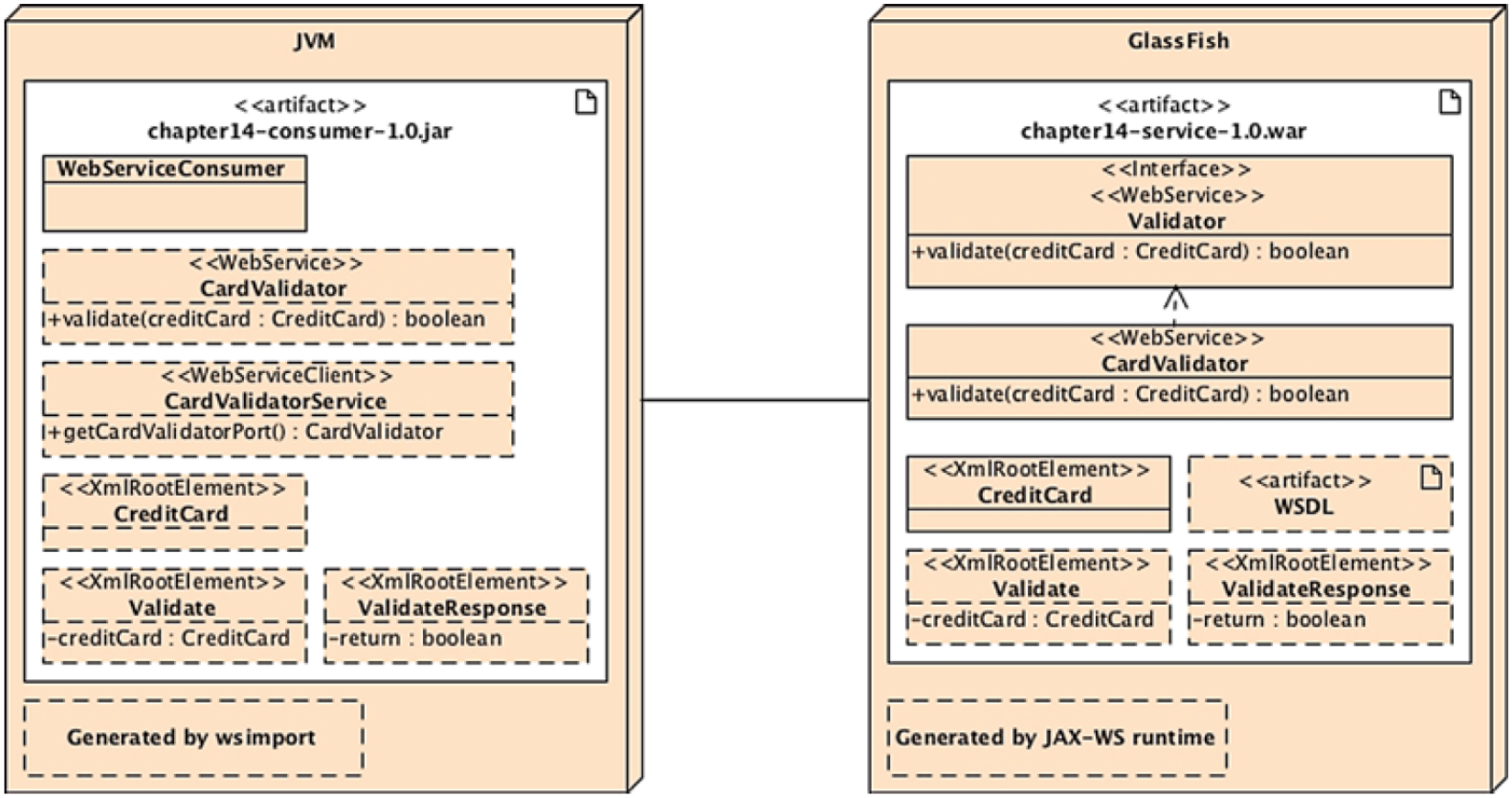
Plan

Diagramma dell'esempio



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

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

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

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

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

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

IL WEBSERVICE

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

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

Con annotazione da WS...

IL WEBSERVICE


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

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

Con annotazione da WS...

...si dichiara l'interfaccia

IL WEBSERVICE


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

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

- Con annotazione da WS...
- ...si dichiara l'interfaccia
- Poi si dichiara il WS

IL WEBSERVICE


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

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

- Con annotazione da WS...
- ...si dichiara l'interfaccia
- Poi si dichiara il WS

Con la classe che implementa l'interfaccia

IL WEBSERVICE

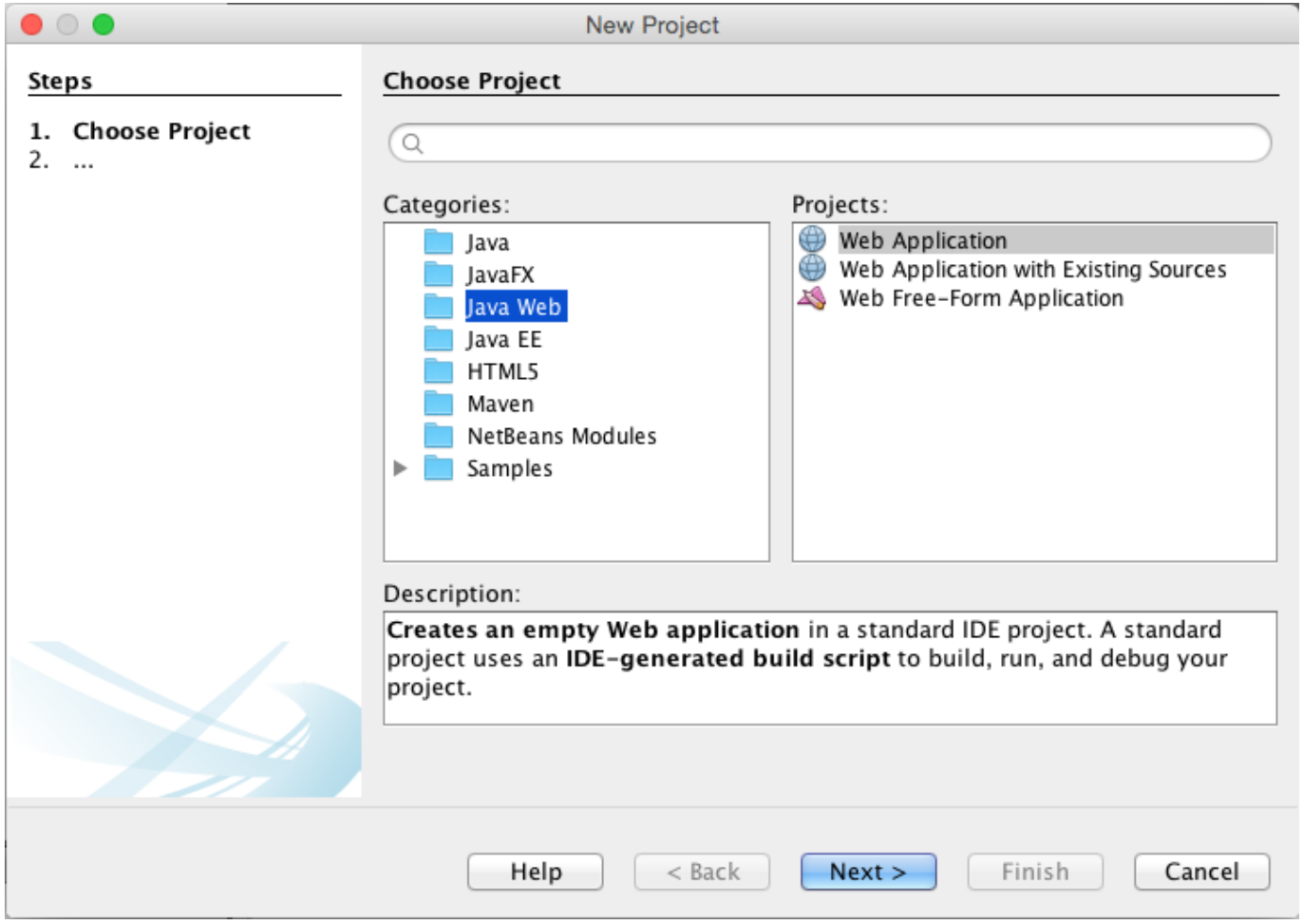

```
@WebService
public interface Validator {
    public boolean validate(CreditCard creditCard);
}

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

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

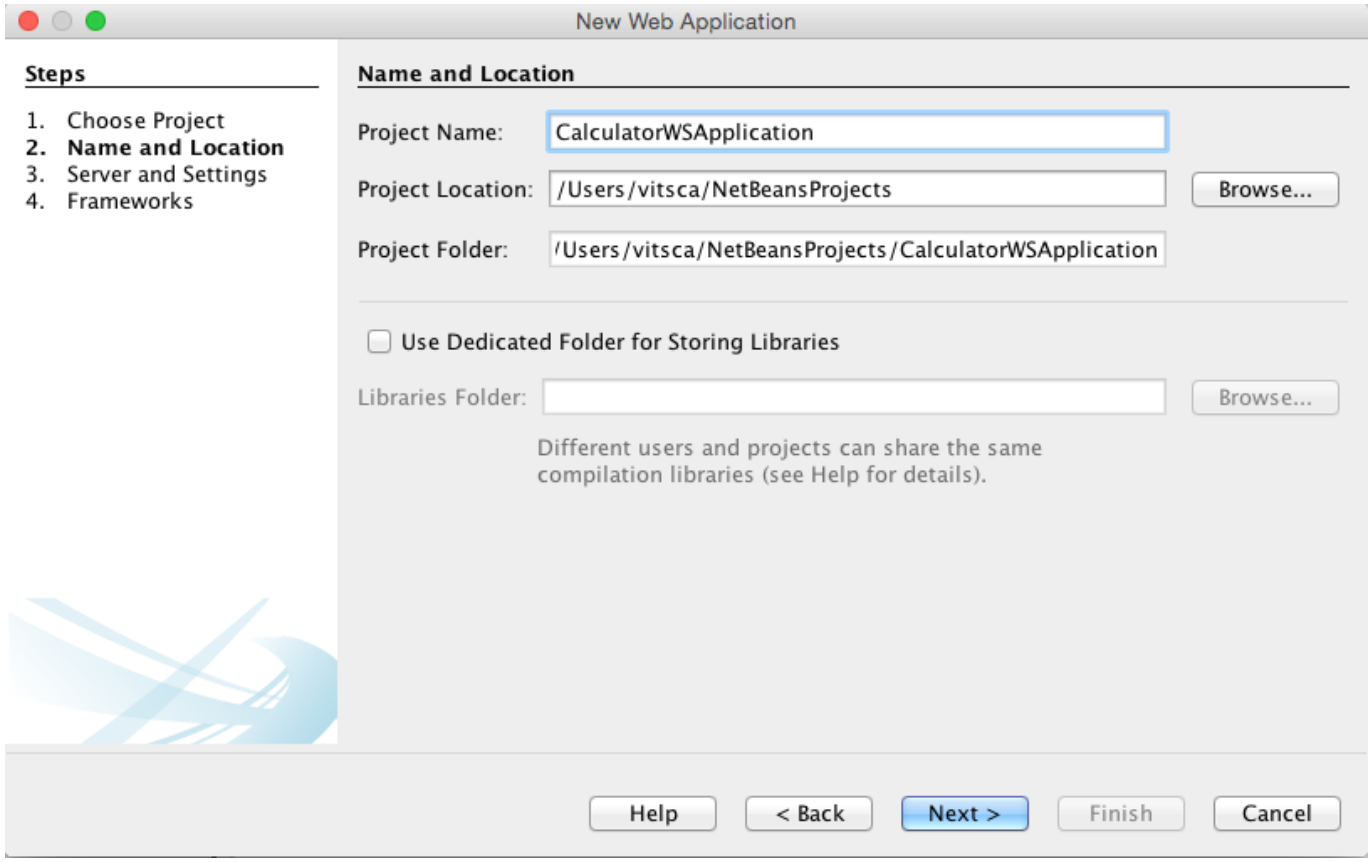
Plan

Plan

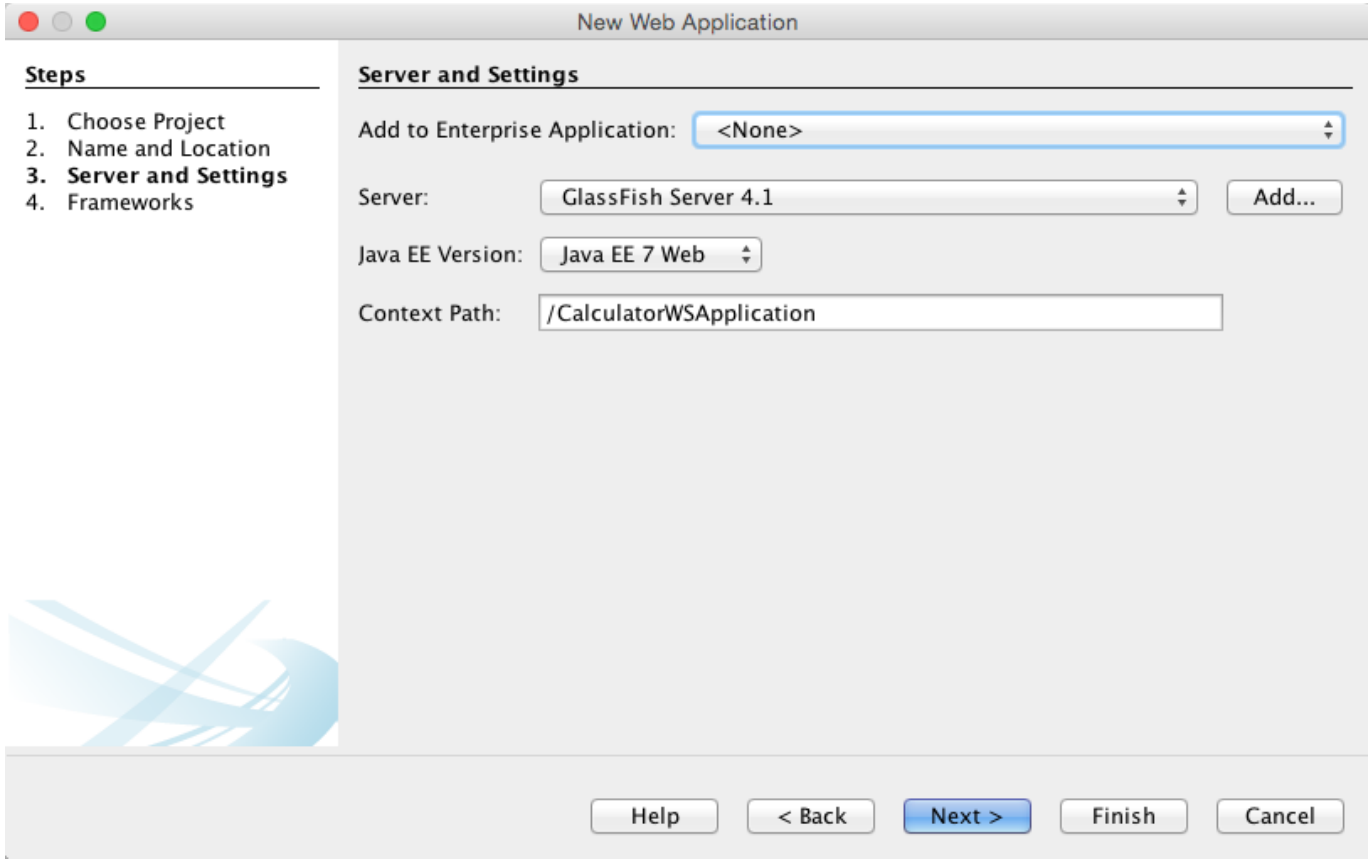
WS in Netbeans: Creazione di un progetto per WS



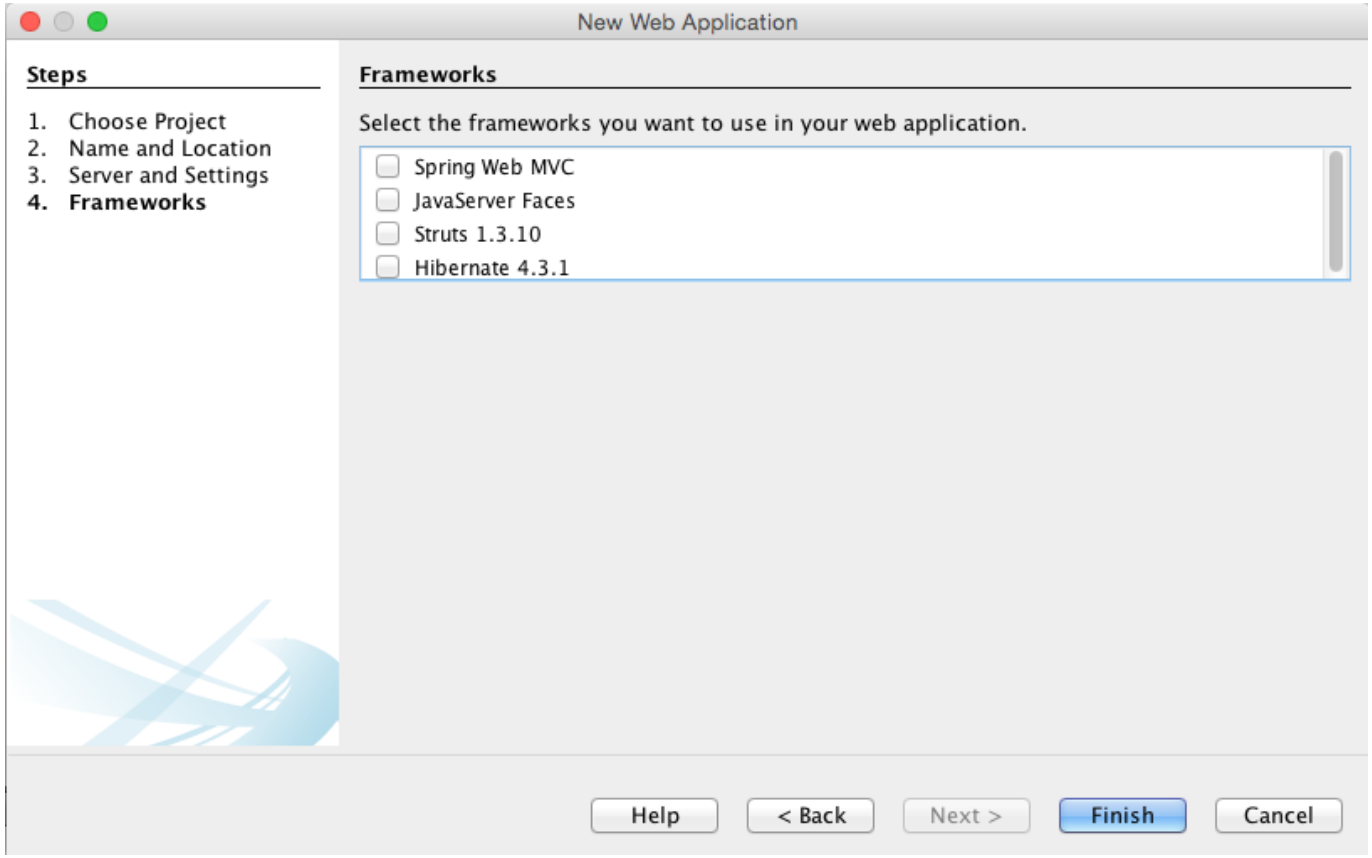
WS in NetBeans: Creazione progetto - 1



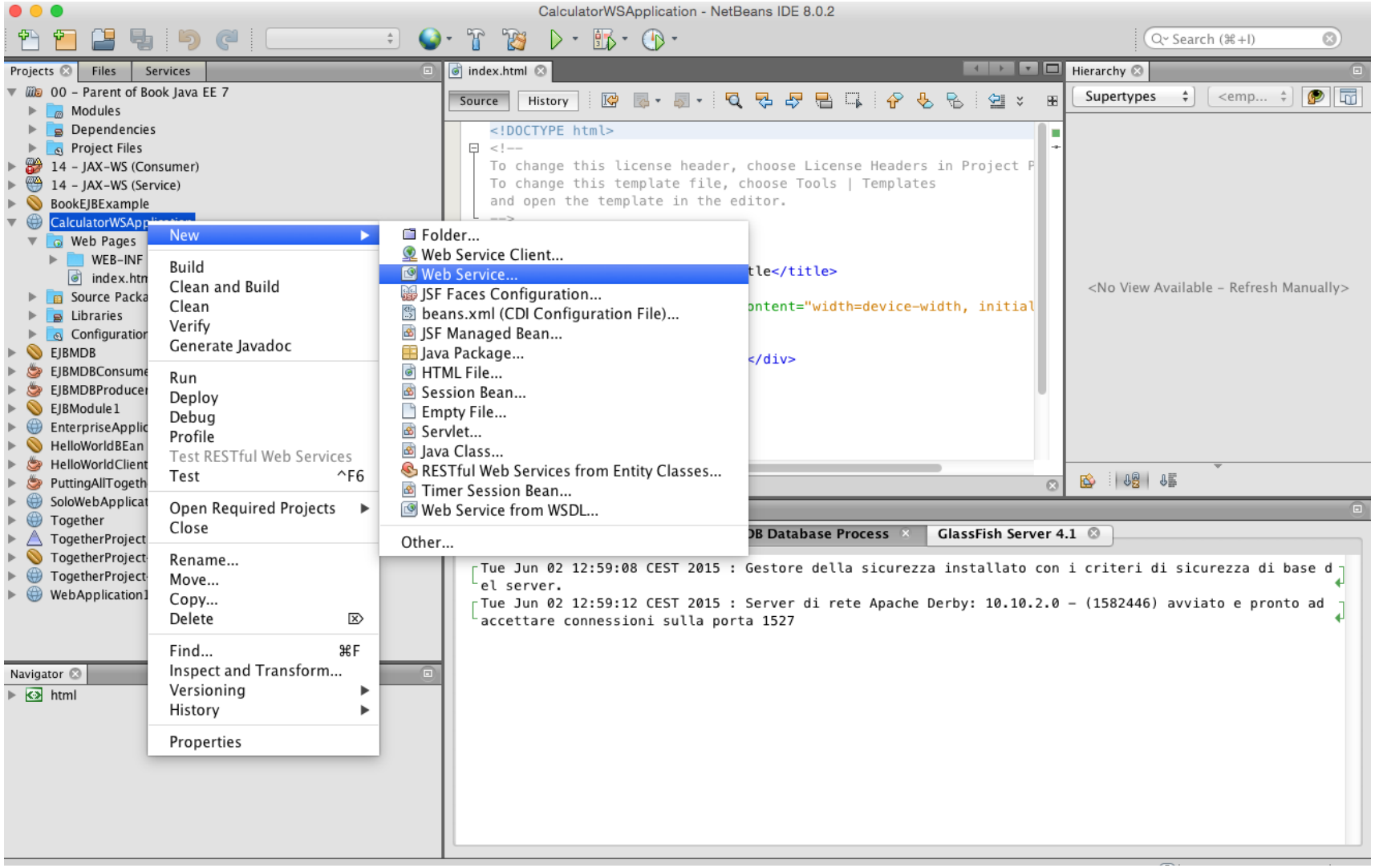
WS in Netbeans: Creazione progetto - 2



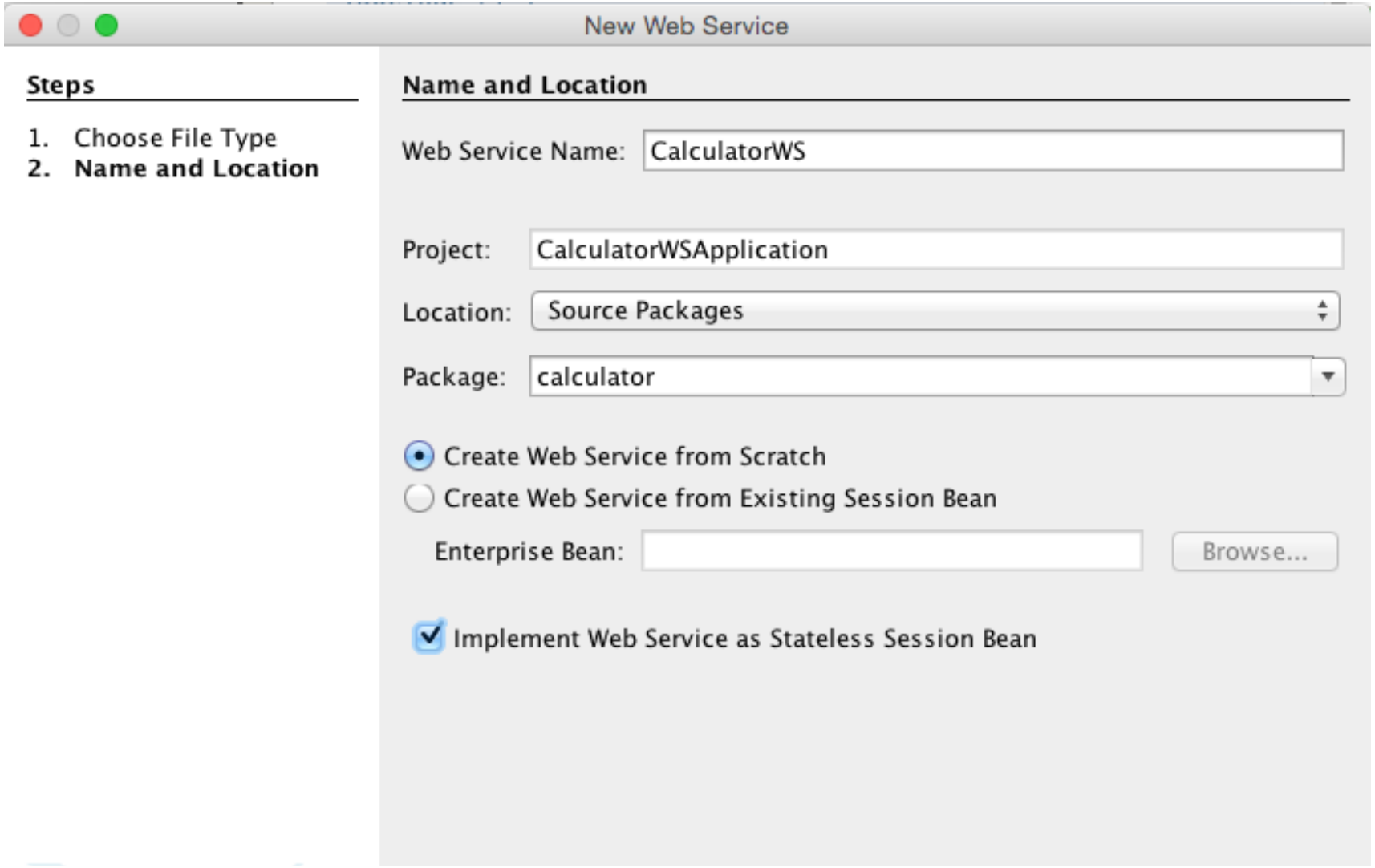
WS in Netbeans: Creazione progetto - 3



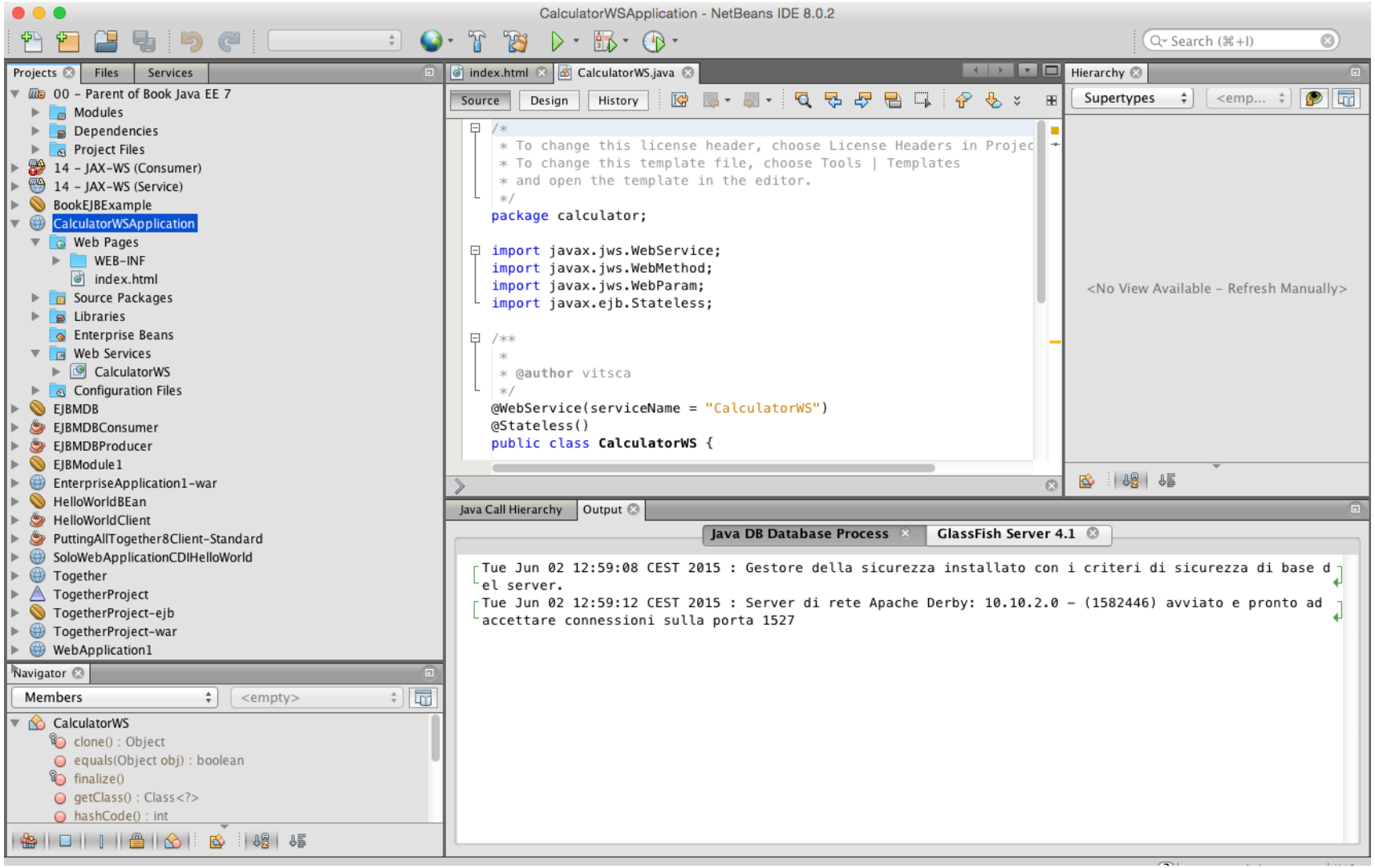
WS in Netbeans: Creazione di un WS - 1



WS in Netbeans: Creazione di un WS - 2



WS in Netbeans: La situazione



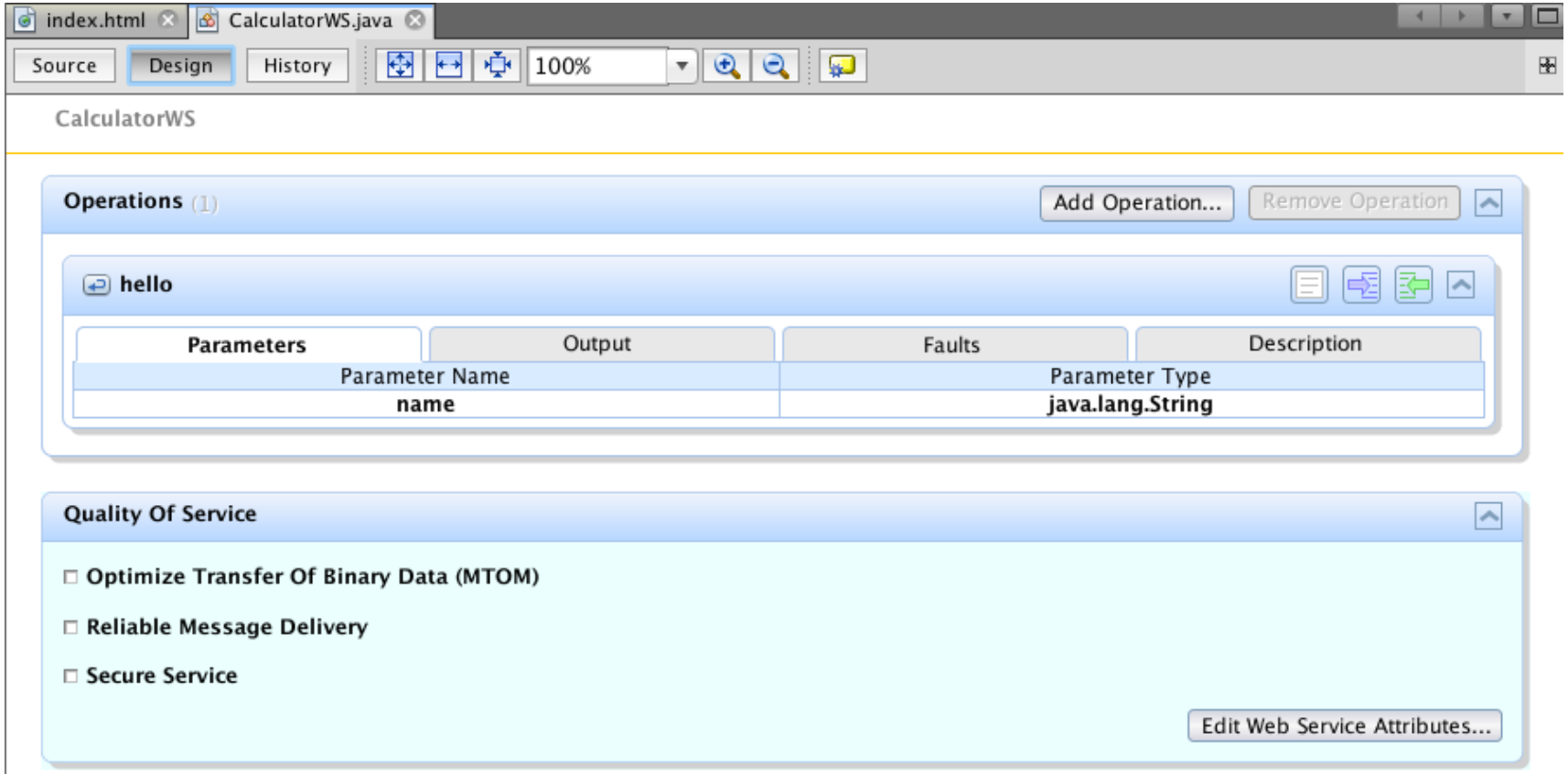
WS in Netbeans: Un WS di Hello

index.html | CalculatorWS.java

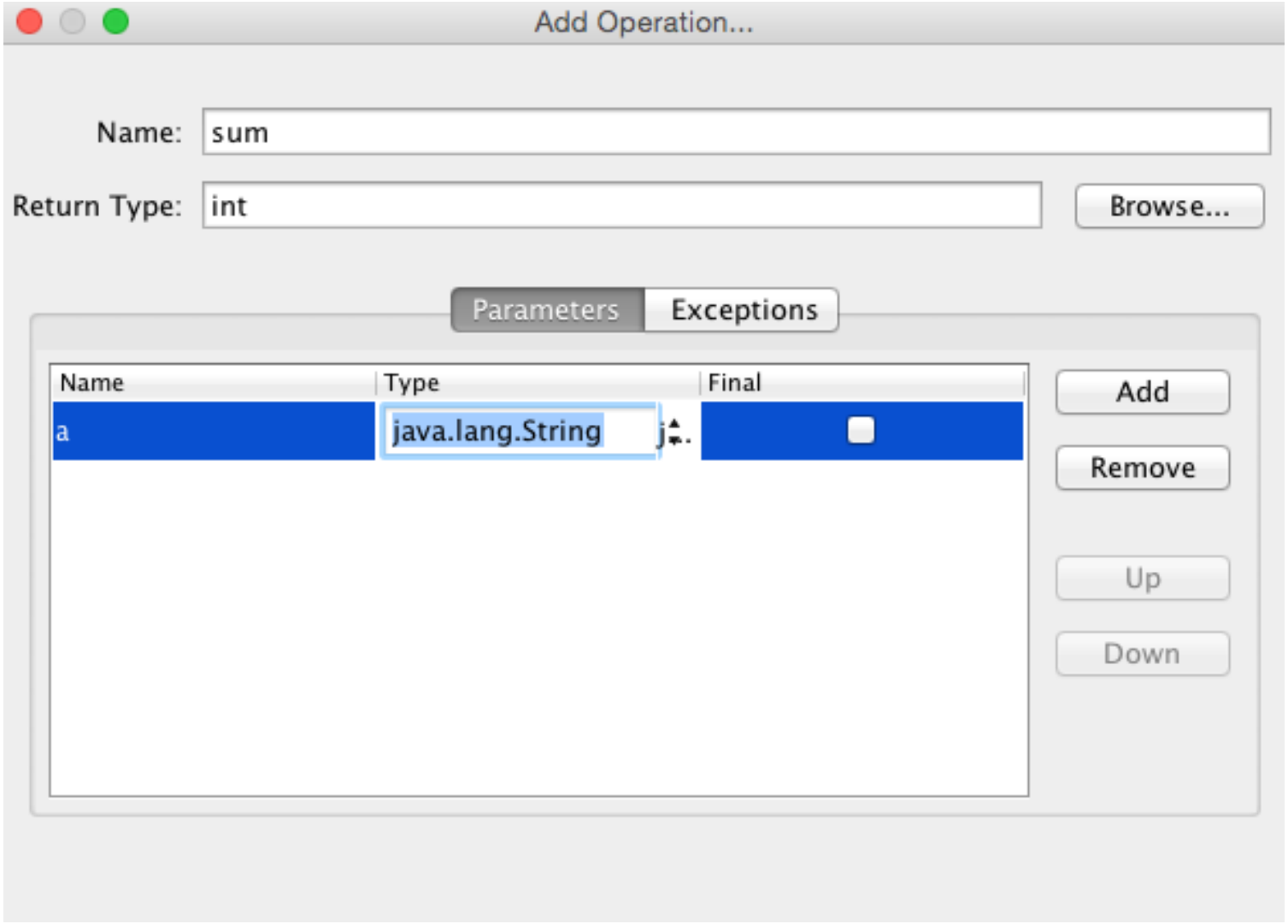
Source | Design | History

```
/*
 * 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 calculator;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.ejb.Stateless;

/**
 *
 * @author vitsca
 */
@WebService(serviceName = "CalculatorWS")
@Stateless()
public class CalculatorWS {

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

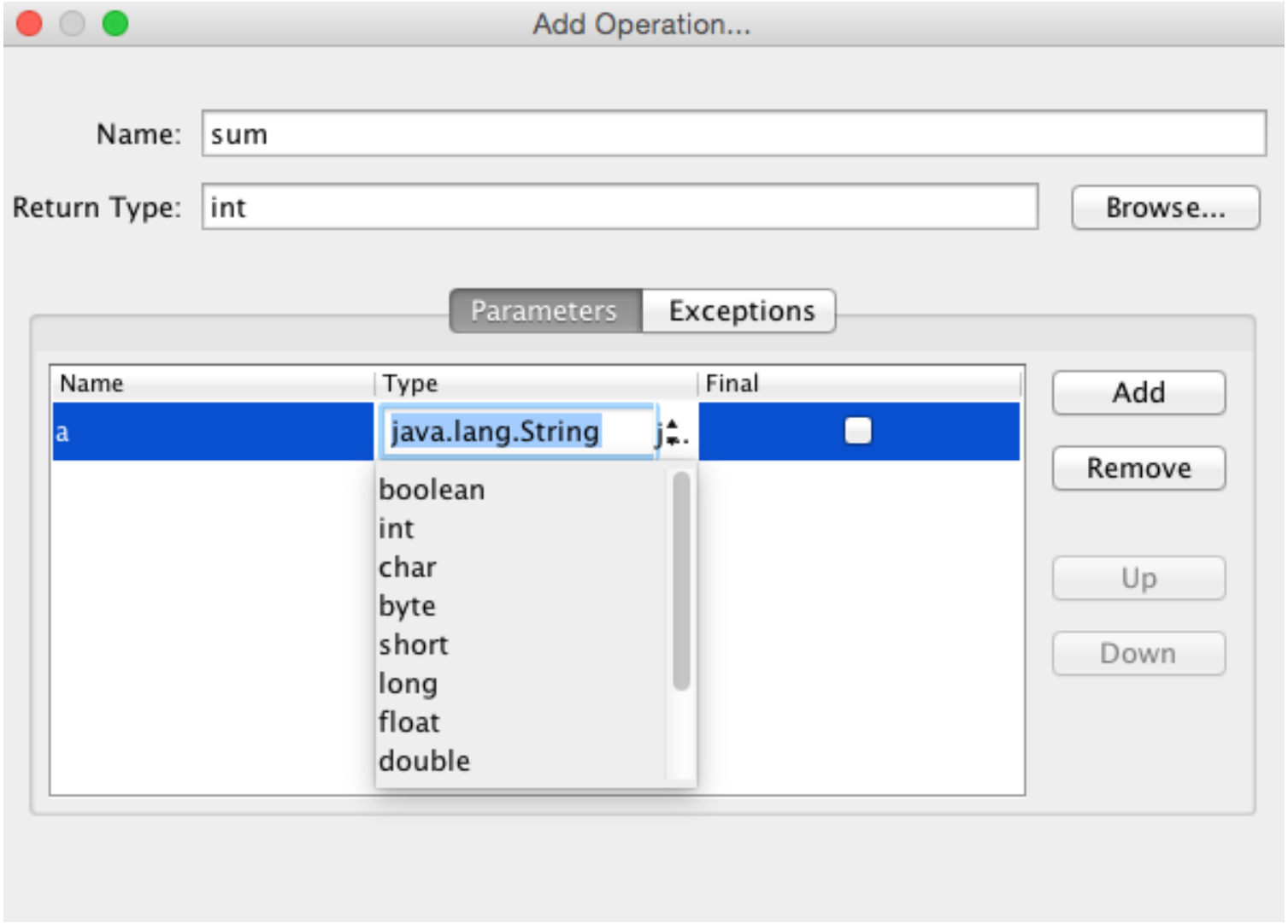

WS in Netbeans: Vista di Design



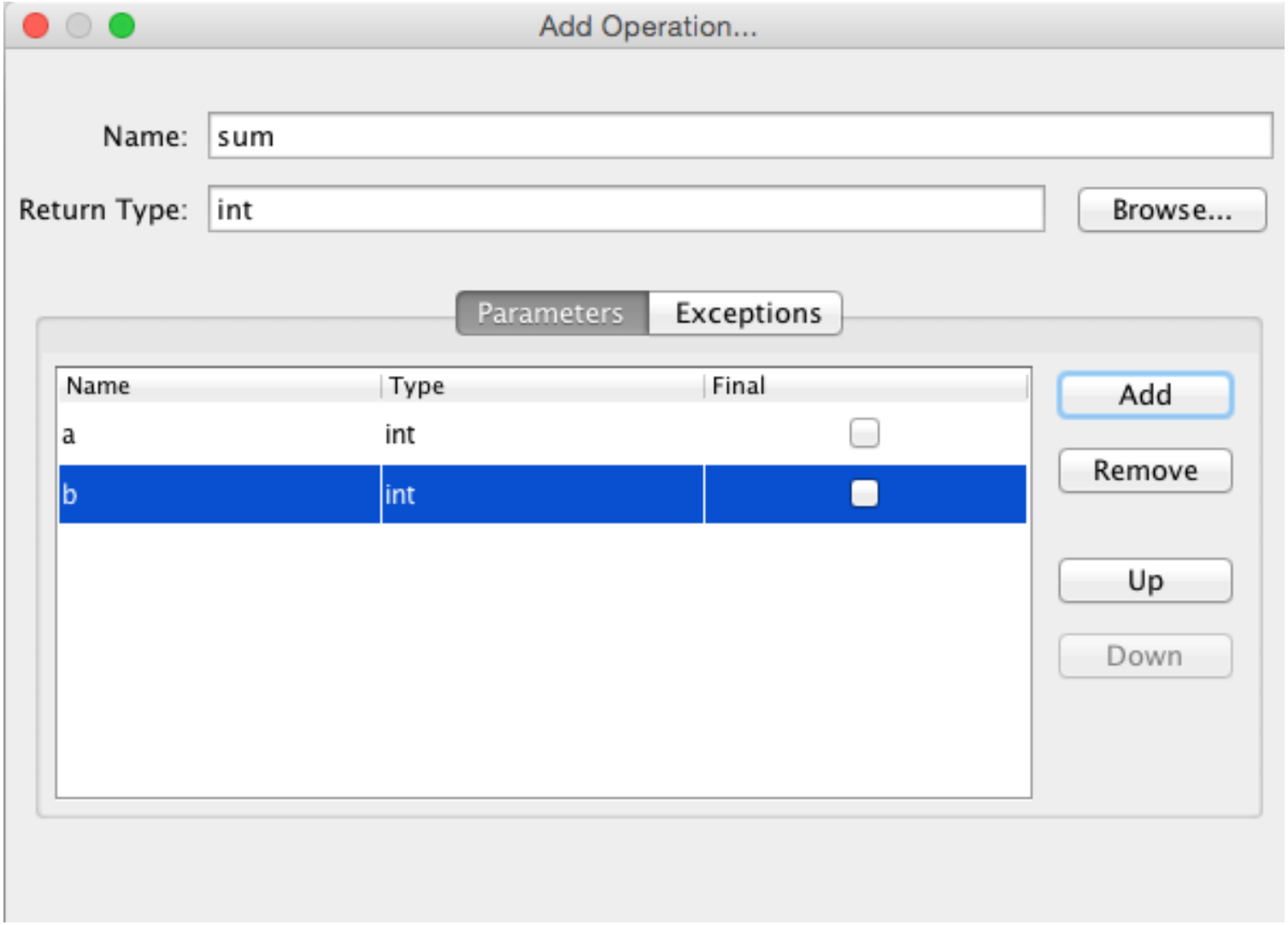
WS in Netbeans: Aggiungiamo una operazione



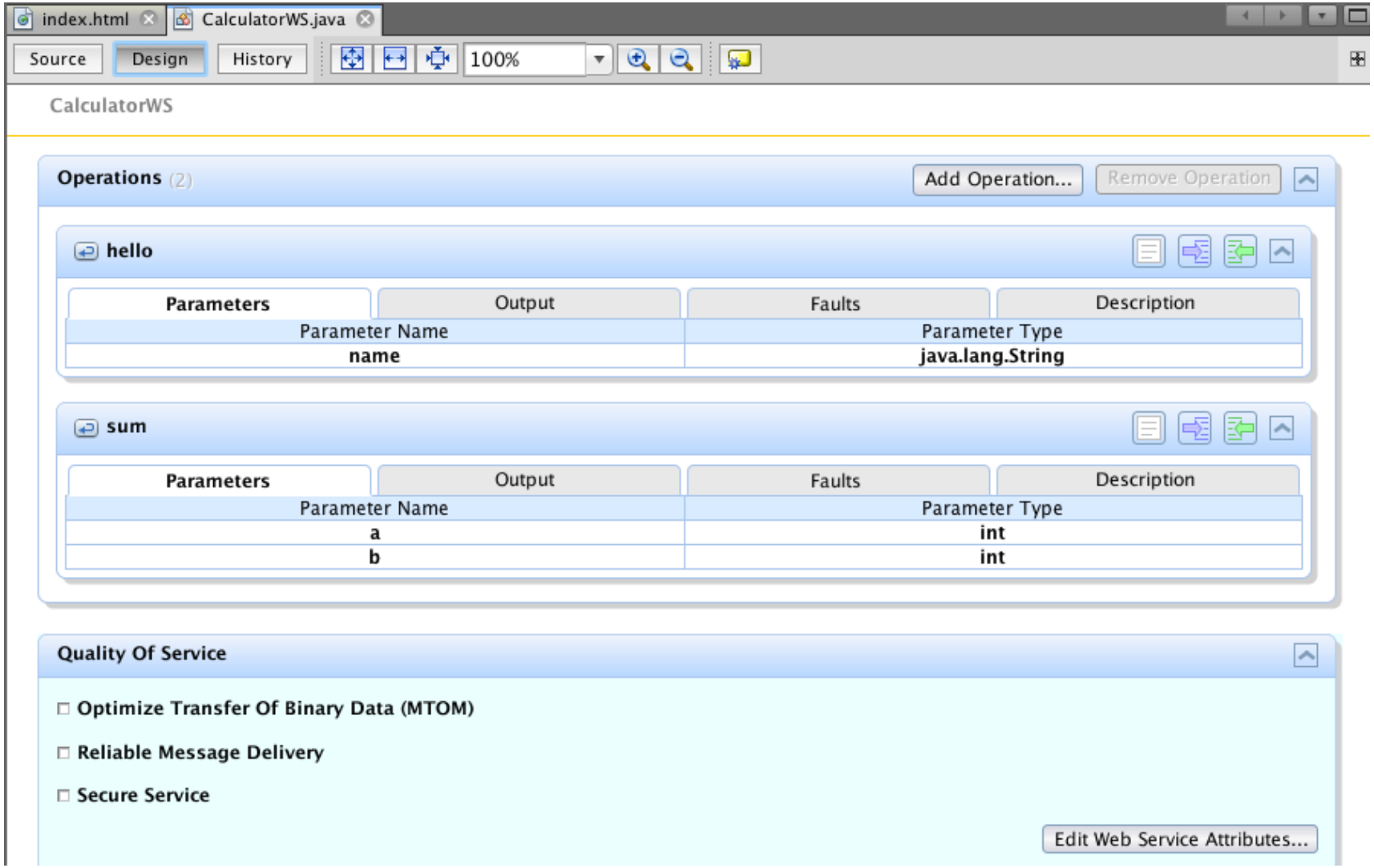
WS in Netbeans: Un parametro . . .



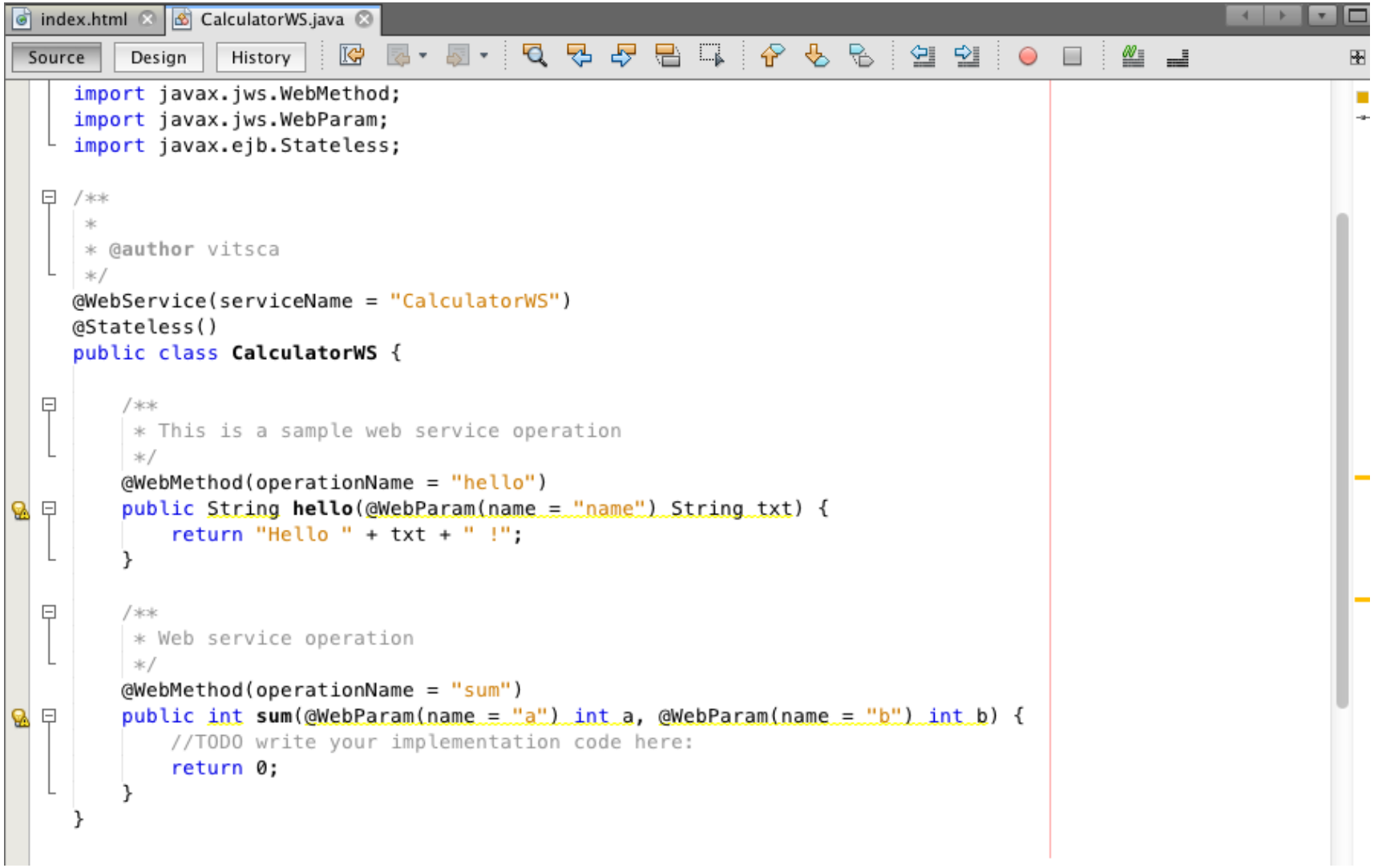
WS in Netbeans: Un secondo parametro



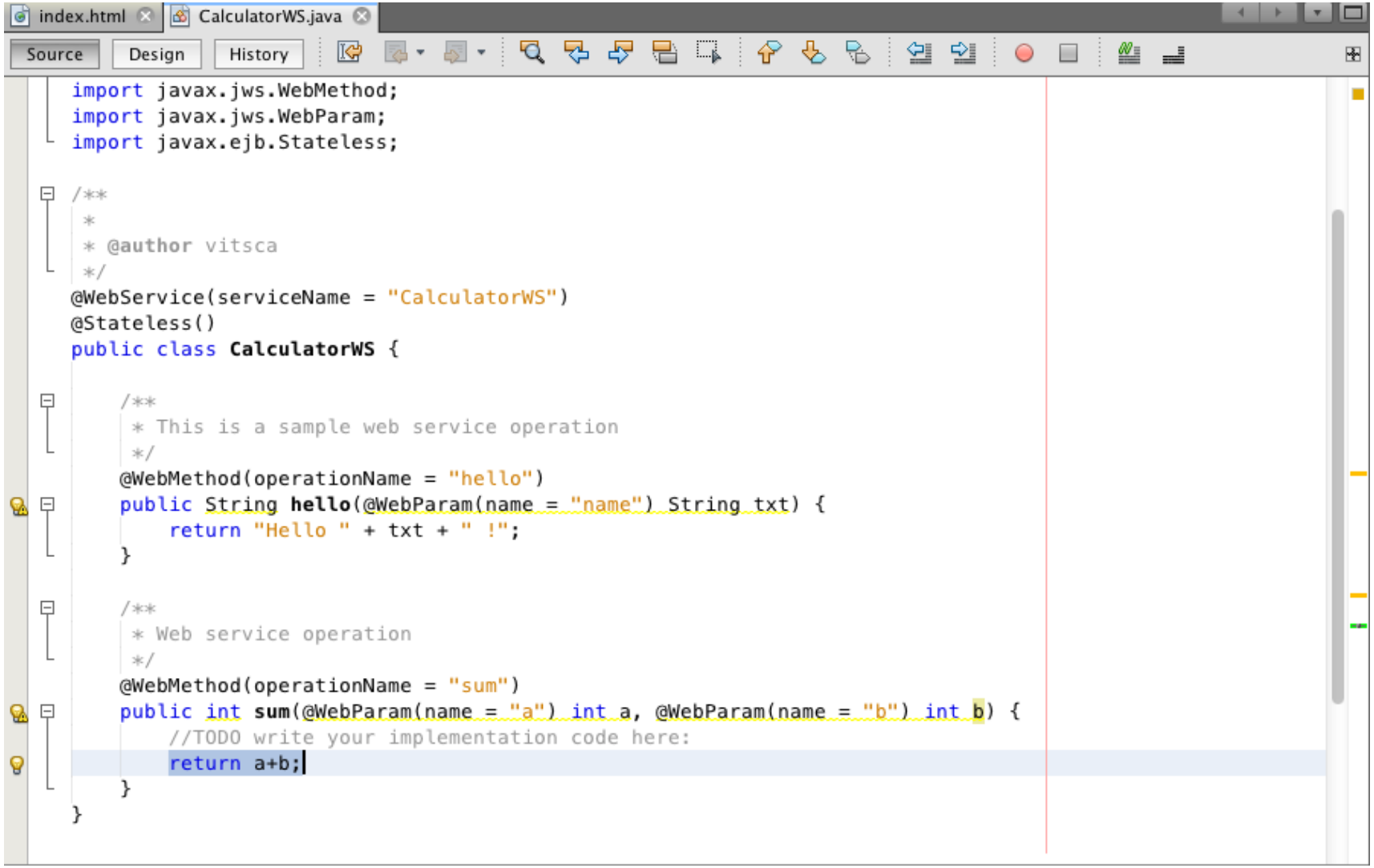
WS in Netbeans: La vista di design



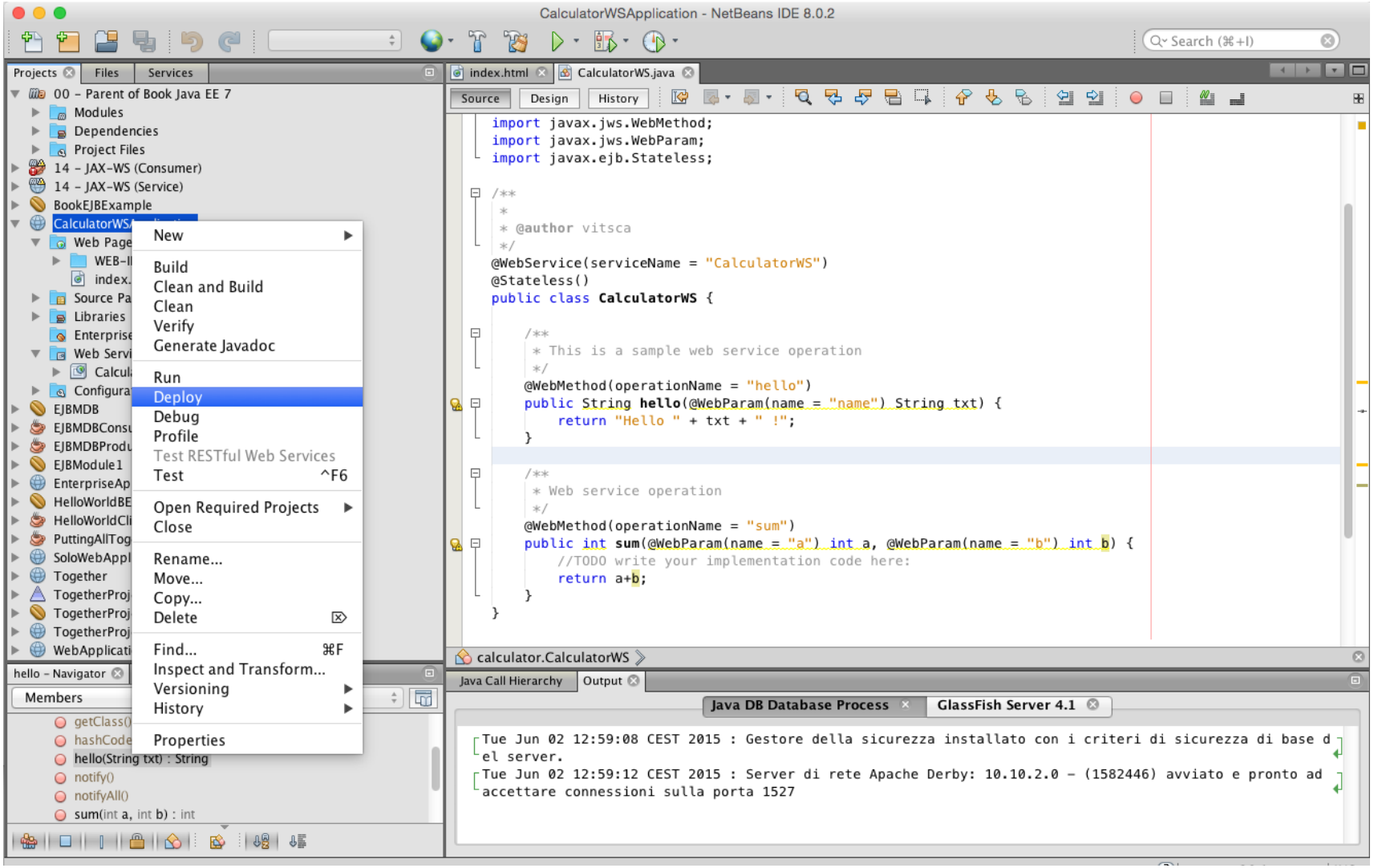
WS in Netbeans: Il codice



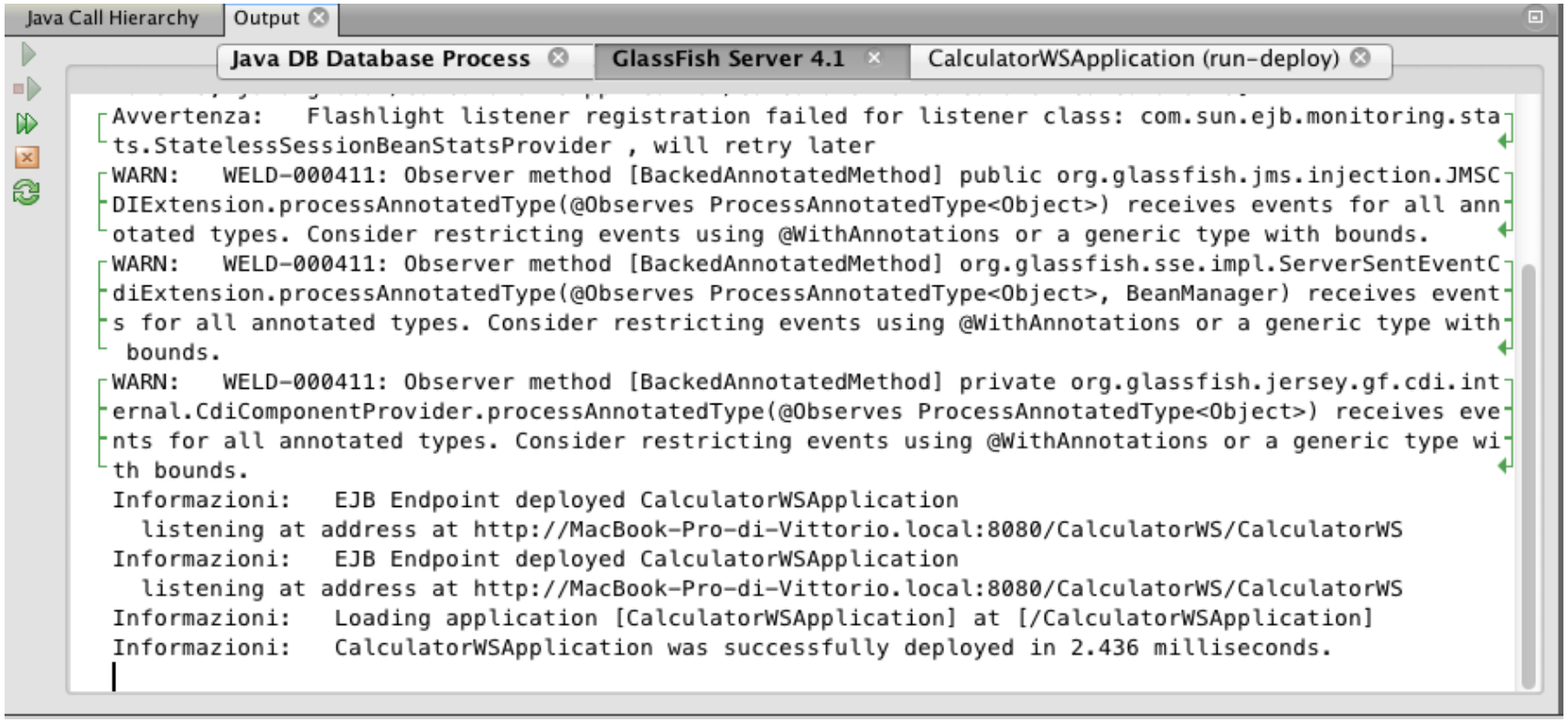
WS in Netbeans: scriviamo qualcosa!



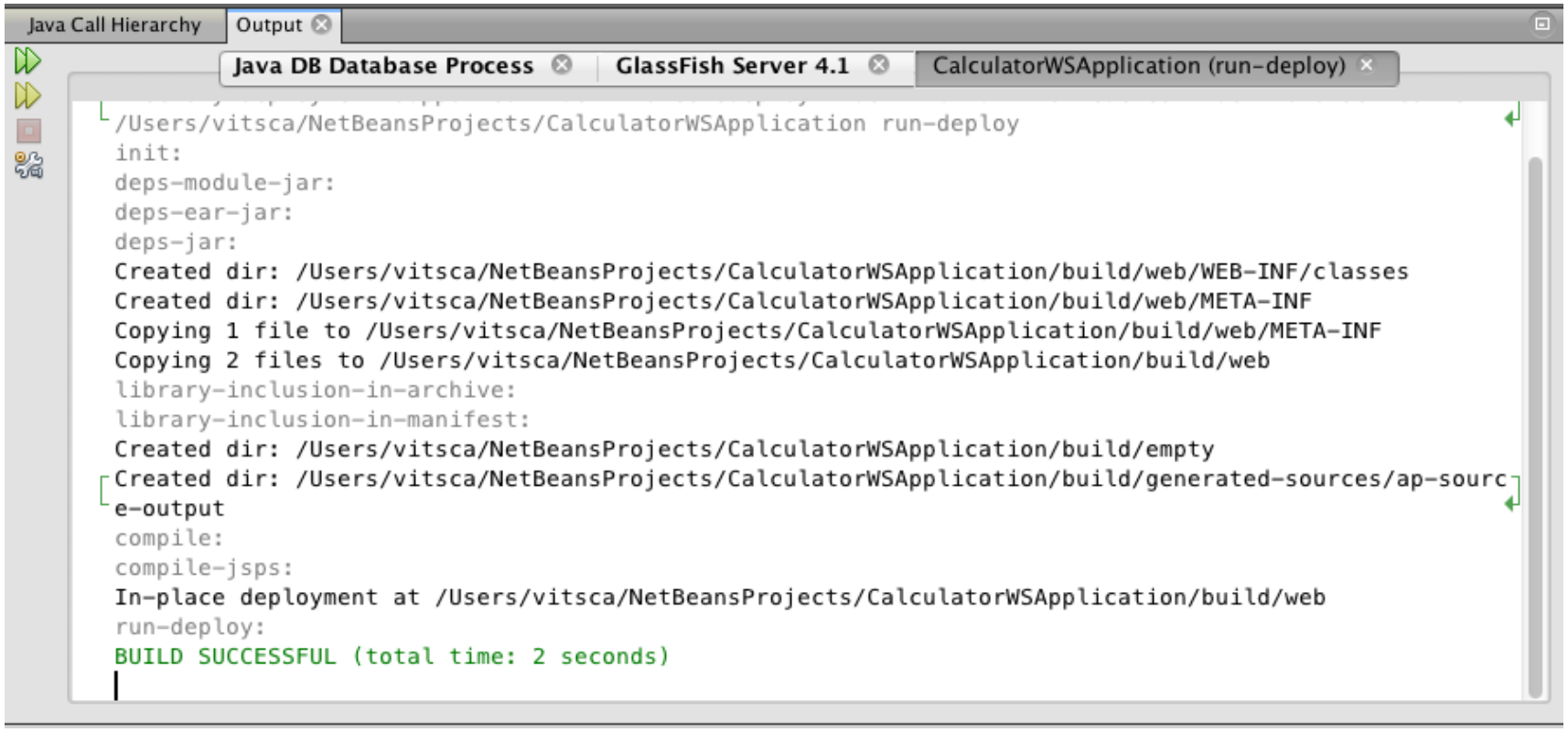
WS in Netbeans: Deploy



WS IN NETBEANS: SUL SERVER

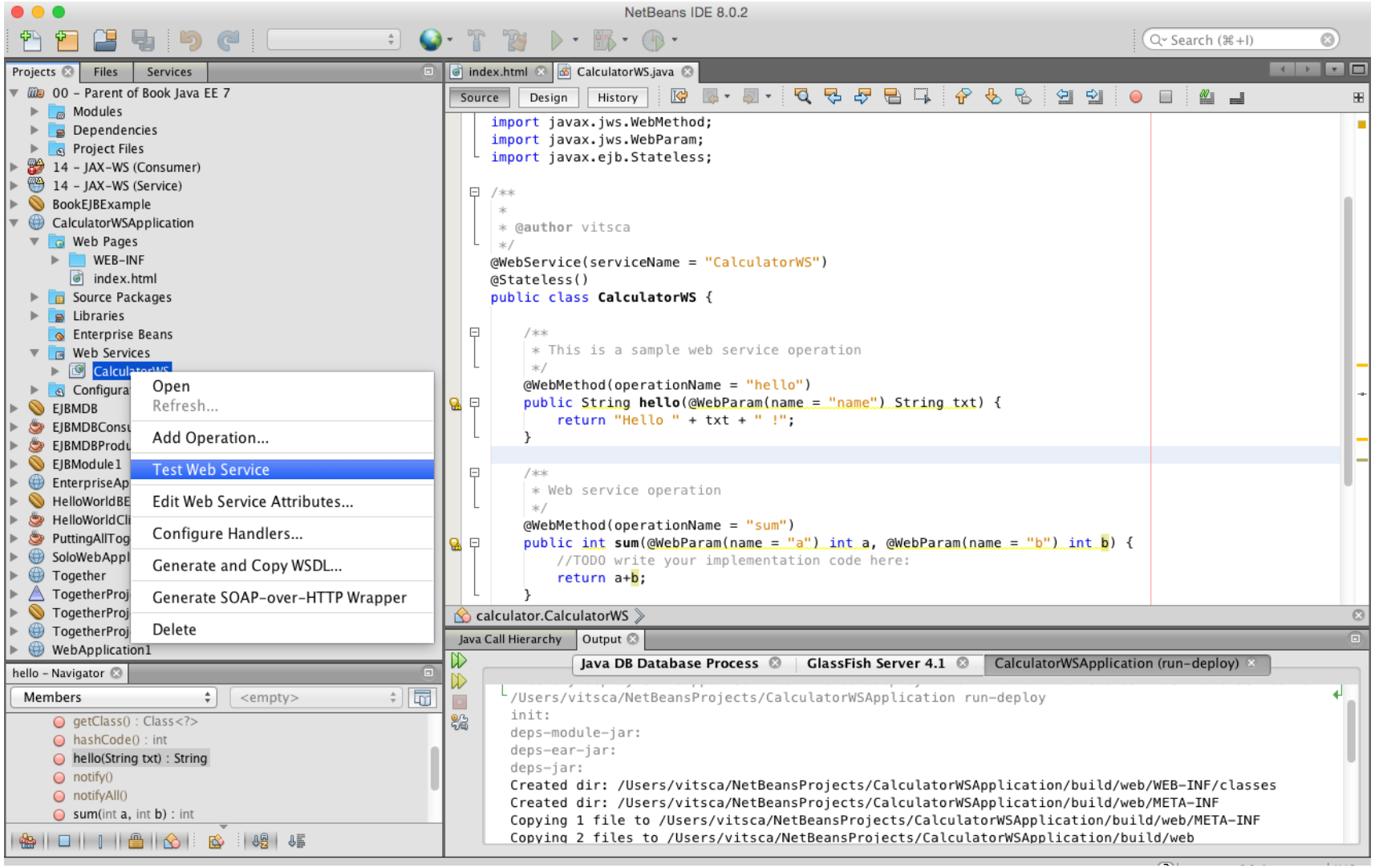


WS in Netbeans: al deploy cosa si fa

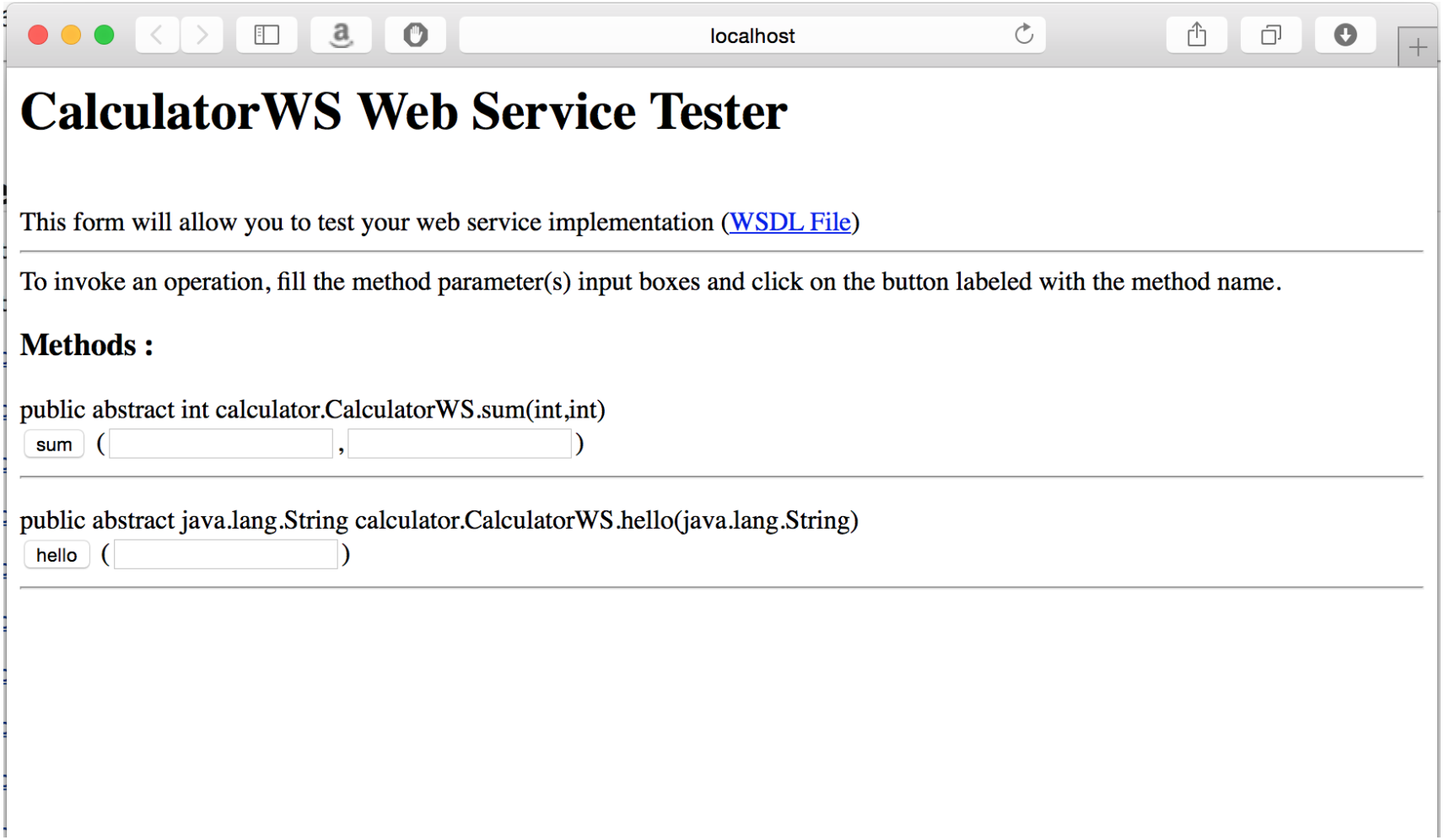


Plan

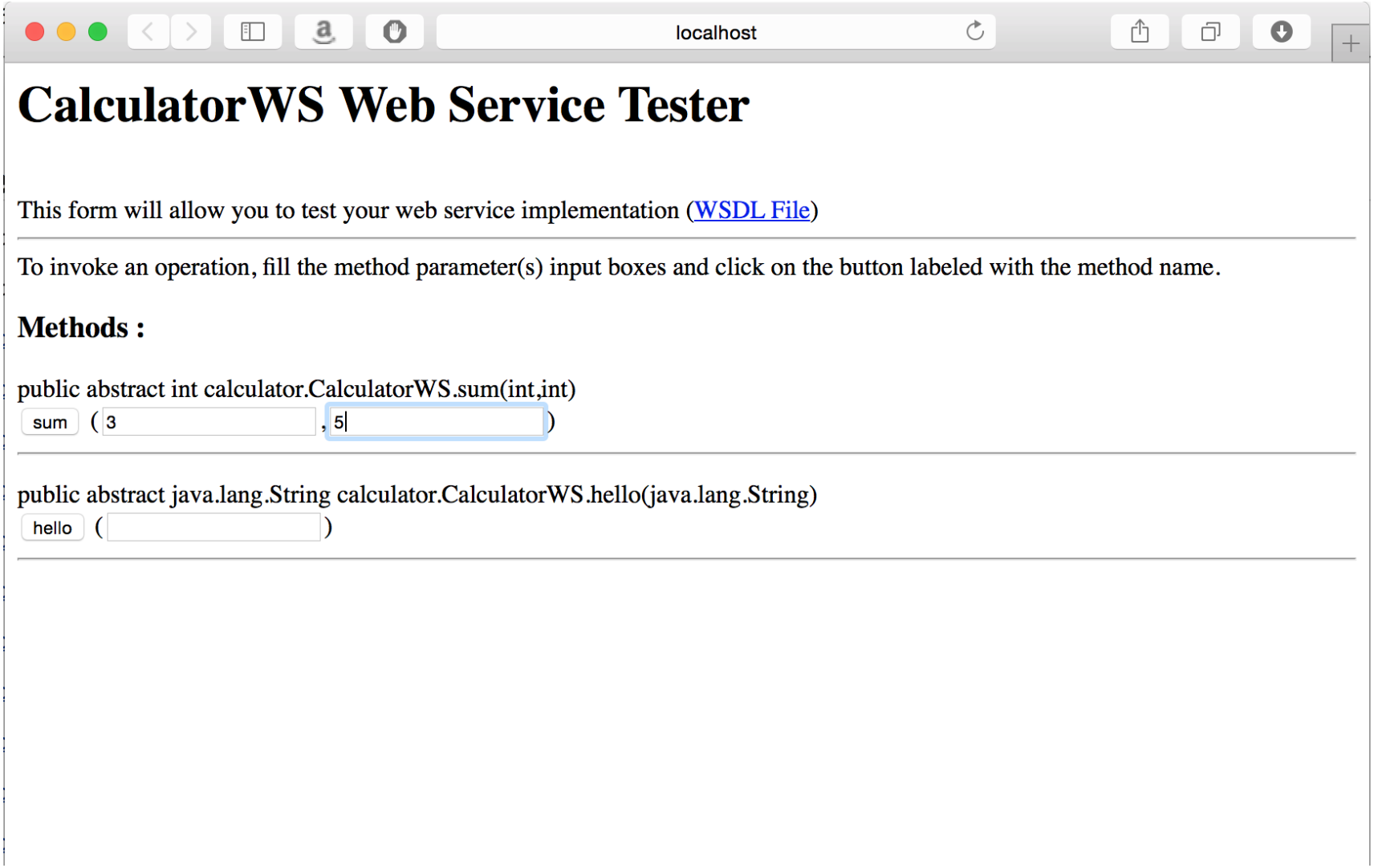
WS in Netbeans: Test - 1



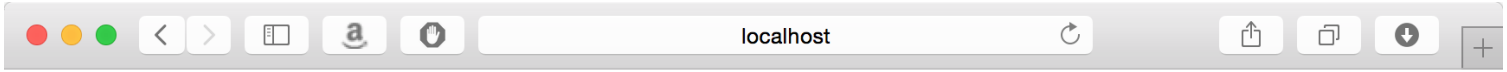
WS IN NETBEANS: TEST - 2



WS IN NETBEANS: TEST - 3



WS IN NETBEANS: TEST - 4



sum Method invocation

Method parameter(s)

Type	Value
int	3
int	5

Method returned

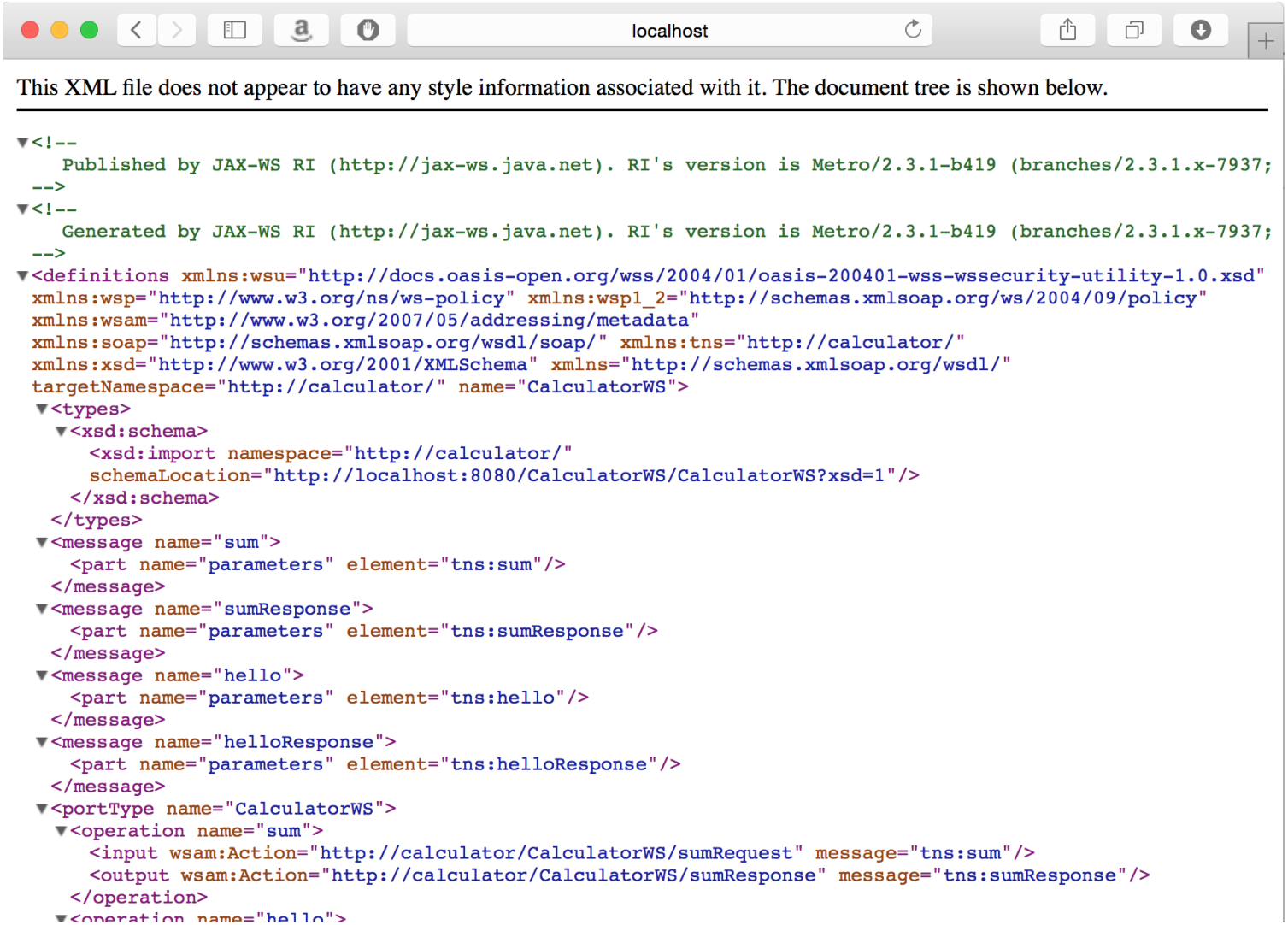
int : "**8**"

SOAP Request

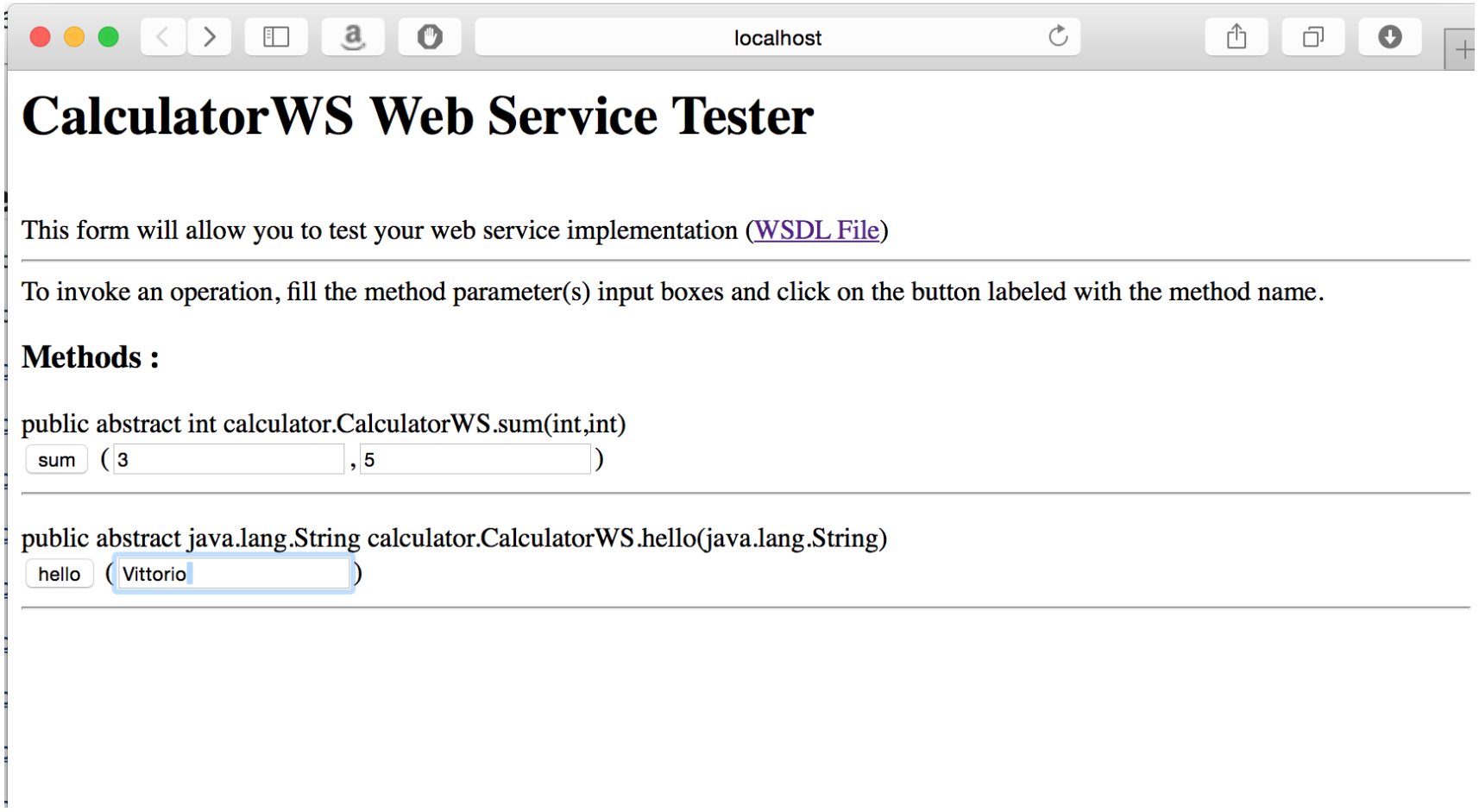
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xml
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:sum xmlns:ns2="http://calculator/">
            <a>3</a>
            <b>5</b>
        </ns2:sum>
    </S:Body>
</S:Envelope>
```

SOAP Response

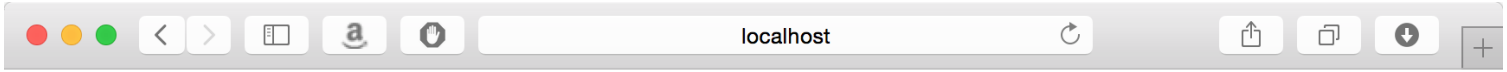
WS IN NETBEANS: TEST - 5



WS IN NETBEANS: TEST - 6



WS IN NETBEANS: TEST - 7



hello Method invocation

Method parameter(s)

Type	Value
java.lang.String	Vittorio

Method returned

java.lang.String : "**Hello Vittorio !**"

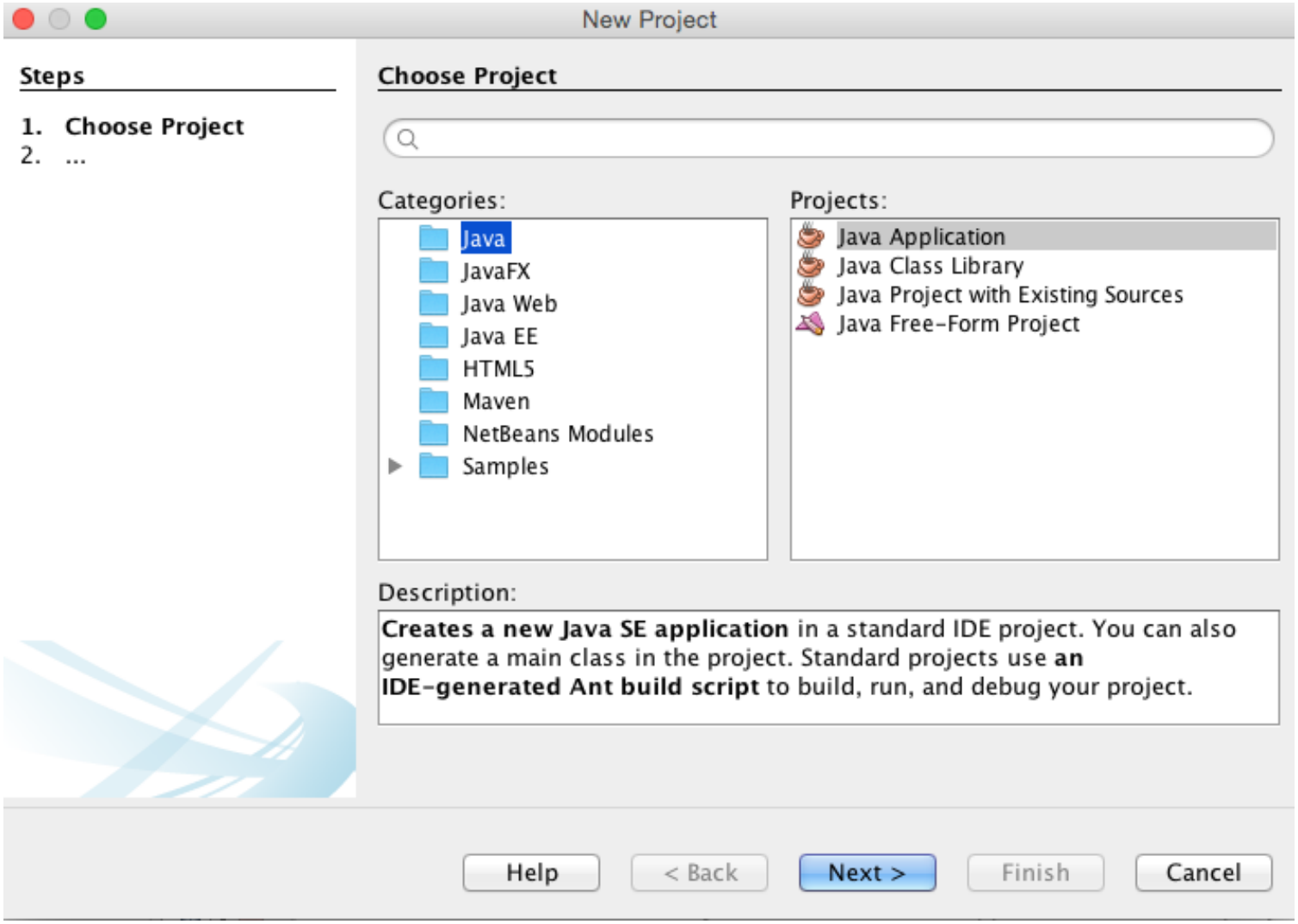
SOAP Request

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

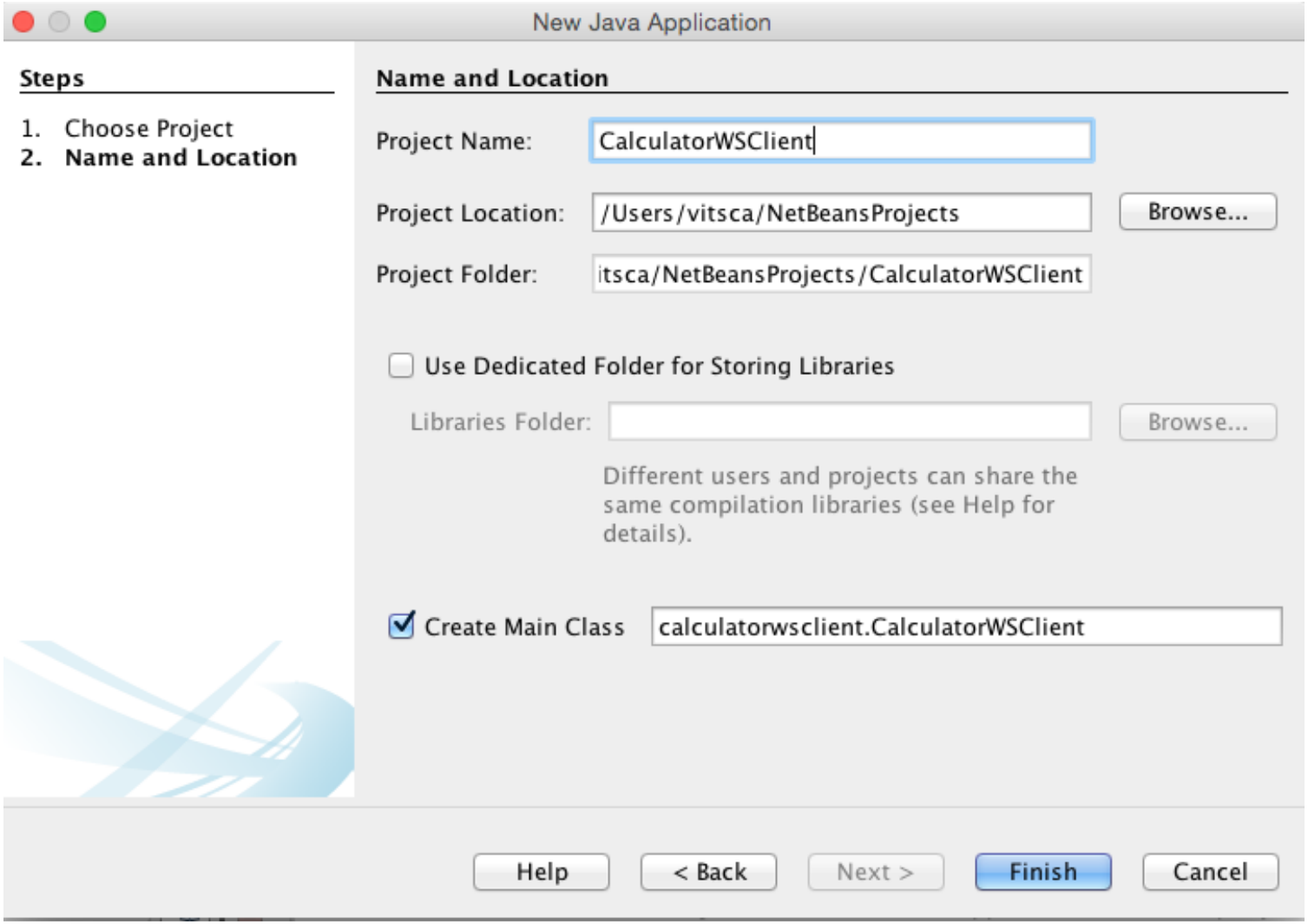
SOAP Response

Plan

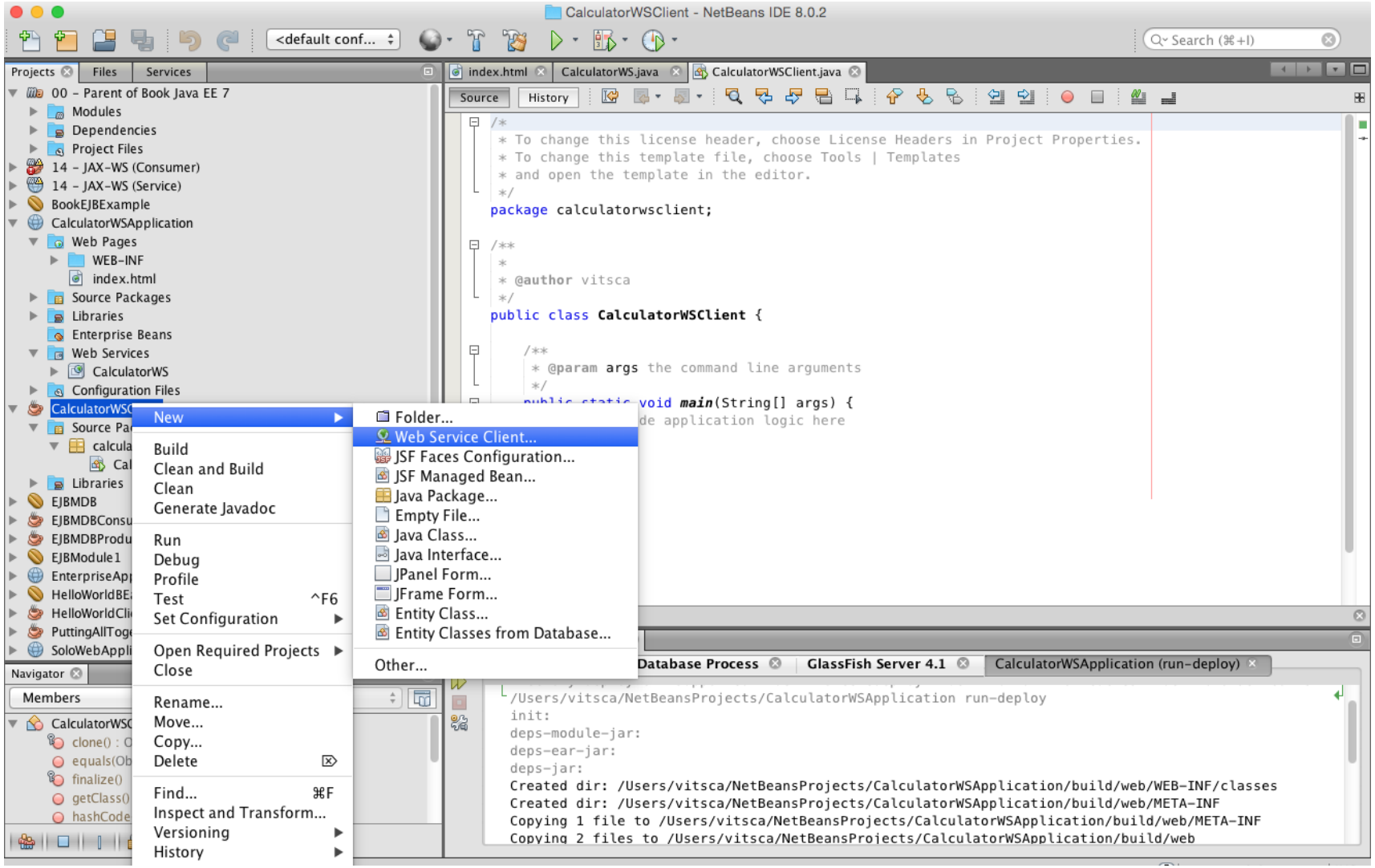
WS in Netbeans: Client - 1



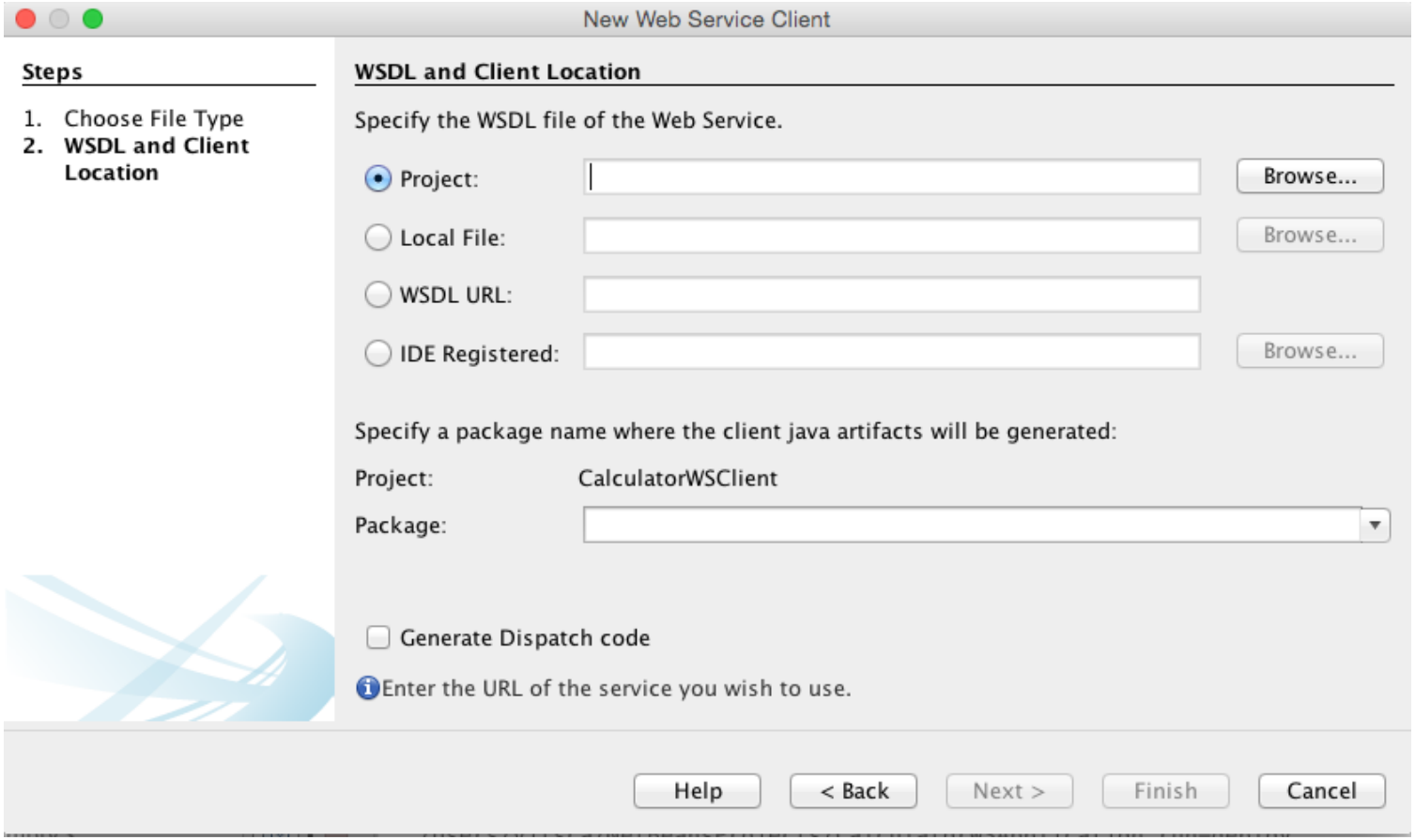
WS in NetBeans: Client - 2



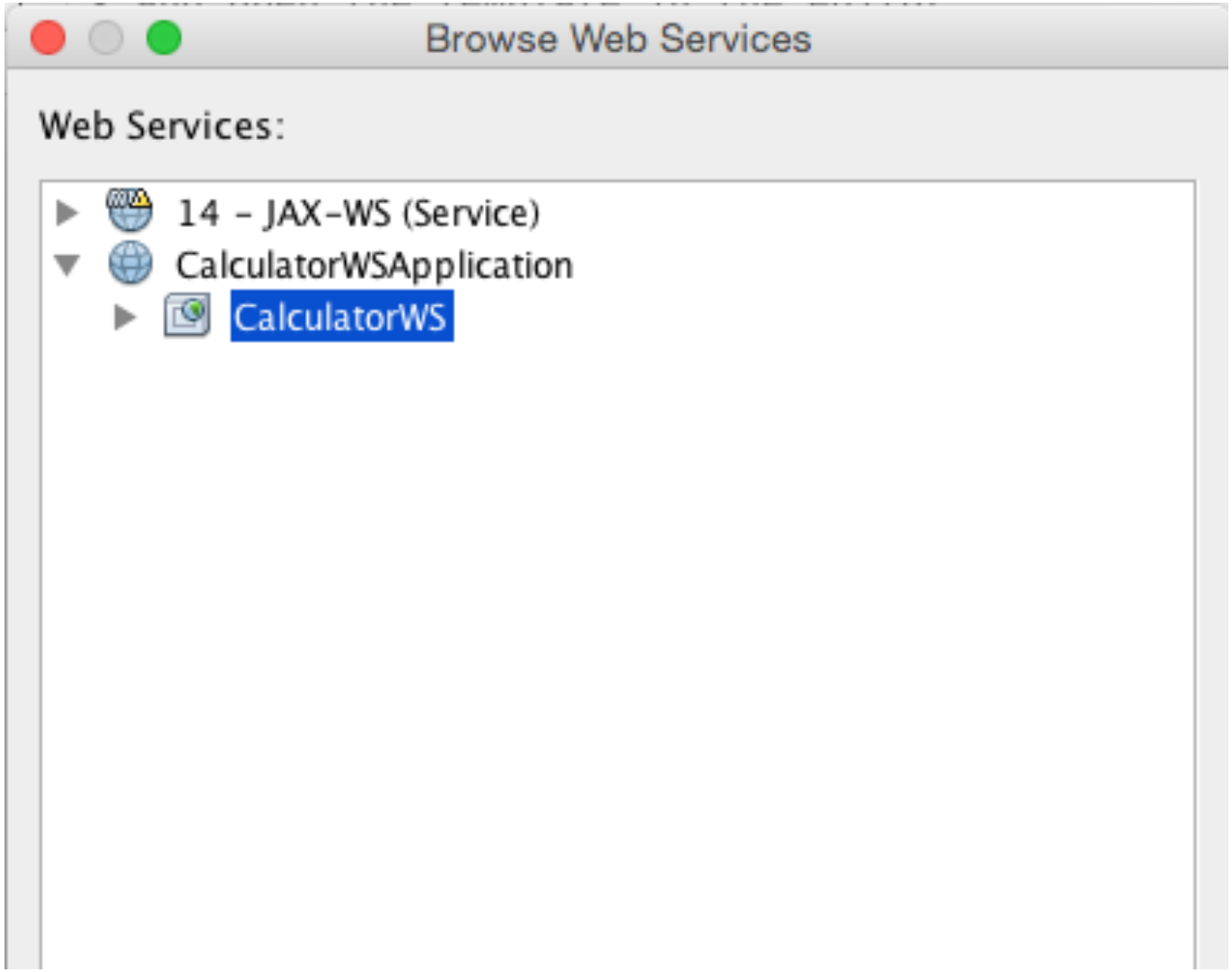
WS in Netbeans: Creiamo un WS Client - 1



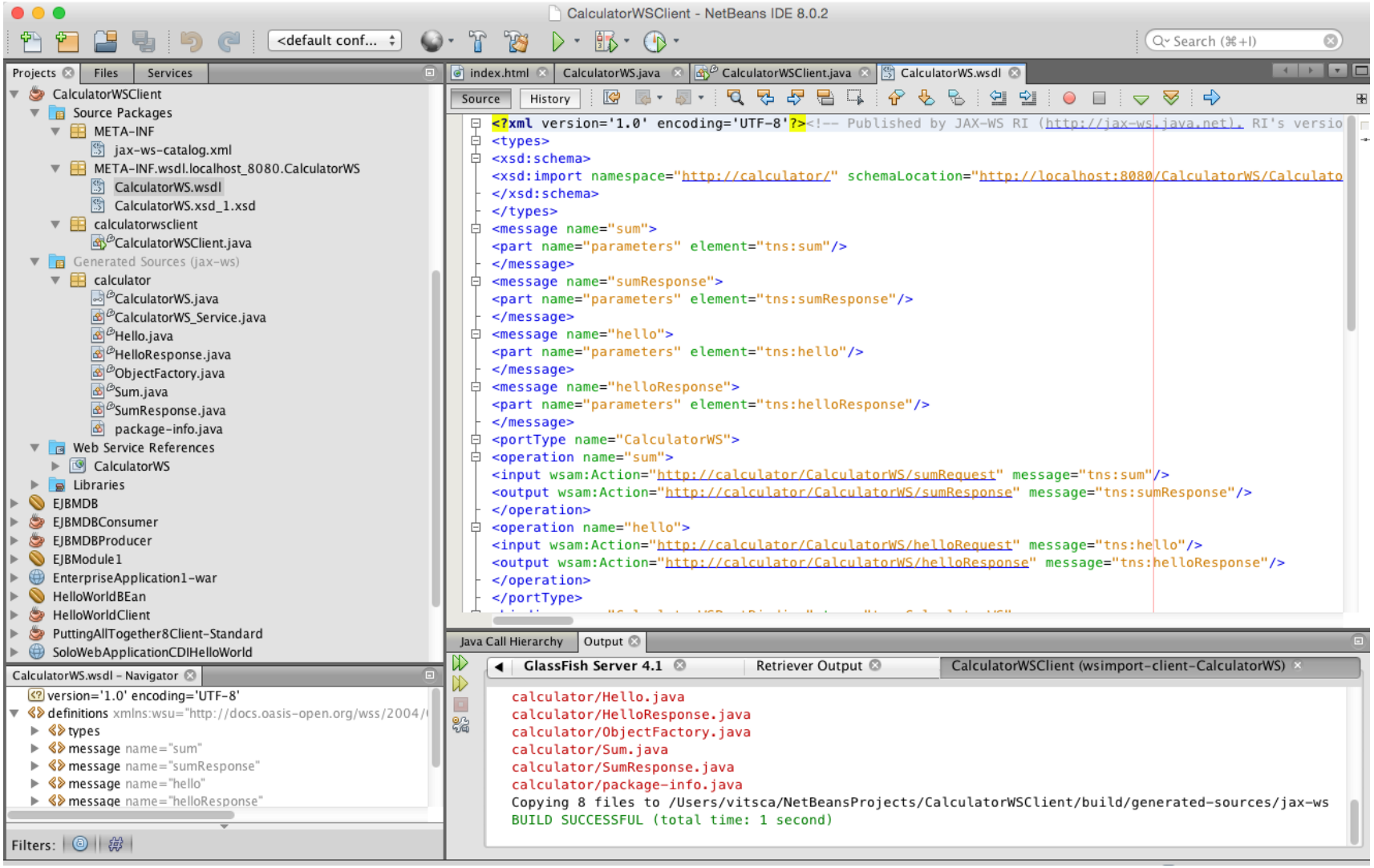
WS in Netbeans: Creiamo un WS Client - 2



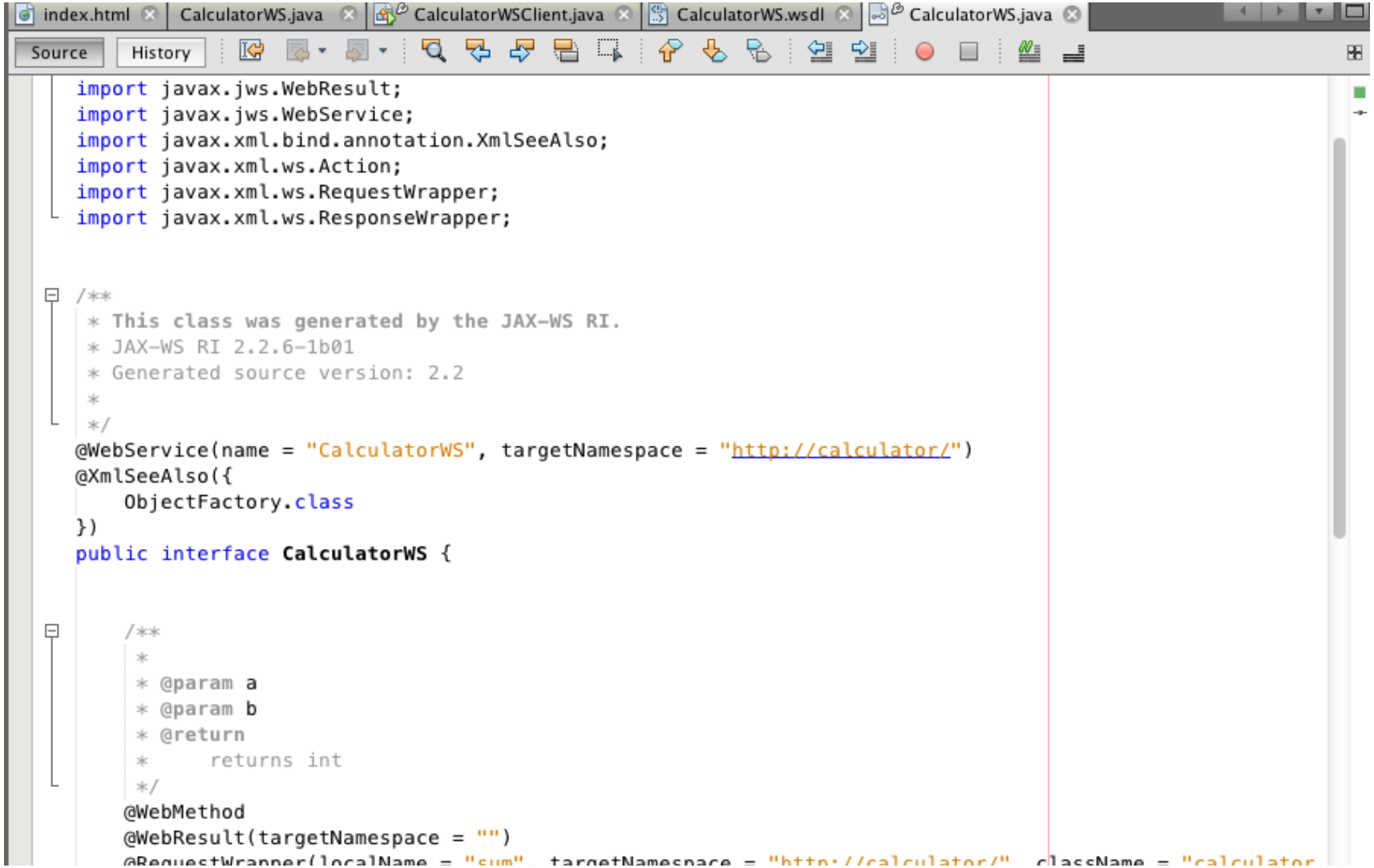
WS in Netbeans: Creiamo un WS Client - 3



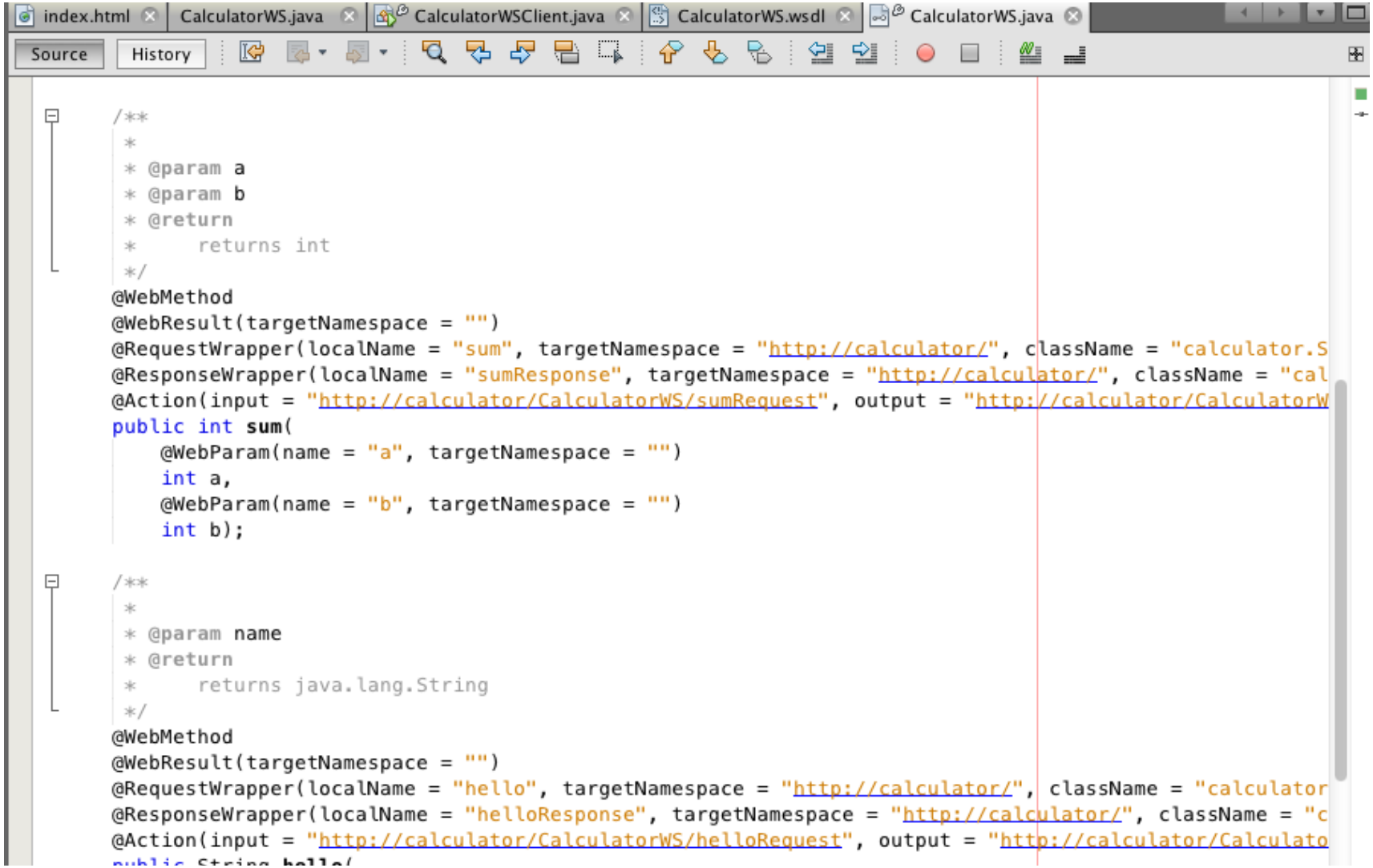
WS in Netbeans: Il file WSDL



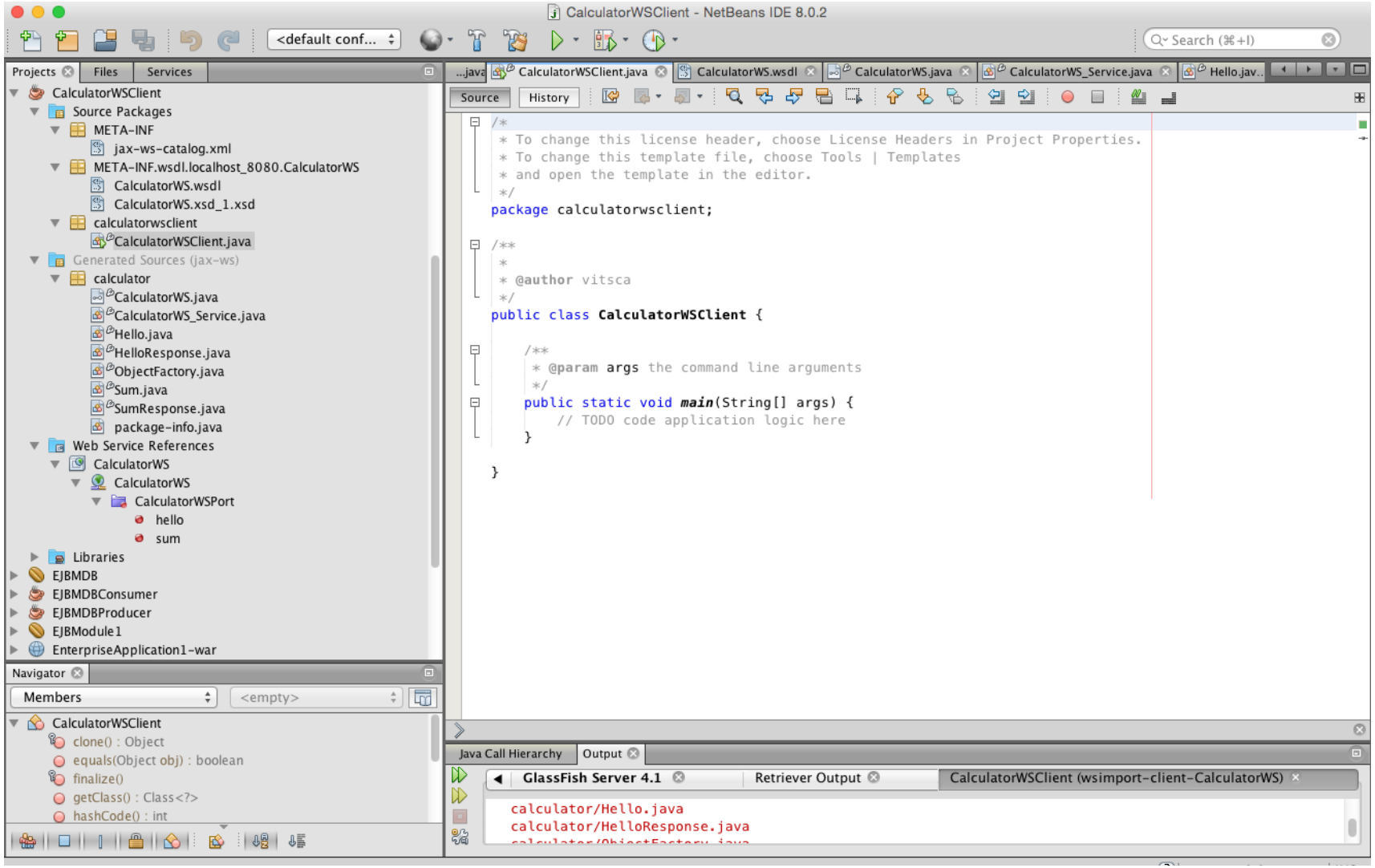
WS in Netbeans: L'interfaccia generata - 1



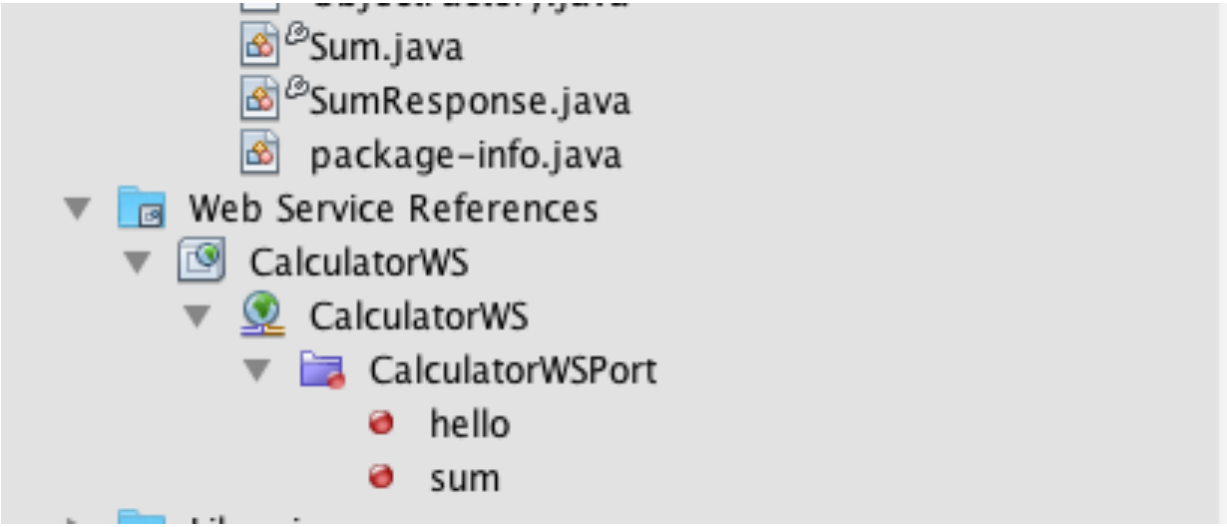
WS in Netbeans: L'interfaccia generata - 2



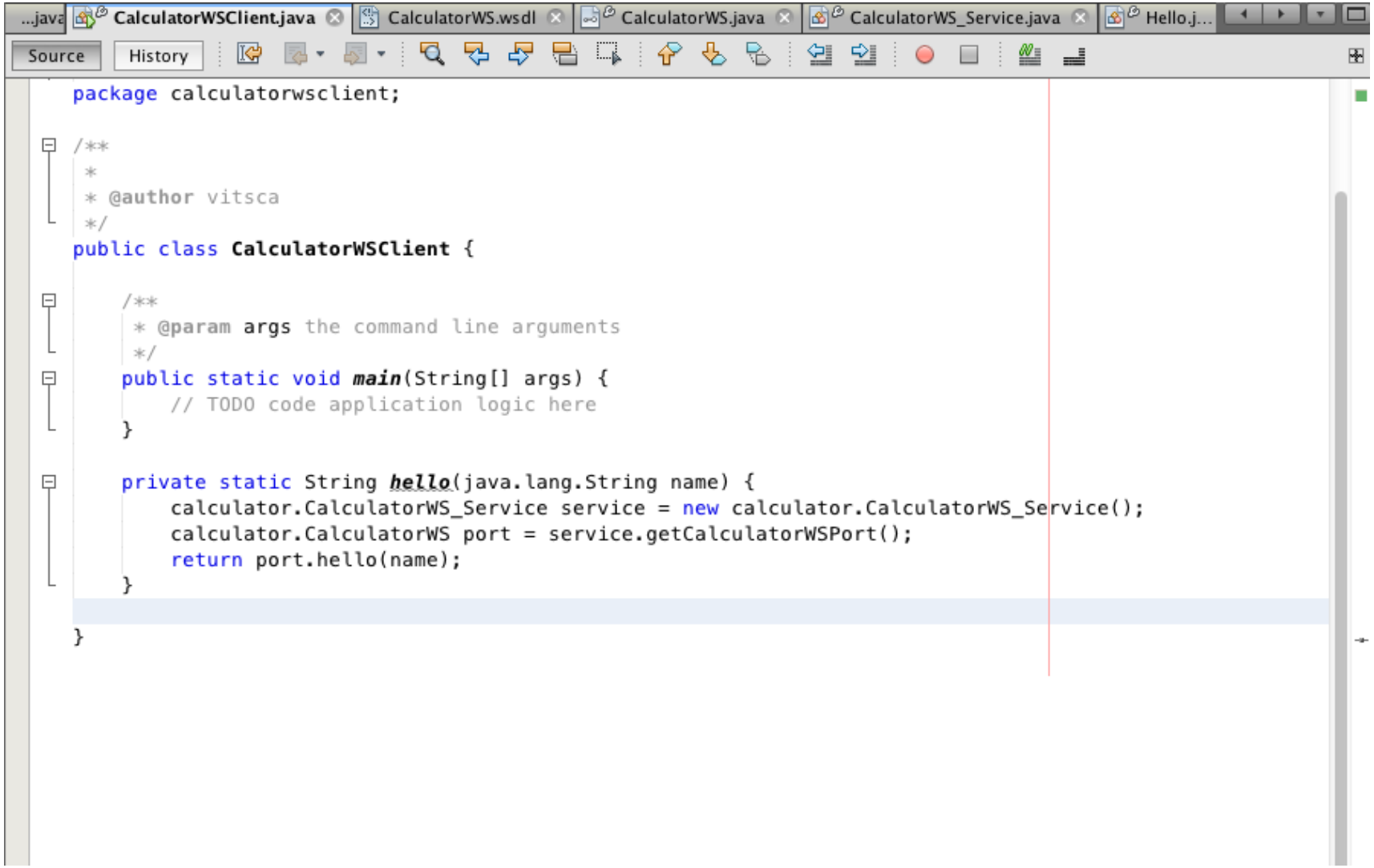
WS in Netbeans: Il client (vuoto)



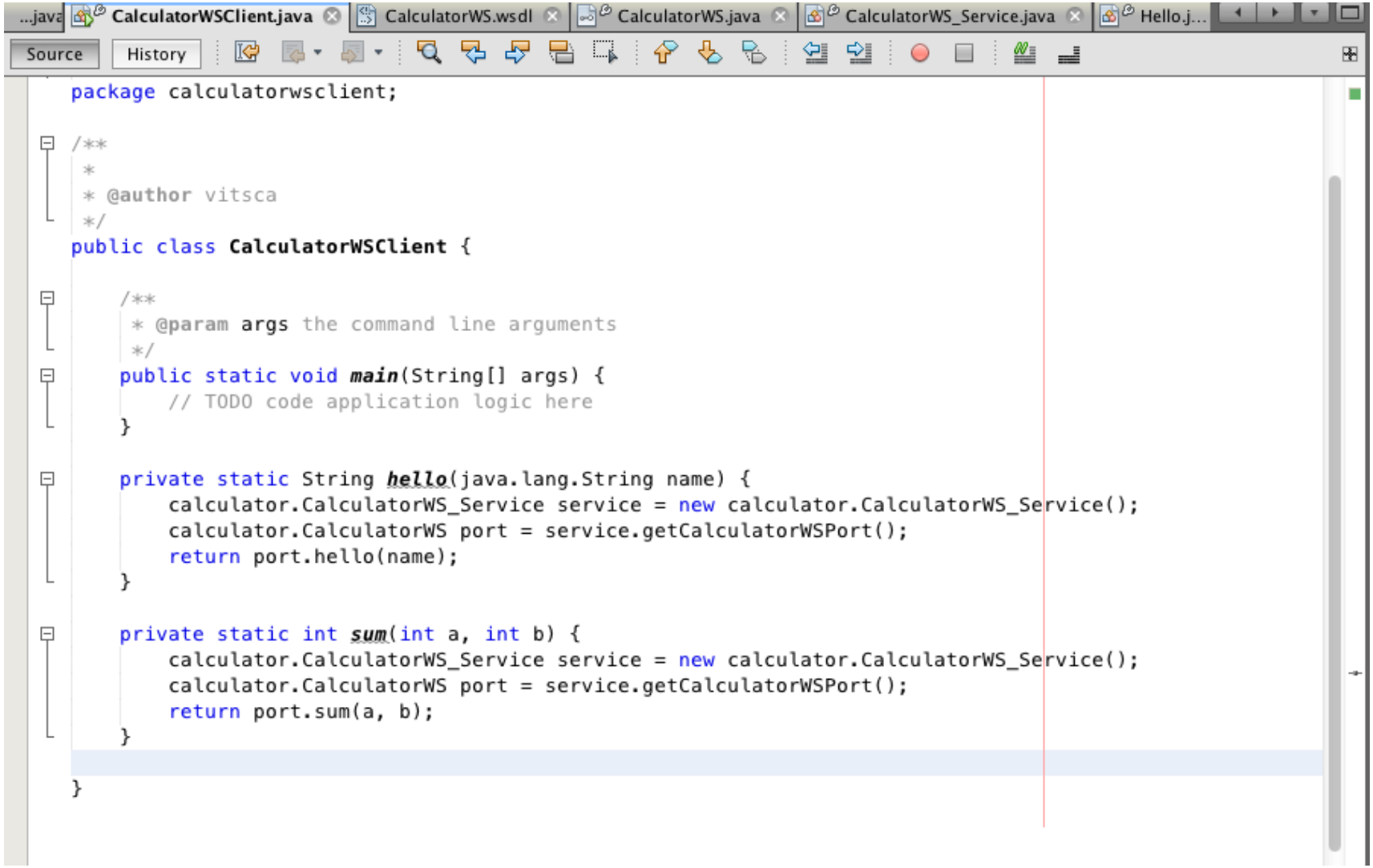
WS in Netbeans: Dal progetto WS . . .



WS IN NETBEANS: DRAG&DROP OPERAZIONE `hello`

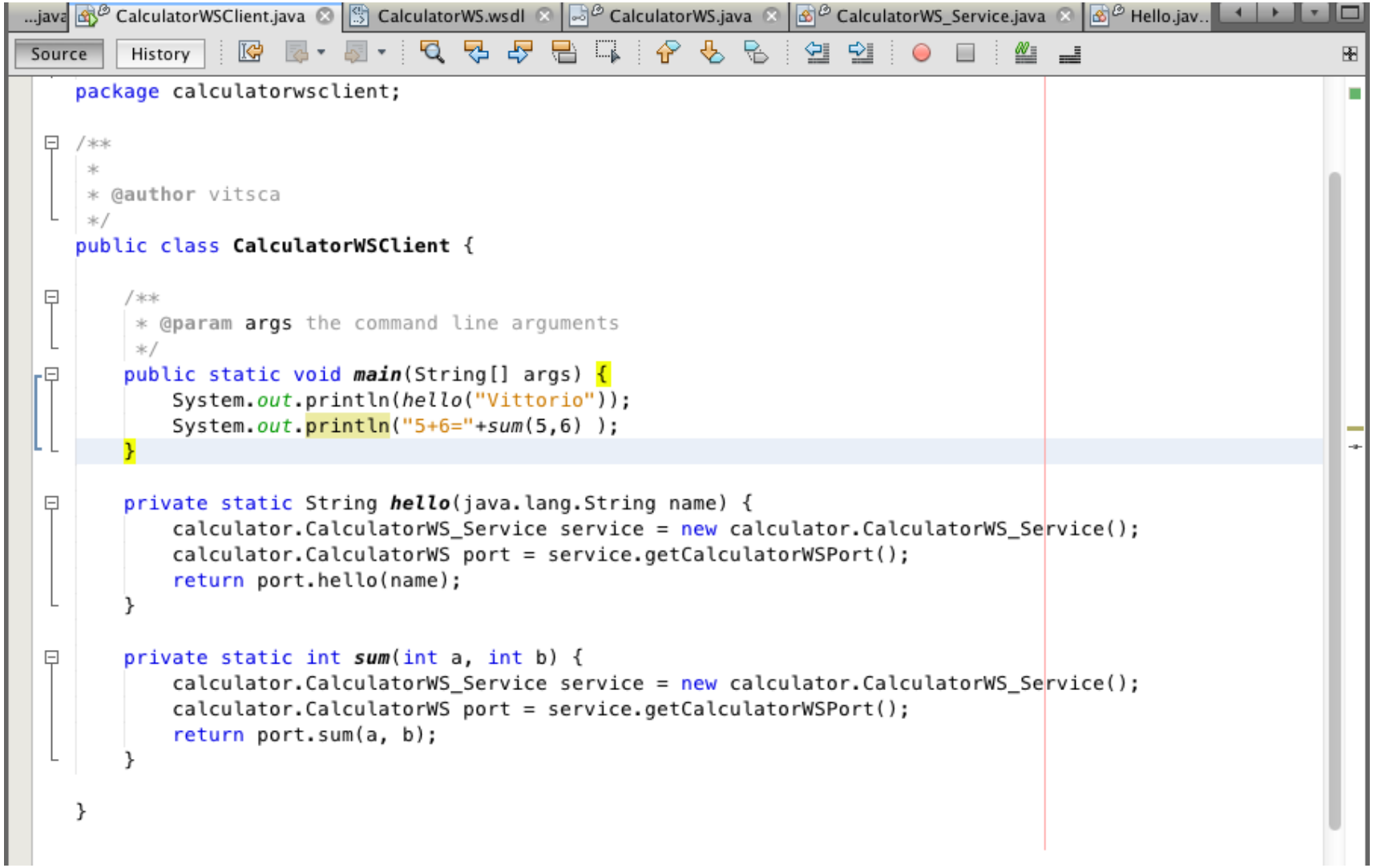


WS IN NETBEANS: DRAG&DROP OPERAZIONE `sum`

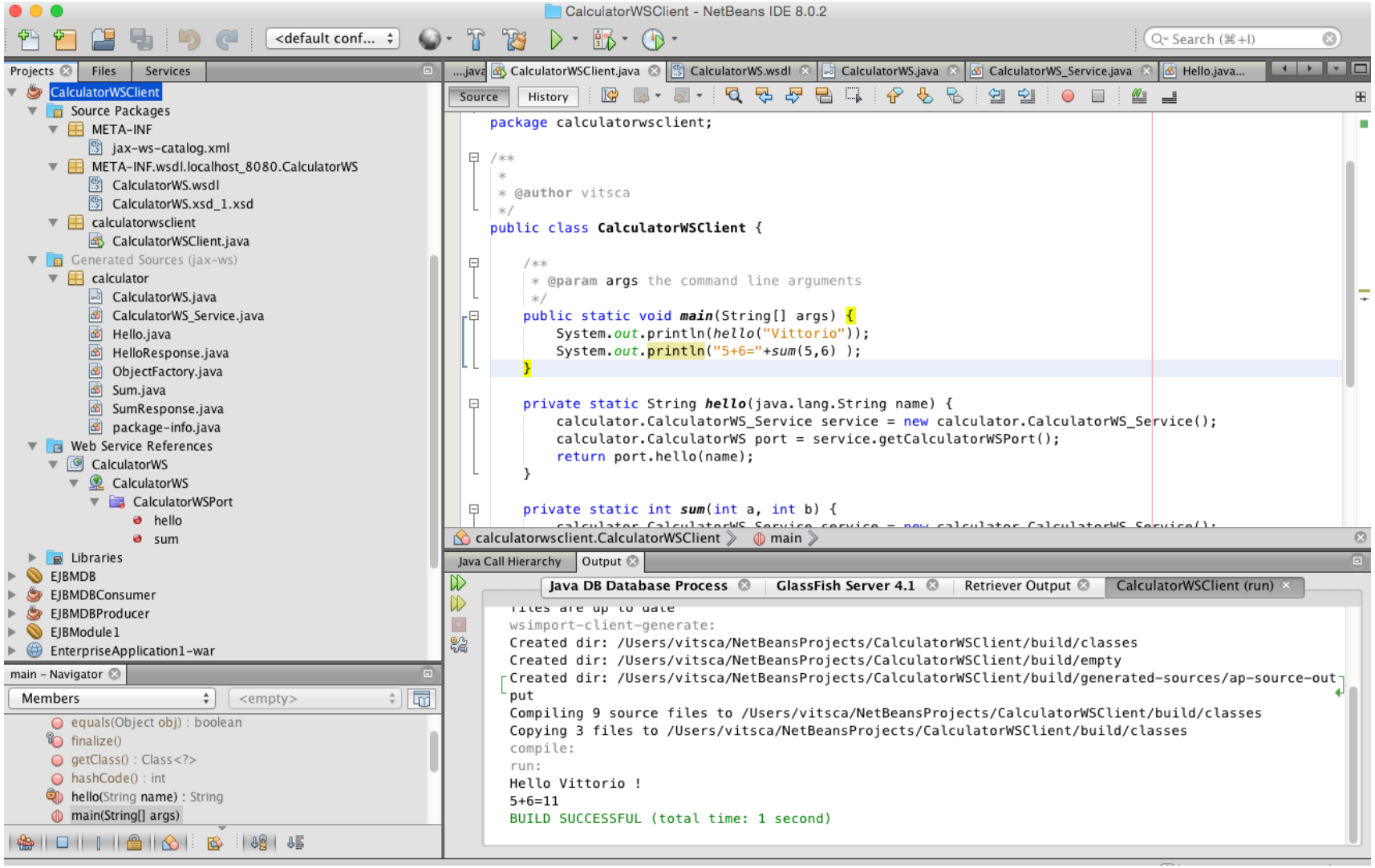



```java
package calculatorwsclient;

/**
 *
 * @author vitsca
 */
public class CalculatorWSClient {

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

    private static String hello(java.lang.String name) {
        calculator.CalculatorWS_Service service = new calculator.CalculatorWS_Service();
        calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.hello(name);
    }

    private static int sum(int a, int b) {
        calculator.CalculatorWS_Service service = new calculator.CalculatorWS_Service();
        calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.sum(a, b);
    }

}
```

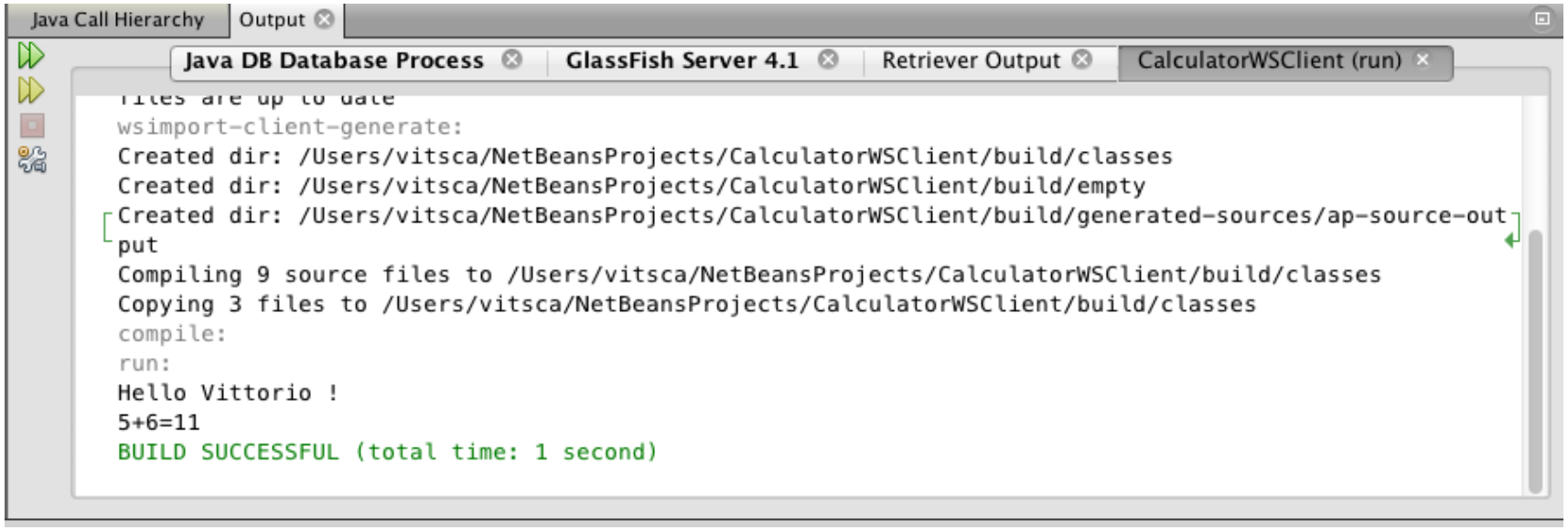

WS IN NETBEANS: . . . UN "POCHINO" DI CODICE



WS in Netbeans: Run . . .



WS in Netbeans: Con i risultati...



Plan

La architettura che abbiamo visto