Modify List View

List

The List View is generated for all DocTypes except Child Tables and Single DocTypes.

https://frappeframework.com/docs/user/en/api/list

Summary:
The "List View" in the Frappe Framework is a feature that provides a comprehensive view of all DocTypes, with the exception of Child Tables and Single DocTypes. This view comes with a variety of functionalities such as filters, sorting, paging, tag-based filtering, and the ability to switch views (e.g., Report, Calendar, Gantt, Kanban).

To customize this List View, a specific JavaScript file (`{doctype}_list.js`) is required in the doctype directory. For example, if you're customizing the "Note" DocType, you'd create a file named `note_list.js`. This file allows for various customizations, from adding fields to fetch, setting default filters, hiding columns, to defining actions when certain buttons are clicked.

Example:

```
// note_list.js
frappe.listview_settings['Note'] = {
    // add fields to fetch
    add_fields: ['title', 'public'],
    // set default filters
    filters: [
        ['public', '=', 1]
    ],
    hide_name_column: true, // hide the last column which shows the `name`
    onload(listview) {
        // triggers once before the list is loaded
    },
    before_render() {
        // triggers before every render of list records
    },

    // set this to true to apply indicator function on draft documents too
    has_indicator_for_draft: false,

    get_indicator(doc) {
        // customize indicator color
        if (doc.public) {
            return [__("Public"), "green", "public,=,Yes"];
        } else {
            return [__("Private"), "darkgrey", "public,=,No"];
        }
    },
    primary_action() {
        // triggers when the primary action is clicked
    },
    get_form_link(doc) {
        // override the form route for this doc
    },
    // add a custom button for each row
    button: {
        show(doc) {
            return doc.reference_name;
        },
        get_label() {
```

```
            return 'View';
        },
        get_description(doc) {
            return __('View {0}', [`${doc.reference_type} ${doc.reference_name}`])
        },
        action(doc) {
            frappe.set_route('Form', doc.reference_type, doc.reference_name);
        }
    },
    // format how a field value is shown
    formatters: {
        title(val) {
            return val.bold();
        },
        public(val) {
            return val ? 'Yes' : 'No';
        }
    }
}
```

Additionally, there's an option to customize the list view using "Client Script" in the system. This is especially useful if the logic is specific to your site. If you wish to share the customization across multiple sites, it's recommended to include them via Apps.

Key Takeaways:

1. **List View Features:** Filters, sorting, paging, and multiple view options.
2. **Customization:** Use `{doctype}_list.js` for specific DocType customizations.
3. **Client Script:** Allows for site-specific logic and broader sharing of customizations.

Examples/Analogies:
Imagine you have a bookshelf (List View) with various books (DocTypes). Now, you want to organize these books based on genre, author, or publication date. The customization options in Frappe are like giving you tools to sort, label, and arrange these books exactly how you want. The Client Script is like a special tool that lets you design unique labels or separators for your bookshelf, specific to your preferences.

▼ Examples

To add a button to the list view of a specific DocType, you should create a `{doctype}_list.js` file in the doctype directory.

Here's how you can do it:

1. Navigate to your `pentaho_processes` app's directory.
2. Go to the directory of the `Inter-app Connection` doctype. It should be something like: `pentaho_processes/pentaho_processes/doctype/inter_app_connection/`.
3. Create a new file named `inter_app_connection_list.js`.
4. Add the following code to this file:

```
frappe.listview_settings['Inter-app Connection'] = {
    onload(listview) {
        listview.page.add_button('Fetch and Log Customers', function() {
            // Call the server-side function
            frappe.call({
                method: 'pentaho_processes.logger.interapp_logger.fetch_and_log_customers',
                callback: function(r) {
                    if(!r.exc) {
                        frappe.msgprint("Customers fetched and logged successfully!");
                    }
                }
            });
        });
    }
};
```

1. Save the file.
2. Clear the cache (you can do this by running `bench clear-cache` in your terminal or clicking the `Reload` button on the Desk UI).
3. Refresh your list view for "Inter-app Connection".

The button should now appear in the list view.