Service Oriented Architecture

Content

Introduction

A SOA Course! Isn't SOA Dead?

What's Special About this Course?

Course Structure

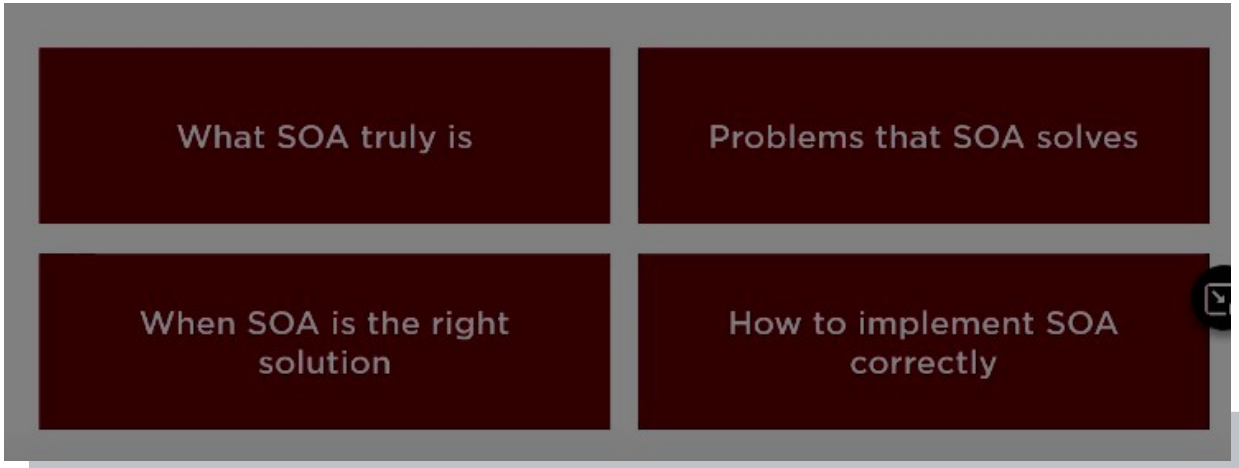
Introduction

In 2009, an article titled SOA is Dead (By ***Anne thomas***) caused various discussion and many people agreed;

SOA problems

- False expectations from implementing SOA
- Misconceptions

We will teach you



In fact, it'is very much

What's Special About this?

Common Problems in SOA Publications

Technology in nature ("How" aspect)

Skip "Why" and "What" aspects

Theoretical in nature

What are the real-world challenges?

How This Course Tackles These Shortcomings?

Understand the business motivations behind SOA

Covers a real-world scenario

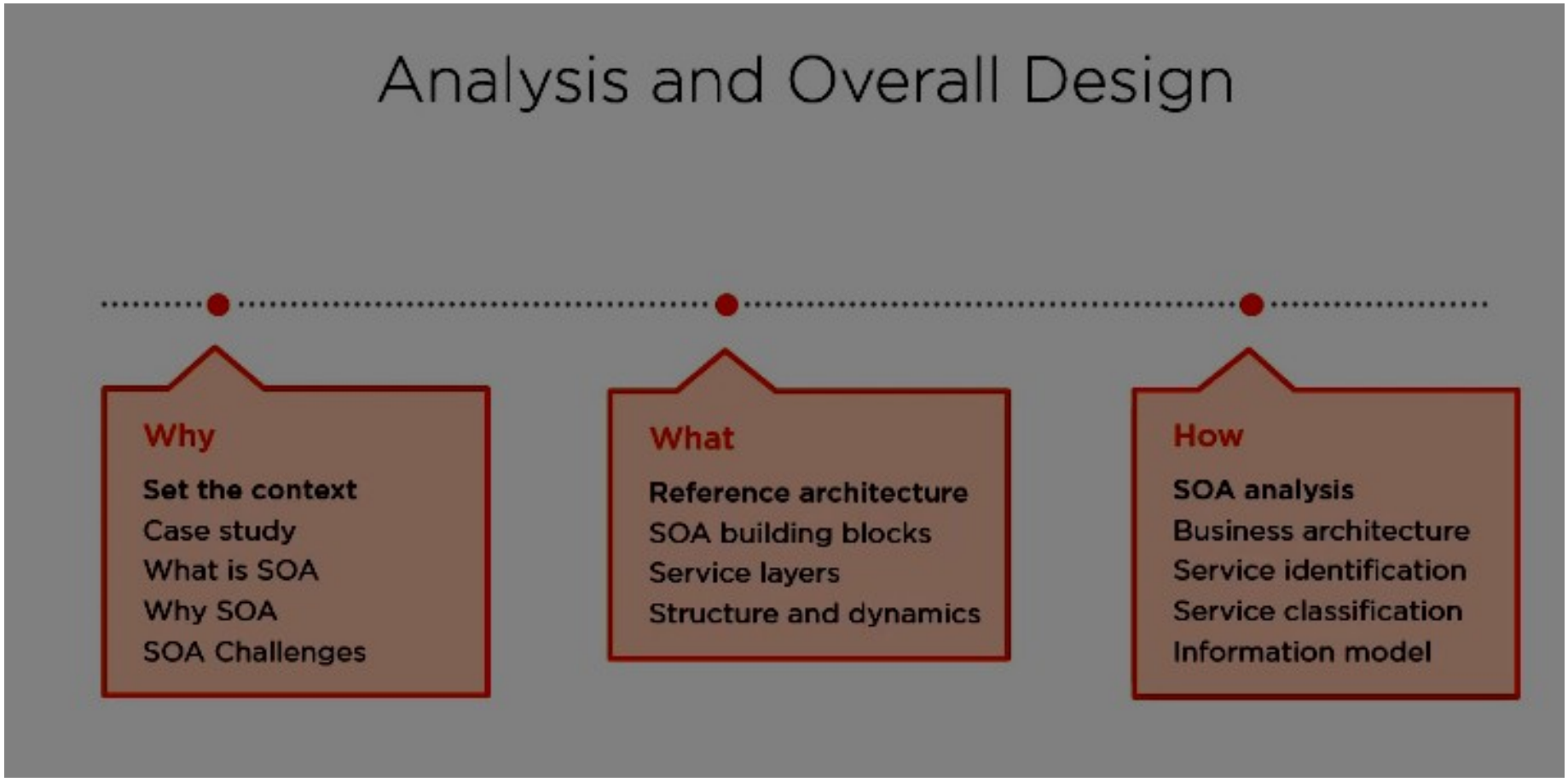
Structure

as we start examining SOA at the high-level architecture,

Part I

Structure

Part I



Structure

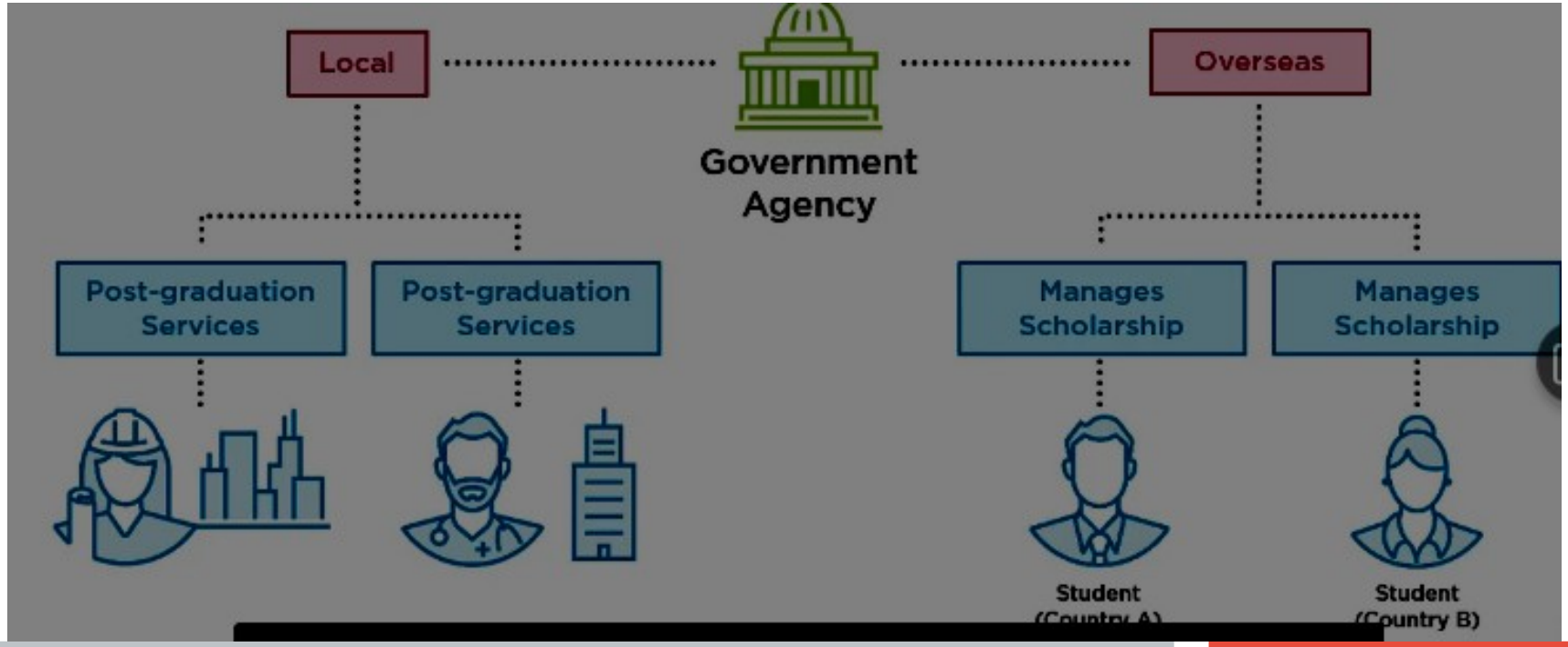
Part II

Design and Implementation

- SOA design principles
- Quality attributes
- Service design
- Service implementation

Case Study

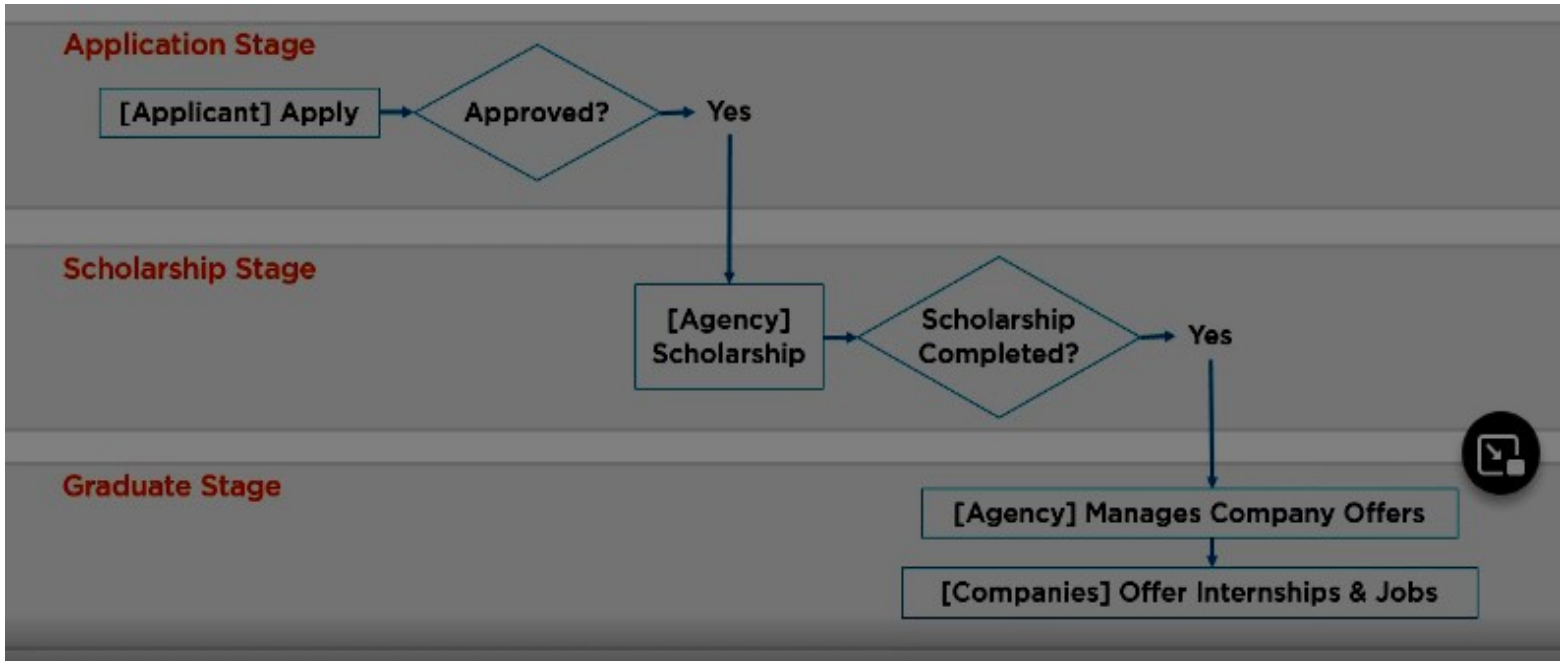
The case study is an adaptation of a real world scenario of a government agency. This agency funds and manages the scholarship of post-graduate students who are study overseas to earn the master , PhD. After completion scholarship the agency also offers post-graduation services which help graduates find internships and jobs. The type of students , thiers allocated funds and other complex business concerns are not relevant to this discussion.



Case Study

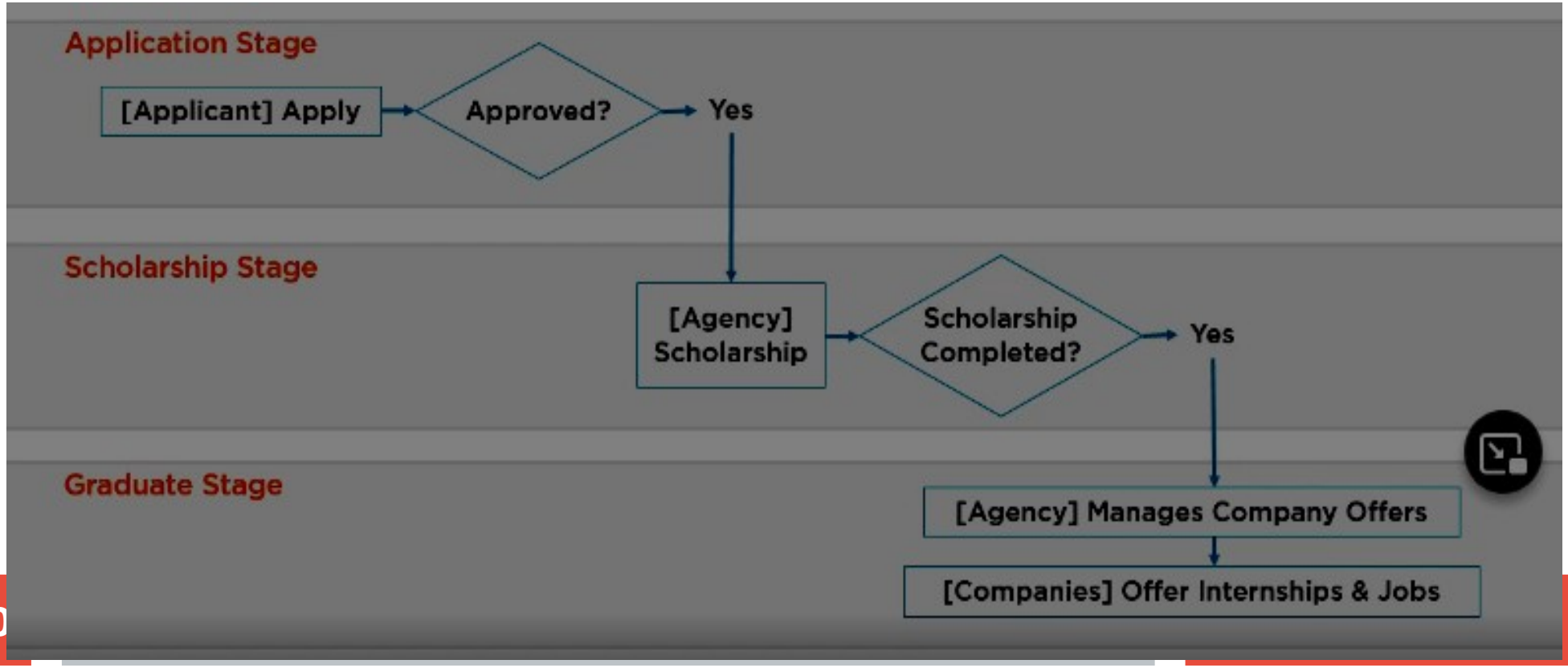
The high-level scenario goes as follows:

Application stage: where the applicant submit thier requests and get either approuved or rejected ; scholarship stage: This is the core of thier buisness where approuved applicants get thier entire scholarship managed in terms of finance and administration; graduate stage: get services that help them get internship son jobs.

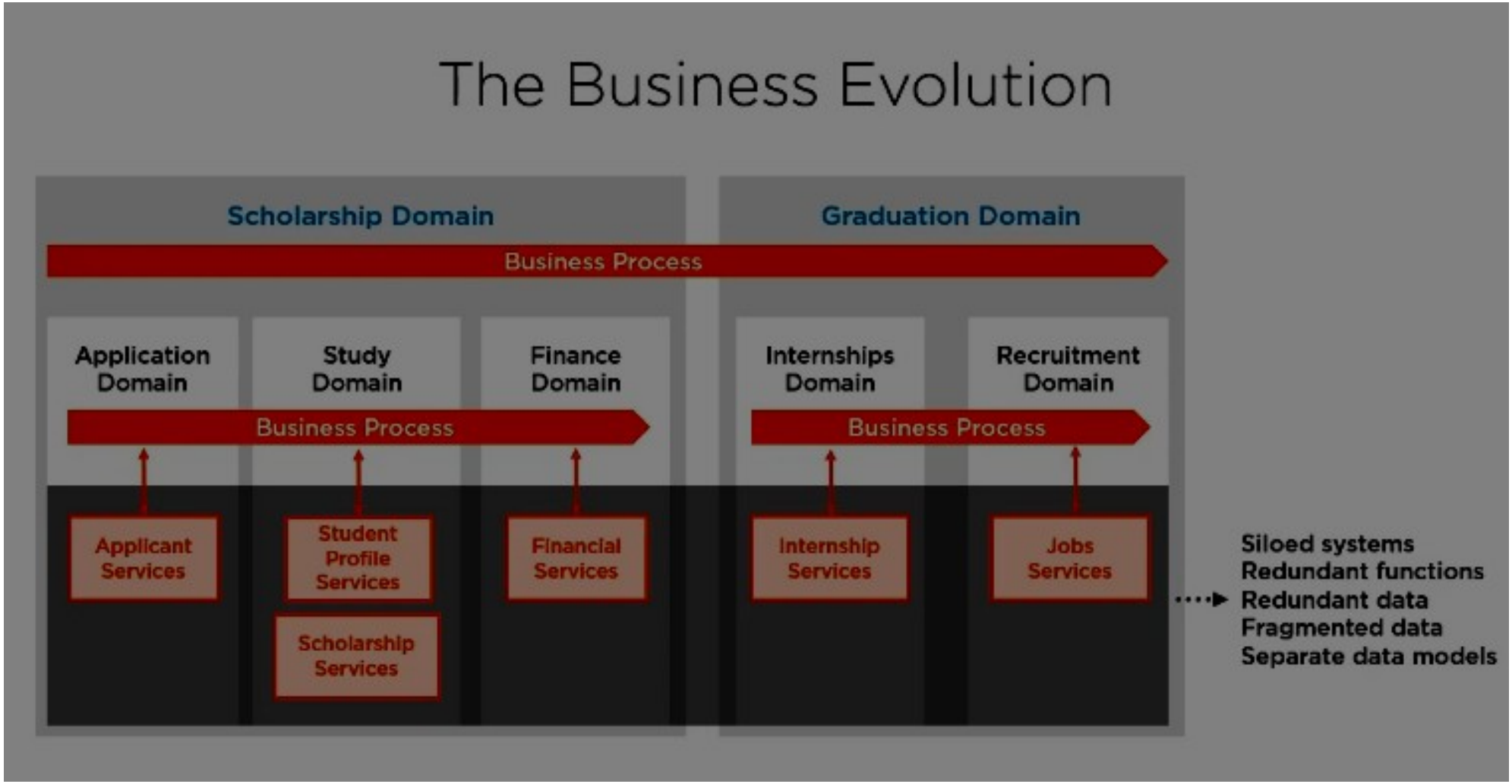


Case Study

In the high-level an applicant to a scholarship, the govmment manage everything related to this scholarship, such as funding, changing the major, grade striking, financial requests, pausing and resuming the scholarship



Case Study : Problems



Case Study : Problems

❖ let's examine the problems that the organisation is facing and t
the course we're going to see how SOA help solving these problem

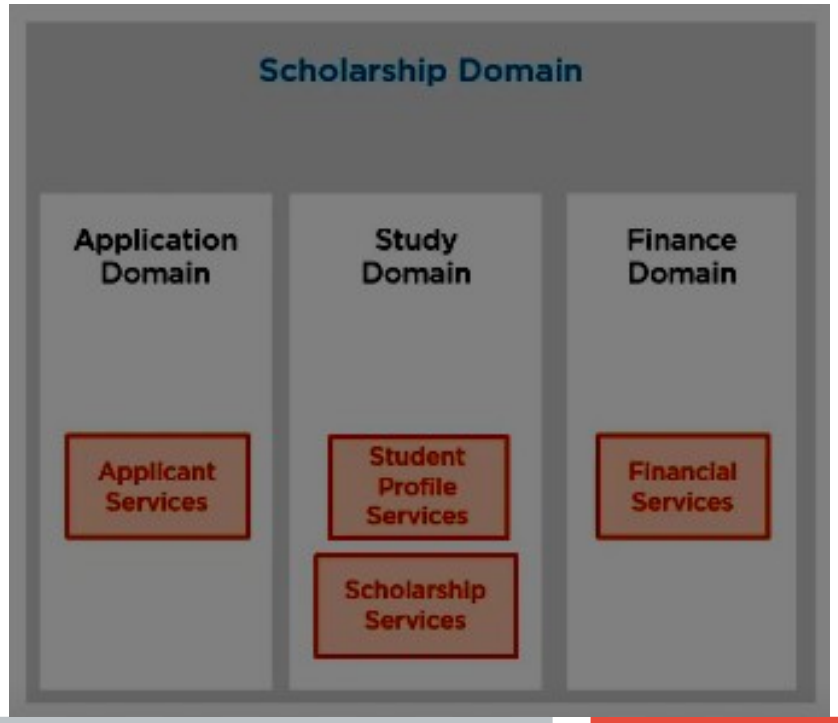
❖ So when the government agency first started. the demands wa
the public services into as soon as possible at all costs.

❖ So the team started by creating a couple of systems that provi
❖ The scope started to get bigger and it was requested to prese
 schoolarship related services such as majour changind and stopp
a scholarship.

Then later, the financial services were requested as
❖ the agency decided to streamline the process by which
❖ they compensate students.

Case Study : Problems

At this stage the agency decided that as the range of busness is
then it must partition the buisness domaines into a set of subdo
ease managment and planning
So a scholarship business doamin was established and include
study and finance each become the owner of the set informatio
that serve this domain.



Case Study : Problems

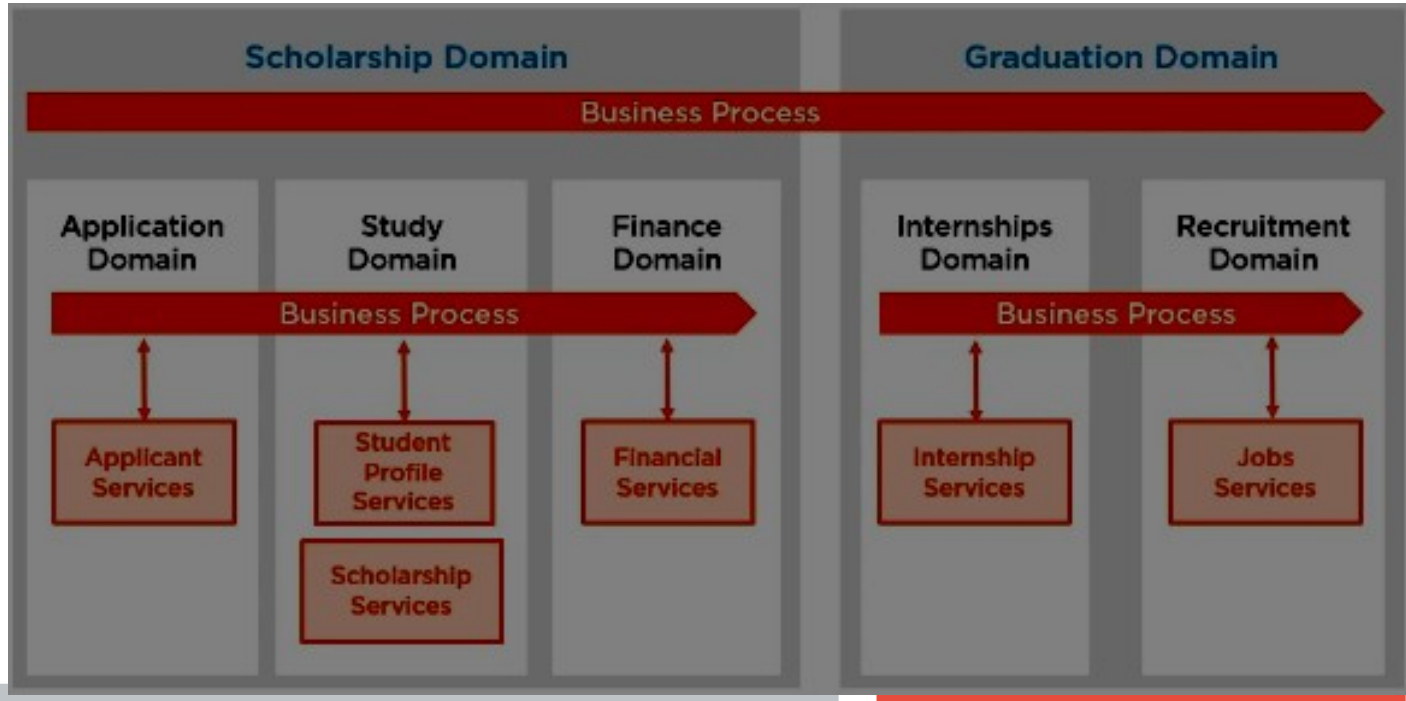
But the business model of the agency kept growing and now a major decision was taken to provide most schoolarship services , first helping graduales land interships, then expending the scope to help them land jo

Case Study : Problems

During expansion came critical moment. Many services exposed t
now required creation of cross-unit or cross-domain process
(collaborate to present public services).

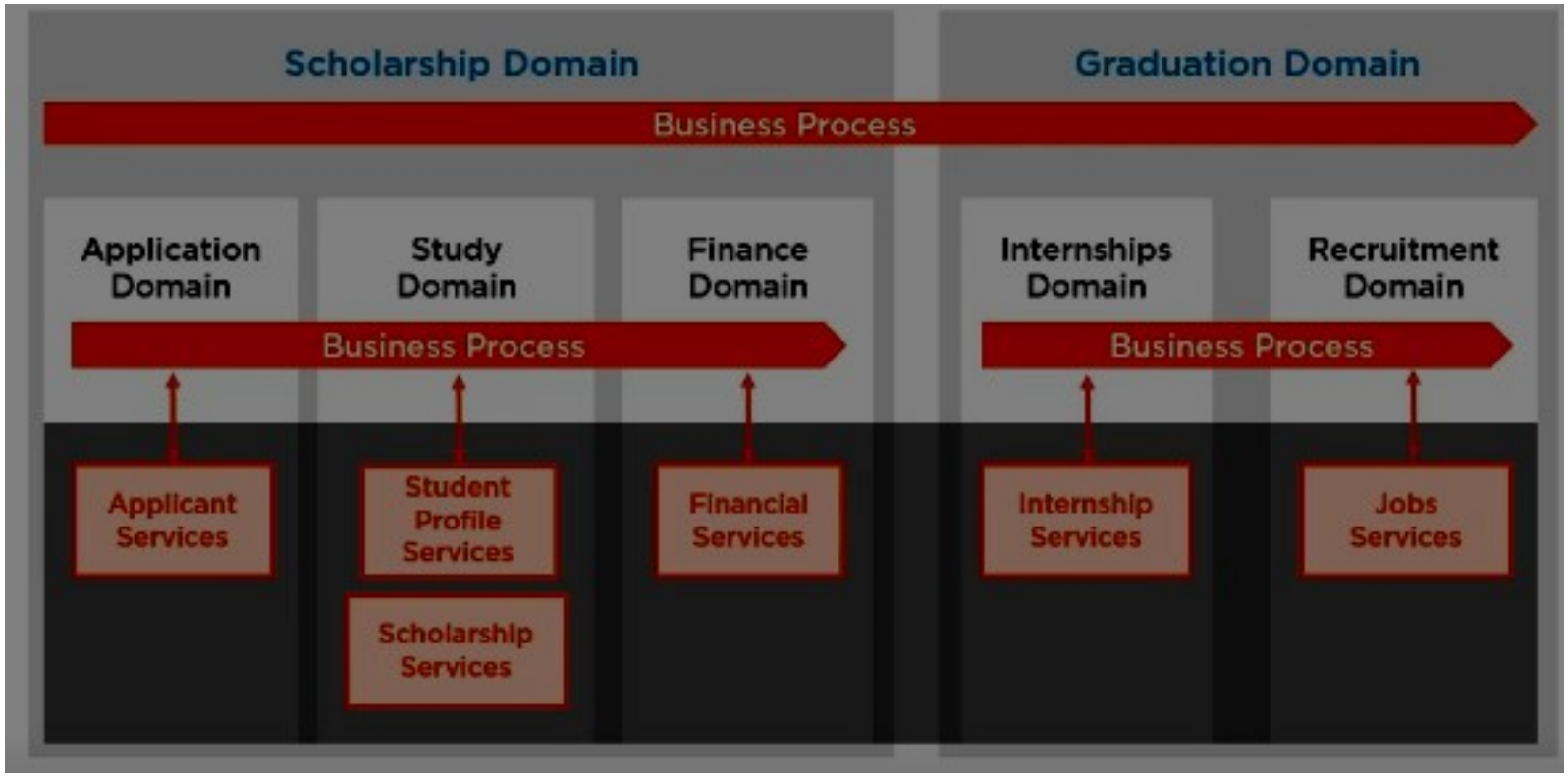
Theses buisness processus use fonctions and data presented by
the informational system in order to perform thier services

-> **this lies the problem.**



Case Study : Problems

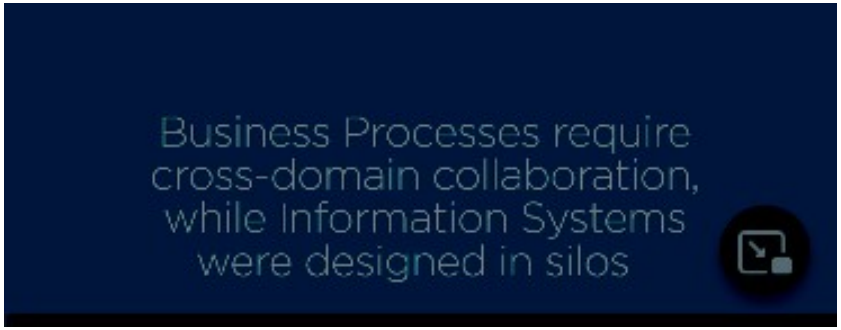
System is initially built in silos without a single holistic view of the entire architecture



This means that each system was created without taking conside the overall growth and architecture

Case Study : Problems

a set of siled systems with redundant functionality, redundant data between data stores, fragmented data between different stores. So what the organization has is an interesting p



Business processes that were from the start designed to croos unit boundaries and required different units to collaborate to achieve requires outcomes. Yet the supporting systems the o that provide functiond and data to the processes are built in silo manner with no regard to cross-unit collaboration.

Case Study : Problems

This makes the agency slow to react to requirements, new servic
slow to take advantage of new technical opportunities.
That could better enable the businesses.

It costs are high since it spends valuable
time and money. working arround the limitations
caused by the IT state.
Both developpment and operations cos
are high to keep the buisness running .

Resulting Issues	Low flexibility/agility High IT costs Lack of customer-centricity

Customers are often required to enter the same information mult
and over the time, this information become inconsistent.

Case Study : Goals

Business Goals

Improve business processes flexibility

- Critical step towards organization agility

Decrease IT costs

Establish customer-centricity

- Single customer experience across units

Case Study : Goals

One More Goal:
New Emerging
Business Need

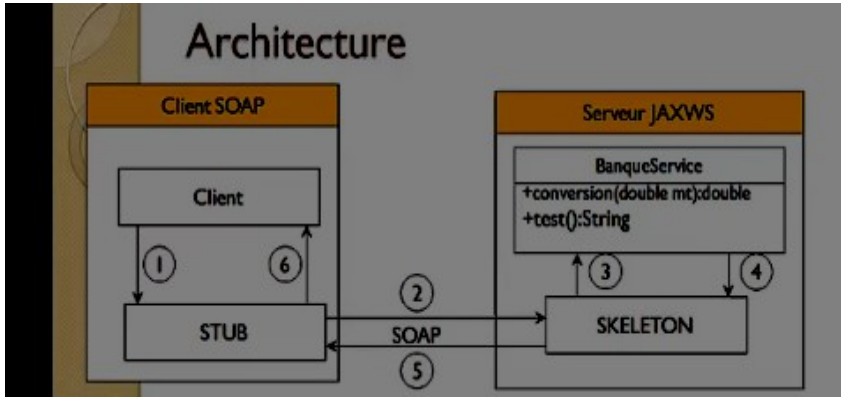
Companies should be able to

- Post their own information
- Themselves ask for certain skills

More efficient than current communication methods

In business terms: extending the ecosystem of the agency

TP : Web service



1- The client asks the stub to call the conversion m

2- the stub connects to the skeleton and sends it a request

3- The skeleton uses the method of the web servic

4- the web service returns the result to the skeleto

5- The skeleton sends the result in a SOAP respons to the STUB

6- the stub provides the result to the client

TP : Web service

```
package ws;

import java.io.Serializable;
/**
 *
 * @author FOTSO
 */
public class Account implements Serializable{
    int id;
    float solde;

    public Account() {
        super();
    }
    public Account(int id, float solde) {
        this.id = id;
        this.solde = solde;
    }
```

wsgen -s . -wsdl -cp ../build/classes jaxws.ws.BankService

TP : Web service

```
@WebService(serviceName = "BanqueWs")
public class BankService {
    @WebMethod(operationName = "concersionEuroGh")
    public double conversion(@WebParam(name = "amount") double mt) {
        return 10 * mt;
    }
    @WebMethod
    public String test() {
        return "test";
    }
    @WebMethod
    public Account getAcccount(@WebParam(name = "code") int id) { //if not public you can not see it
        return new Account(id, (float) Math.random() * 152);
    }
    public List<Account> getAcounptes() {
        List<Account> accounts = new ArrayList<>();
        for (int i = 0; i < 6; i++)  accounts.add(new Account(i, (float) Math.random() * 152));
        return accounts;
    }
}
```

TP : Web service testing (uing oxygen, SoapUI,…)

Analyseur WSDL SOAP

WSDL

Services: BanqueWs

Ports: BankServicePort

Opérations: test

Actions

URL: http://localhost:8088/

Action SOAP:

Version: 1.1 1.2

Requête | Fichiers joints

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns0="http://ws/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns0:test/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Envoyer

Paramètres de la requête: Ouvrir Enregistrer Régénérer

☐ Ouvrir la réponse dans l'éditeur

Réponse

```
<!-- Auto generated server sample response. -->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns0="http://ws/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns0:testResponse>
      <return>STRING</return>
    </ns0:testResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

TP : Web service testing (getAccount)

WSDL

Services: BanqueWs

Ports: BankServicePort

Opérations: getAcccount

Actions

URL: http://localhost:8088/

Action SOAP:

Version: 1.1 1.2

Requête | Fichiers joints

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns0="http://ws/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns0:getAcccount>
      <code>10</code>
    </ns0:getAcccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Envoyer

Paramètres de la requête: Ouvrir Enregistrer Régénérer

Ouvrir la réponse dans l'éditeur

Send requ est

Réponse

```
<!-- Auto generated server sample response. -->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns0="http://ws/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns0:getAcccountResponse>
      <return>
        <id>INT</id>
        <solde>FLOAT</solde>
      </return>
    </ns0:getAcccountResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

TP : Web service testing (getAccount)

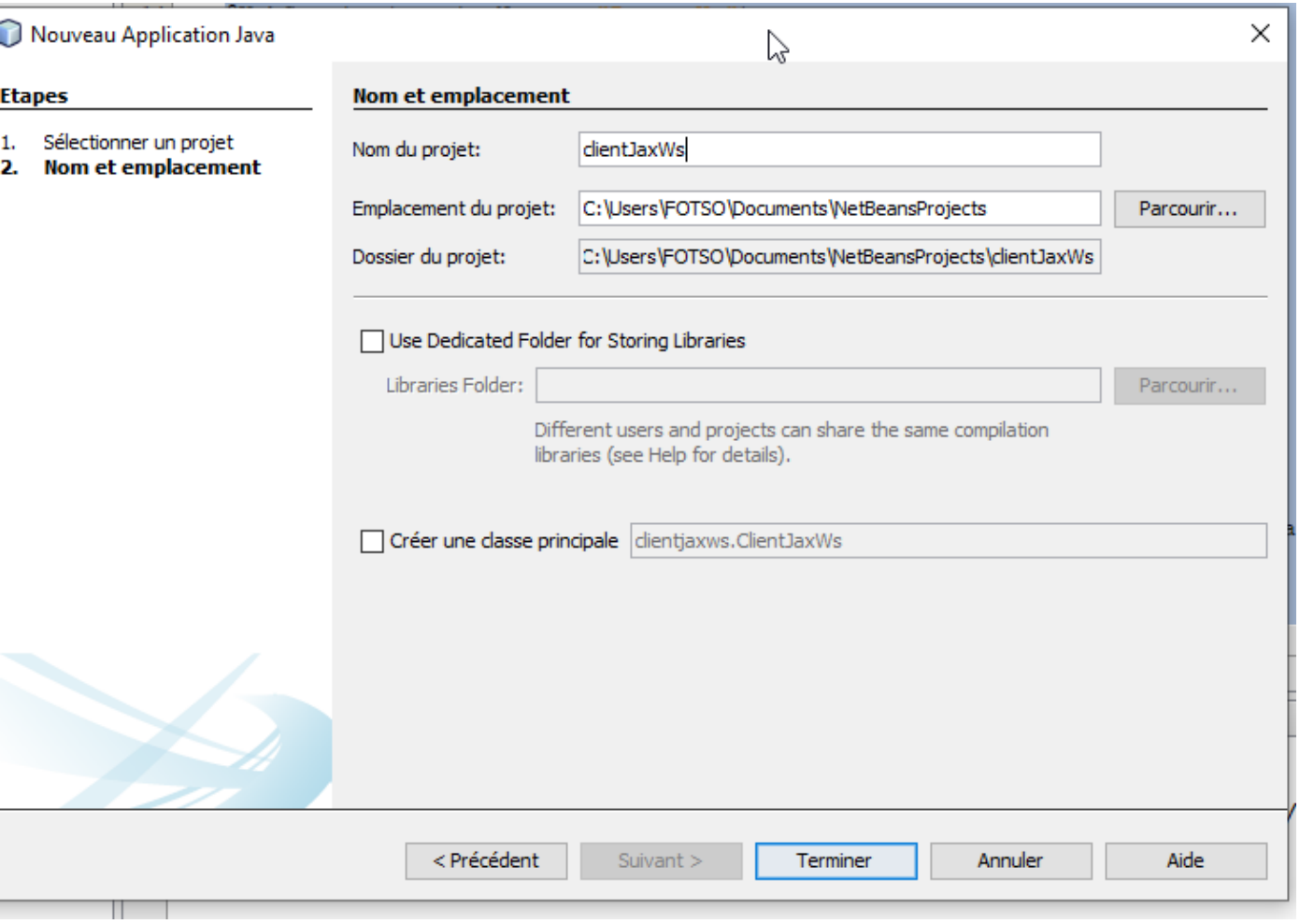
```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns0="http://ws/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns0:getAcounptes/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request: get all accounts

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getAcounptesResponse xmlns:ns2="http://ws/">
         <return>
            <id>0</id>
            <solde>99.93814</solde>
         </return>
         <return>
            <id>1</id>
            <solde>74.77114</solde>
         </return>
         <return>
            <id>2</id>
            <solde>109.61301</solde>
         </return>
         <return>
            <id>3</id>
            <solde>15.757117</solde>
         </return>
         <return>
            <id>4</id>
            <solde>0.121832766</solde>
         </return>
         <return>
            <id>5</id>
            <solde>43.642258</solde>
         </return>
      </ns2:getAcounptesResponse>
   </S:Body>
</S:Envelope>
```

Respone: get accounts

TP : Web service

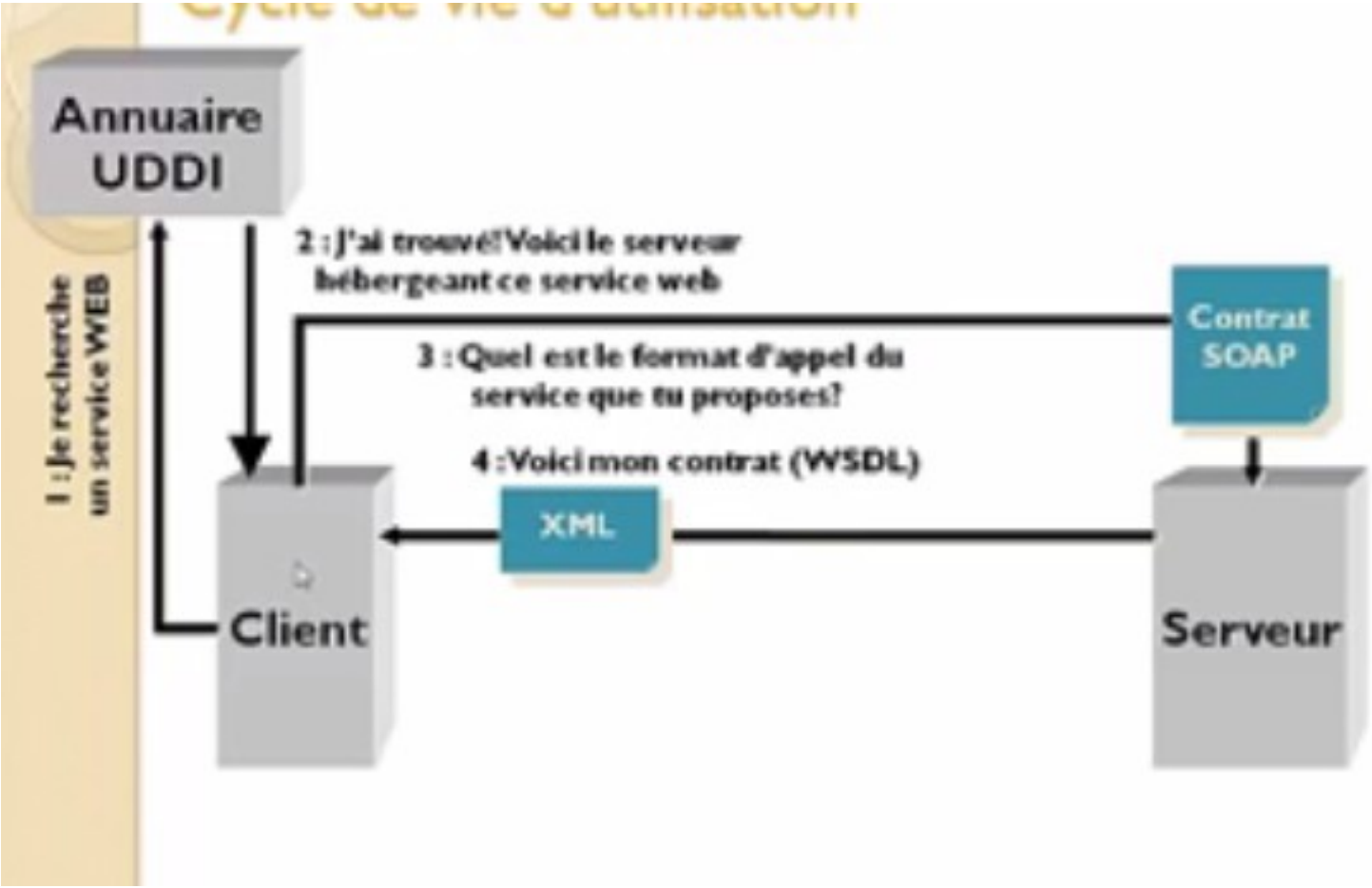


Mainatenant suis une autre entreprise connu lien web Service dois consomme

create java project:
How to create his stub?

TP : Web service

Save wsdl file src java client project



TP : Java customer

```
public class Customer {

    public static void main(String[] args) {
        BankService stub = new BanqueWs().getBankServicePort();
        System.out.println("convert :" + stub.concersionEuroGh(10));
        System.out.println(stub.getAcounptes());
        for (Account acounpte : stub.getAcounptes()) {
            System.out.println("id: "+acounpte.getId()+ " solde:
"+acounpte.getSolde());
            System.out.println("----------------------------------");
        }

    }
}
```

TP : PHP customer

Create and consume SOAP web service using PHP

Prerequisites

Apache HTTP Server 2.4, PHP 7 or more, NUSOAP Library, CURL

Configure PHP SOAP

Consume SOAP Service

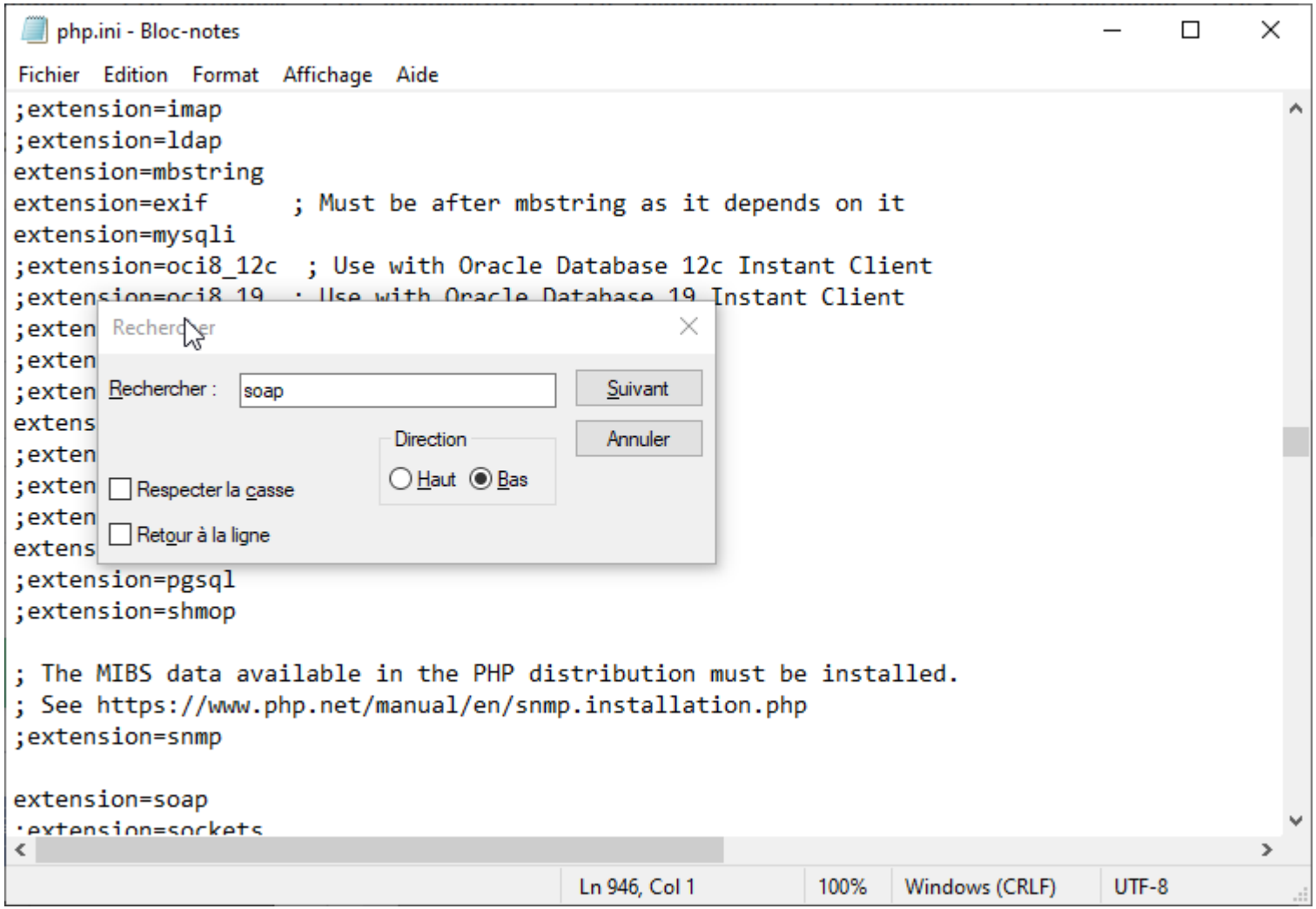
I will show you here 3 different ways of calling or consuming SOAP web service. The first method is using **SoapClient**, the second method is using **NUSOAP** library and the third method is using **CURL**.

Using SoapClient

Using SoapClient you do not need to use any third party library because SoapClient is already available in PHP engine.

The following PHP code using SoapClient calls the converter method, get account ,....

TP : PHP customer (enable soap extension)



TP : PHP customer (code)

```php
<?php
$PORT = 8585;
$amount = 0;
$converted = null;
if (isset($_POST['amount']))
{
    $converted = null;
    $amount =
$_POST['amount'];
    $client = new
SoapClient("http://localhost
:$PORT/BankWs?wsdl");
    $param = new stdClass();
    $param->amount =
$amount;
    $result = $client-
>__soapCall("conversionEURTo
XAF", [$param]);
    $converted = $result-
>return;
} ?>
```

```html
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
       <title>Document</title>
</head>
<body>
<form method="post">
    <label for="amount">Amount in
EUR</label>
    <input type="number" name="amount"
min="0" value="<?=
isset($_POST['amount']) ?
$_POST['amount'] : '' ?>"
            placeholder="Enter the
amount"/>
    <input type="submit"
value="Convert"> <b><?= $converted !==
null ? ' = ' . $converted . ' XAF' :
'' ?></b>
</form>
</body>
```

TP : PHP customer

Create and consume SOAP web service using PHP

Prerequisites

Apache HTTP Server 2.4, PHP 7 or more, NUSOAP Library, CURL

Configure PHP SOAP

Consume SOAP Service

I will show you here 3 different ways of calling or consuming SOAP web service. The first method is using **SoapClient**, the second method is using **NUSOAP** library and the third method is using **CURL**.

Using SoapClient

Using SoapClient you do not need to use any third party library because SoapClient is already available in PHP engine.

The following PHP code using SoapClient calls the converter method, get account ,....

Generation stub client app

MINGW64:/c/Users/FOTSO/Documents/NetBeansProjects/clientJaxWs/src

```
FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$ ls -l
total 4
-rw-r--r-- 1 FOTSO 197121 3817 Nov  1 14:49 BanqueWs.xml

FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$ wsimport BanqueWs.xml
analyse du WSDL...



Génération du code...


Compilation du code...


FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$
```

Documents › NetBeansProjects › clientJaxWs › src › ws

Nom	Modifié le	Type	Taille
Account.class	02/11/2022 02:25	Fichier CLASS	1 Ko
BankService.class	02/11/2022 02:25	Fichier CLASS	2 Ko
BanqueWs.class	02/11/2022 02:25	Fichier CLASS	3 Ko
ConcersionEuroGh.class	02/11/2022 02:25	Fichier CLASS	1 Ko
ConcersionEuroGhResponse.class	02/11/2022 02:25	Fichier CLASS	1 Ko
GetAcccount.class	02/11/2022 02:25	Fichier CLASS	1 Ko
GetAcccountResponse.class	02/11/2022 02:25	Fichier CLASS	1 Ko
GetAcounptes.class	02/11/2022 02:25	Fichier CLASS	1 Ko
GetAcounptesResponse.class	02/11/2022 02:25	Fichier CLASS	1 Ko
ObjectFactory.class	02/11/2022 02:25	Fichier CLASS	4 Ko
package-info.class	02/11/2022 02:25	Fichier CLASS	1 Ko
Test.class	02/11/2022 02:25	Fichier CLASS	1 Ko
TestResponse.class	02/11/2022 02:25	Fichier CLASS	1 Ko

Generation stub client app

MINGW64:/c/Users/FOTSO/Documents/NetBeansProjects/clientJaxWs/src

```
FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$ pip install requests
Collecting requests
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     ---------------------------------------- 62.8/62.8 kB 177.1 kB/s eta 0:00:00
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
     ---------------------------------------- 161.1/161.1 kB 419.1 kB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
     ---------------------------------------- 140.4/140.4 kB 438.2 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.5/61.5 kB 298.2 kB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, reque
sts
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 reque
sts-2.28.1 urllib3-1.26.12
WARNING: There was an error checking the latest version of pip.

FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$
```

MINGW64:/c/Users/FOTSO/Documents/NetBeansProjects/clientJaxWs/src

```
Installing collected packages: urllib3, idna, charset-normalizer, certifi, reque
sts
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 reque
sts-2.28.1 urllib3-1.26.12
WARNING: There was an error checking the latest version of pip.

FOTSO@DESKTOP-LATI2F5 MINGW64 ~/Documents/NetBeansProjects/clientJaxWs/src
$ pip install zeep
Collecting zeep
  Downloading zeep-4.1.0-py2.py3-none-any.whl (100 kB)
     ---------------------------------------- 100.6/100.6 kB 152.1 kB/s eta 0:00:00
Collecting cached-property>=1.3.0
  Downloading cached_property-1.5.2-py2.py3-none-any.whl (7.6 kB)
Collecting isodate>=0.5.4
  Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
     ---------------------------------------- 41.7/41.7 kB 334.9 kB/s eta 0:00:00
Collecting requests-file>=1.5.1
  Downloading requests_file-1.5.1-py2.py3-none-any.whl (3.7 kB)
Collecting lxml>=4.6.0
  Downloading lxml-4.9.1-cp310-cp310-win_amd64.whl (3.6 MB)
     ---------------------------------------- 3.6/3.6 MB 713.8 kB/s eta 0:00:00
Collecting attrs>=17.2.0
  Downloading attrs-22.1.0-py2.py3-none-any.whl (58 kB)
     ---------------------------------------- 58.8/58.8 kB 516.5 kB/s eta 0:00:00
Collecting pytz
  Downloading pytz-2022.6-py2.py3-none-any.whl (498 kB)
     ---------------------------------------- 498.1/498.1 kB 799.7 kB/s eta 0:00:00
Collecting platformdirs>=1.4.0
  Downloading platformdirs-2.5.2-py3-none-any.whl (14 kB)
Requirement already satisfied: requests>=2.7.0 in c:\users\fotso\appdata\local\p
rograms\python\python310\lib\site-packages (from zeep) (2.28.1)
Collecting requests-toolbelt>=0.7.1
  Downloading requests_toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
     ---------------------------------------- 54.5/54.5 kB 282.1 kB/s eta 0:00:00
Requirement already satisfied: six in c:\users\fotso\appdata\local\programs\pyth
on\python310\lib\site-packages (from isodate>=0.5.4->zeep) (1.16.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\fotso\appdata\local\prog
rams\python\python310\lib\site-packages (from requests>=2.7.0->zeep) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\fotso\appdata\loca
l\programs\python\python310\lib\site-packages (from requests>=2.7.0->zeep) (2022
.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\fotso\appdata\l
```

Case of python client: use zeep module

pip3 install zeep

```
from zeep import Client

client = Client(wsdl='http://localhost:8088/BankWs?wsdl')


print('result of conversion: ',
client.service.concersionEuroGh(20))
print('accounts : ', client.service.getAcounptes())
```

Application Authentication with JAX-WS

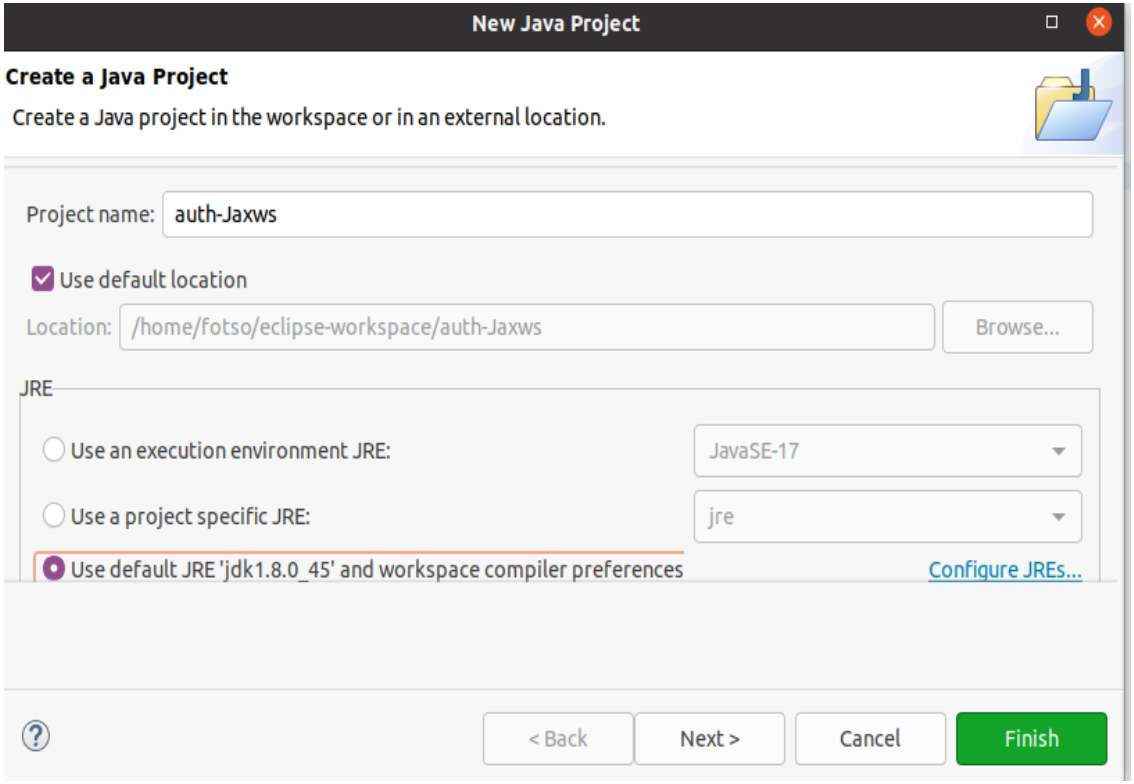
In this section, we show you how to implement the above "application level authentication in JAX-WS". Client provides " username" and "password", attached in SOAP request header and send to server. Server parse the SOAP document and retrieve the provided username and 'password from request header.

Application Authentication with JAX-WS

1. WebService Server :

Create a simple JAX-WS hello world example to handle the authentication in application level.

File : **HelloWorld.java**



Application Authentication with JAX-WS

HelloWorld.java

.java

This interface will contain the declarations of all the methods for the Web Service.
Then we have to create a class that actually implements the above interface, which will be your Endpoint implementation.

Application Authentication with JAX-WS

HelloWorld.java

```java
package auth;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld {

@WebMethod String getHelloWorldAsString();

}
```

HelloWorldImpl.java

```java
package auth;

import javax.jws.WebService;

//Service Implementation Bean
@WebService(endpointInterface = "auth.HelloWorld")
public class HelloWorldImpl implements HelloWorld {

@Override
public String getHelloWorldAsString() {
// TODO Auto-generated method stub
return null;
}

}
```

Application Authentication with JAX-WS

The server would have to read the HTTP request header information which the client put, and validate its identity.

Service ***Endpoint*** Implementation obtains a ***MessageContext*** object through a **WebServiceContext** for accessing the objects of the message.

The ***WebServiceContext*** interface enables a web service endpoint implementation class to access message contexts and security information of the requester.

The service runtime will inject the WebServiceContext on any field marked with @Resource annotation. We will call the WebServiceContext‘s ***getMessageContext()*** method to get MessageContext instance.

Application Authentication with JAX-WS

HelloWorldImpl.java

```
package auth;

import java.util.List;
import java.util.Map;

import javax.annotation.Resource;
import javax.jws.WebService;
import javax.xml.ws.WebServiceContext;
import javax.xml.ws.handler.MessageContext;

//Service Implementation Bean
@WebService(endpointInterface = "auth.HelloWorld")
public class HelloWorldImpl implements HelloWorld {

	@Resource
	WebServiceContext wsctx;

	@Override
	public String getHelloWorldAsString() {
		MessageContext mctx = wsctx.getMessageContext();

		// get detail from request headers
		Map http_headers = (Map)
mctx.get(MessageContext.HTTP_REQUEST_HEADERS);
		List userList = (List)
http_headers.get("Username");
		List passList = (List)
http_headers.get("Password");
		String username = "";
		String password = "";

		if (userList != null) {
			// get username
			username = userList.get(0).toString();
		}

		if (passList != null) {
			// get password
			password = passList.get(0).toString();
		}

		// Should validate username and password with
database
		if (username.equals("root") &&
password.equals("admin")) {
			return "Hello World JAX-WS - Valid User!";
		} else {
			return "Unknown User!";
		}
	}

}
```

Application Authentication with JAX-WS

2. EndPoint Publisher

Create an endpoint publisher to deploy above web service at this URL : “http://localhost:8088/ws/hello”

File : ***HelloWorldPublisher.java***

```
package auth;

import javax.xml.ws.Endpoint;

public class HelloWorldPublisher {

      public static void main(String[] args) {
            String url = "http://localhost:8088/ws/hello";
            Endpoint.publish(url, new HelloWorldImpl());

            System.out.println(url);

      }

}
```

A **QName**, or qualified name, is the full name of an element, attribute. A QName concisely associates the URI of an XML namespace with the local name of an element, attribute, or identifier in that namespace.

Application Authentication with JAX-WS

3. WebService Client

The client will consume the web service so the client has to make a new HTTP Request Header containing its username and password.

To access and manipulate the request contexts of the message the client has to get a BindingProvider from the service port using ***getRequestContext()*** method.

The ***BindingProvider*** interface enables the client to access and manipulate the associated context objects for request and response messages. The Request Context is retrieved as a Map object.

MessageContext.ENDPOINT_ADDRESS_PROPERTY is used to nominate the target service endpoint address. Now we need to add two properties username and password to the Map object.
Then we put this Map object in ***MessageContext.HTTP_REQUEST_HEADERS***.

Application Authentication with JAX-WS

3. WebService Client

Create a web service client to send “username” and “password” for authentication.

File : HelloWorldClient.java

```
import javax.xml.ws.Service;
```

Creates a Service instance. The specified WSDL document location and service qualified name MUST uniquely identify a wsdl:service element.

public interface BindingProvider The BindingProvider interface provides access to the protocol binding and associated context objects for request and response message processing.
Since: JAX-WS 2.0

ENDPOINT_ADDRESS_PROPERTY Standard property: Target service endpoint address.

static String PASSWORD_PROPERTY Standard property: Password for authentication.

static String SESSION_MAINTAIN_PROPERTY Standard property: This boolean property is used by a service client to indicate whether or not it wants to participate in a session with a service endpoint.

static String SOAPACTION_URI_PROPERTY Standard property for SOAPAction.

static String SOAPACTION_USE_PROPERTY Standard property for SOAPAction.

static String USERNAME_PROPERTY Standard property: User name for authentication.

Output

Hello World JAX-WS - Valid User!

Application Authentication with JAX-WS

3. WebService Client

Create a web service client to send “username” and “password” for authentication.

File : HelloWorldClient.java

```
public class HelloWorldClient {

      private static final String WS_URL = "http://localhost:8088/ws/hello?wsdl";

      public static void main(String[] args) throws Exception {

            URL url = new URL(WS_URL);
            QName qname = new QName("http://ws.tafoca.com/", "HelloWorldImplService");

            Service service = Service.create(url, qname);
            HelloWorld hello = service.getPort(HelloWorld.class);

            /****************** UserName & Password ******************************/
            Map<String, Object> req_ctx = ((BindingProvider)
hello).getRequestContext();
            req_ctx.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, WS_URL);
            Map<String, List<String>> headers = new HashMap<String, List<String>>();
            headers.put("Username", Collections.singletonList("root"));
            headers.put("Password", Collections.singletonList("admin"));
            req_ctx.put(MessageContext.HTTP_REQUEST_HEADERS, headers);
            /*********************************************************************/

            System.out.println(hello.getHelloWorldAsString());

      }
}
```

Application Authentication with JAX-WS

4. Tracing SOAP Traffic

From top to bottom, showing how SOAP envelope flows between client and server.

1. Client send request, the username "***root***" and password "***admin***" are included in the SOAP envelope.

```
POST /ws/hello?wsdl HTTP/1.1
Password: admin
Username: root
SOAPAction: ""
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-Type: text/xml; charset=utf-8
User-Agent: Java/1.6.0_13
Host: localhost:8088
Connection: keep-alive
Content-Length: 178

<?xml version="1.0" ?>
    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
        <S:Body>
            <ns2:getHelloWorldAsString xmlns:ns2="http://app/"/>
        </S:Body>
    </S:Envelope>
```

Application Authentication with JAX-WS

4. Tracing SOAP Traffic

From top to bottom, showing how SOAP envelope flows between client and server.

1. Client send request, the username "***root***" and password "***admin***" are included in the SOAP envelope.

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: text/xml; charset=utf-8

<?xml version="1.0" ?>
    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
        <S:Body>
            <ns2:getHelloWorldAsStringResponse
xmlns:ns2="http://ws.mkyong.com/">
                <return>Hello World JAX-WS - Valid User!</return>
            </ns2:getHelloWorldAsStringResponse>
        </S:Body>
    </S:Envelope>
```

https://github.com/tafoca/Tp-Tutorial-tafo-group/tree/main/auth-Jaxws

JAX-WS SOAP Web Service
For CRUD Operations Using Eclipse, SQL

Case study add Database connection service (DAO)

How to generate a web service with top-down approach using JAX-WS by creating Java Classes using wsimport tool and then incorporate them in your java project.

- How to implement web service endpoint interface using an implementation class.
- How to call DAO layer methods from your SOAP Web service implementation class.
- How to connect to SQL database using JDBC and perform CRUD operation.
- How to test and consume a SOAP Web service.

Case study add Database connection service (DAO)

We will implement a User Management Web Service with following key operations:

- *Operation to add a new user to database table.*
- *Operation to update an existing user in database table.*
- *Operation to delete an existing user from database table.*
- *Operation to Fetch any user from database table.*
- *Operation to get all users from database table.*

Case study add Database connection service (DAO)

CREATE TABLE TBL_USERS (user_id INT NOT NULL,user_name VARCHAR(255) NOT NULL,user_category VARCHAR(255),user_active_status BOOLEAN,user_level Float,PRIMARY KEY (user_id));

Step 2: Add Required Libraries in Tomcat Server

Add jar library in lib directory of tomcat

jar_files-tp4/javax.annotation-api-1.2-b03.jar
jar_files-tp4/javax.xml.soap-api-1.3.5.jar
jar_files-tp4/jaxb-api-2.2.9.jar
jar_files-tp4/jaxws-api-2.2.10.jar
jar_files-tp4/jsr181-api-1.0-MR1.jar

Case study add Database connection service (DAO)

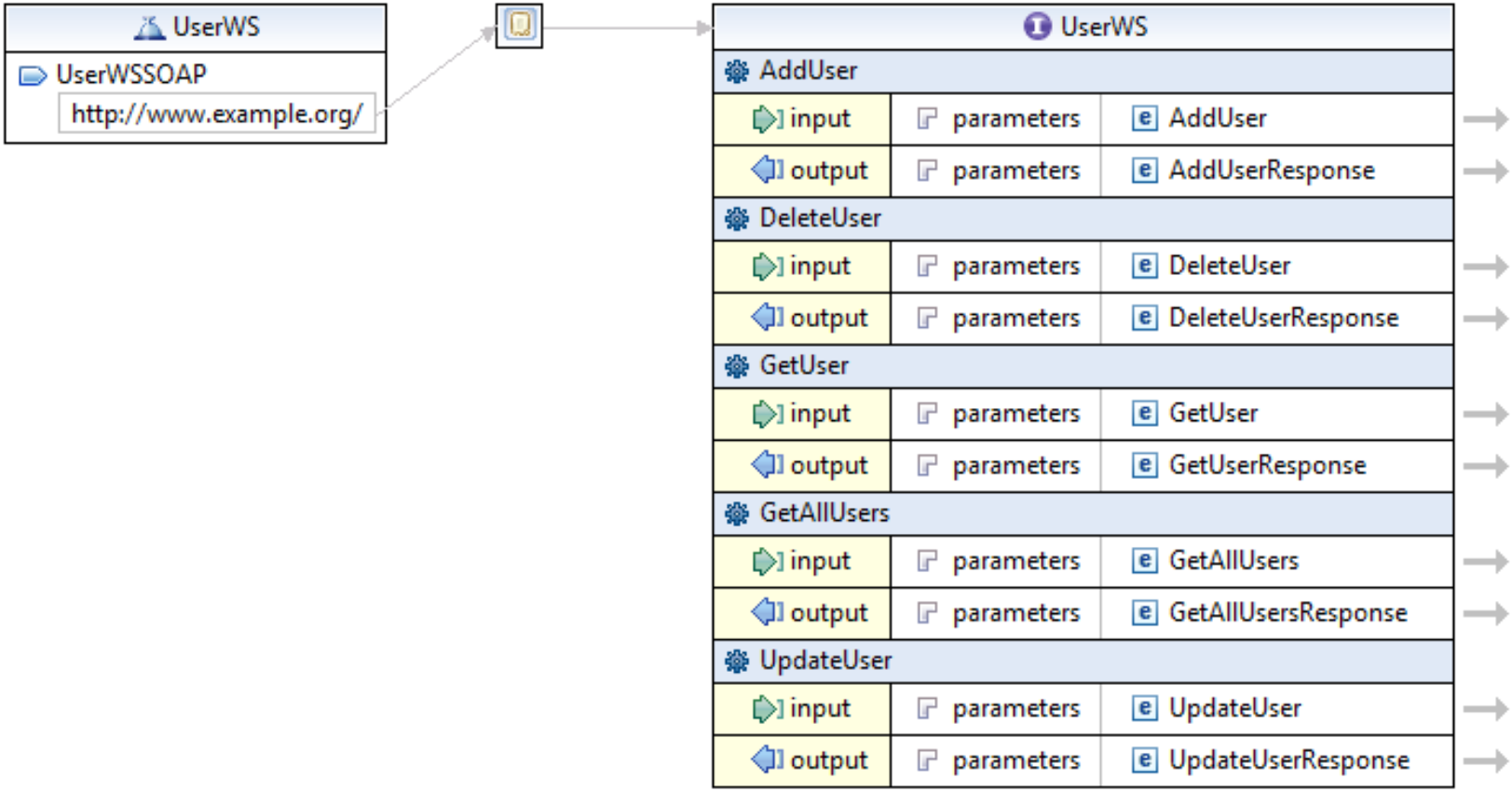
Step 3: Create Dynamic Web Project for JAX-WS SOAP Web Service
Name : JAXWSSoapWebService

Step 4: Top-Down SOAP Web Service: Create XML Schema and WSDL File

For a Top-Down (Contract-First) Web service, we need to first define the contract and then create Java Classes based on that. For this purpose, we will create XSD file (XML Schema) for our User Management use case and then use elements from that XSD in the WSDL file for input and output messages for different operations.
You can create XSD by choosing **File–> New–>XML Schema File** and name it as UserSchema.xsd
Eclipse provide a good tool for creating XSD and defining elements and types. Using Eclipse, I created the schema with Users element containing User where User will have following primitive elements:

- UserID (Integer)
- UserName (String)
- UserCategory (String)
- UserLevel (Double)
- UserActiveStatus (Boolean)

Case study add Database connection service (DAO)



Case study add Database connection service (DAO)

Step 5: Implement DAO Layer with JDBC to for SOAP Web Service to Save Data in Database

For our tutorial, we will use JDBC with MySQL/Postgresl driver to connect with database to perform

all CRUD operations in our dtabase tableÂ TBL_USERSÂ which was created in Step 2.

Our DAO class (UserDAO.java) implements all CRUD operations for our web service to save, fetch, update and delete records from users table.

Case study add Database connection service (DAO)

Step 6: Develop SOAP Web Service Operations by Implementing Service Interface

In this step we implement all service operations by creating an implementation class UserWSImpl.java in which we implement methods of our SEI (Service Endpoint Interface)

The implementation class has following annotations:

- @WebService annotation to mark it as a web service implementation class.
- @WebMethod annotation for our methods which are implementing web service operatons.
- @WebResult annotation to customize our Response element (e.g. to give response element a meaningful name)
- @WebParam annotation to get web service parameters received from the client.

In our Service implementation class, we are calling our DAO class method so that all the logic for database handling is segregated from our service layer.

Case study add Database connection service (DAO)

Step 6: Develop SOAP Web Service Operations by Implementing Service Interface

```
import java.util.List;

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

import entity.Users;

@WebService
public interface UserWS {
@WebMethod public String addUser(@WebParam(name="User") Users addRequest);
@WebMethod public String deleteUser(@WebParam(name="UserID") int userID);
@WebMethod public String updateUser(Users updateUserRequest);
@WebMethod public Users getUser(@WebParam(name="UserName") String userName);
@WebMethod public List<Users> getAllUsers();
}
```

Case study add Database connection service (DAO)

Step 6: Develop SOAP Web Service Operations by Implementing Service Interface

```
package service;
import java.util.List;
import javax.jws.WebService;
import dao.UserDAO;
import entity.Users;
@WebService
public class UserWSImpl implements UserWS {
UserDAO userdao;
@Override
public String addUser(Users addRequest) {
return null;
}

@Override
public String deleteUser(int userID) {
return null;
}

@Override
public String updateUser(Users updateUserRequest) {
// TODO Auto-generated method stub
return null;
}

@Override
public Users getUser(String userName) {
return null;
}
@Override
public List<Users> getAllUsers() {
return null;
}

}
```

Case study add Database connection service (DAO)

Step 6: Develop SOAP Web Service Operations by Implementing Service Interface

```
@Override
@WebMethod
@WebResult(name="ResponseMessage")
public String addUser(@WebParam(name="User")  Users
addRequest) {
return  new UserDAO().addUser(addRequest);
}

@Override
@WebMethod
@WebResult(name="ResponseMessage")
public String deleteUser(@WebParam(name="UserID") int
userID) {
return new UserDAO().deleteUser(userID);
}

@Override
@WebMethod
@WebResult(name="ResponseMessage")
public String updateUser(Users updateUserRequest) {
return new UserDAO().updateUser(updateUserRequest);
}
```

```
@Override
@WebMethod
@WebResult(name="User")
public Users getUser(@WebParam(name="UserName") String userName) {
return new UserDAO().getUser(userName);
}

@Override
@WebMethod
@WebResult(name="Users")
public List<Users> getAllUsers() {
return new UserDAO().getAllUsers();
}
```

```
package service;

import javax.xml.ws.Endpoint;

public class MainUserPublish {

public static void main(String[] args) {
String url = "http://localhost:8086/ws/users";
Endpoint.publish(url, new UserWSImpl());
System.out.println(url);

}

}
```

Case study add Database connection service (DAO)

Step 7: Deployment Web Service

```
package service;

import javax.xml.ws.Endpoint;

public class MainUserPublish {

public static void main(String[] args) {
String url = "http://localhost:8086/ws/users";
Endpoint.publish(url, new UserWSImpl());
System.out.println(url);

}

}
```

http://localhost:8086/ws/users?wsdl

https://github.com/tafoca/Tp-Tutorial-tafo-group/tree/main/UserModule

Case study add Database connection service (DAO)

Step 7: Deployment Web Service : AddUser request

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns0="http://service/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns0:addUser>
         <User>

<userActiveStatus>BOOLEAN</userActiveStatus>

<userCategory>STRING</userCategory>
            <userId>INT</userId>
            <userLevel>DOUBLE</userLevel>
            <userName>STRING</userName>
         </User>
      </ns0:addUser>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<!-- Auto generated server sample response. -->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns0="http://service/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns0:addUserResponse>

<ResponseMessage>STRING</ResponseMessage>
      </ns0:addUserResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```