



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στα Δίκτυα Υπηρεσιών

**Assisting Lecture 9b - Top Down SOAP Web
Services and Php Clients)**

Μύρων Παπαδάκης
Τμήμα Επιστήμης Υπολογιστών

Introduction to Service Networks

CS-592 – Spring 2015

Assisting Lecture 12: Top- Down WS with Apache
Tomcat, Axis2, Eclipse – Part II

Myron Papadakis

Outline

- Previous Lab: Tools for Top-Down Web Services and AreaService Example with clients
- Top-Down Web Services (cont'd)
 - Math Web Service
 - Loan Web Service
- Web Services Clients: Soap and Php

Eclipse Axis2 Lab Example



Lab Example Description

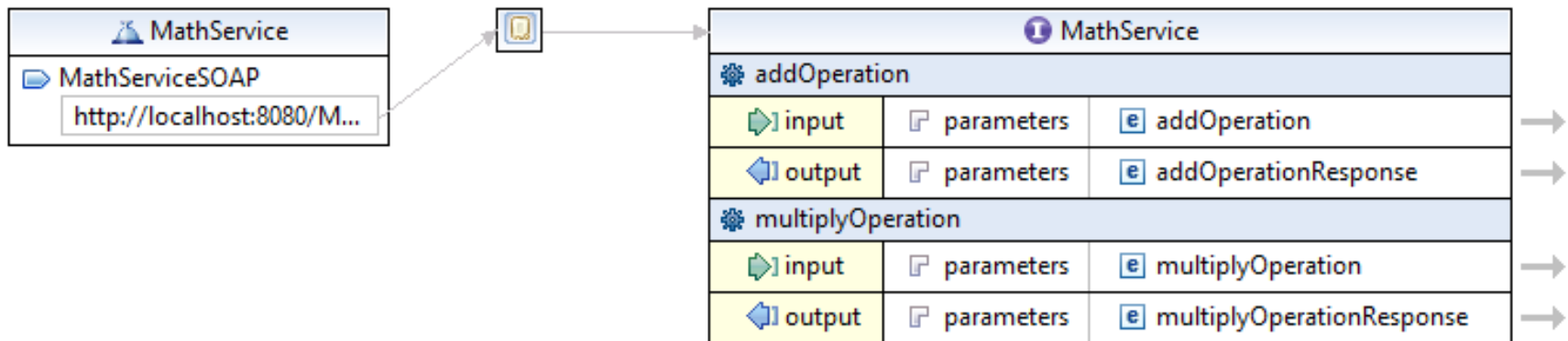
- Create a web service:
 - takes as input two integers
 - Returns the sum of the two integers
 - Returns the multiplication of the integers
- The Web Service will also have to be developed in Eclipse (as a Dynamic Web Project) using the top-down approach.

Lab Assignment

- Step 1: Interface of the Web Service
 - Create the WSDL file for the Math Service
 - Add 2 operations in the WSDL
 - Add: operation that adds two integers
 - Multiply: operation that multiplies two integers
- Step 2: Implementation of the service operations
 - Use Axis2 and create a top-down web service from the WSDL
 - Implement the operations defined in the WSDL file in Java
 - Test the Web service...

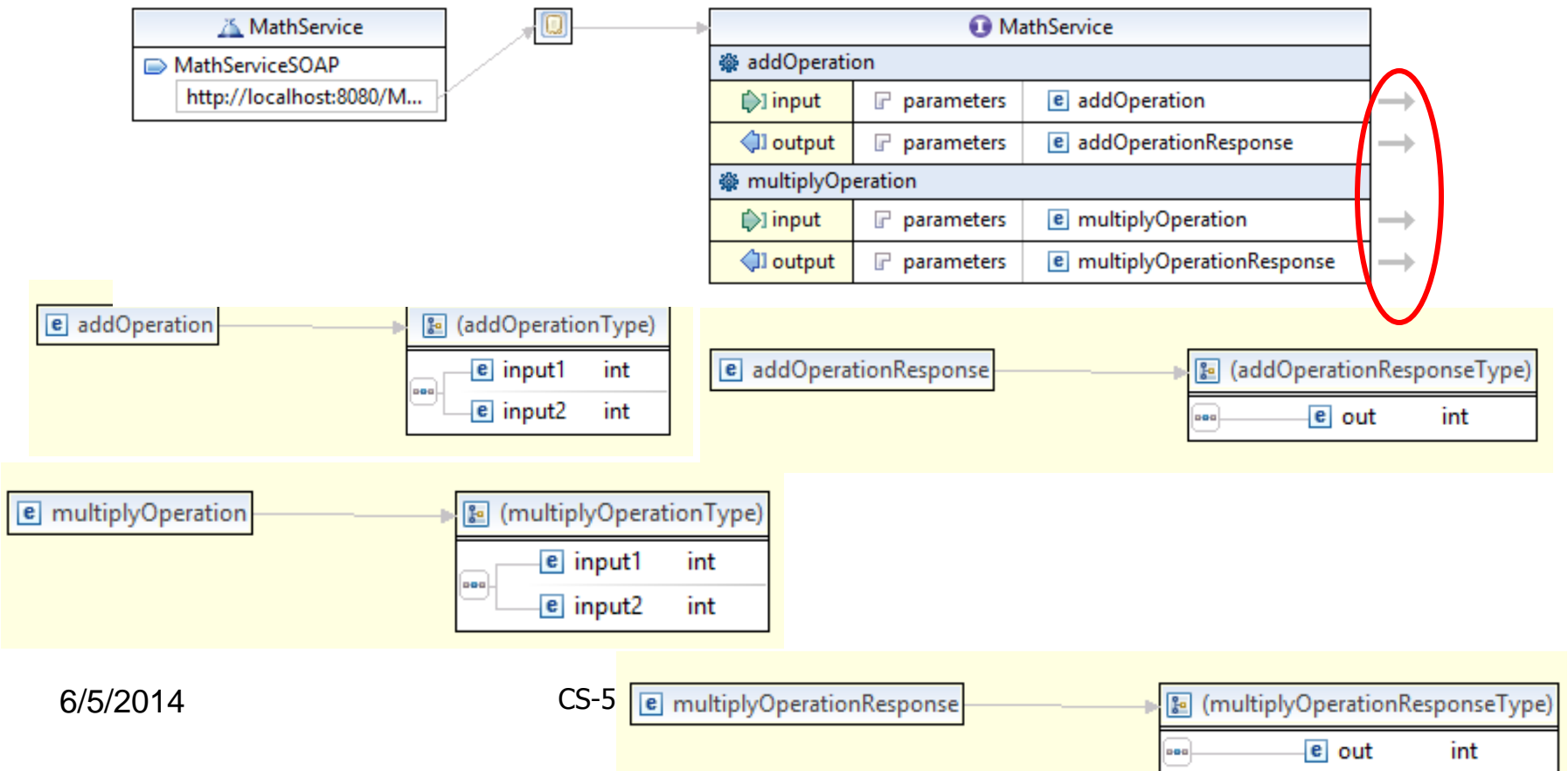
Developing the Math Web Service

- Create a new Dynamic Web Project called “Math”
- Create a folder WSDL on the Math project.
- Right click on this folder and then New → Other → Web Services → WSDL and name this file MathService.wsdl.
- Add the operations to the WSDL File



Developing the Math Web Service

- Both of these operations take as input 2 integers and return an integer as a results.



Developing the Math Web Service

The screenshot shows an IDE window with several tabs: MathIfProcess.bpel, MathService.wsdl, MathIfProcessArtifacts.wsdl, and MathService.wsdl. The main workspace displays a BPEL process diagram. On the left, a box labeled 'MathService' contains a sub-element 'MathServiceSOAP' with the URL 'http://localhost:8080/M...'. An arrow points from this box to a larger box on the right, also labeled 'MathService'. This box contains two operations: 'addOperation' and 'multiplyOperation'. Each operation has an 'input' and an 'output' section, both with 'parameters' and an 'e' icon. The 'addOperation' output is 'addOperationResponse' and the 'multiplyOperation' output is 'multiplyOperationResponse'. Below the workspace are tabs for 'Design' and 'Source'. At the bottom, there are tabs for 'Properties' and 'Console'. The 'Properties' tab is active, showing the 'port' properties:

General	Name: MathServiceSOAP
Documentation	Binding: MathServiceSOAP
Extensions	Address: <u>http://localhost:8080/Math/services/MathService</u>
	Protocol: SOAP

Implementing the operations

- Implement the multiply operation



```
MathlfProcess.bpel | MathlfProcessArtifac | MathService.wsdl | MathServiceSkeleton. X »1
*/
public class MathServiceSkeleton{

    /**
     * Auto generated method signature
     *
     * @param multiplyOperation
     * @return multiplyOperationResponse
     */

    public org.example.www.mathservice.MultiplyOperationResponse multiplyOperation
    (
        org.example.www.mathservice.MultiplyOperation multiplyOperation
    )
    {
        MultiplyOperationResponse r = new MultiplyOperationResponse();
        r.setOut(multiplyOperation.getInput1()*multiplyOperation.getInput2());
        return r;
    }
}
```

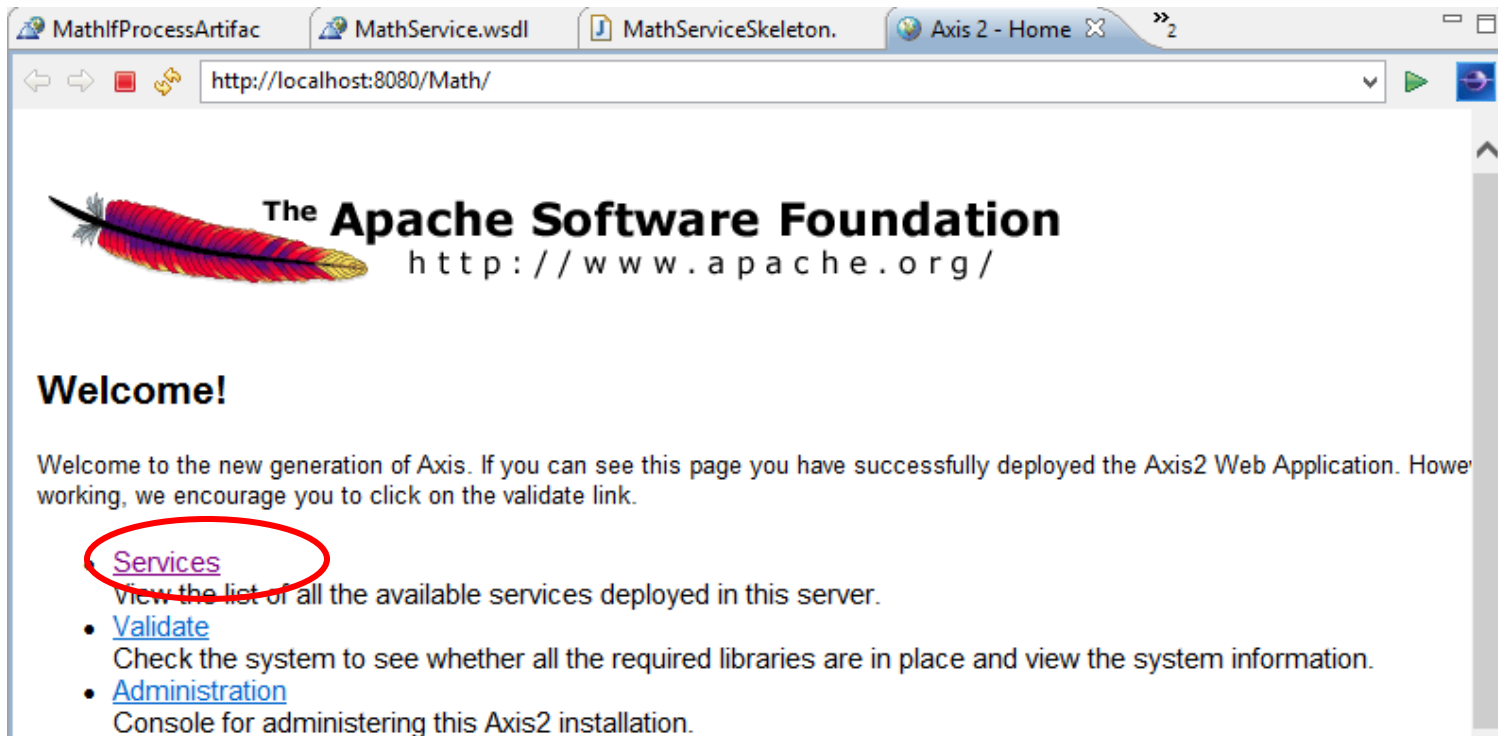
Implementing the operations

- Similarly implement the add operation in the MathServiceSkeleton.java

```
public org.example.www.mathservice.AddOperationResponse addOperation
(
    org.example.www.mathservice.AddOperation addOperation
)
{
    AddOperationResponse r = new AddOperationResponse();
    r.setOut(addOperation.getInput1()+addOperation.getInput2());
    return r;
}
```

Running the Math Project

- Right Click the Project and select Run As → Run on Server
- Restart the server if prompted to do so



Running the Math Project



The screenshot shows a web browser window with the following elements:

- Browser tabs: MathIfProcessArtifac, MathService.wsdl, MathServiceSkeleton., List Services
- Address bar: <http://localhost:8080/Math/services/listServices>
- Header: The Apache Software Foundation logo and <http://www.apache.org/>
- Section: **Available services**
- Service: **MathService** (circled in red)
- Service Description: **MathService**
- Service EPR: **<http://localhost:8080/Math/services/MathService>**
- Service Status: **Active**
- Section: *Available Operations*

Running the Math Project > Test Web Service

- Check if service is running

Web Services Explorer

MathService.wSDL | MathServiceSkeleton. | http://localhost:808 | Web Services Explorer

Web Services Explorer

Navigator

- WSDL Main
- file:/C:/workspace/Math/WSDL/...
- MathService
- MathServiceSOAP
- addOperation
- multiplyOperation

Actions

WSDL Binding Details

Shown below are the details for this SOAP <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

Operations

Name	Documentation
addOperation	--
multiplyOperation	--

Endpoints [Add](#) [Remove](#)

Status

IWAB0381I file:/C:/workspace/Math/WSDL/MathService.wSDL was successfully opened.
IWAB0388I Endpoints were successfully updated.

Running the Math Project > Test Web Service Operations

- Test the addOperation

The screenshot displays the Web Services Explorer interface. On the left, the Navigator pane shows a tree view with 'WSDL Main' expanded to 'MathServiceSOAP', where 'addOperation' is selected. The main pane is divided into three sections: 'Actions', 'Body', and 'Status'. The 'Actions' section contains a text box for 'Endpoints' with the value 'http://localhost:8080/Math/services/MathService'. Below it, the 'Body' section is expanded to show the 'addOperation' operation with two input fields: 'input1' with the value '3' and 'input2' with the value '4'. A red circle highlights the 'addOperation' section in the 'Body' pane. The 'Status' section at the bottom shows the 'addOperationResponse' with the output 'out (int): 7'.

Running the Math Project > Test Web Service Operations

- Test the multiplyOperation

The screenshot displays an IDE interface with two main panels. The left panel, titled 'Navigator', shows a project tree with the following structure:

- WSDL Main
 - file:/C:/workspace/Math/WSDL/...
 - MathService
 - MathServiceSOAP
 - addOperation
 - multiplyOperation

The 'multiplyOperation' is highlighted. The right panel, titled 'Actions', contains the following elements:

- Header: Enter the parameters for the WSDL operation "multiplyOperation" and click Go to invoke.
- Endpoints: A text box containing `http://localhost:8080/Math/services/MathService` with a dropdown arrow.
- Body: A section with a dropdown arrow, containing:
 - [multiplyOperation](#)
 - [input1](#) int: A text box containing the value `3`.
 - [input2](#) int: A text box containing the value `4`.

A red circle is drawn around the 'multiplyOperation' link and the input fields in the 'Body' section. Below the 'Actions' panel is a 'Status' panel with a dropdown arrow and a 'Source' link. It shows the following structure:

- Body
 - multiplyOperationResponse
 - out (int): 12

More Axis2 Web Services – Example 3

Loan Example > Scenario

- Suppose that a client wants to request a loan from a bank
- The client sends the loan request and receives a response afterwards whether the loan was approved or rejected.
- The client sends the following details when asking for a loan
 - Name
 - Address
 - Age
 - Annual Salary
 - Amount Requested
- The bank returns a response to the client. If the loan is rejected for some reason the bank also states the reason of the rejection

Loan Example > Scenario

- The bank examines the loan request and either approves the loan or not.
- **Age:** If the applicant is under 18 years old or above 65 years old the application is rejected. The response contains a message that informs the applicant about the reason
- **Annual Salary:** If the annual salary is less than 20000 then the application is rejected
- **Amount Requested:** If the amount that the client requests is greater than the amount of loan based on years to pay off loan then the loan is not approved.
 - $\text{limit} = \text{annualSalary} * \text{yearsToRepay} * 0.5$
 - $\text{yearsToRepay} = \text{AVERAGE_LIFE_EXPECTANCY} - \text{applicantAge}$;
- In any other case the application is approved.

Steps to create the WS

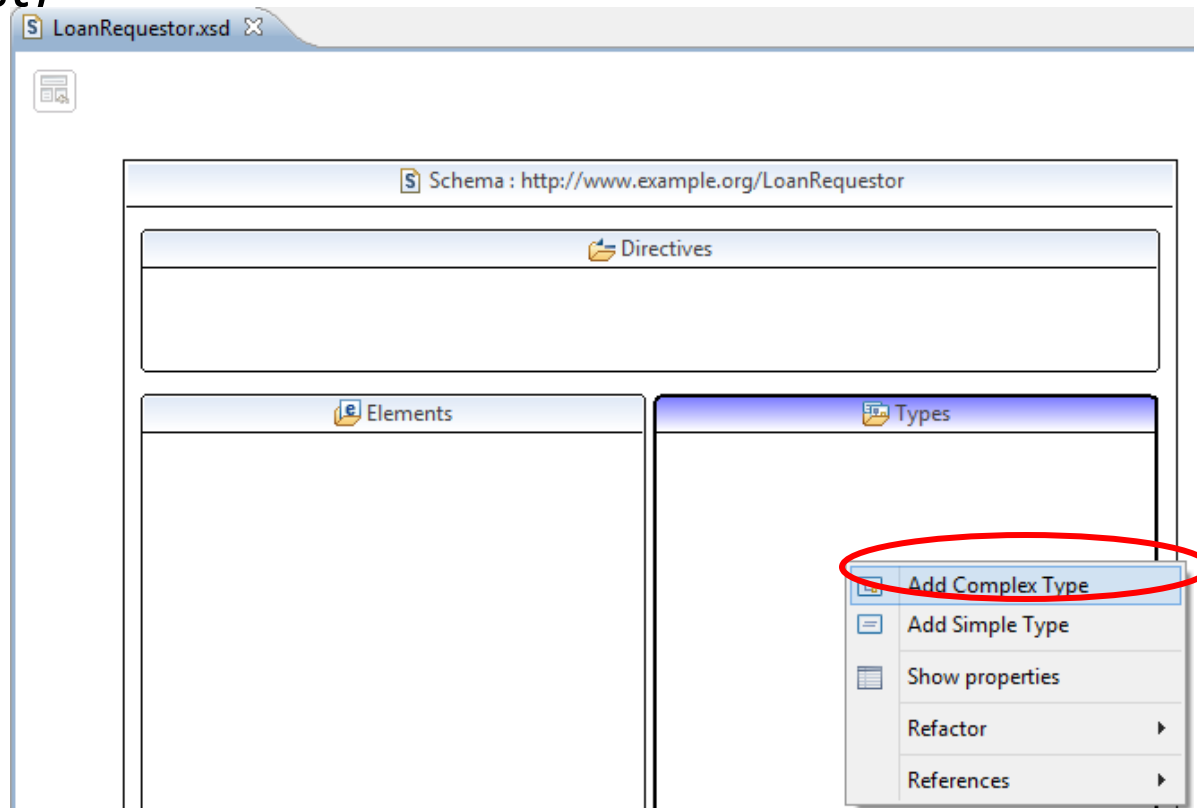
- Create the **XSD Schema** (loan request and loan response)
- Create the WSDL Document for the Web Service and provide the operations of the Web Service
 - Set the types of the request and the responses of the Web Service operations according to the XML Schema
- Create the Web Service from the WSDL using Axis2 (top-down approach)
- Implement the Web Service operations

Eclipse Dynamic Web Project

- Create a new Dynamic Web Project in Eclipse
- Create a folder “WSDL” in the project
 - In this folder we will place the XML schema and the WSDL document itself.
- Create a new XSD document (LoanRequestor.xsd)
 - File → New → Other → XML → XML Schema

XSD Schema (1/6)

- Right click in the Types area and select “Add Complex Type” and name it “processApplType” (will be used for the loan request)



XSD Schema (2/6)

The image shows a screenshot of an XSD Schema editor interface. The main window title is "Schema : http://www.example.org/LoanRequestor". The interface is divided into several panes:

- Directives:** A pane at the top, currently empty. A red circle with the number "1" is placed over this pane.
- Elements:** A pane below Directives, currently empty.
- Types:** A pane on the right side, containing a list of types. The type "processApplType" is selected and circled in red. A red circle with the number "2" is placed over this pane. A context menu is open over "processApplType", with the "Add Element" option highlighted. Other options include "Add Element Ref", "Add Any", "Add Attribute", "Add Attribute Ref", "Add Attribute Group Ref", "Add Any Attribute", "Add Sequence", "Add Choice", and "Add All".

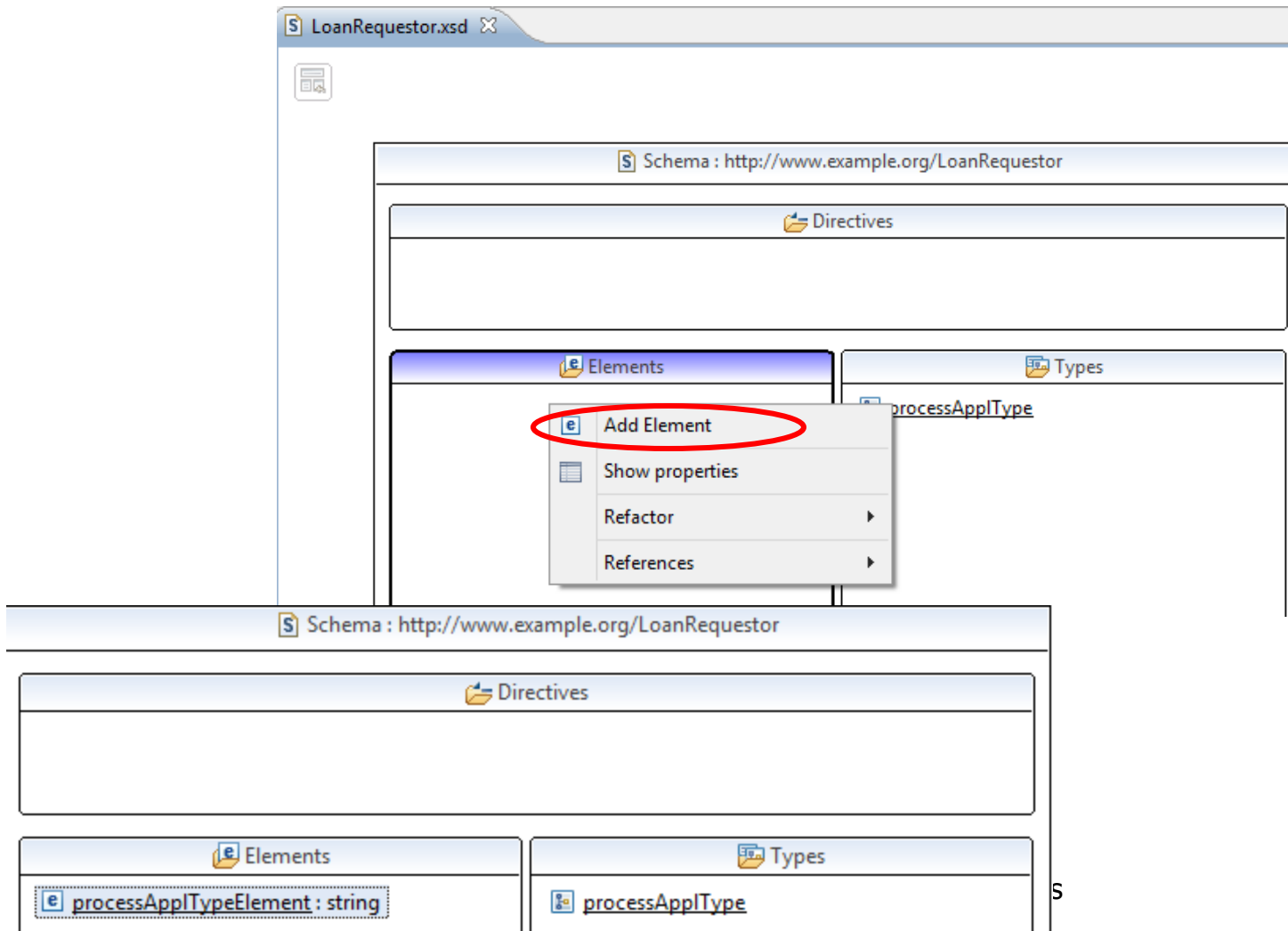
Below the main editor, there is a tab for "*LoanRequestor.xsd".

At the bottom left, there is a detailed view of the "processApplType" type, showing its structure as a sequence of elements:

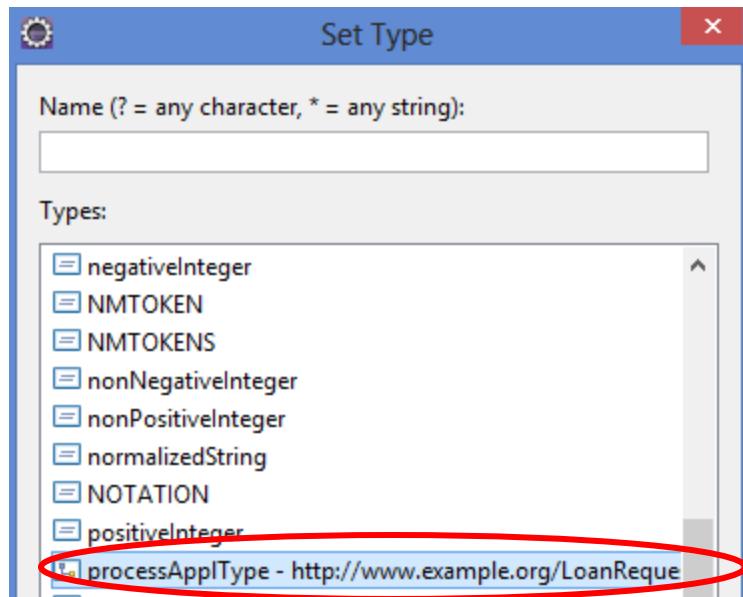
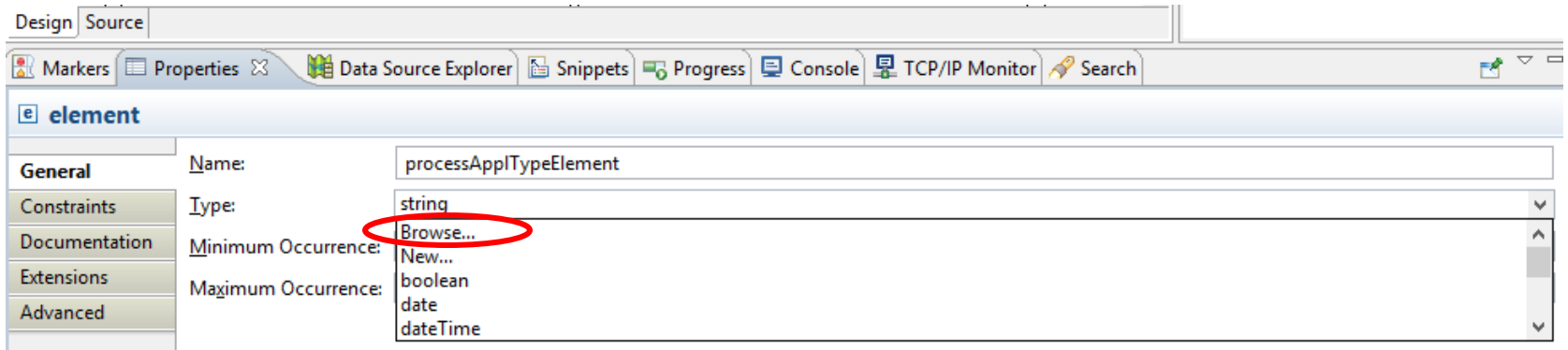
processApplType	
e name	string
e address	string
e age	int
e annualSalary	double
e amountRequested	double

A red circle with the number "3" is placed over this table.

XSD Schema (3/6)

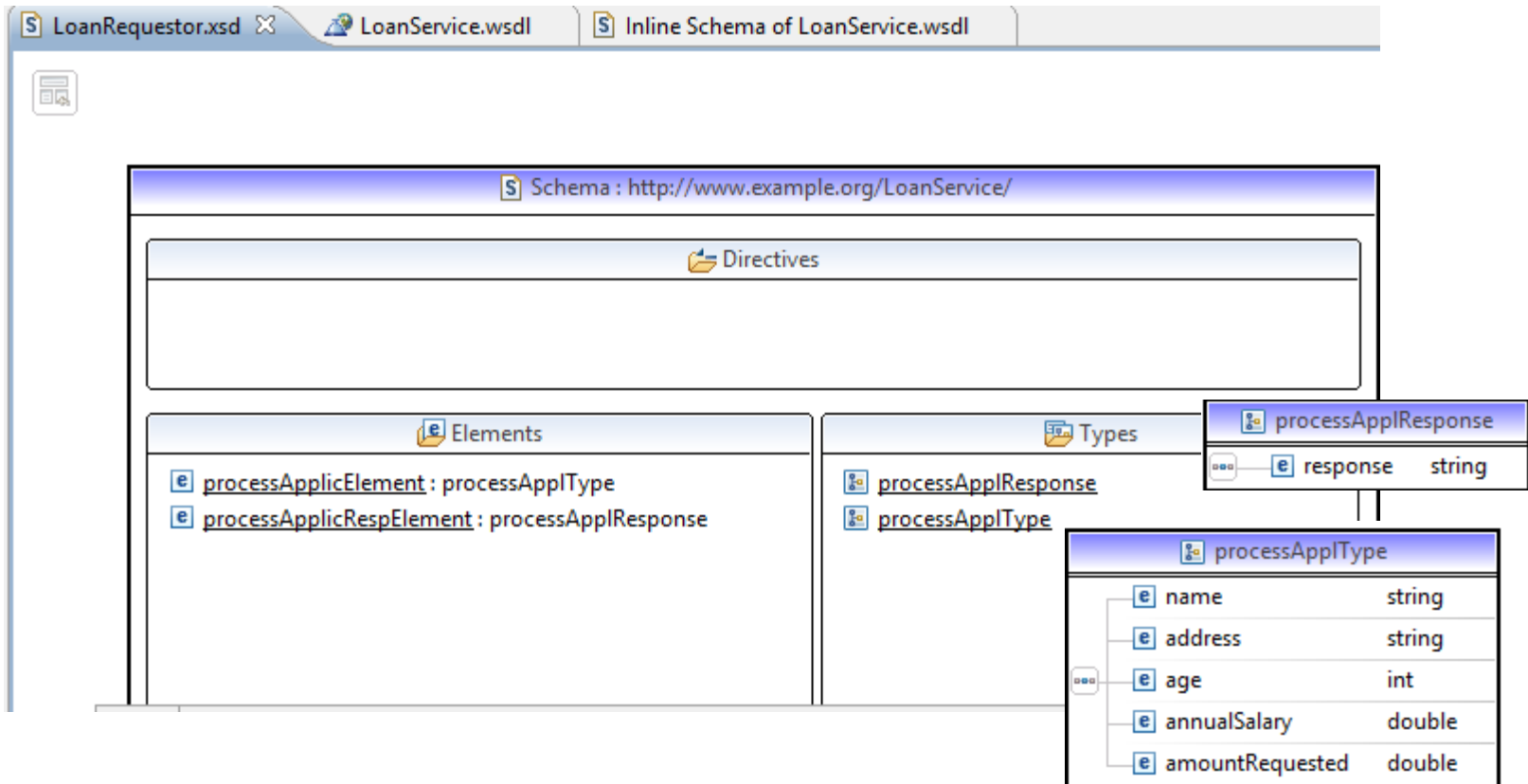


XSD Schema (4/6)



Follow similar steps for the response element..

XSD Schema (5/6)



XSD Schema (6/6)

The screenshot shows the Visual Studio IDE with the XSD Schema for `processApplicElement`. The Design view displays a box for `processApplicElement` with an arrow pointing to a detailed view of the `processAppType` complex type. This type contains five child elements: `name` (string), `address` (string), `age` (int), `annualSalary` (double), and `amountRequested` (double). The Properties window at the bottom shows the following details for the selected element:

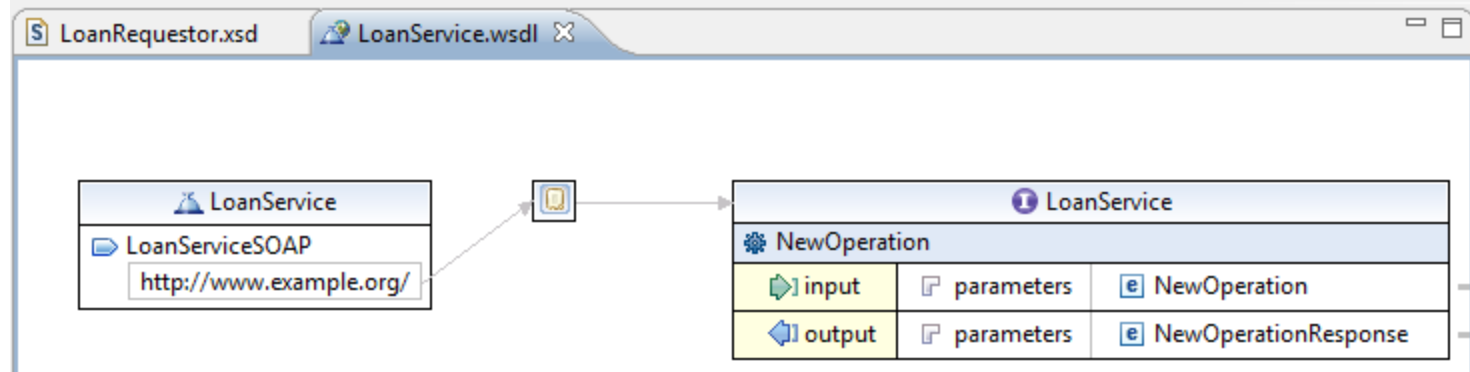
element	
General	Name: processApplicElement
Documentation	Type: tns:processAppType
Extensions	Minimum Occurrence:
Advanced	Maximum Occurrence:

The screenshot shows the Visual Studio IDE with the XSD Schema for `processApplicRespElement`. The Design view displays a box for `processApplicRespElement` with an arrow pointing to a detailed view of the `processAppResponse` complex type. This type contains one child element: `response` (string). The Properties window at the bottom shows the following details for the selected element:

element	
General	Name: processApplicRespElement
Documentation	Type: tns:processAppResponse
Extensions	Minimum Occurrence:
Advanced	Maximum Occurrence:

Create the WSDL

- Create the WSDL file for the Loan Web Service “LoanService.wsdl”
 - Right Click → File → New → Other → WSDL



- Add a single loanOperation (change the default operation)

Create the WSDL

The image shows a Visual Studio interface with two tabs: 'LoanRequestor.xsd' and 'LoanService.wsdl'. The 'LoanService.wsdl' tab is active, displaying a service diagram for 'LoanService'. The diagram shows a 'LoanServiceSOAP' port with the URL 'http://www.example.org/'. An arrow points from this port to a 'LoanService' service. The service has an 'loanOperation' operation. The operation's input parameters are 'loanOperation' and its output parameters are 'loanOperationResponse'. Below the diagram, the 'part' properties window is open, showing the 'Name' as 'parameters' and the 'Element' as 'loanOperation'. The 'Reference Kind' is set to 'Element', which is circled in red.

part	Name	Element	Reference Kind
input	parameters	loanOperation	<input type="radio"/> Type <input checked="" type="radio"/> Element
output	parameters	loanOperationResponse	<input type="radio"/> Type <input checked="" type="radio"/> Element

- Must import the XSD file we created earlier on in order to use the request and response elements...

WSDL File and XSD

LoanRequestor.xsd LoanService.wsdl Inline Schema of LoanService.wsdl

Schema : (no target namespace specified)

Directives

← LoanRequestor.xsd {http://www.example.org/LoanService/}

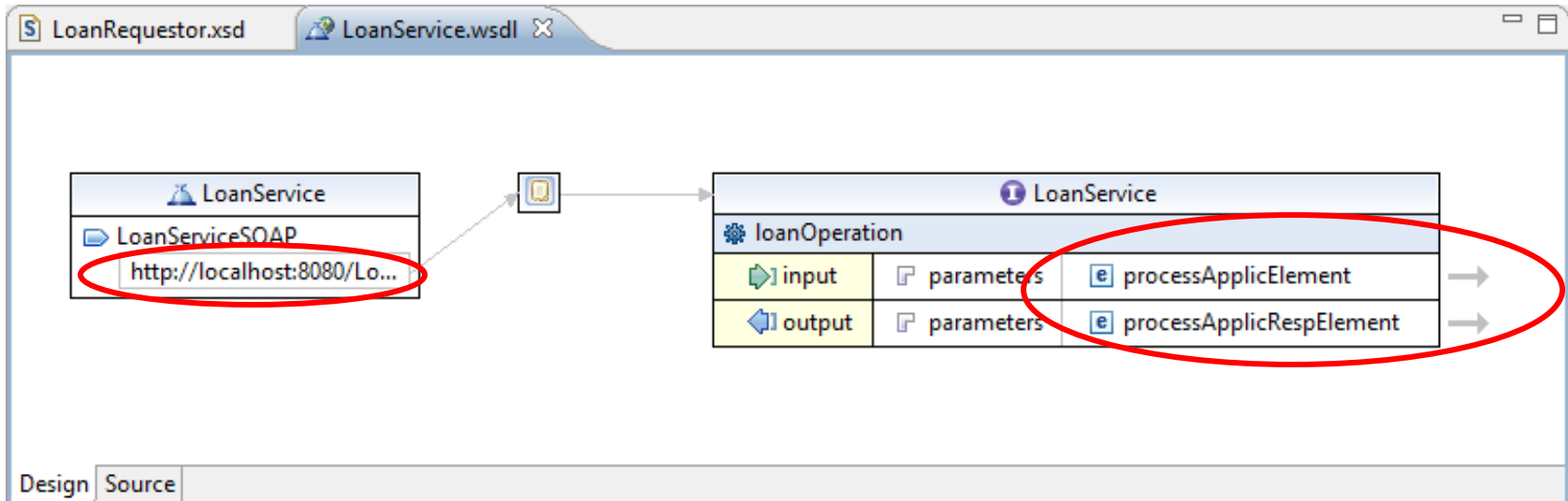
Elements

Types

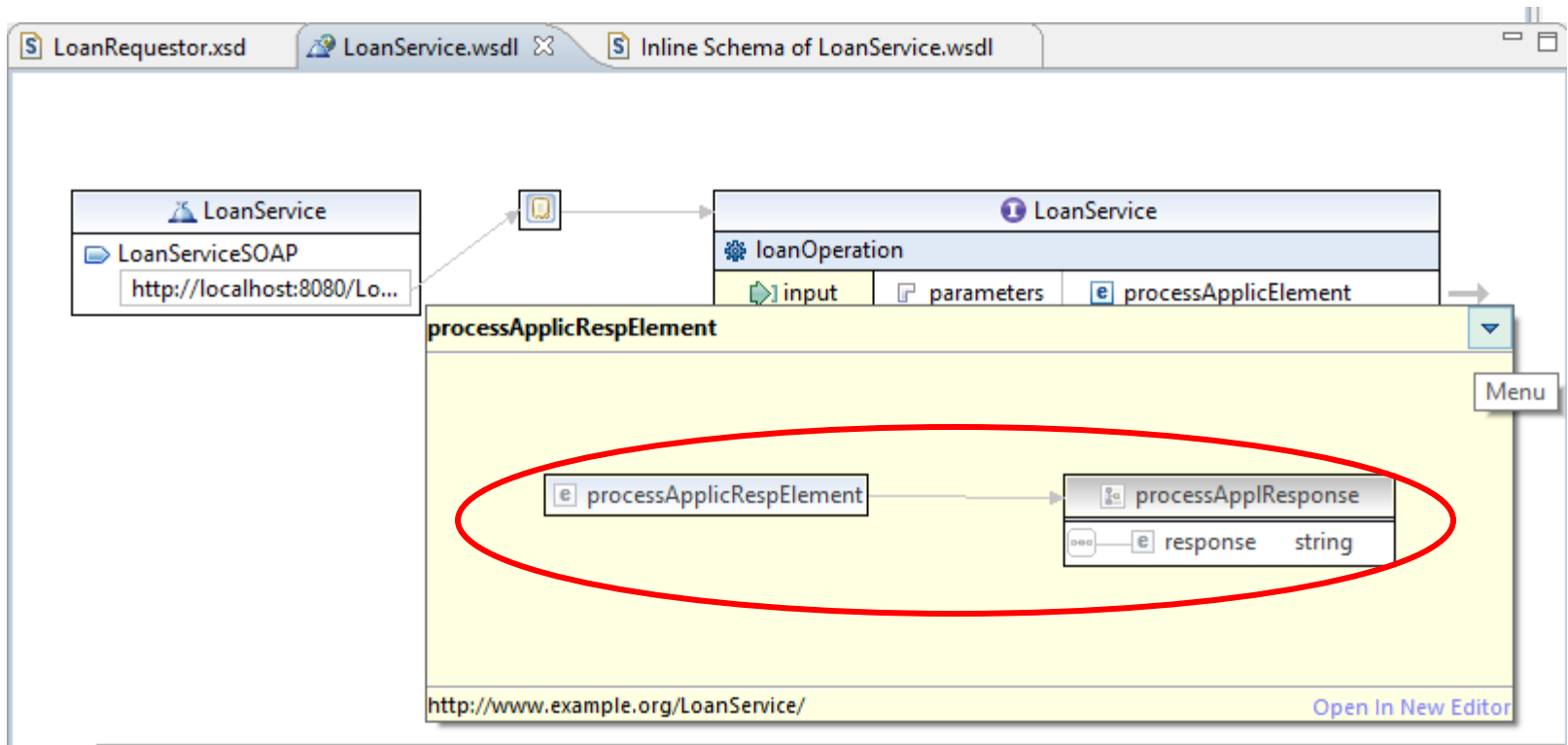
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.example.org/LoanService/" xmlns:w
<wsdl:types>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:import schemaLocation="LoanRequestor.xsd" namespace="http://www.example.org/LoanService/"></xsd:import>
  </xsd:schema>
</wsdl:types>
<wsdl:message name="loanOperationRequest">
```

WSDL File

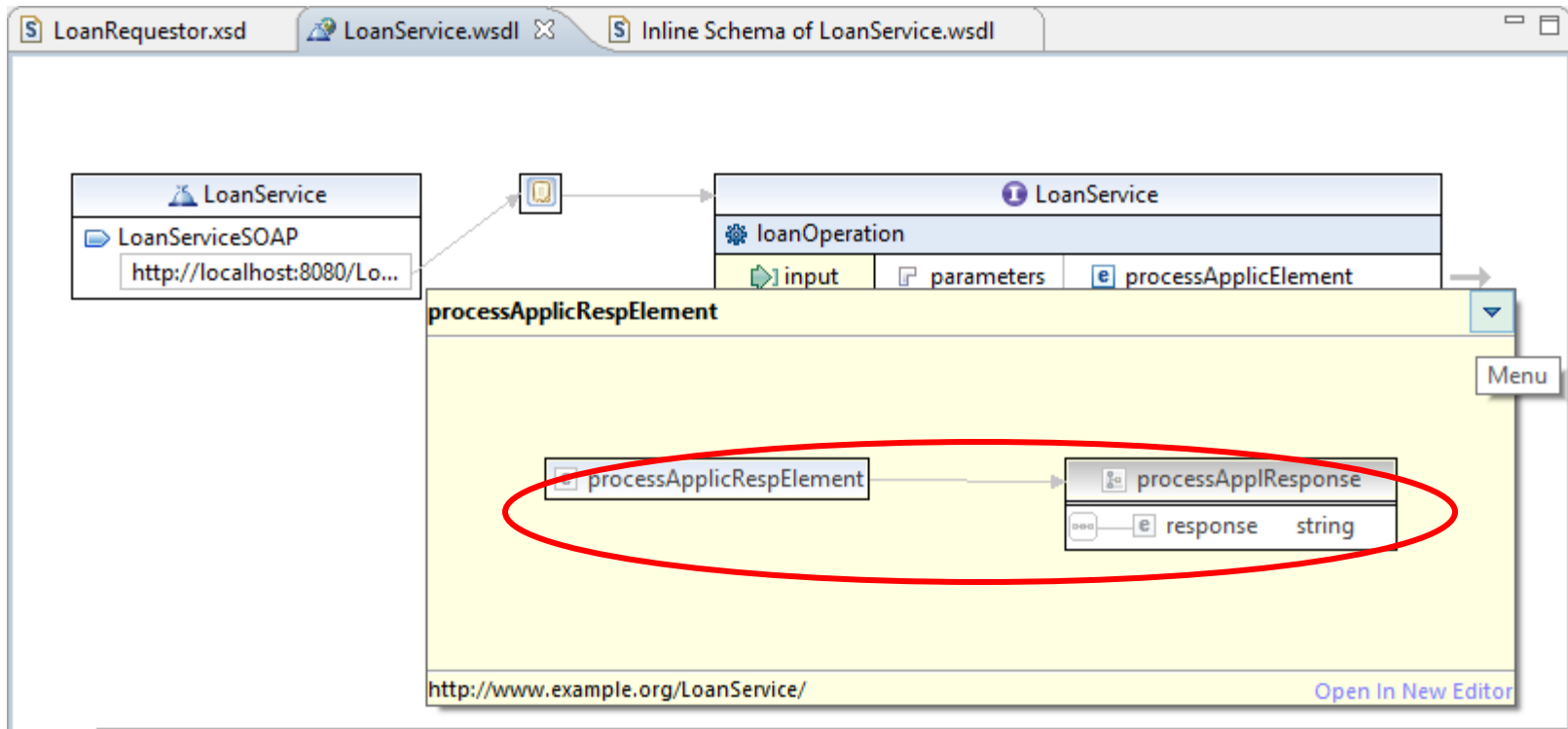
- Use the XSD messages that you created earlier for the request and the response accordingly.
- Do not forget to change the address of the LoanService (Properties View)
 - <http://localhost:8080/LoanWS/services/LoanService>
 - LoanWS: the name of the Eclipse Project



WSDL File



WSDL File



WSDL File

The screenshot displays the Visual Studio IDE with two tabs: LoanRequestor.xsd and LoanService.wsdl. The Design view shows a diagram where a port named LoanServiceSOAP (with address http://localhost:8080/Lo...) is connected to a service named LoanService. The service details are shown in a table below:

LoanService			
loanOperation			
input	parameters	processApplicElement	→
output	parameters	processApplicRespElement	→

The Properties window for the port is shown below, with the Address field circled in red:

port	
General	Name: LoanServiceSOAP
Documentation	Binding: LoanServiceSOAP
Extensions	Address: <u>http://localhost:8080/LoanWS/services/LoanService</u>
	Protocol: SOAP

Axis2 Web Service from WSDL

The image shows a screenshot of the Axis2 Web Service wizard. The main window is titled "Web Service" and contains the following elements:

- Web Services** section: "Select a service implementation or definition and move the sliders to set the level of service and client generation." A slider is set to "Start service".
- Web service type:** "Top down Java bean Web Service" (dropdown menu).
- Service definition:** "/LoanWS/WSDL/LoanService.wsdl" (dropdown menu) with a "Browse..." button.
- Client type:** "Java Proxy" (dropdown menu).
- No client** section: A slider is set to "No client".
- Configuration:** A section for server configuration.
- Start server** button: A button to start the server.
- Publish the Web service** checkbox: A checkbox at the bottom left.

The "Server startup" dialog box is overlaid on the main window. It contains the following text:

Server startup

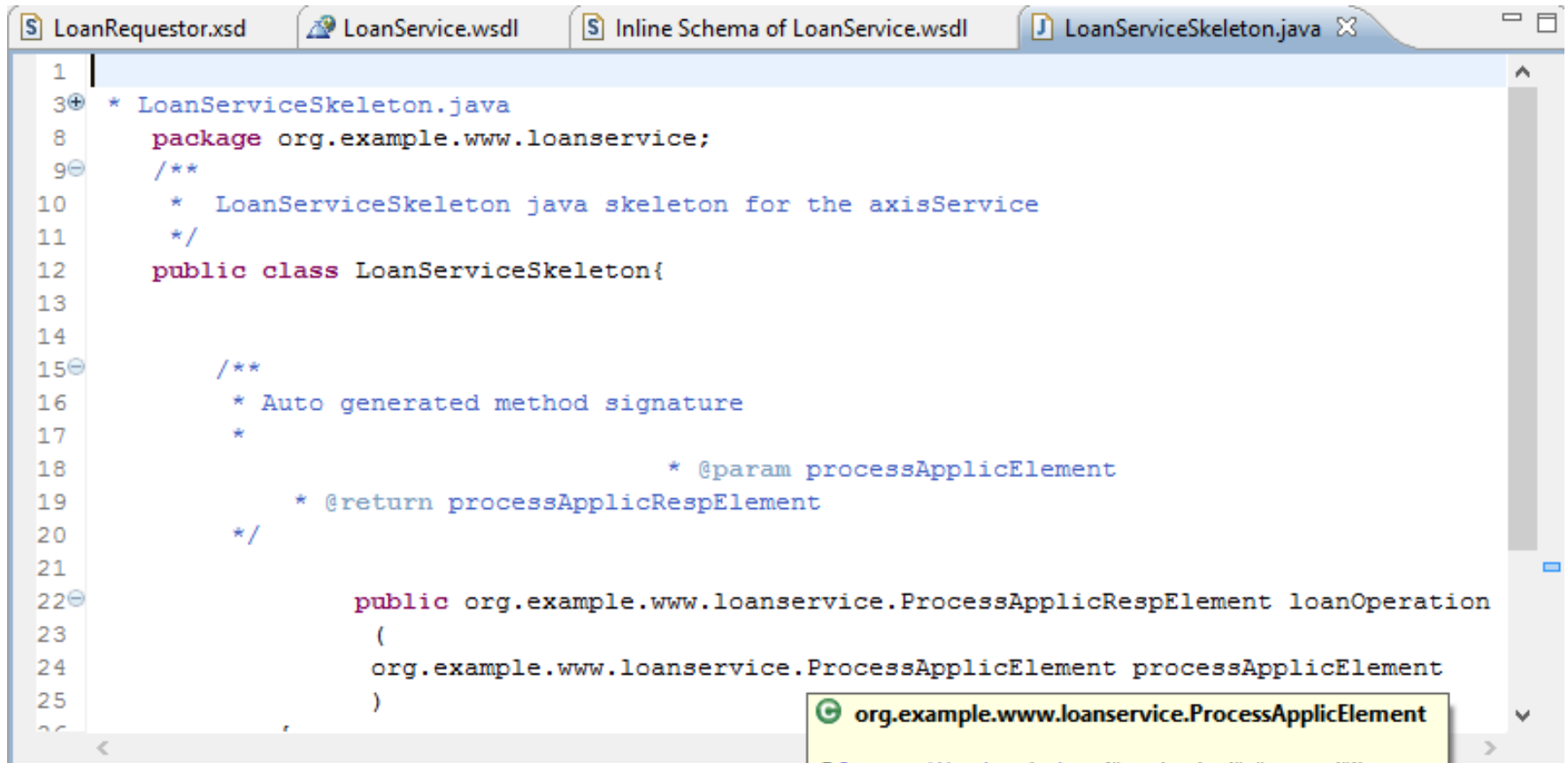
Start the server from this page.

In order to proceed the server "Tomcat v7.0 Server at localhost" must be started. Once the server is started the "next" button will be enabled. The "back" button can be used while the server is starting to change any previous settings in this wizard.

Currently the server is starting.

Publishing to Tomcat v7.0 Server at localhost...: IWAB0200I Starting server.

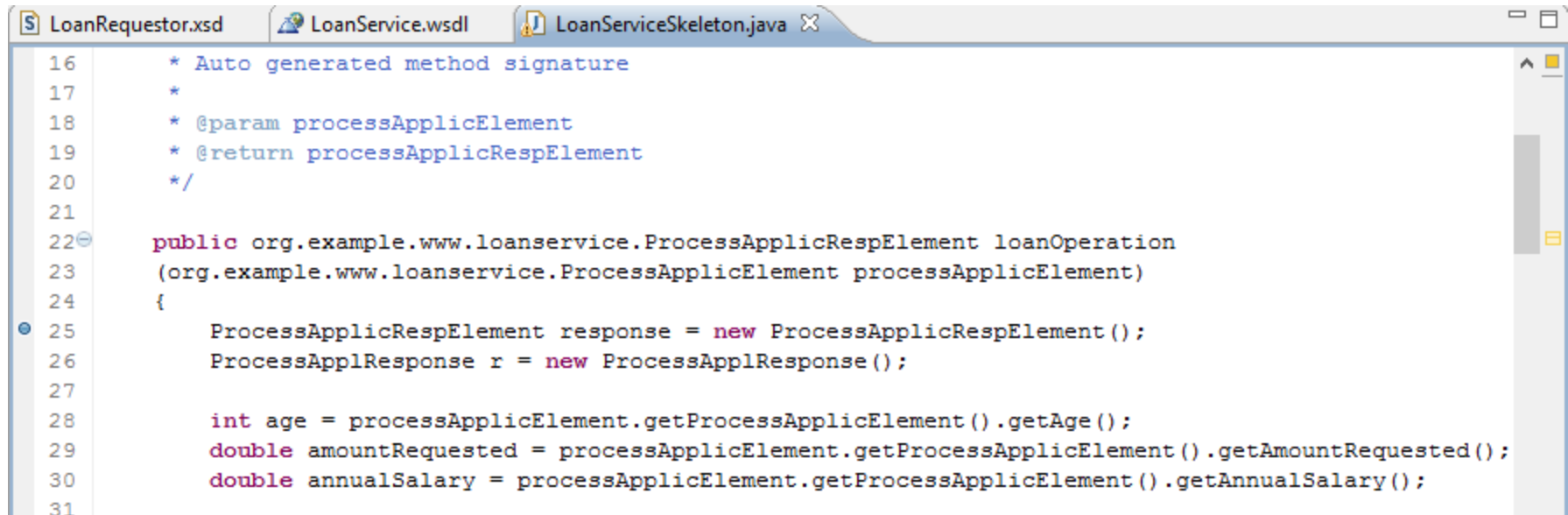
Implementing the operations



```
1 |
3+ * LoanServiceSkeleton.java
8  package org.example.www.loanservice;
9- /**
10  * LoanServiceSkeleton java skeleton for the axisService
11  */
12  public class LoanServiceSkeleton{
13
14
15-  /**
16  * Auto generated method signature
17  *
18  * @param processApplicElement
19  * @return processApplicRespElement
20  */
21
22-  public org.example.www.loanservice.ProcessApplicRespElement loanOperation
23  (
24  org.example.www.loanservice.ProcessApplicElement processApplicElement
25  )
26
```

org.example.www.loanservice.ProcessApplicElement

Implementing the operations



```
LoanRequestor.xsd | LoanService.wsdl | LoanServiceSkeleton.java x
16      * Auto generated method signature
17      *
18      * @param processApplicElement
19      * @return processApplicRespElement
20      */
21
22 public org.example.www.loanservice.ProcessApplicRespElement loanOperation
23 (org.example.www.loanservice.ProcessApplicElement processApplicElement)
24 {
25     ProcessApplicRespElement response = new ProcessApplicRespElement();
26     ProcessApplResponse r = new ProcessApplResponse();
27
28     int age = processApplicElement.getProcessApplicElement().getAge();
29     double amountRequested = processApplicElement.getProcessApplicElement().getAmountRequested();
30     double annualSalary = processApplicElement.getProcessApplicElement().getAnnualSalary();
31
```

⋮

Test if the WS is running

LoanRequestor.xsd LoanService.wsdl Inline Schema of Loan LoanServiceSkeleton.j List Services

http://localhost:8080/LoanWS/services/listServices

Available Operations

- getVersion

[LoanService](#)

Service Description : LoanService

Service EPR : http://localhost:8080/LoanWS/services/LoanService

Service Status : Active

Available Operations

- loanOperation

WS in the Web Services Browser

The screenshot displays the Web Services Explorer interface. The top navigation bar includes tabs for 'LoanService.wsdl', 'LoanServiceSkeleton.', 'List Services', and 'Web Services Explorer'. The main window is divided into three panes:

- Navigator:** Shows a tree view of the WSDL structure. The path is: WSDL Main > file:/C:/workspace/LoanWS/WSDL > LoanService > LoanServiceSOAP > loanOperation. The 'loanOperation' node is selected and highlighted.
- Actions:** Contains the 'Invoke a WSDL Operation' section. It prompts the user to 'Enter the parameters for the WSDL operation "loanOperation" and click Go to invoke.' Below this, there is a text box for 'Endpoints' containing the URL 'http://localhost:8080/LoanWS/services/LoanService'. A 'Body' section is expanded to show a parameter named 'processApplicElement' with a sub-parameter 'name' of type 'string'.
- Status:** Displays two messages: 'IWAB0381I file:/C:/workspace/LoanWS/WSDL/LoanService.wsdl was successfully opened.' and 'IWAB0388I Endpoints were successfully updated.'

Testing the WS

The screenshot shows the Web Services Explorer interface. The top bar contains several tabs: 'LoanService.wsdl', 'LoanServiceSkeleton.', 'List Services', and 'Web Services Explore'. The main area is divided into two panes. The left pane, titled 'Navigator', shows a tree view of the WSDL structure: 'WSDL Main' -> 'file:/C:/workspace/LoanWS/WSDL' -> 'LoanService' -> 'LoanServiceSOAP' -> 'loanOperation'. The right pane, titled 'Actions', shows the details for the 'processApplicElement' action. It lists the following parameters with their data types and corresponding input fields:

- name** string
- address** string
- age** int
- annualSalary** double
- amountRequested** double

Testing the WS

endpoints

▼ **Body**

▼ [processApplicElement](#)

[name](#) string

[address](#) string

[age](#) int

[annualSalary](#) double

[amountRequested](#) double

▼ **Status**

▼ [processApplicRespElement](#)

response (string): Loan Application APPROVED.

Testing the WS

▼ Body

▼ [processApplicElement](#)

[name](#) string
George

[address](#) string
Kapodistrias 21

[age](#) int
30

[annualSalary](#) double
12000

[amountRequested](#) double
10000


Go Reset

i Status

▼ [processApplicRespElement](#)

response (string): Loan Application REJECTED - Reason: Annual Salary \$12000.0 too low. Annual Salary needs to be over \$20000.0 to qualify.

Testing the WS

 Actions

▼ **Body**

▼ [processApplicElement](#)

[name](#) string
Mary

[address](#) string
Inatou 10

[age](#) int
67

[annualSalary](#) double
50000


[amountRequested](#) double
5000

i Status

▼ processApplicRespElement

response (string): Loan Application REJECTED - Reason: Over-aged 67. Age needs to be under 65 years to qualify.

Testing the WS

 **Invoke a WSDL Operation**

Enter the parameters for the WSDL operation "loanOperation" and click **Go** to invoke.

Endpoints

▼ **Body**


▼ [processApplicElement](#)

[name](#) string

[address](#) string

[age](#) int


[annualSalary](#) double


 **Status**

▼ [processApplicRespElement](#)

response (string): Loan Application REJECTED - Reason: Under-aged 17. Age needs to be over 18 years to qualify.

Testing the WS

 Actions



▼ **Body**

▼ [processApplicElement](#)

[name](#) string

[address](#) string

[age](#) int

[annualSalary](#) double

[amountRequested](#) double

i Status

▼ [processApplicRespElement](#)

response (string): Loan Application REJECTED - Reason: You are asking for too much \$4000000.0. Annual Salary \$70000.0, Age 45 years. Your limit is \$875000.0

Loan Client Example using PHP

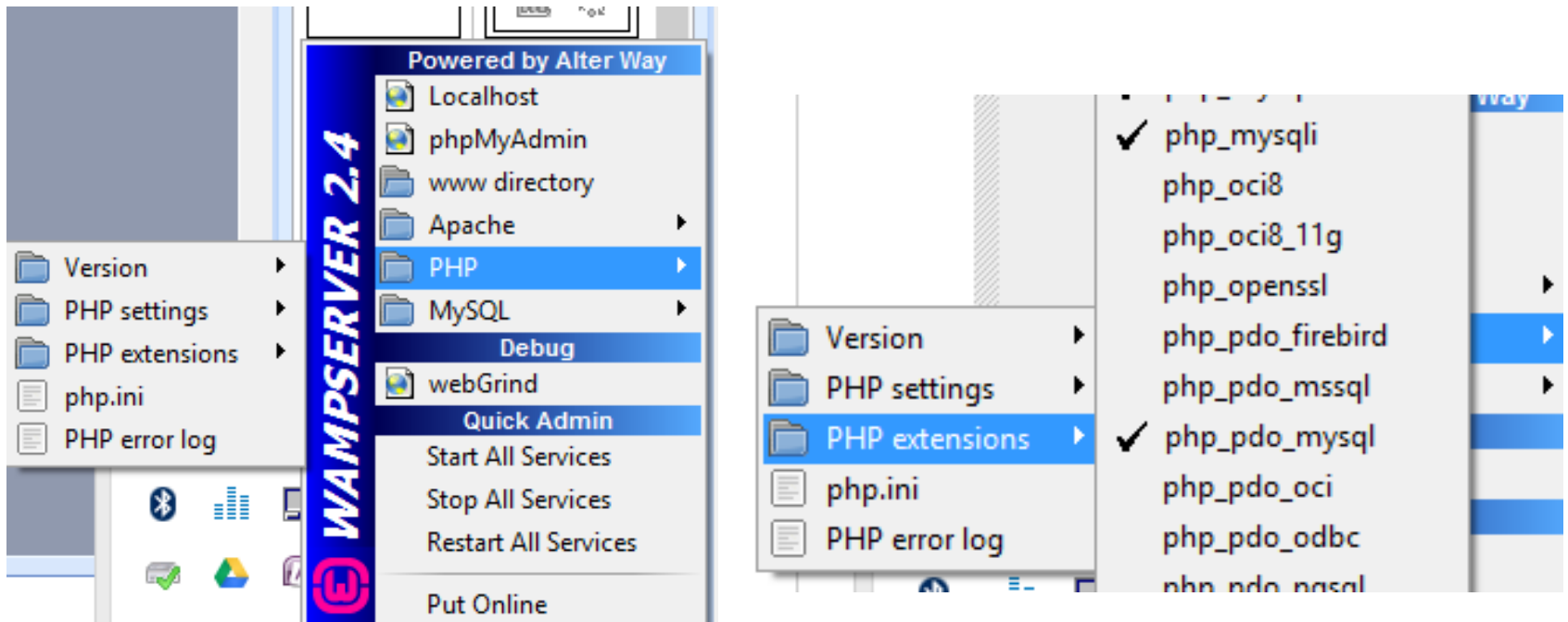
For writing a client manually in Java or testing the Web Service through the Web Services Browser, SOAP UI see the previous Assisting Lecture

Prerequisites

- Php must be installed on your PC
- A program like XAMP or WAMP can be very helpful when developing php and mysql applications
- Download wamp for example
- Work Done in C:\wamp\www
- Place a test.php there to see if php is running

Php and SOAP

- Enable php_soap extensions(or delete the ‘;’ from the corresponding extension of the php.ini file)
- Native SoapClient, see <http://php.net/soapclient>

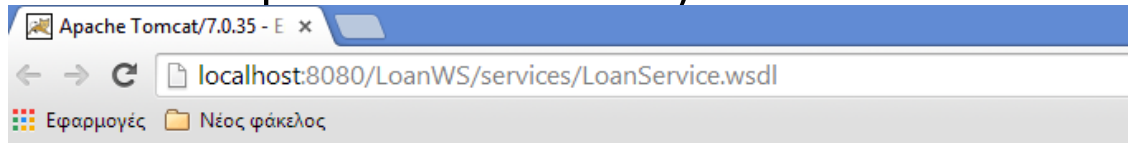


Steps

- A php client can be developed in a very similar way as any other client (jsp client, java client, etc)
 1. Check that the Web Service is running (check the wsdl file)
 - Most common you will have to start Tomcat from the Eclipse IDE (assuming that use one)
 2. Develop the php client
 - Using for example a simple editor like Notepad++
 3. Test the Web service using the client
 1. Through your Web browser (Firefox, Chrome, etc)

Step 1

- Before testing your Web Service (either in php or wherever) always make sure that the service is running.
- Otherwise you will retrieve a fault similar to the following one (after the development of the client)



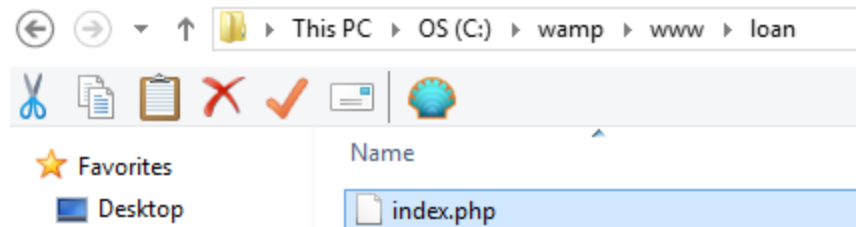
This XML file does not appear to have any style information associated with it. The document tree is shown below

```
▼<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://w
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="LoanService" targetNamespa
▼<wsdl:types>
  ▼<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:import namespace="http://www.example.org/LoanService/" schemaLocation="Lo
  </xsd:schema>
</wsdl:types>
▼<wsdl:message name="loanRequest">
  <wsdl:part name="loanRequest" type="tns:loanRequest"/>
</wsdl:message>
▼<wsdl:message name="loanResponse">
  <wsdl:part name="loanResponse" type="tns:loanResponse"/>
</wsdl:message>
▼<wsdl:portType name="LoanService">
  ▼<wsdl:operation name="loanOperation">
    <wsdl:input message="tns:loanOperationRequest"></wsdl:input>
    <wsdl:output message="tns:loanOperationResponse"></wsdl:output>
  </wsdl:operation>
</wsdl:portType>
</wsdl:definitions>
```

Error!SOAP-ERROR: Parsing WSDL: Couldn't load from 'http://localhost:8080/LoanWS/services/LoanService?wsdl' : failed to load external entity "http://localhost:8080/LoanWS/services/LoanService?wsdl"

Step 2

- Create a php file in the folder you created previously (i.e. index.php)



- We want to create an input form (html) so that users provide their details (name, address, salary, amount requested, etc)
 - You can use some css to style the form etc (e.g. style.css)
- The Web Service will be invoked when users submit their form
- The Web Service responds and users see the result of the loan request (approved or rejected and for which reason)
- In the following slides, a simple example follows (note that I do not tackle exceptional states, etc but you should do..)

Index.php

```
<!DOCTYPE HTML>
<html>
<head>
<link type="text/css" rel="stylesheet" href="style.css">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="author" content="Myron Papadakis">
<title>PHP Client for Loan Scenario</title>
</head>
<body>
  <header class="main">
    <h1>Loan Request</h1>
  </header>
  <section class="main">
    <?php
      if(isset($_POST['name']) && isset($_POST['address']) && isset($_POST['age'])
        && isset($_POST['salary']) && isset($_POST['amount'])) {
          $name = $_POST['name'];
          $address = $_POST['address'];
          $age = $_POST['age'];
          $salary = $_POST['salary'];
```

Index.php

```
$amount = $_POST['amount'];
}
if(isset($_POST['submit'])) {
    $wsdl = 'http://localhost:8080/LoanWS/services/LoanService?wsdl';
    $debug = false;
    try {
        $client = new SoapClient($wsdl);
        if($debug) {
            var_dump ($client->__getFunctions());
            var_dump($client->__getTypes());
        }
        $params = array ('name' => $name, 'address' => $address, 'age' => $age, 'annualSalary' => $salary, 'amountRequested' => $amount);
        $result = $client->loanOperation($params);
        if (is_soap_fault($result)) {
            trigger_error("SOAP Fault: (faultcode: {$result->faultcode}, faultstring: {$result->faultstring})", E_USER_ERROR);
        }
        echo $result->response;
    }
    catch (Exception $e) {
        echo "Error!";
        echo $e -> getMessage ();
    }
}
```

```
array (size=1)
  0 => string 'processApplResponse loanOperation(processApplType $parameters)' (length=62)
```

```
array (size=2)
  0 => string 'struct processApplType {
    string name;
    string address;
    int age;
    double annualSalary;
    double amountRequested;
  }' (length=114)
  1 => string 'struct processApplResponse {
    string response;
  }' (length=48)
```

Index.php

```
    }  
    ?>  
    <form method="post" action="index.php">  
        <label>Name:</label>  
        <input name="name" placeholder="Goes Here">  
        <label>Address:</label>  
        <input name="address" placeholder="Goes Here">  
        <label>Age:</label>  
        <input name="age" placeholder="Goes Here">  
        <label>Salary:</label>  
        <input name="salary" placeholder="Goes Here">  
        <label>Amount requested:</label>  
        <input name="amount" placeholder="Goes Here">  
        <input id="submit" name="submit" type="submit" value="Submit">  
    </form>  
</section>  
</body>  
</html>
```

Step 3

Loan Request

Name:

Address:

Age:

Salary:

Amount requested:

```
<form method="post" action="index.php">
  <label>Name:</label>
  <input name="name" placeholder="Goes Here">
  <label>Address:</label>
  <input name="address" placeholder="Goes Here">
  <label>Age:</label>
  <input name="age" placeholder="Goes Here">
  <label>Salary:</label>
  <input name="salary" placeholder="Goes Here">
  <label>Amount requested:</label>
  <input name="amount" placeholder="Goes Here">
  <input id="submit" name="submit" type="submit" value="Submit">
</form>
```

Index.php (part)

```
label {
  display:block;
  margin-top:10px;
  letter-spacing:1px;
}

/* This section centers our complete page */
.main {
  display:block;
  margin:0 auto;
  width:500px;
}

/* This section centers the form inside our web page*/
form {
  margin:0 auto;
  width:420px;
}
```

Style.css (part)

Demonstration Example

Loan Request

Name:

Address:

Age:

Salary:

Amount requested:

Loan Request

Loan Application REJECTED - Reason: Annual Salary \$10000.0 too low.
Annual Salary needs to be over \$20000.0 to qualify.

Name:

Address:

Age:

Salary:

Amount requested:

References

- **Top-down Approach:**
http://www.eclipse.org/webtools/community/tutorials/TopDownAxis2WebService/td_tutorial.html
- <http://www.soapui.org/Working-with-soapUI/getting-started.html>
- <http://php.net/manual/en/class.soapclient.php>

Τέλος Ενότητας



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «**Εκπαίδευση και Δια Βίου Μάθηση**» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>

•Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

•Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Μύρων Παπαδάκης. «**Εισαγωγή στα Δίκτυα Υπηρεσιών. Assisting Lecture 9b - Top Down SOAP Web Services and Php Clients**)». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: <https://elearn.uoc.gr/course/view.php?id=416/>