JAXWS coding

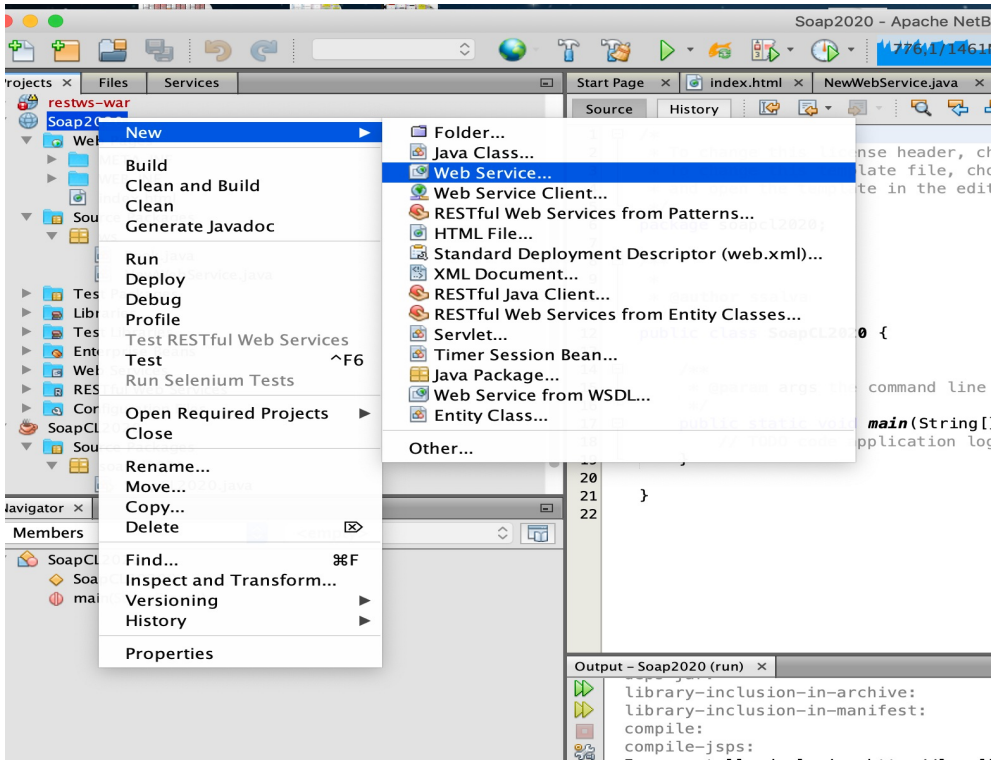
Utilisation de JAXWS dans netbeans

* Utilisation transparente (et facile)
  * Création d'un projet Web (web application)
  * Ajout de Services Web
    * A partir de POJO
    * A partir de WSDL

  * Création de clients par génération de stubs (skelette)

Utilisation de JAXWS dans netbeans

* Exemple de création de service:
* Faire un projet Web Application et



Utilisation de JAXWS dans netbeans

* Exemple de création de service:

Utilisation de JAXWS dans netbeans

* Exemple de service:

ServiceEX - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects | Files | Services | Profiler

- leak
- ServiceEX
  - Web Pages
    - META-INF
    - WEB-INF
    - index.jsp
  - Source Packages
  - Libraries
  - Web Services
  - Configuration Files

Essai.java - Navigator

Members View

- Essai
  - hello(String txt) : String

index.jsp | Essai.java

Source | Design

```
5   package ex;
6
7   import javax.jws.WebService;
8   import javax.jws.WebMethod;
9   import javax.jws.WebParam;
10
11  /**
12   *
13   * @author seb
14   */
15  @WebService(serviceName = "Essai")
16  public class Essai {
17
18      /** This is a sample web service operation */
19      @WebMethod(operationName = "hello")
20      public String hello(@WebParam(name = "name") String txt) {
21          return "Hello " + txt + " !";
22      }
23  }
24
```

Output | Tasks

Apache Tomcat Log | Apache Tomcat | Retriever Output | WebApplication2 (clean)

```
undeploy?path=/WebApplication2
OK - Application non déployée pour le chemin de contexte /WebApplication2
deps-clean:
do-clean:
Deleting directory C:\Users\seb\Documents\NetBeansProjects\WebApplication2\build
check-clean:
clean:
BUILD SUCCESSFUL (total time: 2 seconds)
```

13 | 15 INS

Utilisation de JAXWS dans netbeans

* Exemple de service:

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects | Files | Services | Profiler

- leak
- ServiceEX
  - Web Pages
    - META-INF
    - WEB-INF
    - index.jsp
  - Source Packages
  - Libraries
  - Web Services
  - Configuration Files

Essai.java - Navigator

Members View

- Essai
  - hello(String txt) : String

index.jsp | Essai.java

Source | Design | 100%

Essai

Operations (1) — Add Operation... | Remove Operation

hello

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
name	java.lang.String

Quality Of Service

- Optimize Transfer Of Binary Data (MTOM)
- Reliable Message Delivery
- Secure Service

Advanced ...

Output | Tasks

Apache Tomcat Log | Apache Tomcat | Retriever Output | WebApplication2 (clean)

```
undeploy?path=/WebApplication2
OK - Application non déployée pour le chemin de contexte /WebApplication2
deps-clean:
do-clean:
Deleting directory C:\Users\seb\Documents\NetBeansProjects\WebApplication2\build
check-clean:
clean:
BUILD SUCCESSFUL (total time: 2 seconds)
```

Utilisation de JAXWS dans netbeans

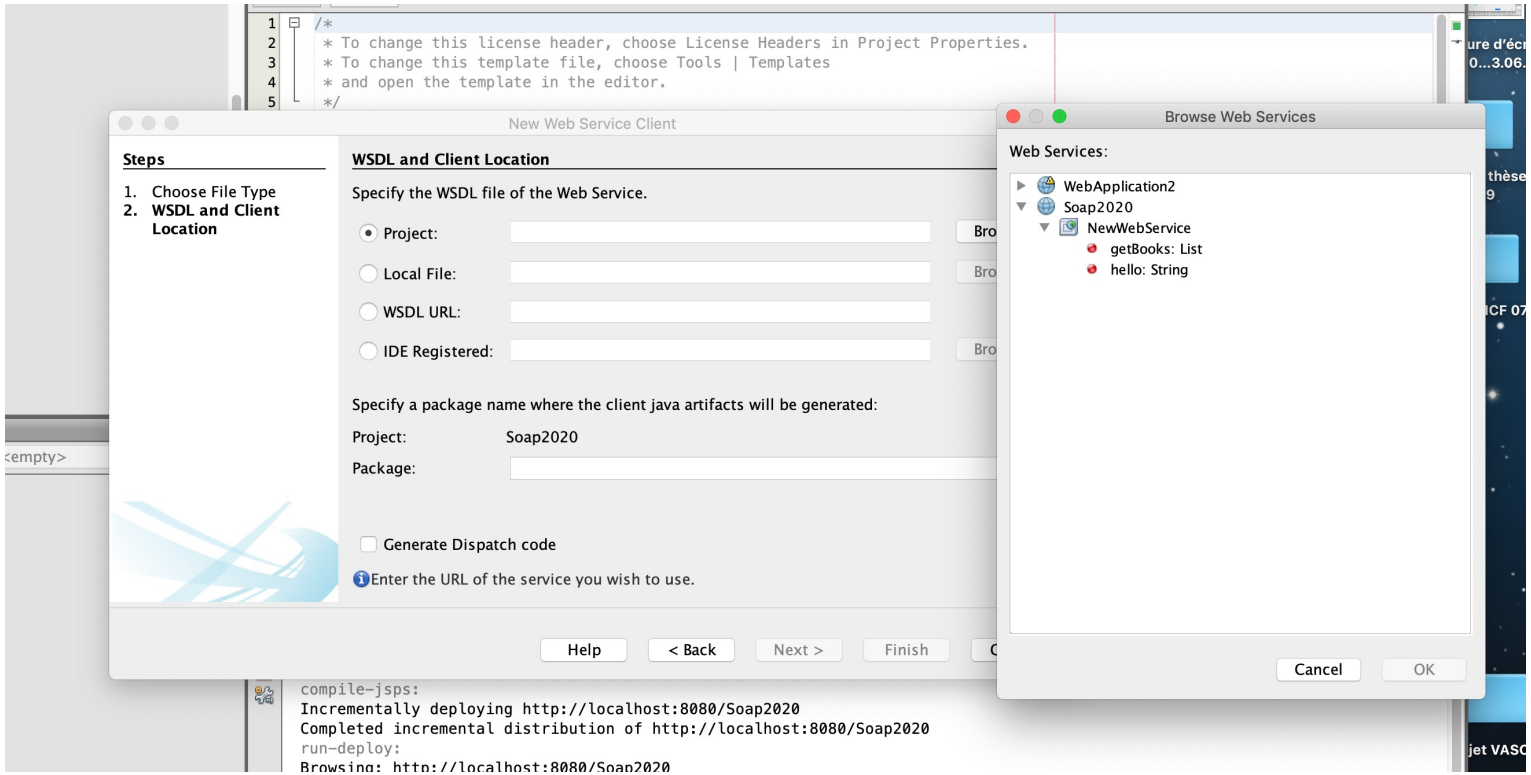
* Consommation d’un service en utilisant un proxy qui est généré (par la cmd wsimport)

  * Création d’un projet Java application
  * New Web service Client => generation de stubs
  * Dans le code, faire insertion de code, appel de service Web

Utilisation de JAXWS dans netbeans

Consommation d'un service

Choix du service :



Utilisation de JAXWS dans netbeans

Consommation d'un service

Insertion de code :

```
6   package soapcl2020;
7
8   /**
9    *
10   * @author ssalva
11   */
12  public class SoapCL2020 {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
22
```

Generate
- Constructor...
- Logger...
- toString()...
- Override Method...
- Add Property...
- Call Web Service Operation...
- Generate REST Client...

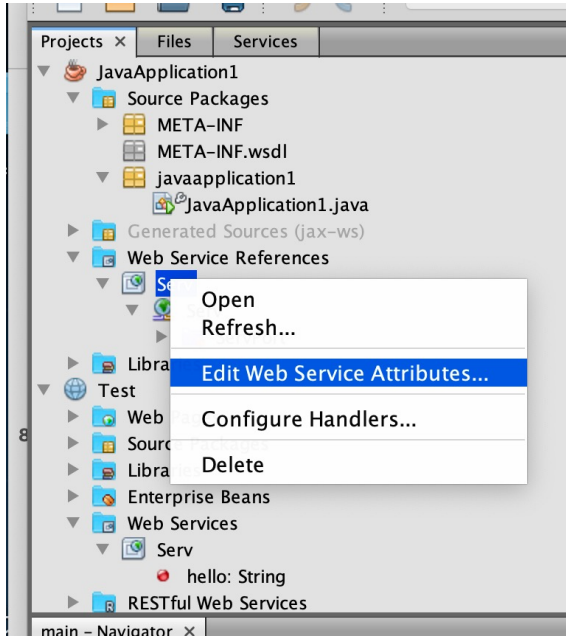
Output

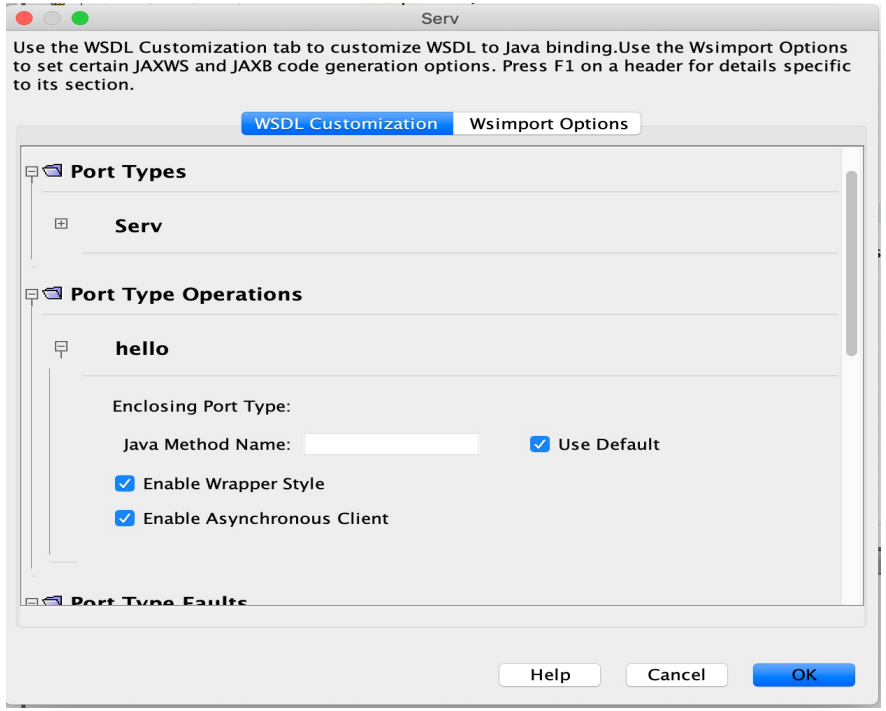
Apache Tomcat or TomEE (1) Log | Apache Tomcat or TomEE (1) | Retriever Output

```
cl/GetBooksResponse.java
cl/Hello.java
cl/HelloResponse.java
cl/NewWebService.java
cl/NewWebService_Service.java
cl/ObjectFactory.java
cl/package-info.java
Copying 8 files to /Users/ssalva/NetBeansProjects/SoapCL2020/build/generated-so
BUILD SUCCESSFUL (total time: 1 second)
```

Utilisation de JAXWS dans netbeans

Consommation d'un service en mode asynchrone





Puis insertion de code comme precedemment

Projets Maven

Code service Web

* Créer un projet Maven
* Ajouter dépendances

```
<dependency>
  <groupId>com.sun.xml.ws</groupId>
  <artifactId>jaxws-rt</artifactId>
  <version>2.3.2?</version>
</dependency>
```

* Créer une classe et faire le code JAXWS

Code service Web

* Configurer endpoints
  * Sous WEB-INF, créer le fichier **sun-jaxws.xml** et ajouter

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints version="2.0" xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime">
 <endpoint implementation="ws.Hello" name="HelloService" url-pattern="/HelloService"/>
</endpoints>
```

Code service Web

* Configurer Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <listener>
    <listener-class>com.sun.xml.ws.transport.http.servlet.WSServlet
ContextListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>HelloService</servlet-name>
    <servlet-class>com.sun.xml.ws.transport.http.servlet.WSSer
vlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloService</servlet-name>
    <url-pattern>/HelloService</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

Code sevice client

* Créer un projet Maven
* Ajouter dans pom.xml lib et build pouir génération de code avec wsimport
* Ajouter dépendances

Code sevice client

```
<dependency>
 <groupId>javax.xml.ws</groupId>
 <artifactId>jaxws-api</artifactId>
 <version>2.3.1</version>
</dependency>
<dependency>
        <groupId>javax.xml</groupId>
        <artifactId>webservices-api</artifactId>
        <version>2.0</version>
        <type>jar</type>
      </dependency>
```

```
<dependency>
        <groupId>jakarta.xml.ws</groupId>
        <artifactId>jakarta.xml.ws-api</artifactId>
        <version>2.3.3</version>
      </dependency>
      <dependency>
        <groupId>com.sun.xml.ws</groupId>
        <artifactId>jaxws-rt</artifactId>
        <version>2.3.3</version>
      </dependency>
```

Code sevice client

* Faire le code client dans un main en lisant le code généré

Code sevice client

* Faire un code client asynchrone (plus difficile)
* Créer un fichier de binding :

* Ex: *async-bindings.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<bindings xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  wsdlLocation="http://localhost:9980/hello?wsdl"
xmlns="http://java.sun.com/xml/ns/jaxws">
  <!-- applies to wsdl:definitions node, that would mean the entire wsdl -->
  <enableAsyncMapping>true</enableAsyncMapping>
</bindings>
```

Code sevice client

* Faire un code client asynchrone (plus difficile)
* Modifier le pom.xml pour prendre en compte le binding :

```
<configuration>
                    <wsdlUrls>
                      <wsdlUrl>http://localhost:8080/hello?wsdl</wsdlUrl>
                    </wsdlUrls>
                    <bindingDirectory>${basedir}/src/main/resources/jaxws</bindingDirectory>
                    <keep>true</keep>
                    <packageName>jcg.demo.jaxws.client.hello</packageName>
                    <sourceDestDir>src/generated/java</sourceDestDir>
 </configuration>
```

SOAP et Springboot

Soap et Springboot

* Oui c’est possible
* Mais il faut écrire les schéma XML (ceux en debut du WSDL) à la main ☹

Soap et Springboot

* Approche Top-down (je passe, a priori Springboot ne fait pas)

* Approche **Bottom-Up**
* D'après la doc spring il faut écrire les schéma XML (ceux en debut du WSDL) à la main ☹
* https://spring.io/guides/gs/producing-web-service/
  * Ici: pas besoin de schémas ? https://blog.sodifrance.fr/creer-des-services-web-soap-simplement-avec-jax-ws-et-spring-boot-part-i/

Soap et Springboot

Create an XML Schema to Define the Domain

```
<xs:element name="getCountryRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"
type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element
name="getCountryResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="country"
type="tns:country"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="country">
    <xs:sequence>
      <xs:element name="name"
type="xs:string"/>
      <xs:element name="population"
type="xs:int"/>
      <xs:element name="capital"
type="xs:string"/>
      <xs:element name="currency"
type="tns:currency"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="currency">
    <xs:restriction base="xs:string">
      <xs:enumeration value="GBP"/>
      <xs:enumeration value="EUR"/>
      <xs:enumeration value="PLN"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Soap et Springboot

Create an XML Schema to Define the Domain

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://localhost:8080/Books"

targetNamespace="http://localhost:8080/Books" elementFormDefault="qualified">
  <xs:element name="getBookRequest">
    <xs:complexType>
    </xs:complexType>
  </xs:element>
  <xs:element name="getBookResponse">
  <xs:complexType>
    <xs:sequence>
<xs:element minOccurs="0" maxOccurs="unbounded" name="book" type="tns:book"
        />
    </xs:sequence>
</xs:complexType>
  </xs:element>
  <xs:complexType name="book">
    <xs:sequence>
      <xs:element name="isbn" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Soap et Springboot

* **Create Country Service Endpoint (code du service)**

```
@Endpoint public class CountryEndpoint {
private static final String NAMESPACE_URI = "http://spring.io/guides/gs-producing-web-service";
private CountryRepository countryRepository;

@PayloadRoot(namespace = NAMESPACE_URI, localPart = "getCountryRequest")
 @ResponsePayload
public GetCountryResponse getCountry(@RequestPayload GetCountryRequest request) {
GetCountryResponse response = new GetCountryResponse();
response.setCountry());
return response; } }
```

Soap et Springboot

* Configuration

Des namespaces, des schemas, des noms d’opération dans

src/…/WebServiceConfig.java

Soap et Springboot

* **Make the Application Executable**

```
@SpringBootApplication
public class ProducingWebServiceApplication {
public static void main(String[] args) {
SpringApplication.run(ProducingWebServiceApplication.class, args); } }
```

Soap et Springboot

* **Faire un client**
* **https://spring.io/guides/gs/consuming-web-service/**

* Plus simple?
* Utilise wsimport pour générer les stubs dans *target/generated-sources*

* *Faire une classe fille à* WebServiceGatewaySupport pour préparer les appels
* *Configurer la sérialisation / deserialisation*
* *Et appel*

Soap et Springboot

* **Faire un client**
* https://spring.io/guides/gs/consuming-web-service/

* Exemple *classe fille à* WebServiceGatewaySupport pour préparer les appels

```
public class BookClient extends WebServiceGatewaySupport {
 private static final Logger log = LoggerFactory.getLogger(BookClient.class);

 public GetBookResponse getbook() {

  GetBookRequest request = new GetBookRequest();

  GetBookResponse response = (GetBookResponse) getWebServiceTemplate()
    .marshalSendAndReceive("http://localhost:8080/ws/books", request,
      new SoapActionCallback(
        "http://localhost:8080/Books"));
  return response;
```

Soap et Springboot

* **Faire un client**
* https://spring.io/guides/gs/consuming-web-service/

```
@SpringBootApplication
 public class ConsumingWebServiceApplication {
public static void main(String[] args) {
SpringApplication.run(ConsumingWebServiceApplication.class, args); }

@Bean CommandLineRunner lookup(CountryClient quoteClient) {
return args -> { String country = "Spain";
 if (args.length > 0) { country = args[0]; }
GetCountryResponse response = quoteClient.getCountry(country);
System.err.println(response.getCountry().getCurrency());
};
 } }
```

PHP et Soap

PHP et les web services

Possibilité de créer des services web et des clients avec PHP

Les web services:
1.Création (manuelle) du fichier WSDL
2.Création du service web en PHP:

Possibilités:
fonctions natives
frameworks:
nusoap par exemple (http://sourceforge.net/projects/nusoap/)
(génération du wsdl, mais parfois il faut quasiment le définir dans le code)

PHP et les web services

Exemple Code de web service (http://stackoverflow.com/questions/2573494/php5-and-soap-wsdl)

```
<?php
function getRot13($pInput) {
   $rot = str_rot13($pInput);
   return($rot);
}

function getMirror($pInput) {
   $mirror = strrev($pInput);
   return $mirror;
}

ini_set("soap.wsdl_cache_enabled", "0");

$server = new SoapServer('scramble.wsdl');

$server->addFunction("getRot13");
$server->addFunction("getMirror");

$server->handle();
```

PHP et les web services

Code client:

```
<?php
// turn off the WSDL cache
ini_set("soap.wsdl_cache_enabled", "0");

$client = new SoapClient("http://localhost/test/scramble.wsdl");

$origtext = "mississipi";

print("The original text : $origtext\n");

$mirror = $client->getMirror($origtext);
print("The mirrored text : $mirror\n");

$scramble = $client->getRot13($mirror);
print("The scrambled text : $scramble\n");
```

Si plusieurs paramètres: utiliser array

PHP et les web services

Code du client: méthode soapcall (méthode PHP)

```
/* Initialize webservice with your WSDL */
$client = new SoapClient("http://localhost/Service1?wsdl");

/* Set your parameters for the request */
$params = array(
  “isbn" => “1234”,
  "description" => “Book",
  "amount" => 500,
);

/* Invoke webservice method with your parameters, in this case: Buy */
$response = $client->__soapCall(“Buy", array($params));

/* Print webservice response */
var_dump($response);
```

2 mots sur la sécurité

service web et sécurité

Il existe plusieurs standards de sécurité :

- WS-Security: standard proposé par Microsoft pour la sécurisation des services interopérables. Plus large que SAML, il intègre le traitement de l'authentification, du chiffrement et de l'intégrité des données.
  - XML encryption : cryptage de documents XML
  - XML signature: signature digitale dans un document XML

- SAML-XML: (security assertions markup language) spécifie une procédure d'authentification

- XKMS: utilisation de clés publiques

service web et sécurité

Implémentations

- Des API payantes ☹, ou
- http://ws.apache.org/wss4j/ en lien avec https://cxf.apache.org/docs/ws-security.html, spingboot compliant ?

- HTTPS
- Ex: https://www.codeflow.site/fr/article/webservices__jax-ws__application-authentication-with-jax-ws

service web et sécurité

Parefeu XML:

se déploie en aval du pare-feu traditionnel

conçu spécifiquement pour inspecter les flux XML et SOAP sur protocoles de base HTTP/HTTPS et éventuellement sur des protocoles tiers

sécurité sur:

- la validation des messages,
- l'« obfuscation » des exceptions,
- la détection des intrusions,
- le routage selon le type de contenu,
- la transition de protocoles
- la sécurité au niveau de la couche de message.

Conclusion

Manque de performance ?

Mais apporte sécurité, transactions, Uniformité

Mode SMTP pour envoi importants

A recommander pour transactions sensibles entre sites