`xdrlib` — Codificar y decodificar datos XDR

Código fuente: Lib/xdrlib.py

El módulo `xdrlib` soporta el estándar de representación externa de datos (*External Data Representation Standard*) como se describe en **RFC 1014**, escrito por Sun Microsystems, Inc. en junio de 1987. Soporta la mayoría de los tipos de datos descritos en el RFC.

El módulo `xdrlib` define dos clases, una para empaquetar variables en la representación XDR, y otra para desempaquetar valores de la representación XDR. También hay dos clases de excepciones.

class `xdrlib.`**`Packer`**

`Packer` es la clase para empaquetar datos en la representación XDR. La clase `Packer` es instanciada sin argumentos.

class `xdrlib.`**`Unpacker`**(*`data`*)

`Unpacker` es la clase complementaria que desempaqueta los valores de datos XDR de un búfer de cadena. El búfer de entrada se proporciona como *data*.

Ver también

RFC 1014 - XDR: estándar de representación externa de datos

Este RFC definió la codificación de datos que era XDR en el momento en que este módulo fue escrito originalmente. Aparentemente ha quedado obsoleto, siendo reemplazado por **RFC 1832**.

RFC 1832 - XDR: estándar de representación externa de datos

El RFC más reciente que proporciona una definición revisada de XDR.

Instancias de la clase *Packer*

Las instancias de la clase `Packer` poseen los siguientes métodos:

`Packer.`**`get_buffer`**()

Retorna el búfer del paquete actual como una cadena de caracteres.

`Packer.`**`reset`**()

Restablece el búfer del paquete a la cadena de caracteres vacía.

En general, puedes empaquetar cualquiera de los tipos de datos XDR más comunes, invocando el método `pack_tipo()` apropiado. Cada método toma un solo argumento, el valor a empaquetar. Los siguientes métodos simples de empaquetado de tipos de datos son soportados: `pack_uint()`, `pack_int()`, `pack_enum()`, `pack_bool()`, `pack_uhyper()`, y `pack_hyper()`.

`Packer.`**`pack_float`**(*`value`*)

Empaqueta el número de punto flotante de precisión simple, *value*.

`Packer.`**`pack_double`**(*`value`*)

Empaqueta el número de punto flotante de doble precisión, *value*.

Los siguientes métodos soportan el empaquetado de cadenas de caracteres, bytes y datos opacos:

`Packer.`**`pack_fstring`**(*`n`*, *`s`*)

Empaqueta una cadena de caracteres de longitud fija, *s*. *n* es la longitud de la cadena de caracteres pero *no* es empaquetada en el búfer de datos. La cadena de caracteres es rellenada con bytes nulos si es necesario, para garantizar que la longitud de los datos sea un múltiplo de 4 bytes.

`Packer.`**`pack_fopaque`**(*n*, *data*)

Empaqueta un flujo de datos opaco de longitud fija, similar a `pack_fstring()`.

`Packer.`**`pack_string`**(*s*)

Empaqueta una cadena de caracteres de longitud variable, *s*. En primer lugar, la longitud de la cadena de caracteres es empaquetada como un entero sin signo, luego los datos de la cadena de caracteres son empaquetados con `pack_fstring()`.

`Packer.`**`pack_opaque`**(*data*)

Empaqueta una cadena de caracteres de datos opacos de longitud variable, similar a `pack_string()`.

`Packer.`**`pack_bytes`**(*bytes*)

Empaqueta una secuencia de bytes de longitud variable, similar a `pack_string()`.

Los siguientes métodos soportan el empaquetamiento de arreglos y listas:

`Packer.`**`pack_list`**(*list*, *pack_item*)

Empaqueta una *lista* de elementos homogéneos. Este método es útil para listas con longitud indeterminada; en otras palabras, el tamaño no está disponible hasta que se haya recorrido toda la lista. Para cada elemento de la lista, un entero sin signo `1` se empaqueta primero, seguido del valor de datos de la lista. *pack_item* es la función que se llama para empaquetar el elemento individual. Al final de la lista, se empaqueta un entero sin signo `0`.

Por ejemplo, para empaquetar una lista de enteros, el código podría parecerse a esto:

```
import xdrlib
p = xdrlib.Packer()
p.pack_list([1, 2, 3], p.pack_int)
```

`Packer.`**`pack_farray`**(*n*, *array*, *pack_item*)

Empaqueta una lista (*arreglo*) de elementos homogéneos de longitud fija. *n* es la longitud de la lista; *no*es empaquetada en el búfer, pero una excepción `ValueError` es lanzada si `len(array)` no es igual a *n*. Al igual que en el método anterior, *pack_item* es la función usada para empaquetar cada elemento.

`Packer.`**`pack_array`**(*list*, *pack_item*)

Empaqueta una *lista* de elementos homogéneos de longitud variable. Primero, la longitud de la lista es empaquetada como un entero sin signo, luego cada elemento es empaquetado como en el método `pack_farray()` de arriba.

Instancias de la clase *Unpacker*

La clase `Unpacker` ofrece los siguientes métodos:

`Unpacker.`**`reset`**(*data*)

Restablece el búfer de cadena de caracteres a la *data* especificada.

`Unpacker.`**`get_position`**()

Retorna la posición actual desempaquetada en el búfer de datos.

`Unpacker.`**`set_position`**(*position*)

Establece la posición de desempaquetado del búfer de datos en la posición *position*. Debes tener cuidado al usar los métodos `get_position()` y `set_position()`.

`Unpacker.`**`get_buffer`**()
Retorna el búfer de datos actual desempaquetado como una cadena de caracteres.

`Unpacker.`**`done`**()
Indica se ha completado el proceso de desempaquetado. Lanza una excepción `Error` si no se han desempaquetado todos los datos.

Además, cada tipo de datos que puede ser empaquetados con un `Packer`, puede ser desempaquetado con un `Unpacker`. Los métodos de desempaquetados son de la forma `unpack_tipo()`, y no tienen argumentos. Estos retornan el objeto desempaquetado.

`Unpacker.`**`unpack_float`**()
Desempaqueta un número de punto flotante de precisión simple.

`Unpacker.`**`unpack_double`**()
Desempaqueta un número de punto flotante de doble precisión, similar a `unpack_float()`.

Adicionalmente, los siguientes métodos desempaquetan cadenas de caracteres, bytes y datos opacos:

`Unpacker.`**`unpack_fstring`**(*n*)
Desempaqueta y retorna una cadena de caracteres de longitud fija. *n* es el número de caracteres esperados. Se asume un relleno con bytes nulos hasta que la longitud de los datos sea un múltiplo de 4 bytes.

`Unpacker.`**`unpack_fopaque`**(*n*)
Desempaqueta y retorna un flujo de datos opaco de longitud fija, similar a `unpack_fstring()`.

`Unpacker.`**`unpack_string`**()
Desempaqueta y retorna una cadena de caracteres de longitud variable. La longitud de la cadena de caracteres es desempaquetada primero como un entero sin signo, luego los datos de la cadena de caracteres son desempaquetados con el método `unpack_fstring()`.

`Unpacker.`**`unpack_opaque`**()
Desempaqueta y retorna un flujo de datos opaco de longitud variable, similar a `unpack_string()`.

`Unpacker.`**`unpack_bytes`**()
Desempaqueta y retorna una secuencia de bytes de longitud variable, similar a `unpack_string()`.

Los siguientes métodos soportan el desempaquetado de arreglos y listas:

`Unpacker.`**`unpack_list`**(*unpack_item*)
Desempaqueta y retorna una lista de elementos homogéneos. La lista se desempaqueta un elemento a la vez desempaquetando primero un indicador entero sin signo. Si el flag es `1`, el elemento se desempaqueta y se agrega a la lista. Un flag de `0` indica el final de la lista. *unpack_item* es la función a la que se invoca para desempaquetar los elementos.

`Unpacker.`**`unpack_farray`**(*n*, *unpack_item*)
Desempaqueta y retorna (como una lista) un arreglo de elementos homogéneos de longitud fija. *n* es el número de elementos de la lista que se esperan en el búfer. Como se mostró anteriormente, *unpack_item* es la función empleada para desempaquetar cada elemento.

`Unpacker.`**`unpack_array`**(*unpack_item*)

Desempaqueta y retorna una *lista* de elementos homogéneos de longitud variable. En primer lugar, la longitud de la lista se desempaqueta como un entero sin signo, luego cada elemento se desempaqueta como en el método `unpack_farray()` de arriba.

Excepciones

En este módulo, las excepciones se codifican como instancias de clase:

exception `xdrlib.`**`Error`**

La clase de excepción base. `Error` tiene un único atributo público, `msg`, que contiene la descripción del error.

exception `xdrlib.`**`ConversionError`**

Clase derivada de `Error`. No contiene variables de instancia adicionales.

A continuación se muestra un ejemplo de cómo detectarías una de estas excepciones:

```
import xdrlib
p = xdrlib.Packer()
try:
    p.pack_double(8.01)
except xdrlib.ConversionError as instance:
    print('packing the double failed:', instance.msg)
```