DWES06 – SERVEIS WEB

Tipus de serveis web

Segons el W3C hi ha dos grans tipus de serveis web:

• **Big Web services**: serveis que exposen operacions perquè les facin servir els servidors. Estan basats en el protocol SOAP.

• **Web API**: serveis basats en l'arquitectura REST -que basa el seu funcionament en els recursos-.

Big Web services

Els coneguts com a *big Web services* són serveis web que envien i reben la informació empaquetada en un format XML anomenat SOAP. Es poden definir a partir de tres funcions, tot i que no totes són estrictament necessàries per treballar:

• Una manera de definir i transmetre els missatges.
• Un llenguatge de definició de la manera com es poden fer servir els serveis.
• Un sistema de descobriment de serveis.

Format dels missatges

Per **definir el format dels missatges** que es transmeten es fa servir el protocol **SOAP**, que està basat en XML.

Per **transportar aquests missatges** se sol fer servir **HTTP** tot i que es poden fer servir altres protocols de transport com els de correu electrònic, FTP.

Definició del servei

Per poder fer servir un servei web que no s'ha fet servir mai i que fins i tot pot haver estat descobert automàticament és molt important disposar d'una manera de conèixer quins serveis ofereix i com s'ha de fer per usar-los.

La manera que tenen aquests serveis web de definir com s'ha de fer per poder usar el servei i quin serveis s'ofereixen és mitjançant un altre protocol anomenat **WSDL** (*Web services description language*), que també està basat en XML.

Descobriment de serveis

Si es pretén que els programes funcionin sols cal que tinguin alguna manera de poder descobrir serveis que no coneixien anteriorment de manera automàtica.

Es va definir **UDDI** (descripció, descoberta i integració universals o *universal description, discovery and integration*) per **proporcionar la capacitat de permetre als programes descobrir serveis web** que s'adeqüin a les necessitats que tinguin.

SOAP

SOAP va ser el primer estàndard de serveis web que es va desenvolupar i actualment és una recomanació del W3C. És un acrònim de *simple object access protocol* (protocol d'accés d'objecte simple), però actualment aquest nom ha perdut sentit ja que ha quedat demostrat que SOAP no és simple, i a més no té gaire a veure amb l'accés a objectes.

Han anat sorgint diferents versions de SOAP al llarg dels anys i actualment se'n fan servir dues, les versions 1.1 i 1.2, l'especificació de les quals es pot trobar a la pàgina del W3C (http://www.w3.org/TR/soap/).

És un entorn de missatges que permet la comunicació entre programes que corren en diferents servidors amb independència del llenguatge i del sistema operatiu en què funcionin.

Aconsegueix la independència del llenguatge i de la plataforma basant els missatges en XML i s'aprofita que l'èxit de la web ha fet que tots els sistemes tinguin implementat el protocol HTTP (tot i que pot funcionar en altres protocols) per fer-lo servir per l'intercanvi de missatges.

A més de definir el format dels missatges també defineix quines són les regles que han de regir aquest intercanvi, què passa en cas d'error i com s'ha de lligar amb els altres protocols.

Amb els serveis web d'aquest tipus s'exposen tota una sèrie de funcions mitjançant les quals es poden fer programes que poden fer que el servei faci tasques determinades i retorni resultats amb els quals es pugui treballar.

Composició dels missatges

Un missatge SOAP és bàsicament un document XML i per tant ha de complir totes les normes de creació de documents XML.

Problemes

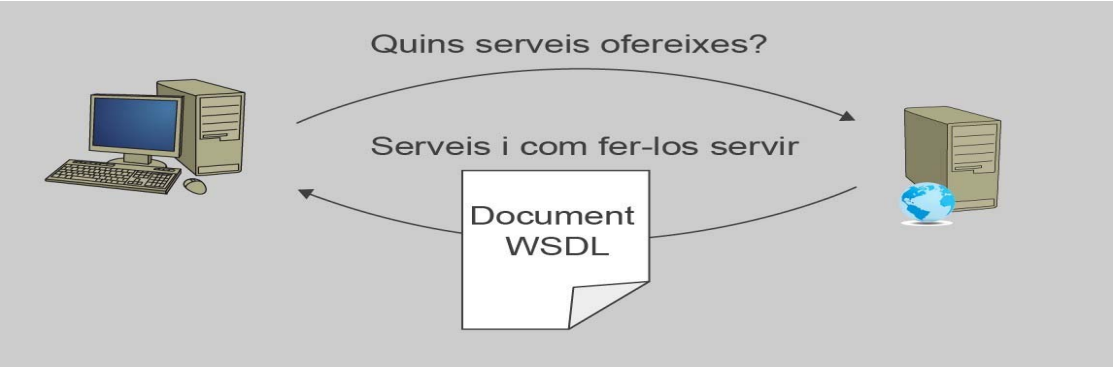
Una de les crítiques que es fa a SOAP és la mateixa que es fa a XML: malgrat tenir molts avantatges té la tendència a fer-se molt gran i això fa que SOAP acabi essent més lent que les altres tecnologies existents.

Tot i que aquests problemes són certs, per molta gent els avantatges que aporta -com la facilitat de lectura per als humans, la facilitat de detecció d'errors i el fet que solucioni els problemes de compatibilitat entre plataformes- el fan una opció que s'ha de tenir en compte.

WSDL

El WSDL (*Web services definition language*) és un llenguatge XML per descriure quins serveis ofereix un servei web i com s'ha de fer per accedir-hi.
Normalment l'ús d'un servei web requereix que el client obtingui abans el fitxer WSDL per saber quin servei s'ofereix i com s'ha de fer servir.

WSDL serveix per obtenir les característiques tècniques d'un servei web



Desenvolupament de serveis web

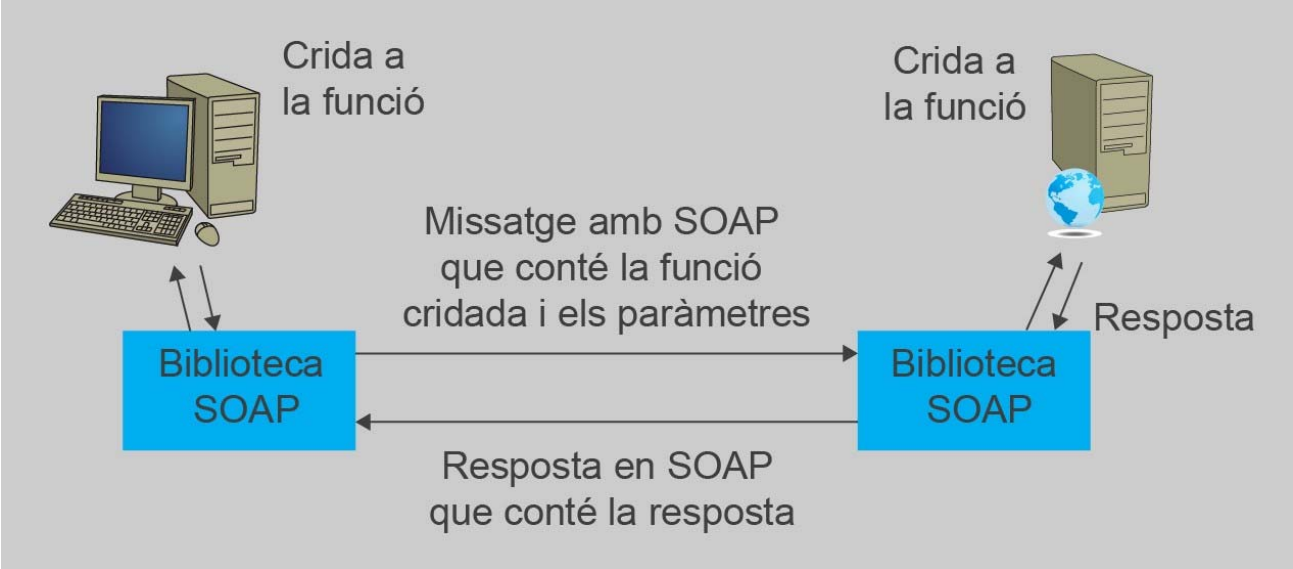
Els avantatges que poden aportar els serveis web (*web services*) no han passat desapercebuts i ràpidament els llenguatges de programació han anat incorporant components per desenvolupar-ne. Per exemple, el JDK (*Java development kit*) de Java incorpora una API per crear serveis web que s'anomena *Java Api for XML Web Services* (JAX-WS).

La majoria de les biblioteques o entorns de desenvolupament s'especialitzen en un dels dos grans grups de serveis web, però l'increment de la popularitat dels serveis REST ha fet que molts dels entorns basats en el protocol SOAP també incorporin la possibilitat de funcionar de forma compatible amb REST.

Desenvolupament de serveis basats en SOAP

Les biblioteques SOAP estan pensades per ser transparents amb vista als programadors, i per tant aquests programadors no solen tenir gaires problemes per fer-les servir. És així perquè les llibreries SOAP s'encarreguen de crear i eliminar els missatges SOAP, i al programa només hi arriben les dades que s'han enviat.

Un client demana un servei i internament la biblioteca crea el paquet SOAP per a les dades que ha d'enviar i l'hi envia. Quan la biblioteca rep les dades elimina el paquet SOAP i envia les dades netes al programa



Les biblioteques SOAP amaguen la implementació als programadors

Per tant, excepte en petits detalls, els programadors es poden despreocupar de l'existència de SOAP i només s'han de preocupar de quins serveis volen oferir.

Exemple fent servir php.

Existeix a php una classe per poder fer feina amb SOAP anomenada SoapClient. Es necessita coneixer el wsdl del servei que volem utilitzar. Per exemple per coneixer els resultats de la eurocopa de França: http://footballpool.dataaccess.eu/data/info.wso?wsdl

```
<?php

        $client = new SoapClient("http://footballpool.dataaccess.eu/data/info.wso?wsdl");

        //print_r($client->_getFunctions()); // per descobrir les funcions disponibles

        $result = $client->AllGames();
        $array = $result->AllGamesResult->tGameInfo;

        echo "<table border='1'>";
        echo "<tr><th>Data</th><th> </th><th> </th><th>Resultat</th></tr>";

        foreach($array as $k=>$v){
                echo "<tr>";
                echo "<td align='right'>" . $v->dPlayDate . "</td>
                <td><img src='{$v->Team1->sCountryFlag}'>" . $v->Team1->sName . "</td>
                <td align='right'>" . $v->Team2->sName . "<img src='{$v->Team1->sCountryFlag}'></td>
                <td align='center'>" . $v->sScore . "</td>";
                echo "</tr>";
        }

        echo "</table>";
?>
```

SERVEIS REST

REST deriva de REpresentational State Transfer .

REST es qualsevol interfície entre sistemes que fa servir HTTP per obtenir dades o generar operacions sobre aquestes dades fent servir preferiblement els formats XML i JSON. Es una alternativa als protocols estàndard d'intercanvi de dades com ara SOAP (Simple Object Access Protocol), que disposen d'una gran capacitat però també de mota complexitat.

Característiques de REST

- Protocol client/servidor sense estat (stateless): Això implica que el sistema **ha de tractar cadascuna de les peticions com si fos l'única, i per tant ha de tenir tot el necessari perquè el servidor pugui fer la tasca**. El servidor mai no tindrà en compte les peticions anteriors per tractar l'actual.

- Les operacions mes importants relacionades amb les dades en un sistema REST son quatre: **POST** (crear), **GET** (llegir), **PUT** (editar), **DELETE** (esborrar).

- Els objectes en REST sempre es manipulen a partir de la URI. Es la URI l'identificador únic de cada recurs.

- Interfície uniforme: per a la transferència de dades en un sistema REST, feim accions concretes (POST, GET, PUT y DELETE) sobre els recursos, identificats amb una URI.

- Sistema de capes: arquitectura jeràrquica entre els components. Cada una d'aquestes capes implementa una determinada funcionalitat dins el sistema REST.

- Utilització de hipermèdia. Part de la informació de les respostes seran hipervincles de navegació associats a altres recursos.

```
{
 "id": 78,
 "nom": "Joan",
 "llinatge": "García",
 "cotxes": [
{"cotxe": "http://servidor/concessionari/api/v1/clients/78/cotxes/1033" },
{"cotxe": "http://servidor/concessionari/api/v1/clients/78/cotxes/7033"}
}
```

Bones pràctiques fent servir REST

Utilitzar els mètodes HTTP per tal de fer les diferents peticions

Els mètodes HTTP milloren la claritat de les peticions Els quatre verbs més utilitzats son els següents:

GET

Llegir un determinat recurs (amb un identificador) o una col·lecció de recursos

PUT

Modificar un recurs específic o una col·lecció de recursos. Es pot fer servir també per crear un determinat recurs si es coneix l'identificador del recurs prèviament.

DELETE

Esborra un determinat recurs a partir del seu identificador.

POST

Crear un nou recurs. S'utilitza per operacions que no coincideixen amb cap mètode.

Fer servir noms de recursos clars i apropiats.

Fer servir noms de recursos clars i apropiats ens ajuda a entendre quina serà la resposta de la petició.

Els noms de recursos apropiats fan que la nostra API sigui més bona d'entendre. Els recursos son tractats com una jerarquía a través dels seus noms URI.

Algunes regles importants a l'hora de dissenyar els nostres noms de recursos:

- Fer servir identificadors dins les nostres URLs en lloc de paràmetres

  www.lloc.com/**usuaris/12345**
  en lloc de
  www.lloc.com/**api?tipus=usuari&id=12345**

- Fer servir la estructura jeràrquica de les URL per simplificar els noms dels recursos
- Dissenyar pensant amb els clients i no amb les dades.
- Els noms de recursos han d'esser noms. Evitar fer servir verbs per construir els noms de recursos. Per especificar la part corresponent al verb farem servir els mètodes HTTP.
- Utilitzar els plurals

  www.lloc.com/clients/33245/comandes/8769/linies/1
  millor que
  www.lloc.com/client/33245/comanda/8769/linia/1

  millor fer servir clients que client_llista.

- Fer servir minúscules separant les paraules amb _ o -
- Fer les URLs el més curt possible.

Fer servir els codis de resposta HTTP per indicar estat.

Els codis de resposta son part del protocol HTTP. Els podem fer servir per comunicar el resultat d'una petició. Per exemple si un recurs es crea correctament a partir d'una petició POST la nostra API pot tornar un 201

Alguns codis d'estat HTTP que es poden fer servir son:

200 OK : Correcte. El més utilitzat.

201 CREATED: Creació correcta (POST o PUT). Pot haver-hi contingut o no dins el cos de la resposta (response body)

204 NO CONTENT : Correcta per les operacions DELETE o PUT (no hi ha cap contingut al cos de la resposta).

400 BAD REQUEST : Error general.

401 UNAUTHORIZED : Error token d'autentificació incorrecta.

403 FORBIDDEN: Error. Usuari no autoritzat..

404 NOT FOUND: Recurs inexistent (a vegades emmascara els codis 401 o 403).

405 METHOD NOT ALLOWED : El mètode especificat no es correcte.

409 CONFLICT: Per exemple esborrat en cascada no suportat, entrada duplicada,...

500 INTERNAL SERVER ERROR: Error intern produït per una excepció.

Oferir JSON i XML

Normalment es fa servir molt més JSON. El ideal seria que els clients poguessin triar la resposta amb xml o json. La utilització de xml suposa també la implementació d'esquemes de validació i espais de noms i per tant suposa molt més complexitat.

Evitar crear respostes complexes.

Es millor crear respostes simples que implementin operacions CRUD sobre els recursos simples que crear respostes complexes.

Fer servir els links hipermèdia.

Si es fan servir links dins les respostes les APIs son més bones d'entendre i fer servir. Per exemple quan es retorna una col·lecció de recursos proporcionar links per la paginació (primer, darrer, anterior, següent) és molt útil o fer servir HTTP Location header amb el link a un recurs creat per PUT o POST.