New Field Documentation

When adding a new Field type to kora, here are all the files to take into consideration when developing the Field.

PHP

- **app/KoraFields/BaseField.php**
    - All new Fields need an app/KoraFields/* model that implements this abstract class.
- **app/Form.php**
    - There are several class variables that need to be considered for new Fields.
        - Create a constant representing the textual name of the new Field
        - $validFieldTypes - Add new Field constant to the appropriate Field sub-category
        - $validFilterFields - Add new Field constant here to support Record data filters
        - $validAssocFields - Add new Field constant here if the Field Record data can be preview in an associator search.
        - $fieldModelMap - Add mapping of new Field constant to the class name of the corresponding app/KoraFields/* model.
        - $jsonFields - Add new Field constant here if the Field uses the MYSQL JSON type for its database column.
        - $enumFields - Add new Field constant here if the Field uses the MYSQL ENUM type for its database column.
- **database/migrations/2017_00_00_000000_CreateRecordsTable.php**
    - If a new database column type is required for the new Field, please add a new add****Column() function to this file.
- **app/Http/Controllers/AdvancedSearchController.php**
    - On submission of an advanced search, the processRequest() function takes the html request data and formats it into a cleaner array that a Field can process.
- **app/Http/Controllers/FieldAjaxController.php**
    - Ajax calls made from javascript to the backend in laravel usually run through controllers. This controller hosts functions that are related to specific Field types, to call functions within the appropriate app/KoraFields/* model.
- **app/Console/Commands/ConvertField.php**
    - If you want to allow the new Field type to be converted to another existing type (or vice versa), add those rules here. Define what in the Field’s options array needs to change, as well as the conversion of existing Record data for that Field.

- **app/FieldValuePreset.php**
  - If the new Field takes advantage of the use of Field Value Presets for its Field options page, add the new Field to the $compatibility array here.
- **app/Http/Controllers/ExodusHelperController.php**
  - No need to add new Fields here, but this file manages converts old kora v2.* Field/Record data into kora v3.0.

JS

- **public/assets/javascripts/fields/options.js**
  - If the new Field needs javascript interaction (user interaction, Ajax requests) with its Field options page, define an initialize function for the Field and add it to the switch statement at the bottom.
- **public/assets/javascripts/fields/show.js**
  - For the single Field page, if the new Field needs validation rules defined for the various Field options, the validateAdvancedOptions() function runs these rules and reports the error messages involved.
- **public/assets/javascripts/records/create.js**
  - If the new Field needs javascript interaction (user interaction, Ajax requests) for the Record create page, define an initialize function for the Field and call it at the bottom.
  - Record presets allow the user to take a stored Record structure, and populate a new Record with the data from that preset. The Record's data, represented in the initializeRecordPresets() function, is the structure of how the data appears in the database. In the contained function, putArray(), define how that information is transferred from the Javascript to the Record input for the new Field.
- **public/assets/javascripts/fields/typedFieldDisplays.js**
  - For displaying Record data, some Fields need to be rendered/initialized after the page loads based on page width, offsets, etc. Those type of rules should be defined in here for a new Field.
- **public/assets/javascripts/fields/typedFieldInputs.js**
  - Similar to typedFieldDisplays.js above, this file holds post-load rendering rules for a Field's inputs relating to Record creation.
- **public/assets/javascripts/records/advanced.js**
  - If the new Field needs javascript interaction (user interaction, Ajax requests) for the Advanced Search page, define an initialize function for the Field and call it at the bottom.
- **public/assets/javascripts/records/batch.js**
  - If the new Field needs javascript interaction (user interaction, Ajax requests) for the Batch Assign Field Values page, define an initialize function for the Field and call it at the bottom.

BLADE

- **resources/views/partials/fields/options/***
  - This folder contains the inputs for defining Field options on the edit Field options page.
  - NOTE: These files contain inputs for a Field's respective default value, and includes other options in an included view. This is because the Combo List Field shares the Field options code, but handles default values differently.
- **resources/views/partials/fields/options/defaults/***
  - As noted above, these files include a Field's option inputs, separate from their default value inputs. Again, this is to allow the Combo-List Field to share this code. Note the use of the $seq variable.
- **resources/views/partials/fields/advanced/***
  - This folder contains the inputs for defining Field options on the create Field page.
- **resources/views/partials/fields/modals/***
  - If the new Field type needs a modal to support either Field options or Record creation, create the modal here.
- **resources/views/partials/records/input/***
  - Create a file here for the new Field's Record input. Define also how input is populated if the Field has a default value, or if editing a Field with existing Record data.
- **resources/views/partials/records/display/***
  - Create a file here for displaying the new Field's Record data.
- **resources/views/partials/records/advanced/***
  - If the new Field supports Advanced Search, create a file for the Field's search input.
- **resources/views/fields/***
  - The files formatted as single*****.blade.php are examples of special views for File Type Fields. kora has "open-in-new-tab" views for most of the Record data belonging to File Type Fields. Some data displays natively in browsers (i.e. an image from a Gallery Field), however some File Type Fields need a custom view. If adding a new File Type Field with a custom single view, do so here.

SCSS

- **resources/assets/scss/partials/inputs/***
  - These files define stylings for various html inputs, but also includes Field files for special Field types. These files can include the styling for Field options, Record input, and Record display. Place new Field stylings here if needed.
- **resources/assets/scss/partials/field-single/***
  - These files define stylings for the single*****.blade.php files found in resources/views/fields/*

Combo-List Field Documentation

The Combo-List Field pairs together two other Field types into one Field. To have the new Field support Combo-List functionality, please consider and modify these files.

PHP

- **app/Form.php**
    - Add the new Field to the $validComboListFieldTypes array in order to activate support.
- **app/RecordPresetController.php**
    - To prep a Record Preset for the create Record page, the getRecordArray() function gathers and formats the data for each Field type. For Combo-List value cards, there is a switch statement that determines what the html input and display values should be per Field type. If the new Field had a more complex data structure, format the values there.

JS

- **public/assets/javascripts/fields/options.js**
    - If the new Field has javascript interactions for the Field options page, recreate those here, modifying the html class names that the JS references to identify the numeric sequence of the sub Field if needed.
    - There are also two switch statements that add default values to the Field. These statements take user inputted data and build the html display and input values to create the Combo-List value card.
- **public/assets/javascripts/fields/options.js**
    - Similar to the Field options page, there are switch statements that define adding a Combo-List value in the create Record context.

BLADE

These Field files need to be modified so they can be used by both the Field itseld, and as a sub Field in the Combo-List Field. Please review a few of each of these to get ideas on how to implement.

- **resources/views/partials/fields/options/[new_field].blade.php**
    - Separate the Field option inputs and the Field default inputs into the config and default folders as partial views. Modify the new partial views to support Combo List.
- **resources/views/partials/records/input/[new_field].blade.php**
    - Field files here have cases for both new Record and edit Record. Add a new case for Combo List in record create.

- **resources/views/partials/records/input/combolist.blade.php**
  - Define here how default values are printed out for Combo-List new Record, and how Record values are printed out for Combo-List edit Record.
- **resources/views/partials/fields/advanced/[new_field].blade.php**
  - Modify the advanced search Field views to support use in Combo List advanced search.