Simple Implementation of a WebService using Eclipse

Author: Gennaro Frazzingaro
Universidad Rey Juan Carlos
campus de Mostòles (Madrid)
GIA – Grupo de Inteligencia Artificial

Grupo de Inteligencia Artificial
Universidad Rey Juan Carlos

Contents

- Web Services introduction
- Axis – a toolkit for Web services
- Development
  - Eclipse (Java IDE) WTP project.
- Walk through a sample

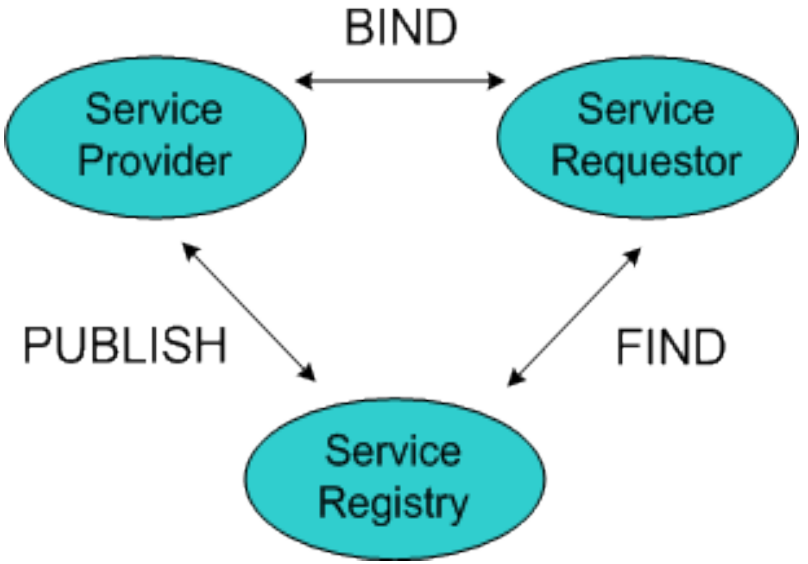
What's a Web Service

"Software applications identified by a URI, whose interface and bindings are capable of being identified, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocls" (W3C).

What's a Web Service (2)

Web services are software system designed to support interoperable Machine to Machine interaction over a network that:

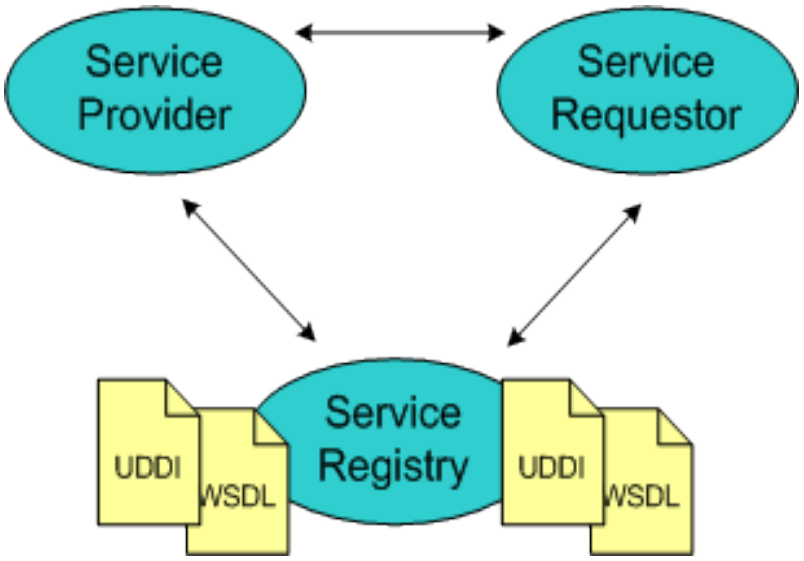
- use open, XML-based standards and transport protocols (SOAP-messages) to exchange data with clients.
- is accessed over a network and executed on a remote system hosting the requested services



Web Service details

Web Services' core is a Web Services Description Language (WSDL) file:

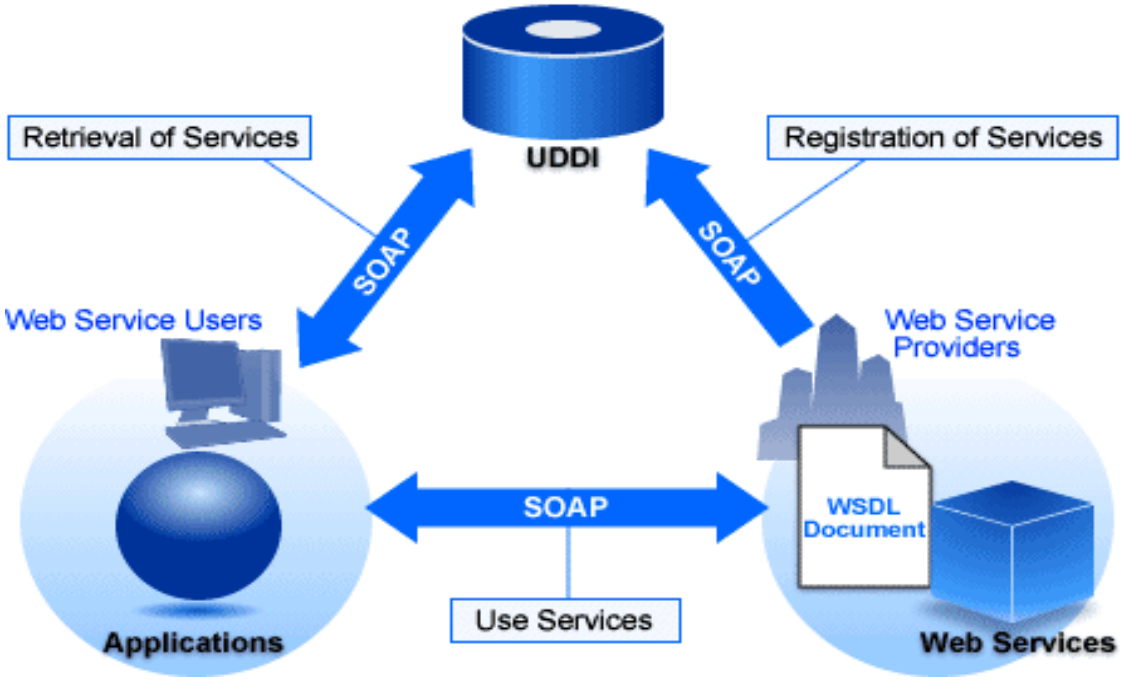
- **WSDL** is an XML-based description language that is used to provide information such as the location of a Web service (i.e., its URI), the communication protocol that is used in the Web service, the messaging format and everything that the Web service user (i.e., the application that calls the Web service) requires in order to call the Web service.
- **UDDI** is a technology to enable the registration and retrieval of Web services.



How does a WS work?

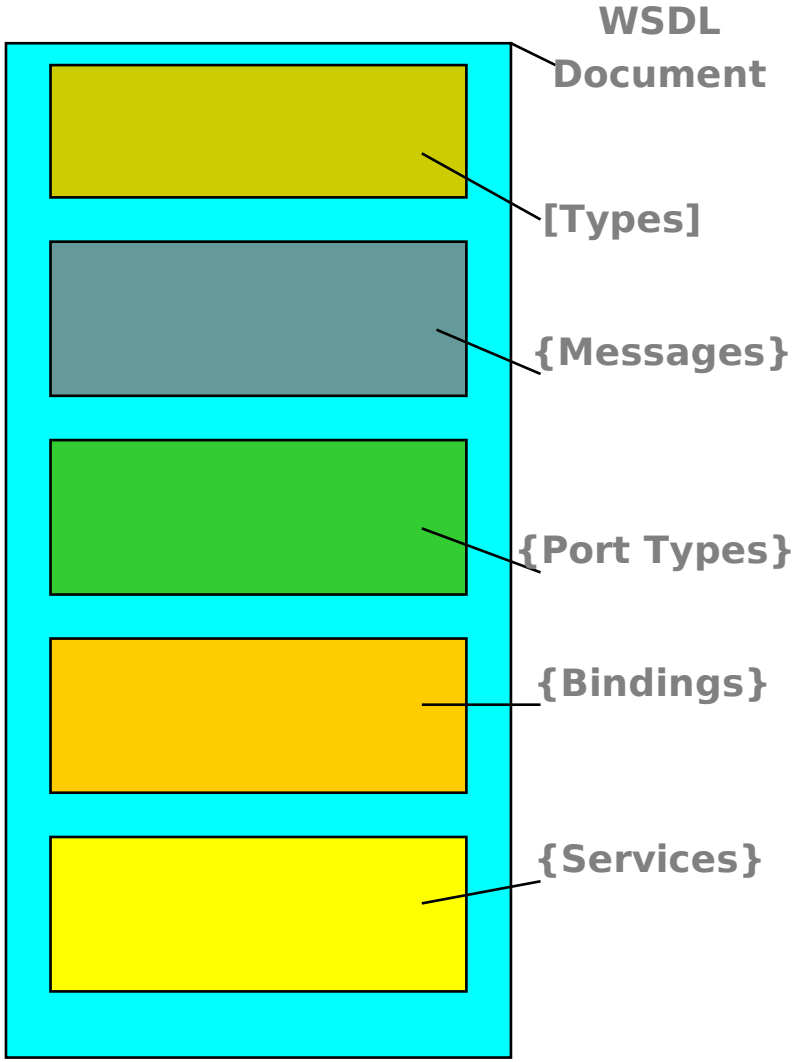
- A WS is published in a UDDI registry
- The client search for the WS in the registry
- The client build the proxy dynamically
- The client invoke the service and receive a response

NOTE: also the UDDI research and publish methods are web services!



WSDL 1.1
Document Structure

- **Types:** platform & language independent vocabulary definition
- **Messages:** composition or aggregation of types
- **Port-Types:** operations' definition in input and output message
- **Binding:** operations and types assignment to a URIs
- **Services:** physical deployment of the ports



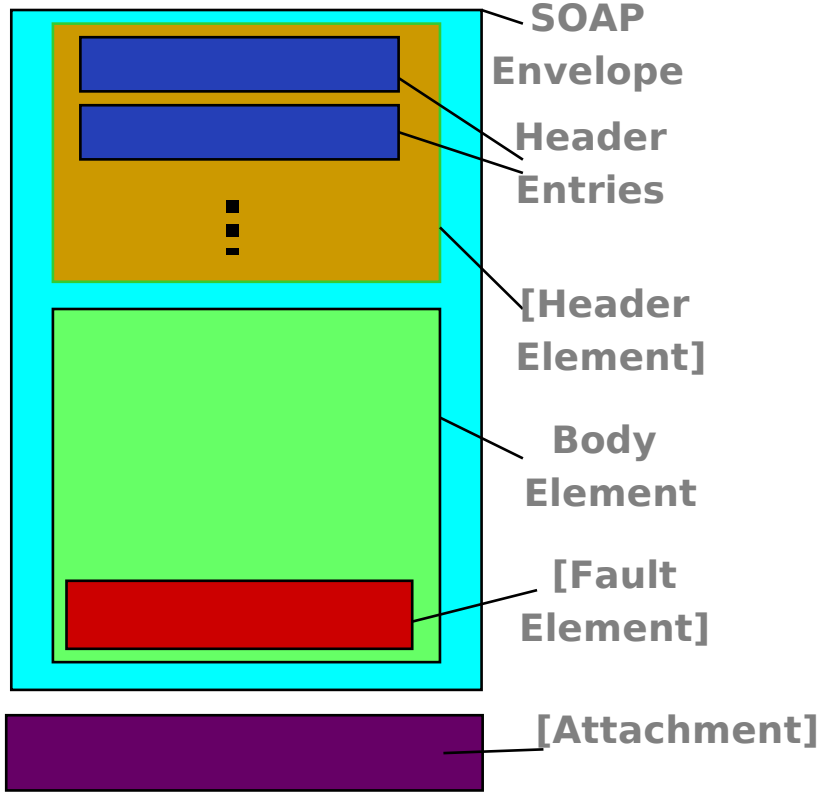
How to speak with a WS

Messages to and from a WS are called SOAP message:

- **Simple Object Access Protocol (SOAP)** is an XML-based communication protocol suitable for Web services.
- SOAP is transport-independent like every common web-protocols (HTTP, HTTPS, SMTP, FTP).

Composition:

- Extern elements:
  - **ENVELOPE** e **ATTACHMENT(S)**
- Intern elements:
  - **Header:** safety, routing, etc…
  - **Body:** real message content (request or response)



AXIS

(official site <http://ws.apache.org/axis/>)

- Apache eXtensible Interaction System (AXIS) is One of the Web service projects of Apache
- It provides a set of libraries for:
  - converting between Java Interfaces and WSDL descriptions.
  - the client and server side to handle invocations (marshalling and un-marshalling)
  - monitoring SOAP requests

How to use Axis with Eclipse

- Download Axis from the official website.
- Put lib folder into Eclipse installation folder and axis folder (inside the webapps folder) into the tomcat folder with the same name.
- Create a new JavaProject including axis library (you can find it into the Tomcat common/lib folder): mail.jar, activation.jar
- Create a sample class

```
package io.me;
public class Ciao {
  public String saluto(String nome){
    return "Ciao "+nome+"!";
  }
}
```

- Compile and Move this class in Tomcat folder

  webapps/axis/WEB-INF/classes

- Activate the service creating a deployment file and ed effettuare la apposita routine di Axis).

How to create a WS with Eclipse and WTP

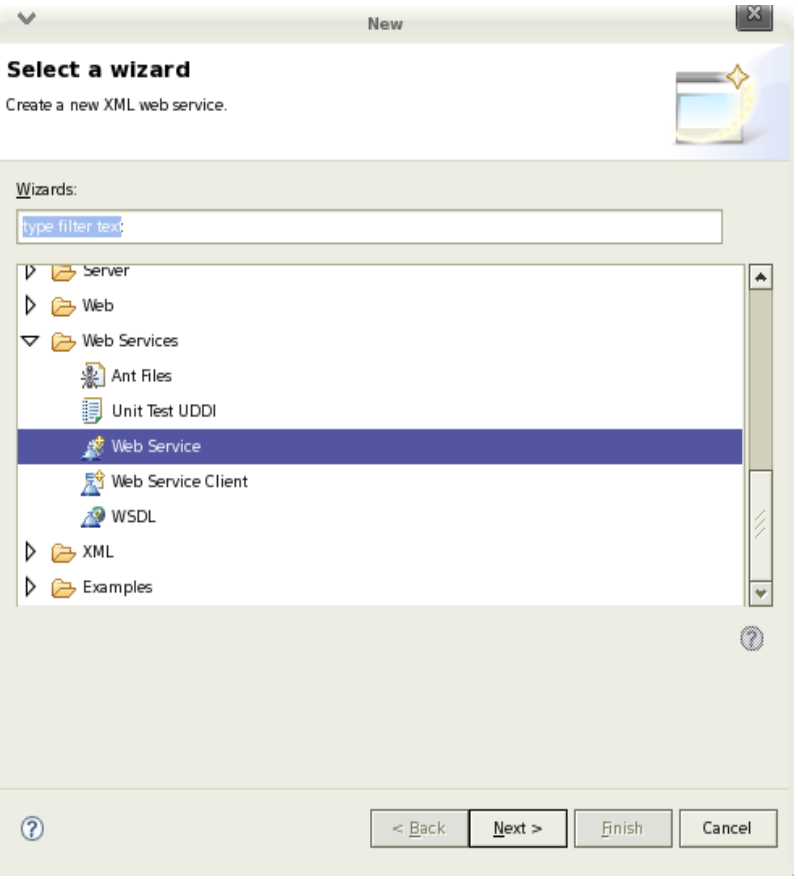
(much more simple!!)

- Download the WebToolsPlatform for Eclipse from http://www.eclipse.org/webtools/main.php

- Create a new Dynamic Web Project
- Create a new class 'Ciao' in a package me

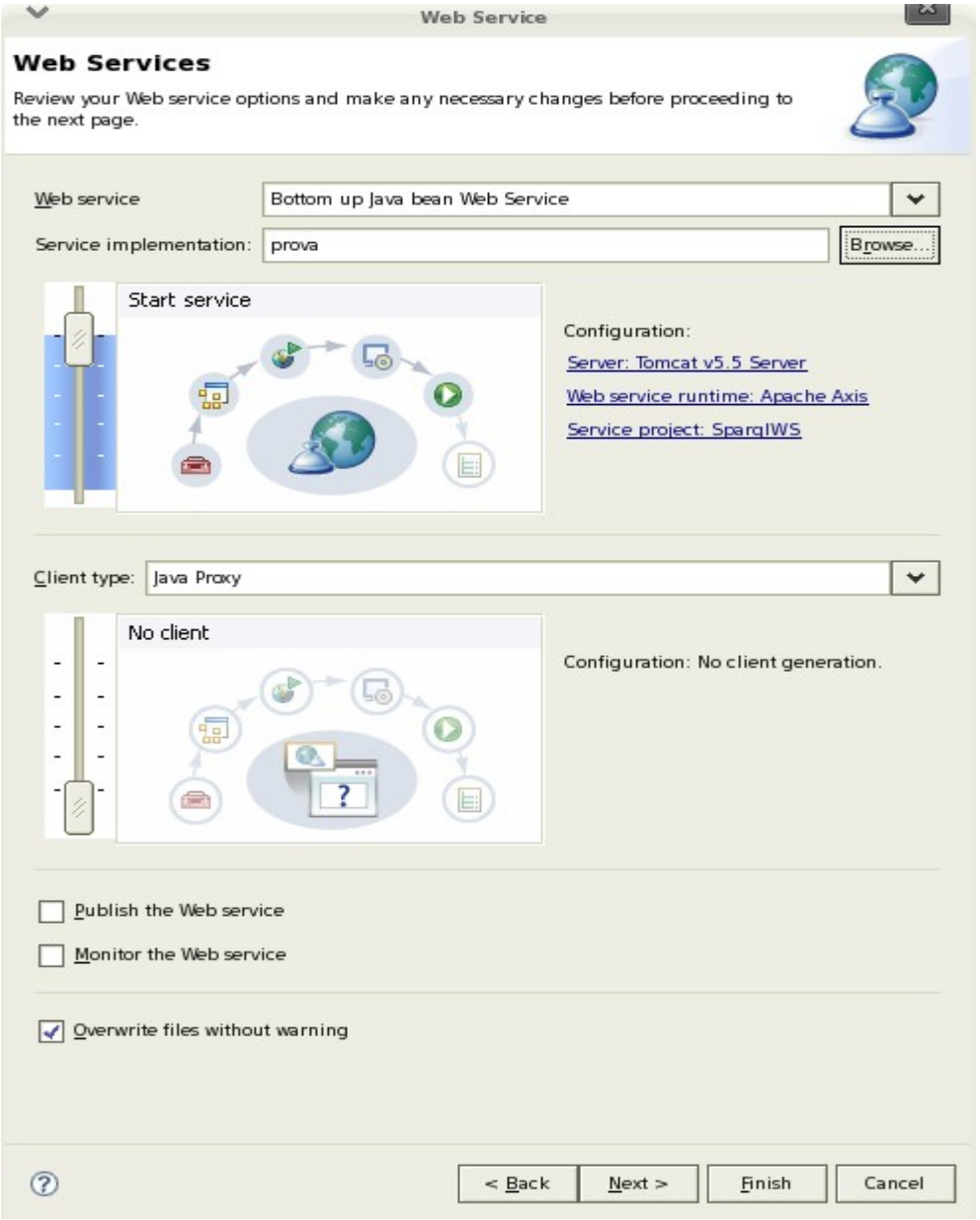
```
package me;
public class Ciao {
  public String saluto(String name){
    return "Ciao "+name+"!";
  }
}
```

- Create a WebService for this class (you need a Tomcat Server started from Eclipse)



Configuring a WS with WTP

- The Service should implement the 'Ciao' Class just written on the Tomcat server running.
- Eclipse could also create, deploy and start a client after the creation of the Service.
- A WSDL file was automatically generated



WSDL automatically generated

- Eclipse automatically created a wsdl file for the WS deployment (in WebContent/wsdl) structured as following:
  - **TYPES**
    - 2 xsd:string `<element name="saluto">` `<element name="wishesResponse">` corresponding to the input and the output parameter
  - **MESSAGE**
    - A message for each type similar to

```
<wsdl:message name="salutoRequest">
<wsdl:part element="impl:saluto" name="parameters"/>
</wsdl:message>
```

WSDL automatically generated (2)

- **PORTTYPE**
  - An input and a output for each operation

```
<wsdl:operation name="saluto">
<wsdl:input message="impl:salutoRequest" name="salutoRequest"/>
<wsdl:output message="impl:salutoResponse" name="salutoResponse"/>
</wsdl:operation>
```

- **BINDING**

```
<wsdl:binding name="provaSoapBinding" type="impl:prova">
```

  - Transport method

```
<wsdlsoap:binding style="document“
transport="http://schemas.xmlsoap.org/soap/http"/>
```

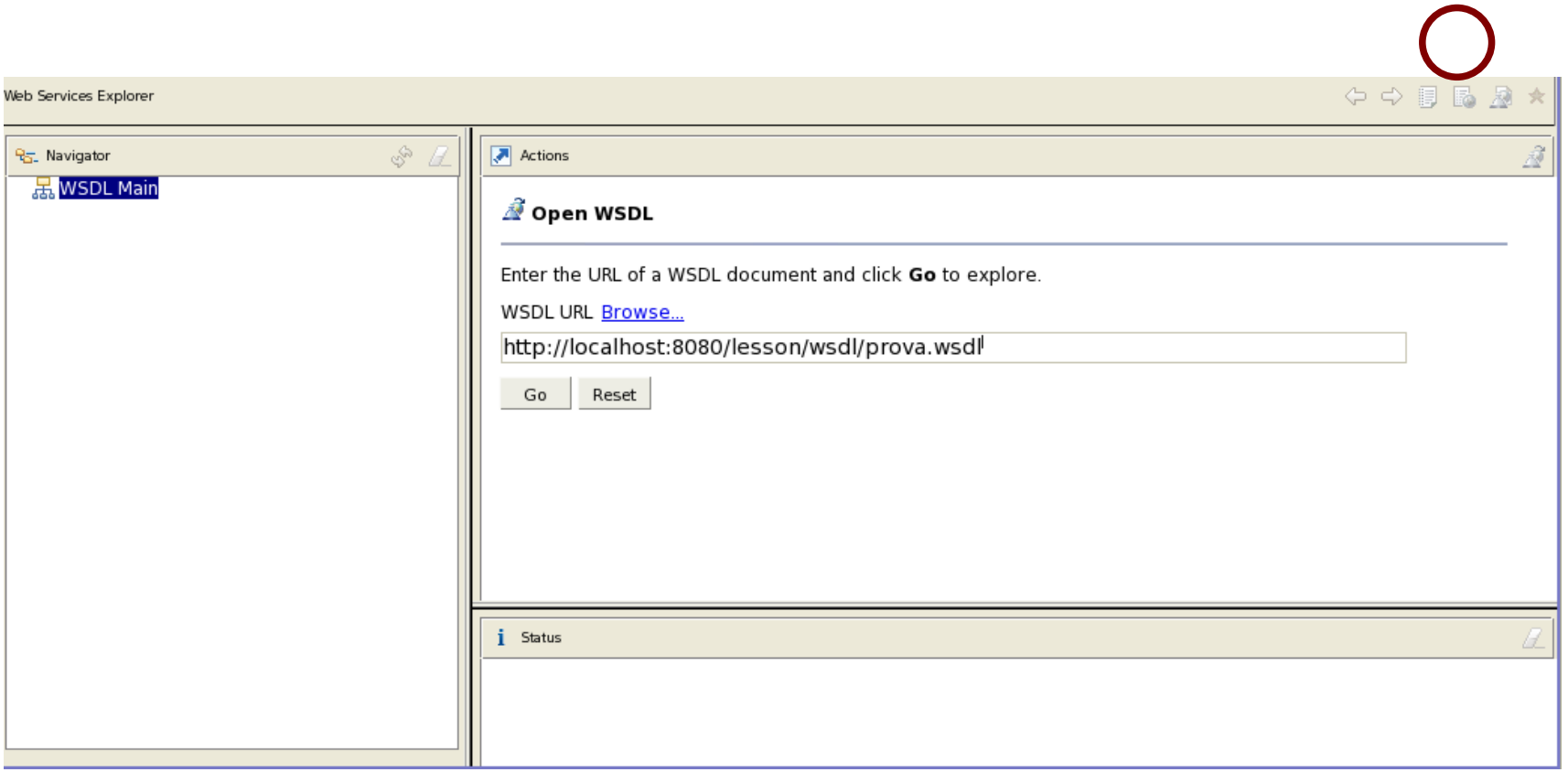
  - Definition of SOAP message content

- **SERVICE**
  - An address for each port declared

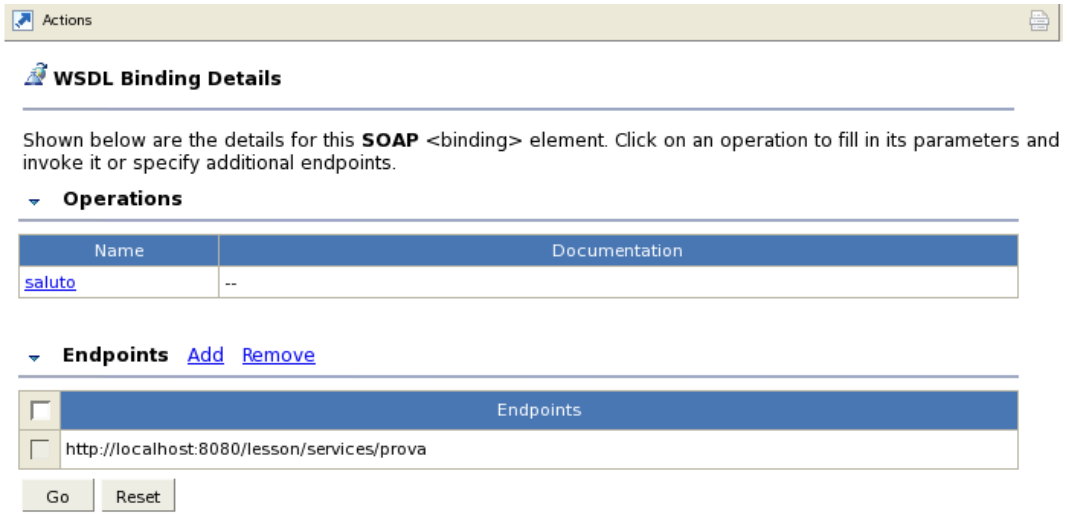
```
<wsdlsoap:address
  location="http://localhost:8080/lesson/services/prova"/>
```

Web Service Explorer

- It's an included plugin to test any Web Service, and it's available on
  Run -> Launch the Web Service Explorer
  menu item
- This is the screenshot of the WSDL Page

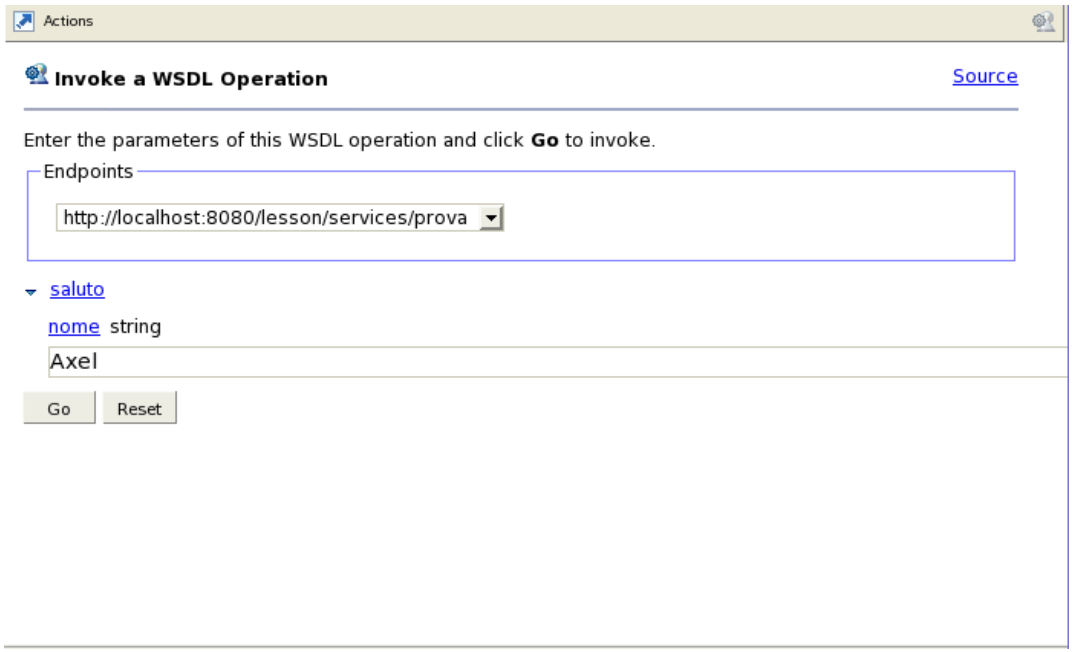


Checking the wsdl file



- Clicking on the 'Go' button you can show the list of function the requested web server has got.

- Choosen the function you can put the value of the parameters which in this case will be just the input of the aforementioned funcion

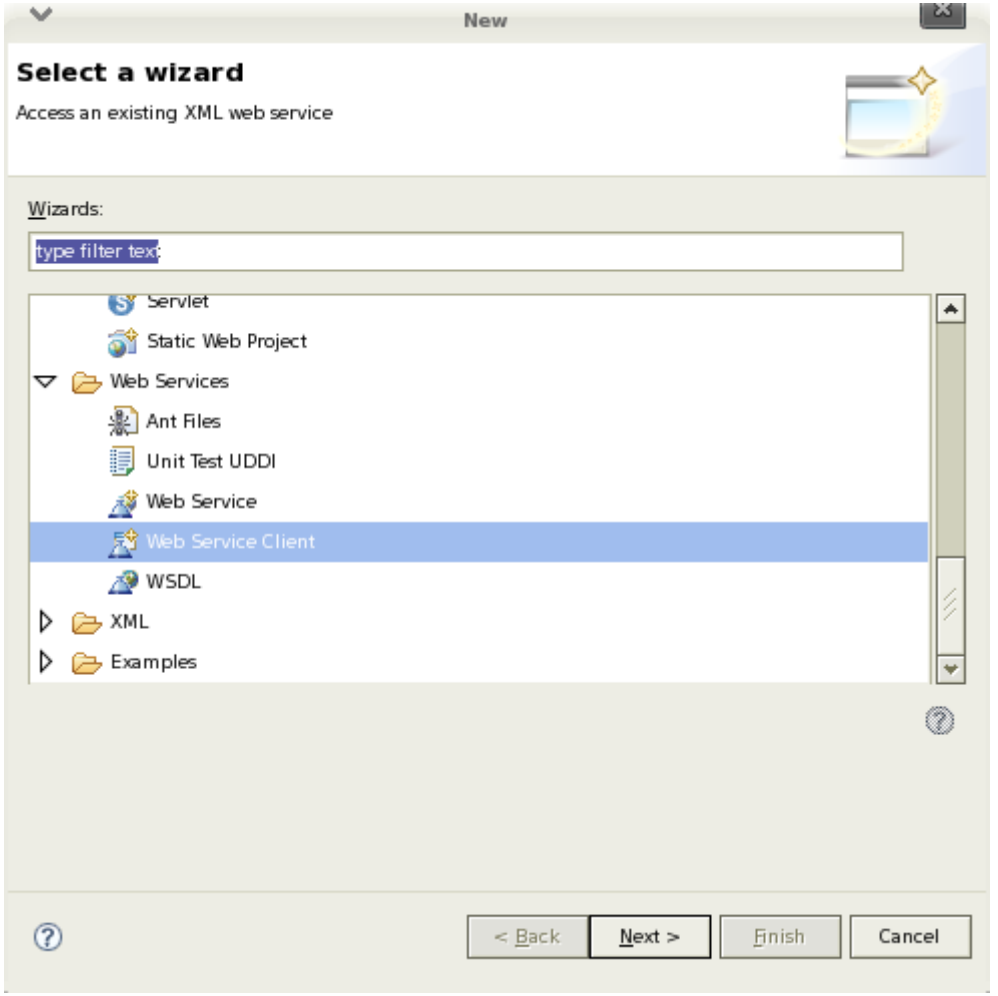


Checking the SOAP messages

- Here the Request and the Response SOAP message (if you click on the 'Source' button)

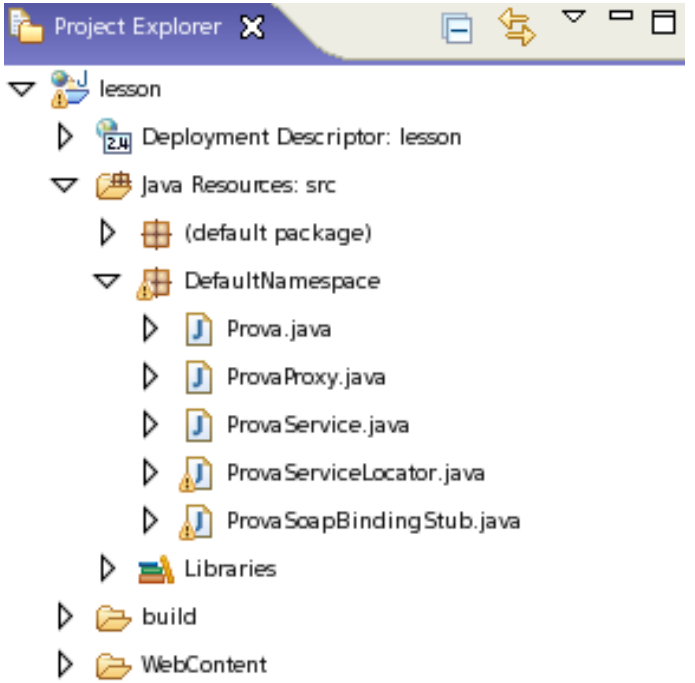
How to deploy a WS client

- To deploy also a client requesting an existing Web Service you need just to specify a valid wsdl file to inform Eclipse how to manage correct SOAP requests and responses.
- It's possible to define a proxy different to the default one (<project_name>/src), for example to manage an existing website
- It's possible to create some static(HTML) and dynamic (Servlets, JSPs) webpages to manage user interfaces (methods often called through the BEANS).



How to deploy a WS client (2)

- Eclipse automatically create 5 class for each method in the WSDL file
  - An interface to call the method itself
  - A proxy class to set the Endpoints
  - A Service interface for the adresses
  - A class for the Service Locator
  - A “Binding Stub” class which really call the WS



References

- Web Service Console Eclipse Plugin
  http://wscep.sourceforge.net/
- Axis Documentation
  http://ws.apache.org/axis/java/

A dimostrative example

and

Questions

(not necessary...)