traittypes Documentation

Release 0.0.6

IPython contributors

September 23, 2016

Contents

CHAPTER 1

Introduction

The *traittypes* module provides a robust reference implementation of trait types for common data structures used in the scipy stack such as

- numpy arrays
- pandas and xarray data structures

which are out of the scope of the main traitlets project but are a common requirement to build applications with traitlets in combination with the scipy stack.

Another goal is to create adequate serialization and deserialization routines for these trait types to be used with the ipywidgets project (`to_json` and `from_json`). These could also return a list of binary buffers as allowed by the current messaging protocol.

CHAPTER 2

Usage

2.1 Example: Validating the Shape of a Numpy Array

We pass a validation function to the `valid` method of the `Array` trait type.

In this example, the validation function is returned by the `shape` closure which stores the tuple in its closure.

```
from traitlets import HasTraits, TraitError
from traittypes import Array

def shape(*dimensions):
    def validator(trait, value):
        if value.shape != dimensions:
            raise TraitError('Expected an of shape %s and got and array with shape %s' % (dimensions,
        else:
            return value
    return validator

class Foo(HasTraits):
    bar = Array(np.identity(2)).valid(shape(2, 2))
foo = Foo()

foo.bar = [1, 2]  # Should raise a TraitError
```

CHAPTER 3

API Reference Documentation

The `SciType` trait type is the base trait type for all Scipy trait types.

It complements the `traitlets.TraitType` with a special API to register custom validators.

class `traittypes.traittypes.`**`SciType`**(*default_value=traitlets.Undefined*, *allow_none=False*, *read_only=None*, *help=None*, ***kwargs*)

A base trait type for numpy arrays, pandas dataframes and series.

`valid`(**validators*)

Register new trait validators

Validators are functions that take two arguments.

- The trait instance
- The proposed value

Validators return the (potentially modified) value, which is either assigned to the HasTraits attribute or input into the next validator.

They are evaluated in the order in which they are provided to the *valid* function.

Example

```
        # Test with a shape constraint
        def shape(*dimensions):
            def validator(trait, value):
                if value.shape != dimensions:
                    raise TraitError('Expected an of shape %s and got and array with shape %s' % (di
                else:
                    return value
            return validator

        class Foo(HasTraits):
            bar = Array(np.identity(2)).valid(shape(2, 2))
        foo = Foo()

        foo.bar = [1, 2]  # Should raise a TraitError
```

The `Array` trait type holds a numpy Array.

class `traittypes.traittypes.`**`Array`**(*default_value=traitlets.Undefined*, *allow_none=False*, *dtype=None*, ***kwargs*)

A numpy array trait type.

The `DataFrame` trait type holds a pandas DataFrame.

class `traittypes.traittypes.`**`DataFrame`**(*default_value=traitlets.Undefined*, *allow_none=False*, *dtype=None*, ***kwargs*)

A pandas dataframe trait type.

The `Series` trait type holds a pandas Series.

class `traittypes.traittypes.`**`Series`**(*default_value=traitlets.Undefined*, *allow_none=False*, *dtype=None*, ***kwargs*)

A pandas series trait type.

A

D

S

V