



**Corso di Laurea in Informatica**  
**III Anno Triennale**  
**Programmazione Distribuita – classe 1**



# Service-Oriented Architecture & SOAP Web Services (1)

Delfina Malandrino

**[dmandrino@unisa.it](mailto:dmandrino@unisa.it)**

<http://www.unisa.it/docenti/delfinamalandrino>

## Organizzazione della lezione

- Service-Oriented Architecture
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

## Organizzazione della lezione

- Service-Oriented Architecture
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

## Service-Oriented Architecture

- La tecnologia (middleware) a servizi è una tecnologia per *l'integrazione* di applicazioni distribuite
  - volta a risolvere problemi pragmatici di *interoperabilità*
  - basata su standard accettati dalla maggior parte dei produttori di software
  - che focalizza l'attenzione sul concetto di *servizio*
- I Web Services rappresentano la tecnologia “*più rappresentativa*” in questo ambito

## Service-Oriented Architecture

- L'architettura orientata ai servizi (SOA)
  - fornisce il contesto metodologico (e di business) in cui utilizzare al meglio le tecnologie basate su servizi

## Service-Oriented Architecture

- I servizi sono **self-contained processes**
  - deployati su una piattaforma di middleware standard (ad esempio Java EE)
  - che possono essere descritti, pubblicati, localizzati ed invocati attraverso la rete
- Ogni pezzo di codice oppure ogni **componente di un'applicazione** di cui è stato fatto il deploy su un sistema **può essere trasformata in un servizio** accessibile via rete
- I servizi riflettono un nuovo approccio "**service-oriented**", basato sull'idea di comporre applicazioni "scoprendo" ed invocando servizi sulla rete piuttosto che scrivere nuove applicazioni
  - Combinazione di funzionalità (già esistenti)

## Service-Oriented Achitecture

- I servizi eseguono funzioni che spaziano fra rispondere a semplici richieste fino all'eseguire complesse logiche di business
- I servizi sono scritti in modo di essere **indipendenti dal contesto** in cui sono usati
  - Service providers e consumers sono debolmente accoppiati
- A livello middleware il loosely coupling richiede che l'approccio service-oriented sia **indipendente dalle specifiche tecnologie o sistemi operativi**
- Servizi e composizione di servizi sono **indipendenti dai linguaggi di programmazione utilizzati**
  - Invocazione attraverso l'utilizzo di self-describing interfacce e di standards

## Service-Oriented Achitecture

- Attraverso l'utilizzo di standard (linguaggi e protocolli)...
- ...gli sviluppatori possono accedere a sistemi ed applicazioni sulla rete sulla base di quello che fanno
  - indipendentemente da come realizzano le loro funzionalità e come sono state implementate

## Service-Oriented Achitecture

- Affinchè si possa parlare di Service-Oriented Architecture, i servizi devono:
  - **Essere neutrali rispetto alla tecnologia**
    - Devono poter essere invocati attraverso tecnologie standard (protocolli, meccanismi di discovery) compliant con accepted standards
  - **Loosely Coupled**
    - Nessuna conoscenza della struttura interna o convenzioni (context) sia a livello client che server
  - **Supportare location transparency**
    - Le definizioni e informazioni di localizzazione devono essere memorizzate in un repository (ad esempio, UDDI) ed accessibili a diversi client
      - Che possono invocarli indipendentemente dalla locazione

## Service-Oriented Achitecture

- I servizi possono essere forniti su una singola macchina o su un gran numero di dispositivi distribuiti su:
  - Una rete locale
  - Una rete distribuita (incluso mobile e ad hoc networks)
- Un caso interessante è quello dei servizi che usano Internet come communication medium e open Internet standards
  - **I Web Services**

## Organizzazione della lezione

- Service-Oriented Architecture
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

## Cosa sono i Web Services

- Applicazioni che possono essere implementate usando diverse tecnologie
  - Simple Object Access Protocol (SOAP) e Representational State Transfer (REST)
- Concetto chiave: “loosely coupled” (debolmente accoppiati)
  - niente è conosciuto dal client (consumer) del servizio circa la sua implementazione
    - linguaggio usato per svilupparlo, la piattaforma che lo esegue, ecc.

## Il paradigma dei Web Services

- Costrutti per supportare lo sviluppo di applicazioni distribuite che siano:
  - Veloci
  - Basso costo
  - Di facile composizione con altri servizi simili
- WS: logica di applicazione disponibile in maniera automatica ed esposta su Internet
  - Ogni porzione di codice e componente può essere trasformata in un web service
- Approccio orientato ai servizi: comporre un'applicazione scoprendo e invocando servizi disponibili sulla rete
  - Invece di costruire nuove applicazioni o invocare applicazioni disponibili

## Requisiti di un Web Service

- Neutri rispetto alla tecnologia
  - ricondotti al minimo comun denominatore tecnologico per essere disponibile in tutti gli ambienti
- Debolmente accoppiati
  - nessuna conoscenza circa le strutture interne, oppure il contesto, da parte del client
- Trasparenza alla locazione
  - definizione e locazione disponibile attraverso registri pubblici, che siano accessibili da un client generico

## Definizione di Web Service

### Dalla W3C:

Un WS è un sistema software progettato per supportare interazioni interoperabili tra computer su una rete.

L'interfaccia è specificata in una descrizione analizzabile automaticamente (come Web Service Definition Language).

L'interazione con gli altri sistemi avviene mediante messaggi SOAP (Simple Object Access Protocol), tipicamente attraverso una richiesta HTTP con serializzazione XML.

## Definizione di Web Service

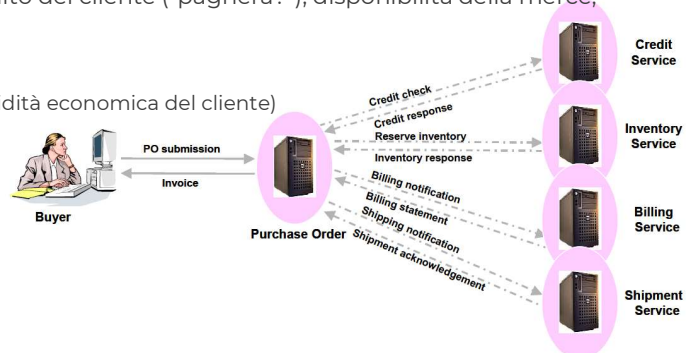
### ■ Cosa possono essere i WS:

- Un task auto-contenuto (depositare fondi su un conto bancario)
- Un processo di business (acquisto automatico di forniture per ufficio)
- Una applicazione (calcolo di previsioni su valore di azioni)
- Una risorsa accessibile mediante servizi (backend di un DB)



## Definizione di Web Service

- Un esempio di gestione di ordini di acquisto:
  - Servizio di ricevimento di ordini di acquisto
  - Risposta sulla base di: buon credito del cliente ("pagherà?"), disponibilità della merce, etc.
  - Servizi coinvolti:
    - Servizio di credito (controlla la validità economica del cliente)
    - Servizio di inventario
    - Servizio di billing
    - Servizio di spedizione



## Una confusione da evitare

- Confronto tra Web services e servizi accessibili via Web
- I servizi accessibili via Web usano una interfaccia universale (browser) ma non sono necessariamente WS
- I WS sono utilizzabili sia automaticamente da altre applicazioni sia da utenti
- Necessario rivedere il concetto di "Software as a Service" (SaaS)

## Software as a Service (SaaS)

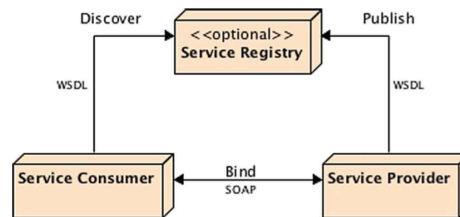
- Concetto apparso per la prima volta con il modello software **Application Service Provider (ASP)**
- Application Service Providers
  - Companies che combinano software e elementi infrastrutturali con servizi di business e professionali per fornire una soluzione completa ai customer sotto forma di servizio richiesto sulla base di una sottoscrizione
  - Sono entità terze che eseguono il deploy, mantengono e gestiscono l'accesso a packaged applications e forniscono software-based services attraverso la rete a customer multipli

## Organizzazione della lezione

- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

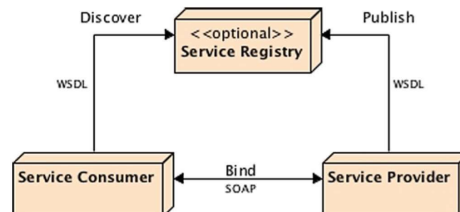
## Ruoli

- Service Provider
  - Organizzazione che fornisce i servizi
- Service Registry
  - Fornisce un repository di service descriptions (WSDL) pubblicati dal Service Provider
- Service Consumer
  - Fruitore dei servizi
    - Individuo
    - Applicazione



## Le interazioni con i Web Services

- Il Web Service può essere (opzionalmente) registrato in un registro con Universal Description Discovery and Integration (UDDI)
- Quando il consumer conosce l'interfaccia del servizio e il formato del messaggio ...
- ... può inviare una richiesta al provider del servizio e ricevere una risposta

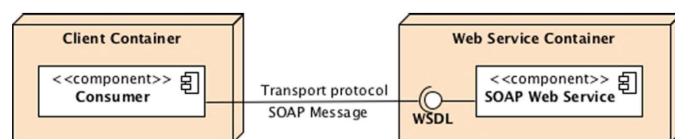


## Organizzazione della lezione

- Service-Oriented architectures
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

## Il ruolo di WSDL

- Serve a specificare l'interfaccia di un Web Service: il contratto garantito
- Ruolo di Interface Definition Language tra i consumer e il servizio (fornito dal provider)
- Fornisce informazioni su:
  - tipo del messaggio
  - porta
  - protocollo di comunicazione
  - operazioni supportate
  - posizione
  - cosa si aspetta il consumatore come valore di ritorno



## Elementi e attributi di WSDL

Element	Description
definitions	Is the root element of the WSDL, and it specifies the global declarations of namespaces that are visible throughout the document
types	Defines the data types to be used in the messages. In this example, it is the XML Schema Definition (CardValidatorService?xsd=1) that describes the parameters passed to the web service request and the response
message	Defines the format of data being transmitted between a web service consumer and the web service itself. Here you have the request (the validate method) and the response (validateResponse)
portType	Specifies the operations of the web service (the validate method). Each operation refers to an input and output message
binding	Describes the concrete protocol (here SOAP) and data formats for the operations and messages defined for a particular port type
portType	Specifies the operations of the web service (the validate method). Each operation refers to an input and output message
binding	Describes the concrete protocol (here SOAP) and data formats for the operations and messages defined for a particular port type
service	Contains a collection of <port> elements, where each port is associated with an endpoint (a network address location or URL)
port	Specifies an address for a binding, thus defining a single communication endpoint

## Un file WSDL: credit card validation

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://chapter14.javaee7.book.agoncal.org/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://chapter14.javaee7.book.agoncal.org/"
  name="CardValidatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace=
        "http://chapter14.javaee7.book.agoncal.org/"
        schemaLocation=
        "http://localhost:8080/chapter14/CardValidatorService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="validate">
    <part name="parameters" element="tns:validate"/>
  </message>
  <message name="validateResponse">
    <part name="parameters" element="tns:validateResponse"/>
  </message>
  <portType name="CardValidator">
    <operation name="validate">
      <input message="tns:validate"/>
      <output message="tns:validateResponse"/>
    </operation>
  </portType>
  ...

```

XML Schema dati

<definitions>

Is the root element of the WSDL, and it specifies the global declarations of namespaces that are visible throughout the document

## Un file WSDL: credit card validation

```
<?xmlversion="1.0"encoding="UTF-8"?>
<definitions
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://chapter14.javaee7.book.agoncal.org/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://chapter14.javaee7.book.agoncal.org/"
  name="CardValidatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace=
        "http://chapter14.javaee7.book.agoncal.org/"
        schemaLocation=
          "http://localhost:8080/chapter14/CardValidatorService?xsd=1">
      </xsd:schema>
    </types>
    <message name="validate">
      <part name="parameters" element="tns:validate"/>
    </message>
    <message name="validateResponse">
      <part name="parameters" element="tns:validateResponse"/>
    </message>
    <portType name="CardValidator">
      <operation name="validate">
        <input message="tns:validate"/>
        <output message="tns:validateResponse"/>
      </operation>
    </portType>
    ...
  </definitions>
```

Definizione del tipo di dato

<types>

Defines the data types to be used in the messages. In this example, it is the XML Schema Definition (CardValidatorService?xsd=1) that describes the parameters passed to the web service request and the response

## Un file WSDL: credit card validation

```
<?xmlversion="1.0"encoding="UTF-8"?>
<definitions
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://chapter14.javaee7.book.agoncal.org/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://chapter14.javaee7.book.agoncal.org/"
  name="CardValidatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace=
        "http://chapter14.javaee7.book.agoncal.org/"
        schemaLocation=
          "http://localhost:8080/chapter14/CardValidatorService?xsd=1">
      </xsd:schema>
    </types>
    <message name="validate">
      <part name="parameters" element="tns:validate"/>
    </message>
    <message name="validateResponse">
      <part name="parameters" element="tns:validateResponse"/>
    </message>
    <portType name="CardValidator">
      <operation name="validate">
        <input message="tns:validate"/>
        <output message="tns:validateResponse"/>
      </operation>
    </portType>
    ...
  </definitions>
```

<message>

Defines the format of data being transmitted between a web service consumer and the web service itself. Here you have the request (the validate method) and the response (validateResponse)

Il formato del messaggio dal consumer al provider ... ed il formato della risposta

## Un file WSDL: credit card validation

```
<?xmlversion="1.0"encoding="UTF-8"?>
<definitions
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://chapter14.javaee7.book.agoncal.org/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://chapter14.javaee7.book.agoncal.org/"
  name="CardValidatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace=
        "http://chapter14.javaee7.book.agoncal.org/"
        schemaLocation=
          "http://localhost:8080/chapter14/CardValidatorService?xsd=1">
    </xsd:schema>
  </types>
  <message name="validate">
    <part name="parameters" element="tns:validate"/>
  </message>
  <message name="validateResponse">
    <part name="parameters" element="tns:validateResponse"/>
  </message>
  <portType name="CardValidator">
    <operation name="validate">
      <input message="tns:validate"/>
      <output message="tns:validateResponse"/>
    </operation>
  </portType>
  ...
```

<portType>

Specifies the operations of the web service (the validate method). Each operation refers to an input and output message

Specifica dell'operazione CardValidator

## Un file WSDL

```
<binding name="CardValidatorPortBinding"
  type="tns:CardValidator">
  <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
  <operation name="validate">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<service name="CardValidatorService">
  <port name="CardValidatorPort"
    binding="tns:CardValidatorPortBinding">
    <soap:address location=
      "http://localhost:8080/chapter14/CardValidatorService"/>
  </port>
</service>
</definitions>
```

Protocollo utilizzato

Codifica SOAP dell'operazione  
... con input e output

<binding>

Describes the concrete protocol (here SOAP) and data formats for the operations and messages defined for a particular port type

## Un file WSDL

```
...
<binding name="CardValidatorPortBinding"
        type="tns:CardValidator">
  <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
  <operation name="validate">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<service name="CardValidatorService">
  <port name="CardValidatorPort"
        binding="tns:CardValidatorPortBinding">
    <soap:address location=
      "http://localhost:8080/chapter14/CardValidatorService"/>
  </port>
</service>
</definitions>
```

**<service>**

Contains a collection of <port> elements, where each port is associated with an endpoint (a network address location or URL)

Posizione del servizio

## Organizzazione della lezione

- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni



## Simple Object Access Protocol (SOAP)

*"SOAP is a lightweight protocol for exchanging structured information in a decentralized, distributed environment."*

*It is an XML based protocol that consists of three parts: an **envelope** that defines a framework for describing what is in a message and how to process it, a set of **encoding rules** for expressing instances of application-defined datatypes, and a **convention** for representing remote procedure calls and responses."*

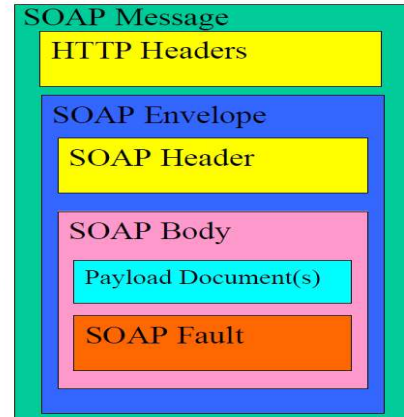
*Draft W3C specification*

## Protocollo per lo scambio di messaggi

- Simple Object Access Protocol (SOAP)
- Fortemente basato su XML: i messaggi scambiati sono strutturati in diversi elementi (envelope, header, body) definiti in XML
- Usa HTTP come protocollo di scambio: richiesta HTTP corrisponde all'invio della richiesta del servizio, mentre la risposta HTTP corrisponde al risultato della invocazione
- Caratteristiche fondamentali: indipendente dalla piattaforma

## SOAP Message structure

- SOAP envelope
  - Identifica il documento XML come messaggio SOAP
- SOAP header (opzionale)
  - contiene informazioni aggiuntive per il message processing
- SOAP Body
  - Contenuto vero e proprio del messaggio
  - Contiene chiamate e risposte
- SOAP fault
  - Contiene informazioni su eventuali errori verificatisi durante il processing



## SOAP Message structure

- La tabella che segue mostra un sottoinsieme dei SOAP elements and attribute
- Come WSDL, anche SOAP definito dalla W3C

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

## SOAP Message structure

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://example.org/2001/06/quotes"
```

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
  <env:Body>
    <m:GetLastTradePrice
      env:encodingStyle="http://www.w3.org/2001/06/soap-encoding"
      xmlns:m="http://example.org/2001/06/quotes">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

Walt Disney Company (DIS) ⓘ

NYSE Valuta in USD

**116,73** +0,23 (+0,20%) ▲

Chiuso · 06/12

After Hours ▼ 116,70 -0,03 (-0,03%) 07/12

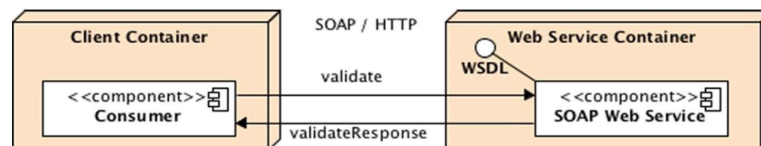
## SOAP Message structure

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

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
  <env:Body>
    <m:GetLastTradePriceResponse
      env:encodingStyle="http://www.w3.org/2001/06/soap-encoding"
      xmlns:m="http://example.org/2001/06/quotes">
      <Price>116,73</Price>
    </m:GetLastTradePriceResponse>
  </env:Body>
</env:Envelope>
```

## SOAP: Tornando al nostro esempio

- WSDL descrive un'interfaccia del web service mentre SOAP fornisce una concreta implementazione definendo i messaggi XML scambiati fra il consumer ed il provider
- Nel nostro esempio, quali sono i messaggi che vengono scambiati?
  - Il consumer invia le informazioni relative alla carta di credito all'interno di un SOAP envelope (Listing 14-3) al metodo **validate** del web service **CardValidator**
  - Il servizio restituisce un altro SOAP envelope (Listing 14-4) con il risultato della validazione (true o false)



## Richiesta e risposta SOAP

### Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

› La “busta” del messaggio

### Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011"expiry_date="10/12"
        control_number="544"type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011"expiry_date="10/12"
        control_number="544"type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo con...
- › ...l'invocazione del servizio

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo con...
- › ... l'invocazione del servizio
- › I parametri passati

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo con...
- › ... l'invocazione del servizio
- › I parametri passati

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” della risposta

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

# Richiesta e risposta SOAP

## Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo con...
- › ... l'invocazione del servizio
- › I parametri passati

## Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/">
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” della risposta
- › Il corpo

Element	Description
Envelope	Defines the message and the namespace used in the document. This is a required root element
Header	Contains any optional attributes of the message or application-specific infrastructure such as security information or network routing
Body	Contains the message being exchanged between applications
Fault	Provides information about errors that occur while the message is processed. This element is optional

## Richiesta e risposta SOAP

### Richiesta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/"
  <soap:Header/>
  <soap:Body>
    <cc:validate>
      <arg0 number="123456789011" expiry_date="10/12"
        control_number="544" type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” del messaggio
- › L'header (vuoto)
- › Il corpo con...
- › ... l'invocazione del servizio
- › I parametri passati

### Risposta

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:cc="http://chapter14.javaee7.book.agoncal.org/"
  <soap:Body>
    <cc:validateResponse>
      <return>false</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

- › La “busta” della risposta
- › Il corpo
- › Il risultato (valore true)

## Organizzazione della lezione

- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni



## Per trovare le informazioni

- Servizio di localizzazione dei servizi
- Pubblicazione da parte del provider della descrizione del servizio in WSDL in un registro UDDI accessibile
- Può essere, in questa maniera, scoperto e scaricato
- E' opzionale, quando si conosce già la locazione del servizio
- UDDI è basato su XML, e contiene informazioni aggiuntive sul servizio (come disponibilità, costo, etc.)
- Poco utilizzato e attualmente in disuso: WS un pò meno "loose coupled"

## Organizzazione della lezione

- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

## Le (tante) specifiche dei WS

Specification	Version	Stand. body	JSR	URL
JAX-WS	2.2a	JCP	224	<a href="http://jcp.org/en/jsr/detail?id=224">http://jcp.org/en/jsr/detail?id=224</a>
Web Services	1.3	JCP	109	<a href="http://jcp.org/en/jsr/detail?id=109">http://jcp.org/en/jsr/detail?id=109</a>
Web Services Metadata	2.1	JCP	181	<a href="http://jcp.org/en/jsr/detail?id=181">http://jcp.org/en/jsr/detail?id=181</a>
JAXB	2.2	JCP	222	<a href="http://jcp.org/en/jsr/detail?id=222">http://jcp.org/en/jsr/detail?id=222</a>
SAAJ	1.3	JCP	67	<a href="http://jcp.org/en/jsr/detail?id=67">http://jcp.org/en/jsr/detail?id=67</a>
<i>JAX-RPC</i>	<i>1.1</i>	<i>JCP</i>	<i>101</i>	<a href="http://jcp.org/en/jsr/detail?id=101">http://jcp.org/en/jsr/detail?id=101</a>
<i>JAXR</i>	<i>1.1</i>	<i>JCP</i>	<i>93</i>	<a href="http://jcp.org/en/jsr/detail?id=93">http://jcp.org/en/jsr/detail?id=93</a>
SOAP	1.2	W3C		<a href="http://www.w3.org/TR/soap/">http://www.w3.org/TR/soap/</a>
XML	1.1	W3C		<a href="http://www.w3.org/TR/xml">http://www.w3.org/TR/xml</a>
WSDL	1.1	W3C		<a href="http://www.w3.org/TR/wsdl">http://www.w3.org/TR/wsdl</a>
UDDI	1.0	OASIS		<a href="http://uddi.org/pubs/uddi_v3.htm">http://uddi.org/pubs/uddi_v3.htm</a>

## Implementazione di riferimento

- Implementazione di riferimento è Metro (Open source) per quanto riguarda il mondo Java
- Implementa JAX-WS (versione 2.2 ora) che permette di costruire e consumare WS in Java
- Metro stack prodotto dalla community di GlassFish
- Anche Apache CXF (precedentemente conosciuto come XFire) e Apache Axis2 implementano lo stack JWS
- JAX-WS permette di mascherare SOAP dal programmatore

## Come sviluppare un Web Service

- Si può partire da due direzioni: dall'alto (specifiche) verso il basso (codice) oppure in direzione opposta
- Approccio top-down: contract first
  - Si parte dal contratto in WSDL, definendo operazioni, messaggi etc.
  - Generazione automatica delle classi da WSDL fornite da Metro (wsimport)
- Approccio bottom-up: program first
  - L'implementazione in Java già esiste
  - Generazione automatica per generare il WSDL a partire dalle classi fornite da Metro (wsген)
- In entrambi i casi il compito del programmatore è facilitato da JAX-WS, che permette di annotare un POJO e farlo "diventare" Web Service
  - Anche qui ...Configuration-by-exception

## Come sviluppare un Web Service

- Il nostro esempio:
  - Web service che valida una carta di credito

## II WS CardValidator

```
@WebService
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

› Annotazione che denota un WS

## II WS CardValidator

```
@WebService
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

› Annotazione che denota un WS

› Nome del POJO

## II WS CardValidator

```
@WebService
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- > Annotazione che denota un WS
- > Nome del POJO
- > Metodo offerto via WS, che restituisce un booleano

## II WS CardValidator

```
@WebService
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- > Annotazione che denota un WS
- > Nome del POJO
- > Metodo offerto via WS, che restituisce un booleano
- > Se la carta è validata (lastDigit pari) ...

## II WS CardValidator

```
@WebService
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- › Annotazione che denota un WS
- › Nome del POJO
- › Metodo offerto via WS, che restituisce un booleano
- › Se la carta è validata (lastDigit pari) ...
- › ... oppure no

## Cosa viene scambiato: la carta di credito

- Un oggetto CreditCard object viene scambiato fra il consumer e il SOAP web service
- I dati da scambiare devono essere documenti XML
- Necessario quindi un metodo per trasformare un oggetto Java in XML
  - **NECESSARIO TRANSFORMARE UNA INVOCAZIONE DI UN METODO JAVA IN UNA CHIAMATA XML SU HTTP**
- JAXB (Java Architecture for XML Binding) permette questo con semplici annotazioni
- In questa maniera il programmatore Java può evitare di scrivere XML

*Listing 14-6.* The CreditCard Class with JAXB Annotations

```
@XmlRootElement
public class CreditCard {

    @XmlAttribute(required = true)
    private String number;
    @XmlAttribute(name = "expiry_date", required = true)
    private String expiryDate;
    @XmlAttribute(name = "control_number", required = true)
    private Integer controlNumber;
    @XmlAttribute(required = true)
    private String type;

    // Constructors, getters, setters
}
```

## La classe CreditCard con le annotazioni JAXB

```

@XmlRootElement
public class CreditCard
{
    @XmlAttribute(required = true)
    private String number;

    @XmlAttribute(name = "expiry_date", required = true)
    private String expiryDate;

    @XmlAttribute(name = "control_number", required = true)
    private Integer controlNumber;

    @XmlAttribute(required = true)
    private String type;

    //Constructors, getters, setters
}

```

> Definizione del root element del file XML

## La classe CreditCard con le annotazioni JAXB

```

@XmlRootElement
public class CreditCard
{
    @XmlAttribute(required = true)
    private String number;

    @XmlAttribute(name = "expiry_date", required = true)
    private String expiryDate;

    @XmlAttribute(name = "control_number", required = true)
    private Integer controlNumber;

    @XmlAttribute(required = true)
    private String type;

    //Constructors, getters, setters
}

```

> Definizione del root element del file XML

> Il nome della classe/tipo

## La classe CreditCard con le annotazioni JAXB

```
@XmlRootElement
public class CreditCard
{
    @XmlAttribute(required = true)
    private String number;

    @XmlAttribute(name = "expiry_date", required = true)
    private String expiryDate;

    @XmlAttribute(name = "control_number", required = true)
    private Integer controlNumber;

    @XmlAttribute(required = true)
    private String type;

    //Constructors, getters, setters
}
```

- > Definizione del root element del file XML
- > Il nome della classe/tipo
- > Un attributo XML, con lo stesso nome del campo Java

## La classe CreditCard con le annotazioni JAXB

```
@XmlRootElement
public class CreditCard
{
    @XmlAttribute(required = true)
    private String number;

    @XmlAttribute(name = "expiry_date", required = true)
    private String expiryDate;

    @XmlAttribute(name = "control_number", required = true)
    private Integer controlNumber;

    @XmlAttribute(required = true)
    private String type;

    //Constructors, getters, setters
}
```

- > Definizione del root element del file XML
- > Il nome della classe/tipo
- > Un attributo XML, con lo stesso nome del campo Java
- > Un attributo obbligatorio, con un nome diverso da quello del campo



## Anatomia di un SOAP WS

- La classe deve:
  - essere annotata con `@WebService` (oppure nel file XML `webservices.xml`)
  - implementare zero o più interfacce (service endpoint) annotate con `@WebService`
  - esser pubblica (non final o astratta)
  - avere un costruttore pubblico di default
  - non avere un `finalize()`
  - essere annotata anche con `@Stateless` o `@Singleton` se si vuole renderla anche un EJB
    - un servizio è sempre stateless!

## SOAP Web Service Endpoints

- JAX-WS permette sia a regolari classi Java che EJBs di essere esposti come Web services
- Vediamo un esempio...

## II WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

← E' un WS

## II WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

← E' un WS  
Ma è anche un EJB Stateless

## Il WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- > E' un WS
- > Ma è anche un EJB Stateless
- > Nome della classe

## Il WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- > E' un WS
- > Ma è anche un EJB Stateless
- > Nome della classe
- > Metodo offerto sia come WS che come EJB

## II WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- › E' un WS
- › Ma è anche un EJB Stateless
- › Nome della classe
- › Metodo offerto sia come WS che come EJB
- › Restituisce vero ...

## II WS CardValidator come EJB

```
@WebService
@Stateless
public class CardValidator
{
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }else{
            return false;
        }
    }
}
```

- › E' un WS
- › Ma è anche un EJB Stateless
- › Nome della classe
- › Metodo offerto sia come WS che come EJB
- › Restituisce vero ...
- › oppure falso

## A cosa serve che un WS sia anche un EJB?

- Si guadagnano i servizi che sono offerti per EJB e non sono offerti per i WS
  - Transazioni, sicurezza in maniera automatica
- Gestione di interceptor, etc., sono possibili in questo caso

## A cosa serve che un WS sia anche un EJB?

- Flessibilità ed efficienza
  - lo stesso "servizio" accessibile per consumer fortemente accoppiati (EJB via RMI-IIOP)
    - EJB chiamato direttamente tramite il protocollo RMI o altre API interne di Java EE
    - Questo approccio è **più efficiente** perché sfrutta la comunicazione nativa del server applicativo e delle risorse Java EE, riducendo l'overhead (ad esempio, la necessità di serializzare e deserializzare i dati come avviene nei Web Services).
  - oppure per consumer debolmente accoppiati (WS via SOAP)
    - Questo approccio lo rende "indipendente dalla piattaforma", permettendo a client su altre piattaforme (ad esempio, un'applicazione scritta in Python o un dispositivo IoT) di consumare il servizio) di consumare il servizio
    - Introdotto un **overhead aggiuntivo** legato al formato del messaggio (XML o JSON) e alla necessità di comunicare tramite protocolli come HTTP che rende questo tipo di interazione meno efficiente rispetto alla chiamata diretta in Java EE

## A cosa serve che un WS sia anche un EJB?

Flessibilità ed efficienza

- Un EJB esposto anche come WS permette di fornire lo stesso servizio con due modalità:
  - **Dipendente dalla piattaforma** (richiesta diretta RMI o simili): per client Java che operano all'interno dello stesso ecosistema, con alte prestazioni
  - **Indipendente dalla piattaforma** (via Web Service): per client esterni o su piattaforme diverse, con maggiore interoperabilità ma minore efficienza

## Organizzazione della lezione

- Service-Oriented Architecture
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni