Gardner API Utility Documentation

Willem van der Schans

Version 1.0.2
4/17/2023 3:33:00 PM

Table of Contents

Readme: Gardner Policy Institute API Utility

Author: Willem van der Schans

Commissioner: Gardner Policy Institute

Description: A Python utility for generating API requests from ConstructionMonitor.com, Utah Real Estate.com, Realtor.com, and the US Census APIs

Notes

1. No functionality for macOS or Linux has been developed or is planned for the future.
2. Documentation is available within the repository.

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: All dependencies are included in the Windows installer

Authentication Requirements

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

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

Changelog

Initial release

Version: 1.0.0
Date: 2023-04-08

Core Functionality

```
1. Optimized support for ConstructionMonitor.com,
    Utah Real Estate.com, Realtor.com, and the US Census APIs
2. Optimized support for generating API requests based on custom input parameters
```

User InterFace

```
1. Optimized ui multithreading for faster processing
2. Simplified user interface for better usability and user experience
```

File Functionality

```
1. Added file browsing support to enhance appending accessibility
```

Logging and Error Handling

```
1. Enhanced logging capabilities for code transparency and easy maintenance

2. Enhanced error handling and exception reporting to prevent hard locks
    while using the programs.
```

Security

```
1. Enhanced security measures for handling sensitive user data using
    locally generated keys
```

GUI

```
1. Improved user interface threading for better usability and error handling
```

Other

```
1. Fixed bugs and issues found in QA
```

Version: 0.9.5
Date: 2023-04-05

```
Improved documentation and code readability for easier use and maintenance
Fixed bugs and issues found in QA
```

Version: 0.9.0
Date: 2023-03-16

```
Enhanced Mainloop and interaction with spawned threads allowing for multiple API requests
to be completed in sequence.
Initial Github Commit
```

Version: 0.8.0
Date: 2023-03-12

```
Added new utility functions for data cleaning, and appending
```

Version: 0.7.0
Date: 2023-03-02

```
Implemented secure storage of authorization keys using an Authorization Utility and
encryption
```

Version: 0.6.0
Date: 2023-02-25

```
Enhanced GUI Utility Improvements.
    - Descriptive processing pop-ups
    - Warning popups
    - MultiThreading
    - Completion Time Estimation
```

Version: 0.5.0
Date: 2023-02-15

```
Added GUI to utility to enhance end-user accesibility.
```

Version: 0.4.0
Date: 2022-12-07

```
Enhanced data processing and analysis functions for more accurate results
    1. Added support for appending to existing new documents to existing CSV files
    2. Added support for storing pulled data in CSV files
Improved user interface for better usability and user experience
```

Version: 0.3.0
Date: 2022-11-15

```
Added support for interacting with Realtor.com
Added support for interacting with ffiec.cfpb.gov [census] API.
```

Version: 0.2.0
Date: 2022-11-03

```
Improved batch processing for interacting with the ConstructionMonitor.com API
Improved batch processing for interacting with the Utah Real Estate.com API
```

Version: 0.1.0
Date: 2022-10-25

```
Added support for interacting with the ConstructionMonitor.com API
```

Version: 0.0.0
Date: 2022-10-15

```
Added Support for UtahRealEstate.com
Added new utility functions for data processing and manipulation
Added documentation for all functions and classes
Improved support for interacting with the US Census API
```

Class Index

Class List

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

File Index

File List

Here is a list of all documented files with brief descriptions:

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 18 of file AuthUtil.py.

Constructor & Destructor Documentation

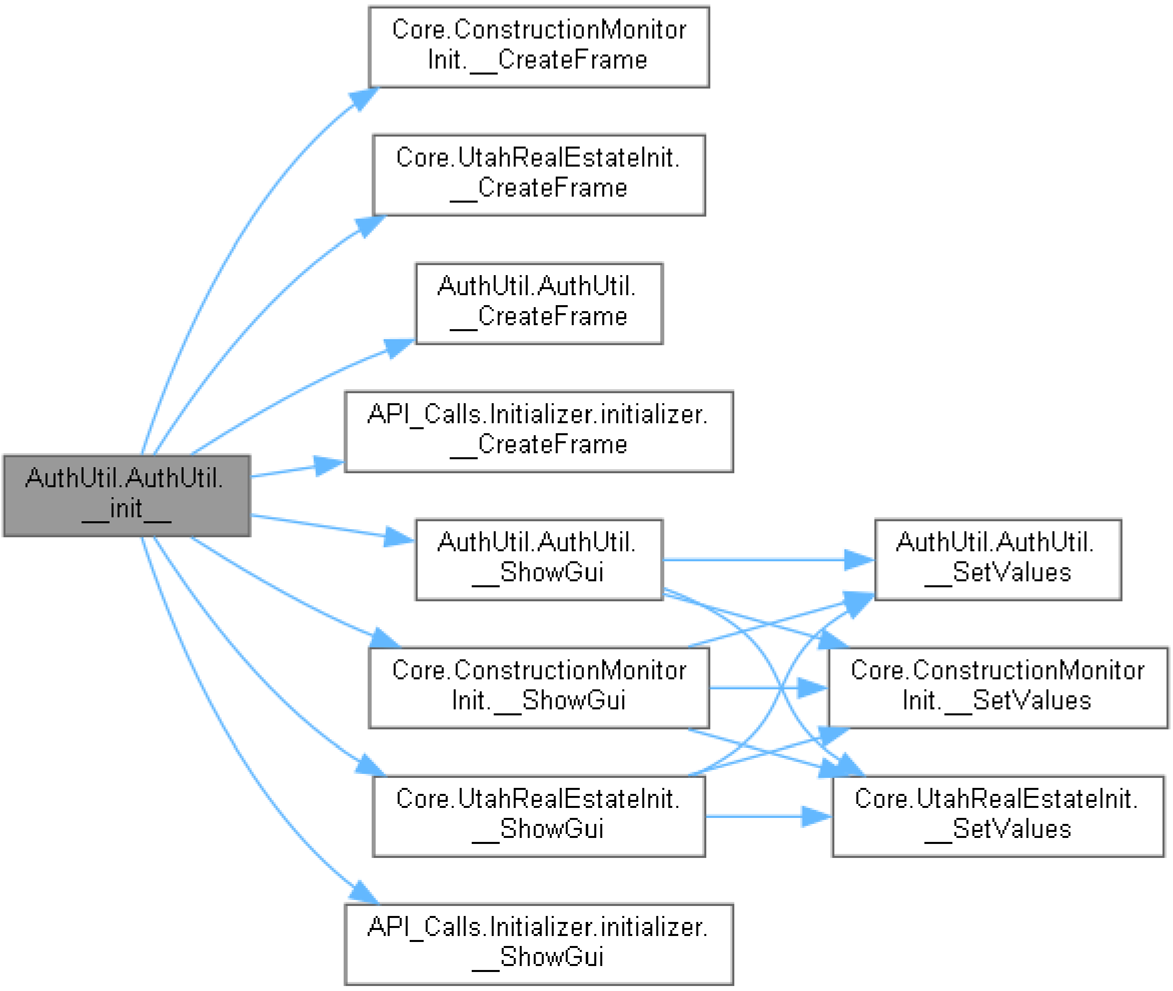
def AuthUtil.AuthUtil.__init__ (*self*)

```
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 20 of file AuthUtil.py.

```
00020     def __init__(self):
00021
00022         """
00023     The __init__ function is called when the class is instantiated.
00024     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.
00025
```

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

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.file_name = None
        self.append_file = None
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
        self.k = None
        self.keyFlag = True
        self.jsonDict = {}
        self.passFlagUre = False
        self.passFlagCm = False
        self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
                           "Submitting will overwrite any old values in an unrecoverable manner."

        if os.path.exists(self.filePath):
            pass
        else:
            if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
                os.mkdir(self.filePath)
            else:
                os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
                os.mkdir(self.filePath)

        if os.path.exists(self.keyPath):
            pass
        else:
            if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
                os.mkdir(self.keyPath)
            else:
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    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...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:
            ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                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...")
            pass

```
Here is the call graph for this function:



Core.ConstructionMonitorInit.__CreateFrame
Core.UtahRealEstateInit.__CreateFrame
AuthUtil.AuthUtil.__CreateFrame
API_Calls.Initializer.initializer.__CreateFrame
AuthUtil.AuthUtil.__init__
AuthUtil.AuthUtil.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__ShowGui
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__ShowGui
Core.UtahRealEstateInit.__SetValues
API_Calls.Initializer.initializer.__ShowGui

Member Function Documentation

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

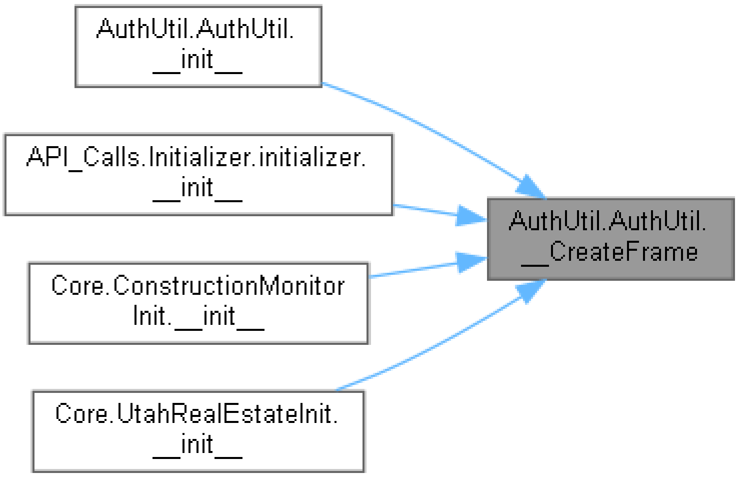
```
The __CreateFrame function creates the GUI layout for the Authentication Utility.
It is called by __init__ and returns a list of lists that contains all the elements
that will be displayed in the window.

Args:
```

```
self: Access the class attributes and methods

Returns:
A list of lists

Doc Author:
Trelent
```

Definition at line 224 of file AuthUtil.py.

```
00224     def __CreateFrame(self):
00225         """
00226     The __CreateFrame function creates the GUI layout for the Authentication Utility.
00227     It is called by __init__ and returns a list of lists that contains all the elements
00228     that will be displayed in the window.
00229
00230     Args:
00231         self: Access the class attributes and methods
00232
00233     Returns:
00234         A list of lists
00235
00236     Doc Author:
00237         Trelent
00238     """
00239         sg.theme('Default1')
00240
00241         line00 = [sg.HSeparator()]
00242
00243         line0 = [sg.Image(ImageLoader("logo.png")),
00244                  sg.Push(),
00245                  sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"), justification="center"),
00246                  sg.Push(),
00247                  sg.Push()]
00248
00249         line1 = [sg.HSeparator()]
00250
00251         line2 = [sg.Push(),
00252                  sg.Text("Utah Real Estate Key: ", justification="center"),
00253                  sg.Push()]
00254
00255         line3 = [sg.Push(),
00256                  sg.Input(default_text="", key="-ureAuth-", disabled=False,
00257                           size=(40, 1)),
00258                  sg.Push()]
00259
00260         line4 = [sg.HSeparator()]
00261
00262         line5 = [sg.Push(),
00263                  sg.Text("Construction Monitor Key: ", justification="center"),
00264                  sg.Push()]
00265
00266         line6 = [sg.Push(),
00267                  sg.Input(default_text="", key="-cmAuth-", disabled=False,
00268                           size=(40, 1)),
00269                  sg.Push()]
00270
00271         line7 = [sg.HSeparator()]
00272
00273         line8 = [sg.Push(),
00274                  sg.Text(self.outcomeText, justification="center"),
00275                  sg.Push()]
00276
00277         line9 = [sg.HSeparator()]
00278
00279         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00280
00281         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]
00282
00283         return layout
```

Here is the caller graph for this function:

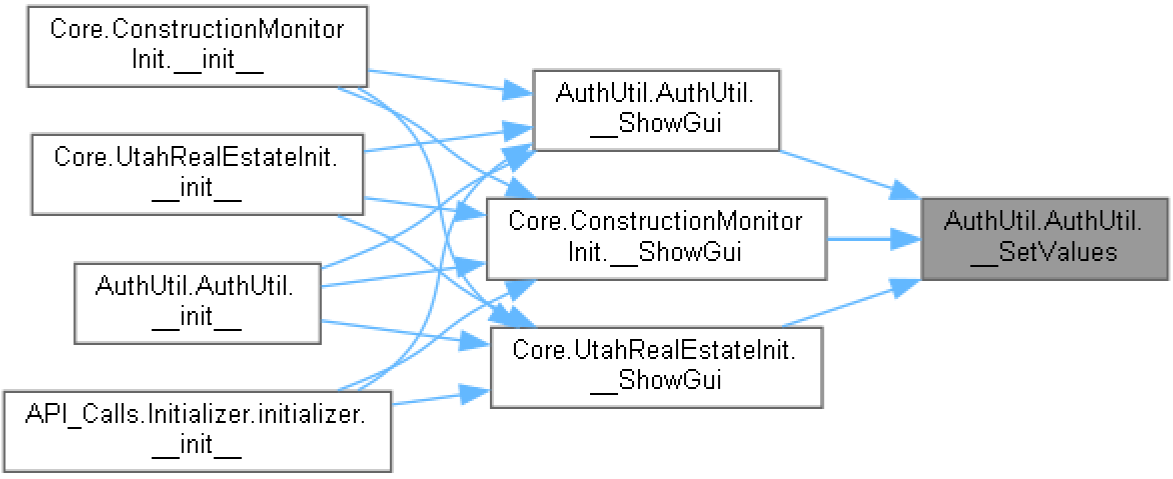


def AuthUtil.AuthUtil.__SetValues (***self*****,** ***values*****)** `[private]`

```
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 104 of file AuthUtil.py.

```
    def __SetValues(self, values):

        """
    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
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    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")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    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")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
            self.passFlagCm = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction Monitor",
                         windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate", windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Construction Monitor", windowType="errorLarge")
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

```
Here is the caller graph for this function:

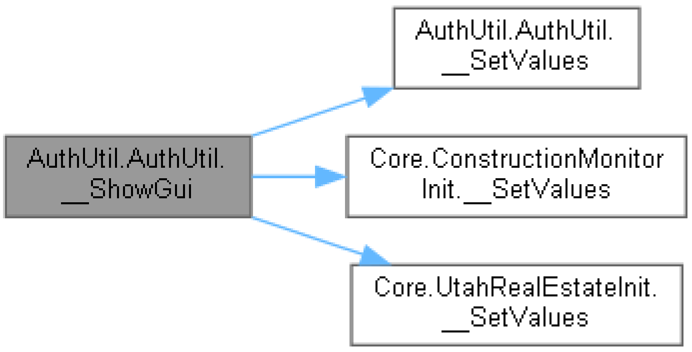

Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
AuthUtil.AuthUtil. __SetValues

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

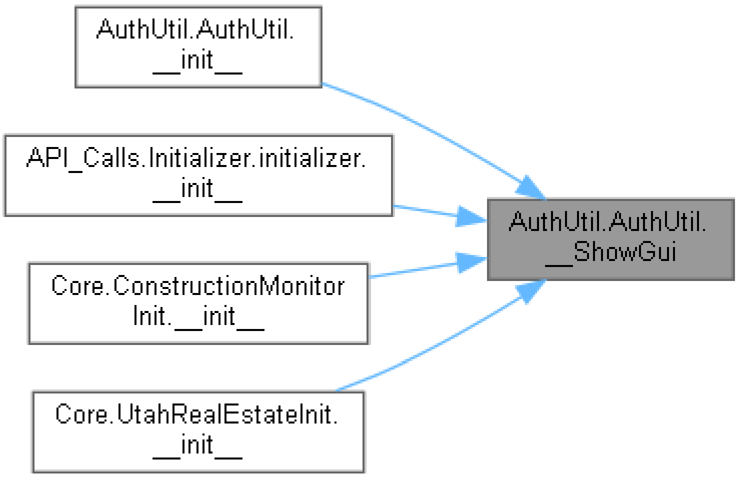
```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the 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 185 of file AuthUtil.py.

```
    def __ShowGui(self, layout, text):

        """
    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
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while not self.passFlagUre or not self.passFlagCm:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                except Exception as e:
                    print(e)
                    RESTError(993)
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

```
Here is the call graph for this function:


AuthUtil.AuthUtil.__SetValues
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__ShowGui

Member Data Documentation

AuthUtil.AuthUtil.append_file

Definition at line 38 of file AuthUtil.py.

AuthUtil.AuthUtil.file_name

Definition at line 37 of file AuthUtil.py.

AuthUtil.AuthUtil.filePath

Definition at line 40 of file AuthUtil.py.

AuthUtil.AuthUtil.jsonDict

Definition at line 43 of file AuthUtil.py.

AuthUtil.AuthUtil.k

Definition at line 41 of file AuthUtil.py.

AuthUtil.AuthUtil.keyFlag

Definition at line 42 of file AuthUtil.py.

AuthUtil.AuthUtil.keyPath

Definition at line 39 of file AuthUtil.py.

AuthUtil.AuthUtil.ListedOrModified

Definition at line 36 of file AuthUtil.py.

AuthUtil.AuthUtil.outcomeText

Definition at line 46 of file AuthUtil.py.

AuthUtil.AuthUtil.passFlagCm

Definition at line 45 of file AuthUtil.py.

AuthUtil.AuthUtil.passFlagUre

Definition at line 44 of file AuthUtil.py.

AuthUtil.AuthUtil.StandardStatus

Definition at line 35 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 41 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 43 of file BatchProcessing.py.

```
00043     def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict,
ColumnSelection, valueObject):
00044
00045         """
00046     The __init__ function is the constructor for a class. It is called when an
object of that class
00047     is created, and it sets up the attributes of that object. In this case, we
are setting up our
00048     object to have a dataframe attribute (which will be used to store all of our
data), as well as
00049     attributes for each parameter in our ReST call.
00050
00051     Args:
00052         self: Represent the instance of the class
00053         RestDomain: Specify the domain of the rest api
00054         NumBatches: Determine how many batches of data to retrieve
00055         ParameterDict: Pass in the parameters that will be used to make the api
call
00056         HeaderDict: Pass the header dictionary from the main function to this
class
00057         ColumnSelection: Determine which columns to pull from the api
00058         valueObject: Pass in the value object that is used to determine what
values are returned
00059
00060     Returns:
00061         An object of the class
00062
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
00066         self.dataframe = None
00067         self.__numBatches = NumBatches
00068         self.__parameterDict = ParameterDict
00069         self.__restDomain = RestDomain
00070         self.__headerDict = HeaderDict
00071         self.__columnSelection = ColumnSelection
00072         self.valueObject = valueObject
00073         self.__maxRequests = 10000
00074         self.__requestCount = math.ceil(self.__numBatches /
(self.__maxRequests / int(self.__parameterDict['size'])))
00075         self.__requestCalls = math.ceil(self.__maxRequests /
int(self.__parameterDict['size']))
00076         self.__dateTracker = None
00077
```

Member Function Documentation

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

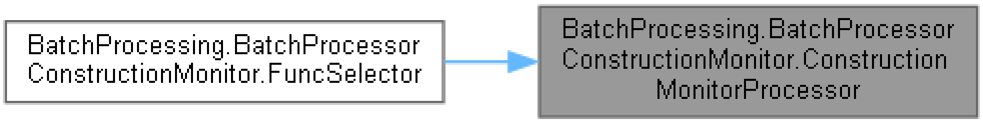
```
The ConstructionMonitorProcessor function will use requests to get data from
ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called
__df (which is local). This
process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at 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 94 of file BatchProcessing.py.

```
00094     def ConstructionMonitorProcessor(self, valueObject):
00095         """
```

```
    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
    """
        __df = None
        for callNum in range(0, self.__requestCount):
            self.__parameterDict["from"] = 0

            if self.__requestCount > 1 and callNum != self.__requestCount - 1:
                __batchNum = self.__requestCalls
                if __df is None:
                    self.__dateTracker = str(date.today())
                else:
                    self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
            elif self.__requestCount == 1:
                __batchNum = self.__numBatches
                self.__dateTracker = str(date.today())
            else:
                __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])) - (
                        self.__requestCount - 1)
                self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')

            self.__parameterDict['dateEnd'] = self.__dateTracker

            for record in range(0, int(math.ceil(__batchNum))):
                if record != 0:
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        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}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)


```

Here is the caller graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

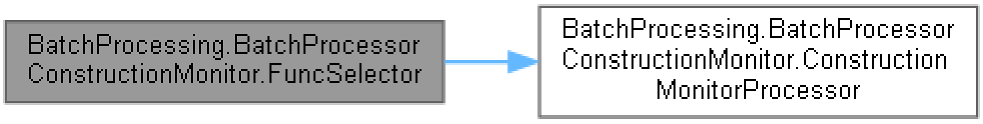
def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (*self*)

```
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 78 of file BatchProcessing.py.

```
    def 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
    """
        self.ConstructionMonitorProcessor(self.valueObject)

```

Here is the call graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

Member Data Documentation

BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection [private]

Definition at line 71 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker `[private]`

Definition at line 76 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__headerDict `[private]`

Definition at line 70 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests `[private]`

Definition at line 73 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__numBatches `[private]`

Definition at line 67 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict `[private]`

Definition at line 68 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls `[private]`

Definition at line 75 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__requestCount `[private]`

Definition at line 74 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.__restDomain `[private]`

Definition at line 69 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.dataframe

Definition at line 66 of file BatchProcessing.py.

BatchProcessing.BatchProcessorConstructionMonitor.valueObject

Definition at line 72 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 172 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 174 of file BatchProcessing.py.

```
00174     def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict,
valueObject):
00175         """
00176     The __init__ function is the constructor for a class. It is called when an
object of that class
00177     is instantiated, and it sets up the attributes of that object. In this case,
we are setting up
00178     the dataframe attribute to be None (which will be set later), and we are also
setting up some
00179     other attributes which will help us make our API calls.
```

```
00180
00181     Args:
00182         self: Represent the instance of the class
00183         RestDomain: Specify the domain of the rest api
00184         NumBatches: Determine how many batches of data to pull from the api
00185         ParameterString: Pass the parameters to the rest api
00186         HeaderDict: Pass in the header information for the api call
00187         valueObject: Create a dataframe from the json response
00188
00189     Returns:
00190         The instance of the class
00191
00192     Doc Author:
00193         Willem van der Schans, Trelent AI
00194     """
00195         self.dataframe = None
00196         self.__numBatches = NumBatches
00197         self.__parameterString = ParameterString
00198         self.__restDomain = RestDomain
00199         self.__headerDict = HeaderDict
00200         self.valueObject = valueObject
00201
```

Member Function Documentation

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

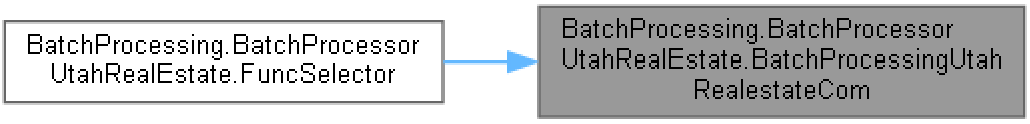
```
The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary information for this
function to work properly. The BatchProcessingUtahRealestateCom function will then use requests 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 219 of file BatchProcessing.py.

```
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221     The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it to
00222         update the progress bar. It also takes in self, which contains all the necessary information for this
00223         function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to get data from
00224         UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is local). This
00225         process will be repeated until all the data has been collected from UtahRealestate.com's ReST API, at which point __df will contain all
00226
00227     Args:
00228         self: Represent the instance of the class
00229         valueObject: Pass the value of a progress bar to the function
00230
00231     Returns:
00232         A dataframe of the scraped data
```

```

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __df = pd.DataFrame()

        for batch in range(self.__numBatches):

            if batch == 0:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
                                        headers=self.__headerDict)

                response_temp = response.json()
                __df = pd.json_normalize(response_temp, record_path=['value'])

            else:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch * 200}",
                                        headers=self.__headerDict)

                response_temp = response.json()
                response_temp = pd.json_normalize(response_temp, record_path=['value'])
                __df = pd.concat([__df, response_temp], ignore_index=True)

            valueObject.setValue(valueObject.getValue() + 1)

        self.dataframe = __df
        valueObject.setValue(-999)
```

Here is the caller graph for this function:

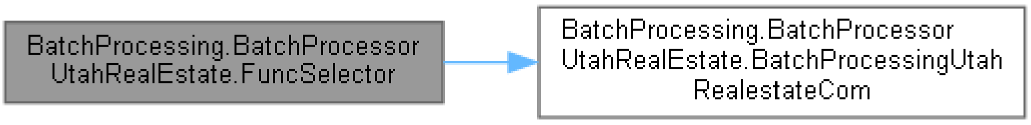

BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector
BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom

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 202 of file BatchProcessing.py.

```
    def 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
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

```
Here is the call graph for this function:


BatchProcessing.BatchProcessor
UtahRealEstate.FuncSelector
BatchProcessing.BatchProcessor
UtahRealEstate.BatchProcessingUtah
RealestateCom

Member Data Documentation

BatchProcessing.BatchProcessorUtahRealEstate.__headerDict [private]

Definition at line 199 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__numBatches [private]

Definition at line 196 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__parameterString [private]

Definition at line 197 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.__restDomain [private]

Definition at line 198 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.dataframe

Definition at line 195 of file BatchProcessing.py.

BatchProcessing.BatchProcessorUtahRealEstate.valueObject

Definition at line 200 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 17 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 19 of file BatchProgressGUI.py.

```
00019     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, 
ColumnSelection=None):
00020 
00021         """
00022     The __init__ function is the first function that gets called when an object 
of this class is created.
00023     It initializes all the variables and sets up a layout for the GUI. It also 
creates a window to display
00024     the dataframe in.
00025 
00026     Args:
00027         self: Represent the instance of the class
00028         BatchesNum: Determine the number of batches that will be created
00029         RestDomain: Specify the domain of the rest api
00030         ParameterDict: Pass the parameters of the request to the class
00031         HeaderDict: Store the headers of the dataframe
00032         Type: Determine the type of dataframe that is being created
00033         ColumnSelection: Select the columns to be displayed in the gui
00034 
00035     Returns:
00036         Nothing
00037 
00038     Doc Author:
00039         Willem van der Schans, Trelent AI
00040     """
00041         self.__parameterDict = ParameterDict
00042         self.__restDomain = RestDomain
00043         self.__headerDict = HeaderDict
00044         self.__columnSelection = ColumnSelection
00045         self.__type = Type
00046         self.dataframe = None
00047 
00048         self.__layout = None
00049         self.__batches = BatchesNum
00050         self.__window = None
00051         self.__batch_counter = 0
00052 
```

Member Function Documentation

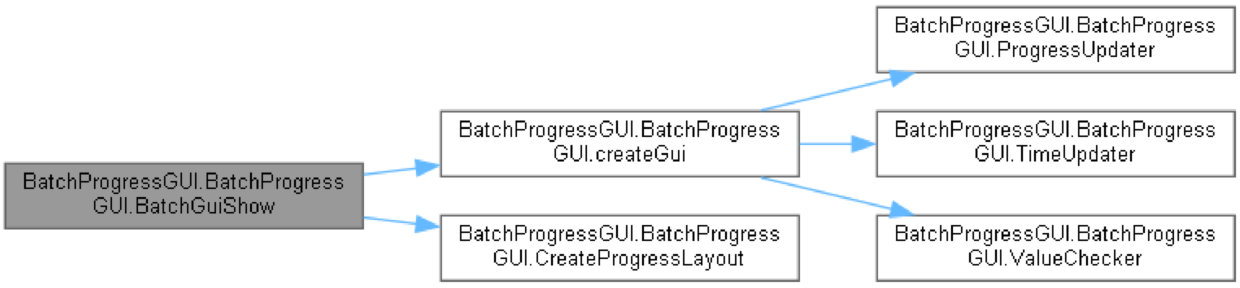
def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (*self*)

```
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 53 of file BatchProgressGUI.py.

```
00053     def BatchGuiShow(self):
00054         """
00055     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.
00056 
00057     Args:
00058         self: Represent the instance of the class
00059 
00060     Returns:
00061         The __type of the batchgui class
00062 
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
```

```
        self.CreateProgressLayout()
        self.createGui(self.__type)

```
Here is the call graph for this function:

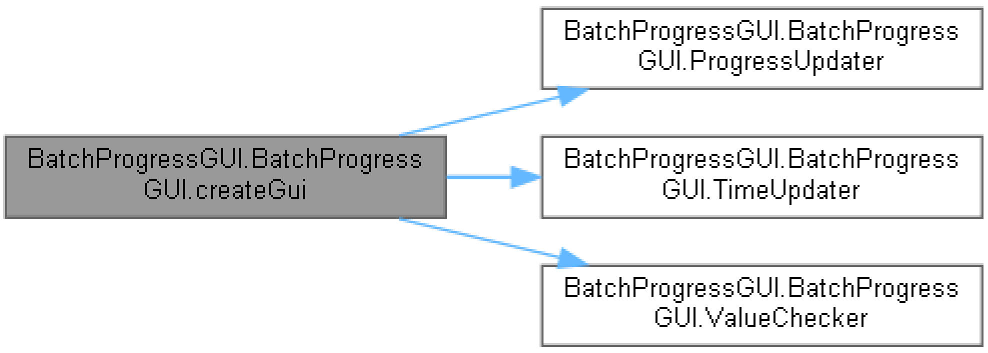

BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.CreateProgressLayout
BatchProgressGUI.BatchProgressGUI.ProgressUpdater
BatchProgressGUI.BatchProgressGUI.TimeUpdater
BatchProgressGUI.BatchProgressGUI.ValueChecker

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

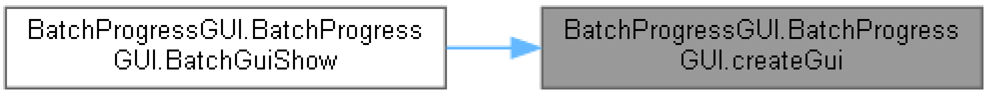
```
The createGui function is the main function that creates the GUI.
It takes in a type parameter which determines what kind of batch processor to use.
The createGui function then sets up all the variables and objects needed for
the program to run, including: window, start_time, update_text, valueObj
(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 104 of file BatchProgressGUI.py.

```
    def 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
    """
        self.__window = sg.Window('Progress', self.__layout, finalize=True,
icon=ImageLoader("taskbar_icon.ico"))

        start_time = datetime.datetime.now().replace(microsecond=0)
        update_text = f"Batch {0} completed"
        self.__window['--progress_text--'].update(update_text)
        self.__window['--progress_bar--'].update(0)
        self.__window['--time_est--'].update("Est time needed 00:00:00")

        valueObj = DataTransfer()
        valueObj.setValue(0)

        if Sourcetype == "construction_monitor":

            processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
                                                                NumBatches=self.__batches,
                                                                ParameterDict=self.__parameterDict,
                                                                HeaderDict=self.__headerDict,
                                                                ColumnSelection=self.__columnSelection,
                                                                valueObject=valueObj)
        elif Sourcetype == "utah_real_estate":
            processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
                                                           NumBatches=self.__batches,
                                                           ParameterString=self.__parameterDict,
                                                           HeaderDict=self.__headerDict,
                                                           valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

```
Here is the call graph for this function:


BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.ProgressUpdater
BatchProgressGUI.BatchProgressGUI.TimeUpdater
BatchProgressGUI.BatchProgressGUI.ValueChecker

Here is the caller graph for this function:

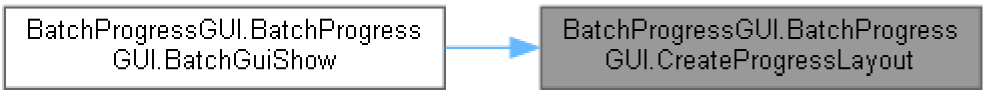

BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui

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 69 of file BatchProgressGUI.py.

```
    def 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
    """
        sg.theme('Default1')

        __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
                   sg.Push()]

        __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
                   sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]

        __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
                           key='--progress_bar--')]

        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

```
Here is the caller graph for this function:

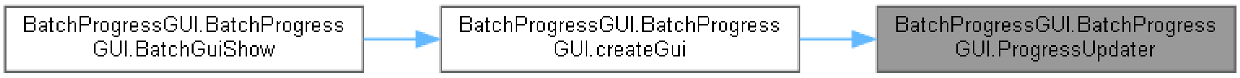

BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.CreateProgressLayout

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 188 of file BatchProgressGUI.py.

```
    def 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
    """
        if valueObj.getValue() != self.__batch_counter:
            self.__batch_counter = valueObj.getValue()

            __update_text = f"Batch {self.__batch_counter}/{self.__batches} completed"

            self.__window.write_event_value('update--progress_bar--', self.__batch_counter)
            self.__window.write_event_value('update--progress_text--', __update_text)
        else:
            pass

```
Here is the caller graph for this function:

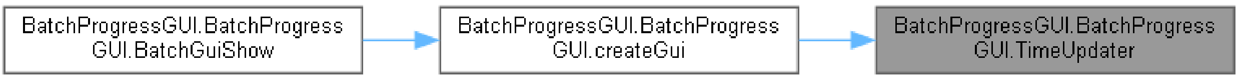

BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ProgressUpdater

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 216 of file BatchProgressGUI.py.

```
00216     def TimeUpdater(self, start_time):
00217 
00218         """
00219     The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to complete
00220     the current batch. It does this by reading the start_time variable passed in, getting the current time,
00221     calculating how much time has passed since start_time was set and then updating a timer string with that value.
00222     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.
00223 
00224     Args:
00225         self: Make the function a method of the class
00226         start_time: Get the time when the function is called
00227 
00228     Returns:
00229         A string that is updated every 0
00230 
00231     Doc Author:
00232         Willem van der Schans, Trelent AI
00233     """
00234         while True:
00235             if self.__batch_counter < self.__batches:
00236 
00237                 __current_time = datetime.datetime.now().replace(microsecond=0)
00238 
00239                 __passed_time = __current_time - start_time
00240 
00241                 __timer_string = f"Time Elapsed {__passed_time}"
00242 
00243                 try:
00244                     self.__window.write_event_value('update--timer--', __timer_string)
00245                 except AttributeError as e:
00246                     print(
00247                         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")
00248                     break
00249 
00250                 __passed_time = __passed_time.total_seconds()
00251 
00252                 try:
00253                     __time_est = datetime.timedelta(
00254                         seconds=(__passed_time * (self.__batches /
self.__batch_counter) - __passed_time)).seconds
00255                 except:
00256                     __time_est = datetime.timedelta(
00257                         seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00258
00259                 __time_est = time.strftime('%H:%M:%S',
time.gmtime(__time_est))
00260
00261                 __end_string = f"Est time needed {__time_est}"
00262                 self.__window.write_event_value('update--time_est--',
__end_string)
00263             else:
00264                 __end_string = f"Est time needed 00:00:00"
00265                 self.__window.write_event_value('update--time_est--',
__end_string)
00266             time.sleep(0.25)
00267
```
Here is the caller graph for this function:

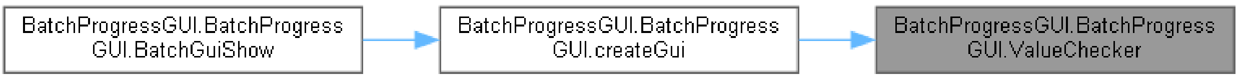

BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.TimeUpdater

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 268 of file BatchProgressGUI.py.

```
00268     def ValueChecker(self, ObjectVal):
00269         """
00270     The ValueChecker function is a thread that checks the value of an object.
00271         It will check if the value has changed, and if it has, it will return
True.
00272         If not, then it returns False.
00273
00274     Args:
00275         self: Represent the instance of the class
00276         ObjectVal: Get the value of the object
00277
00278     Returns:
00279         True if the value of the object has changed, and false if it hasn't
00280
00281     Doc Author:
00282         Willem van der Schans, Trelent AI
00283     """
00284         while True:
00285             time.sleep(0.3)
00286             if self.__batch_counter != ObjectVal.getValue():
00287                 self.__batch_counter = ObjectVal.getValue()
00288                 return True
00289             else:
00290                 return False
```
Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ValueChecker

Member Data Documentation

BatchProgressGUI.BatchProgressGUI.__batch_counter `[private]`

Definition at line 51 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__batches `[private]`

Definition at line 49 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__columnSelection `[private]`

Definition at line 44 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__headerDict `[private]`

Definition at line 43 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__layout `[private]`

Definition at line 48 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__parameterDict `[private]`

Definition at line 41 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__restDomain `[private]`

Definition at line 42 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__type `[private]`

Definition at line 45 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__window `[private]`

Definition at line 50 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.dataframe

Definition at line 46 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.

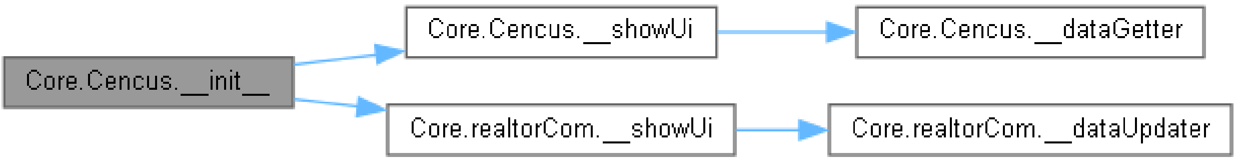
Constructor & Destructor Documentation

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.

```
    def __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
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
        self.uiString = None
        self.link = None

        self.__showUi()
        print(self.link)
        F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
        self.uiString = (
            f"ffiec.cfpb.gov (Mortgage API) request Completed \n
{self.year_arg} data retrieved \n Data Saved at {F.getPath()}")

        PopupWrapped(text=self.uiString, windowType="noticeLarge")

```
Here is the call graph for this function:



Member Function Documentation

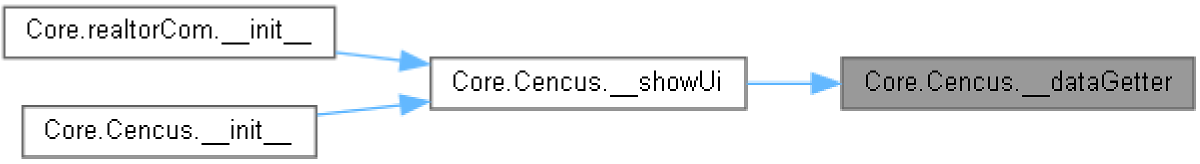
def Core.Cencus.__dataGetter (*self*) `[private]`

```
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.

```
    def __dataGetter(self):
        """
    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
    """
        arg_dict_bu = locals()

        link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link =
"https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
+ f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

Here is the caller graph for this function:



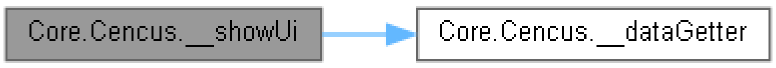
def Core.Cencus.__showUi (*self*) `[private]`

```
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
```

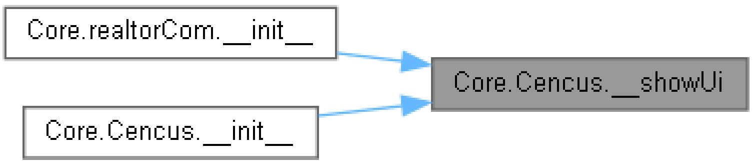
Definition at line 44 of file CFBP/Core.py.

```
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:



Here is the caller 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.

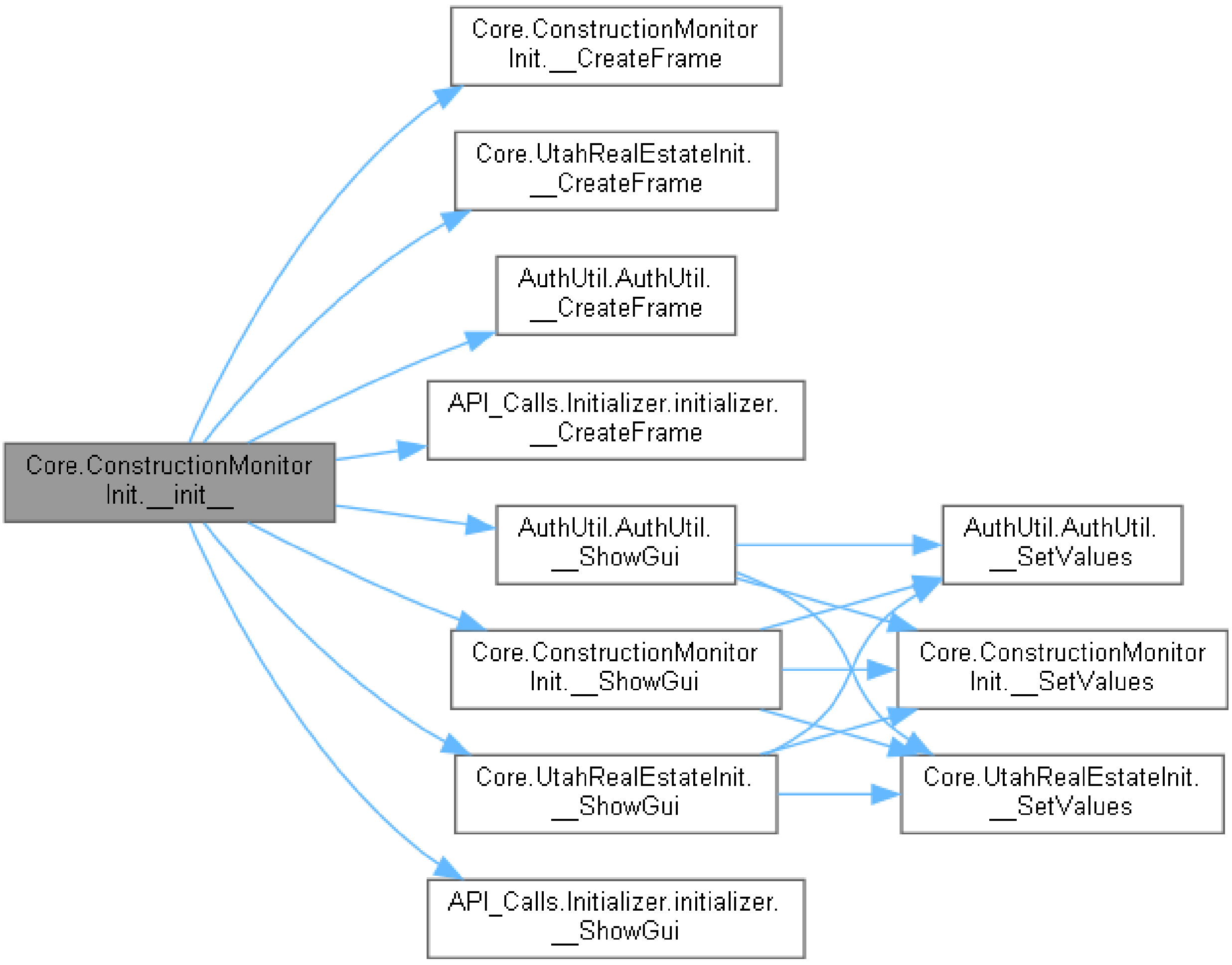
Constructor & Destructor Documentation

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.

```
    def __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
    """
        self.size = None
        self.SourceInclude = None
        self.dateStart = None
        self.dateEnd = None
        self.rest_domain = None
        self.auth_key = None
        self.ui_flag = None
        self.append_file = None

        passFlag = False

        while not passFlag:
            if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json")):
                try:
                    f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                    key = f.readline()
                    f.close()
                    f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json"), "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
                    passFlag = True
                except Exception as e:
                    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")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")

```
Here is the call graph for this function:


Core.ConstructionMonitorInit.__init__
Core.ConstructionMonitorInit.__CreateFrame
Core.UtahRealEstateInit.__CreateFrame
AuthUtil.AuthUtil.__CreateFrame
API_Calls.Initializer.initializer.__CreateFrame
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
API_Calls.Initializer.initializer.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

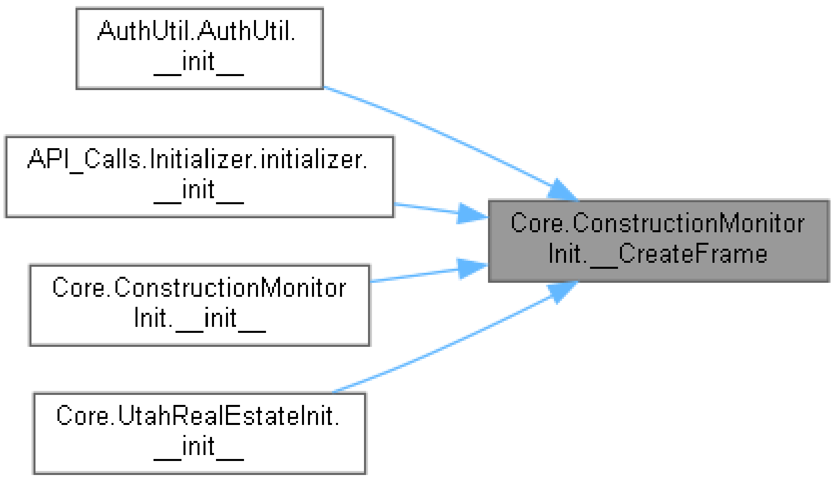
Member Function Documentation

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:

    Returns:
        The layout for the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

```
Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
Core.ConstructionMonitorInit.__CreateFrame

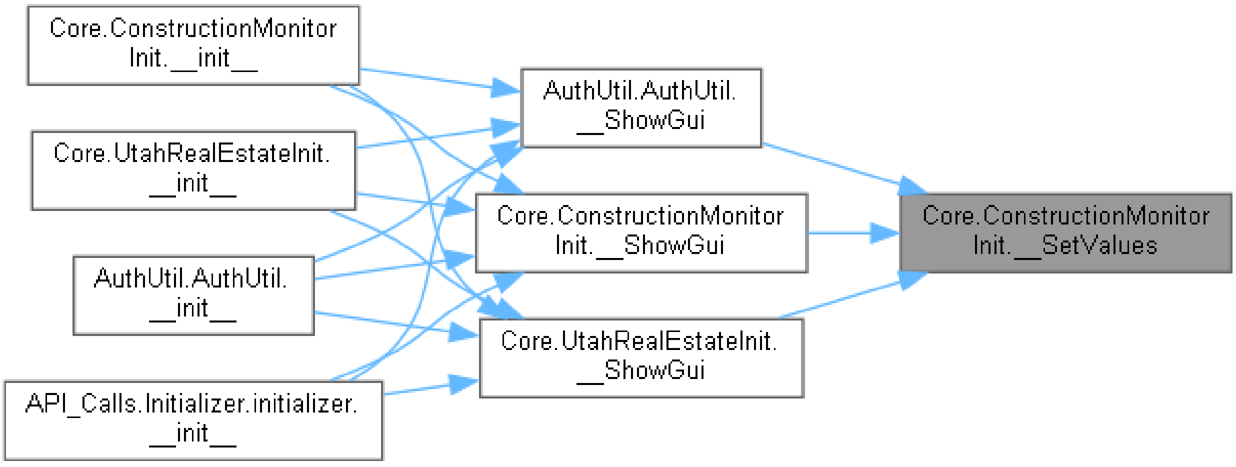
def Core.ConstructionMonitorInit.__SetValues (***self*, *values*) `[private]`**

```
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.

```
    def __SetValues(self, values):

        """
    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
    """
        self.size = 1000

        if values["-Cal-"] != "":
            self.dateStart = values["-Cal-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-EndCal-"] != "":
            self.dateEnd = values["-EndCal-"]
        else:
            self.dateEnd = date.today().strftime("%Y-%m-%d")

        self.rest_domain =
"https://api.constructionmonitor.com/v2/powersearch/?"

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True


```

Here is the caller graph for this function:


Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
Core.ConstructionMonitor Init.__SetValues

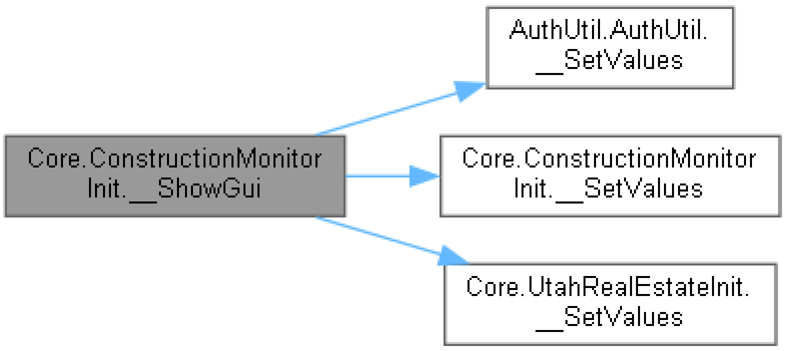
def Core.ConstructionMonitorInit.__ShowGui (*self*, *layout*, *text*) `[private]`

```
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
```

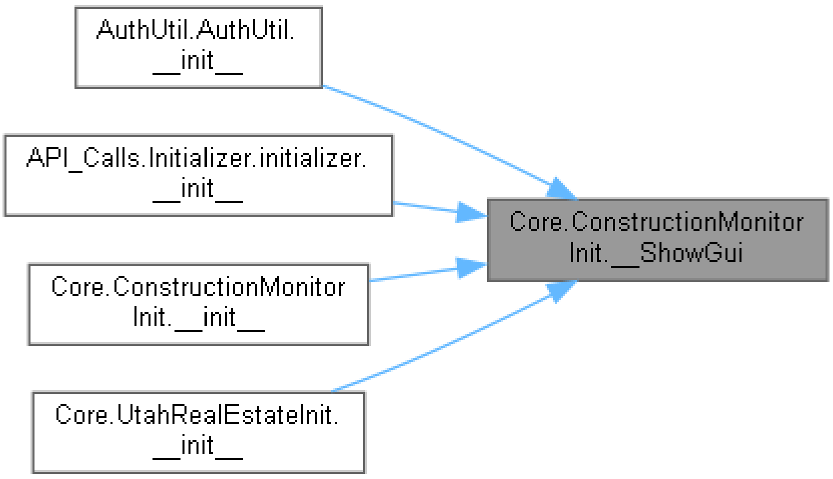
Definition at line 77 of file ConstructionMonitor/Core.py.

```
    def __ShowGui(self, layout, text):

        """
    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
    """
        window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

Here is the call graph for this function:



Here is the caller 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

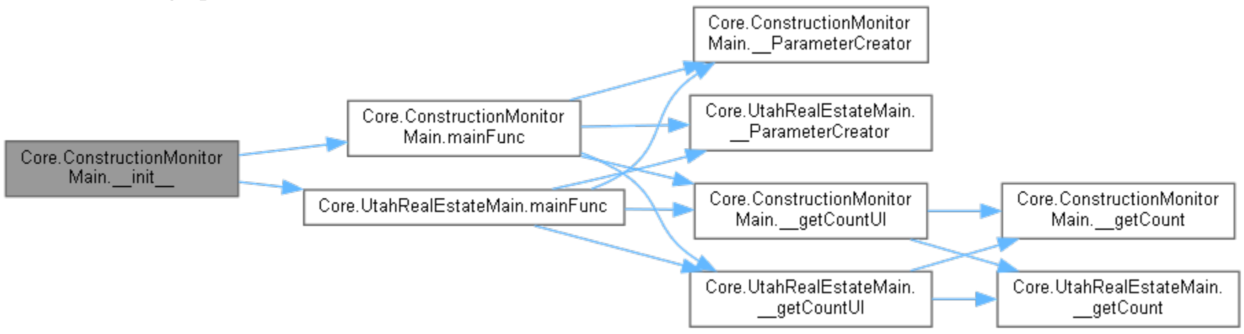
def Core.ConstructionMonitorMain.__init__ (*self*, *siteClass*)

```
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.
```

```

    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
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```
Here is the call graph for this function:



Core.ConstructionMonitorMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Member Function Documentation

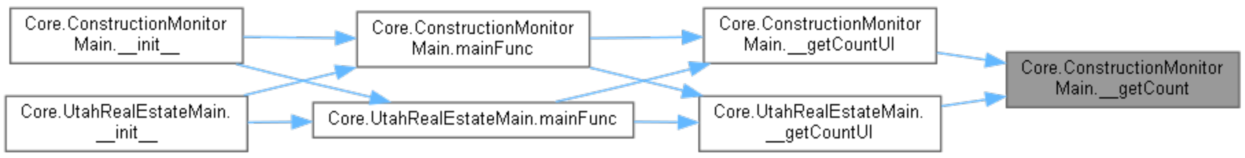
def Core.ConstructionMonitorMain.__getCount (*self*) [private]

```
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:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount

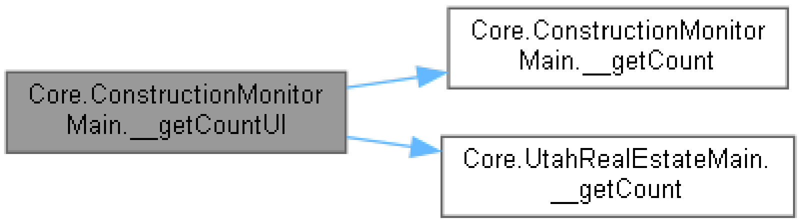
def Core.ConstructionMonitorMain.__getCountUI (*self*) `[private]`

```
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
```

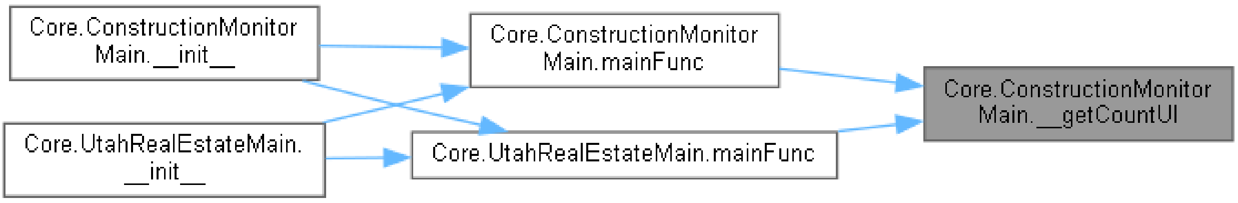
Definition at line 405 of file ConstructionMonitor/Core.py.

```
    def __getCountUI(self):

        """
    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
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

Here is the call graph for this function:


Core.ConstructionMonitor Main.__getCountUI
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI

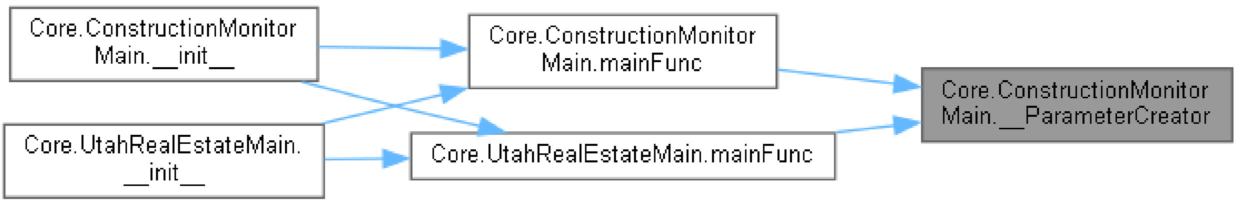
def Core.ConstructionMonitorMain.__ParameterCreator (*self*) `[private]`

```
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.

```
    def __ParameterCreator(self):
        """
    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
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

```

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain.__init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__ParameterCreator

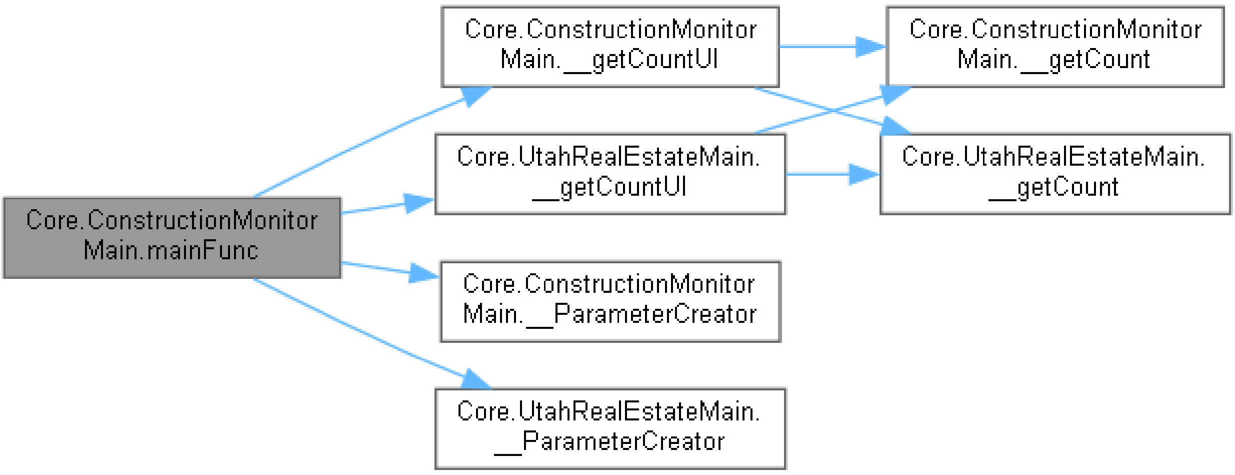
def Core.ConstructionMonitorMain.mainFunc (*self*)

```
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
```

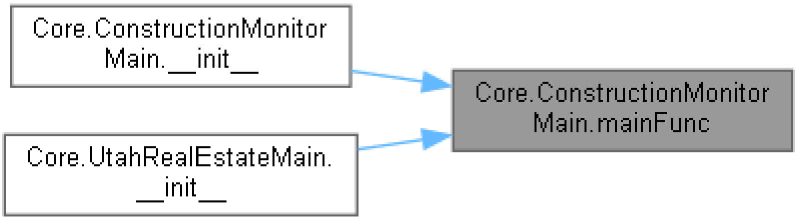
Definition at line 276 of file ConstructionMonitor/Core.py.

```
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,
ColumnSelection=self.__columnSelection,
                                                        BatchesNum=self.__batches,
                                                        Type="construction_monitor")
                  BatchGuiObject.BatchGuiShow()
                  self.dataframe = BatchGuiObject.dataframe
                  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()))}")
                  FileSaver("cm", self.dataframe, self.__appendFile)
              else:
                  RESTError(994)
                  raise SystemExit(994)
```

Here is the call graph for this function:


Core.ConstructionMonitorMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCountUI
Core.UtahRealEstateMain.__getCount
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator

Here is the caller graph for this function:


Core.ConstructionMonitorMain.__init__
Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc

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 4 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 6 of file DataTransfer.py.

```
    def __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
    """
        self.__value = 0

```

Member Function Documentation

def DataTransfer.DataTransfer.getValue (*self*)

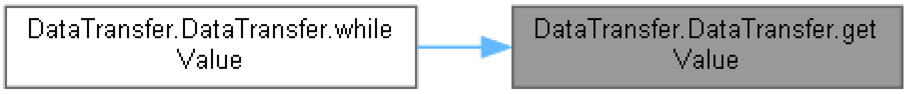
```
The getValue function returns the value of the private variable __value.
This is a getter function that allows access to this private variable.

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 39 of file DataTransfer.py.

```
    def 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
    """
        return self.__value

```

Here is the caller graph for this function:


DataTransfer.DataTransfer.while Value
DataTransfer.DataTransfer.get Value

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 22 of file DataTransfer.py.

```
    def 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
    """
        self.__value = value

```

def DataTransfer.DataTransfer.whileValue (*self*)

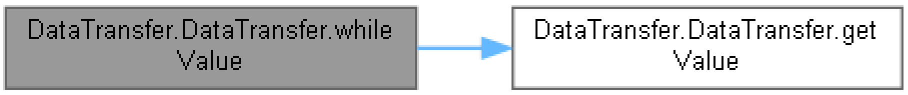
```
The whileValue function is a function that will run the getValue function until it is
told to stop.
This allows for the program to constantly be checking for new values from the sensor.
```

```
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 55 of file DataTransfer.py.

```
00055     def whileValue(self):
00056         """
00057     The whileValue function is a function that will run the getValue function
until it is told to stop.
00058     This allows for the program to constantly be checking for new values from
the sensor.
00059
00060     Args:
00061         self: Refer to the current instance of the class
00062
00063     Returns:
00064         The value of the input
00065
00066     Doc Author:
00067         Willem van der Schans, Trelent AI
00068     """
00069         while True:
00070             self.getValue()
```

Here is the call graph for this function:


DataTransfer.DataTransfer.while
Value
DataTransfer.DataTransfer.get
Value

Member Data Documentation

DataTransfer.DataTransfer.__value [private]

Definition at line 20 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 13 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 15 of file FileSaver.py.

```
00015     def __init__(self, method, outputDF, AppendingPath=None):
00016         """
00017     The __init__ function is called when the class is instantiated.
00018     It sets up the instance of the class, and defines all variables that will
be used by other functions in this class.
00019     The __init__ function takes two arguments: self and method.  The first
argument, self, refers to an instance of a
00020     class (in this case it's an instance of DataFrameSaver). The second argument,
method refers to a string value that
00021     is passed into DataFrameSaver when it's instantiated.
00022
00023     Args:
```

```python
        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
    """
        self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            datetime.datetime.today().strftime('%m%d%Y'))
        self.data = outputDF
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                                              keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                             windowType="noticeLarge")
```

```
00087
00088                   # Logging
00089                   print(
00090                       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")
00091                   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]}")
00092               else:
00093                   PopupWrapped(text=f"File Saved to
{self.docPath.joinpath(self.fileName)}", windowType="noticeLarge")
00094
00095                   # Logging
00096                   print(
00097                       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")
00098           else:
00099               pass
00100
```

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 101 of file FileSaver.py.

```
00101     def getPath(self):
00102         """
00103     The getPath function returns the path to the file.
00104         It is a string, and it joins the docPath with the fileName.
00105
00106     Args:
00107         self: Represent the instance of the class
00108
00109     Returns:
00110         The path to the file
00111
00112     Doc Author:
00113         Willem van der Schans, Trelent AI
00114     """
00115         return str(self.docPath.joinpath(self.fileName))
```

Member Data Documentation

FileSaver.FileSaver.appendFlag

Definition at line 39 of file FileSaver.py.

FileSaver.FileSaver.data

Definition at line 37 of file FileSaver.py.

FileSaver.FileSaver.dataAppending

Definition at line 38 of file FileSaver.py.

FileSaver.FileSaver.docPath

Definition at line 35 of file FileSaver.py.

FileSaver.FileSaver.fileName

Definition at line 40 of file FileSaver.py.

FileSaver.FileSaver.outputFrame

Definition at line 70 of file FileSaver.py.

FileSaver.FileSaver.primaryKey

Definition at line 44 of file FileSaver.py.

FileSaver.FileSaver.uiFlag

Definition at line 41 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 21 of file Initializer.py.

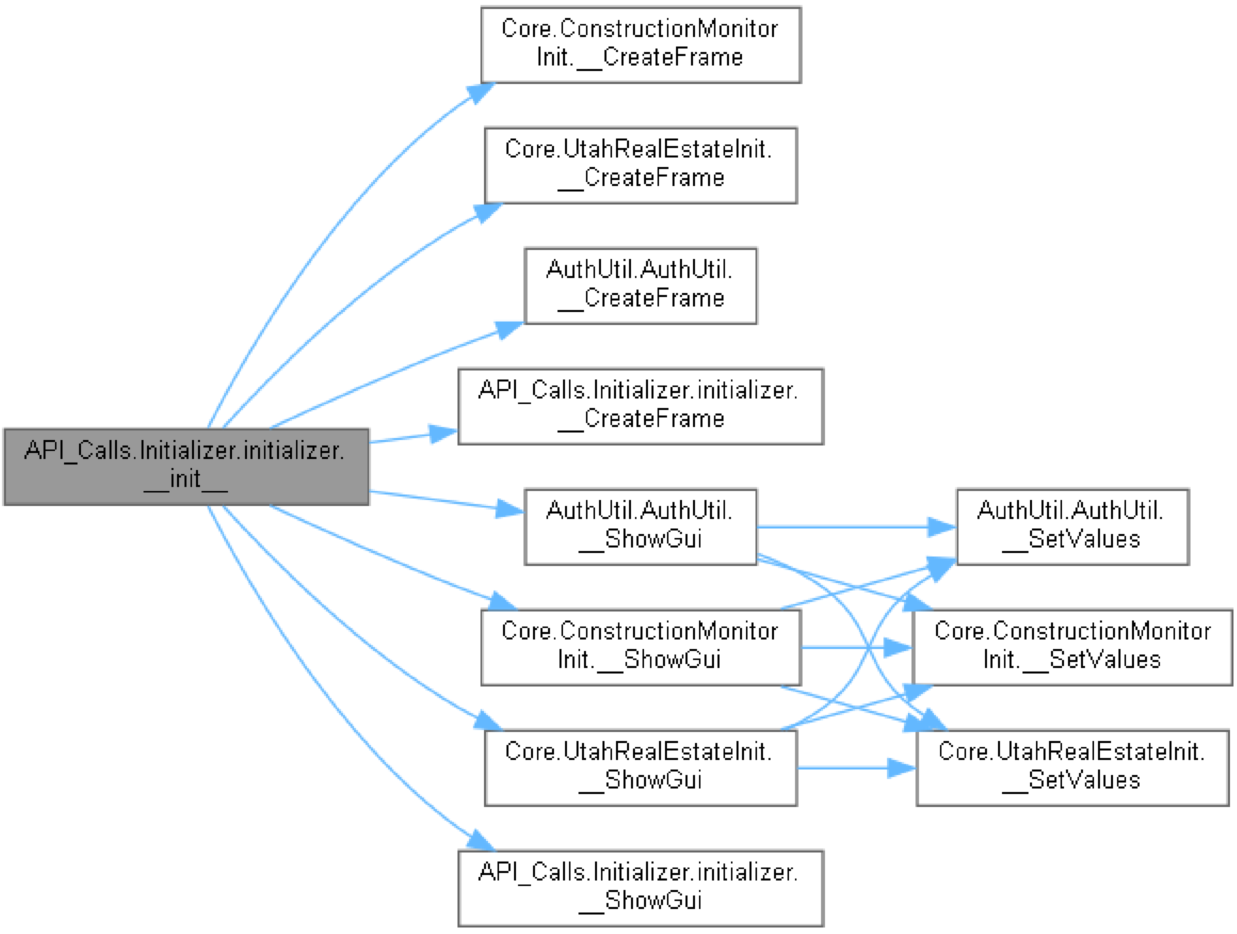
Constructor & Destructor Documentation

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 23 of file Initializer.py.

```
    def __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
    """
        self.classObj = None

        logger()

        print("\n\n------------Initiate Program--------------------\n\n")

        self.__ShowGui(self.__CreateFrame(), "Data Tool")

        print("\n\n-------------Closing Program---------------------\n\n")

```
Here is the call graph for this function:


API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__CreateFrame
Core.UtahRealEstateInit.__CreateFrame
AuthUtil.AuthUtil.__CreateFrame
API_Calls.Initializer.initializer.__CreateFrame
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
API_Calls.Initializer.initializer.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Member Function Documentation

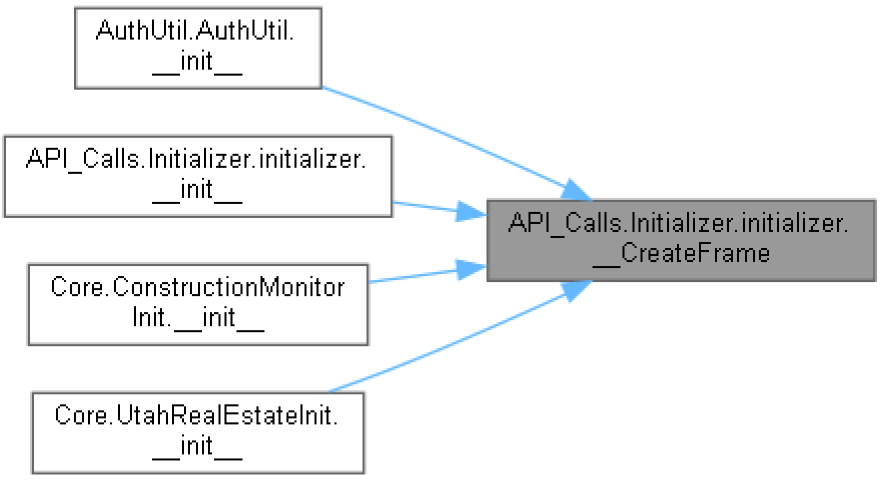
def API_Calls.Initializer.initializer.__CreateFrame (*self*) `[private]`

```
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 121 of file Initializer.py.

```
    def __CreateFrame(self):

        """
    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
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12,
"bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20,
None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)),
sg.Push(), sg.Button("Census", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20,
None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]

        return layout
```

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
API_Calls.Initializer.initializer.__CreateFrame

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

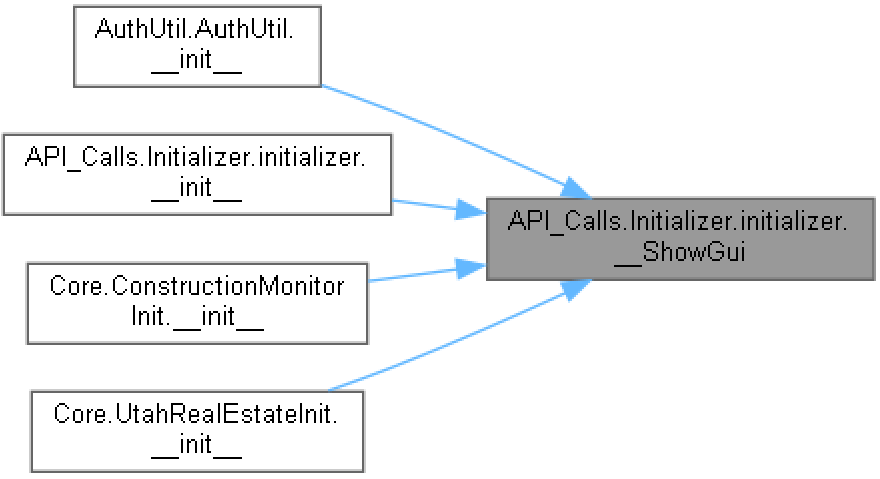
```
The __ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing
a tuple with three elements:
1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;,
etc.)
2) A dictionary containing 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 50 of file Initializer.py.

```
00050     def __ShowGui(self, layout, text):
00051 
00052         """
00053     The __ShowGui function is the main function that displays the GUI.
00054     It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three elements:
00055         1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;, etc.)
00056         2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
00057         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
00058 
00059     Args:
00060         self: Represent the instance of the class
00061         layout: Pass the layout of the window to be created
00062         text: Set the title of the window
00063 
00064     Returns:
00065         A window object
00066 
00067     Doc Author:
00068         Willem van der Schans, Trelent AI
00069     """
00070         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00071                            finalize=True,
00072                            icon=ImageLoader("taskbar_icon.ico"))
00073 
00074         while True:
00075             event, values = window.read()
00076 
00077             if event == "Construction Monitor":
00078                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API Call-----------------")
00079                 ConstructionMonitorMain(ConstructionMonitorInit())
00080                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API Call---------------------\n")
00081             elif event == "Utah Real Estate":
00082                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API Call-----------------")
00083                 UtahRealEstateMain(UtahRealEstateInit())
00084                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API Call---------------------\n")
00085             elif event == "Realtor.Com":
00086                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
```

```
                realtorCom()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API Call---------------------\n")
            elif event == "Census":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Census API Call-----------------")
                Cencus()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Census API Call---------------------\n")
            elif event == "Authorization Utility":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
                try:
                    os.system(f"start {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents folder",
                            windowType="errorLarge")

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    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.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

Here is the caller graph for this function:



AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
API_Calls.Initializer.initializer.__ShowGui

Member Data Documentation

API_Calls.Initializer.initializer.classObj

Definition at line 40 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 13 of file PopupWrapped.py.

Constructor & Destructor Documentation

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

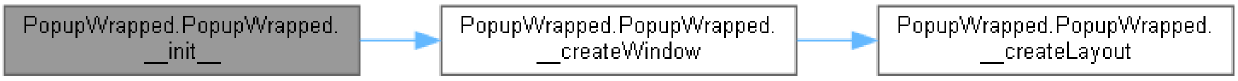
```
The __init__ function is the first function that gets called when an object of this
class is created.
It sets up all the variables and creates a window for us to use.
Args:
self: Represent the instance of the class
text: Set the text of the window
windowType: Determine what type of window to create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

Definition at line 15 of file PopupWrapped.py.

```
00015     def __init__(self, text="", windowType="notice", error=None):
00016         """
00017     The __init__ function is the first function that gets called when an object
of this class is created.
00018     It sets up all the variables and creates a window for us to use.
00019     Args:
00020         self: Represent the instance of the class
00021         text: Set the text of the window
00022         windowType: Determine what type of window to create
00023         error: Display the error message in the window
00024     Returns:
00025         Nothing
00026     Doc Author:
00027         Willem van der Schans, Trelent AI
```

```
    """
        self.__text = text
        self.__type = windowType
        self.__error = error
        self.__layout = []
        self.__windowObj = None
        self.__thread = None
        self.__counter = 0

        self.__createWindow()

```

Here is the call graph for this function:


PopupWrapped.PopupWrapped.
__init__
PopupWrapped.PopupWrapped.
__createWindow
PopupWrapped.PopupWrapped.
__createLayout

Member Function Documentation

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

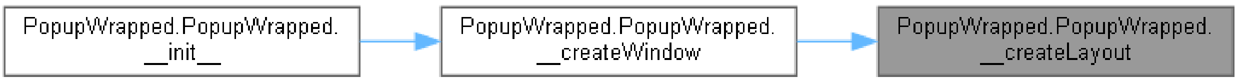
```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates
a layout based on that information.
Args:
self: Refer to the current instance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

Definition at line 39 of file PopupWrapped.py.

```
    def __createLayout(self):
        """
    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
    """
        sg.theme('Default1')
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
                       sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
                       sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
                       sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
```

```
00069                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00070             elif self.__type == "FatalErrorLarge":
00071                 __Line1 = [sg.Push(),
00072                            sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00073                            sg.Text(self.__text, justification="left",
key="-textField-"), sg.Push()]
00074                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00075             elif self.__type == "error":
00076                 __Line1 = [sg.Push(),
00077                            sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00078                            sg.Text(f"{self.__text}: {self.__error}",
justification="center", key="-textField-"),
00079                            sg.Push()]
00080                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00081             elif self.__type == "progress":
00082                 __Line1 = [sg.Push(),
00083                            sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00084
00085             if self.__type == "progress":
00086                 self.__layout = [__Line1, ]
00087             else:
00088                 self.__layout = [__Line1, __Line2]
00089
```

Here is the caller graph for this function:

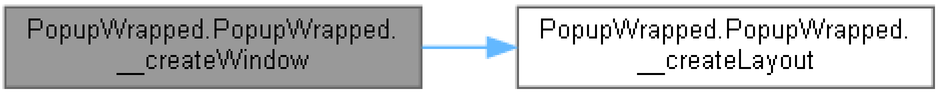

PopupWrapped.PopupWrapped.
__init__
PopupWrapped.PopupWrapped.
__createWindow
PopupWrapped.PopupWrapped.
__createLayout

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

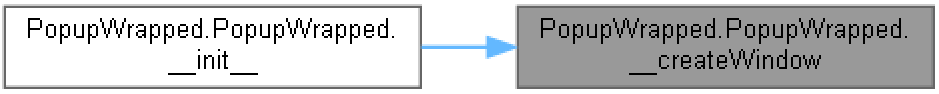
```
The __createWindow function is used to create the window object that will be displayed.
The function takes class variables and a window object. The function first calls
__createLayout, which creates the layout for the window based on what type of message
it is (error, notice, progress). Then it uses 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
&quot;Ok&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 90 of file PopupWrapped.py.

```
00090     def __createWindow(self):
00091         """
00092     The __createWindow function is used to create the window object that will
be displayed.
00093     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
&quot;Ok&quot; or closing the window). Once one of these events occurs
00094     Args:
00095         self: Reference the instance of the class
00096     Returns:
00097         A window object
00098     Doc Author:
00099         Willem van der Schans, Trelent AI
00100     """
00101         self.__createLayout()
00102
00103         if self.__type == "progress":
00104             self.__windowObj = sg.Window(title=self.__type,
layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 50))
        elif self.__type == "noticeLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "errorLarge":
            self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "FatalErrorLarge":
            self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        else:
            self.__windowObj = sg.Window(title=self.__type, layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                if event == "Ok" or event == sg.WIN_CLOSED:
                    break

                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}"
)
                except Exception as e:
                    print(
                        f"PopupWrapped.py | Error = {e} | Log Folder not found please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

```
Here is the call graph for this function:


PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow

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 155 of file PopupWrapped.py.

```python
    def 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
    """
        self.__windowObj.close()

```

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 168 of file PopupWrapped.py.

```python
    def 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
    """
        self.__counter += 1
        if self.__counter == 4:
            self.__counter = 1
        newString = ""
        if self.__type == "notice":
            pass
        elif self.__type == "error":
            pass
        elif self.__type == "progress":
            newString = f"{self.__text}{'.' * self.__counter}"
        self.__windowObj.write_event_value('update-textField-', newString)

        time.sleep(sleep)

```

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 195 of file PopupWrapped.py.

```
    def 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
    """
        event, values = self.__windowObj.read()

        if event.startswith('update'):
            __key_to_update = event[len('update'):]
            self.__windowObj[__key_to_update].update(values[event])
            self.__windowObj.refresh()
```

Member Data Documentation

PopupWrapped.PopupWrapped.__counter [private]

Definition at line 35 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__error [private]

Definition at line 31 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__layout [private]

Definition at line 32 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__text [private]

Definition at line 29 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__thread [private]

Definition at line 34 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__type [private]

Definition at line 30 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__windowObj [private]

Definition at line 33 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

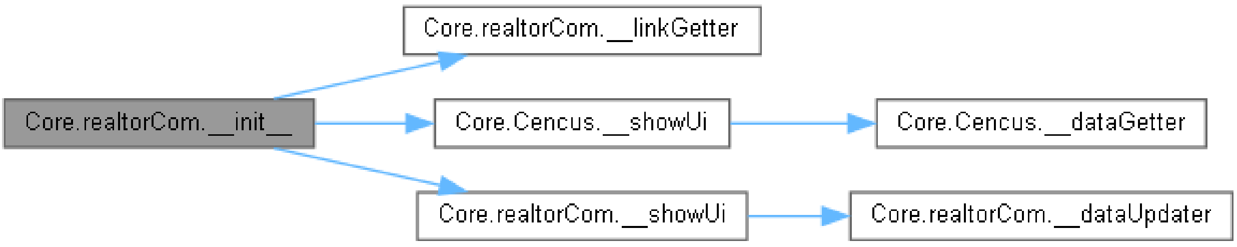
def Core.realtorCom.__init__ (*self*)

```
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
```

```
    """
        self.__page_html = None
        self.__update_date = None
        self.__last_date = None
        self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
        self.__linkDict = {}
        self.dfState = None
        self.dfCounty = None
        self.dfZip = None
        self.uiString = "Files Saved to \n"

        page_html =
requests.get("https://www.realtor.com/research/data/").text
        self.__page_html = BeautifulSoup(page_html, "html.parser")

        self.__linkGetter()
        self.__showUi()

        PopupWrapped(text=self.uiString, windowType="noticeLarge")

```
Here is the call graph for this function:



Member Function Documentation

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

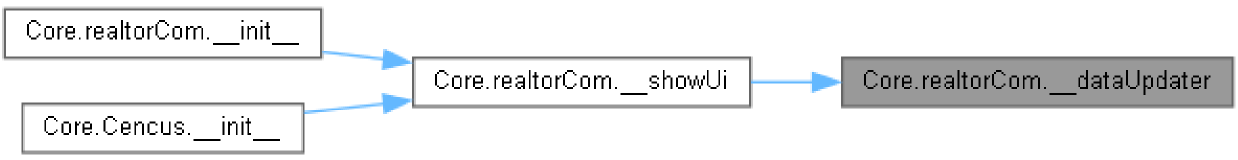
```
The __dataUpdater function is a private function that updates the dataframes for each
of the three
types of realtor data. It takes class variables and return the path to the saved file.
The function first creates an empty
dictionary called tempdf, then iterates through each key in self.__idDict (which
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.

```
    def __dataUpdater(self):

        """
    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
```

```
00108         DataFrame object. Then, depending on which type of realtor data we are dealing with (State/County/Zip), we save
00109
00110     Args:
00111         self: Access the attributes and methods of the class
00112
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]

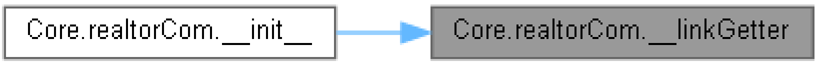
```
The __linkGetter function is a private function that takes the idDict dictionary and adds
a link to each entry in the dictionary. The link is used to access historical data for each
scope symbol.

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
```

Here is the caller graph for this function:



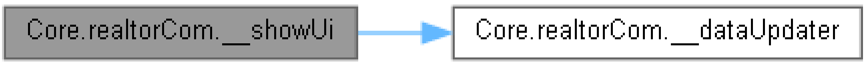
def Core.realtorCom.__showUi (*self*) [private]

```
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
```

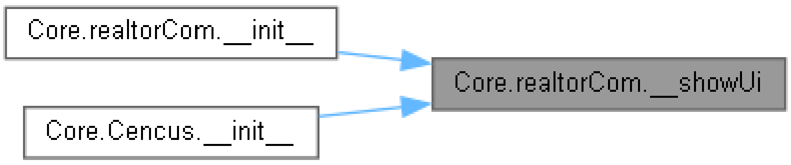
Definition at line 46 of file Realtor/Core.py.

```
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:



Here is the caller 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

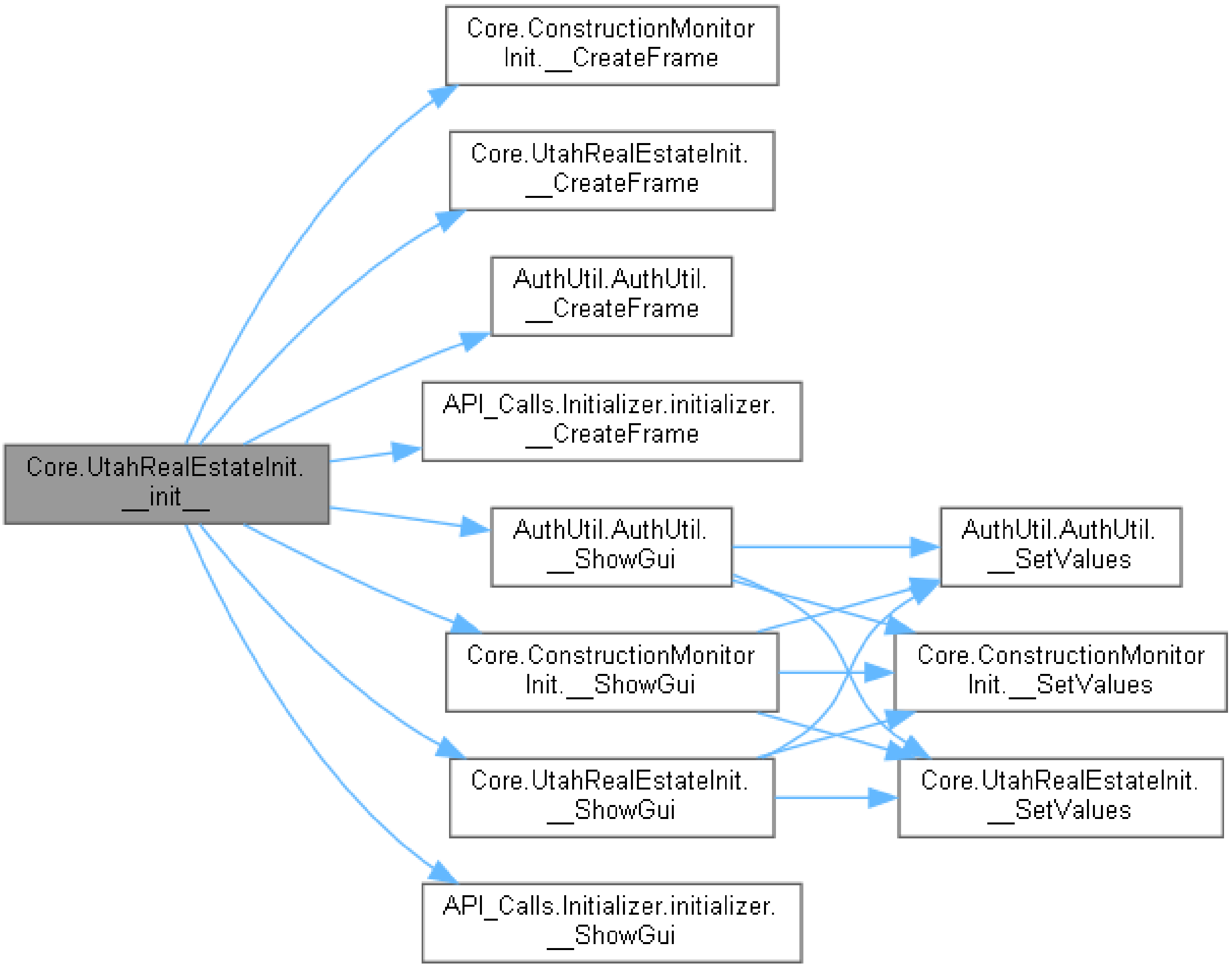
def Core.UtahRealEstateInit.__init__ (*self*)

```
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.

```
    def __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
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

```

Here is the call graph for this function:



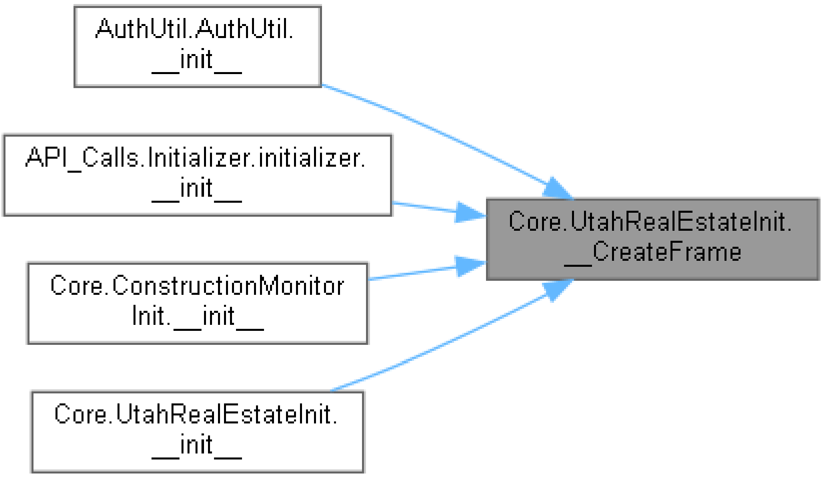
Member Function Documentation

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)),
                sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
                                target="-AppendingFile-")]

        line11 = [sg.HSeparator()]

        line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10, line11,
                  line12]

        return layout

```
Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
Core.UtahRealEstateInit.__CreateFrame

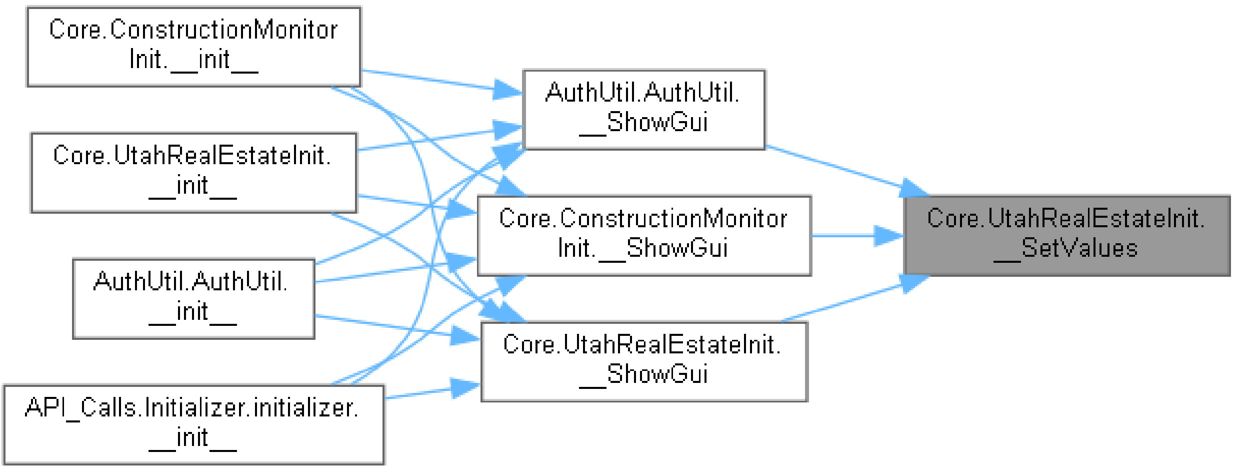
def Core.UtahRealEstateInit.__SetValues (*self*, *values*) `[private]`

```
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:
        A dictionary with the following keys:

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        if values['-selectionFlag-']:
            self.select = "ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,SubdivisionName," \
"StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDate," \
"ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,LotSizeAcres," \
"LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,ParkingTotal," \
"BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffective," \
"OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseContractDate," \
"AssociationFee,AssociationFeeFrequency,OccupantType,PropertySubType,PropertyType," \
                          "StandardStatus,BuyerFinancing"
        else:
            self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None


```

Here is the caller graph for this function:


Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
Core.UtahRealEstateInit.__SetValues

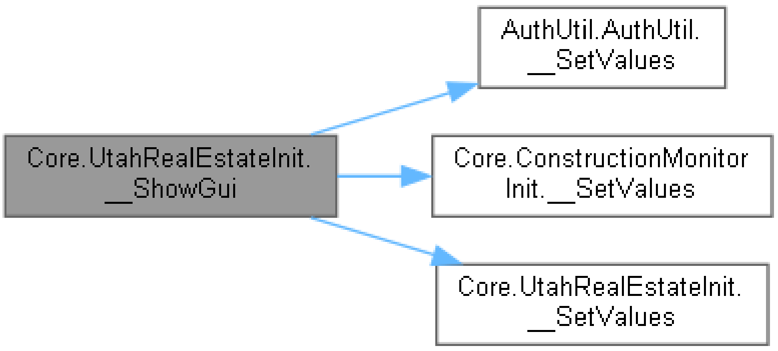
def Core.UtahRealEstateInit.__ShowGui (*self*, *layout*, *text*) `[private]`

```
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
```

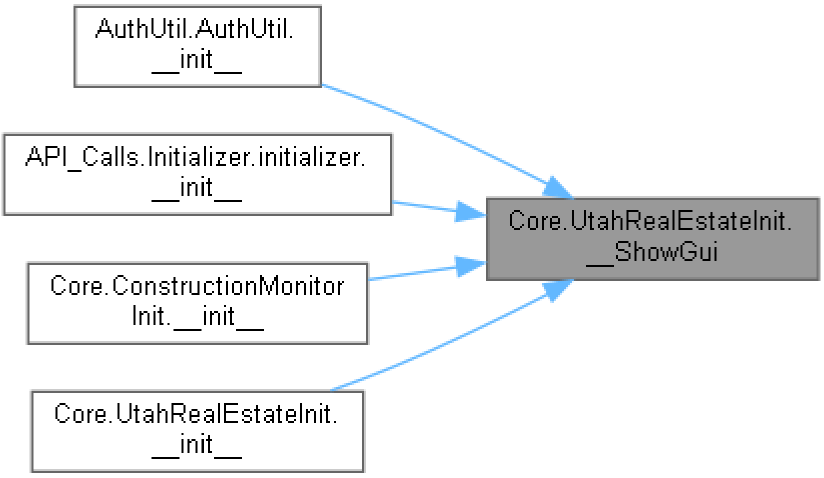
Definition at line 52 of file UtahRealEstate/Core.py.

```
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:



Here is the caller 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*)

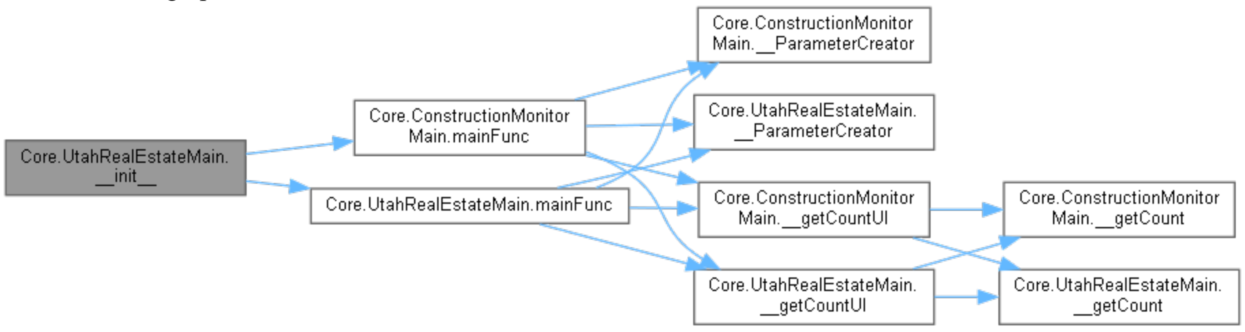
```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to work properly.

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.
```

```
    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
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = 'https://resoapi.utahrealestate.com/reso/odata/Property?'
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
        except AttributeError as e:
            if e is not None:
                print(
                    f"UtahRealEstate/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```

Here is the call graph for this function:


Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Member Function Documentation

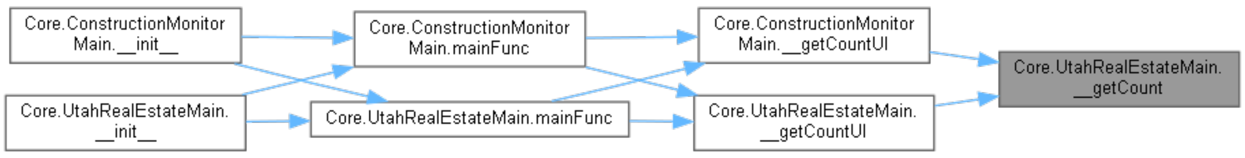
def Core.UtahRealEstateMain.__getCount (*self*) [private]

```
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:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.UtahRealEstateMain. __getCount

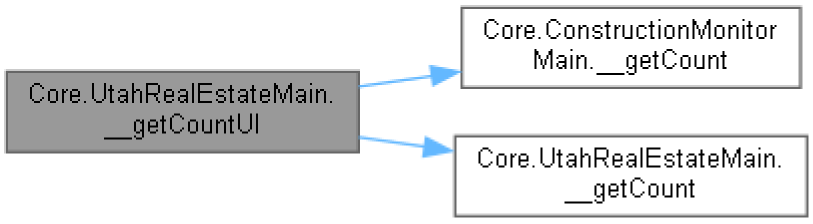
def Core.UtahRealEstateMain.__getCountUI (*self*) `[private]`

```
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
```

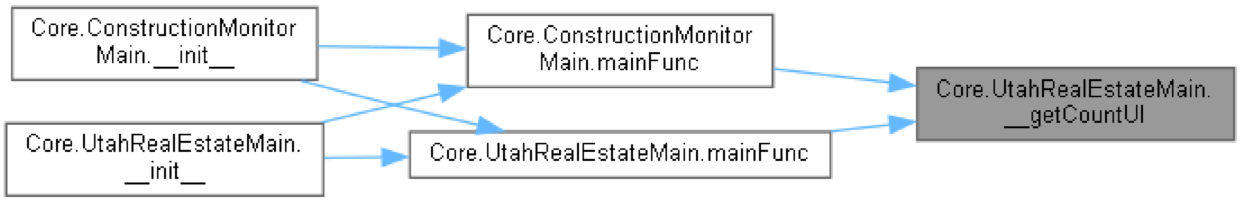
Definition at line 415 of file UtahRealEstate/Core.py.

```
    def __getCountUI(self):

        """
    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
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```

Here is the call graph for this function:


Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.UtahRealEstateMain. __getCountUI

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

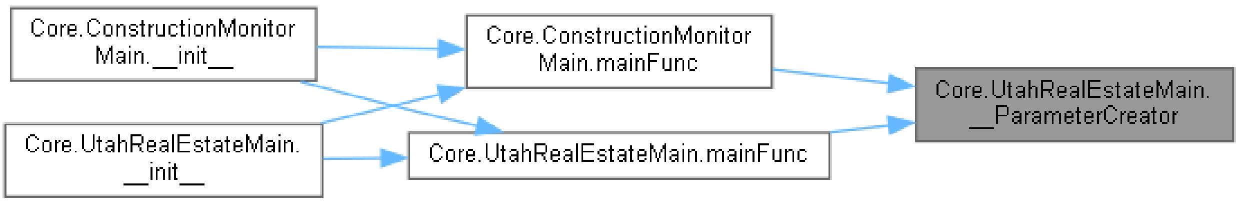
```
The __ParameterCreator function is used to create the filter string for the ReST API
call.
The function takes in a siteClass object and extracts all of its parameters into a
dictionary.
It then creates an appropriate filter string based on those parameters.

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:


Core.ConstructionMonitorMain.__init__
Core.UtahRealEstateMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.UtahRealEstateMain.__ParameterCreator

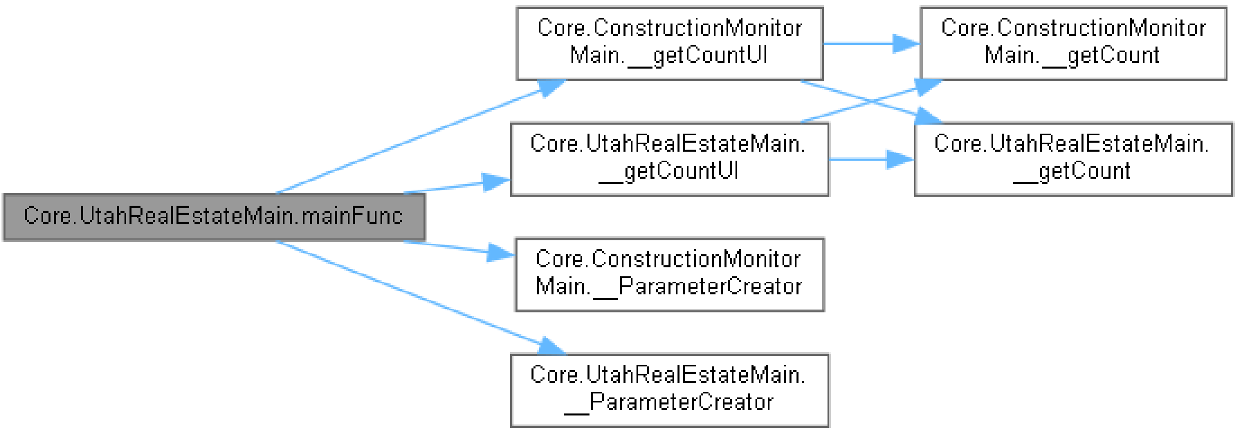
def Core.UtahRealEstateMain.mainFunc (*self*)

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

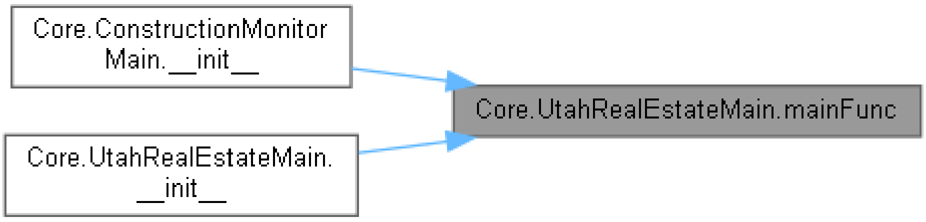
```
00267     def mainFunc(self):
00268 
00269         """
00270     The mainFunc function is the main function of this module. It will be called
by the GUI when a user clicks on
00271     the &quot;Run&quot; button in the GUI. The mainFunc function should contain
all of your code for running your program, 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()
```

```

        self.__ParameterCreator()

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, None)

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            BatchInputGui(self.__batches)
            print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
            BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                              ParameterDict=self.__parameterString,
                                              HeaderDict=self.__headerDict,
                                              BatchesNum=self.__batches,
                                              Type="utah_real_estate")
            BatchGuiObject.BatchGuiShow()
            self.dataframe = BatchGuiObject.dataframe
            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()))}")
            FileSaver("ure", self.dataframe, self.__appendFile)
        else:
            RESTError(994)
            raise SystemExit(994)

```
Here is the call graph for this function:



Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCountUI
Core.UtahRealEstateMain.__getCount
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator

Here is the caller graph for this function:



Core.ConstructionMonitorMain.__init__
Core.UtahRealEstateMain.__init__
Core.UtahRealEstateMain.mainFunc

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

File Documentation

__init__.py

main.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


from Initializer import initializer

initializer()
```

AuthUtil.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import ctypes
import datetime
import json
import os
from pathlib import Path

import PySimpleGUI as sg
from cryptography.fernet import Fernet

from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class AuthUtil:

    def __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
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.file_name = None
        self.append_file = None
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
        self.k = None
        self.keyFlag = True
        self.jsonDict = {}
        self.passFlagUre = False
        self.passFlagCm = False
        self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
                           "Submitting will overwrite any old values in an unrecoverable manner."

        if os.path.exists(self.filePath):
            pass
        else:
            if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
                os.mkdir(self.filePath)
            else:
                os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
                os.mkdir(self.filePath)

        if os.path.exists(self.keyPath):
            pass
        else:
            if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
                os.mkdir(self.keyPath)
            else:
```

```python
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    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...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:
            ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                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...")
            pass

    def __SetValues(self, values):

        """
    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
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    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")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    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")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
            self.passFlagCm = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction Monitor",
                         windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate", windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Construction Monitor", windowType="errorLarge")
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

    def __ShowGui(self, layout, text):

        """
    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
    """
        window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
                            finalize=True,
                            icon=ImageLoader("taskbar_icon.ico"))

        while not self.passFlagUre or not self.passFlagCm:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                except Exception as e:
                    print(e)
                    RESTError(993)
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

    def __CreateFrame(self):
        """
    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
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                  sg.Push(),
                  sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
justification="center"),
                  sg.Push(),
                  sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Push(),
                  sg.Text("Utah Real Estate Key: ", justification="center"),
                  sg.Push()]

        line3 = [sg.Push(),
                  sg.Input(default_text="", key="-ureAuth-", disabled=False,
                           size=(40, 1)),
                  sg.Push()]

        line4 = [sg.HSeparator()]

        line5 = [sg.Push(),
                  sg.Text("Construction Monitor Key: ", justification="center"),
                  sg.Push()]

        line6 = [sg.Push(),
                  sg.Input(default_text="", key="-cmAuth-", disabled=False,
                           size=(40, 1)),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text(self.outcomeText, justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout
```

BatchProcessing.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import math
from datetime import date

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList


def 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
"""
    try:
        document_limit = Argument_Dict["size"]
    except Exception as e:
        # Logging
        print(
            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")
        document_limit = 200

    return int(math.ceil(float(TotalRecords) / float(document_limit)))


class BatchProcessorConstructionMonitor:

    def __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
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.valueObject = valueObject
        self.__maxRequests = 10000
        self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])))
        self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
        self.__dateTracker = None

    def 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
    """
        self.ConstructionMonitorProcessor(self.valueObject)

    def 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
    """
        __df = None
        for callNum in range(0, self.__requestCount):
            self.__parameterDict["from"] = 0

            if self.__requestCount > 1 and callNum != self.__requestCount - 1:
                __batchNum = self.__requestCalls
                if __df is None:
                    self.__dateTracker = str(date.today())
                else:
                    self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
            elif self.__requestCount == 1:
                __batchNum = self.__numBatches
                self.__dateTracker = str(date.today())
            else:
                __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])) - (
                        self.__requestCount - 1)
                self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')

            self.__parameterDict['dateEnd'] = self.__dateTracker

            for record in range(0, int(math.ceil(__batchNum))):
                if record != 0:
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        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}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)


class BatchProcessorUtahRealEstate:

    def __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
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterString = ParameterString
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.valueObject = valueObject

    def 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
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

    def 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
    """
        __df = pd.DataFrame()

        for batch in range(self.__numBatches):

            if batch == 0:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
                                        headers=self.__headerDict)

                response_temp = response.json()
                __df = pd.json_normalize(response_temp, record_path=['value'])

            else:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch * 200}",
                                        headers=self.__headerDict)

                response_temp = response.json()
                response_temp = pd.json_normalize(response_temp, record_path=['value'])
                __df = pd.concat([__df, response_temp], ignore_index=True)

            valueObject.setValue(valueObject.getValue() + 1)

        self.dataframe = __df
        valueObject.setValue(-999)
```

DataChecker.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import os
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.Gui.ImageLoader import ImageLoader


def 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
"""
    __text1 = f"Select existing {Name} csv data file:"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Text("Choose a file: "),
               sg.Input(),
               sg.FileBrowse(file_types=(("Data Files (.csv)", "*.csv"),), initial_folder=DataPath)]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=False,
                       icon=ImageLoader("taskbar_icon.ico"))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":

            break

    window.close()


DataChecker("Construction Monitor", Path(os.path.expanduser('~/Documents')))
```

DataSupportFunctions.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


def 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
"""
    listOut = list(string.split(","))
    return listOut
```

FileSaver.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
from pathlib import Path

import pandas as pd

from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class FileSaver:

    def __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
    """
        self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            datetime.datetime.today().strftime('%m%d%Y'))
        self.data = outputDF
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                                            keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                             windowType="noticeLarge")

                # Logging
                print(
                    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")
                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]}")
            else:
                PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}", windowType="noticeLarge")

                # Logging
                print(
                    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")
        else:
            pass

    def 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
    """
        return str(self.docPath.joinpath(self.fileName))
```

ErrorPopup.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


def 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
"""
    PopupWrapped(
        f"ERROR @ {textString} \n"
        f"Log folder will be opened upon program exit",
        windowType="FatalErrorLarge")
```

ErrorPrint.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime


def 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
"""
    if isinstance(response, int):
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response: {response}")
    else:
        response_txt = response.text
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response: {response_txt}")
```

Logger.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
import sys
from pathlib import Path


def 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
"""
    dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
    if os.path.exists(dir_path):
        pass
    else:
        if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
            os.mkdir(dir_path)
        else:
            os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
            os.mkdir(dir_path)

    filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
        f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
    sys.stdout = open(filePath, 'w')
    sys.stderr = sys.stdin = sys.stdout

    def sorted_ls(path):
        """
    The sorted_ls function takes a path as an argument and returns the files in that directory sorted by modification time.

    Args:
        path: Specify the directory to be sorted

    Returns:
        A list of files in a directory sorted by modification time

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
        return list(sorted(os.listdir(path), key=mtime))

    del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
    for file in del_list:
        os.remove(dir_path.joinpath(file))
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Log file {file} deleted")
```

PrintFunc.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


```

RESTError.py

```python
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime

from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
from API_Calls.Functions.ErrorFunc.ErrorPrint import RESTErrorPrint


def 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
"""
    if isinstance(response, int):
        status_code = response
    else:
        status_code = response.status_code

    if status_code == 200:
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Api Request completed successfully"
        print(textString)
        pass
    elif status_code == 301:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 400:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Bad Request; check input arguments"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 401:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Authentication Error: No keys found"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 402:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 403:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 404:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise NameError(textString)
    elif status_code == 405:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 408:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout by server"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 503:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 701:
        RESTErrorPrint(response)
        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)"
        ErrorPopup(textString)
        raise TypeError(textString)
    elif status_code == 801:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 790:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout within requests"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 791:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 990:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No password input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 991:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No username input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 992:
        RESTErrorPrint(response)
        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)"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 993:
        RESTErrorPrint(response)
        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"
        ErrorPopup(textString)
        print(ValueError(textString))
```

```python
    elif status_code == 994:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: server returned no documents"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 1000:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Catastrophic Error"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 1001:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Main Function Error Break"
        raise SystemError(textString)
    elif status_code == 1100:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User has cancelled the program execution"
        raise KeyboardInterrupt(textString)
    elif status_code == 1101:
        RESTErrorPrint(response)
        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"
        print(textString)
    else:
        RESTErrorPrint(response)
        raise Exception(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | An unknown exception occurred")
```

BatchGui.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/

#
#  2023 - Permission to utilize, edit, copy and maintain and source code related to this project within the project's scope is granted to the Kem C. Gardner Policy institute situated in Salt Lake City, Utah into perpetuity.
#
#  THE CONTENTS OF THIS PROJECT ARE PROPRIETARY AND CONFIDENTIAL.
#  UNAUTHORIZED COPYING, TRANSFERRING OR REPRODUCTION OF THE CONTENTS OF THIS PROJECT, VIA ANY MEDIUM IS STRICTLY PROHIBITED.
#
#  The receipt or possession of the source code and/or any parts thereof does not convey or imply any right to use them
#  for any purpose other than the purpose for which they were provided to you.
#
#
import PySimpleGUI as sg

from API_Calls.Functions.Gui.ImageLoader import ImageLoader


def 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
"""
    __text1 = f"This request will run {batches} batches"
    __text2 = "Do you want to continue?"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Push(),
               sg.Text(__text2, justification="center"),
               sg.Push()]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=False,
                       icon=ImageLoader("taskbar_icon.ico"),
                       size=(290, 100))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":

            break

    window.close()
```

BatchProgressGUI.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/

import datetime
import threading
import time

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.BatchProcessing import BatchProcessorConstructionMonitor, BatchProcessorUtahRealEstate
from API_Calls.Functions.Gui.DataTransfer import DataTransfer
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped

counter = 1


class BatchProgressGUI:

    def __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
    """
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.__type = Type
        self.dataframe = None

        self.__layout = None
        self.__batches = BatchesNum
        self.__window = None
        self.__batch_counter = 0

    def 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
    """
        self.CreateProgressLayout()
        self.createGui(self.__type)

    def 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
    """
        sg.theme('Default1')

        __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--progress_text--"),
                   sg.Push()]

        __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
                   sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]

        __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h', size=(30, 20),
                           key='--progress_bar--')]


        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

    def 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
    """
        self.__window = sg.Window('Progress', self.__layout, finalize=True, icon=ImageLoader("taskbar_icon.ico"))

        start_time = datetime.datetime.now().replace(microsecond=0)
        update_text = f"Batch {0} completed"
        self.__window['--progress_text--'].update(update_text)
        self.__window['--progress_bar--'].update(0)
        self.__window['--time_est--'].update("Est time needed 00:00:00")

        valueObj = DataTransfer()
        valueObj.setValue(0)

        if Sourcetype == "construction_monitor":

            processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
NumBatches=self.__batches,
ParameterDict=self.__parameterDict,
HeaderDict=self.__headerDict,
ColumnSelection=self.__columnSelection,
valueObject=valueObj)
        elif Sourcetype == "utah_real_estate":
            processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
NumBatches=self.__batches,
ParameterString=self.__parameterDict,
HeaderDict=self.__headerDict,
                                                           valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

    def 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
    """
        if valueObj.getValue() != self.__batch_counter:
            self.__batch_counter = valueObj.getValue()

            __update_text = f"Batch {self.__batch_counter}/{self.__batches} completed"

            self.__window.write_event_value('update--progress_bar--', self.__batch_counter)
            self.__window.write_event_value('update--progress_text--', __update_text)
        else:
            pass

    def 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
    """
        while True:
            if self.__batch_counter < self.__batches:

                __current_time = datetime.datetime.now().replace(microsecond=0)

                __passed_time = __current_time - start_time

                __timer_string = f"Time Elapsed {__passed_time}"

                try:
                    self.__window.write_event_value('update--timer--', __timer_string)
                except AttributeError as e:
                    print(
                        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")
                    break

                __passed_time = __passed_time.total_seconds()

                try:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * (self.__batches / self.__batch_counter) - __passed_time)).seconds
                except:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * self.__batches - __passed_time)).seconds

```

```python
                __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))

                __end_string = f"Est time needed {__time_est}"
                self.__window.write_event_value('update--time_est--', __end_string)
            else:
                __end_string = f"Est time needed 00:00:00"
                self.__window.write_event_value('update--time_est--', __end_string)
            time.sleep(0.25)

    def 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
    """
        while True:
            time.sleep(0.3)
            if self.__batch_counter != ObjectVal.getValue():
                self.__batch_counter = ObjectVal.getValue()
                return True
            else:
                return False
```

DataTransfer.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


class DataTransfer:

    def __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
    """
        self.__value = 0

    def 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
    """
        self.__value = value

    def 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
    """
        return self.__value

    def 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
    """
        while True:
            self.getValue()
```

ImageLoader.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import base64
import os
from io import BytesIO
from os.path import join, normpath

from PIL import Image


def 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
"""
    try:
        __path = normpath(join(str(os.getcwd().split("API_Calls", 1)[0]), "API_Calls"))
        __path = normpath(join(__path, "Images"))
        __path = join(__path, file).replace("\\", "/")

        image = Image.open(__path)

        __buff = BytesIO()

        image.save(__buff, format="png")

        img_str = base64.b64encode(__buff.getvalue())

        return img_str
    except Exception as e:
        # We cannot log this error like other errors due to circular imports
        raise e
```

PopupWrapped.py

```
#    This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import os
import time
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.Gui.ImageLoader import ImageLoader


class PopupWrapped():

    def __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
    """
        self.__text = text
        self.__type = windowType
        self.__error = error
        self.__layout = []
        self.__windowObj = None
        self.__thread = None
        self.__counter = 0

        self.__createWindow()

    def __createLayout(self):
        """
    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
    """
        sg.theme('Default1')
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "FatalErrorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "error":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}: {self.__error}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "progress":
            __Line1 = [sg.Push(),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]

        if self.__type == "progress":
            self.__layout = [__Line1, ]
        else:
            self.__layout = [__Line1, __Line2]

    def __createWindow(self):
        """
    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 &quot;Ok&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
    """
        self.__createLayout()

        if self.__type == "progress":
            self.__windowObj = sg.Window(title=self.__type, layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 50))
        elif self.__type == "noticeLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "errorLarge":
            self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "FatalErrorLarge":
            self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
```

```python
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        else:
            self.__windowObj = sg.Window(title=self.__type, layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                if event == "Ok" or event == sg.WIN_CLOSED:
                    break

                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
                except Exception as e:
                    print(
                        f"PopupWrapped.py | Error = {e} | Log Folder not found please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

    def 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
    """
        self.__windowObj.close()

    def 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
    """
        self.__counter += 1
        if self.__counter == 4:
            self.__counter = 1
        newString = ""
        if self.__type == "notice":
            pass
        elif self.__type == "error":
            pass
        elif self.__type == "progress":
            newString = f"{self.__text}{'.' * self.__counter}"
        self.__windowObj.write_event_value('update-textField-', newString)

        time.sleep(sleep)

    def 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
    """
        event, values = self.__windowObj.read()

        if event.startswith('update'):
            __key_to_update = event[len('update'):]
            self.__windowObj[__key_to_update].update(values[event])
            self.__windowObj.refresh()
```

Initializer.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.ErrorFunc.Logger import logger
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
from API_Calls.Sources.CFBP.Core import Cencus
from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
    ConstructionMonitorMain
from API_Calls.Sources.Realtor.Core import realtorCom
from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain, UtahRealEstateInit


class initializer:

    def __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
    """
        self.classObj = None

        logger()

        print("\n\n-------------Initiate Program---------------------\n\n")

        self.__ShowGui(self.__CreateFrame(), "Data Tool")

        print("\n\n-------------Closing Program---------------------\n\n")

    def __ShowGui(self, layout, text):

        """
    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., &quot;Text&quot;, &quot;InputText&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
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Construction Monitor":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API Call-----------------")
                ConstructionMonitorMain(ConstructionMonitorInit())
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API Call---------------------\n")
            elif event == "Utah Real Estate":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API Call-----------------")
                UtahRealEstateMain(UtahRealEstateInit())
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API Call---------------------\n")
            elif event == "Realtor.Com":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
                realtorCom()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API Call---------------------\n")
            elif event == "Census":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Census API Call-----------------")
                Cencus()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Census API Call---------------------\n")
            elif event == "Authorization Utility":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
                try:
                    os.system(f"start {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents folder",
                            windowType="errorLarge")

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    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.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    def __CreateFrame(self):

        """
    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
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(), sg.Button("Census", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]

        return layout
```

CFBP/Core.py

```python
import threading
from datetime import date

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class Cencus:

    def __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
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
        self.uiString = None
        self.link = None

        self.__showUi()
        print(self.link)
        F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
        self.uiString = (
            f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg} data retrieved \n Data Saved at {F.getPath()}")

        PopupWrapped(text=self.uiString, windowType="noticeLarge")

    def __showUi(self):

        """
    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
    """
        uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataGetter,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __dataGetter(self):
        """
    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
    """
        arg_dict_bu = locals()

        link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}" + f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

ConstructionMonitor/Core.py

```python
import copy
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path
import datetime

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class ConstructionMonitorInit:

    def __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
    """
        self.size = None
        self.SourceInclude = None
        self.dateStart = None
        self.dateEnd = None
        self.rest_domain = None
        self.auth_key = None
        self.ui_flag = None
        self.append_file = None

        passFlag = False

        while not passFlag:
            if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                    "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                    "Security").joinpath("auth.json")):
                try:
                    f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                    key = f.readline()
                    f.close()
                    f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json"), "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
                    passFlag = True
                except Exception as e:
                    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")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")

    def __ShowGui(self, layout, text):

        """
    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
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    @staticmethod
    def __CreateFrame():

        """
    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
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

    def __SetValues(self, values):

        """
    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
    """
        self.size = 1000

        if values["-Cal-"] != "":
            self.dateStart = values["-Cal-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-EndCal-"] != "":
            self.dateEnd = values["-EndCal-"]
        else:
            self.dateEnd = date.today().strftime("%Y-%m-%d")

        self.rest_domain = "https://api.constructionmonitor.com/v2/powersearch/?"

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True


class ConstructionMonitorMain:

    def __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
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

    def 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
    """
        self.__ParameterCreator()

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)
        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            BatchInputGui(self.__batches)
            print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
            BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                              ParameterDict=self.__parameterDict,
                                              HeaderDict=self.__headerDict,
                                              ColumnSelection=self.__columnSelection,
                                              BatchesNum=self.__batches,
                                              Type="construction_monitor")
            BatchGuiObject.BatchGuiShow()
            self.dataframe = BatchGuiObject.dataframe
            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()))}")
            FileSaver("cm", self.dataframe, self.__appendFile)
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    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
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

    def __getCount(self):
        """
    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
    """
        __count_resp = None

        try:

            __temp_param_dict = copy.copy(self.__parameterDict)

            __count_resp = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=__temp_param_dict)

            if __count_resp.status_code != 200:
                RESTError(__count_resp)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        __count_resp = __count_resp.json()

        self.__record_val = __count_resp["hits"]["total"]["value"]

        del __count_resp, __temp_param_dict

    def __getCountUI(self):

        """
    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
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

Realtor/Core.py

```python
import threading

import pandas as pd
import requests
from bs4 import *

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class realtorCom:

    def __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
    """
        self.__page_html = None
        self.__update_date = None
        self.__last_date = None
        self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
        self.__linkDict = {}
        self.dfState = None
        self.dfCounty = None
        self.dfZip = None
        self.uiString = "Files Saved to \n"

        page_html = requests.get("https://www.realtor.com/research/data/").text
        self.__page_html = BeautifulSoup(page_html, "html.parser")

        self.__linkGetter()
        self.__showUi()

        PopupWrapped(text=self.uiString, windowType="noticeLarge")

    def __showUi(self):

        """
    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
    """
        uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataUpdater,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __linkGetter(self):

        """
    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
    """
        for key, value in self.__idDict.items():
            for row in self.__page_html.find_all("div", {"class": "monthly"}):
                try:
                    for nestedRow in row.find_all("a"):
                        if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
                            self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
                except Exception as e:
                    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")
                    RESTError(801)
                    raise SystemExit(801)

    def __dataUpdater(self):

        """
    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
    """
        for key, value in self.__idDict.items():
            tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)

            if key == "State":
                self.dfState = tempdf
            elif key == "County":
                self.dfCounty = tempdf
            elif key == "Zip":
                self.dfZip = tempdf

            FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
            self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

UtahRealEstate/Core.py

```python
import copy
import datetime
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class UtahRealEstateInit:

    def __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
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

    def __ShowGui(self, layout, text):

        """
    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
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(993)
            elif event == sg.WIN_CLOSED or event == "Quit":

                break

        window.close()

    @staticmethod
    def __CreateFrame():
        """
    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
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Utah Real Estate Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-", size=(31, 1))]

        line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date", "Close Date"],
                             key="-type-", size=(31, 1))]

        line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
                          disabled=False, size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-DateStart-")]

        line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today().strftime("%Y-%m-%d")), key="-DateEnd-", disabled=False,
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-', target="-DateEnd-")]

        line6 = [[sg.Text("Column Sub-Selection : ", size=(23, None), justification="Right"),
                  sg.Checkbox(text="", default=True, key="-selectionFlag-", size=(15, 1)),
                  sg.Push()]]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                  sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                           size=(20, 1)),
                  sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                                target="-AppendingFile-")]

        line11 = [sg.HSeparator()]

        line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10, line11,
                  line12]

        return layout

    def __SetValues(self, values):

        """
    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
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        if values['-selectionFlag-']:
            self.select = "ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,SubdivisionName," \
                          "StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDate," \
```

```python
                          "ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,LotSizeAcres," \
                          "LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,ParkingTotal," \
                          "BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffective," \
                          "OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseContractDate," \
                          "AssociationFee,AssociationFeeFrequency,OccupantType,PropertySubType,PropertyType," \
                          "StandardStatus,BuyerFinancing"
        else:
            self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None


class UtahRealEstateMain:

    def __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
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = 'https://resoapi.utahrealestate.com/reso/odata/Property?'
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
        except AttributeError as e:
            if e is not None:
                print(
                    f"UtahRealEstate/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

    def 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 &quot;Run&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
    """
        passFlag = False

        while not passFlag:
            if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
                try:
                    f = open(self.keyPath, "rb")
                    key = f.readline()
                    f.close()
                    f = open(self.filePath, "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
                    self.__headerDict = {authDict["ure"]["parameter"]: authkey}
                    passFlag = True
                except Exception as e:
                    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")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ParameterCreator()

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, None)

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            BatchInputGui(self.__batches)
            print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
            BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                              ParameterDict=self.__parameterString,
                                              HeaderDict=self.__headerDict,
                                              BatchesNum=self.__batches,
                                              Type="utah_real_estate")
            BatchGuiObject.BatchGuiShow()
            self.dataframe = BatchGuiObject.dataframe
            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()))}")
            FileSaver("ure", self.dataframe, self.__appendFile)
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    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
    """
        filter_string = ""

        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        if __Source_dict["ListedOrModified"] == "Listing Date":
            filter_string = f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEnd']}"
        elif __Source_dict["ListedOrModified"] == "Modification Date":
            filter_string = f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
        elif __Source_dict["ListedOrModified"] == "Close Date":
            filter_string = f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"

        filter_string = filter_string + f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['StandardStatus']}'"

        if __Source_dict["select"] is not None:
            filter_string = filter_string + f'&$select={__Source_dict["select"]}'

        self.__parameterString = filter_string

    def __getCount(self):
        """
    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
    """
        __count_resp = None

        try:
            __count_resp =
requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
                                        headers=self.__headerDict)

            if __count_resp.status_code != 200:
                RESTError(__count_resp)
                raise SystemExit(0)

            self.__record_val = int(__count_resp.json()["@odata.count"])

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

    def __getCountUI(self):

        """
    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
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress",
error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```

Index