# formly Documentation

*Release 0.7*

Eldarion

Jan 05, 2018

# Contents

*formly* is an app that provides an out of the box solution to building adhoc forms to collect data from end users. It is multi-faceted in that it provides interfaces for building multi-page forms, interfaces for executing the survey, as well as views for reviewing results.

Also, it is non-linear, meaning that you can route users taking the survey to different pages based on what they answered on certain questions. This allows you to create very rich surveys that dive deep on detail you care about while not wasting the time of users who would otherwise have to go through questions that do not apply to them.

This project is brought to you by Midwest Communications.

# CHAPTER 1

# Development

The source repository can be found at https://github.com/eldarion/formly/

## 1.1 Contents

### 1.1.1 ChangeLog

#### 0.14.0

- add hookset to support customizing available field type choices when designing a survey

#### 0.13.0

- fix field mapping bug (#30)
- improve output of MultipleTextField widget (#20)

#### 0.12.0

- fix broken migrations from 0.11.0

#### 0.11.0

- add support for Rating field

### 0.10.2

- fix app to work with a custom user module
- add missing migration for formly.Field

### 0.10.1

- fix Field.form_field() bug when Likert field has no choices

### 0.10

- add Likert-style field widget and presentation

### 0.9

- make label and help_text textfields

### 0.6

- changed field label descriptions to be more suitable for less technical audiences
- made compatible with Django > 1.5
- drop unique constraint on field label

### 0.5

- made urls Django 1.5 compatible
- add maximum_choices field
- drop unique constraint on field label

### 0.4.2

- fixed multiple choice field
- added survey to context

### 0.4.1

- fixed serialization bug, note this is a backwards incompatible change if you have previously stored results

### 0.4

- added authorization checks for all the views

### 0.3

- added ability to control redirection at the end of a survey

#### 0.2

- added ability to change the ordering of fields on a page

#### 0.1

- initial release

### 1.1.2 Installation

- Requirements:
- django-jsonfield==0.9.13
- Optional Requirements (to use the built in templates):
- pinax-theme-bootstrap (not required if you use different block names)
- django-forms-bootstrap (required for form rendering in templates)
- To install:

```
pip install formly
```

- Add `'formly'` to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
    # other apps
    "formly",
)
```

- Mount the `formly.urls` somewhere:

```
urlpatterns = patterns("",
    ...
    url(r"^surveys/", include("formly.urls")),
    ...
)
```

### 1.1.3 Usage

`formly` is designed to be pretty plug-and-play, in that after you install it, using it should be as simple as creating and publishing surveys through the web interface.

After installation, browse to whereever you mounted the urls for *formly* and you'll see an interface to be able to create a new survey. From here you can create a survey and begin editing pages. Pages in *formly* represent each step of the survey. The user will be guided through each page of the survey in the appropriate order saving each page at a time.

For each page, you can give a title and add as many fields as you desire. If the field is a type that requires choices, then you will have the option on the fields detail/edit form to add choices. An optional value that can be supplied for a choice answer is the page to redirect the user to if they select that answer.

If you have multiple choice fields in a single page with conflicting page routing, *formly* resolves to the first page it encounters. For example, if you had a question that had choice B route to page 3 and another question later in the form that had choice C route to page 5 and the user answered both questions with choice B and choice C, the user would go to page 3 next. Keep this in mind when building surveys.

### 1.1.4 Fields

`formly` enables you to create questions with a multitude of field types that will control the dynamic rendering and processing of form input on each page of your survey.

All field types tie directly to a specific `field` and `widget` configuration as found in `django.forms`. The other attributes that can be passed into every field, `label`, `help_text`, and `required` can be set and managed at design time.

For the field types that accept choices, there is the ability to set key/value pairs for each field that are used to populate the choices attribute for the field to be used for both display as well as form validation upon execution.

#### text field

The `text field` is a field for open ended text input and is interpreted as `django.forms.CharField`.

#### textarea

The `textarea` field type is a `django.forms.CharField` with a `django.forms.Textarea` widget to be used to collect longer form text input.

#### radio choices

The `radio choices` field type is a `django.forms.ChoiceField` with a `django.forms.RadioSelect` widget, populated with choices specified at design time.

#### dropdown field

The `dropdown field` is a select field generated from a `django.forms.ChoiceField` with a `django.forms.Select` widget, populated with choices specified at design time.

#### checkbox field

The `checkbox field` is a field generated from a `django.forms.MultipleChoiceField` with a `django.forms.CheckboxInput` widget, populated with choices specified at design time. This field allows for multiple selections.

#### date field

The `date field` provides a way to constrain input to dates only. It is generated from a `django.forms.DateField`.

#### media upload field

The `media upload field` enables users to upload content as a response. It is uses `django.forms.FileField`.

#### boolean field

The `boolean field` renders and processes input using `django.forms.BooleanField`.

#### multiple text field

The `multiple text` field type presents a number of single line fields. The number of fields is specified at design time.

#### likert scale field

The `likert scale` field type is a `django.forms.ChoiceField`, populated with choices specified at design time. The field template `formly/templates/bootstrapform/field.html` emits:

> **<ul class="likert-question">** {{ field }}
>
> </ul>

for hooking in CSS design. The following sample CSS presents a Likert field in familiar horizontal layout. You should add this (or similar) CSS to your project to get Likert-scale presentation.

> **form .likert-question {** list-style:none; width:100%; margin:0; padding:0 0 35px; display:block; border-bottom:2px solid #efefef;
>
> } form .likert-question:last-of-type {
>
> border-bottom:0;
>
> } form .likert-question:before {
>
> content: ‘’; position:relative; top:13px; left:13%; display:block; background-color:#dfdfdf; height:4px; width:75%;
>
> } form .likert-question li {
>
> display:inline-block; width:19%; text-align:center; vertical-align: top;
>
> } form .likert-question li input[type=radio] {
>
> display:block; position:relative; top:0; left:50%; margin-left:-6px;
>
> } form .likert-question li label {
>
> width:100%;
>
> }

#### rating scale field

The `rating scale` field type is a `django.forms.ChoiceField`, populated with choices specified at design time. The field template `formly/templates/bootstrapform/field.html` emits:

> **<ul class="rating-question">** {{ field }}
>
> </ul>

### 1.1.5 Templates

`formly` ships with some stock templates that are based on `pinax-theme-bootstrap` and `django-forms-bootstrap`. You are not required to use these of course and in case you are rolling your own templates, here is what the views in `formly` expect.

### `formly/design/choice_form.html`

**Context** `form`, `choice`, `page`

**Extends** `formly/design/survey_edit_base.html`

This is the template that provides the ability to update the values for a particular choice for a choice field.

### `formly/design/field_confirm_delete.html`

**Context** `form`, `field`

**Extends** `site_base.html`

This is the template is rendered to supply a delete confirmation form for field deletion.

### `formly/design/field_form.html`

**Context** `form`, `field`, `page`, `field_choice_form`

**Extends** `formly/design/survey_edit_base.html`

This is the template is rendered for a user interface to update a field.

### `formly/design/fieldchoice_confirm_delete.html`

**Context** `form`, `fieldchoice`

**Extends** `site_base.html`

This is the template is rendered to supply a delete confirmation form for field choice deletion.

### `formly/design/page_confirm_delete.html`

**Context** `form`, `page`

**Extends** `site_base.html`

This is the template is rendered to supply a delete confirmation form for page deletion.

### `formly/design/page_form.html`

**Context** `form`, `page`, `field_form`

**Extends** `formly/design/survey_edit_base.html`

This is the template is that displays the user interface for updating a page object.

### `formly/design/survey_confirm_delete.html`

**Context** `form`, `survey`

**Extends** `site_base.html`

This is the template is rendered to supply a delete confirmation form for survey deletion.

### `formly/design/survey_detail.html`

**Context** `survey`

**Extends** `site_base.html`

This template displays the detail for a survey.

### `formly/design/survey_edit_base.html`

**Context** `page`

**Extends** `subnav_base.html`

**Extended By** `formly/design/choice_form.html`, `formly/design/field_form.html`, `formly/design/page_form.html`

This a base template to provide some common subnav.

### `formly/design/survey_form.html`

**Context** `form`

**Extends** `site_base.html`

This template hosts the creation form for creating a new survey object.

### `formly/design/survey_list.html`

**Context** `unpublished_surveys`, `published_surveys`

**Extends** `site_base.html`

This template receives all surveys in the system split between two context objects, one for published surveys and the other for unpublished surveys.

### `formly/results/home.html`

**Context** `survey`

**Extends** `site_base.html`

A template for displaying the results of a given survey.

### `formly/run/page.html`

**Context** `form`, `page`

**Extends** `site_base.html`

This template is rendered for the end user to complete a particular survey, it is always rendered with the appropriate page for the user.

#### `formly/bootstrapform/field.html`

**Context** `field`

This modified `django-bootstrap-form` template renders the various field types, including special handling for Likert and Rating fields.

### 1.1.6 Authorization

`formly` ships with an auth backend that by default, when added to your `AUTHENTICATION_BACKENDS` setting will segment the create, edit, delete and results viewing based on the `reauest.user` being the `Survey.creator`.

You can override this by writing your own auth backend and using in it's place.

The permission labels used are as follows:

#### formly.view_survey_list

Can the user see the list of published and unpublished surveys

#### formly.create_survey

Can the user create a survey

#### formly.view_survey_detail

Can the user view the survey's detail. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.change_survey_name

Can the user change the survey's name. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.publish_survey

Can the user publish the survey. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.duplicate_survey

Can the user duplicate the survey. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.edit_survey

Can the user edit the survey. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.view_results

Can the user view the survey's results. The survey object in question is passed to the `has_perm` method of the auth backend.

#### formly.delete_object

Can the user delete the object in question. The object will be either a `Survey`, `Page`, `Field`, or a `FieldChoice`.

### 1.1.7 Callbacks

Callbacks are a way to provide functionality to formly that requires some runtime decision making instead of just a setting. They are callables that are defined in settings and ship some sane defaults.

#### `FORMLY_COMPLETE_REDIRECT_CALLBACK`

**Default** `formly.callbacks.survey_complete_redirect`

**Arguments** `survey`

**Expected Return** a url that will be passed to `redirect()`