
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.javaee7.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 {
  //...
}
```

9

- 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

10

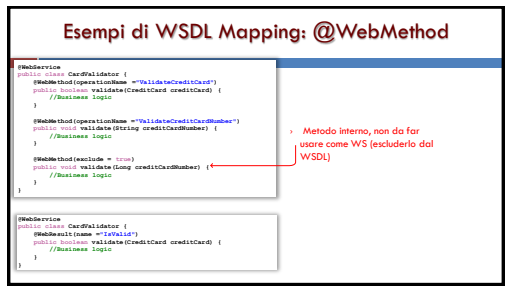
- Di default, tutti i metodi di un SOAP web service sono esposti nel file WSDL e le regole di default vengono usate per il mapping
- Per applicare personalizzazione si può usare l'annotazione @javax.jws.WebMethod sui metodi
- Nel nostro esempio:
  - Due metodi verranno rinominati
  - Un metodo verrà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

10

## Esempi di WSDL Mapping: @WebMethod

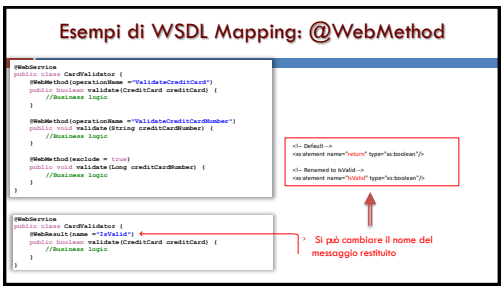

11

## Esempi di WSDL Mapping: @WebMethod




12

## Esempi di WSDL Mapping: @WebMethod




13

## Esempi di WSDL Mapping: @WebParam

15

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

16

## SOAP Binding

17

- Un binding descrive come il web service è legato al protocollo di massaging (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

17

## SOAP Binding

18

- **Document**: Il messaggio SOAP contiene il documento
  - Inviato nel <soap:Body> element senza regole di formattazione addizionali
  - É 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

18

## 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

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.)

20

## 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="arg0" 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"/>
```

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

22

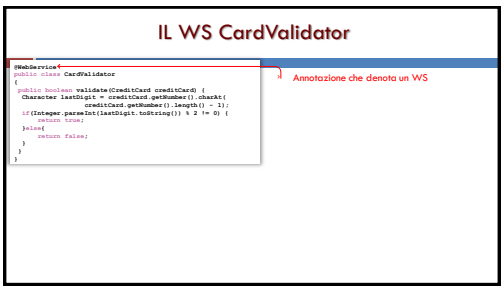
## Gestione delle eccezioni

23

- In Java, quando si verifica un errore, si lanciano le eccezioni e qualche altra classe dovrà gestirle
- Con SOAP non si può procedure 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…

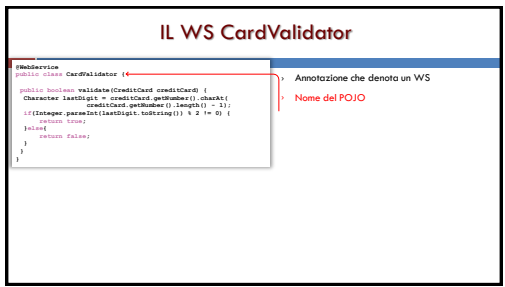
23

## IL WS CardValidator



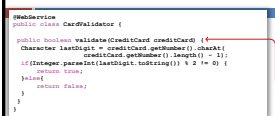

24

## IL WS CardValidator




25

## IL WS CardValidator

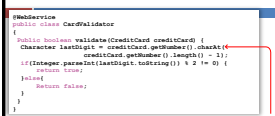


```
@WebService
public class CardValidator {

 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?

26

## IL WS CardValidator

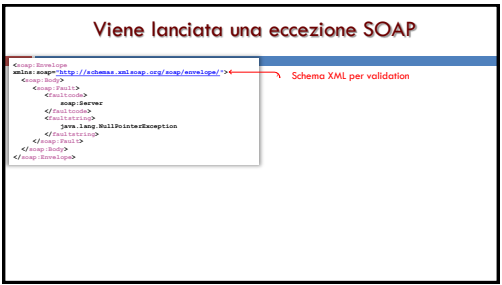


```
@WebService
public class CardValidator
{
 Public boolean validate(CreditCard creditCard) {
  Character lastDigit = creditCard.getNumber().charAt(
                creditCard.getNumber().length() - 1);
  if(Integer.parseInt(lastDigit.toString()) % 2 != 0) {
     return true;
  }else{
     Return false;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

- Annotazione che denota un WS
- Nome del POJO
- Cosa succede se il parametro é null?
- Qui viene lanciata una eccezione `NullPointerException`

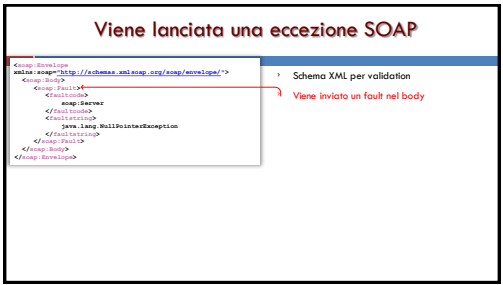
27

## Viene lanciata una eccezione SOAP



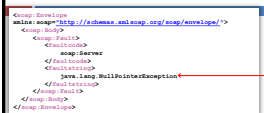

28

## Viene lanciata una eccezione SOAP




29

## Viene lanciata una eccezione SOAP




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

Vediamo un altro esempio…

30

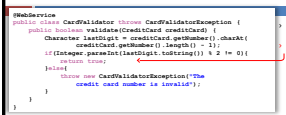
## Lanciare una eccezione specifica

```
@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

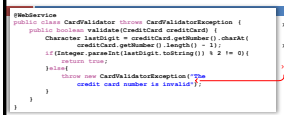
## Lanciare una eccezione specifica




- La classe lancia una eccezione utente
- Quando il numero è dispari, allora

32

## Lanciare una eccezione specifica




- 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>
```

34

## Organizzazione della lezione

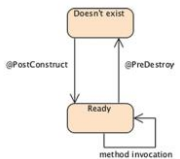
35

- 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

35

## Il ciclo di vita di un WS






- 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

36

## Contesti di un WS

```
@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

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

44

## File XML per deployment:

File opzionale

webservices.xml

```
<webservices xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/javaee_web_services_1_3.xsd"
version="1.3">
<webservice-description>
   <webservice-description-name>
      CardValidatorWS
   </webservice-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>
  </webservice-description>
</webservices>
```

XML schema per la validazione

45

## File XML per deployment: webservices.xml

File opzionale

```
<webservices xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/javaee_web_services_1_3.xsd"
 version="1.3">
 <webservice-description>
   <webservice-description-name>
     CardValidatorWS
   </webservice-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>
 </webservice-description>
</webservices>
```

- XML schema per la validazione
- Nome del WS

46

## File XML per deployment: webservices.xml

File opzionale

```
<webservices xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/javaee_web_services_1_3.xsd"
 version="1.3">
 <webservice-description>
   <webservice-description-name>
     CardValidatorWS
   </webservice-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>
 </webservice-description>
</webservices>
```

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

47

## File XML per deployment:

File opzionale

webservices.xml


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

48

## File XML per deployment:

File opzionale

webservices.xml


- XML schema per la validazione
- Nome del WS
- Informazioni sulla porta
- Interfaccia
- Servlet di implementazione

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

50

## 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

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

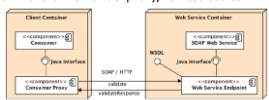



Figure 14.4. A consumer invoking a web service through a proxy

52

## 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 poducer

53

## 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)

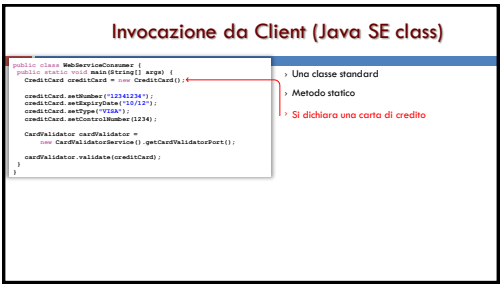
```
public class WebServiceConsumer {
  public static void main(String[] args) {
    CreditCard creditCard = new CreditCard();

    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

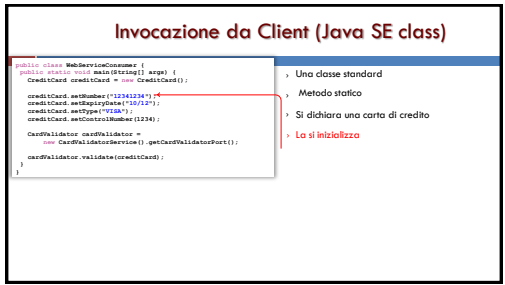
55

## Invocazione da Client (Java SE class)



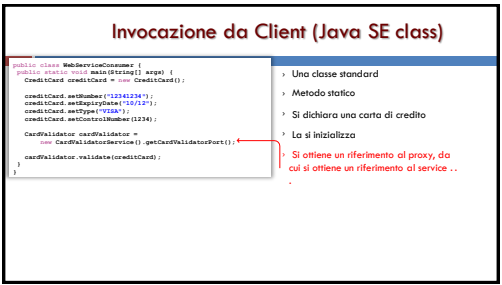

56

## Invocazione da Client (Java SE class)




57

## Invocazione da Client (Java SE class)

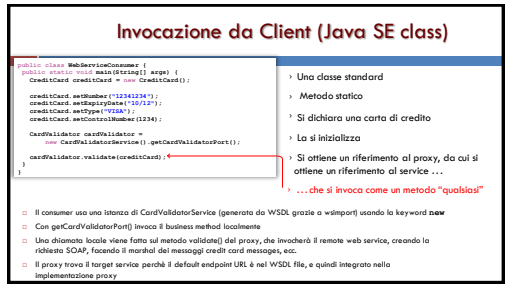


```
public class WebServiceConsumer {
  public static void main(String[] args) {
    CreditCard creditCard = new CreditCard();

    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)



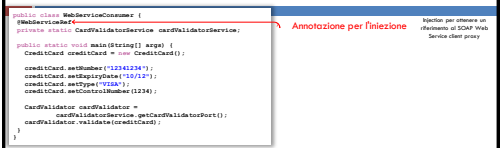
```
public class WebServiceConsumer {
  public static void main(String[] args) {
    CreditCard creditCard = new CreditCard();

    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 perché il default endpoint URL è nel WSDL file, e quindi integrato nella implementazione proxy

59

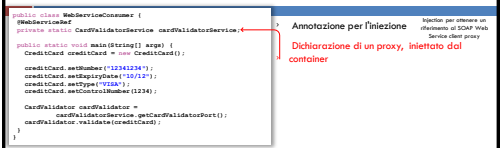
## Invocazione da un Container (invoking with Injection)




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

60

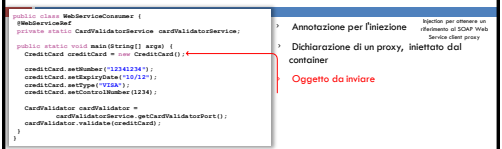
## Invocazione da un Container (invoking with Injection)




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

61

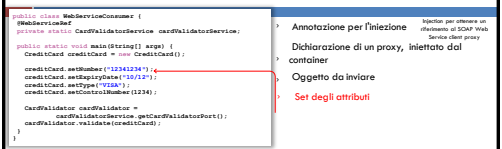
## Invocazione da un Container (invoking with Injection)




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

62

## Invocazione da un Container (invoking with Injection)




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

63

## 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
- Oggetto da inviare
- Set degli attributi
- Uso del proxy iniettato

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

64

## 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
- Oggetto da inviare
- Set degli attributi
- Uso del proxy iniettato
- …per invocare il WS

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

65

## 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

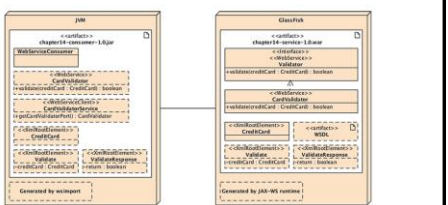
## Diagramma dell'esempio

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, falso 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

## CreditCard: La classe per la carta di credito

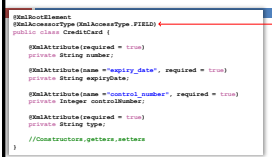
```
@XmlRootElement
@XmlAccessorType (XmlAccessType.FIELD)
public class CreditCard {

    @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



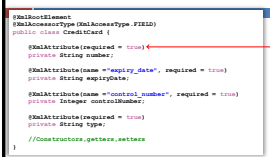

Inizio dell'elemento root dell'XML

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

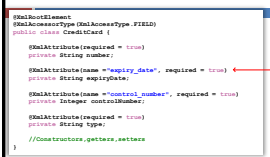



- Inizio dell'elemento root dell'XML
- Tutti i campi saranno mappati su XML
- Attributo obbligatorio

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

71

## CreditCard: La classe per la carta di credito



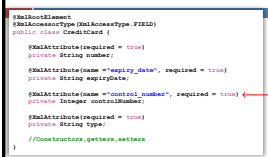
```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class CreditCard {

    @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

74

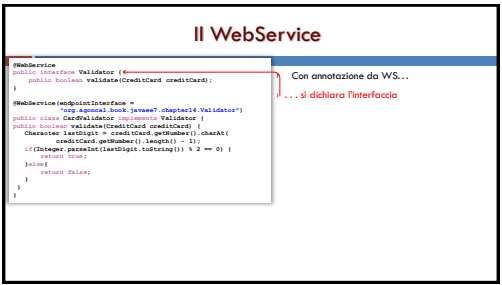
## Il WebService

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

@WebService(endpointInterface =
                "org.agnosci.book.jvmee7.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…

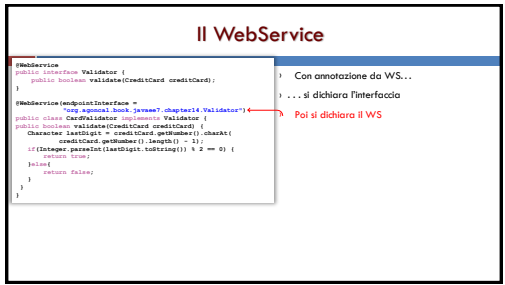
75

## Il WebService



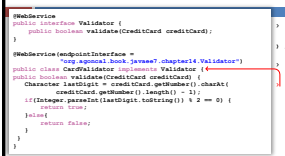

76

## Il WebService




77

## Il WebService

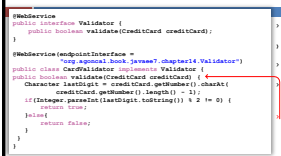


```
@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@WebService(endpointInterface =
            "org.agoncal.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
- Con la classe che implementa l'interfaccia

78

## Il WebService



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

@WebService(endpointInterface =
            "org.agoncal.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
- Con la classe che implementa l'interfaccia
- Metodo offerto come WS

79

## Organizzazione della lezione

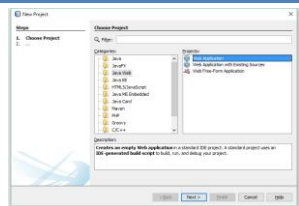
80

- 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

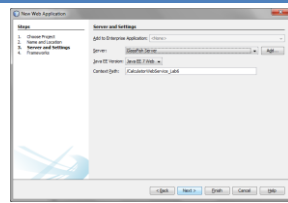
## WS in NetBeans: creazione di un progetto

82

82

## WS in NetBeans: creazione di un progetto

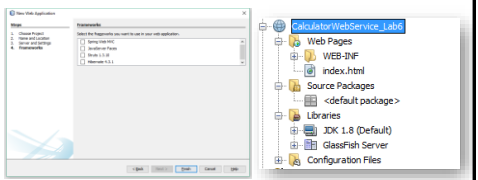
83



83

## WS in NetBeans: creazione di un progetto

84

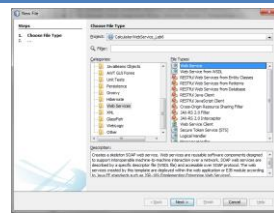



84

## WS in NetBeans: creazione di un progetto

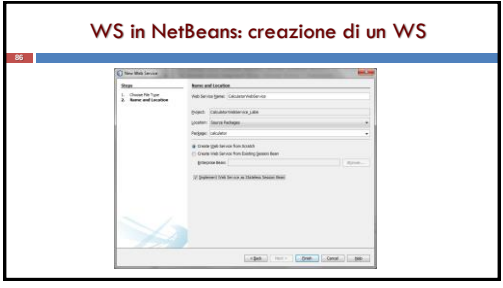
85

Right-click sul progetto → New



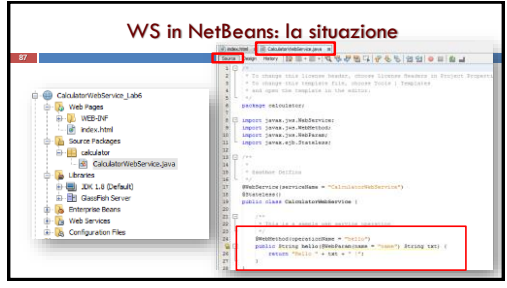
85

## WS in NetBeans: creazione di un WS



86

## WS in NetBeans: la situazione

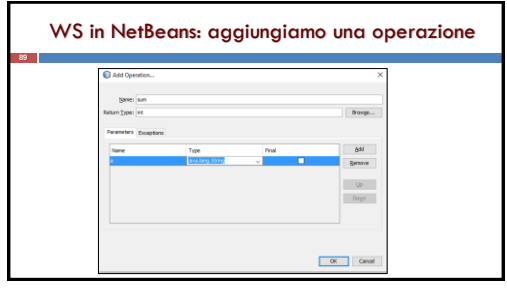



87

## WS in NetBeans: la vista di DESIGN


88

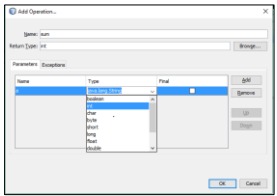
## WS in NetBeans: aggiungiamo una operazione




89

## WS in NetBeans: aggiungiamo un parametro

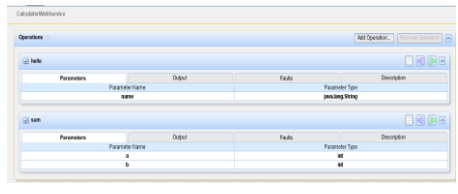
90




90

## WS in NetBeans: dopo aver aggiunto un secondo parametro

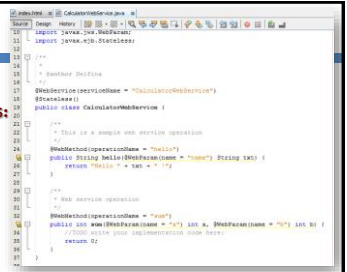
91




91

92

## WS in NetBeans: il codice




92

93

## WS in NetBeans

93

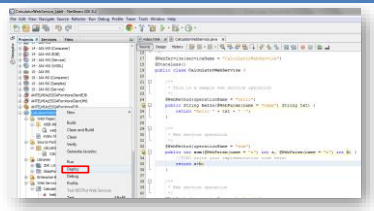
### WS in NetBeans: scriviamo qualcosa (implementiamo somma e moltiplicazione)

```
@WebMethod(operationName = "sum")
public int sum(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {
    //TODO write your implementation code here:
    return a+b;
}

/**
 * Web service operation
 */
@WebMethod(operationName = "multiply")
public int multiply(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {
    //TODO write your implementation code here:
    return 0;
}
}
```

93

## WS in NetBeans: deploy

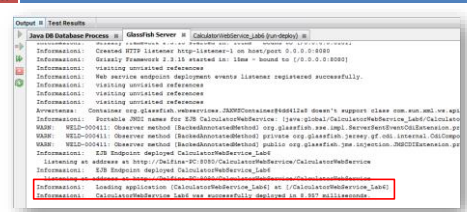
94



94

## WS in NetBeans: sul server

95




95

## Organizzazione della lezione

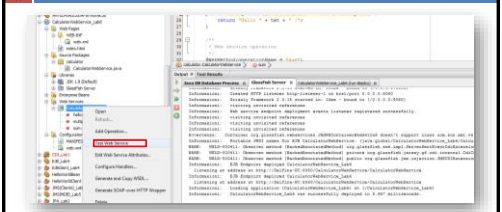
96

- 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

96

## WS in NetBeans: testing

97




97

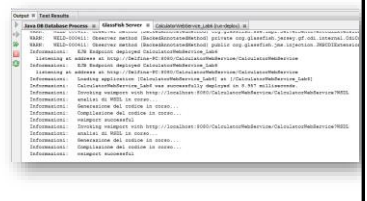
98

WS in NetBeans: Test


99

WS in NetBeans: testing

Cosa è successo sul server



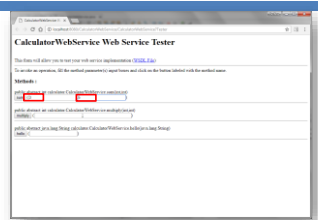
## WS in NetBeans: Test

100


100

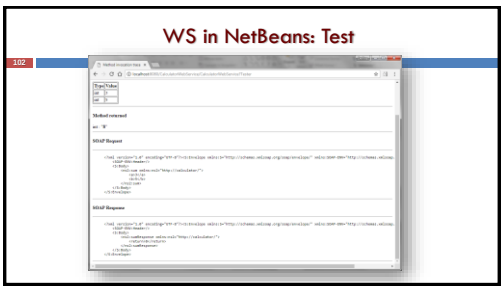
## WS in NetBeans: Test

101

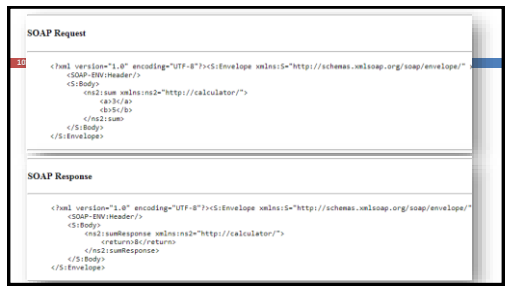



101

## WS in NetBeans: Test



102




103

## Organizzazione della lezione

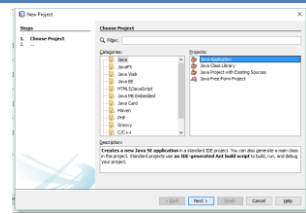
104

- 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

104

## WS in NetBeans: Client

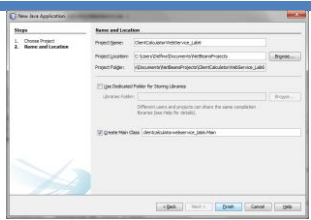
105




105

## WS in NetBeans: Client

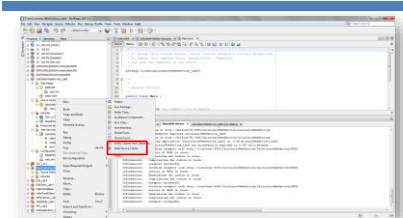
106




106

## WS in NetBeans: creiamo un WS Client

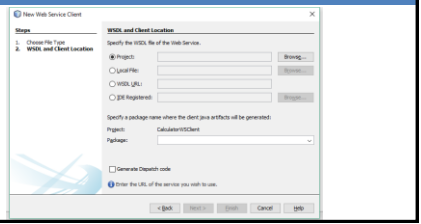
107



107

## WS in NetBeans: creiamo un WS Client

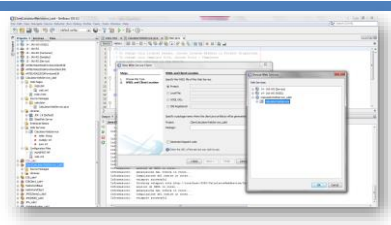
108




108

## WS in NetBeans: creiamo un WS Client

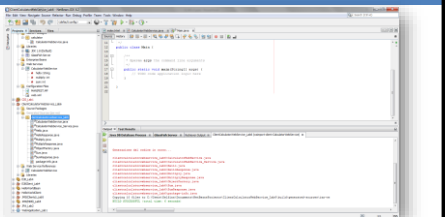
109



109

## La situazione attuale

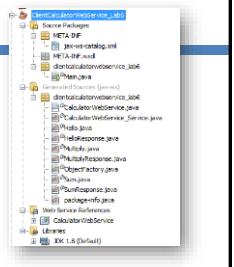
110



110

## WS in NetBeans: i files generati

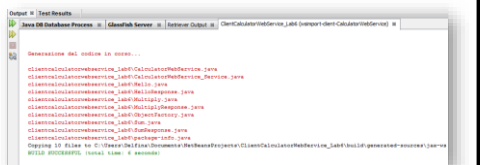
111




111

## WS in NetBeans: Output

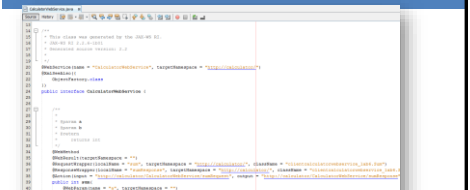
112




112

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

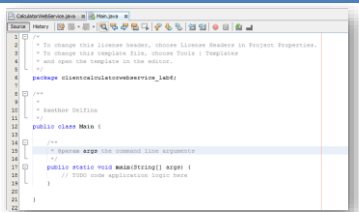
113



113

## WS in NetBeans: il Client (VUOTO)

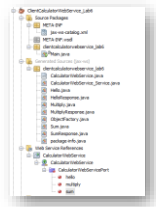
114




114

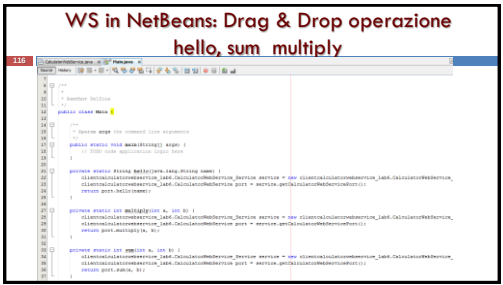
## Dal progetto WS…

115



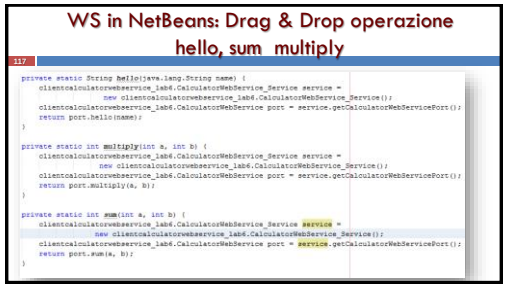

115

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



116

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

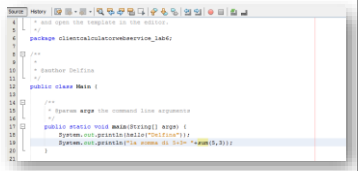


117

## WS in NetBeans: esecuzione

118

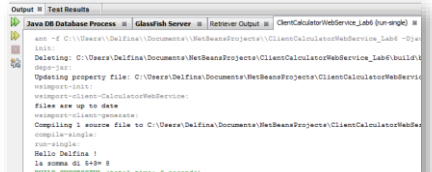
WS in NetBeans:
… un po' di codice




118

## WS in NetBeans: RUN

119

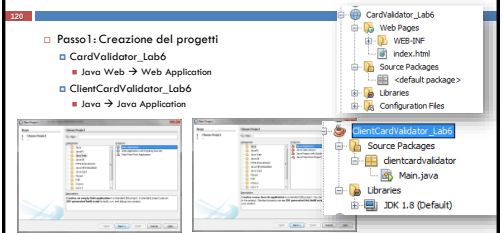



119

## CreditCardValidator Web Service

120

- Passo1: Creazione del progetti
  - CardValidator_Lab6
    - Java Web → Web Application
  - ClientCardValidator_Lab6
    - Java → Java Application




120

## CreditCardValidator Web Service

121

- 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

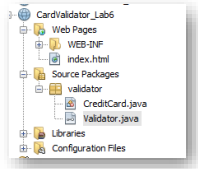
```
@WebService
public interface Validator {

  public boolean validate(CreditCard creditCard);
}
```

122

## La situazione attuale

123

- In NetBeans




123

## CreditCardValidator Web Service

124

- La classe CardValidator

*Listing 14-36.* The CardValidator Web Service Bean

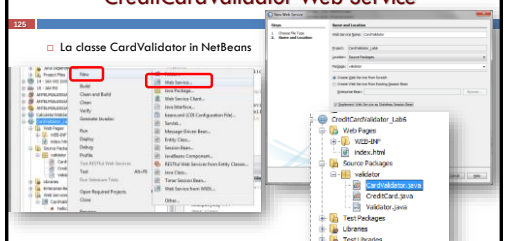
```
@WebService(endpointInterface = "org.agoncal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {

  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

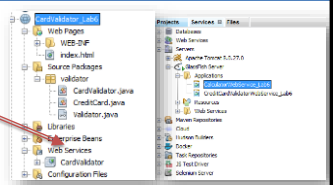



125

## CreditCardValidator Web Service

126

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

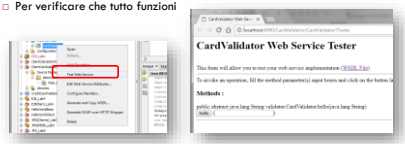



126

## CreditCardValidator Web Service

127

- Per verificare che tutto funzioni



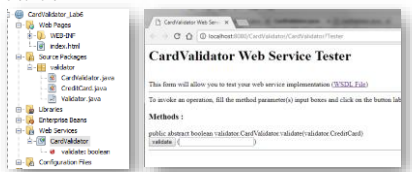

- E' presente il Web Service di default!
- Rimuoviamo hello(), ed aggiungiamo validate()

127

## CreditCardValidator Web Service

128

- Nuova situazione



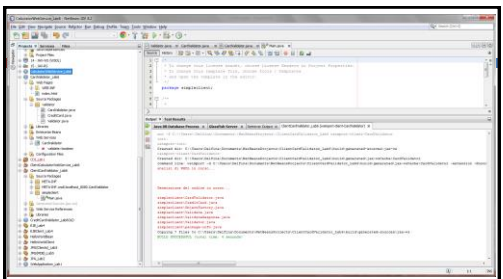

128

## CreditCardValidator Web Service

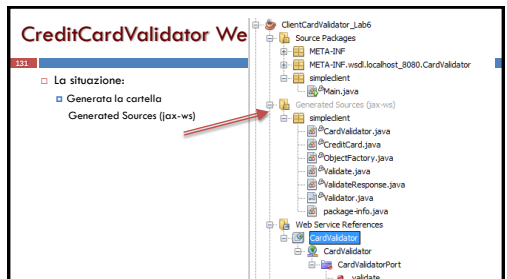
129

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

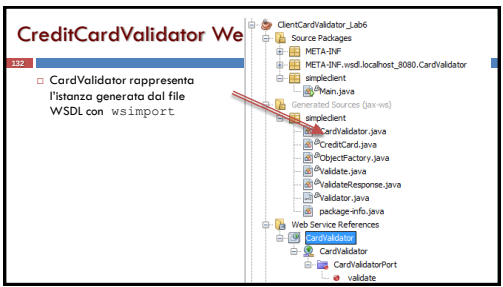
129



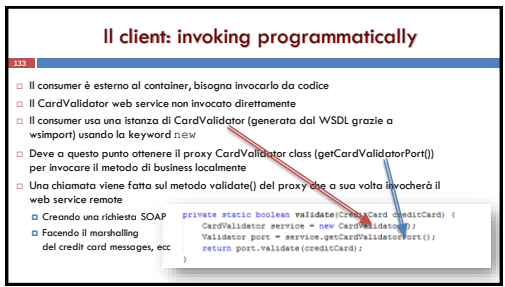
130


CreditCardValidator We
131
La situazione:
Generata la cartella
Generated Sources (jax-ws)
ClientCardValidator_Lab6
Source Packages
META-INF
META-INF.wsdl.localhost_8080.CardValidator
simplecient
Main.java
Generated Sources (jax-ws)
simplecient
CardValidator.java
CreditCard.java
ObjectFactory.java
Validate.java
ValidateResponse.java
Validator.java
package-info.java
Web Service References
CardValidator
CardValidator
CardValidatorPort
validate

131


CreditCardValidator We
132
CardValidator rappresenta l'istanza generata dal file WSDL con wsimport
ClientCardValidator_Lab6
Source Packages
META-INF
META-INF.wsdl.localhost_8080.CardValidator
simpleclient
Main.java
Generated Sources (jax-ws)
simpleclient
CardValidator.java
CreditCard.java
ObjectFactory.java
Validate.java
ValidateResponse.java
Validator.java
package-info.java
Web Service References
CardValidator
CardValidator
CardValidatorPort
validate

132

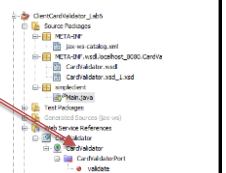

Il client: invoking programmatically
133
Il consumer è esterno al container, bisogna invocarlo da codice
Il CardValidator web service non invocato direttamente
Il consumer usa una istanza di CardValidator (generata dal WSDL grazie a wsimport) usando la keyword new
Deve a questo punto ottenere il proxy 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

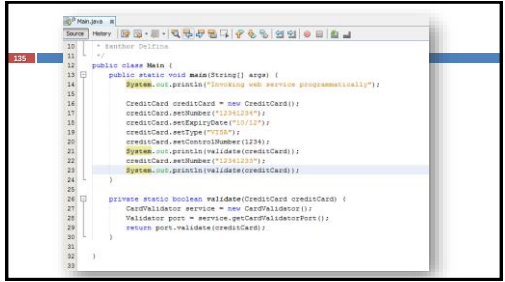
## Il client: invoking programmatically

134

- Drag & Drop da qui nella classe del client



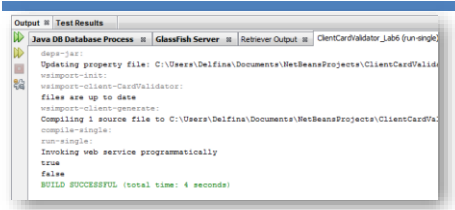

134




135

## Il client: invoking programmatically

136




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