La principal diferencia es que:

1. El método PUT es idempotente en HTTP lo que significa que producirá el mismo resultado si se ejecuta varias veces
2. El método POST no es idempotente, ya que si se ejecuta varias veces está creando varios elementos

Diferencia #2

1. El método POST se utiliza para crear una nueva entidad
2. El método PUT se utiliza para actualizar(reemplazar) una entidad existente, tener presente que si en la trama se envía solo una parte de los valores a atualizar, los demas campos se setearán a null o vacio

Diferencia #3

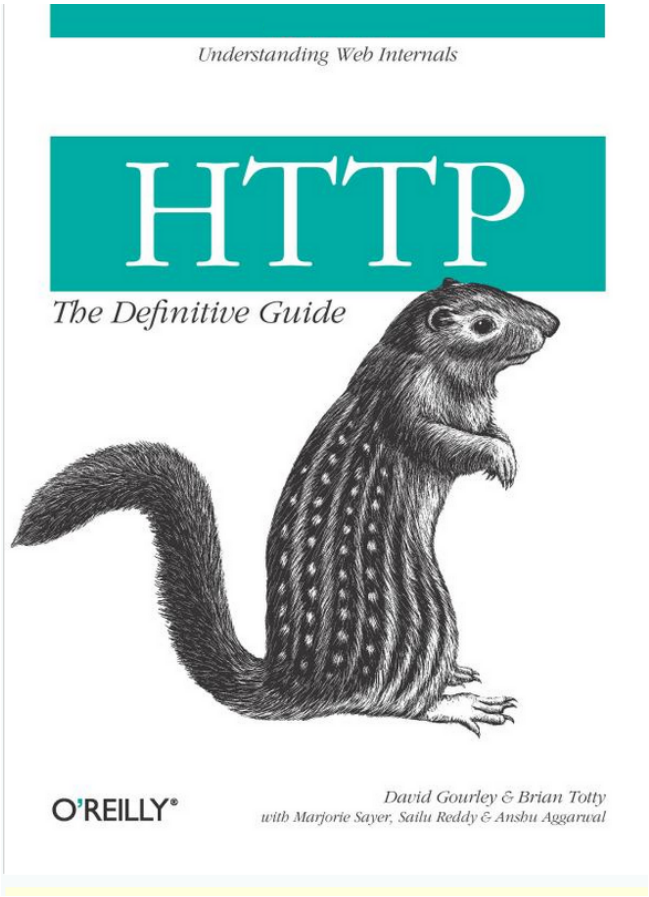
- Métodos seguros: GET y HEAD, ya que NO modifican recursos en el servidor
- Métodos inseguros: POST y PUT, ya que SI modifican recursos en el servidor

Lectura:

La idempotencia es una cosa importante mientras se construye una API RESTful tolerante a fallos. La idempotencia es también la razón de por qué debería usar PUT sobre POST para actualizar un recurso en REST. Por ejemplo, supongamos que un cliente desea actualizar un recurso a través de POST. Dado que POST no es un método idempotent, llamarlo varias veces puede resultar en actualizaciones incorrectas.

En el mundo real es muy probable que su solicitud POST puede tiempo de espera, lo que sucederá con el recurso que. ¿Se actualiza realmente el recurso? ¿Se ha producido el tiempo de espera durante el envío de la solicitud al servidor, o la respuesta al cliente? ¿Podemos volver a intentarlo nuevamente, o necesitamos averiguar primero qué ha sucedido con el recurso? Mediante el uso de métodos idempotentes como PUT, no tiene que responder a esta pregunta, pero podemos enviar la solicitud de forma segura hasta que obtengamos una respuesta del servidor.

Aquí recomendamos una guía que tiene todos estos conceptos y buenas prácticas para diseñar un API REST

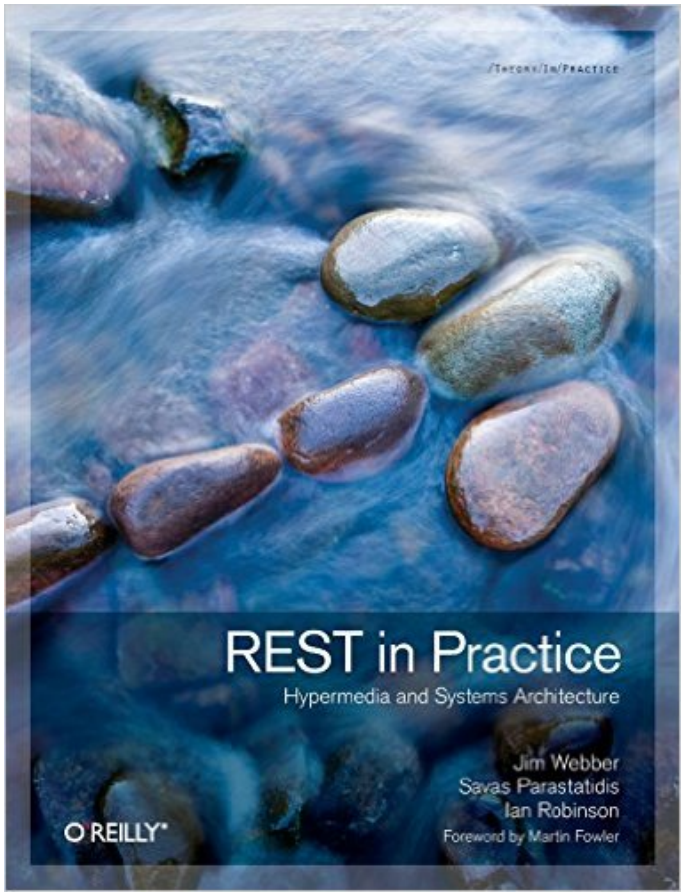


¿Cuándo utilizar los métodos PUT y POST en REST?

Ahora «es tiempo para algunos conocimientos prácticos sobre cuándo usar los métodos PUT y POST para llamar a WebServices RESTful.

1. Usted debe utilizar POST para crear nuevos recursos y PUT para actualizar los recursos existentes
2. Utilice PUT cuando conozca el «id» del objeto, p. Orden, Libro, Empleado
3. Utilice POST cuando necesite que el servidor controle la generación de URL de sus recursos
4. Ejemplos:
    1. PUT/items/1/update
    2. POST/items/create

Usted puede leer más REST en el libro Práctica para aprender más directrices sobre el diseño de una API REST. A veces, leer un par de libros sobre el tema ayuda a reducir la confusión.



Fuente:
http://javarevisited.blogspot.com/2016/10/difference-between-put-and-post-in-restful-web-service.html#ixzz4UuuyQmtc

Julio Pari (IT Architect IBM)

Si te ha interesado este artículo y deseas un apoyo o asesoría en algún requerimiento, envíame un mensaje a: (info@juliopari.com) o sino a través de Linkedin: https://www.linkedin.com/in/juliopari/