TableLayout BreakHandling

Break handling inside tables with the Knuth approach

The difficulty

The normal block-level LMs handle hard breaks in a fashion that the current element list is prematurely ended using a Knuth Penalty with p=-INFINITE. Additional calls to getNextKnuthElements() return the next element list. This is also related to the way that the page breaker does its job. Normally it would handle a whole page-sequence at once, but in the case of a hard break, you get two (or more in case of multiple breaks) element lists (before and after the break) which are all handled separately by the page breaker.

Table layout (and similarly for lists) has the added complexity of creating combined elements lists. It means synchronizing multiple element list sources at the same time. It's not just a matter of returning early when a hard break is encountered but it is also necessary to check where the other simultaneously active element lists are broken, and to find out how to restart the element generation for the next list.

Current status (2005-06-16)

The TableContentLayoutManager is designed to deliver a single element list for a whole table without currently respecting break-before and break-after properties on any level. The class (and its helpers) needs to be made restartable. This page is dedicated to find out how this should be done and to document this decision.

Current status (2005-06-23)

All breaks inside tables are now supported, but the TableLM is not restartable.

Where are hard breaks possible inside tables?

According to the spec:

- fo:table
- fo:table-row
- finally inside the block-level content of table-cells but not on fo:table-cells itself.

Just for completeness, here are the nodes on which hard breaks or possible inside lists as they will be quite similar to tables:

- fo:list-block
- fo:list-item
- finally inside the block-level content of list-item-labels and list-item-bodies.

Approach

It is best to look at the most complex case first, which are the breaks on table-cell content. I expect the rest to automatically click into place after that. The only non-trivial place will be the breaks on fo:table-row if a row is inside a multi-row row group in which case the problem is similar to the breaks inside table-cell content.

Details

So far, I've identified the starting point for handling breaks. This would be TableStepper.getNextStep() where the element lists are checked for possible break points. An additional check for penalties with p=-INFINITE reveals the hard breaks. I've added a boolean array for signalling hard break situations. It is quite easy to halt further processing from there, the difficulty is to figure out how to store the state and restart processing at the right point, since this hard break somewhat disrupts the flow of the algorithm.

In the end it turned out that there is an easy way: We can simply insert hard breaks (penalties p=-INF) into the element lists. The breaker handles these pretty well so far. So the table layout didn't need to be made restartable, yet. I imagine it has to be, later, when changing available IPD needs to be supported.