Gardner Project

Willem van der Schans
Version 1.0
April 2023

Table of Contents

README

Gardner API Utility

A collection of data source api call utilities.

VERSION INFO

1. Python=3.10
2. pandas~=1.5.2
3. requests~=2.28.1
4. beautifulsoup4~=4.11.1
5. pysimplegui~=4.60.4
6. cryptography~=38.0.1
7. pillow~=9.2.0

Note: Use latest viable requirements for versions above

Authentication Requirements

Authentication Keys are needed for utahrealestate.com and constructionmonitor.com

The program provides a safe way to store and use authentication keys

License

Namespace Index

Namespace List

Here is a list of all namespaces with brief descriptions:

Class Index

Class List

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

Namespace Documentation

API_Calls Namespace Reference

Namespaces

- namespace _main_
- namespace Initializer

API_Calls._main_ Namespace Reference

API_Calls.Initializer Namespace Reference

Classes

class initializer

AuthUtil Namespace Reference

Classes

class AuthUtil

BatchGui Namespace Reference

Functions

- def BatchInputGui (batches)

Function Documentation

def BatchGui.BatchInputGui (*batches*)

```
The BatchInputGui function is a simple GUI that asks the user if they want to continue
with the number of batches
that have been selected. This function is called by the BatchInputGui function in order
to confirm that this is what
the user wants.

Args:
batches: Display the number of batches that will be run

Returns:
A boolean value

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 6 of file BatchGui.py.

```
00006 def BatchInputGui(batches):
00007     """
00008 The BatchInputGui function is a simple GUI that asks the user if they want to
continue with the number of batches
00009 that have been selected. This function is called by the BatchInputGui function
in order to confirm that this is what
00010 the user wants.
00011
00012 Args:
00013     batches: Display the number of batches that will be run
00014
00015 Returns:
00016     A boolean value
00017
00018 Doc Author:
00019     Willem van der Schans, Trelent AI
00020 """
00021     __text1 = f"This request will run {batches} batches"
00022     __text2 = "Do you want to continue?"
00023
00024     __Line1 = [sg.Push(),
00025                sg.Text(__text1, justification="center"),
00026                sg.Push()]
00027
00028     __Line2 = [sg.Push(),
00029                sg.Text(__text2, justification="center"),
00030                sg.Push()]
00031
00032     __Line3 = [sg.Push(),
00033                sg.Ok("Continue"),
00034                sg.Cancel(),
00035                sg.Push()]
00036
00037     window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
00038                        modal=True,
00039                        keep_on_top=True,
00040                        disable_close=False,
00041                        icon=ImageLoader("taskbar_icon.ico"),
00042                        size=(290, 100))
00043
00044     while True:
```

```
00045         event, values = window.read()
00046         if event == "Continue":
00047             break
00048         elif event == sg.WIN_CLOSED or event == "Cancel":
00049 
00050             break
00051 
00052     window.close()
```

BatchProcessing Namespace Reference

Classes

- class BatchProcessorConstructionMonitorclass BatchProcessorUtahRealEstate

Functions

- def BatchCalculator (TotalRecords, Argument_Dict)

Function Documentation

def BatchProcessing.BatchCalculator (*TotalRecords*, *Argument_Dict*)

```
The BatchCalculator function takes two arguments:
1. TotalRecords - the total number of records in the database
2. Argument_Dict - a dictionary containing all the arguments passed to this function
by the user

Args:
TotalRecords: Determine the number of batches that will be needed to complete the query
Argument_Dict: Pass in the arguments that will be used to query the database

Returns:
The total number of batches that will be made

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 25 of file BatchProcessing.py.

```
00025 def BatchCalculator(TotalRecords, Argument_Dict):
00026     """
00027 The BatchCalculator function takes two arguments:
00028     1. TotalRecords - the total number of records in the database
00029     2. Argument_Dict - a dictionary containing all the arguments passed to this
function by the user
00030
00031 Args:
00032     TotalRecords: Determine the number of batches that will be needed to complete
the query
00033     Argument_Dict: Pass in the arguments that will be used to query the database
00034
00035 Returns:
00036     The total number of batches that will be made
00037
00038 Doc Author:
00039     Willem van der Schans, Trelent AI
00040 """
00041     try:
00042         document_limit = Argument_Dict["size"]
00043     except Exception as e:
00044         # Logging
00045         print(
00046             f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Batch Calculator document limit
overwritten to 200 from input")
00047         document_limit = 200
00048
00049     return int(math.ceil(float(TotalRecords) / float(document_limit)))
00050
00051
```

BatchProgressGUI Namespace Reference

Classes

class BatchProgressGUIVariables

- int counter = 1

Variable Documentation

int BatchProgressGUI.counter = 1

Definition at line 27 of file BatchProgressGUI.py.

Core Namespace Reference

Classes

- class Cencusclass ConstructionMonitorInit
- class ConstructionMonitorMain
- class realtorCom
- class UtahRealEstateInit
- class UtahRealEstateMain

DataChecker Namespace Reference

Functions

- def DataChecker (Name, DataPath)

Function Documentation

def DataChecker.DataChecker (*Name*, *DataPath*)

```
The DataChecker function is used to check if the user has selected a valid data file.
If the user selects an invalid file, they will be prompted to select another one until
they choose a valid one.

Args:
Name: Display the name of the data file that is being selected
Path: Set the initial folder for the file browser

Returns:
A list of all the data files in a directory

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 24 of file DataChecker.py.

```
00024 def DataChecker(Name, DataPath):
00025     """
00026 The DataChecker function is used to check if the user has selected a valid data
file.
00027     If the user selects an invalid file, they will be prompted to select another
one until
00028     they choose a valid one.
00029
00030 Args:
00031     Name: Display the name of the data file that is being selected
00032     Path: Set the initial folder for the file browser
00033
00034 Returns:
00035     A list of all the data files in a directory
00036
00037 Doc Author:
00038     Willem van der Schans, Trelent AI
00039 """
00040     __text1 = f"Select existing {Name} csv data file:"
00041
00042     __Line1 = [sg.Push(),
00043                sg.Text(__text1, justification="center"),
00044                sg.Push()]
00045
00046     __Line2 = [sg.Text("Choose a file: "),
00047                sg.Input(),
00048                sg.FileBrowse(file_types=(("Data Files (.csv)", "*.csv"),),
initial_folder=DataPath)]
00049
00050     __Line3 = [sg.Push(),
00051                sg.Ok("Continue"),
00052                sg.Cancel(),
00053                sg.Push()]
00054
00055     window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
00056                        modal=True,
00057                        keep_on_top=True,
00058                        disable_close=False,
00059                        icon=ImageLoader("taskbar_icon.ico"))
00060
00061     while True:
00062         event, values = window.read()
```

```
00063         if event == "Continue":
00064             break
00065         elif event == sg.WIN_CLOSED or event == "Cancel":
00066 
00067             break
00068 
00069     window.close()
00070 
00071 
```

DataSupportFunctions Namespace Reference

Functions

- def StringToList (string)

Function Documentation

def DataSupportFunctions.StringToList (*string*)

```
The StringToList function takes a string and converts it into a list.
The function is used to convert the input from the user into a list of strings, which
can then be iterated through.

Args:
string: Split the string into a list

Returns:
A list of strings

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 16 of file DataSupportFunctions.py.

```
00016 def StringToList(string):
00017     """
00018 The StringToList function takes a string and converts it into a list.
00019     The function is used to convert the input from the user into a list of strings,
which can then be iterated through.
00020
00021 Args:
00022     string: Split the string into a list
00023
00024 Returns:
00025     A list of strings
00026
00027 Doc Author:
00028     Willem van der Schans, Trelent AI
00029 """
00030     listOut = list(string.split(","))
00031     return listOut
```

DataTransfer Namespace Reference

Classes

class DataTransfer

ErrorPopup Namespace Reference

Functions

- def ErrorPopup (textString)

Function Documentation

def ErrorPopup.ErrorPopup (*textString*)

```
The ErrorPopup function is used to display a popup window with an error message.
It takes one argument, textString, which is the string that will be displayed in the
popup window.
The function also opens up the log folder upon program exit.

Args:
textString: Display the error message

Returns:
Nothing, but it does print an error message to the console

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 18 of file ErrorPopup.py.

```
00018 def ErrorPopup(textString):
00019     """
00020 The ErrorPopup function is used to display a popup window with an error message.
00021 It takes one argument, textString, which is the string that will be displayed
in the popup window.
00022 The function also opens up the log folder upon program exit.
00023
00024 Args:
00025     textString: Display the error message
00026
00027 Returns:
00028     Nothing, but it does print an error message to the console
00029
00030 Doc Author:
00031     Willem van der Schans, Trelent AI
00032 """
00033     PopupWrapped(
00034         f"ERROR @ {textString} \n"
00035         f"Log folder will be opened upon program exit",
00036         windowType="FatalErrorLarge")
```

ErrorPrint Namespace Reference

Functions

- def RESTErrorPrint (response)

Function Documentation

def ErrorPrint.RESTErrorPrint (*response*)

```
The RESTErrorPrint function is used to print the response from a ReST API call.
If the response is an integer, it will be printed as-is. If it's not an integer,
it will be converted to text and then printed.

Args:
response: Print the response from a rest api call

Returns:
The response text

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 18 of file ErrorPrint.py.

```
00018 def RESTErrorPrint(response):
00019     """
00020 The RESTErrorPrint function is used to print the response from a ReST API call.
00021 If the response is an integer, it will be printed as-is. If it's not an integer,
00022 it will be converted to text and then printed.
00023
00024 Args:
00025     response: Print the response from a rest api call
00026
00027 Returns:
00028     The response text
00029
00030 Doc Author:
00031     Willem van der Schans, Trelent AI
00032 """
00033     if isinstance(response, int):
00034         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Resource Response: {response}")
00035     else:
00036         response_txt = response.text
00037         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Resource Response: {response_txt}")
```

FileSaver Namespace Reference

Classes

class FileSaver

ImageLoader Namespace Reference

Functions

- def ImageLoader (file)

Function Documentation

def ImageLoader.ImageLoader (*file*)

```
The ImageLoader function takes in a file name and returns the image as a base64 encoded
string.
This is used to send images to the API for processing.

Args:
file: Specify the image file to be loaded

Returns:
A base64 encoded image string

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 24 of file ImageLoader.py.

```
00024 def ImageLoader(file):
00025     """
00026 The ImageLoader function takes in a file name and returns the image as a base64
encoded string.
00027 This is used to send images to the API for processing.
00028
00029 Args:
00030     file: Specify the image file to be loaded
00031
00032 Returns:
00033     A base64 encoded image string
00034
00035 Doc Author:
00036     Willem van der Schans, Trelent AI
00037 """
00038     try:
00039         __path = normpath(join(str(os.getcwd().split("API_Calls", 1)[0]),
"API_Calls"))
00040         __path = normpath(join(__path, "Images"))
00041         __path = join(__path, file).replace("\\", "/")
00042
00043         image = Image.open(__path)
00044
00045         __buff = BytesIO()
00046
00047         image.save(__buff, format="png")
00048
00049         img_str = base64.b64encode(__buff.getvalue())
00050
00051         return img_str
00052     except Exception as e:
00053         # We cannot log this error like other errors due to circular imports
00054         raise e
```

Logger Namespace Reference

Functions

- def logger ()

Function Documentation

def Logger.logger ()

```
The logger function creates a log file in the user's AppData directory.
The function will create the directory if it does not exist.
The function will also delete the oldest file when 100 logs have been saved to prevent
bloat.

Args:

Returns:
A file path to the log file that was created

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 22 of file Logger.py.

```
00022 def logger():
00023     """
00024 The logger function creates a log file in the user's AppData directory.
00025 The function will create the directory if it does not exist.
00026 The function will also delete the oldest file when 100 logs have been saved to
prevent bloat.
00027
00028 Args:
00029
00030 Returns:
00031     A file path to the log file that was created
00032
00033 Doc Author:
00034     Willem van der Schans, Trelent AI
00035 """
00036     dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
00037     if os.path.exists(dir_path):
00038         pass
00039     else:
00040         if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00041             os.mkdir(dir_path)
00042         else:
00043             os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00044             os.mkdir(dir_path)
00045
00046     filePath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
00047         f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
00048     sys.stdout = open(filePath, 'w')
00049     sys.stderr = sys.stdin = sys.stdout
00050
00051     def sorted_ls(path):
00052         """
00053     The sorted_ls function takes a path as an argument and returns the files in
that directory sorted by modification time.
00054
00055     Args:
00056         path: Specify the directory to be sorted
00057
00058     Returns:
00059         A list of files in a directory sorted by modification time
00060
```

```
00061     Doc Author:
00062         Willem van der Schans, Trelent AI
00063     """
00064         mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
00065         return list(sorted(os.listdir(path), key=mtime))
00066 
00067     del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
00068     for file in del_list:
00069         os.remove(dir_path.joinpath(file))
00070         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Log file {file} deleted")
```

PopupWrapped Namespace Reference

Classes

class PopupWrapped

PrintFunc Namespace Reference

RESTError Namespace Reference

Functions

- def RESTError (response)

Function Documentation

def RESTError.RESTError (*response*)

```
The RESTError function is a function that checks the status codes.
If it is 200, then everything went well and nothing happens. If it isn't 200, then an
error message will be printed to
the console with information about what happened (i.e., if there was an authentication
error or if the resource wasn't found).
The function also raises an exception and opens an error popup for easy debugging.

Args:
response: Print out the response from the server

Returns:
A text string

Doc Author:
Trelent
```

Definition at line 21 of file RESTError.py.

```
00021 def RESTError(response):
00022     """
00023 The RESTError function is a function that checks the status codes.
00024 If it is 200, then everything went well and nothing happens. If it isn't 200,
then an error message will be printed to
00025 the console with information about what happened (i.e., if there was an
authentication error or if the resource wasn't found).
00026 The function also raises an exception and opens an error popup for easy debugging.
00027
00028 Args:
00029     response: Print out the response from the server
00030
00031 Returns:
00032     A text string
00033
00034 Doc Author:
00035     Trelent
00036 """
00037     if isinstance(response, int):
00038         status_code = response
00039     else:
00040         status_code = response.status_code
00041
00042     if status_code == 200:
00043         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Api Request completed successfully"
00044         print(textString)
00045         pass
00046     elif status_code == 301:
00047         RESTErrorPrint(response)
00048         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Endpoint redirection; check domain
name and endpoint name"
00049         ErrorPopup(textString)
00050         raise ValueError(textString)
00051     elif status_code == 400:
00052         RESTErrorPrint(response)
00053         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Bad Request; check input arguments"
00054         ErrorPopup(textString)
```

```
00055             raise ValueError(textString)
00056         elif status_code == 401:
00057             RESTErrorPrint(response)
00058             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Authentication Error: No keys found"
00059             ErrorPopup(textString)
00060             raise PermissionError(textString)
00061         elif status_code == 402:
00062             RESTErrorPrint(response)
00063             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Authentication Error: Cannot access
decryption Key in %appdata%/roaming/GardnerUtil/security"
00064             ErrorPopup(textString)
00065             raise PermissionError(textString)
00066         elif status_code == 403:
00067             RESTErrorPrint(response)
00068             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Access Error: the resource you are
trying to access is forbidden"
00069             ErrorPopup(textString)
00070             raise PermissionError(textString)
00071         elif status code == 404:
00072             RESTErrorPrint(response)
00073             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Resource not found: the resource
you are trying to access does not exist on the server"
00074             ErrorPopup(textString)
00075             raise NameError(textString)
00076         elif status_code == 405:
00077             RESTErrorPrint(response)
00078             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Method is not valid, request
rejected by server"
00079             ErrorPopup(textString)
00080             raise ValueError(textString)
00081         elif status_code == 408:
00082             RESTErrorPrint(response)
00083             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout by server"
00084             ErrorPopup(textString)
00085             raise TimeoutError(textString)
00086         elif status_code == 503:
00087             RESTErrorPrint(response)
00088             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | The resource is not ready for the
get request"
00089             ErrorPopup(textString)
00090             raise SystemError(textString)
00091         elif status_code == 701:
00092             RESTErrorPrint(response)
00093             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Error in coercing icon to bits
(Imageloader.py)"
00094             ErrorPopup(textString)
00095             raise TypeError(textString)
00096         elif status_code == 801:
00097             RESTErrorPrint(response)
00098             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Resource Error, HTML cannot be
parsed the website's HTML source might be changed"
00099             ErrorPopup(textString)
00100             raise ValueError(textString)
00101         elif status_code == 790:
00102             RESTErrorPrint(response)
00103             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout within requests"
00104             ErrorPopup(textString)
00105             raise TimeoutError(textString)
00106         elif status_code == 791:
00107             RESTErrorPrint(response)
00108             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
00109             ErrorPopup(textString)
00110             raise ValueError(textString)
00111         elif status_code == 990:
00112             RESTErrorPrint(response)
```

```
00113             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No password input"
00114             ErrorPopup(textString)
00115             raise ValueError(textString)
00116         elif status_code == 991:
00117             RESTErrorPrint(response)
00118             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No username input"
00119             ErrorPopup(textString)
00120             raise ValueError(textString)
00121         elif status_code == 992:
00122             RESTErrorPrint(response)
00123             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No authentication input (Basic or
User/PW)"
00124             ErrorPopup(textString)
00125             raise ValueError(textString)
00126         elif status_code == 993:
00127             RESTErrorPrint(response)
00128             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: input values could
not be coerced to arguments"
00129             ErrorPopup(textString)
00130             print(ValueError(textString))
00131         elif status_code == 994:
00132             RESTErrorPrint(response)
00133             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: server returned
no documents"
00134             ErrorPopup(textString)
00135             raise ValueError(textString)
00136         elif status_code == 1000:
00137             RESTErrorPrint(response)
00138             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Catastrophic Error"
00139             ErrorPopup(textString)
00140             raise SystemError(textString)
00141         elif status_code == 1001:
00142             RESTErrorPrint(response)
00143             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Main Function Error Break"
00144             raise SystemError(textString)
00145         elif status_code == 1100:
00146             RESTErrorPrint(response)
00147             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User has cancelled the program
execution"
00148             raise KeyboardInterrupt(textString)
00149         elif status_code == 1101:
00150             RESTErrorPrint(response)
00151             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User returned to main menu using
the exit button"
00152             print(textString)
00153         else:
00154             RESTErrorPrint(response)
00155             raise Exception(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Status Code = {status_code} | An unknown exception occurred")
```

Class Documentation

AuthUtil.AuthUtil Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatusListedOrModified
- file_name
- append_file
- keyPath
- filePath
- k
- keyFlag
- jsonDict
- passFlagUre
- passFlagCm
- outcomeText

Private Member Functions

- def __SetValues (self, values)
- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Detailed Description

Definition at line 30 of file AuthUtil.py.

Constructor & Destructor Documentation

def AuthUtil.AuthUtil.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object, which in this case means that it creates
a new window and displays it on screen.

Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 32 of file AuthUtil.py.

```
00032     def __init__(self):
00033 
00034         """
00035     The __init__ function is called when the class is instantiated.
00036     It sets up the initial state of the object, which in this case means that
it creates a new window and displays it on screen.
00037 
```

```
00038     Args:
00039         self: Represent the instance of the class
00040
00041     Returns:
00042         None
00043
00044     Doc Author:
00045         Willem van der Schans, Trelent AI
00046     """
00047         self.StandardStatus = None
00048         self.ListedOrModified = None
00049         self.file_name = None
00050         self.append_file = None
00051         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00052         self.filePath =
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Secu
rity")
00053         self.k = None
00054         self.keyFlag = True
00055         self.jsonDict = {}
00056         self.passFlagUre = False
00057         self.passFlagCm = False
00058         self.outcomeText = "Please input the plain text keys in the input boxes
above \n " \
00059                            "Submitting will overwrite any old values in an
unrecoverable manner."
00060
00061         if os.path.exists(self.filePath):
00062             pass
00063         else:
00064             if
os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")
):
00065                 os.mkdir(self.filePath)
00066             else:
00067
os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
00068                 os.mkdir(self.filePath)
00069
00070         if os.path.exists(self.keyPath):
00071             pass
00072         else:
00073             if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00074                 os.mkdir(self.keyPath)
00075             else:
00076                 os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00077                 os.mkdir(self.keyPath)
00078
00079         if
os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00080             try:
00081                 f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00082                 self.k = f.readline()
00083                 f.close()
00084             except Exception as e:
00085                 print(e)
00086                 RESTError(402)
00087                 raise SystemExit(402)
00088         else:
00089             self.k = Fernet.generate_key()
00090             f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00091             f.write(self.k)
00092             f.close()
00093
00094             try:
00095                 os.remove(self.filePath.joinpath("auth.json"))
00096             except Exception as e:
00097                 # Logging
00098                 print(
00099                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file -
This can be due to the file not existing. Continuing...")
00100                 pass
```

```
00101
00102             f = open(self.filePath.joinpath("auth.json"), "wb")
00103             f.close()
00104             self.keyFlag = False
00105
00106         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00107
00108         try:
00109
ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35
.av3ra3rvavcr3w"), 2)
00110         except Exception as e:
00111             # Logging
00112             print(
00113                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as
hidden. This is either a Permission error or Input Error. Continuing...")
00114             pass
00115
```

Member Function Documentation

def AuthUtil.AuthUtil.__CreateFrame (*self*) `[private]`

```
The __CreateFrame function creates the GUI layout for the Authentication Utility.
It is called by __init__ and returns a list of lists that contains all the elements
that will be displayed in the window.

Args:
self: Access the class attributes and methods

Returns:
A list of lists

Doc Author:
Trelent
```

Definition at line 236 of file AuthUtil.py.

```
00236     def __CreateFrame(self):
00237         """
00238     The __CreateFrame function creates the GUI layout for the Authentication
Utility.
00239     It is called by __init__ and returns a list of lists that contains all the
elements
00240     that will be displayed in the window.
00241
00242     Args:
00243         self: Access the class attributes and methods
00244
00245     Returns:
00246         A list of lists
00247
00248     Doc Author:
00249         Trelent
00250     """
00251         sg.theme('Default1')
00252
00253         line00 = [sg.HSeparator()]
00254
00255         line0 = [sg.Image(ImageLoader("logo.png")),
00256                  sg.Push(),
00257                  sg.Text("Authentication Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00258                  sg.Push(),
00259                  sg.Push()]
00260
00261         line1 = [sg.HSeparator()]
00262
00263         line2 = [sg.Push(),
00264                  sg.Text("Utah Real Estate Key: ", justification="center"),
```

```
00265                   sg.Push()]
00266
00267         line3 = [sg.Push(),
00268                  sg.Input(default_text="", key="-ureAuth-", disabled=False,
00269                           size=(40, 1)),
00270                  sg.Push()]
00271
00272         line4 = [sg.HSeparator()]
00273
00274         line5 = [sg.Push(),
00275                  sg.Text("Construction Monitor Key: ",
justification="center"),
00276                  sg.Push()]
00277
00278         line6 = [sg.Push(),
00279                  sg.Input(default text="", key="-cmAuth-", disabled=False,
00280                           size=(40, 1)),
00281                  sg.Push()]
00282
00283         line7 = [sg.HSeparator()]
00284
00285         line8 = [sg.Push(),
00286                  sg.Text(self.outcomeText, justification="center"),
00287                  sg.Push()]
00288
00289         line9 = [sg.HSeparator()]
00290
00291         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00292
00293         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10]
00294
00295         return layout
```

def AuthUtil.AuthUtil.__SetValues (*self*, *values*) `[private]`

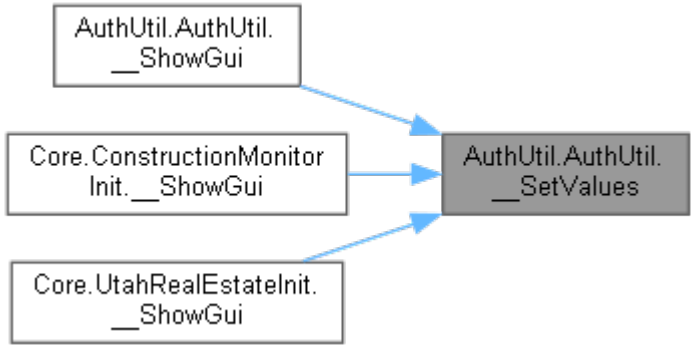
```
The __SetValues function is called when the user clicks on the &quot;OK&quot; button
in the window.
It takes a dictionary of values as an argument, and then uses those values to update
the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

Args:
self: Make the function a method of the class
values: Store the values that are entered into the form

Returns:
A dictionary of the values entered by the user

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 116 of file AuthUtil.py.

```
00116     def __SetValues(self, values):
00117
00118         """
00119     The __SetValues function is called when the user clicks on the &quot;OK&quot;
button in the window.
00120     It takes a dictionary of values as an argument, and then uses those values
to update
00121     the auth.json file with new keys for both Utah Real Estate and Construction
Monitor.
00122
00123     Args:
00124         self: Make the function a method of the class
00125         values: Store the values that are entered into the form
00126
00127     Returns:
00128         A dictionary of the values entered by the user
00129
00130     Doc Author:
00131         Willem van der Schans, Trelent AI
00132     """
00133         ureCurrent = None
```

```
00134         cmCurrent = None
00135         keyFile = None
00136
00137         fernet = Fernet(self.k)
00138
00139         try:
00140             f = open(self.filePath.joinpath("auth.json"), "r")
00141             keyFile = json.load(f)
00142             fileFlag = True
00143         except:
00144             fileFlag = False
00145
00146         if fileFlag:
00147             try:
00148                 ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00149             except Exception as e:
00150                 # Logging
00151                 print(
00152                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Utah Real Estate Key.
Continuing but this should be resolved if URE functionality will be accessed")
00153                 ureCurrent = None
00154
00155             try:
00156                 cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00157             except Exception as e:
00158                 # Logging
00159                 print(
00160                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Construction Monitor
Key. Continuing but this should be resolved if CM functionality will be accessed")
00161                 cmCurrent = None
00162
00163         if values["-ureAuth-"] != "":
00164             self.jsonDict.update(
00165                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00166             self.passFlagUre = True
00167         elif ureCurrent is not None:
00168             self.jsonDict.update(
00169                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00170             self.passFlagUre = True
00171         else:
00172             pass
00173
00174         if values["-cmAuth-"] != "":
00175             self.jsonDict.update(
00176                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00177             self.passFlagCm = True
00178         elif ureCurrent is not None:
00179             self.jsonDict.update(
00180                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00181             self.passFlagUre = True
00182         else:
00183             pass
00184
00185         if not self.passFlagUre and not self.passFlagCm:
00186             PopupWrapped("Please make sure you provide keys for both Utah Real
estate and Construction Monitor",
00187                          windowType="errorLarge")
00188         if self.passFlagCm and not self.passFlagUre:
00189             PopupWrapped("Please make sure you provide a key for Utah Real
estate", windowType="errorLarge")
00190         if not self.passFlagCm and self.passFlagUre:
00191             PopupWrapped("Please make sure you provide a key for Construction
Monitor", windowType="errorLarge")
00192         else:
00193             jsonOut = json.dumps(self.jsonDict, indent=4)
00194             f = open(self.filePath.joinpath("auth.json"), "w")
00195             f.write(jsonOut)
00196
```

Here is the caller graph for this function:



def AuthUtil.AuthUtil.__ShowGui (*self*, *layout*, *text*) `[private]`

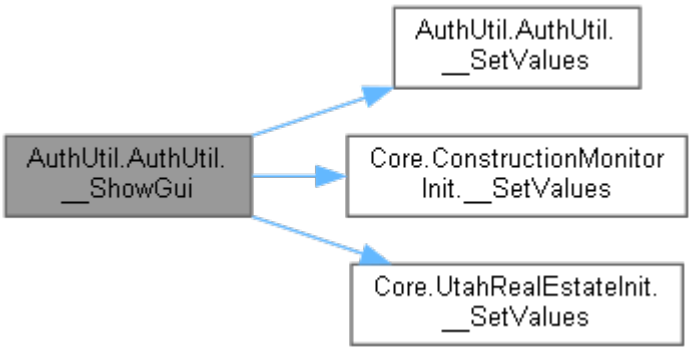
```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the window.

Args:
self: Represent the instance of the class
layout: Pass the layout of the gui to be displayed
text: Set the title of the window

Returns:
A window object
```

Definition at line 197 of file AuthUtil.py.

```
00197     def __ShowGui(self, layout, text):
00198
00199         """
00200     The __ShowGui function is a helper function that displays the GUI to the user.
00201     It takes in two arguments: layout and text. The layout argument is a list
of lists,
00202     which contains all the elements that will be displayed on screen. The text
argument
00203     is simply what will be displayed at the top of the window.
00204
00205     Args:
00206         self: Represent the instance of the class
00207         layout: Pass the layout of the gui to be displayed
00208         text: Set the title of the window
00209
00210     Returns:
00211         A window object
00212     """
00213         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00214                            finalize=True,
00215                            icon=ImageLoader("taskbar_icon.ico"))
00216
00217         while not self.passFlagUre or not self.passFlagCm:
00218             event, values = window.read()
00219
00220             if event == "Submit":
00221                 try:
00222                     self.__SetValues(values)
00223                 except Exception as e:
00224                     print(e)
00225                     RESTError(993)
00226                 finally:
00227                     pass
00228             elif event == sg.WIN_CLOSED or event == "Quit":
00229
00230                 break
00231             else:
00232                 pass
00233
00234         window.close()
```

```
00235
```

Here is the call graph for this function:



Member Data Documentation

AuthUtil.AuthUtil.append_file

Definition at line 50 of file AuthUtil.py.

AuthUtil.AuthUtil.file_name

Definition at line 49 of file AuthUtil.py.

AuthUtil.AuthUtil.filePath

Definition at line 52 of file AuthUtil.py.

AuthUtil.AuthUtil.jsonDict

Definition at line 55 of file AuthUtil.py.

AuthUtil.AuthUtil.k

Definition at line 53 of file AuthUtil.py.

AuthUtil.AuthUtil.keyFlag

Definition at line 54 of file AuthUtil.py.

AuthUtil.AuthUtil.keyPath

Definition at line 51 of file AuthUtil.py.

AuthUtil.AuthUtil.ListedOrModified

Definition at line 48 of file AuthUtil.py.

AuthUtil.AuthUtil.outcomeText

Definition at line 58 of file AuthUtil.py.

AuthUtil.AuthUtil.passFlagCm

Definition at line 57 of file AuthUtil.py.

AuthUtil.AuthUtil.passFlagUre

Definition at line 56 of file AuthUtil.py.

AuthUtil.AuthUtil.StandardStatus

Definition at line 47 of file AuthUtil.py.

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

- AuthUtil.py

BatchProcessing.BatchProcessorConstructionMonitor Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject)
- def FuncSelector (self)
- def ConstructionMonitorProcessor (self, valueObject)

Public Attributes

- dataframevalueObject

Private Attributes

- __numBatches__parameterDict
- __restDomain
- __headerDict
- __columnSelection
- __maxRequests
- __requestCount
- __requestCalls
- __dateTracker

Detailed Description

Definition at line 52 of file BatchProcessing.py.

Constructor & Destructor Documentation

def BatchProcessing.BatchProcessorConstructionMonitor.__init__ (*self*, *RestDomain*, *NumBatches*, *ParameterDict*, *HeaderDict*, *ColumnSelection*, *valueObject*)

```
The __init__ function is the constructor for a class. It is called when an object of
that class
is created, and it sets up the attributes of that object. In this case, we are setting
up our
object to have a dataframe attribute (which will be used to store all of our data),
as well as
attributes for each parameter in our ReST call.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to retrieve
ParameterDict: Pass in the parameters that will be used to make the api call
HeaderDict: Pass the header dictionary from the main function to this class
ColumnSelection: Determine which columns to pull from the api
valueObject: Pass in the value object that is used to determine what values are returned

Returns:
An object of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 54 of file BatchProcessing.py.

```
00054     def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict,
ColumnSelection, valueObject):
00055
00056         """
00057     The __init__ function is the constructor for a class. It is called when an
object of that class
00058     is created, and it sets up the attributes of that object. In this case, we
are setting up our
00059     object to have a dataframe attribute (which will be used to store all of our
data), as well as
00060     attributes for each parameter in our ReST call.
00061
00062     Args:
00063         self: Represent the instance of the class
00064         RestDomain: Specify the domain of the rest api
00065         NumBatches: Determine how many batches of data to retrieve
00066         ParameterDict: Pass in the parameters that will be used to make the api
call
00067         HeaderDict: Pass the header dictionary from the main function to this
class
00068         ColumnSelection: Determine which columns to pull from the api
00069         valueObject: Pass in the value object that is used to determine what
values are returned
00070
00071     Returns:
00072         An object of the class
00073
00074     Doc Author:
00075         Willem van der Schans, Trelent AI
00076     """
00077         self.dataframe = None
00078         self.__numBatches = NumBatches
00079         self.__parameterDict = ParameterDict
00080         self.__restDomain = RestDomain
00081         self.__headerDict = HeaderDict
00082         self.__columnSelection = ColumnSelection
00083         self.valueObject = valueObject
00084         self.__maxRequests = 10000
00085         self.__requestCount = math.ceil(self.__numBatches /
(self.__maxRequests / int(self.__parameterDict['size'])))
00086         self.__requestCalls = math.ceil(self.__maxRequests /
int(self.__parameterDict['size']))
00087         self.__dateTracker = None
00088
```

Member Function Documentation

def BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor (*self*, *valueObject*)

```
The ConstructionMonitorProcessor function will use requests to get data from
ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called
__df (which is local). This
process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at which point __df will contain all

Args:
self: Represent the instance of the object itself
valueObject: Update the progress bar in the gui

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

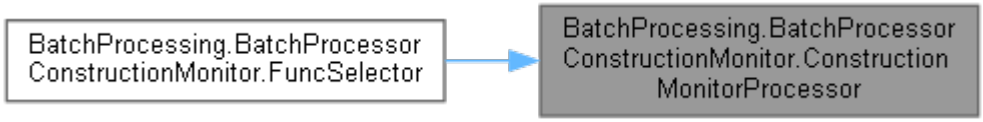
Definition at line 105 of file BatchProcessing.py.

```
00105     def ConstructionMonitorProcessor(self, valueObject):
00106         """
```

```
00107     The ConstructionMonitorProcessor function will use requests to get data from
00108        ConstructionMontior.com's ReST API and store it into a pandas DataFrame
object called __df (which is local). This
00109        process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at which point __df will contain all
00110
00111     Args:
00112         self: Represent the instance of the object itself
00113         valueObject: Update the progress bar in the gui
00114
00115     Returns:
00116         A dataframe
00117
00118     Doc Author:
00119         Willem van der Schans, Trelent AI
00120     """
00121         __df = None
00122         for callNum in range(0, self.__requestCount):
00123             self.__parameterDict["from"] = 0
00124
00125             if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00126                   batchNum = self.  requestCalls
00127                 if __df is None:
00128                     self.__dateTracker = str(date.today())
00129                 else:
00130                     self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00131             elif self.__requestCount == 1:
00132                 __batchNum = self.__numBatches
00133                 self.__dateTracker = str(date.today())
00134             else:
00135                 __batchNum = self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])) - (
00136                         self.__requestCount - 1)
00137                 self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00138
00139             self.__parameterDict['dateEnd'] = self.__dateTracker
00140
00141             for record in range(0, int(math.ceil(__batchNum))):
00142                 if record != 0:
00143                     self.__parameterDict["from"] = record *
int(self.__parameterDict["size"])
00144
00145                 response = requests.post(url=self.__restDomain,
00146                                          headers=self.__headerDict,
00147                                          json=self.__parameterDict)
00148
00149                 counter = 0
00150                 try:
00151                     response = response.json()['hits']['hits']
00152                 except KeyError as e:
00153                     # Logging
00154                     print(
00155                         f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server
Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict}
| Headers = {self.__headerDict}")
00156                     continue
00157
00158                 valueObject.setValue(valueObject.getValue() + 1)
00159
00160                 if record == 0 and callNum == 0:
00161                     __df = pd.json_normalize(response[counter]["_source"])
00162                     __df["id"] = response[counter]['_id']
00163                     __df["county"] =
response[counter]["_source"]['county']['county_name']
00164                     counter += 1
00165
00166                 for i in range(counter, len(response)):
00167                     __tdf = pd.json_normalize(response[i]["_source"])
00168                     __tdf["id"] = response[i]['_id']
00169                     __tdf["county"] =
response[i]["_source"]['county']['county_name']
00170                     __df = pd.concat([__df, __tdf], ignore_index=True)
00171
00172         if self.__columnSelection is not None:
```

```
00173             __col_list = StringToList(self.__columnSelection)
00174             __col_list.append("id")
00175             __col_list.append("county")
00176         else:
00177             pass
00178 
00179         self.dataframe = __df
00180         valueObject.setValue(-999)
00181 
00182 
```

Here is the caller graph for this function:



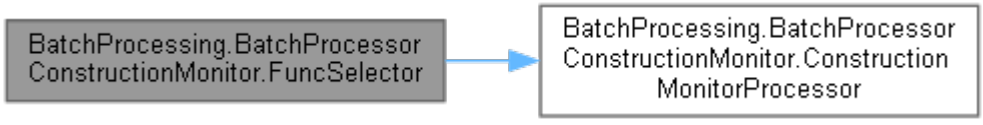
def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (*self*)

```
The FuncSelector function is a function that takes the valueObject and passes it to
the ConstructionMonitorProcessor function.
The ConstructionMonitorProcessor function then uses this valueObject to determine which
of its functions should be called.

Args:
self: Represent the instance of the class

Returns:
The result of the constructionmonitorprocessor function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 89 of file BatchProcessing.py.

```
00089     def FuncSelector(self):
00090         """
00091     The FuncSelector function is a function that takes the valueObject and passes
it to the ConstructionMonitorProcessor function.
00092     The ConstructionMonitorProcessor function then uses this valueObject to
determine which of its functions should be called.
00093 
00094     Args:
00095         self: Represent the instance of the class
00096 
00097     Returns:
00098         The result of the constructionmonitorprocessor function
00099 
00100     Doc Author:
00101         Willem van der Schans, Trelent AI
00102     """
00103         self.ConstructionMonitorProcessor(self.valueObject)
00104 
```

Here is the call graph for this function:



Member Data Documentation

BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection `[private]`

Definition at line 82 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker [private]

Definition at line 87 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__headerDict [private]

Definition at line 81 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests [private]

Definition at line 84 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__numBatches [private]

Definition at line 78 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict [private]

Definition at line 79 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls [private]

Definition at line 86 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__requestCount [private]

Definition at line 85 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__restDomain [private]

Definition at line 80 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.dataframe

Definition at line 77 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.valueObject

Definition at line 83 of file BatchProcessing.py.

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

- BatchProcessing.py

BatchProcessing.BatchProcessorUtahRealEstate Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject)
- def FuncSelector (self)
- def BatchProcessingUtahRealestateCom (self, valueObject)

Public Attributes

- dataframevalueObject

Private Attributes

- __numBatches__parameterString
- __restDomain
- __headerDict

Detailed Description

Definition at line 183 of file BatchProcessing.py.

Constructor & Destructor Documentation

def BatchProcessing.BatchProcessorUtahRealEstate.__init__ (*self*, *RestDomain*, *NumBatches*, *ParameterString*, *HeaderDict*, *valueObject*)

```
The __init__ function is the constructor for a class. It is called when an object of
that class
is instantiated, and it sets up the attributes of that object. In this case, we are
setting up
the dataframe attribute to be None (which will be set later), and we are also setting
up some
other attributes which will help us make our API calls.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to pull from the api
ParameterString: Pass the parameters to the rest api
HeaderDict: Pass in the header information for the api call
valueObject: Create a dataframe from the json response

Returns:
The instance of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 185 of file BatchProcessing.py.

```
00185     def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict,
valueObject):
00186         """
00187     The __init__ function is the constructor for a class. It is called when an
object of that class
00188     is instantiated, and it sets up the attributes of that object. In this case,
we are setting up
00189     the dataframe attribute to be None (which will be set later), and we are also
setting up some
00190     other attributes which will help us make our API calls.
```

```
00191
00192     Args:
00193         self: Represent the instance of the class
00194         RestDomain: Specify the domain of the rest api
00195         NumBatches: Determine how many batches of data to pull from the api
00196         ParameterString: Pass the parameters to the rest api
00197         HeaderDict: Pass in the header information for the api call
00198         valueObject: Create a dataframe from the json response
00199
00200     Returns:
00201         The instance of the class
00202
00203     Doc Author:
00204         Willem van der Schans, Trelent AI
00205     """
00206         self.dataframe = None
00207         self.__numBatches = NumBatches
00208         self.__parameterString = ParameterString
00209         self.__restDomain = RestDomain
00210         self.__headerDict = HeaderDict
00211         self.valueObject = valueObject
00212
```

Member Function Documentation

def BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom (*self*, *valueObject*)

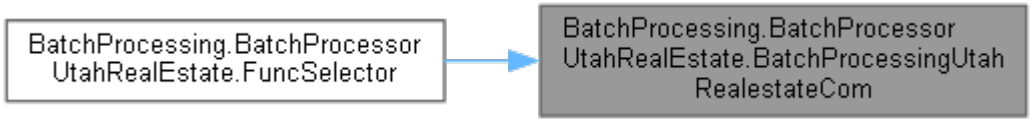
```
The BatchProcessingUtahRealestateCom function is a function that takes in the
valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary
information for this
function to work properly. The BatchProcessingUtahRealestateCom function will then use
requests to get data from
UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df
(which is local). This
process will be repeated until all the data has been collected from UtahRealestate.com's
ReST API, at which point __df will contain all

Args:
self: Represent the instance of the class
valueObject: Pass the value of a progress bar to the function

Returns:
A dataframe of the scraped data

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 230 of file BatchProcessing.py.

```
00230     def BatchProcessingUtahRealestateCom(self, valueObject):
00231         """
00232     The BatchProcessingUtahRealestateCom function is a function that takes in
the valueObject and uses it to
00233         update the progress bar. It also takes in self, which contains all the
necessary information for this
00234         function to work properly. The BatchProcessingUtahRealestateCom function
will then use requests to get data from
00235         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
called __df (which is local). This
00236         process will be repeated until all the data has been collected from
UtahRealestate.com's ReST API, at which point __df will contain all
00237
00238     Args:
00239         self: Represent the instance of the class
00240         valueObject: Pass the value of a progress bar to the function
00241
00242     Returns:
00243         A dataframe of the scraped data
```

```
00244
00245     Doc Author:
00246         Willem van der Schans, Trelent AI
00247     """
00248         __df = pd.DataFrame()
00249
00250         for batch in range(self.__numBatches):
00251
00252             if batch == 0:
00253                 response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00254                                             headers=self.__headerDict)
00255
00256                 response_temp = response.json()
00257                 __df = pd.json_normalize(response_temp, record_path=['value'])
00258
00259             else:
00260                 response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
200}",
00261                                             headers=self.__headerDict)
00262
00263                 response_temp = response.json()
00264                 response_temp = pd.json_normalize(response_temp,
record_path=['value'])
00265                 __df = pd.concat([__df, response_temp], ignore_index=True)
00266
00267             valueObject.setValue(valueObject.getValue() + 1)
00268
00269         self.dataframe = __df
00270         valueObject.setValue(-999)
```

Here is the caller graph for this function:



def BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector (*self*)

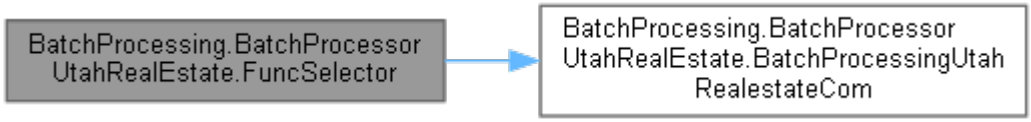
```
The FuncSelector function is a function that takes the valueObject as an argument and
then calls the appropriate
function based on what was selected in the dropdown menu.  The valueObject is passed
to each of these functions
so that they can access all of its attributes.

Args:
self: Represent the instance of the class

Returns:
The function that is selected by the user

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 213 of file BatchProcessing.py.

```
00213     def FuncSelector(self):
00214         """
00215     The FuncSelector function is a function that takes the valueObject as an
argument and then calls the appropriate
00216         function based on what was selected in the dropdown menu.  The
valueObject is passed to each of these functions
00217         so that they can access all of its attributes.
00218
00219     Args:
00220         self: Represent the instance of the class
00221
00222     Returns:
00223         The function that is selected by the user
00224
00225     Doc Author:
00226         Willem van der Schans, Trelent AI
```

```
00227     """
00228         self.BatchProcessingUtahRealestateCom(self.valueObject)
00229
```

Here is the call graph for this function:



Member Data Documentation

BatchProcessing.BatchProcessorUtahRealEstate.__headerDict [private]

Definition at line 210 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__numBatches [private]

Definition at line 207 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__parameterString [private]

Definition at line 208 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__restDomain [private]

Definition at line 209 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.dataframe

Definition at line 206 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.valueObject

Definition at line 211 of file BatchProcessing.py.

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

- BatchProcessing.py

BatchProgressGUI.BatchProgressGUI Class Reference

Public Member Functions

- def __init__ (self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None)
- def BatchGuiShow (self)
- def CreateProgressLayout (self)
- def createGui (self, Sourcetype)
- def ProgressUpdater (self, valueObj)
- def TimeUpdater (self, start_time)
- def ValueChecker (self, ObjectVal)

Public Attributes

dataframePrivate Attributes

- __parameterDict__restDomain
- __headerDict
- __columnSelection
- __type
- __layout
- __batches
- __window
- __batch_counter

Detailed Description

Definition at line 30 of file BatchProgressGUI.py.

Constructor & Destructor Documentation

def BatchProgressGUI.BatchProgressGUI.__init__ (*self*, *BatchesNum*, *RestDomain*, *ParameterDict*, *HeaderDict*, *Type*, *ColumnSelection* = `None`)

```
The __init__ function is the first function that gets called when an object of this
class is created.
It initializes all the variables and sets up a layout for the GUI. It also creates a
window to display
the dataframe in.

Args:
self: Represent the instance of the class
BatchesNum: Determine the number of batches that will be created
RestDomain: Specify the domain of the rest api
ParameterDict: Pass the parameters of the request to the class
HeaderDict: Store the headers of the dataframe
Type: Determine the type of dataframe that is being created
ColumnSelection: Select the columns to be displayed in the gui

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 32 of file BatchProgressGUI.py.

```
00032     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type,
ColumnSelection=None):
00033
00034         """
00035     The __init__ function is the first function that gets called when an object
of this class is created.
00036     It initializes all the variables and sets up a layout for the GUI. It also
creates a window to display
00037     the dataframe in.
00038
00039     Args:
00040         self: Represent the instance of the class
00041         BatchesNum: Determine the number of batches that will be created
00042         RestDomain: Specify the domain of the rest api
00043         ParameterDict: Pass the parameters of the request to the class
00044         HeaderDict: Store the headers of the dataframe
00045         Type: Determine the type of dataframe that is being created
00046         ColumnSelection: Select the columns to be displayed in the gui
00047
00048     Returns:
00049         Nothing
00050
00051     Doc Author:
00052         Willem van der Schans, Trelent AI
00053     """
00054         self.__parameterDict = ParameterDict
00055         self.__restDomain = RestDomain
00056         self.__headerDict = HeaderDict
00057         self.__columnSelection = ColumnSelection
00058         self.__type = Type
00059         self.dataframe = None
00060
00061         self.__layout = None
00062         self.__batches = BatchesNum
00063         self.__window = None
00064         self.__batch_counter = 0
00065
```

Member Function Documentation

def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (*self*)

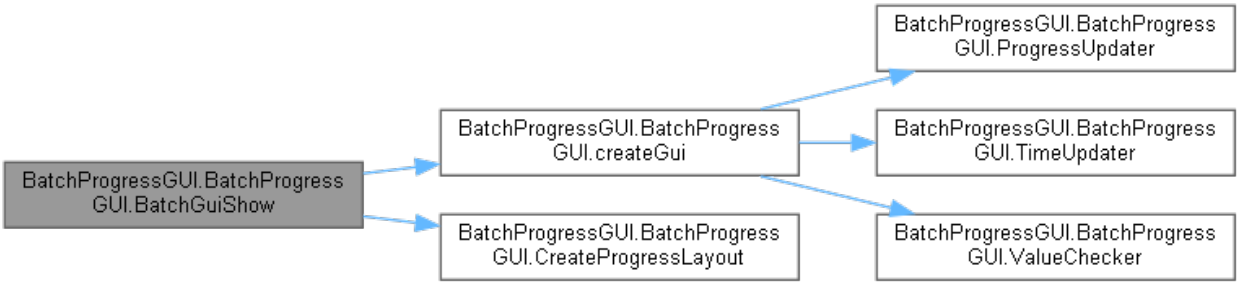
```
The BatchGuiShow function is called by the BatchGui function. It creates a progress
bar layout and then calls the createGui function to create a GUI for batch processing.

Args:
self: Represent the instance of the class

Returns:
The __type of the batchgui class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 66 of file BatchProgressGUI.py.

```
00066     def BatchGuiShow(self):
00067         """
00068     The BatchGuiShow function is called by the BatchGui function. It creates a
progress bar layout and then calls the createGui function to create a GUI for batch
processing.
00069
00070     Args:
00071         self: Represent the instance of the class
00072
00073     Returns:
00074         The __type of the batchgui class
00075
00076     Doc Author:
00077         Willem van der Schans, Trelent AI
00078     """
```

```
00079         self.CreateProgressLayout()
00080         self.createGui(self.__type)
00081
```

Here is the call graph for this function:



def BatchProgressGUI.BatchProgressGUI.createGui (*self*, *Sourcetype*)

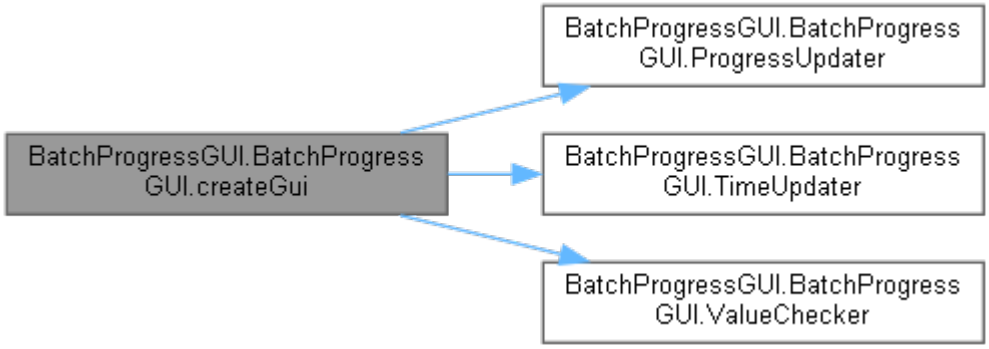
```
The createGui function is the main function that creates the GUI.
It takes in a type parameter which determines what kind of batch processor to use.
The createGui function then sets up all the variables and objects needed for
the program to run, including: window, start_time, update_text, valueObj
(DataTransfer),
processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
and threading objects for TimeUpdater and ValueChecker functions. The createGui
function also starts these threads.

Args:
self: Access the object itself
Sourcetype: Determine which batch processor to use

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 117 of file BatchProgressGUI.py.

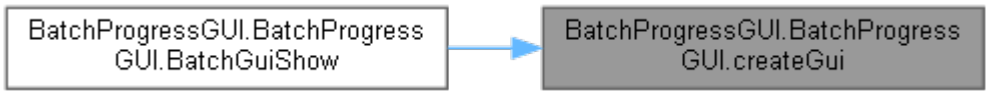
```
00117     def createGui(self, Sourcetype):
00118
00119         """
00120     The createGui function is the main function that creates the GUI.
00121     It takes in a type parameter which determines what kind of batch processor
to use.
00122     The createGui function then sets up all the variables and objects needed for
00123     the program to run, including: window, start_time, update_text, valueObj
(DataTransfer),
00124     processorObject (BatchProcessorConstructionMonitor or
BatchProcessorUtahRealestate),
00125     and threading objects for TimeUpdater and ValueChecker functions. The
createGui function also starts these threads.
00126
00127     Args:
00128         self: Access the object itself
00129         Sourcetype: Determine which batch processor to use
00130
00131     Returns:
00132         The dataframe
00133
00134     Doc Author:
00135         Willem van der Schans, Trelent AI
00136     """
00137         self.__window = sg.Window('Progress', self.__layout, finalize=True,
icon=ImageLoader("taskbar_icon.ico"))
00138
00139         start_time = datetime.datetime.now().replace(microsecond=0)
00140         update_text = f"Batch {0} completed"
00141         self.__window['--progress_text--'].update(update_text)
00142         self.__window['--progress_bar--'].update(0)
00143         self.__window['--time_est--'].update("Est time needed 00:00:00")
00144
00145         valueObj = DataTransfer()
```

```
00146         valueObj.setValue(0)
00147
00148         if Sourcetype == "construction_monitor":
00149
00150             processorObject =
BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00151
NumBatches=self.__batches,
00152
ParameterDict=self.__parameterDict,
00153
HeaderDict=self.__headerDict,
00154
ColumnSelection=self.__columnSelection,
00155
valueObject=valueObj)
00156         elif Sourcetype == "utah_real_estate":
00157             processorObject =
BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00158
NumBatches=self.__batches,
00159
ParameterString=self.__parameterDict,
00160
HeaderDict=self.__headerDict,
00161
valueObject=valueObj)
00162
00163         threading.Thread(target=self.TimeUpdater,
00164                          args=(start_time,),
00165                          daemon=True).start()
00166         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")
00167
00168         batchFuncThread =
threading.Thread(target=processorObject.FuncSelector,
00169                                                daemon=False)
00170         batchFuncThread.start()
00171         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
00172         threading.Thread(target=self.ValueChecker,
00173                          args=(valueObj,),
00174                          daemon=False).start()
00175         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")
00176
00177         while True:
00178
00179             self.ProgressUpdater(valueObj)
00180
00181             if valueObj.getValue() == -999:
00182                 break
00183
00184             window, event, values = sg.read_all_windows()
00185             if event.startswith('update'):
00186                 __key_to_update = event[len('update'):]
00187                 window[__key_to_update].update(values[event])
00188                 window.refresh()
00189                 pass
00190
00191             if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00192                 break
00193
00194             time.sleep(0.1)
00195
00196         self.dataframe = processorObject.dataframe
00197         self.__window.close()
00198
00199         PopupWrapped(text="Api Request Completed", windowType="notice")
00200
```

Here is the call graph for this function:



Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (*self*)

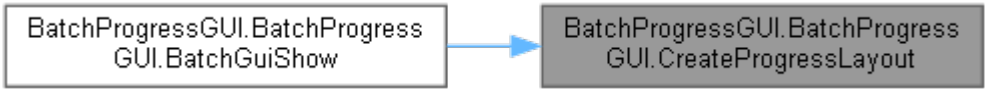
```
The CreateProgressLayout function creates the layout for the progress window.
The function takes in self as a parameter and returns nothing.

Parameters:
    self (object): The object that is calling this function.

Args:
self: Access the class variables and methods

Returns:
A list of lists

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 82 of file BatchProgressGUI.py.

```
00082     def CreateProgressLayout(self):
00083 
00084         """
00085     The CreateProgressLayout function creates the layout for the progress window.
00086         The function takes in self as a parameter and returns nothing.
00087 
00088         Parameters:
00089             self (object): The object that is calling this function.
00090 
00091     Args:
00092         self: Access the class variables and methods
00093 
00094     Returns:
00095         A list of lists
00096 
00097     Doc Author:
00098         Willem van der Schans, Trelent AI
00099     """
00100         sg.theme('Default1')
00101 
00102         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
00103                    sg.Push()]
00104 
00105         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
00106                    sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]
00107 
00108         __Line3 = [
00109             sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
00110                            key='--progress_bar--')]
00111 
```

```
00112
00113         layout = [__Line1, __Line2, __Line3]
00114
00115         self.__layout = layout
00116
```

Here is the caller graph for this function:



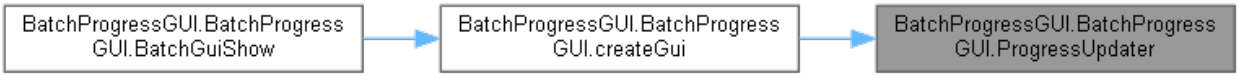
def BatchProgressGUI.BatchProgressGUI.ProgressUpdater (*self*, *valueObj*)

```
The ProgressUpdater function is a callback function that updates the progress bar and
text
in the GUI. It takes in one argument, which is an object containing information about
the
current batch number. The ProgressUpdater function then checks if this value has changed
from
the last time it was called (i.e., if we are on a new batch). If so, it updates both
the progress
bar and text with this new information.

Args:
self: Make the progressupdater function an instance method
valueObj: Get the current value of the batch counter

Returns:
The value of the batch counter

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 201 of file BatchProgressGUI.py.

```
00201     def ProgressUpdater(self, valueObj):
00202         """
00203     The ProgressUpdater function is a callback function that updates the progress
bar and text
00204     in the GUI. It takes in one argument, which is an object containing information
about the
00205     current batch number. The ProgressUpdater function then checks if this value
has changed from
00206     the last time it was called (i.e., if we are on a new batch). If so, it updates
both the progress
00207     bar and text with this new information.
00208
00209     Args:
00210         self: Make the progressupdater function an instance method
00211         valueObj: Get the current value of the batch counter
00212
00213     Returns:
00214         The value of the batch counter
00215
00216     Doc Author:
00217         Willem van der Schans, Trelent AI
00218     """
00219         if valueObj.getValue() != self.__batch_counter:
00220             self.__batch_counter = valueObj.getValue()
00221
00222             __update_text = f"Batch {self.__batch_counter}/{self.__batches}
completed"
00223
00224             self.__window.write_event_value('update--progress_bar--',
self.__batch_counter)
00225             self.__window.write_event_value('update--progress_text--',
__update_text)
00226         else:
00227             pass
00228
```

Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.TimeUpdater (*self*, *start_time*)

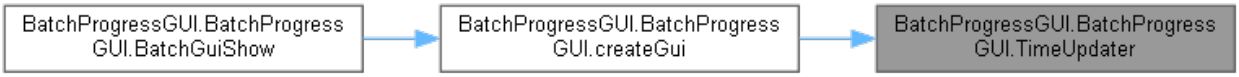
```
The TimeUpdater function is a thread that updates the time elapsed and estimated time
needed to complete
the current batch. It does this by reading the start_time variable passed in, getting
the current time,
calculating how much time has passed since start_time was set and then updating a timer
string with that value.
It then calculates an estimation of how long it will take to finish all batches based
on how many batches have been completed so far.

Args:
self: Make the function a method of the class
start_time: Get the time when the function is called

Returns:
A string that is updated every 0

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 229 of file BatchProgressGUI.py.

```
00229     def TimeUpdater(self, start_time):
00230
00231         """
00232     The TimeUpdater function is a thread that updates the time elapsed and
estimated time needed to complete
00233     the current batch. It does this by reading the start_time variable passed
in, getting the current time,
00234     calculating how much time has passed since start_time was set and then
updating a timer string with that value.
00235     It then calculates an estimation of how long it will take to finish all batches
based on how many batches have been completed so far.
00236
00237     Args:
00238         self: Make the function a method of the class
00239         start_time: Get the time when the function is called
00240
00241     Returns:
00242         A string that is updated every 0
00243
00244     Doc Author:
00245         Willem van der Schans, Trelent AI
00246     """
00247         while True:
00248             if self.__batch_counter < self.__batches:
00249
00250                 __current_time =
datetime.datetime.now().replace(microsecond=0)
00251
00252                 __passed_time = __current_time - start_time
00253
00254                 __timer_string = f"Time Elapsed {__passed_time}"
00255
00256                 try:
00257                     self.__window.write_event_value('update--timer--',
__timer_string)
00258                 except AttributeError as e:
00259                     print(
00260                         f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error,
this is okay if the display looks good, this exception omits fatal crashes due to an
aesthetic error")
00261                     break
00262
00263                 __passed_time = __passed_time.total_seconds()
00264
00265                 try:
00266                     __time_est = datetime.timedelta(
```

```
00267                           seconds=(__passed_time * (self.__batches /
self.__batch_counter) - __passed_time)).seconds
00268                   except:
00269                       __time_est = datetime.timedelta(
00270                           seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00271
00272                   __time_est = time.strftime('%H:%M:%S',
time.gmtime(__time_est))
00273
00274                   __end_string = f"Est time needed {__time_est}"
00275                   self.__window.write_event_value('update--time_est--',
__end_string)
00276               else:
00277                   __end_string = f"Est time needed 00:00:00"
00278                   self.  window.write event value('update--time est--',
__end_string)
00279               time.sleep(0.25)
00280
```

Here is the caller graph for this function:



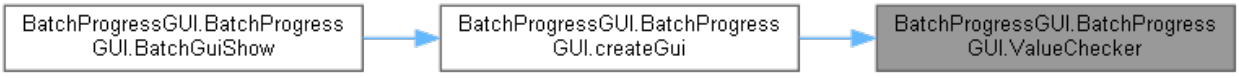
def BatchProgressGUI.BatchProgressGUI.ValueChecker (*self*, *ObjectVal*)

```
The ValueChecker function is a thread that checks the value of an object.
It will check if the value has changed, and if it has, it will return True.
If not, then it returns False.

Args:
self: Represent the instance of the class
ObjectVal: Get the value of the object

Returns:
True if the value of the object has changed, and false if it hasn't

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 281 of file BatchProgressGUI.py.

```
00281     def ValueChecker(self, ObjectVal):
00282         """
00283     The ValueChecker function is a thread that checks the value of an object.
00284         It will check if the value has changed, and if it has, it will return
True.
00285         If not, then it returns False.
00286
00287     Args:
00288         self: Represent the instance of the class
00289         ObjectVal: Get the value of the object
00290
00291     Returns:
00292         True if the value of the object has changed, and false if it hasn't
00293
00294     Doc Author:
00295         Willem van der Schans, Trelent AI
00296     """
00297         while True:
00298             time.sleep(0.3)
00299             if self.__batch_counter != ObjectVal.getValue():
00300                 self.__batch_counter = ObjectVal.getValue()
00301                 return True
00302             else:
00303                 return False
```

Here is the caller graph for this function:



Member Data Documentation

BatchProgressGUI.BatchProgressGUI.__batch_counter [private]

Definition at line 64 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__batches [private]

Definition at line 62 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__columnSelection [private]

Definition at line 57 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__headerDict [private]

Definition at line 56 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__layout [private]

Definition at line 61 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__parameterDict [private]

Definition at line 54 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__restDomain [private]

Definition at line 55 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__type [private]

Definition at line 58 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__window [private]

Definition at line 63 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.dataframe

Definition at line 59 of file BatchProgressGUI.py.

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

- BatchProgressGUI.py

Core.Cencus Class Reference

Public Member Functions

- def __init__ (self, state_arg=None, year_arg=None)

Public Attributes

- state_argyear_arg
- uiString
- link

Private Member Functions

- def __showUi (self)
- def __dataGetter (self)

Detailed Description

Definition at line 12 of file CFBP/Core.py.

Constructor & Destructor Documentation

def Core.Cencus.__init__ (*self*, *state_arg* = None, *year_arg* = None)

```
The __init__ function is called when the class is instantiated.
It's job is to initialize the object with some default values, and do any other setup
that might be necessary.
The __init__ function can take arguments, but it doesn't have to.

Args:
self: Represent the instance of the class
state_arg: Set the state_arg attribute of the class
year_arg: Set the year of data to be retrieved

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 14 of file CFBP/Core.py.

```
00014     def __init__(self, state_arg=None, year_arg=None):
00015         """
00016     The __init__ function is called when the class is instantiated.
00017     It's job is to initialize the object with some default values, and do any
other setup that might be necessary.
00018     The __init__ function can take arguments, but it doesn't have to.
00019
00020     Args:
00021         self: Represent the instance of the class
00022         state_arg: Set the state_arg attribute of the class
00023         year_arg: Set the year of data to be retrieved
00024
00025     Returns:
00026         A popupwrapped object
00027
00028     Doc Author:
00029         Willem van der Schans, Trelent AI
00030     """
00031         self.state_arg = state_arg
00032         self.year_arg = year_arg
```

```
00033         self.uiString = None
00034         self.link = None
00035 
00036         self.__showUi()
00037         print(self.link)
00038         F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00039         self.uiString = (
00040             f"ffiec.cfpb.gov (Mortgage API) request Completed \n
{self.year_arg} data retrieved \n Data Saved at {F.getPath()}")
00041 
00042         PopupWrapped(text=self.uiString, windowType="noticeLarge")
00043 
```

Member Function Documentation

def Core.Cencus.__dataGetter (*self*) `[private]`

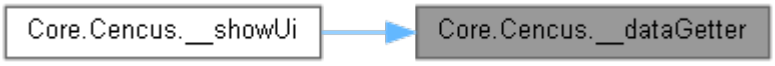
```
The   dataGetter function is a private function that gets the data from the CFPB API.
It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for
the API call.

Args:
self: Represent the instance of the class

Returns:
A response object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 72 of file CFBP/Core.py.

```
00072     def __dataGetter(self):
00073         """
00074     The __dataGetter function is a private function that gets the data from the
CFPB API.
00075     It takes no arguments, but uses self.state_arg and self.year_arg to create
a URL for the API call.
00076 
00077     Args:
00078         self: Represent the instance of the class
00079 
00080     Returns:
00081         A response object
00082 
00083     Doc Author:
00084         Willem van der Schans, Trelent AI
00085     """
00086         arg_dict_bu = locals()
00087 
00088         link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
00089 
00090         if self.state_arg is None:
00091             self.state_arg = "UT"
00092         else:
00093             pass
00094 
00095         if self.year_arg is None:
00096             self.year_arg = str(date.today().year - 1)
00097         else:
00098             pass
00099 
00100         passFlag = False
00101 
00102         while not passFlag:
00103 
00104             self.link =
"https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
+ f"&years={self.year_arg}"
00105 
00106             response = requests.get(self.link)
```

```
00107
00108             if response.status_code == 400:
00109                 self.year_arg = int(self.year_arg) - 1
00110
00111             else:
00112                 passFlag = True
00113
00114         RESTError(response)
00115         raise SystemExit(0)
```

Here is the caller graph for this function:



def Core.Cencus.__showUi (*self*) `[private]`

```
The __showUi function is a function that creates a progress bar window.
The __showUi function takes class variables and returns a windowobj.


Args:
self: Represent the instance of the class

Returns:
The uiobj variable

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 44 of file CFBP/Core.py.

```
00044     def __showUi(self):
00045
00046         """
00047     The __showUi function is a function that creates a progress bar window.
00048     The __showUi function takes class variables and returns a windowobj.
00049
00050
00051     Args:
00052         self: Represent the instance of the class
00053
00054     Returns:
00055         The uiobj variable
00056
00057     Doc Author:
00058         Willem van der Schans, Trelent AI
00059     """
00060         uiObj = PopupWrapped(text="Cenus Request running",
windowType="progress", error=None)
00061
00062         threadGui = threading.Thread(target=self.__dataGetter,
00063                                      daemon=False)
00064         threadGui.start()
00065
00066         while threadGui.is_alive():
00067             uiObj.textUpdate()
00068             uiObj.windowPush()
00069         else:
00070             uiObj.stopWindow()
00071
```

Here is the call graph for this function:

Member Data Documentation

Core.Cencus.link

Definition at line 34 of file CFBP/Core.py.

Core.Cencus.state_arg

Definition at line 31 of file CFBP/Core.py.

Core.Cencus.uiString

Definition at line 33 of file CFBP/Core.py.

Core.Cencus.year_arg

Definition at line 32 of file CFBP/Core.py.

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

- CFBP/Core.py

Core.ConstructionMonitorInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- sizeSourceInclude
- dateStart
- dateEnd
- rest_domain
- auth_key
- ui_flag
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

Detailed Description

Definition at line 24 of file ConstructionMonitor/Core.py.

Constructor & Destructor Documentation

def Core.ConstructionMonitorInit.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the variables that will be used by other functions in this class.


Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 26 of file ConstructionMonitor/Core.py.

```
00026     def __init__(self):
00027 
00028         """
00029     The __init__ function is called when the class is instantiated.
00030     It sets up the variables that will be used by other functions in this class.
00031 
00032 
00033     Args:
00034         self: Represent the instance of the class
00035 
00036     Returns:
00037         None
00038 
00039     Doc Author:
```

```
00040         Willem van der Schans, Trelent AI
00041     """
00042         self.size = None
00043         self.SourceInclude = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.rest_domain = None
00047         self.auth_key = None
00048         self.ui_flag = None
00049         self.append_file = None
00050 
00051         passFlag = False
00052 
00053         while not passFlag:
00054             if
os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpat
h(
00055                     "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
00056 
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00057                     "Security").joinpath("auth.json")):
00058                 try:
00059                     f =
open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00060                         "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00061                     key = f.readline()
00062                     f.close()
00063                     f =
open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00064                         "Security").joinpath("auth.json"), "rb")
00065                     authDict = json.load(f)
00066                     fernet = Fernet(key)
00067                     self.auth_key =
fernet.decrypt(authDict["cm"]["auth"]).decode()
00068                     passFlag = True
00069                 except Exception as e:
00070                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | ConstructionMonitor/Core.py | Error = {e} | Auth.json not found
opening AuthUtil")
00071                     AuthUtil()
00072             else:
00073                 AuthUtil()
00074 
00075         self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00076 
```

Member Function Documentation

def Core.ConstructionMonitorInit.__CreateFrame () `[static]`, `[private]`

```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the GUI window.
This is done by creating each line as a list and then appending it to another list which
will contain all lines.

Args:

Returns:
The layout for the gui

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 116 of file ConstructionMonitor/Core.py.

```
00116     def __CreateFrame():
00117 
00118         """
00119     The __CreateFrame function creates the GUI layout for the application.
```

```
00120         The function returns a list of lists that contains all the elements to
be displayed in the GUI window.
00121         This is done by creating each line as a list and then appending it to
another list which will contain all lines.
00122
00123     Args:
00124
00125     Returns:
00126         The layout for the gui
00127
00128     Doc Author:
00129         Willem van der Schans, Trelent AI
00130     """
00131         sg.theme('Default1')
00132
00133         line00 = [sg.HSeparator()]
00134
00135         line0 = [sg.Image(ImageLoader("logo.png")),
00136                  sg.Push(),
00137                  sg.Text("Construction Monitor Utility", font=("Helvetica",
12, "bold"), justification="center"),
00138                  sg.Push(),
00139                  sg.Push()]
00140
00141         line1 = [sg.HSeparator()]
00142
00143         line3 = [sg.Text("Start Date : ", size=(15, None),
justification="Right"),
00144                  sg.Input(default_text=(date.today() -
timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
00145                           size=(20, 1)),
00146                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-start_date-', target="-Cal-")]
00147
00148         line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00149                  sg.Input(default_text=date.today().strftime("%Y-%m-%d"),
key="-EndCal-",
00150                           size=(20, 1)),
00151                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-start_date-', target="-EndCal-")]
00152
00153         line5 = [sg.HSeparator()]
00154
00155         line6 = [sg.Push(),
00156                  sg.Text("File Settings", font=("Helvetica", 12, "bold"),
justification="center"),
00157                  sg.Push()]
00158
00159         line7 = [sg.HSeparator()]
00160
00161         line8 = [sg.Text("Appending File : ", size=(15, None),
justification="Right"),
00162                  sg.Input(default_text="", key="-AppendingFile-",
disabled=True,
00163                           size=(20, 1)),
00164                  sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
00165                                target="-AppendingFile-")]
00166
00167         line9 = [sg.HSeparator()]
00168
00169         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00170
00171         layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8,
line9, line10]
00172
00173         return layout
00174
```

def Core.ConstructionMonitorInit.__SetValues (*self*, *values*) `[private]`

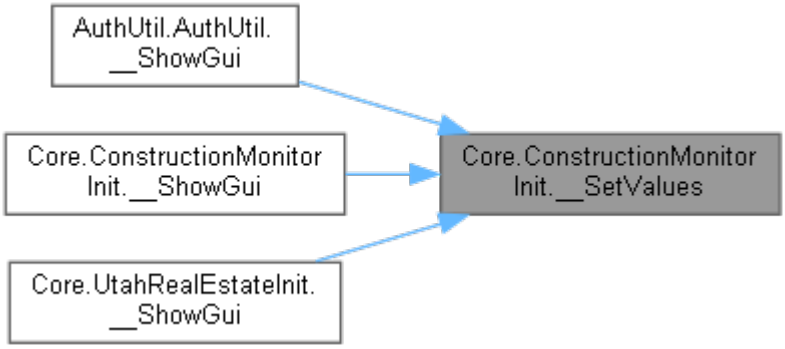
```
The __SetValues function is used to set the values of the variables that are used in
the __GetData function.
The __SetValues function takes a dictionary as an argument, and then sets each variable
based on what is passed into
```

```
the dictionary. The keys for this dictionary are defined by the user when they create
their own instance of this class.

Args:
self: Represent the instance of the class
values: Pass in the values from the ui

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 175 of file ConstructionMonitor/Core.py.

```
00175     def __SetValues(self, values):
00176
00177         """
00178     The __SetValues function is used to set the values of the variables that are
used in the __GetData function.
00179     The __SetValues function takes a dictionary as an argument, and then sets
each variable based on what is passed into
00180     the dictionary. The keys for this dictionary are defined by the user when
they create their own instance of this class.
00181
00182     Args:
00183         self: Represent the instance of the class
00184         values: Pass in the values from the ui
00185
00186     Returns:
00187         A dictionary of values
00188
00189     Doc Author:
00190         Willem van der Schans, Trelent AI
00191     """
00192         self.size = 1000
00193
00194         if values["-Cal-"] != "":
00195             self.dateStart = values["-Cal-"]
00196         else:
00197             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00198
00199         if values["-EndCal-"] != "":
00200             self.dateEnd = values["-EndCal-"]
00201         else:
00202             self.dateEnd = date.today().strftime("%Y-%m-%d")
00203
00204         self.rest_domain =
"https://api.constructionmonitor.com/v2/powersearch/?"
00205
00206         self.SourceInclude = None
00207
00208         if values["-append_file-"] != "":
00209             self.append_file = str(values["-append_file-"])
00210         else:
00211             self.append_file = None
00212
00213         self.ui_flag = True
00214
00215
```

Here is the caller graph for this function:



def Core.ConstructionMonitorInit.__ShowGui (*self*, *layout*, *text*) `[private]`

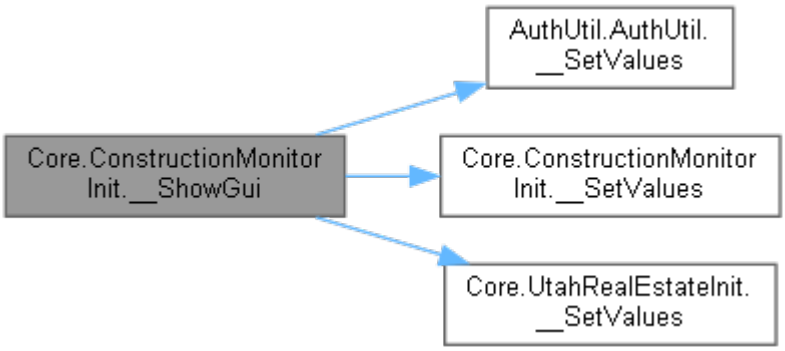
```
The __ShowGui function is the main function that creates and displays the GUI.
It takes in a layout, which is a list of lists containing all the elements to be displayed
on screen.
The text parameter specifies what title should appear at the top of the window.

Args:
self: Refer to the current instance of a class
layout: Determine what the gui will look like
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 77 of file ConstructionMonitor/Core.py.

```
00077     def __ShowGui(self, layout, text):
00078
00079         """
00080     The __ShowGui function is the main function that creates and displays the
GUI.
00081     It takes in a layout, which is a list of lists containing all the elements
to be displayed on screen.
00082     The text parameter specifies what title should appear at the top of the window.
00083
00084     Args:
00085         self: Refer to the current instance of a class
00086         layout: Determine what the gui will look like
00087         text: Set the title of the window
00088
00089     Returns:
00090         A dictionary of values
00091
00092     Doc Author:
00093         Willem van der Schans, Trelent AI
00094     """
00095         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00096                            finalize=True,
00097                            icon=ImageLoader("taskbar_icon.ico"))
00098
00099         while True:
00100             event, values = window.read()
00101
00102             if event == "Submit":
00103                 try:
00104                     self.__SetValues(values)
00105                     break
00106                 except Exception as e:
00107                     print(e)
00108                     RESTError(993)
00109                     raise SystemExit(933)
00110             elif event == sg.WIN_CLOSED or event == "Quit":
00111                 break
```

```
00112
00113         window.close()
00114
```

Here is the call graph for this function:



Member Data Documentation

Core.ConstructionMonitorInit.append_file

Definition at line 49 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.auth_key

Definition at line 47 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.dateEnd

Definition at line 45 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.dateStart

Definition at line 44 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.rest_domain

Definition at line 46 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.size

Definition at line 42 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.SourceInclude

Definition at line 43 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorInit.ui_flag

Definition at line 48 of file ConstructionMonitor/Core.py.

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

- ConstructionMonitor/Core.py

Core.ConstructionMonitorMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

dataframePrivate Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __siteClass__restDomain
- __headerDict
- __columnSelection
- __appendFile
- __parameterDict
- __search_id
- __record_val
- __batches
- __ui_flag

Detailed Description

Definition at line 216 of file ConstructionMonitor/Core.py.

Constructor & Destructor Documentation

def Core.ConstructionMonitorMain.__init__ (*self*, *siteClass*)

```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to run properly.


Args:
self: Represent the instance of the class
siteClass: Identify the site that is being used

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 218 of file ConstructionMonitor/Core.py.

```
00218     def __init__(self, siteClass):
00219 
00220         """
00221     The __init__ function is the first function that runs when an object of this
class is created.
00222     It sets up all the variables and functions needed for this class to run
properly.
```

```
00223
00224
00225     Args:
00226         self: Represent the instance of the class
00227         siteClass: Identify the site that is being used
00228
00229     Returns:
00230         Nothing
00231
00232     Doc Author:
00233         Willem van der Schans, Trelent AI
00234     """
00235         self.__siteClass = siteClass
00236         self.__restDomain = None
00237         self.__headerDict = None
00238         self.  columnSelection = None
00239         self.__appendFile = None
00240
00241         self.__parameterDict = {}
00242         self.__search_id = None
00243         self.__record_val = 0
00244         self.  batches = 0
00245
00246         self.__ui_flag = None
00247
00248         self.dataframe = None
00249
00250         try:
00251             self.mainFunc()
00252         except SystemError as e:
00253             if "Status Code = 1000 | Catastrophic Error" in str(getattr(e,
'message', repr(e))):
00254                 print(
00255                     f"ConstructionMonitor/Core.py | Error = {e} | Cooerced
SystemError in ConstructionMonitorMain class")
00256                 pass
00257         except AttributeError as e:
00258             # This allows for user cancellation of the program using the quit
button
00259             if "'NoneType' object has no attribute 'json'" in str(getattr(e,
'message', repr(e))):
00260                 RESTError(1101)
00261                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Error {e}")
00262                 pass
00263             elif e is not None:
00264                 print(
00265                     f"ConstructionMonitor/Core.py | Error = {e} |
Authentication Error | Please update keys in AuthUtil")
00266                 RESTError(401)
00267                 print(e)
00268                 pass
00269             else:
00270                 pass
00271         except Exception as e:
00272             print(e)
00273             RESTError(1001)
00274             raise SystemExit(1001)
00275
```

Member Function Documentation

def Core.ConstructionMonitorMain.__getCount (*self*) `[private]`

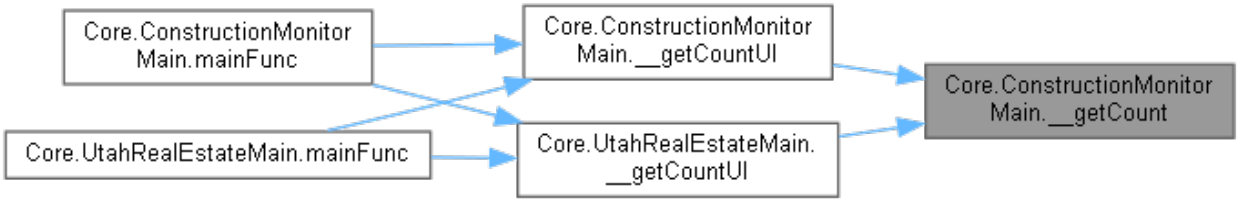
```
The __getCount function is used to get the total number of records that are returned
from a query.
This function is called by the __init__ function and sets the self.__record_val variable
with this value.

Args:
self: Represent the instance of the class
```

```
Returns:
The total number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 356 of file ConstructionMonitor/Core.py.

```
00356     def __getCount(self):
00357         """
00358     The   getCount function is used to get the total number of records that are
returned from a query.
00359     This function is called by the __init__ function and sets the
self.__record_val variable with this value.
00360
00361     Args:
00362         self: Represent the instance of the class
00363
00364     Returns:
00365         The total number of records in the database
00366
00367     Doc Author:
00368         Willem van der Schans, Trelent AI
00369     """
00370         __count_resp = None
00371
00372         try:
00373
00374             __temp_param_dict = copy.copy(self.__parameterDict)
00375
00376             __count_resp = requests.post(url=self.__restDomain,
00377                                          headers=self.__headerDict,
00378                                          json=__temp_param_dict)
00379
00380             if __count_resp.status_code != 200:
00381                 RESTError(__count_resp)
00382
00383         except requests.exceptions.Timeout as e:
00384             print(e)
00385             RESTError(790)
00386             raise SystemExit(790)
00387         except requests.exceptions.TooManyRedirects as e:
00388             print(e)
00389             RESTError(791)
00390             raise SystemExit(791)
00391         except requests.exceptions.MissingSchema as e:
00392             print(e)
00393             RESTError(1101)
00394         except requests.exceptions.RequestException as e:
00395             print(e)
00396             RESTError(405)
00397             raise SystemExit(405)
00398
00399         __count_resp = __count_resp.json()
00400
00401         self.__record_val = __count_resp["hits"]["total"]["value"]
00402
00403         del __count_resp, __temp_param_dict
00404
```

Here is the caller graph for this function:



def Core.ConstructionMonitorMain.__getCountUI (*self*) `[private]`

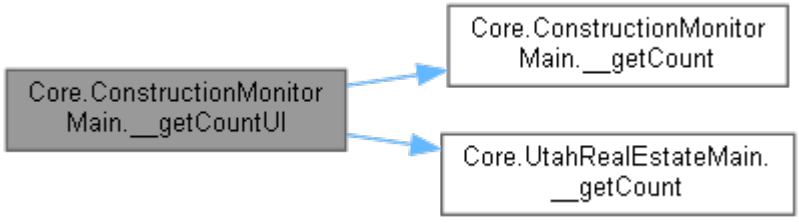
```
The __getCountUI function is a wrapper for the __getCount function.
```

```
It allows the user to run __getCount in a separate thread, so that they can continue
working while it runs.
The function will display a progress bar and update with text as it progresses through
its tasks.

Args:
self: Access the class variables and methods

Returns:
The count of the number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

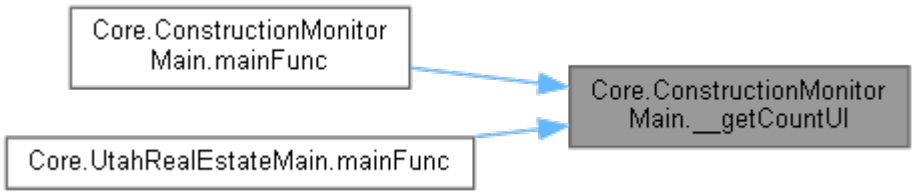
Definition at line 405 of file ConstructionMonitor/Core.py.

```
00405     def __getCountUI(self):
00406
00407         """
00408     The __getCountUI function is a wrapper for the __getCount function.
00409     It allows the user to run __getCount in a separate thread, so that they can
continue working while it runs.
00410     The function will display a progress bar and update with text as it progresses
through its tasks.
00411
00412     Args:
00413         self: Access the class variables and methods
00414
00415     Returns:
00416         The count of the number of records in the database
00417
00418     Doc Author:
00419         Willem van der Schans, Trelent AI
00420     """
00421         if self.__ui_flag:
00422             uiObj = PopupWrapped(text="Batch request running",
windowType="progress", error=None)
00423
00424             threadGui = threading.Thread(target=self.__getCount,
00425                                          daemon=False)
00426             threadGui.start()
00427
00428             while threadGui.is_alive():
00429                 uiObj.textUpdate()
00430                 uiObj.windowPush()
00431             else:
00432                 uiObj.stopWindow()
00433
00434         else:
00435             self.__getCount()
```

Here is the call graph for this function:



Here is the caller graph for this function:



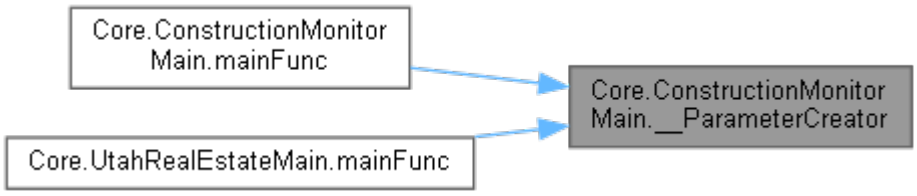
def Core.ConstructionMonitorMain.__ParameterCreator (*self*) `[private]`

```
The __ParameterCreator function is used to create the parameter dictionary that will
be passed into the
__Request function. The function takes in a siteClass object and extracts all of its
attributes, except for
those that start with '__' or are callable. It then creates a dictionary from these
attributes and stores it as
self.__parameterDict.

Args:
self: Make the function a method of the class

Returns:
A dictionary of parameters and a list of non parameter variables

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 317 of file ConstructionMonitor/Core.py.

```
00317     def __ParameterCreator(self):
00318         """
00319     The __ParameterCreator function is used to create the parameter dictionary
that will be passed into the
00320         __Request function. The function takes in a siteClass object and extracts
all of its attributes, except for
00321         those that start with '__' or are callable. It then creates a dictionary
from these attributes and stores it as
00322         self.__parameterDict.
00323
00324     Args:
00325         self: Make the function a method of the class
00326
00327     Returns:
00328         A dictionary of parameters and a list of non parameter variables
00329
00330     Doc Author:
00331         Willem van der Schans, Trelent AI
00332     """
00333         __Source_dict = {key: value for key, value in
self.__siteClass.__dict__.items() if
00334                          not key.startswith('__') and not callable(key)}
00335
00336         self.__restDomain = __Source_dict["rest_domain"]
00337         __Source_dict.pop("rest_domain")
00338         self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00339         __Source_dict.pop("auth_key")
00340         self.__columnSelection = __Source_dict["SourceInclude"]
00341         __Source_dict.pop("SourceInclude")
00342         self.__ui_flag = __Source_dict["ui_flag"]
00343         __Source_dict.pop("ui_flag")
00344         self.__appendFile = __Source_dict["append_file"]
00345         __Source_dict.pop("append_file")
00346
00347         temp_dict = copy.copy(__Source_dict)
00348         for key, value in temp_dict.items():
00349             if value is None:
00350                 __Source_dict.pop(key)
00351             else:
00352                 pass
00353
00354         self.__parameterDict = copy.copy(__Source_dict)
00355
```

Here is the caller graph for this function:



def Core.ConstructionMonitorMain.mainFunc (*self*)

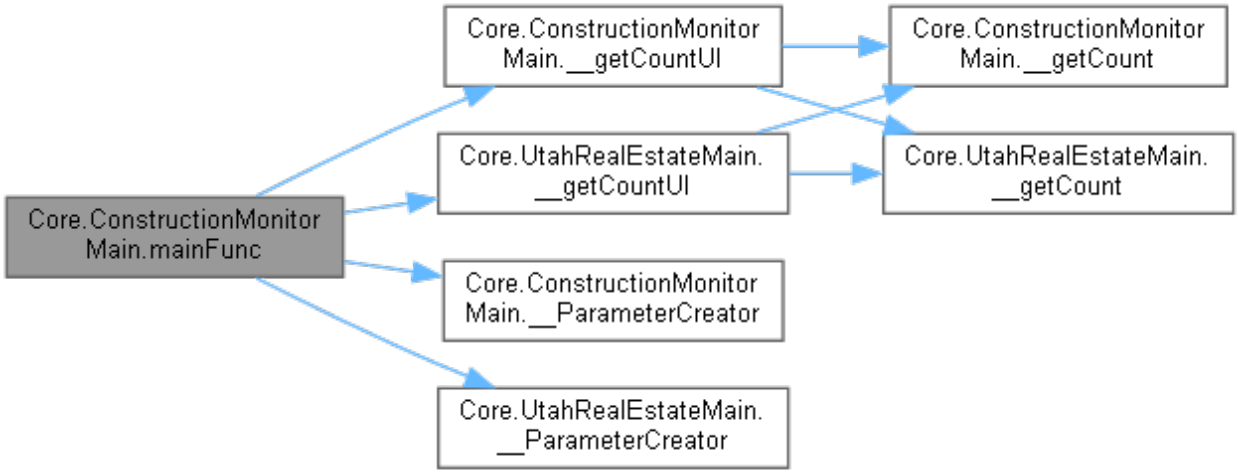
```
The mainFunc function is the main function of this module. It will be called by the
GUI or CLI to execute
the code in this module. The mainFunc function will first create a parameter dictionary
using the __ParameterCreator
method, then it will get a count of all records that match its parameters using the
__getCountUI method, and then
it will calculate how many batches are needed to retrieve all records with those
parameters using BatchCalculator.
After that it asks if you want to continue with retrieving data from Salesforce (if
running in GUI mode). Then it shows
a progress bar for each

Args:
self: Refer to the current object

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 276 of file ConstructionMonitor/Core.py.

```
00276     def mainFunc(self):
00277         """
00278     The mainFunc function is the main function of this module. It will be called
by the GUI or CLI to execute
00279     the code in this module. The mainFunc function will first create a parameter
dictionary using the __ParameterCreator
00280     method, then it will get a count of all records that match its parameters
using the __getCountUI method, and then
00281     it will calculate how many batches are needed to retrieve all records with
those parameters using BatchCalculator.
00282     After that it asks if you want to continue with retrieving data from Salesforce
(if running in GUI mode). Then it shows
00283     a progress bar for each
00284
00285     Args:
00286         self: Refer to the current object
00287
00288     Returns:
00289         The dataframe
00290
00291     Doc Author:
00292         Willem van der Schans, Trelent AI
00293     """
00294         self.__ParameterCreator()
00295
00296         self.__getCountUI()
00297
00298         self.__batches = BatchCalculator(self.__record_val,
self.__parameterDict)
00299         if self.__batches != 0:
00300             startTime = datetime.datetime.now().replace(microsecond=0)
00301             BatchInputGui(self.__batches)
00302             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
00303             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00304
ParameterDict=self.__parameterDict,
00305                                               HeaderDict=self.__headerDict,
00306
ColumnSelection=self.__columnSelection,
00307                                               BatchesNum=self.__batches,
00308                                               Type="construction_monitor")
00309             BatchGuiObject.BatchGuiShow()
00310             self.dataframe = BatchGuiObject.dataframe
00311             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00312             FileSaver("cm", self.dataframe, self.__appendFile)
00313         else:
00314             RESTError(994)
00315             raise SystemExit(994)
```

```
00316
```

Here is the call graph for this function:



Member Data Documentation

Core.ConstructionMonitorMain.__appendFile `[private]`

Definition at line 239 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__batches `[private]`

Definition at line 244 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__columnSelection `[private]`

Definition at line 238 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__headerDict `[private]`

Definition at line 237 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__parameterDict `[private]`

Definition at line 241 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__record_val `[private]`

Definition at line 243 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__restDomain `[private]`

Definition at line 236 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__search_id [private]

Definition at line 242 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__siteClass [private]

Definition at line 235 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.__ui_flag [private]

Definition at line 246 of file ConstructionMonitor/Core.py.

Core.ConstructionMonitorMain.dataframe

Definition at line 248 of file ConstructionMonitor/Core.py.

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

- ConstructionMonitor/Core.py

DataTransfer.DataTransfer Class Reference

Public Member Functions

- def __init__ (self)
- def setValue (self, value)
- def getValue (self)
- def whileValue (self)

Private Attributes

__value

Detailed Description

Definition at line 16 of file DataTransfer.py.

Constructor & Destructor Documentation

def DataTransfer.DataTransfer.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets the initial value of self.__value to 0.

Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 18 of file DataTransfer.py.

```
00018     def __init__(self):
00019         """
00020     The __init__ function is called when the class is instantiated.
00021     It sets the initial value of self.__value to 0.
00022 
00023     Args:
00024         self: Represent the instance of the class
00025 
00026     Returns:
00027         Nothing
00028 
00029     Doc Author:
00030         Willem van der Schans, Trelent AI
00031     """
00032         self.__value = 0
00033 
```

Member Function Documentation

def DataTransfer.DataTransfer.getValue (*self*)

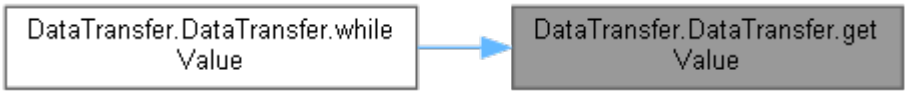
```
The getValue function returns the value of the private variable __value.
This is a getter function that allows access to this private variable.

Args:
self: Represent the instance of the class
```

```
Returns:
The value of the instance variable

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 51 of file DataTransfer.py.

```
00051     def getValue(self):
00052         """
00053     The getValue function returns the value of the private variable __value.
00054     This is a getter function that allows access to this private variable.
00055
00056     Args:
00057         self: Represent the instance of the class
00058
00059     Returns:
00060         The value of the instance variable
00061
00062     Doc Author:
00063         Willem van der Schans, Trelent AI
00064     """
00065         return self.__value
00066
```

Here is the caller graph for this function:



def DataTransfer.DataTransfer.setValue (*self*, *value*)

```
The setValue function sets the value of the object.


Args:
self: Represent the instance of the class
value: Set the value of the instance variable __value

Returns:
The value that was passed to it

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 34 of file DataTransfer.py.

```
00034     def setValue(self, value):
00035         """
00036     The setValue function sets the value of the object.
00037
00038
00039     Args:
00040         self: Represent the instance of the class
00041         value: Set the value of the instance variable __value
00042
00043     Returns:
00044         The value that was passed to it
00045
00046     Doc Author:
00047         Willem van der Schans, Trelent AI
00048     """
00049         self.__value = value
00050
```

def DataTransfer.DataTransfer.whileValue (*self*)

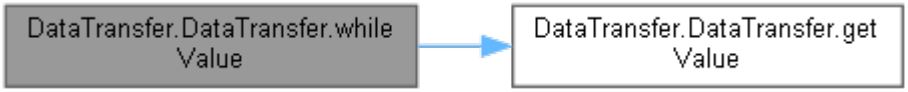
```
The whileValue function is a function that will run the getValue function until it is
told to stop.
This allows for the program to constantly be checking for new values from the sensor.
```

```
Args:
self: Refer to the current instance of the class

Returns:
The value of the input

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 67 of file DataTransfer.py.

```
00067     def whileValue(self):
00068         """
00069     The whileValue function is a function that will run the getValue function
until it is told to stop.
00070     This allows for the program to constantly be checking for new values from
the sensor.
00071 
00072     Args:
00073         self: Refer to the current instance of the class
00074 
00075     Returns:
00076         The value of the input
00077 
00078     Doc Author:
00079         Willem van der Schans, Trelent AI
00080     """
00081         while True:
00082             self.getValue()
```

Here is the call graph for this function:



Member Data Documentation

DataTransfer.DataTransfer.__value [private]

Definition at line 32 of file DataTransfer.py.

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

- DataTransfer.py

FileSaver.FileSaver Class Reference

Public Member Functions

- def __init__ (self, method, outputDF, AppendingPath=None)
- def getPath (self)

Public Attributes

- docPathdata
- dataAppending
- appendFlag
- fileName
- uiFlag
- primaryKey
- outputFrame

Detailed Description

Definition at line 25 of file FileSaver.py.

Constructor & Destructor Documentation

def FileSaver.FileSaver.__init__ (*self*, *method*, *outputDF*, *AppendingPath* = None)

```
The __init__ function is called when the class is instantiated.
It sets up the instance of the class, and defines all variables that will be used by
other functions in this class.
The __init__ function takes two arguments: self and method.  The first argument, self,
refers to an instance of a
class (in this case it's an instance of DataFrameSaver). The second argument, method
refers to a string value that
is passed into DataFrameSaver when it's instantiated.

Args:
self: Represent the instance of the class
method: Determine which dataframe to append the new data to
outputDF: Pass in the dataframe that will be saved to a csv file
AppendingPath: Specify the path to an existing csv file that you want to append your
dataframe to

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 27 of file FileSaver.py.

```
00027     def __init__(self, method, outputDF, AppendingPath=None):
00028         """
00029     The __init__ function is called when the class is instantiated.
00030     It sets up the instance of the class, and defines all variables that will
be used by other functions in this class.
00031     The __init__ function takes two arguments: self and method.  The first
argument, self, refers to an instance of a
00032     class (in this case it's an instance of DataFrameSaver). The second argument,
method refers to a string value that
00033     is passed into DataFrameSaver when it's instantiated.
00034
00035     Args:
```

```
00036         self: Represent the instance of the class
00037         method: Determine which dataframe to append the new data to
00038         outputDF: Pass in the dataframe that will be saved to a csv file
00039         AppendingPath: Specify the path to an existing csv file that you want
to append your dataframe to
00040
00041     Returns:
00042         Nothing
00043
00044     Doc Author:
00045         Willem van der Schans, Trelent AI
00046     """
00047         self.docPath =
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00048             datetime.datetime.today().strftime('%m%d%Y'))
00049         self.data = outputDF
00050         self.dataAppending = None
00051         self.appendFlag = True
00052         self.fileName =
f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00053         self.uiFlag = True
00054
00055         if method.lower() == "ure":
00056             self.primaryKey = "ListingKeyNumeric"
00057         elif method.lower() == "cm":
00058             self.primaryKey = "id"
00059         elif "realtor" in method.lower():
00060             self.primaryKey = None
00061             self.uiFlag = False
00062         elif method.lower() == "cfbp":
00063             self.primaryKey = None
00064             self.uiFlag = False
00065         else:
00066             raise ValueError("method input is invalid choice one of 4 options:
URE, CM, Realtor, CFBP")
00067
00068         if AppendingPath is None:
00069             self.appendFlag = False
00070         else:
00071             self.dataAppending = pd.read_csv(AppendingPath)
00072
00073         if self.appendFlag:
00074             if self.primaryKey is not None:
00075                 # Due to low_memory loading the columns are not typed properly,
00076                 # since we are comparing this will be an issue since we need to
do type comparisons,
00077                 # so here we coerce the types of the primary keys to numeric.
00078                 # If another primary key is ever chosen make sure to core to the
right data type.
00079                 self.dataAppending[self.primaryKey] =
pd.to_numeric(self.dataAppending[self.primaryKey])
00080                 self.data[self.primaryKey] =
pd.to_numeric(self.data[self.primaryKey])
00081
00082                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00083
keep="last")
00084             else:
00085                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(keep="last")
00086         else:
00087             self.outputFrame = self.data
00088
00089         if os.path.exists(self.docPath):
00090             self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00091         else:
00092             os.mkdir(self.docPath)
00093             self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00094
00095         if self.uiFlag:
00096             if self.appendFlag:
00097                 PopupWrapped(text=f"File Appended and Saved to
{self.docPath.joinpath(self.fileName)}",
00098                              windowType="noticeLarge")
```

```
00099
00100                 # Logging
00101                 print(
00102                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | {method} API request Completed | File Appended and Saved to
{self.docPath.joinpath(self.fileName)} | Exit Code 0")
00103                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Appending Statistics | Method: {method} | Appending file rows:
{self.dataAppending.shape[0]}, Total Rows: {(self.dataAppending.shape[0] +
self.data.shape[0])}, Duplicates Dropped {(self.dataAppending.shape[0] +
self.data.shape[0])-self.outputFrame.shape[0]}")
00104             else:
00105                 PopupWrapped(text=f"File Saved to
{self.docPath.joinpath(self.fileName)}", windowType="noticeLarge")
00106
00107                 # Logging
00108                 print(
00109                     f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | {method} API request Completed | File Saved to
{self.docPath.joinpath(self.fileName)} | Exit Code 0")
00110         else:
00111             pass
00112
```

Member Function Documentation

def FileSaver.FileSaver.getPath (*self*)

```
The getPath function returns the path to the file.
It is a string, and it joins the docPath with the fileName.

Args:
self: Represent the instance of the class

Returns:
The path to the file

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 113 of file FileSaver.py.

```
00113     def getPath(self):
00114         """
00115     The getPath function returns the path to the file.
00116         It is a string, and it joins the docPath with the fileName.
00117
00118     Args:
00119         self: Represent the instance of the class
00120
00121     Returns:
00122         The path to the file
00123
00124     Doc Author:
00125         Willem van der Schans, Trelent AI
00126     """
00127         return str(self.docPath.joinpath(self.fileName))
```

Member Data Documentation

FileSaver.FileSaver.appendFlag

Definition at line 51 of file FileSaver.py.

FileSaver.FileSaver.data

Definition at line 49 of file FileSaver.py.

FileSaver.FileSaver.dataAppending

Definition at line 50 of file FileSaver.py.

FileSaver.FileSaver.docPath

Definition at line 47 of file FileSaver.py.

FileSaver.FileSaver.fileName

Definition at line 52 of file FileSaver.py.

FileSaver.FileSaver.outputFrame

Definition at line 82 of file FileSaver.py.

FileSaver.FileSaver.primaryKey

Definition at line 56 of file FileSaver.py.

FileSaver.FileSaver.uiFlag

Definition at line 53 of file FileSaver.py.

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

- FileSaver.py

API_Calls.Initializer.initializer Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

classObjPrivate Member Functions

- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Detailed Description

Definition at line 32 of file Initializer.py.

Constructor & Destructor Documentation

def API_Calls.Initializer.initializer.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the logging, calls the __ShowGui function to create and display
the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 34 of file Initializer.py.

```
00034     def __init__(self):
00035 
00036         """
00037     The __init__ function is called when the class is instantiated.
00038     It sets up the logging, calls the __ShowGui function to create and display
00039     the GUI, and then calls __CreateFrame to create a frame for displaying
widgets.
00040 
00041 
00042     Args:
00043         self: Represent the instance of the class
00044 
00045     Returns:
00046         Nothing
00047 
00048     Doc Author:
00049         Willem van der Schans, Trelent AI
00050     """
00051         self.classObj = None
00052 
00053         logger()
00054 
00055         print("\n\n-------------Initiate Program--------------------\n\n")
00056 
00057         self.__ShowGui(self.__CreateFrame(), "Data Tool")
00058 
```

```
00059         print("\n\n-------------Closing Program--------------------\n\n")
00060
```

Member Function Documentation

def API_Calls.Initializer.initializer.__CreateFrame (*self*) [private]

```
The __CreateFrame function is a helper function that creates the layout for the main
window.
It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

Args:
self: Represent the instance of the class

Returns:
A list of lists, which is then passed to the sg

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 132 of file Initializer.py.

```
00132     def __CreateFrame(self):
00133
00134         """
00135     The __CreateFrame function is a helper function that creates the layout for
the main window.
00136     It returns a list of lists, which is then passed to sg.Window() as its layout
parameter.
00137
00138     Args:
00139         self: Represent the instance of the class
00140
00141     Returns:
00142         A list of lists, which is then passed to the sg
00143
00144     Doc Author:
00145         Willem van der Schans, Trelent AI
00146     """
00147         sg.theme('Default1')
00148
00149         line0 = [sg.HSeparator()]
00150
00151         line1 = [sg.Image(ImageLoader("logo.png")),
00152                  sg.Push(),
00153                  sg.Text("Gardner Data Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00154                  sg.Push(),
00155                  sg.Push()]
00156
00157         line3 = [sg.HSeparator()]
00158
00159         line4 = [sg.Push(),
00160                  sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
justification="center"),
00161                  sg.Push()]
00162
00163         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20,
None)), sg.Push(),
00164                   sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00165
00166         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)),
sg.Push(), sg.Button("Census", size=(20, None)),
00167                   sg.Push()]]
00168
00169         line8 = [sg.HSeparator()]
00170
00171         line9 = [sg.Push(),
00172                  sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
00173                  sg.Push()]
```

```
00174
00175         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20,
None)),
00176                    sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00177
00178         line11 = [sg.HSeparator()]
00179
00180         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]
00181
00182         return layout
```

def API_Calls.Initializer.initializer.__ShowGui (*self*, *layout*, *text*) [private]

```
The __ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing
a tuple with three elements:
1) The type of element to be displayed (e.g., &amp;quot;Text&amp;quot;, &amp;quot;InputText&amp;quot;,
etc.)
2) A dictionary containing any additional parameters for that element (e.g., size,
default value, etc.)
3) An optional key name for the element (used in event handling). If no key name is
provided then one will be generated automatically by PySimpleGUIQt based on its position
in the layout list

Args:
self: Represent the instance of the class
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A window object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 61 of file Initializer.py.

```
00061     def __ShowGui(self, layout, text):
00062
00063         """
00064     The __ShowGui function is the main function that displays the GUI.
00065     It takes two arguments: layout and text. Layout is a list of lists, each
containing a tuple with three elements:
00066         1) The type of element to be displayed (e.g., &amp;quot;Text&amp;quot;,
&amp;quot;InputText&amp;quot;, etc.)
00067         2) A dictionary containing any additional parameters for that element
(e.g., size, default value, etc.)
00068         3) An optional key name for the element (used in event handling). If no
key name is provided then one will be generated automatically by PySimpleGUIQt based
on its position in the layout list
00069
00070     Args:
00071         self: Represent the instance of the class
00072         layout: Pass the layout of the window to be created
00073         text: Set the title of the window
00074
00075     Returns:
00076         A window object
00077
00078     Doc Author:
00079         Willem van der Schans, Trelent AI
00080     """
00081         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00082                            finalize=True,
00083                            icon=ImageLoader("taskbar_icon.ico"))
00084
00085         while True:
00086             event, values = window.read()
00087
00088             if event == "Construction Monitor":
```

```
00089                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API
Call-----------------")
00090                   ConstructionMonitorMain(ConstructionMonitorInit())
00091                   print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API
Call---------------------\n")
00092               elif event == "Utah Real Estate":
00093                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API
Call-----------------")
00094                   UtahRealEstateMain(UtahRealEstateInit())
00095                   print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API
Call---------------------\n")
00096               elif event == "Realtor.Com":
00097                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
00098                   realtorCom()
00099                   print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API
Call---------------------\n")
00100               elif event == "Census":
00101                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Census API Call-----------------")
00102                   Cencus()
00103                   print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Census API Call--------------------\n")
00104               elif event == "Authorization Utility":
00105                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Authorization
Utility-----------------")
00106                   AuthUtil()
00107                   print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Authorization
Utility---------------------\n")
00108               elif event == "Open Data Folder":
00109                   print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
00110                   try:
00111                       os.system(f"start
{Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
00112                   except:
00113                       try:
00114                           os.system(f"start
{Path(os.path.expanduser('~/Documents'))}")
00115                       except Exception as e:
00116
print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
00117                           PopupWrapped(
00118                               text="Documents folder not found. Please create a
Windows recognized documents folder",
00119                               windowType="errorLarge")
00120
00121               elif event in ('Exit', None):
00122                   try:
00123                       break
00124                   except Exception as e:
00125                       print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Error on program exit, for logging
purposes only.")
00126                       break
00127               elif event == sg.WIN_CLOSED or event == "Quit":
00128                   break
00129
00130         window.close()
00131
```

Member Data Documentation

API_Calls.Initializer.initializer.classObj

Definition at line 51 of file Initializer.py.

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

- Initializer.py

PopupWrapped.PopupWrapped Class Reference

Public Member Functions

- def __init__ (self, text="", windowType="notice", error=None)
- def stopWindow (self)
- def textUpdate (self, sleep=0.5)
- def windowPush (self)

Private Member Functions

- def __createLayout (self)
- def __createWindow (self)

Private Attributes

- __text__type
- __error
- __layout
- __windowObj
- __thread
- __counter

Detailed Description

Definition at line 24 of file PopupWrapped.py.

Constructor & Destructor Documentation

def PopupWrapped.PopupWrapped.__init__ (*self*, *text* = "", *windowType* = "notice", *error* = None)

```
The __init__ function is the first function that gets called when an object of this
class is created.
It sets up all the variables and creates a window for us to use.
Args:
self: Represent the instance of the class
text: Set the text of the window
windowType: Determine what type of window to create
error: Display the error message in the window
Returns:
Nothing
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 26 of file PopupWrapped.py.

```
00026     def __init__(self, text="", windowType="notice", error=None):
00027         """
00028     The __init__ function is the first function that gets called when an object
of this class is created.
00029     It sets up all the variables and creates a window for us to use.
00030     Args:
00031         self: Represent the instance of the class
00032         text: Set the text of the window
00033         windowType: Determine what type of window to create
00034         error: Display the error message in the window
00035     Returns:
00036         Nothing
00037     Doc Author:
00038         Willem van der Schans, Trelent AI
```

```
00039     """
00040         self.__text = text
00041         self.__type = windowType
00042         self.__error = error
00043         self.__layout = []
00044         self.__windowObj = None
00045         self.__thread = None
00046         self.__counter = 0
00047 
00048         self.__createWindow()
00049 
```

Member Function Documentation

def PopupWrapped.PopupWrapped.__createLayout (*self*) [private]

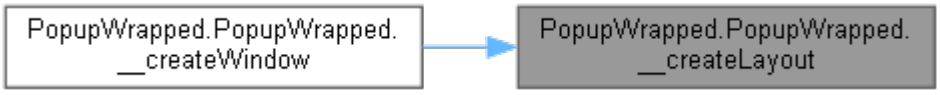
```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates
a layout based on that information.
Args:
self: Refer to the current instance of a class
Returns:
A list of lists
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 50 of file PopupWrapped.py.

```
00050     def __createLayout(self):
00051         """
00052     The __createLayout function is used to create the layout of the window.
00053     The function takes class variables and returns a window layout.
00054     It uses a series of if statements to determine what type of window it is,
then creates a layout based on that information.
00055     Args:
00056         self: Refer to the current instance of a class
00057     Returns:
00058         A list of lists
00059     Doc Author:
00060         Willem van der Schans, Trelent AI
00061     """
00062         sg.theme('Default1')
00063         __Line1 = None
00064         __Line2 = None
00065 
00066         if self.__type == "notice":
00067             __Line1 = [sg.Push(),
00068                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00069                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00070             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00071         elif self.__type == "noticeLarge":
00072             __Line1 = [sg.Push(),
00073                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00074                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00075             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00076         elif self.__type == "errorLarge":
00077             __Line1 = [sg.Push(),
00078                        sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00079                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00080             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00081         elif self.__type == "FatalErrorLarge":
00082             __Line1 = [sg.Push(),
00083                        sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
```

```
00084                           sg.Text(self.__text, justification="left",
key="-textField-"), sg.Push()]
00085                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00086         elif self.__type == "error":
00087             __Line1 = [sg.Push(),
00088                           sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00089                           sg.Text(f"{self.__text}: {self.__error}",
justification="center", key="-textField-"),
00090                           sg.Push()]
00091                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00092         elif self.__type == "progress":
00093             __Line1 = [sg.Push(),
00094                           sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00095
00096         if self.__type == "progress":
00097             self.__layout = [__Line1, ]
00098         else:
00099             self.__layout = [__Line1, __Line2]
00100
```

Here is the caller graph for this function:



def PopupWrapped.PopupWrapped.__createWindow (*self*) `[private]`

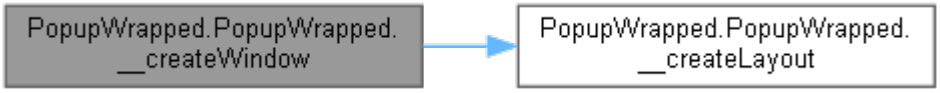
```
The __createWindow function is used to create the window object that will be displayed.
The function takes class variables and a window object. The function first calls
__createLayout, which creates the layout for the window based on what type of message
it is (error, notice, progress). Then it uses PySimpleGUI's Window class to create a
new window with that layout and some other parameters such as title and icon. If this
is not a progress bar or permanent message then we start a timer loop that waits until
either 100 iterations have passed or an event has been triggered (such as clicking
&amp;quot;Ok&amp;quot; or closing the window). Once one of these events occurs
Args:
self: Reference the instance of the class
Returns:
A window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 101 of file PopupWrapped.py.

```
00101     def __createWindow(self):
00102         """
00103     The __createWindow function is used to create the window object that will
be displayed.
00104     The function takes class variables and a window object. The function first
calls __createLayout, which creates the layout for the window based on what type of
message it is (error, notice, progress). Then it uses PySimpleGUI's Window class to
create a new window with that layout and some other parameters such as title and icon.
If this is not a progress bar or permanent message then we start a timer loop that waits
until either 100 iterations have passed or an event has been triggered (such as clicking
&amp;quot;Ok&amp;quot; or closing the window). Once one of these events occurs
00105     Args:
00106         self: Reference the instance of the class
00107     Returns:
00108         A window object
00109     Doc Author:
00110         Willem van der Schans, Trelent AI
00111     """
00112         self.__createLayout()
00113
00114         if self.__type == "progress":
00115             self.__windowObj = sg.Window(title=self.__type,
layout=self.__layout, finalize=True,
00116                                          modal=True,
00117                                          keep_on_top=True,
00118                                          disable_close=False,
```

```
00119
icon=ImageLoader("taskbar_icon.ico"),
00120                                           size=(290, 50))
00121         elif self.__type == "noticeLarge":
00122             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
finalize=True,
00123                                           modal=True,
00124                                           keep_on_top=True,
00125                                           disable_close=False,
00126
icon=ImageLoader("taskbar_icon.ico"))
00127         elif self.__type == "errorLarge":
00128             self.__windowObj = sg.Window(title="Error", layout=self.__layout,
finalize=True,
00129                                           modal=True,
00130                                           keep on top=True,
00131                                           disable_close=False,
00132
icon=ImageLoader("taskbar_icon.ico"))
00133         elif self.__type == "FatalErrorLarge":
00134             self.__windowObj = sg.Window(title="Fatal Error",
layout=self.  layout, finalize=True,
00135                                           modal=True,
00136                                           keep_on_top=True,
00137                                           disable_close=False,
00138
icon=ImageLoader("taskbar_icon.ico"))
00139         else:
00140             self.__windowObj = sg.Window(title=self.__type,
layout=self.__layout, finalize=True,
00141                                           modal=True,
00142                                           keep_on_top=True,
00143                                           disable_close=False,
00144
icon=ImageLoader("taskbar_icon.ico"),
00145                                           size=(290, 80))
00146
00147         if self.__type != "progress" or self.__type.startswith("perm"):
00148             timer = 0
00149             while timer < 100:
00150                 event, values = self.__windowObj.read()
00151                 if event == "Ok" or event == sg.WIN_CLOSED:
00152                     break
00153
00154                 time.sleep(0.1)
00155
00156             if self.__type == "FatalErrorLarge":
00157                 try:
00158                     os.system(
00159                         f"start
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}"
)
00160                 except Exception as e:
00161                     print(
00162                         f"PopupWrapped.py | Error = {e} | Log Folder not found
please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")
00163
00164             self.__windowObj.close()
00165
```

Here is the call graph for this function:



def PopupWrapped.PopupWrapped.stopWindow (*self*)

```
The stopWindow function is used to close the window object that was created in the
startWindow function.
This is done by calling the close() method on self.__windowObj, which will cause it
to be destroyed.
Args:
self: Represent the instance of the class
Returns:
```

```
The window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 166 of file PopupWrapped.py.

```
00166     def stopWindow(self):
00167         """
00168     The stopWindow function is used to close the window object that was created
in the startWindow function.
00169     This is done by calling the close() method on self.  windowObj, which will
cause it to be destroyed.
00170     Args:
00171         self: Represent the instance of the class
00172     Returns:
00173         The window object
00174     Doc Author:
00175         Willem van der Schans, Trelent AI
00176     """
00177         self.__windowObj.close()
00178
```

def PopupWrapped.PopupWrapped.textUpdate (*self*, *sleep* = 0.5)

```
The textUpdate function is a function that updates the text in the text field.
It does this by adding dots to the end of it, and then removing them. This creates
a loading effect for when something is being processed.
Args:
self: Refer to the object itself
sleep: Control the speed of the text update
Returns:
A string that is the current text of the text field
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 179 of file PopupWrapped.py.

```
00179     def textUpdate(self, sleep=0.5):
00180         """
00181     The textUpdate function is a function that updates the text in the text field.
00182     It does this by adding dots to the end of it, and then removing them. This
creates
00183     a loading effect for when something is being processed.
00184     Args:
00185         self: Refer to the object itself
00186         sleep: Control the speed of the text update
00187     Returns:
00188         A string that is the current text of the text field
00189     Doc Author:
00190         Willem van der Schans, Trelent AI
00191     """
00192         self.__counter += 1
00193         if self.__counter == 4:
00194             self.__counter = 1
00195         newString = ""
00196         if self.__type == "notice":
00197             pass
00198         elif self.__type == "error":
00199             pass
00200         elif self.__type == "progress":
00201             newString = f"{self.__text}{'.' * self.__counter}"
00202         self.__windowObj.write_event_value('update-textField-', newString)
00203
00204         time.sleep(sleep)
00205
```

def PopupWrapped.PopupWrapped.windowPush (*self*)

```
The windowPush function is used to update the values of a window object.
The function takes in an event and values from the window object, then checks if the
event starts with 'update'.
```

```
If it does, it will take everything after 'update' as a key for updating that specific
value.
It will then update that value using its key and refresh the window.
Args:
self: Reference the object that is calling the function
Returns:
A tuple containing the event and values
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 206 of file PopupWrapped.py.

```
00206     def windowPush(self):
00207
00208         """
00209     The windowPush function is used to update the values of a window object.
00210         The function takes in an event and values from the window object, then
checks if the event starts with 'update'.
00211         If it does, it will take everything after 'update' as a key for updating
that specific value.
00212         It will then update that value using its key and refresh the window.
00213     Args:
00214         self: Reference the object that is calling the function
00215     Returns:
00216         A tuple containing the event and values
00217     Doc Author:
00218         Willem van der Schans, Trelent AI
00219     """
00220         event, values = self.__windowObj.read()
00221
00222         if event.startswith('update'):
00223             __key_to_update = event[len('update'):]
00224             self.__windowObj[__key_to_update].update(values[event])
00225             self.__windowObj.refresh()
```

Member Data Documentation

PopupWrapped.PopupWrapped.__counter [private]

Definition at line 46 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__error [private]

Definition at line 42 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__layout [private]

Definition at line 43 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__text [private]

Definition at line 40 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__thread [private]

Definition at line 45 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__type [private]

Definition at line 41 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__windowObj [private]

Definition at line 44 of file PopupWrapped.py.

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

- PopupWrapped.py

Core.realtorCom Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- dfStatedfCounty
- dfZip
- uiString

Private Member Functions

- def __showUi (self)
- def __linkGetter (self)
- def __dataUpdater (self)

Private Attributes

- __page_html__update_date
- __last_date
- __idDict
- __linkDict

Detailed Description

Definition at line 12 of file Realtor/Core.py.

Constructor & Destructor Documentation

def Core.realtorCom.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of an object, and it's where you put code that needs to
run before anything else in your class.

Args:
self: Represent the instance of the class

Returns:
A new object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 14 of file Realtor/Core.py.

```
00014     def __init__(self):
00015         """
00016     The __init__ function is called when the class is instantiated.
00017     It sets up the initial state of an object, and it's where you put code that
needs to run before anything else in your class.
00018
00019     Args:
00020         self: Represent the instance of the class
00021
00022     Returns:
00023         A new object
00024
00025     Doc Author:
00026         Willem van der Schans, Trelent AI
```

```
00027     """
00028         self.__page_html = None
00029         self.__update_date = None
00030         self.__last_date = None
00031         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00032         self.__linkDict = {}
00033         self.dfState = None
00034         self.dfCounty = None
00035         self.dfZip = None
00036         self.uiString = "Files Saved to \n"
00037 
00038         page_html =
requests.get("https://www.realtor.com/research/data/").text
00039         self.__page_html = BeautifulSoup(page_html, "html.parser")
00040 
00041         self.  linkGetter()
00042         self.__showUi()
00043 
00044         PopupWrapped(text=self.uiString, windowType="noticeLarge")
00045 
```

Member Function Documentation

def Core.realtorCom.__dataUpdater (*self*) `[private]`

```
The __dataUpdater function is a private function that updates the dataframes for each
of the three
types of realtor data. It takes class variables and return the path to the saved file.
The function first creates an empty
dictionary called tempdf, then iterates through each key in self.__idDict (which
contains all three ids).
For each key, it reads in a csv file from the link associated with that id and saves
it to tempdf as a pandas
DataFrame object. Then, depending on which type of realtor data we are dealing with
(State/County/Zip), we save


Args:
self: Access the attributes and methods of the class

Returns:
The path of the saved file

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 101 of file Realtor/Core.py.

```
00101     def __dataUpdater(self):
00102 
00103         """
00104     The __dataUpdater function is a private function that updates the dataframes
for each of the three
00105         types of realtor data. It takes class variables and return the path to
the saved file. The function first creates an empty
00106         dictionary called tempdf, then iterates through each key in self.__idDict
(which contains all three ids).
00107         For each key, it reads in a csv file from the link associated with that
id and saves it to tempdf as a pandas
00108         DataFrame object. Then, depending on which type of realtor data we are
dealing with (State/County/Zip), we save
00109 
00110 
00111     Args:
00112         self: Access the attributes and methods of the class
00113 
00114     Returns:
00115         The path of the saved file
00116 
00117     Doc Author:
00118         Willem van der Schans, Trelent AI
```

```
00119     """
00120         for key, value in self.__idDict.items():
00121             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00122 
00123             if key == "State":
00124                 self.dfState = tempdf
00125             elif key == "County":
00126                 self.dfCounty = tempdf
00127             elif key == "Zip":
00128                 self.dfZip = tempdf
00129 
00130             FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00131             self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()}
\n"
```

Here is the caller graph for this function:

def Core.realtorCom.__linkGetter (*self*) `[private]`

```
The __linkGetter function is a private function that takes the idDict dictionary and
adds
a link to each entry in the dictionary. The link is used to access historical data for
each
scope symbol.

Args:
self: Refer to the object itself

Returns:
A dictionary of all the links to the history pages

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 74 of file Realtor/Core.py.

```
00074     def __linkGetter(self):
00075 
00076         """
00077     The __linkGetter function is a private function that takes the idDict
dictionary and adds
00078     a link to each entry in the dictionary. The link is used to access historical
data for each
00079     scope symbol.
00080 
00081     Args:
00082         self: Refer to the object itself
00083 
00084     Returns:
00085         A dictionary of all the links to the history pages
00086 
00087     Doc Author:
00088         Willem van der Schans, Trelent AI
00089     """
00090         for key, value in self.__idDict.items():
00091             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00092                 try:
00093                     for nestedRow in row.find_all("a"):
00094                         if "History" in str(nestedRow.get("href")) and key in
str(nestedRow.get("href")):
00095                             self.__idDict[key] = {"id": value, "link":
nestedRow.get("href")}
00096                 except Exception as e:
00097                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
for realtor.com")
00098                     RESTError(801)
00099                     raise SystemExit(801)
00100 
```

def Core.realtorCom.__showUi (*self*) [private]

```
The __showUi function is a helper function that creates and displays the progress window.
It also starts the dataUpdater thread, which will update the progress bar as it runs.


Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 46 of file Realtor/Core.py.

```
00046     def __showUi(self):
00047 
00048         """
00049     The __showUi function is a helper function that creates and displays the
progress window.
00050     It also starts the dataUpdater thread, which will update the progress bar
as it runs.
00051 
00052 
00053     Args:
00054         self: Represent the instance of the class
00055 
00056     Returns:
00057         A popupwrapped object
00058 
00059     Doc Author:
00060         Willem van der Schans, Trelent AI
00061     """
00062         uiObj = PopupWrapped(text="Request running", windowType="progress",
error=None)
00063 
00064         threadGui = threading.Thread(target=self.__dataUpdater,
00065                                      daemon=False)
00066         threadGui.start()
00067 
00068         while threadGui.is_alive():
00069             uiObj.textUpdate()
00070             uiObj.windowPush()
00071         else:
00072             uiObj.stopWindow()
00073 
```

Here is the call graph for this function:

Member Data Documentation

Core.realtorCom.__idDict [private]

Definition at line 31 of file Realtor/Core.py.

Core.realtorCom.__last_date [private]

Definition at line 30 of file Realtor/Core.py.

Core.realtorCom.__linkDict [private]

Definition at line 32 of file Realtor/Core.py.

Core.realtorCom.__page_html [private]

Definition at line 28 of file Realtor/Core.py.

Core.realtorCom.__update_date [private]

Definition at line 29 of file Realtor/Core.py.

Core.realtorCom.dfCounty

Definition at line 34 of file Realtor/Core.py.

Core.realtorCom.dfState

Definition at line 33 of file Realtor/Core.py.

Core.realtorCom.dfZip

Definition at line 35 of file Realtor/Core.py.

Core.realtorCom.uiString

Definition at line 36 of file Realtor/Core.py.

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

- Realtor/Core.py

Core.UtahRealEstateInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatusListedOrModified
- dateStart
- dateEnd
- select
- file_name
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

Detailed Description

Definition at line 24 of file UtahRealEstate/Core.py.

Constructor & Destructor Documentation

def Core.UtahRealEstateInit.__init__ (*self*)

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object.


Args:
self: Represent the instance of the class

Returns:
The __createframe function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 26 of file UtahRealEstate/Core.py.

```
00026     def __init__(self):
00027 
00028         """
00029     The __init__ function is called when the class is instantiated.
00030     It sets up the initial state of the object.
00031 
00032 
00033     Args:
00034         self: Represent the instance of the class
00035 
00036     Returns:
00037         The __createframe function
00038 
00039     Doc Author:
00040         Willem van der Schans, Trelent AI
```

```
00041     """
00042         self.StandardStatus = None
00043         self.ListedOrModified = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.select = None
00047         self.file_name = None
00048         self.append_file = None
00049 
00050         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00051 
```

Member Function Documentation

def Core.UtahRealEstateInit.__CreateFrame () `[static]`, `[private]`

```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the window.
Each element is defined by its type and any additional parameters needed to define it.

Args:

Returns:
A list of lists, which is used to create the gui

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 93 of file UtahRealEstate/Core.py.

```
00093     def __CreateFrame():
00094         """
00095     The __CreateFrame function creates the GUI layout for the application.
00096         The function returns a list of lists that contains all the elements to
be displayed in the window.
00097         Each element is defined by its type and any additional parameters needed
to define it.
00098 
00099     Args:
00100 
00101     Returns:
00102         A list of lists, which is used to create the gui
00103 
00104     Doc Author:
00105         Willem van der Schans, Trelent AI
00106     """
00107         sg.theme('Default1')
00108 
00109         line00 = [sg.HSeparator()]
00110 
00111         line0 = [sg.Image(ImageLoader("logo.png")),
00112                  sg.Push(),
00113                  sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00114                  sg.Push(),
00115                  sg.Push()]
00116 
00117         line1 = [sg.HSeparator()]
00118 
00119         line2 = [sg.Text("MLS Status : ", size=(15, None),
justification="Right"),
00120                  sg.DropDown(default_value="Active", values=["Active",
"Closed"], key="-status-", size=(31, 1))]
00121 
00122         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00123                  sg.DropDown(default_value="Listing Date", values=["Listing
Date", "Modification Date", "Close Date"],
00124                              key="-type-", size=(31, 1))]
00125 
```

```
00126         line4 = [sg.Text("Start Date : ", size=(15, None),
justification="Right"),
00127                  sg.Input(default_text=(date.today() -
timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00128                           disabled=False, size=(20, 1)),
00129                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-start_date-', target="-DateStart-")]
00130
00131         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00132                  sg.Input(default_text=(date.today().strftime("%Y-%m-%d")),
key="-DateEnd-", disabled=False,
00133                           size=(20, 1)),
00134                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-end_date-', target="-DateEnd-")]
00135
00136         line6 = [[sg.Text("Column Sub-Selection : ", size=(23, None),
justification="Right"),
00137                   sg.Checkbox(text="", default=True, key="-selectionFlag-",
size=(15, 1)),
00138                   sg.Push()]]
00139
00140         line7 = [sg.HSeparator()]
00141
00142         line8 = [sg.Push(),
00143                  sg.Text("File Settings", font=("Helvetica", 12, "bold"),
justification="center"),
00144                  sg.Push()]
00145
00146         line9 = [sg.HSeparator()]
00147
00148         line10 = [sg.Text("Appending File : ", size=(15, None),
justification="Right"),
00149                   sg.Input(default_text="", key="-AppendingFile-",
disabled=True,
00150                            size=(20, 1)),
00151                   sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
00152                                 target="-AppendingFile-")]
00153
00154         line11 = [sg.HSeparator()]
00155
00156         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00157
00158         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10, line11,
00159                   line12]
00160
00161         return layout
00162
```

def Core.UtahRealEstateInit.__SetValues (*self*, *values*) `[private]`

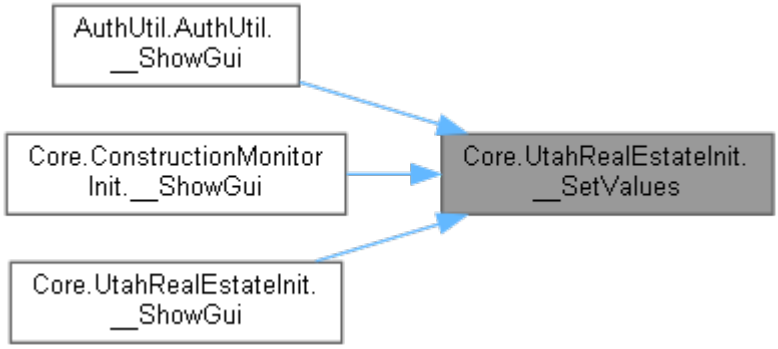
```
The __SetValues function is used to set the values of the variables that are used in
the
__GetData function. The values are passed from a dictionary called 'values' which is
created
by parsing through an XML file using ElementTree. This function also sets default values
for
some of these variables if they were not specified in the XML file.

Args:
self: Represent the instance of the class
values: Pass the values from the gui to this function

Returns:
A dictionary with the following keys:

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 163 of file UtahRealEstate/Core.py.

```
00163     def __SetValues(self, values):
00164
```

```
00165         """
00166     The __SetValues function is used to set the values of the variables that are
used in the
00167         __GetData function. The values are passed from a dictionary called
'values' which is created
00168        by parsing through an XML file using ElementTree. This function also sets
default values for
00169        some of these variables if they were not specified in the XML file.
00170
00171     Args:
00172         self: Represent the instance of the class
00173         values: Pass the values from the gui to this function
00174
00175     Returns:
00176         A dictionary with the following keys:
00177
00178     Doc Author:
00179         Willem van der Schans, Trelent AI
00180     """
00181         self.StandardStatus = values["-status-"]
00182
00183         self.ListedOrModified = values["-type-"]
00184
00185         if values["-DateStart-"] != "":
00186             self.dateStart = values["-DateStart-"]
00187         else:
00188             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00189
00190         if values["-DateEnd-"] != "":
00191             self.dateEnd = values["-DateEnd-"]
00192         else:
00193             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00194
00195         if values['-selectionFlag-']:
00196             self.select =
"ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,Subdi
visionName," \
00197
"StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDa
te," \
00198
"ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,L
otSizeAcres," \
00199
"LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,
ParkingTotal," \
00200
"BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffect
ive," \
00201
"OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseCont
ractDate," \
00202
"AssociationFee,AssociationFeeFrequency,OccupantType,PropertySubType,PropertyType,
" \
00203                             "StandardStatus,BuyerFinancing"
00204         else:
00205             self.select = None
00206
00207         if values["-append_file-"] != "":
00208             self.append_file = str(values["-append_file-"])
00209         else:
00210             self.append_file = None
00211
00212
```

Here is the caller graph for this function:



def Core.UtahRealEstateInit.__ShowGui (*self*, *layout*, *text*) `[private]`

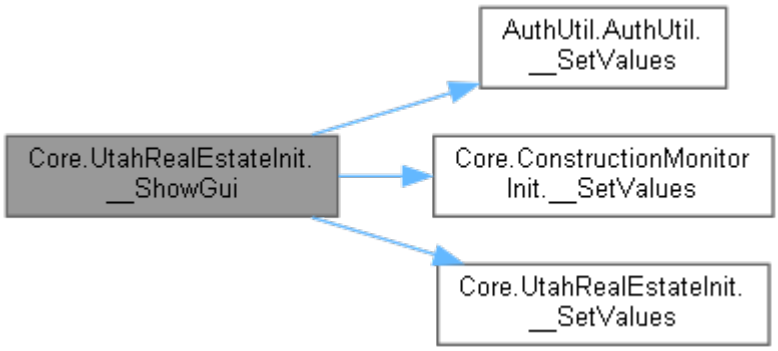
```
The __ShowGui function is a helper function that creates the GUI window and displays
it to the user.
It takes in two parameters: layout, which is a list of lists containing all the elements
for each row;
and text, which is a string containing what will be displayed as the title of the window.
The __ShowGui
method then uses these parameters to create an instance of sg.Window with all its
attributes set accordingly.

Args:
self: Refer to the current class instance
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 52 of file UtahRealEstate/Core.py.

```
00052     def __ShowGui(self, layout, text):
00053
00054         """
00055     The __ShowGui function is a helper function that creates the GUI window and
displays it to the user.
00056     It takes in two parameters: layout, which is a list of lists containing all
the elements for each row;
00057     and text, which is a string containing what will be displayed as the title
of the window. The __ShowGui
00058     method then uses these parameters to create an instance of sg.Window with
all its attributes set accordingly.
00059
00060     Args:
00061         self: Refer to the current class instance
00062         layout: Pass the layout of the window to be created
00063         text: Set the title of the window
00064
00065     Returns:
00066         A dictionary of values
00067
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
00070     """
00071         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00072                            finalize=True,
00073                            icon=ImageLoader("taskbar_icon.ico"))
00074
00075         while True:
00076             event, values = window.read()
00077
00078             if event == "Submit":
00079                 try:
00080                     self.__SetValues(values)
```

```
00081                     break
00082                 except Exception as e:
00083                     print(e)
00084                     RESTError(993)
00085                     raise SystemExit(993)
00086             elif event == sg.WIN_CLOSED or event == "Quit":
00087 
00088                 break
00089 
00090         window.close()
00091 
```

Here is the call graph for this function:



Member Data Documentation

Core.UtahRealEstateInit.append_file

Definition at line 48 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.dateEnd

Definition at line 45 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.dateStart

Definition at line 44 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.file_name

Definition at line 47 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.ListedOrModified

Definition at line 43 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.select

Definition at line 46 of file UtahRealEstate/Core.py.

Core.UtahRealEstateInit.StandardStatus

Definition at line 42 of file UtahRealEstate/Core.py.

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

- UtahRealEstate/Core.py

Core.UtahRealEstateMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

- dataframekeyPath
- filePath
- key

Private Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __batches__siteClass
- __headerDict
- __parameterString
- __appendFile
- __dateStart
- __dateEnd
- __restDomain
- __record_val

Detailed Description

Definition at line 213 of file UtahRealEstate/Core.py.

Constructor & Destructor Documentation

def Core.UtahRealEstateMain.__init__ (*self*, *siteClass*)

```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to work properly.

Args:
self: Represent the instance of the class
siteClass: Determine which site to pull data from

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 215 of file UtahRealEstate/Core.py.

```
00215     def __init__(self, siteClass):
00216 
00217         """
00218     The __init__ function is the first function that runs when an object of this
class is created.
```

```
00219     It sets up all the variables and functions needed for this class to work
properly.
00220
00221     Args:
00222         self: Represent the instance of the class
00223         siteClass: Determine which site to pull data from
00224
00225     Returns:
00226         Nothing
00227
00228     Doc Author:
00229         Willem van der Schans, Trelent AI
00230     """
00231         self.dataframe = None
00232         self.__batches = 0
00233         self.  siteClass = siteClass
00234         self.__headerDict = None
00235         self.__parameterString = ""
00236         self.__appendFile = None
00237         self.__dateStart = None
00238         self.__dateEnd = None
00239         self.  restDomain =
'https://resoapi.utahrealestate.com/reso/odata/Property?'
00240         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00241             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00242         self.filePath =
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00243             "Security").joinpath("auth.json")
00244         self.key = None
00245
00246         try:
00247             self.mainFunc()
00248         except KeyError as e:
00249             # This allows for user cancellation of the program using the quit
button
00250             if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00251                 RESTError(1101)
00252                 print(e)
00253                 pass
00254         except AttributeError as e:
00255             if e is not None:
00256                 print(
00257                     f"UtahRealEstate/Core.py | Error = {e} | Authentication
Error | Please update keys in AuthUtil")
00258                 RESTError(401)
00259                 pass
00260             else:
00261                 pass
00262         except Exception as e:
00263             print(e)
00264             RESTError(1001)
00265             raise SystemExit(1001)
00266
```

Member Function Documentation

def Core.UtahRealEstateMain.__getCount (*self*) `[private]`

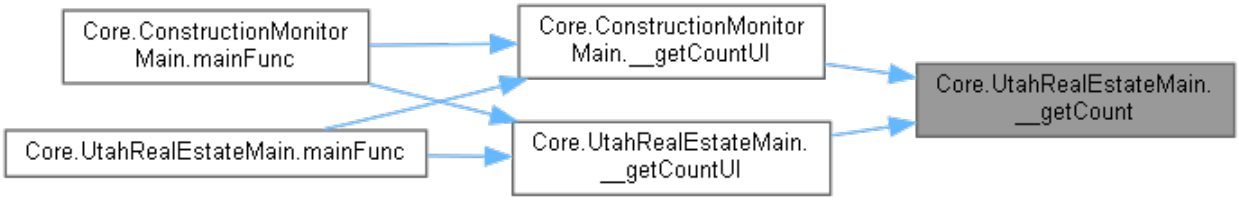
```
The __getCount function is used to determine the number of records that will be returned
by the query.
This function is called when a user calls the count() method on a ReST object. The
__getCount function uses
the $count parameter in OData to return only an integer value representing how many
records would be returned
by the query.

Args:
self: Represent the instance of the class

Returns:
The number of records in the data set
```

```
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 371 of file UtahRealEstate/Core.py.

```
00371     def __getCount(self):
00372         """
00373     The __getCount function is used to determine the number of records that will
be returned by the query.
00374     This function is called when a user calls the count() method on a ReST object.
The __getCount function uses
00375     the $count parameter in OData to return only an integer value representing
how many records would be returned
00376     by the query.
00377
00378     Args:
00379         self: Represent the instance of the class
00380
00381     Returns:
00382         The number of records in the data set
00383
00384     Doc Author:
00385         Willem van der Schans, Trelent AI
00386     """
00387         __count_resp = None
00388
00389         try:
00390             __count_resp =
requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00391                                         headers=self.__headerDict)
00392
00393             if __count_resp.status_code != 200:
00394                 RESTError(__count_resp)
00395                 raise SystemExit(0)
00396
00397             self.__record_val = int(__count_resp.json()["@odata.count"])
00398
00399         except requests.exceptions.Timeout as e:
00400             print(e)
00401             RESTError(790)
00402             raise SystemExit(790)
00403         except requests.exceptions.TooManyRedirects as e:
00404             print(e)
00405             RESTError(791)
00406             raise SystemExit(791)
00407         except requests.exceptions.MissingSchema as e:
00408             print(e)
00409             RESTError(1101)
00410         except requests.exceptions.RequestException as e:
00411             print(e)
00412             RESTError(405)
00413             raise SystemExit(405)
00414
```

Here is the caller graph for this function:



def Core.UtahRealEstateMain.__getCountUI (*self*) `[private]`

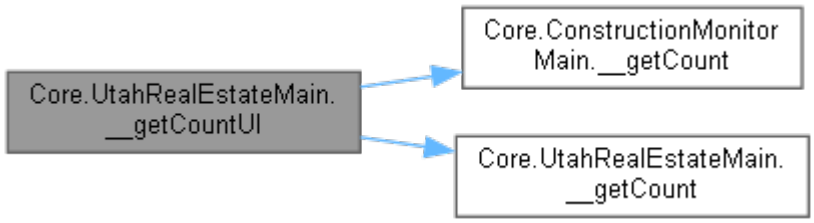
```
The __getCountUI function is a wrapper for the __getCount function.
It creates a progress window and updates it while the __getCount function runs.
The purpose of this is to keep the GUI responsive while running long processes.

Args:
self: Represent the instance of the class
```

```
Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

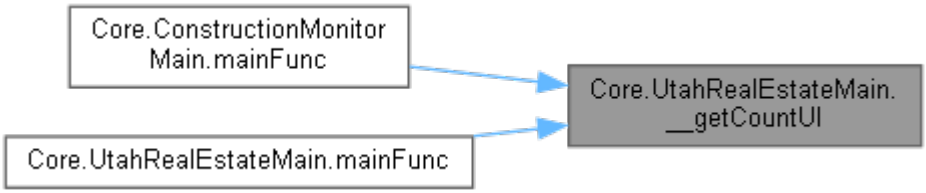
Definition at line 415 of file UtahRealEstate/Core.py.

```
00415     def __getCountUI(self):
00416 
00417         """
00418     The __getCountUI function is a wrapper for the __getCount function.
00419     It creates a progress window and updates it while the __getCount function
runs.
00420     The purpose of this is to keep the GUI responsive while running long processes.
00421 
00422     Args:
00423         self: Represent the instance of the class
00424 
00425     Returns:
00426         A popupwrapped object
00427 
00428     Doc Author:
00429         Willem van der Schans, Trelent AI
00430     """
00431         uiObj = PopupWrapped(text="Batch request running",
windowType="progress", error=None)
00432 
00433         threadGui = threading.Thread(target=self.__getCount,
00434                                      daemon=False)
00435         threadGui.start()
00436 
00437         while threadGui.is_alive():
00438             uiObj.textUpdate()
00439             uiObj.windowPush()
00440         else:
00441             uiObj.stopWindow()
```

Here is the call graph for this function:



Here is the caller graph for this function:



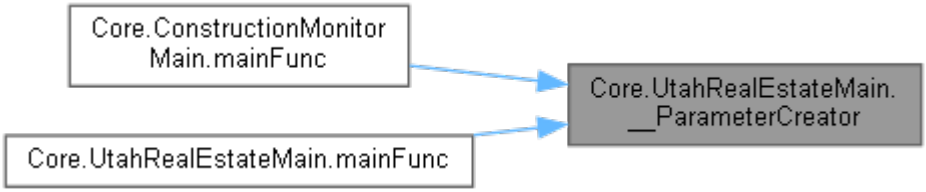
def Core.UtahRealEstateMain.__ParameterCreator (*self*) `[private]`

```
The __ParameterCreator function is used to create the filter string for the ReST API
call.
The function takes in a siteClass object and extracts all of its parameters into a
dictionary.
It then creates an appropriate filter string based on those parameters.

Args:
self: Bind the object to the class

Returns:
A string to be used as the parameter in the api call

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 327 of file UtahRealEstate/Core.py.

```
00327     def __ParameterCreator(self):
00328         """
00329     The __ParameterCreator function is used to create the filter string for the
ReST API call.
00330     The function takes in a siteClass object and extracts all of its parameters
into a dictionary.
00331     It then creates an appropriate filter string based on those parameters.
00332 
00333     Args:
00334         self: Bind the object to the class
00335 
00336     Returns:
00337         A string to be used as the parameter in the api call
00338 
00339     Doc Author:
00340         Willem van der Schans, Trelent AI
00341     """
00342         filter_string = ""
00343 
00344         __Source_dict = {key: value for key, value in
self.__siteClass.__dict__.items() if
00345                          not key.startswith('__') and not callable(key)}
00346 
00347         self.__appendFile = __Source_dict["append_file"]
00348         __Source_dict.pop("append_file")
00349 
00350         temp_dict = copy.copy(__Source_dict)
00351         for key, value in temp_dict.items():
00352             if value is None:
00353                 __Source_dict.pop(key)
00354             else:
00355                 pass
00356 
00357         if __Source_dict["ListedOrModified"] == "Listing Date":
00358             filter_string =
f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingC
ontractDate%20le%20{__Source_dict['dateEnd']}"
00359         elif __Source_dict["ListedOrModified"] == "Modification Date":
00360             filter_string =
f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20a
nd%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00361         elif __Source_dict["ListedOrModified"] == "Close Date":
00362             filter_string =
f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{
__Source_dict['dateEnd']}"
00363 
00364         filter_string = filter_string +
f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['Stan
dardStatus']}'"
00365 
00366         if __Source_dict["select"] is not None:
00367             filter_string = filter_string +
f'&$select={__Source_dict["select"]}'
00368 
00369         self.__parameterString = filter_string
00370 
```

Here is the caller graph for this function:



def Core.UtahRealEstateMain.mainFunc (*self*)

```
The mainFunc function is the main function of this module. It will be called by the
GUI when a user clicks on
```

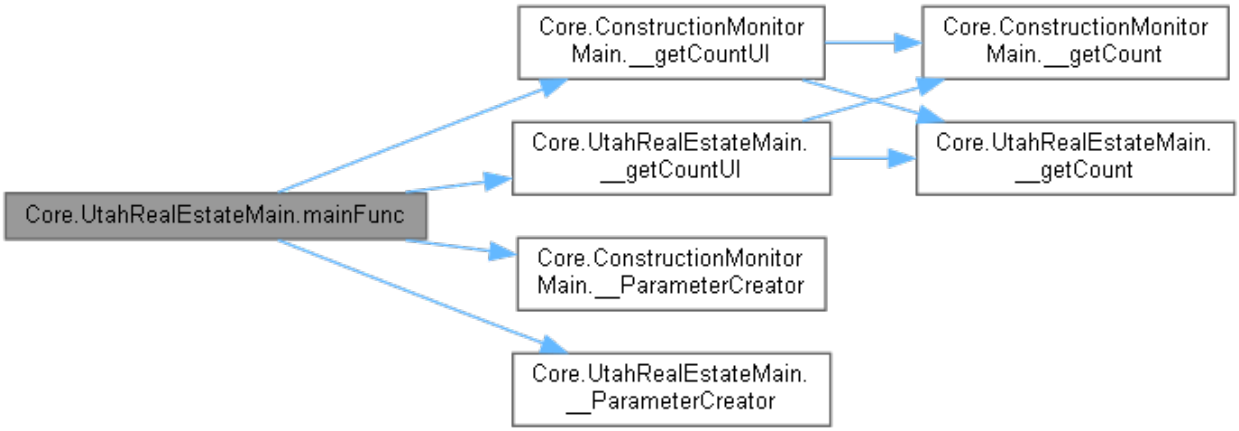
```
the &amp;quot;Run&amp;quot; button in the GUI. The mainFunc function should contain all of your
code for running your program, and it
should return a dataframe that contains all the data you want to display in your final
report.

Args:
self: Reference the object itself

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 267 of file UtahRealEstate/Core.py.

```
00267     def mainFunc(self):
00268
00269         """
00270     The mainFunc function is the main function of this module. It will be called
by the GUI when a user clicks on
00271     the &amp;quot;Run&amp;quot; button in the GUI. The mainFunc function should contain
all of your code for running your program, and it
00272     should return a dataframe that contains all the data you want to display in
your final report.
00273
00274     Args:
00275         self: Reference the object itself
00276
00277     Returns:
00278         A dataframe
00279
00280     Doc Author:
00281         Willem van der Schans, Trelent AI
00282     """
00283         passFlag = False
00284
00285         while not passFlag:
00286             if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00287                 try:
00288                     f = open(self.keyPath, "rb")
00289                     key = f.readline()
00290                     f.close()
00291                     f = open(self.filePath, "rb")
00292                     authDict = json.load(f)
00293                     fernet = Fernet(key)
00294                     authkey =
fernet.decrypt(authDict["ure"]["auth"]).decode()
00295                     self.__headerDict = {authDict["ure"]["parameter"]:
authkey}
00296                     passFlag = True
00297                 except Exception as e:
00298                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening
AuthUtil")
00299                     AuthUtil()
00300             else:
00301                 AuthUtil()
00302
00303         self.__ParameterCreator()
00304
00305         self.__getCountUI()
00306
00307         self.__batches = BatchCalculator(self.__record_val, None)
00308
00309         if self.__batches != 0:
00310             startTime = datetime.datetime.now().replace(microsecond=0)
00311             BatchInputGui(self.__batches)
00312             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
00313             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00314
ParameterDict=self.__parameterString,
00315                                               HeaderDict=self.__headerDict,
00316                                               BatchesNum=self.__batches,
00317                                               Type="utah_real_estate")
```

```
00318             BatchGuiObject.BatchGuiShow()
00319             self.dataframe = BatchGuiObject.dataframe
00320             print(
00321                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00322             FileSaver("ure", self.dataframe, self.__appendFile)
00323         else:
00324             RESTError(994)
00325             raise SystemExit(994)
00326
```

Here is the call graph for this function:



Member Data Documentation

Core.UtahRealEstateMain.__appendFile [private]

Definition at line 236 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__batches [private]

Definition at line 232 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__dateEnd [private]

Definition at line 238 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__dateStart [private]

Definition at line 237 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__headerDict [private]

Definition at line 234 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__parameterString [private]

Definition at line 235 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__record_val [private]

Definition at line 397 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__restDomain [private]

Definition at line 239 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.__siteClass [private]

Definition at line 233 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.dataframe

Definition at line 231 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.filePath

Definition at line 242 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.key

Definition at line 244 of file UtahRealEstate/Core.py.

Core.UtahRealEstateMain.keyPath

Definition at line 240 of file UtahRealEstate/Core.py.

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

- UtahRealEstate/Core.py

Index