【コレクションの使い方】

【要素へのアクセス】

サンプルコード

```
{value
    ||変数を定義する
    let my-array:{Array-of String} =
        {new {Array-of String}, "Apple", "Banana", "Cherry"}

    ||実行結果を確認してください
    my-array[2]
}
```

実行結果

```
Cherry
```

【要素へのアクセス(for)】

サンプルコード

```
{value
    ||変数（配列）を定義する
    let my-array:{Array-of String} =
        {new {Array-of String}, "Apple", "Banana", "Cherry"}

    ||変数を定義する
    let message:VBox = {VBox}

    ||コンテナループにより操作する
    {for element:String in my-array do
        {message.add element}
    }

    ||実行結果を確認してください
    message
}
```

実行結果

```
Apple
Banana
Cherry
```

【要素へのアクセス】

サンプルコード

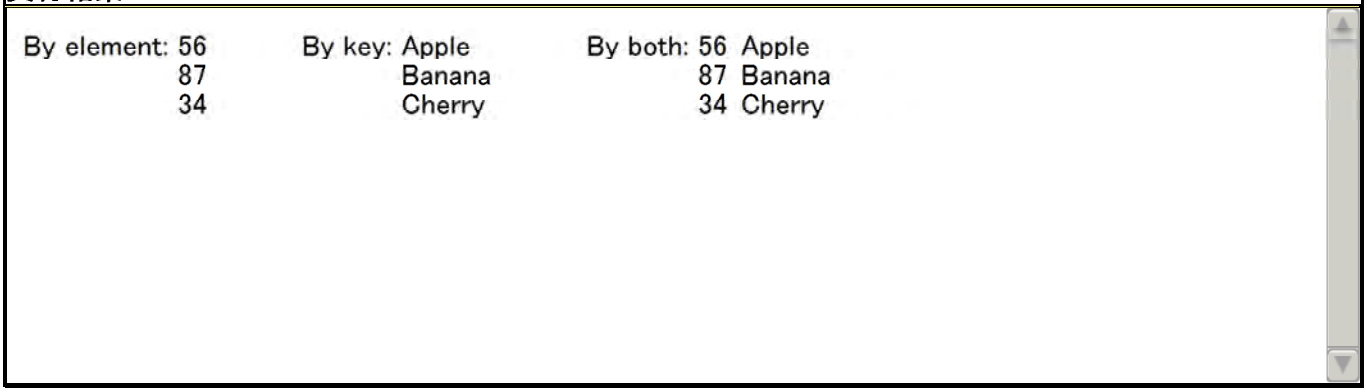
```
||以下のコードはハッシュテーブルの要素へのアクセスサンプルです
{value
    ||変数を定義する
    let my-hash-table:{HashTable-of String, int} =
        {new {HashTable-of String, int},
            "Apple", 56,
            "Banana", 87,
            "Cherry", 34
        }
    let elements:VBox = {VBox}
    let keys:VBox = {VBox}
    let both:VBox = {VBox}

    ||ハッシュテーブルを操作する
    {for element:int in my-hash-table do
        {elements.add element}
    }

    {for key k:String in my-hash-table do
        {keys.add k}
    }

    {for e:int key k:String in my-hash-table do
        {both.add
            {HBox
                spacing = 5pt,
                e,
                k
            }
        }
    }
    ||実行結果を確認してください
    {spaced-hbox
        "By element: ",
        elements,
        {Fill width = 1cm},
        "By key: ",
        keys,
        {Fill width = 1cm},
        "By both: ",
        both
    }
}
```

実行結果

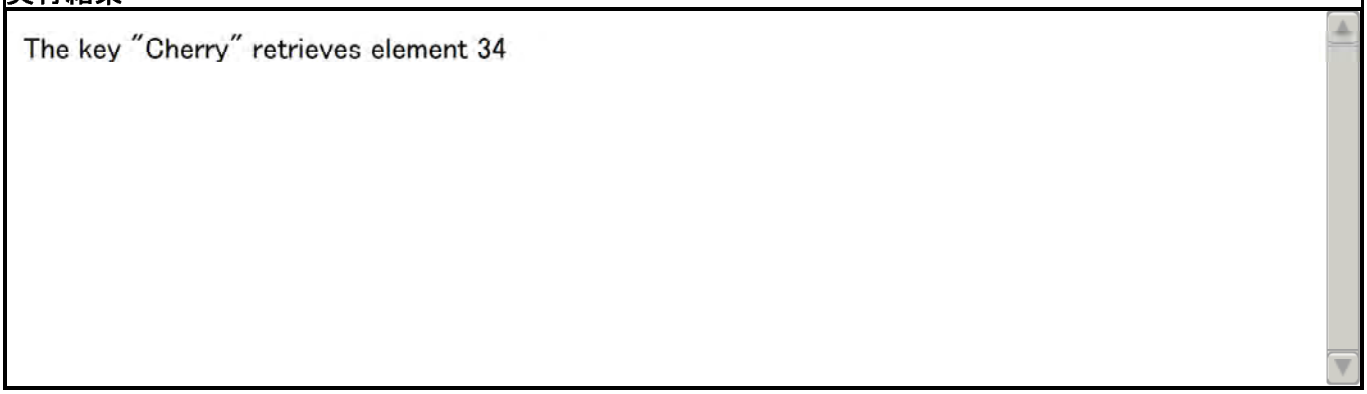


【要素へのアクセス(key)】

サンプルコード

```
||ハッシュテーブルの要素へのアクセスサンプルです
||キーを与えてアクセスします
{let my-hash-table:{HashTable-of String, int} =
    {new {HashTable-of String, int},
        "Apple", 56,
        "Banana", 87,
        "Cherry", 34
    }
}
{let output:#Frame}
{let k:String = "Cherry"}
{let (e:int, exists?:bool) = {my-hash-table.get-if-exists k}}
{if exists? then
    set output =
        {Frame
            {text The key "{value k}" retrieves element {value e}}
        }
 else
    let output =
        {Frame
            {text The key "{value k}" is not found}
        }
}
{value output}
```

実行結果



【要素へのアクセス(keyが存在するか否かチェック)】

サンプルコード

```
{value
    ||ハッシュテーブルの要素へのアクセスサンプルです
    ||キーが存在するかどうかを確認します
    let my-hash-table:{HashTable-of String, int} =
        {new {HashTable-of String, int},
            "Apple", 56,
            "Banana", 87,
            "Cherry", 34
        }
    {let my-key:String = "Orange"}
    let (k:#String, exists?:bool) = {my-hash-table.get-key-if-exists my-key}
    {if not exists? then
        {my-hash-table.set my-key, 62}
    }
    let keys:VBox = {VBox}
    {for key k:String in my-hash-table do
        {keys.add k}
    }
    keys
}
```

実行結果

```
Orange
Apple
Banana
Cherry
```

【要素の追加】

サンプルコード

```
||以下のコードはハッシュテーブルの要素へ追加するサンプルです
{let my-hash-table:{HashTable-of String, int} =
    {new {HashTable-of String, int},
        "Apple", 56,
        "Banana", 87,
        "Cherry", 34
    }
}
{my-hash-table.set "Orange", 62}
{text You added element {my-hash-table.get "Orange"}
    with key "Orange"}
```

実行結果

```
You added element 62 with key "Orange"
```

【要素の削除】

サンプルコード

```
||以下のコードはハッシュテーブルの要素から削除するサンプルです
{let my-hash-table:{HashTable-of String, int} =
    {new {HashTable-of String, int},
        "Apple", 56,
        "Banana", 87,
        "Cherry", 34
    }
}
{let keys:{Array-of String} = {new{Array-of String}}}
{for key k:String in my-hash-table do
    {keys.append k}
}

||ボタン押下により削除します
||結果は実際にコードを書いて実行してください
{CommandButton
    label = "Empty Hash Table",
    {on Action do
        {for key:String in keys do
            {my-hash-table.remove key}
        }
        {popup-message
            {text Size of hash table: {value my-hash-table.size}}
        }
    }
}
```

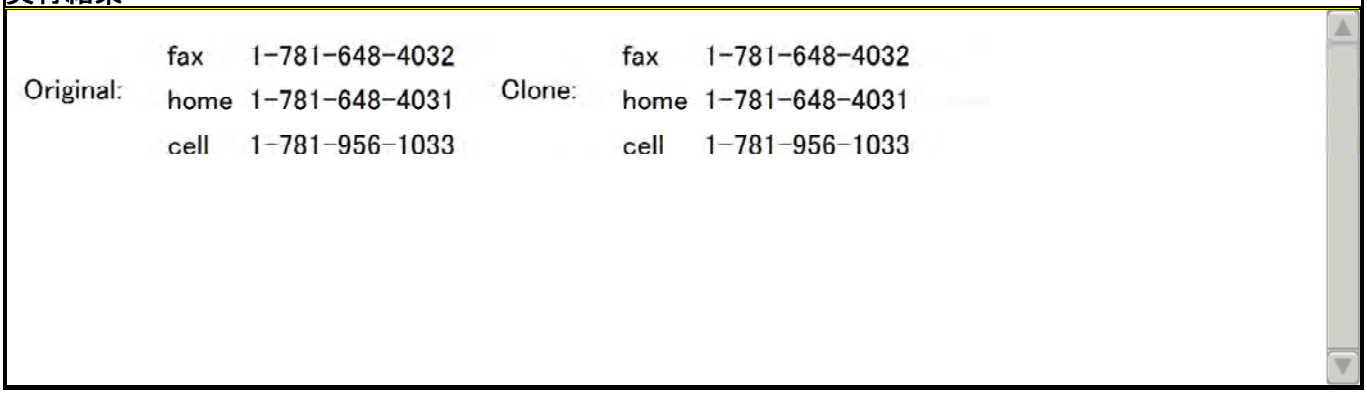
【ハッシュ テーブルのクローン作成】1

サンプルコード

```
||以下のコードはハッシュテーブルの要素変更に関するサンプルです
||詳細はCurl開発者ガイドを確認してください
{let my-hash-table:{HashTable-of String, StringBuf} =
    {new {HashTable-of String, StringBuf},
        "home", {StringBuf "781-648-4031"},
        "cell", {StringBuf "781-956-1033"},
        "fax", {StringBuf "781-648-4032"}
    }
}
{let new-hash-table:{HashTable-of String, StringBuf} =
    {my-hash-table.clone}
}
{let table1:Table = {Table columns = 2}}
{let table2:Table = {Table columns = 2}}

||オリジナルのハッシュ テーブルとクローンの両方が指し示す基本データを変更する場合、
||変更は両方のテーブルに対して行われます。
{for e:StringBuf key k:String in new-hash-table do
    {e.insert '1', 0}
    {e.insert '-', 1}
}
{for e:StringBuf key k:String in my-hash-table do
    {table1.add k}
    {table1.add e}
}
{for e:StringBuf key k:String in new-hash-table do
    {table2.add k}
    {table2.add e}
}
||実行結果を確認してください
{HBox
    "Original:",
    {Fill width = .5cm},
    table1,
    {Fill width = .5cm},
    "Clone:",
    {Fill width = .5cm},
    table2
}
```

実行結果



【ハッシュテーブルのクローン作成】2

サンプルコード

```
||以下のコードは値が50を超える要素をフィルタするサンプルです
{value
    let my-hash-table:{HashTable-of String, int} =
        {new {HashTable-of String, int},
            "Apple", 56,
            "Banana", 87,
            "Cherry", 34
        }
    ||HashTable-of.filter-clone は、
    ||要素の値をフィルタするプロシージャによって返された要素を含むクローンを作成します
    let new-hash-table:{HashTable-of String, int} =
        {my-hash-table.filter-clone
            {proc {element:int}:bool
                {return element > 50}
            }
        }
    let box1:VBox = {VBox}
    let box2:VBox = {VBox}

    ||元々の内容を表示する
    {for e:int key k:String in my-hash-table do
        {box1.add
            {HBox spacing = 5pt, e, k}
        }
    }
    ||フィルタした結果を表示する
    {for e:int key k:String in new-hash-table do
        {box2.add
            {HBox spacing = 5pt, e, k}
        }
    }
    ||実行結果を確認してください
    {HBox
        box1,
        {Fill width = .5cm},
        box2
    }
}
```

実行結果

```
56 Apple     56 Apple
87 Banana    87 Banana
34 Cherry
```

【ハッシュテーブルのクローン作成】3

サンプルコード

```
||文字 'B' で始まるキーをフィルタして取り除くサンプルです
{value
    let my-hash-table:{HashTable-of String, int} =
        {new {HashTable-of String, int},
            "Apple", 56,
            "Banana", 87,
            "Cherry", 34
        }
    ||HashTable-of.filter-keys-clone は、
    ||キーの値をフィルタするプロシージャによって返された要素を含むクローンを作成します
    let new-hash-table:{HashTable-of String, int} =
        {my-hash-table.filter-keys-clone
            {proc {key:String}:bool
                {return key[0] != 'B'}
            }
        }
    let box1:VBox = {VBox}
    let box2:VBox = {VBox}

    ||元々の内容を表示する
    {for e:int key k:String in my-hash-table do
        {box1.add
            {HBox spacing = 5pt, e, k}
        }
    }

    ||フィルタした結果を表示する
    {for e:int key k:String in new-hash-table do
        {box2.add
            {HBox spacing = 5pt, e, k}
        }
    }

    ||実行結果を確認してください
    {HBox
        box1,
        {Fill width = .5cm},
        box2
    }
}
```

実行結果

```
56 Apple     56 Apple
87 Banana    34 Cherry
34 Cherry
```

【ハッシュテーブルのクローン作成】4

サンプルコード

```
||HashTable-of.set を使用して、キーの値 "Frank" に関連付けられた電話番号を置き換えるサンプルです
{let my-hash-table:{HashTable-of String, StringBuf} =
    {new {HashTable-of String, StringBuf},
        "John", {StringBuf "781-555-1234"},
        "Paul", {StringBuf "781-555-5678"},
        "Frank", {StringBuf "781-555-4321"}
    }
}
{let new-hash-table:{HashTable-of String, StringBuf} =
    {my-hash-table.clone}
}
{let table1:Table = {Table columns = 2}}
{let table2:Table = {Table columns = 2}}
{let sb:StringBuf = {StringBuf "781-555-9999"}}
{my-hash-table.set "Frank", sb}

||元々の内容を表示する
{for e:StringBuf key k:String in my-hash-table do
    {table1.add k}
    {table1.add e}
}

||置き換えた結果を表示する
{for e:StringBuf key k:String in new-hash-table do
    {table2.add k}
    {table2.add e}
}

||実行結果を確認してください
{HBox
    "Original:",
    {Fill width = .5cm},
    table1,
    {Fill width = .5cm},
    "Clone:",
    {Fill width = .5cm},
    table2
}
```

実行結果

Original:	John	781-555-1234	Clone:	John 781-555-1234
	Paul	781-555-5678		Paul 781-555-5678
	Frank	781-555-9999		Frank 781-555-4321

【セット 要素へのアクセス】

サンプルコード

```
||以下のコードは、セットの要素へアクセスするためのサンプルです
||セットに関する詳細は、Curl開発者ガイドを確認してください
{value
    let my-set:{Set-of String} =
        {new {Set-of String},
            "Apple",
            "Banana",
            "Cherry"
        }

    let message:VBox = {VBox}

    {for element:String in my-set do
        {message.add element}
    }

    ||実行結果を確認してください
    message
}
```

実行結果

```
Apple
Banana
Cherry
```