Chapter 39 Web Services

Objectives

⍰To describe what a Web service is (§39.1).

⍰To create a Web service class (§39.2).

⍰To publish and test a Web service (§39.3).

⍰To create a Web service client reference (§39.4).

⍰To explain the role of WSDL (§39.4).

⍰To pass object type of arguments in a Web service (§39.5).

⍰To discover how a client communicates with a Web service (§39.5).

⍰To describe what are SOAP request and SOAP response (§39.5).

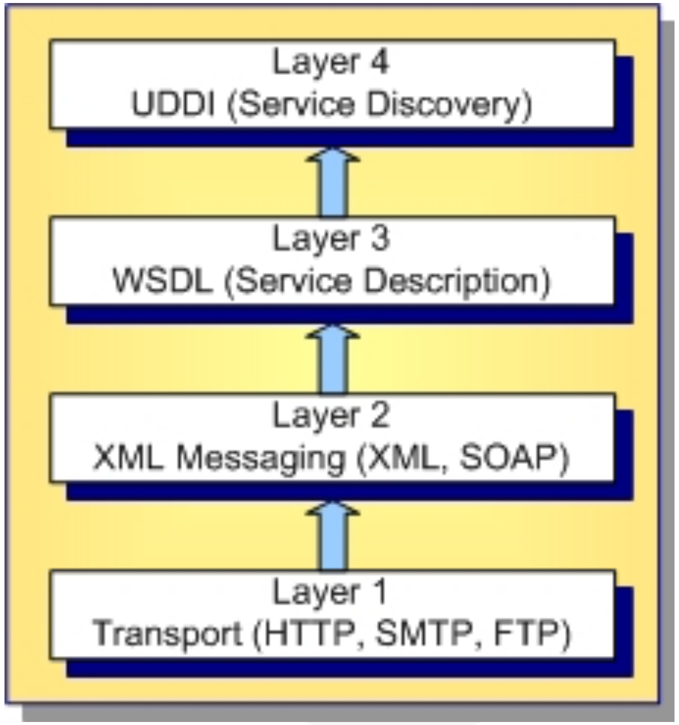
⍰To track a session in Web services (§39.6).

What is a Web Service?

Web service is a technology that enables programs (on different devices) to communicate through HTTP on the Internet. Web services enable a program on one system to invoke a method in an object on another system. You can develop and use Web services using any languages on any platform. Web services are simple and easy to develop.

Web services protocol stack

- Layer 4:
  - UDDI or URI: centralises services into a common registry such that network Web services can publish their location and description, and makes it easy to discover what services are available on the network.
- Layer 3:
  - used for describing the public interface to a specific Web service. The WSDL interface format is typically used for this purpose.
- Layer 2:
  - Messaging Protocol: responsible for encoding messages in a common XML format so that they can be understood at either end of a network connection. Currently, this area includes such protocols as XML-RPC, WS-Addressing, and SOAP.
- Layer 1:
  - Transport Protocol: responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP, as well as the more recent Blocks Extensible Exchange Protocol (BEEP).

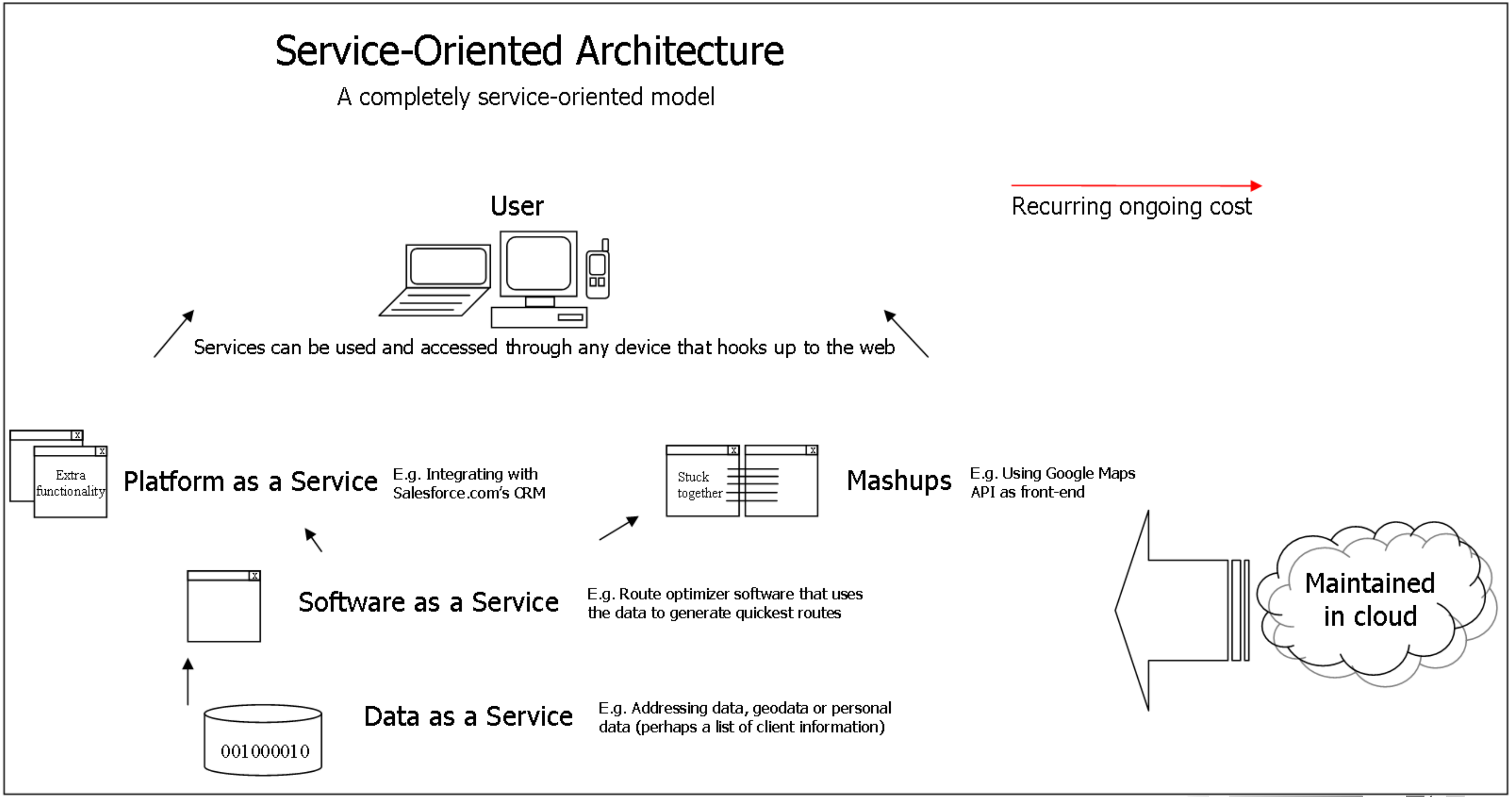

Layer 4
UDDI (Service Discovery)
Layer 3
WSDL (Service Description)
Layer 2
XML Messaging (XML, SOAP)
Layer 1
Transport (HTTP, SMTP, FTP)

What is SOAP?

Web services run on the Web using HTTP. There are several APIs for Web services. A popular standard is the *Simple Object Access Protocol* (SOAP), which is based on XML. The computer on which a Web service resides is referred to as a *server*. The server needs to make the service available to the client, known as *publishing a Web service*. Using a Web service from a client is known as *consuming a Web service*.

SOAP

- SOAP evolved as a successor of XML-RPC, though it borrows its transport and interaction neutrality and the envelope/header/body from elsewhere (probably from WDDX)
- SOAP has three major characteristics:
  1. extensibility (security and WS-routing are among the extensions under development)
  2. neutrality (SOAP can operate over any transport protocol such as HTTP, SMTP, TCP, UDP, or JMS)
  3. independence (SOAP allows for any programming model)



Service-Oriented Architecture

A completely service-oriented model

User

Recurring ongoing cost

Services can be used and accessed through any device that hooks up to the web

Platform as a Service

E.g. Integrating with Salesforce.com's CRM

Mashups

E.g. Using Google Maps API as front-end

Software as a Service

E.g. Route optimizer software that uses the data to generate quickest routes

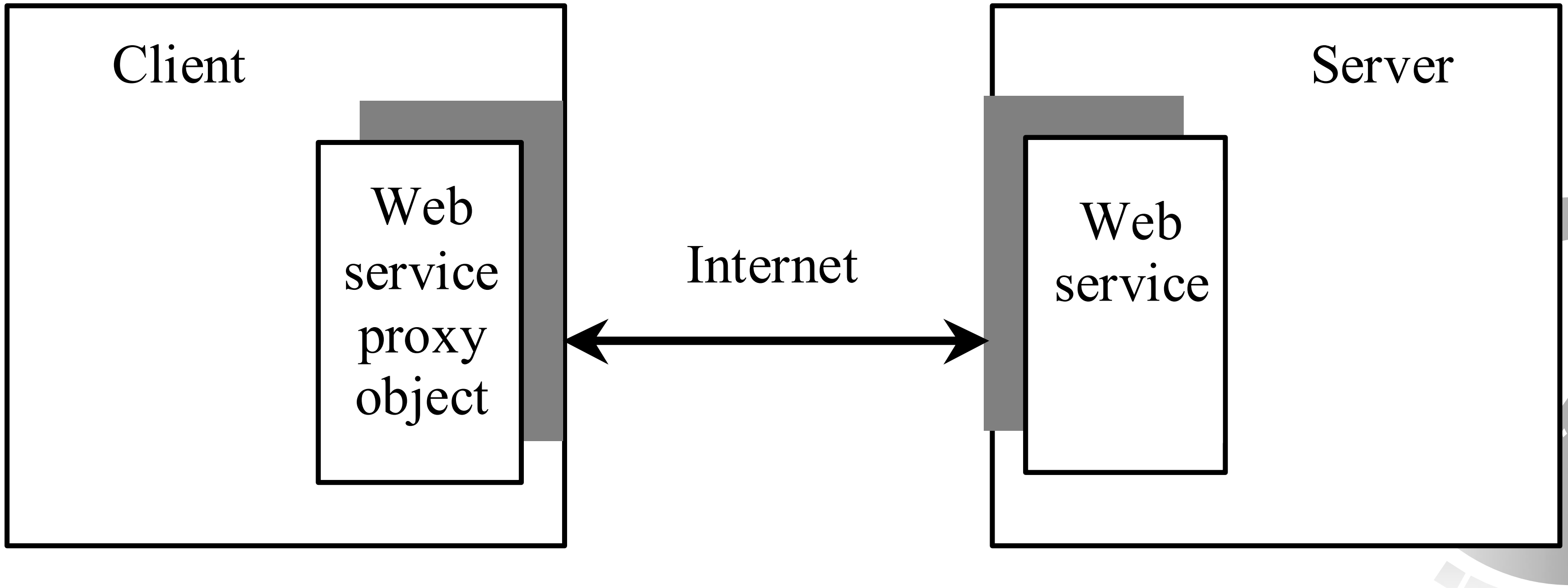
Data as a Service

E.g. Addressing data, geodata or personal data (perhaps a list of client information)

Maintained in cloud

How does a client communicate with a Web service

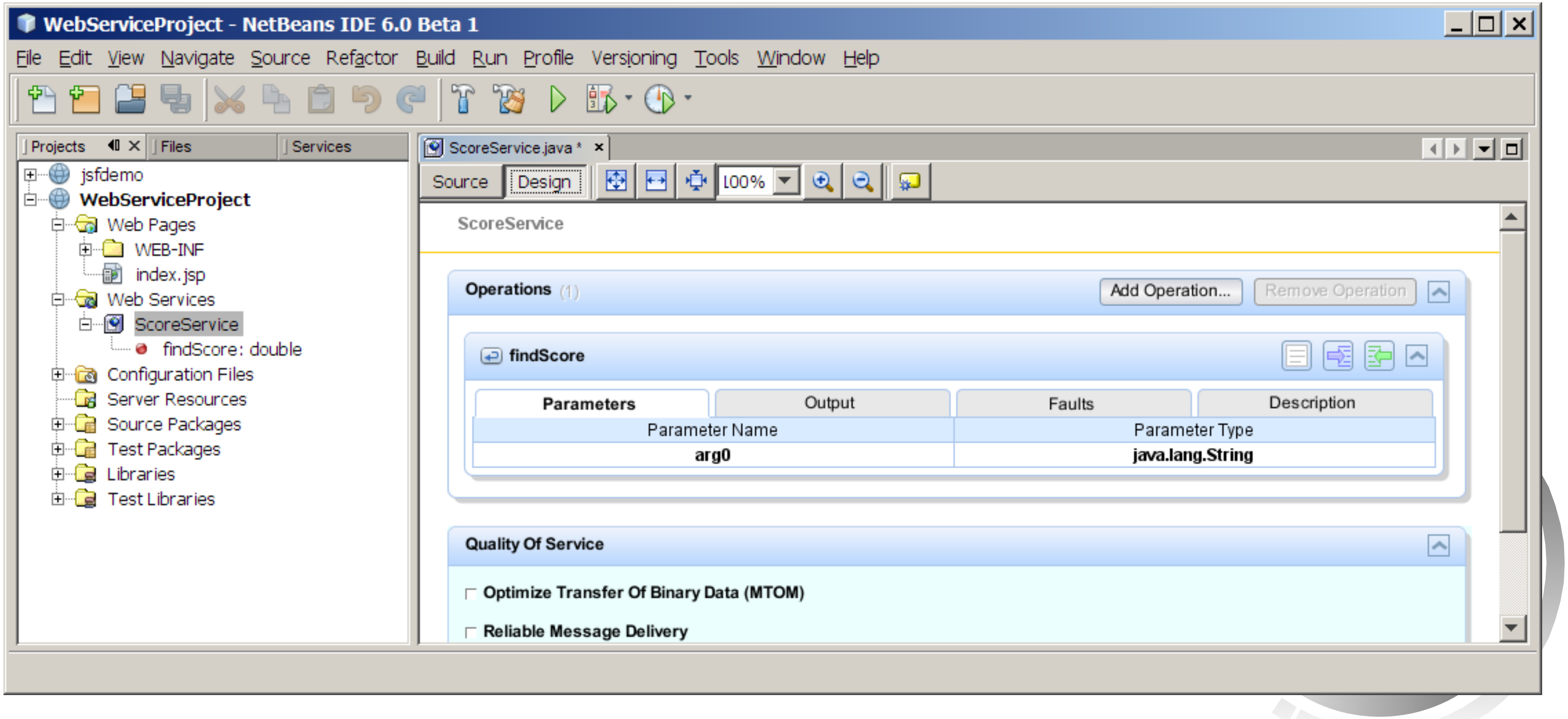
A client interacts with a Web service through a *proxy object*. The proxy object facilitates the communication between the client and the Web service. The client passes arguments to invoke methods on the proxy object. The proxy object sends the request to the server and receives the result back from the server, as shown in Figure 39.1.



Creating Web Services Using NetBeans

Create a Web project, Create a Web service, deploy Web service

For Detailed steps, follow the chapter on the website.

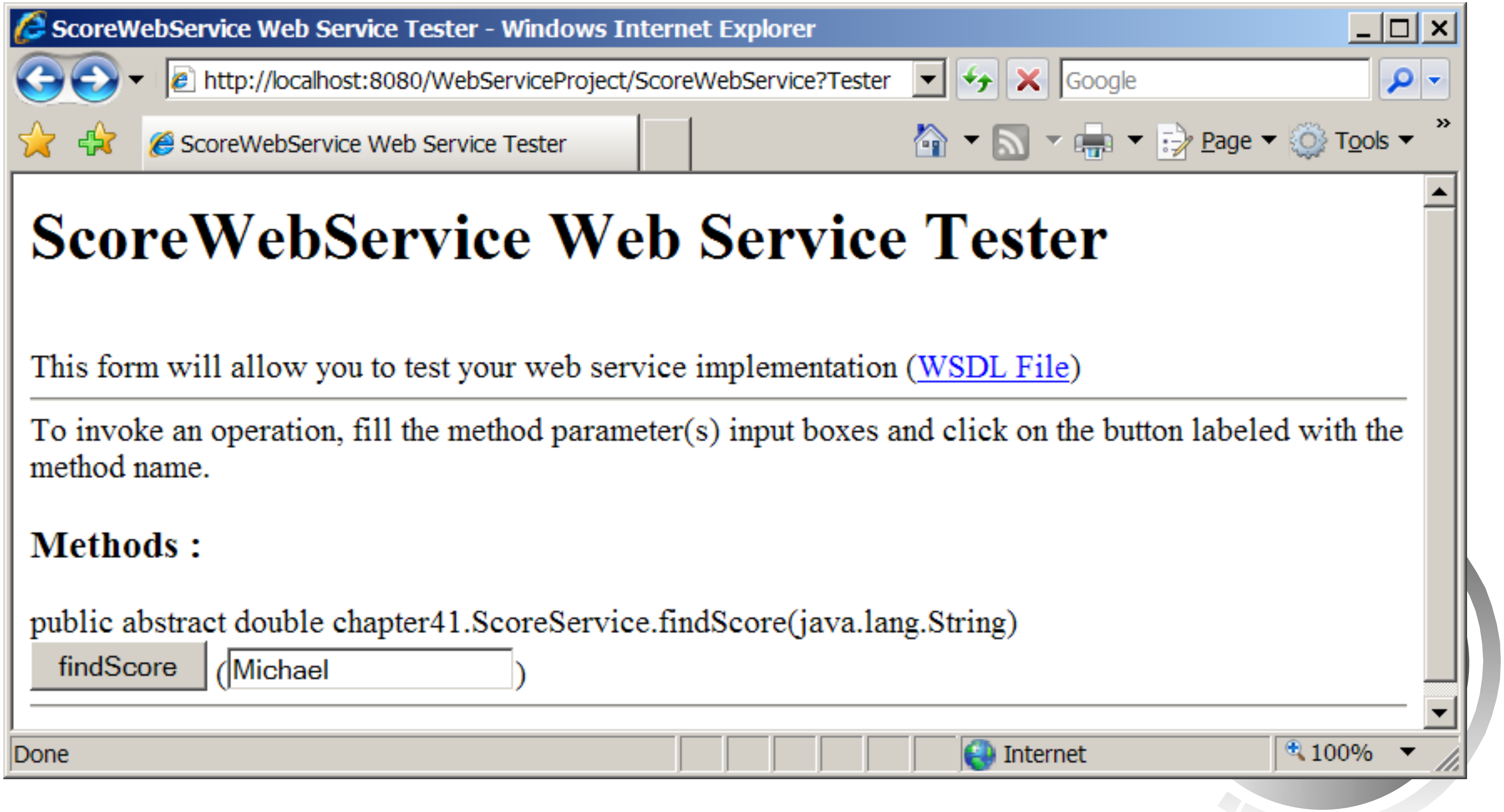


Parameter Name	Parameter Type
arg0	java.lang.String

□ Optimize Transfer Of Binary Data (MTOM)

□ Reliable Message Delivery

Testing Web Services



ScoreWebService Web Service Tester - Windows Internet Explorer
http://localhost:8080/WebServiceProject/ScoreWebService?Tester
Google
ScoreWebService Web Service Tester
Page
Tools
Done
Internet
100%

ScoreWebService Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

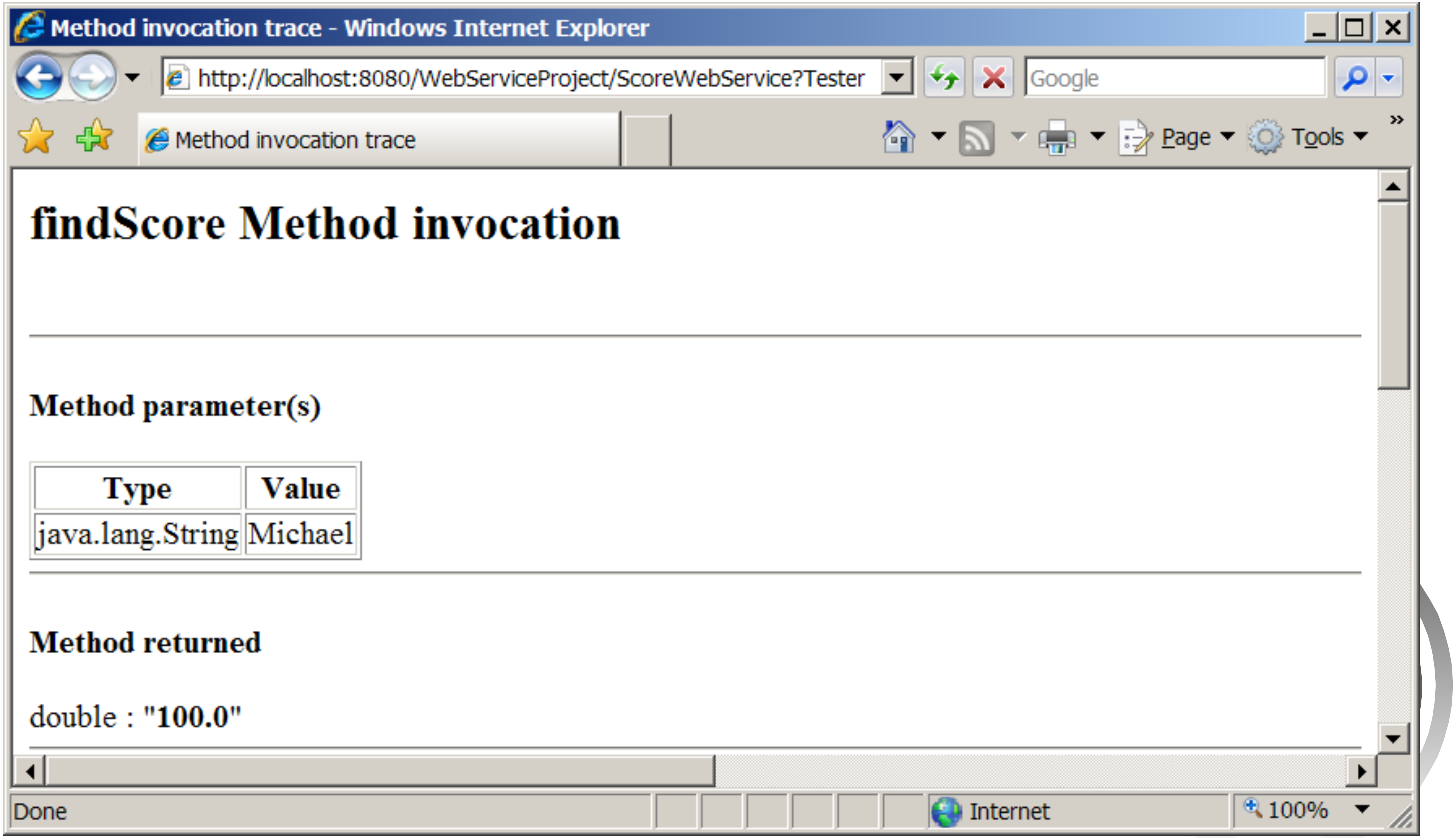
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract double chapter41.ScoreService.findScore(java.lang.String)

findScore (Michael)

Testing Web Services



Method invocation trace - Windows Internet Explorer
http://localhost:8080/WebServiceProject/ScoreWebService?Tester
Google
Method invocation trace
Page
Tools
Done
Internet
100%

findScore Method invocation

Method parameter(s)

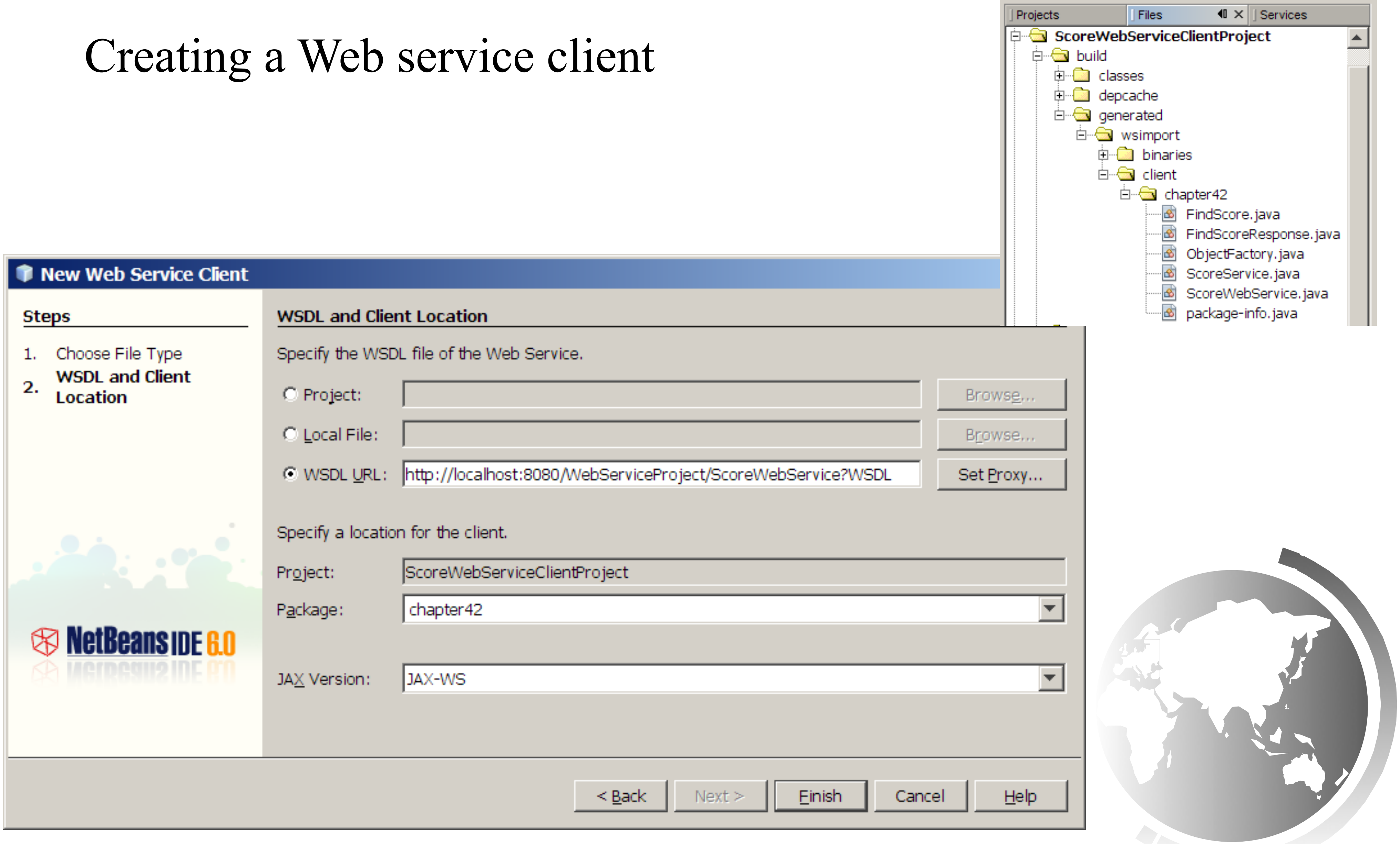
Type	Value
java.lang.String	Michael

Method returned

double : **"100.0"**

Consuming Web Services

Creating a Web service client



```
ScoreWebServiceClientProject
  build
    classes
    depcache
    generated
      wsimport
        binaries
        client
          chapter42
            FindScore.java
            FindScoreResponse.java
            ObjectFactory.java
            ScoreService.java
            ScoreWebService.java
            package-info.java
```

New Web Service Client

Steps

1. Choose File Type
2. **WSDL and Client Location**

WSDL and Client Location

Specify the WSDL file of the Web Service.

○ Project: [Browse...]

○ Local File: [Browse...]

● WSDL URL: http://localhost:8080/WebServiceProject/ScoreWebService?WSDL [Set Proxy...]

Specify a location for the client.

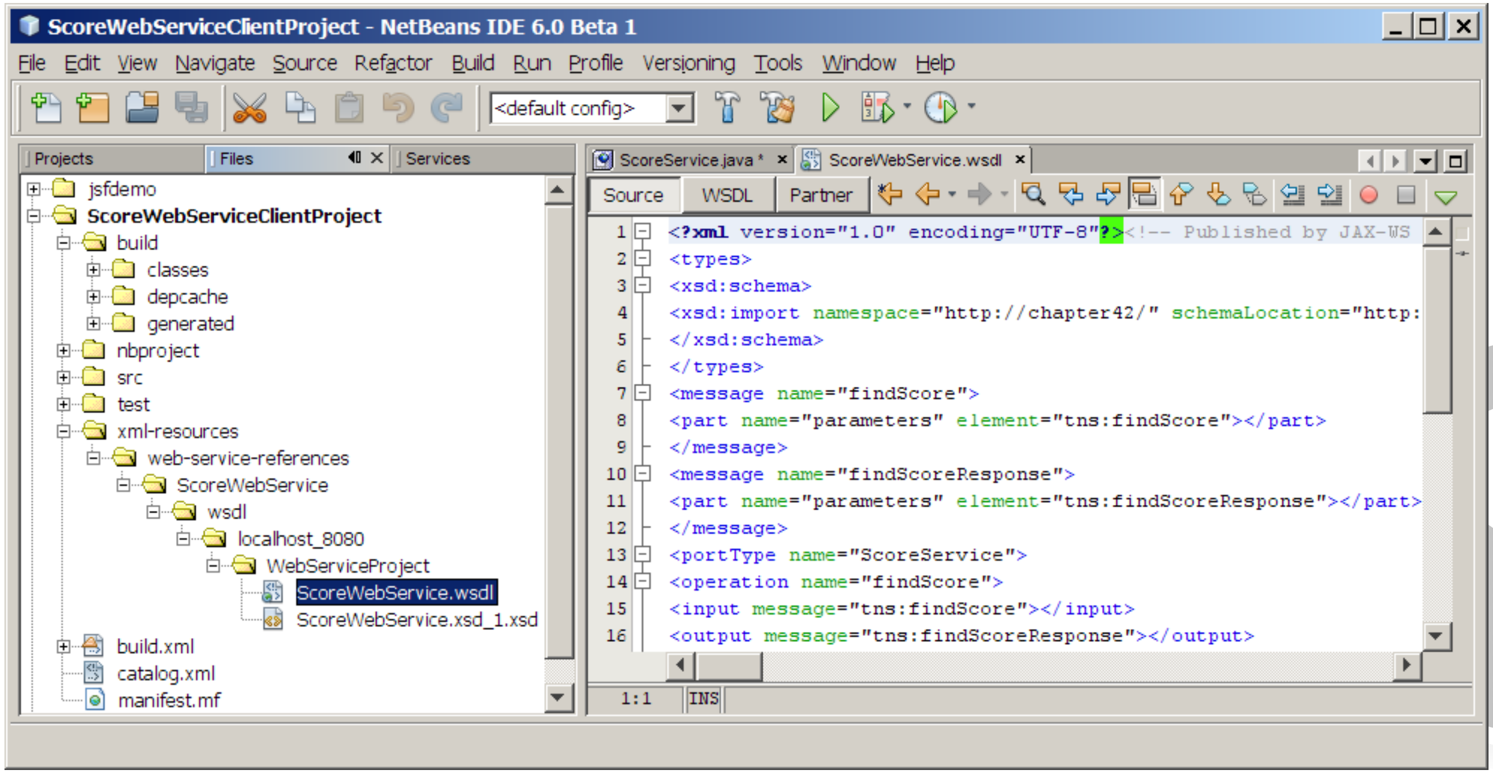
Project: ScoreWebServiceClientProject

Package: chapter42

JAX Version: JAX-WS

What is WSDL?

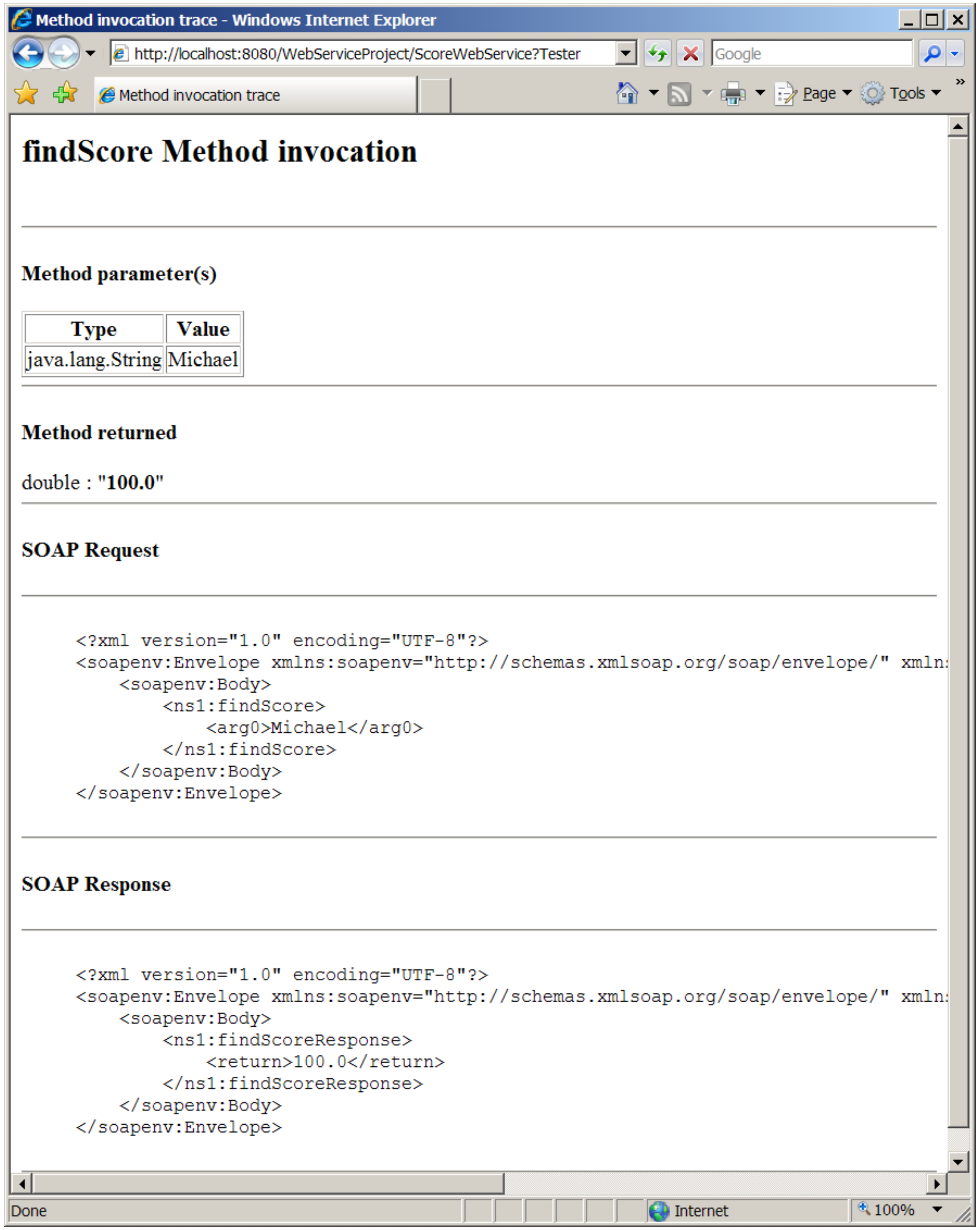
When you created a Web service reference, you entered a WSDL URL, as shown in Figure 39.6. A .wsdl file is created under the xm-resources folder, as shown in Figure 39.8. So *what is WSDL*? WSDL stands for *Web Service Description Language*. A .wsdl file is an XML file that describes the available Web service to the client, i.e., the remote methods, their parameters, and return value types, etc.


ScoreWebServiceClientProject - NetBeans IDE 6.0 Beta 1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
Projects
Files
Services
ScoreService.java *
ScoreWebService.wsdl
Source
WSDL
Partner
1:1
INS

```
jsfdemo
ScoreWebServiceClientProject
    build
        classes
        depcache
        generated
    nbproject
    src
    test
    xml-resources
        web-service-references
            ScoreWebService
                wsdl
                    localhost_8080
                        WebServiceProject
                            ScoreWebService.wsdl
                            ScoreWebService.xsd_1.xsd
    build.xml
    catalog.xml
    manifest.mf
```

```
<?xml version="1.0" encoding="UTF-8"?><!-- Published by JAX-WS
<types>
<xsd:schema>
<xsd:import namespace="http://chapter42/" schemaLocation="http:
</xsd:schema>
</types>
<message name="findScore">
<part name="parameters" element="tns:findScore"></part>
</message>
<message name="findScoreResponse">
<part name="parameters" element="tns:findScoreResponse"></part>
</message>
<portType name="ScoreService">
<operation name="findScore">
<input message="tns:findScore"></input>
<output message="tns:findScoreResponse"></output>
```

Passing and Returning Arguments


Method invocation trace - Windows Internet Explorer
http://localhost:8080/WebServiceProject/ScoreWebService?Tester
Google
Method invocation trace
Page
Tools
Done
Internet
100%

findScore Method invocation

Method parameter(s)

Type	Value
java.lang.String	Michael

Method returned

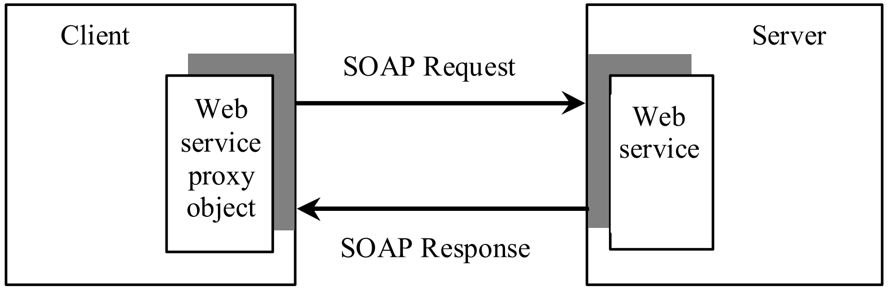
double : "**100.0**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmln:
    <soapenv:Body>
        <ns1:findScore>
            <arg0>Michael</arg0>
        </ns1:findScore>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmln:
    <soapenv:Body>
        <ns1:findScoreResponse>
            <return>100.0</return>
        </ns1:findScoreResponse>
    </soapenv:Body>
</soapenv:Envelope>
```



XML serialization/deserialization

Can you pass an argument of any type between a client and a Web service? No. SOAP only supports primitive types, wrapper types, arrays, String, Date, Time, List, and several other types. It also supports certain custom classes. An object that is sent to or from a server is serialized into XML. The process of serializing/ deserialization objects, called *XML serialization/deserialization*, is performed automatically. For a custom class to be used with Web methods, the class must meet the following requirements:

- The class must have a no-arg constructor.
- Instance variables that should be serialized must have public get and set methods. The classes of these variables must be supported by SOAP.

Web Service Session Tracking

§37.8.3, “Session Tracking Using the Servlet API,” introduced session tracking for servlets using the javax.servlet.http.HttpSession interface. You can use HttpSession to implement session tracking for Web services. To demonstrate this, consider an example that generates random True/False questions for the client and grades the answers on these questions for the client.