gamejolt

Sep 03, 2019

Contents

class gamejolt.**GameJoltAPI**(*game_id*, *private_key*, *username=None*, *token=None*)

data_store_fetch(*key*, *public=False*)
Retrieve the data for a specific key.

Retrieve data for the specified key. If a "username" and "token" is available, it will be retrieved from that specific account. Otherwise, the public key data will be retrieved. You can also pass the 'public=True' parameter to specifically request public data retrieval.

Parameters
- **key** – The key for the data to retrieve.
- **public** – If True, always retrieve public key data.

Returns A *Future* containing the API response.

data_store_get_keys(*public=False*)
Retrieve all available keys in the data store.

If a "username" and "token" is avialable, the keys will be shown for that specific account. Otherwise, the public keys will be shown. You can also pass the 'public=True' parameter to specifically retrieve public keys.

Parameters **public** – If True, retrieve the public keys.

Returns A *Future* containing the API response.

data_store_remove(*key*, *public=False*)
Delete a key and it's data from the data store.

Completely remove a key, and it's data from the data store. If a "username" and "token" is available, the key will be deleted from that specific account. Otherwise, the public key and it's data will be deleted. You can also pass the 'public=True' parameter to specifically target the public data for deletion.

Parameters
- **key** – The key and its data to delete.
- **public** – If True, delete a public key and data.

Returns A *Future* containing the API response.

data_store_set(*key*, *data*, *public=False*)
Set the data for a specific key.

Set data for the specified key. If a "username" and "token" is available, it will be set for that account. Otherwise, the data will be set to the public key. You can also pass the 'public=True' parameter to specifically set the public key data.

Parameters
- **key** – The key to set the data to.
- **data** – The data to send.
- **public** – If True, always set the data to the public key.

Returns A *Future* containing the API response.

data_store_update(*key*, *operation*, *value*, *public=False*)
Perform an update operation on data for a specific key.

You can perform specific operations to previously set key data. The available operations are 'add', 'subtract', 'multiply' and 'divide' for numeric data, and 'append' and 'prepend' for string data. See the Game Jolt API documentation for more information.

If a "username" and "token" is available, the operations are performed for data under that account. Otherwise, the operations are performed on the public key data. You can also pass the 'public=True' parameter to specifically operate on the public data.

Parameters
- **key** – The key to perform the operation on.

- **operation** – The operation to perform.
- **value** – The value to use in the operation.
- **public** – If True, always operate on public data.

Returns A *Future* containing the API response.

scores_add(*sort*, *score=None*, *table_id=None*)
Submit a new game score.

Submit a new score to the default table, or to a specific table. If a "username" and "token" is available, the score will be submitted under that users account. Otherwise, the score will be submitted as a guest.

Parameters
- **sort** – An int of the score.
- **score** – An optional score display name. If None, defaults to the sort value.
- **table_id** – Add the score to a specific table. If None, defaults to the Main table.

Returns A *Future* containing the API response.

scores_fetch(*table_id=None*, *limit=10*)
Fetch the game scores.

Parameters
- **table_id** – Fetch a specific score table. If None, the default score table will be returned.
- **limit** – Set an upper limit on how many scores to return. Defaults to 10.

Returns A *Future* containing the API response.

scores_tables()
Retrieve information on available score tables.

Returns A *Future* containing the API response.

session_close()
Close an open user session.

Returns A *Future* containing the API response.

session_open()
Open a user session.

Returns A *Future* containing the API response.

session_ping(*idle=False*)
Send a keep-alive ping for an open session.

Parameters **idle** – If True, set the session status to 'idle'.

Returns A *Future* containing the API response.

trophies_add_achieved(*trophy_id*)
Set a specific trophy ID as having been achieved.

Parameters **trophy_id** – The ID of the trophy to set to achieved.

Returns A *Future* containing the API response.

trophies_fetch(*achieved=False*, *trophy_id=None*)
Fetch a dictionary of trophies.

Parameters
- **achieved** – If True, only fetch achieved trophies.
- **trophy_id** – Fetch a specific trophy, by ID. Setting this will override the *achieved* parameter.

Returns A *Future* containing the API response.

Index

D

G

S

T