【正規表現とは】

【特殊文字のエスケープ】1

サンプルコード

```
||アプレットでの正規表現
||正規表現の詳細は、Curl開発者ガイドを確認してください
{import * from CURL.LANGUAGE.REGEXP}
{let s:String = "abc ... def"}
{let exp:String = "¥¥."}
{set s = {regexp-subst
              exp,
              s,
              ""
              ,
              replace-all? = true}}

{value s}
```

実行結果

```
abc def
```

【特殊文字のエスケープ】2

サンプルコード

```
||以下のコードでは逐語的文字列を使って文字列と正規表現に予約文字を含めています
{import * from CURL.LANGUAGE.REGEXP}

{let s:String = |"abc ||| def"|}
{let exp:String = |"¥|"|}
{set s = {regexp-subst
              exp,
              s,
              ""
              ,
              replace-all? = true}}
{value s}
```

実行結果

```
abc def
```

【部分文字列に名称を付ける】

サンプルコード

```
||以下のコードは、Matcher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
{import * from CURL.LANGUAGE.REGEXP}
{value
    let matcher:Matcher = {Matcher |"<(?P<name>.*)>"|}
    {if-non-null state = {matcher.match "<foo>"} then
        state["name"]
     else
        "NO-MATCH"
    }
}
```

実行結果

```
foo
```

【単純なマッチング】

サンプルコード

```
||単純な文字列のマッチングサンプルです
{import * from CURL.LANGUAGE.REGEXP}
{regexp-match? "foo", "Hello foobar goodbye"}
```

実行結果

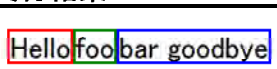
```
true
```

【部分文字列の抽出】

サンプルコード

```
||regexp-match? プロシージャを使用したサンプルです
||詳細は、Curl開発者ガイドを確認してください
{import * from CURL.LANGUAGE.REGEXP}
{value
    let beg:#String, mid:#String, end:#String
    {if {regexp-match? "^(.*)(foo)(.*)$", "Hello foobar goodbye",
            beg, mid, end} then
        {HBox {HBox border-width=1pt, border-color="red", beg},
            {HBox border-width=1pt, border-color="green", mid},
            {HBox border-width=1pt, border-color="blue", end}}
     else
        "No match"}}
```

実行結果



【ソース コードからの検索 / 置換】(replace-all?)

サンプルコード

```
||regexp-substプロシージャのオプション replace-all? を指定した際の挙動を示すサンプルです
||詳細は、Curl開発者ガイドを確認してください
{import * from CURL.LANGUAGE.REGEXP}
{regexp-subst "dbl-(.)", "dbl-x dbl-y dbl-z", "¥¥1¥¥1",
    replace-all?=true}
```

実行結果

```
xx yy zz
```

【ソース コードからの検索 / 置換】(?P=name) シンタックス

サンプルコード

```
||(?P=name) シンタックスを使用したサンプルです
{import * from CURL.LANGUAGE.REGEXP}
{regexp-subst
    |"(?P<year>¥d¥d¥d¥d)-(?P<month>¥d¥d)-(?P<day>¥d¥d)"|,
    "2003-10-11",
    |"(?P=month)/(?P=day)/(?P=year)"|
}
```

実行結果

```
10/11/2003
```

【Matcher と MatchState】

サンプルコード

```
||Matcherを使ったサンプルです
||詳細はCurl開発者ガイドを確認してください
{import * from CURL.LANGUAGE.REGEXP}
{value
    let my-string:String = "Curl is great, I like Curl"
    let m:Matcher = {Matcher "^Curl.*Curl$"}
    {if {regexp-match? m, my-string} then
        {text Match}
     else
        {text No match}
    }
}
```

実行結果

```
Match
```

【リフレクション】

【パッケージと PackageMembers】1

サンプルコード

```
||パッケージへアクセスするためのサンプルコードです
||詳細はCurl開発者ガイドを確認してください
{value
    let current:VBox = {VBox}
    {current.add
        {HBox
            |"{get-current-package} : "|,
            {String {get-current-package}}
        }
    }
    let imports:VBox = {VBox}
    {for p in {get-current-package}.imported-packages do
        {imports.add {String p}}
    }
    {current.add
        {HBox
            |"{get-current-package}.imported-packages : "|,
            imports
        }
    }
    current
}
```

実行結果

```
{get-current-package} :[OpenPackage @0x39BED004 <unnamed>]
{get-current-package}.imported-packages :[Package @0x02F11E00 CURL.IMPLICIT.APPLET]
                                         [Package @0x02EDF930 CURL.LANG]
```

【パッケージと PackageMembers】2

サンプルコード

```
||以下のコードはPackageMemberを利用したサンプルです
{import * from CURL.LANGUAGE.REFLECTION}
{value
    let p:Package =
        {import-package
            {ComponentSelector name = "CURL.RUNTIME.PROCESS"}
        }
    let m:#PackageMember = {p.get-member "get-current-package"}
    {table
        {row
            {cell {String m & ".access"}}
            {cell {String m.access}}
        }
        {row
            {cell {String m & ".constant?"}}
            {cell {String m.constant?}}
        }
        {row
            {cell {String m & ".name"}}
            {cell {String m.name}}
        }
        {row
            {cell {String m & ".package"}}
            {cell {String m.package}}
        }
        {row
            {cell {String m & ".public?"}}
            {cell {String m.public?}}
        }
        {row
            {cell {String m & ".type"}}
            {cell {String m.type}}
        }
    }
}
```

実行結果

```
[PackageMember @0x35A5226C get-current-package].access      BindingAccess.public
[PackageMember @0x35A5226C get-current-package].constant?   true
[PackageMember @0x35A5226C get-current-package].name        get-current-package
[PackageMember @0x35A5226C get-current-package].package     [Package @0x0023D0D4 CURL.RUNTIME.PROCESS]
[PackageMember @0x35A5226C get-current-package].public?     true
[PackageMember @0x35A5226C get-current-package].type        {proc-type {}:Package}
```

【パッケージと PackageMembers】3

サンプルコード

```
||以下はPackageMemberを使ってプロシージャを呼び出すサンプルです
{import * from CURL.LANGUAGE.REFLECTION}

{value
    let p:Package =
        {import-package
            {ComponentSelector name = "CURL.RUNTIME.PROCESS"}
        }
    let m:#PackageMember = {p.get-member "get-current-package"}
    {assert m.type == {type-of get-current-package}}
    {assert {m.get-value} == get-current-package}
    {assert {{m.get-value}} == {get-current-package}}
    {table
        {row
            {cell direct}
            {cell reflection}
        }
        {row
            {cell get-current-package}
            {cell {String m.name}}
        }
        {row
            {cell {type-of get-current-package}}
            {cell {String m.type}}
        }
        {row
            {cell {value get-current-package}}
            {cell {m.get-value}}
        }
        {row
            {cell {get-current-package}}
            {cell {{m.get-value}}}
        }
    }
}
```

実行結果

direct	reflection
get-current-package	get-current-package
[proc-type []:Package]	[proc-type []:Package]
[proc get-current-package]	[proc get-current-package]
[OpenPackage @0x1F8DC004 <unnamed>]	[OpenPackage @0x1F8DC004 <unnamed>]

【PackageMember クラス】名前指定

サンプルコード

```
||名前を指定してPackageMemberを選択するサンプルです
{value
    let p:Package =
        {import-package
            {ComponentSelector name = "CURL.LANGUAGE.REFLECTION"}
        }
    let m:#PackageMember =
        {p.get-member "PackageMember", check-imports? = false}
    {table
        {row
            {cell {String m & ".access"}}
            {cell {String m.access}}
        }
        {row
            {cell {String m & ".constant?"}}
            {cell {String m.constant?}}
        }
        {row
            {cell {String m & ".name"}}
            {cell {String m.name}}
        }
        {row
            {cell {String m & ".package"}}
            {cell {String m.package}}
        }
        {row
            {cell {String m & ".public?"}}
            {cell {String m.public?}}
        }
        {row
            {cell {String m & ".type"}}
            {cell {String m.type}}
        }
    }
}
```

実行結果

```
[PackageMember @0x17DA9474 PackageMember].access        BindingAccess.public
[PackageMember @0x17DA9474 PackageMember].constant?     true
[PackageMember @0x17DA9474 PackageMember].name          PackageMember
[PackageMember @0x17DA9474 PackageMember].package       [Package @0x0023BE94 CURL.LANGUAGE.REFLECTION]
[PackageMember @0x17DA9474 PackageMember].public?       true
[PackageMember @0x17DA9474 PackageMember].type          ClassType
```

【Type、ProcType および ClassType】

サンプルコード

```
||以下のコードではClassMemberのクラスメンバを調べています
{import * from CURL.LANGUAGE.REFLECTION}

{value
    let t:ClassType = ClassMember
    let members:Table =
        {Table
            {row-prototype
                {cell-prototype t}
            },
            {row-prototype
                {cell-prototype "member"},
                {cell-prototype "name"}
            }
        }
    {for m in {t.get-members search-superclasses? = false} do
        {members.add
            {row-prototype
                {cell-prototype m},
                {cell-prototype m.name}
            }
        }
    }
    members
}
```

実行結果

ClassMember

member	name
[ClassProc @0x130685CC]	get-all-filter
[ClassProc @0x130685D4]	get-property-filter
[ClassProc @0x130685DC]	set-property-filter
[ClassProc @0x130685E4]	instance-maker-filter
[ClassProc @0x130685EC]	field-filter
[ClassProc @0x130685F4]	getter-filter
[ClassProc @0x130685FC]	setter-filter
[ClassProc @0x13068604]	method-filter
[ClassProc @0x1306860C]	option-filter
[ClassProc @0x13068614]	class-proc-filter
[ClassProc @0x1306861C]	class-variable-filter
[Getter @0x11FB2CB8]	declaring-class
[Getter @0x11FB2CC4]	defining-class
[Getter @0x11FB2CD0]	name
[Getter @0x11FB2CDC]	type
[Getter @0x11FB2CE8]	access
[Getter @0x11FB2CF4]	public?

【クラス変数】

サンプルコード

```
||以下のコードではClassVariableのクラスメンバを調べています
{import * from CURL.LANGUAGE.REFLECTION}

{value
    let t:ClassType = ClassVariable
    let members:Table =
        {Table
            {row-prototype
                {cell-prototype t}
            },
            {row-prototype
                {cell-prototype "member"},
                {cell-prototype "name"},
                {cell-prototype "type"}
            }
        }
    {for m in {t.get-members search-superclasses? = false} do
        {members.add
            {row-prototype
                {cell-prototype m},
                {cell-prototype m.name},
                {cell-prototype m.type}
            }
        }
    }
    members
}
```

実行結果

```
ClassVariable
member                  name        type
[Getter @0x10B16A24]    public-get? bool
[Getter @0x10B16AA8]    public-set? bool
[Method @0x4D878354]    get-value   {proc-type {}:any}
[Method @0x4D87835C]    set-value   {proc-type {any}:void}
```

【プロパティ】

サンプルコード

```
||以下のコードではPropertyのクラスメンバを調べています
{import * from CURL.LANGUAGE.REFLECTION}

{value
   let t:ClassType = Property
   let members:Table =
       {Table
           {row-prototype
               {cell-prototype t}
           },
           {row-prototype
               {cell-prototype "member"},
               {cell-prototype "name"},
               {cell-prototype "type"}
           }
       }
   {for m in {t.get-members search-superclasses? = false} do
       {members.add
           {row-prototype
               {cell-prototype m},
               {cell-prototype m.name},
               {cell-prototype m.type}
           }
       }
   }
   members
}
```

実行結果

Property

member	name	type
[Method @0x2BBC278C]	get-value	{proc-type {any}:any}
[Method @0x2BBC2794]	set-value	{proc-type {any, any}:void}
[Getter @0x24A12DE4]	access-for-get	BindingAccess
[Getter @0x24A12DF0]	access-for-set	BindingAccess
[Getter @0x24A12DFC]	public-get?	bool
[Getter @0x24A12E08]	public-set?	bool

【コンストラクタとファクトリー】

サンプルコード

```
||以下のコードではリフレクションを使ってオブジェクトを作成します
{import * from CURL.LANGUAGE.REFLECTION}

{value
    let t:ClassType = String
    let members:Table =
        {Table
            {row-prototype
                {cell-prototype t}
            },
            {row-prototype
                {cell-prototype "member"},
                {cell-prototype "name"},
                {cell-prototype "type"},
                {cell-prototype |"{f.new 'X', 3}"|},
                {cell-prototype |"{type-of {f.new 'X', 3}}"|}
            }
        }
    {for m in {t.get-members
                search-superclasses? = false,
                filter = ClassMember.instance-maker-filter
              } do
        let f:Factory = m asa Factory
        {members.add
            {row-prototype
                {cell-prototype f},
                {cell-prototype f.name},
                {cell-prototype f.type},
                {cell-prototype {f.new 'X', 3}},
                {cell-prototype {type-of {f.new 'X', 3}}}
            }
        }
    }
    members
}
```

実行結果

String

member	name	type	{f.new 'X', 3}	{type-of {f.new 'X', 3}}
[Factory @0x20648CCC]	default	{proc-type {...:any}:String}	X3	String
[Factory @0x20648CD4]	repeat-char	{proc-type {char, int}:String}	XXX	String

【メソッド】1

サンプルコード

```
||以下のコードではリフレクションを使ったメソッドを呼び出します
{import * from CURL.LANGUAGE.REFLECTION}
{table
    {row
        {cell direct:}
        {cell
            {{String 'X', 3}.equal?
                {String.repeat-char 'x', 3},
                ignore-case? = true
            }
        }
    }
    {row
        {cell reflection:}
        {cell
            {{(String asa ClassType).get-method "equal?"}.invoke
                {{(String asa ClassType).get-instance-maker "default"}.new
                    'X', 3
                },
                {{(String asa ClassType).get-instance-maker "repeat-char"}.new
                    'x', 3
                },
                ignore-case? = true
            }
        }
    }
}
```

実行結果

```
direct:     false
reflection: false
```

【メソッド】2

サンプルコード

```
||以下のコードではMethodのメソッドを調べています
{import * from CURL.LANGUAGE.REFLECTION}

{value
    let t:ClassType = Method
    let members:Table =
        {Table
            {row-prototype
                {cell-prototype t}
            },
            {row-prototype
                {cell-prototype "member"},
                {cell-prototype "name"},
                {cell-prototype "type"}
            }
        }
    {for m in {t.get-members
                search-superclasses? = false,
                filter = ClassMember.method-filter
              } do
        {members.add
            {row-prototype
                {cell-prototype m},
                {cell-prototype m.name},
                {cell-prototype m.type}
            }
        }
    }
    members
}
```

実行結果

```
Method
member                  name      type
[Method @0x0FDB982C]  invoke-n  {proc-type {any, ...}:{FastArray-of any}}
[Method @0x0FDB9834]  invoke    {proc-type {any, ...}:(any, any, any, any)}
```