Antecedentes de REST: sockets, RPC, SOAP, WSDL

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

Marzo de 2016

©2016 GSyC
Algunos derechos reservados.
Este trabajo se distribuye bajo la licencia
Creative Commons Attribution Share-Alike 4.0

Antecedentes de REST

La tecnología más habitual desde la década de 2010 para intercomunicar aplicaciones en red (no aplicaciones con usuarios), es REST
En el pasado, se emplearon diversos paradigmas (que pueden seguir en uso hoy)

- Sockets de red (*network sockets*)
- RPC. *Remote Procedure Call*
- SOAP (*Simple Object Access Protocol*) + WSDL (*Web Services Description Language*)

Network Sockets

- Los más difundidos son los sockets Berkeley de BSD, en los años 80
- Su principal limitación es que son de muy bajo nivel, simplemente envían y reciben datos entre dos *end points*: dirección IP+Puerto
- Se siguen usando hoy cuando es necesario este acceso de bajo a nivel a la red
- Para desarrollar aplicaciones suelen resultar poco prácticos, obliga al programador a *reinventar la rueda* en todo lo relativo a comunicaciones

RPC

RPC, *Remote Procedure Call*

- En los años 90 se popularizan arquitecturas más complejas que los sockets, basadas en el paradigma RPC
- Se populariza el *middleware*
  Software de sistemas que proporciona al software de aplicaciones funcionalidad por encima de la ofrecida por el SO. Facilita a los desarrolladores las tareas de comunicación y entrada/salida
  - CORBA: *Common Object Request Broker Architecture*, año 1991
  - Java RMI, Remote Method Invocation
  - DCOM (*Distributed Component Object Model*). Protocolo propietario de Microsoft, competidor de CORBA

- El programador invoca a funciones en máquinas remotas de manera prácticamente igual a una llamada a una función local. Se supone que el programador puede olvidar que trabaja en un sistema distribuido
- El propósito era ocultar al programador toda la complejidad propia de la red: (qué recursos hay, dónde están, cómo se accede, qué sucede si fallan)
- Todo esto que en principio parece muy conveniente, cae en desuso en la década de 2000

Inconvenientes del modelo RPC

- Las red es compleja. Ocultar esta complejidad inicialmente hace que programar sea sencillo, pero el software resultante resulta frágil
- Localizar una función local es trivial, una función remota, no
- Las llamadas locales raramente fallan. Las remotas, a menudo
- Las diferencias de latencia son típicamente de varios órdenes de magnitud...
- Las tecnologías RPC son muy heterogéneas
- Los atributos de las clases normalmente se exportan directamente, hay un acople muy fuerte entre el lenguaje de programación y el interfaz que se expone

Inconvenientes del modelo RPC (2)

- Se exportan tipos de datos complejos, propios del modelo de negocio. Proclives a cambiar, generando incompatibilidad
- Los sistemas resultan fuertemente acoplados: en ocasiones es necesario que todo el sistema emplee el mismo lenguaje de programación, un simple cambio de sistema operativo puede llegar a ser conflictivo
- Mala integración con cortafuegos y otros dispositivos de seguridad en red
- Supongamos paradigma cliente-servidor (el más habitual)
  - El cliente tiene que saber exactamente a qué funciones llamar, con qué parámetros de entrada y con qué valores de retorno
  - Típicamente, un pequeño cambio en el cliente o el servidor obliga a cambiar también el servidor o el cliente

Web services centradas en métodos

En los años 2000 aparece la primera generación del paradigma *web service* con tecnologías como XML-RPC, SOAP y WSDL
Tienen ventajas importantes sobre RPC:

- Están basados en HTTP, una de las causas fundamentales del éxito del web, por ser un estándar sencillo y práctico para el intercambio de documentos entre máquina y persona
  - Un *web service* usa el mismo principio y protocolo para la comunicación máquina-máquina
- Están basados en XML, un estándar de datos

- Las ventajas de esta primera generación de web services consiguen reducir mucho el acoplamiento con el sistema operativo y lenguaje de programación
- La integración con cortafuegos y otros dispositivos similares es muy sencilla, pues resulta tráfico web a estos efectos
- Pero el acoplamiento cliente-servidor sigue siendo fuerte, el cliente tiene que saber exactamente a qué función llamar, con qué parámetros y qué valores recibir

SOAP

SOAP (*Simple Object Access Protocol*) es un protocolo que permite la intercomunicación máquina-maquina, mediante llamadas a métodos

- Desarrollado en 1998 por Dave Winer y otros, con respaldo de Microsoft
- SOAP comenzó siendo un protocolo privado de Microsoft, lo que provocó la aparición de un protocolo muy similar, pero libre, XML-RPC
- Normalmente emplea HTTP, aunque también soporta otros protocolos como SMTP y RSS

- SOAP intercambia documentos XML, tanto para los distintos métodos como para los datos
- Los métodos no forman parte de SOAP, sino que los define el programador de la aplicación
- Similar a RPC, pero más sencillo al basarse en mensajes http
- Muy fácil integración con proxies y cortafuegos
- SOAP especifica exactamente cómo codificar las cabeceras HTTP y los documentos XML para intercambiar toda la información

Mensajes SOAP

Un mensaje SOAP es un documento XML ordinario que contiente:

- Un *sobre* (*envelope*) que identifica el documento XML como mensaje SOAP
- Opcionalmente, una cabecera
- El cuerpo, con información sobre las llamadas y las respuestas
- Opcionalmente, un elemento *fault* con información sobre errores

WSDL

Web Services Description Language (WSDL) es un lenguaje basado en XML para describir la funcionalidad ofrecida por un servicio web de primera generación, normalmente SOAP

- v1.0 año 2000, desarrollada por IBM y Microsoft
- v1.1 año 2001, muy similar a 1.1. Adoptada por el W3C (*World Wide Web Consortium*). La más popular
- v2.0 año 2007. No muy extendida

En su momento (sobre el año 2000) SOAP + WSDL fueron técnicas disruptivas.
Pero resultan protocolos bastante complejos, es necesario especificar muchos detalles sobre los métodos a invocar

Componentes de WSDL

- Servicio
  Contiene un conjunto de funciones que se exponen a http
- Puerto
  Dirección donde se ubica el servicio. Típicamente es una cadena con una URL http. El puerto es siempre el mismo para cualquier llamada
- Vínculo (Binding)
  Enlace entre la especificación RPC y la especificación SOAP

Componentes de WSDL (2)

- PortType
  Define un servicios web, las operaciones que se pueden realizar y los mensajes empleados para realizar las operaciones
- Operation
  Equivalente a una función de un lenguaje tradicional
- Message
  Parámetros de la operación
- Types
  Descripción de los tipos de los parámetros

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl"
             xmlns:tns="http://www.tmsws.com/wsdl20sample"
             xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/"
             targetNamespace="http://www.tmsws.com/wsdl20sample">

<documentation>
    This is a sample WSDL 2.0 document.
</documentation>

<!-- Abstract type -->
   <types>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://www.tmsws.com/wsdl20sample"
                 targetNamespace="http://www.example.com/wsdl20sample">

         <xs:element name="request"> ... </xs:element>
         <xs:element name="response"> ... </xs:element>
      </xs:schema>
   </types>
```

```
<!-- Abstract interfaces -->
   <interface name="Interface1">
      <fault name="Error1" element="tns:response"/>
      <operation name="Get" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
   </interface>

<!-- Concrete Binding Over HTTP -->
   <binding name="HttpBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/http">
      <operation ref="tns:Get" whttp:method="GET"/>
   </binding>
```

```
<!-- Concrete Binding with SOAP-->
   <binding name="SoapBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/soap"
            wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
            wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-respon
      <operation ref="tns:Get" />
   </binding>

<!-- Web Service offering endpoints for both bindings-->
   <service name="Service1" interface="tns:Interface1">
      <endpoint name="HttpEndpoint"
                binding="tns:HttpBinding"
                address="http://www.example.com/rest/"/>
      <endpoint name="SoapEndpoint"
                binding="tns:SoapBinding"
                address="http://www.example.com/soap/"/>
   </service>
</description>
```

Fuente: wikipedia

	Acoplamiento con la plataforma	**Acoplamiento cliente-servidor**
Sockets	Muy fuerte	Muy fuerte
RPC	Fuerte (middleware)	Fuerte
Web services 1ª gen. (SOAP, XML-RPC, WSDL)	Débil	Fuerte
Web services 2ª gen. (REST)	Débil	Débil