Concept - Blazing fast vaadin trees

Abstract

As of 5.0-alpha3, trees and lists are hardly usable, refreshing all the time, jumping up and down... So far this has even prevented us from completely dropping the old adminInterface module, because it is still much more usable, responsive. The goal of this concept is to analyze and highlight the performance hiccups in the magnolia trees and lists.

Tasks are also defined in the following parent ticket:

Error rendering macro 'jira'

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

1. Rows with different heights ✓

- **Cause:** This is due to the activation-status icon having a bigger font-size, introducing a bigger row-height.
- **Effect:** This is then likely to produce scroll jumps when the table is repainted, because this is based on calculated row height.
- **Fix:** CSS only.
    - ⚠ Unable to locate Jira server for this macro. It may be due to Application Link configuration.

2. Tables set 'immediate'

- **Cause:** What *immediate* means for Vaadin components is that value changes are immediately propagated to server.
- **Effect:** For tables, the value is the selection, so every time one presses an arrow key and changes selection, we get a server roundtrip, seemingly repainting the table.
- ⚠ However, setting tables non-immediate would no longer update location fragment in the URL or action bar preview as soon as selection changes, but in the next roundtrip.
- **Fix:** Set table non-immediate, and implement an asynchronous scheduled server-side update for selection value changes, so that it does not block client-side selection.

3. Lazy loading in trees ✓

- **Effect:** When getting down a big amount of nodes, at some point selection cannot keep up.
- **Cause:** That is because only a subset of rows (visible row count * cache ratio) has effectively been loaded (**row-based** lazy-loading).
- ℹ We don't need row-based lazy loading in trees.
    - There is already **depth-based** lazy-loading provided by the hierarchical container.
        - This ensures child nodes are loaded only when parent is expanded.

        - Chance is minimal to run into huge loading times when expanding.
            - JCR itself is not optimized for big flat data structures.
            - It was the same in old admin tree.
- How do I turn it off?
    - Row-based lazy loading is provided by the `Table` component.
    - Documentation tells to use `setPageLength(0)` to disable paging...
        - ... until client-side updates it automatically!
- **Fix:** Patch the Vaadin `Table` class to always use the `getPageLength()` getter from the class' internal code (no direct field reference).
    - This enables our `TreeTable` to override `getPageLength()` and return `0` all the time.
    - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- Some lazy loading might still occur when collapsing nodes down the hierarchy, but that's acceptable, given how tightly lazy loading is integrated within the `Table` component.
- There will be a new Table component in a future Vaadin version, but not by the time we ship 5.0.

4. Partial updates

- **Cause:** Vaadin `TreeTable` component normally supports partial updates if container is an `ItemSetChangeNotifier` (not sure why that condition).
    - But this is currently disabled with `// FIXME: This disables partial updates until TreeTable is fixed so it does not change component hierarchy during paint`
        - Forcing partial updates seemed to work basically fine.
        - Vaadin commit said: Currently `TreeTable` changes its child components during paint, which is too late for `AbstractCommunicationManager` to take into account
        - see http://dev.vaadin.com/ticket/8628
- **Effect:** The whole table is repainted when expanding one node.
    - This is not the most harmful factor for user interaction, but it surely has its share of impact when many nodes are expanded.
- Inplace editing is not yet using partial updates, although that was also done in the standalone PoC (ticket already exists).
- **Fix:** Patch `TreeTable#setContainerDataSource()` method to set `containerSupportsPartialUpdates = true`.

5. Column expand ratios ✓

- **Effect:** When comparing M5's config tree with a Vaadin prototype using the same JCR containers, there was still quite a big latency in the config tree when expanding a node.
    - Column widths are actually updated for all the table, when expanding/collapsing nodes!
        - Firebug clearly shows that table cells' width value is bouncing several times within 1-2 pixels of its original value; this is a very expensive CPU operation.
- **Cause n°1:** We set column's expand ratio systematically, even when it's not configured.
- **Cause n°2:** Table seems to send visible columns and their expand ratio even if values have not changed.
- **Fix:** First, do not set expand ratio systematically when building columns.
    - Then patch `TreeTable` client-side classes so that it does not repaint too many times, nor change column width according to max indent.
    - Unable to locate Jira server for this macro. It may be due to Application Link configuration.

6. Scroll position ✓

- **Effect:** The tree might still jump slightly when expanding/collapsing nodes, just for the time it repaints.
- **Cause:** It is likely that there is some scroll position hack in connectors that we should assert how useful it is.
    - FOUND! line 138 in `TableConnector`: `getWidget().updateFirstVisibleAndScrollIfNeeded(uidl);`
        - was introduced with partial updates, see http://dev.vaadin.com/ticket/6722
        - it seems it can be removed with no visible effect (at least without partial updates).
- **Fix:** Patch `TableConnector`to remove this call.

- ⚠ Unable to locate Jira server for this macro. It may be due to Application Link configuration.

7. Subsequent UIDL requests ✓

- **Effect:** When selecting a table row, we send a UIDL request and get location fragment / action bar changes in response, but then a new UIDL request is sent to server with the new location fragment.
- **Cause:** Since Vaadin 7, the `UI` (root element) has the Navigator API built-in, which is aware of the page URI and fragment, but we don't use it at all for location handling.
  - When client-side repaints, Vaadin's `VUI` sees a location change on the client-side, which differs from its server-side state
- **Fix:** We can easily update Vaadin `UI`'s page fragment at the same time we set the magnolia fragment. Then when `VUI`listens to location changes, it is no longer out of sync.
  - ⚠ Unable to locate Jira server for this macro. It may be due to Application Link configuration.

8. Vaadin components in cells

- **Cause:** The status icon is currently served as a plain Vaadin `Label`, using a `ColumnGenerator`. This means that Vaadin has to maintain and update all those label's state, for as many rows as the table has.
- **Effect:** When expanding e.g. modules in config app, there is still a delay of approximately 0.5-1s.
- **Fix:** We should serve the status/permissions icons as style names, using a `CellStyleGenerator`, then we avoid those extra components and make the tree even more responsive.
  - ⚠ This may well require to change how we configure custom columns; we need a new concept here.

9. Scrollbar blinking

- **Effect:** When changing table selection, we can see the scrollbar quickly disappears and reappears.
- **Cause:** There is a client-side hack called `runWebkitOverflowAutoFix()` that briefly sets `overflow:hidden` before reverting it to `auto`, so that scrollbar appearance is recalculated.
  - That fix doesn't make sense for selection changes, although it does for collapse requests.
- **Fix:** Patch the client-side again to process that fix only for collapse requests. Please note that partial updates might behave differently regarding this.

Disculped

- It is interesting to note that our **JCR containers are fine**; they are not causing performance flaws.
  - Some methods however should be left unsupported since the container should be read-only, in other words, persistance is not managed by the container.
  - There is interesting insight on this in the Vaadin book section about SQL container.

- Tried without config app's inplace editing (uses a `FieldFactory`) => no significant improvement
- Tried without our own `MagnoliaTable` and `MagnoliaTreeTable` => no significant improvement
- Tried without `CellStyleGenerator` => no significant improvement
- Tried without our client-side patches that force width and height in the post-layout phase => no significant improvement