

**Service-Oriented Achitecture
& SOAP Web Services (2)**

Corso di Laurea in Informatica, Programmazione Distribuita
 Delfina Malandrino, dmandrino@unisa.it
<http://www.unisa.it/docenti/delfinamalandrino>

1

Organizzazione della lezione

2

- WS in Java
 - ▣ WSDL Mapping
 - ▣ Eccezioni e Fault
 - ▣ Contesto e ciclo di vita
- Putting It All Together
 - ▣ Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - ▣ Il progetto per WS
 - ▣ Testing
 - ▣ WS Client
- Conclusioni

2

Organizzazione della lezione

3

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Putting It All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

3

Vita facile per il programmatore Java

4

- I Web Services, a livello di sistema, sono definiti in termini di XML, operazioni definite da WSDL, scambiando messaggi SOAP
- A livello di programmazione Java, le applicazioni sono definite in termini di oggetti, interfacce e metodi
- Una traduzione da oggetti Java objects ad operazioni WSDL è richiesta
- JAXB runtime (*Java Architecture for XML Binding*) usa le annotazioni per eseguire marshal/unmarshal di una classe da/verso XML
- JWS usa le annotazioni per mappare classi Java in WSDL e per definire
 - come fare il marshal di una invocazione di un metodo in una richiesta SOAP
 - unmarshal di una risposta SOAP in un'istanza del valore restituito dal metodo

4

Vita facile per il programmatore Java

5

- JAX-WS (JSR 224) e WS-Metadata specifications (JSR 181) definiscono due differenti tipi di annotazioni:
 - ▣ *WSDL mapping annotations*: annotazioni che appartengono al package `javax.jws` e permettono il mapping WSDL/Java
 - `@WebMethod`, `@WebResult`, `@WebParam`, e `@OneWay` sono le annotazioni usate per personalizzare i metodi esposti
 - ▣ *SOAP binding annotations*: annotazioni che appartengono al package `javax.jws.soap` e permettono la personalizzazione di SOAP binding
 - `@SOAPBinding` e `@SOAPMessageHandler`

5

Esempi di WSDL Mapping: `@WebService`

6

- L'annotazione `@javax.jws.WebService` marca una classe o una interfaccia come un web service

6

Esempi di WSDL Mapping: @WebService

7

```
@WebService
public class CardValidator {
    //...
```

Una classe

```
@WebService
public interface Validator {
    //...
}

@WebService(endpointInterface =
    "org.agoncal.book.javase7.chapter14.Validator")
public class CardValidator implements Validator {
    //...
```

.... o interfaccia che è un WS

7

Esempi di WSDL Mapping: @WebService

8

- L'annotazione @WebService ha un insieme di attributi (Listing 14-8) che permettono di personalizzare:

- il nome del Web Service nel file WSDL
- la locazione

Listing 14-8. The @WebService API

```
@Retention(RUNTIME) @Target(TYPE)
public @interface WebService {
    String name() default "";
    String targetNamespace() default "";
    String serviceName() default "";
    String portName() default "";
    String wsdlLocation() default "";
    String endpointInterface() default "";
}
```

```
@WebService(portName = "CreditCardValidator",
    serviceName = "ValidatorService")
public class CardValidator {
    //...
```

Possibile cambiare la porta e il servizio

8

Esempi di W

```
@WebService(portName = "CreditCardValidator",
             serviceName = "ValidatorService")
public class CardValidator {
    //...
```

□ Versione originale

```
<service name="CardValidatorService">
  <port name="CardValidatorPort" binding="tns:CardValidatorPortBinding">
    <soap:address location="http://localhost:8080/chapter14/CardValidatorService"/>
  </port>
</service>
</definitions>
```

□ Dopo la modifica

```
<service name="ValidatorService">
  <port name="CreditCardValidator" binding="tns:CreditCardValidatorBinding">
    <soap:address location="http://localhost:8080/chapter14/ValidatorService"/>
  </port>
</service>
```

9

Esempi di WSDL Mapping: @WebMethod

- Di default, tutti i metodi di un SOAP web service sono esposti nel file WSDL e le regole di default vengono usate per il mapping
- Per applicare personalizzazione si può usare l'annotazione `@javax.jws.WebMethod` sui metodi
- Nel nostro esempio:
 - Due metodi verranno rinominati
 - Un metodo verrà escluso dal file WSDL

10

Esempi di WSDL Mapping: @WebMethod

```
[WebService]
public class CardValidator {
    @WebMethod(operationName = "ValidateCreditCard")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }

    @WebMethod(operationName = "ValidateCreditCardNumber")
    public void validate(String creditCardNumber) {
        //Business logic
    }

    @WebMethod(exclude = true)
    public void validate(Long creditCardNumber) {
        //Business logic
    }
}
```

Rename di due metodi

```
[WebService]
public class CardValidator {
    @WebResult(name = "IsValid")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }
}
```

11

Esempi di WSDL Mapping: @WebMethod

```
[WebService]
public class CardValidator {
    @WebMethod(operationName = "ValidateCreditCard")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }

    @WebMethod(operationName = "ValidateCreditCardNumber")
    public void validate(String creditCardNumber) {
        //Business logic
    }

    @WebMethod(exclude = true)
    public void validate(Long creditCardNumber) {
        //Business logic
    }
}
```

Metodo interno, non da far
usare come WS (escluderlo dal
WSDL)

```
[WebService]
public class CardValidator {
    @WebResult(name = "IsValid")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }
}
```

12

Esempi di WSDL Mapping: @WebMethod

```
[WebService]
public class CardValidator {
    @WebMethod(operationName = "ValidateCreditCard")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }

    @WebMethod(operationName = "ValidateCreditCardNumber")
    public void validate(String creditCardNumber) {
        //Business logic
    }

    @WebMethod(exclude = true)
    public void validate(Long creditCardNumber) {
        //Business logic
    }
}
```

```
<!-- Default -->
<xsd:element name="return" type="xs:boolean"/>

<!-- Renamed to isValid -->
<xsd:element name="isValid" type="xs:boolean"/>
```

```
[WebService]
public class CardValidator {
    @WebResult(name = "isValid")
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }
}
```

Si può cambiare il nome del messaggio restituito

13

Esempi di WSDL Mapping: @WebParam

- L'annotazione `@javax.jws.WebParam` permette di personalizzare i parametri dei metodi di un web service
 - ▣ Nomi dei parametri del file WSDL
 - ▣ Il namespace
 - ▣ Tipo

15

Esempi di WSDL Mapping: @WebParam

```

@WebService
public class CardValidator {
    public boolean validate(
        @WebParam(name="Credit-Card", mode = IN)
        CreditCard creditCard) {
        //Business logic
    }
}

```

Viene rinominato in questa maniera ...

Dopo la modifica:

```

<!-- Default -->
<xsd:element name="arg0" type="tns:creditCard" minOccurs="0"/>
<!-- Renamed to Credit-Card -->
<xsd:element name="Credit-Card" type="tns:creditCard" minOccurs="0"/>

```

SOAP Binding

- Un binding descrive come il web service è legato al protocollo di messaging (SOAP)
- Esistono due differenti tipi di programming styles per SOAP binding definiti in WSDL 1.1:
 - ▣ RPC
 - ▣ Document (messaging)
- che influenzano il modo in cui il SOAP body content è strutturato

SOAP Binding

18

- **Document:** Il messaggio SOAP contiene il documento
 - Inviato nel <soap:Body> element senza regole di formattazione aggiuntionali
 - È la scelta di default
- **RPC:** Il messaggio SOAP contiene i parametri ed i valori restituiti
 - Il <soap:Body> contiene un elemento con il nome del metodo remote o procedura da invocare
 - E' presente un elemento per ogni parametro della procedura remota

SOAP Binding

19

- Un SOAP binding (Document or RPC) deve scegliere fra due differenti formati di serialization/deserialization:
 - Literal: Data serializzati secondo un XML schema
 - Encoded: il SOAP encoding specifica come oggetti, strutture, array devono essere serializzati

18

19

SOAP Binding

20

```
@WebService
@SOAPBinding(style = RPC, use = LITERAL)
public class CardValidator {
    public boolean validate(CreditCard creditCard) {
        //Business logic
    }
}
```

Esempio:

Descrive una invocazione RPC (tipo request-reply) e il passaggio di parametri con XML (e non con codifiche binarie per oggetti, arrays, etc.)

SOAP Binding

21

Listing 14-15. WSDL Differences Between Document and RPC Style

```
<!-- Document style -->
<message name="validate">
  <part name="parameters" element="tns:validate"/>
</message>
<message name="validateResponse">
  <part name="parameters" element="tns:validateResponse"/>
</message>
...
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
```

```
<!-- RPC Style -->
<message name="validate">
  <part name="argo" type="tns:creditCard"/>
</message>
<message name="validateResponse">
  <part name="return" type="xsd:boolean"/>
</message>
...
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
```

20

21

Organizzazione della lezione

22

- WS in Java
 - WSDL Mapping
 - **Eccezioni e Fault**
 - Contesto e ciclo di vita
- Putting It All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

Gestione delle eccezioni

23

- In Java, quando si verifica un errore, si lanciano le eccezioni e qualche altra classe dovrà gestirle
- Con SOAP non si può procedere nello stesso modo
 - Consumer e service potrebbero essere scritti in linguaggi diversi e separate dalla rete
- L'idea:
 - Usare SOAP fault in un SOAP message
- JAX-WS runtime **automaticamente** converte le eccezioni Java in SOAP fault messages, restituiti al client
 - Meno lavoro per il programmatore
- Vediamo un esempio...

22

23

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

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

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
- > Cosa succede se il parametro è null?

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
- > Cosa succede se il parametro è null?
- > Qui viene lanciata una eccezione
NullPointerException

Viene lanciata una eccezione SOAP

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>
        soap:Server
      </faultcode>
      <faultstring>
        java.lang.NullPointerException
      </faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>

```

Schema XML per validation

28

Viene lanciata una eccezione SOAP

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>
        soap:Server
      </faultcode>
      <faultstring>
        java.lang.NullPointerException
      </faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>

```

Schema XML per validation

Viene inviato un fault nel body

29

Viene lanciata una eccezione SOAP

```
<?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"
  xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http://schemas.xmlsoap.org/soap/envelope.xsd">
  <soap:Body>
    <soap:Fault>
      <faultcode>
        soap:Server
      </faultcode>
      <faultstring>
        java.lang.NullPointerException
      </faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

- > Schema XML per validation
- > Viene inviato un fault nel body
- > Con il tipo inviato dal codice Java

Vediamo un altro esempio...

30

Lanciare una eccezione specifica

```
@WebService
public class CardValidator throws CardValidatorException {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0){
            return true;
        }else{
            throw new CardValidatorException("The
                credit card number is invalid");
        }
    }
}
```

La classe lancia una eccezione utente

31

Lanciare una eccezione specifica

```

@WebService
public class CardValidator throws CardValidatorException {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if (Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        } else {
            throw new CardValidatorException("The
                credit card number is invalid");
        }
    }
}

```

› La classe lancia una eccezione utente

› Quando il numero è dispari, allora

32

Lanciare una eccezione specifica

```

@WebService
public class CardValidator throws CardValidatorException {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if (Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        } else {
            throw new CardValidatorException("The
                credit card number is invalid");
        }
    }
}

```

› La classe lancia una eccezione utente

› Quando il numero è dispari, allora

› Viene lanciata una eccezione, che JAX-WS
intercetta e rilancia come fault SOAP

33

Lanciare una eccezione specifica

34

Listing 14-24. SOAP Fault in a SOAP Envelope

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>org.agoncal.book.javaee7.chapter14.CardValidatorException</faultstring>
      <detail>
        <ns2:CardValidationFault xmlns:ns2="http://chapter14.javaee7.book.agoncal.org/">
          <message>The credit card number is invalid</message>
        </ns2:CardValidationFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

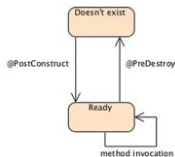
Organizzazione della lezione

35

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Invocare un WS
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

Il ciclo di vita di un WS

36



- I WS SOAP hanno un ciclo di vita simile a quelli dei beans
- Metodi con `PostConstruct` vengono invocati dopo che il WS viene istanziato, mentre metodi con `PreDestroy` vengono invocati prima di chiuderlo

Contesti di un WS

37

```
@WebService
public class CardValidator {
    @Resource
    private WebServiceContext context;
    public boolean validate(CreditCard creditCard) {
        if(!context.isUserInRole("Admin"))
            throw new SecurityException("Only Admins can validate cards");
        //Business logic
    }
}
```

- > Un WS
- > Definizione classe
- > Iniezione del contesto di WS
- > Uso per validare il ruolo dell'utente
- > ... altrimenti si lancia una eccezione

36

42

Deployment Descriptor

44

- SOAP web services permette di definire i metadati usando le annotazioni oppure via XML
- Il deployment descriptor, localizzato sotto la directory WEB-INF, si chiama **webservices.xml**
 - ▣ Sovrascrive oppure estende le annotazioni

File XML per deployment:

webservices.xml

File opzionale

```

<?xml version="1.0" encoding="UTF-8"?>
<webservices xmlns="http://java.sun.com/xml/ns/jaxee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/jaxee
    http://java.sun.com/xml/ns/jaxee/jaxee_web_services_1_3.xsd"
  version="1.3">
  <web-service-description>
    <web-service-description-name>
      CardValidatorWS
    </web-service-description-name>
    <port>
      <port-component-name>
        CardValidator
      </port-component-name>
      <wsdl-port>
        <wsdl-port-overridden-port/>
      </wsdl-port>
      <service-endpoint-interface>
        org.agoncal.book.jaxee7.chapter14.Validator
      </service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>
          CardValidatorServlet
        </servlet-link>
      </service-impl-bean>
    </port>
  </web-service-description>
</webservices>

```

XML schema per la validazione

44

45

File XML per deployment: webservices.xml

File opzionale

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-service xmlns="http://java.sun.com/xml/ns/jaxws"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/jaxws
    http://java.sun.com/xml/ns/jaxws/jaxws_web_services_1_3.xsd"
  version="1.3">
  <web-service-description>
    <web-service-description-name>Nome del WS</web-service-description-name>
  </web-service-description-name>
  <port-component>
    <port-component-name>Nome del WS</port-component-name>
    <card-validator>
      <card-validator-overridden-port/>
      <service-endpoint-interface>
        org.agoncal.book.javaee7.chapter14.Validator
      </service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>
          CardValidatorServlet
        </servlet-link>
      </service-impl-bean>
    </port-component>
  </web-service-description>
</web-service>
```

- XML schema per la validazione
- Nome del WS

46

File XML per deployment: webservices.xml

File opzionale

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-service xmlns="http://java.sun.com/xml/ns/jaxws"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/jaxws
    http://java.sun.com/xml/ns/jaxws/jaxws_web_services_1_3.xsd"
  version="1.3">
  <web-service-description>
    <web-service-description-name>Nome del WS</web-service-description-name>
  </web-service-description-name>
  <port-component>
    <port-component-name>Informazioni sulla porta</port-component-name>
    <card-validator>
      <card-validator-overridden-port/>
      <service-endpoint-interface>
        org.agoncal.book.javaee7.chapter14.Validator
      </service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>
          CardValidatorServlet
        </servlet-link>
      </service-impl-bean>
    </port-component>
  </web-service-description>
</web-service>
```

- XML schema per la validazione
- Nome del WS
- Informazioni sulla porta

47

File XML per deployment:

webservices.xml

File opzionale

```

<?xml version="1.0" encoding="UTF-8"?>
<web-service xmlns="http://java.sun.com/xml/ns/jaxws"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/jaxws
    http://java.sun.com/xml/ns/jaxws/jaxws_web_services_1_3.xsd"
  version="1.3">
  <web-service-description>
    <web-service-description-name>
      CardValidatorWS
    </web-service-description-name>
    <port-component>
      <port-component-name>
        CardValidator
      </port-component-name>
      <wsdl-port>OverriddenPort</wsdl-port>
      <service-endpoint-interface>
        org.agoncal.book.javaee7.chapter14.Validator
      </service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>
          CardValidatorServlet
        </servlet-link>
      </service-impl-bean>
    </port-component>
  </web-service-description>
</web-service>

```

- > XML schema per la validazione
- > Nome del WS
- > Informazioni sulla porta

Interfaccia

File XML per deployment:

webservices.xml

File opzionale

```

<?xml version="1.0" encoding="UTF-8"?>
<web-service xmlns="http://java.sun.com/xml/ns/jaxws"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/jaxws
    http://java.sun.com/xml/ns/jaxws/jaxws_web_services_1_3.xsd"
  version="1.3">
  <web-service-description>
    <web-service-description-name>
      CardValidatorWS
    </web-service-description-name>
    <port-component>
      <port-component-name>
        CardValidator
      </port-component-name>
      <wsdl-port>OverriddenPort</wsdl-port>
      <service-endpoint-interface>
        org.agoncal.book.javaee7.chapter14.Validator
      </service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>
          CardValidatorServlet
        </servlet-link>
      </service-impl-bean>
    </port-component>
  </web-service-description>
</web-service>

```

- > XML schema per la validazione
- > Nome del WS
- > Informazioni sulla porta
- > Interfaccia

> Servlet di implementazione

48

49

Invoking Web Services

50

- Usando il WSDL per descrivere il servizio . . .
- . . . e alcuni stub (proxy) generati automaticamente dal sistema, si può facilmente usare WS senza usare direttamente HTTP, SOAP, etc.
- Una situazione abbastanza simile a quanto viene fatto per RMI
- La differenza rispetto ad RMI è che, sul remote host, il Web service può essere scritto in un linguaggio diverso da Java

Invoking Web Services

51

- WSDL è il contratto standard fra il consumer ed il servizio
- Metro fornisce un utility tool WSDL-to-Java (wsimport) per generare interfacce e classi Java a partire da un file WSDL
- Questi proxy sono la rappresentazione Java di un web service endpoint (servlet or EJB)
- Questi proxy instradano le chiamate locali Java al remote web service usando HTTP

50

51

Invoking Web Services

52

- Quando un metodo sul proxy viene invocato
 - ▢ converte i parametri del metodo in un messaggio SOAP (la richiesta)
 - ▢ Invia la richiesta al web service endpoint
- Per ottenere un risultato, la risposta SOAP viene convertita in una istanza del tipo restituito
- Non si ha necessità di conoscere il "lavoro interno" del proxy, né il suo codice

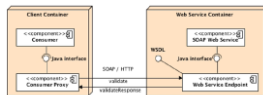


Figure 14-3. A consumer invoking a web service through a proxy

Anatomia di un SOAP Consumer

53

- Poiché JAX-WS è disponibile in Java SE, un SOAP web service consumer può essere:
 - ▢ un qualunque tipo di Java code con una main class in esecuzione nella JVM
 - ▢ una qualunque Java EE component in esecuzione in un container (Web, EJB o application client container)
- Se eseguita in un container, il consumer può ottenere una istanza del proxy attraverso injection oppure programmatically (creandola)
- Per iniettare un web service, bisogna usare l'annotazione `@javax.xml.ws.WebServiceRef` annotation o un CDI producer

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

» Una classe standard

54

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

» Una classe standard

» Metodo statico

55

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

- › Una classe standard
- › Metodo statico
- › Si dichiara una carta di credito

56

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

- › Una classe standard
- › Metodo statico
- › Si dichiara una carta di credito
- › La si inizializza

57

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

- › Una classe standard
- › Metodo statico
- › Si dichiara una carta di credito
- › La si inizializza
- › Si ottiene un riferimento al proxy, da cui si ottiene un riferimento al service ..

58

Invocazione da Client (Java SE class)

```
public class WebServiceConsumer {
    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            new CardValidatorService().getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

- › Una classe standard
- › Metodo statico
- › Si dichiara una carta di credito
- › La si inizializza
- › Si ottiene un riferimento al proxy, da cui si ottiene un riferimento al service ...
- › ... che si invoca come un metodo "qualsiasi"

- ❑ Il consumer usa una istanza di CardValidatorService (generata da WSDL grazie a wsimport) usando la keyword **new**
- ❑ Con getCardValidatorPort() invoca il business method localmente
- ❑ Una chiamata locale viene fatta sul metodo validate() del proxy, che invocherà il remote web service, creando la richiesta SOAP, facendo il marshal dei messaggi credit card messages, ecc.
- ❑ Il proxy trova il target service perchè il default endpoint URL è nel WSDL file, e quindi integrato nella implementazione proxy

59

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

Annotazione per l'iniezione

Injection per ottenere un
riferimento al SOAP Web
Service client proxy

Simple main Java class running in an application client container (ACC)
and using the @WebServiceRef annotation

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

Annotazione per l'iniezione

Dichiarazione di un proxy, iniettato dal
container

Injection per ottenere un
riferimento al SOAP Web
Service client proxy

Simple main Java class running in an application client container (ACC)
and using the @WebServiceRef annotation

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();
        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

> Annotazione per l'iniezione

Injection per ottenere un riferimento al SOAP Web Service client proxy

> Dichiarazione di un proxy, iniettato dal container

> Oggetto da inviare

Simple main Java class running in an application client container (ACC) and using the @WebServiceRef annotation

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();
        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

> Annotazione per l'iniezione

Injection per ottenere un riferimento al SOAP Web Service client proxy

> Dichiarazione di un proxy, iniettato dal container

> Oggetto da inviare

> Set degli attributi

Simple main Java class running in an application client container (ACC) and using the @WebServiceRef annotation

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

> Annotazione per l'iniezione

Injection per ottenere un riferimento al SOAP Web Service client proxy

> Dichiarazione di un proxy, iniettato dal container

> Oggetto da inviare

> Set degli attributi

> Uso del proxy iniettato

Simple main Java class running in an application client container (ACC) and using the @WebServiceRef annotation

Invocazione da un Container (invoking with Injection)

```
public class WebServiceConsumer {
    @WebServiceRef
    private static CardValidatorService cardValidatorService;

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard();

        creditCard.setNumber("12341234");
        creditCard.setExpiryDate("10/12");
        creditCard.setType("VISA");
        creditCard.setControlNumber(1234);

        CardValidator cardValidator =
            cardValidatorService.getCardValidatorPort();
        cardValidator.validate(creditCard);
    }
}
```

> Annotazione per l'iniezione

Injection per ottenere un riferimento al SOAP Web Service client proxy

> Dichiarazione di un proxy, iniettato dal container

> Oggetto da inviare

> Set degli attributi

> Uso del proxy iniettato

> ... per invocare il WS

Simple main Java class running in an application client container (ACC) and using the @WebServiceRef annotation

Organizzazione della lezione

66

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Putting It All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

66

Diagramma dell'esempio

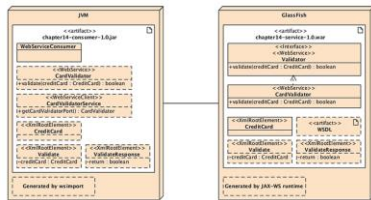
67

- Il SOAP web service `CardValidator` controlla che una carta di credito sia valida
- Ha un metodo che prende in input un oggetto `CreditCard`, applica un qualche algoritmo (☺) e restituisce `true` se la carta è valida, `false` altrimenti
- Dopo il deploy su GlassFish, `wsimport` può essere usato per la generazione degli artifacts necessari per il consumer
- Il consumer può quindi invocare il web service per validare carte di credito

67

Diagramma dell'esempio

68



68

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

Inizio dell'elemento root dell'XML

POJO usato come parametro per il metodo `validate()` del Web Service

69

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

Inizio dell'elemento root dell'XML

Tutti i campi saranno mappati su XML

POJO usato come parametro per il metodo `validate()` del Web Service

70

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

Inizio dell'elemento root dell'XML

Tutti i campi saranno mappati su XML

Attributo obbligatorio

POJO usato come parametro per il metodo `validate()` del Web Service

71

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

- > Inizio dell'elemento root dell'XML
- > Tutti i campi saranno mappati su XML
- > Attributo obbligatorio
- > Attributo obbligatorio, con il nome XML diverso

POJO usato come parametro per il metodo `validate()` del Web Service

72

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

- > Inizio dell'elemento root dell'XML
- > Tutti i campi saranno mappati su XML
- > Attributo obbligatorio
- > Attributo obbligatorio, con il nome XML diverso
- > Attributo obbligatorio, con il nome XML diverso

POJO usato come parametro per il metodo `validate()` del Web Service

73

CreditCard: La classe per la carta di credito

```

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}

```

- › Inizio dell'elemento root dell'XML
- › Tutti i campi saranno mappati su XML
- › Attributo obbligatorio
- › Attributo obbligatorio, con il nome XML diverso
- › Attributo obbligatorio, con il nome XML diverso
- › Attributo obbligatorio

POJO usato come parametro per il metodo `validate()` del Web Service

Il WebService

```

@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
    "org.egonomal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if (Integer.parseInt(lastDigit.toString()) % 2 == 0) {
            return true;
        }
        return false;
    }
}

```

Con annotazione da WS...

Il WebService

```

@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
    "org.agoodal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    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;
        }
    }
}

```

Con annotazione da WS...

... si dichiara l'interfaccia

Il WebService

```

@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
    "org.agoodal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    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;
        }
    }
}

```

Con annotazione da WS...

... si dichiara l'interfaccia

Poi si dichiara il WS

II WebService

```

@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
    "org.egoncal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 == 0) {
            return true;
        }
        return false;
    }
}

```

> Con annotazione da WS...

> ... si dichiara l'interfaccia

> Poi si dichiara il WS

> Con la classe che
implementa l'interfaccia

II WebService

```

@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
    "org.egoncal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    public boolean validate(CreditCard creditCard) {
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);
        if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
            return true;
        }
        return false;
    }
}

```

> Con annotazione da WS...

> ... si dichiara l'interfaccia

> Poi si dichiara il WS

> Con la classe che
implementa l'interfaccia

Metodo offerto come WS

Organizzazione della lezione

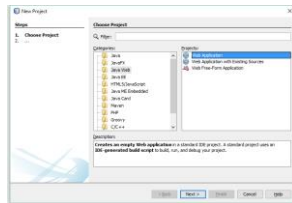
80

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Putting It All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

80

WS in NetBeans: creazione di un progetto

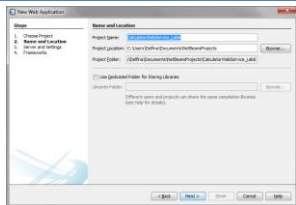
81



81

WS in NetBeans: creazione di un progetto

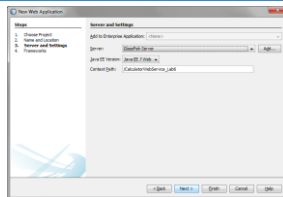
82



82

WS in NetBeans: creazione di un progetto

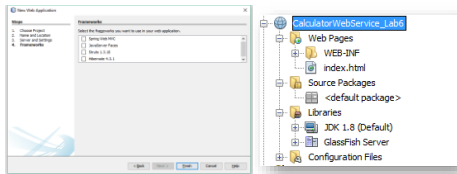
83



83

WS in NetBeans: creazione di un progetto

84

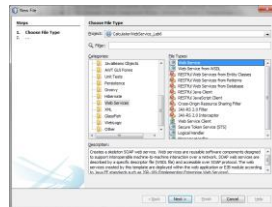


84

WS in NetBeans: creazione di un progetto

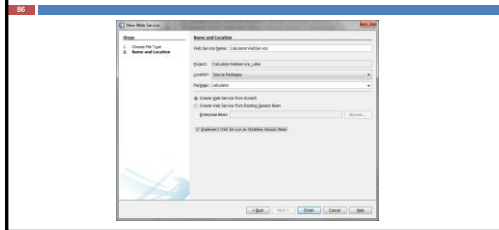
85

Right-click sul progetto → New



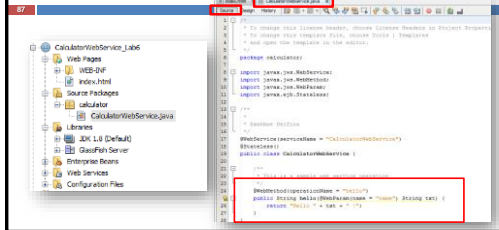
85

WS in NetBeans: creazione di un WS



86

WS in NetBeans: la situazione



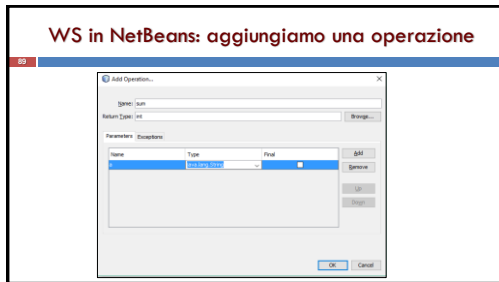
87

WS in NetBeans: la vista di DESIGN



88

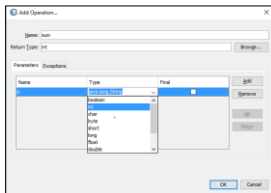
WS in NetBeans: aggiungiamo una operazione



89

WS in NetBeans: aggiungiamo un parametro

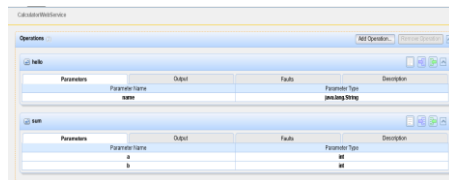
90



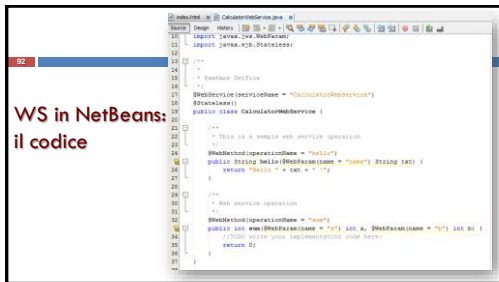
90

WS in NetBeans: dopo aver aggiunto un secondo parametro

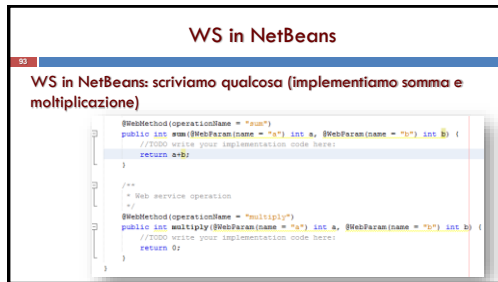
91



91



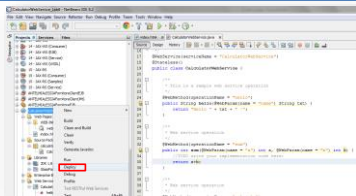
92



93

WS in NetBeans: deploy

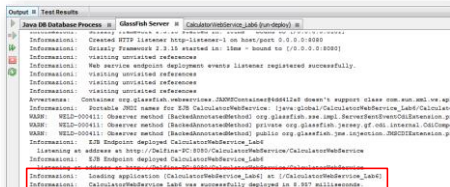
94



94

WS in NetBeans: sul server

95



95

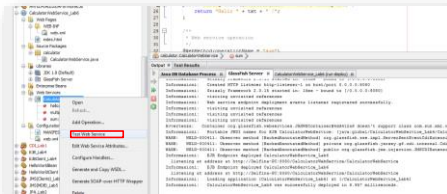
Organizzazione della lezione

96

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Putting it All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

WS in NetBeans: testing

97



96

97

98

WS in NetBeans: Test

CalculatorWebService Web Service Tester

This form will allow you to test your web-service implementation ([WSDL File](#))

To provide an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

```

public abstract int calculate CalculatorWebService.add(int a)
    add();

public abstract int calculate CalculatorWebService.multiply(int a)
    multiply();

public abstract java.lang.String calculate CalculatorWebService.add(java.lang.String)
    addStr();

```

98

99

WS in NetBeans: testing

Cosa è successo sul server

Output | Test Results

```

Java SE Database Process | GlassFish Server | CalculatorWebServiceLab (runDebug).m
...
WAS: WELD-00001: Observer method [BackedAnnotationMethod] private org.glassfish.jersey.gf.cdi.internal.CdiC...
WAS: WELD-00001: Observer method [BackedAnnotationMethod] private org.glassfish.jersey.gf.cdi.internal.CdiC...
Informazioni: EJB Endpoint deployed CalculatorWebServiceLab
Informazioni: Listening at address at http://localhost:8080/CalculatorWebService/CalculatorWebService
Informazioni: EJB Endpoint deployed CalculatorWebServiceLab
Informazioni: Listening at address at http://localhost:8080/CalculatorWebService/CalculatorWebService
Informazioni: Loading application [CalculatorWebServiceLab] as [/CalculatorWebServiceLab]
Informazioni: CalculatorWebServiceLab was successfully deployed in 9.547 milliseconds
Informazioni: Deploying webapp with http://localhost:8080/CalculatorWebService/CalculatorWebService/WEB-INF
Informazioni: Status of WSDL in scope...
Informazioni: Denetazione del codice in corso...
Informazioni: Compilazione del codice in corso...
Informazioni: Import successful
Informazioni: Deploying webapp with http://localhost:8080/CalculatorWebService/CalculatorWebService/WEB-INF
Informazioni: Status of WSDL in scope...
Informazioni: Denetazione del codice in corso...
Informazioni: Compilazione del codice in corso...
Informazioni: Import successful

```

99

WS in NetBeans: Test

101

WS in NetBeans: Test

CalculatorWebService Web Service Tester

This tool will allow you to test your web service implementation ([WSIS 2.0](#)).

To initiate an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods:

- get** (int) returns CalculatorWebServices.get(int)
- getDouble** (int) returns CalculatorWebServices.getDouble(int)
- getLong** (int) returns CalculatorWebServices.getLong(int)

WS in NetBeans: Test



102

SOAP Request

```
<?xml version='1.0' encoding='UTF-8'?><S:Envelope xmlns:S='http://schemas.xmlsoap.org/soap/envelope/'>
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sum xmlns:ns2='http://calculator/'>
      <a>3</a>
      <b>5</b>
    </ns2:sum>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version='1.0' encoding='UTF-8'?><S:Envelope xmlns:S='http://schemas.xmlsoap.org/soap/envelope/'>
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sumResponse xmlns:ns2='http://calculator/'>
      <return>8</return>
    </ns2:sumResponse>
  </S:Body>
</S:Envelope>
```

103

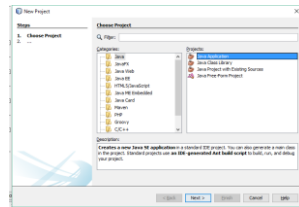
Organizzazione della lezione

104

- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Invocare un WS
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni

WS in NetBeans: Client

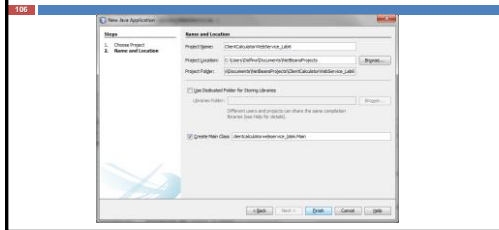
105



104

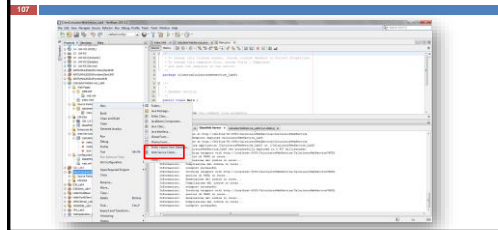
105

WS in NetBeans: Client



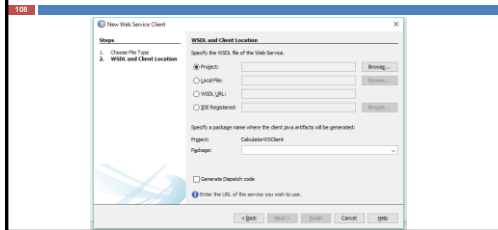
106

WS in NetBeans: creiamo un WS Client



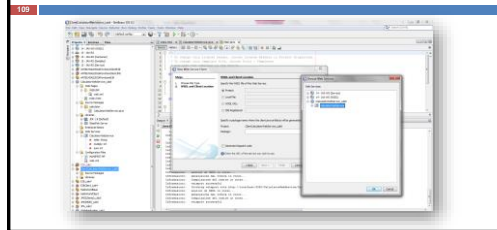
107

WS in NetBeans: creiamo un WS Client



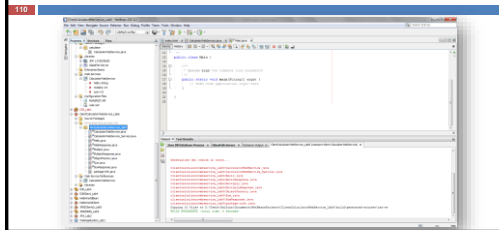
108

WS in NetBeans: creiamo un WS Client

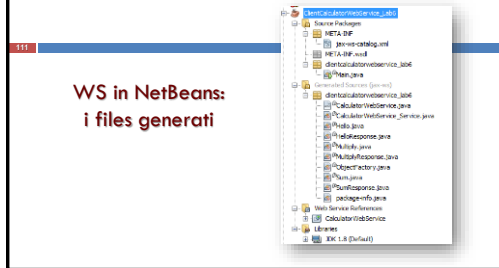


109

La situazione attuale

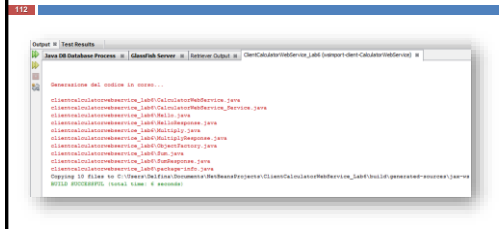


110

WS in NetBeans:
i files generati

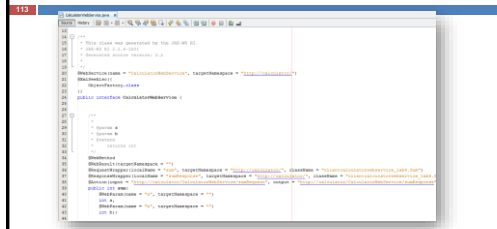
111

WS in NetBeans: Output



112

WS in NetBeans: l'interfaccia generata (sul client)



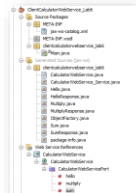
113

WS in NetBeans: il Client (VUOTO)

114

Dal progetto WS...

115



114

115

WS in NetBeans: Drag & Drop operazione hello, sum multiply

```

116  CalculatorWebService.java
117  //
118  //
119  //
120  //
121  //
122  //
123  //
124  //
125  //
126  //
127  //
128  //
129  //
130  //
131  //
132  //
133  //
134  //
135  //
136  //
137  //
138  //
139  //
140  //
141  //
142  //
143  //
144  //
145  //
146  //
147  //
148  //
149  //
150  //
151  //
152  //
153  //
154  //
155  //
156  //
157  //
158  //
159  //
160  //
161  //
162  //
163  //
164  //
165  //
166  //
167  //
168  //
169  //
170  //
171  //
172  //
173  //
174  //
175  //
176  //
177  //
178  //
179  //
180  //
181  //
182  //
183  //
184  //
185  //
186  //
187  //
188  //
189  //
190  //
191  //
192  //
193  //
194  //
195  //
196  //
197  //
198  //
199  //
200  //
201  //
202  //
203  //
204  //
205  //
206  //
207  //
208  //
209  //
210  //
211  //
212  //
213  //
214  //
215  //
216  //
217  //
218  //
219  //
220  //
221  //
222  //
223  //
224  //
225  //
226  //
227  //
228  //
229  //
230  //
231  //
232  //
233  //
234  //
235  //
236  //
237  //
238  //
239  //
240  //
241  //
242  //
243  //
244  //
245  //
246  //
247  //
248  //
249  //
250  //
251  //
252  //
253  //
254  //
255  //
256  //
257  //
258  //
259  //
260  //
261  //
262  //
263  //
264  //
265  //
266  //
267  //
268  //
269  //
270  //
271  //
272  //
273  //
274  //
275  //
276  //
277  //
278  //
279  //
280  //
281  //
282  //
283  //
284  //
285  //
286  //
287  //
288  //
289  //
290  //
291  //
292  //
293  //
294  //
295  //
296  //
297  //
298  //
299  //
300  //
301  //
302  //
303  //
304  //
305  //
306  //
307  //
308  //
309  //
310  //
311  //
312  //
313  //
314  //
315  //
316  //
317  //
318  //
319  //
320  //
321  //
322  //
323  //
324  //
325  //
326  //
327  //
328  //
329  //
330  //
331  //
332  //
333  //
334  //
335  //
336  //
337  //
338  //
339  //
340  //
341  //
342  //
343  //
344  //
345  //
346  //
347  //
348  //
349  //
350  //
351  //
352  //
353  //
354  //
355  //
356  //
357  //
358  //
359  //
360  //
361  //
362  //
363  //
364  //
365  //
366  //
367  //
368  //
369  //
370  //
371  //
372  //
373  //
374  //
375  //
376  //
377  //
378  //
379  //
380  //
381  //
382  //
383  //
384  //
385  //
386  //
387  //
388  //
389  //
390  //
391  //
392  //
393  //
394  //
395  //
396  //
397  //
398  //
399  //
400  //
401  //
402  //
403  //
404  //
405  //
406  //
407  //
408  //
409  //
410  //
411  //
412  //
413  //
414  //
415  //
416  //
417  //
418  //
419  //
420  //
421  //
422  //
423  //
424  //
425  //
426  //
427  //
428  //
429  //
430  //
431  //
432  //
433  //
434  //
435  //
436  //
437  //
438  //
439  //
440  //
441  //
442  //
443  //
444  //
445  //
446  //
447  //
448  //
449  //
450  //
451  //
452  //
453  //
454  //
455  //
456  //
457  //
458  //
459  //
460  //
461  //
462  //
463  //
464  //
465  //
466  //
467  //
468  //
469  //
470  //
471  //
472  //
473  //
474  //
475  //
476  //
477  //
478  //
479  //
480  //
481  //
482  //
483  //
484  //
485  //
486  //
487  //
488  //
489  //
490  //
491  //
492  //
493  //
494  //
495  //
496  //
497  //
498  //
499  //
500  //
501  //
502  //
503  //
504  //
505  //
506  //
507  //
508  //
509  //
510  //
511  //
512  //
513  //
514  //
515  //
516  //
517  //
518  //
519  //
520  //
521  //
522  //
523  //
524  //
525  //
526  //
527  //
528  //
529  //
530  //
531  //
532  //
533  //
534  //
535  //
536  //
537  //
538  //
539  //
540  //
541  //
542  //
543  //
544  //
545  //
546  //
547  //
548  //
549  //
550  //
551  //
552  //
553  //
554  //
555  //
556  //
557  //
558  //
559  //
560  //
561  //
562  //
563  //
564  //
565  //
566  //
567  //
568  //
569  //
570  //
571  //
572  //
573  //
574  //
575  //
576  //
577  //
578  //
579  //
580  //
581  //
582  //
583  //
584  //
585  //
586  //
587  //
588  //
589  //
590  //
591  //
592  //
593  //
594  //
595  //
596  //
597  //
598  //
599  //
600  //
601  //
602  //
603  //
604  //
605  //
606  //
607  //
608  //
609  //
610  //
611  //
612  //
613  //
614  //
615  //
616  //
617  //
618  //
619  //
620  //
621  //
622  //
623  //
624  //
625  //
626  //
627  //
628  //
629  //
630  //
631  //
632  //
633  //
634  //
635  //
636  //
637  //
638  //
639  //
640  //
641  //
642  //
643  //
644  //
645  //
646  //
647  //
648  //
649  //
650  //
651  //
652  //
653  //
654  //
655  //
656  //
657  //
658  //
659  //
660  //
661  //
662  //
663  //
664  //
665  //
666  //
667  //
668  //
669  //
670  //
671  //
672  //
673  //
674  //
675  //
676  //
677  //
678  //
679  //
680  //
681  //
682  //
683  //
684  //
685  //
686  //
687  //
688  //
689  //
690  //
691  //
692  //
693  //
694  //
695  //
696  //
697  //
698  //
699  //
700  //
701  //
702  //
703  //
704  //
705  //
706  //
707  //
708  //
709  //
710  //
711  //
712  //
713  //
714  //
715  //
716  //
717  //
718  //
719  //
720  //
721  //
722  //
723  //
724  //
725  //
726  //
727  //
728  //
729  //
730  //
731  //
732  //
733  //
734  //
735  //
736  //
737  //
738  //
739  //
740  //
741  //
742  //
743  //
744  //
745  //
746  //
747  //
748  //
749  //
750  //
751  //
752  //
753  //
754  //
755  //
756  //
757  //
758  //
759  //
760  //
761  //
762  //
763  //
764  //
765  //
766  //
767  //
768  //
769  //
770  //
771  //
772  //
773  //
774  //
775  //
776  //
777  //
778  //
779  //
780  //
781  //
782  //
783  //
784  //
785  //
786  //
787  //
788  //
789  //
790  //
791  //
792  //
793  //
794  //
795  //
796  //
797  //
798  //
799  //
800  //
801  //
802  //
803  //
804  //
805  //
806  //
807  //
808  //
809  //
810  //
811  //
812  //
813  //
814  //
815  //
816  //
817  //
818  //
819  //
820  //
821  //
822  //
823  //
824  //
825  //
826  //
827  //
828  //
829  //
830  //
831  //
832  //
833  //
834  //
835  //
836  //
837  //
838  //
839  //
840  //
841  //
842  //
843  //
844  //
845  //
846  //
847  //
848  //
849  //
850  //
851  //
852  //
853  //
854  //
855  //
856  //
857  //
858  //
859  //
860  //
861  //
862  //
863  //
864  //
865  //
866  //
867  //
868  //
869  //
870  //
871  //
872  //
873  //
874  //
875  //
876  //
877  //
878  //
879  //
880  //
881  //
882  //
883  //
884  //
885  //
886  //
887  //
888  //
889  //
890  //
891  //
892  //
893  //
894  //
895  //
896  //
897  //
898  //
899  //
900  //
901  //
902  //
903  //
904  //
905  //
906  //
907  //
908  //
909  //
910  //
911  //
912  //
913  //
914  //
915  //
916  //
917  //
918  //
919  //
920  //
921  //
922  //
923  //
924  //
925  //
926  //
927  //
928  //
929  //
930  //
931  //
932  //
933  //
934  //
935  //
936  //
937  //
938  //
939  //
940  //
941  //
942  //
943  //
944  //
945  //
946  //
947  //
948  //
949  //
950  //
951  //
952  //
953  //
954  //
955  //
956  //
957  //
958  //
959  //
960  //
961  //
962  //
963  //
964  //
965  //
966  //
967  //
968  //
969  //
970  //
971  //
972  //
973  //
974  //
975  //
976  //
977  //
978  //
979  //
980  //
981  //
982  //
983  //
984  //
985  //
986  //
987  //
988  //
989  //
990  //
991  //
992  //
993  //
994  //
995  //
996  //
997  //
998  //
999  //
1000  //

```

116

WS in NetBeans: Drag & Drop operazione hello, sum multiply

```

117  private static String hello(java.lang.String name) {
118      clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
119          new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
120      clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
121      return port.hello(name);
122  }
123
124  private static int multiply(int a, int b) {
125      clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
126          new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
127      clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
128      return port.multiply(a, b);
129  }
130
131  private static int sum(int a, int b) {
132      clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
133          new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
134      clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
135      return port.sum(a, b);
136  }

```

117

WS in NetBeans: esecuzione

WS in NetBeans:
... un po' di codice

```

4  * and open the template in the editor.
5  */
6  package clientcalculatorwebservice_lab6;
7
8  /**
9   *
10   * Author Delfina
11   */
12  public class Main {
13
14      /**
15       * Spawns args the command line arguments
16       */
17      public static void main(String[] args) {
18          System.out.println("Hello " + "Delfina");
19          System.out.println("la somma di 5+3= " + sum(5,3));
20      }
21  }

```

WS in NetBeans: RUN

Output | Test Results

Java DB Database Process | GlassFish Server | Retriever Output | ClientCalculatorWebService_Lab6 (run-single) |

```

ant -f C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6 -Djava.class.path=.
init:
Deleting: C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6\build\classes
delete-jar:
Updating property file: C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6\build\classes
waitimport-init:
waitimport-client-CalculatorWebService:
files are up to date
waitimport-client-generate:
Compiling 1 source file to C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6\build\classes
compile-single:
run-single:
Hello Delfina !
la somma di 5+3= 8
BUILD SUCCESSFUL (total time: 6 seconds)

```

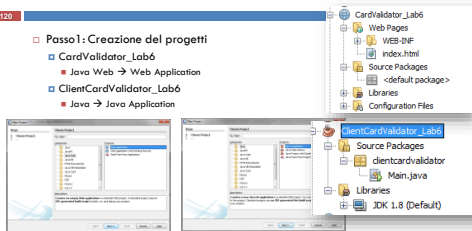
118

119

CreditCardValidator Web Service

Passo 1: Creazione dei progetti

- CardValidator_Lab6
 - Java Web → Web Application
- ClientCardValidator_Lab6
 - Java → Java Application



120

CreditCardValidator Web Service

La classe CreditCard

Listing 14-34. The CreditCard Class with JAXB Annotations

```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
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
}
```

121

CreditCardValidator Web Service

122

□ L'interfaccia

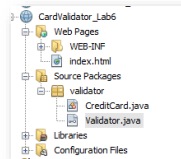
Listing 14-35. The Validator Web Service Interface

```
@WebService  
public interface Validator {  
    public boolean validate(CreditCard creditCard);  
}
```

La situazione attuale

123

□ In NetBeans



122

123

CreditCardValidator Web Service

124

La classe CardValidator

Listing 14-36. The CardValidator Web Service Bean

```
@WebService(endpointInterface = "org.agomcal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {

    public boolean validate(CreditCard creditCard) {

        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length() - 1);

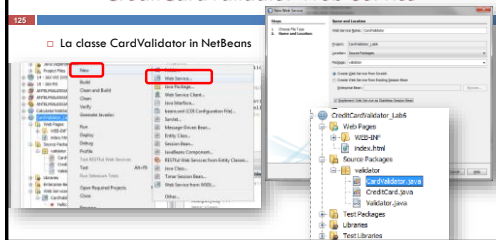
        if (Integer.parseInt(lastDigit.toString()) % 2 == 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

124

CreditCardValidator Web Service

125

La classe CardValidator in NetBeans

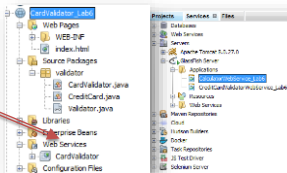


125

CreditCardValidator Web Service

126

- Build & Deploy del Web Service
- Creazione della cartella Web Services con all'interno il Web Service **CardValidator**

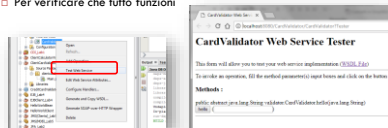


126

CreditCardValidator Web Service

127

- Per verificare che tutto funzioni



- E' presente il Web Service di default!
- Rimuoviamo hello(), ed aggiungiamo validate()

127

CreditCardValidator Web Service

128

□ Nuova situazione



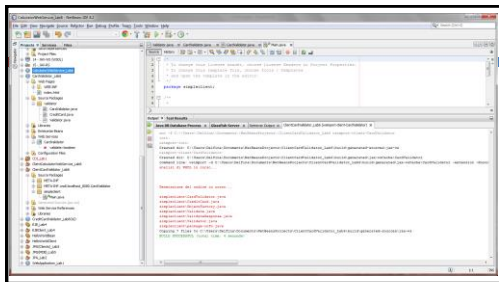
128

CreditCardValidator Web Service

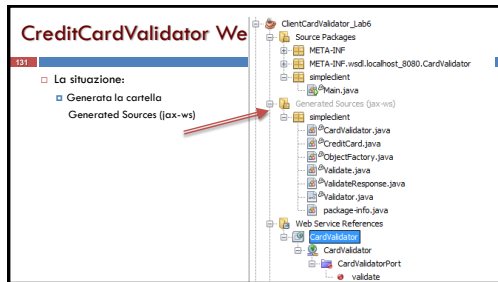
129

- Ora il Consumer: ClientCardValidator_Lab6
- Right-click sul progetto client → new Web Service client

129



130



131

CreditCardValidator We

132

- CardValidator rappresenta l'istanza generata dal file WSDL con wsimport

132

Il client: invoking programmatically

133

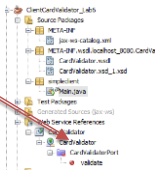
- Il consumer è esterno al container, bisogna invocarlo da codice
- Il CardValidator web service non invocato direttamente
- Il consumer usa una istanza di CardValidator (generata dal WSDL grazie a wsimport) usando la keyword new
- Deve a questo punto ottenere il proxy CardValidator class (getCardValidatorPort()) per invocare il metodo di business localmente
- Una chiamata viene fatta sul metodo validate() del proxy che a sua volta invocherà il web service remote
 - Creando una richiesta SOAP
 - Facendo il marshalling del credit card messages, ecc

```
private static boolean validate(CreditCard creditCard) {
    CardValidator service = new CardValidator();
    Validator port = service.getCardValidatorPort();
    return port.validate(creditCard);
}
```

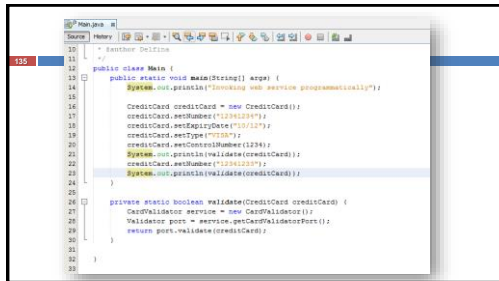
133

Il client: invoking programmatically

- Drag & Drop da qui nella classe del client



134



135

Il client: invoking programmatically

136

```

Output | Test Results
Java DB Database Process | GlassFish Server | Retriever Output | ClientCardValidator_Lab6 (run-single)
depa-jar:
Updating property file: C:\Users\Delfina\Documents\NetBeansProjects\ClientCardValida
wsimport-init:
wsimport-client-CardValidator:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\Delfina\Documents\NetBeansProjects\ClientCardVa
compile-single:
run-single:
Invoking web service programmatically
true
false
BUILD SUCCESSFUL (total time: 4 seconds)
  
```

136

Organizzazione della lezione

- 137
- WS in Java
 - ▣ WSDL Mapping
 - ▣ Eccezioni e Fault
 - ▣ Contesto e ciclo di vita
 - Putting It All Together
 - ▣ Un esempio riassuntivo
 - Supporto ai WS in Netbeans
 - ▣ Il progetto per WS
 - ▣ Testing
 - ▣ WS Client
 - Conclusioni

137



138