



Java ME

Java ME Web Services API



Java ME Web Services API (WSA 1.0)

♦ Web Services API

- ♦ Desenvolupat per la comunitat Java. **JSR 172**
- ♦ API opcional de Java ME per serveis web
- ♦ Té 2 parts
 - **Invocació remota de serveis:** accés a serveis web remots. Subconjunt restringit de JAX-RPC 1.1
 - **XML Parsing:** tractament de fitxer XML. Subconjunt restringit de JAXP amb suport per a SAX 2
- ♦ Dissenyat per treballar amb Connected Device Configuration (CDC) or Connected Limited Device Configuration (CLDC 1.0 o 1.1)



Serveis web

Segons el W3C, un Servei Web és un programari dissenyat per ser interoperable entre màquines a través d'una xarxa (P. ex. Internet).

- ♦ Ha de tenir una interfície que sigui descrita en un format processable per màquines (P. ex. WSDL).
- ♦ Els sistemes interactuen amb els SW segons la seva interfície utilitzant missatges (SOAP Envelopes) o segons REST.
- ♦ La clau és la interoperabilitat (un client PHP, Python, corrent en una plataforma Windows utilitzant un servei web Java en plataforma Linux...).
- ♦ Està pensat per utilitzar estàndards madurs com XML i HTTP.



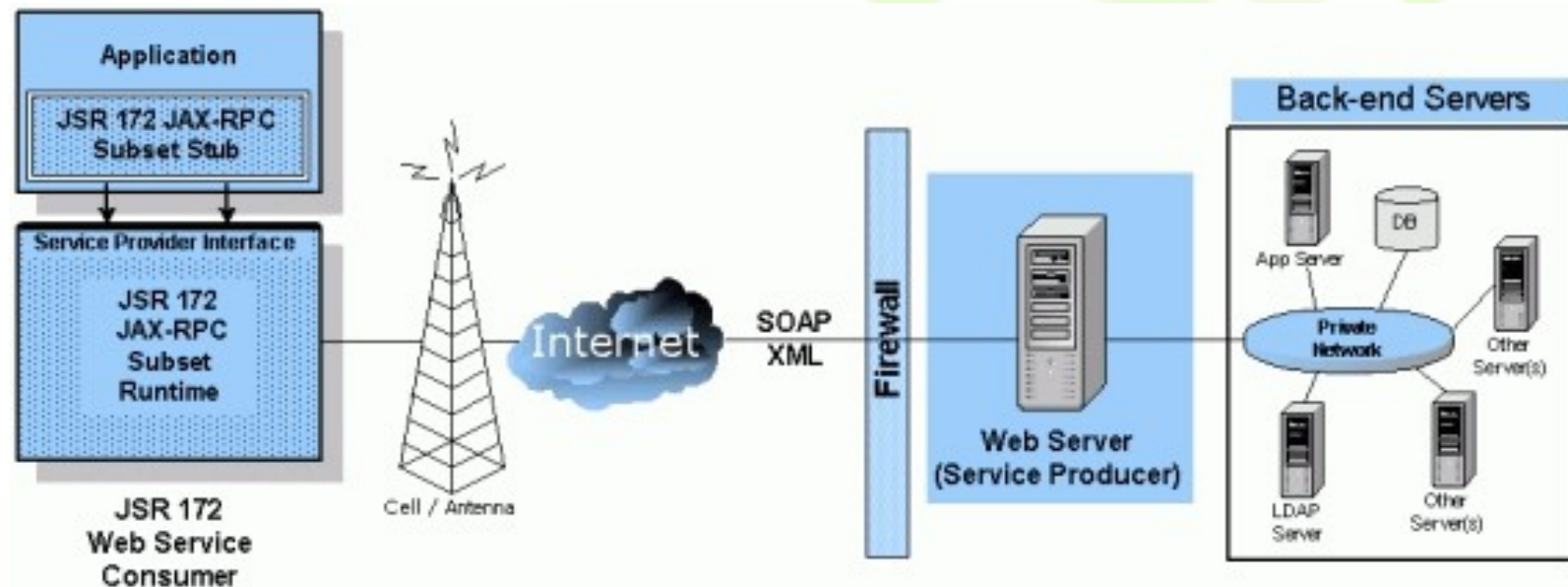
Qui esta al darrera?

- ♦ **Els principals responsables de l'arquitectura i la estandardització dels serveis web són:**
 - ♦ **W3C**
 - Web Services Activity
 - ♦ **OASIS**
 - <http://www.oasis-open.org/>
 - ♦ **WS-I**
 - <http://www.ws-i.org/>



Arquitectura Serveis Web

♦ Arquitectura





Analitzadors Sintàctics (Parsers)

♦ Parse

- ♦ “Parsejar” és analitzar sintàcticament.
- ♦ En informàtica, "parsejar" és el procés d'analitzar una seqüència d'entrada per tal de determinar la seva estructura gramatical respecte a una gramàtica formal específica (en el nostre cas XML).
- ♦ Un Analitzador Sintàctic o "Parser" és un programa informàtic encarregat d'aquesta tasca.
- ♦ A Java analitzar sintàcticament representa transformar documents XML a una estructura d'arbre o classes Java.



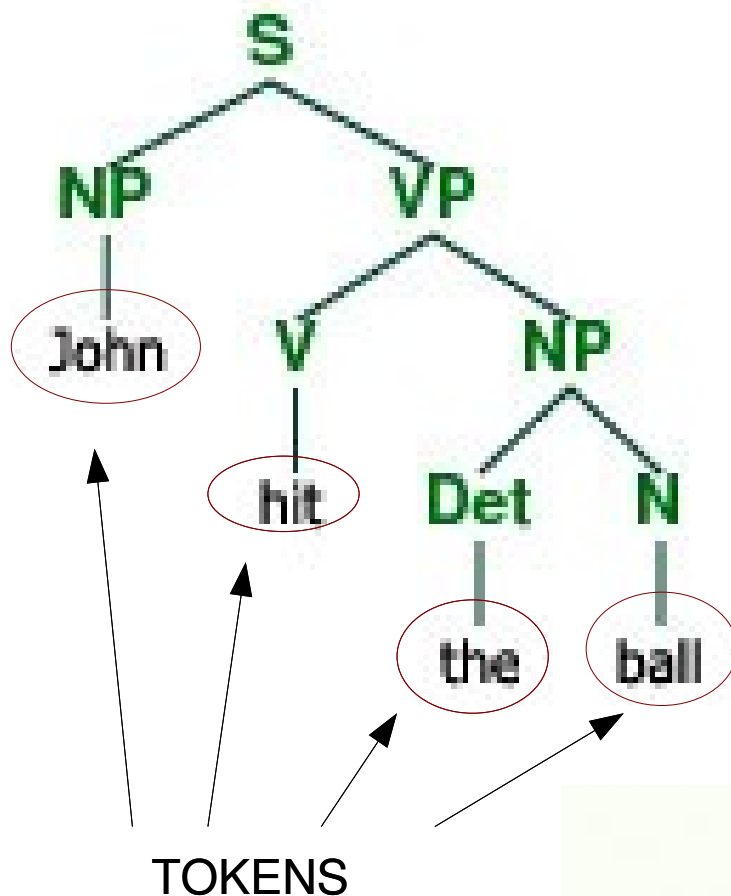
Analitzadors Sintàctics (Parsers)

- ♦ **L'anàlisi sintàctic té 2 fases:**
 - ♦ Identificar les unitat mínimes de significat (tokens).
 - ♦ Construir l'arbre sintàctic que relaciona els tokens.
- ♦ **L'analogia és clara amb l'anàlisi sintàctic en llengua.**
 - ♦ En XML els nodes representen els tokens i l'arbre sintàctic és l'estructura XPath d'un document XML

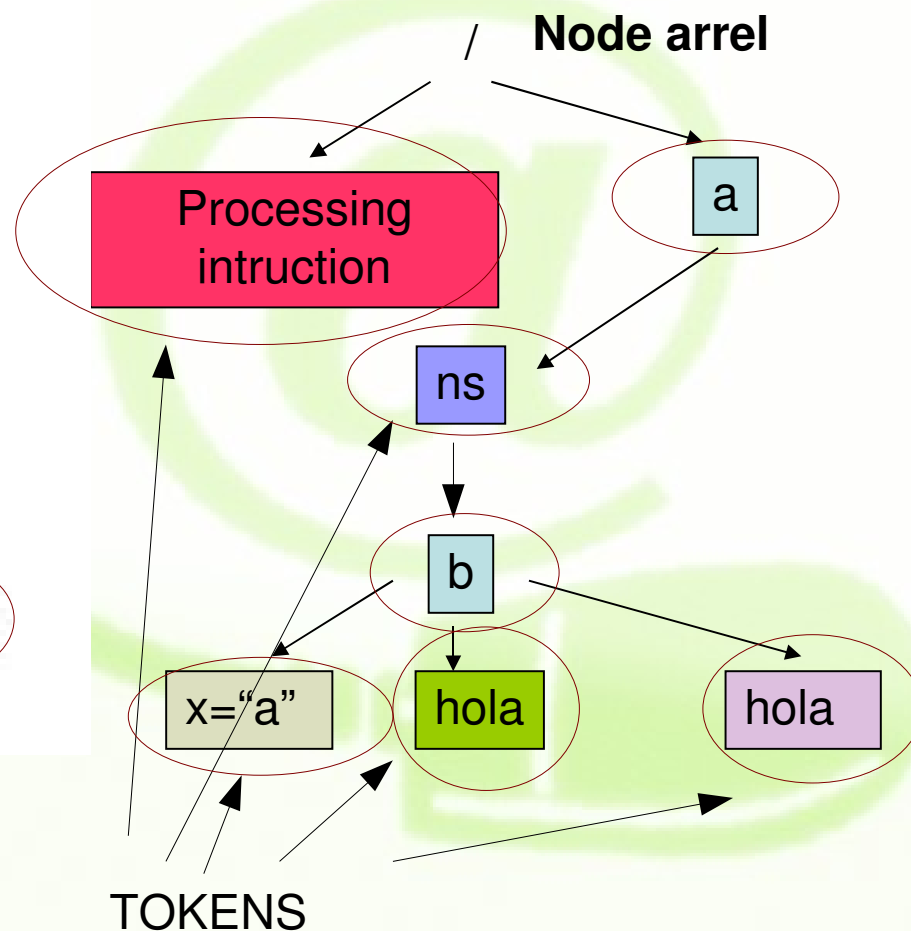


Analitzadors Sintàctics (Parsers)

Anàlisi sintàctic en Llengua



Anàlisi sintàctic en XML









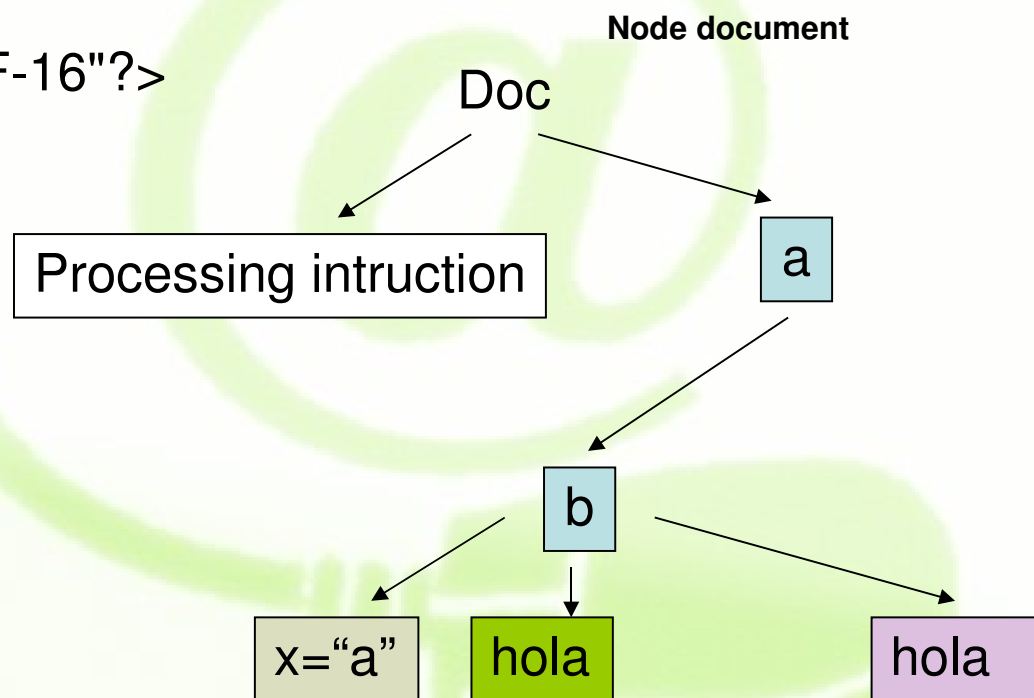
Estructura d'un XML

♦ Un document XML té una estructura d'arbre:

```
<?xml version="1.0" encoding="UTF-16"?>
```

```
<a>
  <b x="a">
    <!-- hola -->
    hola
  </b>
</a>
```

-  Node tipus element
-  Node tipus text
-  Node tipus comentari
-  Node tipus atribut





APIs Java per a l'anàlisi sintàctic

- ▶ **Podem distingir 2 tipus principals d'API:**
 - ▶ **Tree-Based.** Es crea un arbre en memòria amb tota l'estructura del document XML. L'especificació més important és **DOM** del W3C.
 - ▶ **Event-Based.** El document es processa mitjançant esdeveniments amb accés seqüencial. L'especificació més important és **SAX** del grup XML-DEV.

Java ME només suporta SAX. Més adequada a les restriccions de memòria dels dispositius mòbils



SAX vs DOM

DOM

- L'arbre es carrega **complet en memòria.**
- Més **lent.**
- Orientat a documents (Pull)
- Documents petits/mitjans.
- Menys línies de codi.
- L'estructura és important.
- Manipulació dels nodes segons el context.

SAX

- Menys consum de recursos de memòria.
- Més ràpid.
- Orientat a dades (Push)
- Documents grans.
- Més línies de codi.
- Només les dades són importants.
- Manipulació seqüencial



SAX

♦ Simple API for XML (SAX)

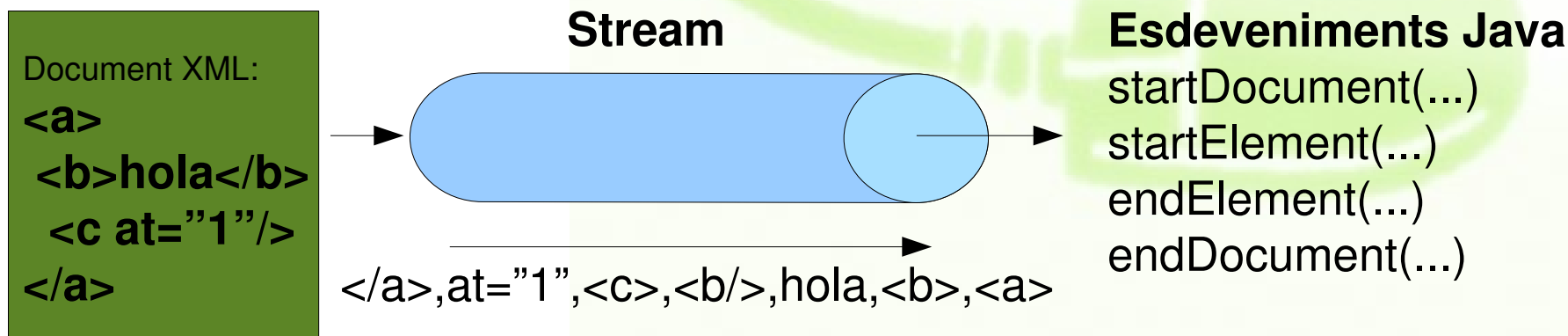
- ♦ Estàndard “de facto” per parsers Java basats en **processament per esdeveniments**.
- ♦ Java ME treballa amb la versió 2.0.
- ♦ Desenvolupat originalment de forma col·laborativa pel grup de treball XML-DEV.
- ♦ Podem trobar altres implementacions en altres llenguatges (Python, Perl, etc.).
 - <http://www.saxproject.org/>



Esdeveniments SAX

♦ Accés seqüencial unidireccional.

- ♦ Analogia amb l'accés seqüencial a fitxers.
- ♦ Sense informació d'estat. No es pot accedir a elements anteriors o posteriors (no hi ha memòria).
- ♦ Poc ús de recursos de memòria.





JAXP

♦ JAXP (Java API for XML processing)

- ♦ Proveeix de capacitat a Java per analitzar sintàcticament, validar i transformar documents XML.
- ♦ Java ME no suporta tot JAXP:
 - Suporta SAX 2.0 (1.0 NO)
 - Suporta espais de noms XML.
 - No suporta DOM
 - No suporta fulls de transformació XSLT
 - Suporta DTD però no XML Schema
- ♦ Paquets

```
javax.xml.parsers  
org.xml.sax  
org.xml.sax.helpers
```



SAX Parsing

♦ Creació d'objectes Parser

```
SAXParserFactory factory = SAXParserFactory.newInstance();  
SAXParser saxParser = factory.newSAXParser();
```

♦ El mètode més important de SAXParser és **parse()**

```
parse(InputSource is, DefaultHandler dh)  
parse(java.io.InputStream is, DefaultHandler dh)
```

• 2 paràmetres

- **InputSource|InputStream:** Origen de dades amb el document XML a “parsejar”
- **Handler:** Classe que especifica com s'ha de “parsejar” el document



SAX Parsing. Exemple

```
<grup nom="1r ESI Tarda">
  <alumne>
    <nom>Pere</nom>
    <cognoms>Abad Abellà</cognoms>
    <foto>http://localhost:8084/GestioAulesServlet/no_avatar.gif</foto>
  </alumne>
  <alumne>
    <nom>Pau</nom>
    <cognoms>Ferret Amela</cognoms>
    <foto>http://localhost:8084/GestioAulesServlet/no_avatar.gif</foto>
  </alumne>
  <alumne>
    <nom>Joan</nom>
    <cognoms>Coll Català</cognoms>
    <foto>http://localhost:8084/GestioAulesServlet/no_avatar.gif</foto>
  </alumne>
  <alumne>
    <nom>Julia</nom>
    <cognoms>Mauri Soler</cognoms>
    <foto>http://localhost:8084/GestioAulesServlet/no_avatar.gif</foto>
  </alumne>
</grup>
```




SAX Parsing. Exemple

♦ Handler

```
public class GrupHandler extends DefaultHandler {  
    ...  
    public void startElement(String uri, String localName, String qName, Attributes attributes)  
        throws SAXException {  
        if ("grup".equals(qName)) {  
            stack.push(new Grup());  
            String nom= attributes.getValue("nom");  
            Grup grup = ((Grup) stack.peek());  
            grup.setNom(nom);  
        } else if ("alumne".equals(qName)) {  
            stack.push(new Alumne());  
        } else if ("nom".equals(qName) || "cognoms".equals(qName) ||  
            "foto".equals(qName)) {  
            stack.push(new StringBuffer());  
            isStackReadyForText = true;  
        }  
    }  
}
```



SAX Parsing. Exemple

♦ Processament de text

```
public void characters(char[] ch, int start, int length) throws SAXException {  
  
    // if stack is not ready, data is not content of recognized element  
    if (isStackReadyForText == true) {  
        ((StringBuffer) stack.peek()).append(ch, start, length);  
    } else {  
        // read data which is not part of recognized element  
    }  
}
```



SAX Parsing. Exemple

◆ EndElement

```
public void endElement(String uri, String localName, String qName) throws SAXException {
    StackReadyForText = false;
    Object tmp = stack.pop();
    if (qName.equals("grup")) {
        grup = (Grup) tmp;
    } else if (qName.equals("alumne")) {
        ((Grup) stack.peek()).getAlumnes().addElement((Alumne) tmp);
    } else if (qName.equals("nom")) {
        ((Alumne) stack.peek()).setNom( tmp.toString() );
    } else if (qName.equals("cognoms")) {
        ((Alumne) stack.peek()).setCognoms(tmp.toString() );
    } else if (qName.equals("foto")) {
        ((Alumne) stack.peek()).setFotoURL(tmp.toString() );
    } else {
        stack.push(tmp);
    }
}
```



La pila de marshalling

♦ Mecanisme de pila:

- ♦ **Push:** Col·loca un objecte a la pila.
- ♦ **Pop:** Treu un objecte de la pila.

♦ StartElement

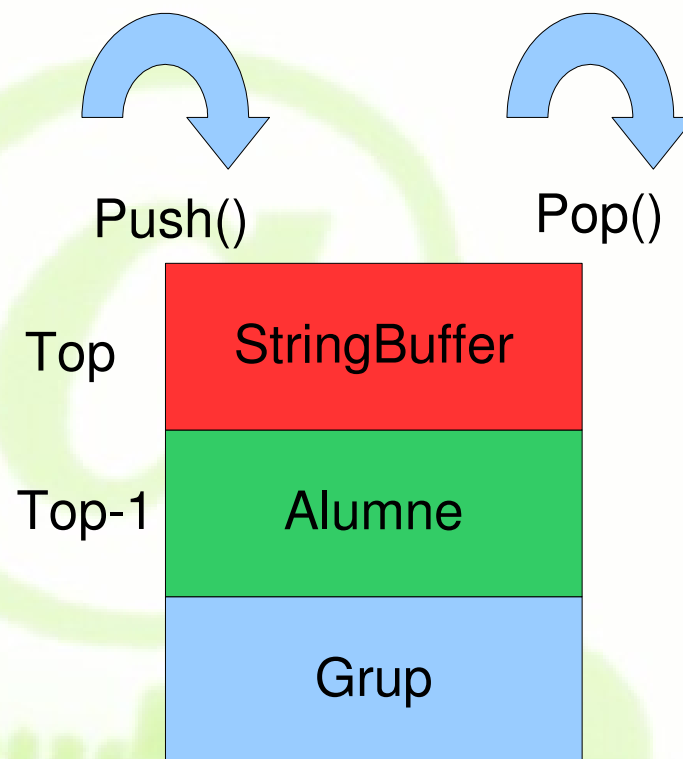
- ♦ Cada cop que comença un element nou, es crea l'objecte equivalent i es col·loca (push) a la pila

♦ EndElement

- ♦ Es treu l'element amb un pop() i s'envia a l'element pare amb un peek()

♦ Funció Characters

- ♦ El nodes simples (o que només tenen text) com **nom**, **cognoms** o **foto** són Objectes StringBuffer
- ♦ S'omple l'objecte amb el valor del text del node simple





Commons-Digester

- ◆ Creat per Apache per donar suport als seus projectes quan necessitaven llegir fitxers de configuració en XML.
- ◆ Permet crear un mapa que transformi documents XML en objectes Java.
- ◆ La idea és crear una sèrie de normes (rules) que funcionen com esdeveniments que s'executen al complir-se una condició o patró (molt similar a XSLT) of nested XML elements is recognized.
- ◆ Commons-Digester proporciona un conjunt de normes per defecte i la possibilitat de crear normes pròpies.
 - **Web:** <http://jakarta.apache.org/commons/digester/>
 - **Javadocs:**
<http://jakarta.apache.org/commons/digester/commons-digester-1.7/docs/api/>



Commons-Digester Exemple

- ♦ El següent exemple transforma aquest document XML en objectes Java (Address.java, AddressBook.java i Person.java)

```
<address-book>
  <person id="1" category="acquaintance">
    <name>Gonzo</name>
    <email type="business">gonzo@muppets.com</email>
    <address>
      <type>home</type>
      <street>123 Maine Ave.</street>
      <city>Las Vegas</city>
      <state>NV</state>
      <zip>01234</zip>
      <country>USA</country>
    </address>
  </person>

  <person id="2" category="rolemodel">
    <name>Kermit</name>
    <email type="business">kermit@muppets.com</email>
    <email type="home">kermie@acme.com</email>
    <address>
      <type>business</type>
      <street>987 Brown Rd</street>
      <city>Las Cruces</city>
      <state>NM</state>
      <zip>75321</zip>
      <country>USA</country>
    </address>
  </person>
</address-book>
```



Commons-Digester Exemple

♦ Fitxer de normes:

```
<!DOCTYPE digester-rules
PUBLIC "-//Jakarta Apache //DTD digester-rules XML V1.0//EN"
"http://jakarta.apache.org/commons/digester/dtds/digester-rules.dtd">

<digester-rules>
  <pattern value="address-book">
    <pattern value="person">
      <object-create-rule classname="Person"/>
      <set-properties-rule/>
      <set-next-rule methodname="addPerson"/>
      <pattern value="name">
        <call-method-rule methodname="setName" paramcount="0"/>
      </pattern>
      <pattern value="email">
        <call-method-rule methodname="addEmail" paramcount="2"/>
        <call-param-rule paramnumber='0' attrname='type' />
        <call-param-rule paramnumber='1' />
      </pattern>
      <pattern value="address">
        <object-create-rule classname="Address"/>
        <set-nested-properties-rule/>
        <set-next-rule methodname="addAddress"/>
      </pattern>
    </pattern>
  </pattern>
</digester-rules>
```





La pila de protocols

- ♦ Els protocols en que es basen el SW es mostren en el següent gràfic:

| | |
|-----------------------|-------------|
| UDDI | Phone book |
| WSDL | Contract |
| SOAP | Envelope |
| HTTP, SMTP, FTP | Mailman |
| TCP/IP, UDP | Post office |
| Programming: DOM, SAX | Speech |
| Schema: DTD, XSD | Vocabulary |
| XML 1.0 | Alphabet |



La pila de protocols

♦ Servei de transport:

- ♦ És l'encarregat de transportar els missatges entre servei i client a través de la xarxa. Exemples: **HTTP**, **SMTP**, **FTP**, **BEEP**.

♦ Missatgeria XML:

- ♦ És responsable de codificar els missatges en un format XML que pugui ser entès tant per client com servei. Exemples: **XML-RPC**, **SOAP** i **REST**.

♦ Descripció del servei:

- ♦ S'utilitza per descriure la interfície pública dels serveis web. Normalment s'utilitza **WSDL**.

♦ Descobriment de serveis:

- ♦ Centralitza serveis web en un registre comú de tal manera que els serveis web puguin **publicar** la seva localització i descripció, d'aquesta manera els possibles clients poden **descobrir** de forma senzilla quins serveis es troben disponibles a la xarxa. Actualment s'utilitza **UDDI**.



Arquitectura Serveis Web

♦ Resumint, en un servei web intervenen tres actors principals:

- ♦ Proveïdor de serveis
- ♦ Client
- ♦ Agent

♦ Compra/venta d'un pis

- ♦ Es pot establir un símil amb la compra/venta d'un pis (comprador, venedor i agent immobiliari).





Missatgeria. XML-RPC, SOAP i REST

Messaging. XML-RPC, SOAP & REST





XML-RPC

- ♦ **XML-RPC és un protocol de crida remota de procediments basat en XML.HTTP és el mecanisme de transport.**
 - ♦ És un protocol molt simple que defineix una sèrie de tipus de dades i comandes (l'especificació completa es pot escriure en un parell de pàgines).
 - ♦ El seu principal avantatge és precisament la seva simplicitat comparada amb altres protocols famosos distribuïts per ser molt complexes (CORBA, RMI, IIOP o SOAP mateix).
 - ♦ Creat per Dave Winer l'any 1995 sota els auspicis de Microsoft. Microsoft el va considerar massa simple i va començar a afegir funcionalitats. A partir d'aquí va deixar de ser tan simple i va ser els inicis del que és ara SOAP.

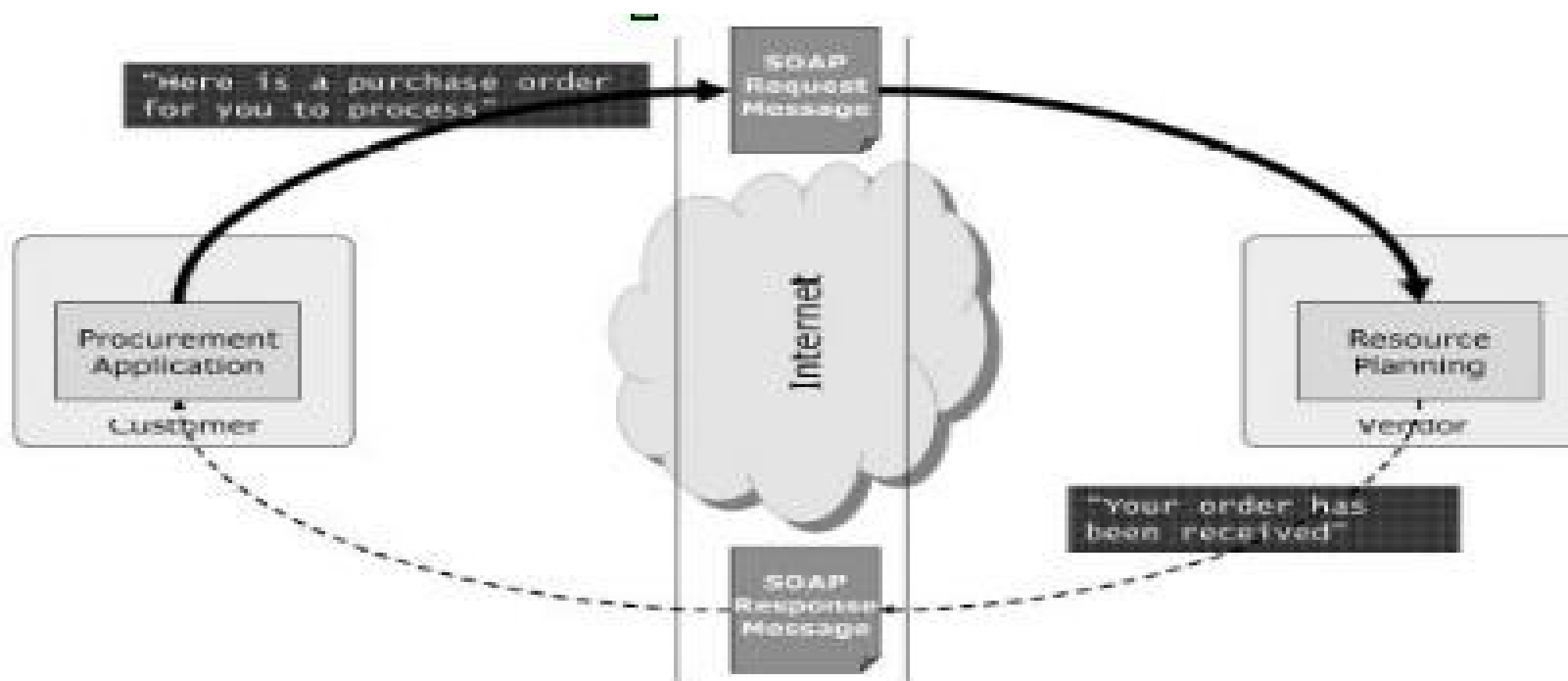


SOAP

- ♦ **SOAP és un protocol per l'intercanvi de missatges en XML a través d'una xarxa.**
 - ♦ Normalment utilitza el protocol **HTTP** però pot utilitzar altres com **SMTP** o **FTP**.
 - ♦ Existeixen diferents patrons de missatges que es poden utilitzar amb SOAP. El més famós és **Remote Procedure Call (RPC)**.
 - ♦ SOAP és la base de la pila de Serveis Web i proveeix a altres protocols superiors d'un marc per a l'intercanvi de missatges.
 - ♦ Segueix el patró: **Service-Oriented architectural(SOA)**.
 - ♦ SOAP a l'estar basat en un format de text (XML) es considera que és unes 10 vegades més lent que altres tecnologies distribuïdes com RMI o IIOP.
 - ♦ Les sigles volien dir: **Simple Object Access Protocol**.
 - ♦ Successor de **XML-RPC**.



Exemple SOAP. XML RPC

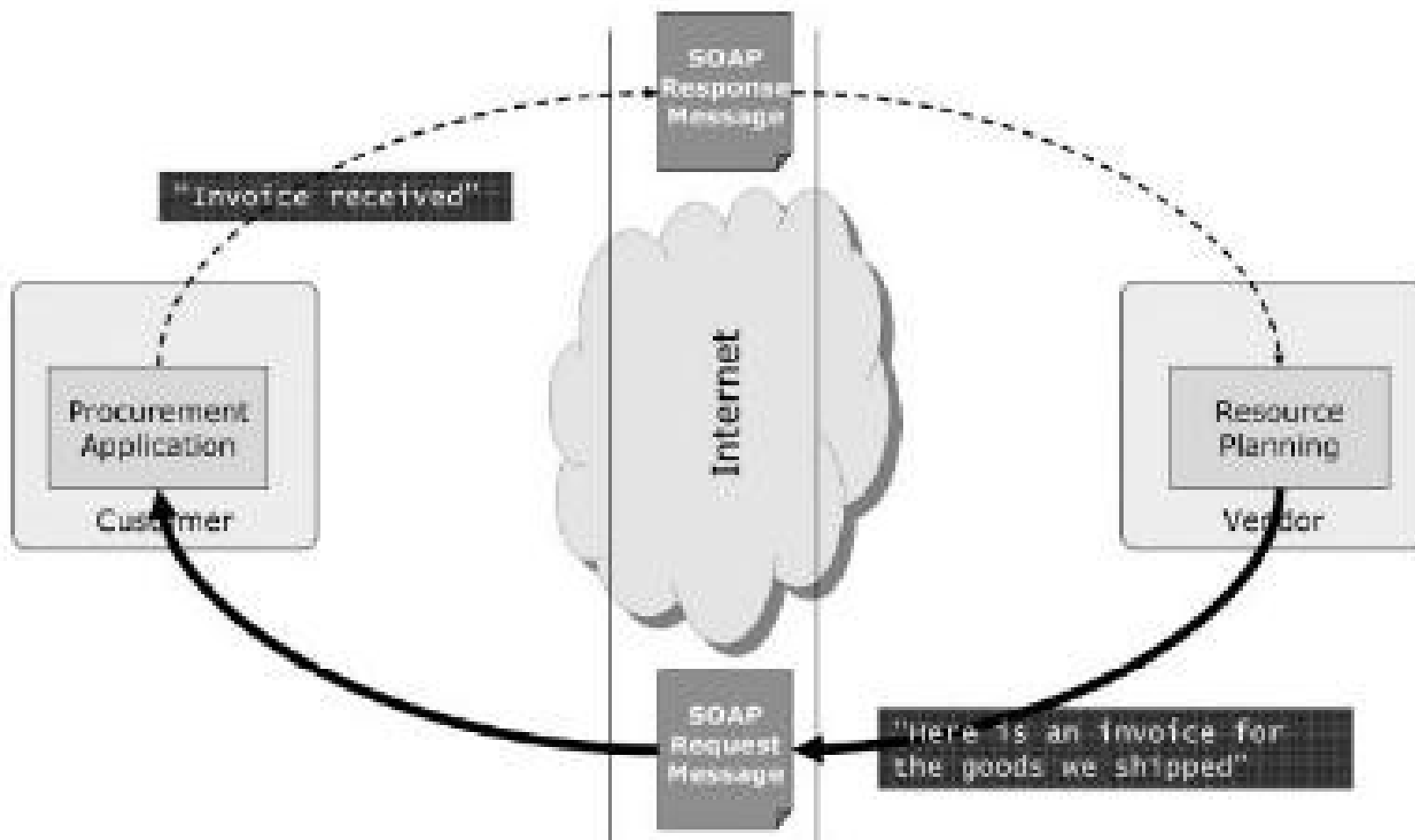


♦ Missatge SOAP de petició

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```



Exemple SOAP. XML RPC





♦ Paquets

`java.rmi`
`javax.microedition.xml.rpc`
`javax.xml.rpc`



Representational State Transfer (REST)

♦ REST

- ♦ És un estil d'arquitectura de programari per sistemes distribuïts hypermedia com la World Wide Web.
- ♦ L'origen del terme es deu a una dissertació doctoral de **Roy Fielding** a l'any 2000. Roy és un dels principals autors de l'especificació del protocol HTTP. Actualment està tenint una important expansió com alternativa a altres arquitectures distribuïdes com SOAP.
- ♦ Originalment REST era una col·lecció de consells per a desenvolupar arquitectures distribuïdes. Actualment s'està utilitzant en un sentit més ampli per descriure qualsevol interfície web simple que utilitzi **XML** i **HTTP** sense cap abstracció extra com SOAP. És el que s'anomena estil REST o l'aproximació REST.



Representational State Transfer (REST)

♦ XML+HTTP

- ♦ És possible escriure aplicacions basades en sense utilitzar **SOAP**. Implementen el model de crida remota de procediments (RPC) però no necessàriament han de ser RESTFul.
- ♦ Els sistemes que utilitzen l'aproximació REST són anomenades sovint **RESTFul**.
- ♦ Els defensors més acèrrims de l'arquitectura REST són anomenats **RESTafarians**.
- ♦ REST no és un estàndard, és un estil.

♦ Referència:

- ♦ http://en.wikipedia.org/wiki/Representational_State_Transfer



Exemple REST

- ♦ **Amazon proporciona serveis web per integrar en pàgines web.**
 - ♦ Podem obtenir informació de productes amb **Amazon ECS** (E-Commerce Services).
- ♦ **Les peticions són crides a una url base:**
 - ♦ **`http://webservices.amazon.com/onca/xml?Service=AWSECommerceService`**
- ♦ **Que es configura a través de paràmetres de Query String (Exemple):**
 - **`AWSAccessKeyId=[IdDelUsuari]`**
 - **`Operation=ItemSearch`**
 - **`SearchIndex=Books`**
 - **`Keywords=dog`**



Exemple REST

- ▶ **La URL és:**

- ▶ **Servei Web Amazon:**

&AWSAccessKeyId=03J88K5T8YE9J45J2682

&Operation=ItemSearch

&SearchIndex=Books

&Keywords=dog

- ▶ **El resultat és un XML amb un esquema definit per Amazon.**

- ▶ **Referències:**

- ▶ **Making REST request with AWS**

- ▶ **Exemple**



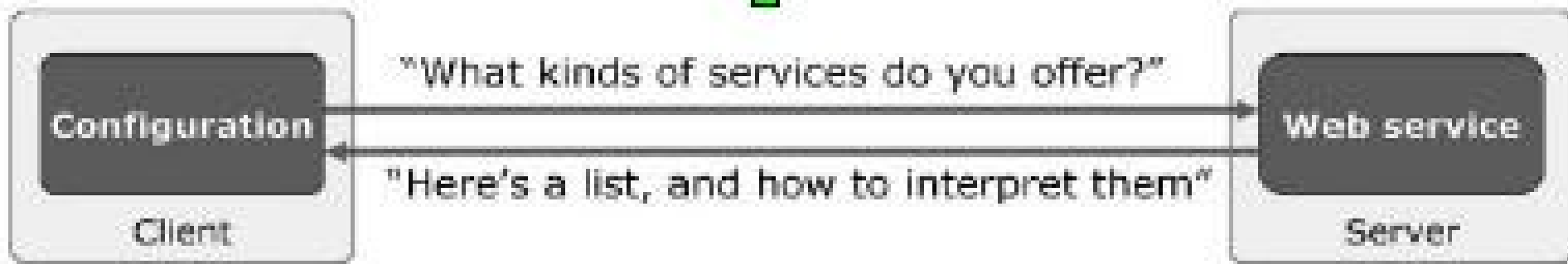
Web Services Description Language (WSDL)

- ♦ **WSDL és un estàndard XML utilitzat per descriure serveis web.**
 - ♦ WSDL descriu la interfície pública del servei web. La interfície descriu com s'ha d'interactuar amb el SW.
 - ♦ WSDL sovint s'utilitza juntament amb SOAP i XMLSchema per proveir Serveis Web per Internet, però també es pot utilitzar amb REST.
 - ♦ Els recursos i els serveis són oferts utilitzant WSDL.
 - ♦ Dos perfils: WS-I (WS-I Basic Profile) o WSRF framework.



Web Services Description Language (WSDL)

- ♦ **WSDL 1.1 no és una recomanació W3C.**
- ♦ **WSDL 2.0 és candidata a recomanació.**



- ♦ **Referència W3C:**
 - ♦ **<http://www.w3.org/TR/wsdl>**
 - ♦ **<http://www.w3.org/TR/wsdl20/>**



Reconeixement 3.0 Unported

Sou lliure de:



copiar, distribuir i comunicar públicament l'obra



fer-ne obres derivades

Amb les condicions següents:



Reconeixement. Heu de reconèixer els crèdits de l'obra de la manera especificada per l'autor o el llicenciador (però no d'una manera que suggereixi que us donen suport o rebeu suport per l'ús que feu l'obra).

- Quan reutilitzeu o distribuïu l'obra, heu de deixar ben clar els termes de la llicència de l'obra.
- Alguna d'aquestes condicions pot no aplicar-se si obteniu el permís del titular dels drets d'autor.
- No hi ha res en aquesta llicència que menyscabi o restringeixi els drets morals de l'autor.

Advertiment

Els drets derivats d'usos legítims o altres limitacions reconegudes per llei no queden afectats per l'anterior
Això és un resum fàcilment llegible del text legal (la llicència completa).

<http://creativecommons.org/licenses/by/3.0/deed.ca>