Block Closures in Pharo

Beuchey Anthony, Corbau Martin, Sasu Daniel

Définition

- Les blocs sont des objets anonymes qui décrivent une méthode.

```
* [ 1 + 2 ].
```

Définition

- Les blocs sont des objets anonymes qui décrivent une méthode.

```
* [ 1 + 2 ].
* [ 5 + 2 ] value.
                → 7
```

Définition

- Les blocs sont des objets anonymes qui décrivent une méthode.

```
* [ 1 + 2 ].
* [ 5 + 2 ] value.
                → 7
* [:x|x + 2] value:7.
                → 9
```

Syntaxe

- `[:argument(s) | méthode]`

```
[:x|x + 2] value:7. → 9
```

Syntaxe

- `[:argument(s) | méthode]`

```
[:x|x + 2] value:7. → 9

[:x :y | x + y ] value:7 value:5. → 12
```

Syntaxe

- `[:argument(s) | méthode]`

```
[:x|x + 2] value:7. → 9

[:x :y | x + y ] value:7 value:5. → 12

[:x :y :z | x + y. x-z] value:7 value:5 value:1. → 6
```

Syntaxe

- `[:argument(s) | méthode]`

```
[:x|x + 2] value:7. → 9

[:x :y | x + y ] value:7 value:5. → 12

[:x :y :z | x + y. x-z] value:7 value:5 value:1. → 6
```

=> Retourne la valeur de la **dernière instruction**

Utilisation

- **Structures de contrôle :**

```
(5 < 5)
```

Utilisation

- Structures de contrôle :

```
(5 < 5)
    ifTrue
```

Utilisation

- **Structures de contrôle :**

```
(5 < 5)
    ifTrue: [ 'True' ]
```

Utilisation

- **Structures de contrôle :**

```
(5 < 5)
    ifTrue: [ 'True' ]
    ifFalse: [ 'False' ].
```

Utilisation

- Structures de contrôle :

```
(5 < 5)
    ifTrue: [ 'True' ]
    ifFalse: [ 'False' ].
→ 'False'
```

Utilisation

- Structures de contrôle :

```
(5 < 5)
    ifTrue: [ 'True' ]
    ifFalse: [ 'False' ].
→ 'False'
```

- Boucles :

```
n := 1.
[ n < 10 ] whileTrue: [ n := n + 1 ].
n. → 10
```

Utilisation

- Structures de contrôle :

```
(5 < 5)
    ifTrue: [ 'True' ]
    ifFalse: [ 'False' ].
→ 'False'
```

- Boucles :

```
n := 1.
[ n < 10 ] whileTrue: [ n := n + 1 ].
n. → 10
```

- Iterateurs :

```
n:=1.
(1 to: 10) do: [:each | n:=n+each].
 n. → 56
```

Utilisation

- Structures de contrôle :

```
(5 < 5)
    ifTrue: [ 'True' ]
    ifFalse: [ 'False' ].
→ 'False'
```

- Boucles :

```
n := 1.
[ n < 10 ] whileTrue: [ n := n + 1 ].
n. → 10
```

- Iterateurs :

```
n:=1.
(1 to: 10) do: [:each | n:=n+each].
 n. → 56

#(2 -3 4 -35 4) collect: [:each | each abs ].
→ #(2 3 4 35 4)
```

Avancé

- Nommer un bloc :

Avancé

- **Nommer un bloc :**

  |a|.

Avancé

- **Nommer un bloc :**

```
|a|.
a := [:x | x+1].
```

Avancé

- **Nommer un bloc :**

```
|a|.
a := [:x | x+1].
…
…
a value:7. → 8
```

Avancé

- **Nombre d'arguments :**

  - `[:x|x + 2] cull:7 cull:3. → 9`

    ```
    [:x :y | x + y] cull:2.
              → Error: This block accepts 2 arguments,
                but was called with 1 argument
    ```

Avancé

- **Nombre d'arguments :**

  - `[:x|x + 2] cull:7 cull:3. → 9`

    ```
    [:x :y | x + y] cull:2.
              → Error: This block accepts 2 arguments,
                but was called with 1 argument
    ```

  - `[:x :y :z :w :v| x + y] value:2 value:3 value:4 value:5 value:6.`
    `→ Error`

Avancé

- **Nombre d'arguments :**

  - `[:x|x + 2] cull:7 cull:3. → 9`

    ```
    [:x :y | x + y] cull:2.
              → Error: This block accepts 2 arguments,
                but was called with 1 argument
    ```

  - `[:x :y :z :w :v| x + y] value:2 value:3 value:4 value:5 value:6.`
    `→ Error`

    Solution :

    `valueWithArguments``: #(1 3 5 7 9). → 25`

Avancé

- **Nombre d'arguments :**

  - `[:x|x + 2] cull:7 cull:3. → 9`

    ```
    [:x :y | x + y] cull:2.
              → Error: This block accepts 2 arguments,
                but was called with 1 argument
    ```

  - `[:x :y :z :w :v| x + y] value:2 value:3 value:4 value:5 value:6.`
    `→ Error`

    Solution :

    valueWithArguments: `#(1 3 5 7 9). → 25`

    → /!\

    Sens ?

Avancé

- **Contexte :**

```
AClass>>aMethod
        | t b |.
        t := 42.
```

Avancé

- **Contexte :**

```
AClass>>aMethod
    | t b |.
    t := 42.
    …
    ...
    t. →42
```

Avancé

- **Contexte :**

```
AClass>>aMethod
    | t b |.
    t := 42.
    …
    ...
    t. →42
    …
    ...
    b := [ t := 10].
```

Avancé

- **Contexte :**

```
AClass>>aMethod
    | t b |.
    t := 42.
    …
    ...
    t. →42
    …
    ...
    b := [ t := 10].
    …
    …
    b value.
    t. → 10
```

Avancé

- **Contexte :**

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

Avancé

- **Contexte :**

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
WitnessClass

|nom|
nom := ‘Je suis la classe témoin’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

Avancé

- Contexte :

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
WitnessClass

|nom|
nom := ‘Je suis la classe témoin’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
|classeAvecBloc classeTemoin res|.
```

Avancé

- **Contexte :**

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
WitnessClass

|nom|
nom := ‘Je suis la classe témoin’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
|classeAvecBloc classeTemoin res|.

classeAvecBloc := BlockClass new.
classeTemoin := WitnessClass new.
```

Avancé

- **Contexte :**

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
WitnessClass

|nom|
nom := ‘Je suis la classe témoin’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
|classeAvecBloc classeTemoin res|.

classeAvecBloc := BlockClass new.
classeTemoin := WitnessClass new.

res := classeTemoin afficheBloc: classeAvecBloc.
```

Avancé

- **Contexte :**

```
BlockClass

|nom|
nom := ‘Je suis la classe avec bloc’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
WitnessClass

|nom|
nom := ‘Je suis la classe témoin’

blocks >>>
        ^[nom].

AfficheBloc : aClass >>>
        ^aClass blocks.
```

```
|classeAvecBloc classeTemoin res|.

classeAvecBloc := BlockClass new.
classeTemoin := WitnessClass new.

res := classeTemoin afficheBloc: classeAvecBloc.

res. → [nom]
res value. → ‘Je suis la classe avec bloc’
```

Avancé

- **Return :**

```
[:x :y :z | x+y. y+z. x+z.] value.
```

Avancé

- **Return :**

```
[:x :y :z | x+y. y+z. ^x+z.] value.
```

Avancé

- **Return :**

```
[:x :y :z | x+y. y+z. ^x+z.] value.

[:x :y :z | x+y. ^y+z. x+z.] value.
```

Avancé

- **Return :**

[:x :y :z | x+y. y+z. ^x+z.] value.

[:x :y :z | x+y. ^y+z. x+z.] value. → ~~XXX~~

Avancé

- **Return :**

[:x :y :z | x+y. y+z. ^x+z.] value.

[:x :y :z | x+y. ^y+z. x+z.] value. → ~~XXX~~

Return de la méthode appelante, pas du bloc !
→ “Non-local return”

Avancé

- **Return :**

```
|a b|.
a := [^15].
b := [^'ERROR'].


-----------------------------------------------
Aclass>>method

  | x y |.
  ...
  ...
  (x == y)
        ifTrue:
              #(1 2 3) do: [:each|each*5].
              a value.
        IfFalse:
              b value.
  ------------------------------------------------------------------------
```

Avancé

- **Exception :**

```
[1+0.] on: ZeroDivide do: [42].
```

Avancé

- **Exception :**

```
[1+0.] on: ZeroDivide do: [42].

[1/0] on: ZeroDivide do: [‘ERROR : DivideByZero’].
```

Avancé

- **Exception :**

  [1+0.] on: ZeroDivide do: [42].

  [1/0] on: ZeroDivide do: [‘ERROR : DivideByZero’].

  [1+’ffhi’] on: ZeroDivide do: [‘unused’].

Avancé

- **Exception :**

```
[1+0.] on: ZeroDivide do: [42].

[1/0] on: ZeroDivide do: [‘ERROR : DivideByZero’].

[1+’ffhi’] on: ZeroDivide do: [‘unused’].

Transcript open.
[1/0] on: ZeroDivide do: [:exception | Transcript show: exception].
```

Avancé

- **Exception :**

[1+0.] on: ZeroDivide do: [42].

[1/0] on: ZeroDivide do: [‘ERROR : DivideByZero’].

[1+’ffhi’] on: ZeroDivide do: [‘unused’].

Transcript open.
[1/0] on: ZeroDivide do: [:exception :x | Transcript show: exception]
value:2. → ~~XXX~~

Conclusion

PHARO C'EST TROP BIEN !

UTILISEZ DES BLOCS !

SAUVEZ DES CHATONS !