

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..... | 2 |
| Class Index | 7 |
| File Index..... | 8 |
| Class Documentation..... | 9 |
| AuthUtil.AuthUtil..... | 9 |
| BatchProcessing.BatchProcessorConstructionMonitor | 18 |
| BatchProcessing.BatchProcessorUtahRealEstate | 23 |
| BatchProgressGUI.BatchProgressGUI..... | 27 |
| Core.Cencus | 36 |
| Core.ConstructionMonitorInit | 40 |
| Core.ConstructionMonitorMain | 48 |
| DataTransfer.DataTransfer | 56 |
| FileSaver.FileSaver | 59 |
| API_Calls.Initializer.initializer..... | 63 |
| PopupWrapped.PopupWrapped | 69 |
| Core.realtorCom | 76 |
| Core.UtahRealEstateInit | 82 |
| Core.UtahRealEstateMain | 90 |
| File Documentation | 99 |
| __init__.py..... | 99 |
| __main__.py | 100 |
| AuthUtil.py..... | 101 |
| BatchProcessing.py | 106 |
| DataChecker.py | 110 |
| DataSupportFunctions.py | 111 |
| FileSaver.py..... | 112 |
| ErrorPopup.py | 114 |
| ErrorPrint.py | 115 |
| Logger.py | 116 |
| PrintFunc.py | 117 |
| RESError.py..... | 118 |
| BatchGui.py..... | 121 |
| BatchProgressGUI.py | 122 |
| DataTransfer.py | 127 |
| ImageLoader.py..... | 129 |
| PopupWrapped.py | 130 |
| Initializer.py | 134 |
| Core.py | 137 |
| Core.py | 139 |
| Core.py | 146 |
| Core.py | 148 |
| Index..... | 155 |

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

License

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

Class Index

Class List

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

| | |
|--|----|
| <u>AuthUtil.AuthUtil</u> | 9 |
| <u>BatchProcessing.BatchProcessorConstructionMonitor</u> | 18 |
| <u>BatchProcessing.BatchProcessorUtahRealEstate</u> | 23 |
| <u>BatchProgressGUI.BatchProgressGUI</u> | 27 |
| <u>Core.Cencus</u> | 36 |
| <u>Core.ConstructionMonitorInit</u> | 40 |
| <u>Core.ConstructionMonitorMain</u> | 48 |
| <u>DataTransfer.DataTransfer</u> | 56 |
| <u>FileSaver.FileSaver</u> | 59 |
| <u>API Calls.Initializer.initializer</u> | 63 |
| <u>PopupWrapped.PopupWrapped</u> | 69 |
| <u>Core.realtorCom</u> | 76 |
| <u>Core.UtahRealEstateInit</u> | 82 |
| <u>Core.UtahRealEstateMain</u> | 90 |

File Index

File List

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

| | |
|--|-----|
| <u>init .py</u> | 99 |
| <u>main .py</u> | 100 |
| <u>AuthUtil.py</u> | 101 |
| <u>BatchProcessing.py</u> | 106 |
| <u>DataChecker.py</u> | 110 |
| <u>DataSupportFunctions.py</u> | 111 |
| <u>FileSaver.py</u> | 112 |
| <u>ErrorPopup.py</u> | 114 |
| <u>ErrorPrint.py</u> | 115 |
| <u>Logger.py</u> | 116 |
| <u>PrintFunc.py</u> | 117 |
| <u>RESError.py</u> | 118 |
| <u>BatchGui.py</u> | 121 |
| <u>BatchProgressGUL.py</u> | 122 |
| <u>DataTransfer.py</u> | 127 |
| <u>ImageLoader.py</u> | 129 |
| <u>PopupWrapped.py</u> | 130 |
| <u>Initializer.py</u> | 134 |
| <u>CFBP/Core.py</u> | 137 |
| <u>ConstructionMonitor/Core.py</u> | 139 |
| <u>Realtor/Core.py</u> | 146 |
| <u>UtahRealEstate/Core.py</u> | 148 |

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
00025         it creates a new window and displays it on screen.
```

```

00026     Args:
00027         self: Represent the instance of the class
00028
00029     Returns:
00030         None
00031
00032     Doc Author:
00033         Willem van der Schans, Trelent AI
00034     """
00035         self.StandardStatus = None
00036         self.ListedOrModified = None
00037         self.file_name = None
00038         self.append_file = None
00039         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00040         self.filePath =
Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath("Secu
rity")
00041         self.k = None
00042         self.keyFlag = True
00043         self.jsonDict = {}
00044         self.passFlagUre = False
00045         self.passFlagCm = False
00046         self.outcomeText = "Please input the plain text keys in the input boxes
above \n " \
00047             "Submitting will overwrite any old values in an
unrecoverable manner."
00048
00049         if os.path.exists(self.filePath):
00050             pass
00051         else:
00052             if
os.path.exists(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData")
):
00053                 os.mkdir(self.filePath)
00054             else:
00055                 os.mkdir(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData"))
00056                 os.mkdir(self.filePath)
00057
00058         if os.path.exists(self.keyPath):
00059             pass
00060         else:
00061             if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00062                 os.mkdir(self.keyPath)
00063             else:
00064                 os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00065                 os.mkdir(self.keyPath)
00066
00067         if
os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00068             try:
00069                 f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00070                 self.k = f.readline()
00071                 f.close()
00072             except Exception as e:
00073                 print(e)
00074                 RESTError(402)
00075                 raise SystemExit(402)
00076         else:
00077             self.k = Fernet.generate_key()
00078             f =
open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00079             f.write(self.k)
00080             f.close()
00081
00082         try:
00083             os.remove(self.filePath.joinpath("auth.json"))
00084         except Exception as e:
00085             # Logging
00086             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..."
)
00088             pass

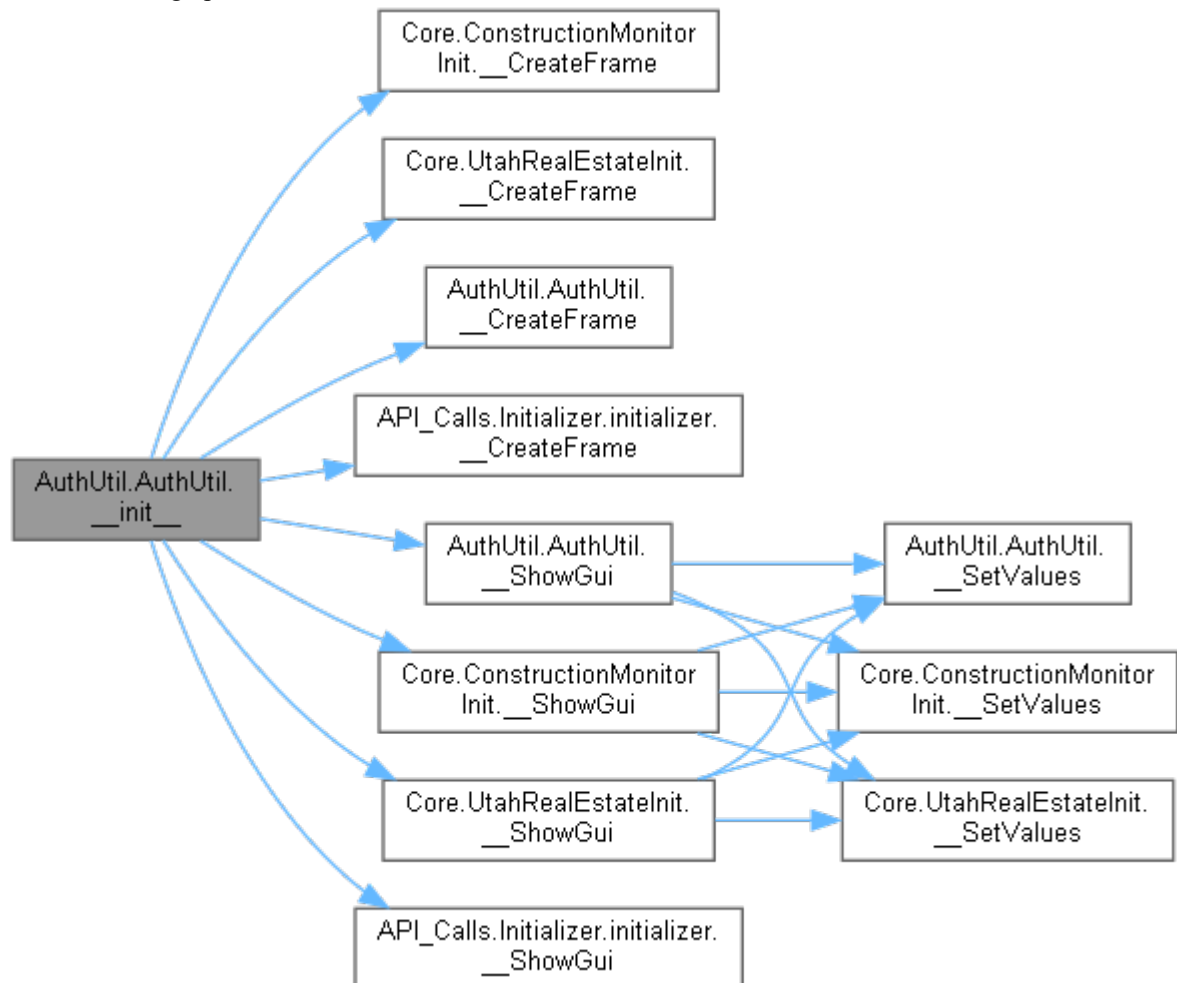
```

```

00089         f = open(self.filePath.joinpath("auth.json"), "wb")
00090         f.close()
00091         self.keyFlag = False
00092
00093     self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00094
00095     try:
00096         ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35
00097         .av3ra3rvavcr3w"), 2)
00098     except Exception as e:
00099         # Logging
00100         print(
00101             f"{datetime.datetime.today().strftime('%m-%d-%Y
00102             %H:%M:%S.%f')}[:-3]} | Authutil.py | Error = {e} | Error when setting the key file as
00103             hidden. This is either a Permission error or Input Error. Continuing...")
00104         pass

```

Here is the call graph for this function:



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

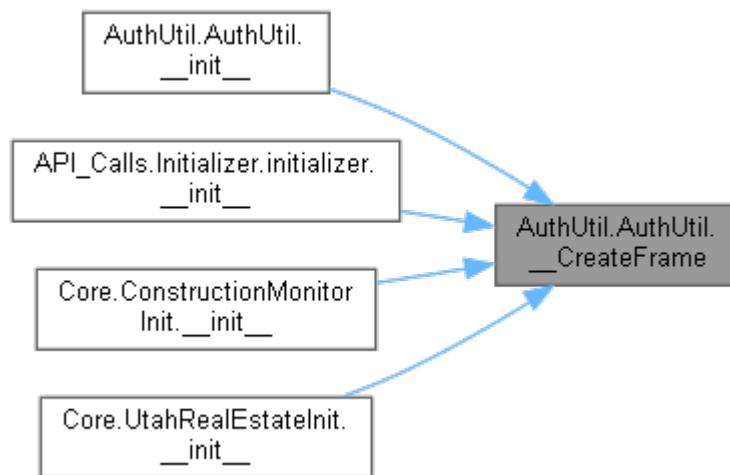
Doc Author:

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
00227         Utility.
00228         It is called by __init__ and returns a list of lists that contains all the
00229         elements
00230         that will be displayed in the window.
00231     Args:
00232         self: Access the class attributes and methods
00233     Returns:
00234         A list of lists
00235     Doc Author:
00236         Trelent
00237     """
00238     sg.theme('Default1')
00239
00240     line00 = [sg.HSeparator()]
00241
00242     line0 = [sg.Image(ImageLoader("logo.png")),
00243             sg.Push(),
00244             sg.Text("Authentication Utility", font=("Helvetica", 12,
00245 "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: ",
00264 justification="center"),
00265             sg.Push()]
00266
00267     line6 = [sg.Push(),
00268             sg.Input(default_text="", key="-cmAuth-", disabled=False,
00269                     size=(40, 1)),
00270             sg.Push()]
00271
00272     line7 = [sg.HSeparator()]
00273
00274     line8 = [sg.Push(),
00275             sg.Text(self.outcomeText, justification="center"),
00276             sg.Push()]
00277
00278     line9 = [sg.HSeparator()]
00279
00280     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00281
00282     layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
00283 line8, line9, line10]
00284
00285     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 "OK" 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](#).

```

00104     def __SetValues(self, values):
00105
00106         """
00107         The __SetValues function is called when the user clicks on the "OK"
button in the window.
00108         It takes a dictionary of values as an argument, and then uses those values
to update
00109         the auth.json file with new keys for both Utah Real Estate and Construction
Monitor.
00110
00111         Args:
00112             self: Make the function a method of the class
00113             values: Store the values that are entered into the form
00114
00115         Returns:
00116             A dictionary of the values entered by the user
00117
00118         Doc Author:
00119             Willem van der Schans, Trelent AI
00120         """
00121         ureCurrent = None
00122         cmCurrent = None
00123         keyFile = None
00124
00125         fernet = Fernet(self.k)
00126
00127         try:
00128             f = open(self.filePath.joinpath("auth.json"), "r")
00129             keyFile = json.load(f)
00130             fileFlag = True
00131         except:
00132             fileFlag = False
00133
00134         if fileFlag:

```

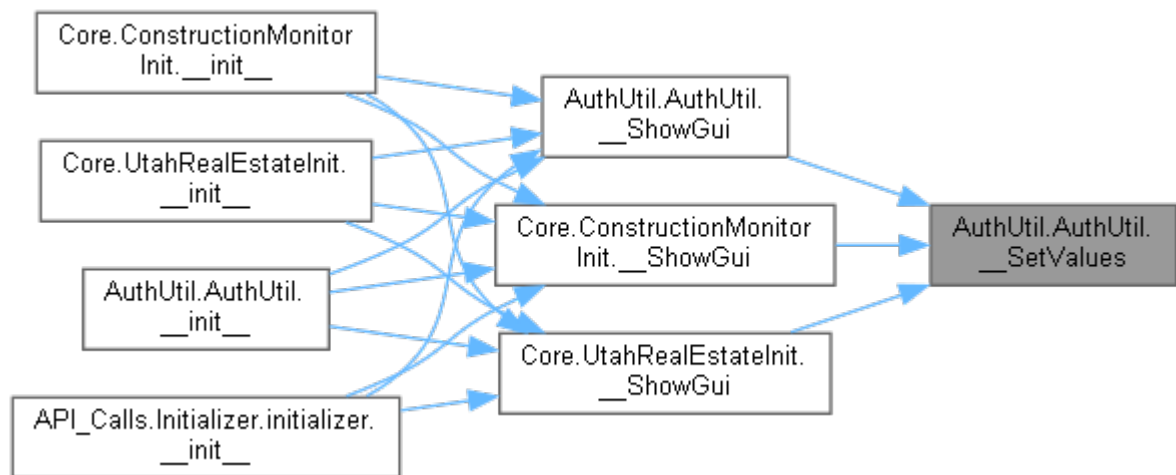


```

00135         try:
00136             ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00137         except Exception as e:
00138             # Logging
00139             print(
00140                 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")
00141             ureCurrent = None
00142
00143         try:
00144             cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00145         except Exception as e:
00146             # Logging
00147             print(
00148                 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")
00149             cmCurrent = None
00150
00151         if values["-ureAuth-"] != "":
00152             self.jsonDict.update(
00153                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00154             self.passFlagUre = True
00155         elif ureCurrent is not None:
00156             self.jsonDict.update(
00157                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00158             self.passFlagUre = True
00159         else:
00160             pass
00161
00162         if values["-cmAuth-"] != "":
00163             self.jsonDict.update(
00164                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00165             self.passFlagCm = True
00166         elif ureCurrent is not None:
00167             self.jsonDict.update(
00168                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00169             self.passFlagUre = True
00170         else:
00171             pass
00172
00173         if not self.passFlagUre and not self.passFlagCm:
00174             PopupWrapped("Please make sure you provide keys for both Utah Real
estate and Construction Monitor",
00175                 windowType="errorLarge")
00176         if self.passFlagCm and not self.passFlagUre:
00177             PopupWrapped("Please make sure you provide a key for Utah Real
estate", windowType="errorLarge")
00178         if not self.passFlagCm and self.passFlagUre:
00179             PopupWrapped("Please make sure you provide a key for Construction
Monitor", windowType="errorLarge")
00180         else:
00181             jsonOut = json.dumps(self.jsonDict, indent=4)
00182             f = open(self.filePath.joinpath("auth.json"), "w")
00183             f.write(jsonOut)
00184

```

Here is the caller graph for this function:



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

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

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

Returns:
 A window object

Definition at line 185 of file [AuthUtil.py](#).

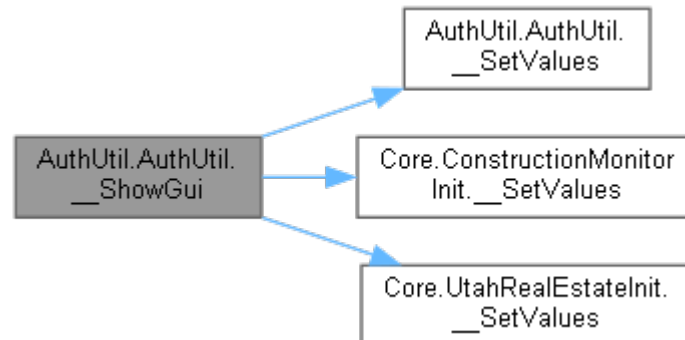
```
00185     def __ShowGui(self, layout, text):
00186
00187         """
00188         The __ShowGui function is a helper function that displays the GUI to the user.
00189         It takes in two arguments: layout and text. The layout argument is a list
00190         of lists,
00191         which contains all the elements that will be displayed on screen. The text
00192         argument
00193         is simply what will be displayed at the top of the window.
00194
00195         Args:
00196         self: Represent the instance of the class
00197         layout: Pass the layout of the gui to be displayed
00198         text: Set the title of the window
00199
00200         Returns:
00201         A window object
00202         """
00203         window = sg.Window(text, layout, grab_anywhere=False,
00204                             return_keyboard_events=True,
00205                             finalize=True,
00206                             icon=ImageLoader("taskbar_icon.ico"))
00207
00208         while not self.passFlagUre or not self.passFlagCm:
00209             event, values = window.read()
00210
00211             if event == "Submit":
00212                 try:
00213                     self.__SetValues(values)
00214                 except Exception as e:
00215                     print(e)
00216                     RESTError(993)
00217                 finally:
00218                     pass
00219             elif event == sg.WIN_CLOSED or event == "Quit":
```

```

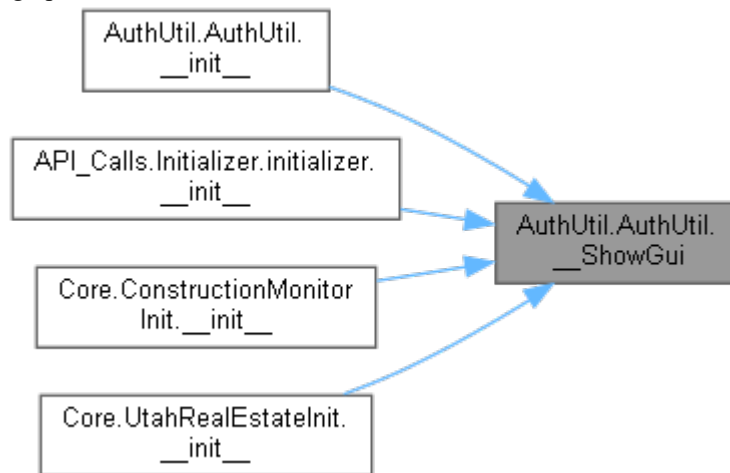
00218         break
00219     else:
00220         pass
00221
00222     window.close()
00223

```

Here is the call graph for this function:



Here is the caller graph for this function:



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,
00044 ColumnSelection, valueObject):
00045         """
00046         The __init__ function is the constructor for a class. It is called when an
00047         object of that class
00048         is created, and it sets up the attributes of that object. In this case, we
00049         are setting up our
00050         object to have a dataframe attribute (which will be used to store all of our
00051         data), as well as
00052         attributes for each parameter in our ReST call.
00053     Args:
00054         self: Represent the instance of the class
00055         RestDomain: Specify the domain of the rest api
00056         NumBatches: Determine how many batches of data to retrieve
00057         ParameterDict: Pass in the parameters that will be used to make the api
00058         call
00059         HeaderDict: Pass the header dictionary from the main function to this
00060         class
00061         ColumnSelection: Determine which columns to pull from the api
00062         valueObject: Pass in the value object that is used to determine what
00063         values are returned
00064     Returns:
00065         An object of the class
00066     Doc Author:
00067         Willem van der Schans, Trelent AI
00068     """
00069     self.dataframe = None
00070     self.__numBatches = NumBatches
00071     self.__parameterDict = ParameterDict
00072     self.__restDomain = RestDomain
00073     self.__headerDict = HeaderDict
00074     self.__columnSelection = ColumnSelection
00075     self.valueObject = valueObject
00076     self.__maxRequests = 10000
00077     self.__requestCount = math.ceil(self.__numBatches /
00078 (self.__maxRequests / int(self.__parameterDict['size'])))
00079     self.__requestCalls = math.ceil(self.__maxRequests /
00080 int(self.__parameterDict['size']))
00081     self.__dateTracker = None
00082 
```

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         """

```

```

00096     The ConstructionMonitorProcessor function will use requests to get data from
00097     ConstructionMontior.com's ReST API and store it into a pandas DataFrame
00098     object called __df (which is local). This
00099     process will be repeated until all the data has been collected from
00100     ConstructionMonitor.com's ReST API, at which point __df will contain all
00101
00102     Args:
00103         self: Represent the instance of the object itself
00104         valueObject: Update the progress bar in the gui
00105
00106     Returns:
00107         A dataframe
00108
00109     Doc Author:
00110         Willem van der Schans, Trelent AI
00111
00112     """
00113     __df = None
00114     for callNum in range(0, self.__requestCount):
00115         self.__parameterDict["from"] = 0
00116
00117         if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00118             batchNum = self.__requestCalls
00119             if __df is None:
00120                 self.__dateTracker = str(date.today())
00121             else:
00122                 self.__dateTracker =
00123                 min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00124             elif self.__requestCount == 1:
00125                 __batchNum = self.__numBatches
00126                 self.__dateTracker = str(date.today())
00127             else:
00128                 __batchNum = self.__numBatches / (self.__maxRequests /
00129                 int(self.__parameterDict['size'])) - (
00130                 self.__requestCount - 1)
00131             self.__dateTracker =
00132             min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00133             self.__parameterDict['dateEnd'] = self.__dateTracker
00134
00135             for record in range(0, int(math.ceil(__batchNum))):
00136                 if record != 0:
00137                     self.__parameterDict["from"] = record *
00138                     int(self.__parameterDict["size"])
00139
00140                     response = requests.post(url=self.__restDomain,
00141                     headers=self.__headerDict,
00142                     json=self.__parameterDict)
00143
00144                     counter = 0
00145                     try:
00146                         response = response.json()['hits']['hits']
00147                     except KeyError as e:
00148                         # Logging
00149                         print(
00150                             f"{datetime.datetime.today().strftime('%m-%d-%Y
00151                             %H:%M:%S.%f')}[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server
00152                             Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict}
00153                             | Headers = {self.__headerDict}")
00154                     continue
00155
00156                     valueObject.setValue(valueObject.getValue() + 1)
00157
00158                     if record == 0 and callNum == 0:
00159                         __df = pd.json_normalize(response[counter]["_source"])
00160                         __df["id"] = response[counter]['_id']
00161                         __df["county"] =
00162                         response[counter]["_source"]['county']['county_name']
00163                         counter += 1
00164
00165                     for i in range(counter, len(response)):
00166                         __tdf = pd.json_normalize(response[i]["_source"])
00167                         __tdf["id"] = response[i]['_id']
00168                         __tdf["county"] =
00169                         response[i]["_source"]['county']['county_name']
00170                         __df = pd.concat([__df, __tdf], ignore_index=True)
00171
00172                     if self.__columnSelection is not None:

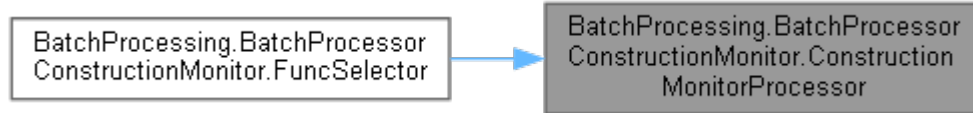
```

```

00162         __col_list = StringToList(self.__columnSelection)
00163         __col_list.append("id")
00164         __col_list.append("county")
00165     else:
00166         pass
00167
00168     self.dataframe = __df
00169     valueObject.setValue(-999)
00170
00171

```

Here is the caller graph for this function:



def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (self)

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

Args:

self: Represent the instance of the class

Returns:

The result of the constructionmonitorprocessor function

Doc Author:

Willem van der Schans, Trelent AI

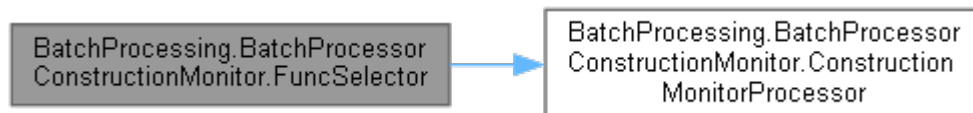
Definition at line 78 of file [BatchProcessing.py](#).

```

00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes
00081         it to the ConstructionMonitorProcessor function.
00082         The ConstructionMonitorProcessor function then uses this valueObject to
00083         determine which of its functions should be called.
00084
00085         Args:
00086             self: Represent the instance of the class
00087
00088         Returns:
00089             The result of the constructionmonitorprocessor function
00090
00091         Doc Author:
00092             Willem van der Schans, Trelent AI
00093         """
00094         self.ConstructionMonitorProcessor(self.valueObject)
00095

```

Here is the call graph for this function:



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
00177     object of that class is instantiated, and it sets up the attributes of that object. In this case,
00178     we are setting up
00179     the dataframe attribute to be None (which will be set later), and we are also
00180     setting up some
00181     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
00222         the valueObject and uses it to
00223         update the progress bar. It also takes in self, which contains all the
00224         necessary information for this
00225         function to work properly. The BatchProcessingUtahRealestateCom function
00226         will then use requests to get data from
00227         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
00228         called __df (which is local). This
00229         process will be repeated until all the data has been collected from
00230         UtahRealestate.com's ReST API, at which point __df will contain all
00231
00232     Args:
00233         self: Represent the instance of the class
00234         valueObject: Pass the value of a progress bar to the function
00235
00236     Returns:
00237         A dataframe of the scraped data

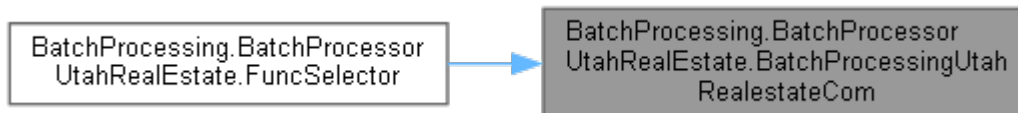
```

```

00233
00234     Doc Author:
00235         Willem van der Schans, Trelent AI
00236     """
00237     __df = pd.DataFrame()
00238
00239     for batch in range(self.__numBatches):
00240
00241         if batch == 0:
00242             response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00243             headers=self.__headerDict)
00244
00245             response_temp = response.json()
00246             __df = pd.json_normalize(response_temp, record_path=['value'])
00247
00248         else:
00249             response =
requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
00250             headers=self.__headerDict)
00251
00252             response_temp = response.json()
00253             response_temp = pd.json_normalize(response_temp,
record_path=['value'])
00254             __df = pd.concat([__df, response_temp], ignore_index=True)
00255
00256             valueObject.setValue(valueObject.getValue() + 1)
00257
00258             self.dataframe = __df
00259             valueObject.setValue(-999)

```

Here is the caller graph for this function:



def BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector (self)

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

Args:
self: Represent the instance of the class

Returns:
The function that is selected by the user

Doc Author:
Willem van der Schans, Trelent AI

Definition at line [202](#) of file [BatchProcessing.py](#).

```

00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an
argument and then calls the appropriate
00205         function based on what was selected in the dropdown menu. The
valueObject is passed to each of these functions
00206         so that they can access all of its attributes.
00207
00208         Args:
00209             self: Represent the instance of the class
00210
00211         Returns:
00212             The function that is selected by the user
00213
00214         Doc Author:
00215             Willem van der Schans, Trelent AI

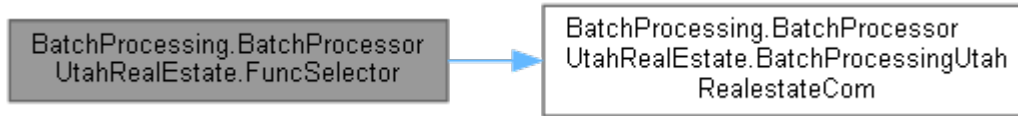
```

```

00216     """
00217         self.BatchProcessingUtahRealestateCom(self.valueObject)
00218

```

Here is the call graph for this function:



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

[dataframe](#)Private 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,
00020                  ColumnSelection=None):
00021         """
00022         The __init__ function is the first function that gets called when an object
00023         of this class is created.
00024         It initializes all the variables and sets up a layout for the GUI. It also
00025         creates a window to display
00026         the dataframe in.
00027     Args:
00028         self: Represent the instance of the class
00029         BatchesNum: Determine the number of batches that will be created
00030         RestDomain: Specify the domain of the rest api
00031         ParameterDict: Pass the parameters of the request to the class
00032         HeaderDict: Store the headers of the dataframe
00033         Type: Determine the type of dataframe that is being created
00034         ColumnSelection: Select the columns to be displayed in the gui
00035     Returns:
00036         Nothing
00037     Doc Author:
00038         Willem van der Schans, Trelent AI
00039     """
00040     self.__parameterDict = ParameterDict
00041     self.__restDomain = RestDomain
00042     self.__headerDict = HeaderDict
00043     self.__columnSelection = ColumnSelection
00044     self.__type = Type
00045     self.dataframe = None
00046
00047     self.__layout = None
00048     self.__batches = BatchesNum
00049     self.__window = None
00050     self.__batch_counter = 0
00051
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
00056         progress bar layout and then calls the createGui function to create a GUI for batch
00057         processing.
00058     Args:
00059         self: Represent the instance of the class
00060     Returns:
00061         The __type of the batchgui class
00062     Doc Author:
00063         Willem van der Schans, Trelent AI
00064     """
00065

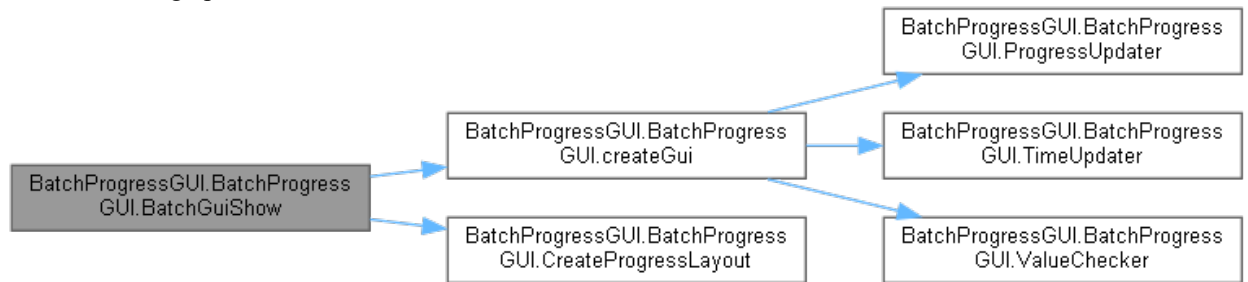
```

```

00066         self.CreateProgressLayout()
00067         self.createGui(self.__type)
00068

```

Here is the call graph for this function:



def BatchProgressGUI.BatchProgressGUI.createGui (self, Sourcetype)

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

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

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI

Definition at line [104](#) of file [BatchProgressGUI.py](#).

```

00104     def createGui(self, Sourcetype):
00105
00106         """
00107         The createGui function is the main function that creates the GUI.
00108         It takes in a type parameter which determines what kind of batch processor
00109         to use.
00110         The createGui function then sets up all the variables and objects needed for
00111         the program to run, including: window, start_time, update_text, valueObj
00112         (DataTransfer),
00113         processorObject (BatchProcessorConstructionMonitor or
00114         BatchProcessorUtahRealestate),
00115         and threading objects for TimeUpdater and ValueChecker functions. The
00116         createGui function also starts these threads.
00117
00118         Args:
00119             self: Access the object itself
00120             Sourcetype: Determine which batch processor to use
00121
00122         Returns:
00123             The dataframe
00124
00125         Doc Author:
00126             Willem van der Schans, Trelent AI
00127         """
00128         self.__window = sg.Window('Progress', self.__layout, finalize=True,
00129         icon=ImageLoader("taskbar_icon.ico"))
00130
00131         start_time = datetime.datetime.now().replace(microsecond=0)
00132         update_text = f"Batch {0} completed"
00133         self.__window['--progress_text--'].update(update_text)
00134         self.__window['--progress_bar--'].update(0)
00135         self.__window['--time_est--'].update("Est time needed 00:00:00")
00136
00137         valueObj = DataTransfer()

```

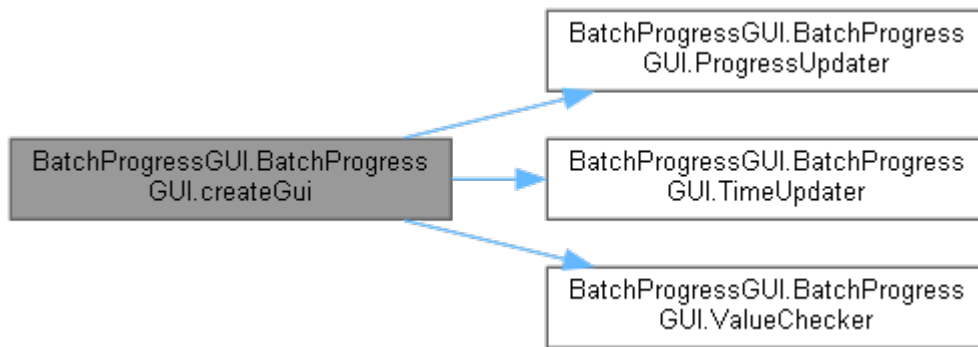


```

00133         valueObj.setValue(0)
00134
00135         if Sourcetype == "construction_monitor":
00136             processorObject =
BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00138
NumBatches=self.__batches,
00139
ParameterDict=self.__parameterDict,
00140
HeaderDict=self.__headerDict,
00141
ColumnSelection=self.__columnSelection,
00142
valueObject=valueObj)
00143         elif Sourcetype == "utah_real_estate":
00144             processorObject =
BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00145
NumBatches=self.__batches,
00146
ParameterString=self.__parameterDict,
00147
HeaderDict=self.__headerDict,
00148
valueObject=valueObj)
00149
00150             threading.Thread(target=self.TimeUpdater,
00151                             args=(start_time,),
00152                             daemon=True).start()
00153             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | TimeUpdater Thread Successfully Started")
00154
00155             batchFuncThread =
threading.Thread(target=processorObject.FuncSelector,
00156                 daemon=False)
00157             batchFuncThread.start()
00158             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | BatchFunc Thread Successfully Started")
00159             threading.Thread(target=self.ValueChecker,
00160                             args=(valueObj,),
00161                             daemon=False).start()
00162             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | ValueChecker Thread Successfully Started")
00163
00164             while True:
00165
00166                 self.ProgressUpdater(valueObj)
00167
00168                 if valueObj.getValue() == -999:
00169                     break
00170
00171                 window, event, values = sg.read_all_windows()
00172                 if event.startswith('update'):
00173                     __key_to_update = event[len('update'):]
00174                     window[__key_to_update].update(values[event])
00175                     window.refresh()
00176                     pass
00177
00178                 if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179                     break
00180
00181                 time.sleep(0.1)
00182
00183             self.dataframe = processorObject.dataframe
00184             self.__window.close()
00185
00186             PopupWrapped(text="Api Request Completed", windowType="notice")
00187

```

Here is the call graph for this function:



Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (self)

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

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

Args:
self: Access the class variables and methods

Returns:
A list of lists

Doc Author:
Willem van der Schans, Trelent AI

Definition at line 69 of file [BatchProgressGUI.py](#).

```

00069     def CreateProgressLayout(self):
00070
00071         """
00072         The CreateProgressLayout function creates the layout for the progress window.
00073         The function takes in self as a parameter and returns nothing.
00074
00075         Parameters:
00076             self (object): The object that is calling this function.
00077
00078         Args:
00079             self: Access the class variables and methods
00080
00081         Returns:
00082             A list of lists
00083
00084         Doc Author:
00085             Willem van der Schans, Trelent AI
00086         """
00087         sg.theme('Default1')
00088
00089         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
00090 justification="center", key="--progress_text--"),
00091 sg.Push()]
00092
00093         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
00094 justification="center", key="--timer--"),
00095 sg.Text(font=("Helvetica", 10), justification="center",
00096 key="--time_est--"), sg.Push()]
00097
00098         __Line3 = [
00099 sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
00100 "#C9c8c8"), orientation='h', size=(30, 20),
00101 key='--progress_bar--')]
  
```

```

00099
00100         layout = [__Line1, __Line2, __Line3]
00101
00102         self.__layout = layout
00103

```

Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.ProgressUpdater (self, valueObj)

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

Args:

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

Returns:

The value of the batch counter

Doc Author:

Willem van der Schans, Trelent AI

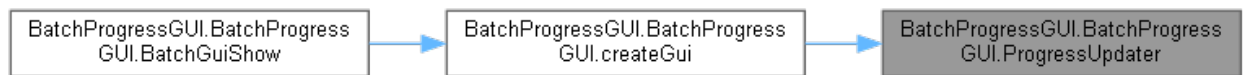
Definition at line [188](#) of file [BatchProgressGUI.py](#).

```

00188     def ProgressUpdater(self, valueObj):
00189         """
00190         The ProgressUpdater function is a callback function that updates the progress
00191         bar and text
00192         in the GUI. It takes in one argument, which is an object containing information
00193         about the
00194         current batch number. The ProgressUpdater function then checks if this value
00195         has changed from
00196         the last time it was called (i.e., if we are on a new batch). If so, it updates
00197         both the progress
00198         bar and text with this new information.
00199
00200         Args:
00201             self: Make the progressupdater function an instance method
00202             valueObj: Get the current value of the batch counter
00203
00204         Returns:
00205             The value of the batch counter
00206
00207         Doc Author:
00208             Willem van der Schans, Trelent AI
00209         """
00210         if valueObj.getValue() != self.__batch_counter:
00211             self.__batch_counter = valueObj.getValue()
00212             __update_text = f"Batch {self.__batch_counter}/{self.__batches}
00213             completed"
00214             self.__window.write_event_value('update--progress_bar--',
00215             self.__batch_counter)
00216             self.__window.write_event_value('update--progress_text--',
00217             __update_text)
00218         else:
00219             pass
00220

```

Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.TimeUpdater (self, start_time)

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

Args:

self: Make the function a method of the class

start_time: Get the time when the function is called

Returns:

A string that is updated every 0

Doc Author:

Willem van der Schans, Trelent AI

Definition at line 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
00220         estimated time needed to complete
00221         the current batch. It does this by reading the start_time variable passed
00222         in, getting the current time,
00223         calculating how much time has passed since start_time was set and then
00224         updating a timer string with that value.
00225         It then calculates an estimation of how long it will take to finish all batches
00226         based on how many batches have been completed so far.
00227
00228         Args:
00229             self: Make the function a method of the class
00230             start_time: Get the time when the function is called
00231
00232         Returns:
00233             A string that is updated every 0
00234
00235         Doc Author:
00236             Willem van der Schans, Trelent AI
00237         """
00238         while True:
00239             if self.__batch_counter < self.__batches:
00240                 __current_time =
00241                 datetime.datetime.now().replace(microsecond=0)
00242                 __passed_time = __current_time - start_time
00243                 __timer_string = f"Time Elapsed {__passed_time}"
00244                 try:
00245                     self.__window.write_event_value('update--timer--',
00246                     __timer_string)
00247                     except AttributeError as e:
00248                         print(
00249                             f"{datetime.datetime.today().strftime('%m-%d-%Y
00250                             %H:%M:%S.%f')}[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error,
00251                             this is okay if the display looks good, this exception omits fatal crashes due to an
00252                             aesthetic error")
00253                         break
00254                 __passed_time = __passed_time.total_seconds()
00255                 try:
00256                     __time_est = datetime.timedelta(

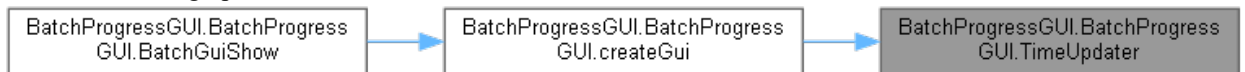
```

```

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:



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

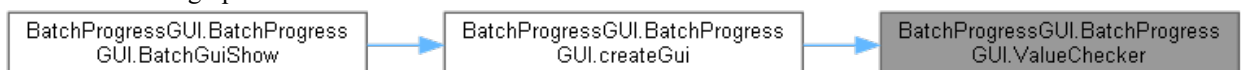
Definition at line [268](#) of file [BatchProgressGUI.py](#).

```

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
00272         True.
00273         If not, then it returns False.
00274
00275         Args:
00276         self: Represent the instance of the class
00277         ObjectVal: Get the value of the object
00278
00279         Returns:
00280         True if the value of the object has changed, and false if it hasn't
00281
00282         Doc Author:
00283         Willem van der Schans, Trelent AI
00284         """
00285         while True:
00286             time.sleep(0.3)
00287             if self.__batch_counter != ObjectVal.getValue():
00288                 self.__batch_counter = ObjectVal.getValue()
00289                 return True
00290             else:
00291                 return False

```

Here is the caller graph for this function:



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_arg](#)[year_arg](#)
- [uiString](#)
- [link](#)

Private Member Functions

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

Detailed Description

Definition at line [12](#) of file [CFBP/Core.py](#).

Constructor & Destructor Documentation

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

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

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

Returns:
A popupwrapped object

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

Definition at line [14](#) of file [CFBP/Core.py](#).

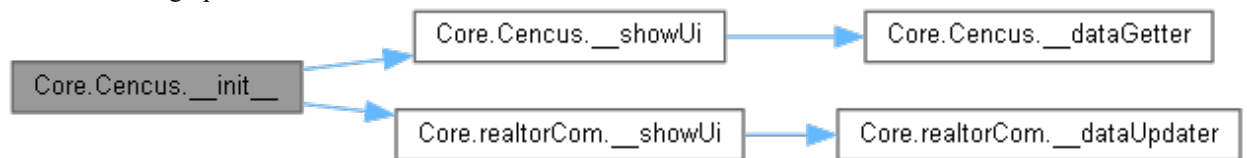
```
00014     def __init__(self, state_arg=None, year_arg=None):
00015         """
00016         The __init__ function is called when the class is instantiated.
00017         It's job is to initialize the object with some default values, and do any
00018         other setup that might be necessary.
00019         The __init__ function can take arguments, but it doesn't have to.
00020
00021         Args:
00022             self: Represent the instance of the class
00023             state_arg: Set the state_arg attribute of the class
00024             year_arg: Set the year of data to be retrieved
00025
00026         Returns:
00027             A popupwrapped object
00028
00029         Doc Author:
00030             Willem van der Schans, Trelent AI
00031         """
00032         self.state_arg = state_arg
00033         self.year_arg = year_arg
```

```

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

```

Here is the call graph for this function:



Member Function Documentation

def Core.Census.__dataGetter (self)[private]

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

Args:
self: Represent the instance of the class

Returns:
A response object

Doc Author:
Willem van der Schans, Trelent AI

Definition at line 72 of file [CFBP/Core.py](#).

```

00072     def __dataGetter(self):
00073         """
00074         The __dataGetter function is a private function that gets the data from the
CFPB API.
00075         It takes no arguments, but uses self.state_arg and self.year_arg to create
a URL for the API call.
00076
00077         Args:
00078             self: Represent the instance of the class
00079
00080         Returns:
00081             A response object
00082
00083         Doc Author:
00084             Willem van der Schans, Trelent AI
00085         """
00086         arg_dict_bu = locals()
00087
00088         link = "https://ffiec.cfbp.gov/v2/data-browser-api/view/csv?"
00089
00090         if self.state_arg is None:
00091             self.state_arg = "UT"
00092         else:
00093             pass
00094
00095         if self.year_arg is None:
00096             self.year_arg = str(date.today().year - 1)
00097         else:
00098             pass
00099
00100         passFlag = False

```

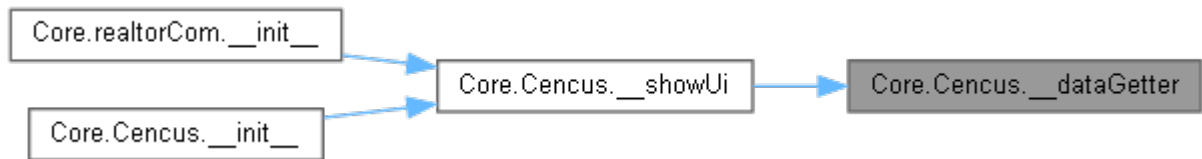


```

00101
00102         while not passFlag:
00103
00104             self.link =
00105             "https://ffiec.cfbp.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
00106             + f"&years={self.year_arg}"
00107
00108             response = requests.get(self.link)
00109
00110             if response.status_code == 400:
00111                 self.year_arg = int(self.year_arg) - 1
00112
00113             else:
00114                 passFlag = True
00115
00116             RESTError(response)
00117             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](#).

```

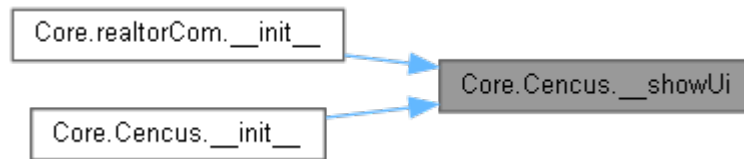
00044     def __showUi(self):
00045
00046         """
00047         The __showUi function is a function that creates a progress bar window.
00048         The __showUi function takes class variables and returns a windowobj.
00049
00050
00051         Args:
00052             self: Represent the instance 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",
00061                               windowType="progress", error=None)
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](#).

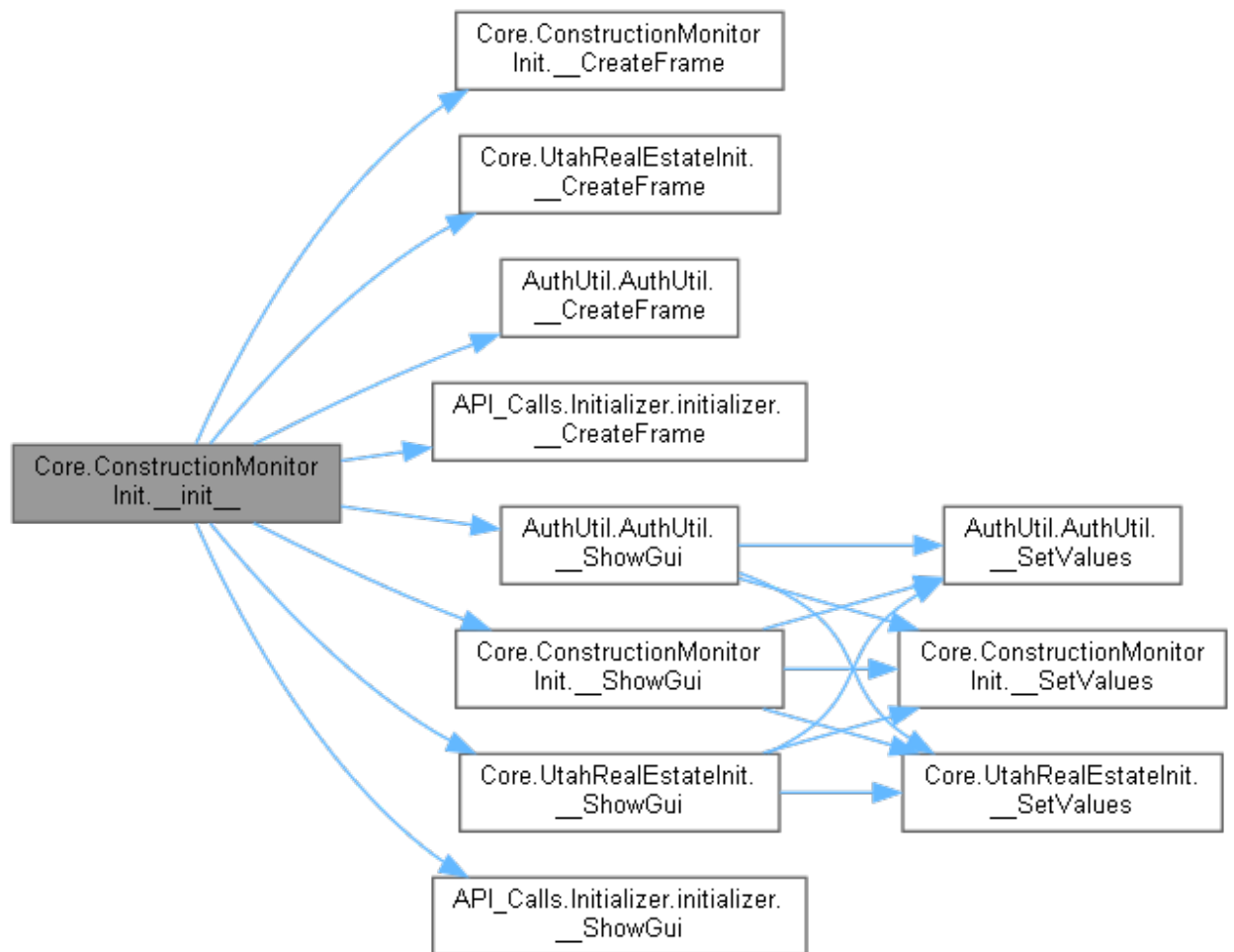
```
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the variables that will be used by other functions in this class.
00031
00032
00033         Args:
00034             self: Represent the instance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

```

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

```

Here is the call graph for this function:



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](#).

```

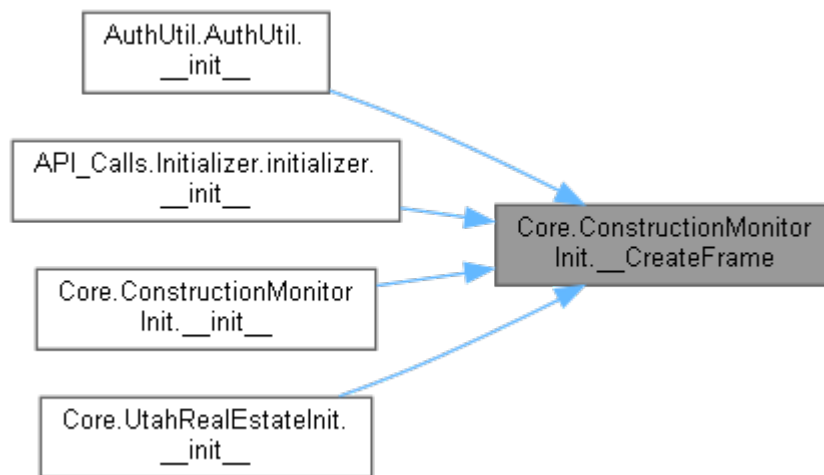
00116     def __CreateFrame():
00117
00118         """
00119         The __CreateFrame function creates the GUI layout for the application.
00120         The function returns a list of lists that contains all the elements to
00121         be displayed in the GUI window.
00122         This is done by creating each line as a list and then appending it to
00123         another list which will contain all lines.
00124
00125         Args:
  
```

```

00124
00125     Returns:
00126         The layout for the gui
00127
00128     Doc Author:
00129         Willem van der Schans, Trelent AI
00130     """
00131     sg.theme('Default1')
00132
00133     line00 = [sg.HSeparator()]
00134
00135     line0 = [sg.Image(ImageLoader("logo.png")),
00136             sg.Push(),
00137             sg.Text("Construction Monitor Utility", font=("Helvetica",
00138 12, "bold"), justification="center"),
00138             sg.Push(),
00139             sg.Push()]
00140
00141     line1 = [sg.HSeparator()]
00142
00143     line3 = [sg.Text("Start Date : ", size=(15, None),
00144 justification="Right"),
00144             sg.Input(default_text=(date.today() -
00145 timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
00146 size=(20, 1)),
00147             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00148 key="-start_date-", target="-Cal-")]
00149
00150     line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00151             sg.Input(default_text=date.today().strftime("%Y-%m-%d"),
00152 key="-EndCal-",
00153 size=(20, 1)),
00154             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00155 key="-start_date-", target="-EndCal-")]
00156
00157     line5 = [sg.HSeparator()]
00158
00159     line6 = [sg.Push(),
00160             sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00161 justification="center"),
00162             sg.Push()]
00163
00164     line7 = [sg.HSeparator()]
00165
00166     line8 = [sg.Text("Appending File : ", size=(15, None),
00167 justification="Right"),
00168             sg.Input(default_text="", key="-AppendingFile-",
00169 disabled=True,
00170 size=(20, 1)),
00171             sg.FileBrowse("Browse File", file_types=[("csv files",
00172 "*.csv")], key="-append_file-",
00173 target="-AppendingFile-")]
00174
00175     line9 = [sg.HSeparator()]
00176
00177     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00178
00179     layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8,
00180 line9, line10]
00181
00182     return layout

```

Here is the caller graph for this function:



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

The __SetValues function is used to set the values of the variables that are used in the __GetData function.
 The __SetValues function takes a dictionary as an argument, and then sets each variable based on what is passed into the dictionary. The keys for this dictionary are defined by the user when they create their own instance of this class.

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

Returns:
 A dictionary of values

Doc Author:
 Willem van der Schans, Trelent AI

Definition at line [175](#) of file [ConstructionMonitor/Core.py](#).

```

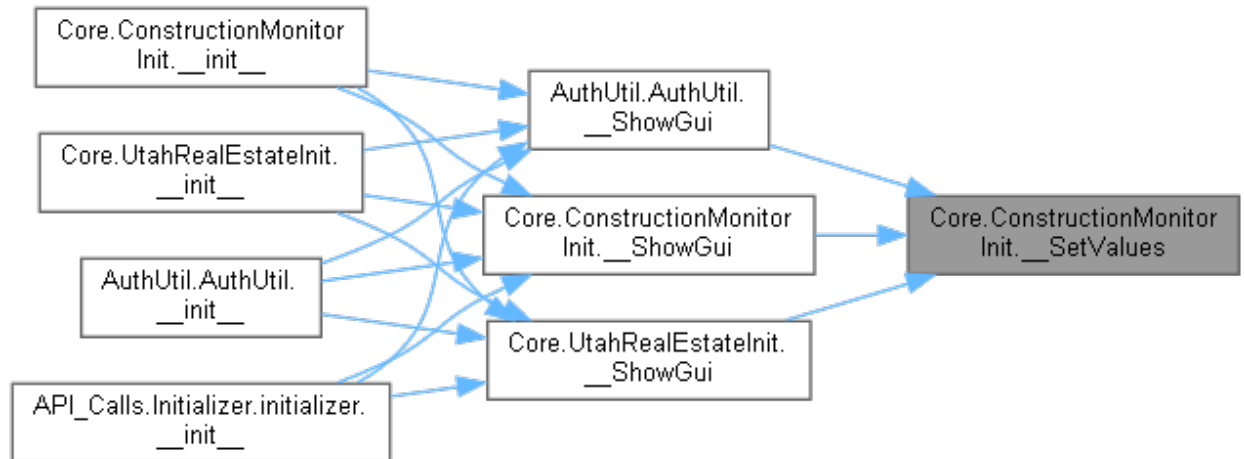
00175     def __SetValues(self, values):
00176
00177         """
00178         The __SetValues function is used to set the values of the variables that are
00179         used in the __GetData function.
00180         The __SetValues function takes a dictionary as an argument, and then sets
00181         each variable based on what is passed into
00182         the dictionary. The keys for this dictionary are defined by the user when
00183         they create their own instance of this class.
00184
00185         Args:
00186             self: Represent the instance of the class
00187             values: Pass in the values from the ui
00188
00189         Returns:
00190             A dictionary of values
00191
00192         Doc Author:
00193             Willem van der Schans, Trelent AI
00194         """
00195         self.size = 1000
00196
00197         if values["-Cal-"] != "":
00198             self.dateStart = values["-Cal-"]
00199         else:
00200             self.dateStart = (date.today() -
00201                             timedelta(days=14)).strftime("%Y-%m-%d")
00202
00203         if values["-EndCal-"] != "":
00204             self.dateEnd = values["-EndCal-"]
00205         else:
00206             self.dateEnd = date.today().strftime("%Y-%m-%d")
  
```

```

00203
00204     self.rest_domain =
00205     "https://api.constructionmonitor.com/v2/powersearch/"
00206
00207     self.SourceInclude = None
00208
00209     if values["-append_file-"] != "":
00210         self.append_file = str(values["-append_file-"])
00211     else:
00212         self.append_file = None
00213
00214     self.ui_flag = True
00215

```

Here is the caller graph for this function:



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

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

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

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI

Definition at line 77 of file [ConstructionMonitor/Core.py](#).

```

00077     def __ShowGui(self, layout, text):
00078
00079         """
00080         The __ShowGui function is the main function that creates and displays the
00081         GUI.
00082         It takes in a layout, which is a list of lists containing all the elements
00083         to be displayed on screen.
00084         The text parameter specifies what title should appear at the top of the window.
00085
00086         Args:
00087         self: Refer to the current instance of a class
00088         layout: Determine what the gui will look like
00089         text: Set the title of the window
00090
00091         Returns:
00092         A dictionary of values
00093
00094         Doc Author:

```

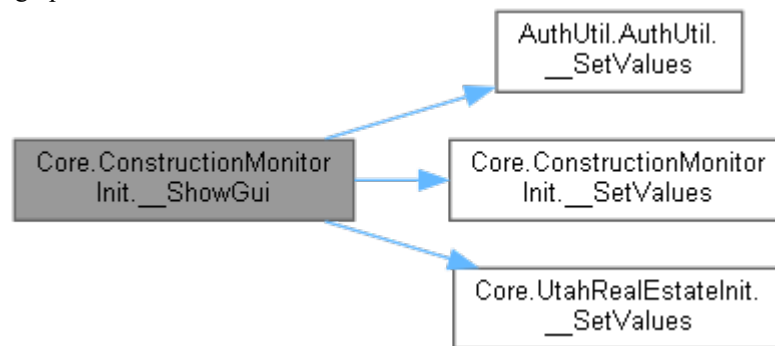


```

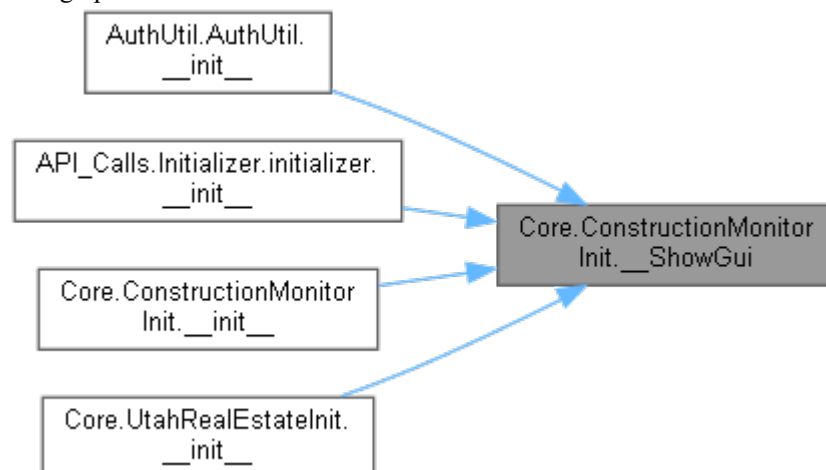
00093         Willem van der Schans, Trelent AI
00094         """
00095         window = sg.Window(text, layout, grab_anywhere=False,
00096         return_keyboard_events=True,
00097         finalize=True,
00098         icon=ImageLoader("taskbar_icon.ico"))
00099         while True:
00100             event, values = window.read()
00101
00102             if event == "Submit":
00103                 try:
00104                     self.__SetValues(values)
00105                     break
00106                 except Exception as e:
00107                     print(e)
00108                     RESError(993)
00109                     raise SystemExit(933)
00110             elif event == sg.WIN_CLOSED or event == "Quit":
00111                 break
00112
00113         window.close()
00114

```

Here is the call graph for this function:



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

[dataframe](#) Private 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](#).

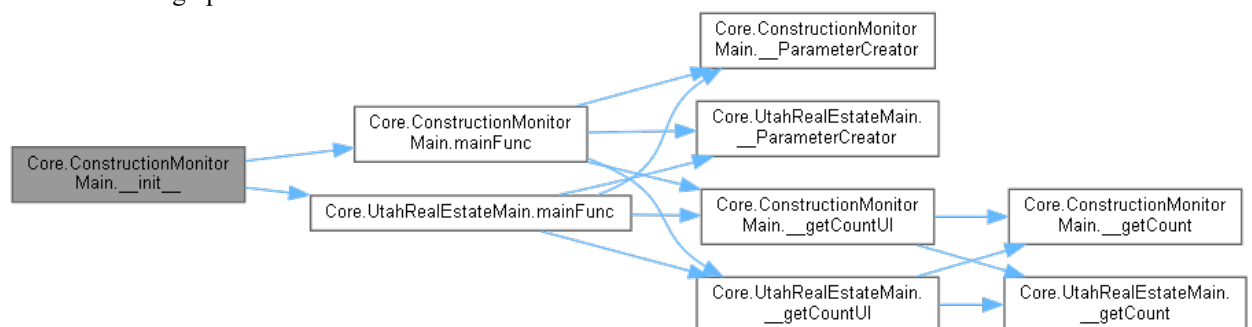
```
00218     def __init__(self, siteClass):
00219
00220         """
00221         The __init__ function is the first function that runs when an object of this
00222         class is created.
00223         It sets up all the variables and functions needed for this class to run
00224         properly.
```

```

00223
00224
00225     Args:
00226         self: Represent the instance of the class
00227         siteClass: Identify the site that is being used
00228
00229     Returns:
00230         Nothing
00231
00232     Doc Author:
00233         Willem van der Schans, Trelent AI
00234     """
00235     self.__siteClass = siteClass
00236     self.__restDomain = None
00237     self.__headerDict = None
00238     self.__columnSelection = None
00239     self.__appendFile = None
00240
00241     self.__parameterDict = {}
00242     self.__search_id = None
00243     self.__record_val = 0
00244     self.__batches = 0
00245
00246     self.__ui_flag = None
00247
00248     self.dataframe = None
00249
00250     try:
00251         self.mainFunc()
00252     except SystemError as e:
00253         if "Status Code = 1000 | Catastrophic Error" in str(getattr(e,
00254         'message', repr(e))):
00255             print(
00256                 f"ConstructionMonitor/Core.py | Error = {e} | Coerced
00257         SystemError in ConstructionMonitorMain class")
00258             pass
00259         except AttributeError as e:
00260             # This allows for user cancellation of the program using the quit
00261         button
00262             if "'NoneType' object has no attribute 'json'" in str(getattr(e,
00263         'message', repr(e))):
00264                 RESTError(1101)
00265                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00266         %H:%M:%S.%f')}[:-3]} | Error {e}")
00267                 pass
00268             elif e is not None:
00269                 print(
00270                     f"ConstructionMonitor/Core.py | Error = {e} |
00271         Authentication Error | Please update keys in AuthUtil")
00272                 RESTError(401)
00273                 print(e)
00274                 pass
00275             else:
00276                 pass
00277         except Exception as e:
00278             print(e)
00279             RESTError(1001)
00280             raise SystemExit(1001)
00281

```

Here is the call graph for this function:



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

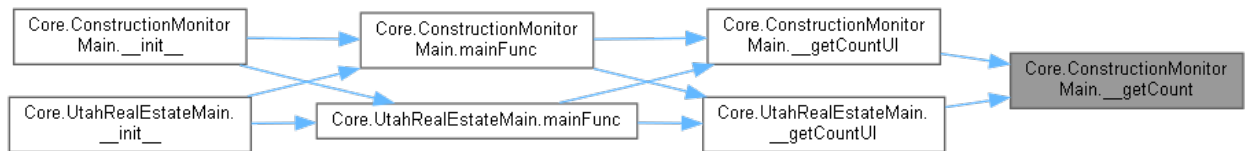
Returns:
The total number of records in the database

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

Here is the caller graph for this function:



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

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

Args:
self: Access the class variables and methods

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

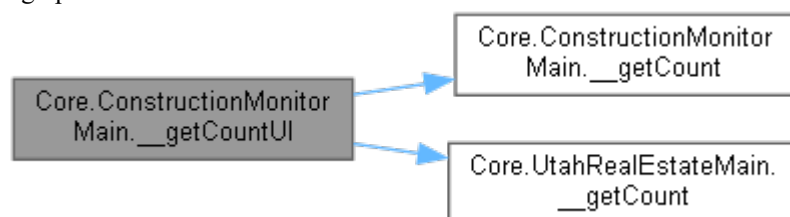
Doc Author:
Willem van der Schans, Trelent AI

Definition at line 405 of file [ConstructionMonitor/Core.py](#).

```

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

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

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

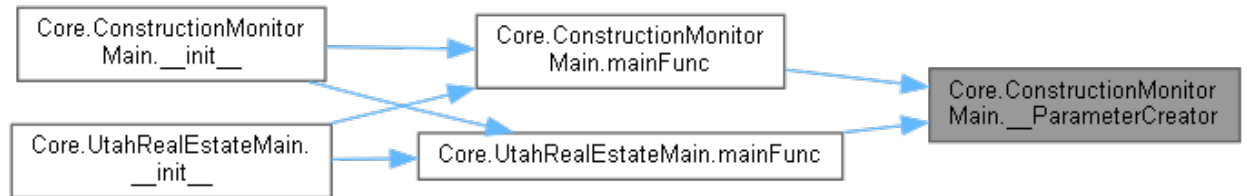
Doc Author:
Willem van der Schans, Trelent AI

Definition at line 317 of file [ConstructionMonitor/Core.py](#).

```

00317     def __ParameterCreator(self):
00318         """
00319         The __ParameterCreator function is used to create the parameter dictionary
00320         that will be passed into the
00321         __Request function. The function takes in a siteClass object and extracts
00322         all of its attributes, except for
00323         those that start with '__' or are callable. It then creates a dictionary
00324         from these attributes and stores it as
00325         self.__parameterDict.
00326
00327         Args:
00328         self: Make the function a method of the class
00329
00330         Returns:
00331         A dictionary of parameters and a list of non parameter variables
00332
00333         Doc Author:
00334         Willem van der Schans, Trelent AI
00335         """
00336         __Source_dict = {key: value for key, value in
00337         self.__siteClass.__dict__.items() if
00338         not key.startswith('__') and not callable(key)}
00339
00340         self.__restDomain = __Source_dict["rest_domain"]
00341         __Source_dict.pop("rest_domain")
00342         self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00343         __Source_dict.pop("auth_key")
00344         self.__columnSelection = __Source_dict["SourceInclude"]
00345         __Source_dict.pop("SourceInclude")
00346         self.__ui_flag = __Source_dict["ui_flag"]
00347         __Source_dict.pop("ui_flag")
00348         self.__appendFile = __Source_dict["append_file"]
00349         __Source_dict.pop("append_file")
00350
00351         temp_dict = copy.copy(__Source_dict)
00352         for key, value in temp_dict.items():
00353             if value is None:
00354                 __Source_dict.pop(key)
00355             else:
00356                 pass
00357
00358         self.__parameterDict = copy.copy(__Source_dict)
  
```

Here is the caller graph for this function:



def Core.ConstructionMonitorMain.mainFunc (self)

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

Args:
self: Refer to the current object

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI

Definition at line 276 of file [ConstructionMonitor/Core.py](#).

```

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

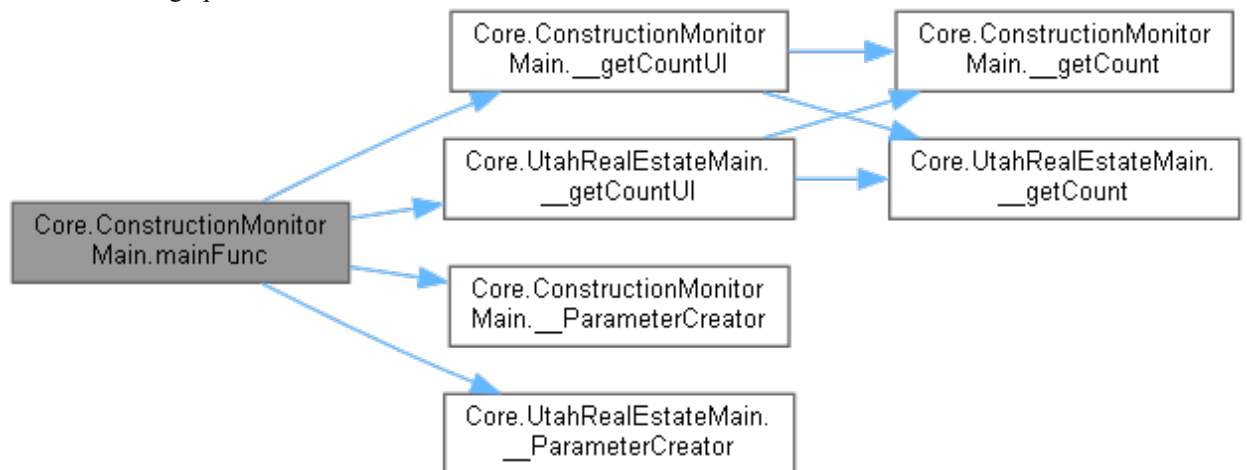


```

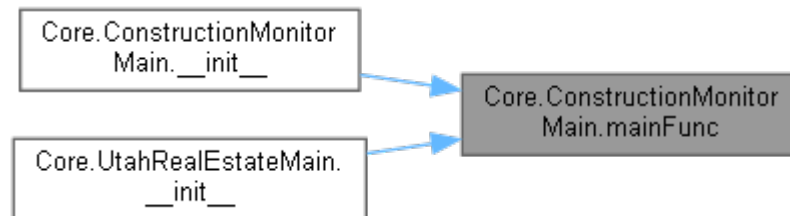
00306
ColumnSelection=self.__columnSelection,
00307                                     BatchesNum=self.__batches,
00308                                     Type="construction_monitor")
00309     BatchGuiObject.BatchGuiShow()
00310     self.dataframe = BatchGuiObject.dataframe
00311     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00312     FileSaver("cm", self.dataframe, self.__appendFile)
00313     else:
00314         RESTError(994)
00315         raise SystemExit(994)
00316

```

Here is the call graph for this function:



Here is the caller graph for this function:



Member Data Documentation

Core.ConstructionMonitorMain.__appendFile [private]

Definition at line 239 of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__batches [private]

Definition at line 244 of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__columnSelection [private]

Definition at line 238 of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__headerDict [private]

Definition at line [237](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__parameterDict[private]

Definition at line [241](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__record_val[private]

Definition at line [243](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__restDomain[private]

Definition at line [236](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__search_id[private]

Definition at line [242](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__siteClass[private]

Definition at line [235](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.__ui_flag[private]

Definition at line [246](#) of file [ConstructionMonitor/Core.py](#).

Core.ConstructionMonitorMain.dataframe

Definition at line [248](#) of file [ConstructionMonitor/Core.py](#).

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

- [ConstructionMonitor/Core.py](#)

DataTransfer.DataTransfer Class Reference

Public Member Functions

- def [__init__](#) (self)
- def [setValue](#) (self, value)
- def [getValue](#) (self)
- def [whileValue](#) (self)

Private Attributes

[__value](#)

Detailed Description

Definition at line [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](#).

```
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
```

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](#).

```

00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054

```

Here is the caller graph for this function:



def DataTransfer.DataTransfer.setValue (self, value)

```

The setValue function sets the value of the object.

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

Returns:
The value that was passed to it

Doc Author:
Willem van der Schans, Trelent AI

```

Definition at line [22](#) of file [DataTransfer.py](#).

```

00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026
00027         Args:
00028             self: Represent the instance of the class
00029             value: Set the value of the instance variable __value
00030
00031         Returns:
00032             The value that was passed to it
00033
00034         Doc Author:
00035             Willem van der Schans, Trelent AI
00036         """
00037         self.__value = value
00038

```

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

Here is the call graph for this function:



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
00019         be used by other functions in this class.
00020         The __init__ function takes two arguments: self and method. The first
00021         argument, self, refers to an instance of a
00022         class (in this case it's an instance of DataFrameSaver). The second argument,
00023         method refers to a string value that
00024         is passed into DataFrameSaver when it's instantiated.
00025
00026     Args:
```

```

00024         self: Represent the instance of the class
00025         method: Determine which dataframe to append the new data to
00026         outputDF: Pass in the dataframe that will be saved to a csv file
00027         AppendingPath: Specify the path to an existing csv file that you want
to append your dataframe to
00028
00029         Returns:
00030             Nothing
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034         """
00035         self.docPath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00036             datetime.datetime.today().strftime('%m%d%Y'))
00037         self.data = outputDF
00038         self.dataAppending = None
00039         self.appendFlag = True
00040         self.fileName =
f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00041         self.uiFlag = True
00042
00043         if method.lower() == "ure":
00044             self.primaryKey = "ListingKeyNumeric"
00045         elif method.lower() == "cm":
00046             self.primaryKey = "id"
00047         elif "realtor" in method.lower():
00048             self.primaryKey = None
00049             self.uiFlag = False
00050         elif method.lower() == "cfbp":
00051             self.primaryKey = None
00052             self.uiFlag = False
00053         else:
00054             raise ValueError("method input is invalid choice one of 4 options:
URE, CM, Realtor, CFBP")
00055
00056         if AppendingPath is None:
00057             self.appendFlag = False
00058         else:
00059             self.dataAppending = pd.read_csv(AppendingPath)
00060
00061         if self.appendFlag:
00062             if self.primaryKey is not None:
00063                 # Due to low_memory loading the columns are not typed properly,
00064                 # since we are comparing this will be an issue since we need to
do type comparisons,
00065                 # so here we coerce the types of the primary keys to numeric.
00066                 # If another primary key is ever chosen make sure to core to the
right data type.
00067                 self.dataAppending[self.primaryKey] =
pd.to_numeric(self.dataAppending[self.primaryKey])
00068                 self.data[self.primaryKey] =
pd.to_numeric(self.data[self.primaryKey])
00069
00070                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00071 keep="last")
00072             else:
00073                 self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(keep="last")
00074             else:
00075                 self.outputFrame = self.data
00076
00077             if os.path.exists(self.docPath):
00078                 self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00079             else:
00080                 os.mkdir(self.docPath)
00081                 self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00082
00083             if self.uiFlag:
00084                 if self.appendFlag:
00085                     PopupWrapped(text=f"File Appended and Saved to
{self.docPath.joinpath(self.fileName)}",
00086 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, Trellent 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, Trellent 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

[classObj](#)Private 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

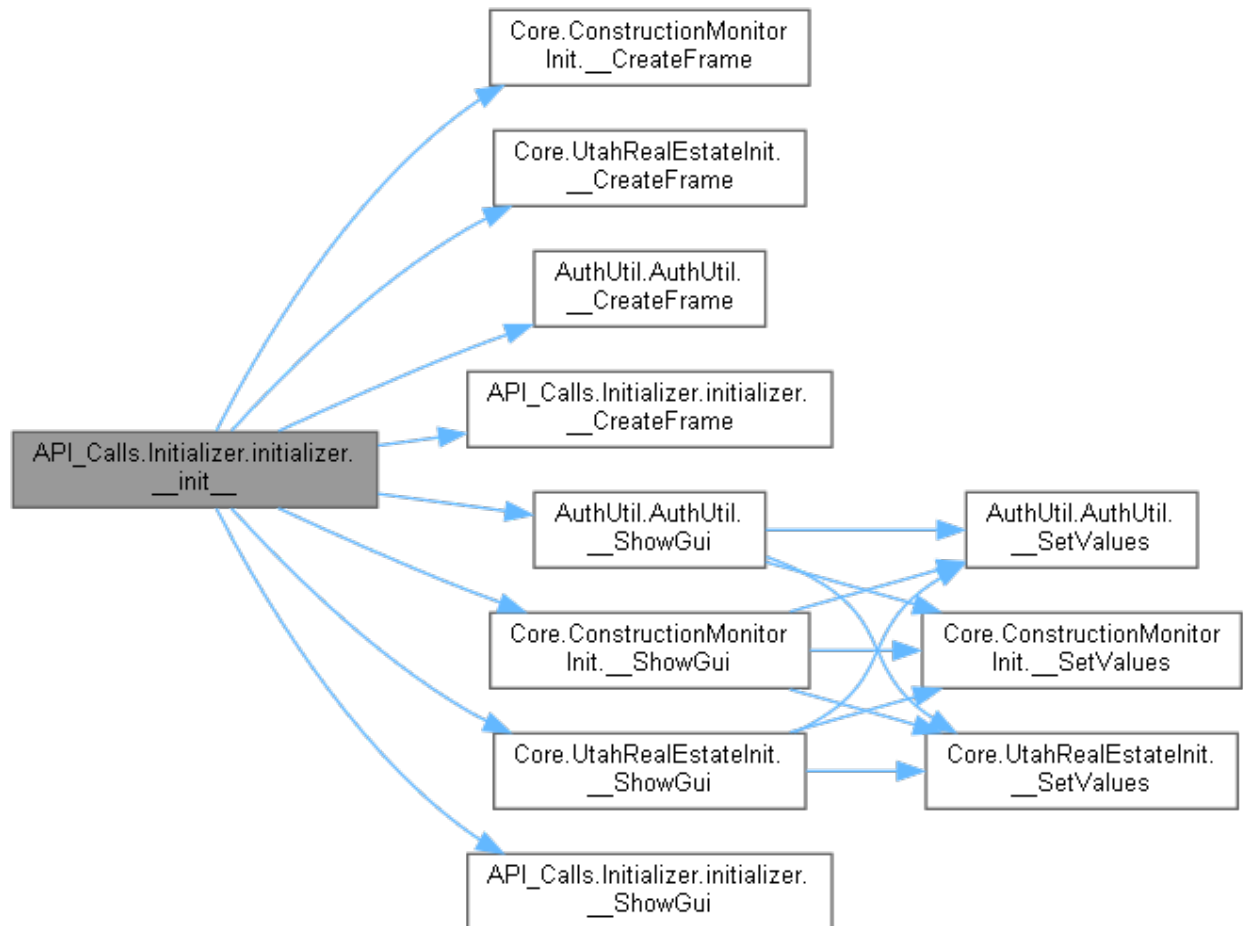
Doc Author:
Willem van der Schans, Trelent AI

Definition at line [23](#) of file [Initializer.py](#).

```
00023     def __init__(self):
00024
00025         """
00026         The __init__ function is called when the class is instantiated.
00027         It sets up the logging, calls the __ShowGui function to create and display
00028         the GUI, and then calls __CreateFrame to create a frame for displaying
00029         widgets.
00030
00031         Args:
00032             self: Represent the instance of the class
00033
00034         Returns:
00035             Nothing
00036
00037         Doc Author:
00038             Willem van der Schans, Trelent AI
00039         """
00040         self.classObj = None
00041
00042         logger()
00043
00044         print("\n\n-----Initiate Program-----\n\n")
00045
00046         self.__ShowGui(self.__CreateFrame(), "Data Tool")
00047
```

```
00048         print("\n\n-----Closing Program-----\n\n")
00049
```

Here is the call graph for this function:



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](#).

```

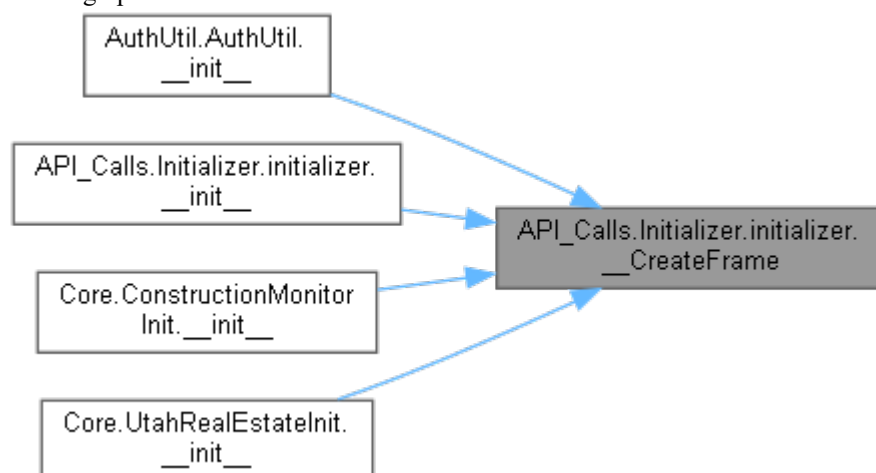
00121     def __CreateFrame(self):
00122
00123         """
00124         The __CreateFrame function is a helper function that creates the layout for
00125         the main window. It returns a list of lists, which is then passed to sg.Window() as its layout
00126         parameter.
00127         Args:
  
```

```

00128         self: Represent the instance of the class
00129
00130     Returns:
00131         A list of lists, which is then passed to the sg
00132
00133     Doc Author:
00134         Willem van der Schans, Trelent AI
00135     """
00136     sg.theme('Default1')
00137
00138     line0 = [sg.HSeparator()]
00139
00140     line1 = [sg.Image(ImageLoader("logo.png")),
00141             sg.Push(),
00142             sg.Text("Gardner Data Utility", font=("Helvetica", 12,
00143 "bold"), justification="center"),
00143             sg.Push(),
00144             sg.Push()]
00145
00146     line3 = [sg.HSeparator()]
00147
00148     line4 = [sg.Push(),
00149             sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
00150 justification="center"),
00151             sg.Push()]
00152
00153     line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20,
00154 None)), sg.Push(),
00155             sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00156
00157     line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)),
00158 sg.Push(), sg.Button("Census", size=(20, None)),
00159             sg.Push()]]
00160
00161     line8 = [sg.HSeparator()]
00162
00163     line9 = [sg.Push(),
00164             sg.Text("Utilities", font=("Helvetica", 10, "bold"),
00165 justification="center"),
00166             sg.Push()]
00167
00168     line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20,
00169 None)),
00170             sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00171
00172     line11 = [sg.HSeparator()]
00173
00174     layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
00175 line11]
00176
00177     return layout

```

Here is the caller graph for this function:



```
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., `"Text"`, `"InputText"`, 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
00055         containing a tuple with three elements:
00056             1) The type of element to be displayed (e.g., "Text",
00057             "InputText", etc.)
00058             2) A dictionary containing any additional parameters for that element
00059             (e.g., size, default value, etc.)
00060             3) An optional key name for the element (used in event handling). If no
00061             key name is provided then one will be generated automatically by PySimpleGUIQt based
00062             on its position in the layout list
00063
00064         Args:
00065             self: Represent the instance of the class
00066             layout: Pass the layout of the window to be created
00067             text: Set the title of the window
00068
00069         Returns:
00070             A window object
00071
00072         Doc Author:
00073             Willem van der Schans, Trelent AI
00074         """
00075         window = sg.Window(text, layout, grab_anywhere=False,
00076         return_keyboard_events=True,
00077                             finalize=True,
00078                             icon=ImageLoader("taskbar_icon.ico"))
00079
00080         while True:
00081             event, values = window.read()
00082
00083             if event == "Construction Monitor":
00084                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
00085 %H:%M:%S.%f')}[:-3]} | -----Initiating Construction Monitor API
00086 Call-----")
00087                 ConstructionMonitorMain(ConstructionMonitorInit())
00088                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
00089 %H:%M:%S.%f')}[:-3]} | -----Closing Construction Monitor API
00090 Call-----\n")
00091             elif event == "Utah Real Estate":
00092                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
00093 %H:%M:%S.%f')}[:-3]} | -----Initiating Utah Real Estate API
00094 Call-----")
00095                 UtahRealEstateMain(UtahRealEstateInit())
00096                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
00097 %H:%M:%S.%f')}[:-3]} | -----Closing Utah Real Estate API
00098 Call-----\n")
00099             elif event == "Realtor.Com":
00100                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
00101 %H:%M:%S.%f')}[:-3]} | -----Initiating Realtor.com API Call-----")

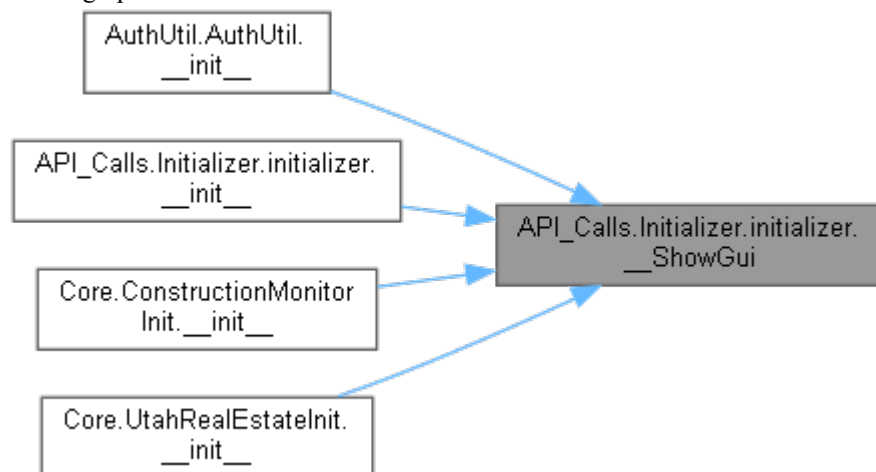
```

```

00087         realtorCom()
00088         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Realtor.com API
Call-----\n")
00089         elif event == "Census":
00090             print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Census API Call-----")
00091             Census()
00092             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Census API Call-----\n")
00093             elif event == "Authorization Utility":
00094                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Authorization
Utility-----")
00095                 AuthUtil()
00096                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Authorization
Utility-----\n")
00097                 elif event == "Open Data Folder":
00098                     print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Data Folder Opened-----")
00099                     try:
00100                         os.system(f"start
{Path(os.path.expanduser('~/.Documents')).joinpath('GardnerUtilData')}")
00101                     except:
00102                         try:
00103                             os.system(f"start
{Path(os.path.expanduser('~/.Documents'))}")
00104                         except Exception as e:
00105                             print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
00106                             PopupWrapped(
00107                                 text="Documents folder not found. Please create a
Windows recognized documents folder",
00108                                 windowType="errorLarge")
00109
00110                 elif event in ('Exit', None):
00111                     try:
00112                         break
00113                     except Exception as e:
00114                         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.")
00115                         break
00116                 elif event == sg.WIN_CLOSED or event == "Quit":
00117                     break
00118
00119             window.close()
00120

```

Here is the caller graph for this function:



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
```

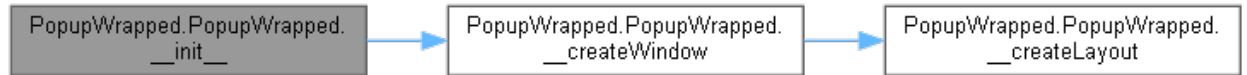


```

00028     """
00029         self.__text = text
00030         self.__type = windowType
00031         self.__error = error
00032         self.__layout = []
00033         self.__windowObj = None
00034         self.__thread = None
00035         self.__counter = 0
00036
00037         self.__createWindow()
00038

```

Here is the call graph for this function:



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](#).

```

00039     def __createLayout(self):
00040         """
00041         The __createLayout function is used to create the layout of the window.
00042         The function takes class variables and returns a window layout.
00043         It uses a series of if statements to determine what type of window it is,
00044         then creates a layout based on that information.
00045         Args:
00046             self: Refer to the current instance of a class
00047         Returns:
00048             A list of lists
00049         Doc Author:
00050             Willem van der Schans, Trelent AI
00051         """
00052         sg.theme('Default1')
00053         __Line1 = None
00054         __Line2 = None
00055         if self.__type == "notice":
00056             __Line1 = [sg.Push(),
00057                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00058                               justification="center"),
00059                        sg.Text(self.__text, justification="center",
00060                               key="-textField-"), sg.Push()]
00061             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00062         elif self.__type == "noticeLarge":
00063             __Line1 = [sg.Push(),
00064                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00065                               justification="center"),
00066                        sg.Text(self.__text, justification="center",
00067                               key="-textField-"), sg.Push()]
00068             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00069         elif self.__type == "errorLarge":
00070             __Line1 = [sg.Push(),
00071                        sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
00072                               justification="center"),
00073                        sg.Text(self.__text, justification="center",
00074                               key="-textField-"), sg.Push()]
00075             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]

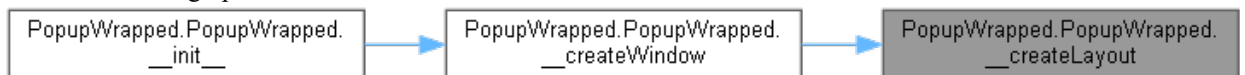
```

```

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:



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 "Ok" 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,

```

```

00105                                     modal=True,
00106                                     keep_on_top=True,
00107                                     disable_close=False,
00108
00109 icon=ImageLoader("taskbar_icon.ico"),                                     size=(290, 50))
00110         elif self.__type == "noticeLarge":
00111             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
00112 finalize=True,
00113                                     modal=True,
00114                                     keep_on_top=True,
00115                                     disable_close=False,
00116
00117 icon=ImageLoader("taskbar_icon.ico"))
00118         elif self.__type == "errorLarge":
00119             self.__windowObj = sg.Window(title="Error", layout=self.__layout,
00120 finalize=True,
00121                                     modal=True,
00122                                     keep_on_top=True,
00123                                     disable_close=False,
00124
00125 icon=ImageLoader("taskbar icon.ico"))
00126         elif self.__type == "FatalErrorLarge":
00127             self.__windowObj = sg.Window(title="Fatal Error",
00128 layout=self.__layout, finalize=True,
00129                                     modal=True,
00130                                     keep_on_top=True,
00131                                     disable_close=False,
00132
00133 icon=ImageLoader("taskbar_icon.ico"),
00134                                     size=(290, 80))
00135
00136         if self.__type != "progress" or self.__type.startswith("perm"):
00137             timer = 0
00138             while timer < 100:
00139                 event, values = self.__windowObj.read()
00140                 if event == "Ok" or event == sg.WIN_CLOSED:
00141                     break
00142
00143                 time.sleep(0.1)
00144
00145             if self.__type == "FatalErrorLarge":
00146                 try:
00147                     os.system(
00148                         f"start
00149 {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
00150                 except Exception as e:
00151                     print(
00152                         f"PopupWrapped.py | Error = {e} | Log Folder not found
00153 please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")
00154
00155             self.__windowObj.close()

```

Here is the call graph for this function:



Here is the caller graph for this function:



def PopupWrapped.PopupWrapped.stopWindow (self)

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

Definition at line [155](#) of file [PopupWrapped.py](#).

```
00155     def stopWindow(self):
00156         """
00157         The stopWindow function is used to close the window object that was created
00158         in the startWindow function.
00159         This is done by calling the close() method on self.__windowObj, which will
00160         cause it to be destroyed.
00161         Args:
00162         self: Represent the instance of the class
00163         Returns:
00164         The window object
00165         Doc Author:
00166         Willem van der Schans, Trelent AI
00167         """
00168         self.__windowObj.close()
00169     
```

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](#).

```
00168     def textUpdate(self, sleep=0.5):
00169         """
00170         The textUpdate function is a function that updates the text in the text field.
00171         It does this by adding dots to the end of it, and then removing them. This
00172         creates
00173         a loading effect for when something is being processed.
00174         Args:
00175         self: Refer to the object itself
00176         sleep: Control the speed of the text update
00177         Returns:
00178         A string that is the current text of the text field
00179         Doc Author:
00180         Willem van der Schans, Trelent AI
00181         """
00182         self.__counter += 1
00183         if self.__counter == 4:
00184             self.__counter = 1
00185         newString = ""
00186         if self.__type == "notice":
00187             pass
00188         elif self.__type == "error":
00189             pass
00190         elif self.__type == "progress":
00191             newString = f"{self.__text}{'.' * self.__counter}"
00192         self.__windowObj.write_event_value('update-textField-', newString)
00193     
```

```
00193         time.sleep(sleep)
00194
```

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](#).

```
00195     def windowPush(self):
00196
00197         """
00198         The windowPush function is used to update the values of a window object.
00199         The function takes in an event and values from the window object, then
00200         checks if the event starts with 'update'.
00201         If it does, it will take everything after 'update' as a key for updating
00202         that specific value.
00203         It will then update that value using its key and refresh the window.
00204     Args:
00205         self: Reference the object that is calling the function
00206     Returns:
00207         A tuple containing the event and values
00208     Doc Author:
00209         Willem van der Schans, Trelent AI
00210     """
00211     event, values = self.__windowObj.read()
00212     if event.startswith('update'):
00213         __key_to_update = event[len('update'):]
00214         self.__windowObj[__key_to_update].update(values[event])
00215         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](#).

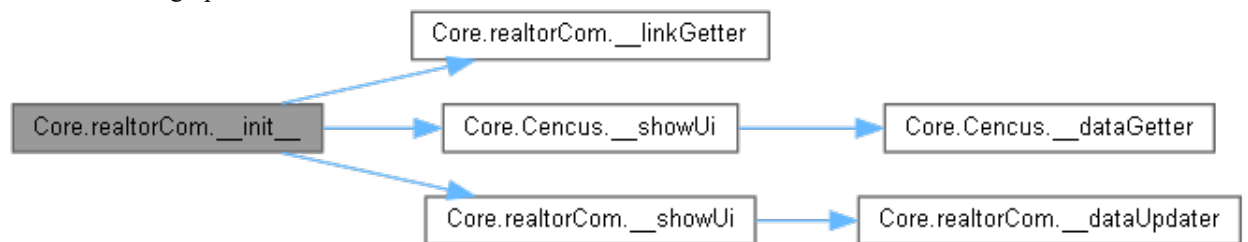
```
00014     def __init__(self):
00015         """
00016         The __init__ function is called when the class is instantiated.
00017         It sets up the initial state of an object, and it's where you put code that
00018         needs to run before anything else in your class.
00019         Args: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

```

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

```

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](#).

```

00101     def __dataUpdater(self):
00102
00103         """
00104         The dataUpdater function is a private function that updates the dataframes
for each of the three
00105         types of realtor data. It takes class variables and return the path to
the saved file. The function first creates an empty
00106         dictionary called tempdf, then iterates through each key in self.__idDict
(which contains all three ids).
00107         For each key, it reads in a csv file from the link associated with that
id and saves it to tempdf as a pandas

```



```

00108         DataFrame object. Then, depending on which type of realtor data we are
00109         dealing with (State/County/Zip), we save
00110
00111     Args:
00112         self: Access the attributes and methods of the class
00113
00114     Returns:
00115         The path of the saved file
00116
00117     Doc Author:
00118         Willem van der Schans, Trelent AI
00119     """
00120     for key, value in self.__idDict.items():
00121         tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00122
00123         if key == "State":
00124             self.dfState = tempdf
00125         elif key == "County":
00126             self.dfCounty = tempdf
00127         elif key == "Zip":
00128             self.dfZip = tempdf
00129
00130         FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00131         self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()}
00132         \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
00078         dictionary and adds
00079         a link to each entry in the dictionary. The link is used to access historical
00080         data for each
00081         scope symbol.
00082
00083     Args:
00084         self: Refer to the object itself
00085
00086     Returns:
00087         A dictionary of all the links to the history pages
00088
00089     Doc Author:
00090         Willem van der Schans, Trelent AI
00091     """
00092     for key, value in self.__idDict.items():
00093         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](#).

```

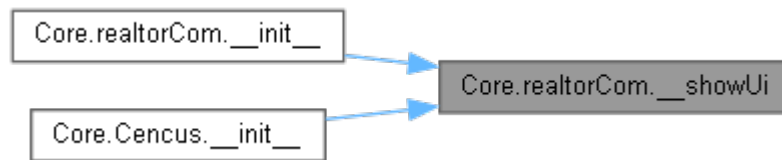
00046     def __showUi(self):
00047
00048         """
00049         The __showUi function is a helper function that creates and displays the
progress window.
00050         It also starts the dataUpdater thread, which will update the progress bar
as it runs.
00051
00052
00053         Args:
00054             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](#).

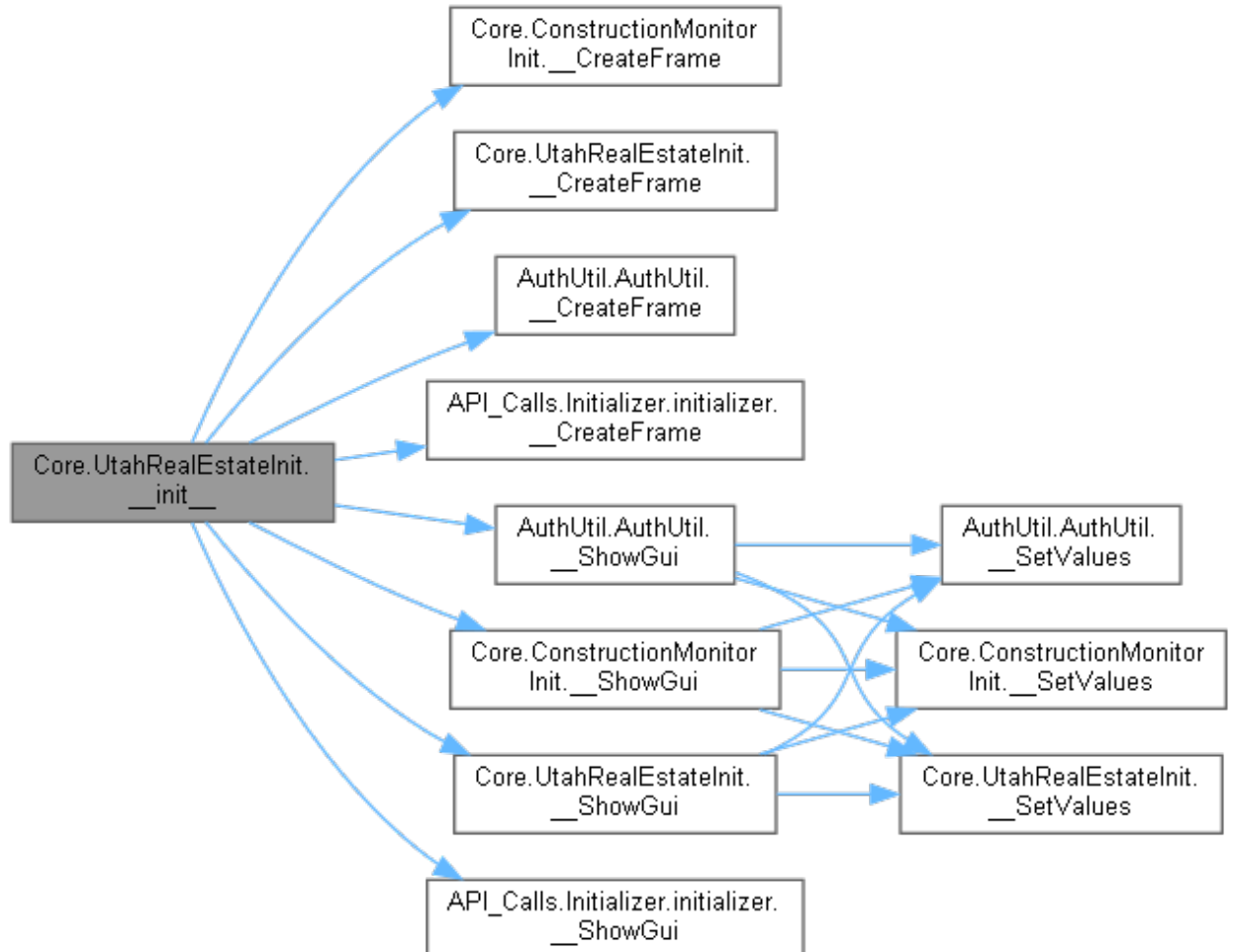
```
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the initial state of the object.
00031
00032
00033         Args:
00034             self: Represent the instance of the class
00035
00036   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

```

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

```

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](#).

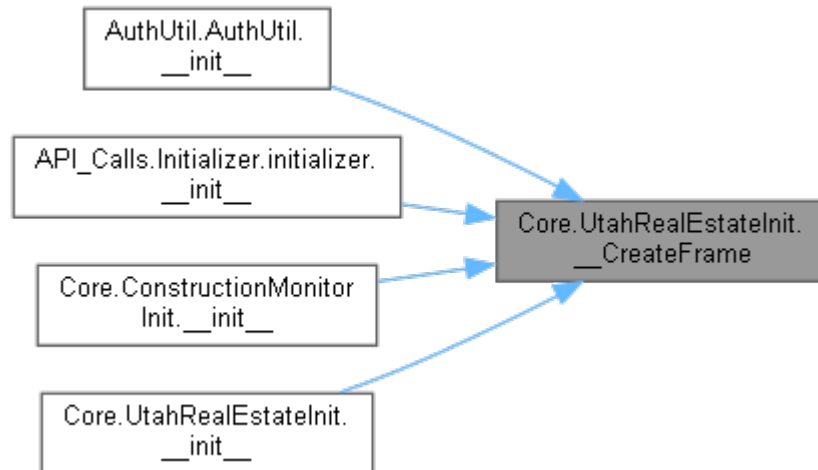
```
00093     def CreateFrame():
00094         """
00095         The CreateFrame function creates the GUI layout for the application.
00096         The function returns a list of lists that contains all the elements to
00097         be displayed in the window.
00098         Each element is defined by its type and any additional parameters needed
00099         to define it.
00100
00101         Args:
00102             Returns:
00103                 A list of lists, which is used to create the gui
00104
00105         Doc Author:
00106             Willem van der Schans, Trelent AI
00107         """
00108         sg.theme('Default1')
00109
00110         line00 = [sg.HSeparator()]
00111
00112         line0 = [sg.Image(ImageLoader("logo.png")),
00113                  sg.Push(),
00114                  sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
00115 "bold"), justification="center"),
00116                  sg.Push(),
00117                  sg.Push()]
00118
00119         line1 = [sg.HSeparator()]
00120
00121         line2 = [sg.Text("MLS Status : ", size=(15, None),
00122 justification="Right"),
00123                  sg.DropDown(default_value="Active", values=["Active",
00124 "Closed"], key="-status-", size=(31, 1))]
00125
00126         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00127                  sg.DropDown(default_value="Listing Date", values=["Listing
00128 Date", "Modification Date", "Close Date"],
00129 key="-type-", size=(31, 1))]
00130
00131         line4 = [sg.Text("Start Date : ", size=(15, None),
00132 justification="Right"),
00133                  sg.Input(default_text=(date.today() -
00134 timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00135 disabled=False, size=(20, 1)),
00136                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
00137 key="-start_date-", target="-DateStart-")]
00138
00139         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00140                  sg.Input(default_text=(date.today()).strftime("%Y-%m-%d"),
00141 key="-DateEnd-", disabled=False,
00142 size=(20, 1)),
00143                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
00144 key="-end_date-", target="-DateEnd-")]
00145
00146         line6 = [[sg.Text("Column Sub-Selection : ", size=(23, None),
00147 justification="Right"),
00148                  sg.Checkbox(text="", default=True, key="-selectionFlag-",
00149 size=(15, 1)),
00150                  sg.Push()]]
00151
00152         line7 = [sg.HSeparator()]
00153
00154         line8 = [sg.Push(),
00155                  sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00156 justification="center"),
00157                  sg.Push()]
00158
00159         line9 = [sg.HSeparator()]
00160
00161         line10 = [sg.Text("Appending File : ", size=(15, None),
00162 justification="Right"),
00163                  sg.Input(default_text="", key="-AppendingFile-",
00164 disabled=True,
00165 size=(20, 1)),
```

```

00151         sg.FileBrowse("Browse File", file_types=[("csv files",
00152             "*.csv")], key='-append_file-',
00153             target="-AppendingFile-")
00154         line11 = [sg.HSeparator()]
00155         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00156         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
00157             line8, line9, line10, line11,
00158             line12]
00159         return layout
00160
00161
00162

```

Here is the caller graph for this function:



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
00167         used in the __GetData function. The values are passed from a dictionary called
00168         'values' which is created
00169         by parsing through an XML file using ElementTree. This function also sets
00170         default values for
00171         some of these variables if they were not specified in the XML file.
00172
00173         Args:
00174             self: Represent the instance of the class
00175             values: Pass the values from the gui to this function
00176
00177         Returns:

```

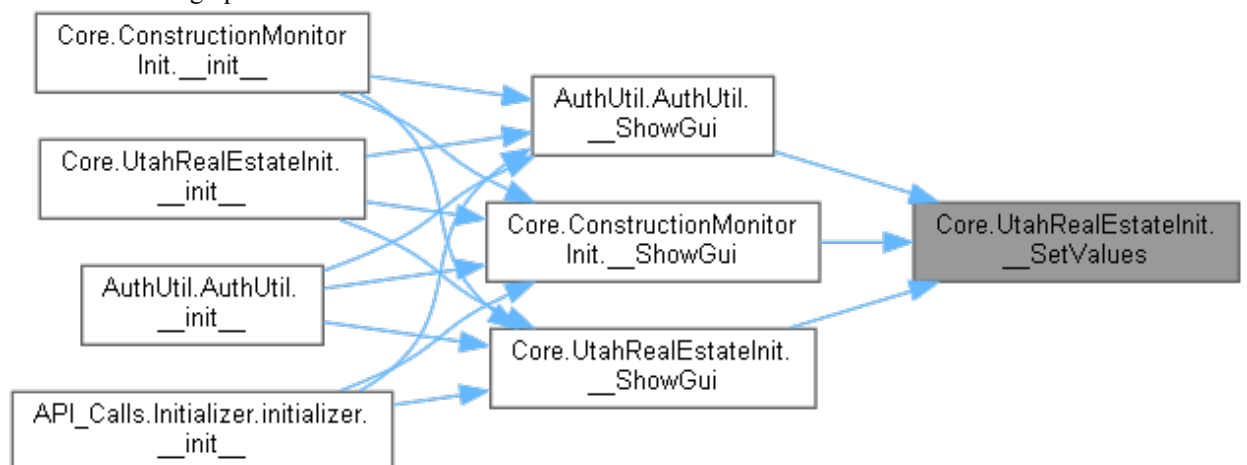


```

00176         A dictionary with the following keys:
00177
00178     Doc Author:
00179         Willem van der Schans, Trelent AI
00180     """
00181         self.StandardStatus = values["-status-"]
00182
00183         self.ListedOrModified = values["-type-"]
00184
00185         if values["-DateStart-"] != "":
00186             self.dateStart = values["-DateStart-"]
00187         else:
00188             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00189
00190         if values["-DateEnd-"] != "":
00191             self.dateEnd = values["-DateEnd-"]
00192         else:
00193             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00194
00195         if values['-selectionFlag-']:
00196             self.select =
"ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,Subdi
visionName," \
00197 "StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDa
te," \
00198 "ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,L
otSizeAcres," \
00199 "LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,
ParkingTotal," \
00200 "BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffect
ive," \
00201 "OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseCont
ractDate," \
00202 "AssociationFee,AssociationFeeFrequency,OccupantType,PropertySubType,PropertyType,
" \
00203             "StandardStatus,BuyerFinancing"
00204         else:
00205             self.select = None
00206
00207         if values["-append_file-"] != "":
00208             self.append_file = str(values["-append_file-"])
00209         else:
00210             self.append_file = None
00211
00212

```

Here is the caller graph for this function:



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

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

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

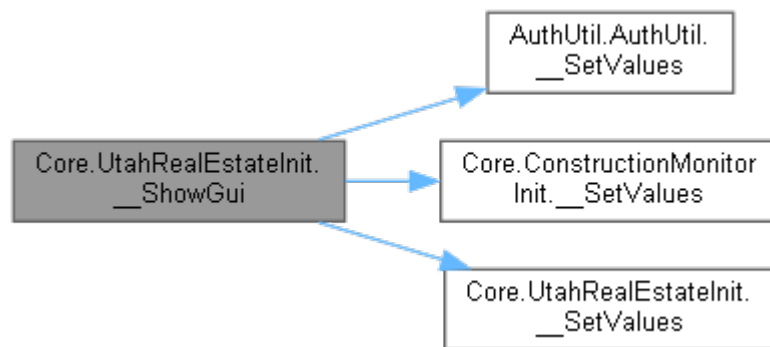
Returns:
 A dictionary of values

Doc Author:
 Willem van der Schans, Trelent AI

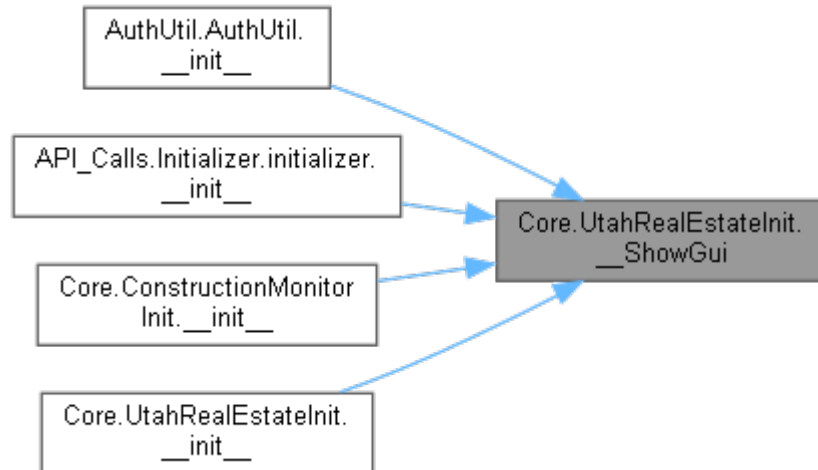
Definition at line 52 of file [UtahRealEstate/Core.py](#).

```
00052     def __ShowGui(self, layout, text):
00053
00054         """
00055         The __ShowGui function is a helper function that creates the GUI window and
00056         displays it to the user.
00057         It takes in two parameters: layout, which is a list of lists containing all
00058         the elements for each row;
00059         and text, which is a string containing what will be displayed as the title
00060         of the window. The __ShowGui
00061         method then uses these parameters to create an instance of sg.Window with
00062         all its attributes set accordingly.
00063
00064         Args:
00065         self: Refer to the current class instance
00066         layout: Pass the layout of the window to be created
00067         text: Set the title of the window
00068
00069         Returns:
00070         A dictionary of values
00071
00072         Doc Author:
00073         Willem van der Schans, Trelent AI
00074         """
00075         window = sg.Window(text, layout, grab_anywhere=False,
00076         return_keyboard_events=True,
00077         finalize=True,
00078         icon=ImageLoader("taskbar_icon.ico"))
00079
00080         while True:
00081             event, values = window.read()
00082
00083             if event == "Submit":
00084                 try:
00085                     self.__SetValues(values)
00086                     break
00087                 except Exception as e:
00088                     print(e)
00089                     RESTError(993)
00090                     raise SystemExit(993)
00091             elif event == sg.WIN_CLOSED or event == "Quit":
00092                 break
00093
00094         window.close()
00095
```

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](#).

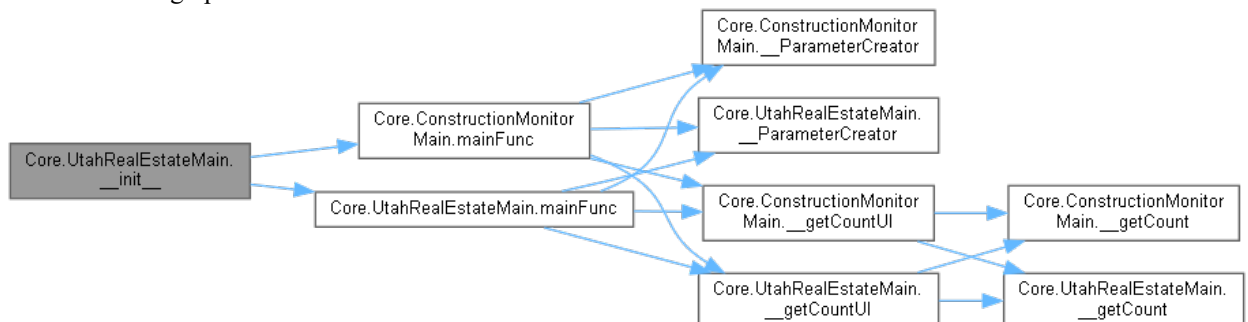
```
00215     def __init__(self, siteClass):
00216
00217         """
00218         The __init__ function is the first function that runs when an object of this
class is created.
```

```

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

```

Here is the call graph for this function:



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

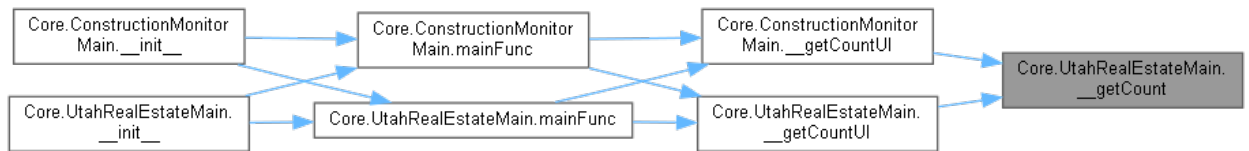
Returns:
The number of records in the data set

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
00374         be returned by the query.
00375         This function is called when a user calls the count() method on a ReST object.
00376         The __getCount function uses
00377         the $count parameter in OData to return only an integer value representing
00378         how many records would be returned
00379         by the query.
00380
00381         Args:
00382             self: Represent the instance of the class
00383
00384         Returns:
00385             The number of records in the data set
00386
00387         Doc Author:
00388             Willem van der Schans, Trelent AI
00389         """
00390         __count_resp = None
00391         try:
00392             __count_resp =
00393             requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00394                         headers=self.__headerDict)
00395
00396             if __count_resp.status_code != 200:
00397                 RESTError(__count_resp)
00398                 raise SystemExit(0)
00399
00400             self.__record_val = int(__count_resp.json()["@odata.count"])
00401
00402         except requests.exceptions.Timeout as e:
00403             print(e)
00404             RESTError(790)
00405             raise SystemExit(790)
00406         except requests.exceptions.TooManyRedirects as e:
00407             print(e)
00408             RESTError(791)
00409             raise SystemExit(791)
00410         except requests.exceptions.MissingSchema as e:
00411             print(e)
00412             RESTError(1101)
00413         except requests.exceptions.RequestException as e:
00414             print(e)
00415             RESTError(405)
00416             raise SystemExit(405)
```

Here is the caller graph for this function:



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

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

Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

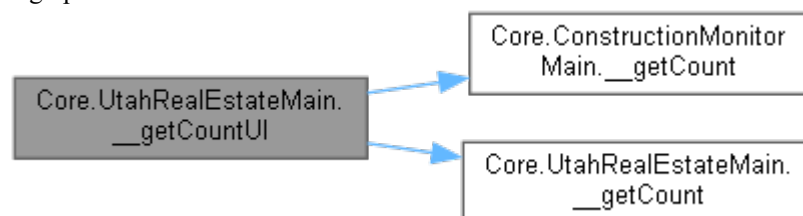
Doc Author:
Willem van der Schans, Trelent AI

Definition at line 415 of file [UtahRealEstate/Core.py](#).

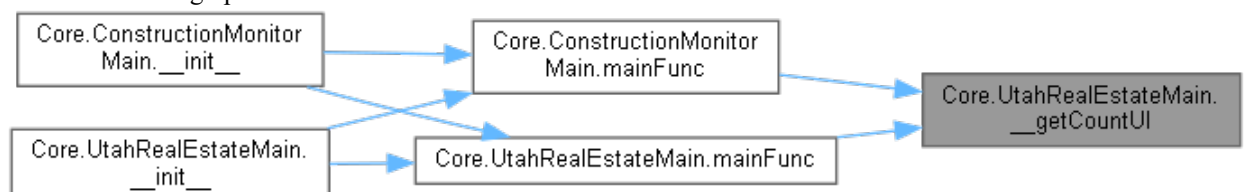
```

00415     def __getCountUI(self):
00416
00417         """
00418         The __getCountUI function is a wrapper for the __getCount function.
00419         It creates a progress window and updates it while the __getCount function
00420         runs.
00421         The purpose of this is to keep the GUI responsive while running long processes.
00422
00423         Args:
00424             self: Represent the instance of the class
00425
00426         Returns:
00427             A popupwrapped object
00428
00429         Doc Author:
00430             Willem van der Schans, Trelent AI
00431         """
00432         uiObj = PopupWrapped(text="Batch request running",
00433                               windowType="progress", error=None)
00434
00435         threadGui = threading.Thread(target=self.__getCount,
00436                                     daemon=False)
00437         threadGui.start()
00438
00439         while threadGui.is_alive():
00440             uiObj.textUpdate()
00441             uiObj.windowPush()
00442         else:
00443             uiObj.stopWindow()
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

Args:
self: Bind the object to the class

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

Doc Author:
Willem van der Schans, Trelent AI

Definition at line [327](#) of file [UtahRealEstate/Core.py](#).

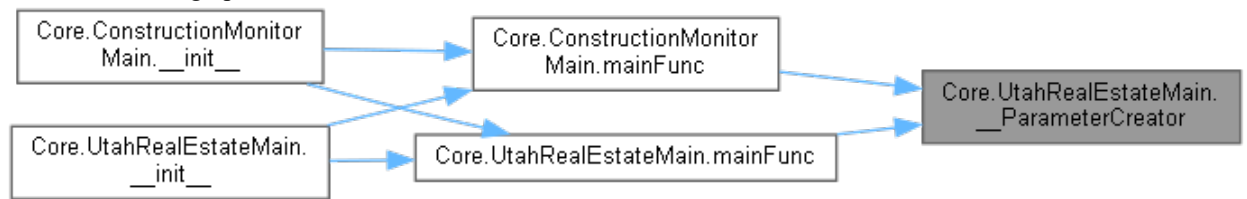
```

00327     def __ParameterCreator(self):
00328         """
00329         The __ParameterCreator function is used to create the filter string for the
00330         ReST API call.
00331         The function takes in a siteClass object and extracts all of its parameters
00332         into a dictionary.
00333         It then creates an appropriate filter string based on those parameters.
00334
00335         Args:
00336             self: Bind the object to the class
00337
00338         Returns:
00339             A string to be used as the parameter in the api call
00340
00341         Doc Author:
00342             Willem van der Schans, Trelent AI
00343         """
00344         filter_string = ""
00345         __Source_dict = {key: value for key, value in
00346             self.__siteClass.__dict__.items() if
00347                 not key.startswith('__') and not callable(key)}
00348         self.__appendFile = __Source_dict["append_file"]
00349         __Source_dict.pop("append_file")
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 =
00359             f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingC
00360             ontractDate%20le%20{__Source_dict['dateEnd']}"
00361         elif __Source_dict["ListedOrModified"] == "Modification Date":
00362             filter_string =
00363             f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20a
00364             nd%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00365         elif __Source_dict["ListedOrModified"] == "Close Date":
00366             filter_string =
00367             f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{
00368             __Source_dict['dateEnd']}"
00369
00370         filter_string = filter_string +
00371         f"%20and%20StandardStatus%20has%20data.Models.StandardStatus'{__Source_dict['Stan
00372         dardStatus']}'"
00373
00374         if __Source_dict["select"] is not None:
00375             filter_string = filter_string +
00376             f'&$select={__Source_dict["select']}'
00377
00378         self.__parameterString = filter_string

```

00370

Here is the caller graph for this function:



def Core.UtahRealEstateMain.mainFunc (self)

The mainFunc function is the main function of this module. It will be called by the GUI when a user clicks on the "Run" 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](#).

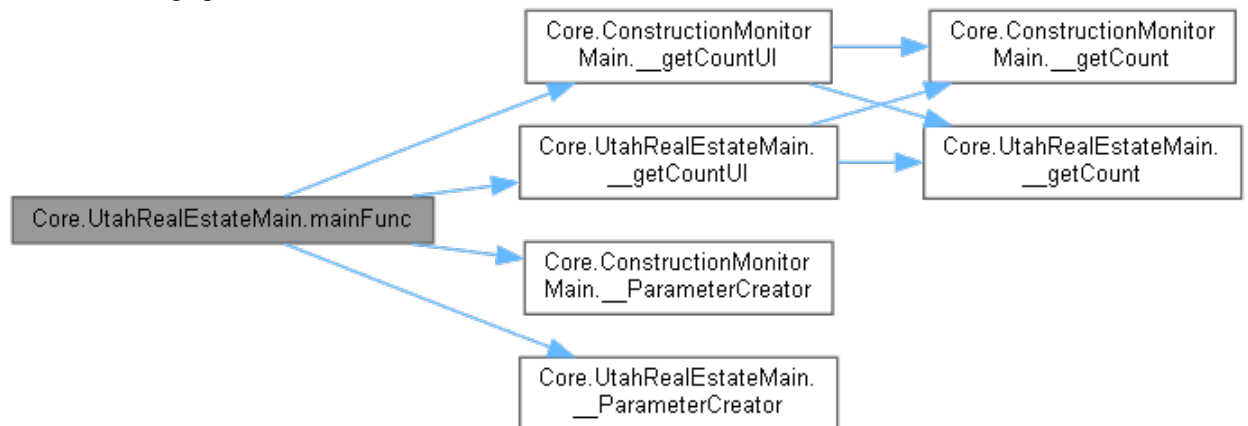
```
00267     def mainFunc(self):
00268         """
00269         The mainFunc function is the main function of this module. It will be called
00270         by the GUI when a user clicks on
00271         the "Run" button in the GUI. The mainFunc function should contain
00272         all of your code for running your program, and it
00273         should return a dataframe that contains all the data you want to display in
00274         your final report.
00275         Args:
00276             self: Reference the object itself
00277         Returns:
00278             A dataframe
00279         Doc Author:
00280             Willem van der Schans, Trelent AI
00281         """
00282         passFlag = False
00283         while not passFlag:
00284             if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00285                 try:
00286                     f = open(self.keyPath, "rb")
00287                     key = f.readline()
00288                     f.close()
00289                     f = open(self.filePath, "rb")
00290                     authDict = json.load(f)
00291                     fernet = Fernet(key)
00292                     authkey =
00293                     fernet.decrypt(authDict["ure"]["auth"]).decode()
00294                     self.__headerDict = {authDict["ure"]["parameter"]}:
00295                     authkey}
00296                     passFlag = True
00297                 except Exception as e:
00298                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00299                     %H:%M:%S.%f')}[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening
00300                     AuthUtil")
00301                     AuthUtil()
00302             else:
00303                 AuthUtil()
```

```

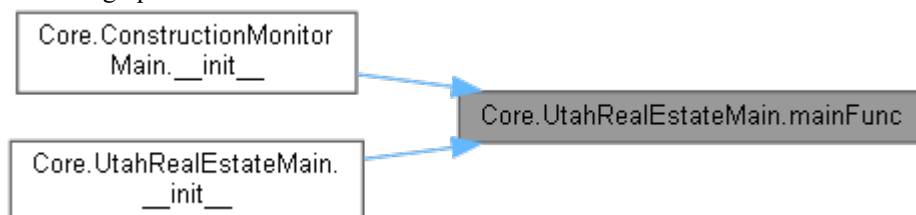
00302
00303     self.__ParameterCreator()
00304
00305     self.__getCountUI()
00306
00307     self.__batches = BatchCalculator(self.__record_val, None)
00308
00309     if self.__batches != 0:
00310         startTime = datetime.datetime.now().replace(microsecond=0)
00311         BatchInputGui(self.__batches)
00312         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for {self.__batches} Batches sent to server")
00313         BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00314         ParameterDict=self.__parameterString,
00315                                     HeaderDict=self.__headerDict,
00316                                     BatchesNum=self.__batches,
00317                                     Type="utah_real_estate")
00318         BatchGuiObject.BatchGuiShow()
00319         self.dataframe = BatchGuiObject.dataframe
00320         print(
00321             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
00322             {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
00323             time.gmtime((datetime.datetime.now().replace(microsecond=0) -
00324             startTime).total_seconds()))}")
00322         FileSaver("ure", self.dataframe, self.__appendFile)
00323     else:
00324         RESTError(994)
00325         raise SystemExit(994)
00326

```

Here is the call graph for this function:



Here is the caller graph for this function:



Member Data Documentation

`Core.UtahRealEstateMain.__appendFile` [private]

Definition at line [236](#) of file [UtahRealEstate/Core.py](#).

`Core.UtahRealEstateMain.__batches` [private]

Definition at line [232](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__dateEnd [private]

Definition at line [238](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__dateStart [private]

Definition at line [237](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__headerDict [private]

Definition at line [234](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__parameterString [private]

Definition at line [235](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__record_val [private]

Definition at line [397](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__restDomain [private]

Definition at line [239](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.__siteClass [private]

Definition at line [233](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.dataframe

Definition at line [231](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.filePath

Definition at line [242](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.key

Definition at line [244](#) of file [UtahRealEstate/Core.py](#).

Core.UtahRealEstateMain.keyPath

Definition at line [240](#) of file [UtahRealEstate/Core.py](#).

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

- `UtahRealEstate/Core.py`

File Documentation

`__init__.py`

main.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 from Initializer import initializer
00005
00006 initializer()
```

AuthUtil.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import ctypes
00005 import datetime
00006 import json
00007 import os
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 from cryptography.fernet import Fernet
00012
00013 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00014 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00015 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00016
00017
00018 class AuthUtil:
00019
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
00026         Args:
00027             self: Represent the instance of the class
00028
00029         Returns:
00030             None
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034         """
00035         self.StandardStatus = None
00036         self.ListedOrModified = None
00037         self.file_name = None
00038         self.append_file = None
00039         self.keyPath =
Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00040         self.filePath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath("Security
")
00041         self.k = None
00042         self.keyFlag = True
00043         self.jsonDict = {}
00044         self.passFlagUre = False
00045         self.passFlagCm = False
00046         self.outcomeText = "Please input the plain text keys in the input boxes above
\n " \
00047
00048         "Submitting will overwrite any old values in an
unrecoverable manner."
00049
00050         if os.path.exists(self.filePath):
00051             pass
00052         else:
00053             if
os.path.exists(Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData")):
00054                 os.mkdir(self.filePath)
00055             else:
00056                 os.mkdir(Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData"))
00057                 os.mkdir(self.filePath)
00058
00059         if os.path.exists(self.keyPath):
00060             pass
00061         else:
00062             if
os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00063                 os.mkdir(self.keyPath)
00064             else:
```



```

00064         os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))
00065                    os.mkdir(self.keyPath)
00066
00067         if
00068         os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00069             try:
00070                 f =
00071                 open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00072                 self.k = f.readline()
00073                 f.close()
00074             except Exception as e:
00075                 print(e)
00076                 RESTError(402)
00077                 raise SystemExit(402)
00078         else:
00079             self.k = Fernet.generate_key()
00080             f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"),
00081                    "wb")
00082             f.write(self.k)
00083             f.close()
00084             try:
00085                 os.remove(self.filePath.joinpath("auth.json"))
00086             except Exception as e:
00087                 # Logging
00088                 print(
00089                     f"{datetime.datetime.today().strftime('%m-%d-%Y
00090                     %H:%M:%S.%f')}[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file - This
00091                     can be due to the file not existing. Continuing...")
00092                 pass
00093
00094             f = open(self.filePath.joinpath("auth.json"), "wb")
00095             f.close()
00096             self.keyFlag = False
00097
00098         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00099
00100         try:
00101             ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3
00102             ra3rvavcr3w"), 2)
00103         except Exception as e:
00104             # Logging
00105             print(
00106                 f"{datetime.datetime.today().strftime('%m-%d-%Y
00107                 %H:%M:%S.%f')}[:-3]} | Authutil.py | Error = {e} | Error when setting the key file as hidden.
00108                 This is either a Permission error or Input Error. Continuing...")
00109             pass
00110
00111         def __SetValues(self, values):
00112             """
00113             The __SetValues function is called when the user clicks on the "OK"
00114             button in the window.
00115             It takes a dictionary of values as an argument, and then uses those values to
00116             update
00117             the auth.json file with new keys for both Utah Real Estate and Construction
00118             Monitor.
00119             Args:
00120                 self: Make the function a method of the class
00121                 values: Store the values that are entered into the form
00122             Returns:
00123                 A dictionary of the values entered by the user
00124             Doc Author:
00125                 Willem van der Schans, Trelent AI
00126             """
00127             ureCurrent = None
00128             cmCurrent = None
00129             keyFile = None
00130
00131             fernet = Fernet(self.k)
00132
00133             try:
00134                 f = open(self.filePath.joinpath("auth.json"), "r")

```

```

00129         keyFile = json.load(f)
00130         fileFlag = True
00131     except:
00132         fileFlag = False
00133
00134     if fileFlag:
00135         try:
00136             ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00137         except Exception as e:
00138             # Logging
00139             print(
00140                 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")
00141             ureCurrent = None
00142
00143         try:
00144             cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00145         except Exception as e:
00146             # Logging
00147             print(
00148                 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")
00149             cmCurrent = None
00150
00151     if values["-ureAuth-"] != "":
00152         self.jsonDict.update(
00153             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00154         self.passFlagUre = True
00155     elif ureCurrent is not None:
00156         self.jsonDict.update(
00157             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00158         self.passFlagUre = True
00159     else:
00160         pass
00161
00162     if values["-cmAuth-"] != "":
00163         self.jsonDict.update(
00164             {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00165         self.passFlagCm = True
00166     elif ureCurrent is not None:
00167         self.jsonDict.update(
00168             {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00169         self.passFlagUre = True
00170     else:
00171         pass
00172
00173     if not self.passFlagUre and not self.passFlagCm:
00174         PopupWrapped("Please make sure you provide keys for both Utah Real estate
and Construction Monitor",
00175                     windowType="errorLarge")
00176     if self.passFlagCm and not self.passFlagUre:
00177         PopupWrapped("Please make sure you provide a key for Utah Real estate",
00178                     windowType="errorLarge")
00179     if not self.passFlagCm and self.passFlagUre:
00180         PopupWrapped("Please make sure you provide a key for Construction
Monitor", windowType="errorLarge")
00181     else:
00182         jsonOut = json.dumps(self.jsonDict, indent=4)
00183         f = open(self.filePath.joinpath("auth.json"), "w")
00184         f.write(jsonOut)
00185
00186     def __ShowGui(self, layout, text):
00187         """
00188         The __ShowGui function is a helper function that displays the GUI to the user.
00189         It takes in two arguments: layout and text. The layout argument is a list of lists,
00190         which contains all the elements that will be displayed on screen. The text
00191         argument
00192         is simply what will be displayed at the top of the window.
00193         Args:

```

```

00194         self: Represent the instance of the class
00195         layout: Pass the layout of the gui to be displayed
00196         text: Set the title of the window
00197
00198     Returns:
00199         A window object
00200     """
00201     window = sg.Window(text, layout, grab_anywhere=False,
00202         return_keyboard_events=True,
00203         finalize=True,
00204         icon=ImageLoader("taskbar_icon.ico"))
00205     while not self.passFlagUre or not self.passFlagCm:
00206         event, values = window.read()
00207
00208         if event == "Submit":
00209             try:
00210                 self.__SetValues(values)
00211             except Exception as e:
00212                 print(e)
00213                 RESTError(993)
00214             finally:
00215                 pass
00216         elif event == sg.WIN_CLOSED or event == "Quit":
00217             break
00218         else:
00219             pass
00220
00221     window.close()
00222
00223 def __CreateFrame(self):
00224     """
00225     The __CreateFrame function creates the GUI layout for the Authentication Utility.
00226     It is called by __init__ and returns a list of lists that contains all the elements
00227     that will be displayed in the window.
00228
00229     Args:
00230         self: Access the class attributes and methods
00231
00232     Returns:
00233         A list of lists
00234
00235     Doc Author:
00236         Trelent
00237     """
00238     sg.theme('Default1')
00239
00240     line00 = [sg.HSeparator()]
00241
00242     line0 = [sg.Image(ImageLoader("logo.png")),
00243         sg.Push(),
00244         sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
00245             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,
00282 line8, line9, line10]
00283     return layout
```

BatchProcessing.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import math
00006 from datetime import date
00007
00008 import pandas as pd
00009 import requests
00010
00011 from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList
00012
00013
00014 def BatchCalculator(TotalRecords, Argument_Dict):
00015     """
00016     The BatchCalculator function takes two arguments:
00017     1. TotalRecords - the total number of records in the database
00018     2. Argument_Dict - a dictionary containing all the arguments passed to this
    function by the user
00019
00020     Args:
00021         TotalRecords: Determine the number of batches that will be needed to complete
    the query
00022         Argument_Dict: Pass in the arguments that will be used to query the database
00023
00024     Returns:
00025         The total number of batches that will be made
00026
00027     Doc Author:
00028         Willem van der Schans, Trelent AI
00029     """
00030     try:
00031         document_limit = Argument_Dict["size"]
00032     except Exception as e:
00033         # Logging
00034         print(
00035             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")
00036         document_limit = 200
00037
00038     return int(math.ceil(float(TotalRecords) / float(document_limit)))
00039
00040
00041 class BatchProcessorConstructionMonitor:
00042
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
00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes
it to the ConstructionMonitorProcessor function.
00081         The ConstructionMonitorProcessor function then uses this valueObject to
determine which of its functions should be called.
00082
00083         Args:
00084             self: Represent the instance of the class
00085
00086         Returns:
00087             The result of the constructionmonitorprocessor function
00088
00089         Doc Author:
00090             Willem van der Schans, Trelent AI
00091         """
00092         self.ConstructionMonitorProcessor(self.valueObject)
00093
00094     def ConstructionMonitorProcessor(self, valueObject):
00095         """
00096         The ConstructionMonitorProcessor function will use requests to get data from
00097         ConstructionMontior.com's ReST API and store it into a pandas DataFrame object
called __df (which is local). This
00098         process will be repeated until all the data has been collected from
ConstructionMonitor.com's ReST API, at which point __df will contain all
00099
00100         Args:
00101             self: Represent the instance of the object itself
00102             valueObject: Update the progress bar in the gui
00103
00104         Returns:
00105             A dataframe
00106
00107         Doc Author:
00108             Willem van der Schans, Trelent AI
00109         """
00110         __df = None
00111         for callNum in range(0, self.__requestCount):
00112             self.__parameterDict["from"] = 0
00113
00114             if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00115                 __batchNum = self.__requestCalls
00116                 if __df is None:
00117                     self.__dateTracker = str(date.today())
00118                 else:
00119                     self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate']).strftime('%Y-%m-%d')
00120                     elif self.__requestCount == 1:
00121                         __batchNum = self.__numBatches
00122                         self.__dateTracker = str(date.today())
00123                     else:
00124                         __batchNum = self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])) - (
00125                             self.__requestCount - 1)
00126                         self.__dateTracker =
min(pd.to_datetime(__df['lastIndexedDate']).strftime('%Y-%m-%d')
00127
00128                 self.__parameterDict['dateEnd'] = self.__dateTracker
00129
00130                 for record in range(0, int(math.ceil(__batchNum))):
00131                     if record != 0:

```

```

00132         self.__parameterDict["from"] = record *
int(self.__parameterDict["size"])
00133
00134         response = requests.post(url=self.__restDomain,
00135                                 headers=self.__headerDict,
00136                                 json=self.__parameterDict)
00137
00138         counter = 0
00139         try:
00140             response = response.json()['hits']['hits']
00141         except KeyError as e:
00142             # Logging
00143             print(
00144                 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}")
00145             continue
00146
00147         valueObject.setValue(valueObject.getValue() + 1)
00148
00149         if record == 0 and callNum == 0:
00150             __df = pd.json_normalize(response[counter]["_source"])
00151             __df["id"] = response[counter]['_id']
00152             __df["county"] =
response[counter]["_source"]['county']['county_name']
00153             counter += 1
00154
00155             for i in range(counter, len(response)):
00156                 __tdf = pd.json_normalize(response[i]["_source"])
00157                 __tdf["id"] = response[i]['_id']
00158                 __tdf["county"] =
response[i]["_source"]['county']['county_name']
00159                 __df = pd.concat([__df, __tdf], ignore_index=True)
00160
00161             if self.__columnSelection is not None:
00162                 __col_list = StringToList(self.__columnSelection)
00163                 __col_list.append("id")
00164                 __col_list.append("county")
00165             else:
00166                 pass
00167
00168             self.dataframe = __df
00169             valueObject.setValue(-999)
00170
00171
00172 class BatchProcessorUtahRealEstate:
00173
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
00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an argument
00205         and then calls the appropriate
00206         function based on what was selected in the dropdown menu. The valueObject
00207         is passed to each of these functions
00208         so that they can access all of its attributes.
00209
00210         Args:
00211             self: Represent the instance of the class
00212
00213         Returns:
00214             The function that is selected by the user
00215
00216         Doc Author:
00217             Willem van der Schans, Trelent AI
00218         """
00219         self.BatchProcessingUtahRealestateCom(self.valueObject)
00220
00221     def BatchProcessingUtahRealestateCom(self, valueObject):
00222         """
00223         The BatchProcessingUtahRealestateCom function is a function that takes in the
00224         valueObject and uses it to
00225         update the progress bar. It also takes in self, which contains all the
00226         necessary information for this
00227         function to work properly. The BatchProcessingUtahRealestateCom function
00228         will then use requests to get data from
00229         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
00230         called __df (which is local). This
00231         process will be repeated until all the data has been collected from
00232         UtahRealestate.com's ReST API, at which point __df will contain all
00233
00234         Args:
00235             self: Represent the instance of the class
00236             valueObject: Pass the value of a progress bar to the function
00237
00238         Returns:
00239             A dataframe of the scraped data
00240
00241         Doc Author:
00242             Willem van der Schans, Trelent AI
00243         """
00244         __df = pd.DataFrame()
00245
00246         for batch in range(self.__numBatches):
00247             if batch == 0:
00248                 response =
00249                 requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00250                             headers=self.__headerDict)
00251
00252                 response_temp = response.json()
00253                 __df = pd.json_normalize(response_temp, record_path=['value'])
00254             else:
00255                 response =
00256                 requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch *
00257                             200}",
00258                             headers=self.__headerDict)
00259
00260                 response_temp = response.json()
00261                 response_temp = pd.json_normalize(response_temp,
00262                                                     record_path=['value'])
00263                 __df = pd.concat([__df, response_temp], ignore_index=True)
00264
00265                 valueObject.setValue(valueObject.getValue() + 1)
00266
00267         self.dataframe = __df
00268         valueObject.setValue(-999)

```


DataChecker.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import os
00005 from pathlib import Path
00006
00007 import PySimpleGUI as sg
00008
00009 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00010
00011
00012 def DataChecker(Name, DataPath):
00013     """
00014     The DataChecker function is used to check if the user has selected a valid data file.
00015     If the user selects an invalid file, they will be prompted to select another one
00016     until they choose a valid one.
00017
00018     Args:
00019         Name: Display the name of the data file that is being selected
00020         Path: Set the initial folder for the file browser
00021
00022     Returns:
00023         A list of all the data files in a directory
00024
00025     Doc Author:
00026         Willem van der Schans, Trelent AI
00027     """
00028     __text1 = f"Select existing {Name} csv data file:"
00029
00030     __Line1 = [sg.Push(),
00031                sg.Text(__text1, justification="center"),
00032                sg.Push()]
00033
00034     __Line2 = [sg.Text("Choose a file: "),
00035                sg.Input(),
00036                sg.FileBrowse(file_types=(("Data Files (.csv)", "*.csv")),
initial_folder=DataPath)]
00037
00038     __Line3 = [sg.Push(),
00039                sg.Ok("Continue"),
00040                sg.Cancel(),
00041                sg.Push()]
00042
00043     window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
00044                          modal=True,
00045                          keep_on_top=True,
00046                          disable_close=False,
00047                          icon=ImageLoader("taskbar_icon.ico"))
00048
00049     while True:
00050         event, values = window.read()
00051         if event == "Continue":
00052             break
00053         elif event == sg.WIN_CLOSED or event == "Cancel":
00054
00055             break
00056
00057     window.close()
00058
00059
00060 DataChecker("Construction Monitor", Path(os.path.expanduser('~/Documents')))
```

DataSupportFunctions.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 def StringToList(string):
00005     """
00006     The StringToList function takes a string and converts it into a list.
00007     The function is used to convert the input from the user into a list of strings,
which can then be iterated through.
00008
00009     Args:
00010         string: Split the string into a list
00011
00012     Returns:
00013         A list of strings
00014
00015     Doc Author:
00016         Willem van der Schans, Trelent AI
00017     """
00018     listOut = list(string.split(", "))
00019     return listOut
```

FileSaver.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 from pathlib import Path
00007
00008 import pandas as pd
00009
00010 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00011
00012
00013 class FileSaver:
00014
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:
00024             self: Represent the instance of the class
00025             method: Determine which dataframe to append the new data to
00026             outputDF: Pass in the dataframe that will be saved to a csv file
00027             AppendingPath: Specify the path to an existing csv file that you want to
append your dataframe to
00028
00029         Returns:
00030             Nothing
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034         """
00035         self.docPath =
Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00036             datetime.datetime.today().strftime('%m%d%Y'))
00037         self.data = outputDF
00038         self.dataAppending = None
00039         self.appendFlag = True
00040         self.fileName =
f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00041         self.uiFlag = True
00042
00043         if method.lower() == "ure":
00044             self.primaryKey = "ListingKeyNumeric"
00045         elif method.lower() == "cm":
00046             self.primaryKey = "id"
00047         elif "realtor" in method.lower():
00048             self.primaryKey = None
00049             self.uiFlag = False
00050         elif method.lower() == "cfbp":
00051             self.primaryKey = None
00052             self.uiFlag = False
00053         else:
00054             raise ValueError("method input is invalid choice one of 4 options: URE,
CM, Realtor, CFBP")
00055
00056         if AppendingPath is None:
00057             self.appendFlag = False
00058         else:
00059             self.dataAppending = pd.read_csv(AppendingPath)
00060
00061         if self.appendFlag:
00062             if self.primaryKey is not None:
00063                 # Due to low memory loading the columns are not typed properly,
00064                 # since we are comparing this will be an issue since we need to do
type comparisons,
```

```

00065         # so here we coerce the types of the primary keys to numeric.
00066         # If another primary key is ever chosen make sure to core to the right
data_type.
00067         self.dataAppending[self.primaryKey] =
pd.to_numeric(self.dataAppending[self.primaryKey])
00068         self.data[self.primaryKey] =
pd.to_numeric(self.data[self.primaryKey])
00069
00070         self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00071 keep="last")
00072         else:
00073         self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(keep="last")
00074         else:
00075         self.outputFrame = self.data
00076
00077         if os.path.exists(self.docPath):
00078         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00079         else:
00080         os.mkdir(self.docPath)
00081         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName),
index=False)
00082
00083         if self.uiFlag:
00084         if self.appendFlag:
00085         PopupWrapped(text=f"File Appended and Saved to
{self.docPath.joinpath(self.fileName)}",
00086 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
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))

```

ErrorPopup.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00005
00006
00007 def ErrorPopup(textString):
00008     """
00009     The ErrorPopup function is used to display a popup window with an error message.
00010     It takes one argument, textString, which is the string that will be displayed in the
00011     popup window.
00012     The function also opens up the log folder upon program exit.
00013     Args:
00014         textString: Display the error message
00015     Returns:
00016         Nothing, but it does print an error message to the console
00017     Doc Author:
00018         Willem van der Schans, Trelent AI
00019     """
00020     PopupWrapped(
00021         f"ERROR @ {textString} \n"
00022         f"Log folder will be opened upon program exit",
00023         windowType="FatalErrorLarge")
```

ErrorPrint.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005
00006
00007 def RESTErrorPrint(response):
00008     """
00009     The RESTErrorPrint function is used to print the response from a ReST API call.
00010     If the response is an integer, it will be printed as-is. If it's not an integer,
00011     it will be converted to text and then printed.
00012
00013     Args:
00014         response: Print the response from a rest api call
00015
00016     Returns:
00017         The response text
00018
00019     Doc Author:
00020         Willem van der Schans, Trelent AI
00021     """
00022     if isinstance(response, int):
00023         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Resource Response: {response}")
00024     else:
00025         response_txt = response.text
00026         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Resource Response: {response_txt}")
```

Logger.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 import sys
00007 from pathlib import Path
00008
00009
00010 def logger():
00011     """
00012     The logger function creates a log file in the user's AppData directory.
00013     The function will create the directory if it does not exist.
00014     The function will also delete the oldest file when 100 logs have been saved to prevent
    bloat.
00015
00016     Args:
00017
00018     Returns:
00019         A file path to the log file that was created
00020
00021     Doc Author:
00022         Willem van der Schans, Trelent AI
00023     """
00024     dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
00025     if os.path.exists(dir_path):
00026         pass
00027     else:
00028         if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00029             os.mkdir(dir_path)
00030         else:
00031             os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00032             os.mkdir(dir_path)
00033
00034     filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
00035         f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
00036     sys.stdout = open(filePath, 'w')
00037     sys.stderr = sys.stdin = sys.stdout
00038
00039     def sorted_ls(path):
00040         """
00041         The sorted_ls function takes a path as an argument and returns the files in that
    directory sorted by modification time.
00042
00043     Args:
00044         path: Specify the directory to be sorted
00045
00046     Returns:
00047         A list of files in a directory sorted by modification time
00048
00049     Doc Author:
00050         Willem van der Schans, Trelent AI
00051     """
00052     mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
00053     return list(sorted(os.listdir(path), key=mtime))
00054
00055     del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
00056     for file in del_list:
00057         os.remove(dir_path.joinpath(file))
00058         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
    %H:%M:%S.%f')}[:-3]} | Log file {file} deleted")
```

PrintFunc.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
00002 on http://www.apache.org/licenses/
00003
```


RESError.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005
00006 from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
00007 from API_Calls.Functions.ErrorFunc.ErrorPrint import RESErrorPrint
00008
00009
00010 def RESError(response):
00011     """
00012     The RESError function is a function that checks the status codes.
00013     If it is 200, then everything went well and nothing happens. If it isn't 200, then
    an error message will be printed to
00014     the console with information about what happened (i.e., if there was an authentication
    error or if the resource wasn't found).
00015     The function also raises an exception and opens an error popup for easy debugging.
00016
00017     Args:
00018         response: Print out the response from the server
00019
00020     Returns:
00021         A text string
00022
00023     Doc Author:
00024         Trelent
00025     """
00026     if isinstance(response, int):
00027         status_code = response
00028     else:
00029         status_code = response.status_code
00030
00031     if status_code == 200:
00032         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Api Request completed successfully"
00033         print(textString)
00034         pass
00035     elif status_code == 301:
00036         RESErrorPrint(response)
00037         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"
00038         ErrorPopup(textString)
00039         raise ValueError(textString)
00040     elif status_code == 400:
00041         RESErrorPrint(response)
00042         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Bad Request; check input arguments"
00043         ErrorPopup(textString)
00044         raise ValueError(textString)
00045     elif status_code == 401:
00046         RESErrorPrint(response)
00047         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Authentication Error: No keys found"
00048         ErrorPopup(textString)
00049         raise PermissionError(textString)
00050     elif status_code == 402:
00051         RESErrorPrint(response)
00052         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"
00053         ErrorPopup(textString)
00054         raise PermissionError(textString)
00055     elif status_code == 403:
00056         RESErrorPrint(response)
00057         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"
00058         ErrorPopup(textString)
00059         raise PermissionError(textString)
00060     elif status_code == 404:
00061         RESErrorPrint(response)
```

```

00062         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"
00063         ErrorPopup(textString)
00064         raise NameError(textString)
00065     elif status_code == 405:
00066         RESTErrorPrint(response)
00067         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"
00068         ErrorPopup(textString)
00069         raise ValueError(textString)
00070     elif status_code == 408:
00071         RESTErrorPrint(response)
00072         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Requests timeout by server"
00073         ErrorPopup(textString)
00074         raise TimeoutError(textString)
00075     elif status_code == 503:
00076         RESTErrorPrint(response)
00077         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"
00078         ErrorPopup(textString)
00079         raise SystemError(textString)
00080     elif status_code == 701:
00081         RESTErrorPrint(response)
00082         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Error in coercing icon to bits
(Imgeloader.py)"
00083         ErrorPopup(textString)
00084         raise TypeError(textString)
00085     elif status_code == 801:
00086         RESTErrorPrint(response)
00087         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"
00088         ErrorPopup(textString)
00089         raise ValueError(textString)
00090     elif status_code == 790:
00091         RESTErrorPrint(response)
00092         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Requests timeout within requests"
00093         ErrorPopup(textString)
00094         raise TimeoutError(textString)
00095     elif status_code == 791:
00096         RESTErrorPrint(response)
00097         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
00098         ErrorPopup(textString)
00099         raise ValueError(textString)
00100     elif status_code == 990:
00101         RESTErrorPrint(response)
00102         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | No password input"
00103         ErrorPopup(textString)
00104         raise ValueError(textString)
00105     elif status_code == 991:
00106         RESTErrorPrint(response)
00107         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | No username input"
00108         ErrorPopup(textString)
00109         raise ValueError(textString)
00110     elif status_code == 992:
00111         RESTErrorPrint(response)
00112         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)"
00113         ErrorPopup(textString)
00114         raise ValueError(textString)
00115     elif status_code == 993:
00116         RESTErrorPrint(response)
00117         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"
00118         ErrorPopup(textString)
00119         print(ValueError(textString))

```

```

00120     elif status_code == 994:
00121         RESTErrorPrint(response)
00122         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Submission Error: server returned no
documents"
00123         ErrorPopup(textString)
00124         raise ValueError(textString)
00125     elif status_code == 1000:
00126         RESTErrorPrint(response)
00127         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Catastrophic Error"
00128         ErrorPopup(textString)
00129         raise SystemError(textString)
00130     elif status_code == 1001:
00131         RESTErrorPrint(response)
00132         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | Main Function Error Break"
00133         raise SystemError(textString)
00134     elif status_code == 1100:
00135         RESTErrorPrint(response)
00136         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Status Code = {status_code} | User has cancelled the program
execution"
00137         raise KeyboardInterrupt(textString)
00138     elif status_code == 1101:
00139         RESTErrorPrint(response)
00140         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"
00141         print(textString)
00142     else:
00143         RESTErrorPrint(response)
00144         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

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003 #
00004 # 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.
00005 #
00006 # THE CONTENTS OF THIS PROJECT ARE PROPRIETARY AND CONFIDENTIAL.
00007 # UNAUTHORIZED COPYING, TRANSFERRING OR REPRODUCTION OF THE CONTENTS OF THIS
PROJECT, VIA ANY MEDIUM IS STRICTLY PROHIBITED.
00008 #
00009 # The receipt or possession of the source code and/or any parts thereof does not
convey or imply any right to use them
00010 # for any purpose other than the purpose for which they were provided to you.
00011 #
00012 #
00013 import PySimpleGUI as sg
00014
00015 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00016
00017
00018 def BatchInputGui(batches):
00019     """
00020     The BatchInputGui function is a simple GUI that asks the user if they want to continue
with the number of batches
00021     that have been selected. This function is called by the BatchInputGui function in
order to confirm that this is what
00022     the user wants.
00023
00024     Args:
00025         batches: Display the number of batches that will be run
00026
00027     Returns:
00028         A boolean value
00029
00030     Doc Author:
00031         Willem van der Schans, Trelent AI
00032     """
00033     __text1 = f"This request will run {batches} batches"
00034     __text2 = "Do you want to continue?"
00035
00036     __Line1 = [sg.Push(),
00037                sg.Text(__text1, justification="center"),
00038                sg.Push()]
00039
00040     __Line2 = [sg.Push(),
00041                sg.Text(__text2, justification="center"),
00042                sg.Push()]
00043
00044     __Line3 = [sg.Push(),
00045                sg.Ok("Continue"),
00046                sg.Cancel(),
00047                sg.Push()]
00048
00049     window = sg.Window("Batch popup", [__Line1, __Line2, __Line3],
00050                        modal=True,
00051                        keep_on_top=True,
00052                        disable_close=False,
00053                        icon=ImageLoader("taskbar_icon.ico"),
00054                        size=(290, 100))
00055
00056     while True:
00057         event, values = window.read()
00058         if event == "Continue":
00059             break
00060         elif event == sg.WIN_CLOSED or event == "Cancel":
00061
00062             break
00063
00064     window.close()
```

BatchProgressGUI.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003 import datetime
00004 import threading
00005 import time
00006
00007 import PySimpleGUI as sg
00008
00009 from API_Calls.Functions.DataFunc.BatchProcessing import
BatchProcessorConstructionMonitor, BatchProcessorUtahRealEstate
00010 from API_Calls.Functions.Gui.DataTransfer import DataTransfer
00011 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00012 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00013
00014 counter = 1
00015
00016
00017 class BatchProgressGUI:
00018
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         """
00042         self.__parameterDict = ParameterDict
00043         self.__restDomain = RestDomain
00044         self.__headerDict = HeaderDict
00045         self.__columnSelection = ColumnSelection
00046         self.__type = Type
00047         self.__dataframe = None
00048
00049         self.__layout = None
00050         self.__batches = BatchesNum
00051         self.__window = None
00052         self.__batch_counter = 0
00053
00054     def BatchGuiShow(self):
00055         """
00056         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.
00057
00058         Args:
00059             self: Represent the instance of the class
00060
00061         Returns:
00062             The __type of the batchgui class
00063
00064         Doc Author:
00065             Willem van der Schans, Trelent AI
00066
00067         """
00068         self.CreateProgressLayout()
```

```

00067         self.createGui(self.__type)
00068
00069     def CreateProgressLayout(self):
00070
00071         """
00072         The CreateProgressLayout function creates the layout for the progress window.
00073         The function takes in self as a parameter and returns nothing.
00074
00075         Parameters:
00076             self (object): The object that is calling this function.
00077
00078         Args:
00079             self: Access the class variables and methods
00080
00081         Returns:
00082             A list of lists
00083
00084         Doc Author:
00085             Willem van der Schans, Trelent AI
00086         """
00087         sg.theme('Default1')
00088
00089         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--progress_text--"),
00090                   sg.Push()]
00091
00092         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10),
justification="center", key="--timer--"),
00093                   sg.Text(font=("Helvetica", 10), justification="center",
key="--time_est--"), sg.Push()]
00094
00095         __Line3 = [
00096             sg.ProgressBar(max_value=self.__batches, bar_color=("#920303",
"#C9c8c8"), orientation='h', size=(30, 20),
00097                           key='--progress_bar--')]
00098
00099
00100         layout = [__Line1, __Line2, __Line3]
00101
00102         self.__layout = layout
00103
00104     def createGui(self, Sourcetype):
00105
00106         """
00107         The createGui function is the main function that creates the GUI.
00108         It takes in a type parameter which determines what kind of batch processor to
use.
00109         The createGui function then sets up all the variables and objects needed for
00110         the program to run, including: window, start_time, update_text, valueObj
(DataTransfer),
00111         processorObject (BatchProcessorConstructionMonitor or
BatchProcessorUtahRealestate),
00112         and threading objects for TimeUpdater and ValueChecker functions. The createGui
function also starts these threads.
00113
00114         Args:
00115             self: Access the object itself
00116             Sourcetype: Determine which batch processor to use
00117
00118         Returns:
00119             The dataframe
00120
00121         Doc Author:
00122             Willem van der Schans, Trelent AI
00123         """
00124         self.__window = sg.Window('Progress', self.__layout, finalize=True,
icon=ImageLoader("taskbar_icon.ico"))
00125
00126         start_time = datetime.datetime.now().replace(microsecond=0)
00127         update_text = f"Batch {0} completed"
00128         self.__window['--progress_text--'].update(update_text)
00129         self.__window['--progress_bar--'].update(0)
00130         self.__window['--time_est--'].update("Est time needed 00:00:00")
00131
00132         valueObj = DataTransfer()
00133         valueObj.setValue(0)
00134

```

```

00135         if Sourcetype == "construction_monitor":
00136
00137             processorObject =
BatchProcessorConstructionMonitor(RestDomain=self. restDomain,
00138
NumBatches=self. batches,
00139
ParameterDict=self. parameterDict,
00140
HeaderDict=self. headerDict,
00141
ColumnSelection=self. columnSelection,
00142
valueObject=valueObj)
00143         elif Sourcetype == "utah_real_estate":
00144             processorObject =
BatchProcessorUtahRealEstate(RestDomain=self. restDomain,
00145
NumBatches=self. batches,
00146
ParameterString=self. parameterDict,
00147
HeaderDict=self. headerDict,
00148
valueObject=valueObj)
00149
00150             threading.Thread(target=self. TimeUpdater,
00151                             args=(start_time,),
00152                             daemon=True).start()
00153             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | TimeUpdater Thread Successfully Started")
00154
00155             batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
00156                                                daemon=False)
00157             batchFuncThread.start()
00158             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | BatchFunc Thread Successfully Started")
00159             threading.Thread(target=self. ValueChecker,
00160                             args=(valueObj,),
00161                             daemon=False).start()
00162             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | ValueChecker Thread Successfully Started")
00163
00164             while True:
00165
00166                 self. ProgressUpdater(valueObj)
00167
00168                 if valueObj.getValue() == -999:
00169                     break
00170
00171                 window, event, values = sg.read_all_windows()
00172                 if event.startswith('update'):
00173                     __key_to_update = event[len('update'):]
00174                     window[__key_to_update].update(values[event])
00175                     window.refresh()
00176                     pass
00177
00178                 if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179                     break
00180
00181                 time.sleep(0.1)
00182
00183                 self. dataframe = processorObject.dataframe
00184                 self. window.close()
00185
00186                 PopupWrapped(text="Api Request Completed", windowType="notice")
00187
00188             def ProgressUpdater(self, valueObj):
00189                 """
00190                 The ProgressUpdater function is a callback function that updates the progress
00191                 bar and text
00192                 in the GUI. It takes in one argument, which is an object containing information
00193                 about the
00194                 current batch number. The ProgressUpdater function then checks if this value has
00195                 changed from
00196                 the last time it was called (i.e., if we are on a new batch). If so, it updates
00197                 both the progress
00198                 bar and text with this new information.

```

```

00195
00196     Args:
00197         self: Make the progressupdater function an instance method
00198         valueObj: Get the current value of the batch counter
00199
00200     Returns:
00201         The value of the batch counter
00202
00203     Doc Author:
00204         Willem van der Schans, Trelent AI
00205     """
00206     if valueObj.getValue() != self.__batch_counter:
00207         self.__batch_counter = valueObj.getValue()
00208
00209     __update_text = f"Batch {self.__batch_counter}/{self.__batches}
completed"
00210
00211     self.__window.write_event_value('update--progress_bar--',
self.__batch_counter)
00212     self.__window.write_event_value('update--progress_text--',
__update_text)
00213     else:
00214         pass
00215
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(
seconds=(__passed_time * (self.__batches /
self.__batch_counter) - __passed_time)).seconds
00254             except:
00255                 __time_est = datetime.timedelta(
seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00256
00257
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
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

```

DataTransfer.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 class DataTransfer:
00005
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026
00027         Args:
00028             self: Represent the instance of the class
00029             value: Set the value of the instance variable __value
00030
00031         Returns:
00032             The value that was passed to it
00033
00034         Doc Author:
00035             Willem van der Schans, Trelent AI
00036         """
00037         self.__value = value
00038
00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054
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()
```


ImageLoader.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import base64
00005 import os
00006 from io import BytesIO
00007 from os.path import join, normpath
00008
00009 from PIL import Image
00010
00011
00012 def ImageLoader(file):
00013     """
00014     The ImageLoader function takes in a file name and returns the image as a base64 encoded
00015     string. This is used to send images to the API for processing.
00016
00017     Args:
00018         file: Specify the image file to be loaded
00019
00020     Returns:
00021         A base64 encoded image string
00022
00023     Doc Author:
00024         Willem van der Schans, Trelent AI
00025     """
00026     try:
00027         __path = normpath(join(str(os.getcwd()).split("API_Calls", 1)[0]),
00028 "API_Calls"))
00029         __path = normpath(join(__path, "Images"))
00030         __path = join(__path, file).replace("\\", "/")
00031
00032         image = Image.open(__path)
00033
00034         __buff = BytesIO()
00035
00036         image.save(__buff, format="png")
00037
00038         img_str = base64.b64encode(__buff.getvalue())
00039
00040         return img_str
00041     except Exception as e:
00042         # We cannot log this error like other errors due to circular imports
00043         raise e
```

PopupWrapped.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import os
00005 import time
00006 from pathlib import Path
00007
00008 import PySimpleGUI as sg
00009
00010 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00011
00012
00013 class PopupWrapped():
00014
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
00018         this class is created.
00019         It sets up all the variables and creates a window for us to use.
00020         Args:
00021             self: Represent the instance of the class
00022             text: Set the text of the window
00023             windowType: Determine what type of window to create
00024             error: Display the error message in the window
00025         Returns:
00026             Nothing
00027         Doc Author:
00028             Willem van der Schans, Trelent AI
00029         """
00030         self.__text = text
00031         self.__type = windowType
00032         self.__error = error
00033         self.__layout = []
00034         self.__windowObj = None
00035         self.__thread = None
00036         self.__counter = 0
00037
00038         self.__createWindow()
00039
00040     def __createLayout(self):
00041         """
00042         The __createLayout function is used to create the layout of the window.
00043         The function takes class variables and returns a window layout.
00044         It uses a series of if statements to determine what type of window it is, then
00045         creates a layout based on that information.
00046         Args:
00047             self: Refer to the current instance of a class
00048         Returns:
00049             A list of lists
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         sg.theme('Default1')
00054         __Line1 = None
00055         __Line2 = None
00056
00057         if self.__type == "notice":
00058             __Line1 = [sg.Push(),
00059                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00060                               justification="center"),
00061                        sg.Text(self.__text, justification="center",
00062                               key="-textField-"), sg.Push()]
00063             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00064         elif self.__type == "noticeLarge":
00065             __Line1 = [sg.Push(),
00066                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
00067                               justification="center"),
00068                        sg.Text(self.__text, justification="center",
00069                               key="-textField-"), sg.Push()]
00070             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00071         elif self.__type == "errorLarge":
00072             __Line1 = [sg.Push(),
```

```

00067             sg.Text(u'\u274C', font=("Helvetica", 20, "bold"),
justification="center"),
00068             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         if self.__type == "progress":
00085             self.__layout = [__Line1, ]
00086         else:
00087             self.__layout = [__Line1, __Line2]
00088
00089     def __createWindow(self):
00090         """
00091         The __createWindow function is used to create the window object that will be
displayed.
00092         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,
00105                                         modal=True,
00106                                         keep_on_top=True,
00107                                         disable_close=False,
00108                                         icon=ImageLoader("taskbar_icon.ico"),
00109                                         size=(290, 50))
00110         elif self.__type == "noticeLarge":
00111             self.__windowObj = sg.Window(title="Notice", layout=self.__layout,
finalize=True,
00112                                         modal=True,
00113                                         keep_on_top=True,
00114                                         disable_close=False,
00115                                         icon=ImageLoader("taskbar_icon.ico"))
00116         elif self.__type == "errorLarge":
00117             self.__windowObj = sg.Window(title="Error", layout=self.__layout,
finalize=True,
00118                                         modal=True,
00119                                         keep_on_top=True,
00120                                         disable_close=False,
00121                                         icon=ImageLoader("taskbar_icon.ico"))
00122         elif self.__type == "FatalErrorLarge":
00123             self.__windowObj = sg.Window(title="Fatal Error",
layout=self.__layout, finalize=True,
00124                                         modal=True,
00125                                         keep_on_top=True,

```

```

00126                                     disable_close=False,
00127                                     icon=ImageLoader("taskbar_icon.ico"))
00128     else:
00129         self.__windowObj = sg.Window(title=self.__type, layout=self.__layout,
00130 finalize=True,
00131                                     modal=True,
00132                                     keep_on_top=True,
00133                                     disable_close=False,
00134                                     icon=ImageLoader("taskbar_icon.ico"),
00135                                     size=(290, 80))
00136     if self.__type != "progress" or self.__type.startswith("perm"):
00137         timer = 0
00138         while timer < 100:
00139             event, values = self.__windowObj.read()
00140             if event == "Ok" or event == sg.WIN_CLOSED:
00141                 break
00142
00143             time.sleep(0.1)
00144
00145             if self.__type == "FatalErrorLarge":
00146                 try:
00147                     os.system(
00148                         f"start
00149 {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}"
00150                     except Exception as e:
00151                         print(
00152                             f"PopupWrapped.py | Error = {e} | Log Folder not found please
00153 search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")
00154
00155                 self.__windowObj.close()
00156
00157 def stopWindow(self):
00158     """
00159     The stopWindow function is used to close the window object that was created in
00160 the startWindow function.
00161     This is done by calling the close() method on self.__windowObj, which will cause
00162 it to be destroyed.
00163     Args:
00164         self: Represent the instance of the class
00165     Returns:
00166         The window object
00167     Doc Author:
00168         Willem van der Schans, Trelent AI
00169     """
00170     self.__windowObj.close()
00171
00172 def textUpdate(self, sleep=0.5):
00173     """
00174     The textUpdate function is a function that updates the text in the text field.
00175 It does this by adding dots to the end of it, and then removing them. This creates
00176 a loading effect for when something is being processed.
00177     Args:
00178         self: Refer to the object itself
00179         sleep: Control the speed of the text update
00180     Returns:
00181         A string that is the current text of the text field
00182     Doc Author:
00183         Willem van der Schans, Trelent AI
00184     """
00185     self.__counter += 1
00186     if self.__counter == 4:
00187         self.__counter = 1
00188         newString = ""
00189         if self.__type == "notice":
00190             pass
00191         elif self.__type == "error":
00192             pass
00193         elif self.__type == "progress":
00194             newString = f"{self.__text}{'.' * self.__counter}"
00195         self.__windowObj.write_event_value('update-textField-', newString)
00196
00197         time.sleep(sleep)
00198
00199 def windowPush(self):
00200     """

```

```

00198     The windowPush function is used to update the values of a window object.
00199     The function takes in an event and values from the window object, then checks
00199     if the event starts with 'update'.
00200     If it does, it will take everything after 'update' as a key for updating that
00200     specific value.
00201     It will then update that value using its key and refresh the window.
00202     Args:
00203         self: Reference the object that is calling the function
00204     Returns:
00205         A tuple containing the event and values
00206     Doc Author:
00207         Willem van der Schans, Trelent AI
00208     """
00209     event, values = self.__windowObj.read()
00210
00211     if event.startswith('update'):
00212         __key_to_update = event[len('update'):]
00213         self.__windowObj[__key_to_update].update(values[event])
00214         self.__windowObj.refresh()

```


Initializer.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found
on http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 from pathlib import Path
00007
00008 import PySimpleGUI as sg
00009
00010 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00011 from API_Calls.Functions.ErrorFunc.Logger import logger
00012 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00013 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00014 from API_Calls.Sources.CFBP.Core import Cencus
00015 from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
ConstructionMonitorMain
00016
00017 from API_Calls.Sources.Realtor.Core import realtorCom
00018 from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain,
UtahRealEstateInit
00019
00020
00021 class initializer:
00022
00023     def __init__(self):
00024
00025         """
00026         The __init__ function is called when the class is instantiated.
00027         It sets up the logging, calls the __ShowGui function to create and display
00028         the GUI, and then calls __CreateFrame to create a frame for displaying widgets.
00029
00030         Args:
00031             self: Represent the instance of the class
00032
00033         Returns:
00034             Nothing
00035
00036         Doc Author:
00037             Willem van der Schans, Trelent AI
00038         """
00039         self.classObj = None
00040
00041         logger()
00042
00043         print("\n\n-----Initiate Program-----\n\n")
00044
00045         self.__ShowGui(self.__CreateFrame(), "Data Tool")
00046
00047         print("\n\n-----Closing Program-----\n\n")
00048
00049     def __ShowGui(self, layout, text):
00050
00051         """
00052         The __ShowGui function is the main function that displays the GUI.
00053         It takes two arguments: layout and text. Layout is a list of lists, each containing
00054         a tuple with three elements:
00055             1) The type of element to be displayed (e.g., "Text",
00056             "InputText", etc.)
00057             2) A dictionary containing any additional parameters for that element (e.g.,
00058             size, default value, etc.)
00059             3) An optional key name for the element (used in event handling). If no key
00060             name is provided then one will be generated automatically by PySimpleGUIQt based on its
00061             position in the layout list
00062
00063         Args:
00064             self: Represent the instance of the class
00065             layout: Pass the layout of the window to be created
00066             text: Set the title of the window
00067
00068         Returns:
00069             A window object
00070
00071         """
```

```

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-----")
00087                 realtorCom()
00088                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Realtor.com API Call-----\n")
00089             elif event == "Census":
00090                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Census API Call-----")
00091                 Census()
00092                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Census API Call-----\n")
00093             elif event == "Authorization Utility":
00094                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Initiating Authorization Utility-----")
00095                 AuthUtil()
00096                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Closing Authorization
Utility-----\n")
00097             elif event == "Open Data Folder":
00098                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | -----Data Folder Opened-----")
00099                 try:
00100                     os.system(f"start
{Path(os.path.expanduser('~\Documents')).joinpath('GardnerUtilData')}")
00101                 except:
00102                     try:
00103                         os.system(f"start
{Path(os.path.expanduser('~\Documents'))}")
00104                     except Exception as e:
00105                         print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
00106                         PopupWrapped(
00107                             text="Documents folder not found. Please create a
Windows recognized documents folder",
00108                             windowType="errorLarge")
00109
00110             elif event in ('Exit', None):
00111                 try:
00112                     break
00113                 except Exception as e:
00114                     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.")
00115                     break
00116             elif event == sg.WIN_CLOSED or event == "Quit":
00117                 break
00118
00119         window.close()
00120

```

```

00121     def __CreateFrame(self):
00122
00123         """
00124         The __CreateFrame function is a helper function that creates the layout for the
00125         main window.
00126         It returns a list of lists, which is then passed to sg.Window() as its layout
00127         parameter.
00128         Args:
00129             self: Represent the instance of the class
00130         Returns:
00131             A list of lists, which is then passed to the sg
00132
00133         Doc Author:
00134             Willem van der Schans, Trelent AI
00135         """
00136         sg.theme('Default1')
00137
00138         line0 = [sg.HSeparator()]
00139
00140         line1 = [sg.Image(ImageLoader("logo.png")),
00141                 sg.Push(),
00142                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"),
00143                        justification="center"),
00144                 sg.Push(),
00145                 sg.Push()]
00146
00147         line3 = [sg.HSeparator()]
00148
00149         line4 = [sg.Push(),
00150                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"),
00151                        justification="center"),
00152                 sg.Push()]
00153
00154         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)),
00155                 sg.Push(),
00156                 sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00157
00158         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
00159                 sg.Button("Census", size=(20, None)),
00160                 sg.Push()]]
00161
00162         line8 = [sg.HSeparator()]
00163
00164         line9 = [sg.Push(),
00165                 sg.Text("Utilities", font=("Helvetica", 10, "bold"),
00166                        justification="center"),
00167                 sg.Push()]
00168
00169         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00170                 sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00171
00172         line11 = [sg.HSeparator()]
00173
00174         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
00175                 line11]
00176
00177         return layout

```

CFBP/Core.py

```
00001 import threading
00002 from datetime import date
00003
00004 import pandas as pd
00005 import requests
00006
00007 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00008 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00009 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00010
00011
00012 class Census:
00013
00014     def __init__(self, state_arg=None, year_arg=None):
00015         """
00016         The __init__ function is called when the class is instantiated.
00017         It's job is to initialize the object with some default values, and do any other
00018         setup that might be necessary.
00019         The __init__ function can take arguments, but it doesn't have to.
00020
00021         Args:
00022             self: Represent the instance of the class
00023             state_arg: Set the state_arg attribute of the class
00024             year_arg: Set the year of data to be retrieved
00025
00026         Returns:
00027             A popupwrapped object
00028
00029         Doc Author:
00030             Willem van der Schans, Trelent AI
00031
00032         """
00033         self.state_arg = state_arg
00034         self.year_arg = year_arg
00035         self.uiString = None
00036         self.link = None
00037
00038         self.__showUi()
00039         print(self.link)
00040         F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00041         self.uiString = (
00042             f"ffiec.cfbp.gov (Mortgage API) request Completed \n {self.year_arg}
00043             data retrieved \n Data Saved at {F.getPath()}")
00044
00045         PopupWrapped(text=self.uiString, windowType="noticeLarge")
00046
00047     def __showUi(self):
00048
00049         """
00050         The __showUi function is a function that creates a progress bar window.
00051         The __showUi function takes class variables and returns a windowobj.
00052
00053         Args:
00054             self: Represent the instance of the class
00055
00056         Returns:
00057             The uiobj variable
00058
00059         Doc Author:
00060             Willem van der Schans, Trelent AI
00061
00062         """
00063         uiObj = PopupWrapped(text="Cenus Request running", windowType="progress",
00064                               error=None)
00065
00066         threadGui = threading.Thread(target=self.__dataGetter,
00067                                       daemon=False)
00068         threadGui.start()
00069
00070         while threadGui.is_alive():
00071             uiObj.textUpdate()
00072             uiObj.windowPush()
00073         else:
00074             uiObj.stopWindow()
```

```

00071
00072     def __dataGetter(self):
00073         """
00074         The __dataGetter function is a private function that gets the data from the CFPB
00075         API.
00076         It takes no arguments, but uses self.state_arg and self.year_arg to create a URL
00077         for the API call.
00078         Args:
00079             self: Represent the instance of the class
00080         Returns:
00081             A response object
00082         Doc Author:
00083             Willem van der Schans, Trelent AI
00084         """
00085         arg_dict_bu = locals()
00086
00087         link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
00088
00089         if self.state_arg is None:
00090             self.state_arg = "UT"
00091         else:
00092             pass
00093
00094         if self.year_arg is None:
00095             self.year_arg = str(date.today().year - 1)
00096         else:
00097             pass
00098
00099         passFlag = False
00100
00101         while not passFlag:
00102
00103             self.link = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" +
00104             f"states={self.state_arg}" + f"&years={self.year_arg}"
00105
00106             response = requests.get(self.link)
00107
00108             if response.status_code == 400:
00109                 self.year_arg = int(self.year_arg) - 1
00110
00111             else:
00112                 passFlag = True
00113
00114         RESTError(response)
00115         raise SystemExit(0)

```

ConstructionMonitor/Core.py

```
00001 import copy
00002 import json
00003 import os
00004 import threading
00005 import time
00006 from datetime import date, timedelta
00007 from pathlib import Path
00008 import datetime
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00018 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00019 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00020 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00021 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00022
00023
00024 class ConstructionMonitorInit:
00025
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the variables that will be used by other functions in this class.
00031
00032         Args:
00033             self: Represent the instance of the class
00034
00035         Returns:
00036             None
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
00042         self.size = None
00043         self.SourceInclude = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.rest_domain = None
00047         self.auth_key = None
00048         self.ui_flag = None
00049         self.append_file = None
00050
00051         passFlag = False
00052
00053         while not passFlag:
00054             if
00055 os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00056             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
00057 Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00058             "Security").joinpath("auth.json")):
00059                 try:
00060                     f =
00061 open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00062                     "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00063                     key = f.readline()
00064                     f.close()
00065                     f =
00066 open(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00067                     "Security").joinpath("auth.json"), "rb")
00068                     authDict = json.load(f)
00069                     fernet = Fernet(key)
00070                     self.auth_key =
00071 fernet.decrypt(authDict["cm"]["auth"]).decode()
00072                     passFlag = True
```

```

00069         except Exception as e:
00070             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | ConstructionMonitor/Core.py | Error = {e} | Auth.json not found
opening AuthUtil")
00071             AuthUtil()
00072         else:
00073             AuthUtil()
00074
00075     self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00076
00077     def __ShowGui(self, layout, text):
00078
00079         """
00080         The __ShowGui function is the main function that creates and displays the GUI.
00081         It takes in a layout, which is a list of lists containing all the elements to
be displayed on screen.
00082         The text parameter specifies what title should appear at the top of the window.
00083
00084         Args:
00085             self: Refer to the current instance of a class
00086             layout: Determine what the gui will look like
00087             text: Set the title of the window
00088
00089         Returns:
00090             A dictionary of values
00091
00092         Doc Author:
00093             Willem van der Schans, Trelent AI
00094         """
00095         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00096                             finalize=True,
00097                             icon=ImageLoader("taskbar_icon.ico"))
00098
00099         while True:
00100             event, values = window.read()
00101
00102             if event == "Submit":
00103                 try:
00104                     self.__SetValues(values)
00105                     break
00106                 except Exception as e:
00107                     print(e)
00108                     RESTError(993)
00109                     raise SystemExit(933)
00110             elif event == sg.WIN_CLOSED or event == "Quit":
00111                 break
00112
00113         window.close()
00114
00115     @staticmethod
00116     def __CreateFrame():
00117
00118         """
00119         The __CreateFrame function creates the GUI layout for the application.
00120         The function returns a list of lists that contains all the elements to be
displayed in the GUI window.
00121         This is done by creating each line as a list and then appending it to another
list which will contain all lines.
00122
00123         Args:
00124
00125         Returns:
00126             The layout for the gui
00127
00128         Doc Author:
00129             Willem van der Schans, Trelent AI
00130         """
00131         sg.theme('Default1')
00132
00133         line00 = [sg.HSeparator()]
00134
00135         line0 = [sg.Image(ImageLoader("logo.png")),
00136                 sg.Push(),
00137                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00138                 sg.Push(),

```

```

00139         sg.Push()]
00140
00141         line1 = [sg.HSeparator()]
00142
00143         line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00144                 sg.Input(default_text=(date.today() -
00145                                timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
00146                                size=(20, 1)),
00147                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00148                                key="-start_date-", target="-Cal-")]
00149
00150         line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00151                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"),
00152                                size=(20, 1)),
00153                 sg.CalendarButton("Select Date", format="%Y-%m-%d",
00154                                key="-start_date-", target="-EndCal-")]
00155
00156         line5 = [sg.HSeparator()]
00157
00158         line6 = [sg.Push(),
00159                 sg.Text("File Settings", font=("Helvetica", 12, "bold"),
00160                        justification="center"),
00161                 sg.Push()]
00162
00163         line7 = [sg.HSeparator()]
00164
00165         line8 = [sg.Text("Appending File : ", size=(15, None),
00166                        justification="Right"),
00167                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00168                        size=(20, 1)),
00169                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")],
00170                        key="-append_file-",
00171                        target="-AppendingFile-")]
00172
00173         line9 = [sg.HSeparator()]
00174
00175         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00176
00177         layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8,
00178                 line9, line10]
00179
00180         return layout
00181
00182     def __SetValues(self, values):
00183         """
00184         The __SetValues function is used to set the values of the variables that are used
00185         in the __GetData function.
00186         The __SetValues function takes a dictionary as an argument, and then sets each
00187         variable based on what is passed into
00188         the dictionary. The keys for this dictionary are defined by the user when they
00189         create their own instance of this class.
00190
00191         Args:
00192             self: Represent the instance of the class
00193             values: Pass in the values from the ui
00194
00195         Returns:
00196             A dictionary of values
00197
00198         Doc Author:
00199             Willem van der Schans, Trelent AI
00200
00201         """
00202         self.size = 1000
00203
00204         if values["-Cal-"] != "":
00205             self.dateStart = values["-Cal-"]
00206         else:
00207             self.dateStart = (date.today() -
00208                              timedelta(days=14)).strftime("%Y-%m-%d")
00209
00210         if values["-EndCal-"] != "":
00211             self.dateEnd = values["-EndCal-"]
00212         else:
00213             self.dateEnd = date.today().strftime("%Y-%m-%d")
00214

```



```

00204     self.rest_domain = "https://api.constructionmonitor.com/v2/powersearch/?"
00205
00206     self.SourceInclude = None
00207
00208     if values["-append_file-"] != "":
00209         self.append_file = str(values["-append_file-"])
00210     else:
00211         self.append_file = None
00212
00213     self.ui_flag = True
00214
00215
00216 class ConstructionMonitorMain:
00217
00218     def __init__(self, siteClass):
00219
00220         """
00221         The __init__ function is the first function that runs when an object of this class
00222         is created.
00223         It sets up all the variables and functions needed for this class to run properly.
00224
00225         Args:
00226             self: Represent the instance of the class
00227             siteClass: Identify the site that is being used
00228
00229         Returns:
00230             Nothing
00231
00232         Doc Author:
00233             Willem van der Schans, Trelent AI
00234         """
00235         self._siteClass = siteClass
00236         self._restDomain = None
00237         self._headerDict = None
00238         self._columnSelection = None
00239         self._appendFile = None
00240
00241         self._parameterDict = {}
00242         self._search_id = None
00243         self._record_val = 0
00244         self._batches = 0
00245
00246         self._ui_flag = None
00247
00248         self.dataframe = None
00249
00250         try:
00251             self.mainFunc()
00252         except SystemError as e:
00253             if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message',
repr(e))):
00254                 print(
00255                     f"ConstructionMonitor/Core.py | Error = {e} | Coerced
SystemError in ConstructionMonitorMain class")
00256                 pass
00257             except AttributeError as e:
00258                 # This allows for user cancellation of the program using the quit button
00259                 if "'NoneType' object has no attribute 'json'" in str(getattr(e,
'message', repr(e))):
00260                     RESTError(1101)
00261                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Error {e}")
00262                     pass
00263                 elif e is not None:
00264                     print(
00265                         f"ConstructionMonitor/Core.py | Error = {e} | Authentication
Error | Please update keys in AuthUtil")
00266                     RESTError(401)
00267                     print(e)
00268                     pass
00269                 else:
00270                     pass
00271             except Exception as e:
00272                 print(e)
00273                 RESTError(1001)
00274                 raise SystemExit(1001)

```

```

00275
00276     def mainFunc(self):
00277         """
00278         The mainFunc function is the main function of this module. It will be called by
the GUI or CLI to execute
00279         the code in this module. The mainFunc function will first create a parameter
dictionary using the __ParameterCreator
00280         method, then it will get a count of all records that match its parameters using
the __getCountUI method, and then
00281         it will calculate how many batches are needed to retrieve all records with those
parameters using BatchCalculator.
00282         After that it asks if you want to continue with retrieving data from Salesforce
(if running in GUI mode). Then it shows
00283         a progress bar for each
00284
00285         Args:
00286             self: Refer to the current object
00287
00288         Returns:
00289             The dataframe
00290
00291         Doc Author:
00292             Willem van der Schans, Trelent AI
00293         """
00294         self.__ParameterCreator()
00295
00296         self.__getCountUI()
00297
00298         self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)
00299         if self.__batches != 0:
00300             startTime = datetime.datetime.now().replace(microsecond=0)
00301             BatchInputGui(self.__batches)
00302             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Request for {self.__batches} Batches sent to server")
00303             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00304
ParameterDict=self.__parameterDict,
00305                                     HeaderDict=self.__headerDict,
00306
ColumnSelection=self.__columnSelection,
00307                                     BatchesNum=self.__batches,
00308                                     Type="construction_monitor")
00309             BatchGuiObject.BatchGuiShow()
00310             self.dataframe = BatchGuiObject.dataframe
00311             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')}[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00312             FileSaver("cm", self.dataframe, self.__appendFile)
00313         else:
00314             RESTError(994)
00315             raise SystemExit(994)
00316
00317     def __ParameterCreator(self):
00318         """
00319         The __ParameterCreator function is used to create the parameter dictionary that
will be passed into the
00320         __Request function. The function takes in a siteClass object and extracts
all of its attributes, except for
00321         those that start with '__' or are callable. It then creates a dictionary from
these attributes and stores it as
00322         self.__parameterDict.
00323
00324         Args:
00325             self: Make the function a method of the class
00326
00327         Returns:
00328             A dictionary of parameters and a list of non parameter variables
00329
00330         Doc Author:
00331             Willem van der Schans, Trelent AI
00332         """
00333         __Source_dict = {key: value for key, value in
self.__siteClass.__dict.items() if
00334                             not key.startswith('__') and not callable(key)}
00335

```

```

00336     self.__restDomain = __Source_dict["rest_domain"]
00337     __Source_dict.pop("rest_domain")
00338     self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00339     __Source_dict.pop("auth_key")
00340     self.__columnSelection = __Source_dict["SourceInclude"]
00341     __Source_dict.pop("SourceInclude")
00342     self.__ui_flag = __Source_dict["ui_flag"]
00343     __Source_dict.pop("ui_flag")
00344     self.__appendFile = __Source_dict["append_file"]
00345     __Source_dict.pop("append_file")
00346
00347     temp_dict = copy.copy(__Source_dict)
00348     for key, value in temp_dict.items():
00349         if value is None:
00350             __Source_dict.pop(key)
00351         else:
00352             pass
00353
00354     self.__parameterDict = copy.copy(__Source_dict)
00355
00356     def __getCount(self):
00357         """
00358         The __getCount function is used to get the total number of records that are
00359         returned from a query.
00360         This function is called by the __init__ function and sets the self.__record_val
00361         variable with this value.
00362
00363         Args:
00364             self: Represent the instance of the class
00365
00366         Returns:
00367             The total number of records in the database
00368
00369         Doc Author:
00370             Willem van der Schans, Trelent AI
00371         """
00372         __count_resp = None
00373
00374         try:
00375             __temp_param_dict = copy.copy(self.__parameterDict)
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
00405     def __getCountUI(self):
00406         """
00407         The __getCountUI function is a wrapper for the __getCount function.
00408         It allows the user to run __getCount in a separate thread, so that they can
00409         continue working while it runs.

```

```

00410     The function will display a progress bar and update with text as it progresses
00411     through its tasks.
00412     Args:
00413         self: Access the class variables and methods
00414     Returns:
00415         The count of the number of records in the database
00416     Doc Author:
00417         Willem van der Schans, Trelent AI
00418     """
00419     if self.__ui_flag:
00420         uiObj = PopupWrapped(text="Batch request running",
00421                               windowType="progress", error=None)
00422         threadGui = threading.Thread(target=self.__getCount,
00423                                       daemon=False)
00424         threadGui.start()
00425         while threadGui.is_alive():
00426             uiObj.textUpdate()
00427             uiObj.windowPush()
00428         else:
00429             uiObj.stopWindow()
00430     else:
00431         self.__getCount()
00432

```

Realtor/Core.py

```
00001 import threading
00002
00003 import pandas as pd
00004 import requests
00005 from bs4 import *
00006
00007 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00008 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00009 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00010
00011
00012 class realtorCom:
00013
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
00018         to run before anything else in your class.
00019
00020         Args:
00021             self: Represent the instance of the class
00022
00023         Returns:
00024             A new object
00025
00026         Doc Author:
00027             Willem van der Schans, Trelent AI
00028         """
00029         self.__page_html = None
00030         self.__update_date = None
00031         self.__last_date = None
00032         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00033         self.__linkDict = {}
00034         self.__dfState = None
00035         self.__dfCounty = None
00036         self.__dfZip = None
00037         self.__uiString = "Files Saved to \n"
00038
00039         page_html = requests.get("https://www.realtor.com/research/data/").text
00040         self.__page_html = BeautifulSoup(page_html, "html.parser")
00041
00042         self.__linkGetter()
00043         self.__showUi()
00044
00045         PopupWrapped(text=self.__uiString, windowType="noticeLarge")
00046
00047     def __showUi(self):
00048         """
00049         The __showUi function is a helper function that creates and displays the progress
00050         window.
00051         It also starts the dataUpdater thread, which will update the progress bar as it
00052         runs.
00053
00054         Args:
00055             self: Represent the instance of the class
00056
00057         Returns:
00058             A popupwrapped object
00059
00060         Doc Author:
00061             Willem van der Schans, Trelent AI
00062         """
00063         uiObj = PopupWrapped(text="Request running", windowType="progress",
00064                               error=None)
00065
00066         threadGui = threading.Thread(target=self.__dataUpdater,
00067                                       daemon=False)
00068         threadGui.start()
00069
00070         while threadGui.is_alive():
00071             uiObj.textUpdate()
```

```

00070         uiObj.windowPush()
00071     else:
00072         uiObj.stopWindow()
00073
00074     def __linkGetter(self):
00075         """
00076         The __linkGetter function is a private function that takes the idDict dictionary
00077         and adds
00078         a link to each entry in the dictionary. The link is used to access historical
00079         data for each
00080         scope symbol.
00081         Args:
00082             self: Refer to the object itself
00083         Returns:
00084             A dictionary of all the links to the history pages
00085         Doc Author:
00086             Willem van der Schans, Trelent AI
00087         """
00088         for key, value in self.__idDict.items():
00089             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00090                 try:
00091                     for nestedRow in row.find_all("a"):
00092                         if "History" in str(nestedRow.get("href")) and key in
00093                             str(nestedRow.get("href")):
00094                                 self.__idDict[key] = {"id": value, "link":
00095                                     nestedRow.get("href")}
00096             except Exception as e:
00097                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
00098 %H:%M:%S.%f')}[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
00099 for realtor.com")
00100                 RESTError(801)
00101                 raise SystemExit(801)
00102
00103     def __dataUpdater(self):
00104         """
00105         The __dataUpdater function is a private function that updates the dataframes for
00106         each of the three
00107         types of realtor data. It takes class variables and return the path to the
00108         saved file. The function first creates an empty
00109         dictionary called tempdf, then iterates through each key in self.__idDict
00110         (which contains all three ids).
00111         For each key, it reads in a csv file from the link associated with that id
00112         and saves it to tempdf as a pandas
00113         DataFrame object. Then, depending on which type of realtor data we are dealing
00114         with (State/County/Zip), we save
00115         Args:
00116             self: Access the attributes and methods of the class
00117         Returns:
00118             The path of the saved file
00119         Doc Author:
00120             Willem van der Schans, Trelent AI
00121         """
00122         for key, value in self.__idDict.items():
00123             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00124             if key == "State":
00125                 self.dfState = tempdf
00126             elif key == "County":
00127                 self.dfCounty = tempdf
00128             elif key == "Zip":
00129                 self.dfZip = tempdf
00130
00131         FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00132         self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"

```

UtahRealEstate/Core.py

```
00001 import copy
00002 import datetime
00003 import json
00004 import os
00005 import threading
00006 import time
00007 from datetime import date, timedelta
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00018 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00019 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00020 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00021 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00022
00023
00024 class UtahRealEstateInit:
00025
00026     def __init__(self):
00027
00028         """
00029         The __init__ function is called when the class is instantiated.
00030         It sets up the initial state of the object.
00031
00032         Args:
00033             self: Represent the instance of the class
00034
00035         Returns:
00036             The __createframe function
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
00042         self.StandardStatus = None
00043         self.ListedOrModified = None
00044         self.dateStart = None
00045         self.dateEnd = None
00046         self.select = None
00047         self.file_name = None
00048         self.append_file = None
00049
00050         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00051
00052     def __ShowGui(self, layout, text):
00053
00054         """
00055         The __ShowGui function is a helper function that creates the GUI window and
00056         displays it to the user.
00057         It takes in two parameters: layout, which is a list of lists containing all the
00058         elements for each row;
00059         and text, which is a string containing what will be displayed as the title of
00060         the window. The __ShowGui
00061         method then uses these parameters to create an instance of sg.Window with all
00062         its attributes set accordingly.
00063
00064         Args:
00065             self: Refer to the current class instance
00066             layout: Pass the layout of the window to be created
00067             text: Set the title of the window
00068
00069         Returns:
00070             A dictionary of values
00071
00072         Doc Author:
00073             Willem van der Schans, Trelent AI
```

```

00070     """
00071     window = sg.Window(text, layout, grab_anywhere=False,
00072         return_keyboard_events=True,
00073         finalize=True,
00074         icon=ImageLoader("taskbar_icon.ico"))
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
00092     @staticmethod
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
00097         displayed in the window.
00098         Each element is defined by its type and any additional parameters needed to
00099         define it.
00100
00101         Args:
00102             Returns:
00103                 A list of lists, which is used to create the gui
00104
00105         Doc Author:
00106             Willem van der Schans, Trelent AI
00107         """
00108         sg.theme('Default1')
00109
00110         line00 = [sg.HSeparator()]
00111
00112         line0 = [sg.Image(ImageLoader("logo.png")),
00113             sg.Push(),
00114             sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
00115                 "bold"), justification="center"),
00116             sg.Push(),
00117             sg.Push()]
00118
00119         line1 = [sg.HSeparator()]
00120
00121         line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00122             sg.DropDown(default_value="Active", values=["Active", "Closed"],
00123                 key="-status-", size=(31, 1))]
00124
00125         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00126             sg.DropDown(default_value="Listing Date", values=["Listing
00127                 Date", "Modification Date", "Close Date"],
00128                 key="-type-", size=(31, 1))]
00129
00130         line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00131             sg.Input(default_text=(date.today() -
00132                 timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00133                 disabled=False, size=(20, 1)),
00134             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00135                 key="-start_date-", target="-DateStart-")]
00136
00137         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00138             sg.Input(default_text=(date.today()).strftime("%Y-%m-%d"),
00139                 key="-DateEnd-", disabled=False,
00140                 size=(20, 1)),
00141             sg.CalendarButton("Select Date", format="%Y-%m-%d",
00142                 key="-end_date-", target="-DateEnd-")]
00143

```



```

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,
size=(20, 1)),
00150                  sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
                                target="-AppendingFile-")]
00151
00152         line11 = [sg.HSeparator()]
00153
00154         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00155
00156         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10, line11,
00159                 line12]
00160
00161         return layout
00162
00163     def __SetValues(self, values):
00164         """
00165         The __SetValues function is used to set the values of the variables that are used
in the
00167         __GetData function. The values are passed from a dictionary called 'values'
which is created
00168         by parsing through an XML file using ElementTree. This function also sets
default values for
00169         some of these variables if they were not specified in the XML file.
00170
00171         Args:
00172             self: Represent the instance of the class
00173             values: Pass the values from the gui to this function
00174
00175         Returns:
00176             A dictionary with the following keys:
00177
00178         Doc Author:
00179             Willem van der Schans, Trelent AI
00180         """
00181         self.StandardStatus = values["-status-"]
00182
00183         self.ListedOrModified = values["-type-"]
00184
00185         if values["-DateStart-"] != "":
00186             self.dateStart = values["-DateStart-"]
00187         else:
00188             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00189
00190         if values["-DateEnd-"] != "":
00191             self.dateEnd = values["-DateEnd-"]
00192         else:
00193             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00194
00195         if values['-selectionFlag-']:
00196             self.select =
"ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,Subdivisi
onName," \
00197             "StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDate,"
\

```

```

00198 "ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,LotSi
00199 zeAcres," \
00200 "LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,Park
00201 ingTotal," \
00202 "BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffective,
00203 " \
00204 "OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseContract
00205 Date," \
00206 "AssociationFee,AssociationFeeFrequency,OccupantType,PropertySubType,PropertyType," \
00207 "StandardStatus,BuyerFinancing"
00208 else:
00209     self.select = None
00210     if values["-append_file-"] != "":
00211         self.append_file = str(values["-append_file-"])
00212     else:
00213         self.append_file = None
00214
00215 class UtahRealEstateMain:
00216     def __init__(self, siteClass):
00217         """
00218         The __init__ function is the first function that runs when an object of this class
00219         is created.
00220         It sets up all the variables and functions needed for this class to work properly.
00221         Args:
00222             self: Represent the instance of the class
00223             siteClass: Determine which site to pull data from
00224         Returns:
00225             Nothing
00226         Doc Author:
00227             Willem van der Schans, Trelent AI
00228         """
00229         self.dataframe = None
00230         self.batches = 0
00231         self.siteClass = siteClass
00232         self.headerDict = None
00233         self.parameterString = ""
00234         self.appendFile = None
00235         self.dateStart = None
00236         self.dateEnd = None
00237         self.restDomain =
00238         'https://resoapi.utahrealestate.com/reso/odata/Property?'
00239         self.keyPath =
00240         Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00241             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00242         self.filePath =
00243         Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00244             "Security").joinpath("auth.json")
00245         self.key = None
00246
00247         try:
00248             self.mainFunc()
00249         except KeyError as e:
00250             # This allows for user cancellation of the program using the quit button
00251             if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00252                 RESTError(1101)
00253                 print(e)
00254                 pass
00255         except AttributeError as e:
00256             if e is not None:
00257                 print(
00258                     f"UtahRealEstate/Core.py | Error = {e} | Authentication Error
00259                     | Please update keys in AuthUtil")
00260                 RESTError(401)
00261                 pass
00262         else:

```

```

00261         pass
00262     except Exception as e:
00263         print(e)
00264         RESTError(1001)
00265         raise SystemExit(1001)
00266
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()
00302
00303             self.\_\_ParameterCreator()
00304
00305             self.\_\_getCountUI()
00306
00307             self.\_\_batches = BatchCalculator(self.\_\_record\_val, None)
00308
00309             if self.\_\_batches != 0:
00310                 startTime = datetime.datetime.now\(\).replace\(microsecond=0\)
00311                 BatchInputGui(self.\_\_batches)
00312                 print(f"{datetime.datetime.today\(\).strftime\('%m-%d-%Y %H:%M:%S.%f'\)\[:-3\]} | Request for {self.\_\_batches} Batches sent to server")
00313                 BatchGuiObject = BatchProgressGUI(RestDomain=self.\_\_restDomain,
00314
00315                 ParameterDict=self.\_\_parameterString,
00316
00317                 HeaderDict=self.\_\_headerDict,
00318                 BatchesNum=self.\_\_batches,
00319                 Type="utah_real_estate")
00318                 BatchGuiObject.BatchGuiShow()
00319                 self.\_\_dataframe = BatchGuiObject.dataframe
00320                 print(
00321                     f"{datetime.datetime.today\(\).strftime\('%m-%d-%Y %H:%M:%S.%f'\)\[:-3\]} | Dataframe retrieved with {self.\_\_dataframe.shape[0]} rows and
{self.\_\_dataframe.shape[1]} columns in {time.strftime\('%H:%M:%S',
time.gmtime\(\(datetime.datetime.now\\(\\).replace\\(microsecond=0\\) -
startTime\).total\_seconds\(\)\)}\)"}"\)
00322                 FileSaver\("ure", self.\\_\\_dataframe, self.\\_\\_appendFile\)
00323             else:
00324                 RESTError\(994\)
00325                 raise SystemExit\(994\)
00326

```

```

00327     def __ParameterCreator(self):
00328         """
00329         The __ParameterCreator function is used to create the filter string for the ReST
00330         API call.
00331         The function takes in a siteClass object and extracts all of its parameters into
00332         a dictionary.
00333         It then creates an appropriate filter string based on those parameters.
00334         Args:
00335             self: Bind the object to the class
00336         Returns:
00337             A string to be used as the parameter in the api call
00338         Doc Author:
00339             Willem van der Schans, Trelent AI
00340         """
00341         filter_string = ""
00342         __Source_dict = {key: value for key, value in
00343 self.__siteClass.__dict__.items() if
00344 not key.startswith(' ') and not callable(key)}
00345
00346         self.__appendFile = __Source_dict["append_file"]
00347         __Source_dict.pop("append_file")
00348
00349         temp_dict = copy.copy(__Source_dict)
00350         for key, value in temp_dict.items():
00351             if value is None:
00352                 __Source_dict.pop(key)
00353             else:
00354                 pass
00355
00356         if __Source_dict["ListedOrModified"] == "Listing Date":
00357             filter_string =
00358 f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContr
00359 actDate%20le%20{__Source_dict['dateEnd']}"
00360         elif __Source_dict["ListedOrModified"] == "Modification Date":
00361             filter_string =
00362 f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%2
00363 0ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00364         elif __Source_dict["ListedOrModified"] == "Close Date":
00365             filter_string =
00366 f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__So
00367 urce_dict['dateEnd']}"
00368
00369         filter_string = filter_string +
00370 f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['Standard
00371 Status']}'"
00372
00373         if __Source_dict["select"] is not None:
00374             filter_string = filter_string + f'&$select={__Source_dict["select"]}'
00375
00376         self.__parameterString = filter_string
00377
00378     def __getCount(self):
00379         """
00380         The __getCount function is used to determine the number of records that will be
00381         returned by the query.
00382         This function is called when a user calls the count() method on a ReST object.
00383         The __getCount function uses
00384         the $count parameter in OData to return only an integer value representing how
00385         many records would be returned
00386         by the query.
00387         Args:
00388             self: Represent the instance of the class
00389         Returns:
00390             The number of records in the data set
00391         Doc Author:
00392             Willem van der Schans, Trelent AI
00393         """
00394         __count_resp = None
00395
00396         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
00415     def __getCountUI(self):
00416
00417         """
00418         The __getCountUI function is a wrapper for the __getCount function.
00419         It creates a progress window and updates it while the __getCount function runs.
00420         The purpose of this is to keep the GUI responsive while running long processes.
00421
00422         Args:
00423             self: Represent the instance of the class
00424
00425         Returns:
00426             A popupwrapped object
00427
00428         Doc Author:
00429             Willem van der Schans, Trelent AI
00430         """
00431         uiObj = PopupWrapped(text="Batch request running", windowType="progress",
error=None)
00432
00433         threadGui = threading.Thread(target=self.__getCount,
00434                                     daemon=False)
00435         threadGui.start()
00436
00437         while threadGui.is_alive():
00438             uiObj.textUpdate()
00439             uiObj.windowPush()
00440         else:
00441             uiObj.stopWindow()

```

Index

- __appendFile
 - Core.ConstructionMonitorMain, 54
 - Core.UtahRealEstateMain, 96
- __batch_counter
 - BatchProgressGUI.BatchProgressGUI, 35
- __batches
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 54
 - Core.UtahRealEstateMain, 96
- __columnSelection
 - BatchProcessing.BatchProcessorConstructionMonitor, 21
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 54
- __counter
 - PopupWrapped.PopupWrapped, 74
- __CreateFrame
 - API_Calls.Initializer.initializer, 64
 - AuthUtil.AuthUtil, 11
 - Core.ConstructionMonitorInit, 42
 - Core.UtahRealEstateInit, 83
- __createLayout
 - PopupWrapped.PopupWrapped, 70
- __createWindow
 - PopupWrapped.PopupWrapped, 71
- __dataGetter
 - Core.Cencus, 37
- __dataUpdater
 - Core.realtorCom, 77
- __dateEnd
 - Core.UtahRealEstateMain, 97
- __dateStart
 - Core.UtahRealEstateMain, 97
- __dateTracker
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
- __error
 - PopupWrapped.PopupWrapped, 74
- __getCount
 - Core.ConstructionMonitorMain, 50
 - Core.UtahRealEstateMain, 92
- __getCountUI
 - Core.ConstructionMonitorMain, 51
 - Core.UtahRealEstateMain, 93
- __headerDict
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
 - BatchProcessing.BatchProcessorUtahRealEstate, 26
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 54
 - Core.UtahRealEstateMain, 97
- __idDict
 - Core.realtorCom, 80
- __init
 - API_Calls.Initializer.initializer, 63
 - AuthUtil.AuthUtil, 9
 - BatchProcessing.BatchProcessorConstructionMonitor, 18
 - BatchProcessing.BatchProcessorUtahRealEstate, 23
 - BatchProgressGUI.BatchProgressGUI, 27
 - Core.Cencus, 36
 - Core.ConstructionMonitorInit, 40
 - Core.ConstructionMonitorMain, 48
 - Core.realtorCom, 76
 - Core.UtahRealEstateInit, 82
 - Core.UtahRealEstateMain, 90
 - DataTransfer.DataTransfer, 56
 - FileSaver.FileSaver, 59
 - PopupWrapped.PopupWrapped, 69
- __init__.py, 99
- __last_date
 - Core.realtorCom, 80
- __layout
 - BatchProgressGUI.BatchProgressGUI, 35
 - PopupWrapped.PopupWrapped, 74
- __linkDict
 - Core.realtorCom, 80
- __linkGetter
 - Core.realtorCom, 78
- __maxRequests
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
- __numBatches
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
 - BatchProcessing.BatchProcessorUtahRealEstate, 26
- __page_html
 - Core.realtorCom, 80
- __ParameterCreator
 - Core.ConstructionMonitorMain, 52
 - Core.UtahRealEstateMain, 94
- __parameterDict
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 55
- __parameterString
 - BatchProcessing.BatchProcessorUtahRealEstate, 26
 - Core.UtahRealEstateMain, 97
- __record_val
 - Core.ConstructionMonitorMain, 55
 - Core.UtahRealEstateMain, 97
- __requestCalls
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
- __requestCount
 - BatchProcessing.BatchProcessorConstructionMonitor, 22

- __restDomain
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
 - BatchProcessing.BatchProcessorUtahRealEstate, 26
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 55
 - Core.UtahRealEstateMain, 97
- __search_id
 - Core.ConstructionMonitorMain, 55
- __SetValues
 - AuthUtil.AuthUtil, 13
 - Core.ConstructionMonitorInit, 44
 - Core.UtahRealEstateInit, 85
- __ShowGui
 - API_Calls.Initializer.initializer, 65
 - AuthUtil.AuthUtil, 15
 - Core.ConstructionMonitorInit, 45
 - Core.UtahRealEstateInit, 87
- __showUi
 - Core.Cencus, 38
 - Core.realtorCom, 79
- __siteClass
 - Core.ConstructionMonitorMain, 55
 - Core.UtahRealEstateMain, 97
- __text
 - PopupWrapped.PopupWrapped, 74
- __thread
 - PopupWrapped.PopupWrapped, 75
- __type
 - BatchProgressGUI.BatchProgressGUI, 35
 - PopupWrapped.PopupWrapped, 75
- __ui_flag
 - Core.ConstructionMonitorMain, 55
- __update_date
 - Core.realtorCom, 80
- __value
 - DataTransfer.DataTransfer, 58
- __window
 - BatchProgressGUI.BatchProgressGUI, 35
- __windowObj
 - PopupWrapped.PopupWrapped, 75
- __main_.py, 100
- API_Calls.Initializer.initializer, 63
 - __CreateFrame, 64
 - __init__, 63
 - __ShowGui, 65
 - __classObj, 68
- append_file
 - AuthUtil.AuthUtil, 16
 - Core.ConstructionMonitorInit, 46
 - Core.UtahRealEstateInit, 88
- appendFlag
 - FileSaver.FileSaver, 61
- auth_key
 - Core.ConstructionMonitorInit, 46
- AuthUtil.AuthUtil, 9
 - __CreateFrame, 11
 - __init__, 9
 - __SetValues, 13
 - __ShowGui, 15
 - append_file, 16
 - file_name, 16
 - filePath, 16
 - jsonDict, 16
 - k, 17
 - keyFlag, 17
 - keyPath, 17
 - ListedOrModified, 17
 - outcomeText, 17
 - passFlagCm, 17
 - passFlagUre, 17
 - StandardStatus, 17
- AuthUtil.py, 101
- BatchGui.py, 121
- BatchGuiShow
 - BatchProgressGUI.BatchProgressGUI, 28
- BatchProcessing.BatchProcessorConstructionMonitor, 18
 - __columnSelection, 21
 - __dateTracker, 22
 - __headerDict, 22
 - __init__, 18
 - __maxRequests, 22
 - __numBatches, 22
 - __parameterDict, 22
 - __requestCalls, 22
 - __requestCount, 22
 - __restDomain, 22
 - ConstructionMonitorProcessor, 19
 - dataframe, 22
 - FuncSelector, 21
 - valueObject, 22
- BatchProcessing.BatchProcessorUtahRealEstate, 23
 - __headerDict, 26
 - __init__, 23
 - __numBatches, 26
 - __parameterString, 26
 - __restDomain, 26
 - BatchProcessingUtahRealestateCom, 24
 - dataframe, 26
 - FuncSelector, 25
 - valueObject, 26
- BatchProcessing.py, 106
- BatchProcessingUtahRealestateCom
 - BatchProcessing.BatchProcessorUtahRealEstate, 24
- BatchProgressGUI.BatchProgressGUI, 27
 - __batch_counter, 35
 - __batches, 35
 - __columnSelection, 35
 - __headerDict, 35
 - __init__, 27
 - __layout, 35
 - __parameterDict, 35
 - __restDomain, 35
 - __type, 35
 - __window, 35
- BatchGuiShow, 28

- createGui, 29
- CreateProgressLayout, 31
- dataframe, 35
- ProgressUpdater, 32
- TimeUpdater, 33
- ValueChecker, 34
- BatchProgressGUI.py, 122
- classObj
 - API_Calls.Initializer.initializer, 68
- ConstructionMonitorProcessor
 - BatchProcessing.BatchProcessorConstructionMonitor, 19
- Core.Cencus, 36
 - __dataGetter, 37
 - __init__, 36
 - __showUi, 38
 - link, 39
 - state_arg, 39
 - uiString, 39
 - year_arg, 39
- Core.ConstructionMonitorInit, 40
 - __CreateFrame, 42
 - __init__, 40
 - __SetValues, 44
 - __ShowGui, 45
 - append_file, 46
 - auth_key, 46
 - dateEnd, 47
 - dateStart, 47
 - rest_domain, 47
 - size, 47
 - SourceInclude, 47
 - ui_flag, 47
- Core.ConstructionMonitorMain, 48
 - __appendFile, 54
 - __batches, 54
 - __columnSelection, 54
 - __getCount, 50
 - __getCountUI, 51
 - __headerDict, 54
 - __init__, 48
 - __ParameterCreator, 52
 - __parameterDict, 55
 - __record_val, 55
 - __restDomain, 55
 - __search_id, 55
 - __siteClass, 55
 - __ui_flag, 55
 - dataframe, 55
 - mainFunc, 53
- Core.py, 137, 139, 146, 148
- Core.realtorCom, 76
 - __dataUpdater, 77
 - __idDict, 80
 - __init__, 76
 - __last_date, 80
 - __linkDict, 80
 - __linkGetter, 78
 - __page_html, 80
 - __showUi, 79
 - __update_date, 80
 - dfCounty, 80
 - dfState, 80
 - dfZip, 80
 - uiString, 80
- Core.UtahRealEstateInit, 82
 - __CreateFrame, 83
 - __init__, 82
 - __SetValues, 85
 - __ShowGui, 87
 - append_file, 88
 - dateEnd, 88
 - dateStart, 88
 - file_name, 88
 - ListedOrModified, 88
 - select, 89
 - StandardStatus, 89
- Core.UtahRealEstateMain, 90
 - __appendFile, 96
 - __batches, 96
 - __dateEnd, 97
 - __dateStart, 97
 - __getCount, 92
 - __getCountUI, 93
 - __headerDict, 97
 - __init__, 90
 - __ParameterCreator, 94
 - __parameterString, 97
 - __record_val, 97
 - __restDomain, 97
 - __siteClass, 97
 - dataframe, 97
 - filePath, 97
 - key, 97
 - keyPath, 97
 - mainFunc, 95
- createGui
 - BatchProgressGUI.BatchProgressGUI, 29
- CreateProgressLayout
 - BatchProgressGUI.BatchProgressGUI, 31
- data
 - FileSaver.FileSaver, 62
- dataAppending
 - FileSaver.FileSaver, 62
- DataChecker.py, 110
- dataframe
 - BatchProcessing.BatchProcessorConstructionMonitor, 22
 - BatchProcessing.BatchProcessorUtahRealEstate, 26
 - BatchProgressGUI.BatchProgressGUI, 35
 - Core.ConstructionMonitorMain, 55
 - Core.UtahRealEstateMain, 97
- DataSupportFunctions.py, 111
- DataTransfer.DataTransfer, 56
 - __init__, 56
 - __value, 58
 - getValue, 56
 - setValue, 57
 - whileValue, 57

- DataTransfer.py, 127
- dateEnd
 - Core.ConstructionMonitorInit, 47
 - Core.UtahRealEstateInit, 88
- dateStart
 - Core.ConstructionMonitorInit, 47
 - Core.UtahRealEstateInit, 88
- dfCounty
 - Core.realtorCom, 80
- dfState
 - Core.realtorCom, 80
- dfZip
 - Core.realtorCom, 80
- docPath
 - FileSaver.FileSaver, 62
- ErrorPopup.py, 114
- ErrorPrint.py, 115
- file_name
 - AuthUtil.AuthUtil, 16
 - Core.UtahRealEstateInit, 88
- fileName
 - FileSaver.FileSaver, 62
- filePath
 - AuthUtil.AuthUtil, 16
 - Core.UtahRealEstateMain, 97
- FileSaver.FileSaver, 59
 - __init__, 59
 - appendFlag, 61
 - data, 62
 - dataAppending, 62
 - docPath, 62
 - fileName, 62
 - getPath, 61
 - outputFrame, 62
 - primaryKey, 62
 - uiFlag, 62
- FileSaver.py, 112
- FuncSelector
 - BatchProcessing.BatchProcessorConstructionMonitor, 21
 - BatchProcessing.BatchProcessorUtahRealEstate, 25
- getPath
 - FileSaver.FileSaver, 61
- getValue
 - DataTransfer.DataTransfer, 56
- ImageLoader.py, 129
- Initializer.py, 134
- jsonDict
 - AuthUtil.AuthUtil, 16
- k
 - AuthUtil.AuthUtil, 17
- key
 - Core.UtahRealEstateMain, 97
- keyFlag
 - AuthUtil.AuthUtil, 17
- keyPath
 - AuthUtil.AuthUtil, 17
 - Core.UtahRealEstateMain, 97
- link
 - Core.Census, 39
- ListedOrModified
 - AuthUtil.AuthUtil, 17
 - Core.UtahRealEstateInit, 88
- Logger.py, 116
- mainFunc
 - Core.ConstructionMonitorMain, 53
 - Core.UtahRealEstateMain, 95
- outcomeText
 - AuthUtil.AuthUtil, 17
- outputFrame
 - FileSaver.FileSaver, 62
- passFlagCm
 - AuthUtil.AuthUtil, 17
- passFlagUre
 - AuthUtil.AuthUtil, 17
- PopupWrapped.PopupWrapped, 69
 - __counter, 74
 - __createLayout, 70
 - __createWindow, 71
 - __error, 74
 - __init__, 69
 - __layout, 74
 - __text, 74
 - __thread, 75
 - __type, 75
 - __windowObj, 75
 - stopWindow, 73
 - textUpdate, 73
 - windowPush, 74
- PopupWrapped.py, 130
- primaryKey
 - FileSaver.FileSaver, 62
- PrintFunc.py, 117
- ProgressUpdater
 - BatchProgressGUI.BatchProgressGUI, 32
- rest_domain
 - Core.ConstructionMonitorInit, 47
- RESError.py, 118
- select
 - Core.UtahRealEstateInit, 89
- setValue
 - DataTransfer.DataTransfer, 57
- size
 - Core.ConstructionMonitorInit, 47
- SourceInclude
 - Core.ConstructionMonitorInit, 47
- StandardStatus
 - AuthUtil.AuthUtil, 17
 - Core.UtahRealEstateInit, 89
- state_arg
 - Core.Census, 39
- stopWindow
 - PopupWrapped.PopupWrapped, 73
- textUpdate
 - PopupWrapped.PopupWrapped, 73
- TimeUpdater
 - BatchProgressGUI.BatchProgressGUI, 33
- ui_flag
 - Core.ConstructionMonitorInit, 47

| | |
|---|--|
| uiFlag | BatchProcessing.BatchProcessorUtahRealEs |
| FileSaver.FileSaver, 62 | tate, 26 |
| uiString | whileValue |
| Core.Cencus, 39 | DataTransfer.DataTransfer, 57 |
| Core.realtorCom, 80 | windowPush |
| ValueChecker | PopupWrapped.PopupWrapped, 74 |
| BatchProgressGUI.BatchProgressGUI, 34 | year_arg |
| valueObject | Core.Cencus, 39 |
| BatchProcessing.BatchProcessorConstructio | |
| nMonitor, 22 | |