Unidad 3 - Clase 2: Otros métodos

Contenidos

Métodos PUT, POST y DELETE

Los métodos que se pueden utilizar para modificar los recursos son HTTP PUT, POST y DELETE.

Los métodos PUT, POST y DELETE se pueden utilizar para modificar los recursos de objeto de negocio y los recursos de estructura de objeto. Sin embargo, las reglas empresariales de un objeto pueden impedir que se actualice mediante una solicitud de la API de REST. Por ejemplo, una solicitud DELETE en un recurso de orden de trabajo puede fallar si las validaciones del objeto de negocio impiden las supresiones a causa del estado actual del objeto de negocio.

Sintaxis

```
método uri HTTP/1.1
parámetro=valor&parámetro=valor&...
```

método es PUT, POST o DELETE

uri es un URI y los parámetros de consulta asociados

parámetro es un parámetro del recurso que se actualiza

valor es el valor del parámetro

Método PUT

Utilice el método PUT para actualizar o insertar un recurso. Una solicitud de actualización debe proporcionar el ID exclusivo del recurso. Para actualizar un recurso de estructura de objeto, se requiere el ID del objeto principal.

Método POST

Utilice el método POST para actualizar o insertar un recurso.

Método DELETE

El método DELETE requiere el ID exclusivo del recurso.

Simultáneo simultáneas de recursos

Se puede controlar el proceso de forma que sólo se actualice o suprima un recurso si éste no ha sido cambiado por otro usuario o aplicación después de la consulta inicial.

Método PUT

Utilice el método PUT para actualizar o insertar un recurso. Una solicitud de actualización debe proporcionar el ID exclusivo del recurso. Para actualizar un recurso de estructura de objeto, se requiere el ID del objeto principal.

Para actualizar o insertar un recurso de estructura de objeto, puede especificar el parámetro **`_action`** para identificar acciones proporcionadas por la infraestructura de integración, por ejemplo, las acciones Change o AddChange.

Ejemplo: actualizar un activo

El método siguiente actualiza el activo que tiene el ID 1234. La descripción del activo se cambia por `mi_nueva_descripción` y el tipo se cambia por `OPERATING`:

```
PUT maxrest/rest/mbo/asset/1234 HTTP/1.1
description=mi_nueva_descripción&type=OPERATING
```

Ejemplo: actualizar un objeto de nivel inferior

El método siguiente actualiza el objeto de negocio `assetspec`, que es un nivel inferior de `asset`. El valor se cambia por `nuevo_valor` y la descripción se cambia por `mi_nueva_descripción`:

```
PUT maxrest/rest/mbo/assetspec/5678 HTTP/1.1
value=mi_nuevo_valor&description=mi_nueva_descripción
```

Ejemplo: establecer el contexto de un objeto de nivel superior

Para insertar o actualizar un objeto de negocio de nivel inferior, es posible que tenga que establecer el contexto del objeto de negocio de nivel superior, como en el ejemplo siguiente, en el que se actualiza la descripción de la línea de OC y se proporciona el ID de la OC para el contexto:

```
PUT maxrest/rest/mbo/po/1234/poline/5678 HTTP/1.1
description=mi_nueva_descripción
```

Método POST

Utilice el método POST para actualizar o insertar un recurso.

Para actualizar un recurso, debe especificar el ID del recurso. Para crear un recurso, debe especificar la clave primaria y todos los campos necesarios que no tienen un valor predeterminado, pero no es necesario ningún ID.

Para actualizar o insertar un recurso de estructura de objeto, puede especificar el parámetro **`_action`** para identificar acciones proporcionadas por la infraestructura de integración, por ejemplo, Change o AddChange.

Para actualizar o insertar un objeto de nivel inferior de un recurso de estructura de objeto, los datos del formulario deben identificar cada aparición del objeto de nivel inferior.

Cuando utilice el método POST para crear un recurso, especifique el parámetro de consulta **`_ulcr`** con el valor `1` para que la respuesta incluya un vínculo para que el cliente acceda al nuevo recurso. De lo contrario, el contenido del recurso se incluye en la respuesta. El vínculo se incluye en la propiedad de cabecera **`Ubicación`** y el código de respuesta HTTP es 201 para identificar que se proporciona un vínculo en lugar de los datos.

Ejemplo: insertar un activo

El método siguiente inserta el activo 127 dentro de la planta BEDFORD utilizando un recurso de objeto de negocio. El activo y la planta forman la clave primaria.

```
POST maxrest/rest/mbo/asset HTTP/1.1
assetnum=127&siteid=BEDFORD&description=mi_nueva_descripción&type=OPERATING
```

Ejemplo: actualizar un activo especificando su ID

El método siguiente actualiza un activo proporcionando el ID como parte del URI:

```
POST maxrest/rest/mbo/asset/1234 HTTP/1.1
description=mi_nueva_descripción&type=OPERATING
```

Ejemplo: especificar una acción

El método siguiente utiliza el parámetro **`_action`** para especificar una acción Change:

```
POST maxrest/rest/mbo/asset/968 HTTP/1.1
_action=Change&description=mi_nueva_descripción
```

Ejemplo: actualizar un registro que tiene varios objetos de nivel inferior

El método siguiente agrega dos líneas de orden de compra a una orden de compra, y cada línea tiene dos costes de orden de compra:

```
POST maxrest/rest/os/mxpo/1234 HTTP/1.1 description=new_po_desc&
poline.id1.polinenum=1&poline.id1.item=ABC&poline.id1.description=nueva_descripción&
poline.id1.pocost.id1-1.costlinenum=1&poline.id1.pocost.id1-1.gldebitacct=nuew_gl_acct_a&
poline.id1.pocost.id1-2.costlinenum=2&poline.id1.pocost.id1-2.gldebitacct=new_gl_acct_b&
poline.id2.polinenum=2&poline.id2.item=XYZ&poline.id2.description=nueva_descripción&
poline.id2.pocost.id2-1.costlinenum=1&poline.id2.pocost.id2-1.gldebitacct=new_gl_acct_c&
poline.id2.pocost.id2-2.costlinenum=2&poline.id2.pocost.id2-2.gldebitacct=new_gl_acct_d&
```

En el ejemplo, los identificadores identifican los parámetros siguientes:

Identificador de grupo	Parámetros identificados
id1	Parámetros que pertenecen a POLINE 1
id1-1	Parámetros que pertenecen a POCOST 1 para POLINE 1
id1-2	Parámetros que pertenecen a POCOST 2 para POLINE 1
id2	Parámetros que pertenecen a POLINE 2
id2-1	Parámetros que pertenecen a POCOST 1 para POLINE 2
id2-2	Parámetros que pertenecen a POCOST 2 para POLINE 2

Propiedades de la cabecera HTTP

Hay varias propiedades de la cabecera HTTP que son relevantes para la API de REST.

Tabla 1. Propiedades de la cabecera HTTP de la API de REST

Propiedad	Descripción
`Accept`	La proporciona el solicitante y se utiliza en la negociación de contenido para determinar el formato de la respuesta.
`Accept-Language`	Proporciona los datos en el idioma solicitado. Esta propiedad sólo se aplica a campos de aplicación que dan soporte a múltiples idiomas. Los valores que se proporcionan (como EN o FR) deben estar soportados por la aplicación. El parámetro de consulta **`_lang`** puede proporcionar el código de idioma y la misma prestación que la propiedad de cabecera **`Accept-Language`**. El parámetro de consulta **`_locale`** habilita que los números y las fechas se devuelvan en el entorno local del solicitante.
`Cache-Control`	Notifica al solicitante si está habilitado el almacenamiento en memoria caché. Si está habilitado, la propiedad **`Cache-Control`** tiene el valor `private` para asegurar que sólo el usuario actual puede reutilizar el contenido en la memoria caché. Cuando no está habilitado el almacenamiento en memoria caché, la propiedad **`Cache-Control`** tiene el valor `no-cache`. La solicitud del cliente puede contener la cabecera **`Cache-Control: no-cache`** para inhabilitar el almacenamiento en memoria caché para una solicitud específica, aunque esté habilitado para el recurso.
`Content-Length`	Contiene la longitud de la respuesta.
`Content-Type`	Notifica al solicitante del formato de la representación que se envía. Por ejemplo, puede especificarse el valor `application/xml` (para formato de respuesta XML) o `application/json` (para formato de respuesta JSON).
`ETag`	Si el almacenamiento en memoria caché está habilitado, contiene el valor Etag para el recurso que se solicita. La memoria caché del navegador del solicitante conserva el valor para futuras solicitudes para el mismo recurso.
`If-None-Match`	Si el almacenamiento en memoria caché está habilitado, contiene el valor Etag para el recurso que se ha solicitado anteriormente, de forma que la API pueda determinar si el contenido de la memoria caché puede volver a utilizarse.
`Last-Modified`	Notifica al solicitante la fecha y hora en las que el recurso se ha modificado por última vez.
`Location`	Contiene el enlace a un recurso (código HTTP 201) creado por un HTTP POST.
`_rlid`	Contiene el ID de la colección de recursos y se utiliza para desplazamiento dentro de la sesión.

Método DELETE

El método DELETE requiere el ID exclusivo del recurso.

Para suprimir un recurso de estructura de objeto, se requiere el ID del objeto principal.

Ejemplo: suprimir un activo

El método siguiente suprime el activo 1234:

```
DELETE maxrest/rest/mbo/asset/1234 HTTP/1.1
```

Simultáneo simultáneas de recursos

Se puede controlar el proceso de forma que sólo se actualice o suprima un recurso si éste no ha sido cambiado por otro usuario o aplicación después de la consulta inicial.

Si el parámetro **`_urs`** se establece en el valor `1`, la respuesta de la consulta incluye el valor de identificador de fila para el recurso.

Ejemplo XML:

```
<PO rowstamp=1234567890>
```

Ejemplo JSON:

```
{"ASSET":{"rowstamp":"1234567890","Attributes":{"ASSETNUM":{"con
tent":"1001"}
```

Ejemplo JSON en formato compacto:

```
{"ASSET":{"rowstamp":"1234567890","ASSETNUM":"1001",
```

Para un recurso de estructura de objeto, se proporciona un valor de identificador de fila para cada objeto que está en la estructura de objeto.

En una solicitud posterior para actualizar el recurso, el valor de identificador de fila se puede proporcionar utilizando el parámetro **`_rowstamp`** para recursos de objeto de negocio y de estructura de objeto, o especificando la propiedad de cabecera **`If-Match`** para objetos de negocio.

Durante el proceso, el valor de identificador de fila se compara con el identificador de fila inicial. Si los valores no coinciden, la operación de actualización o supresión falla y en la respuesta se devuelve el código de retorno HTTP 412.

Método HTTP PATCH

El método HTTP PATCH se utiliza para realizar una actualización parcial de un recurso de OSLC. A diferencia del método PUT, el método PATCH no suprime las propiedades de recurso local que no están incluidas en la petición. El método PATCH se debe especificar utilizando la cabecera x-method-override.

Se aplican las reglas siguientes cuando utiliza el método PATCH para sustituir un recurso de OSLC:

> Se actualizan todas las propiedades literales especificadas en el documento de solicitud. Las propiedades literales que no se especifiquen como parte de la solicitud no se ven afectadas de forma explícita por la infraestructura. Sin embargo, pueden verse implícitamente afectadas por la lógica empresarial que está conectada al recurso.
> Se actualizan todas las propiedades de recursos locales o se sustituyen por los valores de la propiedad correspondiente de la solicitud. Si la propiedad del recurso no está incluida en la solicitud, el recurso local correspondiente no se ve

explícitamente afectado por la infraestructura. Si la propiedad de recurso está incluida, su valor sustituye o actualiza el valor en el servidor.

Los recursos de referencia no pueden actualizarse de forma explícita. No obstante, puede actualizar las propiedades que hagan referencia al recurso, y las propiedades siguen el modelo de actualización de propiedades literales.

La implementación de PATCH tiene dos casos de uso:

Puede sustituir una propiedad de recurso local con el contenido de la petición. Este caso de uso es la implementación predeterminada.

Puede buscar y comparar los elementos de matriz de recurso contenidos en la petición con los elementos existentes en el servidor. Dependiendo de si se ha encontrado una coincidencia de búsqueda, los elementos de matriz de recurso se actualizan o insertan. Un elemento de matriz nunca se suprime de la propiedad del recurso local. Para utilizar este caso de uso, el cliente debe establecer la cabecera de petición HTTP PATCHTYPE en el valor `MERGE` (se distingue entre mayúsculas y minúsculas).

Ejemplo: Actualización de una propiedad literal

El método siguiente actualiza la propiedad de título de la orden de trabajo:

```
POST /maximo/oslc/os/oslcwodetail/abcd
x-method-override: PATCH
{
     "dcterms:title": "Check-out Leaking - Modified for Test"
}
```

A diferencia del método PUT, el método PATCH no actualiza otras propiedades de la orden de trabajo.

Ejemplo: Actualización de una propiedad de recurso

El método siguiente actualiza un registro wplabor especificado y suprime otros datos:

```
POST /maximo/oslc/os/oslcwodetail/abcd
x-method-override: PATCH
{
     "dcterms:title":  "Check-out  Leaking  -  Modified  for  Test",
"spi:wplabor": [
     {
     "spi_wm:wplaborid": "0000000067", "spi_wm:rate": 18.5
     }
]
}
```

Este método se comporta de forma similar al método PUT. El sistema busca un registro wplabor con el ID `0000000067`. Si ese registro wplabor existe, se actualiza. Si no se encuentra ninguna coincidencia, se crea un nuevo registro wplabor con el ID `0000000067`. Se suprime el resto de los registros wplabor para este recurso de orden de trabajo.

Ejemplo: Actualización y fusión de una propiedad de recurso

El método siguiente actualiza el recurso con la cabecera PATCHTYPE establecida en MERGE:

```
POST /maximo/oslc/os/oslcwodetail/abcd
x-method-override: PATCH
PATCHTYPE: MERGE
{
     "dcterms:title": "Check-out Leaking - Modified for Test",
```

```
      "spi:wplabor": [
      {
          "spi_wm:wplaborid": "0000000067", "spi_wm:rate": 18.5
       }
]
}
```

Se busca un registro wplabor cuyo ID sea `0000000067`. Si ese registro wplabor existe, se actualiza. Si no se encuentra ninguna coincidencia, se crea un nuevo registro wplabor con el ID `0000000067`. Debido a que la cabecera PATCHTYPE está establecida en `MERGE`, los demás registros wplabor de este recurso de orden de trabajo se dejan inalterados.

Ejemplo: Realización de una actualización condicional

El método siguiente actualiza el recurso si el valor de ETag es 1234567:

```
POST /maximo/oslc/os/oslcwodetail/abcd
x-method-override: PATCH
if-match: 1234567
```

Si el valor de ETag es `1234567`, se actualiza el recurso de orden de trabajo y se envía un mensaje **`HTTP 204`**.

Si el valor de ETag no es `1234567`, el servidor responde con el mensaje **`HTTP 412 Precondition failed`**. El mensaje implica que algún otro proceso ha actualizado el recurso y el cliente que realiza la petición tiene una copia obsoleta del recurso. El cliente debe ejecutar un método GET en el recurso **`abcd`** para obtener una copia nueva del recurso.