Football Data Connector Documentation

Tony Joseph

Oct 29, 2018

Contents

Football Data Connector is a Python package to connect to football-data.org API

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

Football data connector requires Python 3.4 or later. This package is not tested with any version of Python 2.7. The following third party packages are required, which will be auto-installed if you are using pip.

- python-dateutil
- requests

1.1.2 Installing via pip

The easiest way to install football data connector is using the Python package manager, pip. Use the following command if you have pip installed

```
pip install football-data-connector
```

1.1.3 Installing from github

If you need the latest version, you can install it directly from github repository.

```
pip install git+https://github.com/tony-joseph/football-data-connector.git
```

1.2 User Guide

1.2.1 Creating a Connection to football-data.org API

Before retrieving data, you need to create a connection object.

```
from footballdata.connector import Connector

connection = Connector()
```

The **Connector** constructor also accepts two optional arguments, *api_key* and *api_version*.

```
from footballdata import Connector

connection = Connector(api_key='api key from football-data.org', api_version='v1')
```

Using an API key is recommended. Otherwise you may hit the rate limiting of API. Currently the only supported API version is v1. You can retrieve the list of supported API versions using the class method *Connector.supported_api_versions()supported_api_versions*

Connector object attributes

- **api_key** Gives the API key used to create Connector object
- **api_version** Gives the API version used to create Connector object
- **base_url** Gives the base API URL
- **competitions_endpoint** Gives the API endpoint to fetch competitions
- **fixtures_endpoint** Gives the API endpoint to fetch fixtures

Connector object methods

get_competitions(season='', force_update=False)

Returns a DataSet object contains Competition objects. Accepts two optional arguments, *season* and *force_update*. *season* should be a 4 digit integer representing an year (example 2017). If season is given, only the competitions in the given season will be fetched. Once the values are fetched from API, the results will be cached and, the subsequent calls to get_competitions method will return the cached results. Use *force_update=True* if you want to override the cache and get fresh results from API.

get_fixtures(force_update=False)

Returns a **DataSet** object contains **Fixture** objects. Results will be cached after first API call to avoid unnecessary API hits. Use *force_update=True* if you want to override the cache.

1.2.2 DataSet Objects

Methods like *get_competitions*, *get_fixtures* etc will return a **DataSet** object. **DataSet** is an iterable object. You can use it like any other iterable such as list, tuple etc. Operations like using with a for loop, checking length using len, subscripting, slicing, reversing etc are supported.

1.2.3 Competition Objects

A **Competition** object represents a competition in football-data.org API.

Competition Object Attributes

- id
- caption
- current_match_day
- last_updated
- league
- number_of_games
- number_of_teams
- year

Competition Object Methods

get_teams(force_update=False)

Returns a **DataSet** of **Team** objects. Use *force_update=True* to override cache.

get_fixtures(force_update=False)

Returns a **DataSet** of **Fixture** objects. Use force_update=True to override cache.

get_league_table(force_update=False)

Returns a **DataSet** of **Standing** objects. If league table is not available for a competition, an empty **DataSet** will be returned. Use *force_update=True* to override cache.

1.2.4 Fixture Objects

A **Fixture** object represent a fixture in football-data.org API.

Fixture Object Attributes

- date
- away_team_name
- home_team_name
- match_day
- odds
- result

- status

1.2.5 Team Objects

A **Team** object represents a team in a competition.

Team Object Attributes

- code
- crest_url
- name
- short_name
- squad_market_value

Team Object Methods

get_fixtures(force_update=False)

Returns a **DataSet** of **Fixture** objects representing fixtures of the team for the current season.

get_players(force_update=False)

Returns a **DataSet** for Player objects representing players in the team.

1.2.6 Standing Objects

A **Standing** object represents the standing of a team in a competition.

Standing Object Attributes

- team_name
- crest_uri
- played_games
- wins
- draws
- losses
- home
- away
- points
- position
- goals

- goals_aganist
- goal_difference

1.2.7 Player Objects

A **Player** object represents a player in a team.

Player Object Attributes

- name
- nationality
- position
- contract_until
- date_of_birth
- jersey_number
- market_value

1.3 Caching and Lazy Evaluation

Values in a **DataSet** object are cached during the creation of DataSet. Subsequent calls to methods which returns a **DataSet** will be returning the cached values. These methods will accept an optional parameter, *force_update*, which if set to **True**, will force an API call again and fetch new values. You should force update only on situations where it is absolutely necessary. Otherwise you may hit the API rate limit.

A **DataSet** will not perform any API calls during its creation. There will not be any values in a **DataSet** after its creation. API call is executed only when an action which uses the data is executed such as using in a for loop, checking the length of **DataSet** etc.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search