
bundesliga-tippspiel

Release 1.7.1

Hermann Krumrey

Sep 19, 2020

CONTENTS

1	bundesliga_tippspiel package	3
1.1	Subpackages	3
1.2	Submodules	49
1.3	bundesliga_tippspiel.Config module	49
1.4	bundesliga_tippspiel.enums module	50
1.5	bundesliga_tippspiel.exceptions module	50
1.6	bundesliga_tippspiel.jinja_extras module	50
1.7	bundesliga_tippspiel.main module	51
1.8	bundesliga_tippspiel.template_extras module	51
1.9	Module contents	51
2	bundesliga_tippspiel	53
3	Indices and tables	55
	Python Module Index	57
	Index	61

Contents:

BUNDESLIGA_TIPPSPIEL PACKAGE

1.1 Subpackages

1.1.1 bundesliga_tippspiel.actions package

Submodules

bundesliga_tippspiel.actions.Action module

class `bundesliga_tippspiel.actions.Action.Action`

Bases: `object`

A framework class for actions that can be used by normal site requests as well as API calls to achieve stuff™.

static check_id_or_filters (*_id: Optional[int], filters: List[Optional[Any]]*)

Checks that no filters are applied if a specific ID was provided :param _id: The specific ID :param filters: A list of filters :return: None

execute () → `Dict[str, Any]`

Executes the action after validating user-provided data :return: A JSON-compatible dictionary containing the response :raises `ActionException`: if anything went wrong

execute_with_redirects (*success_url: str, success_msg: str, failure_url: str*) → `werkzeug.wrappers.response.Response`

Executes the action and subsequently redirects accordingly :param success_url: The URL to which to redirect upon success :param success_msg: The message to flash on success :param failure_url: The URI to which to redirect upon failure :return: The redirect

classmethod from_dict (*data: Dict[str, Any]*)

Generates an action from a dictionary :param data: The dictionary containing the relevant data :return: The generated Action object

classmethod from_site_request ()

Generates an Action object from a site request :return: The generated Action object

static get_current_matchday () → `int`

Returns The current matchday

static handle_id_fetch (*_id: int, query_cls: Type[sqlalchemy.ext.declarative.api.Model]*) → `sqlalchemy.ext.declarative.api.Model`

Handles fetching a single object by using it's ID Raises an `ActionException` if an ID does not exist :return: The object identified by the ID :raises `ActionException`: Without fail

prepare_get_response (*result: List[puffotter.flask.db.ModelMixin.ModelMixin], keyword: str*) → Dict[str, Any]

Prepares a GetAction response by :param result: The result to wrap in a response dictionary :param keyword: The keyword to use, e.g: betmatch|player etc. :return: The wrapped response dictionary

static resolve_and_check_matchday (*matchday: Optional[int]*) → Optional[int]

Checks the bound of a matchday :param matchday: The matchday to check :return: None :raises ActionException: If the matchday is invalid

validate_data ()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

class bundesliga_tippspiel.actions.Action.**GetAction** (*_id: Optional[int]*)

Bases: *bundesliga_tippspiel.actions.Action.Action*

Special Action class for 'Get' Actions

__init__ (*_id: Optional[int]*)

Parameters *_id* – The ID to get

bundesliga_tippspiel.actions.ChangeSettingsAction module

class bundesliga_tippspiel.actions.ChangeSettingsAction.**ChangeSettingsAction** (*display_bots: bool*)

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows users to change miscellaneous settings

__init__ (*display_bots: bool*)

Initializes the SetReminderSettingsAction object :param display_bots: Whether to display bots

validate_data ()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetBetAction module

class bundesliga_tippspiel.actions.GetBetAction.**GetBetAction** (*_id: Optional[int] = None, user_id: Optional[int] = None, match_id: Optional[int] = None, matchday: Optional[int] = None, include_other_users_bets: bool = False*)

Bases: *bundesliga_tippspiel.actions.Action.GetAction*

Action that allows retrieving bets from the database

__init__ (*_id: Optional[int] = None, user_id: Optional[int] = None, match_id: Optional[int] = None, matchday: Optional[int] = None, include_other_users_bets: bool = False*)

Initializes the GetBetAction object :param _id: If provided, returns the bet with the specified ID :param user_id: If provided, will only provide bets for the

specified user

Parameters

- **match_id** – If provided, will only provide bets for the specified match
- **matchday** – If provided, will only provide bets for the specified matchday
- **include_other_users_bets** – If True, will include bets from other users even if the match hasn't started yet.

Raises ActionException if any problems occur

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetGoalAction module

```
class bundesliga_tippspiel.actions.GetGoalAction.GetGoalAction(_id: Optional[int] = None, matchday: Optional[int] = None, match_id: Optional[int] = None, player_id: Optional[int] = None, team_id: Optional[int] = None)
```

Bases: *bundesliga_tippspiel.actions.Action.GetAction*

Action that allows retrieving goals from the database

```
__init__(_id: Optional[int] = None, matchday: Optional[int] = None, match_id: Optional[int] = None, player_id: Optional[int] = None, team_id: Optional[int] = None)
```

Initializes the GetGoalAction object :param _id: If provided, returns the goal with the specified ID :param matchday: If provided, will only fetch goals

on the specified matchday

Parameters

- **match_id** – If provided, will only fetch goals that occurred during the specified match
- **player_id** – If provided, will only fetch goals from the specified player
- **team_id** – If provided, will only fetch goals from the specified team

Raises ActionException if any problems occur

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetMatchAction module

```
class bundesliga_tippspiel.actions.GetMatchAction.GetMatchAction (_id: Optional[int] = None, matchday: Optional[int] = None, team_id: Optional[int] = None)
```

Bases: *bundesliga_tippspiel.actions.Action.GetAction*

Action that enables getting Matches

```
__init__ (_id: Optional[int] = None, matchday: Optional[int] = None, team_id: Optional[int] = None)
```

Initializes the GetMatchAction object :param _id: If provided, returns the match with that ID :param matchday: If provided, will return all matches on that matchday :param team_id: If provided, will return all matches of a team :raises: ActionException if any problems occur

```
validate_data ()
```

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetPlayerAction module

```
class bundesliga_tippspiel.actions.GetPlayerAction.GetPlayerAction (_id: Optional[int] = None, team_id: Optional[int] = None)
```

Bases: *bundesliga_tippspiel.actions.Action.GetAction*

Action that allows retrieving players from the database

```
__init__ (_id: Optional[int] = None, team_id: Optional[int] = None)
```

Initializes the GetPlayerAction object :param _id: If provided, will only fetch the selected ID :param team_id: If provided, will only fetch players

in the selected team

Raises ActionException if any problems occur

```
validate_data ()
```

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetReminderSettingsAction module

class `bundesliga_tippspiel.actions.GetReminderSettingsAction`.**GetReminderSettingsAction**
 Bases: `bundesliga_tippspiel.actions.Action.Action`

Action that allows retrieving the user's reminder settings

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.GetTeamAction module

class `bundesliga_tippspiel.actions.GetTeamAction`.**GetTeamAction** (*_id*: *Optional[int]* = *None*)

Bases: `bundesliga_tippspiel.actions.Action.GetAction`

Action that allows retrieving teams from the database

__init__ (*_id*: *Optional[int]* = *None*)

Initializes the GetTeamAction object :param _id: If provided, will only fetch the selected ID :raises: ActionException if any problems occur

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.LeaderboardAction module

class `bundesliga_tippspiel.actions.LeaderboardAction`.**LeaderboardAction** (*matchday*: *Optional[int]* = *None*, *count*: *bool* = *False*, *include_bots*: *bool* = *False*, *bets*: *Optional[List[bundesliga_tippspiel.db.user_generated.Bet.Bet]]* = *None*)

Bases: `bundesliga_tippspiel.actions.Action.Action`

Action that allows fetching a sorted leaderboard

__init__ (*matchday*: *Optional[int]* = *None*, *count*: *bool* = *False*, *include_bots*: *bool* = *False*, *bets*: *Optional[List[bundesliga_tippspiel.db.user_generated.Bet.Bet]]* = *None*)

Initializes the LeaderboardAction object :param matchday: The matchday for which to generate the leaderboard.

If None, will use the most current matchday

Parameters

- **count** – If set to true, will count the amount of bets instead of evaluating their points
- **include_bots** – Whether or not to include bots. Bots are identified by a robot or brain emoji
- **bets** – Allows using a specific set of bets to calculate the leaderboard

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.LoadSettingsAction module

class bundesliga_tippspiel.actions.LoadSettingsAction.**LoadSettingsAction**

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that retrieves miscellaneous settings for the current user

__init__()

Initializes the LoadSettingsAction object :raises: ActionException if any problems occur

validate_data()

Validates data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.PlaceBetsAction module

class bundesliga_tippspiel.actions.PlaceBetsAction.**PlaceBetsAction** (*bets: Dict[Union[str, int], Tuple[Optional[Union[str, int]], Optional[Union[str, int]]], Optional[Union[str, int]]]*)

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows placing bets

__init__ (*bets: Dict[Union[str, int], Tuple[Optional[Union[str, int]], Optional[Union[str, int]]]*)

Initializes the PlaceBetsAction object :param bets: a dictionary mapping match IDs to tuples containing the

home and away scores

Raises ActionException if any problems occur

validate_data()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

bundesliga_tippspiel.actions.SetReminderSettingsAction module

class `bundesliga_tippspiel.actions.SetReminderSettingsAction`.**SetReminderSettingsAction** (*hours: int, reminder_states: Dict[bundesliga_tippspiel.enums.ReminderType, bool]*)

Bases: `bundesliga_tippspiel.actions.Action.Action`

Action that allows setting reminder settings

__init__ (*hours: int, reminder_states: Dict[bundesliga_tippspiel.enums.ReminderType, bool]*)

Initializes the SetReminderSettingsAction object :param hours: How many hours before the match the user wants a reminder :param reminder_states: specifies the states for each of the reminder types

validate_data ()

Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found

Module contents**1.1.2 bundesliga_tippspiel.background package****Submodules****bundesliga_tippspiel.background.match_data module**

`bundesliga_tippspiel.background.match_data`.**get_team_data** (*team_name: str*) → Tuple[str, str, str, Tuple[str, str]]

Generates team short_names, abbreviations and icon URLs for teams :param team_name: The team's full name as specified by OpenLigaDB :return: A tuple containing the name, short_name, abbreviation, svg URL, png URL

`bundesliga_tippspiel.background.match_data`.**parse_goal** (*goal_data: Dict[str, Any], match_id: int*) → Optional[bundesliga_tippspiel.db.match_data.Goal.Goal]

Parses a goal JSON object and generates a Goal object :param match_id: The match ID of the match in which the goal was scored :param goal_data: The goal data to parse :return: The generated Goal object

`bundesliga_tippspiel.background.match_data`.**parse_match** (*match_data: Dict[str, Any]*) → bundesliga_tippspiel.db.match_data.Match.Match

Parses a Match object from JSON match data :param match_data: The match data to parse :return: The generated Match object

`bundesliga_tippspiel.background.match_data`.**parse_player** (*goal_data: Dict[str, Any], team_id: int*) → bundesliga_tippspiel.db.match_data.Player.Player

Parses a Player object from a Goal JSON data object :param goal_data: The data of a goal the player scored :param team_id: The Team of the player :return: The generated Player object

`bundesliga_tippspiel.background.match_data`.**parse_team** (*team_data: Dict[str, Any]*) → bundesliga_tippspiel.db.match_data.Team.Team

Parses team-related JSON data and generates a Team object from that :param team_data: The team data to parse
:return: The generated Team object

bundesliga_tippspiel.background.match_data.**store_in_db** (objects: *List[sqlalchemy.ext.declarative.api.Model]*,
model_cls: *Type[sqlalchemy.ext.declarative.api.Model]*)

Stores a list of objects in the database. While doing so, will either update existing objects or create new one if they don't exist :param objects: The objects to add :param model_cls: The model class of these objects :return: None

bundesliga_tippspiel.background.match_data.**update_match_data** (league: *Optional[str]* = *None*, season: *Optional[str]* = *None*)

Updates the database with the match data for the specified league and season :param league: The league for which to update the data :param season: The season for which to update the data :return: None

bundesliga_tippspiel.background.match_data.**wikimedia_icon_urls** (path: *str*, png_size: *int* = 500) → *Tuple[str, str]*

Generates URL paths to wikimedia-hosted SVG and PNG files :param path: The URL path to the SVG file (without the wikimedia part) :param png_size: The size of the PNG file :return: The URL path to the SVG File, PNG file

bundesliga_tippspiel.background.reminders module

bundesliga_tippspiel.background.reminders.**send_due_reminders** ()
Sends all email reminders that are due :return: None

bundesliga_tippspiel.background.season_events module

bundesliga_tippspiel.background.season_events.**handle_midseason_reminder** () → *bool*

Handles sending out midseason reminders :return: Whether the reminder was sent or not

bundesliga_tippspiel.background.season_events.**handle_postseason_wrapup** () → *bool*

Handles the post-season wrapup :return: None

bundesliga_tippspiel.background.season_events.**handle_preseason_reminder** () → *bool*

Sends a reminder to existing users a week before the start of the season :return: Whether or not the reminder was sent

bundesliga_tippspiel.background.season_events.**handle_season_events** ()
Handles any events that happen once every season :return: None

bundesliga_tippspiel.background.season_events.**load_season_events** () → *List[bundesliga_tippspiel.db.SeasonE*

Loads all event states from the database :return: The event states

Module contents

`bundesliga_tippspiel.background.bg_tasks`: `Dict[str, Tuple[int, Callable]] = {'handle_season': ...}`
 A dictionary containing background tasks for the flask application

1.1.3 bundesliga_tippspiel.db package

Subpackages

`bundesliga_tippspiel.db.match_data` package

Submodules

`bundesliga_tippspiel.db.match_data.Goal` module

```
class bundesliga_tippspiel.db.match_data.Goal.Goal(*args, **kwargs)
    Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.
    api.Model

    Model that describes the “goals” SQL table

    __init__(*args, **kwargs)
        Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword
        arguments

    away_score: int
        The away team’s score after the goal was scored

    home_score: int
        The home team’s score after the goal was scored

    id

    match: bundesliga_tippspiel.db.match_data.Match.Match
        The match in which this goal was scored.

    match_id: int
        The ID of the match in which this goal was scored. Acts as a foreign key.

    minute: int
        The minute in which the goal was scored

    minute_et: int
        This keeps track in which minute of extra time a goal was scored.

    own_goal: bool
        Indicates whether or not this goal was an own goal

    penalty: bool
        Indicates whether or not this goal was a penalty

    player: bundesliga_tippspiel.db.match_data.Player.Player
        The player that scored this goal.

    player_id: int
        The ID of the player that scored this goal. Acts as a foreign key.
```

bundesliga_tippspiel.db.match_data.Match module

```
class bundesliga_tippspiel.db.match_data.Match.Match(*args, **kwargs)
    Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.
    api.Model

    Model that describes the 'matches' SQL table

    __init__(*args, **kwargs)
        Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword
        arguments

    away_current_score: int
        The current score of the away team.

    away_ft_score: int
        The final score of the away team

    away_ht_score: int
        The score of the away team at half time

    away_team: bundesliga_tippspiel.db.match_data.Team.Team
        The away team.

    away_team_id: int
        The ID of the away team. Acts as a foreign key

    bets: List[Bet]
        Bets placed on this match

    property current_score
        Returns The current score formatted as a string

    finished: bool
        Indicates whether or not the match has finished yet

    property ft_score
        Returns The full time score formatted as a string

    goals: List[Goal]
        The goals scored during this match

    home_current_score: int
        The current score of the home team.

    home_ft_score: int
        The final score of the home team

    home_ht_score: int
        The score of the home team at half time

    home_team: bundesliga_tippspiel.db.match_data.Team.Team
        The home team.

    home_team_id: int
        The ID of the home team. Acts as a foreign key

    property ht_score
        Returns The half time score formatted as a string

    id
```


kickoff: **str**

A string representing the kickoff time in UTC in the following format: YYYY-MM-DD:HH-mm-ss If the kickoff time is not known, it should be set to 'TBD'

property **kickoff_datetime**

Returns A datetime object representing the kickoff time

matchday: **int**

The match day of the match

property **minute_display**

This generates a string for displaying the current match minute. Sadly, since OpenligaDB does not provide information on the current minute, this can only offer an approximation. **:return:** A formatted string displaying the current match minute

season: **int**

The season of the match

started: **bool**

Indicates whether or not the match has started yet

bundesliga_tippspiel.db.match_data.Player module

class `bundesliga_tippspiel.db.match_data.Player.Player` (*args, **kwargs)

Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the "players" SQL table

__init__ (*args, **kwargs)

Initializes the Model **:param** args: The constructor arguments **:param** kwargs: The constructor keyword arguments

goals: **List[Goal]**

The goals the player scored.

id

name: **str**

The name of the player

team: `bundesliga_tippspiel.db.match_data.Team.Team`

The team the player is affiliated with.

team_id: **int**

The ID of the team the player is affiliated with. Acts as a foreign key to the 'teams' table.

bundesliga_tippspiel.db.match_data.Team module

class `bundesliga_tippspiel.db.match_data.Team.Team` (*args, **kwargs)

Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'teams' SQL table A Team is the most basic data for a match, it relies on no other data, only primitives

__init__ (*args, **kwargs)

Initializes the Model **:param** args: The constructor arguments **:param** kwargs: The constructor keyword arguments

abbreviation: `str`
A three-letter abbreviation of the team's name. Has to be unique. Example: FCB

icon_png: `str`
The URL of an image file representing the team's logo in PNG format

icon_svg: `str`
The URL of an image file representing the team's logo in SVG format

id

name: `str`
The full name of the team. Has to be unique. Example: FC Bayern München

players: `List[Player]`
The players of this team

season_position_bets: `List[SeasonPositionBet]`
The players of this team

season_team_bets: `List[SeasonTeamBet]`
The players of this team

short_name: `str`
The shortened version of the team's name. Has to be unique. Example: Bayern

Module contents

bundesliga_tippspiel.db.settings package

Submodules

bundesliga_tippspiel.db.settings.DisplayBotsSettings module

```
class bundesliga_tippspiel.db.settings.DisplayBotsSettings.DisplayBotsSettings (*args,  
                                                                           **kwargs)  
  
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.  
api.Model  
  
Database model that specifies whether a user wants to see bots or not  
  
__init__ (*args, **kwargs)  
    Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword  
    arguments  
  
display_bots  
    Whether or not to show bots  
  
id  
  
user: puffotter.flask.db.User.User  
    The user associated with this setting  
  
user_id: int  
    The ID of the user associated with this setting
```

bundesliga_tippspiel.db.settings.ReminderSettings module

```
class bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings (*args,
                                                                    **kwargs)
    Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.
           api.Model
    Database model that keeps track of reminder settings
    __init__ (*args, **kwargs)
        Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword
        arguments
    active
        Whether the reminder is active or not
    get_due_matches () → List[bundesliga_tippspiel.db.match_data.Match.Match]
        Checks if the reminder is due and returns a list of matches that the user still needs to bet on. :return: The
        matches for which the reminder is due
    id
    last_reminder: str
        The time when the last reminder was sent. Format in the form %Y-%m-%d:%H-%M-%S
    property last_reminder_datetime
        Returns The 'last_reminder' parameter as a datetime object
    reminder_time: int
        The time before the next unbet match when the reminder message will be sent. Unit: seconds
    property reminder_time_delta
        Returns The 'reminder_time' parameter as a datetime timedelta
    reminder_type
        The type of reminder
    send_reminder ()
        Sends a reminder message if it's due :return: None
    send_reminder_message (message: str)
        Sends a reminder message using the appropriate method of delivery :param message: The message to send
        :return: None
    set_reminder_time (reminder_time: int)
        Sets the reminder time and resets the time stored as the last reminder :param reminder_time: the new
        reminder time :return: None
    user: puffotter.flask.db.User.User
        The user associated with this setting
    user_id: int
        The ID of the user associated with this setting
```

Module contents

bundesliga_tippspiel.db.user_generated package

Submodules

bundesliga_tippspiel.db.user_generated.Bet module

class `bundesliga_tippspiel.db.user_generated.Bet.Bet` (**args*, ***kwargs*)
Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'bets' SQL table

__init__ (**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

away_score: **int**
The score bet on the away team

evaluate (*when_finished: bool = False*) → int
Evaluates the points score on this bet :param when_finished: Only calculate the value when the match is finished. Otherwise, returns 0

Returns The calculated points

home_score: **int**
The score bet on the home team

id

match: `bundesliga_tippspiel.db.match_data.Match.Match`
The match that this bet refers to

match_id: **int**
The ID of the match that this bet refers to.

user: `puffotter.flask.db.User.User`
The user associated with this bet

user_id: **int**
The ID of the user associated with this bet

bundesliga_tippspiel.db.user_generated.ChatMessage module

class `bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage` (**args*, ***kwargs*)
Bases: `puffotter.flask.db.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'chat_messages' SQL table

__init__ (**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

children: `List[bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage]`
Any child chat messages

creation_time: `float`
The timestamp of when the message was created

delete()
Marks the message as deleted :return: None

deleted: `bool`
Whether the message has been deleted

edit (*new_text: str*)
Edits the message :param new_text: The new message text :return: None

edited: `bool`
Whether the message has been edited

get_text () → `Optional[str]`
Returns The text of the chat message if the user still exists and the message has not been deleted

id: `int`
The ID of the chat message

last_edit: `float`
The timestamp of when the message was last edited

parent: `bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage`
The parent chat message

parent_id: `int`
ID of the parent chat message

text: `str`
The text of the message

user: `Optional[puffotter.flask.db.User.User]`
The user associated with this bet

user_id: `int`
The ID of the user associated with this bet

bundesliga_tippspiel.db.user_generated.SeasonPositionBet module

class `bundesliga_tippspiel.db.user_generated.SeasonPositionBet.SeasonPositionBet` (**args, **kwargs*)

Bases: `puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model`

Model that describes the ‘season_position_bets’ SQL table

__init__ (**args, **kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

id

position: `int`
The position of the team in the table

season: `int`
The season of the season bet

team: *bundesliga_tippspiel.db.match_data.Team.Team*

The team the position bet is for.

team_id: `int`

The ID of the team the position bet is for.

user: *puffotter.flask.db.User.User*

The user associated with this season position bet

user_id: `int`

The ID of the user associated with this season position bet

bundesliga_tippspiel.db.user_generated.SeasonTeamBet module

class *bundesliga_tippspiel.db.user_generated.SeasonTeamBet*.**SeasonTeamBet** (*args, **kwargs)

Bases: *puffotter.flask.db.ModelMixin.ModelMixin*, *sqlalchemy.ext.declarative.api.Model*

Model that describes the 'season_team_bets' SQL table

__init__ (*args, **kwargs)

Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

bet_type: *bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBetType*

The type of the bet

id

season: `int`

The season of the season bet

team: *bundesliga_tippspiel.db.match_data.Team.Team*

The team the user bet on.

team_id: `int`

The ID of the team the user bet on.

user: *puffotter.flask.db.User.User*

The user associated with this bet

user_id: `int`

The ID of the user associated with this bet

class *bundesliga_tippspiel.db.user_generated.SeasonTeamBet*.**SeasonTeamBetType** (value)

Bases: *enum.Enum*

Class that specifies the various season team bets that are possible

LEAST_GOALS_CONCEDED = 'Die wenigsten Tore'

LEAST_GOALS_SCORED = 'Die wenigsten Gegentore'

MOST_GOALS_CONCEDED = 'Die meisten Gegentore'

MOST_GOALS_SCORED = 'Die meisten Tore'

MOST_OWN_GOALS = 'Die meisten Eigentore'

bundesliga_tippspiel.db.user_generated.SeasonWinner module

```
class bundesliga_tippspiel.db.user_generated.SeasonWinner.SeasonWinner (*args,
                                                                    **kwargs)
    Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.
           api.Model
    Model that describes the 'season_winners' SQL table
    __init__ (*args, **kwargs)
        Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword
        arguments
    id
    season: int
        The season for which this is the winner
    property season_string
        Returns The season string, e.g. 2019/20
    user: puffotter.flask.db.User.User
        The user that won the competition
    user_id: int
        The ID of the user that won the competition
```

Module contents**Submodules****bundesliga_tippspiel.db.SeasonEvent module**

```
class bundesliga_tippspiel.db.SeasonEvent.SeasonEvent (*args, **kwargs)
    Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.
           api.Model
    Model that describes the 'season_events' SQL table
    __init__ (*args, **kwargs)
        Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword
        arguments
    event_type: bundesliga_tippspiel.db.SeasonEvent.SeasonEventType
        The type of event
    executed: bool
        Whether the event was executed or not
    id
    season: int
        The season this event is for
class bundesliga_tippspiel.db.SeasonEvent.SeasonEventType (value)
    Bases: enum.Enum
    Enumeration that describes all the possible season event types
    MID_SEASON_REMINDER = 'mid_season_reminder'
```

```
POST_SEASON_WRAPUP = 'post_season_wrapup'
```

```
PRE_SEASON_MAIL = 'pre_season_mail'
```

Module contents

```
bundesliga_tippspiel.db.models: List[sqlalchemy.ext.declarative.api.Model] = [<class 'bun...  
The database models of the application
```

1.1.4 bundesliga_tippspiel.routes package

Subpackages

bundesliga_tippspiel.routes.api package

Submodules

bundesliga_tippspiel.routes.api.getters module

```
bundesliga_tippspiel.routes.api.getters.define_blueprint (blueprint_name: str) →  
flask.blueprints.Blueprint  
Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint
```

```
bundesliga_tippspiel.routes.api.getters.execute_getter (_id: Optional[int],  
action_cls: Type[bundesliga_tippspiel.actions.Action.GetAction])  
→ Dict[str, Any]  
Executes a getter API method using a Getter Action and an optional ID :param _id: Optional ID to use while  
fetching :param action_cls: The action class to use for fetching :return: The result
```

bundesliga_tippspiel.routes.api.putters module

```
bundesliga_tippspiel.routes.api.putters.define_blueprint (blueprint_name: str) →  
flask.blueprints.Blueprint  
Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint
```

Module contents

Submodules

bundesliga_tippspiel.routes.betting module

```
bundesliga_tippspiel.routes.betting.define_blueprint (blueprint_name: str) →  
flask.blueprints.Blueprint  
Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint
```


bundesliga_tippspiel.routes.chat module

`bundesliga_tippspiel.routes.chat.define_blueprint` (*blueprint_name: str*) → `flask.blueprints.Blueprint`

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

bundesliga_tippspiel.routes.information module

`bundesliga_tippspiel.routes.information.define_blueprint` (*blueprint_name: str*) → `flask.blueprints.Blueprint`

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

bundesliga_tippspiel.routes.reminder module

`bundesliga_tippspiel.routes.reminder.define_blueprint` (*blueprint_name: str*) → `flask.blueprints.Blueprint`

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

bundesliga_tippspiel.routes.settings module

`bundesliga_tippspiel.routes.settings.define_blueprint` (*blueprint_name: str*) → `flask.blueprints.Blueprint`

Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

Module contents

`bundesliga_tippspiel.routes.blueprint_generators`: `List[Tuple[Callable[str, flask.blueprints.Blueprint], str]]`
 Defines the functions used to create the various blueprints as well as their names

1.1.5 bundesliga_tippspiel.test package

Subpackages

bundesliga_tippspiel.test.actions package

Submodules

bundesliga_tippspiel.test.actions.ActionTestFramework module

bundesliga_tippspiel.test.actions.GetActionTestFramework module

bundesliga_tippspiel.test.actions.TestAction module

class `bundesliga_tippspiel.test.actions.TestAction.ActionTest` (*methodName='runTest'*)
 Bases: `bundesliga_tippspiel.test.TestFramework._TestFramework`

Class that tests the static helper functions of the Action class

test_getting_current_matchday ()
 Tests getting the current matchday :return: None

bundesliga_tippspiel.test.actions.TestGetBetAction module

```
class bundesliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction (methodName='runTest')
    Bases:          bundesliga_tippspiel.test.actions.GetActionTestFramework.
                  _GetActionTestFramework
    Class that tests the GetBet action

    property action_cls
        Returns The tested Action class

    test_fetching()
        Tests fetching using different methods :return: None

    test_fetching_by_match()
        Explicitly search by match ID, since there was a bug once :return: None

    test_using_filter_and_id()
        Tests that using an ID and an explicit filter does not work :return: None
```

bundesliga_tippspiel.test.actions.TestGetGoalAction module

```
class bundesliga_tippspiel.test.actions.TestGetGoalAction.TestGetGoalAction (methodName='runTe
    Bases:          bundesliga_tippspiel.test.actions.GetActionTestFramework.
                  _GetActionTestFramework
    Class that tests the GetGoal action

    property action_cls
        Returns The tested Action class

    test_fetching()
        Tests fetching using different methods :return: None

    test_using_filter_and_id()
        Tests that using an ID and an explicit filter does not work :return: None
```

bundesliga_tippspiel.test.actions.TestGetMatchAction module

```
class bundesliga_tippspiel.test.actions.TestGetMatchAction.TestGetMatchAction (methodName='run
    Bases:          bundesliga_tippspiel.test.actions.GetActionTestFramework.
                  _GetActionTestFramework
    Class that tests the GetMatch action

    property action_cls
        Returns The tested Action class

    test_fetching()
        Tests fetching using different methods :return: None

    test_using_filter_and_id()
        Tests that using an ID and an explicit filter does not work :return: None
```

bundesliga_tippspiel.test.actions.TestGetPlayerAction module

class bundesliga_tippspiel.test.actions.TestGetPlayerAction.**TestGetPlayerAction** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.actions.GetActionTestFramework.
_GetActionTestFramework

Class that tests the GetTeam action

property action_cls

Returns The tested Action class

test_fetching()
Tests fetching using different methods :return: None

test_using_filter_and_id()
Tests that using an ID and an explicit filter does not work :return: None

bundesliga_tippspiel.test.actions.TestGetReminderSettingsAction module

class bundesliga_tippspiel.test.actions.TestGetReminderSettingsAction.**TestGetReminderSettingsAction**

Bases: bundesliga_tippspiel.test.actions.ActionTestFramework.
_ActionTestFramework

Class that tests the GetReminderSettingsAction action

generate_action() → *bundesliga_tippspiel.actions.GetReminderSettingsAction.GetReminderSettingsAction*
Generates a valid SetReminderSettingsAction object :return: The generated SetReminderSettingsAction

setUp()
Sets up a user for testing :return: None

test_from_dict()
Tests the from_dict method :return: None

test_setting_updating_and_deleting_reminder()
Tests getting a reminder :return: None

bundesliga_tippspiel.test.actions.TestGetTeamAction module

class bundesliga_tippspiel.test.actions.TestGetTeamAction.**TestGetTeamAction** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.actions.GetActionTestFramework.
_GetActionTestFramework

Class that tests the GetTeam action

property action_cls

Returns The tested Action class

test_fetching()
Tests fetching using different methods :return: None

test_using_filter_and_id()
Tests that using an ID and an explicit filter does not work :return: None

bundesliga_tippspiel.test.actions.TestLeaderboardAction module

```
class bundesliga_tippspiel.test.actions.TestLeaderboardAction.TestLeaderboardAction (methodName='setUp')
    Bases:
        bundesliga_tippspiel.test.actions.ActionTestFramework.
        _ActionTestFramework

    Class that tests the Login action

    generate_action () → bundesliga_tippspiel.actions.LeaderboardAction.LeaderboardAction
        Generates a valid LeaderboardAction object :return: The generated LeaderboardAction

    setUp ()
        Sets up users for testing :return: None

    test_leaderboard ()
        Tests that the leaderboard is generated correctly :return: None

    test_leaderboard_for_matchday ()
        Tests generating a leaderboard for a specific matchday :return: None
```

bundesliga_tippspiel.test.actions.TestPlaceBetsAction module

```
class bundesliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction (methodName='setUp')
    Bases:
        bundesliga_tippspiel.test.actions.ActionTestFramework.
        _ActionTestFramework

    Test class that tests the place bets action

    assert_bet (match_id: int, home: int, away: int)
        Tests that a stored bet has the specified parameters :param match_id: The match ID of the bet to test :param
        home: The home score to check for :param away: The away score to check for :return: None

    generate_action () → bundesliga_tippspiel.actions.PlaceBetsAction.PlaceBetsAction
        Generates a new, valid Action object :return: The generated object

    setUp ()
        Sets up a user in the database :return: None

    test_bets_out_of_bounds ()
        Tests using bets that are out of bounds :return: None

    test_invalid_bets ()
        Tests using invalid data to place bets. :return: None

    test_mixing_valid_and_invalid_bets ()
        Tests placing both valid and invalid bets :return: None

    test_parsing_form_data ()
        Tests if parsing the form data works correctly :return: None

    test_placing_bets ()
        Tests placing bets. :return:

    test_updating_bets ()
        Tests placing bets again :return: None
```

bundesliga_tippspiel.test.actions.TestSetReminderSettingsAction module

class bundesliga_tippspiel.test.actions.TestSetReminderSettingsAction.**TestSetReminderSettingsAction**

Bases: bundesliga_tippspiel.test.actions.ActionTestFramework.
_ActionTestFramework

Class that tests the SetReminderSettings action

generate_action () → *bundesliga_tippspiel.actions.SetReminderSettingsAction.SetReminderSettingsAction*
Generates a valid SetReminderSettingsAction object :return: The generated SetReminderSettingsAction

setUp ()
Sets up a user for testing :return: None

test_setting_and_updating_reminder ()
Tests setting a reminder and updating it :return: None

test_setting_invalid_reminder_times ()
Tests setting a reminder time that's not $0 < t < 49$:return: None

Module contents

bundesliga_tippspiel.test.background package

Submodules

bundesliga_tippspiel.test.background.TestMatchDataUpdate module

class bundesliga_tippspiel.test.background.TestMatchDataUpdate.**TestMatchDataUpdate** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Unit test class that tests the match_data_getter script

assert_db_state ()
Performs multiple assertions on a filled database :return: None

test_icon_urls ()
Tests if all team icon URLs are valid :return: None

test_populating_twice ()
Tests populating the database. Twice. :return: None

bundesliga_tippspiel.test.background.TestReminders module

class bundesliga_tippspiel.test.background.TestReminders.**TestReminders** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Class that tests the sending of due email reminders

setUp ()
Sets up a user for testing :return: None

test_with_due_reminder ()
Tests running the action with a due reminder :return: None

test_with_non_due_reminder ()
Tests running the action with a reminder that's not due yet :return: None

test_without_stored_reminders ()
Tests running the action without any stored reminders :return: None

bundesliga_tippspiel.test.background.TestSeasonEvents module

class bundesliga_tippspiel.test.background.TestSeasonEvents.**TestSeasonEvents** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Class that tests the sending of due email reminders

setUp ()
Sets up a user for testing :return: None

test_executing ()
Tests if the events are executed correctly, and only once. :return: None

test_initializing ()
Tests initializing the season-wide database entries :return: None

test_midseason_reminder ()
Tests sending the midseason reminder :return: None

test_postseason_wrapup ()
Tests doing the postseason wrapup :return: None

test_preseason_reminder ()
Tests sending the preseason reminder if it's due :return: None

Module contents

bundesliga_tippspiel.test.models package

Subpackages

bundesliga_tippspiel.test.models.match_data package

Submodules

bundesliga_tippspiel.test.models.match_data.TestGoal module

class bundesliga_tippspiel.test.models.match_data.TestGoal.**TestGoal** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.models.ModelTestFramework._ModelTestFramework

Tests the Goal SQL model

setUp ()
Sets up the data needed by the tests :return: None

test_auto_increment ()
Tests that auto-incrementing works as expected :return: None

test_cascades ()
Tests if cascade deletes work correctly :return: None

test_deleting_from_db()
Tests deleting model objects from the database :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_missing_column_data()
Tests that missing column data is handled correctly :return: None

test_retrieving_from_db()
Tests retrieving model objects from the database :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

bundesliga_tippspiel.test.models.match_data.TestMatch module

class bundesliga_tippspiel.test.models.match_data.TestMatch.**TestMatch** (*methodName='runTest'*)
Bases: `bundesliga_tippspiel.test.models.ModelTestFramework`
`_ModelTestFramework`
Tests the Match SQL model

setUp()
Sets up the data needed by the tests :return: None

test_auto_increment()
Tests that auto-incrementing works as expected :return: None

test_cascades()
Tests if cascade deletes work correctly :return: None

test_deleting_from_db()
Tests deleting model objects from the database :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_kickoff_datetime_conversion()
Tests the kickoff_datetime method :return: None

test_minute_representation()
Tests the representation of the current minute of the match :return: None

test_missing_column_data()
Tests that missing column data is handled correctly :return: None

test_retrieving_from_db()
Tests retrieving model objects from the database :return: None

test_score_representations()
Tests the score representation attributes :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

bundesliga_tippspiel.test.models.match_data.TestPlayer module

```
class bundesliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer (methodName='runTest')
  Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
         _ModelTestFramework
  Tests the Player SQL model

  setUp ()
    Sets up the data needed by the tests :return: None

  test_auto_increment ()
    Tests that auto-incrementing works as expected :return: None

  test_cascades ()
    Tests if cascade deletes work correctly :return: None

  test_deleting_from_db ()
    Tests deleting model objects from the database :return: None

  test_json_representation ()
    Tests the JSON representation of the model :return: None

  test_missing_column_data ()
    Tests that missing column data is handled correctly :return: None

  test_retrieving_from_db ()
    Tests retrieving model objects from the database :return: None

  test_string_representation ()
    Tests the str and repr methods of the model :return: None

  test_uniqueness ()
    Tests that unique attributes are correctly checked :return: None
```

bundesliga_tippspiel.test.models.match_data.TestTeam module

```
class bundesliga_tippspiel.test.models.match_data.TestTeam.TestTeam (methodName='runTest')
  Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
         _ModelTestFramework
  Tests the Team SQL model

  setUp ()
    Sets up the data needed by the tests :return: None

  test_auto_increment ()
    Tests that auto-incrementing works as expected :return: None

  test_cascades ()
    Tests if cascade deletes work correctly :return: None

  test_deleting_from_db ()
    Tests deleting model objects from the database :return: None

  test_json_representation ()
    Tests the JSON representation of the model :return: None

  test_missing_column_data ()
    Tests that missing column data is handled correctly :return: None
```


test_retrieving_from_db()
Tests retrieving model objects from the database :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

Module contents

bundesliga_tippspiel.test.models.settings package

Submodules

bundesliga_tippspiel.test.models.settings.TestReminderSettings module

class bundesliga_tippspiel.test.models.settings.TestReminderSettings.**TestReminderSettings** (
 Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
 _ModelTestFramework
 Tests the ReminderSettings SQL model

setUp()
Sets up the data needed by the tests :return: None

test_auto_increment()
Tests that auto-incrementing works as expected :return: None

test_cascades()
Tests if cascade deletes work correctly :return: None

test_deleting_from_db()
Tests deleting model objects from the database :return: None

test_due()
Tests if due matches can be found correctly

Dates: Now — Reminder 1 — Match 1 — Reminder 2 — Match 2 -0——
 10min——30min——60min——120min- :return: None

test_due_when_bets_placed()
Tests if the is_due method works correctly when bets are placed :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_missing_column_data()
Tests that missing column data is handled correctly :return: None

test_retrieving_from_db()
Tests retrieving model objects from the database :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

Module contents

bundesliga_tippspiel.test.models.user_generated package

Submodules

bundesliga_tippspiel.test.models.user_generated.TestBet module

class bundesliga_tippspiel.test.models.user_generated.TestBet.**TestBet** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
_ModelTestFramework

Tests the Bet SQL model

setUp ()

Sets up the data needed by the tests :return: None

test_auto_increment ()

Tests that auto-incrementing works as expected :return: None

test_cascades ()

Tests if cascade deletes work correctly :return: None

test_deleting_from_db ()

Tests deleting model objects from the database :return: None

test_evaluating_bet_result ()

Tests evaluating the results of a bet :return: None

test_evaluating_unfinished_games ()

Tests that evaluating unfinished games behaves correctly :return: None

test_json_representation ()

Tests the JSON representation of the model :return: None

test_missing_column_data ()

Tests that missing column data is handled correctly :return: None

test_retrieving_from_db ()

Tests retrieving model objects from the database :return: None

test_string_representation ()

Tests the str and repr methods of the model :return: None

test_uniqueness ()

Tests that unique attributes are correctly checked :return: None

bundesliga_tippspiel.test.models.user_generated.TestChatMessage module

class bundesliga_tippspiel.test.models.user_generated.TestChatMessage.**TestChatMessage** (*method*

Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
_ModelTestFramework

Tests the ChatMessage SQL model

setUp ()

Sets up the data needed by the tests :return: None

test_auto_increment ()
Tests that auto-incrementing works as expected :return: None

test_deleting_from_db ()
Tests deleting model objects from the database :return: None

test_json_representation ()
Tests the JSON representation of the model :return: None

test_missing_column_data ()
Tests that missing column data is handled correctly :return: None

test_retrieving_from_db ()
Tests retrieving model objects from the database :return: None

test_string_representation ()
Tests the str and repr methods of the model :return: None

test_using_chat ()
Test typical interactions with the chat systems :return: None

bundesliga_tippspiel.test.models.user_generated.TestSeasonPositionBet module

class bundesliga_tippspiel.test.models.user_generated.TestSeasonPositionBet.**TestSeasonPositionBet**
Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
_ModelTestFramework
Tests the SeasonPositionBet SQL model

setUp ()
Sets up the data needed by the tests :return: None

test_cascades ()
Tests if cascade deletes work correctly :return: None

test_json_representation ()
Tests the JSON representation of the model :return: None

test_missing_column_data ()
Tests that missing column data is handled correctly :return: None

test_string_representation ()
Tests the str and repr methods of the model :return: None

test_uniqueness ()
Tests that unique attributes are correctly checked :return: None

bundesliga_tippspiel.test.models.user_generated.TestSeasonTeamBet module

class bundesliga_tippspiel.test.models.user_generated.TestSeasonTeamBet.**TestSeasonTeamBet**
Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
_ModelTestFramework
Tests the SeasonTeamBet SQL model

setUp ()
Sets up the data needed by the tests :return: None

test_cascades ()
Tests if cascade deletes work correctly :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_missing_column_data()
Tests that missing column data is handled correctly :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

bundesliga_tippspiel.test.models.user_generated.TestSeasonWinner module

class bundesliga_tippspiel.test.models.user_generated.TestSeasonWinner.**TestSeasonWinner** (*met*
Bases: bundesliga_tippspiel.test.models.ModelTestFramework.
_ModelTestFramework
Tests the SeasonWinner SQL model

setUp()
Sets up the data needed by the tests :return: None

test_cascades()
Tests if cascade deletes work correctly :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_missing_column_data()
Tests that missing column data is handled correctly :return: None

test_season_string()
Tests the season string :return: None

test_string_representation()
Tests the str and repr methods of the model :return: None

test_uniqueness()
Tests that unique attributes are correctly checked :return: None

Module contents

Submodules

bundesliga_tippspiel.test.models.ModelTestFramework module

bundesliga_tippspiel.test.models.TestMatchData module

class bundesliga_tippspiel.test.models.TestMatchData.**TestMatchData** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework
Class that tests the various match data models

test_model_relations()
Tests the relations between the models :return: None

bundesliga_tippspiel.test.models.TestSeasonEvent module

```

class bundesliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent (methodName='runTest')
  Bases:          bundesliga_tippspiel.test.models.ModelTestFramework.
                _ModelTestFramework
  Tests the SeasonEvent SQL model

  setUp ()
    Sets up the data needed by the tests :return: None

  test_cascades ()
    Tests if cascade deletes work correctly :return: None

  test_json_representation ()
    Tests the JSON representation of the model :return: None

  test_missing_column_data ()
    Tests that missing column data is handled correctly :return: None

  test_string_representation ()
    Tests the str and repr methods of the model :return: None

  test_uniqueness ()
    Tests that unique attributes are correctly checked :return: None

```

Module contents**bundesliga_tippspiel.test.routes package****Subpackages****bundesliga_tippspiel.test.routes.api package****Submodules****bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework module****bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework module****bundesliga_tippspiel.test.routes.api.TestApiKeyApiRoute module**

```

class bundesliga_tippspiel.test.routes.api.TestApiKeyApiRoute.TestApiKeyApiRoute (methodName='runTest')
  Bases:          bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework.
                _ApiRouteTestFramework
  Tests the /api_key API route

  property route_info
    Provides information about the route :return: The path of the route,
    A list of supported methods, Whether or not the API endpoint requires authorization

  test_successful_call ()
    Tests a successful API call :return: None

```

test_unsuccessful_call()
Tests an unsuccessful API call :return: None

bundesliga_tippspiel.test.routes.api.TestAuthorizeApiRoute module

class bundesliga_tippspiel.test.routes.api.TestAuthorizeApiRoute.**TestAuthorizeApiRoute** (*methodName*)
Bases: bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework.
_ApiRouteTestFramework
Tests the /authorize API route

property route_info
Provides information about the route :return: The path of the route,
A list of supported methods, Whether or not the API endpoint requires authorization

test_expired_api_key()
Tests using an expired API key :return: None

test_successful_call()
Tests a successful API call :return: None

test_unsuccessful_call()
Tests an unsuccessful API call :return: None

bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute module

class bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute.**TestGetBetApiRoute** (*methodName*)
Bases: bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework.
_GetterApiRouteTestFramework
Tests the /bet GET API route

property action_cls
Returns The action class used to fetch data

property keyword
Returns The route keyword

property sample_filters
Returns A sample dictionary of filters with appropriate values

bundesliga_tippspiel.test.routes.api.TestGetGoalApiRoute module

class bundesliga_tippspiel.test.routes.api.TestGetGoalApiRoute.**TestGetGoalApiRoute** (*methodName*)
Bases: bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework.
_GetterApiRouteTestFramework
Tests the /goal GET API route

property action_cls
Returns The action class used to fetch data

property keyword
Returns The route keyword.

property sample_filters**Returns** A sample dictionary of filters with appropriate values**bundesliga_tippspiel.test.routes.api.TestGetMatchApiRoute module****class** `bundesliga_tippspiel.test.routes.api.TestGetMatchApiRoute`.**TestGetMatchApiRoute** (*method*Bases: `bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework`,
`_GetterApiRouteTestFramework`

Tests the /match GET API route

property action_cls**Returns** The action class used to fetch data**property keyword****Returns** The route keyword.**property sample_filters****Returns** A sample dictionary of filters with appropriate values**bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRoute module****class** `bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRoute`.**TestGetPlayerApiRoute** (*meth*Bases: `bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework`,
`_GetterApiRouteTestFramework`

Tests the /player GET API route

property action_cls**Returns** The action class used to fetch data**property keyword****Returns** The route keyword.**property sample_filters****Returns** A sample dictionary of filters with appropriate values**bundesliga_tippspiel.test.routes.api.TestGetTeamApiRoute module****class** `bundesliga_tippspiel.test.routes.api.TestGetTeamApiRoute`.**TestGetTeamApiRoute** (*method*Bases: `bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework`,
`_GetterApiRouteTestFramework`

Tests the /team GET API route

property action_cls**Returns** The action class used to fetch data**property keyword****Returns** The route keyword**property sample_filters**

Returns A sample dictionary of filters with appropriate values

bundesliga_tippspiel.test.routes.api.TestLeaderboardApiRoute module

```
class bundesliga_tippspiel.test.routes.api.TestLeaderboardApiRoute.TestLeaderboardApiRoute
    Bases:          bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework.
                  _ApiRouteTestFramework
    Tests the /leaderboard API route

property route_info
    Provides information about the route :return: The path of the route,
        A list of supported methods, Whether or not the API endpoint requires authorization

test_successful_call ()
    Tests a successful API call :return: None

test_unsuccessful_call ()
    Tests an unsuccessful API call :return: None
```

bundesliga_tippspiel.test.routes.api.TestPutBetApiRoute module

```
class bundesliga_tippspiel.test.routes.api.TestPutBetApiRoute.TestPutBetApiRoute (methodName=
    Bases:          bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework.
                  _ApiRouteTestFramework
    Tests the /bet PUT API route

property route_info
    Provides information about the route :return: The path of the route,
        A list of supported methods, Whether or not the API endpoint requires authorization

setUp ()
    Sets up data for the tests :return: None

test_successful_call ()
    Tests a successful API call :return: None

test_unsuccessful_call ()
    Tests an unsuccessful API call :return: None
```

Module contents

Submodules

bundesliga_tippspiel.test.routes.RouteTestFramework module

bundesliga_tippspiel.test.routes.TestAboutRoute module

```
class bundesliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute (methodName='runTest')
    Bases:          bundesliga_tippspiel.test.routes.RouteTestFramework.
                  _RouteTestFramework
    Class that tests the /about route
```


property route_info

Info about the route to test :return: The route's path,
 the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

test_malformed_data ()

Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()

Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()

Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestBetsRoute module

class bundesliga_tippspiel.test.routes.TestBetsRoute.**TestBetsRoute** (*methodName='runTest'*)

Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
 _RouteTestFramework

Class that tests the /bets route

property route_info

Info about the route to test :return: The route's path,
 the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

setUp ()

Sets up data for the tests :return:

test_getting_specific_matchday ()

Tests retrieving a specific matchday :return: None

test_malformed_data ()

Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()

Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()

Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestChangePasswordRoute module

class bundesliga_tippspiel.test.routes.TestChangePasswordRoute.**TestChangePasswordRoute** (*meth*

Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
 _RouteTestFramework

Class that tests the /change_password route

property route_info

Info about the route to test :return: The route's path,
 the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

test_successful_requests ()

Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestConfirmRoute module

class bundesliga_tippspiel.test.routes.TestConfirmRoute.**TestConfirmRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework

Class that tests the /confirm route

property route_info

Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestDeleteUserRoute module

class bundesliga_tippspiel.test.routes.TestDeleteUserRoute.**TestDeleteUserRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework

Class that tests the /delete_user route

property route_info

Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestEmailReminderRoute module

class bundesliga_tippspiel.test.routes.TestEmailReminderRoute.**TestSetReminderRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework

Class that tests the /set_reminder route

property route_info

Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

```

test_successful_requests ()
    Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
    Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestForgotPasswordRoute module

```

class bundesliga_tippspiel.test.routes.TestForgotPasswordRoute.TestForgotPasswordRoute (method
    Bases:
        bundesliga_tippspiel.test.routes.RouteTestFramework.
        _RouteTestFramework
    Class that tests the /forgot route

    property route_info
        Info about the route to test :return: The route's path,
            the route's primary methods, A phrase found on the route's GET page. None if no such page
            exists, An indicator for if the page requires authentication or not

    test_successful_requests ()
        Tests (a) successful request(s) :return: None

    test_unsuccessful_requests ()
        Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestIndexRoute module

```

class bundesliga_tippspiel.test.routes.TestIndexRoute.TestIndexRoute (methodName='runTest')
    Bases:
        bundesliga_tippspiel.test.routes.RouteTestFramework.
        _RouteTestFramework
    Class that tests the / route

    property route_info
        Info about the route to test :return: The route's path,
            the route's primary methods, A phrase found on the route's GET page. None if no such page
            exists, An indicator for if the page requires authentication or not

    test_malformed_data ()
        Tests that malformed data in the request is handled appropriately :return: None

    test_successful_requests ()
        Tests (a) successful request(s) :return: None

    test_unsuccessful_requests ()
        Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestLeaderboardRoute module

```
class bundesliga_tippspiel.test.routes.TestLeaderboardRoute.TestLeaderboardRoute (methodName='runTest')
  Bases:
    bundesliga_tippspiel.test.routes.RouteTestFramework.
    _RouteTestFramework
  Class that tests the /leaderboard route

  property route_info
    Info about the route to test :return: The route's path,
    the route's primary methods, A phrase found on the route's GET page. None if no such page
    exists, An indicator for if the page requires authentication or not

  setUp ()
    Generates sample match data :return: None

  test_malformed_data ()
    Tests that malformed data in the request is handled appropriately :return: None

  test_successful_requests ()
    Tests (a) successful request(s) :return: None

  test_unsuccessful_requests ()
    Tests (an) unsuccessful request(s) :return: None
```

bundesliga_tippspiel.test.routes.TestLoginRoute module

```
class bundesliga_tippspiel.test.routes.TestLoginRoute.TestLoginRoute (methodName='runTest')
  Bases:
    bundesliga_tippspiel.test.routes.RouteTestFramework.
    _RouteTestFramework
  Class that tests the /login route

  property route_info
    Info about the route to test :return: The route's path,
    the route's primary methods, A phrase found on the route's GET page. None if no such page
    exists, An indicator for if the page requires authentication or not

  test_successful_requests ()
    Tests (a) successful request(s) :return: None

  test_unsuccessful_requests ()
    Tests (an) unsuccessful request(s) :return: None
```

bundesliga_tippspiel.test.routes.TestMatchRoute module

```
class bundesliga_tippspiel.test.routes.TestMatchRoute.TestMatchRoute (methodName='runTest')
  Bases:
    bundesliga_tippspiel.test.routes.RouteTestFramework.
    _RouteTestFramework
  Class that tests the /match route

  property route_info
    Info about the route to test :return: The route's path,
    the route's primary methods, A phrase found on the route's GET page. None if no such page
    exists, An indicator for if the page requires authentication or not
```

```

setUp ()
    Sets up data for the tests :return:

test_successful_requests ()
    Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
    Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestPrivacyRoute module

```

class bundesliga_tippspiel.test.routes.TestPrivacyRoute.TestPrivacyRoute (methodName='runTest')
    Bases:
        bundesliga_tippspiel.test.routes.RouteTestFramework.
        _RouteTestFramework

    Class that tests the /privacy route

    property route_info
        Info about the route to test :return: The route's path,

        the route's primary methods, A phrase found on the route's GET page. None if no such page
        exists, An indicator for if the page requires authentication or not

    test_malformed_data ()
        Tests that malformed data in the request is handled appropriately :return: None

    test_successful_requests ()
        Tests (a) successful request(s) :return: None

    test_unsuccessful_requests ()
        Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestProfileRoute module

```

class bundesliga_tippspiel.test.routes.TestProfileRoute.TestProfileRoute (methodName='runTest')
    Bases:
        bundesliga_tippspiel.test.routes.RouteTestFramework.
        _RouteTestFramework

    Class that tests the /profile route

    property route_info
        Info about the route to test :return: The route's path,

        the route's primary methods, A phrase found on the route's GET page. None if no such page
        exists, An indicator for if the page requires authentication or not

    test_malformed_data ()
        Tests that malformed data in the request is handled appropriately :return: None

    test_successful_requests ()
        Tests (a) successful request(s) :return: None

    test_unsuccessful_requests ()
        Tests (an) unsuccessful request(s) :return: None

```

bundesliga_tippspiel.test.routes.TestRegisterRoute module

class bundesliga_tippspiel.test.routes.TestRegisterRoute.**TestRegisterRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /register route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestSeasonBetsRoute module

class bundesliga_tippspiel.test.routes.TestSeasonBetsRoute.**TestSeasonBetsRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /season_bets route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

setUp ()
Sets up data for the tests :return: None

test_malformed_data ()
Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestSeasonPositionBetsRoute module

class bundesliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.**TestSeasonPositionBetsRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /season_position_bets route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

setUp ()
Sets up data for the tests :return:

test_malformed_data ()
Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute module

class bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.**TestSeasonTeamBetsRoute** (*method*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /season_team_bets route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

setUp ()
Sets up data for the tests :return:

test_malformed_data ()
Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestStatsRoute module

class bundesliga_tippspiel.test.routes.TestStatsRoute.**TestStatsRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /stats route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page exists, An indicator for if the page requires authentication or not

setUp ()
Generates sample match data :return: None

test_malformed_data ()
Tests that malformed data in the request is handled appropriately :return: None

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestTeamRoute module

class bundesliga_tippspiel.test.routes.TestTeamRoute.**TestTeamRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /team route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page
exists, An indicator for if the page requires authentication or not

setUp ()
Sets up data for the tests :return:

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

bundesliga_tippspiel.test.routes.TestUserRoute module

class bundesliga_tippspiel.test.routes.TestUserRoute.**TestUserRoute** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.routes.RouteTestFramework.
_RouteTestFramework
Class that tests the /user route

property route_info
Info about the route to test :return: The route's path,
the route's primary methods, A phrase found on the route's GET page. None if no such page
exists, An indicator for if the page requires authentication or not

setUp ()
Sets up data for the tests :return:

test_successful_requests ()
Tests (a) successful request(s) :return: None

test_unsuccessful_requests ()
Tests (an) unsuccessful request(s) :return: None

Module contents

bundesliga_tippspiel.test.utils package

Submodules

bundesliga_tippspiel.test.utils.TestChartData module

class bundesliga_tippspiel.test.utils.TestChartData.**TestChartData** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Tests chart data generation

test_generating_leaderboard_data ()
Tests generating leaderboard data :return: None

bundesliga_tippspiel.test.utils.TestCrypto module

class bundesliga_tippspiel.test.utils.TestCrypto.**TestCrypto** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Tests cryptographical functions

test_hashing ()
Tests that passwords can be hashed successfully :return: None

test_hashing_strings ()
Tests that password hashing and verifying works with string as well :return: None

test_salt ()
Tests that password hashes are salted :return: None

test_verifying_with_invalid_hash ()
Tests that attempting to verify a password with an incorrectly formatted hash will return False :return: None

bundesliga_tippspiel.test.utils.TestDb module

class bundesliga_tippspiel.test.utils.TestDb.**TestDb** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework

Class that tests database helper functions

test_user_exists_functions ()
Tests functions that check if a user exists or not :return: None

bundesliga_tippspiel.test.utils.TestJson module

class bundesliga_tippspiel.test.utils.TestJson.**TestJson** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework
Unit test class that tests the json methods used for the API
test_jsonifying ()
Tests jsonifying some objects :return: None

bundesliga_tippspiel.test.utils.TestStats module

class bundesliga_tippspiel.test.utils.TestStats.**TestStats** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework
Tests stats generation
test_generating_team_points_data ()
Tests generating team points data :return: None
test_rankings_before_season ()
Tests the generation of rankings before the season has started :return: None

Module contents

Submodules

bundesliga_tippspiel.test.TestConfig module

class bundesliga_tippspiel.test.TestConfig.**TestConfig** (*methodName='runTest'*)
Bases: bundesliga_tippspiel.test.TestFramework._TestFramework
Tests the config class
test_season_calculations ()
Tests if the season calculations are done correctly :return: None

bundesliga_tippspiel.test.TestFramework module

Module contents

1.1.6 bundesliga_tippspiel.utils package

Submodules

bundesliga_tippspiel.utils.chart_data module

`bundesliga_tippspiel.utils.chart_data.generate_leaderboard_data` (*current_matchday*: *Optional[int]* = *None*, *bets*: *Optional[List[bundesliga_tippspiel.db.user_...]]* = *None*, *include_bots*: *bool* = *False*) → *Tuple[int, Dict[str, Tuple[str, List[int]]]]*

Generates leaderboard data for the rankings chart :param *current_matchday*: The current matchday :param *bets*: A list of bets to work on. If not provided, will

load all bets in the database

Parameters *include_bots* – Whether or not to include bots

Returns

A tuple consisting of the matchday to display and the leaderboard data:

username: (colour, list of positions per matchday)

`bundesliga_tippspiel.utils.chart_data.load_leaderboard_history` (*current_matchday*: *Optional[int]* = *None*, *bets*: *Optional[List[bundesliga_tippspiel.db.user_...]]* = *None*, *include_bots*: *bool* = *False*) → *List[List[Tuple[puffotter.flask.db.User.U... int]]]*

Generates historical leaderboard data for chart generation :param *current_matchday*: The current matchday :param *bets*: A list of bets to work on. If not provided, will

load all bets in the database

Parameters *include_bots* – Whether or not to include bots

Returns The list of leaderboard lists

bundesliga_tippspiel.utils.db module

`bundesliga_tippspiel.utils.db.email_exists` (*email*: *str*) → *bool*

Checks if an email address already exists in the database :param *email*: The email to check for :return: True if the email exists in the database, False otherwise

`bundesliga_tippspiel.utils.db.user_exists` (*user_id*: *int*) → *bool*

Checks if a username already exists in the database :param *user_id*: The user's id :return: True if the user exists, False otherwise

`bundesliga_tippspiel.utils.db.username_exists` (*username: str*) → bool

Checks if a username already exists in the database :param username: The username to check :return: True if the username exists in the database, False if not

bundesliga_tippspiel.utils.routes module

`bundesliga_tippspiel.utils.routes.action_route` (*func: Callable*) → Callable

Decorator that catches any ActionExceptions and displays appropriate error messages :param func: The function to wrap :return: The wrapped function

bundesliga_tippspiel.utils.stats module

`bundesliga_tippspiel.utils.stats.create_participation_ranking` (*bets: Optional[List[bundesliga_tippspiel.db.user_g = None, include_bots: bool = False]*) → List[Tuple[puffotter.flask.db.User.User, str]]

Creates a ranking of user's participation percentages :param bets: The bets to analyze. If not provided, will analyze all bets :param include_bots: Whether or not to include bots :return: A sorted list of tuples detailing the participation ranking

`bundesliga_tippspiel.utils.stats.create_point_average_ranking` (*bets: Optional[List[bundesliga_tippspiel.db.user_g = None, include_bots: bool = False]*) → List[Tuple[puffotter.flask.db.User.User, str]]

Creates a ranking of points averages :param bets: The bets to analyze :param include_bots: Whether or not to include bots :return: The ranking

`bundesliga_tippspiel.utils.stats.generate_points_distributions` (*bets: Optional[List[bundesliga_tippspiel.db.user_g = None]*) → Dict[puffotter.flask.db.User.User, Dict[int, int]]

Generates a distribution detailing how often a given amount of points a user earned while betting :param bets: The bets to analyze. If not provided, will analyze all bets :return: A dictionary mapping users to point amounts to their

appearance count

`bundesliga_tippspiel.utils.stats.generate_team_points_table` (*team_points: Dict[bundesliga_tippspiel.db.match_data.Team, int]*) → List[Tuple[bundesliga_tippspiel.db.match_data.Team, int]]

Generates a sorted list of tuples of teams and their points. :param team_points: The points achieved by the teams :return: The sorted list of tuples

```
bundesliga_tippspiel.utils.stats.get_team_points_data (bets: Optional[List[bundesliga_tippspiel.db.user_generated.Bet]] = None) → Dict[puffotter.flask.db.User.User, Dict[bundesliga_tippspiel.db.match_data.Team.Team, int]]
```

Generates information about the amount of points each user achieved betting on specific teams :param bets: The bets to analyze. If not provided, will analyze all bets :return: A dictionary mapping users to dictionaries mapping teams to points

```
bundesliga_tippspiel.utils.stats.get_total_points_per_team (bets: Optional[List[bundesliga_tippspiel.db.user_generated.Bet]] = None) → Dict[bundesliga_tippspiel.db.match_data.Team.Team, int]
```

Retrieves the total amount of points achieved by betting on individual teams :param bets: The bets to analyze. If not provided, will analyze all bets :return: A dictionary mapping the teams to the points achieved by users betting on them

Module contents

1.2 Submodules

1.3 bundesliga_tippspiel.Config module

```
class bundesliga_tippspiel.Config.Config
```

```
Bases: puffotter.flask.Config.Config
```

Configuration for the flask application

```
OPENLIGADB_LEAGUE: str
```

The openligadb league to use

```
OPENLIGADB_SEASON: str
```

The openligadb season to use

```
classmethod environment_variables () → Dict[str, List[str]]
```

Specifies required and optional environment variables :return: The specified environment variables in two lists in

a dictionary, grouped by whether the variables are required or optional

```
classmethod season () → int
```

Returns The current season

```
classmethod season_string (year: Optional[int] = None) → str
```

Returns The season string, e.g. 2020/21 for 2020

1.4 bundesliga_tippspiel.enums module

class `bundesliga_tippspiel.enums.ReminderType` (*value*)

Bases: `enum.Enum`

Class that specifies the different reminder types

EMAIL = 'email'

TELEGRAM = 'telegram'

1.5 bundesliga_tippspiel.exceptions module

exception `bundesliga_tippspiel.exceptions.ActionException` (*reason: str, display_message: str, status_code: int = 400, severity: puffotter.flask.enums.AlertSeverity = <AlertSeverity.DANGER: 'danger'>*)

Bases: `Exception`

An exception that gets raised whenever an action method fails for whatever reason. The reason should be provided as a parameter

__init__ (*reason: str, display_message: str, status_code: int = 400, severity: puffotter.flask.enums.AlertSeverity = <AlertSeverity.DANGER: 'danger'>*)

Initializes the Exception while adding the 'reason' variable :param reason: The reason for the exception :param display_message: The message to display :param status_code: The status code corresponding to the exception

Defaults to 400

Parameters severity – The severity of the exception. Defaults to DANGER

flash ()

Flashes the message so that it can be displayed on the next redirect :return: None

1.6 bundesliga_tippspiel.jinja_extras module

`bundesliga_tippspiel.jinja_extras.get_matchday_total_pill_class` (*points: int*) →

Calculates the appropriate pill badge for an amount of points on a matchday :param points: ^{str}The points for which to get the pill class :return: The appropriate pill badge class

`bundesliga_tippspiel.jinja_extras.get_pill_class` (*points: int*) → str

Calculates the appropriate pill badge for an amount of points :param points: The points for which to get the pill class :return: The appropriate pill badge class

`bundesliga_tippspiel.jinja_extras.jinja_extras` () → Dict[str, Any]

Makes sure that jinja has access to these variables :return: The variables to forward to jinja

1.7 bundesliga_tippspiel.main module

`bundesliga_tippspiel.main.main()`

Initializes and starts the flask application :return: None

1.8 bundesliga_tippspiel.template_extras module

`bundesliga_tippspiel.template_extras.profile_extras()` → Dict[str, Any]

Makes sure that the profile page has access to information on email reminders. :return: The variables to forward to the template

1.9 Module contents

`bundesliga_tippspiel.root_path: str = '/usr/local/lib/python3.6/dist-packages/bundesliga_t`

The root path of the application

`bundesliga_tippspiel.sentry_dsn = 'https://e91e468e84424758bd74e6908af2c565@sentry.namibsu`

The sentry DSN used for exception logging

BUNDESLIGA_TIPPSPIEL

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

bundesliga_tippspiel, 51

bundesliga_tippspiel.actions, 9

bundesliga_tippspiel.actions.Action, 3

bundesliga_tippspiel.actions.ChangeSettingsAction, 4

bundesliga_tippspiel.actions.GetBetAction, 4

bundesliga_tippspiel.actions.GetGoalAction, 5

bundesliga_tippspiel.actions.GetMatchAction, 6

bundesliga_tippspiel.actions.GetPlayerAction, 6

bundesliga_tippspiel.actions.GetReminderSettingsAction, 7

bundesliga_tippspiel.actions.GetTeamAction, 7

bundesliga_tippspiel.actions.LeaderboardAction, 7

bundesliga_tippspiel.actions.LoadSettingsAction, 8

bundesliga_tippspiel.actions.PlaceBetsAction, 8

bundesliga_tippspiel.actions.SetReminderSettingsAction, 9

bundesliga_tippspiel.background, 11

bundesliga_tippspiel.background.match_data, 9

bundesliga_tippspiel.background.reminders, 10

bundesliga_tippspiel.background.season_events, 10

bundesliga_tippspiel.Config, 49

bundesliga_tippspiel.db, 20

bundesliga_tippspiel.db.match_data, 14

bundesliga_tippspiel.db.match_data.Goal, 11

bundesliga_tippspiel.db.match_data.Match, 12

bundesliga_tippspiel.db.match_data.Player, 13

bundesliga_tippspiel.db.match_data.Team, 13

bundesliga_tippspiel.db.SeasonEvent, 19

bundesliga_tippspiel.db.settings, 16

bundesliga_tippspiel.db.settings.DisplayBotsSettings, 14

bundesliga_tippspiel.db.settings.ReminderSettings, 15

bundesliga_tippspiel.db.user_generated, 19

bundesliga_tippspiel.db.user_generated.Bet, 16

bundesliga_tippspiel.db.user_generated.ChatMessage, 16

bundesliga_tippspiel.db.user_generated.SeasonPosition, 17

bundesliga_tippspiel.db.user_generated.SeasonTeamBet, 18

bundesliga_tippspiel.db.user_generated.SeasonWinner, 19

bundesliga_tippspiel.enums, 50

bundesliga_tippspiel.exceptions, 50

bundesliga_tippspiel.jinja_extras, 50

bundesliga_tippspiel.main, 51

bundesliga_tippspiel.routes, 21

bundesliga_tippspiel.routes.api, 20

bundesliga_tippspiel.routes.api.getters, 20

bundesliga_tippspiel.routes.api.putters, 20

bundesliga_tippspiel.routes.betting, 20

bundesliga_tippspiel.routes.chat, 21

bundesliga_tippspiel.routes.information, 21

bundesliga_tippspiel.routes.reminder, 21

bundesliga_tippspiel.routes.settings, 21

bundesliga_tippspiel.template_extras, 51

bundesliga_tippspiel.test, 46

bundesliga_tippspiel.test.actions, 25

bundesliga_tippspiel.test.actions.ActionTestFramework, 32
 21 bundesliga_tippspiel.test.models.user_generated.TestAction, 30
 bundesliga_tippspiel.test.actions.GetActionTestFramework, 30
 21 bundesliga_tippspiel.test.models.user_generated.TestAction, 30
 bundesliga_tippspiel.test.actions.TestAction, 30
 21 bundesliga_tippspiel.test.models.user_generated.TestAction, 30
 bundesliga_tippspiel.test.actions.TestGetBetAction, 31
 22 bundesliga_tippspiel.test.models.user_generated.TestAction, 31
 bundesliga_tippspiel.test.actions.TestGetGoalAction, 31
 22 bundesliga_tippspiel.test.models.user_generated.TestAction, 31
 bundesliga_tippspiel.test.actions.TestGetMatchAction, 32
 22 bundesliga_tippspiel.test.routes, 45
 bundesliga_tippspiel.test.actions.TestGetPlayerAction, 32
 23 bundesliga_tippspiel.test.routes.api, 36
 bundesliga_tippspiel.test.actions.TestGetReminderSettingsAction, 33
 23 bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework, 33
 bundesliga_tippspiel.test.actions.TestGetTeamAction, 33
 23 bundesliga_tippspiel.test.routes.api.GetterApiRouteTestFramework, 33
 bundesliga_tippspiel.test.actions.TestLeaderboardAction, 33
 24 bundesliga_tippspiel.test.routes.api.TestApiKeyApiRouteTestFramework, 33
 bundesliga_tippspiel.test.actions.TestPlaceBetAction, 34
 24 bundesliga_tippspiel.test.routes.api.TestAuthorizedApiRouteTestFramework, 34
 bundesliga_tippspiel.test.actions.TestSeasonEventAction, 34
 25 bundesliga_tippspiel.test.routes.api.TestGetBetApiRouteTestFramework, 34
 bundesliga_tippspiel.test.background, 34
 26 bundesliga_tippspiel.test.routes.api.TestGetGoalApiRouteTestFramework, 34
 bundesliga_tippspiel.test.background.TestMatchDataUpdates, 35
 25 bundesliga_tippspiel.test.routes.api.TestGetMatchApiRouteTestFramework, 35
 bundesliga_tippspiel.test.background.TestReminderSettings, 35
 25 bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRouteTestFramework, 35
 bundesliga_tippspiel.test.background.TestSeasonEvents, 35
 26 bundesliga_tippspiel.test.routes.api.TestGetTeamApiRouteTestFramework, 35
 bundesliga_tippspiel.test.models, 33 bundesliga_tippspiel.test.routes.api.TestLeaderboardApiRouteTestFramework, 33
 bundesliga_tippspiel.test.models.match_data, 36
 29 bundesliga_tippspiel.test.routes.api.TestPutBetApiRouteTestFramework, 36
 bundesliga_tippspiel.test.models.match_data.TestGoal, 36
 26 bundesliga_tippspiel.test.routes.RouteTestFramework, 36
 bundesliga_tippspiel.test.models.match_data.TestMatch, 36
 27 bundesliga_tippspiel.test.routes.TestAboutRoute, 36
 bundesliga_tippspiel.test.models.match_data.TestPlayer, 36
 28 bundesliga_tippspiel.test.routes.TestBetsRoute, 36
 bundesliga_tippspiel.test.models.match_data.TestTeam, 37
 28 bundesliga_tippspiel.test.routes.TestChangePasswordRoute, 37
 bundesliga_tippspiel.test.models.ModelTestFramework, 37
 32 bundesliga_tippspiel.test.routes.TestConfirmRoute, 37
 bundesliga_tippspiel.test.models.settings, 38
 30 bundesliga_tippspiel.test.routes.TestDeleteUserRoute, 38
 bundesliga_tippspiel.test.models.settings.TestReminderSettings, 38
 29 bundesliga_tippspiel.test.routes.TestEmailReminderRoute, 38
 bundesliga_tippspiel.test.models.TestMatchData, 38
 32 bundesliga_tippspiel.test.routes.TestForgotPasswordRoute, 38
 bundesliga_tippspiel.test.models.TestSeasonEvent, 39
 33 bundesliga_tippspiel.test.routes.TestIndexRoute, 39
 bundesliga_tippspiel.test.models.user_generated, 39

bundesliga_tippspiel.test.routes.TestLeaderboardRoute,
40
bundesliga_tippspiel.test.routes.TestLoginRoute,
40
bundesliga_tippspiel.test.routes.TestMatchRoute,
40
bundesliga_tippspiel.test.routes.TestPrivacyRoute,
41
bundesliga_tippspiel.test.routes.TestProfileRoute,
41
bundesliga_tippspiel.test.routes.TestRegisterRoute,
42
bundesliga_tippspiel.test.routes.TestSeasonBetsRoute,
42
bundesliga_tippspiel.test.routes.TestSeasonPositionBetsRoute,
42
bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute,
43
bundesliga_tippspiel.test.routes.TestStatsRoute,
43
bundesliga_tippspiel.test.routes.TestTeamRoute,
44
bundesliga_tippspiel.test.routes.TestUserRoute,
44
bundesliga_tippspiel.test.TestConfig,
46
bundesliga_tippspiel.test.TestFramework,
46
bundesliga_tippspiel.test.utils, 46
bundesliga_tippspiel.test.utils.TestChartData,
45
bundesliga_tippspiel.test.utils.TestCrypto,
45
bundesliga_tippspiel.test.utils.TestDb,
45
bundesliga_tippspiel.test.utils.TestJson,
46
bundesliga_tippspiel.test.utils.TestStats,
46
bundesliga_tippspiel.utils, 49
bundesliga_tippspiel.utils.chart_data,
47
bundesliga_tippspiel.utils.db, 47
bundesliga_tippspiel.utils.routes, 48
bundesliga_tippspiel.utils.stats, 48

INDEX

Symbols

`__init__()` (`bundesliga_tippspiel.actions.Action.GetAction` method), 4

`__init__()` (`bundesliga_tippspiel.actions.ChangeSettingsAction.ChangeSettingsAction` method), 4

`__init__()` (`bundesliga_tippspiel.actions.GetBetAction.GetBetAction` method), 4

`__init__()` (`bundesliga_tippspiel.actions.GetGoalAction.GetGoalAction` method), 5

`__init__()` (`bundesliga_tippspiel.actions.GetMatchAction.GetMatchAction` method), 6

`__init__()` (`bundesliga_tippspiel.actions.GetPlayerAction.GetPlayerAction` method), 6

`__init__()` (`bundesliga_tippspiel.actions.GetTeamAction.GetTeamAction` method), 7

`__init__()` (`bundesliga_tippspiel.actions.LeaderboardAction.LeaderboardAction` method), 7

`__init__()` (`bundesliga_tippspiel.actions.LoadSettingsAction.LoadSettingsAction` method), 8

`__init__()` (`bundesliga_tippspiel.actions.PlaceBetsAction.PlaceBetsAction` method), 8

`__init__()` (`bundesliga_tippspiel.actions.SetReminderSettingsAction.SetReminderSettingsAction` method), 9

`__init__()` (`bundesliga_tippspiel.db.SeasonEvent.SeasonEvent` method), 19

`__init__()` (`bundesliga_tippspiel.db.match_data.Goal.Goal` method), 11

`__init__()` (`bundesliga_tippspiel.db.match_data.Match.Match` method), 12

`__init__()` (`bundesliga_tippspiel.db.match_data.Player.Player` method), 13

`__init__()` (`bundesliga_tippspiel.db.match_data.Team.Team` method), 13

`__init__()` (`bundesliga_tippspiel.db.settings.DisplayBotsSettings.DisplayBotsSettings` method), 14

`__init__()` (`bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings` method), 15

`__init__()` (`bundesliga_tippspiel.db.user_generated.Bet.Bet` method), 16

`__init__()` (`bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage` method), 16

`__init__()` (`bundesliga_tippspiel.db.user_generated.SeasonPosition.BestSeasonPosition` method), 17

`__init__()` (`bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet` method), 18

`__init__()` (`bundesliga_tippspiel.db.user_generated.SeasonWinner.SeasonWinner` method), 19

`__init__()` (`bundesliga_tippspiel.exceptions.ActionException.ActionException` method), 50

A

`ab.Action` (`bundesliga_tippspiel.db.match_data.Team.Team` property), 14

`Action` (class in `bundesliga_tippspiel.actions.Action`), 3

`add_action_cls()` (`bundesliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction` method), 7

`add_action_cls()` (`bundesliga_tippspiel.test.actions.TestGetGoalAction.TestGetGoalAction` method), 8

`add_action_cls()` (`bundesliga_tippspiel.test.actions.TestGetMatchAction.TestGetMatchAction` method), 8

`add_action_cls()` (`bundesliga_tippspiel.test.actions.TestGetPlayerAction.TestGetPlayerAction` method), 9

`add_action_cls()` (`bundesliga_tippspiel.test.actions.TestGetTeamAction.TestGetTeamAction` method), 11

`add_action_cls()` (`bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute.TestGetBetApiRoute` method), 13

`add_action_cls()` (`bundesliga_tippspiel.test.routes.api.TestGetGoalApiRoute.TestGetGoalApiRoute` method), 14

`add_action_cls()` (`bundesliga_tippspiel.test.routes.api.TestGetMatchApiRoute.TestGetMatchApiRoute` method), 15

`add_action_cls()` (`bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRoute.TestGetPlayerApiRoute` method), 16

`add_action_cls()` (`bundesliga_tippspiel.test.routes.api.TestGetTeamApiRoute.TestGetTeamApiRoute` method), 17

property), 35
 action_route() (in module *bundesliga_tippspiel.utils.routes*), 48
 ActionException, 50
 ActionTest (class in module *bundesliga_tippspiel.test.actions.TestAction*), 21
 active (attribute of *bundesliga_tippspiel.db.settings.ReminderSettings*), 15
 assert_bet() (method of *bundesliga_tippspiel.test.actions.TestPlaceBetsAction*), 24
 assert_db_state() (method of *bundesliga_tippspiel.test.background.TestMatchDataUpdateTestMatchDataUpdate*), 25
 away_current_score (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
 away_ft_score (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
 away_ht_score (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
 away_score (attribute of *bundesliga_tippspiel.db.match_data.Goal.Goal*), 11
 away_score (attribute of *bundesliga_tippspiel.db.user_generated.Bet.Bet*), 16
 away_team (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
 away_team_id (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
B
 Bet (class in *bundesliga_tippspiel.db.user_generated.Bet*), 16
 bet_type (attribute of *bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet*), 18
 bets (attribute of *bundesliga_tippspiel.db.match_data.Match.Match*), 12
 bg_tasks (in module *bundesliga_tippspiel.background*), 11
 blueprint_generators (in module *bundesliga_tippspiel.routes*), 21
 bundesliga_tippspiel module, 51
 bundesliga_tippspiel.actions module, 9
 bundesliga_tippspiel.actions.Action module, 3
 bundesliga_tippspiel.actions.ChangeSettingsAction module, 4
 bundesliga_tippspiel.actions.GetBetAction module, 4
 bundesliga_tippspiel.actions.GetGoalAction module, 5
 bundesliga_tippspiel.actions.GetMatchAction module, 6
 bundesliga_tippspiel.actions.GetPlayerAction module, 6
 bundesliga_tippspiel.actions.GetReminderSettingsAction module, 7
 bundesliga_tippspiel.actions.GetTeamAction module, 7
 bundesliga_tippspiel.actions.PlaceBetsAction module, 8
 bundesliga_tippspiel.actions.LeaderboardAction module, 7
 bundesliga_tippspiel.actions.LoadSettingsAction module, 8
 bundesliga_tippspiel.actions.PlaceBetsAction module, 8
 bundesliga_tippspiel.actions.SetReminderSettingsAction module, 9
 bundesliga_tippspiel.background module, 11
 bundesliga_tippspiel.background.match_data module, 9
 bundesliga_tippspiel.background.reminders module, 10
 bundesliga_tippspiel.background.season_events module, 10
 bundesliga_tippspiel.Config module, 49
 bundesliga_tippspiel.db module, 20
 bundesliga_tippspiel.db.match_data module, 14
 bundesliga_tippspiel.db.match_data.Goal module, 11
 bundesliga_tippspiel.db.match_data.Match module, 12
 bundesliga_tippspiel.db.match_data.Player module, 13
 bundesliga_tippspiel.db.match_data.Team module, 13
 bundesliga_tippspiel.db.SeasonEvent module, 19
 bundesliga_tippspiel.db.settings module, 16
 bundesliga_tippspiel.db.settings.DisplayBotsSettings module, 14
 bundesliga_tippspiel.db.settings.ReminderSettings module, 15
 bundesliga_tippspiel.db.user_generated module, 19
 bundesliga_tippspiel.db.user_generated.Bet module, 16
 bundesliga_tippspiel.db.user_generated.ChatMessage module, 16

module, 36
 bundesliga_tippspiel.test.routes.api.ApiRoute module, 42
 module, 33
 bundesliga_tippspiel.test.routes.api.GetBundesligaTestFramework module, 42
 module, 33
 bundesliga_tippspiel.test.routes.api.GetBundesligaTestFramework module, 43
 bundesliga_tippspiel.test.routes.api.TestApiKeyApiRoute module, 43
 module, 33
 bundesliga_tippspiel.test.routes.api.TestApiKeyApiRoute module, 43
 bundesliga_tippspiel.test.routes.api.TestAuthApiRoute module, 44
 module, 34
 bundesliga_tippspiel.test.routes.api.TestAuthApiRoute module, 44
 bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute module, 44
 module, 34
 bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute module, 44
 bundesliga_tippspiel.test.routes.api.TestGetGoalApiRoute module, 46
 module, 34
 bundesliga_tippspiel.test.routes.api.TestGetGoalApiRoute module, 46
 bundesliga_tippspiel.test.routes.api.TestGetMatchApiRoute module, 46
 module, 35
 bundesliga_tippspiel.test.routes.api.TestGetMatchApiRoute module, 46
 bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRoute module, 46
 module, 35
 bundesliga_tippspiel.test.routes.api.TestGetPlayerApiRoute module, 46
 bundesliga_tippspiel.test.routes.api.TestGetTeamApiRoute module, 45
 module, 35
 bundesliga_tippspiel.test.routes.api.TestGetTeamApiRoute module, 45
 bundesliga_tippspiel.test.routes.api.TestLedderborgApiRoute module, 45
 module, 36
 bundesliga_tippspiel.test.routes.api.TestLedderborgApiRoute module, 45
 bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute module, 45
 module, 36
 bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute module, 45
 bundesliga_tippspiel.test.routes.RouteTestBundesliga module, 46
 module, 36
 bundesliga_tippspiel.test.routes.RouteTestBundesliga module, 46
 bundesliga_tippspiel.test.routes.TestAboutRoute module, 46
 module, 36
 bundesliga_tippspiel.test.routes.TestAboutRoute module, 46
 bundesliga_tippspiel.test.routes.TestBetRoute module, 49
 module, 37
 bundesliga_tippspiel.test.routes.TestBetRoute module, 49
 bundesliga_tippspiel.test.routes.TestChangeSettingsRoute module, 47
 module, 37
 bundesliga_tippspiel.test.routes.TestChangeSettingsRoute module, 47
 bundesliga_tippspiel.test.routes.TestConfBundesliga module, 47
 module, 38
 bundesliga_tippspiel.test.routes.TestConfBundesliga module, 47
 bundesliga_tippspiel.test.routes.TestDeleteRoute module, 48
 module, 38
 bundesliga_tippspiel.test.routes.TestDeleteRoute module, 48
 bundesliga_tippspiel.test.routes.TestEmailReminderRoute module, 48
 module, 38
 bundesliga_tippspiel.test.routes.TestEmailReminderRoute module, 48
 bundesliga_tippspiel.test.routes.TestForgotPasswordRoute module, 39
 module, 39
 bundesliga_tippspiel.test.routes.TestForgotPasswordRoute module, 39
 bundesliga_tippspiel.test.routes.TestIndexRoute module, 39
 module, 39
 bundesliga_tippspiel.test.routes.TestIndexRoute module, 39
 bundesliga_tippspiel.test.routes.TestLeaderboardRoute module, 40
 module, 40
 bundesliga_tippspiel.test.routes.TestLeaderboardRoute module, 40
 bundesliga_tippspiel.test.routes.TestLoginRoute module, 40
 module, 40
 bundesliga_tippspiel.test.routes.TestLoginRoute module, 40
 bundesliga_tippspiel.test.routes.TestMatchRoute module, 40
 module, 40
 bundesliga_tippspiel.test.routes.TestMatchRoute module, 40
 bundesliga_tippspiel.test.routes.TestPrivacyRoute module, 41
 module, 41
 bundesliga_tippspiel.test.routes.TestPrivacyRoute module, 41
 bundesliga_tippspiel.test.routes.TestProfileRoute module, 41
 module, 41
 bundesliga_tippspiel.test.routes.TestProfileRoute module, 41
 bundesliga_tippspiel.test.routes.TestRegisterRoute module, 42
 module, 42
 bundesliga_tippspiel.test.routes.TestRegisterRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonBetsRoute module, 42
 module, 42
 bundesliga_tippspiel.test.routes.TestSeasonBetsRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonPositionRoute module, 42
 module, 42
 bundesliga_tippspiel.test.routes.TestSeasonPositionRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute module, 43
 module, 43
 bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute module, 43
 bundesliga_tippspiel.test.routes.TestStatsRoute module, 43
 module, 43
 bundesliga_tippspiel.test.routes.TestStatsRoute module, 43
 bundesliga_tippspiel.test.routes.TestTeamRoute module, 44
 module, 44
 bundesliga_tippspiel.test.routes.TestTeamRoute module, 44
 bundesliga_tippspiel.test.routes.TestUserRoute module, 44
 module, 44
 bundesliga_tippspiel.test.routes.TestUserRoute module, 44
 bundesliga_tippspiel.test.TestConfig module, 46
 module, 46
 bundesliga_tippspiel.test.TestConfig module, 46
 bundesliga_tippspiel.test.TestFramework module, 46
 module, 46
 bundesliga_tippspiel.test.TestFramework module, 46
 bundesliga_tippspiel.test.utils module, 46
 module, 46
 bundesliga_tippspiel.test.utils module, 46
 bundesliga_tippspiel.test.utils.TestChartData module, 45
 module, 45
 bundesliga_tippspiel.test.utils.TestChartData module, 45
 bundesliga_tippspiel.test.utils.TestCrypto module, 45
 module, 45
 bundesliga_tippspiel.test.utils.TestCrypto module, 45
 bundesliga_tippspiel.test.utils.TestDb module, 45
 module, 45
 bundesliga_tippspiel.test.utils.TestDb module, 45
 bundesliga_tippspiel.test.utils.TestJson module, 46
 module, 46
 bundesliga_tippspiel.test.utils.TestJson module, 46
 bundesliga_tippspiel.test.utils.TestStats module, 46
 module, 46
 bundesliga_tippspiel.test.utils.TestStats module, 46
 bundesliga_tippspiel.test.utils module, 49
 module, 49
 bundesliga_tippspiel.test.utils module, 49
 bundesliga_tippspiel.test.utils.chart_data module, 47
 module, 47
 bundesliga_tippspiel.test.utils.chart_data module, 47
 bundesliga_tippspiel.test.utils.db module, 47
 module, 47
 bundesliga_tippspiel.test.utils.db module, 47
 bundesliga_tippspiel.test.utils.routes module, 48
 module, 48
 bundesliga_tippspiel.test.utils.routes module, 48
 bundesliga_tippspiel.test.utils.stats module, 48
 module, 48
 bundesliga_tippspiel.test.utils.stats module, 48

C

bundesliga_tippspiel.test.routes.TestIndexRoute module, 39
 bundesliga_tippspiel.test.routes.TestIndexRoute module, 39
 bundesliga_tippspiel.test.routes.TestLeaderboardRoute module, 40
 bundesliga_tippspiel.test.routes.TestLeaderboardRoute module, 40
 bundesliga_tippspiel.test.routes.TestLoginRoute module, 40
 bundesliga_tippspiel.test.routes.TestLoginRoute module, 40
 bundesliga_tippspiel.test.routes.TestMatchRoute module, 40
 bundesliga_tippspiel.test.routes.TestMatchRoute module, 40
 bundesliga_tippspiel.test.routes.TestPrivacyRoute module, 41
 bundesliga_tippspiel.test.routes.TestPrivacyRoute module, 41
 bundesliga_tippspiel.test.routes.TestProfileRoute module, 41
 bundesliga_tippspiel.test.routes.TestProfileRoute module, 41
 bundesliga_tippspiel.test.routes.TestRegisterRoute module, 42
 bundesliga_tippspiel.test.routes.TestRegisterRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonBetsRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonBetsRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonPositionRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonPositionRoute module, 42
 bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute module, 43
 bundesliga_tippspiel.test.routes.TestSeasonTeamBetsRoute module, 43
 bundesliga_tippspiel.test.routes.TestStatsRoute module, 43
 bundesliga_tippspiel.test.routes.TestStatsRoute module, 43
 bundesliga_tippspiel.test.routes.TestTeamRoute module, 44
 bundesliga_tippspiel.test.routes.TestTeamRoute module, 44
 bundesliga_tippspiel.test.routes.TestUserRoute module, 44
 bundesliga_tippspiel.test.routes.TestUserRoute module, 44
 bundesliga_tippspiel.test.TestConfig module, 46
 bundesliga_tippspiel.test.TestFramework module, 46
 bundesliga_tippspiel.test.utils module, 46
 bundesliga_tippspiel.test.utils.TestChartData module, 45
 bundesliga_tippspiel.test.utils.TestCrypto module, 45
 bundesliga_tippspiel.test.utils.TestDb module, 45
 bundesliga_tippspiel.test.utils.TestJson module, 46
 bundesliga_tippspiel.test.utils.TestStats module, 46
 ChangeSettingsAction (class in bundesliga_tippspiel.actions.ChangeSettingsAction),
 ChatMessage (class in bundesliga_tippspiel.db.user_generated.ChatMessage),
 children (bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage attribute), 16
 Config (class in bundesliga_tippspiel.Config), 49
 create_participation_ranking() (in module bundesliga_tippspiel.utils.stats), 48
 defined_or_filters() (bundesliga_tippspiel.actions.Action.Action static method), 3

create_point_average_ranking() (in module *bundesliga_tippspiel.utils.stats*), 48
 creation_time (in module *bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17
 current_score() (in module *bundesliga_tippspiel.db.match_data.Match.Match* property), 12

D

define_blueprint() (in module *bundesliga_tippspiel.routes.api.getters*), 20
 define_blueprint() (in module *bundesliga_tippspiel.routes.api.putters*), 20
 define_blueprint() (in module *bundesliga_tippspiel.routes.betting*), 20
 define_blueprint() (in module *bundesliga_tippspiel.routes.chat*), 21
 define_blueprint() (in module *bundesliga_tippspiel.routes.information*), 21
 define_blueprint() (in module *bundesliga_tippspiel.routes.reminder*), 21
 define_blueprint() (in module *bundesliga_tippspiel.routes.settings*), 21
 delete() (in module *bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* method), 17
 deleted (in module *bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17
 display_bots (in module *bundesliga_tippspiel.db.settings.DisplayBotsSettings.DisplayBotsSettings* attribute), 14
 DisplayBotsSettings (class in module *bundesliga_tippspiel.db.settings.DisplayBotsSettings*), 14

E

edit() (in module *bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* method), 17
 edited (in module *bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17
 EMAIL (in module *bundesliga_tippspiel.enums.ReminderType* attribute), 50
 email_exists() (in module *bundesliga_tippspiel.utils.db*), 47
 environment_variables() (in module *bundesliga_tippspiel.Config.Config* class method), 49
 evaluate() (in module *bundesliga_tippspiel.db.user_generated.Bet.Bet* method), 16
 event_type (in module *bundesliga_tippspiel.db.SeasonEvent.SeasonEvent* attribute), 19
 execute() (in module *bundesliga_tippspiel.actions.Action.Action* method), 3
 execute_getter() (in module *bundesliga_tippspiel.routes.api.getters*), 20
 execute_with_redirects() (in module *bundesliga_tippspiel.actions.Action.Action* method), 3
 executed (in module *bundesliga_tippspiel.db.SeasonEvent.SeasonEvent* attribute), 19

F

finished (in module *bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 flash() (in module *bundesliga_tippspiel.exceptions.ActionException* method), 50
 from_dict() (in module *bundesliga_tippspiel.actions.Action.Action* class method), 3
 from_site_request() (in module *bundesliga_tippspiel.actions.Action.Action* class method), 3
 ft_score() (in module *bundesliga_tippspiel.db.match_data.Match.Match* property), 12

G

generate_action() (in module *bundesliga_tippspiel.test.actions.TestGetReminderSettingsAction.TestGetReminderSettingsAction* method), 23
 generate_action() (in module *bundesliga_tippspiel.test.actions.TestLeaderboardAction.TestLeaderboardAction* method), 24
 generate_action() (in module *bundesliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction* method), 24
 generate_action() (in module *bundesliga_tippspiel.test.actions.TestSetReminderSettingsAction.TestSetReminderSettingsAction* method), 25
 generate_leaderboard_data() (in module *bundesliga_tippspiel.utils.chart_data*), 47
 generate_points_distributions() (in module *bundesliga_tippspiel.utils.stats*), 48
 generate_team_points_table() (in module *bundesliga_tippspiel.utils.stats*), 48
 get_current_matchday() (in module *bundesliga_tippspiel.actions.Action.Action* static method), 3
 get_due_matches() (in module *bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* method), 15
 get_matchday_total_pill_class() (in module *bundesliga_tippspiel.jinja_extras*), 50
 get_pill_class() (in module *bundesliga_tippspiel.jinja_extras*), 50
 get_team_data() (in module *bundesliga_tippspiel.background.match_data*), 9

get_team_points_data() (in module *bundesliga_tippspiel.utils.stats*), 48
 get_text() (*bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17
 get_total_points_per_team() (in module *bundesliga_tippspiel.utils.stats*), 49
 GetAction (class in *bundesliga_tippspiel.actions.Action*), 4
 GetBetAction (class in *bundesliga_tippspiel.actions.GetBetAction*), 4
 GetGoalAction (class in *bundesliga_tippspiel.actions.GetGoalAction*), 5
 GetMatchAction (class in *bundesliga_tippspiel.actions.GetMatchAction*), 6
 GetPlayerAction (class in *bundesliga_tippspiel.actions.GetPlayerAction*), 6
 GetReminderSettingsAction (class in *bundesliga_tippspiel.actions.GetReminderSettingsAction*), 7
 GetTeamAction (class in *bundesliga_tippspiel.actions.GetTeamAction*), 7
 Goal (class in *bundesliga_tippspiel.db.match_data.Goal*), 11
 goals (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 goals (*bundesliga_tippspiel.db.match_data.Player.Player* attribute), 13
H
 handle_id_fetch() (*bundesliga_tippspiel.actions.Action.Action* static method), 3
 handle_midseason_reminder() (in module *bundesliga_tippspiel.background.season_events*), 10
 handle_postseason_wrapup() (in module *bundesliga_tippspiel.background.season_events*), 10
 handle_preseason_reminder() (in module *bundesliga_tippspiel.background.season_events*), 10
 handle_season_events() (in module *bundesliga_tippspiel.background.season_events*), 10
 home_current_score (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 home_ft_score (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 home_ht_score (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 home_score (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11
 home_score (*bundesliga_tippspiel.db.user_generated.Bet.Bet* attribute), 16
 home_team (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 home_team_id (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 ht_score() (*bundesliga_tippspiel.db.match_data.Match.Match* property), 12
I
 icon_png (*bundesliga_tippspiel.db.match_data.Team.Team* attribute), 14
 icon_svg (*bundesliga_tippspiel.db.match_data.Team.Team* attribute), 14
 id (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11
 id (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12
 id (*bundesliga_tippspiel.db.match_data.Player.Player* attribute), 13
 id (*bundesliga_tippspiel.db.match_data.Team.Team* attribute), 14
 id (*bundesliga_tippspiel.db.SeasonEvent.SeasonEvent* attribute), 19
 id (*bundesliga_tippspiel.db.settings.DisplayBotsSettings.DisplayBotsSettings* attribute), 14
 id (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* attribute), 15
 id (*bundesliga_tippspiel.db.user_generated.Bet.Bet* attribute), 16
 id (*bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17
 id (*bundesliga_tippspiel.db.user_generated.SeasonPositionBet.SeasonPositionBet* attribute), 17
 id (*bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet* attribute), 18
 id (*bundesliga_tippspiel.db.user_generated.SeasonWinner.SeasonWinner* attribute), 19
J
 jinja_extras() (in module *bundesliga_tippspiel.jinja_extras*), 50
K
 keyword() (*bundesliga_tippspiel.test.routes.api.TestGetBetApiRoute.TestGetBetApiRoute* property), 34

keyword() (*bundesliga_tippspiel.test.routes.api.TestGetGoalApiResponse* (attribute), 13
bundesliga_tippspiel.db.match_data.Match.Match
property), 34

keyword() (*bundesliga_tippspiel.test.routes.api.TestGetMatchApiResponse* (attribute), 13
bundesliga_tippspiel.db.SeasonEvent.SeasonEventType
property), 35

keyword() (*bundesliga_tippspiel.test.routes.api.TestGetPlayerApiResponse* (attribute), 13
bundesliga_tippspiel.db.match_data.Goal.Goal
property), 35

keyword() (*bundesliga_tippspiel.test.routes.api.TestGetTeamApiResponse* (attribute), 13
bundesliga_tippspiel.db.match_data.Match.Match
property), 35

kickoff (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12

kickoff_datetime() (*bundesliga_tippspiel.db.match_data.Match.Match* attribute), 12

kickoff_et (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

models (*in module bundesliga_tippspiel.db*), 20

module

L

last_edit (*bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17

last_reminder (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* attribute), 15

last_reminder_datetime() (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* attribute), 15

LeaderboardAction (*class in bundesliga_tippspiel.actions.LeaderboardAction*), 7

LEAST_GOALS_CONCEDED (*bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBetType* attribute), 18

LEAST_GOALS_SCORED (*bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBetType* attribute), 18

load_leaderboard_history() (*in module bundesliga_tippspiel.utils.chart_data*), 47

load_season_events() (*in module bundesliga_tippspiel.background.season_events*), 10

LoadSettingsAction (*class in bundesliga_tippspiel.actions.LoadSettingsAction*), 8

M

main() (*in module bundesliga_tippspiel.main*), 51

match (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

match (*bundesliga_tippspiel.db.user_generated.Bet.Bet* attribute), 16

Match (*class in bundesliga_tippspiel.db.match_data.Match*), 12

match_id (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

match_id (*bundesliga_tippspiel.db.user_generated.Bet.Bet* attribute), 16

bundesliga_tippspiel, 51

bundesliga_tippspiel.actions, 9

bundesliga_tippspiel.actions.Action, 3

bundesliga_tippspiel.actions.ChangeSettingsAction, 4

bundesliga_tippspiel.actions.GetBetAction, 4

bundesliga_tippspiel.actions.GetGoalAction, 5

bundesliga_tippspiel.actions.GetMatchAction, 6

bundesliga_tippspiel.actions.GetPlayerAction, 6

bundesliga_tippspiel.actions.GetReminderSettingAction, 7

bundesliga_tippspiel.actions.GetTeamAction, 7

bundesliga_tippspiel.actions.LeaderboardAction, 7

bundesliga_tippspiel.actions.LoadSettingsAction, 8

bundesliga_tippspiel.actions.PlaceBetsAction, 8

bundesliga_tippspiel.actions.SetReminderSettingAction, 9

bundesliga_tippspiel.background, 11

bundesliga_tippspiel.background.match_data, 9

bundesliga_tippspiel.background.reminders, 10

bundesliga_tippspiel.background.season_events, 10

bundesliga_tippspiel.Config, 49

bundesliga_tippspiel.db, 20

bundesliga_tippspiel.db.match_data, 14

bundesliga_tippspiel.db.match_data.Goal, 11

bundesliga_tippspiel.db.match_data.Match, 11

12	21
bundesliga_tippspiel.db.match_data.Player	bundesliga_tippspiel.test.actions.TestGetBetAct
13	22
bundesliga_tippspiel.db.match_data.Team	bundesliga_tippspiel.test.actions.TestGetGoalAc
13	22
bundesliga_tippspiel.db.SeasonEvent,	bundesliga_tippspiel.test.actions.TestGetMatchA
19	22
bundesliga_tippspiel.db.settings, 16	bundesliga_tippspiel.test.actions.TestGetPlayer
bundesliga_tippspiel.db.settings.DisplayBot	33
14	bundesliga_tippspiel.test.actions.TestGetRemind
bundesliga_tippspiel.db.settings.ReminderSet	28
15	bundesliga_tippspiel.test.actions.TestGetTeamAc
bundesliga_tippspiel.db.user_generated,	23
19	bundesliga_tippspiel.test.actions.TestLeaderboa
bundesliga_tippspiel.db.user_generated.Bet,	24
16	bundesliga_tippspiel.test.actions.TestPlaceBets
bundesliga_tippspiel.db.user_generated.Chat	14
16	bundesliga_tippspiel.test.actions.TestSetRemind
bundesliga_tippspiel.db.user_generated.Season	25
17	bundesliga_tippspiel.test.background,
bundesliga_tippspiel.db.user_generated.Season	26
18	bundesliga_tippspiel.test.background.TestMatchD
bundesliga_tippspiel.db.user_generated.Season	25
19	bundesliga_tippspiel.test.background.TestRemind
bundesliga_tippspiel.enums, 50	25
bundesliga_tippspiel.exceptions, 50	bundesliga_tippspiel.test.background.TestSeason
bundesliga_tippspiel.jinja_extras,	26
50	bundesliga_tippspiel.test.models, 33
bundesliga_tippspiel.main, 51	bundesliga_tippspiel.test.models.match_data,
bundesliga_tippspiel.routes, 21	29
bundesliga_tippspiel.routes.api, 20	bundesliga_tippspiel.test.models.match_data.Tes
bundesliga_tippspiel.routes.api.getters,	26
20	bundesliga_tippspiel.test.models.match_data.Tes
bundesliga_tippspiel.routes.api.putters,	27
20	bundesliga_tippspiel.test.models.match_data.Tes
bundesliga_tippspiel.routes.betting,	28
20	bundesliga_tippspiel.test.models.match_data.Tes
bundesliga_tippspiel.routes.chat, 21	28
bundesliga_tippspiel.routes.information,	bundesliga_tippspiel.test.models.ModelTestFrame
21	32
bundesliga_tippspiel.routes.reminder,	bundesliga_tippspiel.test.models.settings,
21	30
bundesliga_tippspiel.routes.settings,	bundesliga_tippspiel.test.models.settings.TestR
21	29
bundesliga_tippspiel.template_extras,	bundesliga_tippspiel.test.models.TestMatchData,
51	32
bundesliga_tippspiel.test, 46	bundesliga_tippspiel.test.models.TestSeasonEven
bundesliga_tippspiel.test.actions,	33
25	bundesliga_tippspiel.test.models.user_generated
bundesliga_tippspiel.test.actions.ActionTest	32
21	bundesliga_tippspiel.test.models.user_generated
bundesliga_tippspiel.test.actions.GetAction	30
21	bundesliga_tippspiel.test.models.user_generated
bundesliga_tippspiel.test.actions.TestAction	30

bundesliga_tippspiel.test.models.user_generated.TestSeasonPositionBet, 40
 31 bundesliga_tippspiel.test.routes.TestPrivacyRoute, 40
 bundesliga_tippspiel.test.models.user_generated.TestSeasonTeamBet, 41
 31 bundesliga_tippspiel.test.routes.TestProfileRoute, 41
 bundesliga_tippspiel.test.models.user_generated.TestSeasonWinner, 41
 32 bundesliga_tippspiel.test.routes.TestRegisterRoute, 41
 bundesliga_tippspiel.test.routes, 45 42
 bundesliga_tippspiel.test.routes.api, bundesliga_tippspiel.test.routes.TestSeasonBets, 36 42
 bundesliga_tippspiel.test.routes.api.ApiRouteTestFramework, bundesliga_tippspiel.test.routes.TestSeasonPositionBet, 33 42
 bundesliga_tippspiel.test.routes.api.GetBundesligaRouteTestFramework, bundesliga_tippspiel.test.routes.TestSeasonTeamBet, 33 43
 bundesliga_tippspiel.test.routes.api.TestBundesligaRouteTippSpiel, bundesliga_tippspiel.test.routes.TestStatsRoute, 33 43
 bundesliga_tippspiel.test.routes.api.TestBundesligaApiResponse, bundesliga_tippspiel.test.routes.TestTeamRoute, 34 44
 bundesliga_tippspiel.test.routes.api.TestBundesligaRouteTippSpiel, bundesliga_tippspiel.test.routes.TestUserRoute, 34 44
 bundesliga_tippspiel.test.routes.api.TestBundesligaApiResponse, bundesliga_tippspiel.test.TestConfig, 34 46
 bundesliga_tippspiel.test.routes.api.TestBundesligaApiResponse, bundesliga_tippspiel.test.TestFramework, 35 46
 bundesliga_tippspiel.test.routes.api.TestBundesligaApiResponse, bundesliga_tippspiel.test.utils, 46
 35 bundesliga_tippspiel.test.utils.TestChartData, 45
 bundesliga_tippspiel.test.routes.api.TestGetTeamApiResponse, bundesliga_tippspiel.test.utils.TestCrypto, 35 45
 bundesliga_tippspiel.test.routes.api.TestLeaderboardApiResponse, bundesliga_tippspiel.test.utils.TestDb, 36 45
 bundesliga_tippspiel.test.routes.api.TestPutSetApiResponse, bundesliga_tippspiel.test.utils.TestJson, 36 45
 bundesliga_tippspiel.test.routes.RouteTestFramework, bundesliga_tippspiel.test.utils.TestStats, 36 46
 bundesliga_tippspiel.test.routes.TestAboutRoute, bundesliga_tippspiel.test.utils, 46 49
 bundesliga_tippspiel.test.routes.TestBundesligaRouteTippSpiel, bundesliga_tippspiel.test.utils.chart_data, 37 47
 bundesliga_tippspiel.test.routes.TestBundesligaRouteTippSpiel, bundesliga_tippspiel.test.utils.db, 47 47
 37 bundesliga_tippspiel.test.utils.routes, 48
 bundesliga_tippspiel.test.routes.TestConfirmRoute, bundesliga_tippspiel.test.utils.stats, 48 48
 bundesliga_tippspiel.test.routes.TestMostGoalsConceded, (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 38 attribute), 18
 bundesliga_tippspiel.test.routes.TestEmailRoute, (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 38 MOST_GOALS_SCORED (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 bundesliga_tippspiel.test.routes.TestForgotPasswordRoute, (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 39 attribute), 18
 bundesliga_tippspiel.test.routes.TestMostExpensiveGoals, (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 39 attribute), 18
 bundesliga_tippspiel.test.routes.TestLeaderboardRoute, (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonTeamBet), 18
 40 attribute), 18
 bundesliga_tippspiel.test.routes.TestLoginRoute, name (bundesliga_tippspiel.db.match_data.Player.Player), 40
 bundesliga_tippspiel.test.routes.TestMatchRoute, 13

name (*bundesliga_tippspiel.db.match_data.Team.Team* attribute), 14

method), 3
 profile_extras() (in module *bundesliga_tippspiel.template_extras*), 51

O

OPENLIGADB_LEAGUE (*bundesliga_tippspiel.Config.Config* attribute), 49

OPENLIGADB_SEASON (*bundesliga_tippspiel.Config.Config* attribute), 49

own_goal (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

P

parent (*bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17

parent_id (*bundesliga_tippspiel.db.user_generated.ChatMessage.ChatMessage* attribute), 17

parse_goal() (in module *bundesliga_tippspiel.background.match_data*), 9

parse_match() (in module *bundesliga_tippspiel.background.match_data*), 9

parse_player() (in module *bundesliga_tippspiel.background.match_data*), 9

parse_team() (in module *bundesliga_tippspiel.background.match_data*), 9

penalty (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

PlaceBetsAction (class in *bundesliga_tippspiel.actions.PlaceBetsAction*), 8

player (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

Player (class in *bundesliga_tippspiel.db.match_data.Player*), 13

player_id (*bundesliga_tippspiel.db.match_data.Goal.Goal* attribute), 11

players (*bundesliga_tippspiel.db.match_data.Team.Team* attribute), 14

position (*bundesliga_tippspiel.db.user_generated.SeasonPositionBet.SeasonPositionBet* attribute), 17

POST_SEASON_WRAPUP (*bundesliga_tippspiel.db.SeasonEvent.SeasonEventType* attribute), 19

PRE_SEASON_MAIL (*bundesliga_tippspiel.db.SeasonEvent.SeasonEventType* attribute), 20

prepare_get_response() (*bundesliga_tippspiel.actions.Action.Action*

R

reminder_time (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* attribute), 15

reminder_time_delta() (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* property), 15

reminder_type (*bundesliga_tippspiel.db.settings.ReminderSettings.ReminderSettings* attribute), 15

ReminderSettings (class in *bundesliga_tippspiel.db.settings.ReminderSettings*), 15

ReminderType (class in *bundesliga_tippspiel.enums*), 50

resolve_and_check_matchday() (*bundesliga_tippspiel.actions.Action.Action* static method), 4

root_path (in module *bundesliga_tippspiel*), 51

route_info() (*bundesliga_tippspiel.test.routes.api.TestApiKeyApiRoute.TestApiKeyA* property), 33

route_info() (*bundesliga_tippspiel.test.routes.api.TestAuthorizeApiRoute.TestAutho* property), 34

route_info() (*bundesliga_tippspiel.test.routes.api.TestLeaderboardApiRoute.TestLe* property), 36

route_info() (*bundesliga_tippspiel.test.routes.api.TestPutBetApiRoute.TestPutBetA* property), 36

route_info() (*bundesliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute* property), 36

route_info() (*bundesliga_tippspiel.test.routes.TestBetsRoute.TestBetsRoute* property), 37

route_info() (*bundesliga_tippspiel.test.routes.TestChangePasswordRoute.TestChan* property), 37

route_info() (*bundesliga_tippspiel.test.routes.TestConfirmRoute.TestConfirmRoute* property), 38

route_info() (*bundesliga_tippspiel.test.routes.TestDeleteUserRoute.TestDeleteUser* property), 38

route_info() (*bundesliga_tippspiel.test.routes.TestEmailReminderRoute.TestSetRem* property), 38

route_info() (bun- desliga_tippspiel.test.routes.api.TestGetPlayerApiRoute.TestGetP
desliga_tippspiel.test.routes.TestForgotPasswordRoute.TestForgotPasswordRoute
property), 39 sample_filters() (bun-
route_info() (bun- desliga_tippspiel.test.routes.api.TestGetTeamApiRoute.TestGetTea
desliga_tippspiel.test.routes.TestIndexRoute.TestIndexRoute property), 35
property), 39 season (bundesliga_tippspiel.db.match_data.Match.Match
route_info() (bun- attribute), 13
desliga_tippspiel.test.routes.TestLeaderboardRoute.TestLeaderboardRoute
property), 40 attribute), 19
route_info() (bun- season (bundesliga_tippspiel.db.user_generated.SeasonPositionBet.Seaso
desliga_tippspiel.test.routes.TestLoginRoute.TestLoginRoute attribute), 17
property), 40 season (bundesliga_tippspiel.db.user_generated.SeasonTeamBet.SeasonT
route_info() (bun- attribute), 18
desliga_tippspiel.test.routes.TestMatchRoute.TestMatchRoute (bundesliga_tippspiel.db.user_generated.SeasonWinner.SeasonWin
property), 40 attribute), 19
route_info() (bun- season() (bundesliga_tippspiel.Config.Config class
desliga_tippspiel.test.routes.TestPrivacyRoute.TestPrivacyRoute method), 49
property), 41 season_position_bets (bun-
route_info() (bun- desliga_tippspiel.db.match_data.Team.Team
desliga_tippspiel.test.routes.TestProfileRoute.TestProfileRoute attribute), 14
property), 41 season_string() (bun-
route_info() (bun- desliga_tippspiel.Config.Config class method),
desliga_tippspiel.test.routes.TestRegisterRoute.TestRegisterRoute
property), 42 season_string() (bun-
route_info() (bun- desliga_tippspiel.db.user_generated.SeasonWinner.SeasonWinner
desliga_tippspiel.test.routes.TestSeasonBetsRoute.TestSeasonBetsRoute attribute), 19
property), 42 season_team_bets (bun-
route_info() (bun- desliga_tippspiel.db.match_data.Team.Team
desliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.TestSeasonPositionBetsRoute
property), 42 SeasonEvent (class in bun-
route_info() (bun- desliga_tippspiel.db.SeasonEvent), 19
desliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.TestSeasonTeamBetsRoute (class in bun-
property), 43 desliga_tippspiel.db.SeasonEvent), 19
route_info() (bun- SeasonPositionBet (class in bun-
desliga_tippspiel.test.routes.TestStatsRoute.TestStatsRoute desliga_tippspiel.db.user_generated.SeasonPositionBet),
property), 43 17
route_info() (bun- SeasonTeamBet (class in bun-
desliga_tippspiel.test.routes.TestTeamRoute.TestTeamRoute desliga_tippspiel.db.user_generated.SeasonTeamBet),
property), 44 18
route_info() (bun- SeasonTeamBetType (class in bun-
desliga_tippspiel.test.routes.TestUserRoute.TestUserRoute desliga_tippspiel.db.user_generated.SeasonTeamBet),
property), 44 18
SeasonWinner (class in bun-
desliga_tippspiel.db.user_generated.SeasonWinner),
19
S
sample_filters() (bun- send_duplicate_reminders() (in module bun-
desliga_tippspiel.test.routes.api.TestGetBetApiRoute.TestGetBetApiRoute desliga_tippspiel.background.reminders),
property), 34 10
sample_filters() (bun- desliga_tippspiel.test.routes.api.TestGetGoalApiResponse.TestGetGoalApiResponse (bun-
property), 34 desliga_tippspiel.db.settings.ReminderSettings.ReminderSettings
sample_filters() (bun- method), 15
desliga_tippspiel.test.routes.api.TestGetMatchApiResponse.TestGetMatchApiResponse send_reminder_message() (bun-
property), 35 desliga_tippspiel.db.settings.ReminderSettings.ReminderSettings
sample_filters() (bun- method), 15

test_bets_out_of_bounds () (bun- test_due () (bundesliga_tippspiel.test.models.settings.TestReminderSettings
 desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction
 method), 24 test_due_when_bets_placed () (bun-
 test_cascades () (bun- desliga_tippspiel.test.models.settings.TestReminderSettings.TestR
 desliga_tippspiel.test.models.match_data.TestGoal.TestGoal
 method), 29
 method), 26 test_evaluating_bet_result () (bun-
 test_cascades () (bun- desliga_tippspiel.test.models.user_generated.TestBet.TestBet
 desliga_tippspiel.test.models.match_data.TestMatch.TestMatch
 method), 30
 method), 27 test_evaluating_unfinished_games () (bun-
 test_cascades () (bun- desliga_tippspiel.test.models.user_generated.TestBet.TestBet
 desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer
 method), 30
 method), 28 test_executing () (bun-
 test_cascades () (bun- desliga_tippspiel.test.background.TestSeasonEvents.TestSeasonEv
 desliga_tippspiel.test.models.match_data.TestTeam.TestTeam
 method), 26
 method), 28 test_expired_api_key () (bun-
 test_cascades () (bun- desliga_tippspiel.test.routes.api.TestAuthorizeApiRoute.TestAutho
 desliga_tippspiel.test.models.settings.TestReminderSettings.TestReminderSettings
 method), 29 test_fetching () (bun-
 test_cascades () (bun- desliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction
 desliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent
 method), 22
 method), 33 test_fetching () (bun-
 test_cascades () (bun- desliga_tippspiel.test.actions.TestGetGoalAction.TestGetGoalAct
 desliga_tippspiel.test.models.user_generated.TestBet.TestBet
 method), 22
 method), 30 test_fetching () (bun-
 test_cascades () (bun- desliga_tippspiel.test.actions.TestGetMatchAction.TestGetMatchA
 desliga_tippspiel.test.models.user_generated.TestSeasonPositionBet.TestSeasonPositionBet
 method), 31 test_fetching () (bun-
 test_cascades () (bun- desliga_tippspiel.test.actions.TestGetPlayerAction.TestGetPlayer.
 desliga_tippspiel.test.models.user_generated.TestSeasonTeamBet.TestSeasonTeamBet
 method), 31 test_fetching () (bun-
 test_cascades () (bun- desliga_tippspiel.test.actions.TestGetTeamAction.TestGetTeamAc
 desliga_tippspiel.test.models.user_generated.TestSeasonWinner.TestSeasonWinner
 method), 32 test_fetching_by_match () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction
 desliga_tippspiel.test.models.match_data.TestGoal.TestGoal
 method), 22
 method), 26 test_from_dict () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.actions.TestGetReminderSettingsAction.Test
 desliga_tippspiel.test.models.match_data.TestMatch.TestMatch
 method), 23
 method), 27 test_generating_leaderboard_data () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.utils.TestChartData.TestChartData
 desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer
 method), 45
 method), 28 test_generating_team_points_data () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.utils.TestStats.TestStats
 desliga_tippspiel.test.models.match_data.TestTeam.TestTeam
 method), 46
 method), 28 test_getting_current_matchday () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.actions.TestAction.ActionTest
 desliga_tippspiel.test.models.settings.TestReminderSettings.TestReminderSettings
 method), 29 test_getting_specific_matchday () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.routes.TestBetsRoute.TestBetsRoute
 desliga_tippspiel.test.models.user_generated.TestBet.TestBet
 method), 37
 method), 30 test_hashing () (bun-
 test_deleting_from_db () (bun- desliga_tippspiel.test.utils.TestCrypto.TestCrypto
 desliga_tippspiel.test.models.user_generated.TestChatMessage.TestChatMessage
 method), 31 test_hashing_strings () (bun-

<i>desliga_tippspiel.test.utils.TestCrypto.TestCrypto</i> method), 45	<i>desliga_tippspiel.test.actions.TestLeaderboardAction.TestLeaderboard</i> method), 24
test_icon_urls() (bun- <i>desliga_tippspiel.test.background.TestMatchDataUpdate.TestMatchDataUpdate</i> method), 25	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute</i> method), 37
test_initializing() (bun- <i>desliga_tippspiel.test.background.TestSeasonEvents.TestSeasonEvents</i> method), 26	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestBetsRoute.TestBetsRoute</i> method), 37
test_invalid_bets() (bun- <i>desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction</i> method), 24	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestIndexRoute.TestIndexRoute</i> method), 39
test_json_representation() (bun- <i>desliga_tippspiel.test.models.match_data.TestGoal.TestGoal</i> method), 27	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestLeaderboardRoute.TestLeaderboard</i> method), 40
test_json_representation() (bun- <i>desliga_tippspiel.test.models.match_data.TestMatch.TestMatch</i> method), 27	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestPrivacyRoute.TestPrivacyRoute</i> method), 41
test_json_representation() (bun- <i>desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer</i> method), 28	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestProfileRoute.TestProfileRoute</i> method), 41
test_json_representation() (bun- <i>desliga_tippspiel.test.models.match_data.TestTeam.TestTeam</i> method), 28	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestSeasonBetsRoute.TestSeasonBets</i> method), 42
test_json_representation() (bun- <i>desliga_tippspiel.test.models.settings.TestReminderSettings.TestReminderSettings</i> method), 29	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.TestSeason</i> method), 43
test_json_representation() (bun- <i>desliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent</i> method), 33	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.TestSeason</i> method), 43
test_json_representation() (bun- <i>desliga_tippspiel.test.models.user_generated.TestBet.TestBet</i> method), 30	test_malformed_data() (bun- <i>desliga_tippspiel.test.routes.TestStatsRoute.TestStatsRoute</i> method), 43
test_json_representation() (bun- <i>desliga_tippspiel.test.models.user_generated.TestChatMessage.TestChatMessage</i> method), 31	test_midseason_reminder() (bun- <i>desliga_tippspiel.test.background.TestSeasonEvents.TestSeasonE</i> method), 26
test_json_representation() (bun- <i>desliga_tippspiel.test.models.user_generated.TestSeasonPositionBet.TestSeasonPositionBet</i> method), 31	test_minute_representation() (bun- <i>desliga_tippspiel.test.models.match_data.TestMatch.TestMatch</i> method), 27
test_json_representation() (bun- <i>desliga_tippspiel.test.models.user_generated.TestSeasonTeamBet.TestSeasonTeamBet</i> method), 31	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.match_data.TestGoal.TestGoal</i> method), 27
test_json_representation() (bun- <i>desliga_tippspiel.test.models.user_generated.TestSeasonWinLoss.TestSeasonWinLoss</i> method), 32	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.match_data.TestMatch.TestMatch</i> method), 27
test_jsonifying() (bun- <i>desliga_tippspiel.test.utils.TestJson.TestJson</i> method), 46	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer</i> method), 28
test_kickoff_datetime_conversion() (bun- <i>desliga_tippspiel.test.models.match_data.TestMatch.TestMatch</i> method), 27	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.match_data.TestTeam.TestTeam</i> method), 28
test_leaderboard() (bun- <i>desliga_tippspiel.test.actions.TestLeaderboardAction.TestLeaderboardAction</i> method), 24	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.settings.TestReminderSettings.TestR</i> method), 29
test_leaderboard_for_matchday() (bun- <i>desliga_tippspiel.test.models.settings.TestReminderSettings.TestR</i> method), 29	test_missing_column_data() (bun- <i>desliga_tippspiel.test.models.settings.TestReminderSettings.TestR</i> method), 29

desliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent
method), 33

desliga_tippspiel.test.models.settings.TestReminderSettings.TestR
method), 29

test_missing_column_data() (*bun-* *test_retrieving_from_db()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestBet.TestBet
method), 30

test_missing_column_data() (*bun-* *test_retrieving_from_db()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestChatMessage.TestChatMessage
method), 31

test_missing_column_data() (*bun-* *test_salt()* (*bundesliga_tippspiel.test.utils.TestCrypto.TestCrypto*
desliga_tippspiel.test.models.user_generated.TestSeasonPositionBet
method), 31

test_missing_column_data() (*bun-* *test_score_representations()* (*bun-*
desliga_tippspiel.test.models.match_data.TestMatch.TestMatch
desliga_tippspiel.test.models.user_generated.TestSeasonTeamBet
method), 32

test_missing_column_data() (*bun-* *test_season_calculations()* (*bun-*
desliga_tippspiel.test.TestConfig.TestConfig
desliga_tippspiel.test.models.user_generated.TestSeasonWinner
method), 32

test_mixing_valid_and_invalid_bets() (*bun-* *test_season_string()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestSeasonWinner
method), 32

test_mixing_valid_and_invalid_bets() (*bundesliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction*
method), 24

test_model_relations() (*bun-* *test_setting_and_updating_reminder()*
desliga_tippspiel.test.models.TestMatchData.TestMatchData
method), 32

test_parsing_form_data() (*bun-* *test_setting_invalid_reminder_times()*
desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction
method), 24

test_placing_bets() (*bun-* *test_setting_updating_and_deleting_reminder()*
desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsAction
method), 24

test_populating_twice() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.background.TestMatchDataUpdate.TestMatchDataUpdate
method), 25

test_postseason_wrapup() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.background.TestSeasonEvents.TestSeasonEvents
method), 26

test_preseason_reminder() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.background.TestSeasonEvents.TestSeasonEvents
method), 26

test_rankings_before_season() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.utils.TestStats.TestStats
method), 46

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.match_data.TestGoal.TestGoal
method), 27

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent
method), 33

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.match_data.TestMatch.TestMatch
method), 27

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestBet.TestBet
method), 30

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer
method), 28

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestChatMessage.TestChatMessage
method), 31

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.match_data.TestTeam.TestTeam
method), 28

test_retrieving_from_db() (*bun-* *test_string_representation()* (*bun-*
desliga_tippspiel.test.models.user_generated.TestSeasonPositionBet
method), 31

<i>method</i>), 31	<i>method</i>), 41	
test_string_representation() (bun- desliga_tippspiel.test.models.user_generated.TestSeasonTeamBetsRoute.TestProfileRoute.TestProfileRoute <i>method</i>), 32	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestProfileRoute.TestProfileRoute <i>method</i>), 41
test_string_representation() (bun- desliga_tippspiel.test.models.user_generated.TestSeasonWinnerRoute.TestRegisterRoute.TestRegisterRoute <i>method</i>), 32	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestRegisterRoute.TestRegisterRoute <i>method</i>), 42
test_successful_call() (bun- desliga_tippspiel.test.routes.api.TestApiKeyApiRoute.TestApiKeyApiRoute <i>method</i>), 33	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestSeasonBetsRoute.TestSeasonBetsRoute <i>method</i>), 42
test_successful_call() (bun- desliga_tippspiel.test.routes.api.TestAuthorizeApiRoute.TestAuthorizeApiRoute <i>method</i>), 34	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.TestSeasonPositionBetsRoute <i>method</i>), 43
test_successful_call() (bun- desliga_tippspiel.test.routes.api.TestLeaderboardApiRoute.TestLeaderboardApiRoute <i>method</i>), 36	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.TestSeasonTeamBetsRoute <i>method</i>), 43
test_successful_call() (bun- desliga_tippspiel.test.routes.api.TestPutBetApiRoute.TestPutBetApiRoute <i>method</i>), 36	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestStatsRoute.TestStatsRoute <i>method</i>), 43
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestTeamRoute.TestTeamRoute <i>method</i>), 44
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestBetsRoute.TestBetsRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestUserRoute.TestUserRoute <i>method</i>), 44
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestChangePasswordRoute.TestChangePasswordRoute <i>method</i>), 37	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.match_data.TestGoal.TestGoal <i>method</i>), 27
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestConfirmRoute.TestConfirmRoute <i>method</i>), 38	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.match_data.TestMatch.TestMatch <i>method</i>), 27
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestDeleteUserRoute.TestDeleteUserRoute <i>method</i>), 38	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.match_data.TestPlayer.TestPlayer <i>method</i>), 28
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestEmailReminderRoute.TestEmailReminderRoute <i>method</i>), 38	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.match_data.TestTeam.TestTeam <i>method</i>), 29
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestForgotPasswordRoute.TestForgotPasswordRoute <i>method</i>), 39	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.settings.TestReminderSettings.TestReminderSettings <i>method</i>), 29
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestIndexRoute.TestIndexRoute <i>method</i>), 39	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.TestSeasonEvent.TestSeasonEvent <i>method</i>), 33
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestLeaderboardRoute.TestLeaderboardRoute <i>method</i>), 40	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.user_generated.TestBet.TestBet <i>method</i>), 30
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestLoginRoute.TestLoginRoute <i>method</i>), 40	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.user_generated.TestSeasonPositionBetsRoute.TestSeasonPositionBetsRoute <i>method</i>), 31
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestMatchRoute.TestMatchRoute <i>method</i>), 41	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.user_generated.TestSeasonTeamBetsRoute.TestSeasonTeamBetsRoute <i>method</i>), 32
test_successful_requests() (bun- desliga_tippspiel.test.routes.TestPrivacyRoute.TestPrivacyRoute <i>method</i>), 41	test_successful_requests() (bun- desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute <i>method</i>), 37	test_uniqueness() (bun- desliga_tippspiel.test.models.user_generated.TestSeasonWinnerRoute.TestSeasonWinnerRoute <i>method</i>), 32

method), 32	method), 42	
test_unsuccessful_call() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.api.TestApiKeyApiRoute.TestApiKeyApiRoute	desliga_tippspiel.test.routes.TestSeasonBetsRoute.TestSeasonBets	desliga_tippspiel.test.routes.TestSeasonBetsRoute.TestSeasonBets
method), 33	method), 42	method), 42
test_unsuccessful_call() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.api.TestAuthorizeApiRoute.TestAuthorizeApiRoute	desliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.TestSeason	desliga_tippspiel.test.routes.TestSeasonPositionBetsRoute.TestSeason
method), 34	method), 43	method), 43
test_unsuccessful_call() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.api.TestLeaderboardApiRoute.TestLeaderboardApiRoute	desliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.TestSeason	desliga_tippspiel.test.routes.TestSeasonTeamBetsRoute.TestSeason
method), 36	method), 43	method), 43
test_unsuccessful_call() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.api.TestPutBetApiRoute.TestPutBetApiRoute	desliga_tippspiel.test.routes.TestStatsRoute.TestStatsRoute	desliga_tippspiel.test.routes.TestStatsRoute
method), 36	method), 43	method), 43
test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.TestAboutRoute.TestAboutRoute	desliga_tippspiel.test.routes.TestTeamRoute.TestTeamRoute	desliga_tippspiel.test.routes.TestTeamRoute
method), 37	method), 44	method), 44
test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-	test_unsuccessful_requests() (bun-
desliga_tippspiel.test.routes.TestBetsRoute.TestBetsRoute	desliga_tippspiel.test.routes.TestUserRoute.TestUserRoute	desliga_tippspiel.test.routes.TestUserRoute
method), 37	method), 44	method), 44
test_unsuccessful_requests() (bun-	test_updating_bets() (bun-	test_updating_bets() (bun-
desliga_tippspiel.test.routes.TestChangePasswordRoute.TestChangePasswordRoute	desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsA	desliga_tippspiel.test.actions.TestPlaceBetsAction.TestPlaceBetsA
method), 37	method), 24	method), 24
test_unsuccessful_requests() (bun-	test_user_exists_functions() (bun-	test_user_exists_functions() (bun-
desliga_tippspiel.test.routes.TestConfirmRoute.TestConfirmRoute	desliga_tippspiel.test.utils.TestDb.TestDb	desliga_tippspiel.test.utils.TestDb.TestDb
method), 38	method), 45	method), 45
test_unsuccessful_requests() (bun-	test_using_chat() (bun-	test_using_chat() (bun-
desliga_tippspiel.test.routes.TestDeleteUserRoute.TestDeleteUserRoute	desliga_tippspiel.test.models.user_generated.TestChatMessage.Te	desliga_tippspiel.test.models.user_generated.TestChatMessage.Te
method), 38	method), 31	method), 31
test_unsuccessful_requests() (bun-	test_using_filter_and_id() (bun-	test_using_filter_and_id() (bun-
desliga_tippspiel.test.routes.TestEmailReminderRoute.TestEmailReminderRoute	desliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction	desliga_tippspiel.test.actions.TestGetBetAction.TestGetBetAction
method), 39	method), 22	method), 22
test_unsuccessful_requests() (bun-	test_using_filter_and_id() (bun-	test_using_filter_and_id() (bun-
desliga_tippspiel.test.routes.TestForgotPasswordRoute.TestForgotPasswordRoute	desliga_tippspiel.test.actions.TestGetGoalAction.TestGetGoalAct	desliga_tippspiel.test.actions.TestGetGoalAction.TestGetGoalAct
method), 39	method), 22	method), 22
test_unsuccessful_requests() (bun-	test_using_filter_and_id() (bun-	test_using_filter_and_id() (bun-
desliga_tippspiel.test.routes.TestIndexRoute.TestIndexRoute	desliga_tippspiel.test.actions.TestGetMatchAction.TestGetMatchA	desliga_tippspiel.test.actions.TestGetMatchAction.TestGetMatchA
method), 39	method), 22	method), 22
test_unsuccessful_requests() (bun-	test_using_filter_and_id() (bun-	test_using_filter_and_id() (bun-
desliga_tippspiel.test.routes.TestLeaderboardRoute.TestLeaderboardRoute	desliga_tippspiel.test.actions.TestGetPlayerAction.TestGetPlayer	desliga_tippspiel.test.actions.TestGetPlayerAction.TestGetPlayer
method), 40	method), 23	method), 23
test_unsuccessful_requests() (bun-	test_using_filter_and_id() (bun-	test_using_filter_and_id() (bun-
desliga_tippspiel.test.routes.TestLoginRoute.TestLoginRoute	desliga_tippspiel.test.actions.TestGetTeamAction.TestGetTeamAc	desliga_tippspiel.test.actions.TestGetTeamAction.TestGetTeamAc
method), 40	method), 23	method), 23
test_unsuccessful_requests() (bun-	test_verifying_with_invalid_hash() (bun-	test_verifying_with_invalid_hash() (bun-
desliga_tippspiel.test.routes.TestMatchRoute.TestMatchRoute	desliga_tippspiel.test.utils.TestCrypto.TestCrypto	desliga_tippspiel.test.utils.TestCrypto.TestCrypto
method), 41	method), 45	method), 45
test_unsuccessful_requests() (bun-	test_with_due_reminder() (bun-	test_with_due_reminder() (bun-
desliga_tippspiel.test.routes.TestPrivacyRoute.TestPrivacyRoute	desliga_tippspiel.test.background.TestReminders.TestReminders	desliga_tippspiel.test.background.TestReminders.TestReminders
method), 41	method), 25	method), 25
test_unsuccessful_requests() (bun-	test_with_non_due_reminder() (bun-	test_with_non_due_reminder() (bun-
desliga_tippspiel.test.routes.TestProfileRoute.TestProfileRoute	desliga_tippspiel.test.background.TestReminders.TestReminders	desliga_tippspiel.test.background.TestReminders.TestReminders
method), 41	method), 25	method), 25
test_unsuccessful_requests() (bun-	test_without_stored_reminders() (bun-	test_without_stored_reminders() (bun-
desliga_tippspiel.test.routes.TestRegisterRoute.TestRegisterRoute	desliga_tippspiel.test.background.TestReminders.TestReminders	desliga_tippspiel.test.background.TestReminders.TestReminders

<i>method</i>), 25	22
TestAboutRoute (class in bun- desliga_tippspiel.test.routes.TestAboutRoute), 36	TestGetMatchApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestGetMatchApiRoute), 35
TestApiKeyApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestApiKeyApiRoute), 33	TestGetPlayerAction (class in bun- desliga_tippspiel.test.actions.TestGetPlayerAction), 23
TestAuthorizeApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestAuthorizeApiRoute), 34	TestGetPlayerApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestGetPlayerApiRoute), 35
TestBet (class in bun- desliga_tippspiel.test.models.user_generated.TestBet), 30	TestGetReminderSettingsAction (class in bun- desliga_tippspiel.test.actions.TestGetReminderSettingsAction), 23
TestBetsRoute (class in bun- desliga_tippspiel.test.routes.TestBetsRoute), 37	TestGetTeamAction (class in bun- desliga_tippspiel.test.actions.TestGetTeamAction), 23
TestChangePasswordRoute (class in bun- desliga_tippspiel.test.routes.TestChangePasswordRoute), 37	TestGetTeamApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestGetTeamApiRoute), 35
TestChartData (class in bun- desliga_tippspiel.test.utils.TestChartData), 45	TestGoal (class in bun- desliga_tippspiel.test.models.match_data.TestGoal), 26
TestChatMessage (class in bun- desliga_tippspiel.test.models.user_generated.TestChatMessage), 30	TestIndexRoute (class in bun- desliga_tippspiel.test.routes.TestIndexRoute), 39
TestConfig (class in bun- desliga_tippspiel.test.TestConfig), 46	TestJson (class in bun- desliga_tippspiel.test.utils.TestJson), 46
TestConfirmRoute (class in bun- desliga_tippspiel.test.routes.TestConfirmRoute), 38	TestLeaderboardAction (class in bun- desliga_tippspiel.test.actions.TestLeaderboardAction), 24
TestCrypto (class in bun- desliga_tippspiel.test.utils.TestCrypto), 45	TestLeaderboardApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestLeaderboardApiRoute), 36
TestDb (class in bundesliga_tippspiel.test.utils.TestDb), 45	TestLeaderboardRoute (class in bun- desliga_tippspiel.test.routes.TestLeaderboardRoute), 40
TestDeleteUserRoute (class in bun- desliga_tippspiel.test.routes.TestDeleteUserRoute), 38	TestLoginRoute (class in bun- desliga_tippspiel.test.routes.TestLoginRoute), 40
TestForgotPasswordRoute (class in bun- desliga_tippspiel.test.routes.TestForgotPasswordRoute), 39	TestMatch (class in bun- desliga_tippspiel.test.models.match_data.TestMatch), 27
TestGetBetAction (class in bun- desliga_tippspiel.test.actions.TestGetBetAction), 22	TestMatchData (class in bun- desliga_tippspiel.test.models.TestMatchData), 32
TestGetBetApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestGetBetApiRoute), 34	TestMatchDataUpdate (class in bun- desliga_tippspiel.test.background.TestMatchDataUpdate), 25
TestGetGoalAction (class in bun- desliga_tippspiel.test.actions.TestGetGoalAction), 22	TestMatchRoute (class in bun- desliga_tippspiel.test.routes.TestMatchRoute), 40
TestGetGoalApiRoute (class in bun- desliga_tippspiel.test.routes.api.TestGetGoalApiRoute), 34	TestPlaceBetsAction (class in bun- desliga_tippspiel.test.actions.TestPlaceBetsAction), 24
TestGetMatchAction (class in bun- desliga_tippspiel.test.actions.TestGetMatchAction),	

method), 4
validate_data() (bun-
desliga_tippspiel.actions.ChangeSettingsAction.ChangeSettingsAction
method), 4
validate_data() (bun-
desliga_tippspiel.actions.GetBetAction.GetBetAction
method), 5
validate_data() (bun-
desliga_tippspiel.actions.GetGoalAction.GetGoalAction
method), 5
validate_data() (bun-
desliga_tippspiel.actions.GetMatchAction.GetMatchAction
method), 6
validate_data() (bun-
desliga_tippspiel.actions.GetPlayerAction.GetPlayerAction
method), 6
validate_data() (bun-
desliga_tippspiel.actions.GetReminderSettingsAction.GetReminderSettingsAction
method), 7
validate_data() (bun-
desliga_tippspiel.actions.GetTeamAction.GetTeamAction
method), 7
validate_data() (bun-
desliga_tippspiel.actions.LeaderboardAction.LeaderboardAction
method), 8
validate_data() (bun-
desliga_tippspiel.actions.LoadSettingsAction.LoadSettingsAction
method), 8
validate_data() (bun-
desliga_tippspiel.actions.PlaceBetsAction.PlaceBetsAction
method), 8
validate_data() (bun-
desliga_tippspiel.actions.SetReminderSettingsAction.SetReminderSettingsAction
method), 9

W

wikimedia_icon_urls() (in module bun-
desliga_tippspiel.background.match_data),
10