

# Les Architectures Orientées Services

SOA

**Gérer la Sécurité dans les AOS et les Services Web - Suite**

# Plan

1. La Sécurité dans les Services Web
2. Authentification dans JAX-WS

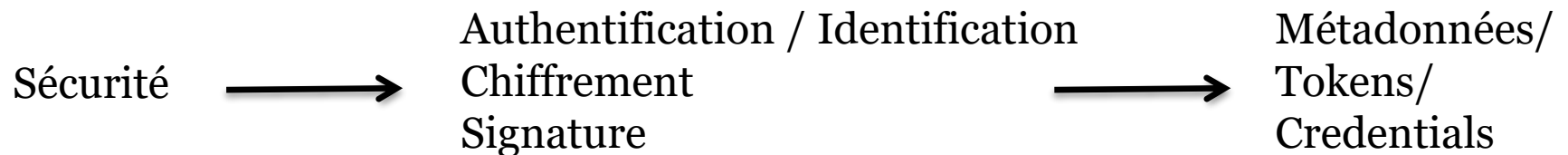
## La Sécurité dans les Services Web

- Les spécifications de la pile permettant la sécurisation des Web Services au travers de méthodes basées sur les messages SOAP eux-mêmes.

### WS-Security

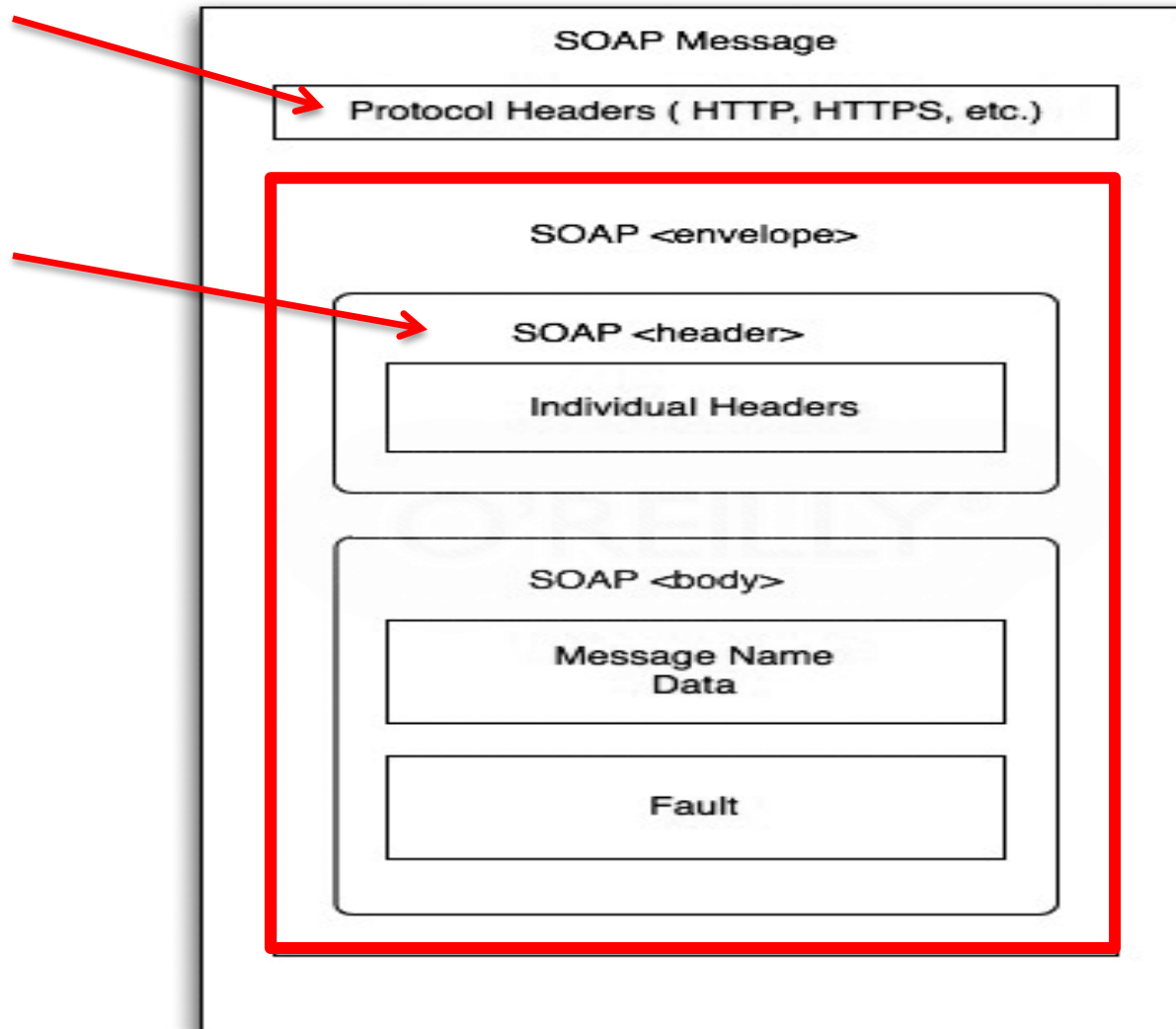
- WS-Security propose de sécuriser de manière intrinsèque les messages SOAP :
  - Assurer la **confidentialité** d'un fragment du message SOAP avec **XML Encryption**.
  - Assurer **l'intégrité** d'un fragment du message SOAP en le signant avec **XML Signature**.
  - **Certifier** l'identité de l'accédant auprès du serveur SOAP avec **SAML**.

# La Sécurité dans les Services Web

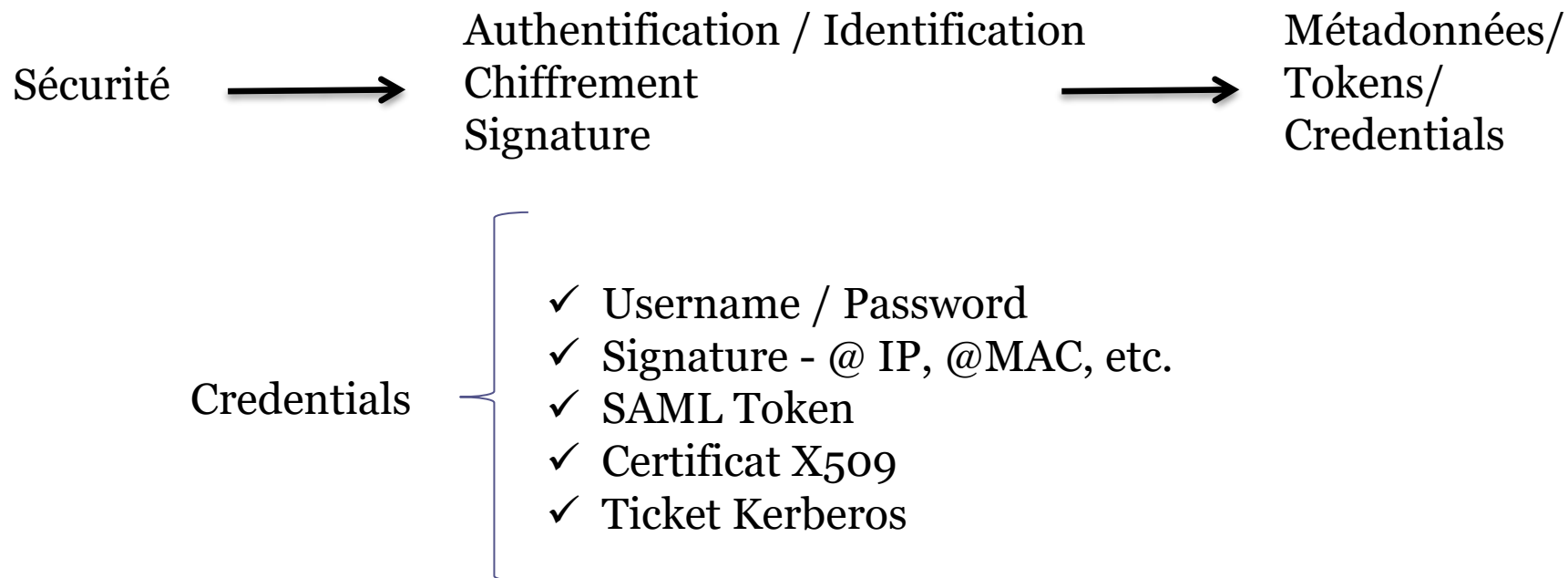


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Header>
    ...
  </soapenv:Header>
  <soapenv:Body>
    <!-- Contenu de la Requête -->
  </soapenv:Body>
</soapenv:Envelope>
```

# La Sécurité dans les Services Web



# La Sécurité dans les Services Web



- Ces Credentials dans JAX-WS sont appelés **Message Context**

# La Sécurité dans les Services Web

Exemples : Identification / **Authentication** Client via

- 1) Username / Password dans HTTP Header
- 2) Adresse IP dans SOAP Header



# La Sécurité dans les Services Web

Exemples : Identification / **Authentication** Client via

- 1) Username / Password dans HTTP Header
- 2) Adresse IP dans SOAP Header

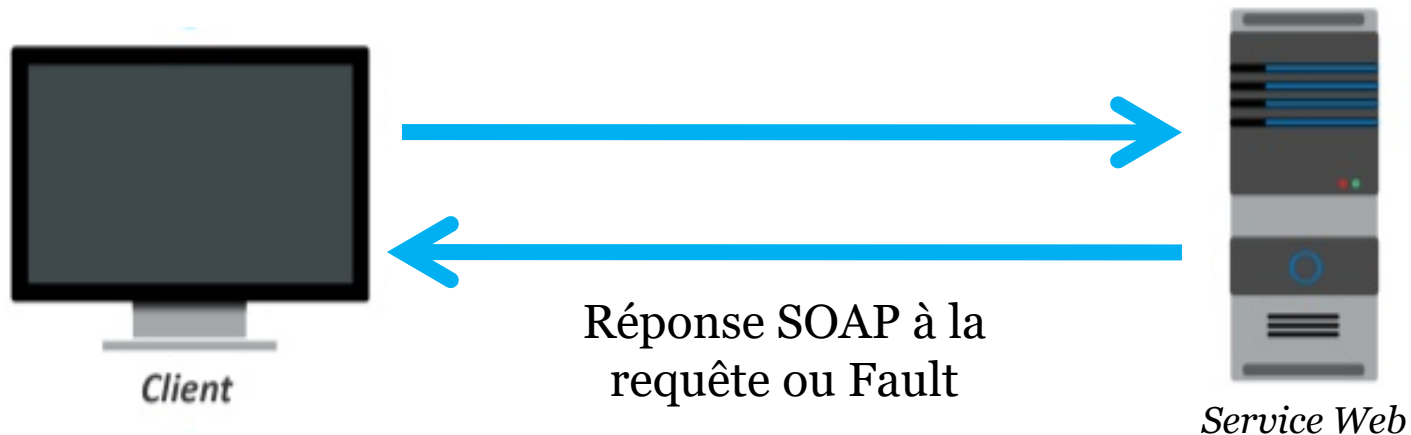




# La Sécurité dans les Services Web

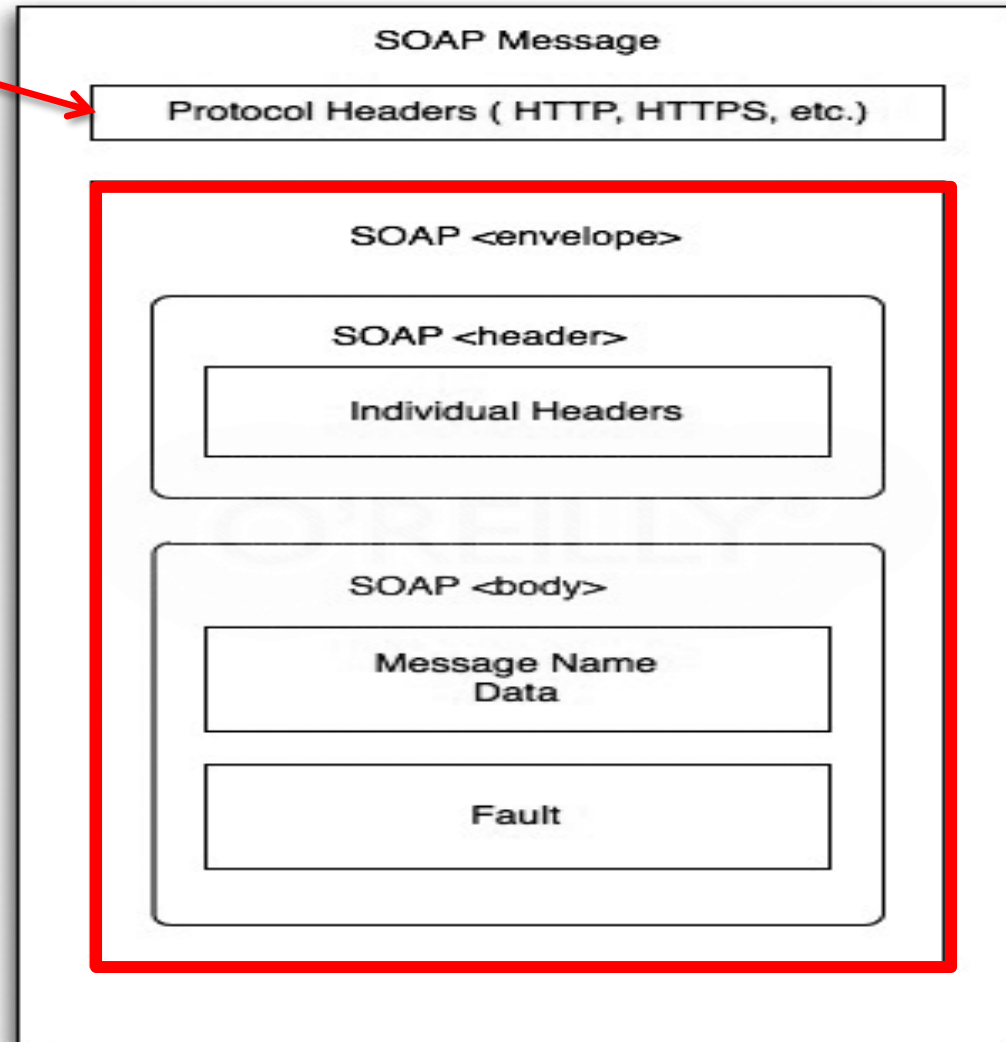
Exemples : Identification / **Authentication** Client via

- 1) Username / Password dans HTTP Header
- 2) Adresse IP dans SOAP Header



# La Sécurité dans les Services Web

1



## Authentication dans JAX-WS

Exemples : Identification / Authentication Client via

- 1) **Username / Password dans HTTP Header**
- 2) Adresse IP dans SOAP Header



## Authentification dans JAX-WS

### Exemple 1 : Implémentation coté **Client**

=> Ajouter Username/Password à l'entête HTTP du message SOAP

<b>Username</b>	<b>master1</b>
<b>Password</b>	<b>soaws</b>

1. Récupérer le **Message Context** de la requête SOAP
2. Créer et stocker les credentials
3. Ajouter les credentials au Message Context récupéré – autant que Header HTTP

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Client**

### 1. Récupérer le Message Context de le requête SOAP

Interface BindingProvider + méthode getRequestContext

```
BindingProvider    bp = (BindingProvider)  stub;  
  
Map <String, Object>    reqContext = bp.getRequestContext() ;
```

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Client**

### 2. Créer et stocker les credentials

Collection HashMap	Username	master1
	Password	soaws

```
HashMap <String, List<String>> headers =  
    new HashMap<String, List<String>>();  
  
headers.put("Username", Collections.singletonList("master1"));  
  
headers.put("Password", Collections.singletonList("soaws"));
```

## Authentification dans JAX-WS

### Exemple 1 : Implémentation coté **Client**

3. Ajouter les credentials au Message Context récupéré – autant que Header HTTP

```
reqContext.put (MessageContext.HTTP_REQUEST_HEADERS, headers) ;
```

## Authentication dans JAX-WS

### Exemple 1 : Implémentation coté **Client**

```
ConvertImplService service = new ConvertImplService()
Convertisseur stub = service.getConvertImplPort();

//1
BindingProvider    bp = (BindingProvider)  stub;
Map <String, Object>   reqContext = bp.getRequestContext();

//2
HashMap <String, List<String>> headers =
                                new HashMap<String, List<String>>();
headers.put("Username", Collections.singletonList("master1"));
headers.put("Password", Collections.singletonList("soaws"));

//3
reqContext.put(MessageContext.HTTP_REQUEST_HEADERS, headers);

System.out.println(stub.getDinarFromEuro(10));
```



## Authentification dans JAX-WS

### Exemple 1 : Implémentation coté **Service Web**

=> Vérifier Username/Password, puis autoriser ou refuser l'accès à l'opération.

1. Récupérer le Message Context de la requête SOAP reçue
2. Récupérer le HTTP Header du Message Context
3. Récupérer les credentials Username/Password du header http
4. Vérifier et comparer - exécuter l'opération invoquée ou renvoyer un Fault

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Service Web**

### 1. Récupérer le Message Context de le requête SOAP reçue

Interface `WebServiceContext` + méthode `getMessageContext`

```
@Resource  
WebServiceContext wsContext;  
  
MessageContext msgContext = wsContext.getMessageContext();
```

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Service Web**

### 2. Récupérer le HTTP Header du Message Context

```
Map httpHeaders =  
    (Map) msgContext.get (MessageContext.HTTP_REQUEST_HEADERS) ;
```

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Service Web**

### 3. Récupérer les credentials Username/Password du header HTTP

```
List usernameList = (List) httpHeaders.get("Username");  
List passwordList = (List) httpHeaders.get("Password");
```

# Authentification dans JAX-WS

## Exemple 1 : Implémentation coté **Service Web**

4. **Vérifier et comparer** - exécuter l'opération invoquée ou renvoyer un Fault .

```
private Boolean isAuthenticated ( ) {  
  
    1) ...  
    2) ...  
    3) ...  
    4) ...  
  
    return true/false;  
}
```

## Authentication dans JAX-WS

### Exemple 1 : Implémentation coté **Service Web**

4. **Vérifier et comparer** - exécuter l'opération invoquée ou renvoyer un Fault .

```
private boolean isAuthenticated() {  
  
    MessageContext msgContext = wsContext.getMessageContext();  
  
    Map httpHeaders = (  
        (Map)msgContext.get(MessageContext.HTTP_REQUEST_HEADERS);  
  
    List usernameList = (List) httpHeaders.get("Username");  
    List passwordList = (List) httpHeaders.get("Password");  
  
    if(usernameList.contains("master1") &&  
        passwordList.contains("soaws"))  
        return true;  
    return false;  
}
```

## Authentification dans JAX-WS

### Exemple 1 : Implémentation coté **Service Web**

4. **Vérifier et comparer** - exécuter l'opération invoquée ou renvoyer un Fault .

```
@Override
@WebMethod
public double getDinarFromEuro(double euro) {

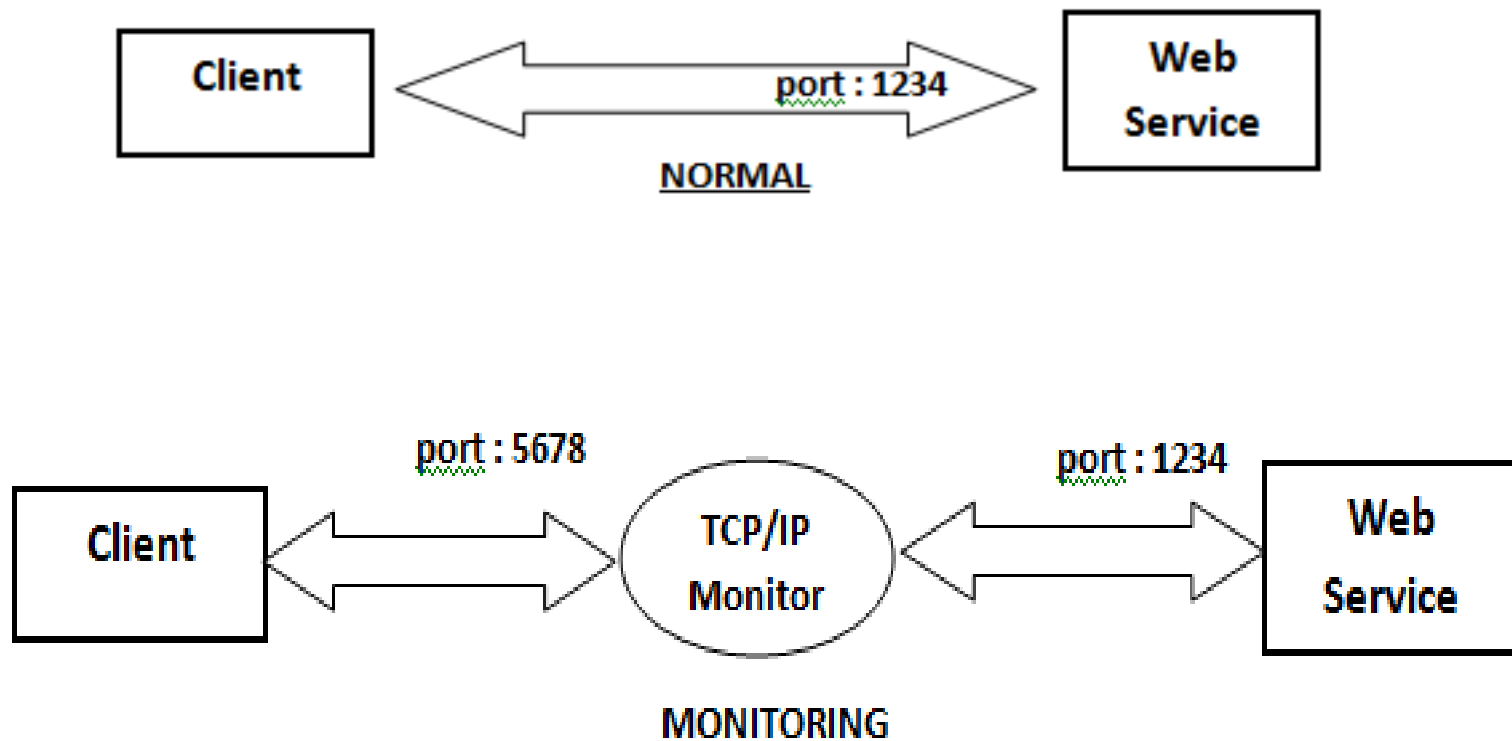
    if(isAuthenticated())
        return euro * 215;
    else
        throw new HTTPException(401);

}
```

## Authentification dans JAX-WS

### Exemple 1 : **Tracer les messages SOAP** – TCP/IP Monitor d'Eclipse

- Héberger un serveur au milieu :





## Authentification dans JAX-WS

### Exemple 1 : **Tracer les messages SOAP** – TCP/IP Monitor d'Eclipse

➤ Héberger un serveur au milieu:

- 1) Depuis le projet du **client**, ouvrir la classe d'implémentation `<ServiceName>` générée via wsimport.
- 2) Rechercher et remplacer tous les numéros de port 1234 par 5678.
- 3) Depuis le menu principal d'Eclipse, cliquer sur *Window* → *Preferences* → *Run/Debug* → *TCP/IP Monitor*.

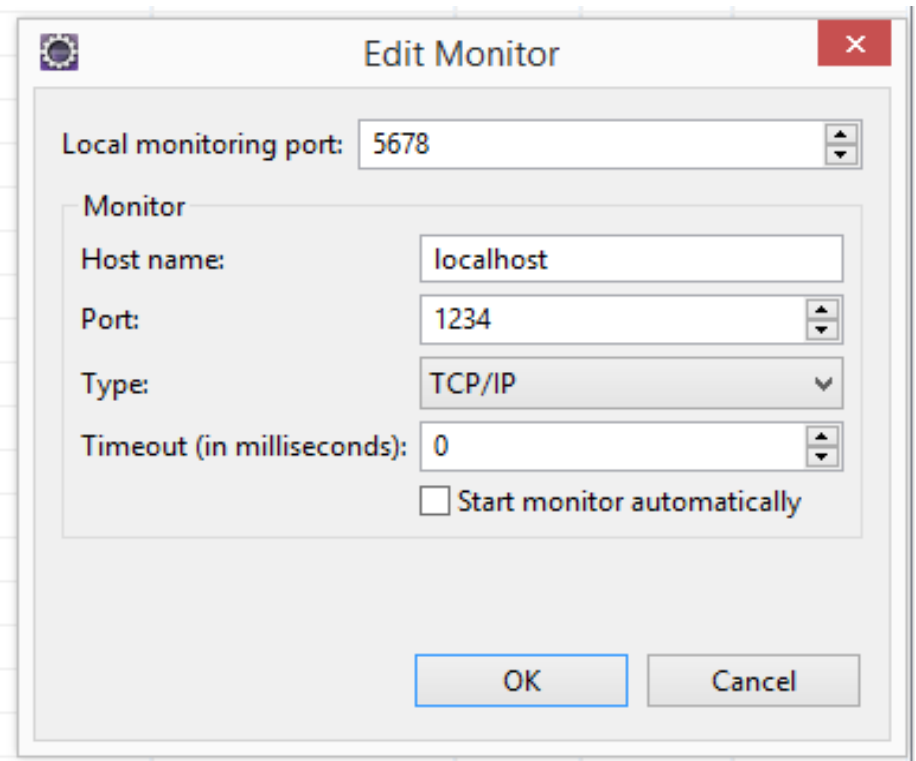
## Authentification dans JAX-WS

### Exemple 1 : Tracer les messages SOAP – TCP/IP Monitor d'Eclipse

➤ Héberger un serveur au milieu:

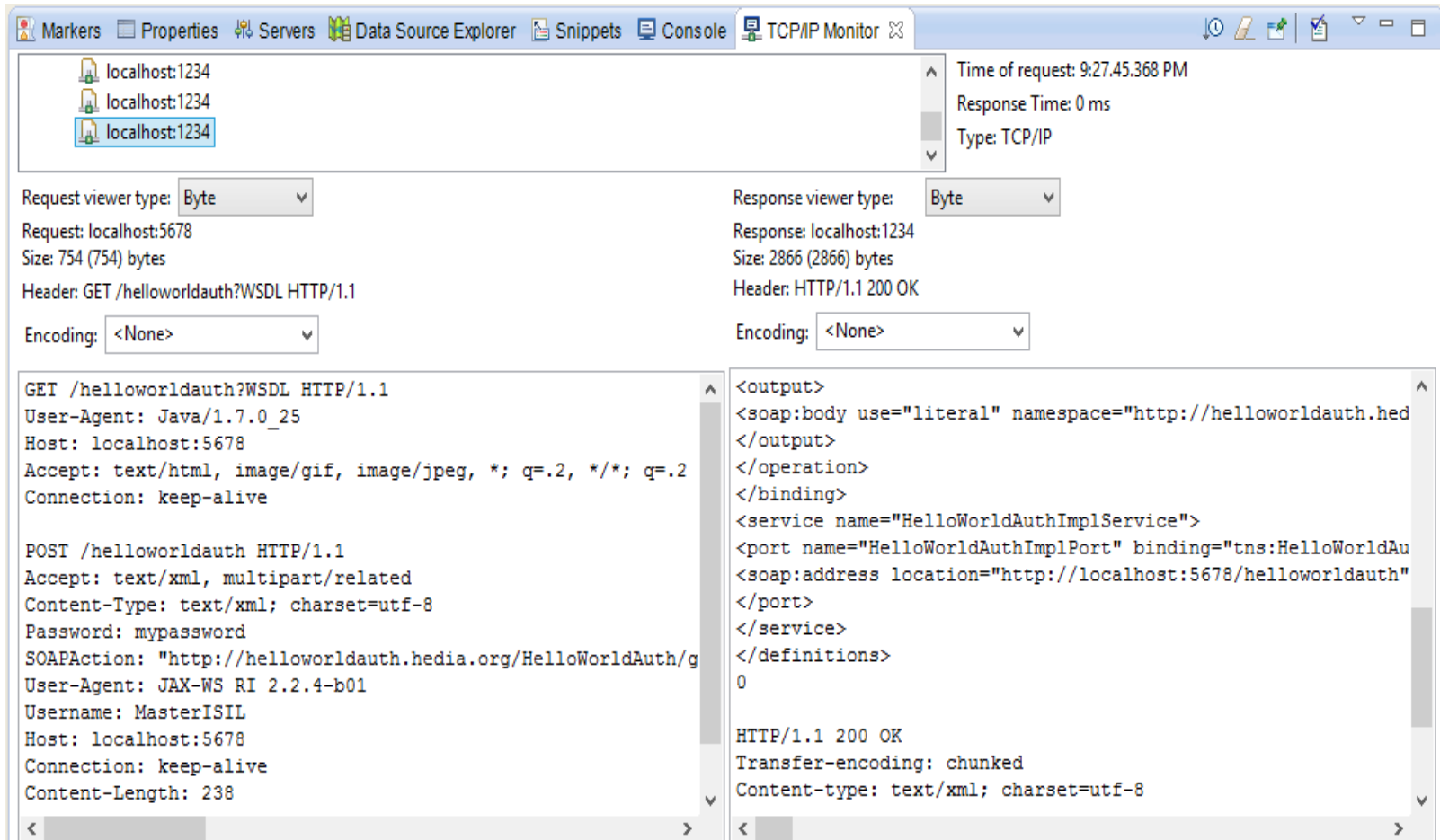
4) Add puis remplir les informations :

5) Ok et Start Moniteur ajouté.



# Authentification dans JAX-WS

## Exemple 1 : Tracer les messages SOAP – TCP/IP Monitor d'Eclipse



The screenshot displays the Eclipse TCP/IP Monitor interface. The top toolbar includes icons for Markers, Properties, Servers, Data Source Explorer, Snippets, Console, and TCP/IP Monitor. The main window is divided into several sections:

- Markers:** A list of three entries labeled 'localhost:1234', with the bottom one selected.
- Request/Response Summary:**
  - Request:** localhost:5678, Size: 754 (754) bytes, Header: GET /helloworldauth?WSDL HTTP/1.1, Encoding: <None>.
  - Response:** localhost:1234, Size: 2866 (2866) bytes, Header: HTTP/1.1 200 OK, Encoding: <None>.
- Request Viewer (Byte):**

```
GET /helloworldauth?WSDL HTTP/1.1
User-Agent: Java/1.7.0_25
Host: localhost:5678
Accept: text/html, image/gif, image/jpeg, *: q=.2, */*; q=.2
Connection: keep-alive

POST /helloworldauth HTTP/1.1
Accept: text/xml, multipart/related
Content-Type: text/xml; charset=utf-8
Password: mypassword
SOAPAction: "http://helloworldauth.hedia.org>HelloWorldAuth/g
User-Agent: JAX-WS RI 2.2.4-b01
Username: MasterISIL
Host: localhost:5678
Connection: keep-alive
Content-Length: 238
```
- Response Viewer (Byte):**

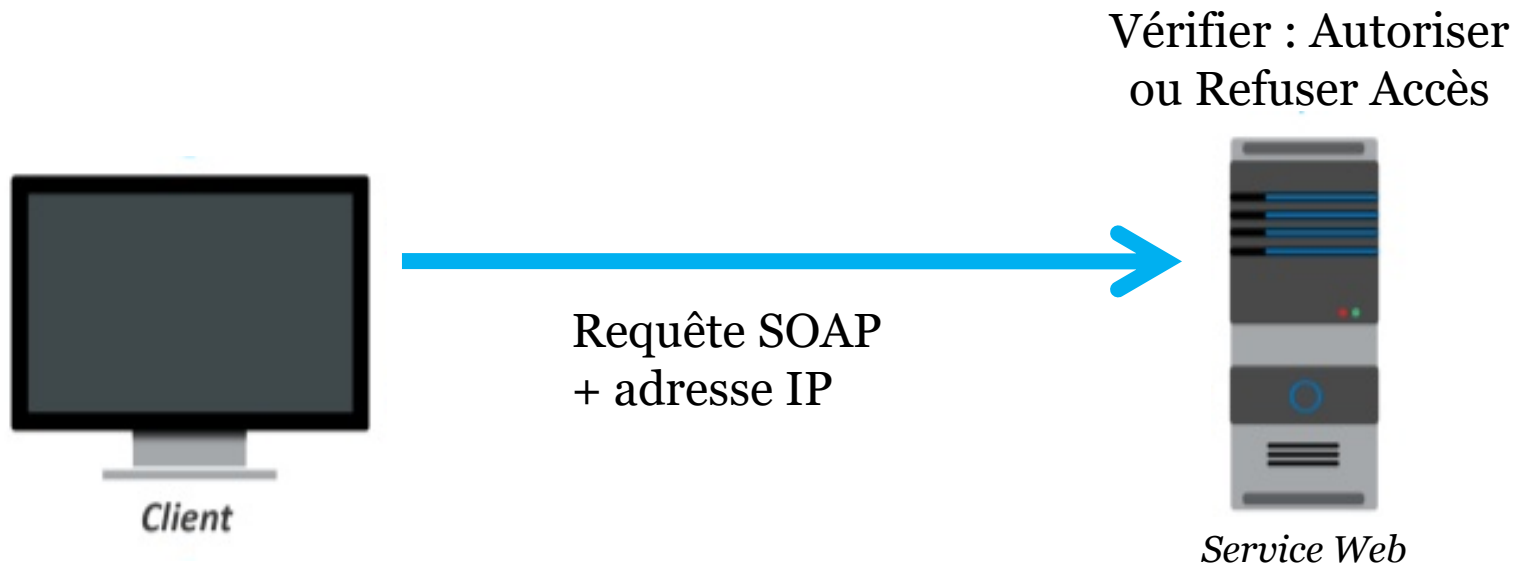
```
<output>
<soap:body use="literal" namespace="http://helloworldauth.hed
</output>
</operation>
</binding>
<service name="HelloWorldAuthImplService">
  <port name="HelloWorldAuthImplPort" binding="tns:HelloWorldAu
  <soap:address location="http://localhost:5678/helloworldauth"
  </port>
</service>
</definitions>
0

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

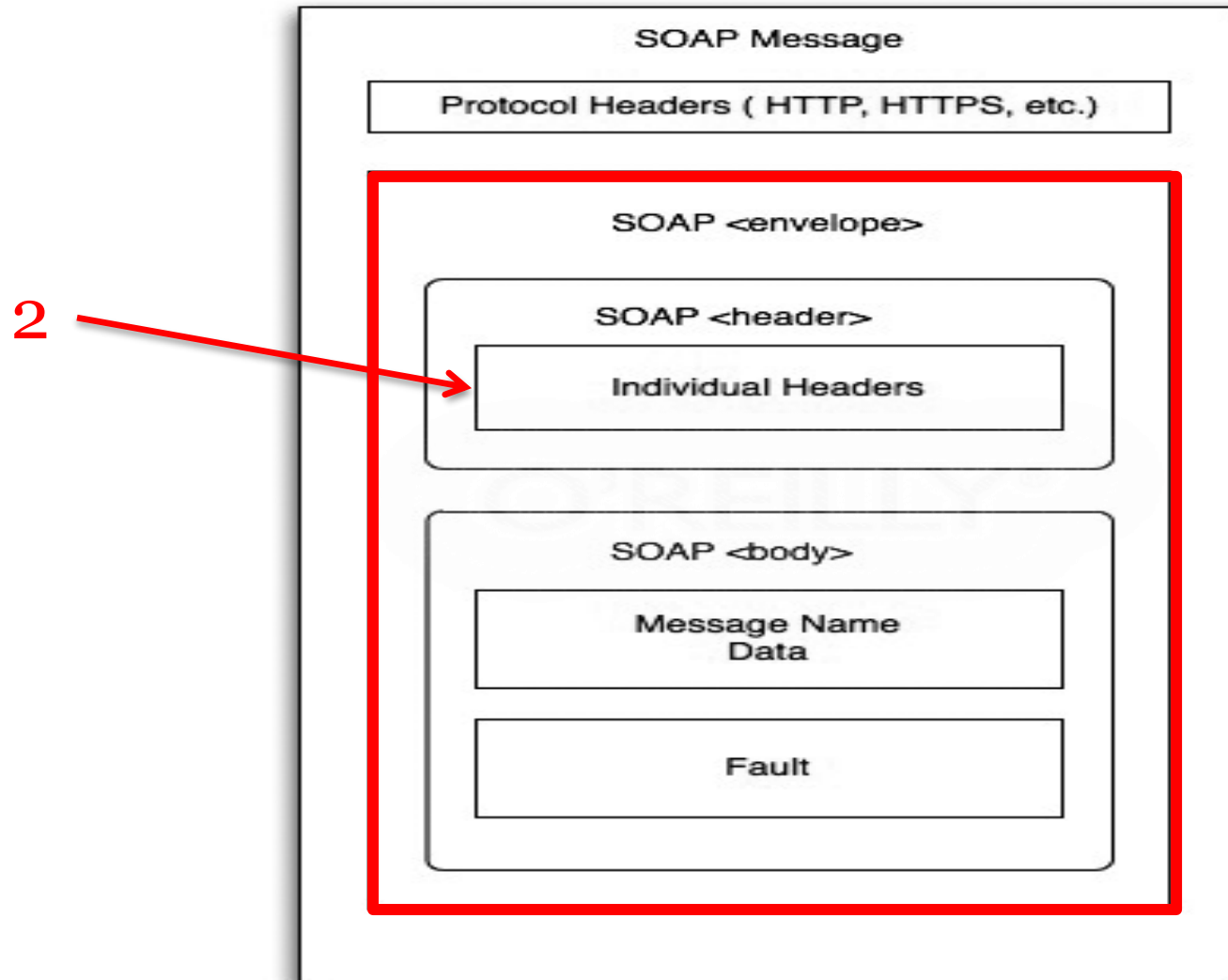
# Authentification dans JAX-WS

Exemples : Identification / Authentification Client via

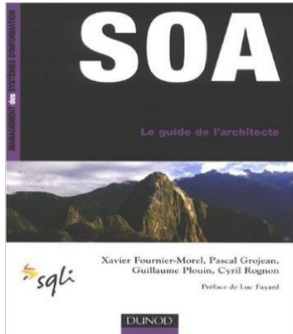
- 1) Username / Password dans HTTP Header
- 2) **Adresse IP dans SOAP Header**



## Authentication dans JAX-WS

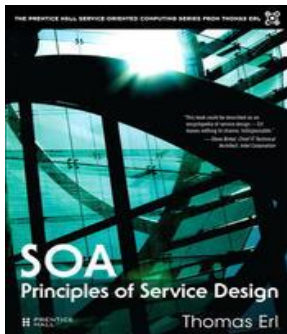


# Ressources



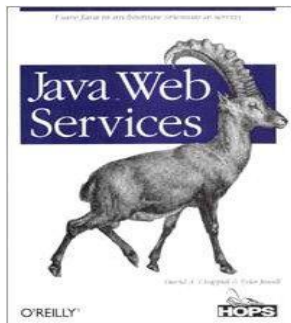
## **Le guide de l'architecte du SI**

- ✓ Auteur : Xavier Fournier-Morel, Pascal Grosjean, ...
- ✓ Éditeur : Dunod
- ✓ Edition : Octobre 2006 - 302 pages - ISBN : 2100499726



## **SOA Principles of Service Design**

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



## **Java Web Services**

- ✓ Auteur : David Chappell & Tyler Jewell
- ✓ Éditeur : O'Reilly
- ✓ Edition : Mars 2002 - 276 pages - ISBN : 0-596-00269-6

# Ressources

## **Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing**

- ✓ Auteur : Armando Fox and David Patterson
- ✓ Éditeur : Strawberry Canyon LLC
- ✓ Edition : Aout 2012 - 412 pages - ISBN : 0984881212

Livre blanc SOA : Architecture Logique : Principes, structures et bonnes pratiques Auteur: Gilbert Raymond.Version 2.

Cours – Mickael Baron – SOA et Microservices

- ✓ [http://mbaron.developpez.com/#page\\_soa](http://mbaron.developpez.com/#page_soa)

Cours – Koushik Kothagal - Developing SOAP Web Services with JAX-WS

- ✓ [https://javabrainz.io/courses/javaee\\_jaxws/lessons/Introduction-to-Web-Services](https://javabrainz.io/courses/javaee_jaxws/lessons/Introduction-to-Web-Services)