SpringField Documentation

Release 0.7.7

Mike Thornton

May 03, 2017

Contents

Contents:

CHAPTER 1

Entities

class springfield.**Entity**(**values*)

flatten()
Get the values as basic Python types

get(*key*, *default=None*, *empty=False*)
Get a value by key. If passed an iterable, get a dictionary of values matching keys.

Parameters **empty** – boolean - Include empty values

jsonify()
Return a dictionary suitable for JSON encoding.

to_json()
Convert the entity to a JSON string.

update(*values*)
Update attibutes. Ignore keys that aren't fields. Allows dot notation.

CHAPTER 2

Fields

class springfield.fields.**AdaptableTypeField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that has a specific type and can be adapted to another type.

adapt(*value*)
Convert the *value* to the *self.type* for this *Field*

classmethod register_adapter(*from_cls*, *func*)
Register a function that can handle adapting from *from_cls* for this field.

TODO This may be a bad idea, re-evaluate how to register adapters.

class springfield.fields.**BooleanField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that contains a *bool*.

adapt(*value*)
Adapt *value* to a *bool*.

Parameters **value** – A boolean-like value.

A *float*, *int*, or *long* will be converted to:

- *True* if equal to *1*
- *False* if equal to *0*

String values will be converted to (case-insensitive):

- *True* if equal to "yes", "true", "1", or "on"
- *False* if equal to "no", "false", "0", or "off"

type
alias of bool

class springfield.fields.**BytesField**(*encoding='base64'*, **args*, ***kwargs*)
A *Field* that contains binary *bytes*.

The field has an encoding to use for json/unicode conversion, such as *'base64'* (the default) or *'hex'*.

If *encoding* == *None*, no encoding/decoding is performed for JSON/unicode values which mean JSON itself will have to escape the bytes using unicode escapes where necessary. This is most suitable for cases where the "bytes" are known to be ASCII 7-bit safe.

The encoding is used in *adapt* if the input is a *unicode* instance, and in *jsonify* always.

adapt(*value*)
If the input is unicode, decode it into bytes. If it is already bytes, it is returned unchanged.

If an encoding was specific for the field, it is applied here if the input is *unicode*.

This assumes that the unicode only contains code points in the valid ranges for a byte - e.g. 0-255.

Parameters **value** – Value to decode

Returns *bytes* object

jsonify(*value*)
Encode the bytes into a unicode string suitable for json encoding.

If an encoding was specified for the field, it is applied here.

type
alias of `str`

class springfield.fields.**CollectionField**(*field*, **args*, ***kwargs*)
A *Field* that can contain an ordered list of values matching a specific *Field* type.

adapt(*value*)
Adapt all values of an iterable to the *CollectionField*'s field type.

flatten(*value*)
Convert all values of an iterable to the *CollectionField*'s field type's native Python type.

jsonify(*value*)
Convert all values of an iterable to the *CollectionField*'s field type's JSON type.

class springfield.fields.**DateTimeField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
Field whose value is a Python *datetime.datetime*

adapt(*value*)
Adapt *value* to a *datetime.datetime* instance.

Parameters **value** – A date-like value. RFC3339 formatted date-strings are supported.

If *dateutil* is installed, *dateutil.parser.parse* is used which supports many date formats.

jsonify(*value*)
Get the date as a RFC3339 date-string

type
alias of `datetime`

class springfield.fields.**EmailField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
Field with an email value

class springfield.fields.**EntityField**(*entity*, **args*, ***kwargs*)
Field that can contain an `Entity`

flatten(*value*)
Convert an `Entity` to a *dict* containing native Python types.

jsonify(*value*)
Convert an `Entity` into a JSON object

type
Determine the type of the Entity that will be instantiated.

There are three ways to reference an Entity when using an EntityField:

- **'self': A byte string referencing the class that is defining this** EntityField as an attribute.
- **'{dotted.name.kls}': A byte string referencing an importable callable** that can be instantiated at field-instantiation time.
- {Entity}: A type that subclasses *Entity*.

The order of operations during instantiation and resolution of the above references is important; During the creation of an *Entity*, the metaclass will call *init()* for fields defined on the class. This is useful for the 'self' reference so that the *EntityField* can be initialized with the class that is being instantiated during creation of the instance. For dotted-name class strings, this is too early since the dotted-name reference may not exist yet. For this reason, resolving the dotted-name reference is deferred to be as late as possible, in this case on the first read of the *type* property of this field.

The dotted-name references are stored in a map on the *EntityField* class to prevent resolving and importing the dotted-name on every instance of this *EntityField*.

Returns:

***type*: A type to use when instantiating the Entity for this** EntityField.

class springfield.fields.**Field**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A field

adapt(*value*)
Convert the value from the input type to the expected type if needed.

Returns The adapted value

flatten(*value*)
Get the value as a basic Python type

Parameters **value** – An Entity's value for this *Field*

init(*cls*)
Initialize the field for its owner Entity class. Any specialization that needs to be done based on the Entity class itself should be done here.

Parameters **cls** – An Entity class.

jsonify(*value*)
Get the value as a suitable JSON type

Parameters **value** – An Entity's value for this *Field*

make_descriptor(*name*)
Create a descriptor for this *Field* to attach to an Entity.

class springfield.fields.**FieldDescriptor**(*name*, *field*)
A descriptor that handles setting and getting *Field* values on an Entity.

class springfield.fields.**FloatField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that contains a *float*.

adapt(*value*)
Adapt *value* to a *float*.

Parameters **value** – Can be an *int*, *float*, *long*, or a *str* or *unicode* that looks like a *float*.

long values will remain 'long's.

type
alias of `float`

class springfield.fields.**IdField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that is used as the primary identifier for an `Entity`

TODO This should accept another Field type to contain the ID

class springfield.fields.**IntField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that contains an *int*.

adapt(*value*)
Adapt *value* to an *int*.

Parameters **value** – Can be an *int*, *float*, *long*, or a *str* or *unicode* that looks like an *int*.

float or *long* values must represent an integer, i.e. no decimal places.

type
alias of `int`

class springfield.fields.**SlugField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
Field whose value is a slugified string.

A slug is a string converted to lowercase with whitespace replace with a "-" and non-ascii chars converted to their ascii equivalents.

adapt(*value*)
Adapt *value* to a slugified string.

Parameters **value** – Any string-like value

class springfield.fields.**StringField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
A *Field* that contains a unicode string.

adapt(*value*)
Adapt *value* to *unicode*.

type
alias of `unicode`

class springfield.fields.**UrlField**(*default=Empty*, *doc=None*, **args*, ***kwargs*)
Field with a URL value

adapt(*value*)
Validate that the *value* has a valid URL format containing a scheme and network location using urlparse.

Parameters **value** – A url-like value.

Returns URL with sheme and network location in lower case.

CHAPTER 3

timeutil

springfield.timeutil.**date_parse**(*date*)
Parse an RFC3339 formatted time string into a datetime object.

Assumes input is UTC.

springfield.timeutil.**generate_rfc3339**(*value*)
Converts a datetime to an RFC3339 formatted time string.

Input is always converted to UTC.

Parameters **value** – A datetime instance

springfield.timeutil.**utcnow**()
Returns the current time in TZ aware UTC.

Returns datetime instance

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

s

t

Index

U