1

Organizzazione della lezione

2

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

2

Organizzazione della lezione

3

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

3

Service-Oriented Achitecture

4

- 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

4

Service-Oriented Achitecture

5

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

5

Service-Oriented Achitecture

6

- 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)

6

Service-Oriented Achitecture

7

- 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

7

Service-Oriented Achitecture

8

- 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

8

Service-Oriented Achitecture

9

- 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

9

Service-Oriented Achitecture

10

- 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

10

Organizzazione della lezione

11

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

11

Cosa sono i Web Services

12

- Banalmente, qualcosa accessibile sul Web che offre un servizio (service)
  - pagine HTML, servlet, . . .
- 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.

12

Il paradigma dei Web Services

13

- 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

13

Requisiti di un Web Service

14

- 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

14

Definizione di Web Service

15

- 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.

15

Definizione di Web Service

16

- 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)

16

Organizzazione della lezione

20

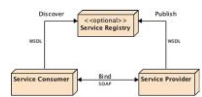
- I Web Services
  - Definizioni
  - I ruoli nei WS
- Gli standard
  - WSDL
  - SOAP
  - UDDI
  - Le specifiche di WS
- Conclusioni

20

Ruoli

21

- 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

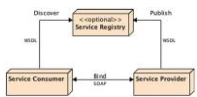


21

Le interazioni con i Web Services

22

- 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



22

Organizzazione della lezione

23

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

23

Organizzazione della lezione

24

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

24

Il ruolo di WSDL

25

- 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

25

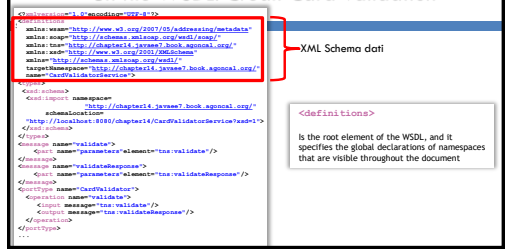
Elementi e attributi di WSDL

26

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
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

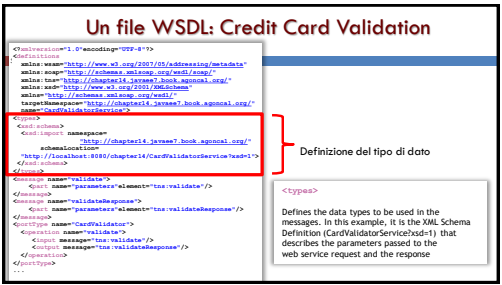
26

Un file WSDL: Credit Card Validation



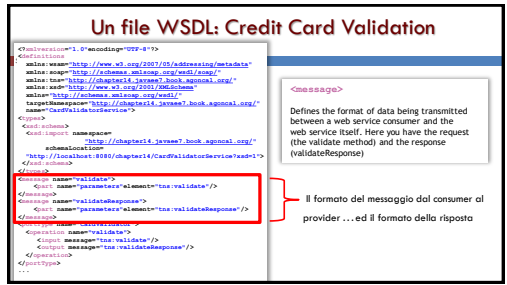
27

Un file WSDL: Credit Card Validation



28

Un file WSDL: Credit Card Validation

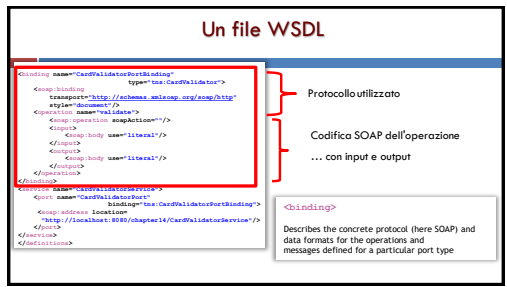


29

Un file WSDL: Credit Card Validation

30

Un file WSDL



31

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

32

Organizzazione della lezione

33

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

33

SOAP

34

"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

34

Protocollo per lo scambio di messaggi

35

- Simple Object Access Protocoll (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, mente la risposta HTTP corrisponde al risultato della invocazione
- Caratteristiche fondamentali: indipendente dalla piattaforma

35

SOAP
Message structure

36

- 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

36

Attributi ed eventi SOAP

48

- 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

37

SOAP

Message structure

38

```
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>
```

38

SOAP

Message structure

39

```
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>150.37</Price>
        </m:GetLastTradePriceResponse>
    </env:Body>
</env:Envelope>
```

39

SOAP: Tornando al nostro esempio

39

- 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 `card validator`
  - Il servizio restituisce un altro SOAP envelope (Listing 14-4) con il risultato della validazione (true o false)

40

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="1234567890123" 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>true</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

41


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="1234567890123"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>true</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

42


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="1234567890123"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>true</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

43

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="1234567890123"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>true</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

44

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="1234567890123"expiry_date="10/12"
            control_number="544"type="Visa"/>
    </cc:validate>
  </soap:Body>
</soap:Envelope>
```

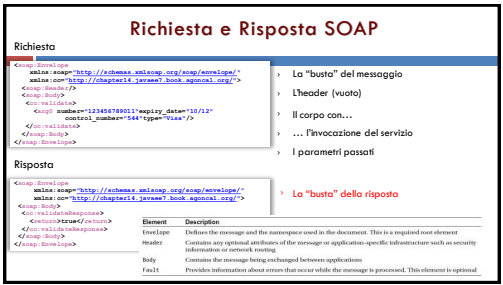
- La "busta" del messaggio
- L'header (vuoto)
- Il corpo con…
- … l'invocazione del servizio
- I parametri passati

Risposta

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

45


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="1234567890123"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>true</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

46


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="1234567890123"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>true</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

47

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>true</return>
    </cc:validateResponse>
  </soap:Body>
</soap:Envelope>
```

- La "busta" della risposta
- Il corpo
- Il risultato (valore true)

48

Organizzazione della lezione

49

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

49

Per trovare le informazioni

50

- 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 addizionali sul servizio (come disponibilità, costo, etc.)
- Poco utilizzato e attualmente in disuso: WS un pò meno "loose coupled"

50

Organizzazione della lezione

51

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

51

Una breve storia dei Web Services

52

- Necessario per comprendere tutti gli standard
- Protocolli distribuiti: lotta tra standard, tra ecosistemi diversi con diversi precursori:
  - DCOM (Microsoft), RPC/RMI (Sun/Oracle), CORBA
- Nessun vincitore né vinti: uso di HTTP, protocollo affermatosi come standard sul WWW
- SOAP 1.0 fu lanciato da Microsoft e appoggiato da IBM
- Standard successivi: SOAP, WSDL, UDDI
- Supporto da Java a partire dal 2002
- Standard diversi gestiti da organismi diversi

52

Le (tante) specifiche dei WS

53

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

53

Implementazione di riferimento

54

- 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

54

Come sviluppare un Web Service

55

- 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 (wsgen)
- 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

55

Come sviluppare un Web Service

56

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

56

Il 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

57

Il 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

58

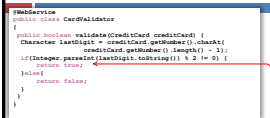
Il WS CardValidator

```
@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

59

Il WS CardValidator

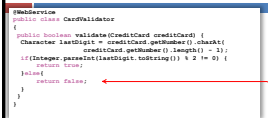


```
@WebService
public class CardValidator
{
 public boolean validate(CreditCard creditCard) {
  Character lastDigit = creditCard.getNumber().charAt(
                creditCard.getNumber().length() - 1);
  if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
     return true;
  }else{
     return false;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

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

60

Il 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 dispari) …
- …oppure no

61

Cosa viene scambiato: la carta di credito

62

- 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
}
```

62

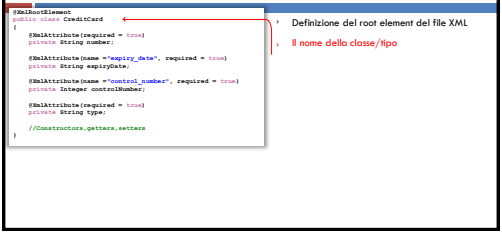
La classe CreditCard con le annotazioni JAXB

```
@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

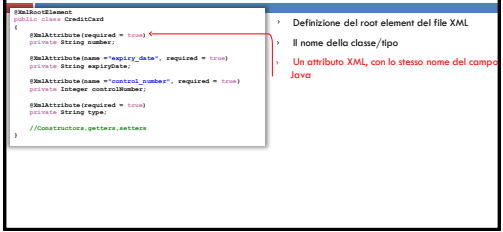
63

La classe CreditCard con le annotazioni JAXB



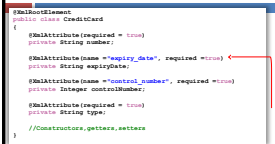
64

La classe CreditCard con le annotazioni JAXB



65

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

66

Anatomia di un SOAP WS

67

- 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!

67

SOAP Web Service Endpoints

68

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

68

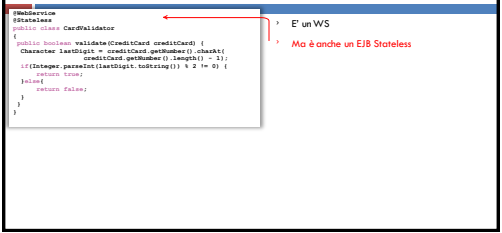
Il WS CardValidator come EJB

```
@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

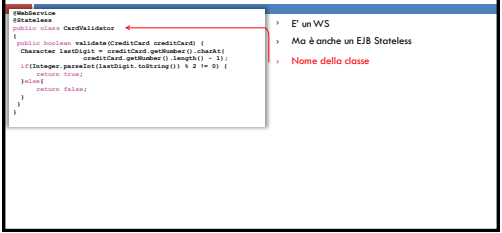
69

Il WS CardValidator come EJB



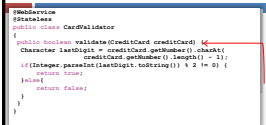
70

Il WS CardValidator come EJB



71

Il WS CardValidator come EJB

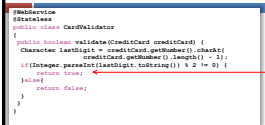


```
@WebService
@Stateless
public class CardValidator
{
 public boolean validate(CreditCard creditCard) {
  Character lastDigit = creditCard.getNumber().charAt(
                        creditCard.getNumber().length() - 1);
  if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
      return true;
  }else{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

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

72

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
- Restituisce vero . . .

73

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
- Restituisce vero . . .
- oppure falso

74

A cosa serve 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
- Flessibilità ed efficienza: lo stesso "servizio" accessibile per consumer fortemente accoppiati (EJB via RMI-IIOP) oppure per consumer debolmente accoppiati (WS via SOAP)
- Stesso servizio offerto in maniera dipendente (ed efficiente) e indipendente dalla piattaforma (meno efficiente)

75

Organizzazione della lezione

76

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

76