
bundesliga-tippspiel

Release 0.17.2

Jan 18, 2019

CONTENTS

1	bundesliga_tippspiel package	3
1.1	Subpackages	3
1.2	Submodules	22
1.3	bundesliga_tippspiel.config module	22
1.4	bundesliga_tippspiel.run module	22
1.5	Module contents	22
2	bundesliga_tippspiel	23
3	Indices and tables	25
	Python Module Index	27
	Index	29

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[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: bundesliga_tippspiel.types.exceptions.ActionException, failure_url: str*) → `str`

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 `handle_id_fetch` (*_id: int, query_cls: flask_sqlalchemy.model.DefaultMeta*) → `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[bundesliga_tippspiel.models.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

bundesliga_tippspiel.actions.ApiKeyDeleteAction module

class bundesliga_tippspiel.actions.ApiKeyDeleteAction.**ApiKeyDeleteAction** (*api_key: str*)
Bases: *bundesliga_tippspiel.actions.Action.Action*
Action that allows a user to delete an API key

__init__ (*api_key: str*)
Initializes the ApiKeyDeleteAction object :param api_key: The API key to delete :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.ApiKeyGenAction module

class bundesliga_tippspiel.actions.ApiKeyGenAction.**ApiKeyGenAction** (*username: str, password: str*)
Bases: *bundesliga_tippspiel.actions.Action.Action*
Action that allows a user to generate a new API key

__init__ (*username: str, password: str*)
Initializes the ApiKeyGenAction object :param username: The user's username :param password: The user's password :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.ChangePasswordAction module

class bundesliga_tippspiel.actions.ChangePasswordAction.**ChangePasswordAction** (*old_password: str, new_password: str, password_repeat: str*)
Bases: *bundesliga_tippspiel.actions.Action.Action*
Action that allows a user to change their password

__init__ (*old_password: str, new_password: str, password_repeat: str*)
Initializes the ChangePasswordAction object :param old_password: The old password for verification purposes :param new_password: The new password :param password_repeat: The new password repeated :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.ConfirmAction module

class bundesliga_tippspiel.actions.ConfirmAction.**ConfirmAction** (*user_id: int,*
confirm_key: str)

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows the confirmation of previously registered users

__init__ (*user_id: int, confirm_key: str*)

Initializes the ConfirmAction object :param user_id: The user's ID :param confirm_key: The user's confirmation key :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.DeleteUserAction module

class bundesliga_tippspiel.actions.DeleteUserAction.**DeleteUserAction** (*password: str*)

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows a user to delete their account

__init__ (*password: str*)

Initializes the DeleteUserAction object :param password: The password of the user for verification purposes :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.ForgotPasswordAction module

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)

Bases: *bundesliga_tippspiel.actions.Action.Action*

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)

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

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.GetEmailReminderAction module

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.Action*

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.Action*

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.Action*

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.GetTeamAction module

```

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

```

Bases: *bundesliga_tippspiel.actions.Action.Action*

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:
                                                                    Op-
                                                                    tional[int]
                                                                    =
                                                                    None)
```

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows fetching a sorted leaderboard

```
__init__ (matchday: Optional[int] = None)
    Initializes the LeaderboardAction object :param matchday: The matchday for which to generate the leader-
    board.

    If None, will use the most current matchday
```

```
validate_data ()
    Validates user-provided data :return: None :raises ActionException: if any data discrepancies are found
```

bundesliga_tippspiel.actions.LoginAction module

```
class bundesliga_tippspiel.actions.LoginAction.LoginAction (username: str, pass-
                                                                    word: str, remember:
                                                                    bool)
```

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows the logging in of users

```
__init__ (username: str, password: str, remember: bool)
    Initializes the LoginAction object :param username: The user's username :param password: The user's
    password :param remember: If set to true, the login session will be persistent :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.PlaceBetsAction module

```
class bundesliga_tippspiel.actions.PlaceBetsAction.PlaceBetsAction (bets:
                                                                    Dict[str,
                                                                    Tuple[str,
                                                                    str]])
```

Bases: *bundesliga_tippspiel.actions.Action.Action*

Action that allows placing bets

```
__init__ (bets: Dict[str, Tuple[str, str]])
    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.RegisterAction module

bundesliga_tippspiel.actions.SendDueEmailRemindersAction module

bundesliga_tippspiel.actions.SetEmailReminderAction module

Module contents

1.1.2 bundesliga_tippspiel.models package

Subpackages

bundesliga_tippspiel.models.auth package

Submodules

bundesliga_tippspiel.models.auth.ApiKey module

class bundesliga_tippspiel.models.auth.ApiKey.**ApiKey**(*args, **kwargs)

Bases: `bundesliga_tippspiel.models.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the 'api_keys' SQL table An ApiKey is used for API access using HTTP basic auth

MAX_AGE = 2592000

The maximum age of an API key in seconds

__init__(*args, **kwargs)

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

creation_time

The time at which this API key was created as a UNIX timestamp

has_expired() → bool

Checks if the API key has expired. API Keys expire after 30 days :return: True if the key has expired, False otherwise

id

key_hash

The hash of the API key

user

The user associated with this API key

user_id

The ID of the user associated with this API key

verify_key (*key: str*) → bool

Checks if a given key is valid :param key: The key to check :return: True if the key is valid, False otherwise

bundesliga_tippspiel.models.auth.User module

class bundesliga_tippspiel.models.auth.User.**User** (*args, **kwargs)

Bases: `bundesliga_tippspiel.models.ModelMixin.ModelMixin`, `sqlalchemy.ext.declarative.api.Model`

Model that describes the ‘users’ SQL table A User stores a user’s information, including their email address, username and password hash

__init__ (*args, **kwargs)

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

api_keys

bets

confirmation_hash

The account’s confirmation hash. This is the hash of a key emailed to the user. Only once the user follows the link in the email containing the key will their account be activated

confirmed

The account’s confirmation status. Logins should be impossible as long as this value is False.

email

The user’s email address

get_id() → str

Method required by flask-login :return: The user’s ID as a unicode string

id

is_active

Property required by flask-login :return: True

is_anonymous

Property required by flask-login :return: True if the user is not confirmed, False otherwise

is_authenticated

Property required by flask-login :return: True if the user is confirmed, False otherwise

password_hash

The user’s hashed password, salted and hashed.

username

The user’s username

verify_password (*password: str*) → bool

Verifies a password against the password hash :param password: The password to check :return: True if the password matches, False otherwise

Module contents

bundesliga_tippspiel.models.match_data package

Submodules

bundesliga_tippspiel.models.match_data.Goal module

class bundesliga_tippspiel.models.match_data.Goal.**Goal**(*args, **kwargs)

Bases: `bundesliga_tippspiel.models.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

The away team’s score after the goal was scored

home_score

The home team’s score after the goal was scored

id

match

The match in which this goal was scored.

match_id

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

minute

The minute in which the goal was scored

minute_et

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

own_goal

Indicates whether or not this goal was an own goal

penalty

Indicates whether or not this goal was a penalty

player

The player that scored this goal.

player_id

The ID of the player that scored this goal. Acts as a foreign key.

bundesliga_tippspiel.models.match_data.Match module

class bundesliga_tippspiel.models.match_data.Match.**Match**(*args, **kwargs)

Bases: `bundesliga_tippspiel.models.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`

The current score of the away team.

`away_ft_score`

The final score of the away team

`away_ht_score`

The score of the away team at half time

`away_team`

The away team.

`away_team_id`

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

`bets`

`current_score`

Returns The current score formatted as a string

`finished`

Indicates whether or not the match has finished yet

`ft_score`

Returns The full time score formatted as a string

`goals`

`home_current_score`

The current score of the home team.

`home_ft_score`

The final score of the home team

`home_ht_score`

The score of the home team at half time

`home_team`

The home team.

`home_team_id`

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

`ht_score`

Returns The half time score formatted as a string

`id`

`kickoff`

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'

`kickoff_datetime`

Returns A datetime object representing the kickoff time

`matchday`

The match day of the match

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

started

Indicates whether or not the match has started yet

bundesliga_tippspiel.models.match_data.Player module

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

Bases: `bundesliga_tippspiel.models.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**id****name**

The name of the player

team

The team the player is affiliated with.

team_id

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

bundesliga_tippspiel.models.match_data.Team module

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

Bases: `bundesliga_tippspiel.models.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

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

icon_png

The URL of an image file representing the team’s logo in PNG format

icon_svg

The URL of an image file representing the team’s logo in SVG format

id**name**

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

players

short_name

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

Module contents

bundesliga_tippspiel.models.user_generated package

Submodules

bundesliga_tippspiel.models.user_generated.Bet module

class bundesliga_tippspiel.models.user_generated.Bet.**Bet** (*args, **kwargs)

Bases: `bundesliga_tippspiel.models.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

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

The score bet on the home team

id

match

The match that this bet refers to

match_id

The ID of the match that this bet refers to.

user

The user associated with this bet

user_id

The ID of the user associated with this bet

bundesliga_tippspiel.models.user_generated.EmailReminder module

Module contents

Submodules

bundesliga_tippspiel.models.ModelMixin module

class bundesliga_tippspiel.models.ModelMixin.**ModelMixin**
Bases: object

A mixin class that specifies a couple of methods all database models should implement

id = Column(None, Integer(), table=None, primary_key=True, nullable=False)
The ID is the primary key of the table and increments automatically

Module contents

1.1.3 bundesliga_tippspiel.routes package

Subpackages

bundesliga_tippspiel.routes.api package

Submodules

bundesliga_tippspiel.routes.api.getters module

bundesliga_tippspiel.routes.api.putters module

bundesliga_tippspiel.routes.api.update module

bundesliga_tippspiel.routes.api.user_management module

Module contents

bundesliga_tippspiel.routes.api.**load_routes()**
Loads all API routes :return: None

Submodules

bundesliga_tippspiel.routes.authentication module

bundesliga_tippspiel.routes.authentication.**login()**
Page that allows the user to log in :return: The generated HTML

bundesliga_tippspiel.routes.authentication.**logout()**
Logs out the user :return:

bundesliga_tippspiel.routes.betting module

bundesliga_tippspiel.routes.betting.**bets** (*matchday: Optional[int] = None*)

Displays all matches for a matchday with entries for betting :param matchday: The matchday to display :return: None

bundesliga_tippspiel.routes.betting.**match** (*match_id: int*)

Displays a single match :param match_id: The ID of the match to display :return: The Response

bundesliga_tippspiel.routes.errors module

bundesliga_tippspiel.routes.errors.**error_handling** (*error:*

werkzeug.exceptions.HTTPException)

Custom redirect for 401 errors :param error: The error that caused the error handler to be called :return: A redirect to the login page

bundesliga_tippspiel.routes.information module

bundesliga_tippspiel.routes.information.**leaderboard** ()

Displays a leaderboard. :return: The Response

bundesliga_tippspiel.routes.information.**team** (*team_id: int*)

Displays information about a single team :param team_id: The ID of the team to display :return: The response

bundesliga_tippspiel.routes.information.**user** (*user_id: int*)

Shows a page describing a user/ :param user_id: The ID of the user to display :return: The request response

bundesliga_tippspiel.routes.static module

bundesliga_tippspiel.routes.static.**about** ()

The about page/"Impressum" for the website :return: The generated HTML

bundesliga_tippspiel.routes.static.**index** ()

The index/home page :return: The generated HTML

bundesliga_tippspiel.routes.static.**privacy** ()

Page containing a privacy disclaimer :return: The generated HTML

bundesliga_tippspiel.routes.user_management module

Module contents

bundesliga_tippspiel.routes.**load_routes** ()

Loads all application routes :return: None

1.1.4 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.TestApiKeyDeleteAction module

bundesliga_tippspiel.test.actions.TestApiKeyGenAction module

bundesliga_tippspiel.test.actions.TestChangePasswordAction module

bundesliga_tippspiel.test.actions.TestConfirmAction module

bundesliga_tippspiel.test.actions.TestDeleteUserAction module

bundesliga_tippspiel.test.actions.TestForgotPasswordAction module

bundesliga_tippspiel.test.actions.TestGetBetAction module

bundesliga_tippspiel.test.actions.TestGetEmailReminderAction module

bundesliga_tippspiel.test.actions.TestGetGoalAction module

bundesliga_tippspiel.test.actions.TestGetMatchAction module

bundesliga_tippspiel.test.actions.TestGetPlayerAction module

bundesliga_tippspiel.test.actions.TestGetTeamAction module

bundesliga_tippspiel.test.actions.TestLeaderboardAction module

bundesliga_tippspiel.test.actions.TestLoginAction module

bundesliga_tippspiel.test.actions.TestPlaceBetsAction module

bundesliga_tippspiel.test.actions.TestRegisterAction module

bundesliga_tippspiel.test.actions.TestSendDueEmailRemindersAction module

bundesliga_tippspiel.test.actions.TestSetEmailReminderAction module

Enumeration that defines the various levels of severity an alert can have

```
DANGER = 'danger'
    Translates to a red alert

INFO = 'info'
    Translates to a blue alert

SUCCESS = 'success'
    Translates to a green alert

WARNING = 'warning'
    Translates to a yellow alert
```

bundesliga_tippspiel.types.exceptions module

```
exception bundesliga_tippspiel.types.exceptions.ActionException (reason:
    str, display_message:
    str, status_code:
    int = 400, severity:
    bundesliga_tippspiel.types.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:
    bundesliga_tippspiel.types.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

Module contents

1.1.6 bundesliga_tippspiel.utils package

Submodules

bundesliga_tippspiel.utils.chart_data module

`bundesliga_tippspiel.utils.chart_data.generate_leaderboard_data()` → Tuple[int, Dict[str, Tuple[str, List[int]]]]

Generates leaderboard data for the rankings chart :return: A tuple consisting of the matchday to display
and the leaderboard data: username: (colour, list of positions per matchday)

bundesliga_tippspiel.utils.crypto module

`bundesliga_tippspiel.utils.crypto.generate_hash(password: str)` → bytes

Salts and hashes a password to generate a hash for storage in a database :param password: The password to hash
:return: The hash of the password

`bundesliga_tippspiel.utils.crypto.generate_random(length: int)` → bytes

Generates a random byte string consisting of alphanumeric characters Thanks @ Albert (<https://stackoverflow.com/users/281021/albert>) <https://stackoverflow.com/questions/2257441> :param length: The length of the string to generate :return: The generated random byte string

`bundesliga_tippspiel.utils.crypto.verify_password(password: str, hashed: str)`

Verifies that a password matches a given hash :param password: The password to verify :param hashed: The hash to verify the password against :return: True if the password matches, otherwise False

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.email module

bundesliga_tippspiel.utils.env module

`bundesliga_tippspiel.utils.env.load_secrets(secrets_file: str)`

Loads a JSON file filled with configuration details and secrets into os.environ :param secrets_file: The file to load :return: None

`bundesliga_tippspiel.utils.env.resolve_env_variable` (*env_key*: str, *_type*: type = <class 'str'>, *default*: object = None) → object

Resolves an environment key. A non-existent environment key will lead to a `KeyError` unless the app is in testing mode, in which case database-related variables won't cause a `KeyError`. `KeyErrors` can also be provided using the 'default' argument :param env_key: The environment key to resolve :param _type: The type of the environment variable :param default: An optional default value :return: The resolved environment variable.

None if the app is in testing mode and the variable is db-related

bundesliga_tippspiel.utils.initialize module

`bundesliga_tippspiel.utils.initialize.initialize_app`()

Initializes the App :return: None

`bundesliga_tippspiel.utils.initialize.initialize_db` (*db_uri*: str)

Initializes the SQLAlchemy database :param db_uri: The URI of the database to initialize :return: None

`bundesliga_tippspiel.utils.initialize.initialize_login_manager`()

Initializes the login manager :return: None

bundesliga_tippspiel.utils.json module

`bundesliga_tippspiel.utils.json.jsonify_models` (*data*: Dict[str, Any], *deep*: bool = True) → Dict[str, Any]

Serializes a dictionary and calls the `__json__()` method on all objects that have one. :param data: The data to serialize :param deep: Indicates whether or not child models are included.

If False, only IDs will be included.

Returns The serialized data

bundesliga_tippspiel.utils.match_data_getter module

bundesliga_tippspiel.utils.recaptcha module

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.routes.api` (*func*: Callable) → Callable

Decorator that handles common API patterns and ensures that the JSON response will always follow a certain pattern :param func: The function to wrap :return: The wrapper function

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

Decorator to make unauthorized API calls respond with JSON properly :param func: The function to wrap :return: The wrapped function

Module contents

1.2 Submodules

1.3 bundesliga_tippspiel.config module

1.4 bundesliga_tippspiel.run module

1.5 Module contents

`bundesliga_tippspiel.app = <Flask 'bundesliga_tippspiel'>`
The Flask App

`bundesliga_tippspiel.db = <SQLAlchemy engine=None>`
The SQLAlchemy database connection

`bundesliga_tippspiel.login_manager = <flask_login.login_manager.LoginManager object>`
The Flask-Login Login Manager

BUNDESLIGA_TIPPSPIEL

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

- bundesliga_tippspiel, 22
- bundesliga_tippspiel.actions, 9
 - Action, 3
 - ApiKeyDeleteAction, 4
 - ApiKeyGenAction, 4
 - ChangePasswordAction, 4
 - ConfirmAction, 5
 - DeleteUserAction, 5
 - GetBetAction, 5
 - GetGoalAction, 6
 - GetMatchAction, 7
 - GetPlayerAction, 7
 - GetTeamAction, 7
 - LeaderboardAction, 8
 - LoginAction, 8
 - PlaceBetsAction, 8
- bundesliga_tippspiel.models, 15
 - auth, 11
 - ApiKey, 9
 - User, 10
 - match_data, 14
 - Goal, 11
 - Match, 11
 - match_data.Player, 13
 - match_data.Team, 13
 - ModelMixin, 15
 - user_generated, 15
 - Bet, 14
 - routes, 16
 - api, 15
 - authentication, 15
 - betting, 16
 - errors, 16
 - information, 16
 - static, 16
 - test, 18
 - actions, 18
 - models, 18
 - auth, 18
 - match_data, 18
 - user_generated, 18
 - routes, 18
 - api, 18
 - utils, 18
 - types, 20
 - enums, 18
 - exceptions, 19
 - utils, 22
 - chart_data, 20
 - crypto, 20
 - db, 20
 - env, 20

bundesliga_tippspiel.utils.initialize,
21
bundesliga_tippspiel.utils.json,21
bundesliga_tippspiel.utils.routes,21

Symbols

- `__init__()` (*bundesliga_tippspiel.actions.ApiKeyDeleteAction.ApiKeyDeleteAction* method), 4
- `__init__()` (*bundesliga_tippspiel.actions.ApiKeyGenAction.ApiKeyGenAction* method), 4
- `__init__()` (*bundesliga_tippspiel.actions.ChangePasswordAction.ChangePasswordAction* method), 4
- `__init__()` (*bundesliga_tippspiel.actions.ConfirmAction.ConfirmAction* method), 5
- `__init__()` (*bundesliga_tippspiel.actions.DeleteUserAction.DeleteUserAction* method), 5
- `__init__()` (*bundesliga_tippspiel.actions.GetBetAction.GetBetAction* method), 5
- `__init__()` (*bundesliga_tippspiel.actions.GetGoalAction.GetGoalAction* method), 6
- `__init__()` (*bundesliga_tippspiel.actions.GetMatchAction.GetMatchAction* method), 7
- `__init__()` (*bundesliga_tippspiel.actions.GetPlayerAction.GetPlayerAction* method), 7
- `__init__()` (*bundesliga_tippspiel.actions.GetTeamAction.GetTeamAction* method), 7
- `__init__()` (*bundesliga_tippspiel.actions.LeaderboardAction.LeaderboardAction* method), 8
- `__init__()` (*bundesliga_tippspiel.actions.LoginAction.LoginAction* method), 8
- `__init__()` (*bundesliga_tippspiel.actions.PlaceBetsAction.PlaceBetsAction* method), 8
- `__init__()` (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* method), 9
- `__init__()` (*bundesliga_tippspiel.models.auth.User.User* method), 10
- `__init__()` (*bundesliga_tippspiel.models.match_data.Goal.Goal* method), 11
- `__init__()` (*bundesliga_tippspiel.models.match_data.Match.Match* method), 11
- `__init__()` (*bundesliga_tippspiel.models.match_data.Player.Player* method), 13
- `__init__()` (*bundesliga_tippspiel.models.match_data.Team.Team* method), 13
- `__init__()` (*bundesliga_tippspiel.models.user_generated.Bet.Bet* method), 14
- `__init__()` (*bundesliga_tippspiel.types.exceptions.ActionException* method), 19

attribute), 12
 away_team_id (bundesliga_tippspiel.models.match_data.Match.Match *attribute*), 12
B
 Bet (class in bundesliga_tippspiel.models.user_generated.Bets), 14
 bets (bundesliga_tippspiel.models.auth.User.User *attribute*), 10
 bets (bundesliga_tippspiel.models.match_data.Match.Match *attribute*), 12
 bets () (in module bundesliga_tippspiel.routes.betting), 16
 bundesliga_tippspiel (module), 22
 bundesliga_tippspiel.actions (module), 9
 bundesliga_tippspiel.actions.Action (module), 3
 bundesliga_tippspiel.actions.ApiKeyDeleteAction (module), 4
 bundesliga_tippspiel.actions.ApiKeyGenAction (module), 4
 bundesliga_tippspiel.actions.ChangePasswordAction (module), 4
 bundesliga_tippspiel.actions.ConfirmAction (module), 5
 bundesliga_tippspiel.actions.DeleteUserAction (module), 5
 bundesliga_tippspiel.actions.GetBetAction (module), 5
 bundesliga_tippspiel.actions.GetGoalAction (module), 6
 bundesliga_tippspiel.actions.GetMatchAction (module), 7
 bundesliga_tippspiel.actions.GetPlayerAction (module), 7
 bundesliga_tippspiel.actions.GetTeamAction (module), 7
 bundesliga_tippspiel.actions.LeaderboardAction (module), 8
 bundesliga_tippspiel.actions.LoginAction (module), 8
 bundesliga_tippspiel.actions.PlaceBetsAction (module), 8
 bundesliga_tippspiel.models (module), 15
 bundesliga_tippspiel.models.auth (module), 11
 bundesliga_tippspiel.models.auth.ApiKey (module), 9
 bundesliga_tippspiel.models.auth.User (module), 10
 bundesliga_tippspiel.models.match_data (module), 14
 bundesliga_tippspiel.models.match_data.Goal (module), 11
 bundesliga_tippspiel.models.match_data.Match (module), 11
 bundesliga_tippspiel.models.match_data.Player (module), 13
 bundesliga_tippspiel.models.match_data.Team (module), 13
 bundesliga_tippspiel.models.ModelMixin (module), 15
 bundesliga_tippspiel.models.user_generated (module), 15
 bundesliga_tippspiel.models.user_generated.Bet (module), 14
 bundesliga_tippspiel.routes (module), 16
 bundesliga_tippspiel.routes.api (module), 15
 bundesliga_tippspiel.routes.authentication (module), 15
 bundesliga_tippspiel.routes.betting (module), 16
 bundesliga_tippspiel.routes.errors (module), 16
 bundesliga_tippspiel.routes.information (module), 16
 bundesliga_tippspiel.routes.static (module), 16
 bundesliga_tippspiel.test (module), 18
 bundesliga_tippspiel.test.actions (module), 18
 bundesliga_tippspiel.test.models (module), 18
 bundesliga_tippspiel.test.models.auth (module), 18
 bundesliga_tippspiel.test.models.match_data (module), 18
 bundesliga_tippspiel.test.models.user_generated (module), 18
 bundesliga_tippspiel.test.routes (module), 18
 bundesliga_tippspiel.test.routes.api (module), 18
 bundesliga_tippspiel.test.utils (module), 18
 bundesliga_tippspiel.types (module), 20
 bundesliga_tippspiel.types.enums (module), 18
 bundesliga_tippspiel.types.exceptions (module), 19
 bundesliga_tippspiel.utils (module), 22
 bundesliga_tippspiel.utils.chart_data (module), 20
 bundesliga_tippspiel.utils.crypto (module), 20

bundesliga_tippspiel.utils.db (module), 20
 bundesliga_tippspiel.utils.env (module), 20
 bundesliga_tippspiel.utils.initialize (module), 21
 bundesliga_tippspiel.utils.json (module), 21
 bundesliga_tippspiel.utils.routes (module), 21

C

ChangePasswordAction (class in bundesliga_tippspiel.actions.ChangePasswordAction), 4
 check_id_or_filters() (bundesliga_tippspiel.actions.Action.Action static method), 3
 ConfirmAction (class in bundesliga_tippspiel.actions.ConfirmAction), 5
 confirmation_hash (bundesliga_tippspiel.models.auth.User.User attribute), 10
 confirmed (bundesliga_tippspiel.models.auth.User.User attribute), 10
 creation_time (bundesliga_tippspiel.models.auth.ApiKey.ApiKey attribute), 9
 current_score (bundesliga_tippspiel.models.match_data.Match.Match attribute), 12

D

DANGER (bundesliga_tippspiel.types.enums.AlertSeverity attribute), 19
 db (in module bundesliga_tippspiel), 22
 DeleteUserAction (class in bundesliga_tippspiel.actions.DeleteUserAction), 5

E

email (bundesliga_tippspiel.models.auth.User.User attribute), 10
 email_exists() (in module bundesliga_tippspiel.utils.db), 20
 error_handling() (in module bundesliga_tippspiel.routes.errors), 16
 evaluate() (bundesliga_tippspiel.models.user_generated.Bet.Bet static method), 14
 execute() (bundesliga_tippspiel.actions.Action.Action method), 3
 execute_with_redirects() (bundesliga_tippspiel.actions.Action.Action method), 3

F

finished (bundesliga_tippspiel.models.match_data.Match.Match attribute), 12
 flash() (bundesliga_tippspiel.types.exceptions.ActionException method), 19
 from_dict() (bundesliga_tippspiel.actions.Action.Action class method), 3
 from_site_request() (bundesliga_tippspiel.actions.Action.Action class method), 3
 ft_score (bundesliga_tippspiel.models.match_data.Match.Match attribute), 12

G

generate_hash() (in module bundesliga_tippspiel.utils.crypto), 20
 generate_leaderboard_data() (in module bundesliga_tippspiel.utils.chart_data), 20
 generate_random() (in module bundesliga_tippspiel.utils.crypto), 20
 get_id() (bundesliga_tippspiel.models.auth.User.User method), 10
 GetBetAction (class in bundesliga_tippspiel.actions.GetBetAction), 5
 GetGoalAction (class in bundesliga_tippspiel.actions.GetGoalAction), 6
 GetMatchAction (class in bundesliga_tippspiel.actions.GetMatchAction), 7
 GetPlayerAction (class in bundesliga_tippspiel.actions.GetPlayerAction), 7
 GetTeamAction (class in bundesliga_tippspiel.actions.GetTeamAction), 7
 Goal (class in bundesliga_tippspiel.models.match_data.Goal), 11
 goals (bundesliga_tippspiel.models.match_data.Match.Match attribute), 12
 goals (bundesliga_tippspiel.models.match_data.Player.Player attribute), 13

H

handle_id_fetch() (bundesliga_tippspiel.actions.Action.Action static method), 3
 has_expired() (bundesliga_tippspiel.models.auth.ApiKey.ApiKey method), 9
 home_current_score (bundesliga_tippspiel.models.match_data.Match.Match attribute), 12

attribute), 12
 home_ft_score (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 home_ht_score (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 home_score (*bundesliga_tippspiel.models.match_data.Goal.Goal* *attribute*), 11
 home_score (*bundesliga_tippspiel.models.user_generated.Bet.Bet* *attribute*), 14
 home_team (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 home_team_id (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 ht_score (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12

I
 icon_png (*bundesliga_tippspiel.models.match_data.Team.Team* *attribute*), 13
 icon_svg (*bundesliga_tippspiel.models.match_data.Team.Team* *attribute*), 13
 id (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* *attribute*), 9
 id (*bundesliga_tippspiel.models.auth.User.User* *attribute*), 10
 id (*bundesliga_tippspiel.models.match_data.Goal.Goal* *attribute*), 11
 id (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 id (*bundesliga_tippspiel.models.match_data.Player.Player* *attribute*), 13
 id (*bundesliga_tippspiel.models.match_data.Team.Team* *attribute*), 13
 id (*bundesliga_tippspiel.models.ModelMixin.ModelMixin* *attribute*), 15
 id (*bundesliga_tippspiel.models.user_generated.Bet.Bet* *attribute*), 14
 index () (in module *bundesliga_tippspiel.routes.static*), 16
 INFO (*bundesliga_tippspiel.types.enums.AlertSeverity* *attribute*), 19
 initialize_app () (in module *bundesliga_tippspiel.utils.initialize*), 21
 initialize_db () (in module *bundesliga_tippspiel.utils.initialize*), 21
 initialize_login_manager () (in module *bundesliga_tippspiel.utils.initialize*), 21
 is_active (*bundesliga_tippspiel.models.auth.User.User* *attribute*), 10
 is_anonymous (*bundesliga_tippspiel.models.auth.User.User* *attribute*), 10

J
 jsonify_models () (in module *bundesliga_tippspiel.utils.json*), 21

K
 key_hash (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* *attribute*), 9
 kickoff (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12
 kickoff_datetime (*bundesliga_tippspiel.models.match_data.Match.Match* *attribute*), 12

L
 leaderboard () (in module *bundesliga_tippspiel.routes.information*), 16
 LeaderboardAction (class in *bundesliga_tippspiel.actions.LeaderboardAction*), 8
 load_routes () (in module *bundesliga_tippspiel.routes*), 16
 load_routes () (in module *bundesliga_tippspiel.routes.api*), 15
 load_secrets () (in module *bundesliga_tippspiel.utils.env*), 20
 login () (in module *bundesliga_tippspiel.routes.authentication*), 15
 login_manager (in module *bundesliga_tippspiel*), 22
 LoginAction (class in *bundesliga_tippspiel.actions.LoginAction*), 8
 logout () (in module *bundesliga_tippspiel.routes.authentication*), 15

M
 match (*bundesliga_tippspiel.models.match_data.Goal.Goal* *attribute*), 11
 match (*bundesliga_tippspiel.models.user_generated.Bet.Bet* *attribute*), 14
 Match (class in *bundesliga_tippspiel.models.match_data.Match*), 11
 match () (in module *bundesliga_tippspiel.routes.betting*), 16
 match_id (*bundesliga_tippspiel.models.match_data.Goal.Goal* *attribute*), 11
 match_id (*bundesliga_tippspiel.models.user_generated.Bet.Bet* *attribute*), 14

matchday (*bundesliga_tippspiel.models.match_data.Match* attribute), 12

MAX_AGE (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* attribute), 9

minute (*bundesliga_tippspiel.models.match_data.Goal.Goal* attribute), 11

minute_display (*bundesliga_tippspiel.models.match_data.Match.Match* attribute), 12

minute_et (*bundesliga_tippspiel.models.match_data.Goal.Goal* attribute), 11

ModelMixin (class in *bundesliga_tippspiel.models.ModelMixin*), 15

N

name (*bundesliga_tippspiel.models.match_data.Player.Player* attribute), 13

name (*bundesliga_tippspiel.models.match_data.Team.Team* attribute), 13

O

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

P

password_hash (*bundesliga_tippspiel.models.auth.User.User* attribute), 10

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

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

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

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

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

players (*bundesliga_tippspiel.models.match_data.Team.Team* attribute), 13

prepare_get_response () (*bundesliga_tippspiel.actions.Action.Action* method), 3

privacy () (in module *bundesliga_tippspiel.routes.static*), 16

R

resolve_and_check_matchday () (*bundesliga_tippspiel.actions.Action.Action* static method), 3

resolve_env_variable () (in module *bundesliga_tippspiel.utils.env*), 20

S

short_name (*bundesliga_tippspiel.models.match_data.Team.Team* attribute), 14

started (*bundesliga_tippspiel.models.match_data.Match.Match* attribute), 13

SUCCESS (*bundesliga_tippspiel.types.enums.AlertSeverity* attribute), 19

T

team (*bundesliga_tippspiel.models.match_data.Player.Player* attribute), 13

Team (class in *bundesliga_tippspiel.models.match_data.Team*), 13

team () (in module *bundesliga_tippspiel.routes.information*), 16

team_id (*bundesliga_tippspiel.models.match_data.Player.Player* attribute), 13

U

user (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* attribute), 9

user (*bundesliga_tippspiel.models.user_generated.Bet.Bet* attribute), 14

User (class in *bundesliga_tippspiel.models.auth.User*), 10

user () (in module *bundesliga_tippspiel.routes.information*), 16

user_exists () (in module *bundesliga_tippspiel.utils.db*), 20

user_id (*bundesliga_tippspiel.models.auth.ApiKey.ApiKey* attribute), 9

user_id (*bundesliga_tippspiel.models.user_generated.Bet.Bet* attribute), 14

username (*bundesliga_tippspiel.models.auth.User.User* attribute), 10

username_exists () (in module *bundesliga_tippspiel.utils.db*), 20

V

validate_data () (*bundesliga_tippspiel.actions.Action.Action* method), 4

validate_data () (*bundesliga_tippspiel.actions.ApiKeyDeleteAction.ApiKeyDeleteAction* method), 4

validate_data () (*bundesliga_tippspiel.actions.ApiKeyGenAction.ApiKeyGenAction* method), 4

validate_data () (*bundesliga_tippspiel.actions.ChangePasswordAction.ChangePasswordAction* method), 4

`validate_data()` (*bundesliga-tippspiel.actions.ConfirmAction.ConfirmAction* method), 5

`validate_data()` (*bundesliga-tippspiel.actions.DeleteUserAction.DeleteUserAction* method), 5

`validate_data()` (*bundesliga-tippspiel.actions.GetBetAction.GetBetAction* method), 6

`validate_data()` (*bundesliga-tippspiel.actions.GetGoalAction.GetGoalAction* method), 6

`validate_data()` (*bundesliga-tippspiel.actions.GetMatchAction.GetMatchAction* method), 7

`validate_data()` (*bundesliga-tippspiel.actions.GetPlayerAction.GetPlayerAction* method), 7

`validate_data()` (*bundesliga-tippspiel.actions.GetTeamAction.GetTeamAction* method), 8

`validate_data()` (*bundesliga-tippspiel.actions.LeaderboardAction.LeaderboardAction* method), 8

`validate_data()` (*bundesliga-tippspiel.actions.LoginAction.LoginAction* method), 8

`validate_data()` (*bundesliga-tippspiel.actions.PlaceBetsAction.PlaceBetsAction* method), 9

`verify_key()` (*bundesliga-tippspiel.models.auth.ApiKey.ApiKey* method), 9

`verify_password()` (*bundesliga-tippspiel.models.auth.User.User* method), 10

`verify_password()` (*in module bundesliga-tippspiel.utils.crypto*), 20

W

`WARNING` (*bundesliga-tippspiel.types.enums.AlertSeverity* attribute), 19