countrydata Documentation

Release 0.0.1

Álvaro Mondéjar Rubio

Mar 22, 2020

INTRODUCTION

CountryData is a Python library and command line interface to retrieve updated data for all the countries in the world fastly.

CHAPTER

ONE

INSTALLING

1.1 From Pypi

```
pip3 install countrydata
```

1.2 From Source

```
git clone https://github.com/mondeja/countrydata.git
cd countrydata
python3 setup.py install
```

CHAPTER

TWO

BASIC USAGE

2.1 Library

The main exposed interface for the library is `Country`, and can be initialized for a country indicating explicitly what data we are using or implicitly.

Explicitly, we can use whatever data field included. For example, retrieve Spain using ISO-3361-1 Alpha3 code, using `a3` as optional argument:

```
>>> from countrydata import Country
>>> country = Country(a3="ESP")
>>> country.data
{'codes': {'a2': 'ES', 'a3': 'ESP', 'num': 724, 'gec': 'SP'}}
```

Implicitly, countrydata tries to infer what field you are using. Let's try to retrieve Spain using ISO-3361-1 Numeric3 code as positional argument:

```
>>> country = Country(724)
>>> country.data
{'codes': {'a2': 'ES', 'a3': 'ESP', 'num': 724, 'gec': 'SP'}}
```

Warning: Keep in mind that positional arguments countries initialization may currently initialize unexpected countries.

2.2 Command line client

Run `countrydata --help` or `countrydata <command> --help` to see documentation for all commands.

```
$ countrydata get-value numeric ESP
724

$ countrydata get-value "ISO-3361-1 Alpha2" 724
ES
```

CHAPTER

THREE

OVERVIEW

CountryData maintains an updated set of countries data periodically tested. It allows you to write applications that needs updated information about countries and other geographical areas.

Currently, the data is built based on ISO-3361-1 Aplha2 country codes, including information from top level territories, without including country subdivisions.

All the data is scraped from online sources which acts as providers, but is validated by countrydata tests before add it to the code, so the library always include data previously validated for the publicly available tests.

3.1 Supported fields

Name	Identifier	Provider	Official source	Valid field names
ISO-3361-1 Alpha2	a2	Wikipedia	International Organization for Standardization (ISO)	`a2` `A2` `ISO-3361-1 Alpha2` `ISO_3361_1_Alpha2` `alpha2` `ALPHA2` `ISO-3361-1 Alpha2` `ISO_3361_1_Alpha2`
ISO-3361-1 Alpha3	a3	Wikipedia	International Organization for Standardization (ISO)	`a3` `A3` `ISO-3361-1 Alpha3` `ISO_3361_1_Alpha3` `alpha3` `ALPHA3` `ISO-3361-1 Alpha3` `ISO_3361_1_Alpha3`
ISO-3361-1 Numeric3	num	Wikipedia	International Organization for Standardization (ISO)	`num` `NUM` `ISO-3361-1 Numeric3` `ISO_3361_1_Numeric3` `numeric` `NUMERIC` `ISO-3361-1 Numeric3` `ISO_3361_1_Numeric3`

3.2 Available data

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
AD	AND	20
AE	ARE	784
AF	AFG	4
AG	ATG	28
AI	AIA	660
AL	ALB	8
AM	ARM	51
AO	AGO	24

Continued on next page

Table 1 – continued from previous page

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
AQ	ATA	10
AR	ARG	32
AS	ASM	16
AT	AUT	40
AU	AUS	36
AW	ABW	533
AX	ALA	248
AZ	AZE	31
BA	BIH	70
BB	BRB	52
BD	BGD	50
BE	BEL	56
BF	BFA	854
BG	BGR	100
BH	BHR	48
BI	BDI	108
BJ	BEN	204
BL	BLM	652
BM	BMU	60
BN	BRN	96
BO	BOL	68
BQ	BES	535
BR	BRA	76
BS	BHS	44
BT	BTN	64
BV	BVT	74
BW	BWA	72
BY	BLR	112
BZ	BLZ	84
CA	CAN	124
CC	CCK	166
CD	COD	180
CF	CAF	140
CG	COG	178
CH	CHE	756
CI	CIV	384
CK	COK	184
CL	CHL	152
CM	CMR	120
CN	CHN	156
CO	COL	170
CR	CRI	188
CU	CUB	192
CV	CPV	132
CW	CUW	531
CX	CXR	162
CY	CYP	196
CZ	CZE	203
DE	DEU	276

Continued on next page

Table 1 – continued from previous page

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
DJ	DJI	262
DK	DNK	208
DM	DMA	212
DO	DOM	214
DZ	DZA	12
EC	ECU	218
EE	EST	233
EG	EGY	818
EH	ESH	732
ER	ERI	232
ES	ESP	724
ET	ETH	231
FI	FIN	246
FJ	FJI	242
FK	FLK	238
FM	FSM	583
FO	FRO	234
FR	FRA	250
GA	GAB	266
GB	GBR	826
GD	GRD	308
GE	GEO	268
GF	GUF	254
GG	GGY	831
GH	GHA	288
GI	GIB	292
GL	GRL	304
GM	GMB	270
GN	GIN	324
GP	GLP	312
GQ	GNQ	226
GR	GRC	300
GS	SGS	239
GT	GTM	320
GU	GUM	316
GW	GNB	624
GY	GUY	328
HK	HKG	344
HM	HMD	334
HN	HND	340
HR	HRV	191
HT	HTI	332
HU	HUN	348
ID	IDN	360
IE	IRL	372
IL	ISR	376
IM	IMN	833
IN	IND	356
IO	IOT	86

Continued on next page

Table 1 – continued from previous page

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
IQ	IRQ	368
IR	IRN	364
IS	ISL	352
IT	ITA	380
JE	JEY	832
JM	JAM	388
JO	JOR	400
JP	JPN	392
KE	KEN	404
KG	KGZ	417
KH	KHM	116
KI	KIR	296
KM	COM	174
KN	KNA	659
KP	PRK	408
KR	KOR	410
KW	KWT	414
KY	CYM	136
KZ	KAZ	398
LA	LAO	418
LB	LBN	422
LC	LCA	662
LI	LIE	438
LK	LKA	144
LR	LBR	430
LS	LSO	426
LT	LTU	440
LU	LUX	442
LV	LVA	428
LY	LBY	434
MA	MAR	504
MC	MCO	492
MD	MDA	498
ME	MNE	499
MF	MAF	663
MG	MDG	450
MH	MHL	584
MK	MKD	807
ML	MLI	466
MM	MMR	104
MN	MNG	496
MO	MAC	446
MP	MNP	580
MQ	MTQ	474
MR	MRT	478
MS	MSR	500
MT	MLT	470
MU	MUS	480
MV	MDV	462

Continued on next page

Table 1 – continued from previous page

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
MW	MWI	454
MX	MEX	484
MY	MYS	458
MZ	MOZ	508
NA	NAM	516
NC	NCL	540
NE	NER	562
NF	NFK	574
NG	NGA	566
NI	NIC	558
NL	NLD	528
NO	NOR	578
NP	NPL	524
NR	NRU	520
NU	NIU	570
NZ	NZL	554
OM	OMN	512
PA	PAN	591
PE	PER	604
PF	PYF	258
PG	PNG	598
PH	PHL	608
PK	PAK	586
PL	POL	616
PM	SPM	666
PN	PCN	612
PR	PRI	630
PS	PSE	275
PT	PRT	620
PW	PLW	585
PY	PRY	600
QA	QAT	634
RE	REU	638
RO	ROU	642
RS	SRB	688
RU	RUS	643
RW	RWA	646
SA	SAU	682
SB	SLB	90
SC	SYC	690
SD	SDN	729
SE	SWE	752
SG	SGP	702
SH	SHN	654
SI	SVN	705
SJ	SJM	744
SK	SVK	703
SL	SLE	694
SM	SMR	674

Continued on next page

Table 1 – continued from previous page

ISO-3361-1 Alpha2	ISO-3361-1 Alpha3	ISO-3361-1 Numeric3
SN	SEN	686
SO	SOM	706
SR	SUR	740
SS	SSD	728
ST	STP	678
SV	SLV	222
SX	SXM	534
SY	SYR	760
SZ	SWZ	748
TC	TCA	796
TD	TCD	148
TF	ATF	260
TG	TGO	768
TH	THA	764
TJ	TJK	762
TK	TKL	772
TL	TLS	626
TM	TKM	795
TN	TUN	788
TO	TON	776
TR	TUR	792
TT	TTO	780
TV	TUV	798
TW	TWN	158
TZ	TZA	834
UA	UKR	804
UG	UGA	800
UM	UMI	581
US	USA	840
UY	URY	858
UZ	UZB	860
VA	VAT	336
VC	VCT	670
VE	VEN	862
VG	VGB	92
VI	VIR	850
VN	VNM	704
VU	VUT	548
WF	WLF	876
WS	WSM	882
YE	YEM	887
YT	MYT	175
ZA	ZAF	710
ZM	ZMB	894
ZW	ZWE	716

CHAPTER

FOUR

SETTING THE ENVIRONMENT

To set up a complete Python3 development environment (Linux based systems only):

```
git clone https://github.com/mondeja/countrydata.git
cd countrydata
python3 -m pip install -U virtualenv
python3 -m virtualenv venv
. venv/bin/activate
python3 -m pip install -r dev-requirements.txt
```

4.1 Commands

You need to install NodeJS with NPM or Yarn to run frequent commands easily:

- **build** Build dinamic documentation files.
- **dev:test** Test with pytest in your current environment.
- **dev:install** Performs a development install.
- **dev:install:req** Install countrydata development requirements.
- **doc** Build countrydata sphinx documentation.
- **doc:watch** Build countrydata sphinx documentation watching for changes in files.
- **flake8** Test countrydata code styling using flake8.
- **release** Release countrydata package to Pypa.
- **release:test** Release countrydata package to Pypa using testing endpoint.
- **test** Test countrydata against all available environments using tox.

CHAPTER

FIVE

TESTING

All the testing of the library is performed by tox for multiple Python environments and pytest.

Continuous testing is provided by Travis (for unit tests and style checks on Linux).

5.1 Commands

- **dev:test** Test with pytest in your current environment.
- **release:test** Release countrydata package to Pypa using testing endpoint.
- **test** Test countrydata against all available environments using tox.

CHAPTER

SIX

TODO

	Example checked
	Example unchecked

CHAPTER

SEVEN

CHANGELOG

Commit 729752f5

- **Added country names dissambiguation using dbpedia.** by *Álvaro Mondéjar* at *2020-03-22 21:00:12*
- **Minor bug fixed in documentation.** by *Álvaro Mondéjar* at *2020-03-21 00:41:54*
- **Documentation improvements** by *Álvaro Mondéjar* at *2020-03-20 23:07:55*
- **Dinamic Country interface properties added to documentation** by *Álvaro Mondéjar* at *2020-03-16 23:49:07*
- **Minor fix on script [ci skip]** by *Álvaro Mondéjar* at *2020-03-16 23:08:41*
- **Data field base specification documented.** by *Álvaro Mondéjar* at *2020-03-16 23:05:28*
- **Documentation improvements** by *Álvaro Mondéjar* at *2020-03-16 19:59:01*
- **Available data table added to documentation** by *Álvaro Mondéjar* at *2020-03-16 18:39:46*
- **Supported fields table added to documentation** by *Álvaro Mondéjar* at *2020-03-15 21:30:54*
- **Bump version to release script, badges and classifiers added.** by *Álvaro Mondéjar* at *2020-03-15 20:27:10*

Fig. 1: Caption: An example image

CHAPTER

EIGHT

COUNTRYDATA

8.1 countrydata.interface

class countrydata.**Country**(**args*, ***kwargs*)

Represents a country and exposes his data. Their initialization is very flexible, allowing to construct it using any field of the data that refers to a country. This possibilities that any data field can be used as reference and this class acts as a country data factory.

Can be initialized explicitly by optional arguments (safer) or implicitly, delegating to the library the inference process of discover what field of the data you are using.

Implicitly, the class can be initalized defining a country value of any type of data as first positional argument. For example, let's try to initialize the United States using ISO-3361-1 Alpha2 codes implicitly:

```
>>> country = Country("US")
>>> country.a3
'USA'
>>> country.num
840
```

Explicitly, the class can be initialized using one of the supported field names. This way is faster and safer. United States can be initialized by his ISO-3361-1 Numeric3 number:

```
>>> country = Country(num=840)
>>> country.a3
'USA'
```

property a2
Returns ISO-3361-1 Alpha2.

property a3
Returns ISO-3361-1 Alpha3.

property num
Returns ISO-3361-1 Numeric3.

property codes
Returns ISO-3361-1 Alpha2, ISO-3361-1 Alpha3 and ISO-3361-1 Numeric3.

CHAPTER

NINE

COUNTRYDATA

9.1 countrydata.spec

class countrydata.spec.**CountryDataEntity**

Base class for all specification fields. All abstract properties are mandatory and must be overwritten for each field specficiation.

abstract classmethod property readable

Readable representation of a field.

abstract classmethod property ids

Valid identifiers for a field. The first identifier defined is the main identifier, and will be exposed as a property of `countrydata.spec.Country` interface.

abstract classmethod property field_path

Path to the field inside the data. This needs to be defined as `path/to/category:location.of.data` where `path/to/category` is the path in the data files tree and `location.of.data` indicates the JSON attribute where is located the value for the country formatted as Javascript `.` characters getters syntax.

For example, the location of ISO-3361-1 Alpha2 code values are in `countrydata/data/files/general/<country-file>.json:codes.a2`, so the value of this property at specification is `general:codes.a2`.

abstract classmethod property type

Data type of the field. Needs to be a valid builtin python data type. Will be used to cast values and for testing.

abstract classmethod property null

Boolean value that specifies if the field can be nullable or not.

abstract classmethod property assertions

List of tuples of two functions that takes a value as first parameter. Assertions are used to infere what fields are passed on interface initializations and testing the library.

The assertions described here must not contain assertions for type data or nullable values testing because this assertions are included dinamically using *`null`* and *`type`* properties.

Example of ISO-3361-1 Alpha2 specification assertion:

```
>>> [
>>>     (
>>>         lambda value: value.isupper(),
>>>         lambda value: "%s code for %s is not uppercased" % (ISO_3361_1_
↪Alpha2.readable, value),
```

(continues on next page)

(continued from previous page)

```
>>>     ),
>>> ]
```

abstract classmethod property data
Information about data providers and data sources. Represented by a dictionary of, at least, one key:

- **provider*** Dictionary with `readable` and `url` keys that indicates the data provider of the field.
- **source** Dictionary with `readable` and `url` keys that indicates the data official source of the field. This value is optional. If not is defined, this means that the provider is the same as the official source of the data.

classmethod property data_provider
Returns the data provider defined at attribute `provider` of *`data`* class property.

classmethod property data_source
Returns the data source defined at attribute `source` of *`data`* class property. If not defined, returns the data provider defined at attribute `provider` of *`data`* class property.

classmethod test(*value*)
Test using default Python `assert` statments if a value can be considered a valid value that follows the specification of the field.

Parameters **value** (*any*) – Value to test against the field restrictions.

classmethod pytest(*value*)
Test using Pytest style statments (`assertion, "Failure message"`) if a value can be considered a valid value that follows the specification of the field.

Parameters **value** (*any*) – Value to test against the field specification restrictions.

classmethod property category_dirpath
Returns the absolute path to the files directory of the category group for the field.

classmethod property categories_chain
Returns all nested categories that belongs to the field specification.

classmethod property groups_chain
Returns all nested groups that wraps the field specification into nested dictionaries.

classmethod filepath(*value*)
Returns the ISO-3361-1 Alpha2 data filename correspondent to the value passed. Needs to be a valid value for the field of the specification.

Parameters **value** (*any*) – Field specification value to use for search the correspondent ISO-3361-1 Alpha2 data filename.

classmethod random_value()
Returns a random country between the values of specification field.

classmethod property valid_field_names
Returns a list of strings that can be used to initializate the country explicitly by field-value at `Country` interface.

INDEX