RI Requirements and Documentation

Table of Contents

Overall Goal

High quality UI reference implementation that runs across phones, tablets, desktop which leverages Dojo 1.9 features and demonstrates latest best practices. The RI should be complete enough to serve as a good starting point for development of more extensive applications.

General Requirements

- Implement CRUD operations on the Model shown below, including the ability to search on several fields.

- Appropriate separation of concerns via MVC (or whatever MV* “pattern” is used).

- The usage of the underlying “framework” and reference implementation should be as straightforward as possible for both simple and more complex scenarios.

- Approach to communication with back-end rest services, including proper error handling.

- Devices & Browsers
  - Same as those documented on dojotoolkit.org, with following notes/discussion topics:
    - How well can Android prior to 4.x be supported due to the browser issues? (Note: Chrome beta is now available on 4.x and supposedly Google plans to support it across all devices.)

  - Must run on phone, tablet, desktop:
  - Goal is as much re-use as possible with appropriate usage of responsive design.

- Desktop/click support

- Scrollable components must support a clickable scroll bar; i.e., it is not obvious to an end user to click on a bar and drag it.

- i18n approach should be demonstrated.

- Approach to SEO should be demonstrated with following considerations:
    - How to address SEO considerations with single page architecture

- Proper “browser back button” support

- Each “view” has its own uri that is bookmarkable.

- Various types of input fields must be demonstrated with an approach to how validation errors are fed back to user
    - E.g., numeric input field with decimals
    - Handling basic error feedback from a REST service invocation

- CSS styling - Must appear professional across all devices

Views

Common Elements

- more … Button - Provides additional options for the given view; for now the ref impl can navigate to a generic “more …” view that has some simple text on it with ability to navigate back to originating view.

- Status element - A “single” selection list. Allowable values are populated from a json “status” uri.

List

- Search/Sort Button - Displays Search view.

- Create Button - Displays Detail view.

- Checkbox above list of items - When checked, will check all of the items in the list.

- Delete Button - Always appears but is disabled until either the “all” checkbox or one of the checkboxes next to a particular “item/row” is checked. When clicked, will prompt user to verify the “delete” was intended. If confirmed, will delete the items from the model and the items will disappear from the view. If one of the items deleted is currently displayed in the Detail view (e.g., as could occur with tablet/desktop or even phone in landscape), the view will be blanked out (all elements removed).

- List of items - Initial display (e.g.., when List view is first rendered such as invoking from a menu): A configurable - from an application level json file - number of items are displayed; e.g., twenty. The items to be displayed are the most recent - createdDate - by “requestedBy” (default to “user1” for now

since there is no login at this point). When a search/sort is performed from the Search/Sort view, the list of items and order will be determined by the search criteria. For each item, the following fields will be displayed: id, first 25 chars of description, and status. The format can be multi-line or grid; exact format not yet specified.
Note: Changes to to the model will be reflected in the List view; e.g., when creating, updating, or deleting items in the Detail view.

Search/Sort

- List Button - Navigates to previously populated List view.

- Submit Search Button - Invokes the search service and displays the List view. The search will pass all fields to the "search service" which will return all items that match the search criteria; an "and clause" is implied for each search field value.

Search

- All of the following search fields are optional; if none is entered and Submit Search button is pressed the default criteria specified under the List view should be used.

- Id input field - Validation: Must be a positive integer and <= 1,000,000,000.

- Requested By - Validation: Must be < fifty chars. Search is exact match on the string. (Could search on beginning of string also.)

- Requested Finish From/To Date - Validation: From Date must be <= To Date. If From Date is empty, defaults for search purposes to "beginning of time" (view field remains blank). If To Date is emtpy, defaults to "end of time" (view field remains blank).

Sort

- 1st selection field - Contains list of fields (from a json config file) that the List can be sorted on. First item in list will be default (based on physical sequence of items in json config file).

- 2nd selection field - Contains exact same list of fields as "1st selection field" for "secondary" sort. First item in list will be default (similar to "1st selection field").

Detail

- List Button - Navigates to List view. This button's label changes to "Cancel" after any elements have been changed. When "Cancel" is available and pressed, a "confirmation dialog" will be displayed verifying the changes are to be cancelled; if "yes" is chosen to cancel the changes, the the Detail screen is displayed with the original state of the model and the button's label changes back to "List".

- Save Button - Saves the screen elements to the underlying model. For items to be created (e.g., "Id" element is blank), a new "Id" will be generated and displayed when the "save" is successful. The button always appears but is disabled when:
    - The "signed in" person is not an "administrator" or is not either the "Requested By" or "Assigned To" person; i.e., "administrators" can change any Request, and if not an "administrator" only a "logged in" person that is either the Requested By or Assigned To person

can change the Request.
    - No changes have been made to any of the fields; e.g., after an item has just been displayed from the List view, or when an item was just updated or created with no subsequent changes made to any of the elements.

- Copy Button - The purpose of this button is to easily enable a user to copy/clone an existing item (so all of the fields don't have to be re-entered). This button is enabled when the "Id" field is populated and no elements have been changed (a "Save" or "Cancel" would be required first before enabling "Copy"). It is also enabled even though the user is not authorized to change the item. When this button is pressed, the following occurs:
    - The following fields are blanked out: Id, Actual Finish Date, Created Date, Updated Date.
    - The Save button is enabled.
    - The Copy button is disabled.
    - The List button becomes a Cancel button (which will cancel the "copy" and re-render the original item).
    - The Delete button is disabled (since Id is now blanked out).
    - No changes are saved to the datastore until the user clicks the Save button.

- Delete Button - Enabled when an "Id" value exists. When clicked, a "confirmation dialog" is displayed; if confirmed, the underlying datastore item is deleted. The List view is re-rendered with the item removed.

Model

Notes:

- All "not null" strings must contain at least one character.

Request - A generic entity that could represent "requests" for service, an "order" entity, etc. Note: Below was extracted from a much more complex information model & is therefore significantly simplified.

```
id                  integer /* pk - auto-generate */
requestType         string /* not null; must be one of predefined set of values; loaded from json uri */
description         string /* not null */
status              string /* not null; must be one of predefined set of values - loaded from json uri */
priority            string /* not null; must be one of predefined set of values - loaded from json uri;
e.g., 1-high 2-medium 3-low; will keep numeric prefix for sorting */
requestedBy         string /* not null; fk - Person */
requestedFinishDate datetime /* not null; must be > createdDate - to illustrate error msg approach */
assignedTo          string /* nullable; fk - Person */
actualFinishDate    datetime /* nullable; must be > createdDate - to illustrate error msg approach */
estimatedUnits      numeric /* nullable; allow 2 decimals - to illustrate edit & error msg approach */
unitType            string /* nullable; must be one of predefined set of values; must be not null if
estimatedUnits is not null; loaded from json uri; hours, days, … */
createdDate         datetime /* auto-generate */
updatedDate         timestamp /* nullable; auto-generate for each update */
```

Person - This entity isn't being maintained by the reference implementation, but is used for demonstrating validations and protecting/enabling fields for an administrator.

```
id                  string  /* pk; will assume email address for simplicity */
```

firstName	string /* not null */
middleName	string /* nullable */
lastName	string /* not null */
administrator	boolean /* not null; true - is an administrator; otherwise not an administrator */
password	string /* nullable */

Wireframes

Phone

Link to Phone Wireframes

Tablet/Desktop

Link to Tablet/Desktop Wireframes

JSON Examples (just need a few more Requests)

Request

Person

Request Type

Status

Priority

Unit Type