Faculté des Sciences de Gabès

Cours Sécurité des Services Orientés Web

Chapitre 3 : SOAP : Simple Object Access Protocol

Faîçal Felhi

felhi_fayssal@yahoo.fr

Le Web et le client serveur

- Proposition Web actuelle insuffisante
- Autres plates-formes client / serveur
  - Java RMI
    - mono-langage : Java, multi-plateforme (JVM), SUN
    - Pas réaliste pour une application industrielle (performance, sécurité, …)
  - CORBA / IIOP
    - Multilangage, multi-plateforme, Multi-vendeurs, OMG
    - Installation « coûteuse » si on doit acheter un ORB
      - Mais les open-sources sont gratuits et souvent plus complet
        - www.objectweb.org
  - DCOM
    - multi-langages, plateforme Win32, Propriétaire Microsoft
    - protocole orienté connexion
      - Échange de nombreux paquets pour créer/maintenir une session
    - Faible diffusion
      - Pas disponible sur MacOS, NT3.51, Win95, WinCE2
      - Coûteux sur UNIX, MVS, VMS ou NT

Faîçal Felhi - felhi_fayssal@yahoo.fr

Le bilan...

- Approche insatisfaisante :
  - Protocoles sophistiqués
    - Coût d'installation (faite par un administrateur, consomme des ressources : machines, personnels, …)
    - Difficile à porter sur d'autres plates-formes
  - Règles de fonctionnement strictes en environnement ouvert (le Net)
    - Environnement sécurisé (intérieur d'un intranet)
    - Incapacité à fonctionner en présence de pare-feu (utilisation impossible sur Internet)
      - Les nouvelles version de CoRBA peuvent ouvrir un port sur un pare-feu comme le port 80 d'HTTP

Faîçal Felhi - felhi_fayssal@yahoo.fr

... et ses conséquences

- Le Web a besoin d'un nouveau protocole
  - Multi-langages, multi-plateformes
  - Respectant les formats d'échanges du Web
    - Réponses et requêtes en XML (Extensible Markup Language)
  - Facile à implémenter sur différents protocoles de transport
    - RPC (remote procedure call) est un protocole réseau permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications.
    - HTTP
    - ou autre MOM (Message-oriented middleware) désigne une architecture logicielle basée sur des composants (les middleware) qui permettent l'échange de messages entre applications réparties sur un réseau
  - Permettant de franchir les « firewalls »
    - ATTENTION : on perd le contrôle d'accès à faible granularité
  - Avec une spécification non propriétaire garantie par un organisme indépendant
    - W3C
- La réponse : SOAP (Simple Object Access Protocol)

Faîçal Felhi - felhi_fayssal@yahoo.fr

La philosophie S.O.A.P

- SOAP codifie simplement une pratique existante
  - Utilisation conjointe de XML et HTTP
- SOAP est un protocole **minimal** pour appeler des méthodes sur des serveurs, services, composants, objets
  - Ne pas imposer une API ou un runtime
  - Ne pas imposer l'utilisation d'un ORB (CORBA, DCOM, …) ou d'un serveur web particulier (Apache, IIS, …)
  - Ne pas imposer un modèle de programmation
    - Plusieurs modèles peuvent être utilisés conjointement
  - Et "ne pas réinventer une nouvelle technologie"
- SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies
  - Vous pouvez écrire votre 1er appel SOAP en moins d'une heure !!
  - Il vous a fallu combien de temps en CORBA, RMI, DCOM ?

Faîçal Felhi - felhi_fayssal@yahoo.fr

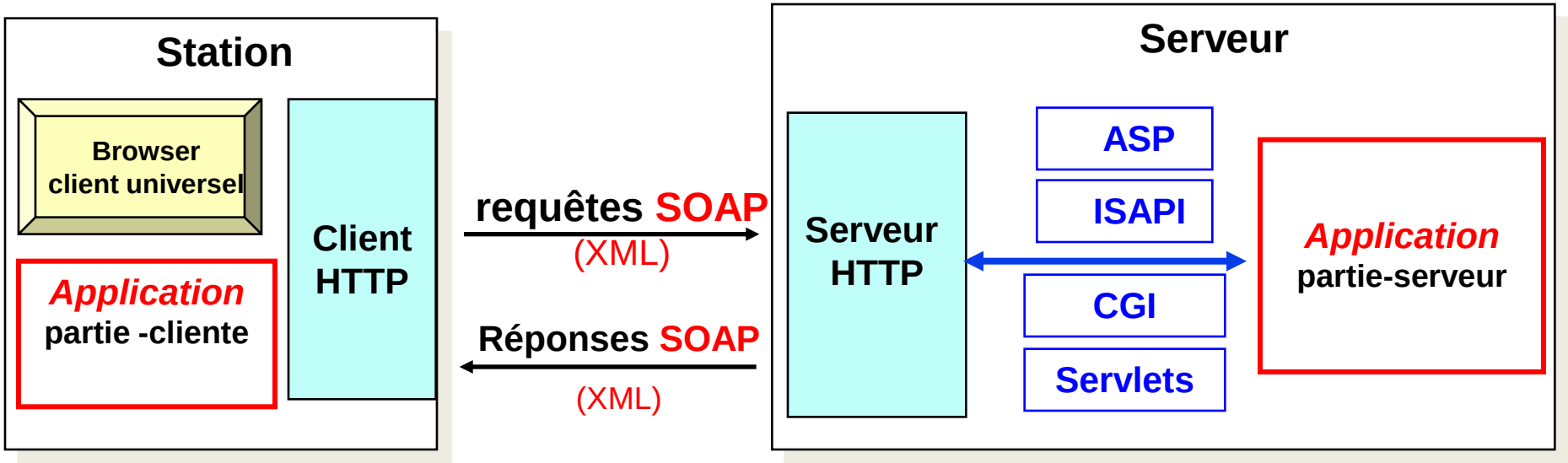
Les 3 aspects d'un appel SOAP

- SOAP peut être vu comme un autre RPC Objets
  - Les requêtes contiennent les paramètres IN et INOUT
  - Les réponses contiennent les paramètres INOUT et OUT
- SOAP peut être vu comme un protocole d'échange de "message"
  - La requête contient un seul message (appel sérialisé d'une méthode sur un objet)
  - La réponse contient un seul message (retour sérialisé d'un appel de méthode sur un objet)
- SOAP peut être vu comme un format d'échange de documents
  - La requête contient un document XML
  - Le serveur retourne une version transformée
- Ces vues ne sont pas imposées par le protocole

Faîçal Felhi - felhi_fayssal@yahoo.fr

En résumé

- SOAP = HTTP + XML



Faîçal Felhi - felhi_fayssal@yahoo.fr

Pourquoi utiliser HTTP ?

- HTTP (HyperText Transfer Protocol) est devenu de facto le protocole de communication de l'Internet
- HTTP est disponible sur ***toutes*** les plates-formes – très rapidement
- HTTP est un protocole simple, qui ne requière que peu de support pour fonctionner correctement
- HTTP est un protocole sans connexion
  - Peu de paquets sont nécessaires pour échanger des informations
- HTTP offre un niveau de sécurité simple et effectif
- HTTP est le seul protocole utilisable à travers des pare-feu

Faîçal Felhi - felhi_fayssal@yahoo.fr

Fonctionnement d'HTTP

- HTTP utilise un protocole requête/réponse basé sur du texte
- La première ligne de la requête contient 3 éléments
  - Verbe : POST/GET/HEAD
  - URI : /default.htm
  - Protocole : HTTP/1.0 - HTTP/1.1
- La première ligne de la réponse contient 2 éléments
  - État : 200, 402
  - Phrase : OK, Unauthorized
- Les lignes suivantes contiennent un nombre arbitraire d'entête
- Le "contenu" suit une ligne d'entête vide
  - Utilisé essentiellement pour les réponses et pour les requêtes POST

Faîçal Felhi - felhi_fayssal@yahoo.fr

Fonctionnement d'HTTP

HTTP Request

```
GET /bar/foo.txt HTTP/1.1
```

ou

```
POST /bar/foo.cgi HTTP/1.1
Content-Type: text/plain
Content-Length: 13

Goodbye, World
```

HTTP Response

```
200 OK
Content-Type: text/plain
Content-Length: 12

Hello, World
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Pourquoi utiliser XML (Extensible Markup Language) ?

- Utilise du texte (peut être lu et écrit directement)
  - ATTENTION : le texte est globalement peut lisible et vite complexe pour un humain
- Construire correctement du texte XML est simple
  - Pas d'éléments qui se recouvrent (uniquement des imbrications)
  - Les attributs sont clairement identifiés (dir="in")
  - Les caractères "<", ">", "&" doivent être précédés d'un caractère d'échappement (ou il faut utiliser CDATA)
- XML est aujourd'hui adopté par tous les acteurs de l'Internet: plates-formes, éditeurs, …

Faîçal Felhi - felhi_fayssal@yahoo.fr

Pourquoi utiliser XML (Extensible Markup Language) ?

- XML permet une extensibilité aisée par l'utilisation d'espaces de nommage (namespaces et URIs)
- XML permet d'ajouter du typage et de la structure à des informations
  - L'information peut être sauvegardée n'importe où sur le Net
  - Les données fournies par de multiples sources peuvent être agrégées en une seule unité
  - Chaque partie à sa propre structure XML
  - Chaque partie peut définir des types spécifiques
- W3C n'impose pas un API mais en recommande un (DOM: Document Object Model)
  - D'autres sont utilisés : SAX(Simple **API** for XML), strcat

Faîçal Felhi - felhi_fayssal@yahoo.fr

Exemple de requête utilisant HTTP

- **Demande de cotation à un serveur**

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
     "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
     "http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Exemple de réponse utilisant HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
      <m:GetLastTradePriceResponse
                xmlns:m="Some-URI">
          <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

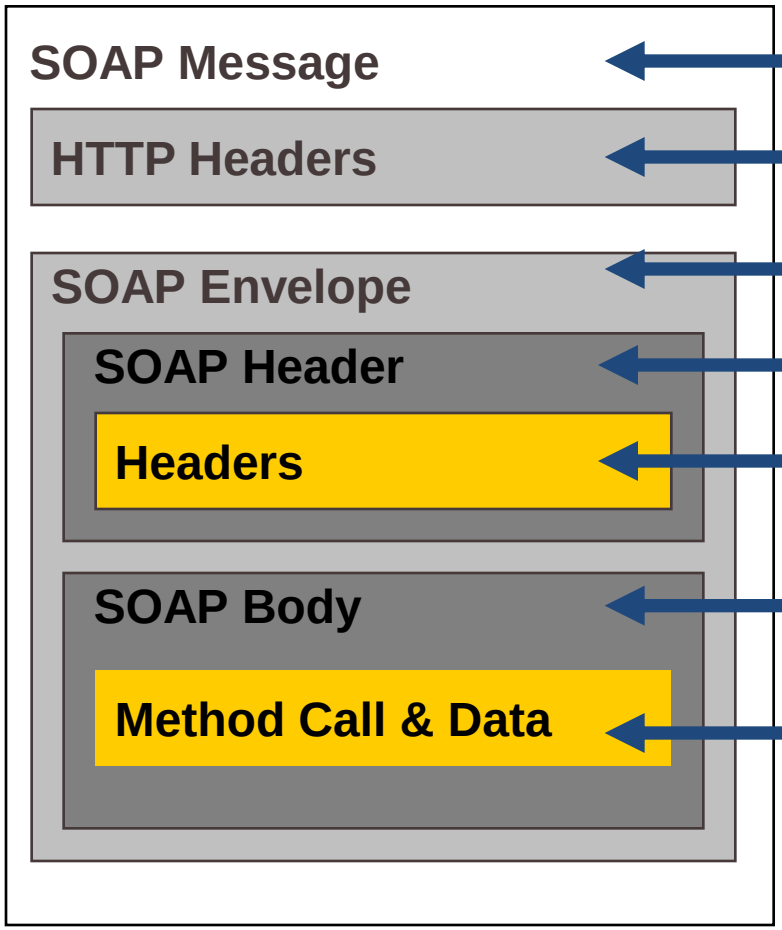

Faîçal Felhi - felhi_fayssal@yahoo.fr

Eléments de SOAP

- L'enveloppe (enveloppe)
  - Définit la structure du message
- Les règles d'encodage (encoding rules)
  - Définit le mécanisme de sérialisation permettant de construire le message pour chacun des types de données pouvant être échangés
- Fonctionnement en modèle client / serveur (RPC representation)
  - Définit comment sont représentés les appels de procédure et les réponses
- Proposer une mise en œuvre sur HTTP (HTTP Extension Framework)
  - RFC 2774
  - Définir l'échange de message SOAP sur HTTP

Faîçal Felhi - felhi_fayssal@yahoo.fr

SOAP Message Structure



Le message SOAP Complet

Entête standard HTTP
et entête SOAP HTTP

Enveloppe

Entête

Entête individuelle

Corps qui contient les
appels de méthodes SOAP

Appel de méthode et description
en XML de données

Faîçal Felhi - felhi_fayssal@yahoo.fr

Modèle de message

- SOAP permet une communication par message
  - d'un expéditeur vers un récepteur
- Structure d'un message
  - Enveloppe / Envelope
    - Élément racine
    - Namespace :
      `SOAP-ENV`
      `http://schemas.xmlsoap.org/soap/envelope/`
  - Entête / Header
    - Élément optionnel
    - Contient des entrées non applicatives
      - Transactions, sessions, …
  - Corps / Body
    - Contient les entrées du message
      - Nom d'une procédure, valeurs des paramètres, valeur de retour
    - Peut contenir les éléments « fault » (erreurs)

Faîçal Felhi - felhi_fayssal@yahoo.fr

Entête d'un Message

- Contient des entrées non applicatives
  - Transactions, sessions, …
- L'attribut mustUnderstand
  - Si absent ou = 0
    - l'élément est optionnel pour l'application réceptrice
  - Si =1
    - l'élément doit être compris de l'application réceptrice
    - Si ce n'est pas le cas, le traitement du message par le récepteur doit échouer
- Exemple

```
<SOAP-ENV:Header>
   <t:Transaction xmlns:t="some-URI"
                  SOAP-ENV:mustUnderstand="1">
          5
   </t:Transaction>
</SOAP-ENV:Header>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Corps d'un Message

- Contient des entrées applicatives
- Encodage des entrées
- Namespace pour l'encodage
  - **`SOAP-ENC`**
    `http://schemas.xmlsoap.org/soap/encoding/`
  - xsd : XML Schema

Faîçal Felhi - felhi_fayssal@yahoo.fr

SOAP sur HTTP

- Utilise le modèle POST Requête/Réponse
- Requête
  - Type MIME : `text/xml`
  - Champs d'entête supplémentaire de la requête
    - SOAPAction : URI

```
SOAPAction: "http://electrocommerce.org/abc#MyMessage"
SOAPAction: "myapp.sdl"
SOAPAction: ""
SOAPAction:
```

  - Envelope SOAP
- Réponse
  - Status
    - 2xx : le récepteur a correctement reçu, compris et accepté le message inclus
    - 500 (Internal Server Error): le récepteur n'accepte pas le message
  - Envelope SOAP
    - La réponse
    - Le détail des erreurs

Faîçal Felhi - felhi_fayssal@yahoo.fr

Un exemple d'échange

```
POST /path/foo.pl HTTP/1.1
Content-Type: text/xml
SOAPAction: interfaceURI#Add
Content-Length: nnnn

<soap:Envelope xmlns:soap='uri for soap'>
  <soap:Body>
    <Add xmlns='interfaceURI'>
      <arg1>24</arg1>
      <arg2>53.2</arg2>
    </Add>
  </soap:Body>
</soap:Envelope>
```

```
200 OK
Content-Type: text/xml
Content-Length: nnnn

<soap:Envelope
    xmlns:soap='uri for soap'>
  <soap:Body>
    <AddResponse xmlns='interfaceURI' >
      <sum>77.2</sum>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Types de message SOAP

- SOAP définit trois types de message
  - Appel (Call) - obligatoire
  - Réponse (Response) - optionnel
  - Erreur (Fault) - optionnel

Faîçal Felhi - felhi_fayssal@yahoo.fr

Appel simple

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml
Content-Length: nnnn
SOAPMethodName: Some-Namespace-URI#GetLastTradePrice

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-
    org:soap.v1”>
    <SOAP:Body>
        <m:GetLastTradePrice
          xmlns:m="Some-Namespace-URI”>
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP:Body>
</SOAP:Envelope>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Réponse

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1”>
    <SOAP:Body>
        <m:GetLastTradePriceResponse
          xmlns:m="Some-Namespace-URI”>
            <return>34.5</return>
        </m:GetLastTradePriceResponse>
    </SOAP:Body>
</SOAP:Envelope>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Erreur

```
<SOAP:Envelope
  xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1>
  <SOAP:Body>
    <SOAP:Fault>
      <faultcode>200</faultcode>
      <faultstring>
        SOAP Must Understand Error
      </faultstring>
      <runcode>1</runcode>
    </SOAP:Fault>
   <SOAP:Body>
</SOAP:Envelope>
```


Faîçal Felhi - felhi_fayssal@yahoo.fr

Sécurité

- Basé sur la sécurité dans http
  - HTTPS
  - Certificats X.509
- Les Firewalls peuvent filtrer les messages facilement
- Pas de transfert de code applicatif
  - Uniquement des données
- Chaque développeur, choisi de rendre visible telle ou telle méthode
- Les paramètres sont typés lors du transport

Faîçal Felhi - felhi_fayssal@yahoo.fr

Portée de SOAP

- SOAP est simple et extensible
  - Il permet de réaliser des appels de méthode sur le Web
  - Indépendant des OS, des modèles objets, des langages
  - Transport des messages par HTTP + XML sur le fil
  - Fonctionne avec l'infrastructure Internet existante
  - Permet l'interopérabilité entre OS, langages et modèles objets
- Ce n'est pas un système réparti à objets
  Il ne couvre donc pas les fonctions suivantes :
  - Pas de ramassage des miettes (fragments)
  - Pas de contrôle de types, pas de gestion de version
  - Pas de dialogue entre deux serveurs HTTP
  - Pas de passage d'objets par référence
    - Nécessite ramassage des miettes en réparti et HTTP bi-directionnel

Faîçal Felhi - felhi_fayssal@yahoo.fr

Autres Extensions

- Transport
  - SOAP sur SMTP/FTP/POP3/IMAP4/RMI-IIOP
    - Voir implémentation IBM/Apache
  - SOAP sur MOM (JMS)
- Encodage
  - XMI (UML)
    - Voir implémentation IBM/Apache
  - Litteral XML
    - DOM org.w3c.dom.Element sérialisé
    - Voir implémentation IBM/Apache

Faîçal Felhi - felhi_fayssal@yahoo.fr

Implémentations de SOAP

Voir http://www.Soapware.org

- JAXM, JAX-RPC RI
- Apache SOAP 2.2
- Apache AXIS
- SoapRMI
- SOAPDirect
- InstantXML
- .Net
- ...

Faîçal Felhi - felhi_fayssal@yahoo.fr