# Wetterturnier

Generated by Doxygen 1.6.1

Mon Aug 3 18:53:22 2015

# Contents



# 1 Class Index

## 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:



# 2 Class Documentation

## 2.1 pywetterturnier::database::database Class Reference

### Public Member Functions

- def __init__

- def __connect__
- def **cursor**
- def **execute**
- def **executemany**
- def **commit**
- def **create_group**
- def **create_user**
- def **get_cities**
- def **get_stations_for_city**
- def **current_tournament**
- def **all_tournament_dates**
- def **get_cityall_bet_data**
- def **get_bet_data**
- def **upsert_bet_data**
- def **upsert_points_data**
- def **upsert_deadman_points**
- def **get_obs_data**
- def **get_parameter_names**
- def **get_parameter_id**
- def **get_user_id_and_create_if_necessary**
- def **get_user_id**
- def **get_username_by_id**
- def **get_active_groups**
- def **get_group_id**
- def **create_groupuser**
- def **get_deadman_points**
- def **close**

### Public Attributes

- **config**
- **db**
- **prefix**

## 2.1.1 Member Function Documentation

### 2.1.1.1 def pywetterturnier::database::database::__connect__ ( *self*)

```
Open database connection
```

### 2.1.1.2 def pywetterturnier::database::database::__init__ ( *self*, *config*)

```
The database class is handling the connection to the
mysql database and different calls and stats.
```

The documentation for this class was generated from the following file:

- pywetterturnier/database.py

## 2.2 pywetterturnier::getobs::getobs Class Reference

### Public Member Functions

- def **__init__**
- def **get_maximum_Sd**
- def **get_columns**
- def **load_obs**
- def **check_record**
- def **load_sunshine**
- def **__add_obs_value__**
- def **prepare**
- def prepare_fun_TTm
- def prepare_fun_TTn
- def prepare_fun_TTd
- def prepare_fun_PPP
- def prepare_fun_dd
- def prepare_fun_ff
- def prepare_fun_fx
- def prepare_fun_N
- def prepare_fun_Wv
- def prepare_fun_Wn
- def prepare_fun_RR
- def **prepare_fun_Sd**

### Public Attributes

- **config**
- **city**
- **date**
- **db**
- **table**
- **data**
- **columns**
- **stations**
- **maxSd**

### 2.2.1 Member Function Documentation

#### 2.2.1.1 def pywetterturnier::getobs::getobs::prepare_fun_dd ( *self*, *parameter*, *station*)

```
Helper function for the wind direction at 12 UTC from database
column dd. Values will be returned in 1/10 degrees but rounded
to full 10 degrees. E.g., observed '138' degrees will be converted
into '1400' (1/10 degrees, rounded to full 10 degrees).
Special case: also depends on database column ff. The following
cases will be used:
   1) if dd not observed/received: return None
   2) if dd==0 and ff==0:          return 0.0
   3) if dd==0 and ff>0:           return None (variable wind direction)
   4) else:                        return value
```

```
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.2 def pywetterturnier::getobs::getobs::prepare_fun_ff ( *self*, *parameter*, *station*)

```
Helper function for the wind speed at 12 UTC from database
column ff. Values will be in 1/10 knots but rounded to full knots.
E.g., if 3.2m/s observed -> 6.22kt -> Return value will be 60.
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.3 def pywetterturnier::getobs::getobs::prepare_fun_fx ( *self*, *parameter*, *station*)

```
Helper function for the maximum gust speed (fx > 25kt) from
database column fx1. Return value will be in 1/10 knots but
rounded to full knots.
E.g., if 21.2m/s observed -> 41.21kt -> Return value will be 410.
Special cases:
   1) no observation available but +30h
observation (row) is in database:
Assume that there were no gusts at all    return 0
   2) no observations available and +30h
observation (row) not yet in database     return None
   3) observation available, below 25 knots     return 0
   4) observation available, >= 25 knots        return value
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.4 def pywetterturnier::getobs::getobs::prepare_fun_N ( *self*, *parameter*, *station*)

```
Helper function for cloud cover at 12 UTC. Return value
will be in 1/10 octas, rounded to full octas [0,10,20,30,...,80].
Observations based on database column cc.
   1) if observation is available         return value
   2) if observation not recorded but
12 UTC database entry exists we
assume that there were no clouds    return 0
   3) else                                return None
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
```

```
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.5 def pywetterturnier::getobs::getobs::prepare_fun_PPP ( *self*, *parameter*, *station*)

```
Helper function for mean sea level pressure at 12 UTC from
database column pmsl. Return value will be in 1/10 hPa.
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.6 def pywetterturnier::getobs::getobs::prepare_fun_RR ( *self*, *parameter*, *station*)

```
Helper function for 24h sum of precipitation based on
database column 'rrr12' at +18 and +30h (as the reported
observations are 12h sums this means from 06 UTC today
to 06 UTC tomorrow). Returns precipitation in 1/10 mm
OR -30 if there was no precipitation at all.
   First: if database entry for 18 UTC is here but
there is no recorded amount of precipitation we
have to assume that there was no precipitation.
The same for +30h (06 UTC next day).
   Second:
If observed precipitation amount is negative (some
stations send -0.1mm/12h for no precipitation) we

   TODO RETO this is a problem!
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.7 def pywetterturnier::getobs::getobs::prepare_fun_TTd ( *self*, *parameter*, *station*)

```
Helper function for dew point temperature. Returns 12 UTC observed
dew point temperature from database column td in 1/10 degrees Celsius.
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.8 def pywetterturnier::getobs::getobs::prepare_fun_TTm ( *self*, *parameter*, *station*)

```
Helper function for TTM, maximum temperature. Returns 18 UTC maximum
temperature, either from column tmax12 or - if tmax12 not available
but tmax24 exists - maximum temperature from tmax24.
Temperature will be in 1/10 degrees Celsius.
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.9 def pywetterturnier::getobs::getobs::prepare_fun_TTn ( *self*, *parameter*, *station*)

```
Helper function for TTN, minimum temperature. Returns 06 UTC minimum
temperature. Simply the tmin12 column at 06 UTC in 1/10 degrees Celsius.
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.10 def pywetterturnier::getobs::getobs::prepare_fun_Wn ( *self*, *parameter*, *station*)

```
Helper function for significant weather observatioins between
12 UTC and 18 UTC (afternoon) based on database table w1.
Value will be in 1/10 levels [0,10,20,...,90].
   1) if observation not recorded but
18 UTC database entry exists we
assume that there were no clouds       return 0
   2) if observation not recorded and
18 UTC database entry not available    return None
   3) observation here BUT
the observed value is < 10 (note:
BUFR messages, 10+ are for automated
significant weather instruments)       return None
   3) else                                   return value
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

#### 2.2.1.11 def pywetterturnier::getobs::getobs::prepare_fun_Wv ( *self*, *parameter*, *station*)

```
Helper function for significant weather observatioins between
06 UTC and 12 UTC (forenoon) based on database table w1.
```

```
Value will be in 1/10 levels [0,10,20,...,90].
   1) if observation not recorded but
12 UTC database entry exists we
assume that there were no clouds       return 0
   2) if observation not recorded and
12 UTC database entry not available    return None
   3) observation here BUT
the observed value is < 10 (note:
BUFR messages, 10+ are for automated
significant weather instruments)       return None
   3) else                                   return value
Args:
   parameter (string): string with parameter short name (e.g., TTm, N, RR)
   station (class): stationclass object.
Returns:
   numeric: observed value if loading data was successful
   None: if observation not available or nor recorded
```

The documentation for this class was generated from the following file:

- pywetterturnier/getobs.py

## 2.3 pywetterturnier::groupbets::groupbets Class Reference

### Public Member Functions

- def __init__
- def **get_groups**
- def **check_and_create_groupuser**
- def **current_tournament**
- def **users_in_group**
- def **compute_bets**
- def **upsert_bets**

### Public Attributes

- **config**
- **db**

### 2.3.1 Member Function Documentation

#### 2.3.1.1 def pywetterturnier::groupbets::groupbets::__init__ ( *self*, *config*)

```
The database class is handling the connection to the
mysql database and different calls and stats.
```

The documentation for this class was generated from the following file:

- pywetterturnier/groupbets.py

## 2.4 pywetterturnier::importbets::importbets Class Reference

**Public Member Functions**

- def **__init__**
- def **loadfile**
- def **loadfilecontent**
- def **takedata**
- def **identify_city**
- def **extract_bettimes**
- def **extract_parameter_points**
- def **extract_sum_points**
- def **extract_obs**
- def **__insert_obs_to_db__**
- def **__get_wmo_number__**
- def **extract_bets**
- def **__insert_bet_to_db__**
- def **__insert_betstat_to_db__**
- def **wp_create_user**
- def **wp_user_exists**
- def **wp_get_param_id**
- def **close**

**Public Attributes**

- **config**
- **db**
- **raw_lines**
- **tournamentdate**
- **obs1**
- **obs2**
- **points**
- **bettimes**

**Static Public Attributes**

- **data1** = None
- **data2** = None
- **cityID** = None
- string **prefix** = 'wp_'
- string **db_users** = 'users'
- string **db_param** = 'wetterturnier_param'
- string **db_bets** = 'wetterturnier_bets'
- string **db_betstat** = 'wetterturnier_betstat'
- string **db_cities** = 'wetterturnier_cities'
- string **db_obs** = 'wetterturnier_obs'
- **unames** = None

The documentation for this class was generated from the following file:

- pywetterturnier/importbets.py

## 2.5 pywetterturnier::judgingclass20021206::judging Class Reference

### Public Member Functions

- def **__init__**
- def **__prepare_for_database__**
- def **points_to_database**
- def **get_points**
- def **__residuals__**
- def **__points_TTm__**
- def **__points_TTn__**
- def **__points_TTd__**
- def **__points_TTmTTnTTd__**
- def **__points_N__**
- def **__points_Sd__**
- def **__points_dd__**
- def **__points_ff__**
- def **__points_fx__**
- def **__points_Wv__**
- def **__points_Wn__**
- def **__points_WvWn__**
- def **__points_PPP__**
- def **__points_RR__**

### Public Attributes

- **quiet**
- **tdate_min**

### Static Public Attributes

- int **tdate_min** = 12027
- **tdate_max** = None

The documentation for this class was generated from the following file:

- pywetterturnier/judgingclass20021206.py

## 2.6 pywetterturnier::migrategroups::migrategroups Class Reference

### Public Member Functions

- def __init__
- def __download_file__
- def __read_rawfile__
- def **create_groups_and_users**

**Public Attributes**

- **config**
- **db**
- **rawfile**
- **data**

### 2.6.1 Member Function Documentation

#### 2.6.1.1 def pywetterturnier::migrategroups::migrategroups::__download_file__ ( *self*)

```
Downloading the html file and store locally
```

#### 2.6.1.2 def pywetterturnier::migrategroups::migrategroups::__init__ ( *self*, *config*)

```
Downloading the groups list file from wetterturnier.de,
Parse the content and create groups and its users if they
are not allready existing.
```

#### 2.6.1.3 def pywetterturnier::migrategroups::migrategroups::__read_rawfile__ ( *self*)

```
Parsing raw file and import groups and users.
```

The documentation for this class was generated from the following file:

- pywetterturnier/migrategroups.py

## 2.7 pywetterturnier::stationclass::stationclass Class Reference

**Public Member Functions**

- def **__init__**
- def **show**

**Public Attributes**

- **ID**
- **wmo**
- **cityID**
- **name**
- **nullconfig**
- **changed**

The documentation for this class was generated from the following file:

- pywetterturnier/stationclass.py

# Index