8 Page model set

```
;; -*- Scheme -*-
;;              Amended Pagemodel.dsl

;; Additional characteristics

;; Ruby character sequence cannot be extended over the Kanji
(declare-characteristic ruby-style "-//JIS//TR X 0010//JP" #t)

;; Patterns of line.
(declare-characteristic line-style "-//JIS//TR X 0010//JP" rule)

;; Ruby and emphasized mark don't increase line space.
(declare-characteristic layout-rule "-//JIS//TR X 0010//JP" #t)

;;
;; Page model

;; Default height of header/footer, for 2 lines.
(define *header-height* (* (cadddr *page-spec*) 2))
(define *footer-height* (* (cadddr *page-spec*) 2))

;; *page-region-y-offset* is distance from top left. Change into bottom left.
;; In addition to, region contains header/footer.

(define *pr-y-off*
  (- *paper-height* *page-region-y-offset*
     *page-region-height* *footer-height*))

;;;;;;; improvement (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define-page-model standard-rect-page
;  (filling-direction 'top-to-bottom)
;  (width *paper-width*)
;  (height *paper-height*)
;  (region
;    (x-origin *page-region-x-offset*)
;    (y-origin *pr-y-off*)
```

```
;   (width *page-region-width*)
;   (height (+ *page-region-height* *header-height* *footer-height*))
;    (header
;      (height *header-height*)
;     (width *page-region-width*)
;      (generate (HEADER-CONTENT 'rect)))
;   (footer
;      (height *footer-height*)
;      (width *page-region-width*)
;      (generate (FOOTER-CONTENT 'rect)))
;   ))

;;;;;;; improvement (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define *verso-pr-x-off*
;  (- *paper-width* *page-region-x-offset* *page-region-width*))

;;;;;;; improvement (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define-page-model standard-verso-page
;  (filling-direction 'top-to-bottom)
;  (width *paper-width*)
;  (height *paper-height*)
;  (region
;   (x-origin *verso-pr-x-off*)
;   (y-origin *pr-y-off*)
;   (width *page-region-width*)
;   (height (+ *page-region-height* *header-height* *footer-height*))
;   (header
;    (height *header-height*)
;    (width *page-region-width*)
;    (generate (PAGE-HEADER 'verso)))
;   (footer
;    (height *footer-height*)
;    (width *page-region-width*)
;    (generate (PAGE-FOOTER 'verso)))
;   ))

;;
;; Specification of headline and page number.
;;
```

```
;;      *has-header-nonburu*
;;      *has-header-hasira*
;;      *has-footer-nonburu*
;;      *has-footer-hasira*
;;      *hasira-rect* (string)
;;      *hasira-verso* (string)
;;
;;   space for Headline and page number, Em.
;;

;;;;;;; improvement (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define (PAGE-HEADER side)
;  (make paragraph
;        use: *header-footer-style* ;;*paragraph-style*
;        quadding: (if (eq side 'verso) 'end 'start)
;        (case side
;          (('rect)
;           (sosofo-append ((if *has-header-nonburu*
;                               (MAKE-NONBURU) #f)
;                           (literal "    ")
;                           (if *has-header-hasira*
;                               (literal *hasira-rect*) #f)
;          )))
;         (else
;           (sosofo-append ((if *has-header-hasira*
;               (literal *hasira-verso*) #f)
;           (if *has-header-nonburu*
;               (progn
;                (literal "    ")
;                (MAKE-NONBURU)) #f)
;           )))
;        )))

;;;;;;; improvement (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define (PAGE-FOOTER side)
;  (make paragraph
;        use: *header-footer-style* ;;*paragraph-style*
;        space-before: *base-font-size* ;; 1 line space
;        quadding: (if (eq side 'verso) 'end 'start)
```

```
;       (case side
;         (('rect)
;          (sosofo-append ((if *has-footer-nonburu*
;                              (MAKE-NONBURU) #f)
;                          (literal "    ")
;                          (if *has-footer-hasira*
;                              (literal *hasira-rect*) #f)
;          )))
;         (else
;           (sosofo-append ((if *has-footer-hasira*
;               (literal *hasira-verso*) #f)
;           (if *has-footer-nonburu*
;               (progn
;                 (literal "    ")
;                 (MAKE-NONBURU)) #f)
;          )))
;       )))

(define MAKE-NONBURU
  ;;(literal "page ")
  (page-number-sosofo)  ;; Number only
)

;;
;; Set footnote
;;
(define (MAKE-FOOTNOTE-EXP num)  ;; Expression footnote
  (make-numbering num *footnote-number-desc*))

;;;;;;; improvement (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;
;(define (FOOTNOTE)
;  (let ((footnote-exp (MAKE-FOOTNOTE-EXP (footnote-number 'page))))
;     ;; First, only footnote number every page.
;     ;; reference mark position of note is where (FOOTNOTE) is called .
;    (make sequence
;          (make included-container-area                to be revised
;                use: *footnote-style*
;                display?: #f
;                (literal footnote-exp))
```

```
;         (make paragraph
;              label: 'footnote
;              (literal footnote-exp)
;              (literal "    ")
;              (process-children)))))

;;;;;;; (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;------Footnote number only-------;
(define (FOOTNOTE)
   (make superscript
      font-size: (* (inherited-font-size) *superscript-font-size-factor*)
      (literal (MAKE-FOOTNOTE-EXP (element-number (current-node))))))

;;;;;;; (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;------Footnote contents----------;
(define (FOOTNOTE-CONTENTS)
   (with-mode FOOT-MODE
      (process-node-list (node-matching-list (parent (current-node)) "FN"))))

;;;;;;; (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(mode FOOT-MODE
   (element FN
      (make paragraph
         (literal (MAKE-FOOTNOTE-EXP (element-number (current-node))))
         (process-children))))

;;;;;;; (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;----Searches in the subgrove whose roots are each member of nl for element matching pattern.----;
(define (node-matching-list nl pattern)
   (let (
         (first (node-list-first  nl))
         (rest (node-list-rest  nl)))
      (if (node-list-empty? first)
         first
         (node-list
            (if (string=? pattern (gi first))
               first
               (node-list-rest first))
            (node-matching-list (children first) pattern)
```

```
            (node-matching-list rest pattern)))))

;;;;;;; improvement (included-container-area);;;;;;;;;;;;;;;;;;;;;;;;;;
;(define (FOOTNOTE-SEPARATOR)
;  (make rule
;        orientation: 'horizontal
;        line-thickness: 1pt
;        ))

;;
;; Set column
;;
;;;;;;; improvement (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define *column-width* (* (cadr *page-spec*)(caddr *page-spec*)))
;(define *column-width+gap*
;  (let ((gap (cddddr *page-spec*)))
;    (if (null? gap)
;        *column-width*
;        (* (cadr *page-spec*)(+ (caddr *page-spec*) (car gap))))))

;;;;;;; improvement (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define (filling-dir fld) ;; Filling direction is perpendicularity if
                          ;; how to set type is vertical.
;  (if (equal fld 'vertical)
;      'left-to-right
;      'top-to-bottom))

;;;;;;; improvement (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define-column-set-model standard-one-column-model
;  (filling-direction (filling-dir *writing-dir*))
;  (column-subset
;   (column
;    (x-origin 0)
;    (width *column-width*)
;    (footnote-separator
;     (generate (FOOTNOTE-SEPARATOR))
;   (flow '(footnote footnote) )
;   ))
;))
```

```
;;;;;;; improvement (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  (filling-direction (filling-dir *writing-dir*))
;  (column-subset
;   (column
;    (x-origin 0)
;    (width *column-width*))
;   (column
;    (x-origin *column-width+gap*)
;   (width *column-width*)
;   (footnote-separator
;   (generate (FOOTNOTE-SEPARATOR))
; (flow '(footnote footnote) )
;   ))
;))

;;;;;;; improvement (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;(define-column-set-model standard-three-column-model
;  (filling-direction (filling-dir *writing-dir*))
;  (column-subset
;   (column
;    (x-origin 0)
;    (width *column-width*))
;   (column
;    (x-origin *column-width+gap*)
;    (width *column-width*))
;   (column
;    (x-origin (* 2 *column-width+gap*))
;    (width *column-width*)
;    (footnote-separator
;     (generate (FOOTNOTE-SEPARATOR))
;   (flow '(footnote footnote) )
;   ))
;))

;; Functions to generate top level flow object.
;;      Call with top level construction rule.
;;
```

```
;;;;;;; improvement (page-sequence, column-set-sequence, included-container-area);;;
;(define (STANDARD-PAGE-SEQUENCE)
;  (case *column-number*
;    ((1)
;     (make page-sequence                                    to be revised
;           initial-page-models: (standard-rect-page standard-verso-page)
;           repeat-page-models: (standard-rect-page standard-verso-page)
;           ;;content-map: '((footnote footnote))
;           (make column-set-sequence                        to be revised
;                 column-set-model: standard-one-column-model
;                 (process-children-trim))       ))
;    ((2)
;     (make page-sequence                                    to be revised
;           initial-page-models: (standard-rect-page standard-verso-page)
;           repeat-page-models: (standard-rect-page standard-verso-page)
;           ;;content-map: '((footnote footnote))
;           (make column-set-sequence                        to be revised
;                 column-set-model: standard-two-column-model
;                 (process-children-trim))       ))
;   ((3)
;    (make page-sequence                                     to be revised
;          initial-page-models: (standard-rect-page standard-verso-page)
;          repeat-page-models: (standard-rect-page standard-verso-page)
;           ;;content-map: '((footnote footnote))
;           (make column-set-sequence                        to be revised
;                 column-set-model: standard-three-column-model
;                 (process-children-trim))       ))
;    ))

;;;;;;; (page-sequence, column-set-sequence, included-container-area);;;;
(define (STANDARD-PAGE-SEQUENCE)
  (make simple-page-sequence
    font-family-name: *base-font-family*
    font-size: *base-font-size*
    line-spacing: (cadddr *page-spec*)
    left-header: (make sequence
                       font-size: (- *base-font-size* 1pt)
                       line-spacing: (cadddr *page-spec*)
                       font-posture: 'italic
```

```
                (if-front-page
                 (empty-sosofo)
                 (sosofo-append
                   (if *has-header-nonburu*
                     MAKE-NONBURU (empty-sosofo))
                   (if *has-header-hasira*
                     (literal *hasira-verso*) (empty-sosofo)))))
right-header: (make sequence
                font-size: (- *base-font-size* 1pt)
                line-spacing: (cadddr *page-spec*)
                font-posture: 'italic
                (if-front-page
                  (sosofo-append
                    (if *has-header-hasira*
                      (literal *hasira-rect*) (empty-sosofo))
                    (if *has-header-nonburu*
                      MAKE-NONBURU (empty-sosofo)))
                  (empty-sosofo)))
left-footer: (make sequence
                font-size: (- *base-font-size* 1pt)
                line-spacing: (cadddr *page-spec*)
                font-posture: 'italic
                (if-front-page
                  (empty-sosofo)
                  (sosofo-append
                    (if *has-footer-nonburu*
                      MAKE-NONBURU (empty-sosofo))
                    (if *has-footer-hasira*
                      (literal *hasira-verso*) (empty-sosofo)))))
right-footer: (make sequence
                font-size: (- *base-font-size* 1pt)
                line-spacing: (cadddr *page-spec*)
                font-posture: 'italic
                (if-front-page
                  (sosofo-append
                    (if *has-footer-hasira*
                      (literal *hasira-rect*) (empty-sosofo))
                    (if *has-footer-nonburu*
                      MAKE-NONBURU (empty-sosofo)))
```

```
                    (empty-sosofo)))
  top-margin: *page-region-y-offset*
  bottom-margin: (- *paper-height* *page-region-height* *page-region-y-offset*)
  left-margin: *page-region-x-offset*
  right-margin: (- *paper-width* *page-region-width* *page-region-x-offset*)
  header-margin: (- *page-region-y-offset* *header-height*)
  footer-margin: *pr-y-off*
  page-width: *page-region-width*
  page-height: *page-region-height*
  quadding: 'justify
  page-n-columns: *column-number*
      (process-children-trim)
  ))

;;;;;;; (column-set-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Characteristic definition for multi column-sets on simple-page-sequence.
(declare-characteristic page-n-columns
      "UNREGISTERED::James Clark//Characteristic::page-n-columns" 1)

;;;;;;; (page-sequence);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Function definition to make verso and recto have difference sosofo.
(define if-front-page
  (external-procedure "UNREGISTERED::James Clark//Procedure::if-front-page"))
```