Sumario

Que es un WebServices?....2
- Arquitectura....2
- Estándares empleados....3
- Ventajas de los servicios web....3
- Inconvenientes de los servicios web....3
- Razones para crear servicios Web....4
- Tipos de Servicios....4
- Protocolo HTTP....4
  - Versiones....5
  - Solicitud HTTP: (Request)....6
  - Request (Ejemplo)....6
  - URL....6
  - Respuesta HTTP: (Response)....7
  - Reponse (Ejemplo)....7

REST....7
- Que es?....7
- Conceptos....8
  - Nivel 0 Comunicación....8
  - Nivel 1 Recursos....10
  - Nivel 2 HTTP Verbs....10
  - Nivel 3 HATEOAS....11
- JAX-RS....12
  - Especificación....12

Que es un WebServices?

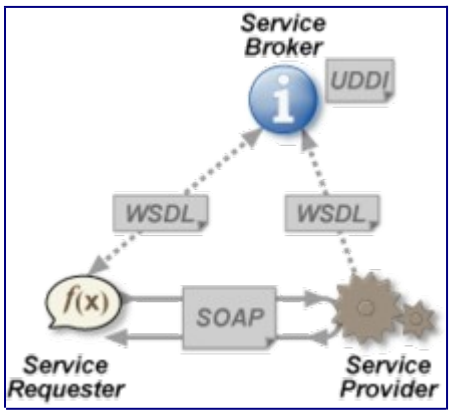
Un **servicio web** (en inglés, *web service* o *web services*) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web.

El W3C define un servicio web como:

> Un servicio web es un sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable. Cuenta con una interfaz descrita en un formato procesable por un equipo informático (específicamente en WSDL), a través de la que es posible interactuar con el mismo mediante el intercambio de mensajes SOAP, típicamente transmitidos usando serialización XML sobre HTTP conjuntamente con otros estándares web.

Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados.

Arquitectura



Arquitectura de los Servicios Web SOAP

En la arquitectura de servicios web existen tres partes: proveedor de servicios web, el que pide el servicio web y el publicador. El proveedor de servicios envía al publicador del servicio un fichero WSDL con la definición del servicio web. El que pide el servicio contacta con el publicador y descubre quién es el proveedor (protocolo WSDL) y contacta con el proveedor (protocolo SOAP). El proveedor valida la petición de servicio y envía el dato estructurado en formato XML utilizando el protocolo SOAP. El fichero XML es validado de nuevo por el que pide el servicio utilizando un fichero XSD.

Estándares empleados

- *Web Services Protocol Stack*: conjunto de servicios y protocolos de los servicios web.
- XML (*Extensible Markup Language*): formato estándar para los datos que se vayan a intercambiar.
- SOAP (*Simple Object Access Protocol*) o XML-RPC (*XML Remote Procedure Call*): protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como *Hypertext Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP), o *Simple Mail Transfer Protocol* (SMTP).
- WSDL (*Web Services Description Language*): es el lenguaje de la interfaz pública para los servicios web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios web.
- UDDI (*Universal Description, Discovery and Integration*): protocolo para publicar la información de los servicios web. Permite comprobar qué servicios web están disponibles.
- WS-Security (*Web Service Security*): protocolo de seguridad aceptado como estándar por OASIS (*Organization for the Advancement of Structured Information Standards*). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.
- REST (*Representational State Transfer*): arquitectura que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etcétera) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.
- GraphQL, arquitectura alternativa a REST.

Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

Inconvenientes de los servicios web

- Para realizar transacciones, no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (*Common Object Request Broker Architecture*).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como *Java Remote Method Invocation* (RMI), CORBA o *Distributed Component Object Model* (DCOM). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en *firewall* cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

Razones para crear servicios Web

La principal razón para usar servicios Web es que se pueden utilizar con HTTP sobre *Transmission Control Protocol* (TCP) en el puerto de red 80. Dado que las organizaciones protegen sus redes mediante *firewalls* (que filtran y bloquean gran parte del tráfico de Internet), cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores web. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otras computadoras en red. Las que había eran *ad hoc* y poco conocidas, tales como *Electronic Data Interchange* (EDI), *Remote Procedure Call* (RPC), u otras API.

Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.

Se espera que para los próximos años mejoren la calidad y cantidad de servicios ofrecidos basados en los nuevos estándares.

Tipos de Servicios

1. SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call): protocolos sobre los que se establece el intercambio.

2. REST (Representational State Transfer): Arquitectura que, haciendo uso del protocolo HTTP proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etcétera) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

3. GraphQL, Arquitectura alternativa a REST.

Protocolo HTTP

Es el protocolo de transferencia de hipertexto que es usado en la Web. HTTP significa HyperText Transfer Protocol, o Protocolo de Transferencia de Hipertexto. Este protocolo fue desarrollado por las instituciones internacionales W3C y IETF y se usa en todo tipo de transacciones a través de Internet.

Este protocolo opera por petición y respuesta entre el cliente y el servidor. A menudo las peticiones tienen que ver con archivos, ejecución de un programa, consulta a una base de datos, traducción y otras funcionalidades. Toda la información que opera en la Web mediante este protocolo es identificada mediante el URL o dirección.La típica transacción de protocolo HTTP se compone de

un encabezado seguido por una línea en blanco y luego un dato. Este encabezado define la acción requerida por el servidor.

El protocolo opera con códigos de respuesta de tres dígitos, que comunican si conexión fue rechazada, si se realizó con éxito, si ha sido redirigida hacia otro URL, si existe un error por parte del cliente, o bien, por parte del servidor.

Los navegadores web tienden a complementar la acción del HTTP como ocurre, por ejemplo, con las denominadas "cookies", que permiten almacenar información de la sesión, función de la que no dispone este protocolo, ya que opera sin estado.

Versiones

HTTP ha pasado por múltiples versiones del protocolo, muchas de las cuales son compatibles con las anteriores. El RFC 2145 describe el uso de los números de versión de HTTP. El cliente le dice al servidor al principio de la petición la versión que usa, y el servidor usa la misma o una anterior en su respuesta.

0.9 (lanzada en 1991)

Obsoleta. Soporta sólo un comando, GET, y además no especifica el número de versión HTTP. No soporta cabeceras. Como esta versión no soporta POST, el cliente no puede enviarle mucha información al servidor.

HTTP/1.0 (mayo de 1996)

Esta es la primera revisión del protocolo que especifica su versión en las comunicaciones, y todavía se usa ampliamente, sobre todo en servidores proxy. Permite los métodos de petición GET, HEAD y POST.

HTTP/1.1 (junio de 1999)

Versión más usada actualmente; Las conexiones persistentes están activadas por defecto y funcionan bien con los proxies. También permite al cliente enviar múltiples peticiones a la vez por la misma conexión (pipelining) lo que hace posible eliminar el tiempo de Round-Trip delay por cada petición.

HTTP/1.2 (febrero de 2000)

Los primeros borradores de 1995 del documento *PEP — an Extension Mechanism for HTTP* (el cual propone el Protocolo de Extensión de Protocolo, abreviado PEP) los hizo el World Wide Web Consortium y se envió al Internet Engineering Task Force. El PEP inicialmente estaba destinado a convertirse en un rango distintivo de HTTP/1.2. En borradores posteriores, sin embargo, se eliminó la referencia a HTTP/1.2. El RFC 2774 (experimental), *HTTP Extension Framework,* incluye en gran medida a PEP. Se publicó en febrero de 2000.

HTTP/2 (mayo de 2015)

Artículo principal: *HTTP/2*

En el año 2012 aparecen los primeros borradores de la nueva versión de HTTP (HTTP/2). Esta nueva versión no modifica la semántica de aplicación de http (todos los conceptos básicos continúan sin cambios). Sus mejoras se enfocan en como se empaquetan los datos y en el transporte. Por ejemplo, añade el uso de una única conexión, la compresión de cabeceras

o el servicio 'server push'. Los exploradores más importantes solo soportan HTTP 2.0 sobre TLS usando la extensión ALPN que requiere TLSv1.2 o superior.

Solicitud HTTP: (Request)

Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor. Comprende:

1. Una línea de solicitud: una línea que especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. La línea está formada por tres elementos que deben estar separados por un espacio: el método, la dirección URL y la versión del protocolo utilizada por el cliente (por lo general, HTTP/1.0)

2. Los campos del encabezado de solicitud: un conjunto de líneas opcionales que permiten aportar información adicional sobre la solicitud y/o el cliente (navegador, sistema operativo, etc.). Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

3. El cuerpo de la solicitud: un conjunto de líneas opcionales que deben estar separadas de las líneas precedentes por una línea en blanco y, por ejemplo,permiten que se envíen datos por un comando POST durante la transmisión de datos al servidor utilizando un formulario.

Request (Ejemplo)

```
GET http://es.kioskea.net HTTP/1.0 Accept : Text/html If-Modified-Since : Saturday,15-January-2000 14:37:11 GMT User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

URL

● Es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet.

● URL significa "Uniform Resource Locator", o bien, "Localizador Uniforme de Recursos". Se trata de una serie de caracteres que responden a un formato estándar y que permiten clasificar recursos subidos a Internet para su descarga y utilización.

● URI ("Uniform Resource Identifier" o "Identificador Uniforme de Recursos"), suele confundirse cuando hablamos de URL en enlaces sobre Internet.

Respuesta HTTP: (Response)

Una respuesta HTTP es un conjunto de líneas que el servidor envía al cliente.

Comprende:

1. Una línea de respuesta: una línea que especifica el tipo de documento solicitado

2. Los campos del encabezado de la respuesta a la solicitud.

3. El cuerpo de la respuesta.

4. Un código de respuesta.

Reponse (Ejemplo)

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Título del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
  .
  .
  .
</body>
</html>
```

Para mayor información leer http en el capitulo 1 (Parte 01) de esta colección de apuntes.

REST

Que es?

Esta pregunta es una de las más habituales en nuestros días. Para algunas personas REST es una arquitectura , para otras es un patrón de diseño , para otras un API. ¿Qué es exactamente? . REST o

Representational State Transfer es un ESTILO de Arquitectura a la hora de realizar una comunicación entre cliente y servidor.

La Transferencia de Estado Representacional o REST por sus siglas en Inglés (Representational State Transfer) es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.Si bien el término REST se refiere a un conjunto de principios de arquitectura, en la actualidad se utiliza en un sentido más amplio para describir cualquier interfaz web simple que utiliza XML o JSON y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP.

Los sistemas que siguen los principios REST se denominan a menudo como RESTful.

Conceptos

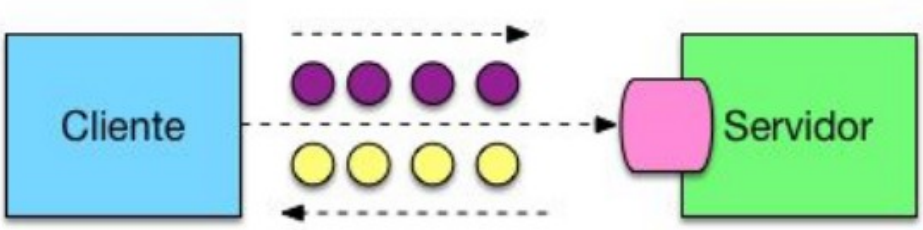
● Stateless: Es un protocolo cliente/servidor sin estado. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente, ni el servidor necesitan recordar ningún estado de las comunicaciones anteriores.

● CRUD: Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información. HTTP tiene un conjunto de operaciones, las más importantes son: GET, POST, PUT Y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD (ABM en español, Alta, Baja, Modificación y consulta) que se requieren para la persistencia de datos.

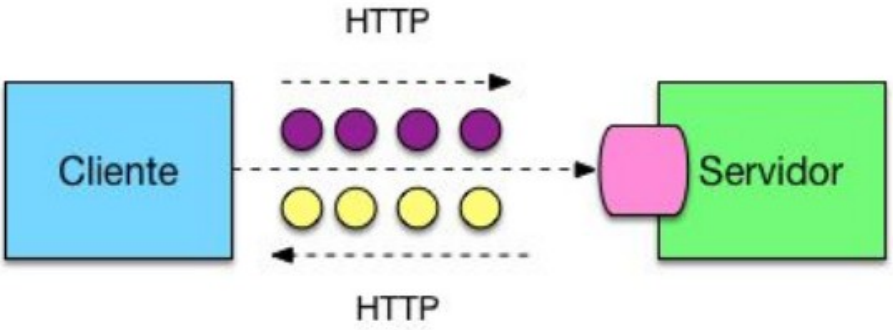
● URI: Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de una URI.

Nivel 0 Comunicación.

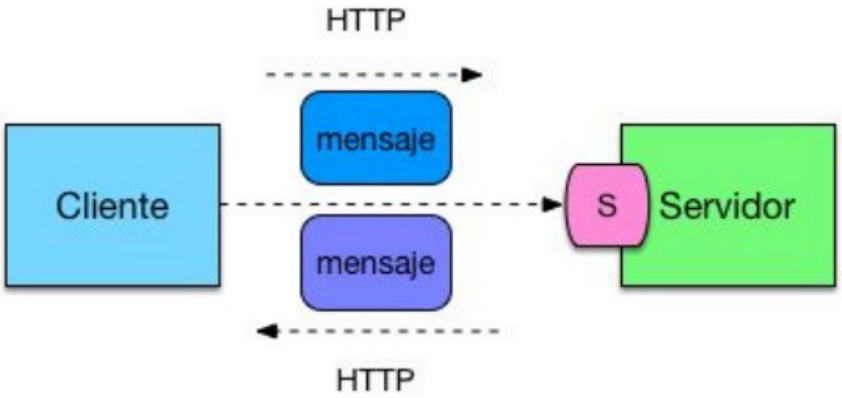
Vamos a intentar explicarlo esto paso a paso . Habitualmente cuando realizamos una comunicación cliente servidor accedemos al servidor en un punto de acceso , le enviamos una información y recibimos un resultado.



Hay muchas formas de realizar esta operación, hoy en día una de las necesidades más claras es que esa comunicación sea abierta y podemos acceder desde cualquier sitio. Así que para ello, usaremos el protocolo de comunicación HTTP.



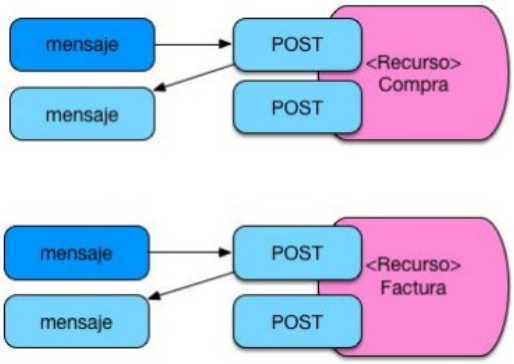
El siguiente paso es decidir qué tipología de mensajes enviamos. Como punto de partida podemos mandar a un servicio un mensaje en formato XML o JSON. El servicio lo recepcionará y nos devolverá una respuesta.



Esto es lo que se conoce en Arquitecturas REST como el **nivel 0**. No tenemos ningún tipo de organización.

Nivel 1 Recursos.

En vez de tener servicios con métodos diversos, se declaran Recursos. ¿Qué es un Recurso? . Un recurso hace referencia a un concepto importante de nuestro negocio (Facturas, Compras etc). Este estilo permite un primer nivel de organización permitiendo acceder a cada uno de los recursos de forma independiente, favoreciendo la reutilización y aumentando la flexibilidad.



Nivel 2 HTTP Verbs.

Por ahora las peticiones las hemos realizado usando GET o POST indistintamente . En el nivel 2 las operaciones pasan a ser categorizadas de forma más estricta. Dependiendo de cada tipo de operación se utilizará un método diferente de envío.

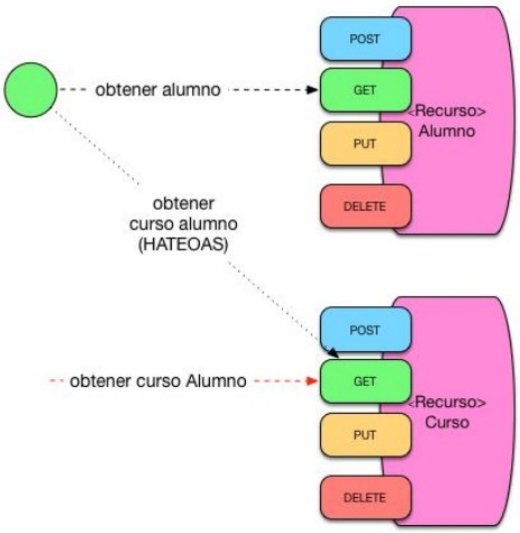
1. GET : Se usará para solicitar consultar a los recursos.
2. POST : Se usará para insertar nuevos recursos.
3. PUT : Se usará para actualizar recursos.
4. DELETE : Se usará para borrar recursos.

Nivel 3 HATEOAS.

Queda un nivel más que es el que se denomina HATEOAS (Hypertext As The Engine Of Application State). ¿Para qué sirve ?. Supongamos que queremos acceder a un recurso Alumno via REST . Si tenemos una Arquitectura a nivel 2 primero accederemos a ese recurso utilizando GET. En segundo lugar deberemos acceder al recurso de Cursos para añadir al alumno al curso (linea roja) .

Esto implica que el cliente que accede a los servicios REST asume un acoplamiento muy alto, debe conocer la url del Alumno y la del Curso. Sin embargo si el recurso del Alumno contiene un link al recurso de Curso esto no hará falta. Podríamos tener una estructura JSON como la siguiente:

```
{nombre:"Diego", apellidos:"Peralta", cursos: "http://servicio/cursos" }
```



Podremos acceder directamente al recurso de Curso utilizando las propiedades del Alumno esto es HATEOAS. De esta forma se aumenta la flexibilidad y se reduce el acoplamiento. Construir arquitecturas sobre estilo REST no es sencillo y hay que ir paso a paso.

JAX-RS

Es la API de Java para servicios RESTful (JAX-RS) es una especificación del lenguaje de programación Java que brinda soporte en la creación de servicios web de acuerdo con el patrón arquitectónico Representational State Transfer (REST). JAX-RS utiliza anotaciones, introducidas en Java SE 5, para simplificar el desarrollo y la implementación de clientes y puntos finales del servicio web. (Endpoint)

Desde la versión 1.1 en adelante, JAX-RS es una parte oficial de Java EE 6.

Una característica notable de ser parte oficial de Java EE es que no es necesaria ninguna configuración para comenzar a utilizar JAX-RS. Para entornos que no son Java EE 6, se requiere una pequeña entrada en el descriptor de despliegue web.xml.

Especificación

JAX-RS proporciona algunas anotaciones para ayudar a mapear una clase de recursos (un POJO) como un recurso web. Las anotaciones usan el paquete Java javax.ws.rs.

Incluyen:

● **@Path** Especifica la ruta relativa para una clase o método de recursos.

● **@GET, @PUT, @POST, @DELETE y @HEAD** Especifican el tipo de solicitud HTTP de un recurso.

● **@Produces** Especifica los tipos de medios de Internet de respuesta (utilizados para la negociación de contenido).

● **@Consumes** Especifica los tipos de medios de Internet aceptados.

Además, proporciona anotaciones adicionales a los parámetros del método para extraer información de la solicitud. Todas las anotaciones @ * Param toman una clave de alguna forma que se utiliza para buscar el valor requerido.

● **@PathParam** Vincula el parámetro de método a un segmento de ruta.

● **@QueryParam** Vincula el parámetro de método al valor de un parámetro de consulta HTTP.

● **@MatrixParam** Vincula el parámetro de método al valor de un parámetro de matriz HTTP.

● **@HeaderParam** Vincula el parámetro de método a un valor de encabezado HTTP.

● **@CookieParam** Vincula el parámetro de método a un valor de cookie.

● **@FormParam** Vincula el parámetro de método a un valor de formulario.

● **@DefaultValue** Especifica un valor predeterminado para los enlaces anteriores cuando no se encuentra la clave.

● **@Context** Devuelve todo el contexto del objeto (por ejemplo @Context HttpServletRequest request).