objectfactory

Release 0.1.0

Devin A. Conley

Sep 28, 2021

CONTENTS:

objectfactory is a python package to easily implement the factory design pattern for object creation, serialization, and polymorphism

- designed to support polymorphism
- integrates seamlessly with marshmallow and other serialization frameworks
- schema inherent in class definition
- load any object with a generic interface
- serialize objects to JSON

CHAPTER

ONE

QUICKSTART

objectfactory is a python package to easily implement the factory design pattern for object creation, serialization, and polymorphism

- designed to support polymorphism
- integrates seamlessly with marshmallow and other serialization frameworks
- schema inherent in class definition
- load any object with a generic interface
- serialize objects to JSON

1.1 Example

Simple **shapes** example:

```
import objectfactory

@objectfactory.register
class Square( objectfactory.Serializable ):
    side = objectfactory.Field()

    def get_area( self ):
        return self.side * self.side

@objectfactory.register
class Triangle( objectfactory.Serializable ):
    base = objectfactory.Field()
    height = objectfactory.Field()

    def get_area( self ):
        return 0.5 * self.base * self.height

serialized_data = [
    {"_type": "Square", "side": 2.0},
    {"_type": "Triangle", "base": 1.75, "height": 2.50},
    {"_type": "Square", "side": 1.5},
```

(continues on next page)

(continued from previous page)

```
]

for data in serialized_data:
    shape = objectfactory.create( data )
    print( 'class type: {}, shape area: {}'.format( type( shape ), shape.get_area() )␣
↪)
```

Output:

```
class type: <class '__main__.Square'>, shape area: 4.0
class type: <class '__main__.Triangle'>, shape area: 2.1875
class type: <class '__main__.Square'>, shape area: 2.25
```

See more examples here

1.2 Install

Use pip for installation

```
pip install objectfactory
```

CHAPTER

TWO

OBJECTFACTORY

objectfactory is a python package to easily implement the factory design pattern for object creation, serialization, and polymorphism

2.1 objectfactory.base

base module

implements abstract base classes for objectfactory

class objectfactory.base.**FieldABC**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: abc.ABC

abstract base class for serializable field

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

abstract marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.base.**SerializableABC**

Bases: abc.ABC

abstract base class for serializable object

abstract deserialize(*body: dict*)

deserialize model from dictionary

Parameters **body** – serialized data to load into object

abstract serialize(*include_type: bool = True*, *use_full_type: bool = True*) → dict

serialize model to dictionary

Parameters

- **include_type** – if true, type information will be included in body
- **use_full_type** – if true, the fully qualified path with be specified in body

Returns serialized object as dict

2.2 objectfactory.factory

factory module

implements serializable object factory

class objectfactory.factory.**Factory**(*name*)
Bases: object

factory class for registering and creating serializable objects

create(*body: dict*, *object_type: Type[T] = <class 'objectfactory.serializable.Serializable'>*) → T
create object from dictionary

Parameters

- **body** – serialized object data
- **object_type** – (optional) specified object type

Raises **TypeError** – if the object is not an instance of the specified type

Returns deserialized object of specified type

register(*serializable:* objectfactory.serializable.Serializable)
decorator to register class with factory

Parameters **serializable** – serializable object class

Returns registered class

objectfactory.factory.**create**(*body: dict*, *object_type: Type[T] = <class 'objectfactory.serializable.Serializable'>*) → T
create object from dictionary with the global factory

Parameters

- **body** – serialized object data
- **object_type** – (optional) specified object type

Raises **TypeError** – if the object is not an instance of the specified type

Returns deserialized object of specified type

objectfactory.factory.**register**(*serializable:* objectfactory.serializable.Serializable)
decorator to register class with the global factory

Parameters **serializable** – serializable object class

Returns registered class

2.3 objectfactory.field

field module

implements serializable fields

class objectfactory.field.**Boolean**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

serializable field for boolean data

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**Field**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.base.FieldABC*

base class for serializable field

this is a class level descriptor for abstracting access to fields of serializable objects

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**Float**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

serializable field for float data

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**Integer**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

serializable field for integer data

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**List**(*default=None*, *key=None*, *field_type=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

field type for list of serializable objects

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **field_type** – specified type for list of nested objects
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**Nested**(*default=None*, *key=None*, *field_type=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

field type for nested serializable object

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **field_type** – specified type for nested object
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

class objectfactory.field.**String**(*default=None*, *key=None*, *required=False*, *allow_none=True*)

Bases: *objectfactory.field.Field*

serializable field for string data

Parameters

- **default** – default value for field if unset
- **key** – dictionary key to use for field serialization
- **required** – whether this field is required to deserialize an object
- **allow_none** – whether null should be considered a valid value

marshmallow()

create generic marshmallow field to do actual serialization

Returns associated marshmallow field

2.4 objectfactory.serializable

serializable module

implements base class and metaclass for serializable objects

class objectfactory.serializable.**Meta**(*name*, *bases*, *attributes*, *schema=None*)

Bases: abc.ABCMeta

metaclass for serializable classes

this is a metaclass to be used for collecting relevant field information when defining a new serializable class

define a new serializable object class, collect and register all field descriptors, construct marshmallow schema

Parameters

- **name** – class name
- **bases** – list of base classes to inherit from
- **attributes** – dictionary of class attributes
- **schema** – (optional) predefined marshmallow schema

Returns newly defined class

class objectfactory.serializable.**Serializable**

Bases: *objectfactory.base.SerializableABC*

base class for serializable objects

deserialize(*body: dict*)

deserialize model from dictionary

Parameters **body** – serialized data to load into object

classmethod from_dict(*body: dict*)

constructor to set data with dictionary

Parameters **body** – dictionary

Returns new instance of serializable object

classmethod from_kwargs(***kwargs*)
constructor to set field data by keyword args

Parameters **kwargs** – keyword arguments by field

Returns new instance of serializable object

serialize(*include_type: bool = True*, *use_full_type: bool = True*) → dict
serialize model to dictionary

Parameters

- **include_type** – if true, type information will be included in body
- **use_full_type** – if true, the fully qualified path with be specified in body

Returns serialized object as dict

- genindex

PYTHON MODULE INDEX

O

INDEX