



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



Service-Oriented Architecture & SOAP Web Services (2)

Delfina Malandrino

dmalandrino@unisa.it

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

Organizzazione della lezione

- 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

Organizzazione della lezione

- 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

Vita facile per il programmatore Java

- 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

Vita facile per il programmatore Java

- 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

Esempi di WSDL Mapping: @WebService

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

Esempi di WSDL mapping: @WebService

```

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

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

```

Una classe
.... o interfaccia che è un WS

Esempi di WSDL mapping: @WebService

- 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

```

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

```

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 "";
}

```

Possibile cambiare la porta e il servizio

Esempi di WSDL

- Versione originale

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

- Dopo la modifica

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

Esempi di WSDL mapping: @WebService

- 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

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

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)

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

- **Document:** Il messaggio SOAP contiene il documento
 - Inviato nel <soap:Body> element senza regole di formattazione addizionali
 - Preferibile quando i messaggi SOAP rappresentano dati complessi (es. ordini, contratti)
 - E' la scelta di default

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Order xmlns="http://example.com/order">
      <OrderID>12345</OrderID>
      <Customer>John Doe</Customer>
    </Order>
  </soap:Body>
</soap:Envelope>
```

SOAP binding

- **RPC:** Il messaggio SOAP contiene i parametri ed i valori restituiti
 - Il <soap:Body> contiene un elemento con il nome del metodo remoto o procedura da invocare
 - E' presente un elemento per ogni parametro della procedura remota
 - Preferibile quando si devono invocare operazioni semplici

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <ns2:add xmlns:ns2="http://example.com/math">
    <a>10</a>
    <b>20</b>
  </ns2:add>
</soap:Body>
</soap:Envelope>
```

SOAP binding

- Un SOAP binding (Document or RPC) deve scegliere fra due differenti formati di serialization/deserialization: **Literal ed Encoding**
 - Modi in cui i dati contenuti nei messaggi SOAP vengono:
 - serializzati - conversione in XML
 - deserializzati - ricostruzione degli oggetti dai dati XML
- Questi formati influenzano il modo in cui il contenuto del messaggio è strutturato e interpretato

SOAP binding

- 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

SOAP binding

- **Literal:** Data serializzati secondo un XML schema
 - Il formato Literal è maggiormente interoperabile rispetto a Encoded, poiché i dati sono conformi a standard XML ben definiti
- Esempio: supponiamo che il messaggio SOAP scambi informazioni su un ordine, con uno schema che definisce **<Order>** con elementi **<OrderID>** e **<Customer>**
- Il messaggio in formato Literal sarà:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Order xmlns="http://example.com/order">
      <OrderID>12345</OrderID>
      <Customer>John Doe</Customer>
    </Order>
  </soap:Body>
</soap:Envelope>

```

SOAP binding

- **Encoded:** Il formato Encoded utilizza le regole di codifica definite da SOAP (SOAP Encoding Rules)
- I dati sono serializzati secondo un insieme di regole specifiche per rappresentare oggetti complessi, riferimenti e collegamenti tra i dati.
- Esempio: Per un oggetto complesso che rappresenta un ordine e include un riferimento a un cliente, il messaggio in formato Encoded potrebbe essere:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Order xmlns="http://example.com/order" xsi:type="ns:OrderType" xmlns:ns="http://example.com/order">
      <OrderID xsi:type="xsd:int">12345</OrderID>
      <Customer href="#cust1" />
    </Order>
    <Customer id="cust1" xsi:type="ns:CustomerType">
      <Name xsi:type="xsd:string">John Doe</Name>
    </Customer>
  </soap:Body>
</soap:Envelope>
```

SOAP binding

```

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

Esempio:

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

SOAP binding

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"/>
```

Organizzazione della lezione

- 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

- 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...

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

II WS CardValidator

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

> Annotazione che denota un WS
 > Nome del POJO

II WS CardValidator

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

> Annotazione che denota un WS
 > Nome del POJO
 > 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

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

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
- › Con il tipo inviato dal codice Java

Vediamo un altro esempio...

Viene lanciata 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

Viene lanciata 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 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

Viene lanciata una eccezione specifica

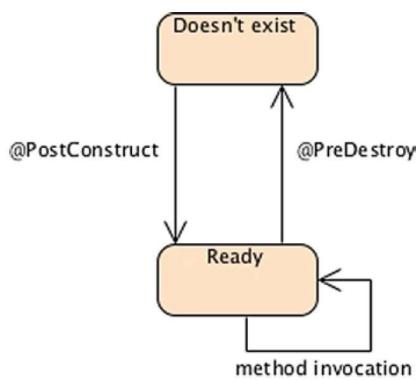
Listing 14-24. SOAP Fault in a SOAP Envelope

```
<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

- 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



- 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

I contesti di un WS

- Un SOAP web service ha un environment context e può accedervi iniettando una reference di javax.xml.ws.WebServiceContext con l'annotazione @Resource annotation
- All'interno di questo contest, il web service può ottenere runtime information
 - l'endpoint implementation class, il message context, security information circa la richiesta appena servita, ecc.

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

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

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

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

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

Contesti di un WS: i metodi disponibili

Method	Description
getMessageContext	Returns the MessageContext for the request being served at the time this method is called. It can be used to access the SOAP message headers, body, and so on.
getUserPrincipal	Returns the Principal that identifies the sender of the request currently being serviced.
isUserInRole	Returns a Boolean indicating whether the authenticated user is included in the specified logical role.
getEndpointReference	Returns the EndpointReference associated with this endpoint.

Deployment Descriptor

- 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 opzionale

File XML per deployment: 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

File opzionale

File XML per deployment: 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

Nome del WS

File opzionale

File XML per deployment: 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
- > Nome del WS

Informazioni sulla porta

File opzionale

File XML per deployment: 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
- > Nome del WS

Informazioni sulla porta

Interfaccia

File opzionale

File XML per deployment: 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
- > Nome del WS
- > Informazioni sulla porta
- > Interfaccia
- > Servlet di implementazione

Invoking Web Services

- 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

- 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

Invoking Web Services

- 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

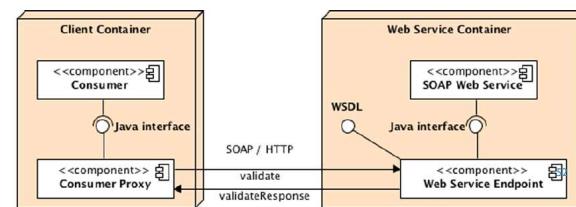


Figure 14-5. A consumer invoking a web service through a proxy

Anatomia di un SOAP Consumer

- 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

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

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

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

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
- ...

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

Invocazione da un container (invoking with injection)

```
public class WebServiceConsumer {
    @WebServiceRef ← Annotazione per l'iniezione
    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);
    }
}
```

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; <--> Annotazione per l'iniezione
                                                               Injection per ottenere un
                                                               riferimento al SOAP Web
                                                               Service client proxy

    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);
    }
}
```

Dichiarazione di un proxy, iniettato dal container

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; <--> Annotazione per l'iniezione
                                                               Injection per ottenere un
                                                               riferimento al SOAP Web
                                                               Service client proxy

    public static void main(String[] args) {
        CreditCard creditCard = new CreditCard(); <--> Dichiarazione di un proxy, iniettato dal
                                                               container

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

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

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

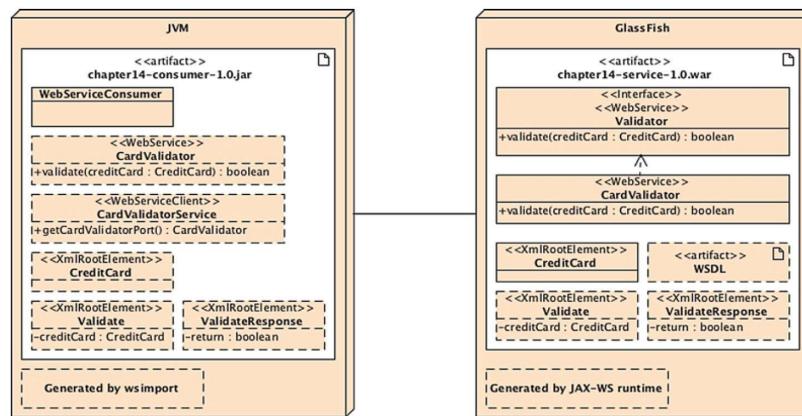
Organizzazione della lezione

- 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

Diagramma dell'esempio

- 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

Diagramma dell'esempio



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

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

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

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

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

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 Web Service

```
@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...

Il Web Service

```
@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

Il Web Service

```

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

@WebService(endpointInterface =
            "org.agoncal.book.javaee7.chapter14.Validator") ←
public class CardValidator implements Validator { → Poi si dichiara il WS
    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 Web Service

```

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

@WebService(endpointInterface =
            "org.agoncal.book.javaee7.chapter14.Validator") ←
public class CardValidator implements Validator { → Poi si dichiara il WS
    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
Con la classe che implementa l'interfaccia

Il Web Service

```

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

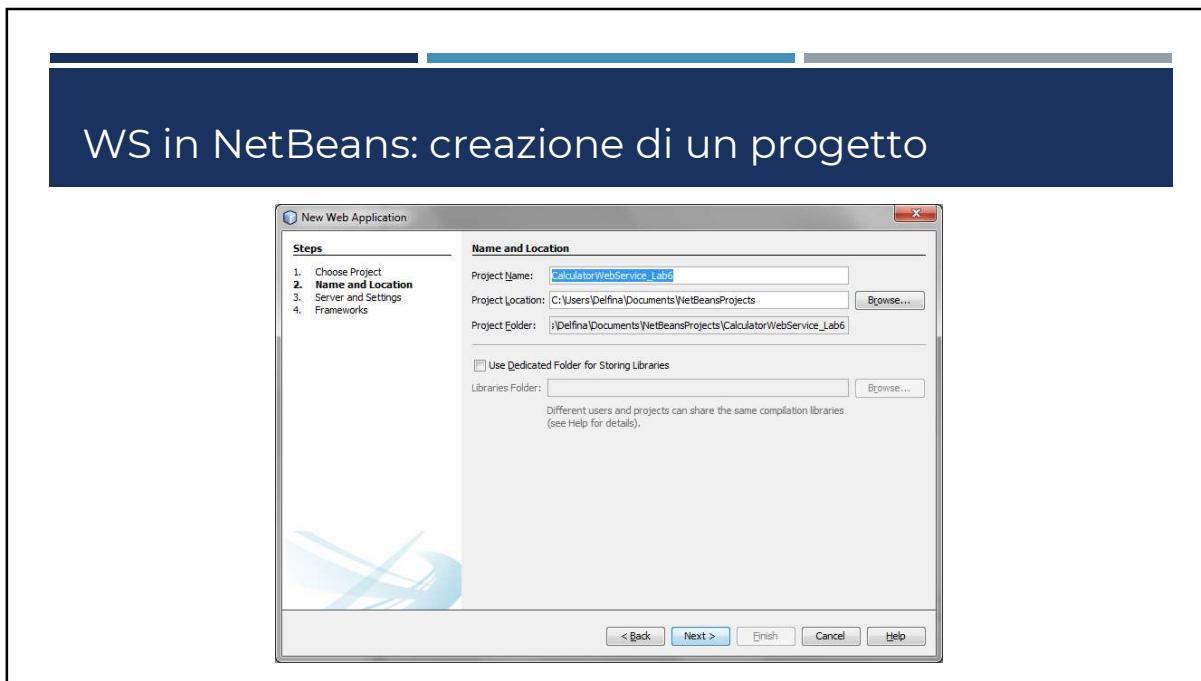
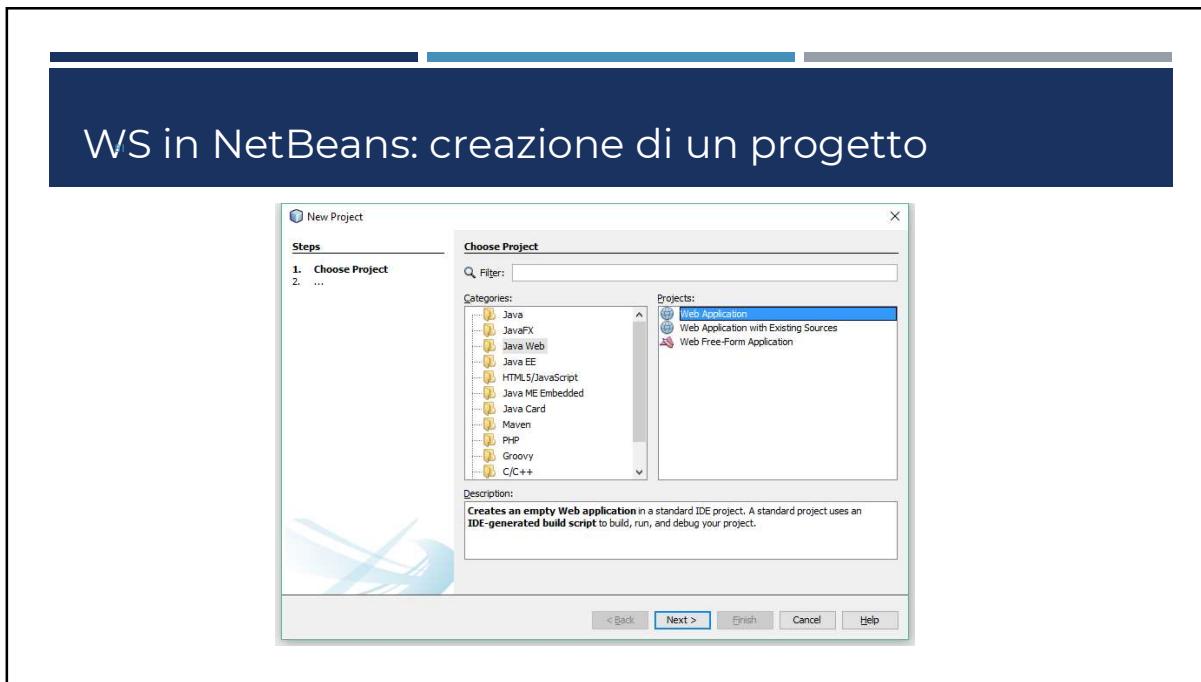
@WebService(endpointInterface =
            "org.agoncal.book.javaee7.chapter14.Validator")
public class CardValidator implements Validator {
    public boolean validate(CreditCard creditCard) { ←
        Character lastDigit = creditCard.getNumber().charAt(
            creditCard.getNumber().length()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

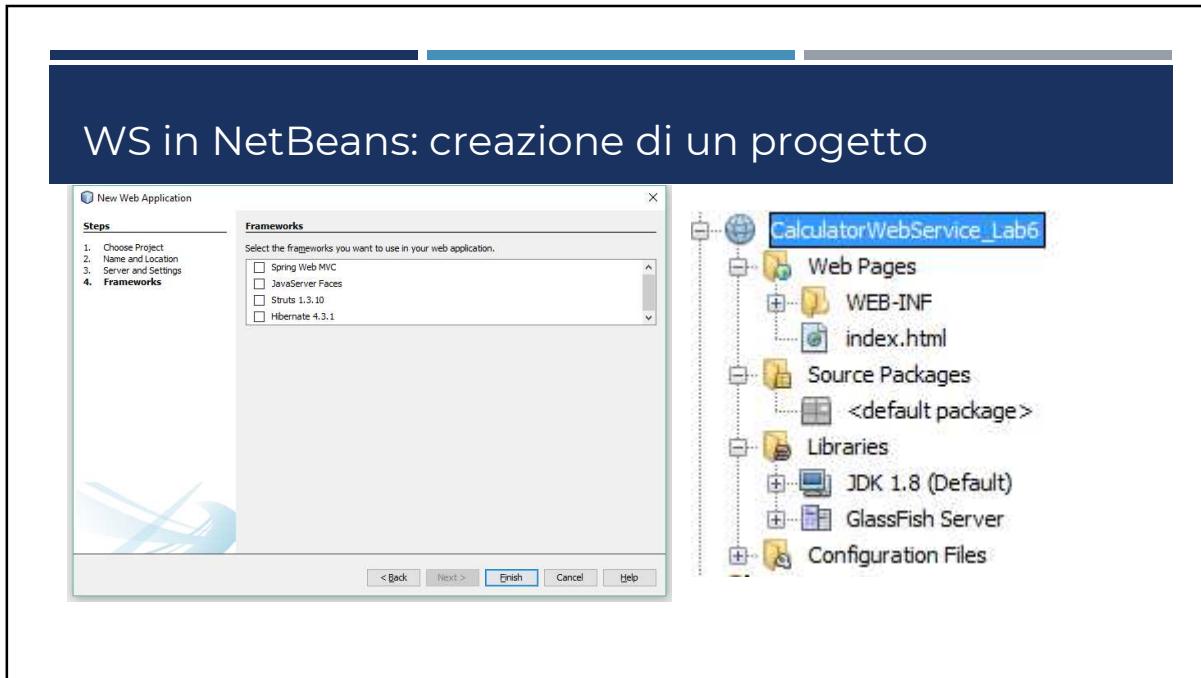
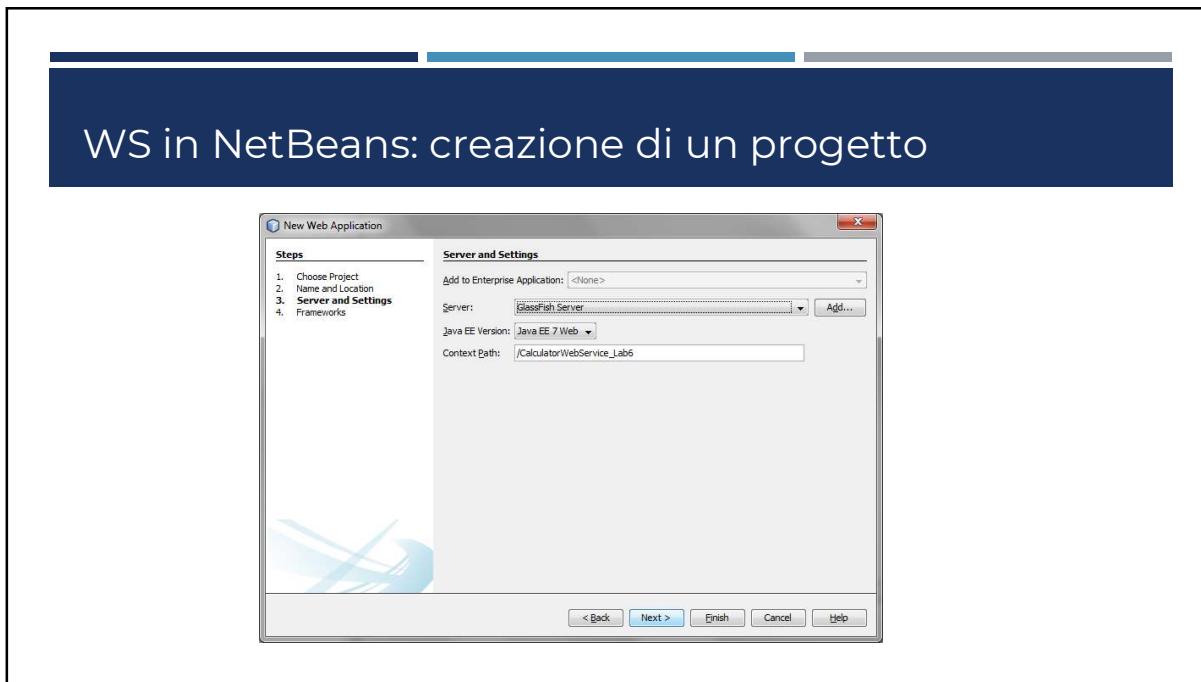
```

- › Con annotazione da WS...
- › ... si dichiara l'interfaccia
- › Poi si dichiara il WS
- Con la classe che implementa l'interfaccia
- › Metodo offerto come WS

Organizzazione della lezione

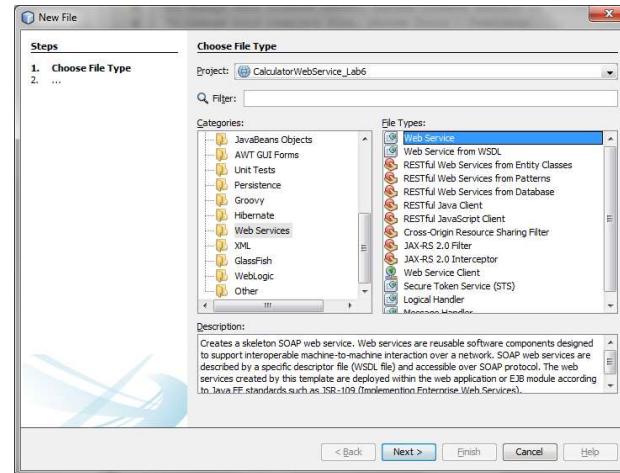
- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Putting It All Together
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni



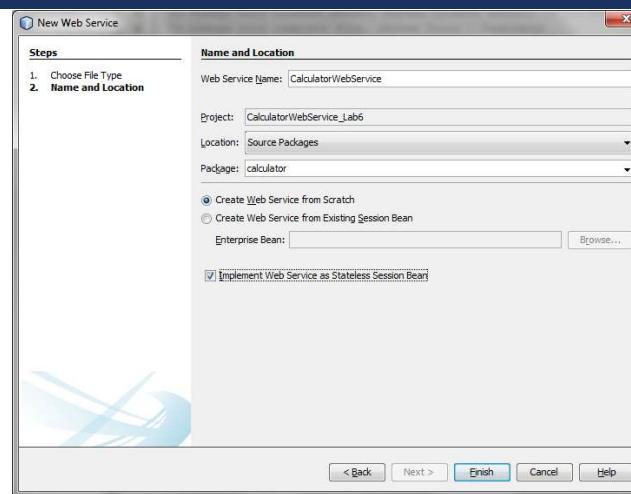


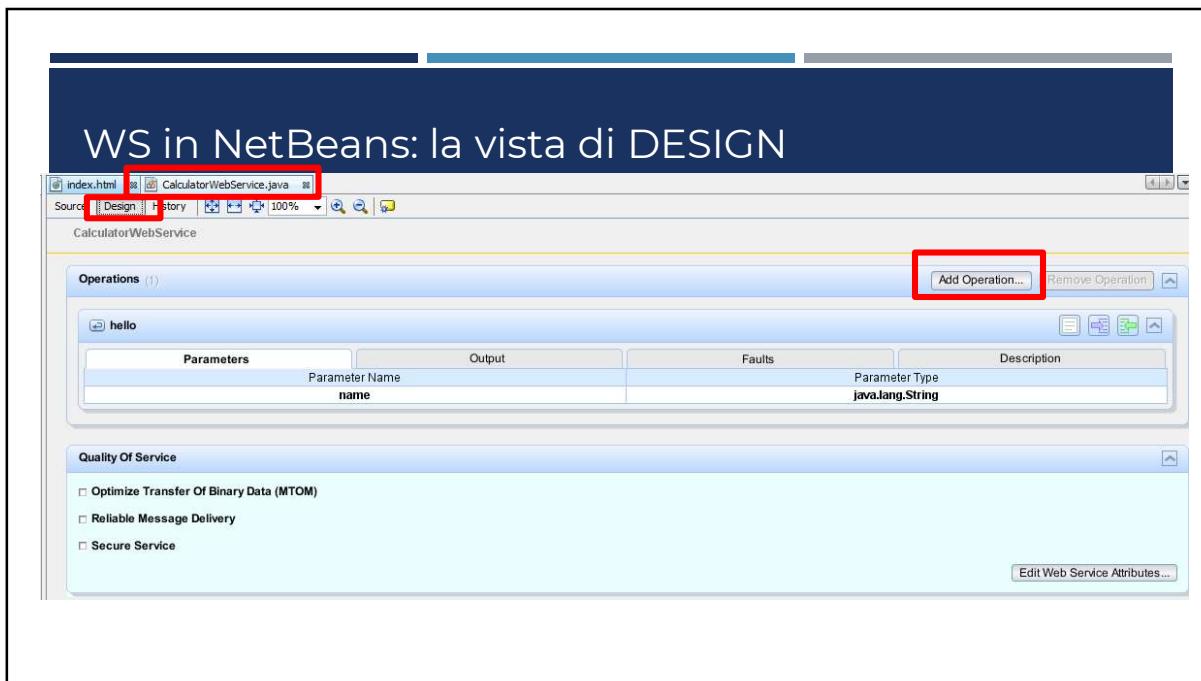
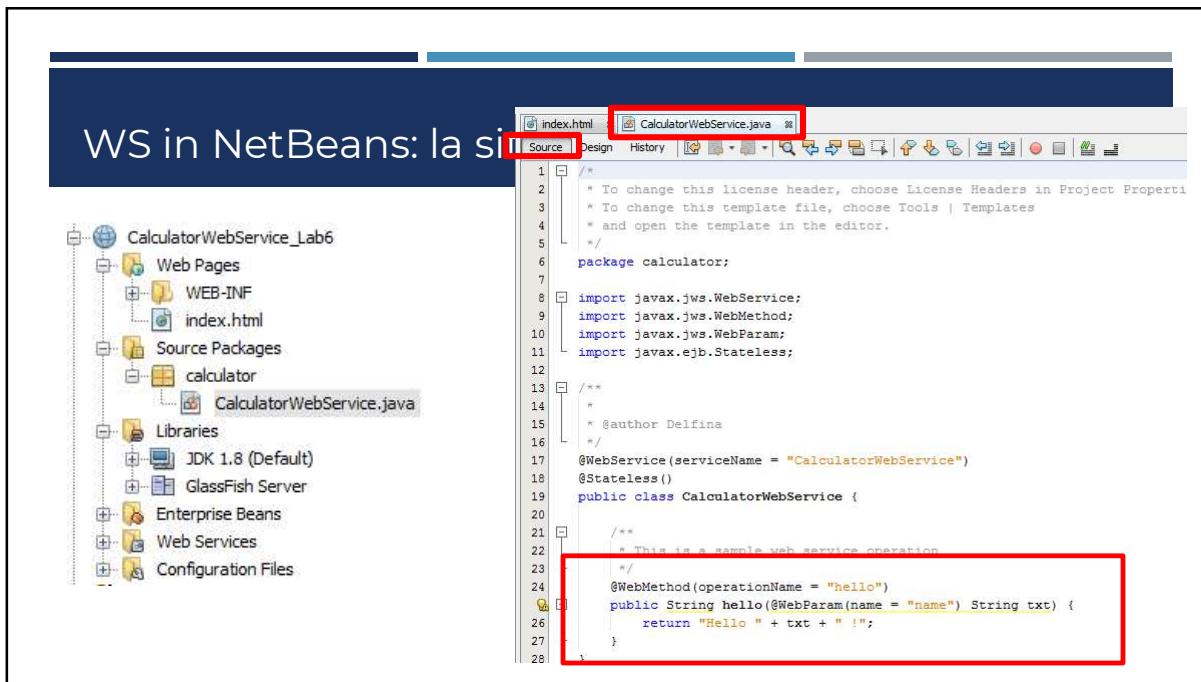
WS in NetBeans: creazione di un progetto

Right-click sul progetto →
New

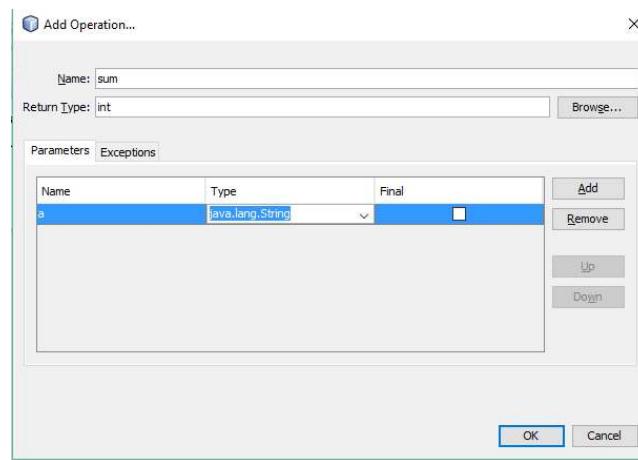


WS in NetBeans: creazione di un WS

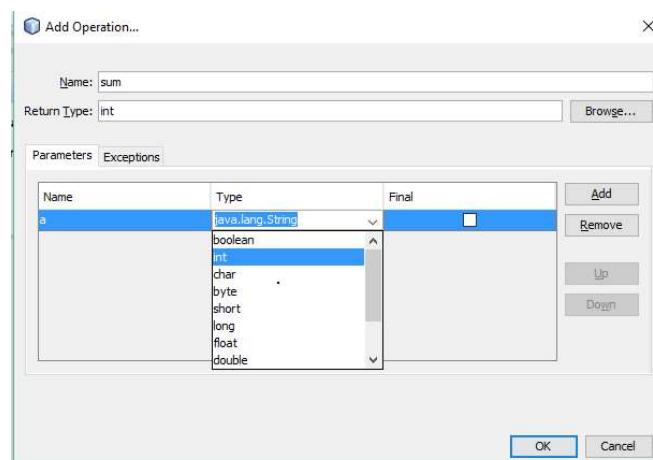




WS in NetBeans: aggiungiamo una operazione



WS in NetBeans: aggiungiamo un parametro



WS in NetBeans: dopo aver aggiunto un secondo parametro



WS in NetBeans: il codice

```

index.html   CalculatorWebService.java
Source Design History | 
10 import javax.jws.WebParam;
11 import javax.ejb.Stateless;
12
13 /**
14 * @author Delfina
15 */
16 @WebService(serviceName = "CalculatorWebService")
17 @Stateless()
18 public class CalculatorWebService {
19
20
21     /**
22      * This is a sample web service operation
23      */
24     @WebMethod(operationName = "hello")
25     public String hello(@WebParam(name = "name") String txt) {
26         return "Hello " + txt + " !";
27     }
28
29     /**
30      * Web service operation
31      */
32     @WebMethod(operationName = "sum")
33     public int sum(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {
34         //TODO write your implementation code here:
35         return 0;
36     }
37 }
```

WS in NetBeans

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

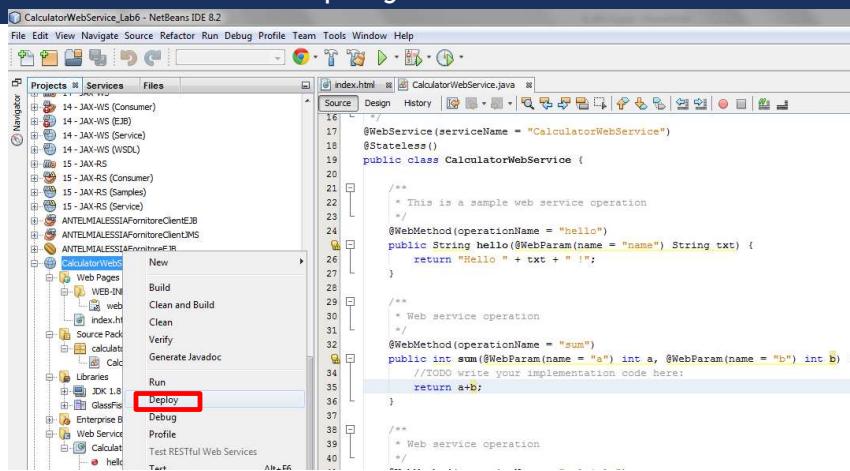
```

@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;
}

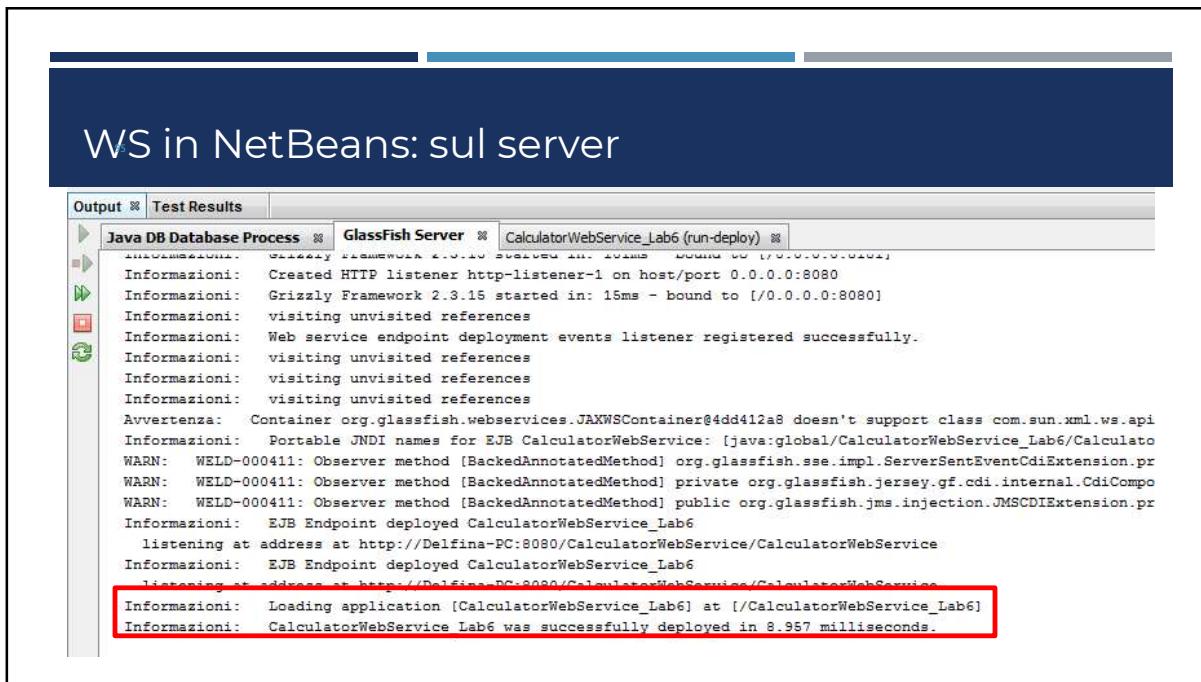
/**
 * Web service operation
 */
@WebMethod(operationName = "multiply")
public int multiply(@WebParam(name = "a") int a, @WebParam(name = "b") int b) {
    //TODO write your implementation code here:
    return 0;
}

```

WS in NetBeans: deploy



WS in NetBeans: sul server



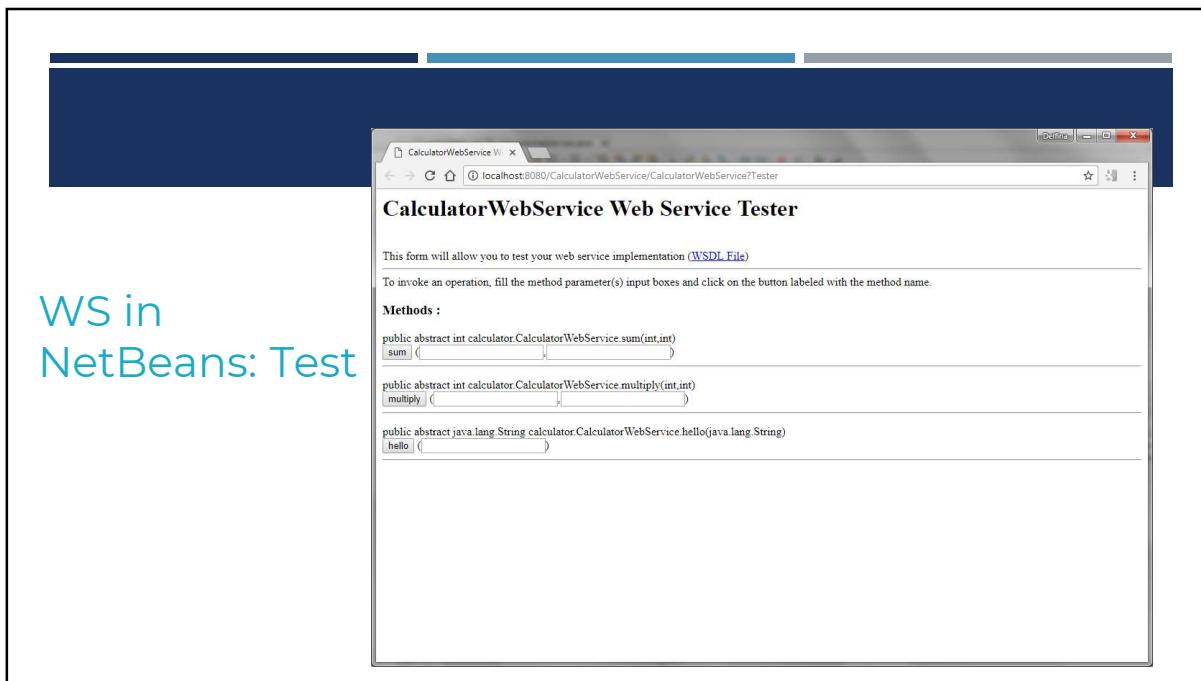
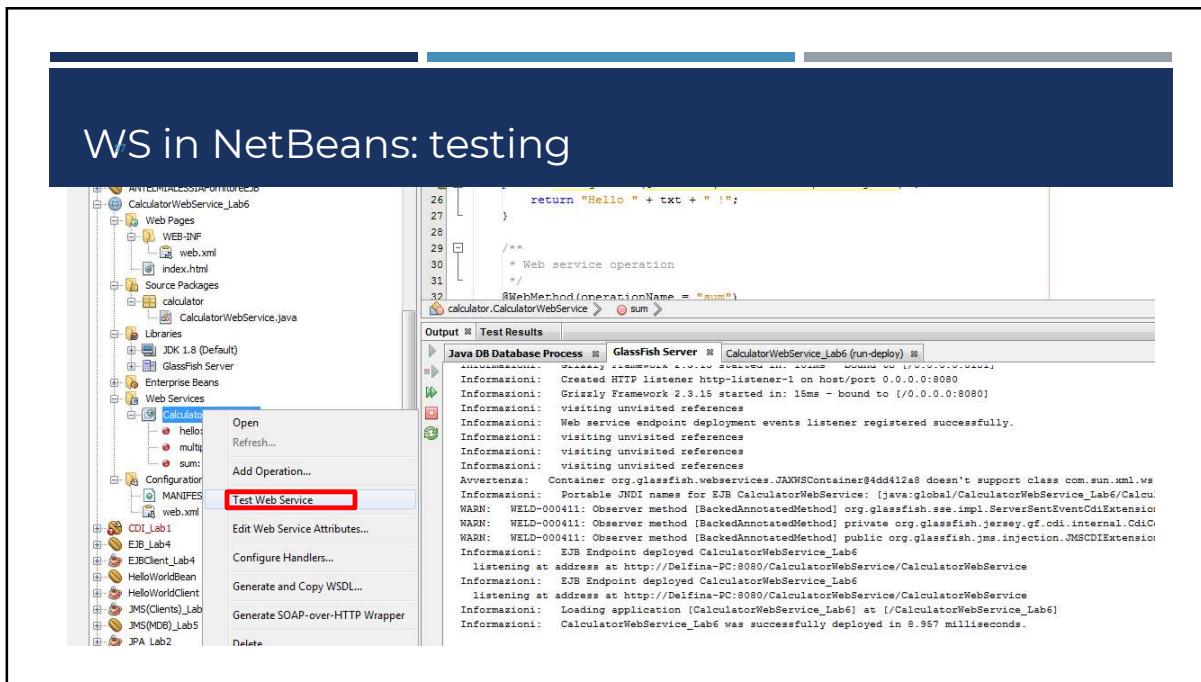
```

Output * Test Results
Java DB Database Process * GlassFish Server * CalculatorWebService_Lab6 (run-deploy) *
INFORMAZIONI: Grizzly Framework 2.3.15 started in: 10ms - bound to [/0.0.0.0:8080]
Informazioni: Created HTTP listener http-listener-1 on host/port 0.0.0.0:8080
Informazioni: Grizzly Framework 2.3.15 started in: 15ms - bound to [/0.0.0.0:8080]
Informazioni: visiting unvisited references
Informazioni: Web service endpoints deployment events listener registered successfully.
Informazioni: visiting unvisited references
Informazioni: visiting unvisited references
Informazioni: visiting unvisited references
Avvertenza: Container org.glassfish.webservices.JAXWSContainer@4dd412a8 doesn't support class com.sun.xml.ws.api
Informazioni: Portable JNDI names for EJB CalculatorWebService: [java:global/CalculatorWebService_Lab6/Calculato
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventCdiExtension.pr
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.gf.cdi.internal.CdiCompo
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSCDIExtension.pr
Informazioni: EJB Endpoint deployed CalculatorWebService_Lab6
listening at address at http://Delfina-PC:8080/CalculatorWebService/CalculatorWebService
Informazioni: EJB Endpoint deployed CalculatorWebService_Lab6
listening at address at http://Delfina-PC:8080/CalculatorWebService/CalculatorWebService
Informazioni: Loading application [CalculatorWebService_Lab6] at [/CalculatorWebService_Lab6]
Informazioni: CalculatorWebService Lab6 was successfully deployed in 8.957 milliseconds.

```

Organizzazione della lezione

- 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

Cosa è successo sul server

```

Output Test Results
Java DB Database Process GlassFish Server CalculatorWebService_Lab6 (run-deploy)
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.gf.cdi.internal.CdiC...
WARN: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSCDIExtensi...
Informazioni: EJB Endpoint deployed CalculatorWebService_Lab6
Informazioni: EJB Endpoint deployed CalculatorWebService_Lab6
listening at address at http://Delfina-PC:8080/CalculatorWebService/CalculatorWebService
Informazioni: Loading application [CalculatorWebService_Lab6] at [/CalculatorWebService_Lab6]
Informazioni: CalculatorWebService_Lab6 was successfully deployed in 8.357 milliseconds.
Informazioni: Invoking wsimport with http://localhost:8080/CalculatorWebService/CalculatorWebService?WSDL
Informazioni: analisi di WSDL in corso...
Informazioni: Generazione del codice in corso...
Informazioni: Compilazione del codice in corso...
Informazioni: wsimport successful
Informazioni: Invoking wsimport with http://localhost:8080/CalculatorWebService/CalculatorWebService?WSDL
Informazioni: analisi di WSDL in corso...
Informazioni: Generazione del codice in corso...
Informazioni: Compilazione del codice in corso...
Informazioni: wsimport successful

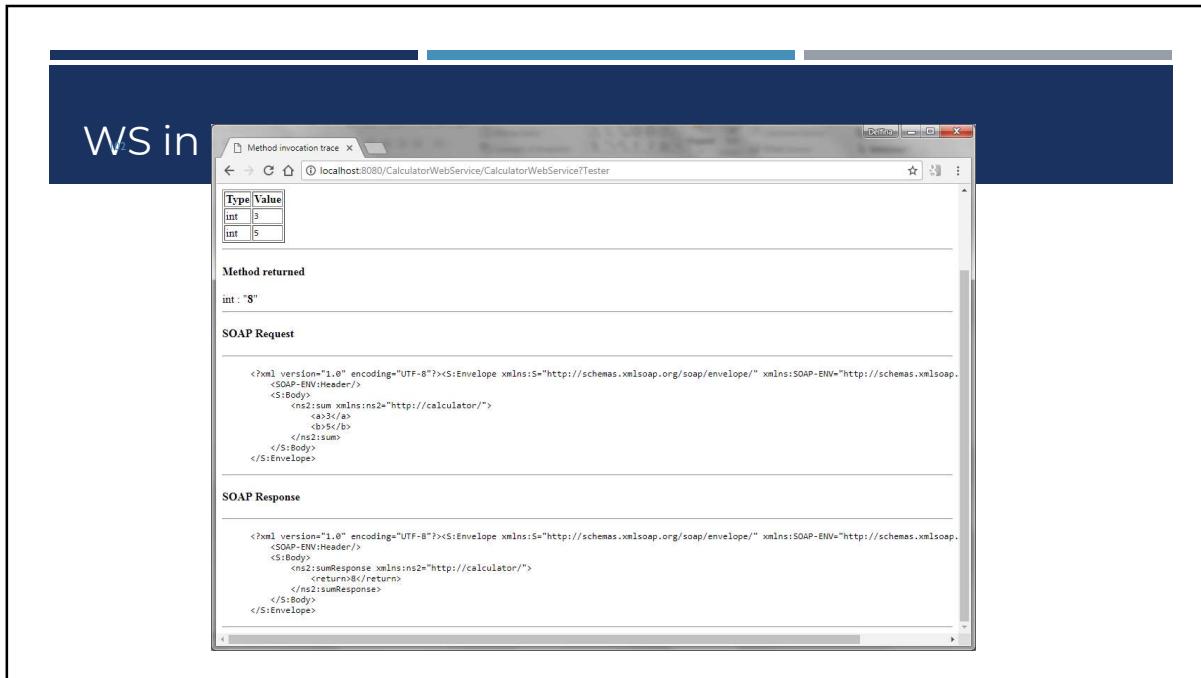
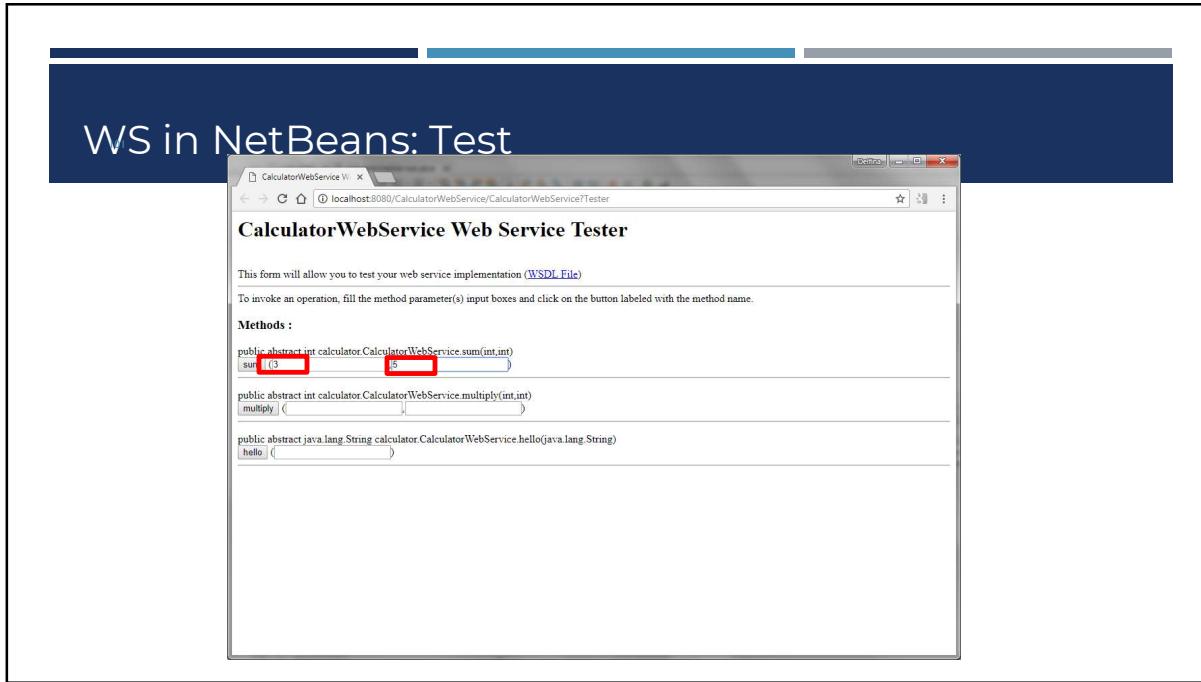
```

WS in NetBeans: testing

```

<!--
Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.1-b419 (branches/2.3.1.x-7937; 2014-08-04T08:11:03+0000) JAXWS-RI/2
-->
<!--
Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.1-b419 (branches/2.3.1.x-7937; 2014-08-04T08:11:03+0000) JAXWS-RI/2
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://calculator/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://calculator/" name="calculatorWebService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://calculator/" schemaLocation="http://localhost:8080/CalculatorWebService/CalculatorWebService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="sum">
    <part name="parameters" element="tns:sum"/>
  </message>
  <message name="sumResponse">
    <part name="parameters" element="tns:sumResponse"/>
  </message>
  <message name="multiply">
    <part name="parameters" element="tns:multiply"/>
  </message>
  <message name="multiplyResponse">
    <part name="parameters" element="tns:multiplyResponse"/>
  </message>
  <message name="hello">
    <part name="parameters" element="tns:hello"/>
  </message>
  <message name="helloResponse">
    <part name="parameters" element="tns:helloResponse"/>
  </message>
  <portType name="CalculatorWebService">
    <operation name="sum">
      <input wsam:Action="http://calculator/CalculatorWebService/sumRequest" message="tns:sum"/>
      <output wsam:Action="http://calculator/CalculatorWebService/sumResponse" message="tns:sumResponse"/>
    </operation>
    <operation name="multiply">
      <input wsam:Action="http://calculator/CalculatorWebService/multiplyRequest" message="tns:multiply"/>
      <output wsam:Action="http://calculator/CalculatorWebService/multiplyResponse" message="tns:multiplyResponse"/>
    </operation>
  </portType>

```



SOAP Request

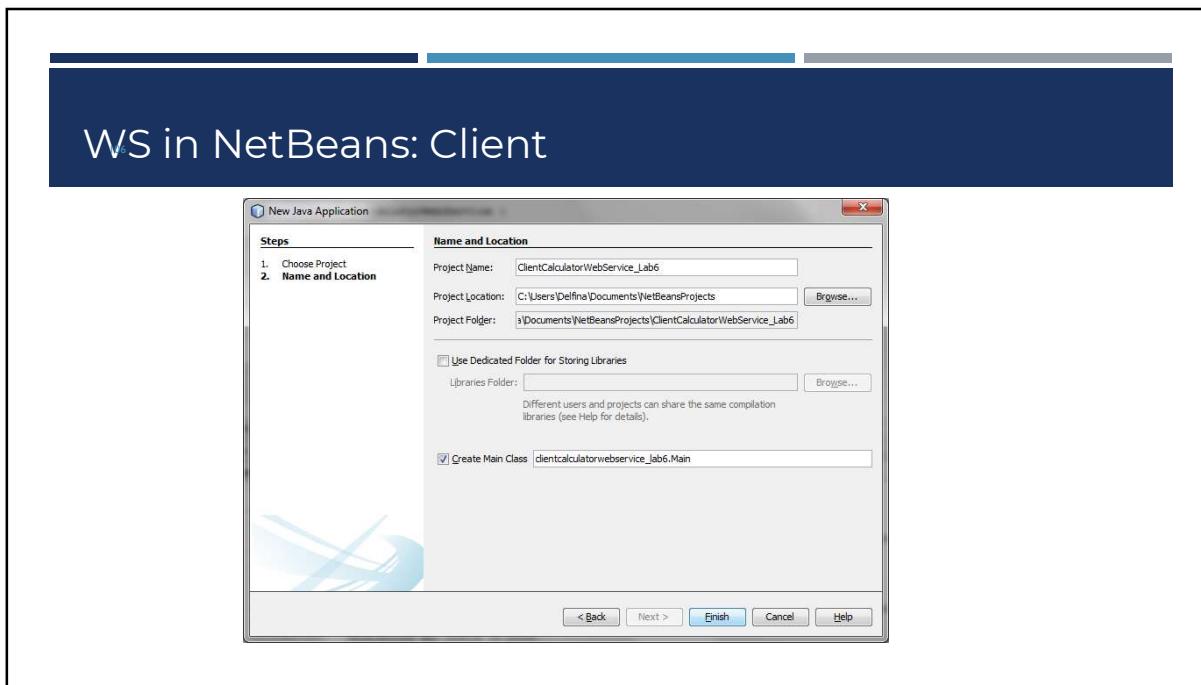
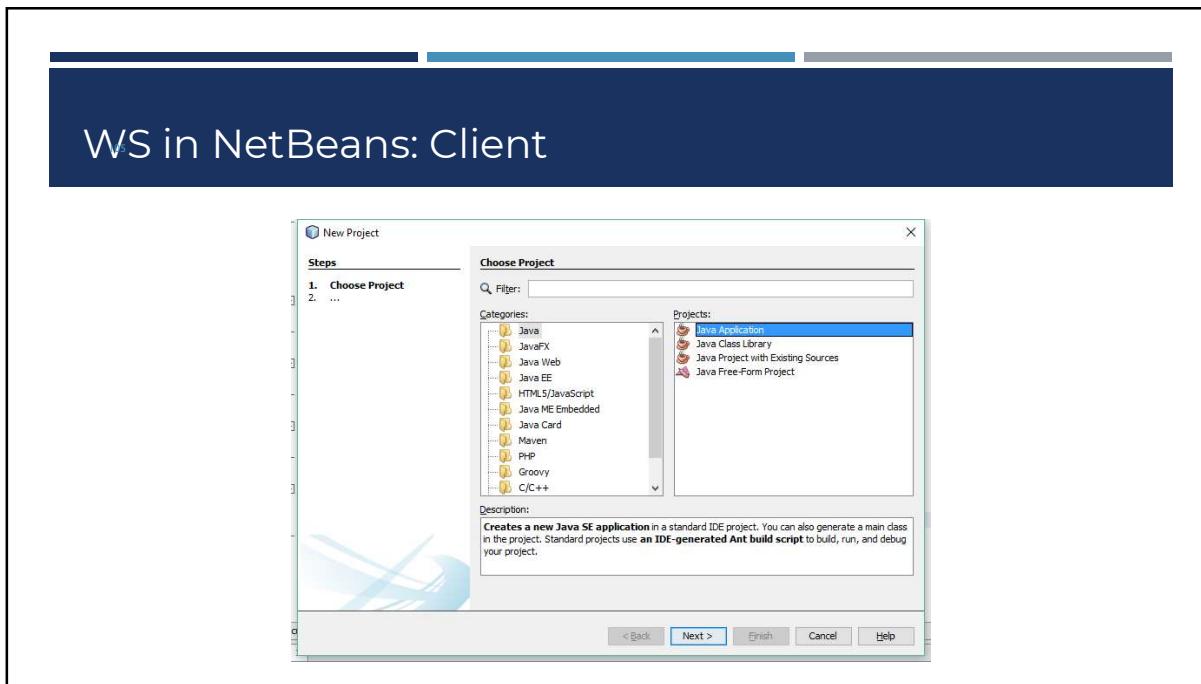
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sum xmlns:ns2="http://calculator/">
      <a>3</a>
      <b>5</b>
    </ns2:sum>
  </S:Body>
</S:Envelope>
```

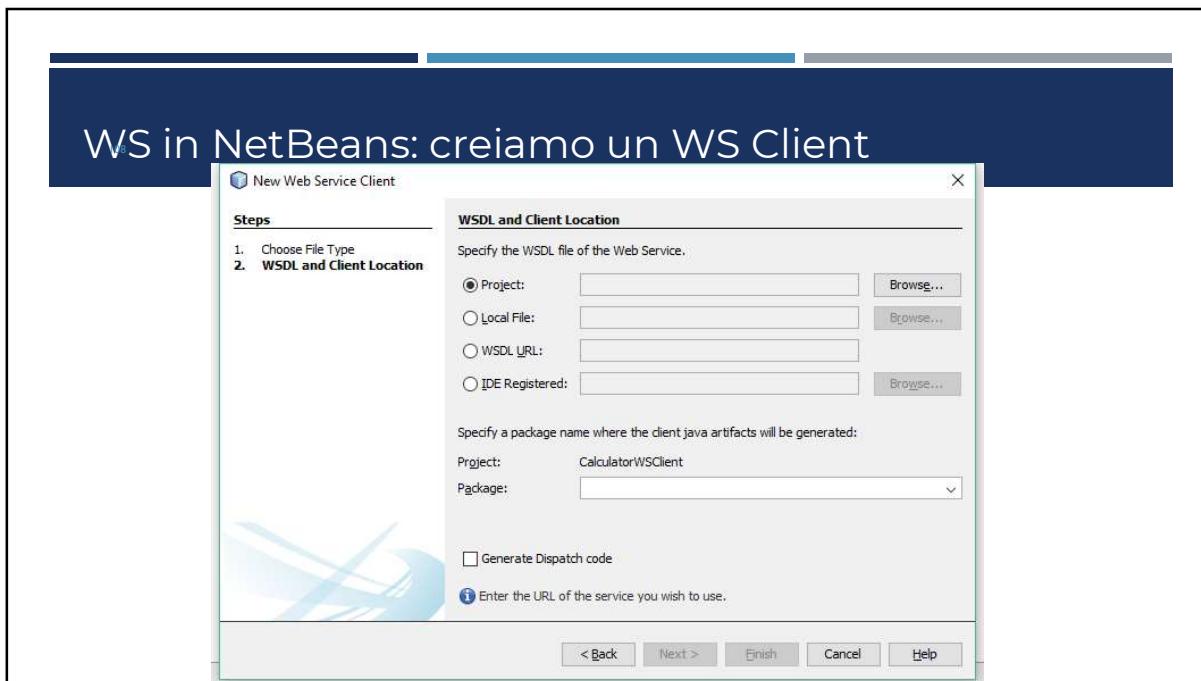
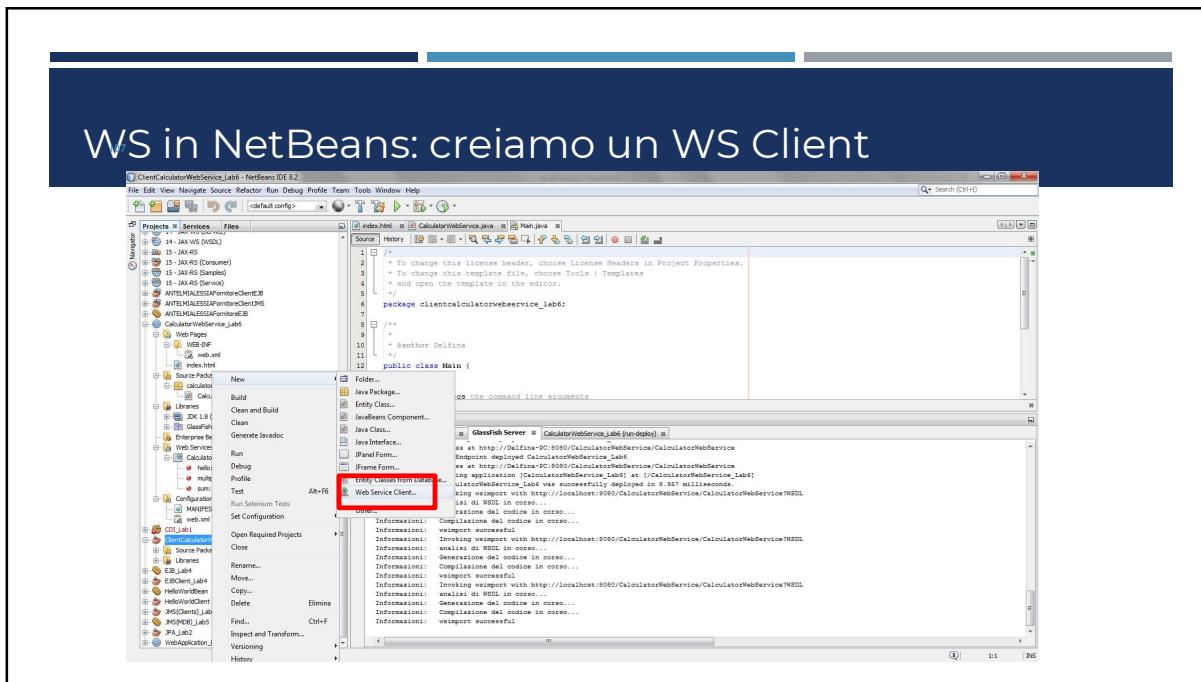
SOAP Response

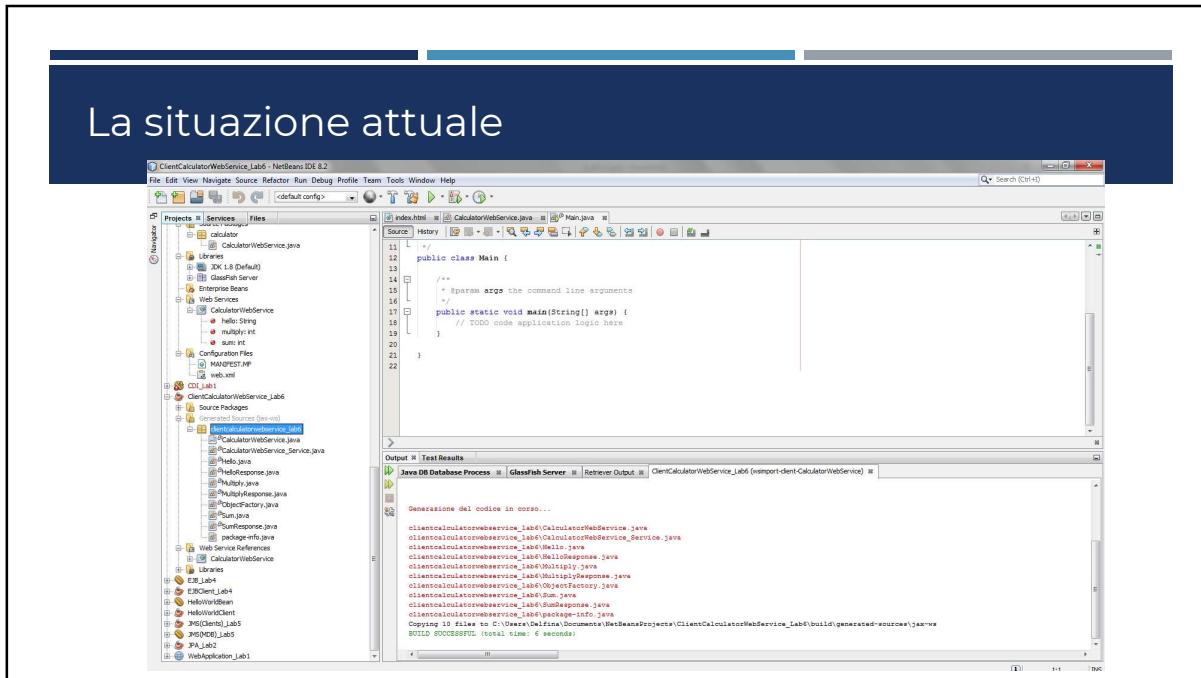
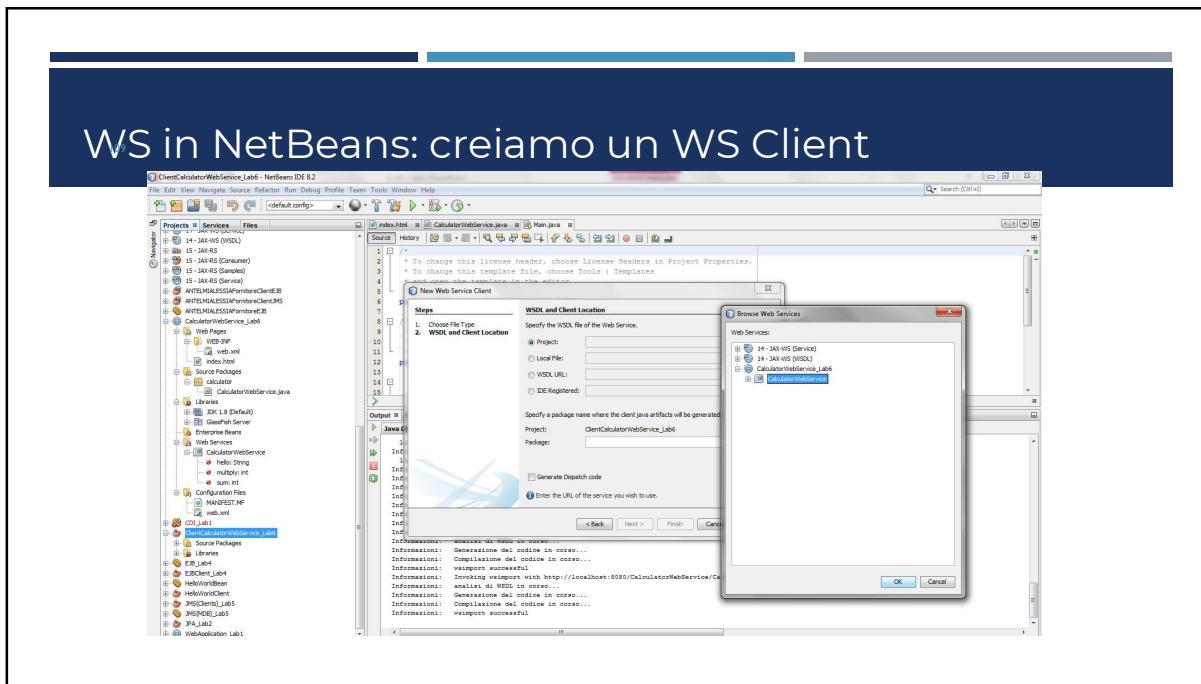
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sumResponse xmlns:ns2="http://calculator/">
      <return>8</return>
    </ns2:sumResponse>
  </S:Body>
</S:Envelope>
```

Organizzazione della lezione

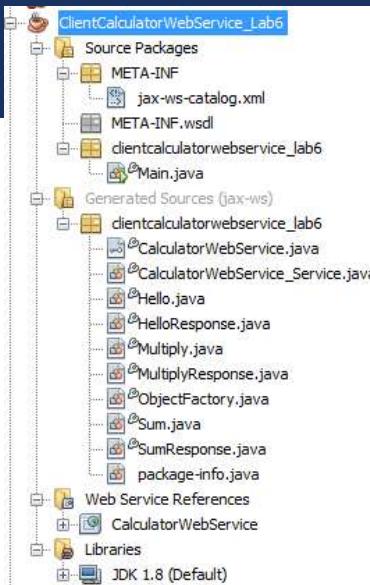
- WS in Java
 - WSDL Mapping
 - Eccezioni e Fault
 - Contesto e ciclo di vita
- Invocare un WS
 - Un esempio riassuntivo
- Supporto ai WS in Netbeans
 - Il progetto per WS
 - Testing
 - WS Client
- Conclusioni







WS in NetBeans: i files generati



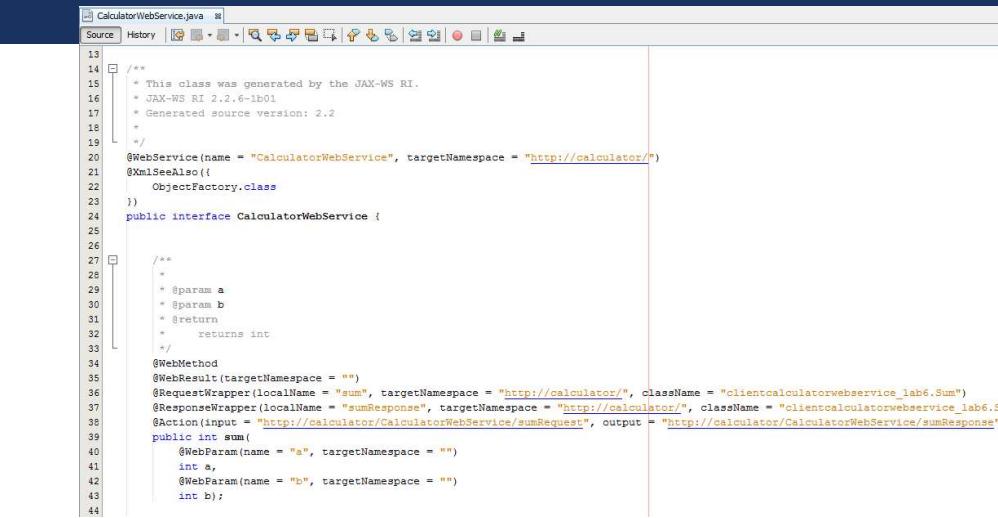
WS in NetBeans: Output

The screenshot shows the NetBeans Output window with the following content:

```
Generazione del codice in corso...

clientcalculatorwebservice_lab6\CalculatorWebService.java
clientcalculatorwebservice_lab6\CalculatorWebService_Service.java
clientcalculatorwebservice_lab6>Hello.java
clientcalculatorwebservice_lab6>HelloResponse.java
clientcalculatorwebservice_lab6\Multiply.java
clientcalculatorwebservice_lab6\MultiplyResponse.java
clientcalculatorwebservice_lab6\ObjectFactory.java
clientcalculatorwebservice_lab6\Sum.java
clientcalculatorwebservice_lab6\SumResponse.java
clientcalculatorwebservice_lab6\package-info.java
Copying 10 files to C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6\build\generated-sources\jax-ws
BUILD SUCCESSFUL (total time: 6 seconds)
```

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

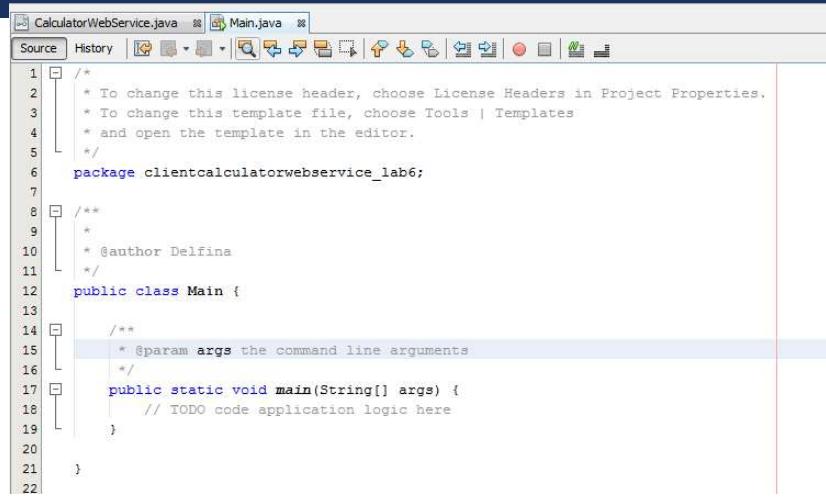


```

13 /**
14  * This class was generated by the JAX-WS RI.
15  * JAX-WS RI 2.2.6-1b01
16  * Generated source version: 2.2
17  *
18 */
19
20 @WebService(name = "CalculatorWebService", targetNamespace = "http://calculator/")
21 @XmlSeeAlso({
22     ObjectFactory.class
23 })
24 public interface CalculatorWebService {
25
26
27     /**
28      *
29      * @param a
30      * @param b
31      * @return
32      *     returns int
33     */
34     @WebMethod
35     @WebResult(targetNamespace = "")
36     @RequestWrapper(localName = "sum", targetNamespace = "http://calculator/", className = "clientcalculatorwebservice_lab6.Sum")
37     @ResponseWrapper(localName = "sumResponse", targetNamespace = "http://calculator/", className = "clientcalculatorwebservice_lab6.S")
38     @Action(input = "http://calculator/CalculatorWebService/sumRequest", output = "http://calculator/CalculatorWebService/sumResponse")
39     public int sum(
40         @WebParam(name = "a", targetNamespace = "")
41         int a,
42         @WebParam(name = "b", targetNamespace = "")
43         int b);
44

```

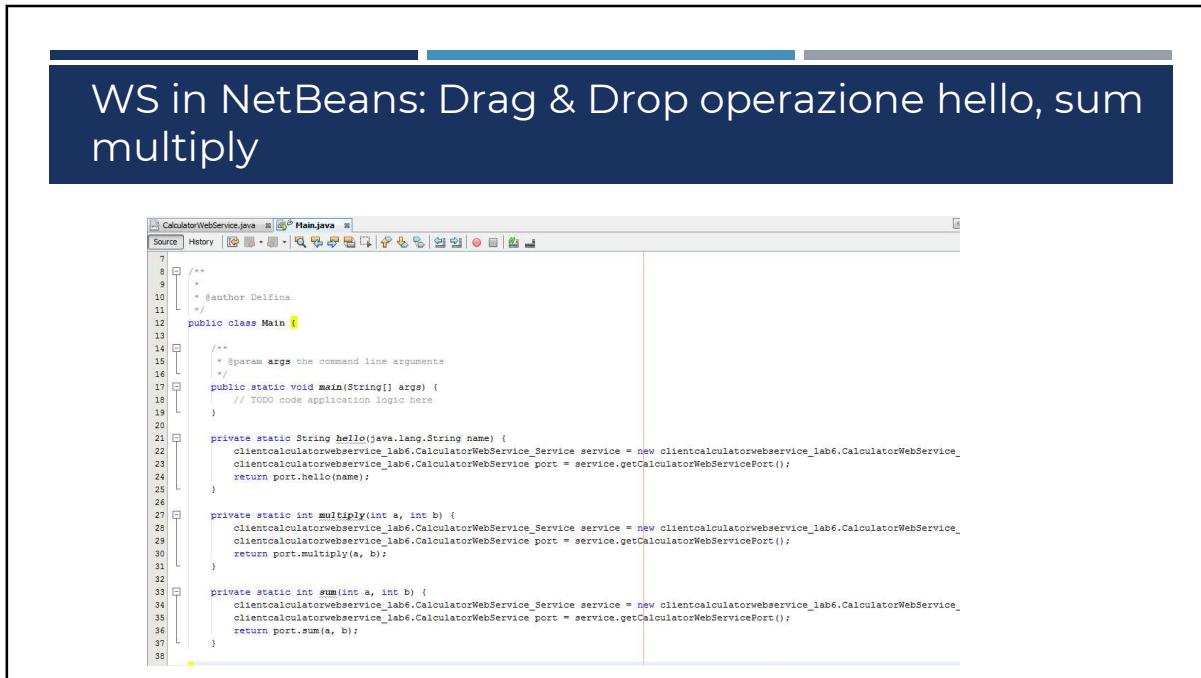
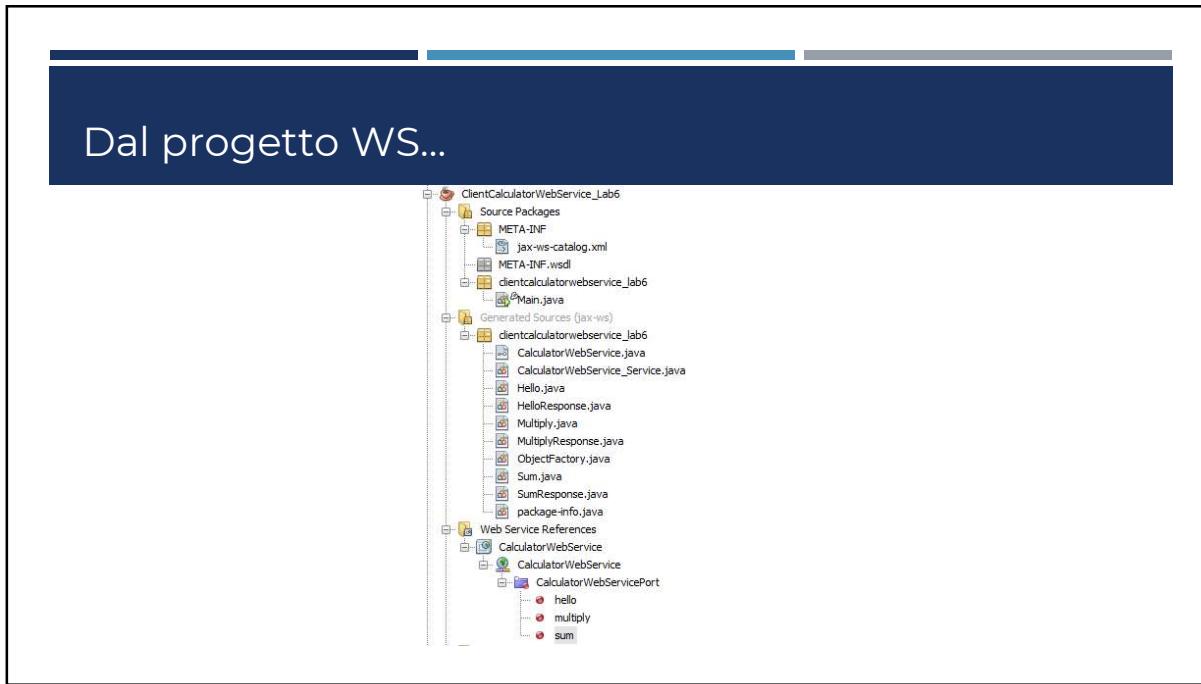
WS in NetBeans: il Client (VUOTO)



```

1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
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      * @param args the command line arguments
16     */
17     public static void main(String[] args) {
18         // TODO code application logic here
19     }
20
21 }
22

```



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

```
private static String hello(java.lang.String name) {
    clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
        new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
    clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
    return port.hello(name);
}

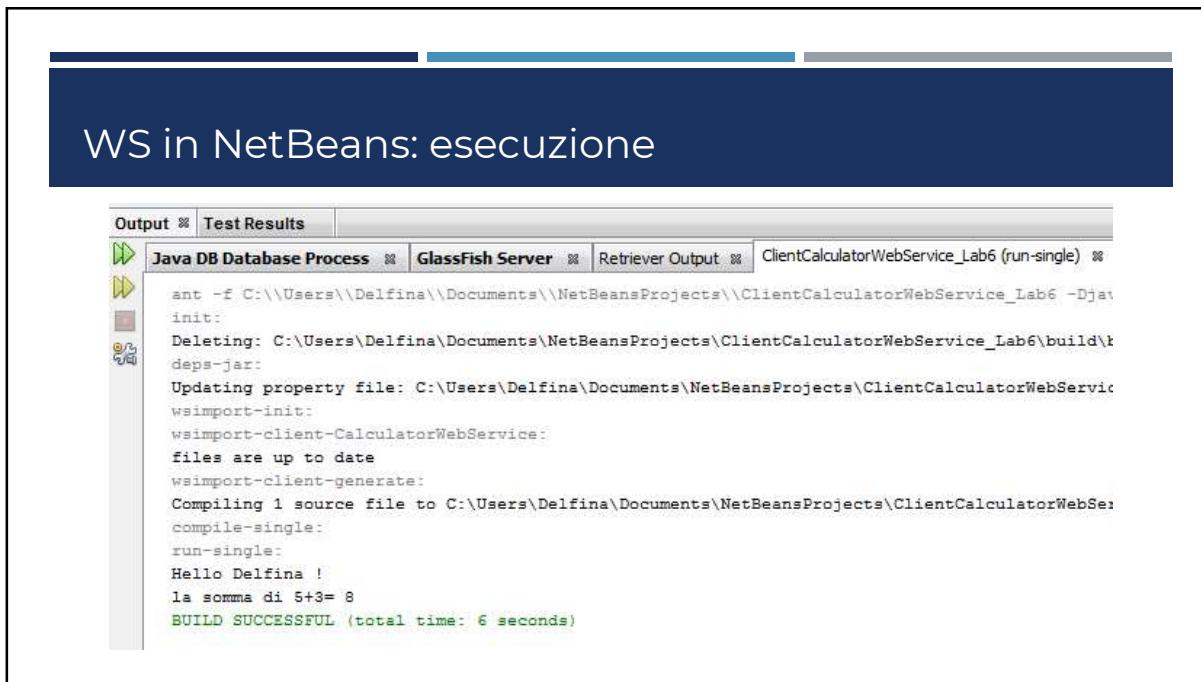
private static int multiply(int a, int b) {
    clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
        new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
    clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
    return port.multiply(a, b);
}

private static int sum(int a, int b) {
    clientcalculatorwebservice_lab6.CalculatorWebService_Service service =
        new clientcalculatorwebservice_lab6.CalculatorWebService_Service();
    clientcalculatorwebservice_lab6.CalculatorWebService port = service.getCalculatorWebServicePort();
    return port.sum(a, b);
}
```

WS in NetBeans: esecuzione

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

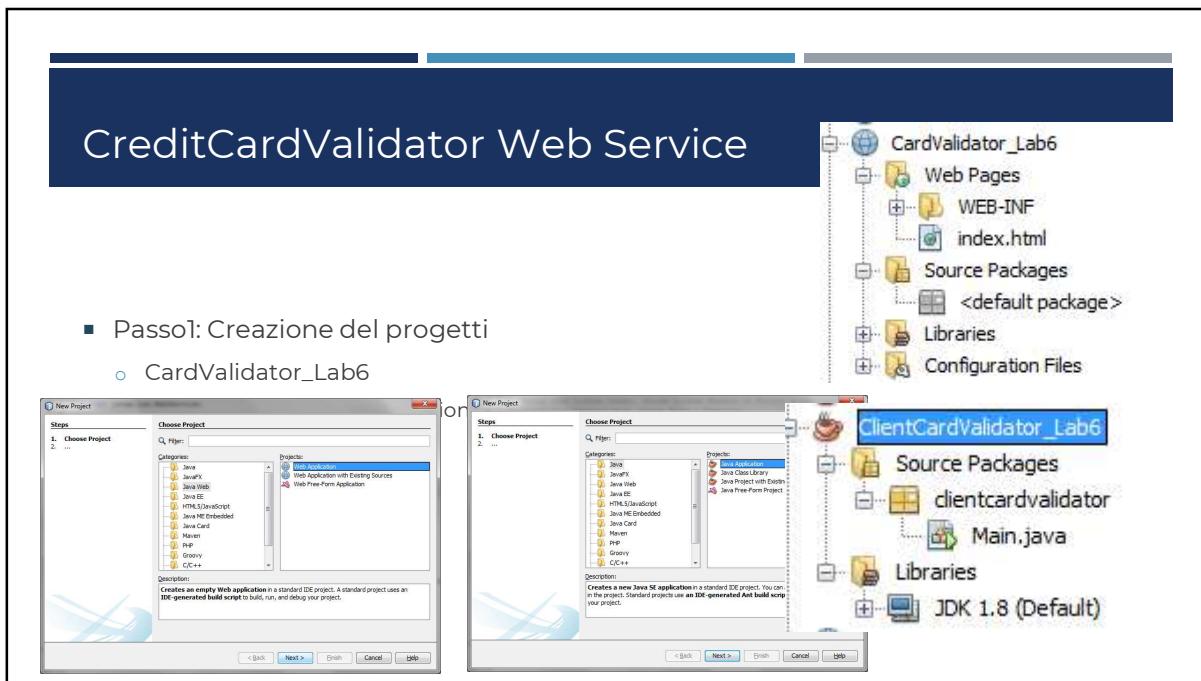
WS in NetBeans: esecuzione



```

Output & Test Results
Java DB Database Process & GlassFish Server & Retriever Output & ClientCalculatorWebService_Lab6 (run-single) &
ant -f C:\\\\Users\\\\Delfina\\\\Documents\\\\NetBeansProjects\\\\ClientCalculatorWebService_Lab6 -Djetty.home=C:\\\\Users\\\\Delfina\\\\Documents\\\\NetBeansProjects\\\\ClientCalculatorWebService_Lab6
init:
Deleting: C:\\Users\\Delfina\\Documents\\NetBeansProjects\\ClientCalculatorWebService_Lab6\\build\\target
deps-jar:
Updating property file: C:\\Users\\Delfina\\Documents\\NetBeansProjects\\ClientCalculatorWebService_Lab6\\build\\target\\wsimport-init.jaxws.properties
wsimport-init:
wsimport-client-CalculatorWebService:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\\Users\\Delfina\\Documents\\NetBeansProjects\\ClientCalculatorWebService_Lab6\\src\\main\\java
compile-single:
run-single:
Hello Delfina !
la somma di 5+3= 8
BUILD SUCCESSFUL (total time: 6 seconds)

```



CreditCardValidator Web Service

Listing 14-34. The CreditCard Class with JAXB Annotations

- La classe CreditCard

```
@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  
}
```

CreditCardValidator Web Service

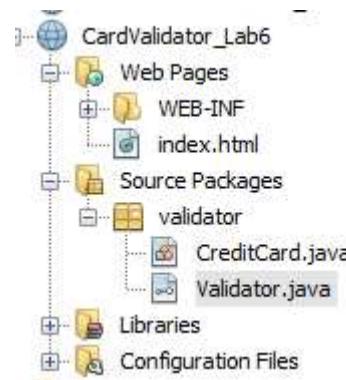
Listing 14-35. The Validator Web Service Interface

- L'interfaccia

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

La situazione attuale

- In NetBeans



CreditCardValidator Web Service

Listing 14-36. The CardValidator Web Service Bean

```

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

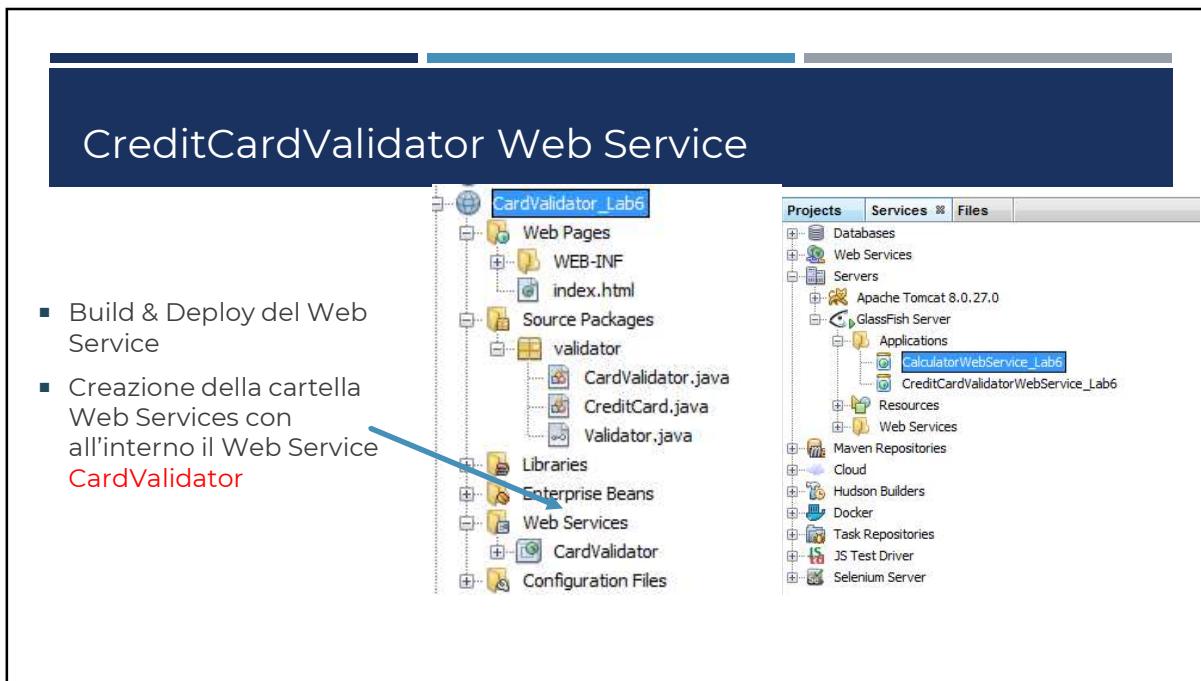
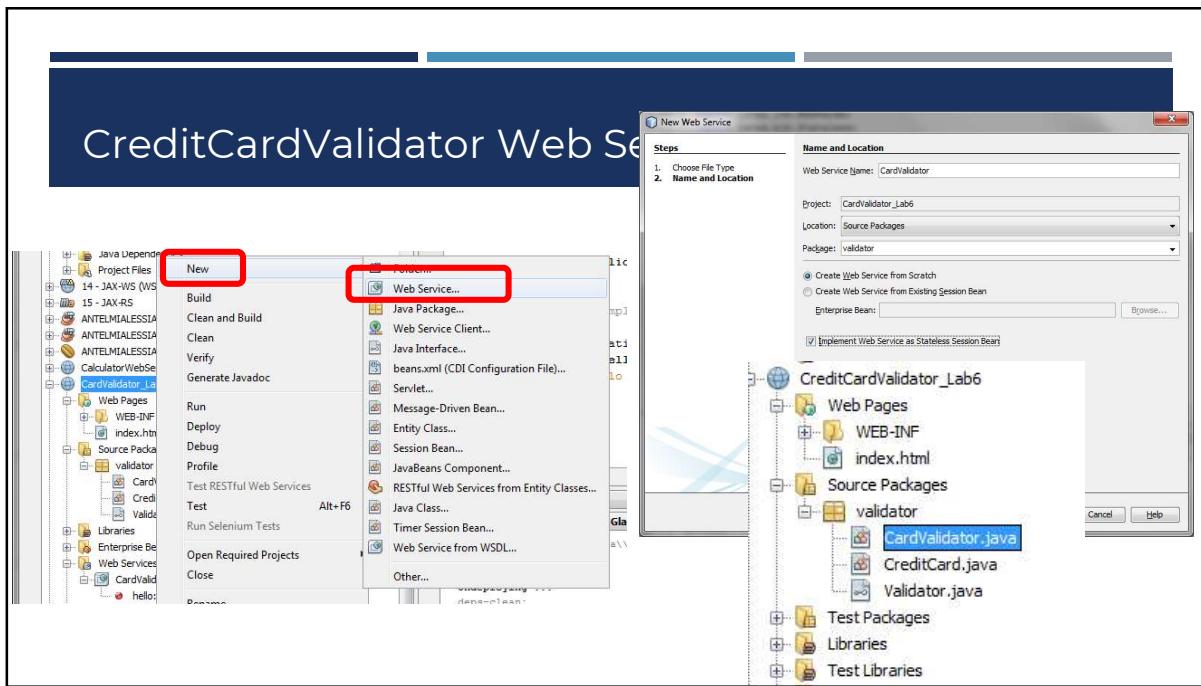
    public boolean validate(CreditCard creditCard) {

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

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

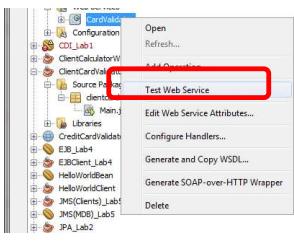
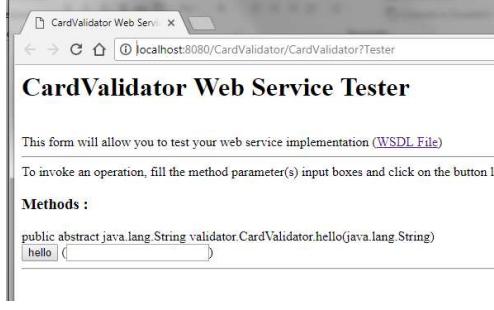
```

- La classe CardValidator



CreditCardValidator Web Service

- Per verificare che tutto funzioni

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

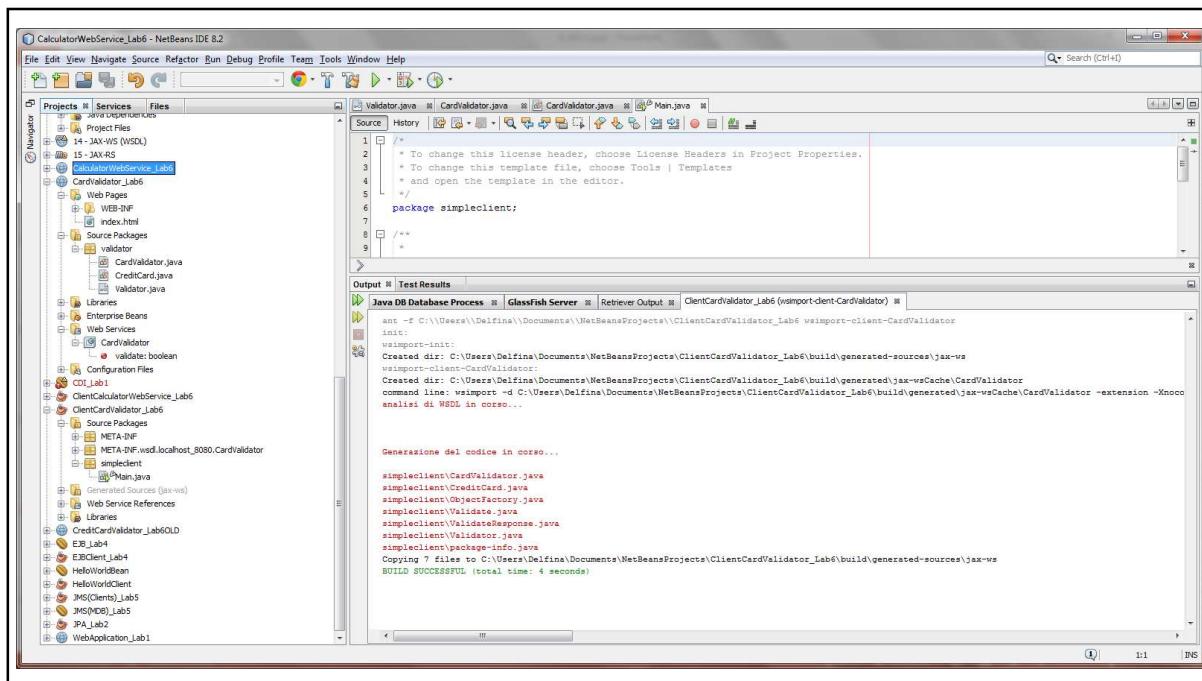
CreditCardValidator Web Service

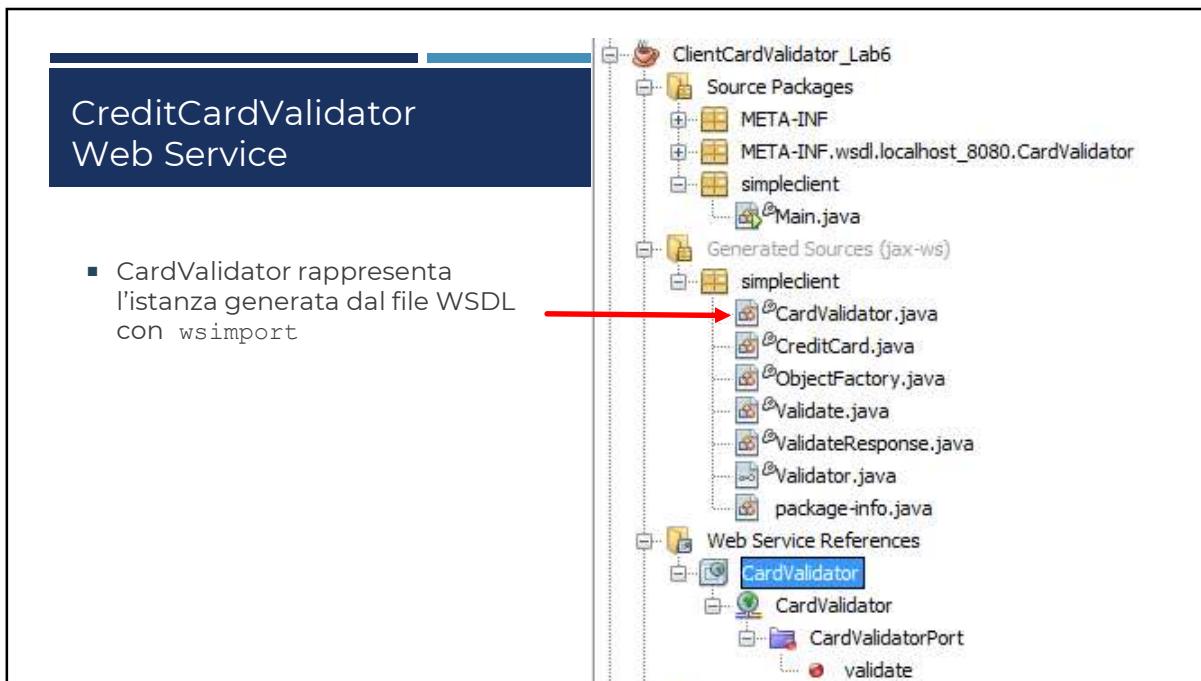
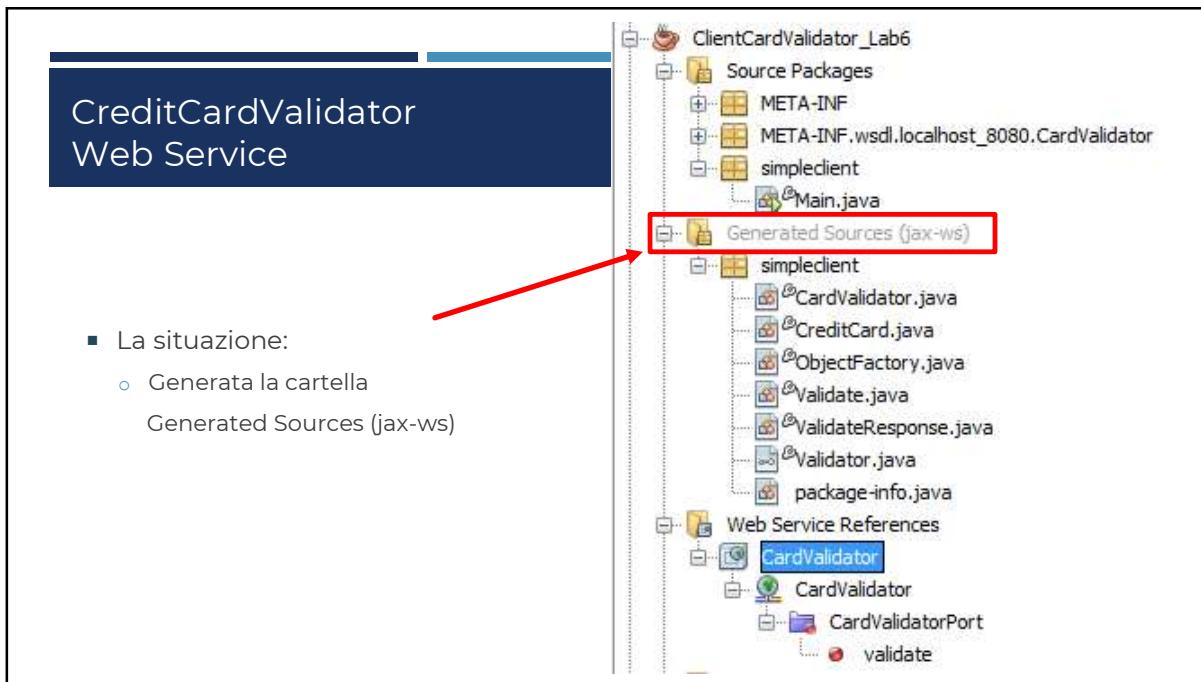
- Nuova situazione




CreditCardValidator Web Service

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





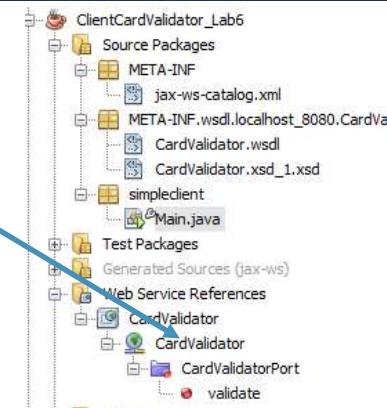
Il client: invoking programmatically

- 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 dei credit card messages, ecc

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

Il client: invoking programmatically

- Drag & Drop da qui nella classe del client



The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Shows the file name "Main.java".
- Toolbar:** Includes icons for Source, History, and various project management tools.
- Code Editor:** Displays the Java code for the Main class. The code initializes a CreditCard object, sets its number, expiry date, and type, and then prints validation results twice. It also contains a validate method that creates a CardValidator service and returns the result of calling validate on its port.

```
10  * @author Delfina
11  */
12  public class Main {
13      public static void main(String[] args) {
14          System.out.println("Invoking web service programmatically");
15
16          CreditCard creditCard = new CreditCard();
17          creditCard.setNumber("12341234");
18          creditCard.setExpiryDate("10/12");
19          creditCard.setType("VISA");
20          creditCard.setControlNumber(1234);
21          System.out.println(validate(creditCard));
22          creditCard.setNumber("12341233");
23          System.out.println(validate(creditCard));
24      }
25
26      private static boolean validate(CreditCard creditCard) {
27          CardValidator service = new CardValidator();
28          Validator port = service.getCardValidatorPort();
29          return port.validate(creditCard);
30      }
31
32  }
33 }
```

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Shows the tabs "Output" and "Test Results".
- Output Window:** Displays the build log for the project "ClientCardValidator_Lab6 (run-single)". The log shows the execution of various Ant tasks: deps-jar, Updating property file, wsimport-init, wsimport-client-CardValidator (files are up to date), wsimport-client-generate, Compiling 1 source file, compile-single, run-single, and finally BUILD SUCCESSFUL (total time: 4 seconds).

```
deps-jar:
Updating property file: C:\Users\Delina\Documents\NetBeansProjects\ClientCardValidator_Lab6\src\main\java\com\client\CardValidator.java
wsimport-init:
wsimport-client-CardValidator:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\Delina\Documents\NetBeansProjects\ClientCardValidator_Lab6\src\main\java
compile-single:
run-single:
Invoking web service programmatically
true
false
BUILD SUCCESSFUL (total time: 4 seconds)
```

Organizzazione della lezione

- 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

