

Introducción a los servicios web RESTful

REFS:

[HTTPS://DOCS.ORACLE.COM/JAVAEE/7/TUTORIAL/JAXRS.HTM#GIEPU](https://docs.oracle.com/javaee/7/tutorial/jaxrs.htm#giepu)

[HTTPS://DOSIDEAS.COM/NOTICIAS/JAVA/314-INTRODUCCION-A-LOS-SERVICIOS-WEB-RESTFUL](https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful)

(REST - Representational State Transfer)

- ▶ La Transferencia de Estado Representacional (REST - Representational State Transfer) fue ganando amplia adopción en toda la web como una alternativa más simple a SOAP y a los servicios web basados en el Lenguaje de Descripción de Servicios Web (Web Services Description Language - WSDL). Ya varios grandes proveedores de Web están migrando a esta tecnología, incluyendo a Yahoo, Google y Facebook, quienes marcaron como obsoletos a sus servicios SOAP y WSDL y pasaron a usar un modelo más fácil de usar, orientado a los recursos.

Presentando REST



- ▶ REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

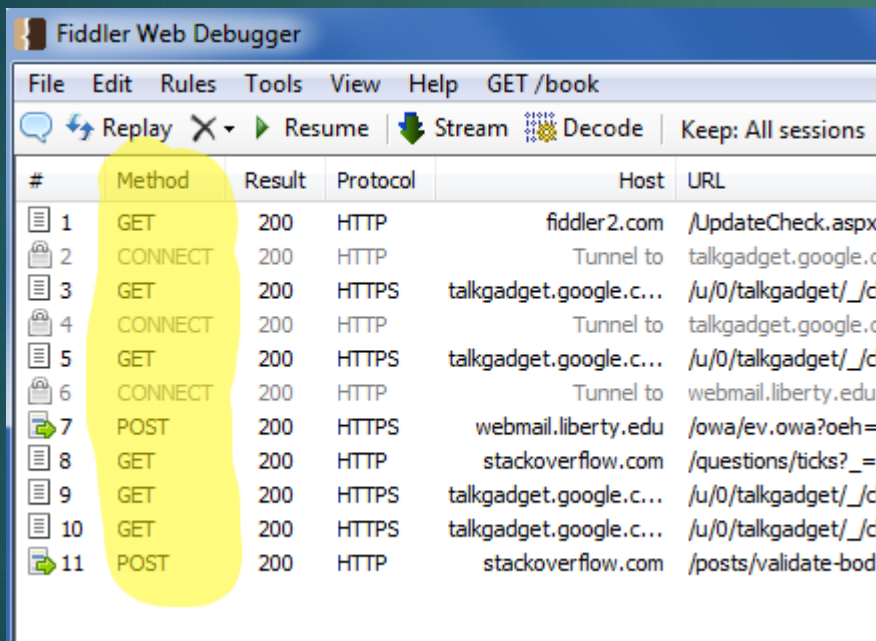
Los 4 principios de REST

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

- ▶ utiliza los métodos HTTP de manera explícita
- ▶ no mantiene estado
- ▶ expone URIs con forma de directorios
- ▶ transfiere XML, JavaScript Object Notation (JSON), o ambos

REST utiliza los métodos HTTP de manera explícita

- ▶ GET
/agregarusuario?nombre=Miguel
HTTP/1.1

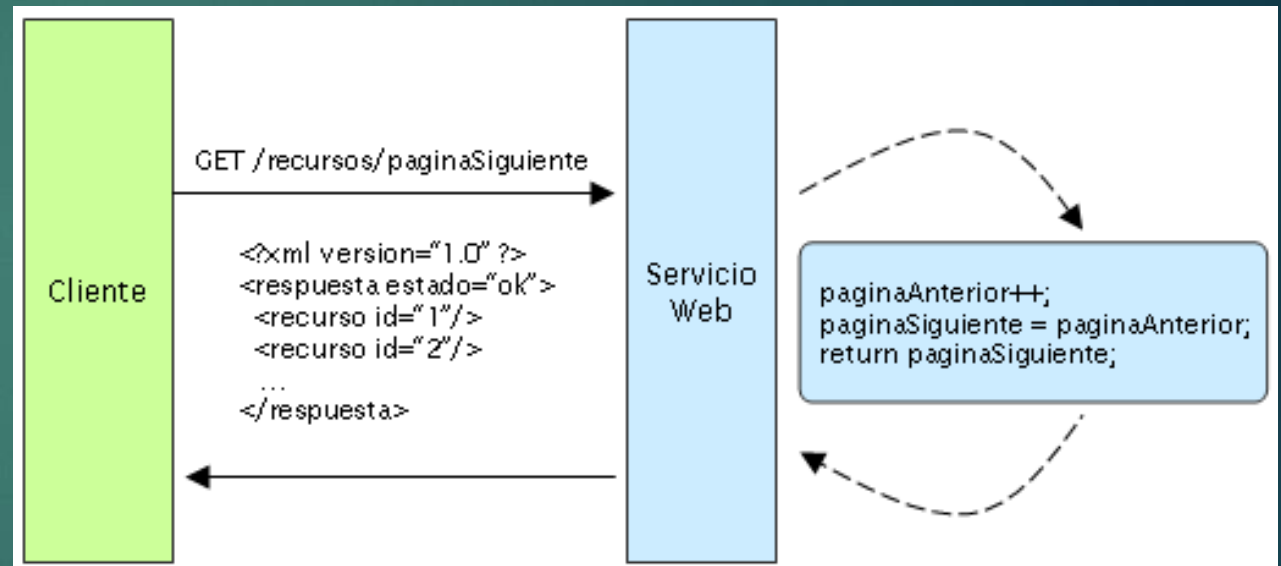


#	Method	Result	Protocol	Host	URL
1	GET	200	HTTP	fiddler2.com	/UpdateCheck.aspx
2	CONNECT	200	HTTP	Tunnel to	talkgadget.google.c...
3	GET	200	HTTPS	talkgadget.google.c...	/u/0/talkgadget/_c...
4	CONNECT	200	HTTP	Tunnel to	talkgadget.google.c...
5	GET	200	HTTPS	talkgadget.google.c...	/u/0/talkgadget/_c...
6	CONNECT	200	HTTP	Tunnel to	webmail.liberty.edu
7	POST	200	HTTPS	webmail.liberty.edu	/owa/ev.owa?oeh=...
8	GET	200	HTTP	stackoverflow.com	/questions/ticks?_=...
9	GET	200	HTTPS	talkgadget.google.c...	/u/0/talkgadget/_c...
10	GET	200	HTTPS	talkgadget.google.c...	/u/0/talkgadget/_c...
11	POST	200	HTTP	stackoverflow.com	/posts/validate-bod...

- ▶ se usa POST para crear un recurso en el servidor
- ▶ se usa GET para obtener un recurso
- ▶ se usa PUT para cambiar el estado de un recurso o actualizarlo
- ▶ se usa DELETE para eliminar un recurso

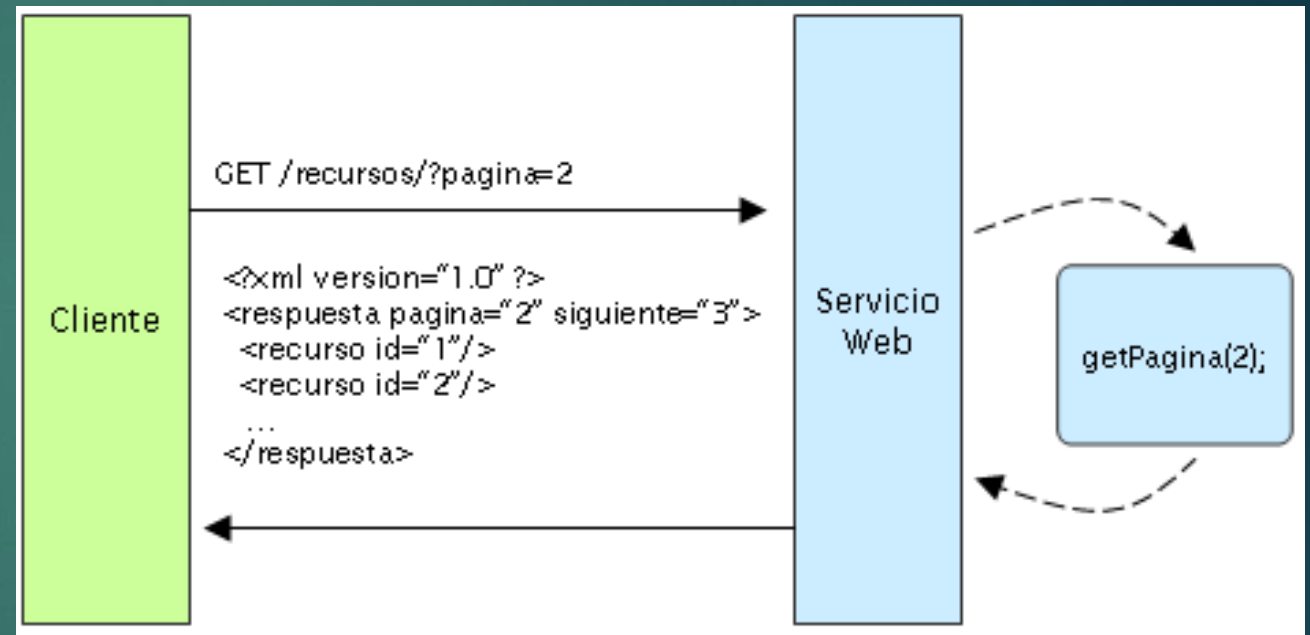
REST no mantiene estado

- ▶ Los servicios web REST necesitan escalar para poder satisfacer una demanda en constante crecimiento. Se usan clusters de servidores con balanceadores de carga y alta disponibilidad, proxies, y gateways de manera de conformar una topología servicable, que permita transferir peticiones de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio web.



REST no mantiene estado

- Una petición completa e independiente hace que el servidor no tenga que recuperar ninguna información de contexto o estado al procesar la petición. Una aplicación o cliente de servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta. De esta manera, el no mantener estado mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor elimina la necesidad de sincronizar los datos de la sesión con una aplicación externa.



REST expone URIs con forma de directorios

- ▶ Las URI de los servicios web REST deben ser intuitivas. Pensemos en las URI como una interfaz auto-documentada que necesita de muy poca o ninguna explicación o referencia para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.
- ▶ Una forma de lograr este nivel de usabilidad es definir URIs con una estructura al estilo de los directorios. De acuerdo a esta definición, una URI no es solamente una cadena de caracteres delimitada por barras, sino más bien un árbol con subordinados y padres organizados como nodos. Por ejemplo, en un servicio de hilos de discusiones que tiene temas varios, se podría definir una estructura de URIs como esta:

`http://www.miservicio.org/discusion/temas/{tema}`

`http://www.miservicio.org/discusion/2008/12/23/{tema}`

REST transfiere XML, JSON, o ambos

- La representación de un recurso en general refleja el estado actual del mismo y sus atributos al momento en que el cliente de la aplicación realiza la petición. La representación del recurso son simples "fotos" en el tiempo.

