Enumitem パッケージ

@monaqa

2020/04/14

目次

本ドキュメントは Enumitem パッケージ (v2.0.0) の仕様および使い方について述べたものです.

1　Enumitem パッケージの概要

Enumitem は， 組版用言語 SATYSFI において， 豊富な箇条書きリストや番号付きのリストを出力するためのパッケージです. SATYSFI には itemize というパッケージが標準で用意されていますが, enumitem パッケージでは itemize パッケージと比較してより豊富な機能を提供します（2020 年 4 月 14 日現在）. Enumitem パッケージを用いることで, 具体的に以下のような恩恵を受けることができます：

- デフォルトで豊富なスタイルを選択できる
- 番号付き箇条書き環境をネストさせることができる[*1]
- 定義リストを作成できる
- ネストごとに箇条書きのスタイルを変更できる
- ユーザ自身がスタイルを容易に拡張できる

以下，基本的な使い方，具体的なコマンドの一覧，そしてカスタマイズの方法について，順に説明していきます．

なお，本パッケージは 2020 年に v2.0.0 となり，その際にユーザーインターフェースを大きく変更しました．それに伴い本ドキュメントも大幅に変更しています．

2　基本的な使い方

Enumitem は標準と同様，+listing 及び +enumerate という箇条書きインターフェースを提供します．デフォルトでは以下のように箇条書きを書くことができます．

```
+listing{
  * hoge
  * fuga
    ** fuga1
      *** fuga11
      *** fuga12
    ** fuga2
}
```

- hoge
- fuga
  - fuga1
    - fuga11
    - fuga12
  - fuga2

```
+enumerate{
```

1 2020 年 2 月現在，標準ではサポートされていません．

```
  * hoge
  * fuga
    ** fuga1
      *** fuga11
      *** fuga12
    ** fuga2
}
```

1. hoge
2. fuga
    i. fuga1
        (1) fuga11
        (2) fuga12
    ii. fuga2

このように，番号つき箇条書き環境のネストもサポートしています．ネストの深さによってインデックスの種類を変えることができます．

また，文章の途中で箇条書きをはさみたくなることもあると思いますが，そんなときは \listing や \enumerate コマンドを使えば改段落を行わずに箇条書きを挿入できます．

```
+p{
  寿限無，寿限無，五劫のすりきれ，
  海砂利水魚の
  \listing{
    * 水行末
    * 雲来末
    * 風来末
  }
  食う寝るところに住むところ ...
}
```

寿限無，寿限無，五劫のすりきれ，海砂利水魚の

- 水行末
- 雲来末
- 風来末

食う寝るところに住むところ ...

3　+genlisting を用いたスタイルの指定

ここまでの内容は標準で提供されている itemize と大きな違いはありません．Enumitem パッケージの特徴は，より一般的な箇条書きを書くためのコマンド +genlisting を提供している点にあります．

3.1　スタイル指定子を用いた指定

まずは +genlisting を使用した箇条書きの例です．

```
+genlisting(Enumitem.dot-arabic)(Enumitem.default-item){
  * hoge
    ** hoge1
    ** hoge2
  * fuga
  * piyo
}
```

1. hoge
    1. hoge1
    2. hoge2
2. fuga
3. piyo

このように，アラビア数字 + ドット をラベルに持つような箇条書きが生成されました．+genlisting は +genlisting(labelf)(textf){item} という形で使用します． 第 1 引数の labelf はラベルの**スタイル指定子**であり，Enumitem モジュールでは dot-arabic をはじめとして 20 種類以上が予め定義されています．またその実態は単なる関数ですから，ユーザの手で自由に作成することもできます（詳細は後述）．textf は本文の**テキスト処理文脈指定**

子であり，こちらも default-item という指定子が Enumitem モジュールで定義されています．

+genlisting は番号付きの箇条書きだけでなく，番号のつかない箇条書きもサポートしています．以下は番号のつかない箇条書きを出力するためのコードと結果の例です．

```
+genlisting(Enumitem.white-bullet)(Enumitem.default-item){
    * hoge
      ** hoge1
      ** hoge2
    * fuga
    * piyo
  }
```

- hoge
  - hoge1
  - hoge2
- fuga
- piyo

なお，上の例では white-bullet などの指定子名に Enumitem を付けていますが，実際のマークアップでいちいちこれらを書くのは鬱陶しいかもしれません．このような場合，Enumitem モジュールを open することで名前空間を省略することができます．

```
@require: enumitem

open Enumitem

in
document(| 中略 |)<
  中略
>
```

このように書くと，Enumitem モジュール内で定義された raw-arabic などの関数を Enu-

mitem.raw-arabic と書かずに直接使えるようになります．ただし他のパッケージやユーザで定義した関数名と衝突しないようにするのは使用者の責任となります．パッケージを読み込むだけで関数名が衝突するという事態を避けるため，open しない限りモジュール名を省略できないようにしています．

3.2 指定子一覧

以下は，Enumitem モジュール内で定義されているスタイル指定子の一覧です．番号付きの箇条書きには以下の指定子を使用できます．

- アラビア数字系
  - 1 `raw-arabic`
  - 2. `dot-arabic`
  - (3) `paren-arabic`
  - [4] `bracket-arabic`
- ローマ数字系
  - i `raw-roman`
  - II `raw-Roman`
  - iii. `dot-roman`
  - IV. `dot-Roman`
  - (v) `paren-roman`
  - (VI) `paren-Roman`
  - [vii] `bracket-roman`
  - [VIII] `bracket-Roman`
- アルファベット系
  - a `raw-alph`
  - B `raw-Alph`
  - c. `dot-alph`
  - D. `dot-Alph`
  - (e) `paren-alph`
  - (F) `paren-Alph`
  - [g] `bracket-alph`

[H] bracket-Alph

また，番号の無い箇条書きには以下の指定子を使用できます．

- bullet
- white-bullet

3.3　ユーザ指定関数による指定

4　テキストの処理文脈変更

5　動的なフラグを用いたラベル操作

5.1　+xgenlisting コマンド

たとえば，以下のような To-Do リストを作成したくなったとします．

☐ ミルクを買う．

☐ The SATYSFIbook を読む．

☑ SATYSFI を完全に理解する．

☐ 課題を解く．

ここで，ラベルを作成する関数は既に用意されているとします．たとえば以下の関数 square-label is-checked ctx は，第 1 引数 is-checked が true のときチェック済みの，false のときチェック済みでないチェックボックスを描画する関数です．

```
let square-label is-checked ctx =
  let fs value = (get-font-size ctx) *' value in
  let fsize = fs 1.0 in
  let gr-square (x, y) =
    stroke 0.5pt Color.black
      (Gr.rectangle (0pt, 0pt) (fs 0.5, fs 0.5 ))
      |> shift-graphics (x, y +' fs 0.15)
  in
  let gr-mark-done (x, y) =
    stroke 0.5pt Color.black (
      Gr.poly-line (fs (0.0 -. 0.1), fs 0.45)
```

```
        [(fs 0.15, fs 0.15); (fs 0.75, fs 0.85)])
        |> shift-graphics (x, y +' fs 0.15)
  in
  let gr point =
    if is-checked then
      [gr-square point; gr-mark-done point]
    else
      [gr-square point]
  in
  inline-skip 10pt ++ (inline-graphics fsize fsize 0pt gr)
```

今までに紹介された +genlisting を用いても，上のように「3 番目のみチェックが付いたリスト」を実現することはできます．以下のようにパターンマッチや if 文などを用いて，チェックを付けたい場所のみ場合分けして処理すればよいのです．

```
+genlisting(fun depth idx ctx -> (
  let checked = match idx with
    | 3 -> true
    | _ -> false
  in
  square-label checked ctx
))(default-item){
  * ミルクを買う.
  * The \SATySFi;book を読む.
  * \SATySFi; を完全に理解する.
  * 課題を解く.
}
```

しかしこれはあまり直観的ではなく，また編集もしやすいとはいえません．「ミルクを買う」にチェックを付けたいとき，「ミルクを買う」というテキストから離れた場所を編集しなければならないのは手間ですし，「どこにチェックが付いているのか」をひと目で判断することができません．また，今回はアイテムの数が 4 つだったため数えるのも楽でしたが，もっと長いリストの途中にチェックをつけるためにいちいち数えなければならないのは手間です．このように，スタイル指定子を用いたレイアウトの指定は最初から最後まで一貫した規則を持つ箇条書きの生成には適しているものの，例外があったり，動的にレイアウトを変更したかったりするケースにはあまり適していません．

Enumitem パッケージではこのようなケースに対応するため，本文中でパラメータに値をセットし，その値をラベルに反映させる方法を提供しています．たとえば先程の To-Do リストは，プリアンブルにて以下のように定義された +todo-list 及び \done コマンドで作成したものです．

```
let done = EnumitemBase.make-item-local-param false
let-inline \done = {\set-item(done)(true);}
let-block +todo-list item = '<
  +xgenlisting(
    fun depth idx ctx -> square-label (Param.get done) ctx
  )(default-item)(item);
>
```

この +todo-list 及び \done コマンドを使うと，先程の箇条書きは以下のようにシンプルに書くことができます．

```
+todo-list{
  * ミルクを買う．
  * The \SATySFi;book を読む．
  * \done; \SATySFi; を完全に理解する．
  * 課題を解く．
}
```

□ ミルクを買う．

□ The SATYSFIbook を読む．

☑ SATYSFI を完全に理解する．

□ 課題を解く．

このように，本文に \done; というコマンドが付いているアイテムに限って，チェックボックスにチェックマークが付くようになります．\done の位置はどこにあってもかまいません．

何が起きたのか，もうすこし詳細に説明します．ポイントは以下の 4 点です．

- 箇条書きのアイテム内で一時的な値を保持するための器（パラメータ）を定義することができる．

- パラメータには，本文中で \set-item(param)(value); と打つことで一時的に値を設定することができる．
- +xgenlisting を使うと，第 1 引数でラベルのスタイルを指定する際に，Param.get 関数でパラメータの値を受け取り，その値に応じてラベルの出力を変更することができる．
- \set-item で変更した値はそのアイテムが終わると破棄され，デフォルト値にもどる．

今回定義した done パラメータは bool 型を持つ値でしたが，実際には int，inline-text，int -> int -> context -> inline-boxes 型など様々な型を持つパラメータを作成することができます．したがって，上記の例のようにフラグとして使うだけではなく，パラメータの値に応じてインデックスの値を変更したり，ラベルそのもののレイアウトを変えたり，と多彩なカスタマイズが可能となります．

5.2 +xgenlisting の注意点

+xgenlisting は便利なコマンドですが，+genlisting にはない欠点が存在します．それは「本文に副作用のあるコマンドを入れると（おそらく）不具合が起きる」ということです．たとえば，内部でカウンタをインクリメントさせる \footnote を +genlisting の本文で用いると，（実装にもよるとは思いますが）脚注の数字が 2 つインクリメントされる可能性があります．

理由は実装上の事情にあります．実は，「本文中にあるコマンドを読み取ってラベルに反映する」という行為は少し不自然なのです．なぜなら本来，ラベルを組み終わってはじめて後続する本文のテキスト幅が分かり，本文を組むことができるようになるからです．ラベルを組まないと本文が組めない，しかし本文を読まないとラベルを組めない，というのが本実装を困難にする点でした．

そこで，+xgenlisting ではその問題を解決するため，「ラベルを組む前に read-inline で本文を読み，読み終わったものは一度破棄する」という方式で実装を行いました．本文中のコマンドは read-inline で読まれることにより評価され，\set-item がある場合はパラメータが一時的な値へとセットされます．その後であれば既にユーザ指定が分かっているためラベルを組むことができ，ラベルを組んでから再度改めて本文を組むことができます．

そしてこの方法だと（お気付きの通り），本文は read-inline によって 2 度評価されます．副作用のないコマンドであれば何度評価されても結果は変わりませんから問題ありませんが，副作用のあるコマンド，特に 1 度だけ呼ばれることを想定しているコマンドを

+xgenlisting の中に入れると，このことによって挙動がおかしくなるのです．

6　+gendescription コマンド

工事中

(i) あああああ

いいいいいい

(iii) えええええ