# newforms Documentation

***Release 0.4.1***

**Jonathan Buchanan**

March 12, 2014

**Note:** Unless specified otherwise, API items documented herein live under the `forms` namespace object in the browser, or whatever namespace you care to use when requiring newforms in Node.js.

Contents:

CHAPTER 1

# Forms

## 1.1 Guide

TBD

## 1.2 API

**class** `BaseForm` ( [*kwargs*] )

A collection of Fields that knows how to validate and display itself.

**Arguments**

- **kwargs** (*Object*) – form options.

`kwargs.`**`data`**

input form data, where property names are field names. A form with data is considered to be "bound" and ready for use validating and coercing the given data.

`kwargs.`**`files`**

input file data.

`kwargs.`**`autoId`**

a template for use when automatically generating `id` attributes for fields, which should contain a `{name}` placeholder for the field name – defaults to `id_{name}`.

`kwargs.`**`prefix`**

a prefix to be applied to the name of each field in this instance of the form - using a prefix allows you to easily work with multiple instances of the same Form object in the same HTML `<form>`, or to safely mix Form objects which have fields with the same names.

`kwargs.`**`initial`**

initial form data, where property names are field names - if a field's value is not specified in `data`, these values will be used when rendering field widgets.

`kwargs.`**`errorConstructor`**

the constructor function to be used when creating error details - defaults to `ErrorList()`.

`kwargs.`**`labelSuffix`**

a suffix to be used when generating labels in one of the convenience methods which renders the entire Form - defaults to `':'`.

`kwargs.`**`emptyPermitted`**

if `true`, the form is allowed to be empty – defaults to `false`.

**isValid**()
Determines whether or not the form has errors, triggering cleaning of the form first if necessary.

**Returns** `true` if the form is bound and has no errors, `false` otherwise. If errors are being ignored, returns `false`.

**errors**()
Getter for errors, which first cleans the form if there are no errors defined yet.

**class BoundField**(*form*, *field*, *name*)
A field and its associated data.

**Arguments**

- **form** (*Form*) – a form.
- **field** (*Field*) – one of the form's fields.
- **name** (*String*) – the name under which the field is held in the form

**DeclarativeFieldsMeta**(*{prototypeProps*, *constructorProps}*)
This function is responsible for setting up form fields when a new Form constructor is being created. It pops any Fields it finds off the form's prototype properties object, determines if any forms are also being mixed-in via a __mixin__ property and handles inheritance of Fields from any form which is being inherited from such that fields will be given this order of precedence should there be a naming conflict with any of these three sources.

1.Fields specified in the prototype properties

2.Fields from a mixed-in form

3.Fields from the Form being inherited from

**class Form**([*kwargs*])
Inherits from BaseForm and registers DeclarativeFieldsMeta to be used to set up Fields when this constructor is inherited from.

It is intended as the entry point for defining your own forms. You can do this using its `extend()` function, which is provided by Concur

..js:function:: Form.extend({prototypeProps, constructorProps})

Creates a new constrctor which inherits from Form. The new form's fields and prototype properties, such as validation methods, should be passed as `prototypeProps`.

CHAPTER 2

# Fields

## 2.1 Guide

TBD

## 2.2 API

**class** `Field`([*kwargs*])

An object that is responsible for doing validation and normalisation, or "cleaning" – for example: an EmailField makes sure its data is a valid e-mail address – and makes sure that acceptable "blank" values all have the same representation.

**Arguments**

- **kwargs** (*Object*) – field options.

`kwargs.`**`required`**
determines if the field is required - defaults to `true`.

`kwargs.`**`widget`**
overrides the widget used to render the field - if not provided, the field's default will be used.

`kwargs.`**`label`**
the label to be displayed for the field - if not provided, will be generated from the field's name.

`kwargs.`**`initial`**
an initial value for the field to be used if none is specified by the field's form.

`kwargs.`**`helpText`**
help text for the field.

`kwargs.`**`errorMessages`**
custom error messages for the field.

`kwargs.`**`showHiddenInitial`**
specifies if it is necessary to render a hidden widget with initial value after the widget.

`kwargs.`**`validators`**
list of addtional validators to use

**class** `CharField`([*kwargs*])

Validates that its input is a valid string.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**`maxLength`**
a maximum valid length for the input string.

`kwargs.`**`minLength`**
a minimum valid length for the input string.

**class** **`IntegerField`**([*kwargs*])
Validates that its input is a valid integer.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**`maxValue`**
a maximum value for the input.

`kwargs.`**`minValue`**
a minimum value for the input.

**class** **`FloatField`**([*kwargs*])
Validates that its input is a valid float.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**`maxValue`**
a maximum value for the input.

`kwargs.`**`minValue`**
a minimum value for the input.

**class** **`DecimalField`**([*kwargs*])
Validates that its input is a decimal number.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**`maxValue`**
a maximum value for the input.

`kwargs.`**`minValue`**
a minimum value for the input.

`kwargs.`**`maxDigits`**
the maximum number of digits the input may contain.

`kwargs.`**`decimalPlaces`**
the maximum number of decimal places the input may contain.

**class** **`BaseTemporalField`**([*kwargs*])
Base field for fields which validate that their input is a date or time.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**`inputFormats`**
a list of time.strptime input formats which are considered valid.

**class** **`DateField`**([*kwargs*])
Validates that its input is a date.

**class** **TimeField**([*kwargs*])
Validates that its input is a time.

**class** **DateTimeField**([*kwargs*])
Validates that its input is a date/time.

**class** **RegexField**([*kwargs*])
Validates that its input matches a given regular expression.

**Arguments**

- **regex** (*RegExp|String*) – a regular expression.
- **kwargs** (*Object*) – field options, as specified in Field and CharField.

**class** **EmailField**([*kwargs*])
Validates that its input appears to be a valid e-mail address.

**class** **FileField**([*kwargs*])
Validates that its input is a valid uploaded file.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**allowEmptyFile**
`true` if empty files are allowed - defaults to `false`.

**class** **ImageField**([*kwargs*])
Validates that its input is a valid uploaded image.

**Arguments**

- **kwargs** (*Object*) – field options, as specified in FileField.

**class** **URLField**([*kwargs*])
Validates that its input appears to be a valid URL.

**class** **BooleanField**([*kwargs*])
Normalises its input to a `Boolean`.

**class** **NullBooleanField**([*kwargs*])
A field whose valid values are `null`, `true` and `false`.

Invalid values are cleaned to `null`.

**class** **ChoiceField**([*kwargs*])
Validates that its input is one of a valid list of choices.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.`**choices**
a list of choices - each choice should be specified as a list containing two items; the first item is a value which should be validated against, the second item is a display value for that choice, for example:

```
{choices: [[1, 'One'], [2, 'Two']]}
```

Defaults to `[]`.

**class** **TypedChoiceField**([*kwargs*])
A ChoiceField which returns a value coerced by some provided function.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in ChoiceField.

`kwargs.coerce`
a function which takes the String value output by ChoiceField's clean method and coerces it to another type - defaults to a function which returns the given value unaltered.

`kwargs.emptyValue`
the value which should be returned if the selected value can be validly empty - defaults to `''`.

**class `MultipleChoiceField`** ( [*kwargs*] )
Validates that its input is one or more of a valid list of choices.

**class `TypedMultipleChoiceField`** ( [*kwargs*] )
A MultipleChoiceField} which returns values coerced by some provided function.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in MultipleChoiceField.

`kwargs.coerce`
function which takes the String values output by MultipleChoiceField's toJavaScript method and coerces it to another type - defaults to a function which returns the given value unaltered.

`kwargs.emptyValue`
the value which should be returned if the selected value can be validly empty - defaults to `''`.

**class `FilePathField`** ( [*kwargs*] )
Allows choosing from files inside a certain directory.

**Arguments**

- **path** (*String*) – The absolute path to the directory whose contents you want listed - this directory must exist.
- **kwargs** (*Object*) – field options additional to those supplied in ChoiceField.

`kwargs.match`
a regular expression pattern - if provided, only files with names matching this expression will be allowed as choices.

`kwargs.recursive`
if `true`, the directory will be descended into recursively and all descendants will be listed as choices - defaults to `false`.

**class `ComboField`** ( [*kwargs*] )
A Field whose `clean()` method calls multiple Field `clean()` methods.

**Arguments**

- **kwargs** (*Object*) – field options additional to those specified in Field.

`kwargs.fields`
fields which will be used to perform cleaning, in the order they're given.

**class `MultiValueField`** ( [*kwargs*] )
A Field that aggregates the logic of multiple Fields.

Its `clean()` method takes a "decompressed" list of values, which are then cleaned into a single value according to `this.fields`. Each value in this list is cleaned by the corresponding field – the first value is cleaned by the first field, the second value is cleaned by the second field, etc. Once all fields are cleaned, the list of clean values is "compressed" into a single value.

Subclasses should not have to implement `clean()`. Instead, they must implement `compress()`, which takes a list of valid values and returns a "compressed" version of those values – a single value.

You'll probably want to use this with MultiWidget.

**Arguments**

- **kwargs** (*Object*) – field options additional to those supplied in Field.

kwargs.**fields**
a list of fields to be used to clean a "decompressed" list of values.

**class SplitDateTimeField**([*kwargs*])
A MultiValueField consisting of a DateField and a TimeField.

**Arguments**

- **kwargs** (*Object*) – field options options, as specified in Field and MultiValueField.

**class IPAddressField**([*kwargs*])
Validates that its input is a valid IPv4 address.

**class GenericIPAddressField**([*kwargs*])
Validates that its input is a valid IPv4 or IPv6 address.

**class SlugField**([*kwargs*])
Validates that its input is a valid slug.

CHAPTER 3

# Widgets

## 3.1 Guide

TBD

## 3.2 API

**class** `Widget` ( [*kwargs* ] )
An HTML form widget.

A widget handles the rendering of HTML, and the extraction of data from an object that corresponds to the widget.

**Arguments**

- **kwargs** (*Object*) – widget options.

`kwargs.attrs`
HTML attributes for the rendered widget.

**class** `Input` ( [*kwargs* ] )
An HTML <input> widget.

**class** `TextInput` ( [*kwargs* ] )
An HTML <input type="text"> widget

**Arguments**

- **kwargs** (*Object*) – widget options.

**class** `PasswordInput` ( [*kwargs* ] )
An HTML `<input type="password">` widget.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Input.

`kwargs.renderValue`
if `false` a value will not be rendered for this field - defaults to `false`.

**class** `HiddenInput` ( [*kwargs* ] )

An HTML <input type="hidden"> widget.

**param Object kwargs** widget options.

**class `MultipleHiddenInput`** ( [*kwargs* ] )
A widget that handles <input type="hidden"> for fields that have a list of values.

**class `FileInput`** ( [*kwargs* ] )

An HTML <input type="file"> widget.

**param Object kwargs** widget options.

**class `ClearableFileInput`** ( [*kwargs* ] )

**Arguments**

- **kwargs** (*Object*) – widget options.

**class `DateInput`** ( [*kwargs* ] )
A <input type="text"> which, if given a Date object to display, formats it as an appropriate date string.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Input.

(kwargs attribute)

`kwargs.`**`format (String}`**
a time.strftime date format string.

**class `DateTimeInput`** ( [*kwargs* ] )
A <input type="text"> which, if given a Date object to display, formats it as an appropriate datetime string.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Input.

(kwargs attribute)

`kwargs.`**`format (String}`**
a time.strftime datetime format string.

**class `TimeInput`** ( [*kwargs* ] )
A <input type="text"> which, if given a Date object to display, formats it as an appropriate time string.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Input.

(kwargs attribute)

`kwargs.`**`format (String}`**
a time.strftime time format string.

**class `CheckboxInput`** ( [*kwargs* ] )
An HTML `<input type="checkbox">` widget.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Widget.

`kwargs.`**`checkTest`**
a function which takes a value and returns `true` if the checkbox should be checked for that value.

**class `Textarea`** ( [*kwargs* ] )
An HTML `<textarea>` widget.

**Arguments**

- **kwargs** (*Object*) – widget options

Default rows and cols HTML attributes will be used if not provided.

**class** `Select` ([*kwargs*])
An HTML `<select>` widget.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Widget.

`kwargs.choices`
choices to be used when rendering the widget, with each choice specified as an Array in `[value, text]` format.

**class** `NullBooleanSelect` ([*kwargs*])
A `<select>` widget intended to be used with NullBooleanField.

**Arguments**

- **kwargs** (*Object*) – widget options, as specified in Select. Any `choices` provided will be overrridden with the specific choices this widget requires.

**class** `SelectMultiple` ([*kwargs*])
An HTML `<select>` widget which allows multiple selections.

**Arguments**

- **kwargs** (*Object*) – widget options, as specified in Select.

**class** `RadioSelect` ([*kwargs*])
Renders a single select as a list of `<input type="radio">` elements.

**Arguments**

- **kwargs** (*Object*) – widget options additional to those specified in Select.

`kwargs.renderer`
a custom RadioFieldRenderer constructor.

**class** `RadioFieldRenderer` (*name*, *value*, *attrs*, *choices*)
An object used by RadioSelect to enable customisation of radio widgets.

**Arguments**

- **name** (*String*) – the field name.
- **value** (*String*) – the selected value.
- **attrs** (*Object*) – HTML attributes for the widget.
- **choices** (*Array*) – choices to be used when rendering the widget, with each choice specified as an Array in `[value, text]` format.

**class** `RadioInput` (*name*, *value*, *attrs*, *choice*, *index*)
An object used by RadioFieldRenderer that represents a single `<input type="radio">`.

**Arguments**

- **name** (*String*) – the field name.
- **value** (*String*) – the selected value.
- **attrs** (*Object*) – HTML attributes for the widget.
- **choice** (*Array*) – choice details to be used when rendering the widget, specified as an Array in `[value, text]` format.
- **index** (*Number*) – the index of the radio button this widget represents.

**class** `CheckboxSelectMultiple` ([*kwargs*])
Multiple selections represented as a list of `<input type="checkbox">` widgets.

**Arguments**

- **kwargs** (*Object*) – widget options, as specified in Select.

**class** `MultiWidget` (*widgets*[, *kwargs* ])
A widget that is composed of multiple widgets.

You'll probably want to use this class with MultiValueField.

**Arguments**

- **widgets** (*Array*) – the list of widgets composing this widget.
- **kwargs** (*Object*) – widget options.

**class** `SplitDateTimeWidget` ([*kwargs* ])
Splits Date input into two `<input type="text">` elements.

**Arguments**

- **kwargs** (*Object*) – widget optionsadditional to those specified in MultiWidget.
- **[dateFormat]** (*String*) – a time.strftime date format string
- **[timeFormat]** (*String*) – a time.strftime time format string

**class** `SplitHiddenDateTimeWidget` ([*kwargs* ])
Splits Date input into two `<input type="hidden">` elements.

CHAPTER 4

# FormSets

## 4.1 Guide

TBD

## 4.2 API

**class** `BaseFormSet` ( [*kwargs* ] )

A collection of instances of the same Form.

**Arguments**

- **kwargs** (*Object*) – configuration options.

`kwargs.`**`data`**

input form data, where property names are field names.

`kwargs.`**`files`**

input file data.

`kwargs.`**`autoId`**

a template for use when automatically generating `id` attributes for fields, which should contain a `{name}` placeholder for the field name – defaults to `id_{name}`.

`kwargs.`**`prefix`**

a prefix to be applied to the name of each field in each form instance.

`kwargs.`**`initial`**

a list of initial form data objects, where property names are field names - if a field's value is not specified in `data`, these values will be used when rendering field widgets.

`kwargs.`**`errorConstructor`**

the constructor function to be used when creating error details - defaults to ErrorList.

**`formsetFactory`** (*form*[, *kwargs* ] )

Returns a FormSet constructor for the given Form constructor.

**Arguments**

- **form** (*Form*) – the constructor for the Form to be managed.
- **kwargs** (*Object*) – arguments defining options for the created FormSet constructor - all arguments other than those defined below will be added to the new formset constructor's

`prototype`, so this object can also be used to define new methods on the resulting formset, such as a custom `clean` method.

`kwargs.`**`formset`**
: the constructuer which will provide the prototype for the created FormSet constructor – defaults to `BaseFormSet`.

`kwargs.`**`extra`**
: the number of extra forms to be displayed – defaults to `1`.

`kwargs.`**`canOrder`**
: if `true`, forms can be ordered – defaults to `false`.

`kwargs.`**`canDelete`**
: if `true`, forms can be deleted – defaults to `false`.

`kwargs.`**`maxNum`**
: the maximum number of forms to be displayed – defaults to `0`.

CHAPTER 5

# Models

## 5.1 Stability

The models modules is a work-in-progress based on limited experimentation with a sync API - it is likely to change extensively to be able to handle an async Model API.

## 5.2 API

**`ModelInterface`**
A means of hooking newforms up with information about your model layer.

**`throwsIfNotFound`**
Set to `true` if an exception is thrown when a model can't be found.

**`notFoundErrorConstructor`**
Constructor of error to ve thrown when a model can't be found. Any exceptions which do not have this constructor will be rethrown.

**`notFoundValue`**
Value returned to indicate not found, instead of throwing an exception.

**`prepareValue`**
Given a model instance, should return the id which will be used to search for valid choices on submission.

**`findById`**
Finds a model instance by id, given the model query which was passed to newforms and the id of the selected model.

**class `ModelChoiceField`**(*modelQuery*[, *kwargs*])
A ChoiceField which retrieves its choices as objects returned by a given function.

**Arguments**

- **modelQuery** (*Function*) – an object which performs a query for model instances - this is expected to have an `__iter__` method which returns a list of instances.
- **kwargs** (*Object*) – field options

`kwargs.`**`cacheChoice`**
if `true`, the model query function will only be called the first time it is needed, otherwise it will be called every time the field is rendered.

CHAPTER 6

# Utilities

## 6.1 Core utilities

Utilities which are of partcular interest when inplementing your own form components are exposed on the `forms` namespace object.

**class `ErrorObject`** (*errors*)

`ErrorObject.`**`set`**`()`

`ErrorObject.`**`get`**`()`

`ErrorObject.`**`isPopulated`**`()`

`ErrorObject.`**`asUL`**`()`

`ErrorObject.`**`asText`**`()`

`ErrorObject.`**`defaultRendering`**`()`

`ErrorObject.`**`toString`**`()`

**class `ErrorList`** (*errors*)

`ErrorList.`**`extend`**`()`

`ErrorList.`**`isPopulated`**`()`

`ErrorList.`**`asUL`**`()`

`ErrorList.`**`asText`**`()`

`ErrorList.`**`defaultRendering`**`()`

`ErrorList.`**`toString`**`()`

**class `ValidationError`** (*message*, *{code: null*, *params: null}*)

Thrown by validation methods to indicate that invalid data was detected. The message argument can be a list of strings or a string, in which case an error code and a message parameters object can be provided to enable customisation of the resulting error message based on the code.

This constructor is actually provided by the validators package, but is exposed as part of newforms' exports for convenience.

**`formData`** (*form*)

Creates an object representation of a form's contents.