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 {
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

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

Il Web Service

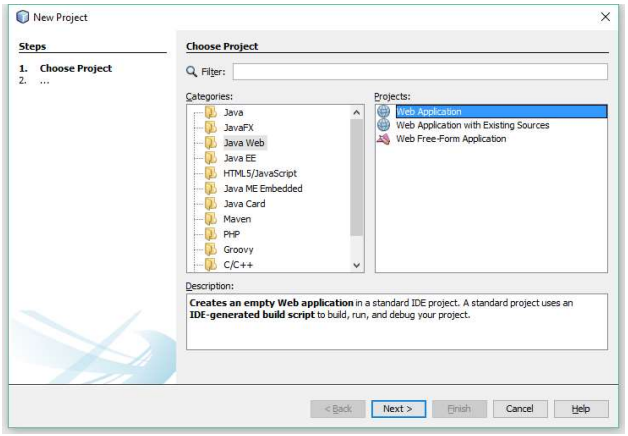
```
@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

@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

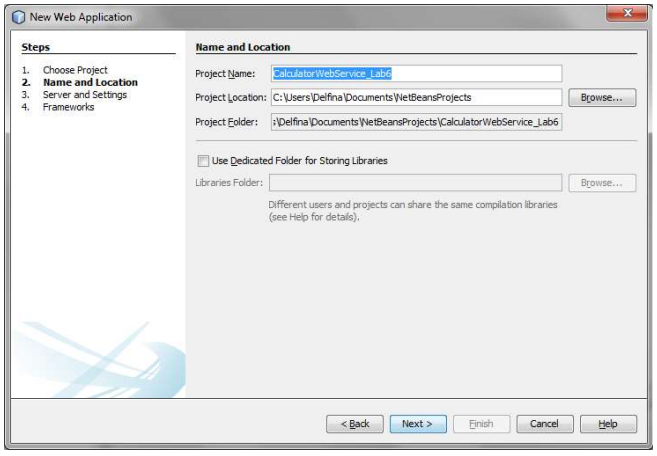
Organizzazione della lezione

- 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

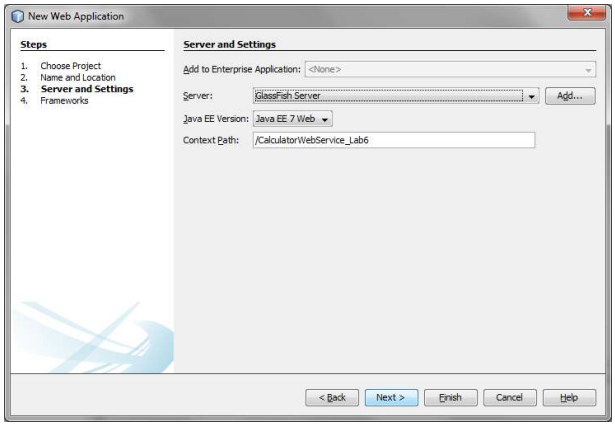
WS in NetBeans: creazione di un progetto



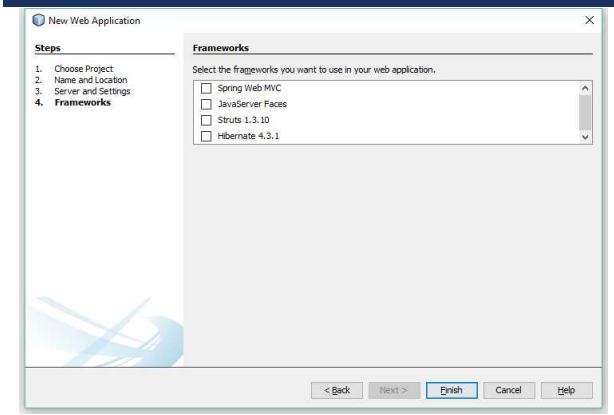
WS in NetBeans: creazione di un progetto

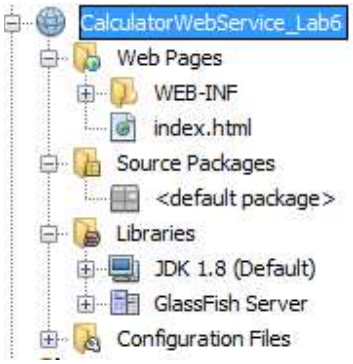


WS in NetBeans: creazione di un progetto



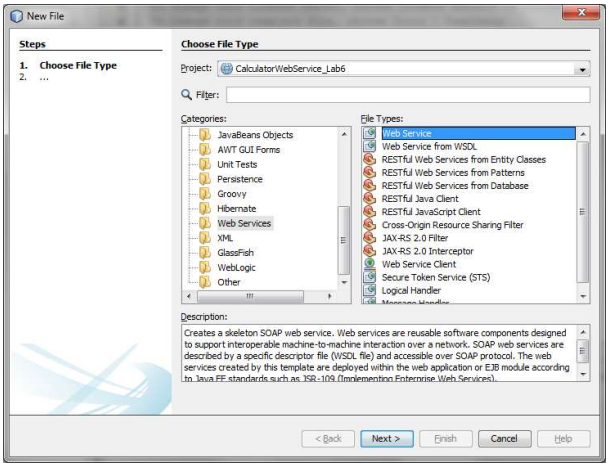
WS in NetBeans: creazione di un progetto



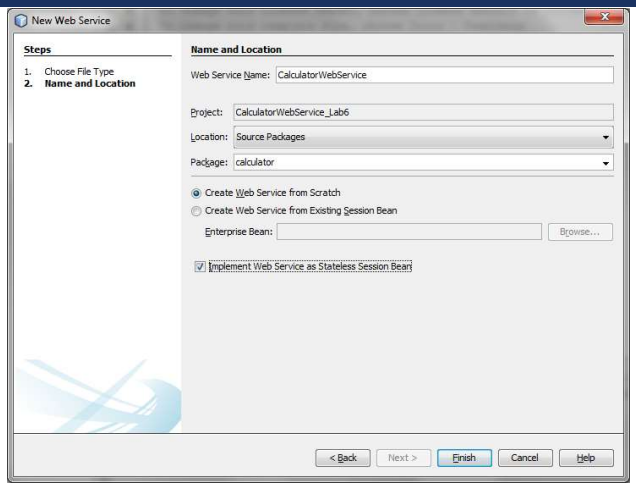


WS in NetBeans: creazione di un progetto

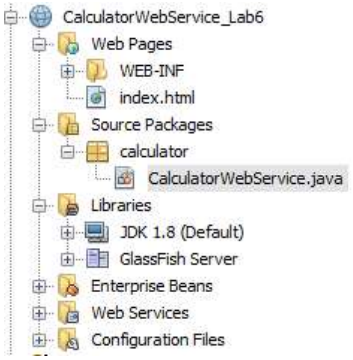
Right-click sul progetto → New

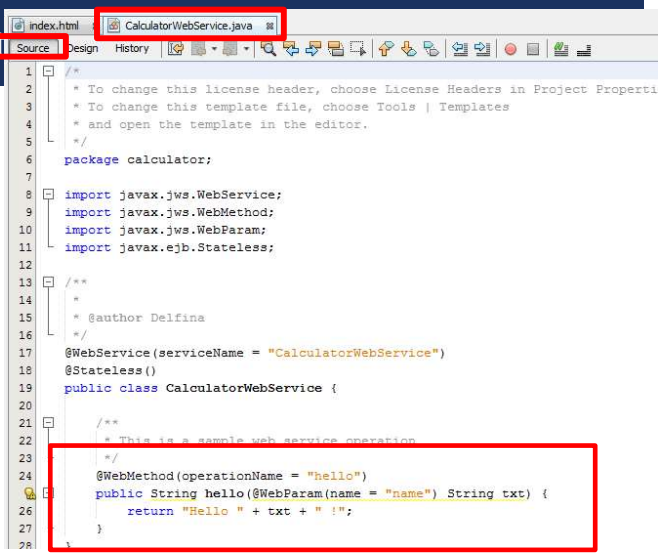


WS in NetBeans: creazione di un WS

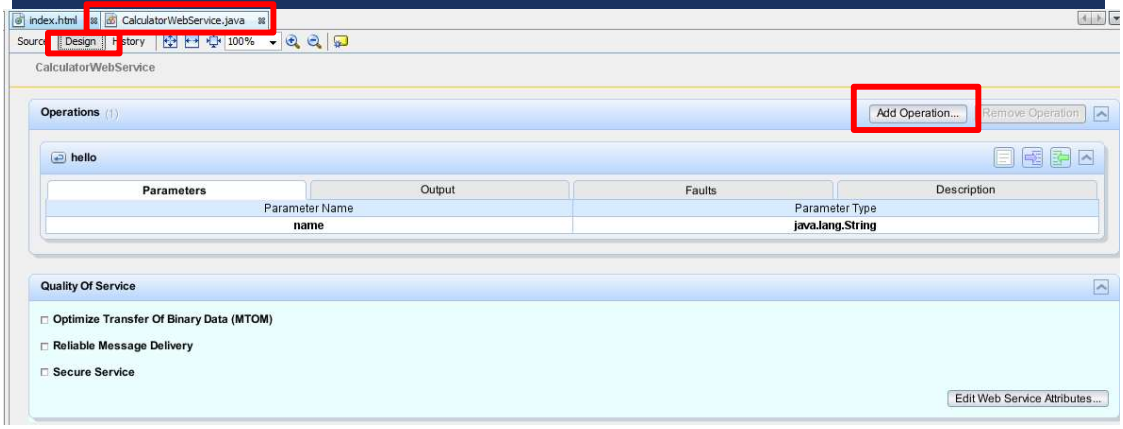


WS in NetBeans: la si

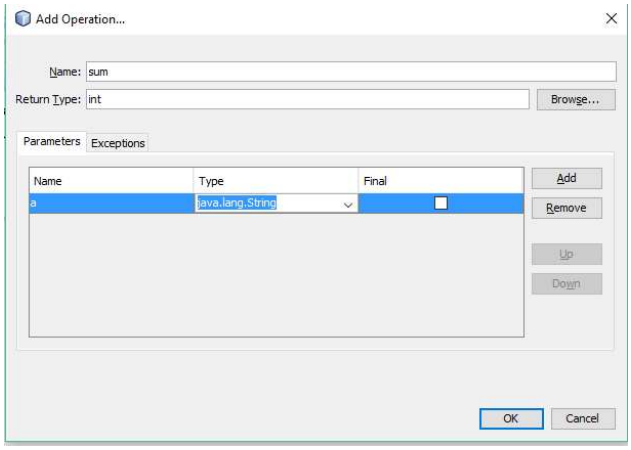




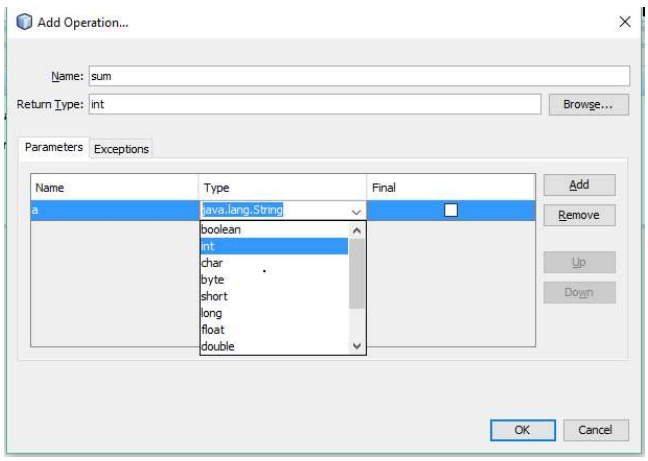
WS in NetBeans: la vista di DESIGN



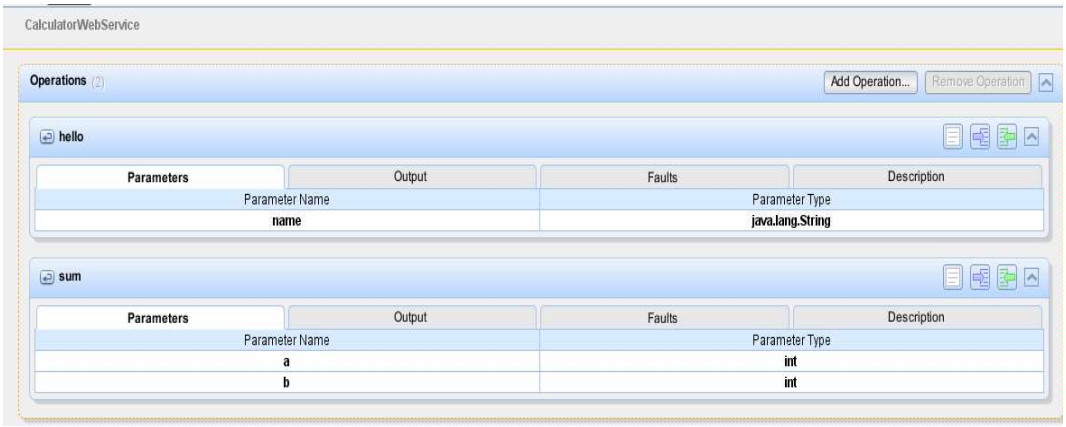
WS in NetBeans: aggiungiamo una operazione



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

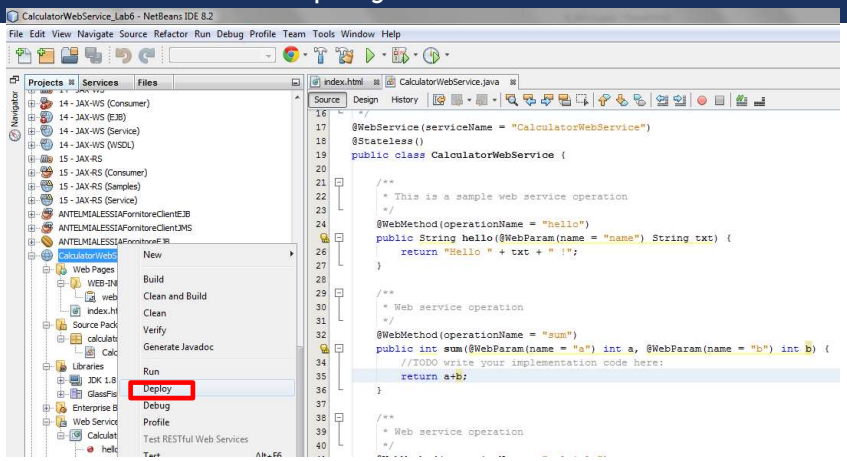
```
import javax.jws.WebParam;
import javax.ejb.Stateless;

/**
 *
 * @author Delfina
 */
@WebService(serviceName = "CalculatorWebService")
@Stateless()
public class CalculatorWebService {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }

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

WS in NetBeans

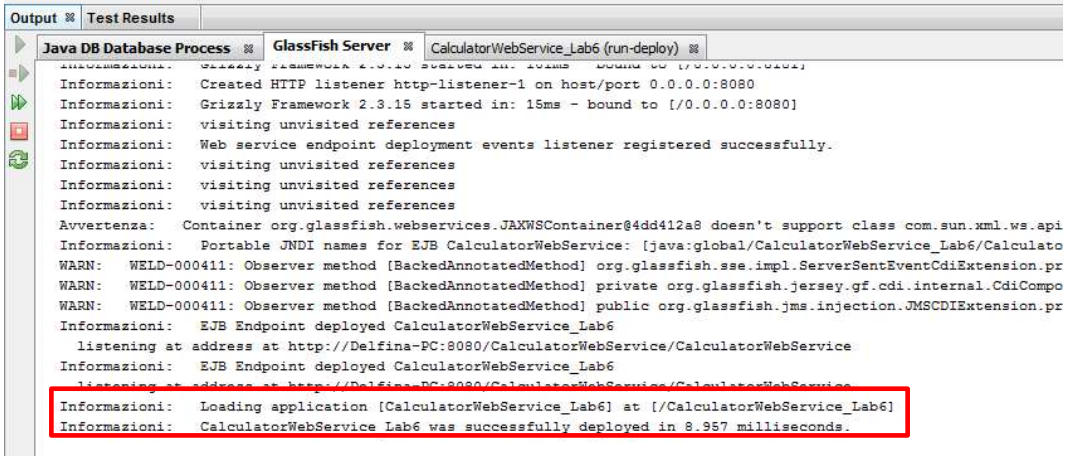
WS in NetBeans: scriviamo qualcosa (implementiamo somma e moltiplicazione)

```
    @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



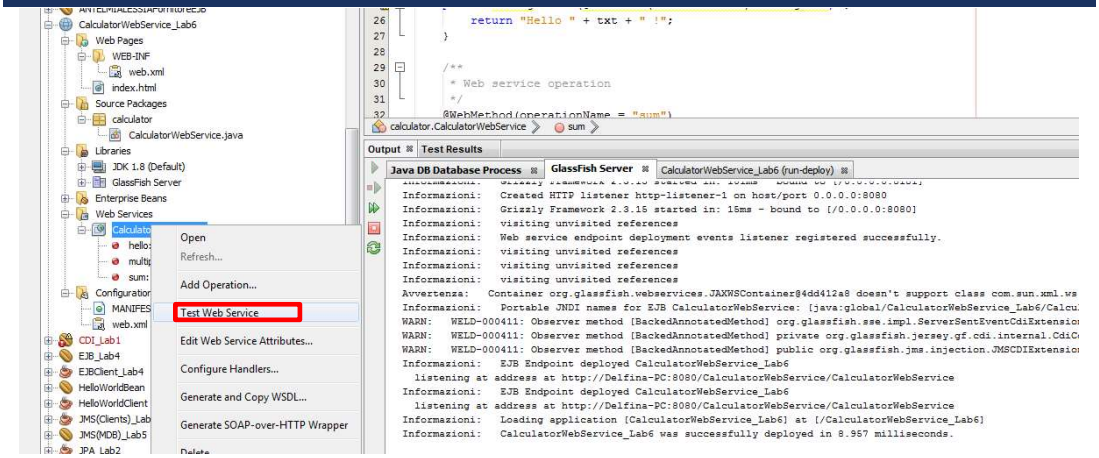
WS in NetBeans: sul server



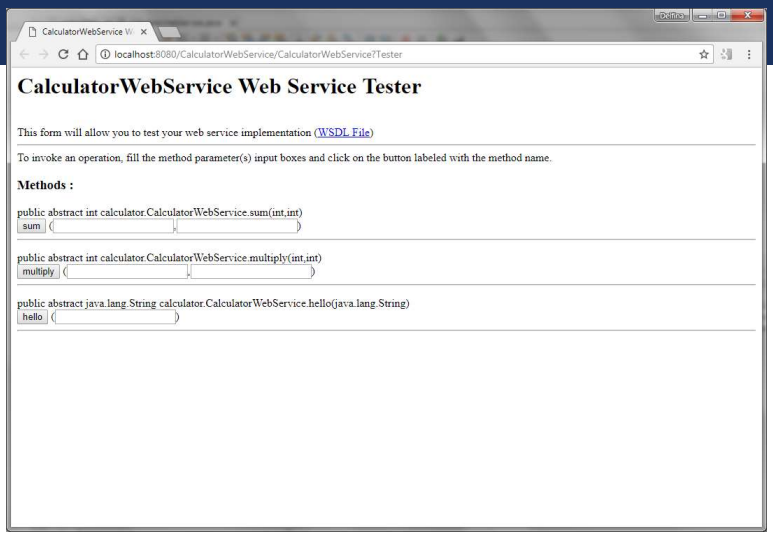
Organizzazione della lezione

- 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

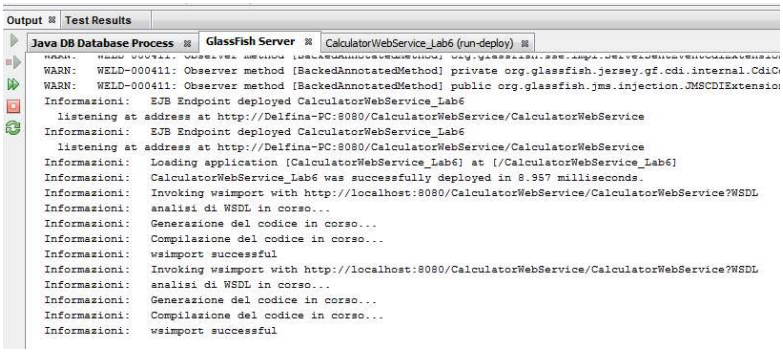


WS in NetBeans: Test

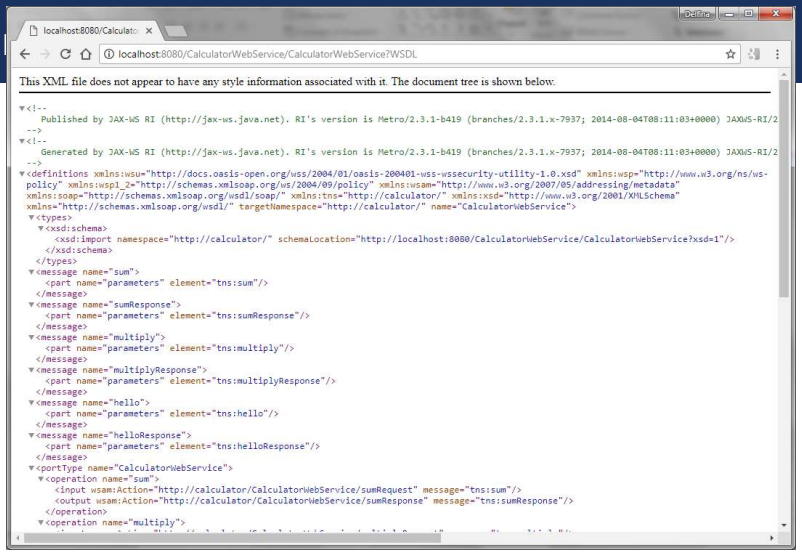


WS in NetBeans: testing

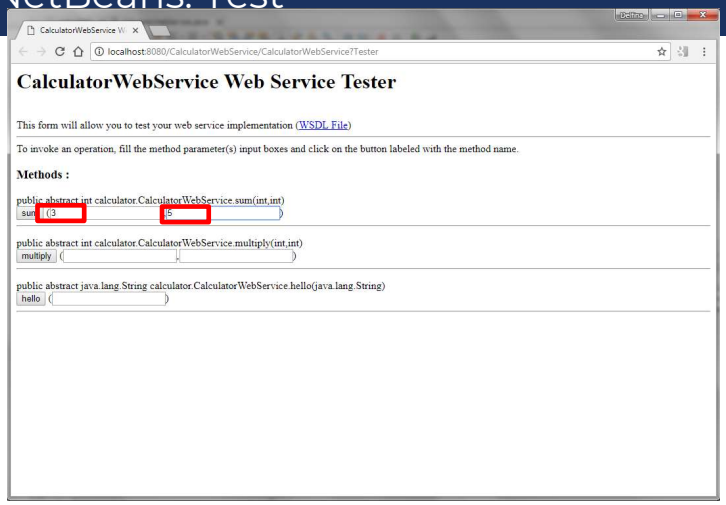
Cosa è successo sul server



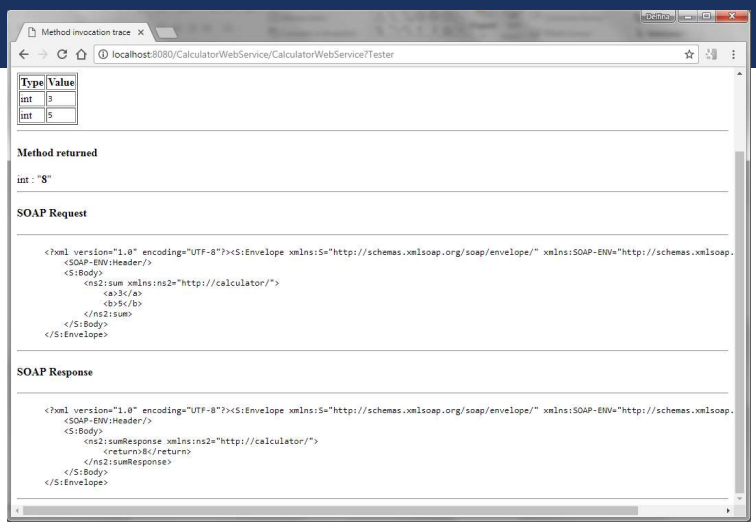
WS i



WS in NetBeans: Test



WS in



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>
```

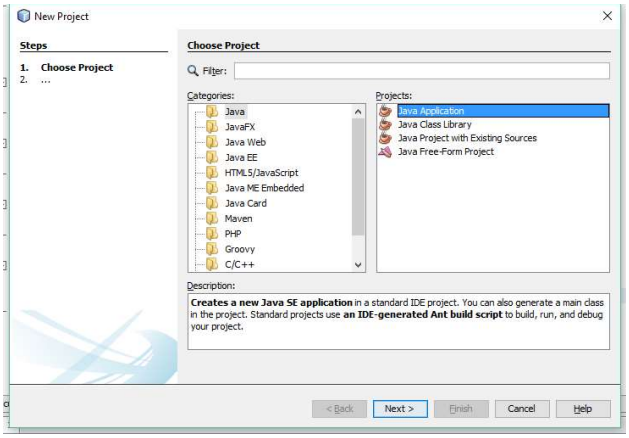
SOAP Response

```
<?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>
```

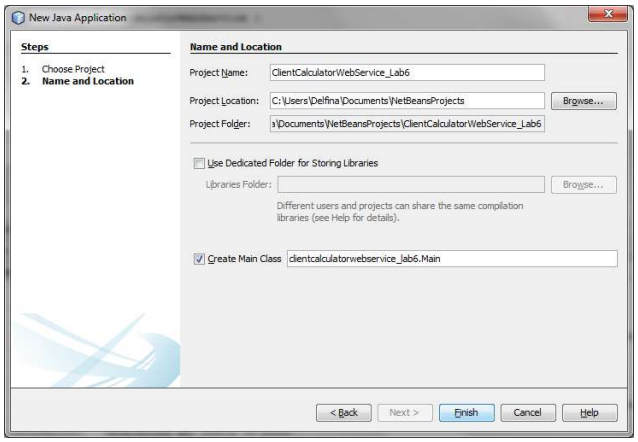
Organizzazione della lezione

- 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

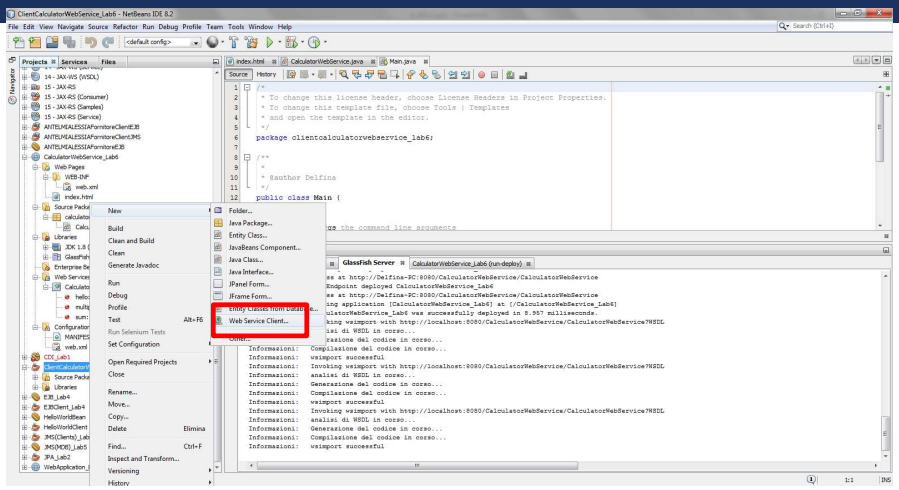
WS in NetBeans: Client



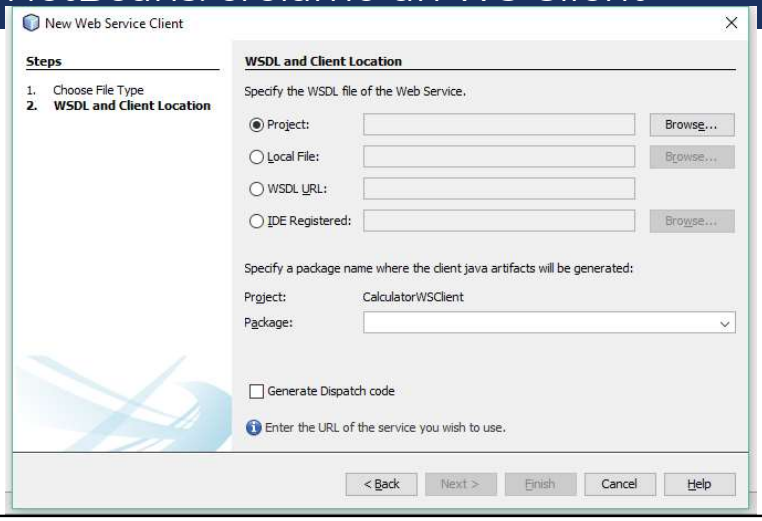
WS in NetBeans: Client



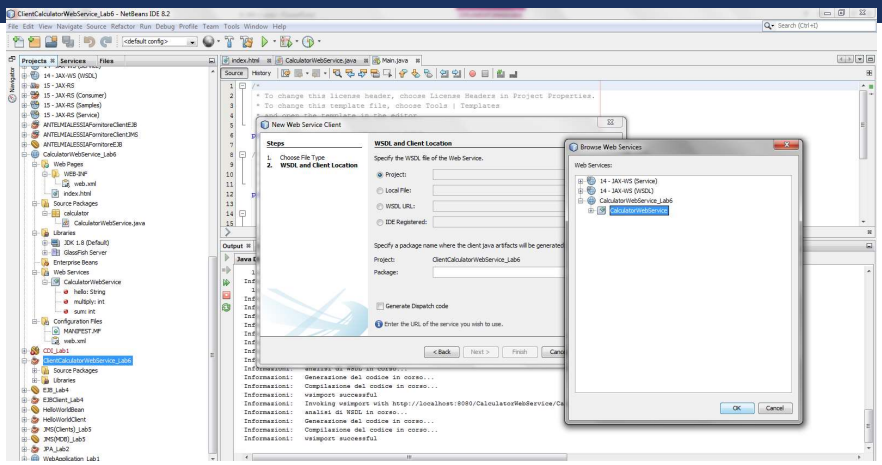
WS in NetBeans: creiamo un WS Client



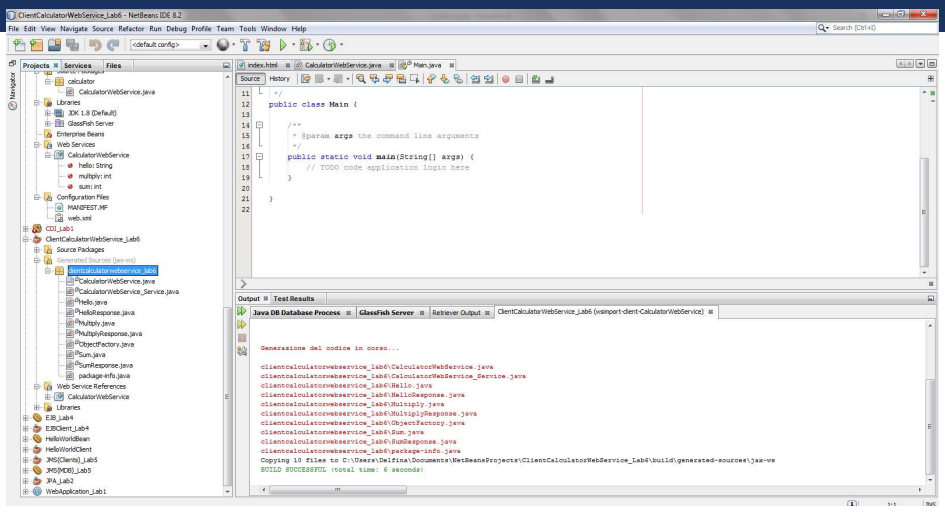
WS in NetBeans: creiamo un WS Client



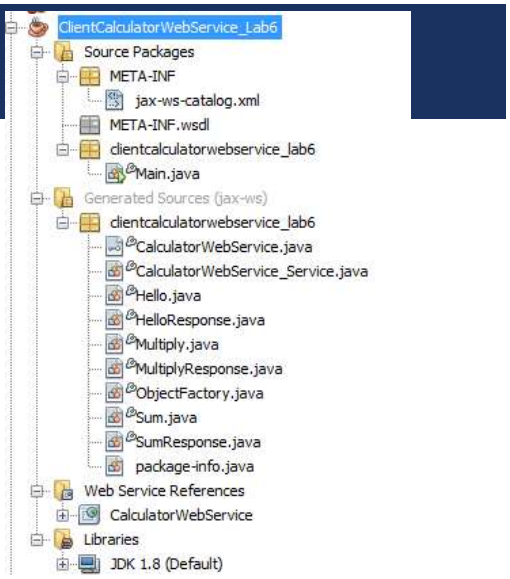
WS in NetBeans: creiamo un WS Client



La situazione attuale



WS in NetBeans: i files generati



WS in NetBeans: Output

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

CalculatorWebService.java

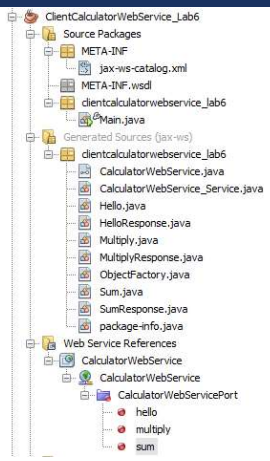
```
/**
 * This class was generated by the JAX-WS RI.
 * JAX-WS RI 2.2.6-1b01
 * Generated source version: 2.2
 *
 */
@WebService(name = "CalculatorWebService", targetNamespace = "http://calculator/")
@XmlSeeAlso({
    ObjectFactory.class
})
public interface CalculatorWebService {


    /**
     *
     * @param a
     * @param b
     * @return
     *     returns int
     */
    @WebMethod
    @WebResult(targetNamespace = "")
    @RequestWrapper(localName = "sum", targetNamespace = "http://calculator/", className = "clientcalculatorwebservice_lab6.Sum")
    @ResponseWrapper(localName = "sumResponse", targetNamespace = "http://calculator/", className = "clientcalculatorwebservice_lab6.S
    @Action(input = "http://calculator/CalculatorWebService/sumRequest", output = "http://calculator/CalculatorWebService/sumResponse"
    public int sum(
        @WebParam(name = "a", targetNamespace = "")
        int a,
        @WebParam(name = "b", targetNamespace = "")
        int b);
```

WS in NetBeans: il Client (VUOTO)

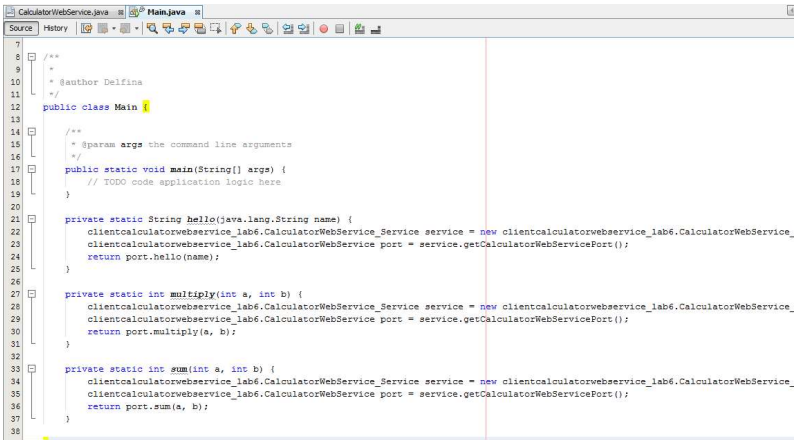
CalculatorWebService.java | Main.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package clientcalculatorwebservice_lab6;

/**
 *
 * @author Delfina
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Dal progetto WS...



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



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

Source | History

```
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         System.out.println(hello("Delfina"));
19         System.out.println("la somma di 5+3= "+sum(5,3));
20     }
21
```

WS in NetBeans: esecuzione

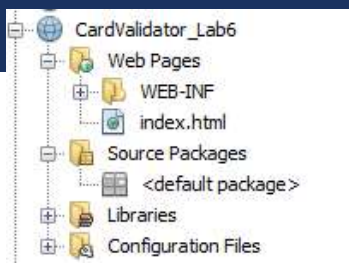
Output | Test Results

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

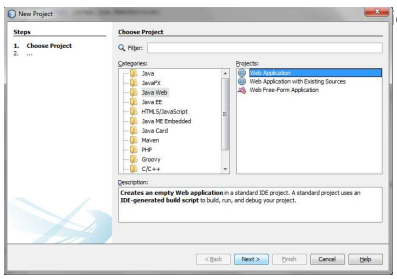
```
ant -f C:\\Users\\Delfina\\Documents\\NetBeansProjects\\ClientCalculatorWebService_Lab6 -Dja
init:
Deleting: C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebService_Lab6\build\
deps-jar:
Updating property file: C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebServic
wsimport-init:
wsimport-client-CalculatorWebService:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\Delfina\Documents\NetBeansProjects\ClientCalculatorWebSe
compile-single:
run-single:
Hello Delfina !
la somma di 5+3= 8
BUILD SUCCESSFUL (total time: 6 seconds)
```

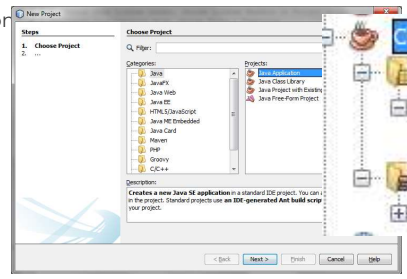
CreditCardValidator Web Service

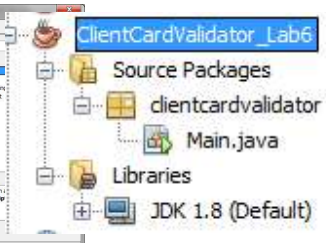
- Passo1: Creazione del progetti
  - CardValidator_Lab6



or







CreditCardValidator Web Service

- La classe CreditCard

Listing 14-34. The CreditCard Class with JAXB Annotations

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

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

  // Constructors, getters, setters
}
```

CreditCardValidator Web Service

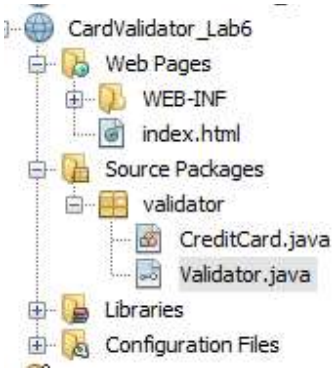
- L'interfaccia

Listing 14-35. The Validator Web Service Interface

```
@WebService
public interface Validator {

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

La situazione attuale

- In NetBeans



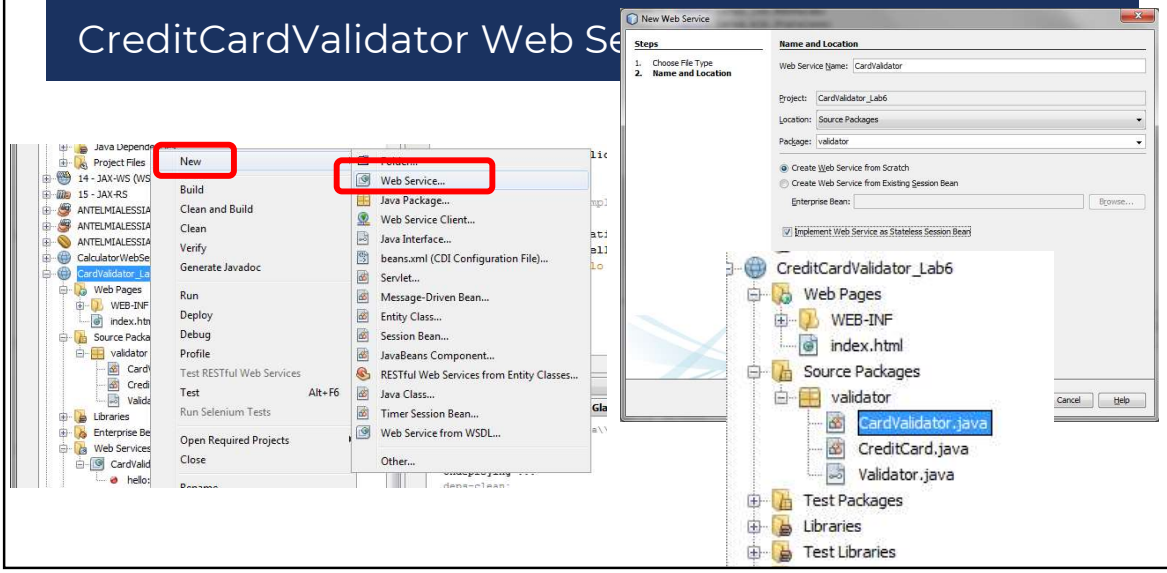
CreditCardValidator Web Service

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;
    }
  }
}
```

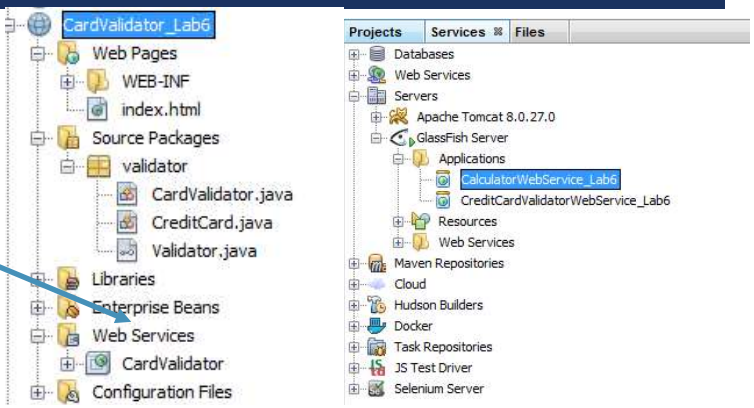
- La classe CardValidator

CreditCardValidator Web Se



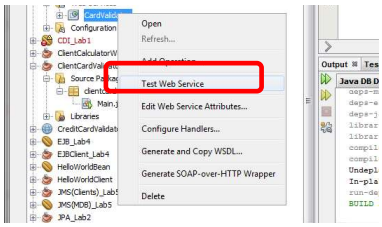
CreditCardValidator Web Service

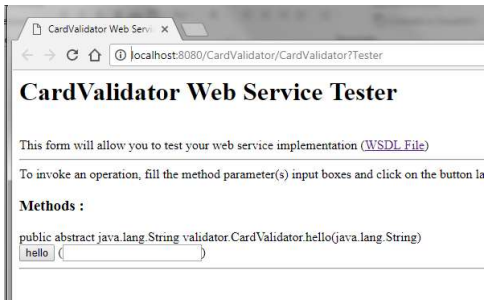
- Build & Deploy del Web Service
- Creazione della cartella Web Services con all'interno il Web Service CardValidator



CreditCardValidator Web Service

- Per verificare che tutto funzioni

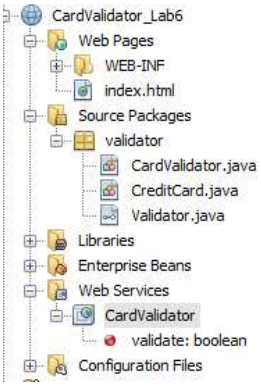


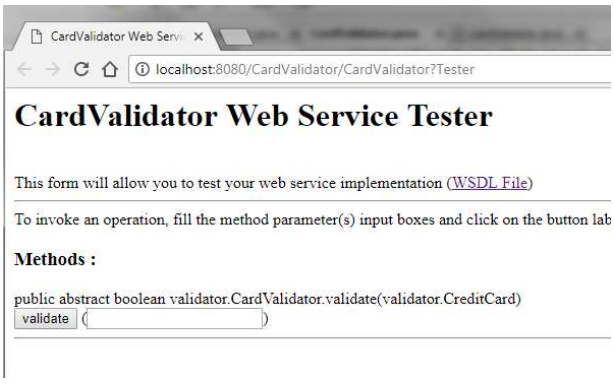


- E' presente il Web Service di default!
- Rimuoviamo hello(), ed aggiungiamo validate()

CreditCardValidator Web Service

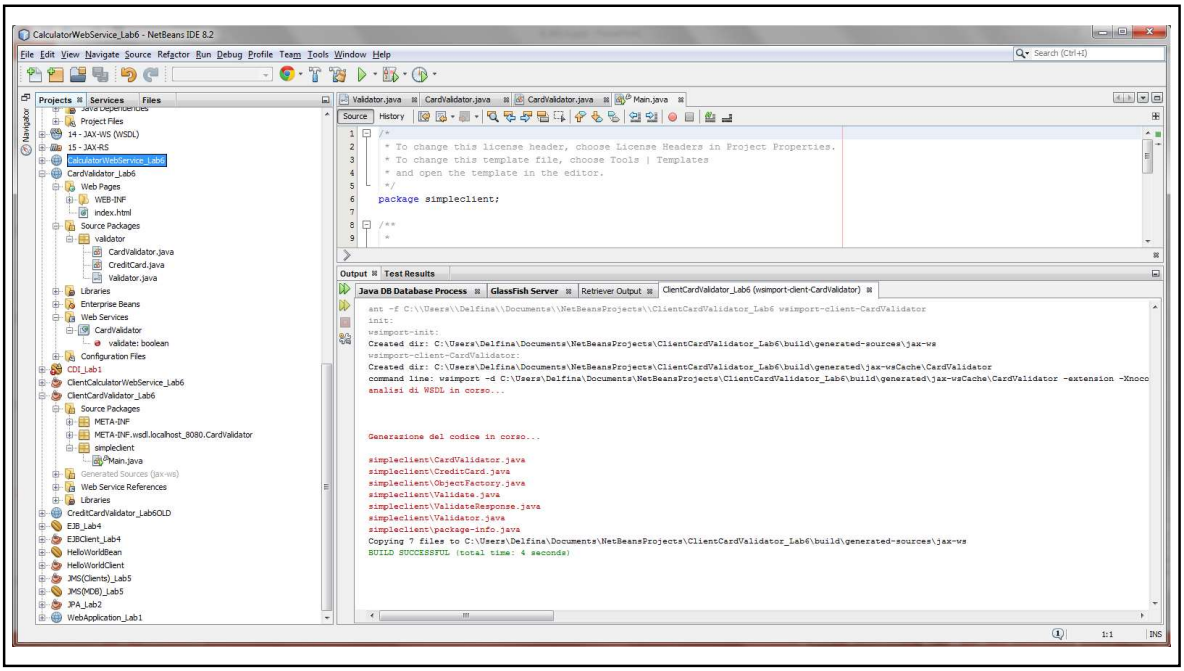
- Nuova situazione





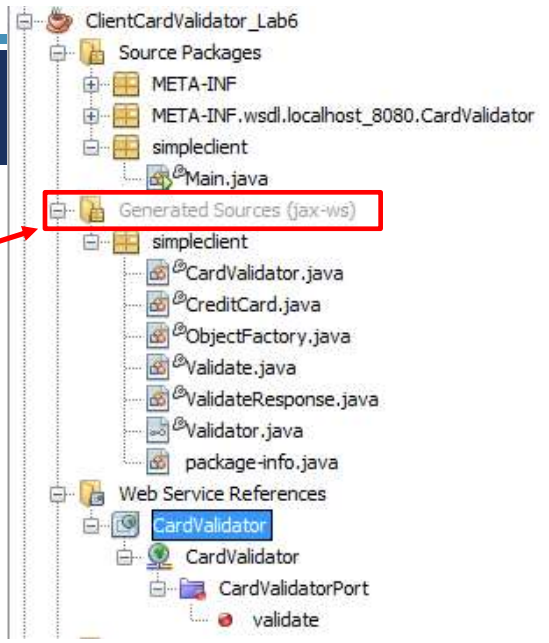
CreditCardValidator Web Service

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



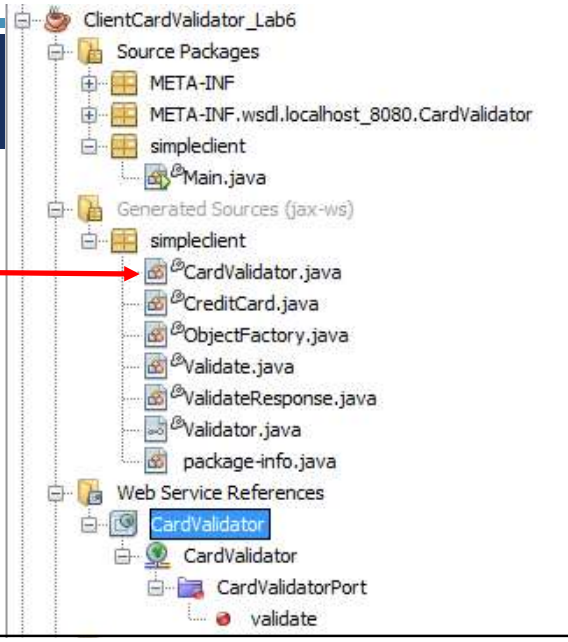
CreditCardValidator Web Service

- La situazione:
  - Generata la cartella Generated Sources (jax-ws)



CreditCardValidator Web Service

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



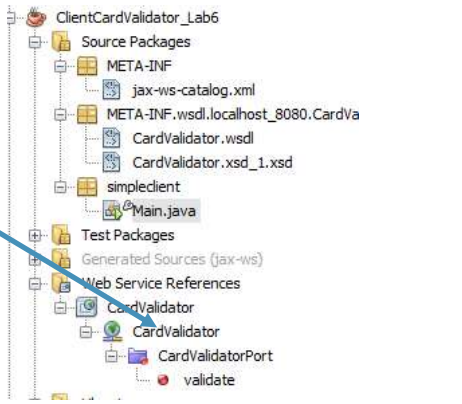
Il client: invoking programmatically

- 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 del 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



Il cl

Main.java

Source | History

```
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
```

Il client: invoking programmatically

Output | Test Results

Java DB Database Process | GlassFish Server | Retriever Output | ClientCardValidator_Lab6 (run-single)

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

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