace Schema Instance http://www.w3.org/1999/XMLSchema-instance.

Il tipo semplice è importato dal namespace degli Schemi XML, http://www.w3.org/1999/XMLSchema.

Formato dei messaggi SOAP

```
<element name="Book">
 <complexType>
  <element name="author" type="xsd:string"/>
  <element name="preface" type="xsd:string"/>
  <element name="intro" type="xsd:string"/>
 </complexType>
</element>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <e:Book xmlns:e="http://www.aaa.it/test">
   <author>Henry Ford</author>
   <preface>Prefatory text</preface>
   <intro>This is a book.</intro>
  </e:Book>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Serializzazione di una struttura semplice.

Il messaggio in basso contiene l'immagine serialzzata di una struttura definita nello schema in alto.

Book è l'accessor dell'intera struttura, mentre *author*, *preface*, *intro* sono gli accessor dei singoli membri.

Formato dei messaggi SOAP

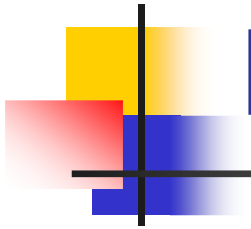
```
<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="phoneNumber"
maxOccurs="unbounded"/>
  </complexType>
  <anyAttribute/>
</element>
<element name="Person">
  <complexType>
    <element name="name" type="string"/>
    <element name="phoneNumbers" type="ArrayOfPhoneNumbers"/>
  </complexType>
</element>
```

```
<p:Person>
  <name>John Hancock</name>
  <phoneNumbers SOAP-ENC:arrayType="p:phoneNumber[2]">
    <phoneNumber>206-555-1212</phoneNumber>
    <phoneNumber>1-888-123-4567</phoneNumber>
  </phoneNumbers>
</p:Person>
```

Serializzazione di un vettore.

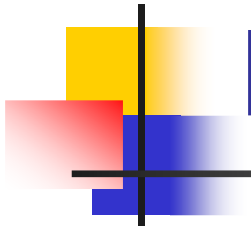
Lo schema in alto dichiara l'elemento *ArrayOfPhoneNumbers* come estensione dell'elemento *Array* del SOAP encoding. I componenti dell'array saranno contenuti negli elementi figli *phoneNumber*.

Nell'istanza di messaggio SOAP in basso, l'attributo obbligatorio *arrayType* specifica quale sarà il contenuto effettivo dell'array *phoneNumbers*: due elementi di tipo *phoneNumber*.



Formato dei messaggi SOAP

- Formato del messaggio
  - Document: struttura definita per mezzo di XML schema
  - Rpc: struttura definita da "SOAP-RPC"
- Codifica dei dati
  - Literal: codifica definita per mezzo di XML schema
  - Encoded: codifica definita da "SOAP encoding rules"
- Combinazioni più frequenti
  - Document/literal (standard in .NET)
  - Rpc/encoded (usato dai Java server)



Formato dei messaggi SOAP

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message MUST use the SOAP Encoding namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

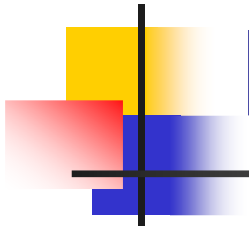
Formato dei messaggi SOAP

- Esempio: un Client formatta un messaggio SOAP per richiedere informazioni su un prodotto da un Web Service che simula un magazzino.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <getProductDetails xmlns="http://magazzino.example.com/ws">
            <productId>827635</productId>
        </getProductDetails>
    </soap:Body>
</soap:Envelope>
```

- Quello che segue è il testo con il quale il 'warehouse' web service potrebbe inviare il suo messaggio di risposta con le informazioni richieste.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
    <getProductDetailsResponse xmlns="http://magazzino.example.com/ws">
    <getProductDetailsResult>
        <productName>Toptimate, set da 3 pezzi</productName>
        <productId>827635</productId>
        <description>Set di valigie; 3 pezzi; poliestere; nero.</description>
        <price>96.50</price>
    </getProductDetailsResult>
    </getProductDetailsResponse> </soap:Body> </soap:Envelope>
```

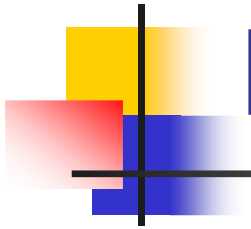


HTTP binding

- SOAP attraverso HTTP-GET
  - Codifica in HTTP ("url-encoded")
  - Risposta codificata in XML
  - Utilizzabile solo per semplici chiamate (mancano le intestazioni, i dati sono non strutturati)
- SOAP attraverso HTTP-POST
  - La parte dati di POST contiene chiamate di procedura codificate in SOAP
  - La risposta è codificata con SOAP
  - Nessuna restrizione

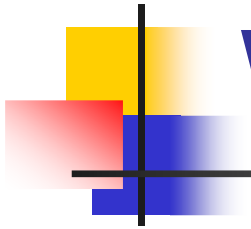
HTTP binding

- POST invia al server il path della risorsa richiesta, seguito da un blocco di dati ("*payload*"). Nel caso di SOAP, il payload sarà costituito dal messaggio SOAP vero e proprio.
- Un messaggio HTTP POST che trasporti un *payload* SOAP deve necessariamente inserire nel suo header (il gruppo di righe che apre la richiesta HTTP) il campo **SOAPAction**.
- Il contenuto del campo SOAPAction è un URI (anche vuoto) che dovrebbe **fornire delle informazioni riguardanti il contenuto del messaggio SOAP allegato.**



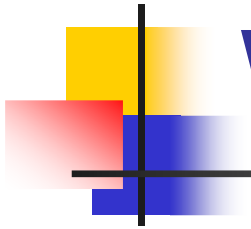
HTTP binding

- Per inviare la risposta a un messaggio SOAP, il server dovrà attenersi allo standard HTTP.
- **Il contenuto della risposta sarà il messaggio SOAP di ritorno**, eventualmente indicante un errore di elaborazione (tramite l'elemento *Fault*).
- Se il messaggio SOAP ha generato un errore, il server dovrà restituire il messaggio SOAP riguardante l'errore stesso unitamente al codice **HTTP 500/Internal Server Error**.



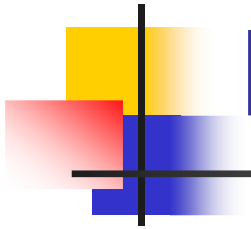
Web Services in .NET

- Infrastruttura basata su IIS e ASP.NET
- per l'implementazione di web service l'ambiente .NET fornisce:
  - Classi di base
  - Attributi
  - Protocolli
- Visual Studio .NET permette di
  - implementare web service
  - provarne il funzionamento (testing)
  - gestire IIS
  - generare codice per la creazione dei proxy (wsdl.exe)



Web Services in .NET

- System.Web.Services
  - Sviluppo del servizio (WebService, WebMethod, …)
- System.Web.Services.Configuration
  - Per estendere SOAP
- System.Web.Services.Description
  - Per creare/manipolare descrizioni in WSDL
- System.Web.Services.Discovery
  - Per usare DISCO
- System.Web.Services.Protocols
  - Per l'implementazione del protocollo di comunicazione (SOAP-HTTP, …)
- System.Xml.Serialization
  - Per la serializzazione di documenti XML



Esempio

- In Asmx

  Usando la direttiva @WebService

  <%@ WebService

  Language="C#" Class="MyWebService" %>

  Derivando da una classe Base

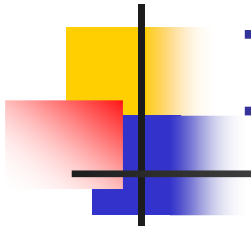
  public class MyWebService : WebService {}

  Utilizzando gli attributi di .NET

  [WebMethod(Description="Comment")]

  […]

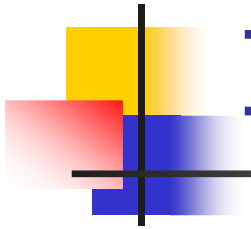
  public Returntype MyWebMethod(…) {}



Il servizio Orologio

- Orologio.asmx

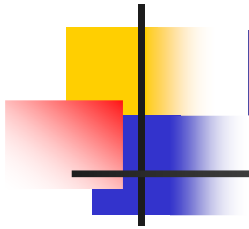
```
<%@ WebService Language="C#" Class="Orologio" %>

using System;
using System.Web.Services;

public class Orologio : WebService {

        [WebMethod(Description="restituisce l'ora corrente")]
        public string leggiOra(bool shortForm) {
                if (shortform) return DateTime.Now.ToShortTimeString();
                else return DateTime.Now.ToLongTimeString();
        }
}
```



Il Client

- Il codice proxy può essere generato automaticamente da wsdl.exe

  Wsdl.exe
  * */namespace:ClientOrologio*
  * */out:ProxyClienteOrologio.cs*
  * http://localhost/.../Orologio.asmx*
- Oppure utilizzando Visual Studio aggiungendo riferimento Web

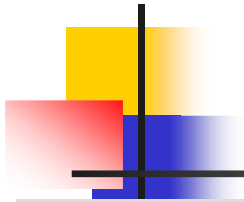


Esempio: HTTP-POST

- Codifica della chiamata al metodo String LeggiOra(bool shortForm) del webservice Orologio
- Chiamata *http://…/Orologio.asmx/LeggiOra?shortForm=true*

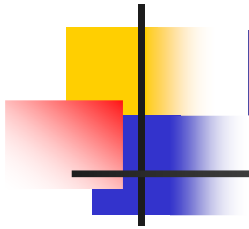
- Risposta inviata dal server

```
HTTP/1.1 200 OK Content-Type: text/xml;
charset=utf-8 Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://tempuri.org/">string</string>
```



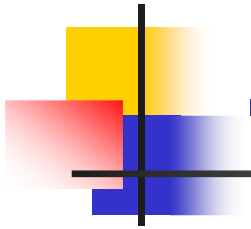
Esempio: Soap over HTTP

```
POST /Orologio/Orologio.asmx HTTP/1.1
Content-type: text/xml; charset=utf-8
SOAPAction: http://... /LeggiOra
Content-length: 198
User-Agent: ?????
Host: localhost
Accept: text/html, image/gif, image/jpeg, *;
Connection: keep-alive

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
                <LeggiOra xmlns="http://tempuri.org/"
                        <shortForm>    true    </shortForm>
                < /LeggiOra>
        </soap:Body>
</soap:Envelope>
```



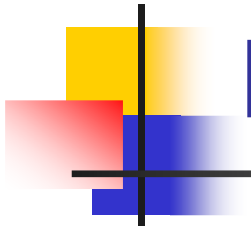
Esempio: la risposta

```
HTTP/1.1 200 OK Content-Type: text/xml; charset=utf-8 Content-
Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
        xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
                <LeggiOraResponse xmlns="http://tempuri.org/">
                        <LeggiOraResult>string</LeggiOraResult>
                </LeggiOraResponse>
        </soap:Body>
</soap:Envelope>
```



.NET e SOAP

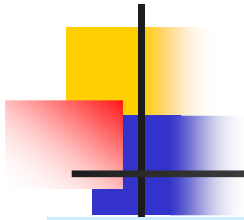
- Permette di definire il formato e la codifica dei messaggi
- Permette di codificare tutti i tipi di dato supportati in modo automatico
- Permette di sviluppare in modo semplici gli header dei messaggi
- Gestione del ciclo di vita del software



Formato e Codifica dei Messaggi

```
[SoapRpcMethod(
Use=SoapBindingUse.Encoded
Action=http://localhost/Sample/AddAddressRpc          // SOAP action
RequestNamespace="http://.../Sample/Request",
RequestElementName="AddAddressRpcRequest",// SOAP element name
ResponseNamespace="http://.../Sample/Response",
ResponseElementName="AddAddressRpcResponse")] // SOAP element name
[WebMethod(Description="Indirizzo di un utente")]
public void AddAddressRpc(long userID, Address address) { ... }
```

- attributi SoapRpcService,SoapRpcMethod
  - Use: SoapBindingUse.(Literal | Encoded)
  - Action: URI per il campo SOAPAction
  - RequestNameSpace, RequestElementName
  - ResponseNameSpace, ResponseElementName

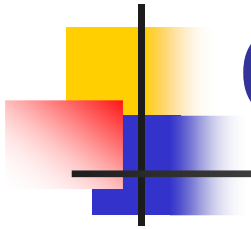


Formato e Codifica dei Messaggi (2)

```
[SoapDocumentMethod(Use=SoapBindingUse.Literal,
Action="http://localhost/Sample/AddAddressDocLit",// SOAPAction
RequestNamespace="http://localhost/Sample/Request",
RequestElementName="AddAddressDocLitRequest",// SOAP element name
ResponseNamespace="http://localhost/Sample/Response",
ResponseElementName="AddAddressDocLitResponse")]// SOAP element name
[WebMethod(Description="…")]
public void AddAddressDocLit(long userID, Address address) { ... }
```

```
[SoapDocumentService(Use=SoapBindingUse.Encoded)]
public class Orologio : WebService { ... }
```

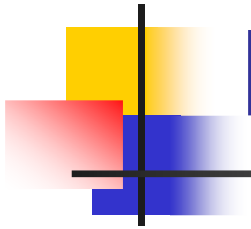
- attributi
  - SoapDocumentService
  - SoapDocumentMethod



Codifica dei tipi di dato

- Serializzazione dei tipi di dato .NET
  - Basato sulle regole di codifica SOAP http://schemas.xmlsoap.org/soap/encoding
  - Modificato mediante gli attributi (System.Web.Serialization)

SoapAttributeAttribute	Serializing field as XML attribute
SoapElementAttribute	Serializing field as XML element
SoapIgnoreAttribute	No serialization of field
SoapIncludeAttribute	Including a type
SoapEnumAttribute	Adapting name of enumeration



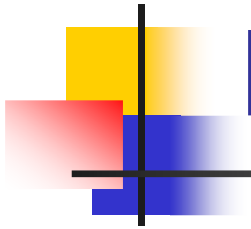
Esempi: codifica

Riscrittura del metodo LeggiOra, che restituisce una struct TimeDesc:

```
[WebMethod(Description="…")]
        public TimeDesc LeggiOra() {
                TimeDesc td = new TimeDesc();
                // ...
                return td;
        }
```

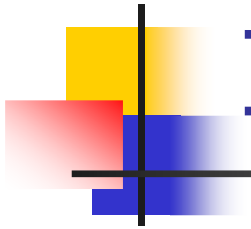
- Codifica della struttura in .NET

```
public struct TimeDesc {
[SoapAttribute] public string TimeLong;
[SoapAttribute] public string TimeShort;
[SoapAttribute (AttributeName = "ZoneID")] public int   TimeZone;
   }
```



Esempio: codifica (2)

```
…
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" ...
<soap:Body
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <types:LeggiOraResponse>
                    <LeggiOraResult href="#id1" />
      </types:LeggiOraResponse>
      <types:TimeDesc id="id1" xsi:type="types:TimeDesc"
                    types:TimeLong="10:00:25"
                    types:TimeShort="10:00“
                    types:ZoneID="1" />
      </soap:Body>
</soap:Envelope>
```

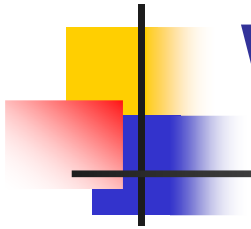


Intestazione

- L'intestazione è utilizzata per aggiungere meta-dati ai messaggi
- Non ci sono limiti sui dati che vengono aggiunti
- Tutte le intestazioni devono avere gli attributi
  - Actor: destinatario del campo
  - mustUnderstand: il campo deve necessariamente essere gestito
- .NET
  - Class SoapHeader: per la gestione dei campi
  - Attributo SoapHeaderAttribute: definizione dei campi per i metodi

Public string Actor{set;get;}
Public bool MustUnderstand{set;get;}
Public bool DidUnderstand{get;set;}

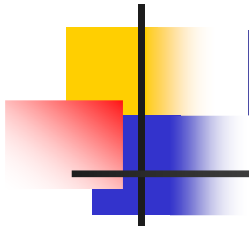
Public SoapHeaderDirection Direction {get;set;}
Public string MemberName{set;get;}



Web Services con PHP

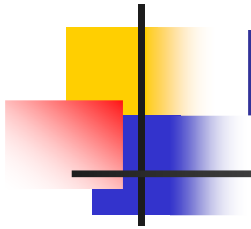
- NuSOAP: Libreria per lo sviluppo di server e client in PHP
- La libreria deve essere inclusa come qualsiasi altra librearia php
  - Require_once("nusoap.php");

- Definisco la funzione server
  - $server = new soap_server;
  - $server->register("Esempio");



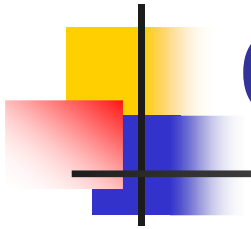
Definiamo i servizi

- Per definire i servizi offerti è sufficiente definire delle funzioni
  - *Function calcoloIVA($tasso,$valore){ … }*
- Per offrire il servizio bisogna registrarlo
  - *$server->register('calcoloIVA);*
- Ed infine occorre attivare il servizio
  $server->service($_SERVER['HTTP_RAW_POST_DATA_']);



Definiamo un cliente

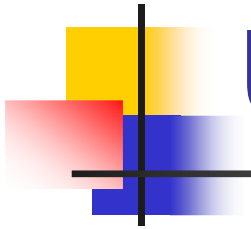
- Come nel caso del server è necessario includere la libreria
- Successivamente dichiariamo l'oggetto client
  - $client = new soapclient('http:// .../server.php');
- E richiamare il servizio
  - $response = $client->call('calcoloIVA', $parametri);



Come definire i parametri

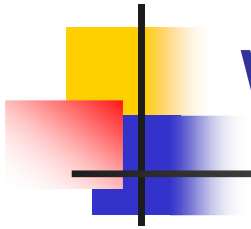
- I parametri vengono definiti tramite un array associativo
  - *$param = array('tasso' => 20, 'valore' => 111,23);*
- Nel caso di fault posso verificare lo stato e mostrare il relativo messaggio

```
If($client->fault){
        echo $client->faultcode;
        echo $client->faultstring;
}
```



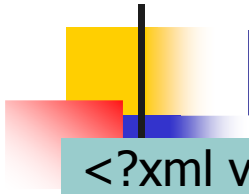
Uso dei proxy

- È possibile anche usare un oggetto proxy
  - $wsdl = "…..?wsdl";
  - $client = new soapclient($wsdl,true);
  - $proxy = $client->getProxy();
  - $response = $proxy->calcoloIVA(10, 123,45);



Web Service Description Language

- WSDL è un linguaggio per la descrizione dell'interfaccia (IDL) basato su XML
- Ogni WSD descrive
  - I tipi di dato usati
  - La struttura del messaggio
  - Le operazioni (metodi)
  - I protocolli usati per chiamare le operazioni
  - L'indirizzo del servizio web

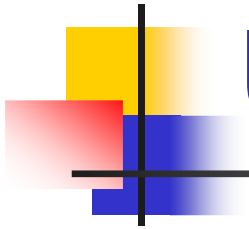


Esempio

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
        xmlns:tns="http://localhost/time/"
        xmlns:s="http://www.w3.org/2001/XMLSchema"
        ...
        targetNamespace="http://localhost/time/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types />
    <message name="GetTimeSoapIn" />
    <message name="GetTimeSoapOut">
        <part name="GetTimeResult" type="s:string" />
    </message>
    <portType name="TimeServiceSoap">
        <operation name="GetTime">
            <input message="tns:GetTimeSoapIn" />
            <output message="tns:GetTimeSoapOut" />
        </operation>
    </portType>
    ...
```

Esempio

```
    ...
<binding name="TimeServiceSoap" type="tns:TimeServiceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"style="rpc" />
<operation name="GetTime">
<soap:operation soapAction="http://dotnet.jku.at/time/GetTime" style="rpc"/>
        <input>
            <soap:body use="encoded" namespace="http://localhost/time/"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
        <output>
            <soap:body use="encoded" namespace="http://localhost/time/"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </output>
    </operation>
    </binding><service name="TimeService">
    <documentation>Richiesta dell'ora</documentation>
    <port name="TimeServiceSoap" binding="tns:TimeServiceSoap">
        <soap:address location="http://localhost/time/TimeService.asmx" />
    </port>    </service></definitions>
```

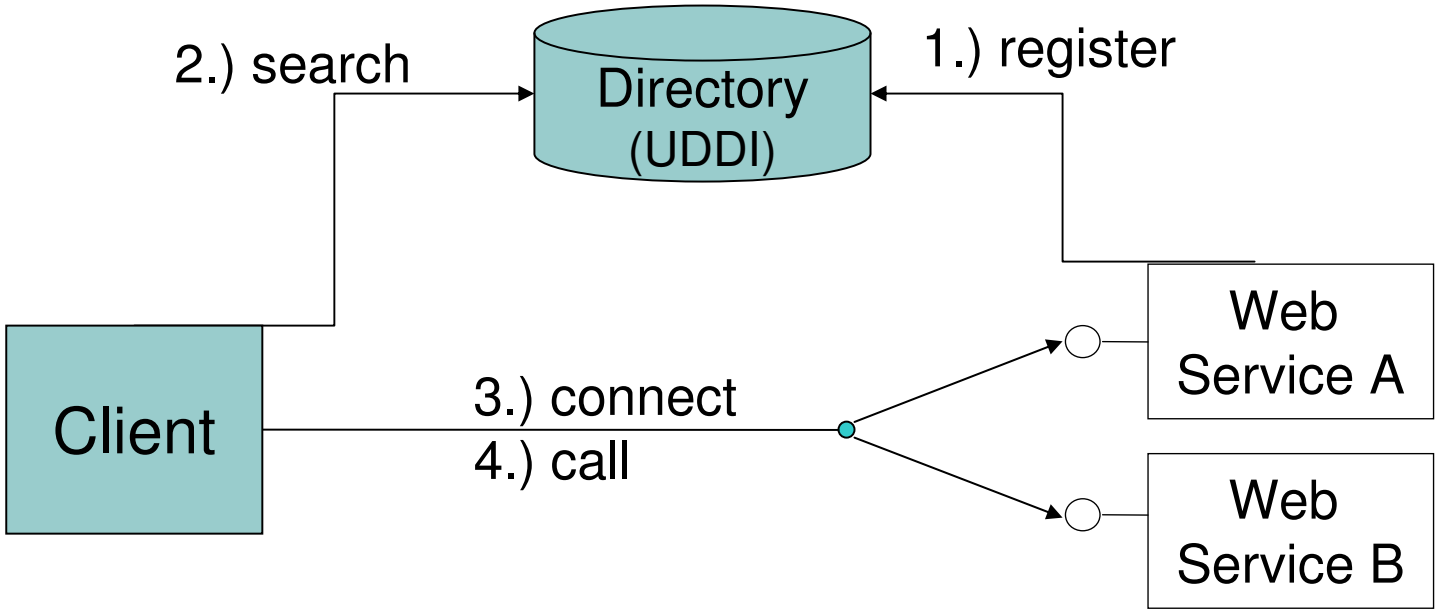


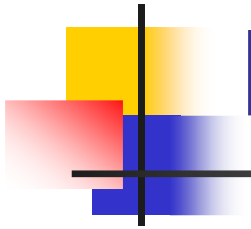
UDDI

- Universal Description Discovery and Integration) è un registry (DB ordinato ed indicizzato), basato su XML ed indipendente dalla piattaforma hardware, che permette alle aziende la pubblicazione dei propri dati e dei servizi offerti su internet.
- L'UDDI è uno degli standard alla base del funzionamento dei Web Service: è stato progettato per essere interrogato da messaggi in SOAP e per fornire il collegamento ai documenti WSDL che descrivono i vincoli protocollari ed i formati dei messaggi necessari per l'interazione con i Web Service elencati nella propria directory.

Universal, Description, Discovery, Integration

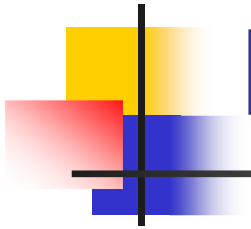
- http://www.uddi.org
  - Definizione di protocolli standard per la ricerca e l'uso dei servizi web
  - Fornisce l'interfaccia al servizio web





DISCO

- Tecnica proprietaria di Microsoft per l'uso di servizi web dinamici
- File DISCO
  - Contiene documenti XML che definisco le URI per accedere ai servizi web
  - Può essere ottenuto mediante una richiesta UDDI
- Namespace in .NET
  - System.Web.Service.Discovery



Esercitazione

- Realizzare un web service che dato un tasso di interesse ed un capitale da finanziare, calcoli e stampi la rata mensile del finanziamento
- Realizzare un web service che dato il nome di un file di testo, controlli (sul server) se e' presente ed eventualmente ne stampa il contenuto