Gardner API Utility Documentation

Willem van der Schans
Version 1.0.2
4/17/2023 3:33:00 PM

Table of Contents

Readme: Gardner Policy Institute API Utility

Author: Willem van der Schans

Commissioner: Gardner Policy Institute

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

Notes

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

VERSION INFO

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

Note: All dependencies are included in the Windows installer

Authentication Requirements

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

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

Changelog

Initial release

Version: 1.0.0
Date: 2023-04-08

Core Functionality

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

User InterFace

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

File Functionality

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

Logging and Error Handling

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

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

Security

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

GUI

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

Other

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

Version: 0.9.5
Date: 2023-04-05

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

Version: 0.9.0
Date: 2023-03-16

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

Version: 0.8.0
Date: 2023-03-12

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

Version: 0.7.0
Date: 2023-03-02

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

Version: 0.6.0
Date: 2023-02-25

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

Version: 0.5.0
Date: 2023-02-15

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

Version: 0.4.0
Date: 2022-12-07

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

Version: 0.3.0
Date: 2022-11-15

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

Version: 0.2.0
Date: 2022-11-03

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

Version: 0.1.0
Date: 2022-10-25

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

Version: 0.0.0
Date: 2022-10-15

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

License

Class Index

Class List

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

File Index

File List

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

Class Documentation

AuthUtil.AuthUtil Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

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

Private Member Functions

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

Detailed Description

Definition at line 18 of file AuthUtil.py.

Constructor & Destructor Documentation

def AuthUtil.AuthUtil.__init__ (*self*)

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

Args:
self: Represent the instance of the class

Returns:
None

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

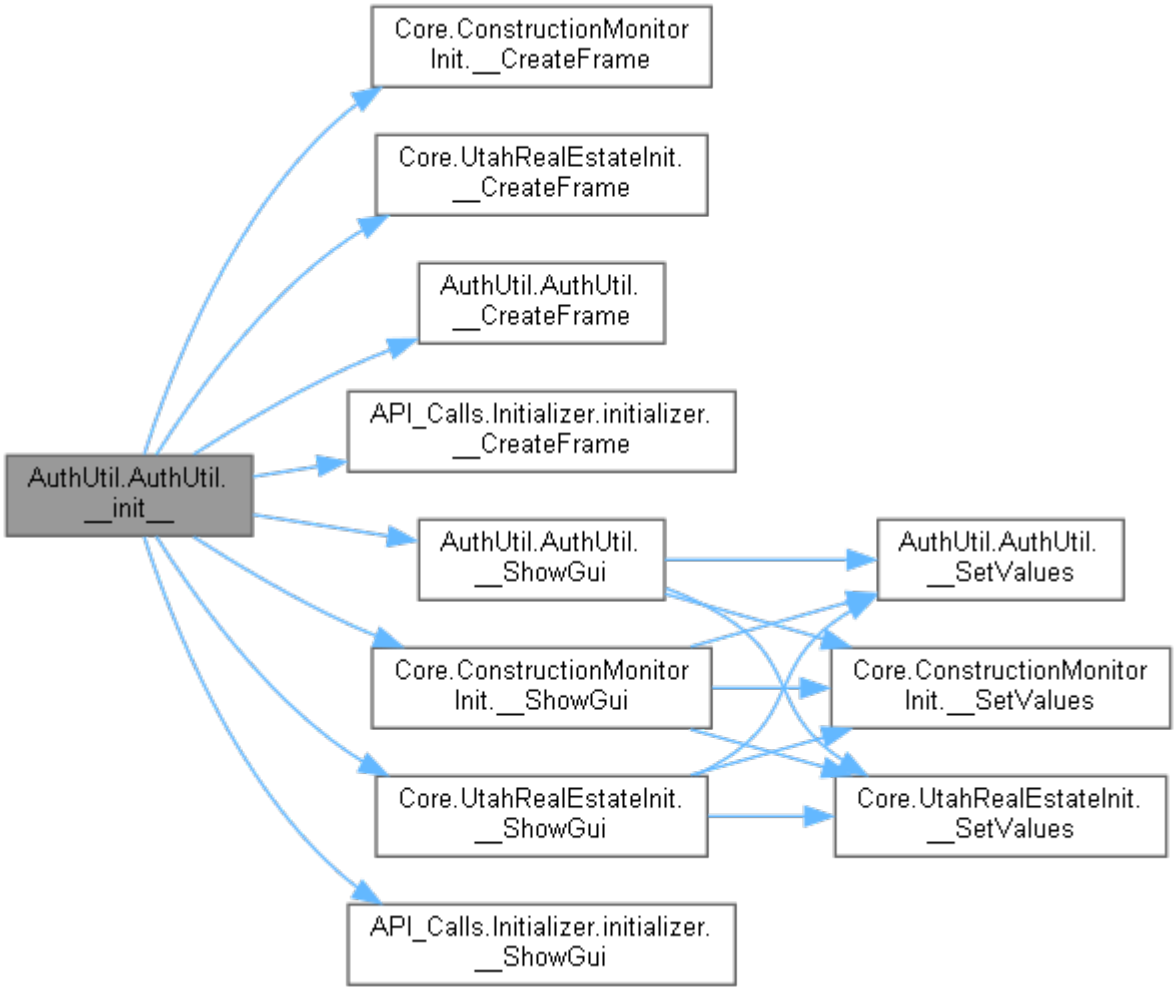
Definition at line 20 of file AuthUtil.py.

```
00020     def __init__(self):
00021 
00022         """
00023     The __init__ function is called when the class is instantiated.
00024     It sets up the initial state of the object, which in this case means that
it creates a new window and displays it on screen.
00025 
```

```
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(
00087                     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
00090             f = open(self.filePath.joinpath("auth.json"), "wb")
00091             f.close()
00092             self.keyFlag = False
00093
00094         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00095
00096         try:
00097
ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35
.av3ra3rvavcr3w"), 2)
00098         except Exception as e:
00099             # Logging
00100             print(
00101                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as
hidden. This is either a Permission error or Input Error. Continuing...")
00102             pass
00103
```

Here is the call graph for this function:



Member Function Documentation

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

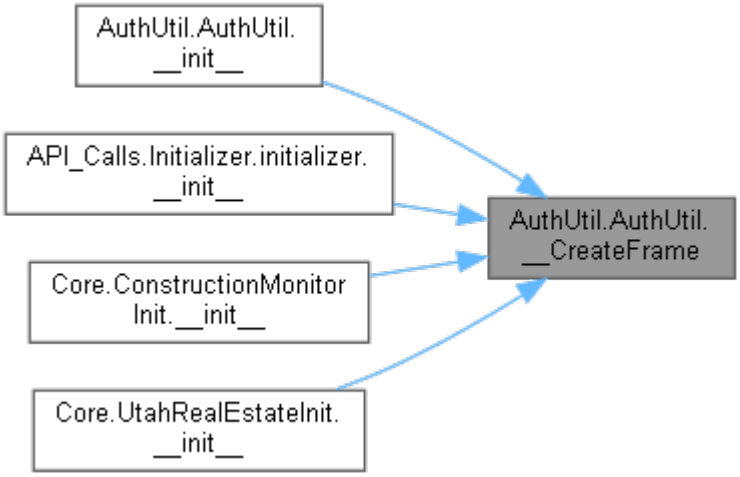
```
The __CreateFrame function creates the GUI layout for the Authentication Utility.
It is called by __init__ and returns a list of lists that contains all the elements
that will be displayed in the window.

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

Here is the caller graph for this function:



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

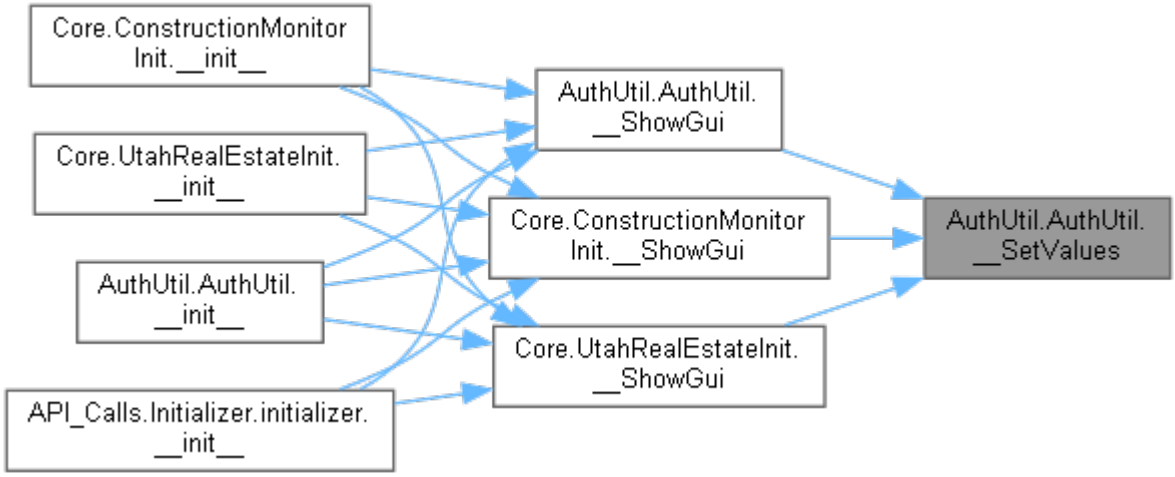
```
The __SetValues function is called when the user clicks on the &amp;quot;OK&amp;quot; button
in the window.
It takes a dictionary of values as an argument, and then uses those values to update
the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

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

Returns:
A dictionary of the values entered by the user

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

Definition at line 104 of file AuthUtil.py.

```
00104     def __SetValues(self, values):
00105
00106         """
00107     The __SetValues function is called when the user clicks on the &amp;quot;OK&amp;quot;
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]`

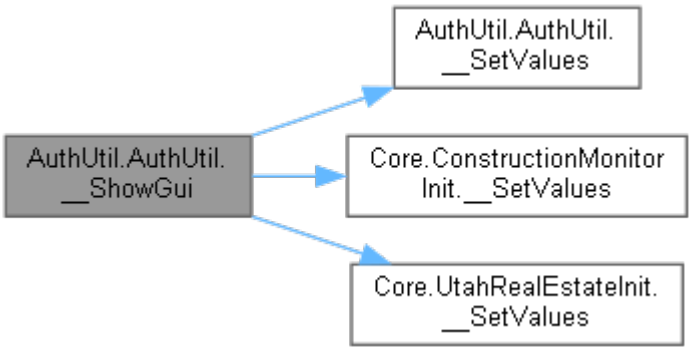
```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the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

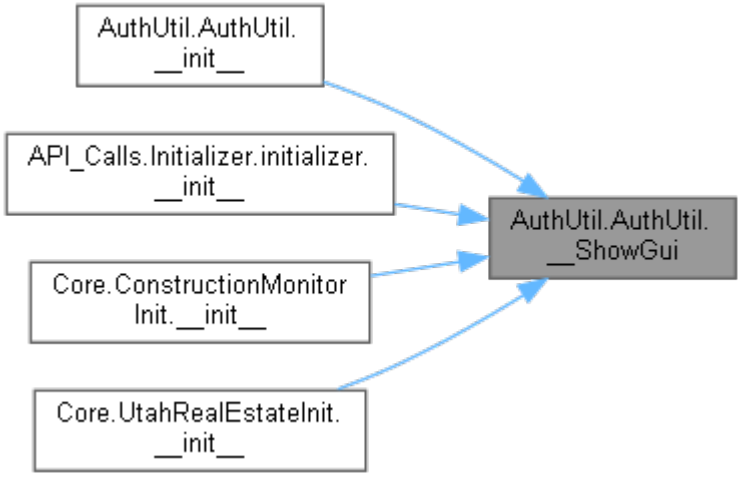
```
00185     def __ShowGui(self, layout, text):
00186
00187         """
00188     The __ShowGui function is a helper function that displays the GUI to the user.
00189     It takes in two arguments: layout and text. The layout argument is a list
of lists,
00190     which contains all the elements that will be displayed on screen. The text
argument
00191     is simply what will be displayed at the top of the window.
00192
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,
return_keyboard_events=True,
00202                            finalize=True,
00203                            icon=ImageLoader("taskbar_icon.ico"))
00204
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
```

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

Member Function Documentation

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

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

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

Returns:
A dataframe

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

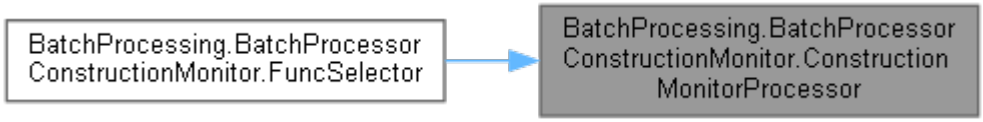
Definition at line 94 of file BatchProcessing.py.

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

```
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 
```

Here is the caller graph for this function:



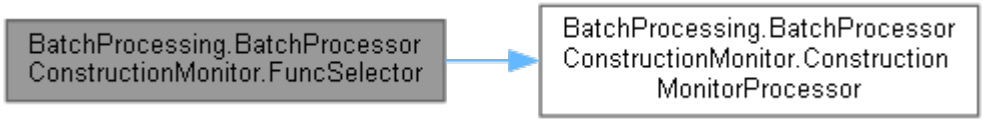
def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (*self*)

```
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
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 
```

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
object of that class
00177     is instantiated, and it sets up the attributes of that object. In this case,
we are setting up
00178     the dataframe attribute to be None (which will be set later), and we are also
setting up some
00179     other attributes which will help us make our API calls.
```

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

Member Function Documentation

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

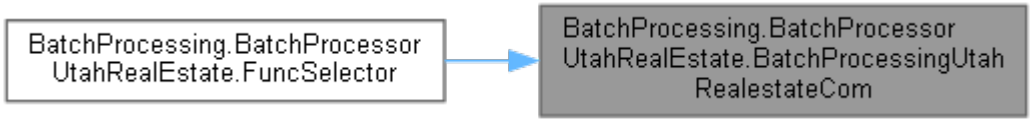
```
The BatchProcessingUtahRealestateCom function is a function that takes in the
valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary
information for this
function to work properly. The BatchProcessingUtahRealestateCom function will then use
requests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

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

Returns:
A dataframe of the scraped data

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

Definition at line 219 of file BatchProcessing.py.

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

```
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 *
200}",
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*)

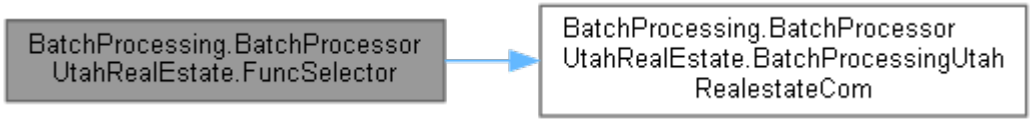
```
The FuncSelector function is a function that takes the valueObject as an argument and
then calls the appropriate
function based on what was selected in the dropdown menu.  The valueObject is passed
to each of these functions
so that they can access all of its attributes.

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

dataframePrivate Attributes

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

Detailed Description

Definition at line 17 of file BatchProgressGUI.py.

Constructor & Destructor Documentation

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

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

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

Returns:
Nothing

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

Definition at line 19 of file BatchProgressGUI.py.

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

Member Function Documentation

def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (*self*)

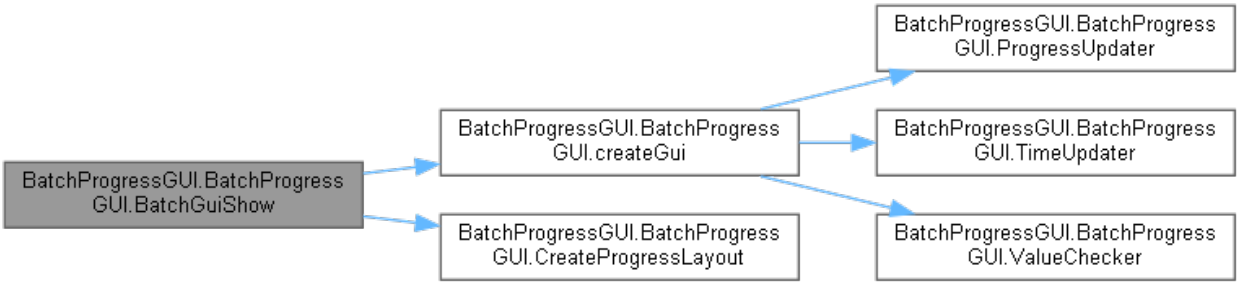
```
The BatchGuiShow function is called by the BatchGui function. It creates a progress
bar layout and then calls the createGui function to create a GUI for batch processing.

Args:
self: Represent the instance of the class

Returns:
The __type of the batchgui class

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

Definition at line 53 of file BatchProgressGUI.py.

```
00053     def BatchGuiShow(self):
00054         """
00055     The BatchGuiShow function is called by the BatchGui function. It creates a
progress bar layout and then calls the createGui function to create a GUI for batch
processing.
00056
00057     Args:
00058         self: Represent the instance of the class
00059
00060     Returns:
00061         The __type of the batchgui class
00062
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
```

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

Here is the call graph for this function:



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

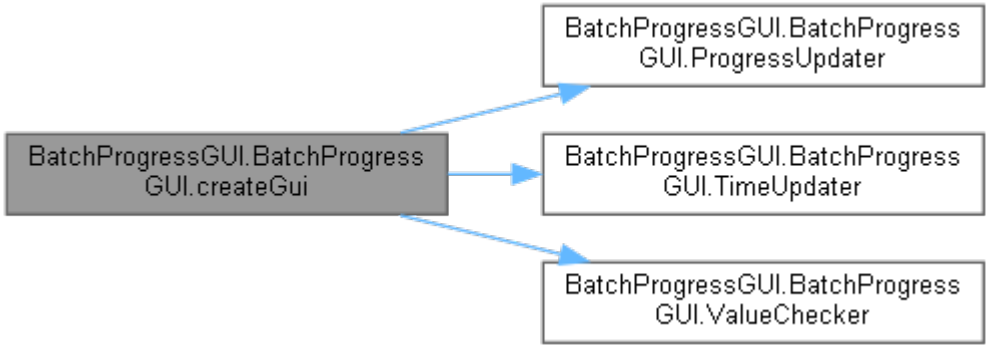
```
The createGui function is the main function that creates the GUI.
It takes in a type parameter which determines what kind of batch processor to use.
The createGui function then sets up all the variables and objects needed for
the program to run, including: window, start_time, update_text, valueObj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

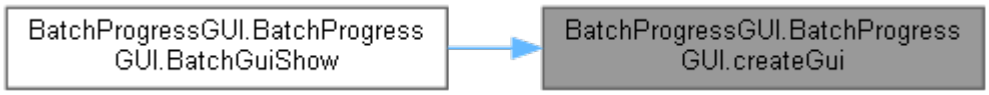
```
00104     def createGui(self, Sourcetype):
00105
00106         """
00107     The createGui function is the main function that creates the GUI.
00108     It takes in a type parameter which determines what kind of batch processor
to use.
00109     The createGui function then sets up all the variables 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
```

Here is the call graph for this function:



Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (*self*)

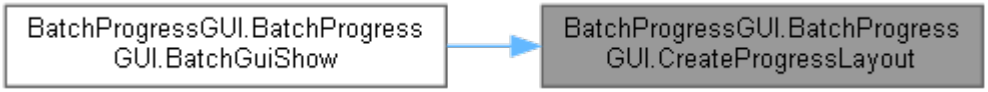
```
The CreateProgressLayout function creates the layout for the progress window.
The function takes in self as a parameter and returns nothing.

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),
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
```

Here is the caller graph for this function:



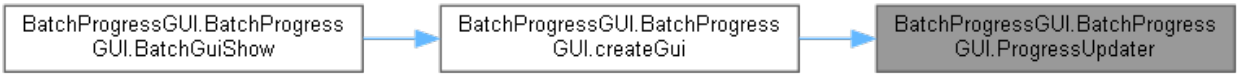
def BatchProgressGUI.BatchProgressGUI.ProgressUpdater (*self*, *valueObj*)

```
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
bar and text
00191     in the GUI. It takes in one argument, which is an object containing information
about the
00192     current batch number. The ProgressUpdater function then checks if this value
has changed from
00193     the last time it was called (i.e., if we are on a new batch). If so, it updates
both the progress
00194     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
```

Here is the caller graph for this function:



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

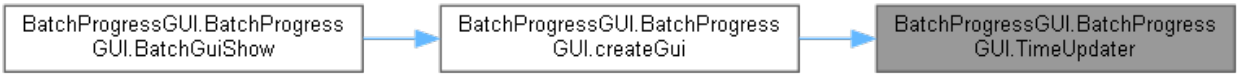
```
The TimeUpdater function is a thread that updates the time elapsed and estimated time
needed to complete
the current batch. It does this by reading the start_time variable passed in, getting
the current time,
calculating how much time has passed since start_time was set and then updating a timer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

```
00254                           seconds=(__passed_time * (self.__batches /
self.__batch_counter) - __passed_time)).seconds
00255                   except:
00256                       __time_est = datetime.timedelta(
00257                           seconds=(__passed_time * self.__batches -
__passed_time)).seconds
00258
00259                   __time_est = time.strftime('%H:%M:%S',
time.gmtime(__time_est))
00260
00261                   __end_string = f"Est time needed {__time_est}"
00262                   self.__window.write_event_value('update--time_est--',
__end_string)
00263               else:
00264                   __end_string = f"Est time needed 00:00:00"
00265                   self.  window.write event value('update--time est--',
__end_string)
00266               time.sleep(0.25)
00267
```

Here is the caller graph for this function:



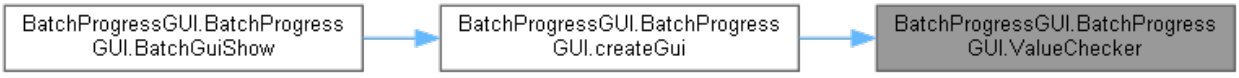
def BatchProgressGUI.BatchProgressGUI.ValueChecker (*self*, *ObjectVal*)

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

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

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

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

Definition at line 268 of file BatchProgressGUI.py.

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

Here is the caller graph for this function:



Member Data Documentation

BatchProgressGUI.BatchProgressGUI.__batch_counter [private]

Definition at line 51 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__batches [private]

Definition at line 49 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__columnSelection [private]

Definition at line 44 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__headerDict [private]

Definition at line 43 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__layout [private]

Definition at line 48 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__parameterDict [private]

Definition at line 41 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__restDomain [private]

Definition at line 42 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__type [private]

Definition at line 45 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.__window [private]

Definition at line 50 of file BatchProgressGUI.py.

BatchProgressGUI.BatchProgressGUI.dataframe

Definition at line 46 of file BatchProgressGUI.py.

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

- BatchProgressGUI.py

Core.Cencus Class Reference

Public Member Functions

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

Public Attributes

- state_argyear_arg
- uiString
- link

Private Member Functions

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

Detailed Description

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

Constructor & Destructor Documentation

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

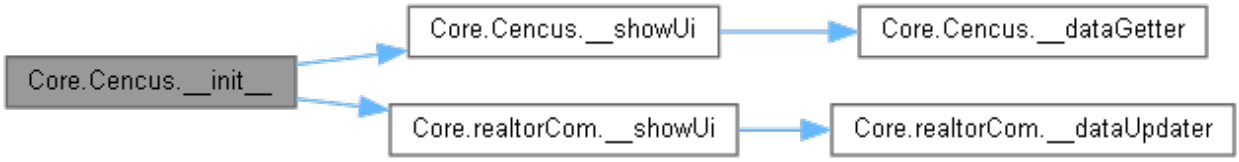
```
The __init__ function is called when the class is instantiated.
It's job is to initialize the object with some default values, and do any other setup
that might be necessary.
The __init__ function can take arguments, but it doesn't have to.

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

Returns:
A popupwrapped object

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

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

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

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

Here is the call graph for this function:



Member Function Documentation

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

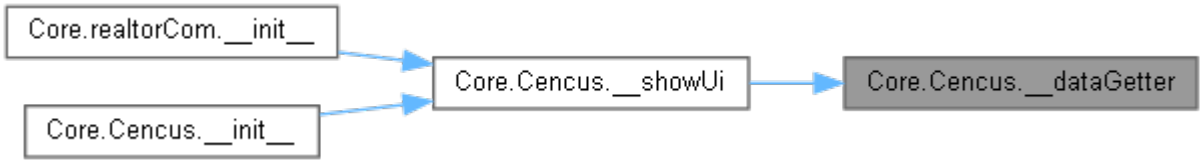
```
The __dataGetter function is a private function that gets the data from the CFPB API.
It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for
the API call.

Args:
self: Represent the instance of the class

Returns:
A response object

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

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

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

```
00101
00102         while not passFlag:
00103
00104             self.link =
"https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?" + f"states={self.state_arg}"
+ f"&years={self.year_arg}"
00105
00106             response = requests.get(self.link)
00107
00108             if response.status_code == 400:
00109                 self.year_arg = int(self.year_arg) - 1
00110
00111             else:
00112                 passFlag = True
00113
00114         RESTError(response)
00115         raise SystemExit(0)
```

Here is the caller graph for this function:



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

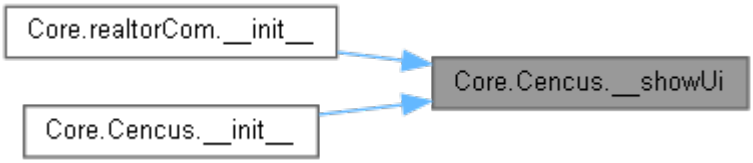
```
The __showUi function is a function that creates a progress bar window.
The __showUi function takes class variables and returns a windowobj.


Args:
self: Represent the instance of the class

Returns:
The uiobj variable

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

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

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

Here is the call graph for this function:

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*)

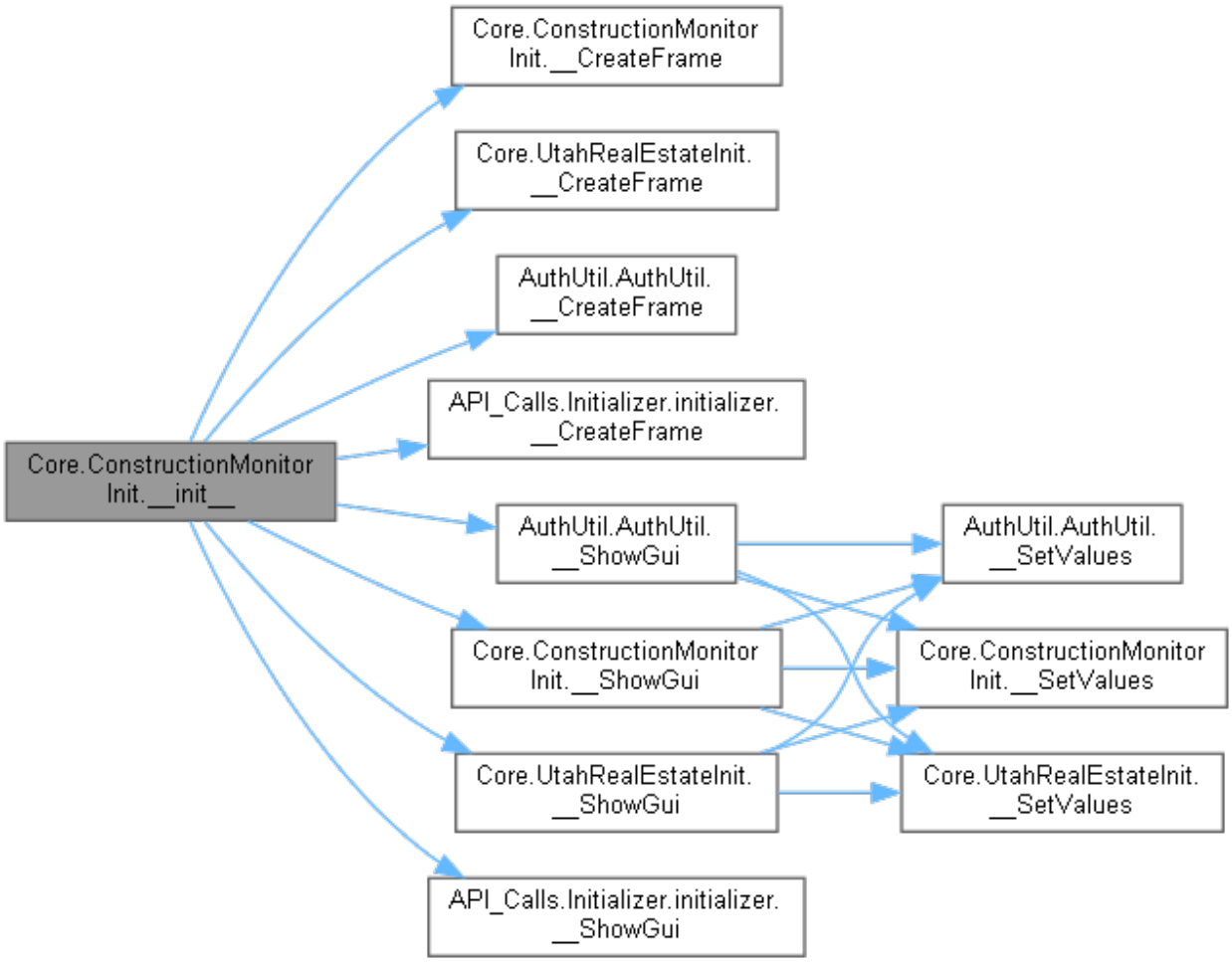
```
The __init__ function is called when the class is instantiated.
It sets up the variables that will be used by other functions in this class.


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]`

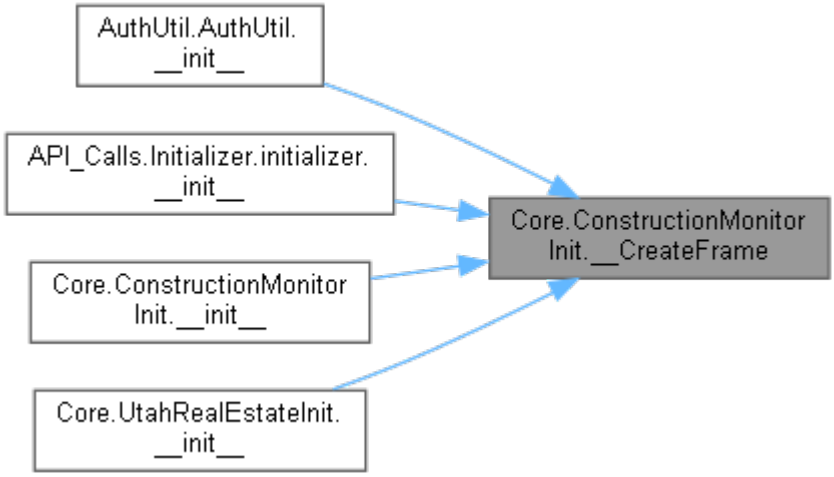
```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the GUI window.
This is done by creating each line as a list and then appending it to another list which
will contain all lines.

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
be displayed in the GUI window.
00121         This is done by creating each line as a list and then appending it to
another list which will contain all lines.
00122
00123     Args:
```

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

Here is the caller graph for this function:



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

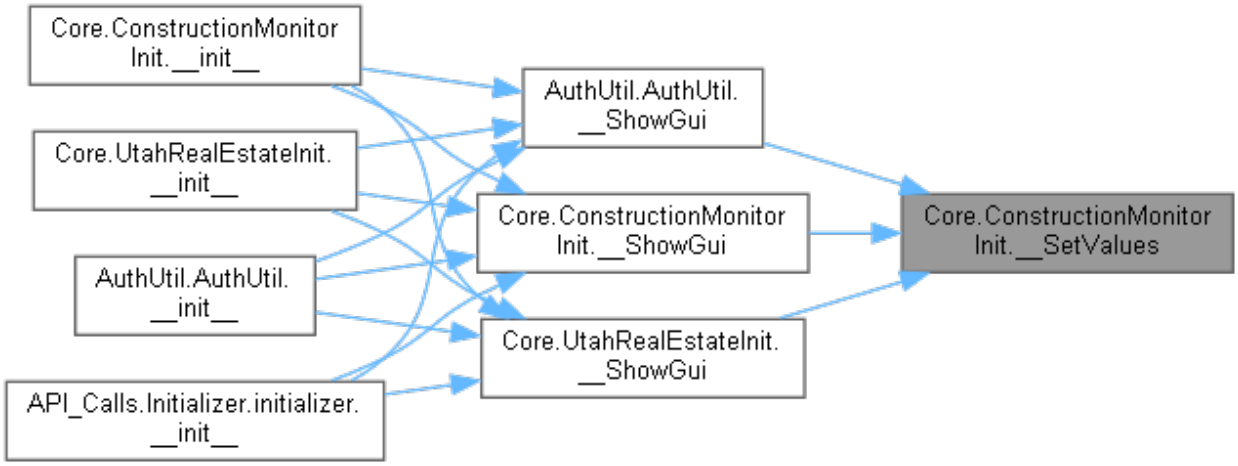
```
The __SetValues function is used to set the values of the variables that are used in
the __GetData function.
The __SetValues function takes a dictionary as an argument, and then sets each variable
based on what is passed into
the dictionary. The keys for this dictionary are defined by the user when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

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

Here is the caller graph for this function:



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

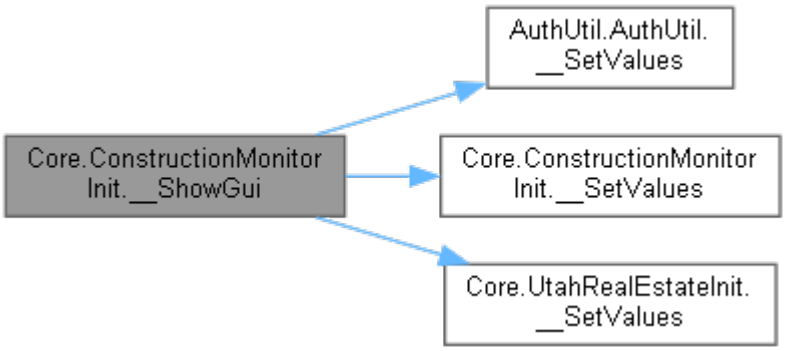
```
The __ShowGui function is the main function that creates and displays the GUI.
It takes in a layout, which is a list of lists containing all the elements to be displayed
on screen.
The text parameter specifies what title should appear at the top of the window.

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

Returns:
A dictionary of values

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

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

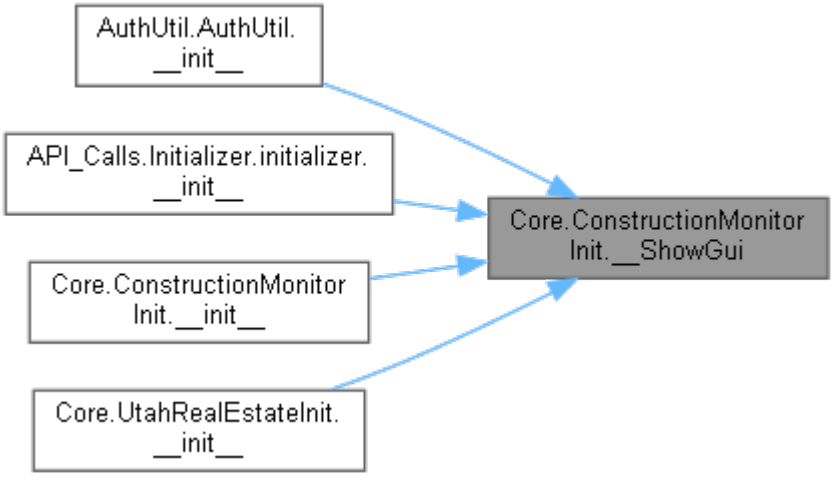
```
00077     def __ShowGui(self, layout, text):
00078
00079         """
00080     The __ShowGui function is the main function that creates and displays the
GUI.
00081     It takes in a layout, which is a list of lists containing all the elements
to be displayed on screen.
00082     The text parameter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

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

Here is the call graph for this function:



Here is the caller graph for this function:



Member Data Documentation

Core.ConstructionMonitorInit.append_file

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

Core.ConstructionMonitorInit.auth_key

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

Core.ConstructionMonitorInit.dateEnd

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

Core.ConstructionMonitorInit.dateStart

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

Core.ConstructionMonitorInit.rest_domain

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

Core.ConstructionMonitorInit.size

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

Core.ConstructionMonitorInit.SourceInclude

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

Core.ConstructionMonitorInit.ui_flag

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

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

- ConstructionMonitor/Core.py

Core.ConstructionMonitorMain Class Reference

Public Member Functions

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

Public Attributes

dataframePrivate Member Functions

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

Private Attributes

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

Detailed Description

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

Constructor & Destructor Documentation

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

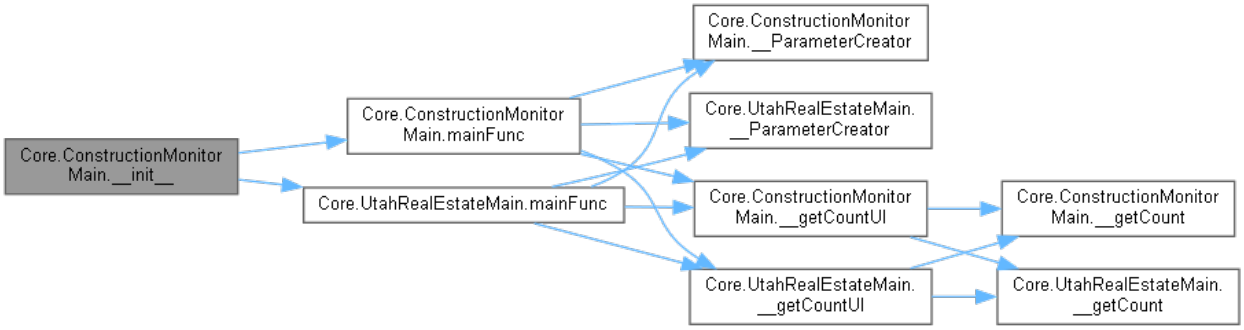
```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to run properly.


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

Returns:
Nothing

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

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

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

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

Here is the call graph for this function:



Member Function Documentation

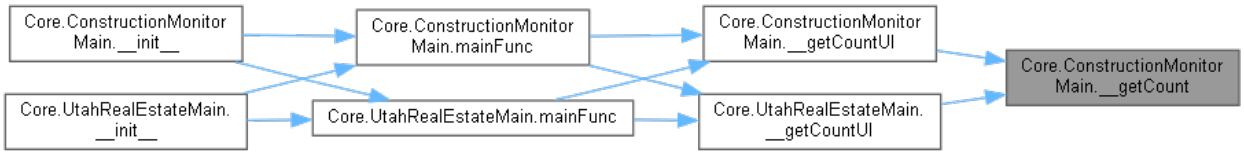
def Core.ConstructionMonitorMain.__getCount (*self*) [private]

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

Here is the caller graph for this function:



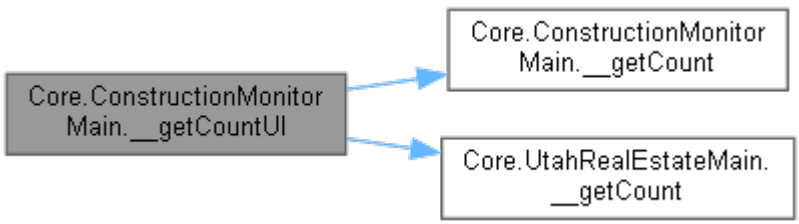
def Core.ConstructionMonitorMain.__getCountUI (*self*) `[private]`

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

Args:
self: Access the class variables and methods

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

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

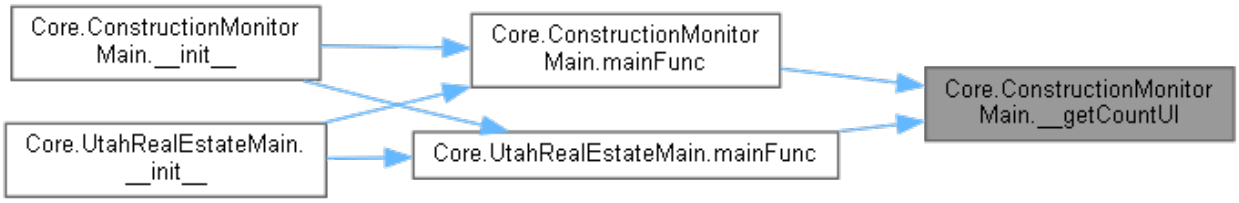
Definition at line 405 of file ConstructionMonitor/Core.py.

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

Here is the call graph for this function:



Here is the caller graph for this function:



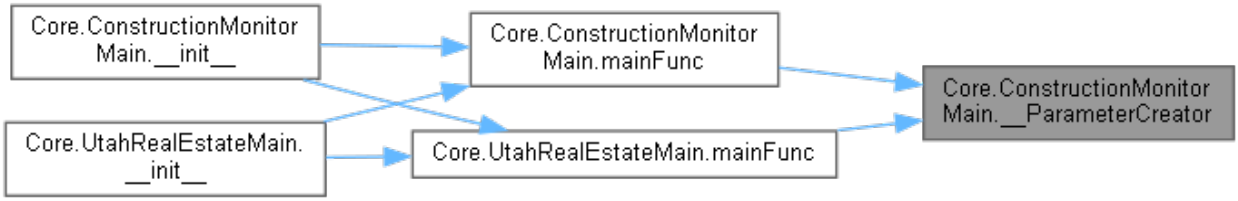
def Core.ConstructionMonitorMain.__ParameterCreator (*self*) `[private]`

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

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

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

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

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

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

Here is the caller graph for this function:



def Core.ConstructionMonitorMain.mainFunc (*self*)

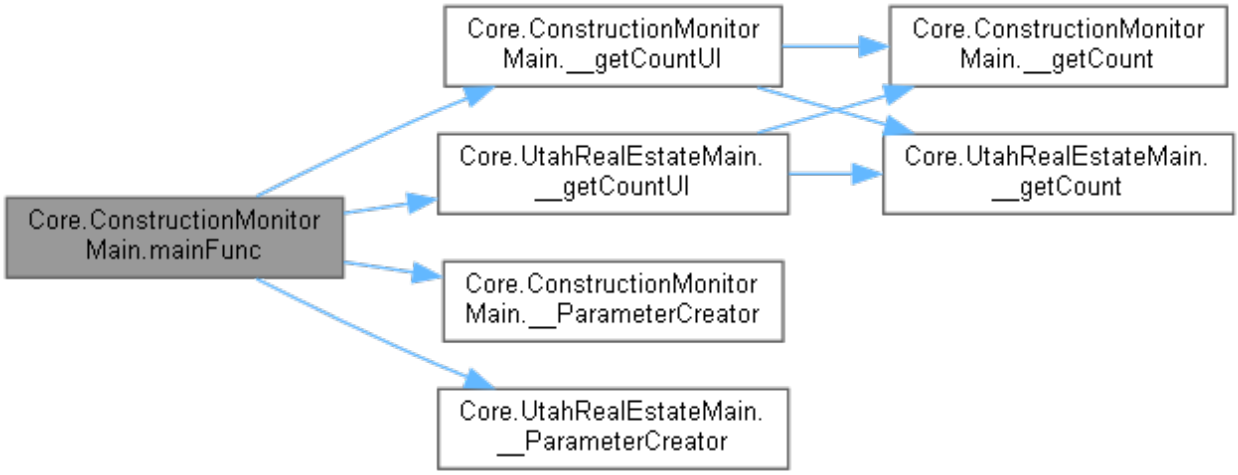
```
The mainFunc function is the main function of this module. It will be called by the
GUI or CLI to execute
the code in this module. The mainFunc function will first create a parameter dictionary
using the __ParameterCreator
method, then it will get a count of all records that match its parameters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

Args:
self: Refer to the current object

Returns:
The dataframe

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

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

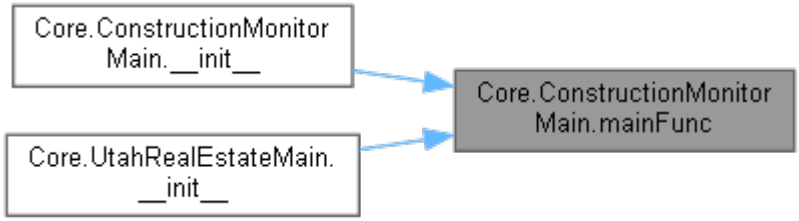
```
00276     def mainFunc(self):
00277         """
00278     The mainFunc function is the main function of this module. It will be called
by the GUI or CLI to execute
00279     the code in this module. The mainFunc function will first create a parameter
dictionary using the __ParameterCreator
00280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

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

Here is the call graph for this function:



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*)

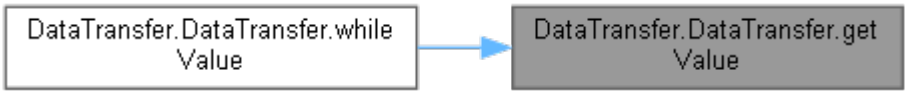
```
The getValue function returns the value of the private variable __value.
This is a getter function that allows access to this private variable.

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*)

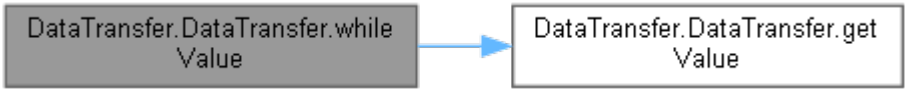
```
The whileValue function is a function that will run the getValue function until it is
told to stop.
This allows for the program to constantly be checking for new values from the sensor.
```

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

Here is the call graph for this function:



Member Data Documentation

DataTransfer.DataTransfer.__value [private]

Definition at line 20 of file DataTransfer.py.

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

- DataTransfer.py

FileSaver.FileSaver Class Reference

Public Member Functions

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

Public Attributes

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

Detailed Description

Definition at line 13 of file FileSaver.py.

Constructor & Destructor Documentation

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

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

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

Returns:
Nothing

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

Definition at line 15 of file FileSaver.py.

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

```
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, Trelent AI
```

Definition at line 101 of file FileSaver.py.

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

Member Data Documentation

FileSaver.FileSaver.appendFlag

Definition at line 39 of file FileSaver.py.

FileSaver.FileSaver.data

Definition at line 37 of file FileSaver.py.

FileSaver.FileSaver.dataAppending

Definition at line 38 of file FileSaver.py.

FileSaver.FileSaver.docPath

Definition at line 35 of file FileSaver.py.

FileSaver.FileSaver.fileName

Definition at line 40 of file FileSaver.py.

FileSaver.FileSaver.outputFrame

Definition at line 70 of file FileSaver.py.

FileSaver.FileSaver.primaryKey

Definition at line 44 of file FileSaver.py.

FileSaver.FileSaver.uiFlag

Definition at line 41 of file FileSaver.py.

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

- FileSaver.py

API_Calls.Initializer.initializer Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

classObjPrivate Member Functions

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

Detailed Description

Definition at line 21 of file Initializer.py.

Constructor & Destructor Documentation

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

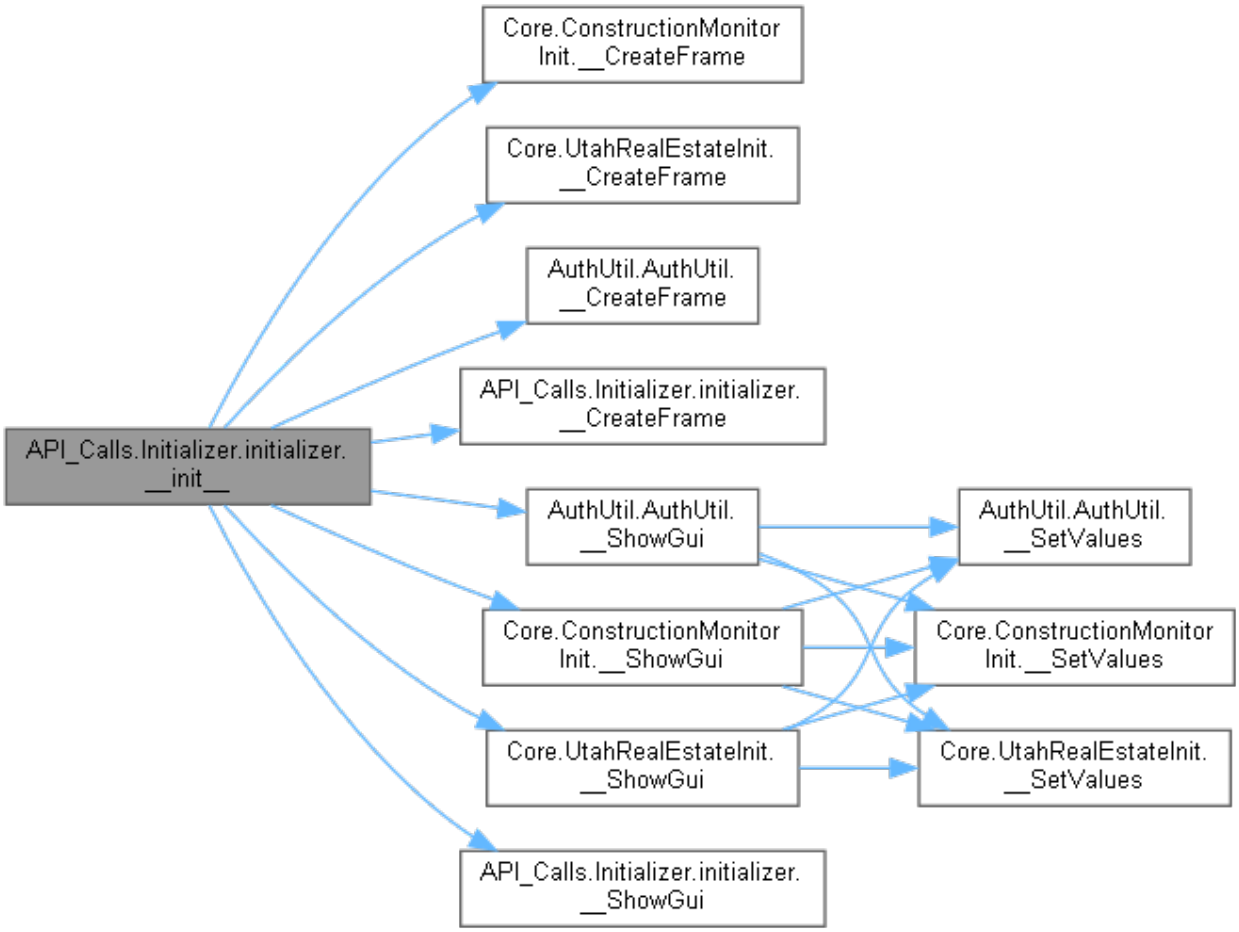
```
The __init__ function is called when the class is instantiated.
It sets up the logging, calls the __ShowGui function to create and display
the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


Args:
self: Represent the instance of the class

Returns:
Nothing

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

Definition at line 23 of file Initializer.py.

```
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 
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]`

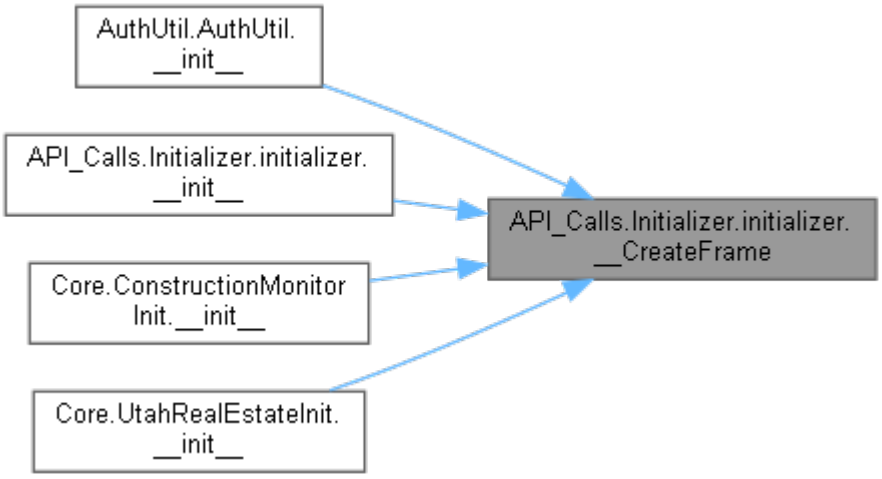
```
The __CreateFrame function is a helper function that creates the layout for the main
window.
It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

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
the main window.
00125     It returns a list of lists, which is then passed to sg.Window() as its layout
parameter.
00126
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,
"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"),
justification="center"),
00150                  sg.Push()]
00151
00152         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20,
None)), sg.Push(),
00153                   sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00154
00155         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)),
sg.Push(), sg.Button("Census", size=(20, None)),
00156                   sg.Push()]]
00157
00158         line8 = [sg.HSeparator()]
00159
00160         line9 = [sg.Push(),
00161                  sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
00162                  sg.Push()]
00163
00164         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20,
None)),
00165                    sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00166
00167         line11 = [sg.HSeparator()]
00168
00169         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]
00170
00171         return layout
```

Here is the caller graph for this function:



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

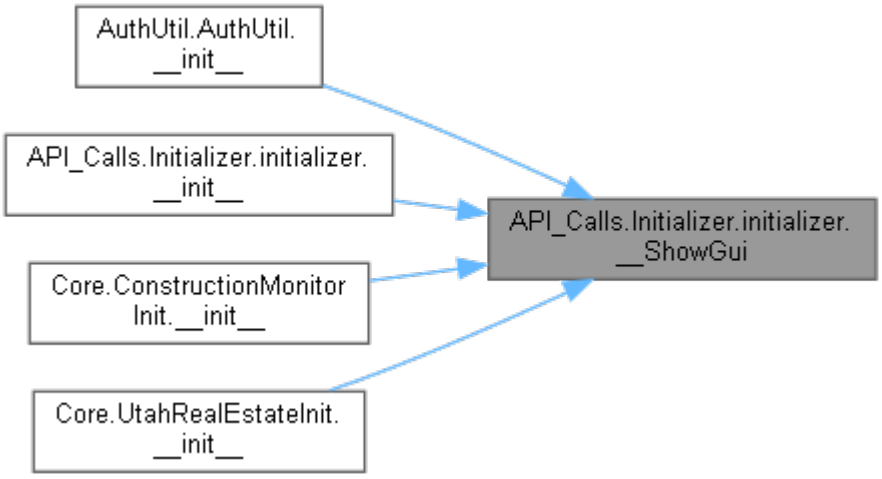
```
The __ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing
a tuple with three elements:
1) The type of element to be displayed (e.g., &amp;quot;Text&amp;quot;, &amp;quot;InputText&amp;quot;,
etc.)
2) A dictionary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

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

Returns:
A window object

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

Definition at line 50 of file Initializer.py.

```
00050     def __ShowGui(self, layout, text):
00051
00052         """
00053     The __ShowGui function is the main function that displays the GUI.
00054     It takes two arguments: layout and text. Layout is a list of lists, each
containing a tuple with three elements:
00055         1) The type of element to be displayed (e.g., &amp;quot;Text&amp;quot;,
&amp;quot;InputText&amp;quot;, etc.)
00056         2) A dictionary containing any additional parameters for that element
(e.g., size, default value, etc.)
00057         3) An optional key name for the element (used in event handling). If no
key name is provided then one will be generated automatically by PySimpleGUIQt based
on its position in the layout list
00058
00059     Args:
00060         self: Represent the instance of the class
00061         layout: Pass the layout of the window to be created
00062         text: Set the title of the window
00063
00064     Returns:
00065         A window object
00066
00067     Doc Author:
00068         Willem van der Schans, Trelent AI
00069     """
00070         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00071                            finalize=True,
00072                            icon=ImageLoader("taskbar_icon.ico"))
00073
00074         while True:
00075             event, values = window.read()
00076
00077             if event == "Construction Monitor":
00078                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API
Call-----------------")
00079                 ConstructionMonitorMain(ConstructionMonitorInit())
00080                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API
Call--------------------\n")
00081             elif event == "Utah Real Estate":
00082                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API
Call-----------------")
00083                 UtahRealEstateMain(UtahRealEstateInit())
00084                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API
Call--------------------\n")
00085             elif event == "Realtor.Com":
00086                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
```

```
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                 Cencus()
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)

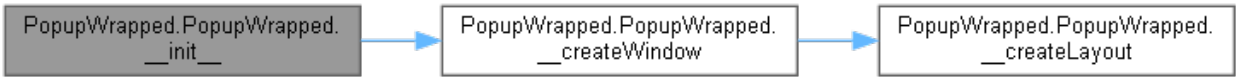
```
The __init__ function is the first function that gets called when an object of this
class is created.
It sets up all the variables and creates a window for us to use.
Args:
self: Represent the instance of the class
text: Set the text of the window
windowType: Determine what type of window to create
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]`

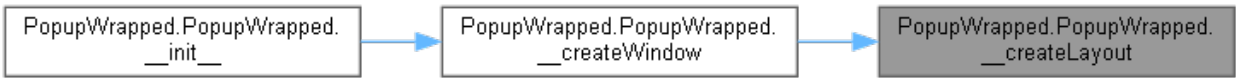
```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates
a layout based on that information.
Args:
self: Refer to the current instance 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,
then creates a layout based on that information.
00044     Args:
00045         self: Refer to the current instance of a class
00046     Returns:
00047         A list of lists
00048     Doc Author:
00049         Willem van der Schans, Trelent AI
00050     """
00051         sg.theme('Default1')
00052         __Line1 = None
00053         __Line2 = None
00054 
00055         if self.__type == "notice":
00056               Line1 = [sg.Push(),
00057                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00058                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00059               __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00060         elif self.__type == "noticeLarge":
00061             __Line1 = [sg.Push(),
00062                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00063                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00064               __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00065         elif self.__type == "errorLarge":
00066             __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
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]`

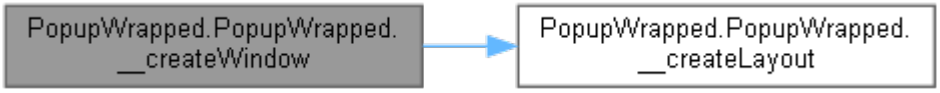
```
The __createWindow function is used to create the window object that will be displayed.
The function takes class variables and a window object. The function first calls
__createLayout, which creates the layout for the window based on what type of message
it is (error, notice, progress). Then it uses 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
&amp;quot;Ok&amp;quot; or closing the window). Once one of these events occurs
Args:
self: Reference the instance of the class
Returns:
A window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 90 of file PopupWrapped.py.

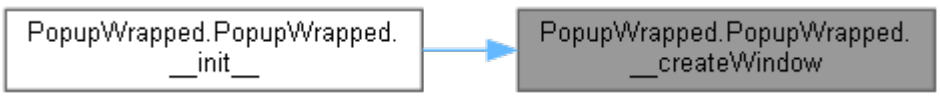
```
00090     def __createWindow(self):
00091         """
00092     The __createWindow function is used to create the window object that will
be displayed.
00093     The function takes class variables and a window object. The function first
calls __createLayout, which creates the layout for the window 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
&amp;quot;Ok&amp;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, finalize=True,
00130                                           modal=True,
00131                                           keep_on_top=True,
00132                                           disable_close=False,
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
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}"
)
00149                 except Exception as e:
00150                     print(
00151                         f"PopupWrapped.py | Error = {e} | Log Folder not found
please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")
00152
00153             self.__windowObj.close()
00154
```

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

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
creates
00172     a loading effect for when something is being processed.
00173     Args:
00174         self: Refer to the object itself
00175         sleep: Control the speed of the text update
00176     Returns:
00177         A string that is the current text of the text field
00178     Doc Author:
00179         Willem van der Schans, Trelent AI
00180     """
00181         self.__counter += 1
00182         if self.__counter == 4:
00183             self.__counter = 1
00184         newString = ""
00185         if self.__type == "notice":
00186             pass
00187         elif self.__type == "error":
00188             pass
00189         elif self.__type == "progress":
00190             newString = f"{self.__text}{'.' * self.__counter}"
00191         self.__windowObj.write_event_value('update-textField-', newString)
00192
```

```
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
checks if the event starts with 'update'.
00200         If it does, it will take everything after 'update' as a key for updating
that 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()
```

Member Data Documentation

PopupWrapped.PopupWrapped.__counter `[private]`

Definition at line 35 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__error `[private]`

Definition at line 31 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__layout `[private]`

Definition at line 32 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__text `[private]`

Definition at line 29 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__thread [private]

Definition at line 34 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__type [private]

Definition at line 30 of file PopupWrapped.py.

PopupWrapped.PopupWrapped.__windowObj [private]

Definition at line 33 of file PopupWrapped.py.

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

- PopupWrapped.py

Core.realtorCom Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- dfStatedfCounty
- dfZip
- uiString

Private Member Functions

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

Private Attributes

- __page_html__update_date
- __last_date
- __idDict
- __linkDict

Detailed Description

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

Constructor & Destructor Documentation

def Core.realtorCom.__init__ (*self*)

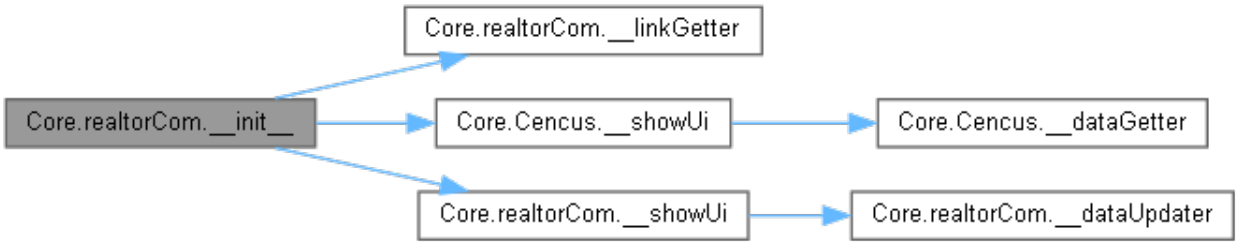
```
The __init__ function is called when the class is instantiated.
It sets up the initial state of an object, and it's where you put code that needs to
run before anything else in your class.

Args:
self: Represent the instance of the class

Returns:
A new object

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

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

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

```
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
dealing with (State/County/Zip), we save
00109
00110
00111     Args:
00112         self: Access the attributes and methods of the class
00113
00114     Returns:
00115         The path of the saved file
00116
00117     Doc Author:
00118         Willem van der Schans, Trelent AI
00119     """
00120         for key, value in self.__idDict.items():
00121             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00122
00123             if key == "State":
00124                 self.dfState = tempdf
00125             elif key == "County":
00126                 self.dfCounty = tempdf
00127             elif key == "Zip":
00128                 self.dfZip = tempdf
00129
00130             FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00131             self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()}
\n"
```

Here is the caller graph for this function:

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

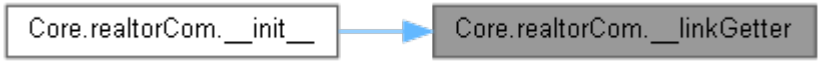
```
The __linkGetter function is a private function that takes the idDict dictionary and
adds
a link to each entry in the dictionary. The link is used to access historical data for
each
scope symbol.

Args:
self: Refer to the object itself

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

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

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

```
00074     def __linkGetter(self):
00075
00076         """
00077     The __linkGetter function is a private function that takes the idDict
dictionary and adds
00078     a link to each entry in the dictionary. The link is used to access historical
data for each
00079     scope symbol.
00080
00081     Args:
00082         self: Refer to the object itself
00083
00084     Returns:
00085         A dictionary of all the links to the history pages
00086
00087     Doc Author:
00088         Willem van der Schans, Trelent AI
00089     """
00090         for key, value in self.__idDict.items():
00091             for row in self.__page_html.find_all("div", {"class": "monthly"}):
```

```
00092                 try:
00093                     for nestedRow in row.find_all("a"):
00094                         if "History" in str(nestedRow.get("href")) and key in
str(nestedRow.get("href")):
00095                             self.__idDict[key] = {"id": value, "link":
nestedRow.get("href")}
00096                 except Exception as e:
00097                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
for realtor.com")
00098                     RESTError(801)
00099                     raise SystemExit(801)
00100
```

Here is the caller graph for this function:



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

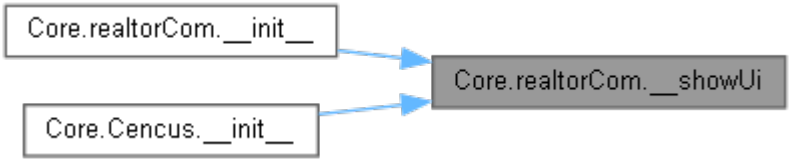
```
The __showUi function is a helper function that creates and displays the progress window.
It also starts the dataUpdater thread, which will update the progress bar as it runs.


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*)

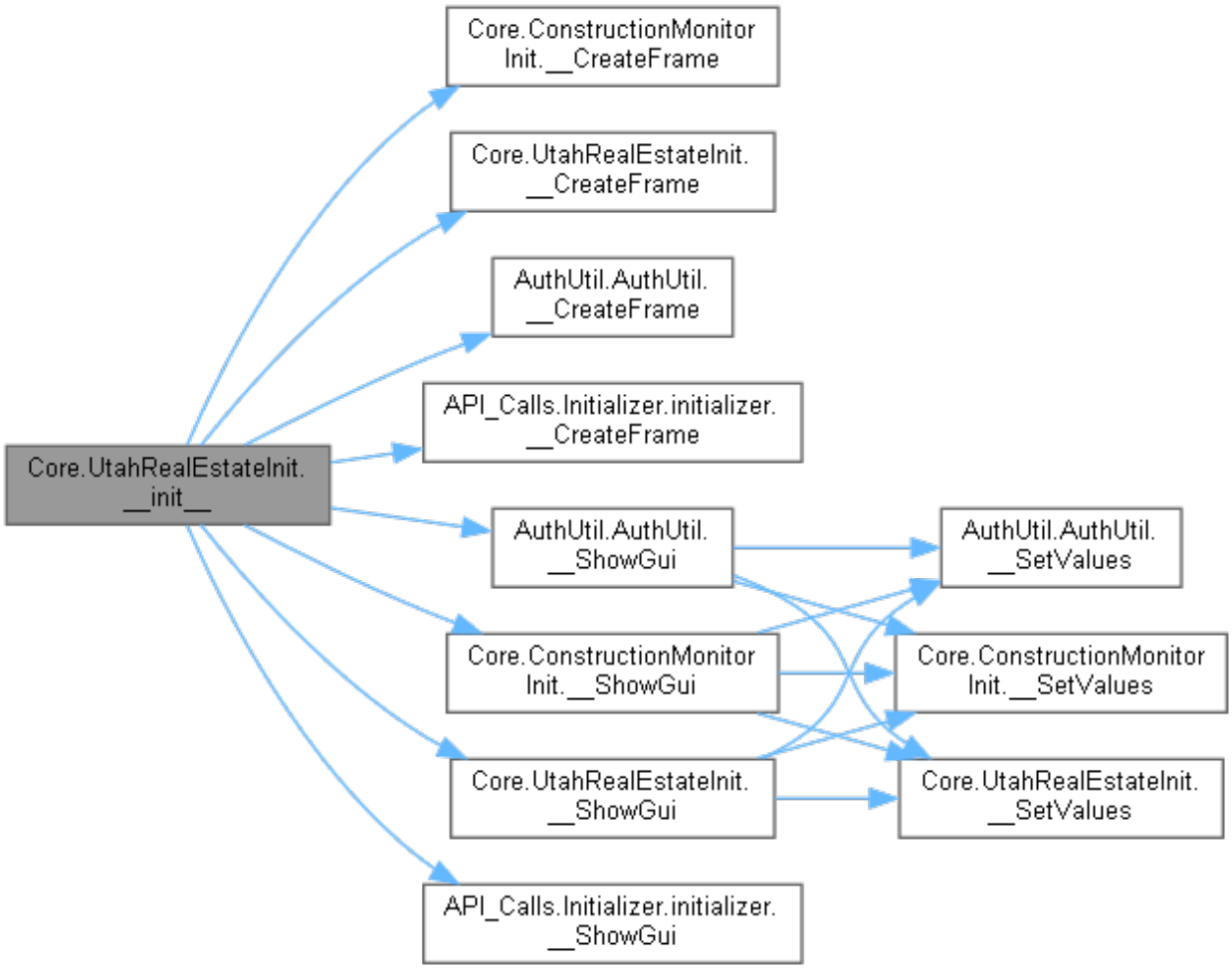
```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object.


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]`

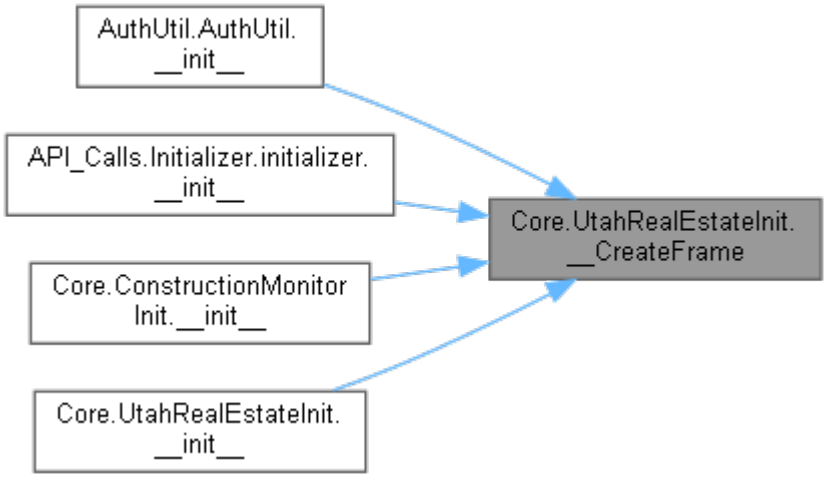
```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed
in the window.
Each element is defined by its type and any additional parameters needed to define it.

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

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

Here is the caller graph for this function:



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

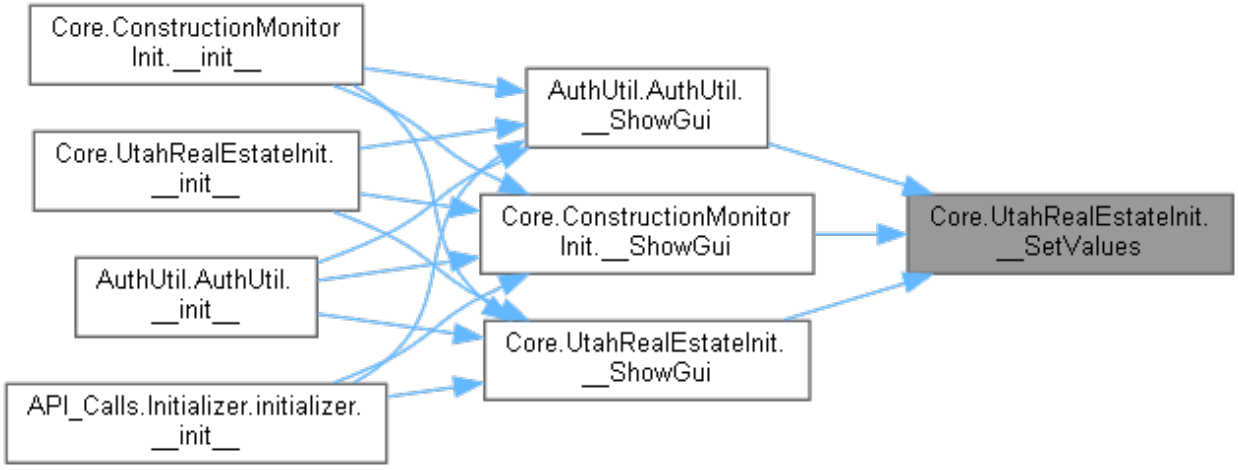
```
The __SetValues function is used to set the values of the variables that are used in
the
__GetData function. The values are passed from a dictionary called 'values' which is
created
by parsing through an XML file using ElementTree. This function also sets default values
for
some of these variables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

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

Here is the caller graph for this function:



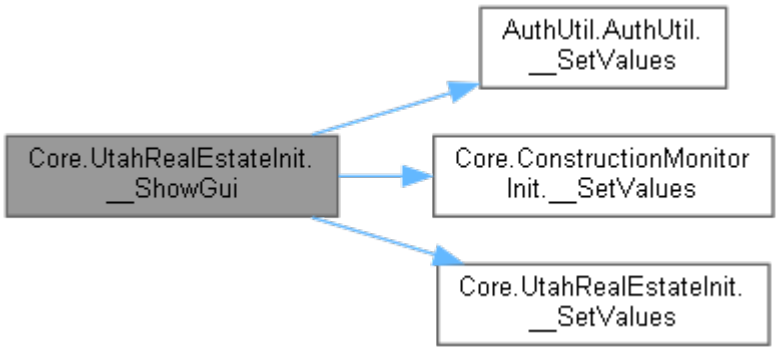
def Core.UtahRealEstateInit.__ShowGui (*self*, *layout*, *text*) [private]

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

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

Returns:
A dictionary of values

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

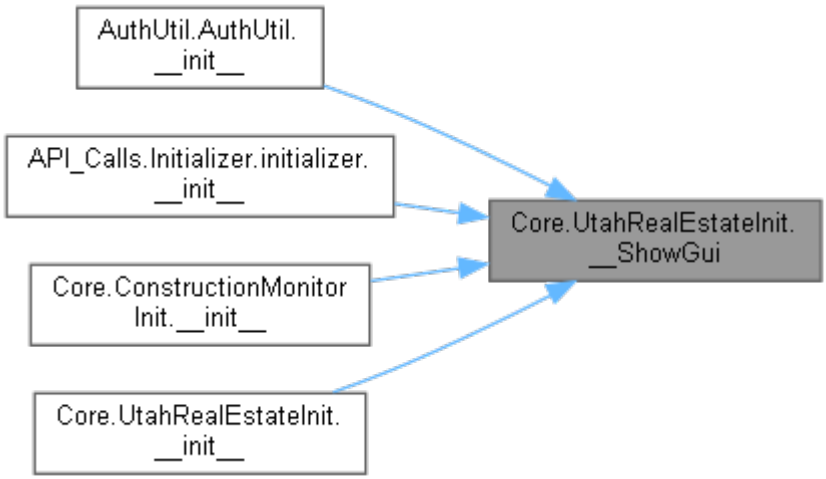
Definition at line 52 of file UtahRealEstate/Core.py.

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

Here is the call graph for this function:



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*)

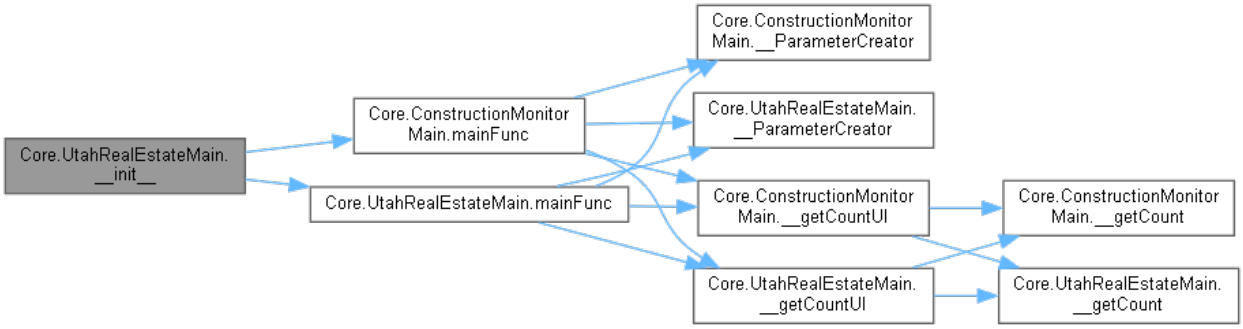
```
The __init__ function is the first function that runs when an object of this class is
created.
It sets up all the variables and functions needed for this class to work properly.

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

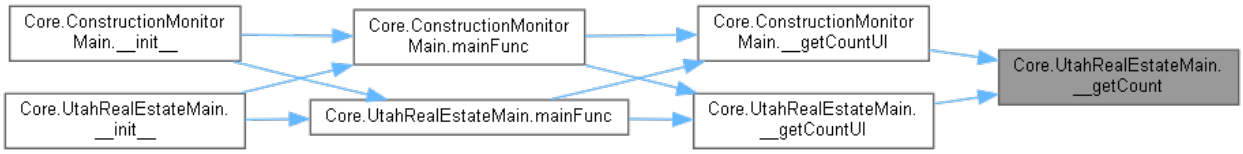
def Core.UtahRealEstateMain.__getCount (*self*) `[private]`

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

Here is the caller graph for this function:



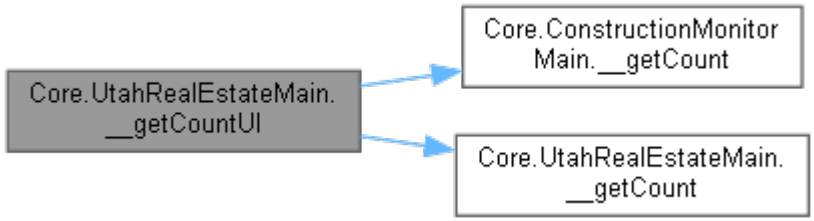
def Core.UtahRealEstateMain.__getCountUI (*self*) `[private]`

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

Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

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

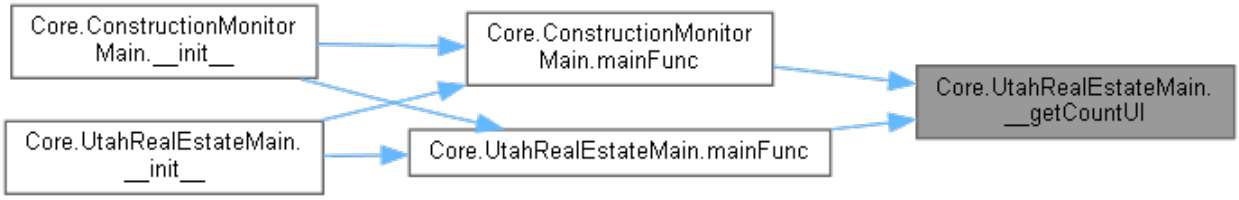
Definition at line 415 of file UtahRealEstate/Core.py.

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

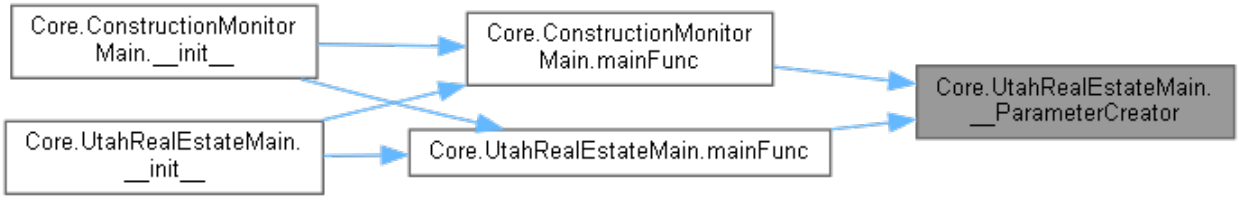
```
The __ParameterCreator function is used to create the filter string for the ReST API
call.
The function takes in a siteClass object and extracts all of its parameters into a
dictionary.
It then creates an appropriate filter string based on those parameters.

Args:
self: Bind the object to the class

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

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

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

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

```
00370
```

Here is the caller graph for this function:



def Core.UtahRealEstateMain.mainFunc (*self*)

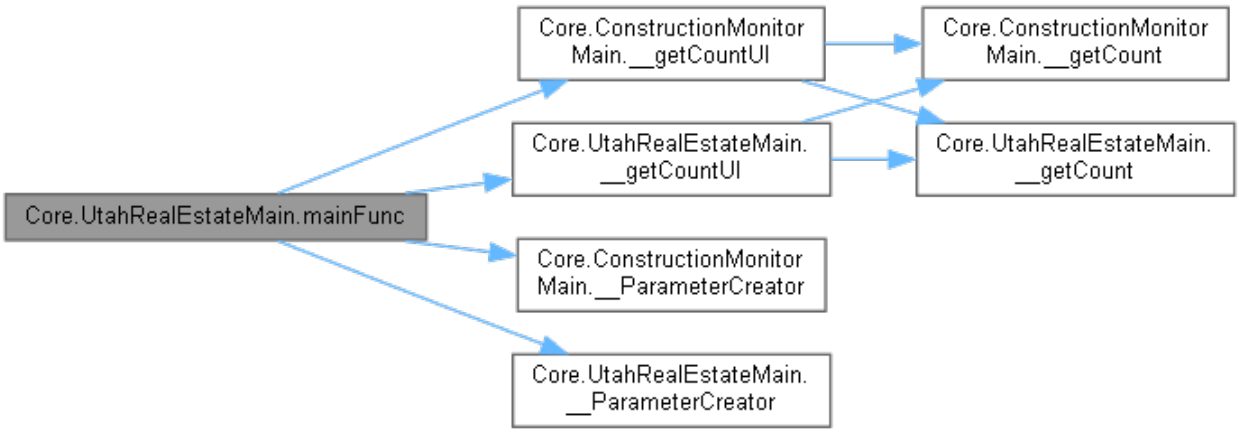
```
The mainFunc function is the main function of this module. It will be called by the
GUI when a user clicks on
the &amp;quot;Run&amp;quot; button in the GUI. The mainFunc function should contain all of your
code for running your program, and it
should return a dataframe that contains all the data you want to display in your final
report.

Args:
self: Reference the object itself

Returns:
A dataframe

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

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

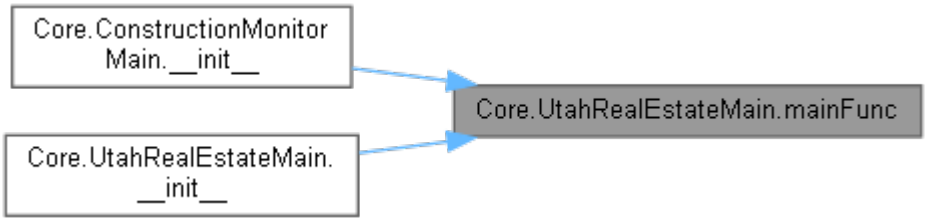
```
00267     def mainFunc(self):
00268 
00269         """
00270     The mainFunc function is the main function of this module. It will be called
by the GUI when a user clicks on
00271     the &amp;quot;Run&amp;quot; button in the GUI. The mainFunc function should contain
all of your code for running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

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

Here is the call graph for this function:



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                            "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(
00087                     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
00090             f = open(self.filePath.joinpath("auth.json"), "wb")
00091             f.close()
00092             self.keyFlag = False
00093
00094         self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00095
00096         try:
00097
ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3
ra3rvavcr3w"), 2)
00098         except Exception as e:
00099             # Logging
00100             print(
00101                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as hidden.
This is either a Permission error or Input Error. Continuing...")
00102             pass
00103
00104     def __SetValues(self, values):
00105
00106         """
00107     The __SetValues function is called when the user clicks on the &amp;quot;OK&amp;quot;
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 
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 of lists,
00190     which contains all the elements that will be displayed on screen. The text
argument
00191     is simply what will be displayed at the top of the window.
00192 
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,
return_keyboard_events=True,
00202                            finalize=True,
00203                            icon=ImageLoader("taskbar_icon.ico"))
00204 
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 
00218                 break
00219             else:
00220                 pass
00221 
00222         window.close()
00223 
00224     def __CreateFrame(self):
00225         """
00226     The __CreateFrame function creates the GUI layout for the Authentication Utility.
00227     It is called by __init__ and returns a list of lists that contains all the elements
00228     that will be displayed in the window.
00229 
00230     Args:
00231         self: Access the class attributes and methods
00232 
00233     Returns:
00234         A list of lists
00235 
00236     Doc Author:
00237         Trelent
00238     """
00239         sg.theme('Default1')
00240 
00241         line00 = [sg.HSeparator()]
00242 
00243         line0 = [sg.Image(ImageLoader("logo.png")),
00244                  sg.Push(),
00245                  sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
justification="center"),
00246                  sg.Push(),
00247                  sg.Push()]
00248 
00249         line1 = [sg.HSeparator()]
00250 
00251         line2 = [sg.Push(),
00252                  sg.Text("Utah Real Estate Key: ", justification="center"),
00253                  sg.Push()]
00254 
00255         line3 = [sg.Push(),
00256                  sg.Input(default_text="", key="-ureAuth-", disabled=False,
00257                           size=(40, 1)),
00258                  sg.Push()]
00259 
00260         line4 = [sg.HSeparator()]
00261 
00262         line5 = [sg.Push(),
00263                  sg.Text("Construction Monitor Key: ", justification="center"),
00264                  sg.Push()]
00265 
00266         line6 = [sg.Push(),
00267                  sg.Input(default_text="", key="-cmAuth-", disabled=False,
00268                           size=(40, 1)),
```

```
00269                   sg.Push()]
00270
00271         line7 = [sg.HSeparator()]
00272
00273         line8 = [sg.Push(),
00274                  sg.Text(self.outcomeText, justification="center"),
00275                  sg.Push()]
00276
00277         line9 = [sg.HSeparator()]
00278
00279         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00280
00281         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10]
00282
00283         return layout
```

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
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
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221     The BatchProcessingUtahRealestateCom function is a function that takes in the
valueObject and uses it to
00222         update the progress bar. It also takes in self, which contains all the
necessary information for this
00223         function to work properly. The BatchProcessingUtahRealestateCom function
will then use requests to get data from
00224         UtahRealestate.com's ReST API and store it into a pandas DataFrame object
called __df (which is local). This
00225         process will be repeated until all the data has been collected from
UtahRealestate.com's ReST API, at which point __df will contain all
00226
00227     Args:
00228         self: Represent the instance of the class
00229         valueObject: Pass the value of a progress bar to the function
00230
00231     Returns:
00232         A dataframe of the scraped data
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 *
200}",
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)
```

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
until
00016     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
popup window.
00011 The function also opens up the log folder upon program exit.
00012
00013 Args:
00014     textString: Display the error message
00015
00016 Returns:
00017     Nothing, but it does print an error message to the console
00018
00019 Doc Author:
00020     Willem van der Schans, Trelent AI
00021 """
00022     PopupWrapped(
00023         f"ERROR @ {textString} \n"
00024         f"Log folder will be opened upon program exit",
00025         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
on http://www.apache.org/licenses/
00002
00003
```

RESTError.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 RESTErrorPrint
00008
00009
00010 def RESTError(response):
00011     """
00012 The RESTError 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         RESTErrorPrint(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         RESTErrorPrint(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         RESTErrorPrint(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         RESTErrorPrint(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         RESTErrorPrint(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         RESTErrorPrint(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
(Imageloader.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 #


00005 #
00006 #
00007 #

00008 #
00009 #

00010 #
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         self.__parameterDict = ParameterDict
00042         self.__restDomain = RestDomain
00043         self.__headerDict = HeaderDict
00044         self.__columnSelection = ColumnSelection
00045         self.__type = Type
00046         self.dataframe = None
00047
00048         self.__layout = None
00049         self.__batches = BatchesNum
00050         self.__window = None
00051         self.__batch_counter = 0
00052
00053     def BatchGuiShow(self):
00054         """
00055     The BatchGuiShow function is called by the BatchGui function. It creates a
progress bar layout and then calls the createGui function to create a GUI for batch
processing.
00056
00057     Args:
00058         self: Represent the instance of the class
00059
00060     Returns:
00061         The __type of the batchgui class
00062
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
00066         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
bar and text
00191     in the GUI. It takes in one argument, which is an object containing information
about the
00192     current batch number. The ProgressUpdater function then checks if this value has
changed from
00193     the last time it was called (i.e., if we are on a new batch). If so, it updates
both the progress
00194     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(
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
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
string.
00015 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]),
"API_Calls"))
00028         __path = normpath(join(__path, "Images"))
00029         __path = join(__path, file).replace("\\", "/")
00030
00031         image = Image.open(__path)
00032
00033         __buff = BytesIO()
00034
00035         image.save(__buff, format="png")
00036
00037         img_str = base64.b64encode(__buff.getvalue())
00038
00039         return img_str
00040     except Exception as e:
00041         # We cannot log this error like other errors due to circular imports
00042         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
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
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, then
creates a layout based on that information.
00044     Args:
00045         self: Refer to the current instance of a class
00046     Returns:
00047         A list of lists
00048     Doc Author:
00049         Willem van der Schans, Trelent AI
00050     """
00051         sg.theme('Default1')
00052         __Line1 = None
00053         __Line2 = None
00054
00055         if self.__type == "notice":
00056             __Line1 = [sg.Push(),
00057                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00058                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00059             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00060         elif self.__type == "noticeLarge":
00061             __Line1 = [sg.Push(),
00062                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"),
justification="center"),
00063                        sg.Text(self.__text, justification="center",
key="-textField-"), sg.Push()]
00064             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00065         elif self.__type == "errorLarge":
00066             __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
00085           if self.__type == "progress":
00086               self.__layout = [__Line1, ]
00087           else:
00088               self.__layout = [__Line1, __Line2]
00089
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 &amp;quot;Ok&amp;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,
finalize=True,
00130                                          modal=True,
00131                                          keep_on_top=True,
00132                                          disable_close=False,
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
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
00149                 except Exception as e:
00150                     print(
00151                         f"PopupWrapped.py | Error = {e} | Log Folder not found please
search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")
00152
00153             self.__windowObj.close()
00154
00155     def stopWindow(self):
00156         """
00157     The stopWindow function is used to close the window object that was created in
the startWindow function.
00158     This is done by calling the close() method on self.__windowObj, which will cause
it to be destroyed.
00159     Args:
00160         self: Represent the instance of the class
00161     Returns:
00162         The window object
00163     Doc Author:
00164         Willem van der Schans, Trelent AI
00165     """
00166         self.__windowObj.close()
00167
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 creates
00172     a loading effect for when something is being processed.
00173     Args:
00174         self: Refer to the object itself
00175         sleep: Control the speed of the text update
00176     Returns:
00177         A string that is the current text of the text field
00178     Doc Author:
00179         Willem van der Schans, Trelent AI
00180     """
00181         self.__counter += 1
00182         if self.__counter == 4:
00183             self.__counter = 1
00184         newString = ""
00185         if self.__type == "notice":
00186             pass
00187         elif self.__type == "error":
00188             pass
00189         elif self.__type == "progress":
00190             newString = f"{self.__text}{'.' * self.__counter}"
00191         self.__windowObj.write_event_value('update-textField-', newString)
00192
00193         time.sleep(sleep)
00194
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 checks
if the event starts with 'update'.
00200         If it does, it will take everything after 'update' as a key for updating that
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, \
00016     ConstructionMonitorMain
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
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
00050     def __ShowGui(self, layout, text):
00051
00052         """
00053     The __ShowGui function is the main function that displays the GUI.
00054     It takes two arguments: layout and text. Layout is a list of lists, each containing
a tuple with three elements:
00055         1) The type of element to be displayed (e.g., &amp;quot;Text&amp;quot;,
&amp;quot;InputText&amp;quot;, etc.)
00056         2) A dictionary containing any additional parameters for that element (e.g.,
size, default value, etc.)
00057         3) An optional key name for the element (used in event handling). If no key
name is provided then one will be generated automatically by PySimpleGUIQt based on its
position in the layout list
00058
00059     Args:
00060         self: Represent the instance of the class
00061         layout: Pass the layout of the window to be created
00062         text: Set the title of the window
00063
00064     Returns:
00065         A window object
00066
```

```
00067     Doc Author:
00068         Willem van der Schans, Trelent AI
00069     """
00070         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00071                            finalize=True,
00072                            icon=ImageLoader("taskbar_icon.ico"))
00073
00074         while True:
00075             event, values = window.read()
00076
00077             if event == "Construction Monitor":
00078                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API
Call-----------------")
00079                 ConstructionMonitorMain(ConstructionMonitorInit())
00080                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API
Call---------------------\n")
00081             elif event == "Utah Real Estate":
00082                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API
Call-----------------")
00083                 UtahRealEstateMain(UtahRealEstateInit())
00084                 print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API
Call---------------------\n")
00085             elif event == "Realtor.Com":
00086                 print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
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                 Cencus()
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
main window.
00125     It returns a list of lists, which is then passed to sg.Window() as its layout
parameter.
00126
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, "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"),
justification="center"),
00150                  sg.Push()]
00151
00152         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)),
sg.Push(),
00153                   sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00154
00155         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
sg.Button("Census", size=(20, None)),
00156                   sg.Push()]]
00157
00158         line8 = [sg.HSeparator()]
00159
00160         line9 = [sg.Push(),
00161                  sg.Text("Utilities", font=("Helvetica", 10, "bold"),
justification="center"),
00162                  sg.Push()]
00163
00164         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00165                    sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00166
00167         line11 = [sg.HSeparator()]
00168
00169         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10,
line11]
00170
00171         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 Cencus:
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
setup that might be necessary.
00018     The __init__ function can take arguments, but it doesn't have to.
00019
00020     Args:
00021         self: Represent the instance of the class
00022         state_arg: Set the state_arg attribute of the class
00023         year_arg: Set the year of data to be retrieved
00024
00025     Returns:
00026         A popupwrapped object
00027
00028     Doc Author:
00029         Willem van der Schans, Trelent AI
00030     """
00031         self.state_arg = state_arg
00032         self.year_arg = year_arg
00033         self.uiString = None
00034         self.link = None
00035
00036         self.__showUi()
00037         print(self.link)
00038         F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00039         self.uiString = (
00040             f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg}
data retrieved \n Data Saved at {F.getPath()}")
00041
00042         PopupWrapped(text=self.uiString, windowType="noticeLarge")
00043
00044     def __showUi(self):
00045
00046         """
00047     The __showUi function is a function that creates a progress bar window.
00048     The __showUi function takes class variables and returns a windowobj.
00049
00050
00051     Args:
00052         self: Represent the instance of the class
00053
00054     Returns:
00055         The uiobj variable
00056
00057     Doc Author:
00058         Willem van der Schans, Trelent AI
00059     """
00060         uiObj = PopupWrapped(text="Cenus Request running", windowType="progress",
error=None)
00061
00062         threadGui = threading.Thread(target=self.__dataGetter,
00063                                      daemon=False)
00064         threadGui.start()
00065
00066         while threadGui.is_alive():
00067             uiObj.textUpdate()
00068             uiObj.windowPush()
00069         else:
00070             uiObj.stopWindow()
```

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

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

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

```
00275
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
returned from a query.
00359     This function is called by the __init__ function and sets the self.__record_val
variable with this value.
00360 
00361     Args:
00362         self: Represent the instance of the class
00363 
00364     Returns:
00365         The total number of records in the database
00366 
00367     Doc Author:
00368         Willem van der Schans, Trelent AI
00369     """
00370         __count_resp = None
00371 
00372         try:
00373 
00374             __temp_param_dict = copy.copy(self.__parameterDict)
00375 
00376             __count_resp = requests.post(url=self.__restDomain,
00377                                          headers=self.__headerDict,
00378                                          json=__temp_param_dict)
00379 
00380             if __count_resp.status_code != 200:
00381                 RESTError(__count_resp)
00382 
00383         except requests.exceptions.Timeout as e:
00384             print(e)
00385             RESTError(790)
00386             raise SystemExit(790)
00387         except requests.exceptions.TooManyRedirects as e:
00388             print(e)
00389             RESTError(791)
00390             raise SystemExit(791)
00391         except requests.exceptions.MissingSchema as e:
00392             print(e)
00393             RESTError(1101)
00394         except requests.exceptions.RequestException as e:
00395             print(e)
00396             RESTError(405)
00397             raise SystemExit(405)
00398 
00399         __count_resp = __count_resp.json()
00400 
00401         self.__record_val = __count_resp["hits"]["total"]["value"]
00402 
00403         del __count_resp, __temp_param_dict
00404 
00405     def __getCountUI(self):
00406 
00407         """
00408     The __getCountUI function is a wrapper for the __getCount function.
00409     It allows the user to run __getCount in a separate thread, so that they can
continue working while it runs.
```

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

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
to run before anything else in your class.
00018
00019     Args:
00020         self: Represent the instance of the class
00021
00022     Returns:
00023         A new object
00024
00025     Doc Author:
00026         Willem van der Schans, Trelent AI
00027     """
00028         self.__page_html = None
00029         self.__update_date = None
00030         self.__last_date = None
00031         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00032         self.__linkDict = {}
00033         self.dfState = None
00034         self.dfCounty = None
00035         self.dfZip = None
00036         self.uiString = "Files Saved to \n"
00037
00038         page_html = requests.get("https://www.realtor.com/research/data/").text
00039         self.__page_html = BeautifulSoup(page_html, "html.parser")
00040
00041         self.__linkGetter()
00042         self.__showUi()
00043
00044         PopupWrapped(text=self.uiString, windowType="noticeLarge")
00045
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
00074     def __linkGetter(self):
00075
00076         """
00077     The __linkGetter function is a private function that takes the idDict dictionary
and adds
00078     a link to each entry in the dictionary. The link is used to access historical
data for each
00079     scope symbol.
00080
00081     Args:
00082         self: Refer to the object itself
00083
00084     Returns:
00085         A dictionary of all the links to the history pages
00086
00087     Doc Author:
00088         Willem van der Schans, Trelent AI
00089     """
00090         for key, value in self.__idDict.items():
00091             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00092                 try:
00093                     for nestedRow in row.find_all("a"):
00094                         if "History" in str(nestedRow.get("href")) and key in
str(nestedRow.get("href")):
00095                             self.__idDict[key] = {"id": value, "link":
nestedRow.get("href")}
00096                 except Exception as e:
00097                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links
for realtor.com")
00098                     RESTError(801)
00099                     raise SystemExit(801)
00100
00101     def __dataUpdater(self):
00102
00103         """
00104     The __dataUpdater function is a private function that updates the dataframes for
each of the three
00105         types of realtor data. It takes class variables and return the path to the
saved file. The function first creates an empty
00106         dictionary called tempdf, then iterates through each key in self.__idDict
(which contains all three ids).
00107         For each key, it reads in a csv file from the link associated with that id
and saves it to tempdf as a pandas
00108         DataFrame object. Then, depending on which type of realtor data we are dealing
with (State/County/Zip), we save
00109
00110
00111     Args:
00112         self: Access the attributes and methods of the class
00113
00114     Returns:
00115         The path of the saved file
00116
00117     Doc Author:
00118         Willem van der Schans, Trelent AI
00119     """
00120         for key, value in self.__idDict.items():
00121             tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00122
00123             if key == "State":
00124                 self.dfState = tempdf
00125             elif key == "County":
00126                 self.dfCounty = tempdf
00127             elif key == "Zip":
00128                 self.dfZip = tempdf
00129
00130             FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00131             self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

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
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
00052     def __ShowGui(self, layout, text):
00053
00054         """
00055     The __ShowGui function is a helper function that creates the GUI window and
displays it to the user.
00056     It takes in two parameters: layout, which is a list of lists containing all the
elements for each row;
00057     and text, which is a string containing what will be displayed as the title of
the window. The __ShowGui
00058     method then uses these parameters to create an instance of sg.Window with all
its attributes set accordingly.
00059
00060     Args:
00061         self: Refer to the current class instance
00062         layout: Pass the layout of the window to be created
00063         text: Set the title of the window
00064
00065     Returns:
00066         A dictionary of values
00067
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
```

```
00070     """
00071         window = sg.Window(text, layout, grab_anywhere=False,
return_keyboard_events=True,
00072                            finalize=True,
00073                            icon=ImageLoader("taskbar_icon.ico"))
00074
00075         while True:
00076             event, values = window.read()
00077
00078             if event == "Submit":
00079                 try:
00080                     self.__SetValues(values)
00081                     break
00082                 except Exception as e:
00083                     print(e)
00084                     RESTError(993)
00085                     raise SystemExit(993)
00086             elif event == sg.WIN_CLOSED or event == "Quit":
00087
00088                 break
00089
00090         window.close()
00091
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
displayed in the window.
00097         Each element is defined by its type and any additional parameters needed to
define it.
00098
00099     Args:
00100
00101     Returns:
00102         A list of lists, which is used to create the gui
00103
00104     Doc Author:
00105         Willem van der Schans, Trelent AI
00106     """
00107         sg.theme('Default1')
00108
00109         line00 = [sg.HSeparator()]
00110
00111         line0 = [sg.Image(ImageLoader("logo.png")),
00112                  sg.Push(),
00113                  sg.Text("Utah Real Estate Utility", font=("Helvetica", 12,
"bold"), justification="center"),
00114                  sg.Push(),
00115                  sg.Push()]
00116
00117         line1 = [sg.HSeparator()]
00118
00119         line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00120                  sg.DropDown(default_value="Active", values=["Active", "Closed"],
key="-status-", size=(31, 1))]
00121
00122         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00123                  sg.DropDown(default_value="Listing Date", values=["Listing
Date", "Modification Date", "Close Date"],
00124                              key="-type-", size=(31, 1))]
00125
00126         line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00127                  sg.Input(default_text=(date.today() -
timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
00128                           disabled=False, size=(20, 1)),
00129                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-start_date-', target="-DateStart-")]
00130
00131         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00132                  sg.Input(default_text=(date.today().strftime("%Y-%m-%d")),
key="-DateEnd-", disabled=False,
00133                           size=(20, 1)),
00134                  sg.CalendarButton("Select Date", format="%Y-%m-%d",
key='-end_date-', target="-DateEnd-")]
00135
```

```
00136         line6 = [[sg.Text("Column Sub-Selection : ", size=(23, None),
justification="Right"),
00137                   sg.Checkbox(text="", default=True, key="-selectionFlag-",
size=(15, 1)),
00138                   sg.Push()]]
00139
00140         line7 = [sg.HSeparator()]
00141
00142         line8 = [sg.Push(),
00143                  sg.Text("File Settings", font=("Helvetica", 12, "bold"),
justification="center"),
00144                  sg.Push()]
00145
00146         line9 = [sg.HSeparator()]
00147
00148         line10 = [sg.Text("Appending File : ", size=(15, None),
justification="Right"),
00149                   sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00150                            size=(20, 1)),
00151                   sg.FileBrowse("Browse File", file_types=[("csv files",
"*.csv")], key='-append_file-',
00152                                 target="-AppendingFile-")]
00153
00154         line11 = [sg.HSeparator()]
00155
00156         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00157
00158         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7,
line8, line9, line10, line11,
00159                   line12]
00160
00161         return layout
00162
00163     def __SetValues(self, values):
00164
00165         """
00166     The __SetValues function is used to set the values of the variables that are used
in the
00167         __GetData function. The values are passed from a dictionary called 'values'
which is created
00168        by parsing through an XML file using ElementTree. This function also sets
default values for
00169        some of these variables if they were not specified in the XML file.
00170
00171     Args:
00172         self: Represent the instance of the class
00173         values: Pass the values from the gui to this function
00174
00175     Returns:
00176         A dictionary with the following keys:
00177
00178     Doc Author:
00179         Willem van der Schans, Trelent AI
00180     """
00181         self.StandardStatus = values["-status-"]
00182
00183         self.ListedOrModified = values["-type-"]
00184
00185         if values["-DateStart-"] != "":
00186             self.dateStart = values["-DateStart-"]
00187         else:
00188             self.dateStart = (date.today() -
timedelta(days=14)).strftime("%Y-%m-%d")
00189
00190         if values["-DateEnd-"] != "":
00191             self.dateEnd = values["-DateEnd-"]
00192         else:
00193             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00194
00195         if values['-selectionFlag-']:
00196             self.select =
"ListingKeyNumeric,StateOrProvince,CountyOrParish,City,PostalCity,PostalCode,Subdivisi
onName," \
00197
"StreetName,StreetNumber,ParcelNumber,UnitNumber,UnparsedAddress,MlsStatus,CloseDate,"
\
```

```
00198
"ClosePrice,ListPrice,OriginalListPrice,LeaseAmount,LivingArea,BuildingAreaTotal,LotSi
zeAcres," \
00199
"LotSizeSquareFeet,LotSizeArea,RoomsTotal,Stories,BedroomsTotal,MainLevelBedrooms,Park
ingTotal," \
00200
"BasementFinished,AboveGradeFinishedArea,TaxAnnualAmount,YearBuilt,YearBuiltEffective,
" \
00201
"OnMarketDate,ListingContractDate,CumulativeDaysOnMarket,DaysOnMarket,PurchaseContract
Date," \
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
00213 class UtahRealEstateMain:
00214
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
00267     def mainFunc(self):
00268
00269         """
00270     The mainFunc function is the main function of this module. It will be called by
the GUI when a user clicks on
00271     the &amp;quot;Run&amp;quot; button in the GUI. The mainFunc function should contain all
of your code for running your program, and it
00272     should return a dataframe that contains all the data you want to display in your
final report.
00273
00274     Args:
00275         self: Reference the object itself
00276
00277     Returns:
00278         A dataframe
00279
00280     Doc Author:
00281         Willem van der Schans, Trelent AI
00282     """
00283         passFlag = False
00284
00285         while not passFlag:
00286             if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00287                 try:
00288                     f = open(self.keyPath, "rb")
00289                     key = f.readline()
00290                     f.close()
00291                     f = open(self.filePath, "rb")
00292                     authDict = json.load(f)
00293                     fernet = Fernet(key)
00294                     authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00295                     self.__headerDict = {authDict["ure"]["parameter"]: authkey}
00296                     passFlag = True
00297                 except Exception as e:
00298                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening
AuthUtil")
00299                     AuthUtil()
00300             else:
00301                 AuthUtil()
00302
00303         self.__ParameterCreator()
00304
00305         self.__getCountUI()
00306
00307         self.__batches = BatchCalculator(self.__record_val, None)
00308
00309         if self.__batches != 0:
00310             startTime = datetime.datetime.now().replace(microsecond=0)
00311             BatchInputGui(self.__batches)
00312             print(f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Request for {self.__batches} Batches sent to server")
00313             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00314
ParameterDict=self.__parameterString,
00315                                               HeaderDict=self.__headerDict,
00316                                               BatchesNum=self.__batches,
00317                                               Type="utah_real_estate")
00318             BatchGuiObject.BatchGuiShow()
00319             self.dataframe = BatchGuiObject.dataframe
00320             print(
00321                 f"{datetime.datetime.today().strftime('%m-%d-%Y
%H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and
{self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S',
time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
00322             FileSaver("ure", self.dataframe, self.__appendFile)
00323         else:
00324             RESTError(994)
00325             raise SystemExit(994)
00326
```

```
00327     def __ParameterCreator(self):
00328         """
00329     The __ParameterCreator function is used to create the filter string for the ReST
API call.
00330     The function takes in a siteClass object and extracts all of its parameters into
a dictionary.
00331     It then creates an appropriate filter string based on those parameters.
00332
00333     Args:
00334         self: Bind the object to the class
00335
00336     Returns:
00337         A string to be used as the parameter in the api call
00338
00339     Doc Author:
00340         Willem van der Schans, Trelent AI
00341     """
00342         filter_string = ""
00343
00344         __Source_dict = {key: value for key, value in
self.__siteClass.__dict__.items() if
00345                          not key.startswith('  ') and not callable(key)}
00346
00347         self.__appendFile = __Source_dict["append_file"]
00348         __Source_dict.pop("append_file")
00349
00350         temp_dict = copy.copy(__Source_dict)
00351         for key, value in temp_dict.items():
00352             if value is None:
00353                 __Source_dict.pop(key)
00354             else:
00355                 pass
00356
00357         if __Source_dict["ListedOrModified"] == "Listing Date":
00358             filter_string =
f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContr
actDate%20le%20{__Source_dict['dateEnd']}"
00359         elif __Source_dict["ListedOrModified"] == "Modification Date":
00360             filter_string =
f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%2
0ModificationTimestamp%20le%20{__Source_dict['dateEnd']}T:23:59:59Z"
00361         elif __Source_dict["ListedOrModified"] == "Close Date":
00362             filter_string =
f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__So
urce_dict['dateEnd']}"
00363
00364         filter_string = filter_string +
f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['Standard
Status']}'"
00365
00366         if __Source_dict["select"] is not None:
00367             filter_string = filter_string + f'&$select={__Source_dict["select"]}'
00368
00369         self.__parameterString = filter_string
00370
00371     def __getCount(self):
00372         """
00373     The __getCount function is used to determine the number of records that will be
returned by the query.
00374     This function is called when a user calls the count() method on a ReST object.
The __getCount function uses
00375     the $count parameter in OData to return only an integer value representing how
many records would be returned
00376     by the query.
00377
00378     Args:
00379         self: Represent the instance of the class
00380
00381     Returns:
00382         The number of records in the data set
00383
00384     Doc Author:
00385         Willem van der Schans, Trelent AI
00386     """
00387         __count_resp = None
00388
00389         try:
```

```
00390             __count_resp =
requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00391                                         headers=self.__headerDict)
00392
00393             if __count_resp.status_code != 200:
00394                 RESTError(__count_resp)
00395                 raise SystemExit(0)
00396
00397             self.__record_val = int(__count_resp.json()["@odata.count"])
00398
00399         except requests.exceptions.Timeout as e:
00400             print(e)
00401             RESTError(790)
00402             raise SystemExit(790)
00403         except requests.exceptions.TooManyRedirects as e:
00404             print(e)
00405             RESTError(791)
00406             raise SystemExit(791)
00407         except requests.exceptions.MissingSchema as e:
00408             print(e)
00409             RESTError(1101)
00410         except requests.exceptions.RequestException as e:
00411             print(e)
00412             RESTError(405)
00413             raise SystemExit(405)
00414
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