

Programación de Servicios y Procesos



Servicios RESTful CRUD

IES Nervión
Miguel A. Casado Alías
Raúl Sánchez Galán

Servicios RESTful de tipo CRUD

- **Create, Read, Update, Delete**
- Es el tipo de servicio RESTful más sencillo de diseñar
- Hace referencia a operaciones de mantenimiento de datos, normalmente sobre tablas de una BD
- Tendremos dos tipos de recursos:
 - **Colecciones:** listas o contenedores de entidades. Se suelen corresponder con una tabla de la BD
 - **Entidades:** ocurrencias o instancias concretas dentro de una colección. Se suelen corresponder con un registro de una tabla.

Tipos de recursos

Método	EndPoint	Uso
GET	<i>/service/ejemploSencilloRESTful.php/personas</i>	Obtener listado de personas 
GET	<i>/service/ejemploSencilloRESTful.php/personas/{id}</i>	Obtener una persona 

Colecciones

- Las identificaremos con una URI derivada del nombre de la entidad que contienen. Por ejemplo:

<http://www.server.com/rest/libro>
- Dicha colección representaría todos los libros del sistema
- Se suele usar la siguiente correspondencia:

Método HTTP	Operación
GET	Leer todas las entidades dentro de la colección
PUT	Actualización múltiple y/o masiva
DELETE	Borrar la colección y todas sus entidades
POST	Crear una nueva entidad dentro de la colección

Ejemplo de endpoints - Colecciones

Método	EndPoint	Uso
GET	<i>/service/ejemploSencilloRESTful.php /personas</i>	Obtener listado de personas
DELETE	<i>/service/ejemploSencilloRESTful.php /personas</i>	Borra el listado de personas
POST	<i>/service/ejemploSencilloRESTful.php /personas</i>	Crear una nueva entidad. (Enviar los datos en el body)
PUT	<i>/service/ejemploSencilloRESTful.php /personas</i>	Actualización múltiple. (Enviar los datos en el body)

Entidades

- Las identificaremos concatenando a la URI de la colección correspondiente un identificador de entidad. Por ejemplo:

<http://www.server.com/rest/libro/AF74gt0>

- Representaría al libro con identificador AF74gt0
- Se suele usar la siguiente correspondencia :

Método HTTP	Operación
GET	Leer los datos de una entidad en concreto
PUT	Actualizar una entidad existente o crearla si no existe
DELETE	Borrar una entidad en concreto
POST	Añadir información a una entidad ya existente

Ejemplo de endpoints - Entidades

Método	EndPoint	Uso
GET	<i>/service/ejemploSencilloRESTful.php /personas/{id}</i>	Obtener datos de una persona
DELETE	<i>/service/ejemploSencilloRESTful.php /personas/{id}</i>	Borra una persona
PUT	<i>/service/ejemploSencilloRESTful.php /personas/{id}</i>	Crear una nueva entidad o actualizarla. (Enviar los datos en el body)
POST	<i>/service/ejemploSencilloRESTful.php /personas</i>	Añadir información. (Enviar los datos en el body)

Lectura de recursos

- Usaremos el verbo GET tanto para leer colecciones como entidades
- Para obtener todos los miembros de una colección, hacemos GET sobre la URI de la colección. ¿Qué devolveremos? Dos opciones:
 - Lista con enlaces (URI) a todas las entidades
 - Lista de entidades con todos sus datos

```
1 GET /rest/libro HTTP/1.1
2 Host: www.server.com
3 Accept: application/json
4
```


Respuesta a GET con solo enlaces

```
1  HTTP/1.1 200 Ok
2  Content-Type: application/json; charset=utf-8
3
4  ["http://www.server.com/rest/libro/45",
5   "http://www.server.com/rest/libro/465",
6   "http://www.server.com/rest/libro/4342"]
```

Respuesta a GET con toda la información

```
1  HTTP/1.1 200 Ok
2  Content-Type: application/json;charset=utf-8
3
4  [ {
5      "id": "http://www.server.com/rest/libro/45",
6      "author": "Rober Jones",
7      "title": "Living in Spain",
8      "genre": "biographic",
9      "price": { "currency": "$", "amount": 33.2}
10 },
11 {
12     "id": "http://www.server.com/rest/libro/465",
13     "author": "Enrique Gómez",
14     "title": "Desventuras de un informático en Paris",
15     "genre": "scifi",
16     "price": { "currency": "€", "amount": 10}
17 },
18 {
19     "id": "http://www.server.com/rest/libro/4342",
20     "author": "Jane Doe",
21     "title": "Anonymous",
22     "genre": "scifi",
23     "price": { "currency": "$", "amount": 4}
24 }
```

Entidades con recursos hijos (detalles)

- A veces el volumen de información de una entidad es muy elevado y puede ser buena idea dividirla **en recursos hijos**

Por ejemplo, si de un libro queremos ofrecer además de autor, título, precio, ISBN y número de páginas, la información de todos sus capítulos, podríamos hacer una **colección de capítulos** dentro de libro, con URIs como:

- `/rest/libro/Ae3Fsr7/capitulo` información de todos los capítulos
- `/rest/libro/Ae3Fsr7/capitulo/3` nos daría info del capítulo 3

Al hacer GET sobre `/rest/libro/Ae3Fsr7` se ofrecerá la información de autor, título, precio, ISBN, número de páginas y las **URIs** de todos los capítulos del libro

Filtrar una colección

- Podemos hacer uso de query strings cuando no queramos obtener todas las entidades de una colección, sino solo las que cumplen algunos requisitos.

```
1 GET /rest/libro?precio_max=20eur&genero=scifi HTTP/1.1
2 Host: www.server.com
3 Accept: application/json
4
```

Borrar

- Se usa el método DELETE
- Al borrar una colección, se deben borrar todas sus entidades
- Al borrar una entidad, se deben borrar sus entidades hijas
- Con query strings hacemos borrado selectivo

Peticiones: Respuesta:

```
1 DELETE /rest/libro/465 HTTP/1.1
```

```
2 Host: www.server.com
```

```
3
```

```
1 HTTP/1.1 204 No Content
```

```
2
```

```
1 DELETE /rest/libro?genero=scifi HTTP/1.1
```

```
2 Host: www.server.com
```

```
3
```

Crear entidades

- Podemos crear una entidad haciendo PUT sobre una URI que no esté asociada a ninguna, y mandando los datos del nuevo recurso.
- También podríamos crearla haciendo POST sobre una colección
- POST tiene la ventaja de que la lógica de creación de URIS no recae en el cliente. Además, implica que la entidad se asocia a la colección (en PUT no tendría porqué ser así). Problema de POST: NO es idempotente.
- El servidor responde con código 201 en caso de éxito
- La cabecera Location lleva la URI del recurso creado

Ejemplo petición crear con PUT

```
1  PUT /rest/libro/465 HTTP/1.1
2  Host: www.server.com
3  Accept: application/json
4  Content-Type: application/json
5
6  {
7    "author": "Enrique Gómez Salas",
8    "title": "Desventuras de un informático en Paris",
9    "genre": "scifi",
10   "price": { "currency": "€", "amount": 50 }
11 }
```


Ejemplo respuesta petición crear con PUT

```
1  HTTP/1.1 201 Created
2  Location: http://www.server.com/rest/libro/465
3  Content-Type: application/json;charset=utf-8
4
5  {
6    "id": "http://www.server.com/rest/libro/465",
7    "author": "Enrique Gómez Salas",
8    "title": "Desventuras de un informático en Paris",
9    "genre": "scifi",
10   "price": { "currency": "€", "amount": 50 }
11 }
```


Ejemplo petición crear con POST

```
1  POST /rest/libro HTTP/1.1
2  Host: www.server.com
3  Accept: application/json
4  Content-Type: application/json
5
6  {
7    "author": "Enrique Gómez Salas",
8    "title": "Desventuras de un informático en Paris",
9    "genre": "scifi",
10   "price": { "currency": "€", "amount": 50}
11 }
```

Ejemplo respuesta petición crear con POST

```
1  HTTP/1.1 201 Created
2  Location: http://www.server.com/rest/libro/3d7ef
3  Content-Type: application/json;charset=utf-8
4
5  {
6    "id": "http://www.server.com/rest/libro/3d7ef",
7    "author": "Enrique Gómez Salas",
8    "title": "Desventuras de un informático en Paris",
9    "genre": "scifi",
10   "price": { "currency": "€", "amount": 50}
11 }
```

¿Cómo leer el body de una petición HTTP?

- `file_get_contents('php://input')`
 - Devuelve el contenido del body de una petición
- `$objectJson= json_decode($data);`
 - Devuelve un objeto, con los atributos del json