Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

Cátedra de Paradigmas de Programación

Diccionarios en Smalltalk

Autores:

- Carlos Lombardi – carlombardi@gmail.com
- Guido Vilariño – gvilarino@gmail.com

Historial de Revisiones

Versión	Revisado por	Fecha	Detalles
V 1.0	Carlos	06-2006	Versión para release.
V 1.1	Guido	04-10-2007	Corregí errores explicativos y conceptuales tanto en texto como en los ejemplos de código. Actualicé mi dirección de correo electrónico.

Diccionarios en Smalltalk

Un Dictionary es un conjunto de asociaciones entre claves y valores, donde tanto las claves como los valores son objetos cualesquiera.

Dicho de otra forma: a un Dictionary puedo pedirle que asocie un objeto valor a un objeto clave. Cada clave puede tener asociado un único objeto valor en un Dictionary; si asocio un valor a una clave que ya tenía otro valor asociado, reemplaza el valor viejo por el nuevo.

Para esto se le envía al Dictionary el mensaje `at:put:` que me diga qué objeto tiene asociado a una clave. Si no tiene ninguno, da error... a menos que se lo pida de una forma que evita el error.

La forma standard de pedir el valor asociado a una clave es enviarle al Dictionary el mensaje `at:`; si quiero especificar qué quiero que haga si no encuentra la clave, uso `at:ifAbsent:`.

Un Dictionary es una Collection, entonces los mensajes comunes a todas las Collections se los voy a poder enviar a un Dictionary.

Para estos mensajes, se toman sólo los valores, p.ej. si pregunto `includes:` estoy preguntando si contiene un valor, y en `select:` / `collect:` / etc. el parámetro va a ser un valor. A pesar de esto, el resultado del `select:` será un Dictionary, para los valores para los que el bloque dé true va a incluir el par clave/valor correspondiente.

Un ejemplo de todo esto: un sencillo diccionario de traducción de palabras. De esta forma creo un Dictionary y le pongo algunas asociaciones

```
dct := Dictionary new.
dct at: 'uno' put: 'one'.      "Clave 'uno', valor 'one'"
dct at: 'dos' put: 'two'.      "Clave 'dos', valor 'two'"
dct at: 'tres' put: 'three'.
dct at: 'cuatro' put: 'four'.
```

Le pido a dct un elemento

```
dct at: 'tres'                          "devuelve 'three'"
dct at: 'cinco'                               "tira error"
dct at: 'cinco' ifAbsent: ['no se']    "devuelve 'no se'"
dct at: 'tres'  ifAbsent: ['no se']    "devuelve 'three'"
```

Le pido cosas de Collection

```
dct size.              "devuelve 4"
dct includes: 'dos'    "false, trabaja sobre los valores"
dct includes: 'two'    "true"
dct select: [:pal | pal size = 4]
    "a Dictionary('cuatro' -> 'four'), Analiza sobre los
 valores pero devuelve un nuevo diccionario respetando las
claves"
```

Ahora, ¿qué pasa si queremos usar las claves, o necesitamos trabajar con los pares clave/valor? Vamos de a poco.

Si a un Dictionary le envío el mensaje keys me devuelve un Set con las claves, en este caso:

```
dct keys "devuelve a Set('uno' 'dos' 'cuatro' 'tres')"
```

Este es un nuevo objeto Set, al que le puedo hacer/preguntar lo que quiera, obviamente que si lo modifico el diccionario no se ve afectado. P.ej.

```
dct keys select: [:pal | pal size = 4]
                                    "devuelve a Set('tres')"
```

Ahora veamos cómo trabajar con claves y valores a la vez.

Primero me pregunto, si quiero trabajar con los pares clave/valor, es probable que tenga que representar cada par como un objeto. ¿Qué objetos van a ser estos?

Respuesta: van a ser instancias de la clase Association. Una Association es un par clave/valor, entiende los mensajes key y value.

Si a un Dictionary le envío el mensaje associations, va a devolver una OrderedCollection con sus pares clave/valor representados mediante Associations, p.ej.

```
dct associations
```

devuelve una OrderedCollection con los pares: 'uno' -> 'one' / 'dos' -> 'two' / 'cuatro' -> 'four' / 'tres' -> 'three' (el printString de Association es clave -> valor)

Entonces, si obtengo uno de estos pares, p.ej.

```
dct associations first
```

obtengo un objeto al que le puedo pedir key y value. P.ej.

```
dct associations first key     "respuesta posible: 'uno'"
dct associations first value   "respuesta posible: 'one'"
```

Otra vez, esta es una colección a la que puedo hacerle cualquier cosa, p.ej.

```
dct associations select:
             [:assoc | assoc key size = assoc value size]
```

Devuelve

```
an OrderedCollection('uno' -> 'one' 'dos' -> 'two')
```

Terminamos con otro ejemplo, una implementación de un depósito que se acuerda del stock de cada artículo usando un diccionario donde la clave es el artículo y el valor es la cantidad de unidades del artículo en el depósito.

#Depósito
variables
artículos: Dictionary

métodos

```
cuantoTenésDe: unArtículo
    ^artículos at: unArtículo

pone: cant de: unArtículo
    artículos at: unArtículo
              put: (artículos at: unArtículo) + cant

saca: cant de: unArtículo
    “Análogo anterior.”

cantidadTotalDeUnidades
    ^artículos inject: 0 into: [:x :elem | x + elem].
    "trabaja sobre los valores, que son las cantidades"

valorTotalDeposito
    ^artículos associations
         inject: 0
         into: [:x :assoc |
                   x + assoc key precio * assoc value].
    "assoc key: el artículo, le pido el precio
     assoc value: la cantidad de ese artículo en el
depósito"
```