A service-oriented programming language

Ivan Lanese

(Original slides from Fabrizio Montesi)

Web services

- Web services are a main technology to realize SOAs
  - … but REST services are more used nowadays
- Web services allow to perform integration
  - Web services communicate using open protocols (XML, HTTP, SOAP, …)
  - Web services are language- and platform- independent
- Web services can exploit and be exploited by other web services
  - Jolie can invoke external web services
  - Jolie can expose its behaviors as web services
- Web service interfaces are described by WSDL documents
  - We will study WSDL basics in next lessons
  - Jolie has tools for
    - Translating a WSDL document into an iol Jolie interface
    - Generating a WSDL document exposing a Jolie input port

Testing your web service

- If you build a new web service or want to use a 3rd party service you would need to test it

- The SoapUI tool from http://www.soapui.org/ allows you to generate sample SOAP invocations and check the replies
  - Download and install the tool (the open source version is for free)
  - Choose New SOAP project
  - Load your WSDL
  - Choose to generate sample requests for all operations

- You can edit the different requests to test various behaviours

Invoking a web service from Jolie

- We need to find a web service, and its WSDL description
- We need to translate the WSDL document into a Jolie iol interface
  - Good: there is a tool, wsdl2jolie, to do this
  - Less good: WSDL is more precise than Jolie type system, hence you may lose some information
- Syntax: wsdl2jolie wsdl-url > iol-file
- You need to remove the first line from the file

Invoking a web service from Jolie: an example

- Let us invoke the calculator web service at
  http://www.dneonline.com/calculator.asmx
  whose WSDL is available at
  http://www.dneonline.com/calculator.asmx?wsdl

- We apply wsdl2jolie

- A sample client is

```
include "console.iol"
include "calc.iol"

main
{
x.intA=5;
x.intB=8;
Add@CalculatorSoap(x)(y);
println@Console(y.AddResult)();
Multiply@CalculatorSoap(x)(y);
println@Console(y.MultiplyResult)()
}
```

Exposing a Jolie web service

- We need a Jolie service with at least one soap input port
- We need to generate the WSDL document corresponding to the input port
  - We can use jolie2wsdl to do this
- Syntax:
  jolie2wsdl --namespace [target_name_space]
  --portName [name_of_the_port] --portAddr [address_string]
  --o [output_filename] filename.ol
- portAddr is http://...
- We attach the generated WSDL document to the port by setting the following properties of the soap protocol:
  - .wsdl to the url of the .wsdl file
  - .wsdl.port to the name of the port inside the .wsdl file
  - .dropRootValue to true

Exposing a Jolie web service: limitations

- All types occurring in the port definition need to be complex types (with a suitable type definition)
  - It is better if each operation uses a distinct type
- For all of them the root should have type void
- These limitations are due to the definition of SOAP

Exposing a Jolie web service: example

- We want to export our calculator service
- We update it to match the requirements
- We set soap as protocol for the input port
- We execute jolie2wsdl
  jolie2wsdl --namespace mytest.test.com --portName CalcPort
  --portAddr http://localhost:8000 --outputFile myWsdl.wsdl server.ol
- We attach the generated WSDL document to the port

```
inputPort CalcPort {
Location: "socket://localhost:8000"
Protocol: soap {
          .wsdl = "./myWsdl.wsdl";
          .wsdl.port = "CalcPortServicePort";
          .dropRootValue = true
          }
Interfaces: Interf
}
```