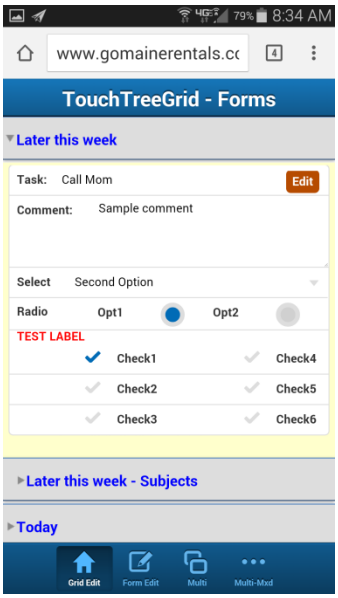
TouchTreeGrid – Embedded Forms

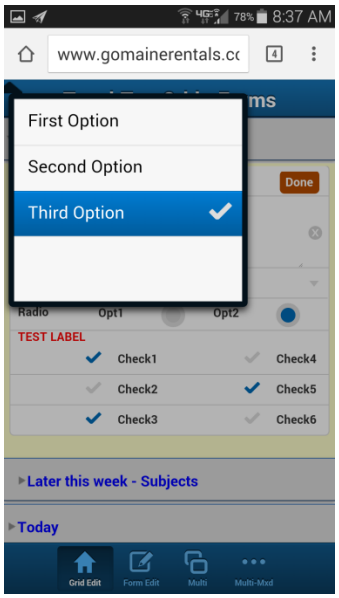
TouchTreeGrid now supports embedded Forms within TreeGrids with minimal additional configuration. Forms can be embedded in Content rows (aka Leafs) and/or Category rows.

See Demo at: http://www.gomainerentals.com/Sencha/TTG_Forms/index.html

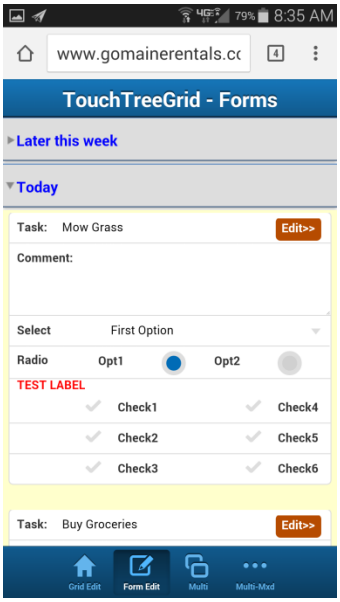
Forms Example – Tree Grids

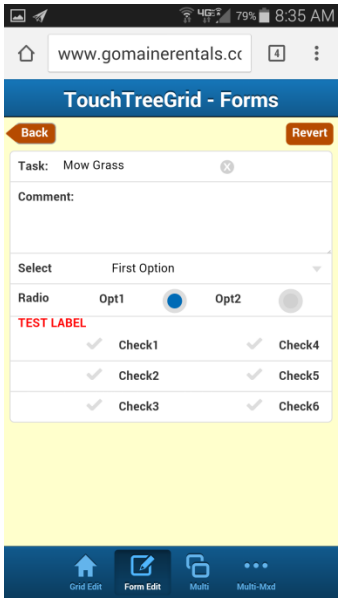
Edit within Grid:



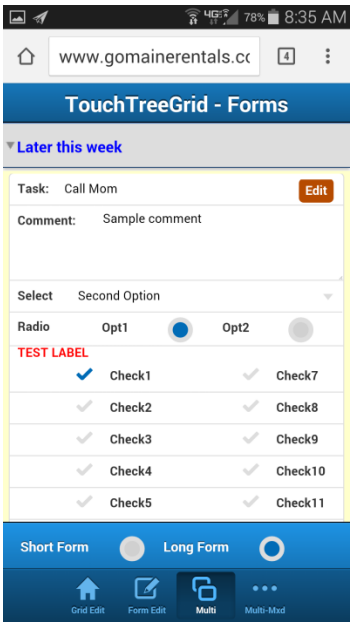


Read-only within Grid, edit within stand-alone Form:

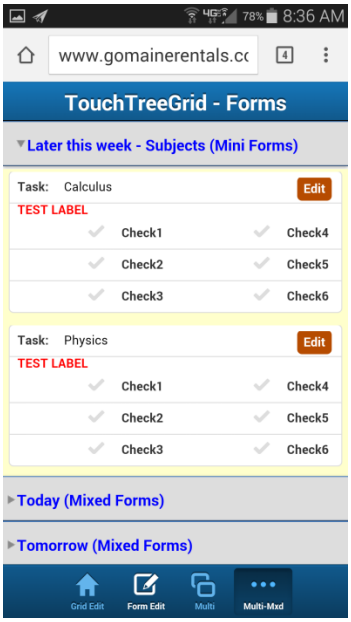




Toggle Multi-Forms within Grid:



Record-based Mixed Forms:



Demonstrates

The “Grid Edit” tab demonstrates editing directly with the grid. The “Form Edit” tab demonstrates read-only display within the expanded grid where user navigates to a separate Form Panel for purposes of edit. The “Multi” tab demonstrates how different formats can be easily toggled (example Short-form vs. Long-form). The “Multi-Mixed” tab demonstrates how different forms can be easily displayed based on the record being displayed. It is also possible to display different forms based on category. This implementation supports a form-level “Read-only” config which hopefully will be supported by future release of Sencha Touch.

Known Limitations

This is a work in progress. I am releasing now as supported functionality may be useful to some as is. Edit functionality works best across all the form component types with Touch versions 2.3.0 and up. Currently only the following form elements are supported: textfield, textareafield, selectfield, checkboxfield, radiofield, labels, buttons, images. Some of the others may or may not work without additional enhancements; I just haven't spent much time on them yet. Edit of textfield and textareafield does not seem to work smoothly on iOS devices (still working on this) so I disabled edit for these devices in all but the “Form Edit” example (see MyFormPanel.js postInit method). “Grid Edit” does work for the Android devices I tested. Read-only mode works on all devices. If editing is required, I propose to either use the “Form Edit” approach for optimal performance and usability, or a hybrid solution where form edit is used for iOS devices and “Grid Edit” used for all other supported devices.

Performance degrades the more form rows are displayed at the same time. This is particularly true when using radio buttons which I struggled the most with to support. I recommend config “singleExpand: true” to display in accordion mode and I recommend against implementing expandAll feature (disabled in my examples).

Basic Config

Embedding Forms within your Tree Grid is very easy:

- Create Form Panel classes just like you normally would
  - Manually add "**readOnly: true**" (or false) config to define initial edit mode
    - Note: any form components that support "readOnly: true" config would never be updated to non-readOnly by TouchTreeGrid, so only define this config at the component level if truly intended.
  - Use "**name**" config for each form element to map to matching field name within the store used by the grid. Each radio group requires the same "name" config.
  - Add special purpose **postInit()** method to your form to handle any customizations to the form after being rendered (see examples).
  - Assign '**cls**' config to each Form and reference in CSS file as recommended in "mycss.css" from examples (see CSS selector 'myForm').

- Define your TreeGrid same as you normally would with following additional configs:
  - Content rows require "**contentItemTplOverride**" config which references the form to display. Example:
    - contentItemTplOverride:
      '{[this.renderer_myForm("TouchTreeGrid.view.MyFormPanel", "320px", values)]}'

      "**this.renderer_myForm**" is an internally provided method which must be called to renderer your form.
      - 1st parameter specifies form class to display (required)
      - 2nd parameter defines fixed height to display form within grid (required)
      - 3rd parameter = "values". This is required for internal usage. "values" is object defining current record along with a number of other useful data (refer to Sencha's API on Ext.XTemplate)
      - 4th parameter is optional where you can pass a configuration object when the instance of the form is created similar to:
        myForm = Ext.create(myClass, myConfigs);

  - Category rows would similarly utilize "**categItemTplOverride**" config to render forms.

  - **variableHeights: true** (required)

  - **infinite: false** (recommended, but test as true for your implementation for potentially improved performance)

- Assign '**cls**' config to each grid instance and first reference "**x-touchtreegrid-list**" followed by your custom CSS selector ("**touchtreegrid-myForm**" in the examples). Refer to "mycss.css" file in the examples for recommended CSS.

Program Flow – "Grid Edit" Example

- All examples launched from "TouchTreeGrid.view.Main"
- "Grid Edit" references "TouchTreeGrid.view.MyFormPanel" form
- MyFormPanel postInit() method has logic to update text of Edit button from "Edit" to "Done" based on form instance readOnly status.
  - Note: both the record instance of myFormPanel.config.readOnly and record.readOnly updated after initial form rendering and used to control readOnly vs. edit modes for that record until store is reloaded.
- MyFormPanel postInit() also adds tap listener to Edit button (i.e. for each instance of the form) to call onMyFormEditBtnTap() method
- MyFormPanel onMyFormEditBtnTap() method toggles readOnly status and refreshes the form for that record.

Program Flow – "Form Edit" Example

- "Form Edit" adds grid instance and form panel instance to "formEdit" container (card layout).
- Form "TouchTreeGrid.view.MyFormPanel2" used for this example
- TouchTreeGridController has controller actions for switching between edit and readOnly mode
  - onMyForm2EditBtnTap() method handles edit button tap from grid form instance, sets form panel as new active item within "formEdit" container, and updates form panel instance from record for purposes of editing.
  - onMyForm2BackBtnTap() method updates record from form and includes custom logic for radio buttons.
  - onMyForm2RevertBtnTap() method resets the form to unedited state.
- I added 'ease-in' animation to node expand for this example. Refer to end of mycss.css for relevant css. Note this required adding cls: 'touchtreegrid-myForm-animate' to this grid instance. Animation not as appealing for grid edit examples as the form gets refreshed and animation applied when toggling between readOnly and edit mode.

Program Flow – "Multi" Example

- "Multi" example adds "multiexample" grid instance and toolbar with ShortForm/LongForm radio buttons to "multiForms" container (vbox layout).
- "formType: short" config added to "multiexample" grid instance to track shortForm/longForm state.
- Listeners on "check" event defined within "TouchTreeGrid.view.Main" call onShortFormTypeCheck() and onLongFormTypeCheck() methods which update grid "formType" config and refresh the grid similar to:

```
multiexample.config.formType='long';  // toggle  'short' and 'long'
```

multiexample.doRefreshList(true); // refreshes grid applying toggled form

- Multi-forms implemented with config:
  contentItemTplOverride: '{[this.scope.config.formType===”short” ?
  this.renderer_myForm("TouchTreeGrid.view.MyFormPanel", "320px", values) :
  this.renderer_myForm("TouchTreeGrid.view.MyFormPanelLong", "420px", values)]}',

  Note: “this.scope” references the grid instance itself and is useful for many purposes.
- Both MyFormPanel and MyFormPanelLong have postInit() and onMyFormEditBtnTap() methods to handle edit vs. readOnly modes

Program Flow – “Mixed” Example

- “Mixed” example adds “multirec” grid instance to tab panel
- “Columns” config includes renderer config which specifies renderer method to call to display text for Category rows:
  renderer: 'this.**renderer_CategText**(values)'
- “renderers” config includes definition of renderer_CategText() function:
  renderer_CategText: function (values){
  return values.text + ' (' + values.**categType** + ')';
  }
- “**categType**” is a field provided in “TaskMultiRec” store which specifies category descriptive text regarding form types for that category.
- “**contentItemTplOverride**” config specifies custom renderer function to call to specify which form to display for current record:
  contentItemTplOverride: '{[this.**renderer_multiFormRec**(values)]}'
- “renderers” config includes definition of renderer_multiFormRec() function:

```
renderer_multiFormRec: function (values) {
   if (values.formType === 'mini'){
      return this.renderer_myForm("TouchTreeGrid.view.MyFormPanelMini", "160px", values)
   }
   else if (values.formType === 'long'){
      return this.renderer_myForm("TouchTreeGrid.view.MyFormPanelLong", "420px", values)
   }
   else {
      return this.renderer_myForm("TouchTreeGrid.view.MyFormPanel", "320px", values);
   }
}
```

- “**formType**” is a field provided in “TaskMultiRec” store which specifies form to display for eachrecord