Gardner API Utility Documentation 1.2.0

Willem van der Schans, DoxyGen

1	Class Index	1
	1.1 Class List	1
2	File Index	3
	2.1 File List	3
3	Class Documentation	5
	3.1 AuthUtil.AuthUtil Class Reference	5
	3.1.1 Detailed Description	5
	3.1.2 Constructor & Destructor Documentation	6
	3.1.2.1init()	6
	3.1.3 Member Function Documentation	8
	3.1.3.1CreateFrame()	8
	3.1.3.2SetValues()	10
	3.1.3.3ShowGui()	12
	3.1.4 Member Data Documentation	14
	3.1.4.1 append_file	14
	3.1.4.2 file_name	14
	3.1.4.3 filePath	15
	3.1.4.4 jsonDict	15
	3.1.4.5 k	15
	3.1.4.6 keyFlag	15
	3.1.4.7 keyPath	15
	3.1.4.8 ListedOrModified	15
	3.1.4.9 outcomeText	16
	3.1.4.10 passFlagCm	16
	3.1.4.11 passFlagUre	16
	3.1.4.12 popupFlag	16
	3.1.4.13 StandardStatus	16
	3.2 BatchProcessing.BatchProcessorConstructionMonitor Class Reference	17
	3.2.1 Detailed Description	17
	3.2.2 Constructor & Destructor Documentation	17
	3.2.2.1init()	18
	3.2.3 Member Function Documentation	19
	3.2.3.1 ConstructionMonitorProcessor()	19
	3.2.3.2 FuncSelector()	
	3.2.4 Member Data Documentation	
	3.2.4.1columnSelection	
	3.2.4.3headerDict	

3.2.4.4maxRequests		22
3.2.4.5numBatches	:	22
3.2.4.6parameterDict	:	22
3.2.4.7requestCalls	:	22
3.2.4.8requestCount	:	22
3.2.4.9restDomain	:	23
3.2.4.10 dataframe	:	23
3.2.4.11 valueObject	:	23
3.3 BatchProcessing.BatchProcessorUtahRealEstate Class Reference	:	23
3.3.1 Detailed Description	:	24
3.3.2 Constructor & Destructor Documentation	:	24
3.3.2.1init()	:	24
3.3.3 Member Function Documentation	:	25
3.3.3.1 BatchProcessingUtahRealestateCom()	:	25
3.3.3.2 FuncSelector()	:	26
3.3.4 Member Data Documentation	:	27
3.3.4.1headerDict	:	27
3.3.4.2numBatches	:	27
3.3.4.3parameterString	:	27
3.3.4.4restDomain	:	27
3.3.4.5 dataframe	:	28
3.3.4.6 valueObject	:	28
3.4 BatchProgressGUI.BatchProgressGUI Class Reference	:	28
3.4.1 Detailed Description	:	29
3.4.2 Constructor & Destructor Documentation	:	29
3.4.2.1init()	:	29
3.4.3 Member Function Documentation	:	30
3.4.3.1 BatchGuiShow()	:	30
3.4.3.2 createGui()	:	31
3.4.3.3 CreateProgressLayout()	:	33
3.4.3.4 ProgressUpdater()	:	34
3.4.3.5 TimeUpdater()	:	35
3.4.3.6 ValueChecker()	:	36
3.4.4 Member Data Documentation	:	37
3.4.4.1batch_counter	:	37
3.4.4.2batches	;	38
3.4.4.3columnSelection	:	38
3.4.4.4headerDict	;	38
3.4.4.5 <u>layout</u>	:	38

3.4.4.6parameterDict	 	38
3.4.4.7restDomain	 	38
3.4.4.8type	 	39
3.4.4.9window	 	39
3.4.4.10 dataframe	 	39
3.5 Core.CFBP Class Reference	 	39
3.5.1 Detailed Description	 	40
3.5.2 Constructor & Destructor Documentation	 	40
3.5.2.1init()	 	40
3.5.3 Member Function Documentation	 	41
3.5.3.1dataGetter()	 	41
3.5.3.2showUi()	 	42
3.5.4 Member Data Documentation	 	43
3.5.4.1 link	 	44
3.5.4.2 state_arg	 	44
3.5.4.3 uiString	 	44
3.5.4.4 year_arg	 	44
3.6 Core.ConstructionMonitorInit Class Reference	 	44
3.6.1 Detailed Description	 	45
3.6.2 Constructor & Destructor Documentation	 	45
3.6.2.1init()	 	45
3.6.3 Member Function Documentation	 	47
3.6.3.1CreateFrame()	 	47
3.6.3.2SetValues()	 	49
3.6.3.3ShowGui()	 	50
3.6.4 Member Data Documentation	 	52
3.6.4.1 append_file	 	52
3.6.4.2 auth_key	 	53
3.6.4.3 dateEnd	 	53
3.6.4.4 dateStart	 	53
3.6.4.5 rest_domain	 	53
3.6.4.6 size	 	53
3.6.4.7 SourceInclude	 	53
3.6.4.8 ui_flag	 	54
3.7 Core.ConstructionMonitorMain Class Reference	 	54
3.7.1 Detailed Description	 	54
3.7.2 Constructor & Destructor Documentation	 	55
3.7.2.1init()	 	55
3.7.3 Member Function Documentation		

3.7.3.1getCount()	 . 56
3.7.3.2getCountUI()	 . 57
3.7.3.3ParameterCreator()	 . 59
3.7.3.4 mainFunc()	 . 60
3.7.4 Member Data Documentation	 . 62
3.7.4.1appendFile	 . 62
3.7.4.2batches	 . 63
3.7.4.3columnSelection	 . 63
3.7.4.4headerDict	 . 63
3.7.4.5parameterDict	 . 63
3.7.4.6record_val	 . 63
3.7.4.7restDomain	 . 63
3.7.4.8search_id	 . 64
3.7.4.9siteClass	 . 64
3.7.4.10ui_flag	 . 64
3.7.4.11 dataframe	 . 64
3.8 DataTransfer.DataTransfer Class Reference	 . 64
3.8.1 Detailed Description	 . 65
3.8.2 Constructor & Destructor Documentation	 . 65
3.8.2.1init()	 . 65
3.8.3 Member Function Documentation	 . 65
3.8.3.1 getValue()	 . 66
3.8.3.2 setValue()	 . 66
3.8.3.3 while Value()	 . 67
3.8.4 Member Data Documentation	 . 68
3.8.4.1value	 . 68
3.9 FileSaver.FileSaver Class Reference	 . 68
3.9.1 Detailed Description	 . 69
3.9.2 Constructor & Destructor Documentation	 . 69
3.9.2.1init()	 . 69
3.9.3 Member Function Documentation	 . 70
3.9.3.1 getPath()	 . 71
3.9.4 Member Data Documentation	 . 71
3.9.4.1 appendFlag	 . 71
3.9.4.2 data	 . 71
3.9.4.3 dataAppending	 . 72
3.9.4.4 docPath	 . 72
3.9.4.5 fileName	 . 72
3.9.4.6 outputFrame	 . 72

3.9.4.7 primaryKey	. 72
3.9.4.8 uiFlag	. 72
3.10 API_Calls.Initializer.initializer Class Reference	. 73
3.10.1 Detailed Description	. 73
3.10.2 Constructor & Destructor Documentation	. 73
3.10.2.1init()	. 73
3.10.3 Member Function Documentation	. 74
3.10.3.1CreateFrame()	. 75
3.10.3.2ShowGui()	. 76
3.10.4 Member Data Documentation	. 78
3.10.4.1 classObj	. 78
3.11 PopupWrapped.PopupWrapped Class Reference	. 79
3.11.1 Detailed Description	. 79
3.11.2 Constructor & Destructor Documentation	. 79
3.11.2.1init()	. 80
3.11.3 Member Function Documentation	. 81
3.11.3.1createLayout()	. 81
3.11.3.2createWindow()	. 82
3.11.3.3 openFile()	. 85
3.11.3.4 stopWindow()	. 85
3.11.3.5 textUpdate()	. 86
3.11.3.6 windowPush()	. 87
3.11.4 Member Data Documentation	. 87
3.11.4.1counter	. 87
3.11.4.2docpath	. 87
3.11.4.3error	. 88
3.11.4.4errorFlag	. 88
3.11.4.5layout	. 88
3.11.4.6text	. 88
3.11.4.7thread	. 88
3.11.4.8type	. 88
3.11.4.9windowObj	. 89
3.12 Core.realtorCom Class Reference	. 89
3.12.1 Detailed Description	. 89
3.12.2 Constructor & Destructor Documentation	. 90
3.12.2.1init()	. 90
3.12.3 Member Function Documentation	. 91
3.12.3.1dataUpdater()	. 91
3.12.3.2linkGetter()	. 92

3.12.3.3showUi()	93
3.12.4 Member Data Documentation)4
3.12.4.1idDict) 4
3.12.4.2last_date	95
3.12.4.3linkDict	95
3.12.4.4page_html	95
3.12.4.5update_date	95
3.12.4.6 dfCounty	95
3.12.4.7 dfState	95
3.12.4.8 dfZip	96
3.12.4.9 uiString	96
3.13 Settings.settings Class Reference	96
3.13.1 Detailed Description	96
3.13.2 Member Data Documentation	96
3.13.2.1 settingCFBPLink	97
3.13.2.2 settingCMRestDomain	97
3.13.2.3 settingDownloadSourceLink	97
3.13.2.4 settingGenerationToolLink	97
3.13.2.5 settingGithubApiUrl) 7
3.13.2.6 settingRealtorLink	98
3.13.2.7 settingURERestDomain	98
3.13.2.8 settingVersion	98
3.14 Core.UtahRealEstateInit Class Reference	98
3.14.1 Detailed Description	99
3.14.2 Constructor & Destructor Documentation	99
3.14.2.1init()	99
3.14.3 Member Function Documentation)0
3.14.3.1CreateFrame())0
3.14.3.2SetValues())2
3.14.3.3ShowGui())3
3.14.4 Member Data Documentation)5
3.14.4.1 append_file)5
3.14.4.2 dateEnd)6
3.14.4.3 dateStart)6
3.14.4.4 file_name)6
3.14.4.5 ListedOrModified)6
3.14.4.6 select)6
3.14.4.7 StandardStatus)6
3.15 Core.UtahRealEstateMain Class Reference)7

	3.15.1 Detailed Description
	3.15.2 Constructor & Destructor Documentation
	3.15.2.1init()
	3.15.3 Member Function Documentation
	3.15.3.1getCount()
	3.15.3.2getCountUI()
	3.15.3.3ParameterCreator()
	3.15.3.4 mainFunc()
	3.15.4 Member Data Documentation
	3.15.4.1appendFile
	3.15.4.2batches
	3.15.4.3dateEnd
	3.15.4.4dateStart
	3.15.4.5headerDict
	3.15.4.6parameterString
	3.15.4.7record_val
	3.15.4.8restDomain
	3.15.4.9siteClass
	3.15.4.10 dataframe
	3.15.4.11 filePath
	3.15.4.12 key
	3.15.4.13 keyPath
4	File Documentation 119
	4.1initpy
	4.2 _mainpy
	4.3 AuthUtil.py
	4.4 BatchProcessing.py
	4.5 DataSupportFunctions.py
	4.6 FileSaver.py
	4.7 Settings.py
	4.8 versionChecker.py
	4.9 ErrorPopup.py
	4.10 ErrorPrint.py
	4.11 Logger.py
	4.12 RESTError.py
	4.13 BatchGui.py
	4.14 BatchProgressGUI.py
	4.15 DataTransfer.py

4.16 ImageLoader.py	140
4.17 PopupWrapped.py	140
4.18 Initializer.py	144
4.19 CFBP/Core.py	147
4.20 ConstructionMonitor/Core.py	149
4.21 Realtor/Core.py	155
4.22 UtahRealEstate/Core.py	157

Chapter 1

Class Index

1.1 Class List

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

AuthUtil.AuthUtil
BatchProcessing.BatchProcessorConstructionMonitor
BatchProcessing.BatchProcessorUtahRealEstate
BatchProgressGUI.BatchProgressGUI
Core.CFBP
Core.ConstructionMonitorInit
Core.ConstructionMonitorMain
DataTransfer.DataTransfer
FileSaver.FileSaver
API_Calls.Initializer.initializer
PopupWrapped
Core.realtorCom
Settings.settings
Core.UtahRealEstateInit
Core.UtahRealEstateMain

2 Class Index

Chapter 2

File Index

2.1 File List

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

initpy
_mainpy
AuthUtil.py
BatchProcessing.py
DataSupportFunctions.py
FileSaver.py
Settings.py
versionChecker.py
ErrorPopup.py
ErrorPrint.py
Logger.py
RESTError.py
BatchGui.py
BatchProgressGUI.py
DataTransfer.py
ImageLoader.py
PopupWrapped.py
Initializer.py
CFBP/Core.py
ConstructionMonitor/Core.py
Realtor/Core.py
Litah Raai Estato / Cora ny

4 File Index

Chapter 3

Class Documentation

3.1 AuthUtil.AuthUtil Class Reference

Public Member Functions

def __init__ (self)

Public Attributes

- StandardStatus
- ListedOrModified
- file name
- append_file
- keyPath
- filePath
- k
- keyFlag
- jsonDict
- passFlagUre
- passFlagCm
- outcomeText
- popupFlag

Private Member Functions

- def __SetValues (self, values)
- def <u>ShowGui</u> (self, layout, text)
- def __CreateFrame (self)

3.1.1 Detailed Description

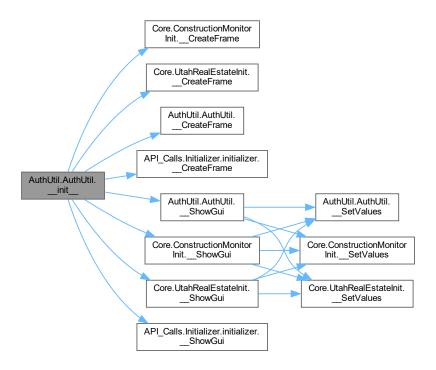
Definition at line 18 of file AuthUtil.py.

3.1.2 Constructor & Destructor Documentation

```
3.1.2.1 __init__()
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
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
         The __init__ function is called when the class is instantiated.
00023
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
00027
             self: Represent the instance of the class
00028
00029
         Returns:
00030
             None
00031
00032
         Doc Author:
         Willem van der Schans, Trelent AI
00033
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 <math>n "
00047
                                "Submitting will overwrite any old values in an unrecoverable manner."
00048
00049
             if os.path.exists(self.filePath):
00050
00051
             else:
00052
                 if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
00053
                     os.mkdir(self.filePath)
00054
00055
                     os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
00056
                     os.mkdir(self.filePath)
00057
00058
             if os.path.exists(self.keyPath):
00059
00060
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
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"))
              except Exception as e:
    # Logging
00084
00085
                  00086
00087
    Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
00088
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
           trv:
00097
     00098
           except Exception as e:
00099
               # Logging
              00100
00101
     {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:



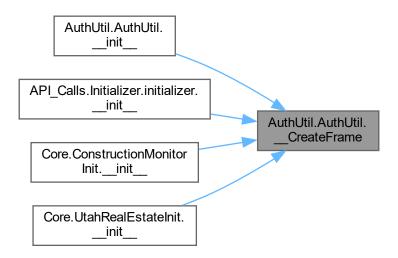
3.1.3 Member Function Documentation

3.1.3.1 __CreateFrame()

Definition at line 235 of file AuthUtil.py.

```
def __CreateFrame(self):
00235
00236
00237
          The __CreateFrame function creates the GUI layout for the Authentication Utility.
00238
          It is called by __init__ and returns a list of lists that contains all the elements
00239
          that will be displayed in the window.
00240
00241
00242
              self: Access the class attributes and methods
00243
00244
00245
             A list of lists
00246
00247
         Doc Author:
          Trelent
00248
00249
00250
              sq.theme('Default1')
00251
00252
              line00 = [sq.HSeparator()]
00253
00254
              line0 = [sg.Image(ImageLoader("logo.png")),
00255
                       sa.Push(),
00256
                       sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
     justification="center"),
00257
                       sg.Push(),
00258
                       sq.Push()]
00259
00260
              line1 = [sg.HSeparator()]
00261
00262
              line2 = [sq.Push(),
                       sg.Text("Utah Real Estate API Key: ", justification="center"),
00263
00264
                       sq.Push()1
00265
00266
              line3 = [sg.Push(),
                       sg.Input(default_text="123", key="-ureAuth-", disabled=False,
00267
00268
                                size=(40, 1)),
                       sq.Push()]
00269
00270
              line4 = [sg.HSeparator()]
00271
00272
00273
              line5 = [sg.Push(),
                       sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
00274
00275
                       sg.Push()]
00276
00277
              line6 = [sg.Push(),
00278
                       sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
00279
                                size=(40, 1)),
00280
                       sq.Push()]
00281
00282
              line7 = [sg.HSeparator()]
00283
00284
              line8 = [sg.Push(),
00285
                       sg.Text(self.outcomeText, justification="center"),
00286
                       sg.Push()]
00287
00288
              line9 = [sg.HSeparator()]
00289
00290
              line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00291
00292
              layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]
00293
00294
              return layout
```

Here is the caller graph for this function:



3.1.3.2 __SetValues()

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

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

Args:

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

Returns

A dictionary of the values entered by the user

Doc Author:

Willem van der Schans, Trelent AI

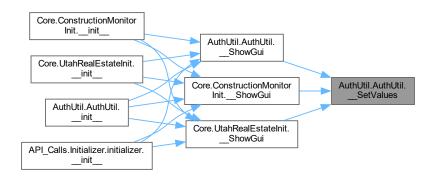
Definition at line 104 of file AuthUtil.py.

```
00104 def __SetValues(self, values):
00105
00106 """
00107 The __SetValues function is called when the user clicks on the "OK" button in the window.
00108 It takes a dictionary of values as an argument, and then uses those values to update
00109 the auth.json file with new keys for both Utah Real Estate and Construction Monitor.
00110
```

```
00111
          Args:
00112
              self: Make the function a method of the class
00113
              values: Store the values that are entered into the form
00114
00115
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
              cmCurrent = None
00122
00123
              keyFile = None
00124
              self.popupFlag = False
00125
00126
              fernet = Fernet(self.k)
00127
00128
00129
                  f = open(self.filePath.joinpath("auth.json"), "r")
00130
                  keyFile = json.load(f)
00131
                  fileFlag = True
00132
              except:
00133
                  fileFlag = False
00134
00135
              # Try initial decoding, if fails pass and write new keys and files
00136
              if fileFlag:
00137
                     ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00138
                  except Exception as e:
00139
00140
                      # Logging
00141
                      00142
      |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE
      functionality will be accessed")
00143
                      ureCurrent = None
00144
00145
00146
                     cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00147
                  except Exception as e:
00148
                      # Logging
                      print(
00149
                         f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ Authutil.py
00150
      |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM
      functionality will be accessed")
00151
                      cmCurrent = None
00152
00153
              if values["-ureAuth-"] != "":
00154
                  self.jsonDict.update(
      {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00155
                  self.passFlagUre = True
00156
00157
              elif ureCurrent is not None:
00158
                 self.jsonDict.update(
00159
                      {"ure": {"parameter": "Authorization", "auth":
     fernet.encrypt(ureCurrent.encode()).decode()}})
00160
                  self.passFlagUre = True
00161
00162
                 pass
00163
00164
              if values["-cmAuth-"] != "":
                 if values["-cmAuth-"].startswith("Basic"):
00165
00166
                      self.jsonDict.update(
00167
                         {"cm": {"parameter": "Authorization",
                                  "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00168
00169
                      self.passFlagCm = True
00170
                  else:
                      PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction
00171
     Monitor",
00172
                                   windowType="AuthError")
                      self.popupFlag = True
00173
00174
00175
              elif ureCurrent is not None:
00176
                self.jsonDict.update(
                      {"cm": {"parameter": "Authorization", "auth":
00177
      fernet.encrypt(cmCurrent.encode()).decode()}})
00178
                  self.passFlagUre = True
00179
              else:
00180
                 pass
00181
00182
              if not self.passFlagUre and not self.passFlagCm:
                  PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction
00183
```

```
Monitor",
00184
                               windowType="errorLarge")
00185
              if self.passFlagCm and not self.passFlagUre:
00186
                 PopupWrapped("Please make sure you provide a key for Utah Real estate",
     windowType="errorLarge")
00187
             if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
00188
                 PopupWrapped("Please make sure you provide a key for Construction Monitor",
     windowType="errorLarge")
00189
             if self.popupFlag:
00190
00191
              else:
               jsonOut = json.dumps(self.jsonDict, indent=4)
00192
                  f = open(self.filePath.joinpath("auth.json"), "w")
00193
                  f.write(jsonOut)
00195
```

Here is the caller graph for this function:



3.1.3.3 __ShowGui()

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.

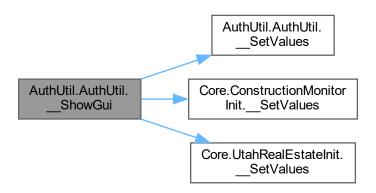
```
Args:
self: Represent the instance of the class
layout: Pass the layout of the gui to be displayed
text: Set the title of the window

Returns:
A window object
```

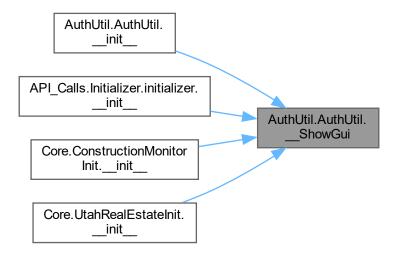
```
Definition at line 196 of file AuthUtil.py.
```

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

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 append_file

AuthUtil.AuthUtil.append_file

Definition at line 38 of file AuthUtil.py.

3.1.4.2 file_name

AuthUtil.AuthUtil.file_name

Definition at line 37 of file AuthUtil.py.

3.1.4.3 filePath

AuthUtil.AuthUtil.filePath

Definition at line 40 of file AuthUtil.py.

3.1.4.4 jsonDict

AuthUtil.AuthUtil.jsonDict

Definition at line 43 of file AuthUtil.py.

3.1.4.5 k

AuthUtil.AuthUtil.k

Definition at line 41 of file AuthUtil.py.

3.1.4.6 keyFlag

AuthUtil.AuthUtil.keyFlag

Definition at line 42 of file AuthUtil.py.

3.1.4.7 keyPath

AuthUtil.AuthUtil.keyPath

Definition at line 39 of file AuthUtil.py.

3.1.4.8 ListedOrModified

AuthUtil.AuthUtil.ListedOrModified

Definition at line 36 of file AuthUtil.py.

3.1.4.9 outcomeText

AuthUtil.AuthUtil.outcomeText

Definition at line 46 of file AuthUtil.py.

3.1.4.10 passFlagCm

AuthUtil.AuthUtil.passFlagCm

Definition at line 45 of file AuthUtil.py.

3.1.4.11 passFlagUre

AuthUtil.AuthUtil.passFlagUre

Definition at line 44 of file AuthUtil.py.

3.1.4.12 popupFlag

AuthUtil.AuthUtil.popupFlag

Definition at line 124 of file AuthUtil.py.

3.1.4.13 StandardStatus

AuthUtil.AuthUtil.StandardStatus

Definition at line 35 of file AuthUtil.py.

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

AuthUtil.py

3.2 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

- dataframe
- valueObject

Private Attributes

- __numBatches
- parameterDict
- __restDomain
- headerDict
- __columnSelection
- __maxRequests
- __requestCount
- __requestCalls
- __dateTracker

3.2.1 Detailed Description

Definition at line 41 of file BatchProcessing.py.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 __init__()

```
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.
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.
         def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):
00043
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
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
             self.__restDomain = RestDomain
00069
             self.__headerDict = HeaderDict
00070
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
```

3.2.3 Member Function Documentation

3.2.3.1 ConstructionMonitorProcessor()

```
\tt def Batch Processing. Batch Processor Construction Monitor. Construction Monitor Processor ( \tt in the monitor Processor ( \tt in t
                              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
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
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:
                          Willem van der Schans, Trelent AI
00108
00109
00110
                             _df = None
00111
                           for callNum in range(0, self.__requestCount):
                                  self.__parameterDict["from"] = 0
00112
00113
00114
                                  if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00115
                                          __batchNum = self.__requestCalls
                                          if __df is None:
00116
00117
                                                  self.__dateTracker = str(date.today())
00118
                                                 self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00119
00120
                                  elif self.__requestCount == 1:
                                          __batchNum = self.__numBatches
self.__dateTracker = str(date.today())
00121
00122
00123
                                  else:
                                          __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size']))
00124
            - (
00125
                                                         self.__requestCount - 1)
                                         \verb|self.__dateTracker = \min(pd.to_datetime(\__df['lastIndexedDate'])).strftime('%Y-%m-%d')|
00126
00127
                                  self.__parameterDict['dateEnd'] = self.__dateTracker
00128
00129
00130
                                  for record in range(0, int(math.ceil(__batchNum))):
00131
                                          if record != 0:
                                                 self.__parameterDict["from"] = record * int(self.__parameterDict["size"])
00132
```

```
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:
                           # Logging
00142
00143
                           print(
                              f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00144
      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:
                           __df = pd.json_normalize(response[counter]["_source"])
00150
                           __df["id"] = response[counter]['_id']
00151
00152
                           __df["county"] = response[counter]["_source"]['county']['county_name']
00153
                           counter += 1
00154
00155
                       for i in range(counter, len(response)):
                           __tdf = pd.json_normalize(response[i]["_source"])
00156
                           __tdf["id"] = response[i]['_id']
00157
                            _tdf["county"] = response[i]["_source"]['county']['county_name']
00158
00159
                           __df = pd.concat([__df, __tdf], ignore_index=True)
00160
              if self.__columnSelection is not None:
    __col_list = StringToList(self.__columnSelection)
00161
00162
                  __col_list.append("id")
00163
                   __col_list.append("county")
00164
00165
              else:
00166
00167
00168
              self.dataframe = __df
00169
              valueObject.setValue(-999)
00170
00171
```

Here is the caller graph for this function:



3.2.3.2 FuncSelector()

```
\begin{tabular}{ll} \tt def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector ( \\ self ) \end{tabular}
```

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

Args:

```
self: Represent the instance of the class
Returns:
The result of the constructionmonitorprocessor function
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 78 of file BatchProcessing.py.
         def FuncSelector(self):
00079
00080
          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.
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:
         Willem van der Schans, Trelent AI
00090
00091
00092
             self.ConstructionMonitorProcessor(self.valueObject)
```

Here is the call graph for this function:

00093



3.2.4 Member Data Documentation

3.2.4.1 columnSelection

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.} \underline{\quad } {\tt columnSelection} \quad [private]$

Definition at line 71 of file BatchProcessing.py.

3.2.4.2 __dateTracker

BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker [private]

Definition at line 76 of file BatchProcessing.py.

3.2.4.3 _headerDict

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.} \underline{\quad \text{headerDict} \quad [private]}$

Definition at line 70 of file BatchProcessing.py.

3.2.4.4 __maxRequests

BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests [private]

Definition at line 73 of file BatchProcessing.py.

3.2.4.5 numBatches

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.} \underline{\quad } {\tt numBatches} \quad [{\tt private}]$

Definition at line 67 of file BatchProcessing.py.

3.2.4.6 __parameterDict

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.} \underline{\hspace{0.5cm}} {\tt parameterDict} \quad [{\tt private}]$

Definition at line 68 of file BatchProcessing.py.

3.2.4.7 requestCalls

BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls [private]

Definition at line 75 of file BatchProcessing.py.

3.2.4.8 __requestCount

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.} \underline{\hspace{0.5cm}} {\tt requestCount} \quad [private]$

Definition at line 74 of file BatchProcessing.py.

3.2.4.9 __restDomain

BatchProcessing.BatchProcessorConstructionMonitor.__restDomain [private]

Definition at line 69 of file BatchProcessing.py.

3.2.4.10 dataframe

BatchProcessing.BatchProcessorConstructionMonitor.dataframe

Definition at line 66 of file BatchProcessing.py.

3.2.4.11 valueObject

 ${\tt BatchProcessing.BatchProcessorConstructionMonitor.valueObject}$

Definition at line 72 of file BatchProcessing.py.

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

· BatchProcessing.py

3.3 BatchProcessing.BatchProcessorUtahRealEstate Class Reference

Public Member Functions

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

Public Attributes

- · dataframe
- valueObject

Private Attributes

- __numBatches
- __parameterString
- restDomain
- headerDict

3.3.1 Detailed Description

Definition at line 172 of file BatchProcessing.py.

3.3.2 Constructor & Destructor Documentation

```
3.3.2.1 __init__()
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.
         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
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:
         Willem van der Schans, Trelent AI
00193
00194
00195
             self.dataframe = None
             self.\__numBatches = NumBatches
00196
             self.__parameterString = ParameterString
00197
             self.\__restDomain = RestDomain
00198
             self.__headerDict = HeaderDict
00199
             self.valueObject = valueObject
00200
00201
```

3.3.3 Member Function Documentation

3.3.3.1 BatchProcessingUtahRealestateCom()

```
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

```
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
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
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
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.ison normalize(response temp, record path=['value'])
00247
00248
                  else:
                      response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch
00249
      * 200}",
00250
                                              headers=self.__headerDict)
00251
00252
                      response_temp = response.json()
```

Here is the caller graph for this function:



3.3.3.2 FuncSelector()

```
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
          The FuncSelector function is a function that takes the valueObject as an argument and then calls the
00204
      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:
          Willem van der Schans, Trelent AI
00215
00216
00217
              self.BatchProcessingUtahRealestateCom(self.valueObject)
00218
```

Here is the call graph for this function:



3.3.4 Member Data Documentation

3.3.4.1 __headerDict

BatchProcessing.BatchProcessorUtahRealEstate.__headerDict [private]

Definition at line 199 of file BatchProcessing.py.

3.3.4.2 __numBatches

BatchProcessing.BatchProcessorUtahRealEstate.__numBatches [private]

Definition at line 196 of file BatchProcessing.py.

3.3.4.3 __parameterString

 ${\tt BatchProcessing.BatchProcessorUtahRealEstate.} \underline{\quad} {\tt parameterString\quad} [{\tt private}]$

Definition at line 197 of file BatchProcessing.py.

3.3.4.4 __restDomain

BatchProcessing.BatchProcessorUtahRealEstate.__restDomain [private]

Definition at line 198 of file BatchProcessing.py.

3.3.4.5 dataframe

BatchProcessing.BatchProcessorUtahRealEstate.dataframe

Definition at line 195 of file BatchProcessing.py.

3.3.4.6 valueObject

 ${\tt BatchProcessing.BatchProcessorUtahRealEstate.valueObject}$

Definition at line 200 of file BatchProcessing.py.

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

· BatchProcessing.py

3.4 BatchProgressGUI.BatchProgressGUI Class Reference

Public Member Functions

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

Public Attributes

· dataframe

Private Attributes

- __parameterDict
- __restDomain
- headerDict
- __columnSelection
- __type
- __layout
- __batches
- window
- __batch_counter

3.4.1 Detailed Description

Definition at line 17 of file BatchProgressGUI.py.

3.4.2 Constructor & Destructor Documentation

```
3.4.2.1 init ()
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.
         def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):
00020
00022
         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
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
             Type: Determine the type of dataframe that {\color{red} \mathbf{is}} being created
00032
00033
             ColumnSelection: Select the columns to be displayed in the gui
00034
00035
         Returns:
00036
             Nothing
```

Doc Author:

00037

```
00039
               Willem van der Schans, Trelent AI
00040
00041
               self.__parameterDict = ParameterDict
               self.__restDomain = RestDomain
self.__headerDict = HeaderDict
00042
00043
00044
               self.__columnSelection = ColumnSelection
00045
               self.__type = Type
00046
               self.dataframe = None
00047
00048
               self.__layout = None
00049
               self.__batches = BatchesNum
               self.__window = None
00050
00051
               self.__batch_counter = 0
00052
```

3.4.3 Member Function Documentation

3.4.3.1 BatchGuiShow()

```
def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (
               self )
The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and then calls the
Aras:
self: Represent the instance of the class
The __type of the batchgui class
Willem van der Schans, Trelent AI
Definition at line 53 of file BatchProgressGUI.py.
         def BatchGuiShow(self):
00054
         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:
         Willem van der Schans, Trelent AI
00064
00065
00066
             self.CreateProgressLayout()
00067
```

Here is the call graph for this function:

00068

self.createGui(self.__type)

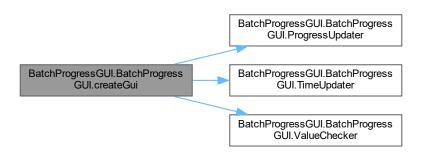


3.4.3.2 createGui()

```
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 thread
Aras:
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.pv.
00104
          def createGui(self, Sourcetype):
00105
00106
00107
          The createGui function is the main function that creates the GUI.
00108
          It takes \frac{in}{n} a type parameter which determines what kind of batch processor to use.
00109
          The createGui function then sets up all the variables {\sf and} objects needed for
00110
          the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
00111
          \verb|processorObject| (Batch \verb|ProcessorConstructionMonitor| or Batch \verb|ProcessorUtah Realestate|), \\
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:
          Willem van der Schans, Trelent AI
00122
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)
              self.__window['--time_est--'].update("Est time needed 00:00:00")
00130
00131
00132
              valueObj = DataTransfer()
00133
              valueObj.setValue(0)
00134
00135
              if Sourcetype == "construction_monitor":
00136
00137
                  processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00138
                                                                      {\tt NumBatches=self.\_\_batches,}
00139
                                                                      ParameterDict=self. parameterDict,
00140
                                                                      {\tt HeaderDict=self.\_\_headerDict,}
00141
                                                                      ColumnSelection=self.__columnSelection,
00142
                                                                      valueObject=valueObj)
              elif Sourcetype == "utah_real_estate":
00143
                  processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00144
```

```
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 \ Thr
              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,
                                                                          args=(valueObj,),
00160
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(valueObi)
00167
00168
                                           if valueObj.getValue() == -999:
00169
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
00177
00178
                                          if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179
00180
                                          time.sleep(0.1)
00181
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:



3.4.3.3 CreateProgressLayout()

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
00076
                  self (object): The object that is calling this function.
00077
00078
         Args:
00079
             self: Access the class variables and methods
08000
00081
          Returns:
00082
              A list of lists
00083
00084
         Doc Author:
          Willem van der Schans, Trelent AI
00085
00086
00087
              sq.theme('Default1')
00088
00089
              __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center",
      key="--progress\_text--"),
00090
                         sq.Push()]
00091
              __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
00092
                         sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]
00093
00094
00095
              __Line3 = [
```

Here is the caller graph for this function:



3.4.3.4 ProgressUpdater()

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

Args:

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

Returns:

The value of the batch counter

Doc Author:

Willem van der Schans, Trelent AI

Definition at line 188 of file BatchProgressGUI.py.

```
def ProgressUpdater(self, valueObj):
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
              self: Make the progressupdater function an instance method
00197
00198
              valueObj: Get the current value of the batch counter
00199
00200
          Returns:
              The value of the batch counter
00201
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)
self.__window.write_event_value('update--progress_text--', __update_text)
00212
                 else:
00214
00215
```

Here is the caller graph for this function:



3.4.3.5 TimeUpdater()

```
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 be

```
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.
         It then calculates an estimation of how long it will take to finish all batches based on how many
00222
     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:
          Willem van der Schans, Trelent AI
00232
00233
00234
              while True:
00235
                  if self.__batch_counter < self.__batches:</pre>
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
                      trv:
00244
                          self.__window.write_event_value('update--timer--', __timer_string)
00245
                      except AttributeError as e:
00246
                         print(
                              f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00247
     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
00249
00250
                       _passed_time = __passed_time.total_seconds()
00251
00252
                          __time_est = datetime.timedelta(
00253
                              seconds=(__passed_time * (self.__batches / self.__batch_counter) -
00254
      __passed_time)).seconds
00255
                      except:
00256
                          __time_est = datetime.timedelta(
                              \verb|seconds=(\__passed_time * self.\__batches - \__passed_time)|).seconds|
00257
00258
                        _time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))
00259
00260
                        _end_string = f"Est time needed {__time_est}"
00261
                      self.__window.write_event_value('update--time_est--', __end_string)
00262
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:



3.4.3.6 ValueChecker()

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

Here is the caller graph for this function:



3.4.4 Member Data Documentation

3.4.4.1 __batch_counter

```
BatchProgressGUI.__batch_counter [private]
```

Definition at line 51 of file BatchProgressGUI.py.

3.4.4.2 __batches

```
BatchProgressGUI.BatchProgressGUI.__batches [private]
```

Definition at line 49 of file BatchProgressGUI.py.

3.4.4.3 __columnSelection

```
BatchProgressGUI.BatchProgressGUI.__columnSelection [private]
```

Definition at line 44 of file BatchProgressGUI.py.

3.4.4.4 headerDict

```
BatchProgressGUI.BatchProgressGUI._headerDict [private]
```

Definition at line 43 of file BatchProgressGUI.py.

3.4.4.5 __layout

```
BatchProgressGUI.BatchProgressGUI.__layout [private]
```

Definition at line 48 of file BatchProgressGUI.py.

3.4.4.6 parameterDict

```
BatchProgressGUI.BatchProgressGUI.__parameterDict [private]
```

Definition at line 41 of file BatchProgressGUI.py.

3.4.4.7 __restDomain

```
BatchProgressGUI.BatchProgressGUI.__restDomain [private]
```

Definition at line 42 of file BatchProgressGUI.py.

3.4.4.8 __type

BatchProgressGUI.BatchProgressGUI.__type [private]

Definition at line 45 of file BatchProgressGUI.py.

3.4.4.9 __window

```
BatchProgressGUI. BatchProgressGUI. window [private]
```

Definition at line 50 of file BatchProgressGUI.py.

3.4.4.10 dataframe

BatchProgressGUI.BatchProgressGUI.dataframe

Definition at line 46 of file BatchProgressGUI.py.

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

· BatchProgressGUI.py

3.5 Core.CFBP Class Reference

Public Member Functions

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

Public Attributes

- · state arg
- year_arg
- uiString
- link

Private Member Functions

- def showUi (self)
- def <u>__dataGetter</u> (self)

3.5.1 Detailed Description

Definition at line 15 of file CFBP/Core.py.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 __init__()

```
def Core.CFBP.__init__ (
                                  self,
                                  state_arg = None,
                                  year_arg = None )
            _init__ function is called when the class is instantiated.
Its 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 17 of file CFBP/Core.py.
                     def __init__(self, state_arg=None, year_arg=None):
00017
00018
00019
                                _init__ function is called when the class is instantiated.
00020
                      Its job is to initialize the object with some default values, and do any other setup that might be
00021
                     The __init__ function can take arguments, but it doesn't have to.
00022
00023
00024
                             self: Represent the instance of the class
                              state_arg: Set the state_arg attribute of the class
00026
                             year_arg: Set the year of data to be retrieved
00027
00028
00029
                             A popupwrapped object
00030
00031
                    Doc Author:
                    Willem van der Schans, Trelent AI
00032
00033
00034
                             self.state_arg = state_arg
00035
                              self.year_arg = year_arg
                              self.uiString = None
00036
00037
                             self.link = None
00038
00039
                              eventReturn = confirmDialog()
                              if eventReturn == "Continue":
00040
                                      startTime = datetime.datetime.now().replace(microsecond=0)
00041
00042
                                       self.__showUi()
00043
                                      print(
                                                f" \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ API \ Link = ( A
00044
             {self.link}")
00045
                                      F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
```

```
00046
00047
                                                                                                                                        f'' \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:\%M:\%S.\%f')[:-3]} \ | \ \texttt{Data retrieved with } 
                                     in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) startTime).total_seconds()))}")
 00048
 00049
                                                                                                                self.uiString = (
 00050
                                                                                                                                          \texttt{f"ffiec.fpb.gov} \  \, \texttt{(Mortgage API)} \  \, \texttt{request Completed } \\   \, \texttt{n \{self.year\_arg\}} \  \, \texttt{data retrieved } \\  \, \texttt{n} \  \, \texttt{n \{self.year\_arg\}} \\  \, \texttt{data retrieved } \\  \, \texttt{n \{self.year\_arg\}} \\  \, \texttt{n \{self.ye
                                     Data Saved at {F.getPath()}")
 00051
 00052
                                                                                                               PopupWrapped(text=self.uiString, windowType="noticeLarge")
 00053
                                                                                       else:
 00054
                                                                                                              print(
00055
                                                                                                                                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
                                     Request")
 00056
00057
```

Here is the call graph for this function:



3.5.3 Member Function Documentation

3.5.3.1 __dataGetter()

```
def Core.CFBP.__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.
self: Represent the instance of the class
Returns:
A response object
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 86 of file CFBP/Core.py.
00086
         def __dataGetter(self):
00087
00088
         The __dataGetter function is a private function that gets the data from the CFPB API.
00089
         It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.
00090
00091
00092
             self: Represent the instance of the class
00093
```

Returns:

00094

```
00095
              A response object
00096
00097
          Doc Author:
          Willem van der Schans, Trelent AI
00098
00099
00100
              arg_dict_bu = locals()
00101
00102
              link = settings.settingCFBPLink
00103
00104
              if self.state_arg is None:
00105
                  self.state_arg = "UT"
00106
              else:
00107
00108
00109
              if self.year_arg is None:
                  self.year_arg = str(datetime.date.today().year - 1)
00110
00111
              else:
00112
00113
00114
              passFlag = False
00115
00116
              while not passFlag:
00117
                  self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"
00118
00119
00120
                  response = requests.get(self.link)
00121
00122
                  if response.status_code == 400:
00123
                      self.year_arg = int(self.year_arg) - 1
00124
00125
                  else:
00126
                      passFlag = True
00127
              RESTError (response)
00128
00129
              raise SystemExit(0)
```

Here is the caller graph for this function:



3.5.3.2 __showUi()

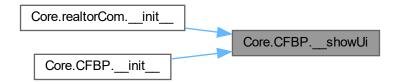
Definition at line 58 of file CFBP/Core.py.

```
def __showUi(self):
00059
00060
00061
          The \_showUi function is a function that creates a progress bar window.
00062
          The __showUi function takes class variables and returns a windowobj.
00063
00064
00065
          Args:
00066
              self: Represent the instance of the class
00067
00068
          Returns:
              The uiobj variable
00069
00070
00071
          Doc Author:
00072
              Willem van der Schans, Trelent AI
00073
00074
              uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)
00075
00076
              threadGui = threading.Thread(target=self.__dataGetter,
00077
                                            daemon=False)
00078
              threadGui.start()
00079
00080
              while threadGui.is_alive():
00081
                  uiObj.textUpdate()
00082
                  uiObj.windowPush()
00083
              else:
00084
                  uiObj.stopWindow()
00085
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.5.4 Member Data Documentation

3.5.4.1 link

Core.CFBP.link

Definition at line 37 of file CFBP/Core.py.

3.5.4.2 state_arg

Core.CFBP.state_arg

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

3.5.4.3 uiString

Core.CFBP.uiString

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

3.5.4.4 year_arg

Core.CFBP.year_arg

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

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

· CFBP/Core.py

3.6 Core.ConstructionMonitorInit Class Reference

Public Member Functions

def __init__ (self)

Public Attributes

- size
- SourceInclude
- 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 ()
```

3.6.1 Detailed Description

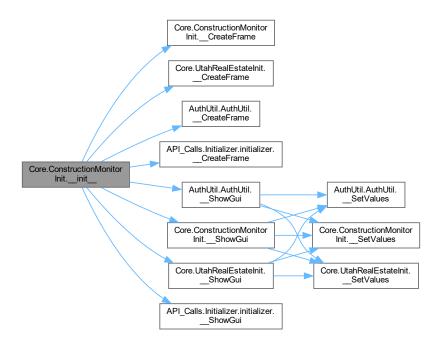
Definition at line 25 of file ConstructionMonitor/Core.py.

3.6.2 Constructor & Destructor Documentation

Definition at line 27 of file ConstructionMonitor/Core.py.

```
def __init__(self):
00028
00029
00030
                      _ function is called when the class is instantiated.
00031
          It sets up the variables that will be used by other functions in this class.
00032
00033
00034
          Args:
00035
              self: Represent the instance of the class
00036
00037
          Returns:
00038
              None
00039
00040
          Doc Author:
00041
              Willem van der Schans, Trelent AI
00042
00043
              self.size = None
00044
              self.SourceInclude = None
00045
              self.dateStart = None
00046
              self.dateEnd = None
00047
              self.rest_domain = None
00048
              self.auth_key = None
              self.ui_flag = None
00049
              self.append_file = None
00050
00051
00052
              passFlag = False
00053
00054
              while not passFlag:
                   if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00055
00056
                           "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
                       Path (os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00057
                           "Security").joinpath("auth.json")):
00058
00059
                           f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00060
                               "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00061
                           key = f.readline()
00062
00063
                           f.close()
                           f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
    "Security").joinpath("auth.json"), "rb")
00064
00065
00066
                           authDict = json.load(f)
00067
                           fernet = Fernet(key)
                           self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
00068
00069
                           passFlag = True
                       except Exception as e:
00070
00071
                           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")
00072
                           AuthUtil()
00073
00074
                       AuthUtil()
00075
00076
              self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00077
```

Here is the call graph for this function:



3.6.3 Member Function Documentation

3.6.3.1 CreateFrame()

```
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:

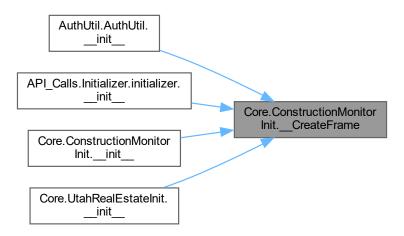
Returns:
The layout for the gui

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

Definition at line 117 of file ConstructionMonitor/Core.py.

```
00121
                            The function returns a list of lists that contains all the elements to be displayed \frac{in}{in} the GUI
00122
                           This is done by creating each line as a list and then appending it to another list which will
           contain all lines.
00123
00124
                    Args:
00125
00126
00127
                            The layout for the gui
00128
00129
                    Doc Author:
                    Willem van der Schans, Trelent AI
00130
00131
00132
                            sg.theme('Default1')
00133
00134
                           line00 = [sg.HSeparator()]
00135
00136
                            line0 = [sg.Image(ImageLoader("logo.png")),
00137
                                              sq.Push(),
00138
                                              sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"),
            justification="center"),
00139
                                             sq.Push(),
00140
                                             sg.Push()]
00141
                            line1 = [sg.HSeparator()]
00142
00143
                            line3 = [sq.Text("Start Date : ", size=(15, None), justification="Right"),
00144
                                              sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
00145
           key="-Cal-",
00146
                                                                size=(20, 1)),
00147
                                             sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]
00148
                            line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00149
                                              \verb|sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-", k
00150
00151
                                                                size=(20, 1)),
                                              sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00152
           target="-EndCal-")]
00153
00154
                            line5 = [sg.HSeparator()]
00155
00156
                            line6 = [sq.Push(),
                                              sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00157
00158
                                              sg.Push()]
00159
00160
                            line7 = [sg.HSeparator()]
00161
                            00162
00163
00164
                                                                 size=(20, 1)),
00165
                                              sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                                                                          target="-AppendingFile-")]
00166
00167
00168
                            line9 = [sg.HSeparator()]
00169
00170
                            line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00171
00172
                            layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]
00173
00174
                            return layout
00175
```

Here is the caller graph for this function:

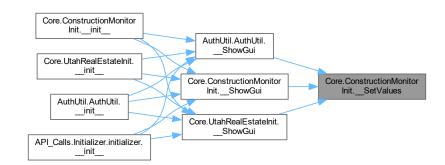


3.6.3.2 __SetValues()

```
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 i
the dictionary. The keys for this dictionary are defined by the user when they create their own instance of this
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 176 of file ConstructionMonitor/Core.py.
         def __SetValues(self, values):
00176
00177
00178
00179
         The \_SetValues function is used to set the values of the variables that are used in the \_GetData
     function.
00180
               _SetValues function takes a dictionary as an argument, <mark>and</mark> then sets each variable based on what
         The _
     is passed into
00181
         the dictionary. The keys for this dictionary are defined by the user when they create their own
     instance of this class.
00182
```

```
00183
          Args:
00184
             self: Represent the instance of the class
00185
              values: Pass in the values from the ui
00186
00187
00188
              A dictionary of values
00189
00190
          Doc Author:
          Willem van der Schans, Trelent AI
00191
00192
00193
              self.size = 1000
00194
00195
              if values["-Cal-"] != "":
00196
                 self.dateStart = values["-Cal-"]
00197
              else:
00198
                  self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00199
00200
              if values["-EndCal-"] != "":
00201
                  self.dateEnd = values["-EndCal-"]
00202
00203
                  self.dateEnd = date.today().strftime("%Y-%m-%d")
00204
00205
              self.rest_domain = settings.settingCMRestDomain
00206
00207
              self.SourceInclude = None
00208
              if values["-append_file-"] != "":
00209
                  self.append_file = str(values["-append_file-"])
00210
              else:
00211
00212
                  self.append_file = None
00213
00214
              self.ui_flag = True
00215
00216
```

Here is the caller graph for this function:



3.6.3.3 ShowGui()

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

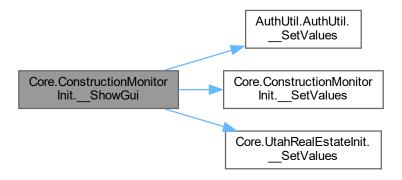
Returns:
A dictionary of values

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

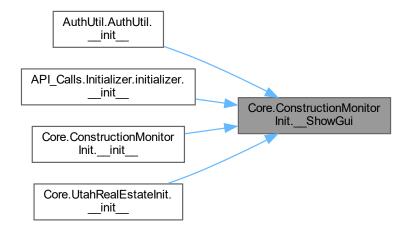
Definition at line 78 of file ConstructionMonitor/Core.py.

```
00078
          def __ShowGui(self, layout, text):
00079
00080
          The __ShowGui function is the main function that creates and displays the GUI.
00081
00082
          It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
00083
          The text parameter specifies what title should appear at the top of the window.
00084
00085
00086
              self: Refer to the current instance of a class
00087
              layout: Determine what the gui will look like
00088
              text: Set the title of the window
00089
00090
00091
              A dictionary of values
00092
00093
          Doc Author:
          Willem van der Schans, Trelent AI
00094
00095
00096
              window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00097
                                 finalize=True,
00098
                                 icon=ImageLoader("taskbar_icon.ico"))
00099
00100
              while True:
00101
                  event, values = window.read()
00102
00103
                  if event == "Submit":
00104
                      try:
00105
                          self.__SetValues(values)
00106
00107
                      except Exception as e:
00108
                         print(e)
00109
                          RESTError (993)
00110
                          raise SystemExit(933)
00111
                  elif event == sg.WIN_CLOSED or event == "Quit":
00112
00113
00114
              window.close()
00115
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.4 Member Data Documentation

3.6.4.1 append_file

Core.ConstructionMonitorInit.append_file

Definition at line 50 of file ConstructionMonitor/Core.py.

3.6.4.2 auth_key

Core.ConstructionMonitorInit.auth_key

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

3.6.4.3 dateEnd

Core.ConstructionMonitorInit.dateEnd

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

3.6.4.4 dateStart

Core.ConstructionMonitorInit.dateStart

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

3.6.4.5 rest_domain

Core.ConstructionMonitorInit.rest_domain

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

3.6.4.6 size

Core.ConstructionMonitorInit.size

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

3.6.4.7 SourceInclude

Core.ConstructionMonitorInit.SourceInclude

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

3.6.4.8 ui_flag

Core.ConstructionMonitorInit.ui_flag

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

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

· ConstructionMonitor/Core.py

3.7 Core.ConstructionMonitorMain Class Reference

Public Member Functions

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

Public Attributes

dataframe

Private Member Functions

- def __ParameterCreator (self)
- def getCount (self)
- def <u>getCountUI</u> (self)

Private Attributes

- siteClass
- __restDomain
- __headerDict
- __columnSelection
- __appendFile
- __parameterDict
- __search_id
- · __record_val
- __batches
- __ui_flag

3.7.1 Detailed Description

Definition at line 217 of file ConstructionMonitor/Core.py.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 __init__() 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 219 of file ConstructionMonitor/Core.py. 00219 def __init__(self, siteClass): 00220 00221 00222 The __init__ function is the first function that runs when an object of this class is created. 00223 It sets up all the variables and functions needed for this class to run properly. 00224 00225 00226 Args: 00227 self: Represent the instance of the class 00228 siteClass: Identify the site that is being used 00229 00230 Returns: 00231 Nothing 00232 00233 Doc Author: 00234 Willem van der Schans, Trelent AI 00235 00236 self.__siteClass = siteClass 00237 self.__restDomain = None 00238 self.__headerDict = None 00239 self.__columnSelection = None 00240 self.__appendFile = None 00241 self.__parameterDict = {} 00242 00243 self.__search_id = None

if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):

f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in

This allows for user cancellation of the program using the quit button

00248 00249

00250 00251 00252

00253

00254

00255

00256

00257 00258

00259

self.__record_val = 0
self.__batches = 0

self.__ui_flag = None

self.dataframe = None

self.mainFunc()

print(

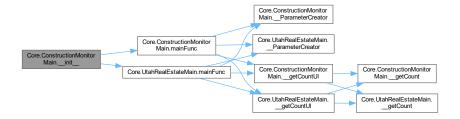
ConstructionMonitorMain class")

except SystemError as e:

except AttributeError as e:

```
00260
                if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
00261
                   RESTError (1101)
00262
                    print(f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ Error \ \{e\}") \} 
00263
                   pass
00264
                elif e is not None:
00265
                   print(
00266
                       keys in AuthUtil")
00267
                   RESTError (401)
00268
                   print(e)
00269
00270
                else:
00271
00272
            except Exception as e:
00273
                print(e)
00274
                RESTError (1001)
00275
                raise SystemExit(1001)
00276
```

Here is the call graph for this function:



3.7.3 Member Function Documentation

3.7.3.1 __getCount()

Definition at line 372 of file ConstructionMonitor/Core.py.

```
00372
          def __getCount(self):
00373
          The __getCount function is used to get the total number of records that are returned from a query.
00374
00375
          This function is called by the __init__ function and sets the self.__record_val variable with this
00376
00377
00378
              self: Represent the instance of the class
00379
00380
00381
              The total number of records in the database
00382
00383
         Doc Author:
          Willem van der Schans, Trelent AI
00384
00385
00386
               count resp = None
00387
00388
              try:
00389
00390
                   __temp_param_dict = copy.copy(self.__parameterDict)
00391
00392
                   __count_resp = requests.post(url=self.__restDomain,
00393
                                                 headers=self.__headerDict,
00394
                                                 json=__temp_param_dict)
00395
00396
              except requests.exceptions.Timeout as e:
00397
                  print(e)
00398
                  RESTError (790)
                  raise SystemExit(790)
00399
00400
              except requests.exceptions.TooManyRedirects as e:
00401
                  print(e)
00402
                  RESTError (791)
                  raise SystemExit(791)
00403
00404
              {\tt except} \ {\tt requests.exceptions.MissingSchema} \ {\tt as} \ {\tt e:}
00405
                  print(e)
00406
                   RESTError (1101)
00407
              except requests.exceptions.RequestException as e:
00408
                   print(e)
00409
                   RESTError (405)
00410
                  raise SystemExit (405)
00411
00412
              __count_resp = __count_resp.json()
00413
00414
              self.__record_val = __count_resp["hits"]["total"]["value"]
00415
00416
              del __count_resp, __temp_param_dict
00417
```

Here is the caller graph for this function:



3.7.3.2 __getCountUI()

```
\begin{tabular}{ll} \tt def Core.ConstructionMonitorMain.\_getCountUI ( \\ & self ) & [private] \end{tabular}
```

```
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

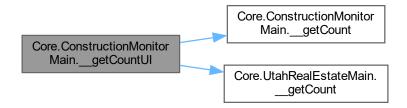
Returns:
The count of the number of records in the database

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

Definition at line 418 of file ConstructionMonitor/Core.py.

```
def __getCountUI(self):
00418
00419
00420
00421
          The
                _getCountUI function \overline{\mathsf{is}} a wrapper \overline{\mathsf{for}} the __getCount function.
00422
          It allows the user to run __getCount in a separate thread, so that they can continue working while it
00423
          The function will display a progress bar and update with text as it progresses through its tasks.
00424
00425
              self: Access the class variables and methods
00426
00427
00428
          Returns:
00429
              The count of the number of records in the database
00430
00431
          Doc Author:
00432
              Willem van der Schans, Trelent AI
00433
00434
              if self.__ui_flag:
00435
                   uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00436
00437
                   threadGui = threading.Thread(target=self.__getCount,
00438
                                                  daemon=False)
00439
                  threadGui.start()
00440
00441
                   while threadGui.is_alive():
00442
                       uiObj.textUpdate()
00443
                       uiObj.windowPush()
00444
00445
                       uiObj.stopWindow()
00446
              else:
00448
                  self.__getCount()
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.7.3.3 ParameterCreator()

Definition at line 333 of file ConstructionMonitor/Core.py.

```
def ___ParameterCreator(self):
00333
00334
00335
           The __ParameterCreator function is used to create the parameter dictionary that will be passed into
      the
00336
                __Request function. The function takes in a siteClass object and extracts all of its attributes,
      except for
00337
               those that start with '\_' or are callable. It then creates a dictionary from these attributes and
      stores it as
00338
              self.__parameterDict.
00339
00340
          Args:
00341
              self: Make the function a method of the class
00342
00343
          Returns:
00344
              A dictionary of parameters and a list of non parameter variables
00345
00346
          Doc Author:
00347
               Willem van der Schans, Trelent AI
00348
               __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if not key.startswith('__') and not callable(key)}
00349
00350
00351
00352
               self. restDomain = Source dict["rest domain"]
                 _Source_dict.pop("rest_domain")
00353
               self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00354
               ___Source_dict.pop("auth_key")
self.__columnSelection = __Source_dict["SourceInclude"]
00355
00356
00357
               __Source_dict.pop("SourceInclude")
```

```
00358
              self.__ui_flag = __Source_dict["ui_flag"]
00359
              __Source_dict.pop("ui_flag")
00360
              self.__appendFile = __Source_dict["append_file"]
00361
              __Source_dict.pop("append_file")
00362
00363
              temp_dict = copy.copy(__Source_dict)
00364
              for key, value in temp_dict.items():
00365
                  if value is None:
00366
                      __Source_dict.pop(key)
00367
00368
00369
00370
              self.__parameterDict = copy.copy(__Source_dict)
00371
```

Here is the caller graph for this function:



3.7.3.4 mainFunc()

```
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 __ParameterCrea 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 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 277 of file ConstructionMonitor/Core.py.

```
00277 def mainFunc(self):
00278 """

00279 The mainFunc function is the main function of this module. It will be called by the GUI or CLI to execute

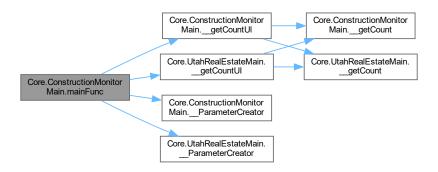
00280 the code in this module. The mainFunc function will first create a parameter dictionary using the __ParameterCreator

00281 method, then it will get a count of all records that match its parameters using the __getCountUI method, and then

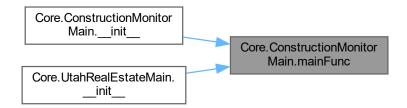
00282 it will calculate how many batches are needed to retrieve all records with those parameters using BatchCalculator.
```

```
00283
                            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
 00284
00285
 00286
                           Args:
 00287
                                       self: Refer to the current object
 00288
 00289
                            Returns:
 00290
                                       The dataframe
 00291
 00292
                           Doc Author:
                            Willem van der Schans, Trelent AI
 00293
 00294
 00295
                                        self.__ParameterCreator()
 00296
 00297
                                       print(
00298
                                                  f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict =
                 {self.__parameterDict}")
 00299
                                       print(
00300
                                                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
                 {self.__restDomain}")
 00301
00302
                                        self.__getCountUI()
 00303
00304
                                        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)
00305
00306
                                       print(
                                                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
00307
                 {self.__batches} | Rows {self.__record_val}")
00308
00309
                                        if self.__batches != 0:
00310
                                                   startTime = datetime.datetime.now().replace(microsecond=0)
00311
                                                   eventReturn = BatchInputGui(self.__batches, self.__record_val)
                                                   if eventReturn == "Continue":
00312
00313
                                                              print(
                                                                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
00314
                 {self.__batches} batches sent to server")
00315
                                                              BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00316
                                                                                                                                                               ParameterDict=self.__parameterDict,
00317
                                                                                                                                                               HeaderDict=self.__headerDict,
00318
                                                                                                                                                               ColumnSelection=self.__columnSelection,
00319
                                                                                                                                                               BatchesNum=self.__batches,
00320
                                                                                                                                                               Type="construction_monitor")
00321
                                                              BatchGuiObject.BatchGuiShow()
00322
                                                               self.dataframe = BatchGuiObject.dataframe
00323
                                                              print(
00324
                                                                           \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('\%m-\%d-\%Y \%H:\%M:\%S.\%f')[:-3]} \; | \; \; \texttt{Dataframe of the property of the prope
                 retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in
                  startTime).total_seconds()))}")
00325
                                                             FileSaver("cm", self.dataframe, self.__appendFile)
00326
                                                   else:
 00327
                                                             print(
00328
                                                                          f" \{ date time.date time.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ Request for the strength of 
                 {self.__batches} batches canceled by user")
 00329
                                        else:
00330
                                                  RESTError (994)
00331
                                                   raise SystemExit(994)
00332
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.7.4 Member Data Documentation

3.7.4.1 __appendFile

Core.ConstructionMonitorMain.__appendFile [private]

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

3.7.4.2 __batches

Core.ConstructionMonitorMain.__batches [private]

Definition at line 245 of file ConstructionMonitor/Core.py.

3.7.4.3 __columnSelection

Core.ConstructionMonitorMain.__columnSelection [private]

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

3.7.4.4 __headerDict

Core.ConstructionMonitorMain.__headerDict [private]

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

3.7.4.5 __parameterDict

 ${\tt Core.ConstructionMonitorMain.} \underline{\hspace{0.5cm}} {\tt parameterDict} \hspace{0.2cm} [{\tt private}]$

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

3.7.4.6 record val

Core.ConstructionMonitorMain.__record_val [private]

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

3.7.4.7 __restDomain

Core.ConstructionMonitorMain.__restDomain [private]

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

3.7.4.8 __search_id

```
Core.ConstructionMonitorMain.__search_id [private]
```

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

3.7.4.9 __siteClass

```
Core.ConstructionMonitorMain.__siteClass [private]
```

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

3.7.4.10 __ui_flag

Core.ConstructionMonitorMain.__ui_flag [private]

Definition at line 247 of file ConstructionMonitor/Core.py.

3.7.4.11 dataframe

Core.ConstructionMonitorMain.dataframe

Definition at line 249 of file ConstructionMonitor/Core.py.

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

ConstructionMonitor/Core.py

3.8 DataTransfer.DataTransfer Class Reference

Public Member Functions

- def __init__ (self)
- def setValue (self, value)
- def getValue (self)
- def while Value (self)

Private Attributes

value

3.8.1 Detailed Description

Definition at line 4 of file DataTransfer.py.

3.8.2 Constructor & Destructor Documentation

```
3.8.2.1 __init__()
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.
self: Represent the instance of the class
Returns:
Nothing
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 6 of file DataTransfer.py.
         def __init__(self):
00007
         The \_init\_ function is called when the class is instantiated. It sets the initial value of self.\_value to 0.
00008
00009
00010
00011
              self: Represent the instance of the class
00012
00013
00014
         Returns:
00015
             Nothing
00016
00017
         Doc Author:
          Willem van der Schans, Trelent AI
00018
```

3.8.3 Member Function Documentation

 $self._value = 0$

00019

00021

3.8.3.1 getValue()

```
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.
Aras:
self: Represent the instance of the class
The value of the instance variable
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 39 of file DataTransfer.py.
         def getValue(self):
00039
00040
         The getValue function returns the value of the private variable \_value.
00041
00042
         This is a getter function that allows access to this private variable.
00043
00044
00045
             self: Represent the instance of the class
00046
00047
00048
             The value of the instance variable
00049
00050
         Doc Author:
         Willem van der Schans, Trelent AI
00051
00052
00053
             return self.__value
```

Here is the caller graph for this function:



3.8.3.2 setValue()

00054

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

```
Args:
self: Represent the instance of the class
value: Set the value of the instance variable __value
Returns:
The value that was passed to it
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 22 of file DataTransfer.py.

```
def setValue(self, value):
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
00032
             The value that was passed to it
00033
00034
          Willem van der Schans, Trelent AI
00035
00036
00037
              self.___value = value
00038
```

3.8.3.3 whileValue()

```
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:

def DataTransfer.DataTransfer.whileValue (

self)

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
          The whileValue function is a function that will run the getValue function until it is told to stop.
00057
          This allows for the program to constantly be checking for new values from the sensor.
00058
00059
00060
              self: Refer to the current instance of the class
00061
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:



3.8.4 Member Data Documentation

3.8.4.1 __value

```
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

3.9 FileSaver.FileSaver Class Reference

Public Member Functions

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

Public Attributes

- docPath
- data
- dataAppending
- appendFlag
- fileName
- uiFlag
- primaryKey
- outputFrame

3.9.1 Detailed Description

Definition at line 13 of file FileSaver.py.

3.9.2 Constructor & Destructor Documentation

```
3.9.2.1 __init__()
def FileSaver.FileSaver.__init__ (
               self,
               method.
               outputDF,
               AppendingPath = None)
     __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 clas
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 tha
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.
         def __init__(self, method, outputDF, AppendingPath=None):
00016
00017
         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
00018
     in this class.
00019
         The __init__ function takes two arguments: self and method. The first argument, self, refers to an
     instance of a
         class (\frac{1}{1} this case it's an instance of DataFrameSaver). The second argument, method refers to a
00020
     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
             AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to
00027
00028
00029
        Returns:
00030
             Nothing
00031
```

Doc Author:

Willem van der Schans, Trelent AI

self.data = outputDF

datetime.datetime.today().strftime('%m%d%Y'))

00032

00033

00034

00035

00037

```
00038
                             self.dataAppending = None
                             self.appendFlag = True
self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00039
00040
00041
                             self.uiFlag = True
00042
00043
                             if method.lower() == "ure":
00044
                                     self.primaryKey = "ListingKeyNumeric"
00045
                             elif method.lower() == "cm":
00046
                                     self.primaryKey = "id"
                             elif "realtor" in method.lower():
00047
                                     self.primaryKey = None
00048
00049
                                     self.uiFlag = False
                             elif method.lower() == "cfbp":
00050
                                     self.primaryKey = None
00051
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,
                                              # since we are comparing this will be an issue since we need to do type comparisons,
00064
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)
                             else:
00079
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="savedLarge")
00087
00088
                                              # Logging
00089
                                             print(
00090
                                                       \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ | \ \{ \texttt{method} \} \ \ \texttt{API} \} \} = \{ \texttt{method} \} = \{ \texttt{method} \} \} = \{ \texttt{method} \} = \{ \texttt{method} \} \} = \{ \texttt{method} \} = \{ \texttt{method} \} \} = \{ \texttt{method} \} = \{ \texttt{method} \} \} = \{ \texttt{method} \} = \{ \texttt{method} \} \} = \{ \texttt{method} \}
            request Completed | File Appended and Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0") print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Appending
00091
             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:
                                             PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}",
00093
            windowType="savedLarge")
00094
00095
                                              # Logging
00096
                                             print(
                                                      00097
            request Completed | File Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
00098
                            else:
00099
00100
```

3.9.3 Member Function Documentation

3.9.3.1 getPath()

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

return str(self.docPath.joinpath(self.fileName))

3.9.4 Member Data Documentation

3.9.4.1 appendFlag

00115

FileSaver.FileSaver.appendFlag

Definition at line 39 of file FileSaver.py.

3.9.4.2 data

FileSaver.FileSaver.data

Definition at line 37 of file FileSaver.py.

3.9.4.3 dataAppending

```
FileSaver.FileSaver.dataAppending
```

Definition at line 38 of file FileSaver.py.

3.9.4.4 docPath

```
FileSaver.FileSaver.docPath
```

Definition at line 35 of file FileSaver.py.

3.9.4.5 fileName

FileSaver.FileSaver.fileName

Definition at line 40 of file FileSaver.py.

3.9.4.6 outputFrame

FileSaver.FileSaver.outputFrame

Definition at line 70 of file FileSaver.py.

3.9.4.7 primaryKey

FileSaver.FileSaver.primaryKey

Definition at line 44 of file FileSaver.py.

3.9.4.8 uiFlag

FileSaver.FileSaver.uiFlag

Definition at line 41 of file FileSaver.py.

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

FileSaver.py

3.10 API Calls.Initializer.initializer Class Reference

Public Member Functions

def __init__ (self)

Public Attributes

classObj

Private Member Functions

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

3.10.1 Detailed Description

Definition at line 22 of file Initializer.py.

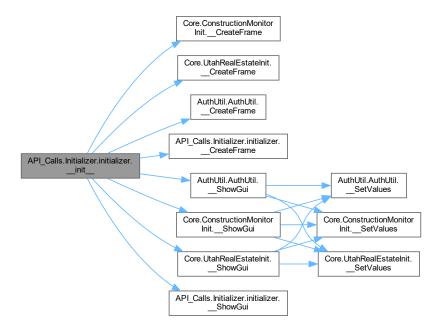
3.10.2 Constructor & Destructor Documentation

```
3.10.2.1 __init__()
```

Definition at line 24 of file Initializer.py.

```
def __init__(self):
00025
00026
00027
        The \_init\_ function is called when the class is instantiated.
00028
        It sets up the logging, calls the __ShowGui function to create and display
00029
        the GUI, and then calls __CreateFrame to create a frame for displaying widgets.
00030
00031
00032
        Args:
00033
            self: Represent the instance of the class
00034
00035
        Returns:
00036
            Nothing
00037
00038
        Doc Author:
        Willem van der Schans, Trelent AI
00039
00040
00041
            self.classObj = None
00042
00043
            logger()
00044
00045
            print ("\n\n-----\n\n")
00046
00047
            self.__ShowGui(self.__CreateFrame(), "Data Tool")
00048
            print("\n\n----\n\n")
00049
00050
```

Here is the call graph for this function:

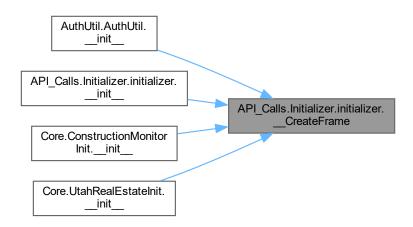


3.10.3 Member Function Documentation

3.10.3.1 __CreateFrame()

```
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 sq.Window() as its layout parameter.
Aras:
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 136 of file Initializer.py.
00136
         def __CreateFrame(self):
00137
00138
00139
         The __CreateFrame function is a helper function that creates the layout for the main window.
00140
         It returns a list of lists, which is then passed to sg.Window() as its layout parameter.
00141
00142
00143
             self: Represent the instance of the class
00144
00145
00146
             A list of lists, which is then passed to the sg
00147
00148
         Willem van der Schans, Trelent AI
         Doc Author:
00149
00150
00151
             sg.theme('Default1')
00152
00153
             line0 = [sg.HSeparator()]
00154
00155
             line1 = [sg.Image(ImageLoader("logo.png")),
00156
                     sg.Push(),
                      sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
00157
00158
                      sg.Push(),
00159
                      sq.Push()]
00160
00161
             line3 = [sq.HSeparator()]
00162
00163
             line4 = [sq.Push(),
00164
                     sq.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
00165
                      sq.Push()]
00166
00167
             line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                       sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00168
00169
             00170
00171
00172
                       sq.Push()]]
00173
             line8 = [sg.HSeparator()]
00174
00175
             line9 = [sg.Push(),
00176
                      sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
00177
00178
                      sq.Push()]
00179
             line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00180
00181
                        sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00182
00183
             line11 = [sq.HSeparator()]
00184
             layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]
00185
00186
00187
             return layout
```

Here is the caller graph for this function:



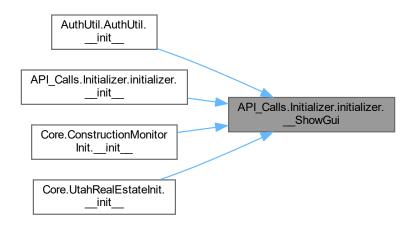
3.10.3.2 __ShowGui()

```
def API_Calls.Initializer.initializer.__ShowGui (
              self,
              layout,
              text ) [private]
The \_ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three elements:
1) The type of element to be displayed (e.g., "Text", "InputText", etc.)
2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
3) An optional key name for the element (used in event handling). If no key name is provided then one will be gen
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 51 of file Initializer.py.
         def __ShowGui(self, layout, text):
00051
00052
00053
00054
             _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
00055
     elements:
```

```
00056
             1) The type of element to be displayed (e.g., " Text", " InputText", etc.)
             2) A dictionary containing any additional parameters for that element (e.g., size, default value,
00057
00058
             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
00059
00060
00061
             self: Represent the instance of the class
             layout: Pass the layout of the window to be created
00062
00063
             text: Set the title of the window
00064
00065
00066
             A window object
00067
00068
        Doc Author:
00069
             Willem van der Schans, Trelent AI
00070
00071
             # Todo Gitlab Update
00072
             versionChecker()
00073
             window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00074
00075
                                finalize=True,
00076
                                icon=ImageLoader("taskbar icon.ico"))
00077
00078
             while True:
00079
                 event, values = window.read()
00080
                 if event == "Construction Monitor":
00081
00082
                         f"\n\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ |
00083
                 --Initiating Construction Monitor API Call------")
00084
                     ConstructionMonitorMain(ConstructionMonitorInit())
00085
                     print(
                        f"{\tt datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ |
00086
               ----Closing Construction Monitor API Call------\n")
               elif event == "Utah Real Estate":
00087
00088
                   print(
                         f"\n\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ |
00089
               ---Initiating Utah Real Estate API Call-----")
00090
                    UtahRealEstateMain(UtahRealEstateInit())
00091
                     print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00092
               ----Closing Utah Real Estate API Call-----\n")
00093
               elif event == "Realtor.Com":
00094
                   print(
00095
                         f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ |
               ----Initiating Realtor.com API Call----
00096
                    realtorCom()
00097
                     print(
00098
                         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
               ----Closing Realtor.com API Call----
00099
               elif event == "CFPB Mortgage":
00100
                   print(
00101
                         f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ |
               ----Initiating ffiec.cfpb API Call----
00102
                    CFBP()
00103
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00104
                ---Closing ffiec.cfpb API Call----
00105
                 elif event == "Authorization Utility":
                   print(
00106
00107
                         f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ |
                 --Initiating Authorization Utility--
00108
                     AuthUtil()
                     print(
00110
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
                --Closing Authorization Utility--
00111
                elif event == "Open Data Folder":
00112
                   print(
                         f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00113
                 -Data Folder Opened-----
00114
                     try:
00115
                        os.system(f"start
     {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
00116
                     except:
00117
00118
                            os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
00119
                         except Exception as e:
                           print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00120
```

```
00122
                                  text="Documents folder not found. Please create a Windows recognized documents
     folder",
00123
                                  windowType="errorLarge")
00124
00125
                 elif event in ('Exit', None):
00126
                      try:
00127
                         break
00128
                      except Exception as e:
00129
                         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.")
00130
00131
                 elif event == sg.WIN_CLOSED or event == "Quit":
00132
                     break
00133
00134
             window.close()
00135
```

Here is the caller graph for this function:



3.10.4 Member Data Documentation

3.10.4.1 classObj

API_Calls.Initializer.initializer.classObj

Definition at line 41 of file Initializer.py.

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

· Initializer.py

3.11 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)
- def openFile (self)

Private Member Functions

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

Private Attributes

- __text
- __type
- __error
- __layout
- __windowObj
- thread
- __counter
- __docpath
- __errorFlag

3.11.1 Detailed Description

Definition at line 15 of file PopupWrapped.py.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 __init__()

00057

```
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 17 of file PopupWrapped.py.
          def __init__(self, text="", windowType="notice", error=None):
00017
00018
00019
          The __init__ function is the first function that gets called when an object of this class is created.
00020
          It sets up all the variables and creates a window for us to use.
00021
          Args:
             self: Represent the instance of the class
00022
00023
              text: Set the text of the window
00024
              windowType: Determine what type of window to create
00025
              error: Display the error message in the window
00026
         Returns:
00027
             Nothing
00028
         Doc Author:
          Willem van der Schans, Trelent AI
00029
00030
00031
              self.\__text = text
00032
              self.__type = windowType
              self.__error = error
00033
00034
              self.__layout = []
00035
              self.__windowObj = None
00036
              self.__thread = None
00037
              self.\_counter = 0
00038
              self.__docpath = None
00039
              self.__errorFlag = False
00040
00041
                  if "File Appended and Saved to " in self.__text:
00042
                 self.__docpath = str(self.__text[27:])
elif "File Saved to " in self.__text:
00043
00044
00045
                     self.__docpath = str(self.__text[14:])
00046
                  else:
                     pass
00047
00048
              except Exception as e:
00049
                 if self.__type == "savedLarge":
00050
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py
      | Error = {e} | Error creating self.__docpath open file button not available")
00052
                     self.__errorFlag = True
00053
                 else:
00054
                    pass
00055
             self.__createWindow()
```

Here is the call graph for this function:



3.11.3 Member Function Documentation

3.11.3.1 __createLayout()

Definition at line 58 of file PopupWrapped.py.

```
def __createLayout(self):
00058
00059
00060
                 \_createLayout function rac{is}{i} used to create the layout of the window.
00061
           The function takes class variables and returns a window layout.
00062
           It uses a series of if statements to determine what type of window it is, then creates a layout based
00063
          Args:
00064
               self: Refer to the current instance of a class
00065
          Returns:
00066
              A list of lists
00067
           Doc Author:
           Willem van der Schans, Trelent AI
00068
00069
00070
               sg.theme('Default1')
               __Line1 = None
00071
               __Line2 = None
00072
00073
               if self.__type == "notice":
00074
00075
                   __Line1 = [sg.Push(),
00076
                                sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00077
                                sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00078
                     _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
               elif self.__type == "noticeLarge":
00079
                    __Line1 = [sg.Push(),
00080
                                sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00081
00082
               __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
elif self.__type == "savedLarge":
00083
00084
                   if self.__errorFlag:
00085
00086
                        \_Line1 = [sg.Push(),
00087
                                    sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
```

```
00088
                                   sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00089
                       __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00090
                   else:
00091
                       \_Line1 = [sg.Push(),
                                   sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00092
00093
00094
                       __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10,
      1)), sg.Push()]
00095
              elif self.__type == "errorLarge":
00096
                  \_Line1 = [sg.Push(),
                              sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00097
                              sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00098
00099
                    _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
              elif self.__type == "FatalErrorLarge":
00100
00101
                 \_Line1 = [sg.Push(),
                              sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                              sq.Text(self.__text, justification="left", key="-textField-"), sq.Push()]
00103
                    _{\rm Line2} = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00104
00105
              elif self.__type == "error":
00106
                  \_Line1 = [sg.Push(),
00107
                              sg. Text (u' \setminus u274C', \ font=("Helvetica", \ 20, \ "bold"), \ justification="center"), \\
                              sg.Text(f"{self.__text}: {self.__error}", justification="center",
00108
     key="-textField-"),
00109
                              sa.Push()1
00110
                    _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
              elif self.__type == "AuthError":
00111
                  \_Line1 = [sg.Push(),
00112
                              sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00113
                              sg.Text(f"{self.__text}", justification="center", key="-textField-"),
00114
00115
                              sq.Push()]
                   __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
00116
                              sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
00117
              elif self.__type == "versionWindow":
00118
                  \_Line1 = [sg.Push(),
00119
                              sg.Text(f"{self.__text}", justification="left", key="-textField-"),
00120
0.0121
                              sq.Push()]
00122
                   __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00123
                              sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
              elif self.__type == "progress":
00124
00125
                  \_Line1 = [sg.Push(),
00126
                              sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00127
00128
              if self.__type == "progress":
00129
                   self.__layout = [__Line1, ]
              else:
00130
00131
                   self.__layout = [__Line1, __Line2]
00132
```

Here is the caller graph for this function:



3.11.3.2 __createWindow()

```
def PopupWrapped.PopupWrapped._createWindow ( self \ ) \quad [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 function first calls __creates first first calls __creates first first calls __creates first first first calls __creates first f

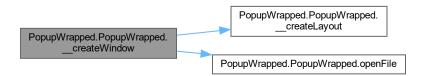
```
Args:
self: Reference the instance of the class
Returns:
A window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 133 of file PopupWrapped.py.

```
def __createWindow(self):
00134
00135
          The __createWindow function is used to create the window object that will be displayed.
00136
          The function takes class variables and a window object. The function first calls __createLayout, which
      creates the layout for the window based on what type of message it is (error, notice, progress). Then it
      uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as
      title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits
      until either 100 iterations have passed or an event has been triggered (such as clicking " Ok" or
      closing the window). Once one of these events occurs
00137
         Args:
00138
              self: Reference the instance of the class
00139
          Returns:
00140
              A window object
00141
          Doc Author:
00142
             Willem van der Schans, Trelent AI
00143
00144
             self.__createLayout()
00145
00146
              if self.__type == "progress":
                  self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
00147
     finalize=True.
00148
                                                modal=True,
00149
                                                keep_on_top=True,
00150
                                                disable_close=False,
00151
                                                icon=ImageLoader("taskbar_icon.ico"),
00152
                                                size=(290, 50))
              elif self.__type == "noticeLarge":
00153
00154
                  self.__windowObj = sq.Window(title="Notice", layout=self.__layout, finalize=True,
00155
                                                modal=True,
00156
                                                keep_on_top=True,
00157
                                                disable_close=False,
00158
                                                icon=ImageLoader("taskbar_icon.ico"))
              elif self.__type == "savedLarge":
00159
                 self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00160
00161
                                                modal=True,
00162
                                                keep_on_top=False,
00163
                                                disable_close=False,
00164
                                                icon=ImageLoader("taskbar_icon.ico"))
00165
              elif self.__type == "errorLarge":
00166
                  self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
00167
                                                modal=True,
00168
                                                keep_on_top=True,
00169
                                                disable_close=False,
00170
                                                icon=ImageLoader("taskbar_icon.ico"))
00171
              elif self.__type == "FatalErrorLarge":
00172
                  self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
00173
                                                modal=True,
00174
                                                keep_on_top=True,
00175
                                                disable_close=False,
00176
                                                icon=ImageLoader("taskbar_icon.ico"))
00177
              elif self.__type == "AuthError":
00178
                  self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout,
      finalize=True,
00179
                                                modal=True,
00180
                                                keep_on_top=True,
00181
                                                disable_close=False,
                                                icon=ImageLoader("taskbar_icon.ico"))
00182
00183
              elif self.__type == "versionWindow":
00184
                  self.__windowObj = sg.Window(title="Update Notice", layout=self.__layout, finalize=True,
00185
                                                modal=True,
00186
                                                keep_on_top=True,
00187
                                                disable_close=False,
00188
                                                icon=ImageLoader("taskbar_icon.ico"))
00189
              else:
                 self.__windowObj = sq.Window(title=self.__type.capitalize(), layout=self.__layout,
00190
      finalize=True,
00191
                                                modal=True,
00192
                                                keep on top=True,
00193
                                                disable close=False,
00194
                                                icon=ImageLoader("taskbar_icon.ico"),
```

```
00195
                                                     size=(290, 80))
00196
00197
                if self.__type != "progress" or self.__type.startswith("perm"):
00198
00199
                    while timer < 100:</pre>
00200
                         event, values = self.__windowObj.read()
00201
                         if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
00202
00203
                         elif event == "Open Generation Tool [Web Browser]":
00204
                             webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
00205
00206
                         elif event == "Open File":
00207
                             threadFile = threading. Thread(target=self.openFile,
00208
                                                               daemon=False)
00209
                             threadFile.start()
00210
                             time.sleep(3)
00211
                             break
                         elif event == "Download":
00212
                              # Todo Gitlab Update
00213
00214
                             webbrowser.open(settings.settingDownloadSourceLink, new=2,
00215
                                               autoraise=True)
00216
                         time.sleep(0.1)
00217
00218
00219
                    if self.__type == "FatalErrorLarge":
00220
                         try:
00221
                             os.svstem(
00222
                                  f"start
        \{ \texttt{Path} \, (\texttt{os.path.expandvars} \, (\texttt{r'\$APPDATA\$'}) \, ) \, . \, \\ \texttt{joinpath} \, (\texttt{'GardnerUtil'}) \, . \, \\ \texttt{joinpath} \, (\texttt{'Logs'}) \, \} \, \texttt{"}) 
00223
                         except Exception as e:
00224
                             print(
00225
                                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
       PopupWrapped.py | Error = {e} | Log Folder not found please search manually for
       %APPDATA%\Roaming\GardnerUtil\Logs\n")
00226
00227
                    self.__windowObj.close()
00228
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.11.3.3 openFile()

```
def PopupWrapped.PopupWrapped.openFile (
               self )
The openFile function opens the file that is associated with the
document object. It does this by calling os.system and passing it
self.__docpath as an argument.
Args:
self: Represent the instance of the object itself
Returns:
The filepath of the document
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 290 of file PopupWrapped.py.
00290
         def openFile(self):
00291
00292
         The openFile function opens the file that is associated with the
00293
             document object. It does this by calling os.system and passing it
00294
             self.__docpath as an argument.
00295
00296
00297
             self: Represent the instance of the object itself
00298
00299
00300
             The filepath of the document
00301
00302
         Doc Author:
         Willem van der Schans, Trelent AI
00303
00304
00305
             os.system(self.__docpath)
```

Here is the caller graph for this function:



3.11.3.4 stopWindow()

Definition at line 229 of file PopupWrapped.py.

```
def stopWindow(self):
00230
00231
          The stopWindow function is used to close the window object that was created in the startWindow
00232
          This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
00233
00234
             self: Represent the instance of the class
00235
          Returns:
00236
            The window object
00237
          Doc Author:
          Willem van der Schans, Trelent AI
00238
00239
00240
              self.__windowObj.close()
00241
```

3.11.3.5 textUpdate()

Definition at line 242 of file PopupWrapped.py.

def PopupWrapped.PopupWrapped.textUpdate (

```
def textUpdate(self, sleep=0.5):
00242
00243
00244
          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
00245
00246
          a loading effect for when something is being processed.
00247
          Args:
00248
              self: Refer to the object itself
00249
              sleep: Control the speed of the text update
00250
          Returns:
00251
              A string that is the current text of the text field
00252
          Doc Author:
          Willem van der Schans, Trelent AI
00253
00254
00255
              self.__counter += 1
00256
              if self.__counter == 4:
00257
                  self.\__counter = 1
00258
              newString = ""
00259
              if self.__type == "notice":
00260
00261
              elif self.__type == "error":
00262
00263
              elif self.__type == "progress":
00264
                  newString = f"{self.__text}{'.' * self.__counter}"
00265
              self.__windowObj.write_event_value('update-textField-', newString)
00266
00267
              time.sleep(sleep)
00268
```

3.11.3.6 windowPush()

```
00272
         The windowPush function is used to update the values of a window object.
00273
             The function takes in an event and values from the window object, then checks if the event starts
     with 'update'.
00274
             If it does, it will take everything after 'update' as a key for updating that specific value.
00275
             It will then update that value using its key and refresh the window.
         Args:
00277
             self: Reference the object that is calling the function
00278
        Returns:
00279
            A tuple containing the event and values
00280
         Doc Author:
        Willem van der Schans, Trelent AI
00281
00282
00283
             event, values = self.__windowObj.read()
00284
00285
             if event.startswith('update'):
00286
                 __key_to_update = event[len('update'):]
                 self.__windowObj[__key_to_update].update(values[event])
00287
                 self.__windowObj.refresh()
00288
00289
```

3.11.4 Member Data Documentation

3.11.4.1 __counter

PopupWrapped.__counter [private]

Definition at line 37 of file PopupWrapped.py.

3.11.4.2 docpath

PopupWrapped. __docpath [private]

Definition at line 38 of file PopupWrapped.py.

3.11.4.3 __error

```
PopupWrapped.PopupWrapped.__error [private]
```

Definition at line 33 of file PopupWrapped.py.

3.11.4.4 __errorFlag

```
PopupWrapped.PopupWrapped.__errorFlag [private]
```

Definition at line 39 of file PopupWrapped.py.

3.11.4.5 __layout

PopupWrapped.PopupWrapped.__layout [private]

Definition at line 34 of file PopupWrapped.py.

3.11.4.6 __text

PopupWrapped.PopupWrapped.__text [private]

Definition at line 31 of file PopupWrapped.py.

3.11.4.7 thread

PopupWrapped.PopupWrapped.__thread [private]

Definition at line 36 of file PopupWrapped.py.

3.11.4.8 __type

PopupWrapped.PopupWrapped.__type [private]

Definition at line 32 of file PopupWrapped.py.

3.11.4.9 __windowObj

```
PopupWrapped.PopupWrapped.__windowObj [private]
```

Definition at line 35 of file PopupWrapped.py.

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

· PopupWrapped.py

3.12 Core.realtorCom Class Reference

Public Member Functions

```
• def __init__ (self)
```

Public Attributes

- dfState
- dfCounty
- dfZip
- uiString

Private Member Functions

- def showUi (self)
- def __linkGetter (self)
- def __dataUpdater (self)

Private Attributes

- __page_html
- __update_date
- __last_date
- __idDict
- __linkDict

3.12.1 Detailed Description

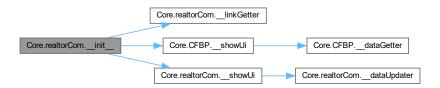
Definition at line 16 of file Realtor/Core.py.

3.12.2 Constructor & Destructor Documentation

00058

```
3.12.2.1 __init__()
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
self: Represent the instance of the class
Returns:
A new object
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 18 of file Realtor/Core.py.
00018
          def __init__(self):
00019
          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
00021
      anything else in your class.
00022
00023
00024
             self: Represent the instance of the class
00025
         Returns:
00026
00027
            A new object
00028
00029
        Doc Author:
         Willem van der Schans, Trelent AI
00030
00031
00032
              self.__page_html = None
00033
              self.__update_date = None
00034
              self.__last_date = None
              self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00035
00036
              self.__linkDict = {}
00037
              self.dfState = None
00038
              self.dfCounty = None
00039
              self.dfZip = None
00040
              self.uiString = "Files Saved to \n"
00041
00042
              eventReturn = confirmDialog()
00043
              if eventReturn == "Continue":
00044
                  page_html = requests.get(settings.settingRealtorLink).text
00045
                  self.__page_html = BeautifulSoup(page_html, "html.parser")
00046
                  startTime = datetime.datetime.now().replace(microsecond=0)
00047
                  self.__linkGetter()
00048
                 print(
                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Link Dictionary =
00049
      {self.__idDict}")
00050
                 self.__showUi()
00051
                  PopupWrapped(text=self.uiString, windowType="noticeLarge")
00052
                 print(
00053
                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
     in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0)
     startTime).total_seconds()))}")
00054
             else:
00055
                print(
                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
00056
      Request")
00057
```

Here is the call graph for this function:



3.12.3 Member Function Documentation

3.12.3.1 dataUpdater()

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 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 114 of file Realtor/Core.py.

```
00114
         def __dataUpdater(self):
00115
00116
00117
          The __dataUpdater function is a private function that updates the dataframes for each of the three
00118
              types of realtor data. It takes class variables and return the path to the saved file. The
      function first creates an empty
00119
              dictionary called tempdf, then iterates through each key in self.__idDict (which contains all
      three ids).
00120
              For each key, it reads in a csv file from the link associated with that id and saves it to tempdf
      as a pandas
00121
              DataFrame object. Then, depending on which type of realtor data we are dealing with
      (State/County/Zip), we save
00122
00123
00124
          Args:
              self: Access the attributes and methods of the class
00125
00126
00127
          Returns:
              The path of the saved file
00128
00129
```

```
00130
          Doc Author:
00131
              Willem van der Schans, Trelent AI
00132
00133
              for key, value in self.__idDict.items():
00134
                  tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00135
00136
                  if key == "State":
00137
                      self.dfState = tempdf
00138
                  elif key == "County":
                      self.dfCounty = tempdf
00139
00140
                  elif key == "Zip":
00141
                      self.dfZip = tempdf
00142
                  FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00143
00144
                  self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

Here is the caller graph for this function:



3.12.3.2 __linkGetter()

```
\begin{tabular}{ll} $\operatorname{def Core.realtorCom.}\_\operatorname{linkGetter} \ ( \\ & self \ ) & [private] \end{tabular}
```

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 87 of file Realtor/Core.py.

```
00087
          def __linkGetter(self):
00088
00089
00090
          The __linkGetter function is a private function that takes the idDict dictionary and adds
00091
          a link to each entry in the dictionary. The link is used to access historical data for each
00092
          scope symbol.
00093
00094
          Args:
00095
              self: Refer to the object itself
00096
00097
00098
              A dictionary of all the links to the history pages
00099
00100
         Doc Author:
```

```
00101
                Willem van der Schans, Trelent AI
00102
00103
                for key, value in self.__idDict.items():
                    for row in self.__page_html.find_all("div", {"class": "monthly"}):
00104
00105
                         try:
00106
                             for nestedRow in row.find_all("a"):
                                  if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
    self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
00107
00108
00109
                         except Exception as e:
                             print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00110
      Realtor/Core.py | Error = {e} | Error while getting document links for realtor.com")
00111
                             RESTError (801)
00112
                             raise SystemExit(801)
00113
```

Here is the caller graph for this function:



3.12.3.3 __showUi()

```
\begin{tabular}{ll} $\operatorname{def Core.realtorCom.\_showUi} & ( \\ & self ) & [\operatorname{private}] \end{tabular}
```

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 59 of file Realtor/Core.py.

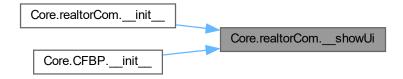
```
00059
         def __showUi(self):
00060
00061
00062
          The __showUi function is a helper function that creates and displays the progress window.
00063
          It also starts the dataUpdater thread, which will update the progress bar as it runs.
00064
00065
00066
         Args:
00067
              self: Represent the instance of the class
00068
00069
          Returns:
00070
              A popupwrapped object
00071
00072
          Doc Author:
```

```
00073
              Willem van der Schans, Trelent AI
00074
00075
              uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)
00076
00077
              threadGui = threading.Thread(target=self.__dataUpdater,
00078
00079
              threadGui.start()
08000
00081
              while threadGui.is_alive():
                  uiObj.textUpdate()
                  uiObj.windowPush()
00084
              else:
00085
                  uiObj.stopWindow()
00086
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.12.4 Member Data Documentation

3.12.4.1 __idDict

Core.realtorCom.__idDict [private]

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

3.12.4.2 __last_date

Core.realtorCom.__last_date [private]

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

3.12.4.3 __linkDict

Core.realtorCom.__linkDict [private]

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

3.12.4.4 __page_html

Core.realtorCom.__page_html [private]

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

3.12.4.5 __update_date

Core.realtorCom.__update_date [private]

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

3.12.4.6 dfCounty

Core.realtorCom.dfCounty

Definition at line 38 of file Realtor/Core.py.

3.12.4.7 dfState

Core.realtorCom.dfState

Definition at line 37 of file Realtor/Core.py.

3.12.4.8 dfZip

```
Core.realtorCom.dfZip
```

Definition at line 39 of file Realtor/Core.py.

3.12.4.9 uiString

```
Core.realtorCom.uiString
```

Definition at line 40 of file Realtor/Core.py.

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

· Realtor/Core.py

3.13 Settings.settings Class Reference

Static Public Attributes

- str settingVersion = "1.2.0"
- str settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest"
- str settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
- str settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApiUtility/releases/latest'
- str settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
- str settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"
- str settingRealtorLink = "https://www.realtor.com/research/data/"
- str settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"

3.13.1 Detailed Description

Definition at line 6 of file Settings.py.

3.13.2 Member Data Documentation

3.13.2.1 settingCFBPLink

str Settings.settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
[static]

Definition at line 21 of file Settings.py.

3.13.2.2 settingCMRestDomain

str Settings.settings.settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"
[static]

Definition at line 25 of file Settings.py.

3.13.2.3 settingDownloadSourceLink

 $str\ Settings.settingSownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApi \leftarrow Utility/releases/latest' [static]$

Definition at line 17 of file Settings.py.

3.13.2.4 settingGenerationToolLink

str Settings.settings.settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
[static]

Definition at line 15 of file Settings.py.

3.13.2.5 settingGithubApiUrl

str Settings.settings.settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApi↔ Utility/releases/latest" [static]

Definition at line 11 of file Settings.py.

3.13.2.6 settingRealtorLink

```
str Settings.settings.settingRealtorLink = "https://www.realtor.com/research/data/" [static]
```

Definition at line 29 of file Settings.py.

3.13.2.7 settingURERestDomain

```
str Settings.settinguRERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"
[static]
```

Definition at line 33 of file Settings.py.

3.13.2.8 settingVersion

```
str Settings.settingVersion = "1.2.0" [static]
```

Definition at line 9 of file Settings.py.

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

· Settings.py

3.14 Core.UtahRealEstateInit Class Reference

Public Member Functions

def init (self)

Public Attributes

- StandardStatus
- ListedOrModified
- 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 ()

3.14.1 Detailed Description

Definition at line 25 of file UtahRealEstate/Core.py.

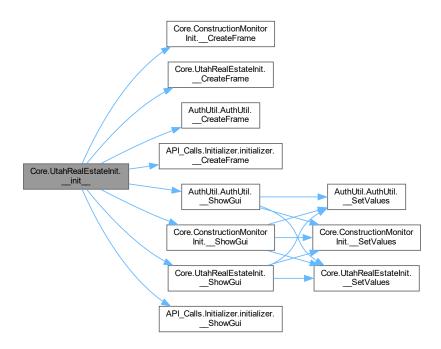
3.14.2 Constructor & Destructor Documentation

3.14.2.1 __init__()

Definition at line 27 of file UtahRealEstate/Core.py.

```
def __init__(self):
00029
00030
         The __init__ function is called when the class is instantiated.
00031
         It sets up the initial state of the object.
00032
00033
00034
        Args:
             self: Represent the instance of the class
00035
00036
00037
        Returns:
00038
             The __createframe function
00039
00040
        Doc Author:
         Willem van der Schans, Trelent AI
00041
00042
00043
             self.StandardStatus = None
00044
             self.ListedOrModified = None
00045
             self.dateStart = None
```

Here is the call graph for this function:



3.14.3 Member Function Documentation

3.14.3.1 CreateFrame()

```
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:

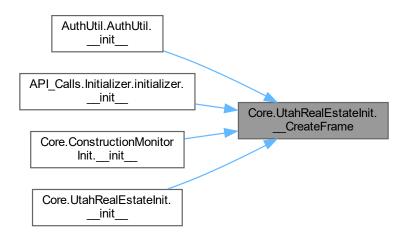
Returns:
A list of lists, which is used to create the gui

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

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

```
def __CreateFrame():
00093
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
00100
00101
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
                     sq.Push(),
                     sg.Text("Utah Real Estate Utility", font=("Helvetica", 12, "bold"),
00113
     justification="center"),
00114
                     sa.Push().
00115
                     sg.Push()]
00116
             line1 = [sq.HSeparator()]
00117
00118
             00119
00120
     size=(31, 1))1
00121
             00122
00123
      "Close Date"],
00124
                                 key="-type-", size=(31, 1))]
00125
             line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00126
                      sg.Input (default\_text = (date.today() - timedelta(days = 14)).strftime("%Y-%m-%d"),
00127
     key="-DateStart-",
00128
                              disabled=False, size=(20, 1)),
                     sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00129
     target="-DateStart-")]
00130
00131
             line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                     sg.Input(default_text=(date.today().strftime("%Y-%m-%d")), key="-DateEnd-",
00132
     disabled=False,
00133
                              size=(20, 1)),
00134
                     sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-',
     target="-DateEnd-")]
00135
00136
             line7 = [sg.HSeparator()]
00137
00138
             line8 = [sq.Push(),
00139
                      sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00140
                      sq.Push()]
00141
00142
             line9 = [sg.HSeparator()]
00143
00144
             line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                      sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00145
00146
                               size=(20, 1)),
00147
                       sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00148
                                    target="-AppendingFile-")]
00149
00150
             line11 = [sg.HSeparator()]
00151
00152
             line12 = [sq.Push(), sq.Submit(focus=True), sq.Quit(), sq.Push()]
00153
00154
             layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
00155
                      line121
00156
00157
             return lavout
00158
```

Here is the caller graph for this function:



3.14.3.2 SetValues()

```
\begin{tabular}{ll} $\operatorname{def Core.UtahRealEstateInit.} \underline{\quad \  } & \\ & self, \\ & values \ ) & [private] \\ \end{tabular}
```

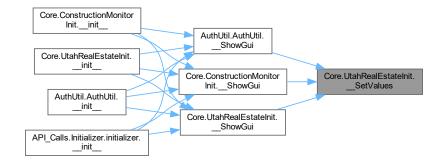
The __SetValues function is used to set the values of the variables that are used in the __GetData function. The values are passed from a dictionary called 'values' which is created by parsing through an XML file using ElementTree. This function also sets default values for some of these variables if 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 159 of file UtahRealEstate/Core.py.

```
00168
              self: Represent the instance of the class
00169
              values: Pass the values from the gui to this function
00170
00171
00172
              A dictionary with the following keys:
00173
00174
         Doc Author:
          Willem van der Schans, Trelent AI
00175
00176
00177
              self.StandardStatus = values["-status-"]
00178
00179
              self.ListedOrModified = values["-type-"]
00180
              if values["-DateStart-"] != "":
00181
00182
                 self.dateStart = values["-DateStart-"]
00183
              else:
00184
                  self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00185
              if values["-DateEnd-"] != "":
00186
                 self.dateEnd = values["-DateEnd-"]
00187
00188
              else:
00189
                  self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00190
00191
              self.select = None
00192
              if values["-append_file-"] != "":
00193
                  self.append_file = str(values["-append_file-"])
00194
00195
              else:
00196
                  self.append_file = None
00197
00198
```

Here is the caller graph for this function:



3.14.3.3 __ShowGui()

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

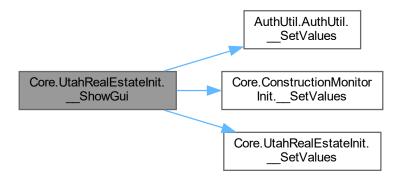
Returns:
A dictionary of values

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

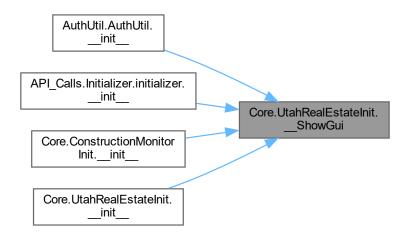
Definition at line 53 of file UtahRealEstate/Core.py.

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

Here is the call graph for this function:



Here is the caller graph for this function:



3.14.4 Member Data Documentation

3.14.4.1 append_file

Core.UtahRealEstateInit.append_file

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

3.14.4.2 dateEnd

Core.UtahRealEstateInit.dateEnd

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

3.14.4.3 dateStart

Core.UtahRealEstateInit.dateStart

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

3.14.4.4 file_name

Core.UtahRealEstateInit.file_name

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

3.14.4.5 ListedOrModified

Core.UtahRealEstateInit.ListedOrModified

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

3.14.4.6 select

Core.UtahRealEstateInit.select

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

3.14.4.7 StandardStatus

Core.UtahRealEstateInit.StandardStatus

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

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

UtahRealEstate/Core.py

3.15 Core.UtahRealEstateMain Class Reference

Public Member Functions

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

Public Attributes

- dataframe
- keyPath
- filePath
- key

Private Member Functions

- def __ParameterCreator (self)
- def <u>getCount</u> (self)
- def <u>getCountUI</u> (self)

Private Attributes

- __batches
- siteClass
- __headerDict
- __parameterString
- __appendFile
- dateStart
- __dateEnd
- __restDomain
- __record_val

3.15.1 Detailed Description

Definition at line 199 of file UtahRealEstate/Core.py.

3.15.2 Constructor & Destructor Documentation

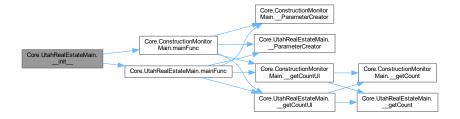
3.15.2.1 __init__()

```
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.
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 201 of file UtahRealEstate/Core.py.
00201
          def __init__(self, siteClass):
00202
00203
00204
          The __init__ function is the first function that runs when an object of this class is created.
00205
          It sets up all the variables and functions needed for this class to work properly.
00206
00207
00208
              self: Represent the instance of the class
00209
              siteClass: Determine which site to pull data from
00210
00211
         Returns:
00212
             Nothing
00213
00214
         Doc Author:
00215
              Willem van der Schans, Trelent AI
00216
00217
              self.dataframe = None
              self.\_batches = 0
00218
              self.__siteClass = siteClass
00219
              self.__headerDict = None
00220
              self.__parameterString = ""
00221
              self.__appendFile = None
00222
00223
              self.__dateStart = None
              self.__dateEnd = None
00224
00225
              self.__restDomain = settings.settingURERestDomain
00226
              self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                  "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00227
              self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00228
                  "Security").joinpath("auth.json")
00229
              self.key = None
00230
00231
              self.__record_val = None
00232
00233
00234
                  self.mainFunc()
00235
              except KeyError as e:
                  \ensuremath{\sharp} This allows for user cancellation of the program using the quit button
00236
                  if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00237
00238
                     RESTError (1101)
00239
                     print(e)
00240
                      pass
00241
                  else:
00242
                     pass
00243
              except Exception as e:
00244
                  print(e)
                  RESTError(1001)
00245
```

raise SystemExit(1001)

00246

Here is the call graph for this function:



3.15.3 Member Function Documentation

3.15.3.1 __getCount()

```
\begin{tabular}{ll} $\operatorname{def Core.UtahRealEstateMain.} \underline{\quad \  } & \\ & self \end{tabular} ) & [\operatorname{private}] \end{tabular}
```

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 366 of file UtahRealEstate/Core.py.

```
def __getCount(self):
00367
00368
          The __getCount function is used to determine the number of records that will be returned by the query.
00369
          This function is called when a user calls the count() method on a ReST object. The __getCount function
      uses
00370
          the $count parameter in OData to return only an integer value representing how many records would be
     returned
00371
          by the query.
00372
00373
          Args:
00374
              self: Represent the instance of the class
00375
00376
         Returns:
00377
              The number of records in the data set
00378
00379
         Doc Author:
          Willem van der Schans, Trelent AI
00380
00381
00382
              __count_resp = None
00383
00384
              try:
```

```
00385
                  __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00386
                                               headers=self.__headerDict)
00387
00388
              except requests.exceptions.Timeout as e:
00389
                  print(e)
00390
                  RESTError (790)
00391
                  raise SystemExit(790)
00392
              except requests.exceptions.TooManyRedirects as e:
00393
                  print(e)
00394
                  RESTError (791)
00395
                  raise SystemExit (791)
00396
              except requests.exceptions.MissingSchema as e:
00397
                  print(e)
00398
                  RESTError (1101)
00399
              except requests.exceptions.RequestException as e:
00400
                  print(e)
00401
                  RESTError (405)
00402
                  raise SystemExit (405)
00403
00404
              self.__record_val = int(__count_resp.json()["@odata.count"])
00405
```

Here is the caller graph for this function:



3.15.3.2 __getCountUI()

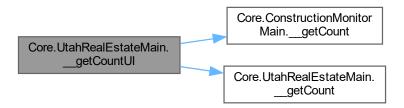
00414

```
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.
self: Represent the instance of the class
Returns:
A popupwrapped object
Doc Author:
Willem van der Schans, Trelent AI
Definition at line 406 of file UtahRealEstate/Core.py.
00406
           def __getCountUI(self):
00407
00408
           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.
00409
00410
00411
00412
00413
```

self: Represent the instance of the class

```
00415
00416
          Returns:
00417
             A popupwrapped object
00418
00419
          Doc Author:
00420
              Willem van der Schans, Trelent AI
00421
00422
              uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00423
              threadGui = threading.Thread(target=self.__getCount,
00424
00425
                                            daemon=False)
00426
              threadGui.start()
00427
00428
              while threadGui.is_alive():
00429
                  uiObj.textUpdate()
00430
                  uiObj.windowPush()
00431
              else:
                  uiObj.stopWindow()
00432
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.15.3.3 __ParameterCreator()

```
def Core.UtahRealEstateMain.\_ParameterCreator ( self ) [private]
```

```
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
```

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.

Definition at line 325 of file UtahRealEstate/Core.py.

```
00325
          def __ParameterCreator(self):
00326
               _ParameterCreator function is used to create the filter string for the ReST API call.
00327
00328
          The function takes in a siteClass object and extracts all of its parameters into a dictionary.
00329
          It then creates an appropriate filter string based on those parameters.
00330
00331
00332
              self: Bind the object to the class
00333
00334
          Returns:
00335
              A string to be used as the parameter in the api call
00336
00337
          Doc Author:
          Willem van der Schans, Trelent AI
00338
00339
00340
              filter_string = ""
00341
00342
              __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                               not key.startswith('__') and not callable(key)}
00343
00344
00345
              self.__appendFile = __Source_dict["append_file"]
00346
              __Source_dict.pop("append_file")
00347
00348
              temp_dict = copy.copy(__Source_dict)
00349
              for key, value in temp_dict.items():
00350
                  if value is None:
00351
                      __Source_dict.pop(key)
00352
00353
                      pass
00354
00355
              if __Source_dict["ListedOrModified"] == "Listing Date":
00356
                  filter_string =
      f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%201e%20{__Source_dict['dateEn
00357
              elif __Source_dict["ListedOrModified"] == "Modification Date":
00358
                  filter_string =
      f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:002%20and%20ModificationTimestamp%20le%20{__Source_dict['dateStart']}
00359
              elif __Source_dict["ListedOrModified"] == "Close Date":
00360
                  filter_string =
      f"$filter=CloseDate%20gt%20{__Source_dict['dateEnd']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"
00361
00362
              filter string = filter string +
      f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus′{__Source_dict['StandardStatus']}'"
00363
00364
              self.__parameterString = filter_string
00365
```

Here is the caller graph for this function:



3.15.3.4 mainFunc()

```
\begin{tabular}{ll} def & Core. Utah Real Estate Main.main Func ( \\ & self ) \end{tabular}
```

The mainFunc function is the main function of this module. It will be called by the GUI when a user clicks on the " Run" button in the GUI. The mainFunc function should contain all of your code for running your pro should return a dataframe that contains all the data you want to display in your final report.

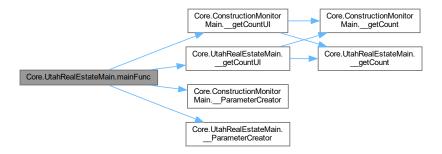
```
Args:
self: Reference the object itself
Returns:
A dataframe
Doc Author:
Willem van der Schans, Trelent AI
```

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

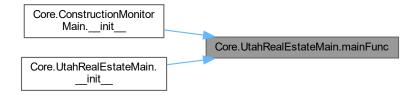
```
00248
          def mainFunc(self):
00249
00250
00251
          The mainFunc function is the main function of this module. It will be called by the GUI when a user
      clicks on
00252
          the " Run" button in the GUI. The mainFunc function should contain all of your code for
      running your program, and it
00253
          should return a dataframe that contains all the data you want to display in your final report.
00254
00255
00256
              self: Reference the object itself
00257
          Returns:
00258
00259
             A dataframe
00260
          Doc Author:
00261
          Willem van der Schans, Trelent AI
00262
00263
00264
              passFlag = False
00265
              while not passFlag:
00266
00267
                   if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00268
00269
                           f = open(self.keyPath, "rb")
00270
                           key = f.readline()
00271
                           f.close()
00272
                           f = open(self.filePath, "rb")
00273
                           authDict = json.load(f)
00274
                           fernet = Fernet(key)
00275
                           authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00276
                           self.__headerDict = {authDict["ure"]["parameter"]: authkey}
00277
                           passFlag = True
00278
                       except Exception as e:
00279
                          print(
00280
                               f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
     UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
00281
                           AuthUtil()
00282
                  else:
                       AuthUtil()
00283
00284
00285
              self. ParameterCreator()
00286
00287
              print (
                  f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String =
00288
      {self.
              _parameterString}")
00289
              print(
00290
                   f''' \{ \text{datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \} \mid \text{Rest Domain} = \{ \text{Rest Domain} = \text{Rest Domain} \} 
      {self.__restDomain}")
00291
```

```
00292
                                                    self.__getCountUI()
 00293
 00294
                                                    if self.__record_val is None:
 00295
                                                                  self.__record_val = 0
 00296
 00297
                                                    self.__batches = BatchCalculator(self.__record_val, None)
 00298
 00299
00300
                                                                  f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \mid Batches = f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \mid Batches = f"\{datetime.datetime.today().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftim
                       {self.__batches} | Rows {self.__record_val}")
 00301
 00302
                                                    if self.__batches != 0:
 00303
                                                                   startTime = datetime.datetime.now().replace(microsecond=0)
                                                                   eventReturn = BatchInputGui(self.__batches, self.__record_val)
 00304
 00305
                                                                   if eventReturn == "Continue":
 00306
                                                                                print(
00307
                                                                                                 f'''{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
                      {self.__batches} batches sent to server")
 00308
                                                                                 BatchGuiObject = BatchProgressGUI (RestDomain=self.__restDomain,
00309
                                                                                                                                                                                                                ParameterDict=self.__parameterString,
                                                                                                                                                                                                               HeaderDict=self.__headerDict,
BatchesNum=self.__batches,
00310
00311
                                                                                                                                                                                                                Type="utah_real_estate")
00312
 00313
                                                                                 BatchGuiObject.BatchGuiShow()
00314
                                                                                 self.dataframe = BatchGuiObject.dataframe
00315
                                                                                 print(
00316
                                                                                                f''' \{ \text{datetime.datetime.today().strftime('\%m-\%d-\%Y \%H:\%M:\%S.\%f')[:-3]} \ | \ \text{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()))}")
00317
                                                                               FileSaver("ure", self.dataframe, self.__appendFile)
00318
                                                                   else:
                                                                               print(
00319
                                                                                               f" \{ \texttt{datetime.datetime.today().strftime('\%m-\%d-\%Y \%H:\%M:\%S.\%f')[:-3] } \ | \ \texttt{Request for the property of the property o
00320
                      {self.__batches} batches canceled by user")
00321
                                                   else:
                                                                  RESTError (994)
00322
00323
                                                                   raise SystemExit(994)
00324
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.15.4 Member Data Documentation

3.15.4.1 __appendFile

Core.UtahRealEstateMain.__appendFile [private]

Definition at line 222 of file UtahRealEstate/Core.py.

3.15.4.2 __batches

Core.UtahRealEstateMain.__batches [private]

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

3.15.4.3 __dateEnd

Core.UtahRealEstateMain.__dateEnd [private]

Definition at line 224 of file UtahRealEstate/Core.py.

3.15.4.4 __dateStart

```
Core.UtahRealEstateMain.__dateStart [private]
```

Definition at line 223 of file UtahRealEstate/Core.py.

3.15.4.5 __headerDict

```
Core.UtahRealEstateMain.__headerDict [private]
```

Definition at line 220 of file UtahRealEstate/Core.py.

3.15.4.6 __parameterString

```
Core.UtahRealEstateMain.__parameterString [private]
```

Definition at line 221 of file UtahRealEstate/Core.py.

3.15.4.7 __record_val

```
Core.UtahRealEstateMain.__record_val [private]
```

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

3.15.4.8 restDomain

```
Core.UtahRealEstateMain.__restDomain [private]
```

Definition at line 225 of file UtahRealEstate/Core.py.

3.15.4.9 __siteClass

Core.UtahRealEstateMain.__siteClass [private]

Definition at line 219 of file UtahRealEstate/Core.py.

3.15.4.10 dataframe

Core.UtahRealEstateMain.dataframe

Definition at line 217 of file UtahRealEstate/Core.py.

3.15.4.11 filePath

Core.UtahRealEstateMain.filePath

Definition at line 228 of file UtahRealEstate/Core.py.

3.15.4.12 key

Core.UtahRealEstateMain.key

Definition at line 230 of file UtahRealEstate/Core.py.

3.15.4.13 keyPath

Core.UtahRealEstateMain.keyPath

Definition at line 226 of file UtahRealEstate/Core.py.

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

· UtahRealEstate/Core.py

Chapter 4

File Documentation

4.1 __init__.py

4.2 _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()
```

4.3 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
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
00018 class AuthUtil:
00019
00020
         def __init__(self):
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
         Aras:
00027
             self: Represent the instance of the class
00028
00029
         Returns:
00030
             None
```

```
00031
00032
                 Doc Author:
                 Willem van der Schans, Trelent AI
00033
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
                        self.jsonDict = {}
00043
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
00051
                        else:
00052
                               if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
00053
                                     os.mkdir(self.filePath)
00054
                               else:
                                      os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
00055
00056
                                      os.mkdir(self.filePath)
00057
00058
                        if os.path.exists(self.keyPath):
00059
00060
                        else:
00061
                               if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00062
                                      os.mkdir(self.keyPath)
00063
                               else:
                                      \verb|os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))| \\
00064
                                      os.mkdir(self.keyPath)
00065
00066
                        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00067
00068
                                      f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00069
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()
                               f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00078
00079
                               f.write(self.k)
00080
                               f.close()
00081
00082
00083
                                      os.remove(self.filePath.joinpath("auth.json"))
00084
                               except Exception as e:
00085
                                       # Logging
00086
                                      print(
                                               f" \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ Authutil.py \ | \ Auth
00087
          Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
00088
00089
00090
                               f = open(self.filePath.joinpath("auth.json"), "wb")
                               f.close()
00091
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
                               print(
00100
                                      f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py | Error =
00101
           {e} | Error when setting the key file as hidden. This is either a Permission error or Input Error.
          Continuing...")
00102
00103
00104
                 def SetValues(self, values):
00105
                         ....
00106
```

4.3 AuthUtil.py 121

```
00107
                _SetValues function is called when the user clicks on the "OK" button in the window.
00108
          It takes a dictionary of values as an argument, and then uses those values to update
00109
          the auth.json file with new keys for both Utah Real Estate and Construction Monitor.
00110
00111
00112
              self: Make the function a method of the class
00113
              values: Store the values that are entered into the form
00114
00115
00116
              A dictionary of the values entered by the user
00117
00118
          Doc Author:
          Willem van der Schans, Trelent AI
00119
00120
00121
              ureCurrent = None
00122
              cmCurrent = None
00123
              keyFile = None
00124
              self.popupFlag = False
00125
00126
              fernet = Fernet(self.k)
00127
00128
                  f = open(self.filePath.joinpath("auth.json"), "r")
00129
00130
                  keyFile = json.load(f)
00131
                  fileFlag = True
00132
              except:
                  fileFlag = False
00133
00134
00135
              \# Try initial decoding, if fails pass and write new keys and files
00136
              if fileFlag:
00137
00138
                      ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00139
                  except Exception as e:
00140
                      # Logging
                      print(
   f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py
00141
00142
      |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE
      functionality will be accessed")
00143
                      ureCurrent = None
00144
00145
                      cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00146
00147
                  except Exception as e:
                      # Logging
00148
00149
                      print(
                           f'' \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ Authutil.py \} 
00150
      |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM
      functionality will be accessed")
00151
                      cmCurrent = None
00152
00153
              if values["-ureAuth-"] != "":
00154
                  self.jsonDict.update(
                      {"ure": {"parameter": "Authorization", "auth":
00155
      fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00156
                  self.passFlagUre = True
00157
              elif ureCurrent is not None:
00158
                self.jsonDict.update(
                      {"ure": {"parameter": "Authorization", "auth":
00159
      fernet.encrypt(ureCurrent.encode()).decode()})
00160
                  self.passFlagUre = True
00161
              else:
00162
                 pass
00163
00164
              if values["-cmAuth-"] != "":
                  if values["-cmAuth-"].startswith("Basic"):
00165
00166
                      self.jsonDict.update(
                          {"cm": {"parameter": "Authorization",
00167
                                   "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00168
00169
                      self.passFlagCm = True
00170
                  else:
00171
                      PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction
     Monitor",
00172
                                   windowType="AuthError")
                      self.popupFlag = True
00173
00174
00175
              elif ureCurrent is not None:
00176
                 self.jsonDict.update(
                      {"cm": {"parameter": "Authorization", "auth":
00177
      fernet.encrypt(cmCurrent.encode()).decode()}})
00178
                  self.passFlagUre = True
00179
              else:
```

```
00180
00181
00182
              if not self.passFlagUre and not self.passFlagCm:
00183
                 PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction
     Monitor",
                               windowType="errorLarge")
00184
00185
              if self.passFlagCm and not self.passFlagUre:
                  PopupWrapped("Please make sure you provide a key for Utah Real estate",
00186
      windowType="errorLarge")
00187
              if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
00188
                  PopupWrapped("Please make sure you provide a key for Construction Monitor",
      windowType="errorLarge")
00189
             if self.popupFlag:
00190
00191
              else:
00192
                 jsonOut = json.dumps(self.jsonDict, indent=4)
00193
                  f = open(self.filePath.joinpath("auth.json"), "w")
00194
                  f.write(jsonOut)
00195
00196
          def __ShowGui(self, layout, text):
00197
00198
00199
          The ShowGui function is a helper function that displays the GUI to the user.
00200
          It takes in two arguments: layout and text. The layout argument is a list of lists,
00201
          which contains all the elements that will be displayed on screen. The text argument
00202
          is simply what will be displayed at the top of the window.
00203
00204
00205
              self: Represent the instance of the class
00206
              layout: Pass the layout of the qui to be displayed
00207
              text: Set the title of the window
00208
00209
          Returns:
          A window object
00210
00211
00212
              window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00213
                                 finalize=True,
                                 icon=ImageLoader("taskbar_icon.ico"))
00214
00215
00216
              while not self.passFlagUre or not self.passFlagCm:
00217
                  event, values = window.read()
00218
                  if event == "Submit":
00219
00220
                      try:
00221
                          self.__SetValues(values)
00222
                      except Exception as e:
00223
                          print(e)
00224
                          RESTError (993)
00225
                      finally:
00226
00227
                  elif event == sg.WIN_CLOSED or event == "Quit":
00228
00229
00230
                  else:
00231
00232
00233
              window.close()
00234
00235
          def __CreateFrame(self):
00236
          The __CreateFrame function creates the GUI layout for the Authentication Utility.
00237
00238
          It is called by __init__ and returns a list of lists that contains all the elements
00239
          that will be displayed in the window.
00240
00241
          Args:
00242
             self: Access the class attributes and methods
00243
00244
         Returns:
00245
            A list of lists
00246
00247
         Doc Author:
          Trelent
00248
00249
00250
              sg.theme('Default1')
00251
00252
              line00 = [sg.HSeparator()]
00253
00254
              line0 = [sg.Image(ImageLoader("logo.png")),
00255
                       sq.Push(),
                       sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
00256
      justification="center"),
```

```
00257
                       sg.Push(),
00258
                       sg.Push()]
00259
00260
              line1 = [sg.HSeparator()]
00261
00262
              line2 = [sg.Push(),
00263
                       sg.Text("Utah Real Estate API Key: ", justification="center"),
00264
00265
00266
              line3 = [sg.Push(),
00267
                       sg.Input(default_text="123", key="-ureAuth-", disabled=False,
00268
                                 size=(40, 1)),
00269
                       sg.Push()]
00270
00271
              line4 = [sg.HSeparator()]
00272
00273
              line5 = [sq.Push(),
00274
                       sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
00275
                       sq.Push()]
00276
00277
              line6 = [sq.Push(),
00278
                       sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
00279
                                 size=(40, 1)),
00280
                       sg.Push()]
00281
              line7 = [sg.HSeparator()]
00282
00283
00284
              line8 = [sq.Push(),
00285
                       sg.Text(self.outcomeText, justification="center"),
00286
                       sq.Push()]
00287
00288
              line9 = [sg.HSeparator()]
00289
00290
              line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00291
00292
              layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]
00293
00294
              return lavout
```

4.4 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 \underline{i}n 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
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
          is created, and it sets up the attributes of that object. In this case, we are setting up our
00047
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
              self.__parameterDict = ParameterDict
00068
00069
              self.\__restDomain = RestDomain
              self.__headerDict = HeaderDict
00070
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
00084
              self: Represent the instance of the class
00085
00086
00087
             The result of the constructionmonitorprocessor function
00088
00089
          Doc Author:
          Willem van der Schans, Trelent AI
00090
00091
00092
              self.ConstructionMonitorProcessor(self.valueObject)
00093
00094
          def ConstructionMonitorProcessor(self, valueObject):
00095
00096
          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
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:
          Willem van der Schans, Trelent AI
00108
00109
```

```
00110
              for callNum in range(0, self.__requestCount):
00111
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:
                      __batchNum = self.__numBatches
00121
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
                      response = requests.post(url=self.__restDomain,
00134
                                                headers=self._headerDict,
00135
00136
                                                json=self.__parameterDict)
00137
00138
                      counter = 0
00139
                          response = response.json()['hits']['hits']
00140
00141
                       except KeyError as e:
00142
                           # Logging
                          print(
00143
                             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00144
      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"])
                          __df["id"] = response[counter]['_id']
00151
00152
                            _df["county"] = response[counter]["_source"]['county']['county_name']
00153
                           counter += 1
00154
00155
                       for i in range(counter, len(response)):
                          __tdf = pd.json_normalize(response[i]["_source"])
00156
                          __tdf["id"] = response[i]['_id']
00157
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
00167
00168
              self.dataframe =
00169
              valueObject.setValue(-999)
00170
00172 class BatchProcessorUtahRealEstate:
00173
00174
                _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
          other attributes which will help us make our API calls.
00179
00180
00181
00182
              self: Represent the instance of the class
              RestDomain: Specify the domain of the rest api
00183
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 \underline{in} the header information \underline{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:
              Willem van der Schans, Trelent AI
00193
00194
00195
              self.dataframe = None
00196
              self.__numBatches = NumBatches
              self.__parameterString = ParameterString
00197
              self.__restDomain = RestDomain
00198
              self.__headerDict = HeaderDict
00199
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:
              The function that is selected by the user
00212
00213
00214
          Doc Author:
00215
              Willem van der Schans, Trelent AI
00216
00217
              self.BatchProcessingUtahRealestateCom(self.valueObject)
00218
          {\tt def~BatchProcessingUtahRealestateCom(self,~valueObject):}\\
00219
00220
          The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it
00221
      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
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()
                      __df = pd.json_normalize(response_temp, record_path=['value'])
00246
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.ison()
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)
```

4.5 DataSupportFunctions.py

```
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.
         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:
         Willem van der Schans, Trelent AI
00016
00017 """
          listOut = list(string.split(","))
00018
00019
          return listOut
```

4.6 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
          def __init__(self, method, outputDF, AppendingPath=None):
00015
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
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
              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
              if method.lower() == "ure":
00043
                 self.primaryKey = "ListingKeyNumeric"
00044
              elif method.lower() == "cm":
00045
                  self.primaryKey = "id"
00046
              elif "realtor" in method.lower():
00047
```

```
self.primaryKey = None
00048
00049
                                self.uiFlag = False
00050
                         elif method.lower() == "cfbp":
00051
                                self.primaryKey = None
00052
                                self.uiFlag = False
00053
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
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)
                                 self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
00081
00082
00083
                         if self.uiFlag:
00084
                                 if self.appendFlag:
00085
                                       PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                                                               windowType="savedLarge")
00086
00087
00088
                                        # Logging
00089
                                       print(
                                               f" \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ \{ method \} \ API \} 
00090
          request Completed | File Appended and Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0") print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Appending
00091
           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
00093
                                       PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}",
           windowType="savedLarge")
00094
00095
                                        # Logging
00096
00097
                                                f" \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ \{ method \} \ API \} = \{ (datetime.datetime.today().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().st
           request Completed | File Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
00098
                         else:
00099
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))
```

4.7 Settings.py 129

4.7 Settings.py

```
This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 # Setting NameSpace for maintenance
00005
00006 class settings:
00007
             Version Checker
00008
                 Update accordingly using semantic versioning: https://semver.org/
00009
          settingVersion = "1.2.0"
00010
                 Update in conjunction with settingDownloadSourceLink
00011
          settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest"
00012
00013
             PopUpWrapped
00014
                  Singular Reference free to change
00015
          settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
00016
                 Update in conjunction with settingDownloadSourceLink
00017
          settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApiUtility/releases/latest'
00018
00019
          # CFBP Source
00020
                 This link downloads csv's immediately so minimal change is likely required in the source code
          settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
00021
00022
00023
          # ConstructionMonitor Source
                 Check the REST call and data parser when updating this
00024
00025
          settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"
00026
00027
          # Realtor.Com Source
00028
                 Updating This link likely requires a rewrite of the html parser
          settingRealtorLink = "https://www.realtor.com/research/data/"
00029
00030
00031
          # UtahRealEstate Source
00032
                  API links are generated with hard references so updating this link requires a large code
      rewrite
00033
          settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"
```

4.8 versionChecker.py

```
00001 #
         This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002 import requests
00003
00004 from API_Calls.Functions.DataFunc.Settings import settings
00005 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00006
00007
00008 def versionChecker():
00009
00010 The versionChecker function is used to check if the current version of the program is up-to-date.
00011 It does this by comparing the latest release on GitHub.
00012 If they are not equal, it will pop up a window telling you that there's an update available.
00013
00014 Args:
00015
00016 Returns:
         A popup window with the current version and latest version
00017
00018
00019 Doc Author:
00020
          Willem van der Schans, Trelent AI
00021 """
          # Todo Gitlab Update
00023
          current_version = settings.settingVersion
00024
          # Todo Gitlab Update
00025
          response = requests.get(settings.settingGithubApiUrl)
00026
          latest_version = response.json()['name']
00027
          text_string = f"A different version is tagged as latest release \n \n"
                        f"Running version: {current_version}\n" \
00028
00029
                        f"Latest version: {latest_version}'
00030
         print(text_string)
00031
00032
          if current_version != latest_version:
00033
              PopupWrapped(text_string, windowType="versionWindow")
```

4.9 ErrorPopup.py

```
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
00007 def ErrorPopup(textString):
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"
              f"Log folder will be opened upon program exit",
00024
              windowType="FatalErrorLarge")
00025
```

4.10 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):
80000
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:
         The response text
00017
00018
00020 Willem van der Schans, Trelent AI 00021 """
00019 Doc Author:
00022
        if isinstance(response, int):
00023
             print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response:
     {response}")
00024
             response_txt = response.text
00025
             print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Resource Response:
      {response_txt}")
```

4.11 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
```

4.12 RESTError.py 131

```
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:
          A file path to the log file that was created
00019
00020
00021 Doc Author:
          Willem van der Schans, Trelent AI
00023 """
          dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
00025
          if os.path.exists(dir path):
00026
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
          filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
00034
00035
              f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
          sys.stdout = open(filePath, 'w')
00036
          sys.stderr = sys.stdin = sys.stdout
00037
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
00044
              path: Specify the directory to be sorted
00045
00046
          Returns:
              A list of files in a directory sorted by modification time
00047
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))
               print (f" \{ aatetime.datetime.today().strftime('%m-%d-%Y %H: %M: %S. %f')[:-3] \} \ | \ Log \ file \ \{ file \} \} 
00058
     deleted")
```

4.12 RESTError.py

```
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
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)
                                                raise ValueError(textString)
00039
00040
                                   elif status code == 400:
00041
                                                RESTErrorPrint (response)
                                                \texttt{textString} = \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:\%M:\%S.\%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()} \{ \texttt{datetime.datetime.today().strftime()} \} 
00042
                     {status_code} | Bad Request; check input arguments"
00043
                                                ErrorPopup(textString)
00044
                                                raise ValueError(textString)
00045
                                   elif status_code == 401:
                                                RESTErrorPrint (response)
00046
                                                \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.\%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()}(\texttt{datetime.datetime.today().strftime()}(\texttt{Status Code}) \mid \texttt{Status Code}) \mid \texttt{Status Code} = \texttt{f()}(\texttt{Status Code}) \mid \texttt{Status Code} \mid \texttt{St
00047
                    {status_code} | Authentication Error: No keys found"
00048
                                                ErrorPopup (textString)
00049
                                                raise PermissionError(textString)
                                   elif status_code == 402:
00050
00051
                                                RESTErrorPrint (response)
                                                \texttt{textString} = \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ | \ \texttt{Status Code} = \texttt{f()} \{ \texttt{datetime.datetime.today().strftime()} \} \} 
00052
                     {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
                                                \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()}(\texttt{Status Code}) = \texttt{f()}(\texttt{St
                    {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
                                                {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)
08000
                                   elif status code == 701:
00081
                                                RESTErrorPrint (response)
                                                \texttt{textString} = \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()} \{ \texttt{datetime.datetime.today().strftime()} \} 
00082
                     {status_code} | Error in coercing icon to bits (Imageloader.py)'
00083
                                                ErrorPopup(textString)
                                                raise TypeError(textString)
00084
00085
                                   elif status_code == 801:
00086
                                                RESTErrorPrint (response)
                                                \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()}(\texttt{datetime.datetime.today().strftime()}(\texttt{Status Code}) \mid \texttt{Status Code}) \mid \texttt{Status Code} = \texttt{f()}(\texttt{Status Code}) \mid \texttt{Status Code} \mid \texttt{St
00087
                     {status_code} | Resource Error, HTML cannot be parsed the website's HTML source might be changed"
```

4.13 BatchGui.py 133

```
00088
                                                             ErrorPopup(textString)
00089
                                                             raise ValueError(textString)
00090
                                            elif status_code == 790:
00091
                                                             RESTErrorPrint (response)
                                                             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
00092
                          {status code} | Requests timeout within requests"
00093
                                                             ErrorPopup(textString)
00094
                                                              raise TimeoutError(textString)
                                            elif status_code == 791:
00095
00096
                                                             RESTErrorPrint (response)
                                                             \texttt{textString} = \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \; | \; \; \texttt{Status Code} = \texttt{f()} \{ \texttt{Status Code} = \texttt{f()} \{ \texttt{Status Code} = \texttt{f()} \} \} 
00097
                          {status_code} | Too many redirects, Bad url"
00098
                                                             ErrorPopup(textString)
00099
                                                             raise ValueError(textString)
00100
                                            elif status_code == 990:
                                                             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:
                                                             RESTErrorPrint (response)
00106
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)
                                                            raise ValueError(textString)
00109
                                            elif status_code == 992:
00110
00111
                                                             RESTErrorPrint (response)
                                                             \texttt{textString} = \texttt{f"} \{ \texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \ | \ \texttt{Status Code} = \texttt{f()} \{ \texttt{datetime.datetime.today().strftime()} \} \} 
00112
                          {status_code} | No authentication input (Basic or User/PW)'
00113
                                                             ErrorPopup (textString)
00114
                                                             raise ValueError(textString)
00115
                                            elif status_code == 993:
00116
                                                             RESTErrorPrint (response)
                                                             \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today()}.\texttt{strftime('%m-%d-%Y %H:$M:$S.$f')[:-3]} \mid \texttt{Status Code} = \texttt{f()}(\texttt{formal}) \mid \texttt{f()}(\texttt{form
00117
                          {status_code} | Submission Error: input values could not be coerced to arguments"
00118
                                                             ErrorPopup(textString)
00119
                                                             print(ValueError(textString))
                                            elif status_code == 994:
00121
                                                             RESTErrorPrint (response)
                                                             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
00122
                          {status_code} | Submission Error: server returned no documents"
00123
                                                            ErrorPopup(textString)
00124
                                                             raise ValueError(textString)
00125
                                            elif status_code == 1000:
00126
                                                             RESTErrorPrint (response)
00127
                                                             \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} \mid \texttt{Status Code} = \texttt{f()}(\texttt{Status Code}) = \texttt{f()}(\texttt{St
                          {status_code} | Catastrophic Error"
00128
                                                             ErrorPopup(textString)
00129
                                                             raise SystemError(textString)
00130
                                            elif status_code == 1001:
00131
                                                             RESTErrorPrint (response)
                                                             \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today()}.\texttt{strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]}\} \mid \texttt{Status Code} = \texttt{f()}(\texttt{datetime.datetime.today()}.\texttt{strftime()}(\texttt{datetime.datetime.datetime.today()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.today()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.da
00132
                          {status_code} | Main Function Error Break"
00133
                                                            raise SystemError(textString)
00134
                                            elif status_code == 1100:
00135
                                                             RESTErrorPrint (response)
                                                             textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
00136
                          {status_code} | User has cancelled the program execution"
00137
                                                             raise KeyboardInterrupt(textString)
00138
                                            elif status_code == 1101:
00139
                                                             RESTErrorPrint (response)
00140
                                                             \texttt{textString} = \texttt{f"}\{\texttt{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]}\} \mid \texttt{Status Code} = \texttt{f()}\{\texttt{datetime.datetime.datetime.today().strftime()} + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime()}) + \texttt{f()}(\texttt{datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.datetime.da
                          {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")
```

4.13 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 import PySimpleGUI as sg
```

```
00005 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00006
00007
00008 def BatchInputGui(batches, documentCount=None):
00009
00010 The BatchInputGui function is a simple GUI that displays the number of batches and pages
00011 that will be requested. It also gives the user an option to cancel or continue with their request.
00012
00013
00014 Args:
00015
          batches: Determine how many batches will be run
00016
          documentCount: Determine how many documents will be retrieved
00017
00018 Returns:
00019
          The event, which is the button that was pressed
00020
00021 Doc Author:
00022
          Willem van der Schans, Trelent AI
00023 """
00024
          event = None
          __text1 = f"This request will run {batches}"
else:
00025
00026
00027
00028
                _text1 = f"This request will run {batches} batches and will retrieve {documentCount} rows"
00029
00030
           _text2 = "Press Continue to start request"
00031
00032
          _{\rm Line1} = [sg.Push(),
00033
                     sg.Text(__text1, justification="center"),
00034
                     sq.Push()]
00035
00036
          \_Line2 = [sg.Push(),
00037
                     sg.Text(__text2, justification="center"),
00038
                     sg.Push()]
00039
          \_Line3 = [sg.Push(),
00040
                     sg.Ok("Continue"),
00041
00042
                     sg.Cancel(),
00043
                     sg.Push()]
00044
          window = sg.Window("Popup", [__Line1, __Line2, __Line3],
00045
00046
                              modal=True,
00047
                              keep_on_top=True,
00048
                              disable close=True,
00049
                              icon=ImageLoader("taskbar_icon.ico"))
00050
00051
          while True:
00052
              event, values = window.read()
00053
              if event == "Continue":
00054
00055
              elif event == sg.WIN_CLOSED or event == "Cancel":
00056
00057
00058
          window.close()
00059
00060
          return event
00061
00062
00063 def confirmDialog():
00064
00065 The confirmDialog function is a simple confirmation dialog that asks the user if they want to continue
      with the request.
00066 The function takes no arguments and returns the button event to allow for process confirmation.
00067
00068 Args:
00069
00070 Returns:
00071
          The event that was triggered,
00072
00073 Doc Author:
00074
          Willem van der Schans, Trelent AI
00075 """
00076
          event = None
          __text1 = f"This request can take multiple minutes to complete"
00077
          __text2 = "Press Continue to start the request"
00078
00079
           _{\rm Line1} = [sg.Push(),
08000
00081
                     sg.Text(__text1, justification="center"),
00082
                     sq.Push()]
00083
00084
          _{\rm Line2} = [sg.Push(),
```

```
00085
                      sg.Text(__text2, justification="center"),
00086
                      sq.Push()]
00087
00088
          \__Line3 = [sg.Push(),
00089
                      sg.Ok("Continue"),
00090
                      sg.Cancel(),
00091
                     sg.Push()]
00092
00093
          window = sg.Window("Popup", [__Line1, __Line2, __Line3],
                              modal=True,
00094
00095
                              keep_on_top=True,
00096
                              disable_close=True,
00097
                              icon=ImageLoader("taskbar_icon.ico"))
00098
00099
          while True:
             event, values = window.read()
00101
              if event == "Continue":
00102
00103
              elif event == sq.WIN_CLOSED or event == "Cancel":
00104
00105
00106
          window.close()
00107
00108
          return event
```

4.14 BatchProgressGUI.py

```
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 \text{ counter} = 1
00015
00016
00017 class BatchProgressGUI:
00018
00019
          def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):
00020
00021
00022
                      _ 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
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
              self._headerDict = HeaderDict
00043
              self.__columnSelection = ColumnSelection
00044
00045
              self.__type = Type
00046
              self.dataframe = None
00048
              self. lavout = 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
00058
              self: Represent the instance of the class
00059
00060
          Returns:
00061
              The __type of the batchgui class
00062
00063
          Doc Author:
          Willem van der Schans, Trelent AI
00064
00065
00066
              self.CreateProgressLayout()
00067
              self.createGui(self.__type)
00068
00069
          def CreateProgressLavout(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
00076
                  self (object): The object that is calling this function.
00077
00078
00079
              self: Access the class variables and methods
00080
00081
          Returns:
              A list of lists
00082
00083
00084
          Doc Author:
              Willem van der Schans, Trelent AI
00085
00086
00087
              sg.theme('Default1')
00088
00089
                _Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center",
      key="--progress_text--"),
00090
                          sq.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
              __Line3 = [
00095
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 \underline{i}n 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
          the program to run, including: window, start_time, update_text, valueObj (DataTransfer), processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
00110
00111
          and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts
00112
     these threads.
00113
00114
00115
              self: Access the object itself
              Sourcetype: Determine which batch processor to use
00116
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)
                      self.__window['--time_est--'].update("Est time needed 00:00:00")
00130
00131
00132
                      valueObj = DataTransfer()
00133
                      valueObj.setValue(0)
00134
00135
                       if Sourcetype == "construction_monitor":
00136
                             processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00137
                                                                                                                  NumBatches=self.__batches,
00138
00139
                                                                                                                  ParameterDict=self.__parameterDict,
00140
                                                                                                                  HeaderDict=self._headerDict,
00141
                                                                                                                  ColumnSelection=self.__columnSelection,
00142
                                                                                                                  valueObject=valueObj)
00143
                      elif Sourcetype == "utah_real_estate":
                             processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00144
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
         Successfully Started")
00154
00155
                      batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
00156
                                                                                daemon=False)
00157
                      batchFuncThread.start()
                      \label{lem:main} print(f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ BatchFunc\ Thread().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().strftime().st
00158
         Successfully Started")
00159
                      threading. Thread (target=self. ValueChecker,
00160
                                                  args=(valueObj,),
00161
                                                  daemon=False).start()
                      00162
         Successfully Started")
00163
00164
                      while True:
00165
00166
                             self.ProgressUpdater(valueObj)
00167
00168
                             if valueObj.getValue() == -999:
00169
00170
                             window, event, values = sg.read_all_windows()
00171
00172
                             if event.startswith('update'):
                                      _key_to_update = event[len('update'):]
00173
00174
                                    window[__key_to_update].update(values[event])
00175
                                    window.refresh()
00176
00177
00178
                             if event == sq.WIN_CLOSED or event == "Cancel" or event == "Exit":
00179
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
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:
          Willem van der Schans, Trelent AI
00204
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)
                  self.__window.write_event_value('update--progress_text--', __update_text)
00212
00213
              else:
00214
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
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
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:</pre>
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
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
00249
                      __passed_time = __passed_time.total_seconds()
00250
00251
00252
                          __time_est = datetime.timedelta(
00253
                             seconds=(__passed_time * (self.__batches / self.__batch_counter) -
00254
      __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
```

4.15 DataTransfer.py 139

```
00277
00278
00279
              True if the value of the object has changed, and false if it hasn't
00280
00281
         Doc Author:
          Willem van der Schans, Trelent AI
00282
00283
00284
              while True:
00285
                  time.sleep(0.3)
                  if self.__batch_counter != ObjectVal.getValue():
                     self.__batch_counter = ObjectVal.getValue()
00288
                      return True
00289
                  else:
                      return False
```

4.15 DataTransfer.py

```
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
          The __init__ function is called when the class is instantiated. It sets the initial value of self.__value to 0.
00008
00009
00010
00011
          Args:
00012
              self: Represent the instance of the class
00013
00014
          Returns:
00015
              Nothing
00016
00017
          Doc Author:
          Willem van der Schans, Trelent AI
00018
00019
00020
              self.\_value = 0
00021
          def setValue(self, value):
00022
00023
          The setValue function sets the value of the object.
00024
00025
00026
00027
00028
              self: Represent the instance of the class
00029
              value: Set the value of the instance variable __value
00030
00031
00032
              The value that was passed to it
00033
00034
          Doc Author:
          Willem van der Schans, Trelent AI
00035
00036
00037
              self.__value = value
00038
          def getValue(self):
00039
00040
00041
          The getValue function returns the value of the private variable _
00042
          This is a getter function that allows access to this private variable.
00043
00044
00045
              self: Represent the instance of the class
00046
00047
          Returns:
00048
              The value of the instance variable
00049
00050
          Doc Author:
          Willem van der Schans, Trelent AI
00051
00052
00053
              return self.__value
00054
          def whileValue(self):
00055
00056
00057
          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.
00059
```

```
00060
         Args:
00061
             self: Refer to the current instance of the class
00062
          Returns:
00063
00064
              The value of the input
00065
00066
         Doc Author:
          Willem van der Schans, Trelent AI
00067
00068
00069
              while True:
00070
                  self.getValue()
```

4.16 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
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.
00017 Args:
00018
          file: Specify the image file to be loaded
00019
00020 Returns:
00021
          A base64 encoded image string
00023 Doc Author:
00024
          Willem van der Schans, Trelent AI
00025 """
00026
              __path = normpath(join(str(os.getcwd().split("API_Calls", 1)[0]), "API_Calls"))
00027
              __path = normpath(join(__path, "External Files"))
__path = normpath(join(__path, "Images"))
__path = join(__path, file).replace("\\", "/")
00028
00029
00030
00031
00032
              image = Image.open(__path)
00033
00034
               __buff = BytesIO()
00035
00036
               image.save(__buff, format="png")
00037
00038
               img_str = base64.b64encode(__buff.getvalue())
00039
00040
               return img_str
00041
           except Exception as e:
               # We cannot log this error like other errors due to circular imports
00042
00043
               raise e
```

4.17 PopupWrapped.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/
00002 import datetime
00003 import os
00004 import threading
00005 import time
00006 import webbrowser
00007 from pathlib import Path
00008
00009 import PySimpleGUI as sg
00010
00011 from API_Calls.Functions.DataFunc.Settings import settings
```

```
00012 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00013
00014
00015 class PopupWrapped():
00016
          def __init__(self, text="", windowType="notice", error=None):
00017
00018
00019
          The __init__ function is the first function that gets called when an object of this class is created.
00020
          It sets up all the variables and creates a window for us to use.
00021
          Args:
00022
               self: Represent the instance of the class
00023
               text: Set the text of the window
00024
               windowType: Determine what type of window to create
               error: Display the error message in the window
00026
          Returns:
00027
             Nothing
00028
          Doc Author:
00029
               Willem van der Schans, Trelent AI
00030
               self.__text = text
00031
               self.__type = windowType
00032
               self.__error = error
00033
               self.__layout = []
00034
               self.__windowObj = None
00035
00036
               self.__thread = None
               self.\_counter = 0
00037
               self.__docpath = None
00038
               self.__errorFlag = False
00039
00040
00041
                   if "File Appended and Saved to " in self.__text:
00042
00043
                       self.__docpath = str(self.__text[27:])
                   elif "File Saved to " in self.__text:
00044
                      self.__docpath = str(self.__text[14:])
00045
00046
                   else:
00047
               except Exception as e:
    if self.__type == "savedLarge":
00048
00049
                       print(
                          f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py
00051
      | Error = {e} | Error creating self.__docpath open file button not available")
00052
                       self.__errorFlag = True
00053
                   else:
                       pass
00054
00055
00056
               self.__createWindow()
00057
00058
                _createLayout(self):
00059
00060
                __createLayout function is used to create the layout of the window.
00061
          The function takes class variables and returns a window layout.
00062
          It uses a series of if statements to determine what type of window it is, then creates a layout based
     on that information.
00063
00064
              self: Refer to the current instance of a class
00065
          Returns:
00066
               A list of lists
00067
          Doc Author:
               Willem van der Schans, Trelent AI
00068
00069
00070
               sg.theme('Default1')
               __Line1 = None
00071
               __Line2 = None
00072
00073
00074
               if self.__type == "notice":
00075
                  \_Line1 = [sg.Push(),
00076
                               sg.Text(u')u2713', font=("Helvetica", 20, "bold"), justification="center"),
00077
                               sq.Text(self.__text, justification="center", key="-textField-"), sq.Push()]
00078
                    _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00079
               elif self.__type == "noticeLarge":
                  __Line1 = [sg.Push(),
00080
                               sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"), sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00081
00082
00083
                    _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00084
               elif self.__type == "savedLarge":
00085
                  if self.__errorFlag:
00086
                       \_Line1 = [sg.Push(),
                                   sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00087
00088
00089
                        __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00090
```

```
00091
                        \_Line1 = [sq.Push(),
                                    sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00092
00093
00094
                        __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10,
      1)), sg.Push()]
               elif self.__type == "errorLarge":
00095
00096
                   \_Line1 = [sg.Push(),
00097
                               sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                                sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00098
                    __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00099
               elif self.__type == "FatalErrorLarge":
00100
00101
                   \_Line1 = [sg.Push(),
                                sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
00102
00104
                     _Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
               elif self.__type == "error":
                   __Line1 = [sg.Push(),
00106
                                sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(f"{self.__text}: {self.__error}", justification="center",
00107
00108
      key="-textField-"),
00109
                                sq.Push()]
00110
                     __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
               elif self.__type == "AuthError":
00111
00112
                   __Line1 = [sg.Push(),
                                sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
sg.Text(f"{self.__text}", justification="center", key="-textField-"),
00113
00114
00115
                                sa.Push()1
                    __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
00116
                                sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
00117
               elif self.__type == "versionWindow":
00118
00119
                   \__Line1 = [sg.Push(),
                                sg.Text(f"{self.__text}", justification="left", key="-textField-"),
00120
00121
                                sg.Push()]
00122
                    __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00123
                                sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
00124
               elif self.__type == "progress":
                   \_Line1 = [sg.Push(),
00125
                                sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00126
00127
               if self.__type == "progress":
00128
                   self.__layout = [__Line1, ]
00129
               else:
00130
00131
                    self.__layout = [__Line1, __Line2]
00132
00133
           def __createWindow(self):
00134
00135
                 _createWindow function {	ext{is}} used to create the window object that will be displayed.
00136
           The function takes class variables and a window object. The function first calls __createLayout, which
      creates the layout for the window based on what type of message it is (error, notice, progress). Then it
       uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as
       title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits
       until either 100 iterations have passed or an event has been triggered (such as clicking "Ok" or
      closing the window). Once one of these events occurs
00137
00138
               self: Reference the instance of the class
00139
           Returns:
00140
               A window object
00141
           Doc Author:
               Willem van der Schans, Trelent AI
00142
00143
00144
               self.__createLayout()
00145
00146
                if self.__type == "progress":
00147
                   self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
      finalize=True,
00148
                                                    modal=True,
00149
                                                    keep_on_top=True,
00150
                                                    disable_close=False,
00151
                                                    icon=ImageLoader("taskbar_icon.ico"),
00152
                                                    size=(290, 50))
00153
               elif self.__type == "noticeLarge":
00154
                    self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00155
                                                   modal=True,
00156
                                                    keep on top=True,
00157
                                                    disable_close=False,
00158
                                                    icon=ImageLoader("taskbar icon.ico"))
               elif self.__type == "savedLarge":
00159
                    self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00160
00161
                                                    modal=True,
                                                    keep_on_top=False,
00162
00163
                                                    disable close=False.
```

```
00164
                                                icon=ImageLoader("taskbar_icon.ico"))
00165
              elif self.__type == "errorLarge":
00166
                  self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
00167
                                                modal=True,
00168
                                                 keep_on_top=True,
00169
                                                 disable_close=False,
00170
                                                 icon=ImageLoader("taskbar_icon.ico"))
00171
              elif self.__type == "FatalErrorLarge":
00172
                  self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                                modal=True,
00173
00174
                                                 keep_on_top=True,
00175
                                                disable_close=False,
00176
                                                icon=ImageLoader("taskbar_icon.ico"))
              elif self.__type == "AuthError":
00177
00178
                  self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout,
     finalize=True,
00179
                                                modal=True.
                                                keep_on_top=True,
00180
00181
                                                 disable_close=False,
00182
                                                icon=ImageLoader("taskbar_icon.ico"))
              elif self.__type == "versionWindow":
00183
00184
                  self.__windowObj = sq.Window(title="Update Notice", layout=self.__layout, finalize=True,
00185
                                                modal=True,
00186
                                                keep on top=True,
00187
                                                disable_close=False,
00188
                                                icon=ImageLoader("taskbar icon.ico"))
00189
              else:
                 self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
00190
      finalize=True,
00191
                                                modal=True.
00192
                                                 keep_on_top=True,
00193
                                                disable_close=False,
00194
                                                icon=ImageLoader("taskbar_icon.ico"),
00195
                                                size=(290, 80))
00196
00197
              if self.__type != "progress" or self.__type.startswith("perm"):
00198
                  timer = 0
00199
                  while timer < 100:
                      event, values = self.__windowObj.read()
if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
00200
00201
00202
                          break
00203
                      elif event == "Open Generation Tool [Web Browser]":
00204
                           webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
00205
00206
                      elif event == "Open File":
00207
                           threadFile = threading.Thread(target=self.openFile,
00208
                                                          daemon=False)
00209
                           threadFile.start()
00210
                           time.sleep(3)
00211
00212
                      elif event == "Download":
00213
                           # Todo Gitlab Update
00214
                           webbrowser.open(settings.settingDownloadSourceLink, new=2,
00215
                                           autoraise=True)
00216
00217
                      time.sleep(0.1)
00218
00219
                  if self.__type == "FatalErrorLarge":
00220
                      try:
                          os.system(
00221
00222
                              f"start
      {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
00223
                      except Exception as e:
00224
                          print(
                              f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00225
      PopupWrapped.py | Error = {e} | Log Folder not found please search manually for
      %APPDATA%\Roaming\GardnerUtil\Logs\n")
00226
00227
                  self.__windowObj.close()
00228
00229
          def stopWindow(self):
00230
00231
          The stopWindow function is used to close the window object that was created in the startWindow
      function.
00232
          This is done by calling the close() method on self._windowObj, which will cause it to be destroyed.
00233
          Args:
00234
              self: Represent the instance of the class
00235
          Returns:
00236
             The window object
00237
          Doc Author:
00238
              Willem van der Schans, Trelent AI
```

```
00239
00240
              self.__windowObj.close()
00241
00242
          def textUpdate(self, sleep=0.5):
00243
00244
          The textUpdate function is a function that updates the text in the text field.
00245
          It does this by adding dots to the end of it, and then removing them. This creates
00246
          a loading effect for when something is being processed.
00247
00248
              self: Refer to the object itself
00249
              sleep: Control the speed of the text update
00250
00251
              A string that is the current text of the text field
          Doc Author:
          Willem van der Schans, Trelent AI
00253
00254
00255
              self.__counter += 1
              if self.__counter == 4:
    self.__counter = 1
00256
00257
              newString = ""
00258
00259
              if self.__type == "notice":
00260
00261
              elif self.__type == "error":
00262
00263
              elif self.__type == "progress":
                  newString = f"{self.__text}{'.' * self.__counter}"
00264
00265
              self.__windowObj.write_event_value('update-textField-', newString)
00266
00267
              time.sleep(sleep)
00268
00269
          def windowPush (self):
00270
00271
          The windowPush function is used to update the values of a window object.
00272
00273
              The function takes in an event and values from the window object, then checks if the event starts
     with 'update'.
00274
              If it does, it will take everything after 'update' as a key for updating that specific value.
00275
              It will then update that value using its key and refresh the window.
00276
00277
              self: Reference the object that is calling the function
00278
          Returns:
00279
             A tuple containing the event and values
00280
          Doc Author:
00281
              Willem van der Schans, Trelent AI
00282
00283
              event, values = self.__windowObj.read()
00284
00285
              if event.startswith('update'):
00286
                  __key_to_update = event[len('update'):]
00287
                  self.__windowObj[__key_to_update].update(values[event])
00288
                  self.__windowObj.refresh()
00289
00290
          def openFile(self):
00291
00292
          The openFile function opens the file that is associated with the
00293
              document object. It does this by calling os.system and passing it
00294
              self.__docpath as an argument.
00295
00296
00297
             self: Represent the instance of the object itself
00298
00299
00300
              The filepath of the document
00301
00302
          Doc Author:
          Willem van der Schans, Trelent AI
00303
00304
00305
              os.system(self.__docpath)
```

4.18 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
```

4.18 Initializer.py 145

```
00006 from pathlib import Path
00008 import PySimpleGUI as sg
00009
00010 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00011 from API_Calls.Functions.DataFunc.versionChecker import versionChecker
00012 from API_Calls.Functions.ErrorFunc.Logger import logger
00013 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00014 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00015 from API_Calls.Sources.CFBP.Core import CFBP
00016 from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
         ConstructionMonitorMain
00018 from API_Calls.Sources.Realtor.Core import realtorCom
00019 from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain, UtahRealEstateInit
00020
00021
00022 class initializer:
00023
00024
         def __init__(self):
00025
00026
00027
         The __init__ function is called when the class is instantiated.
00028
         It sets up the logging, calls the __ShowGui function to create and display
00029
         the GUI, and then calls __CreateFrame to create a frame for displaying widgets.
00030
00031
00032
         Aras:
00033
             self: Represent the instance of the class
00034
00035
         Returns:
00036
             Nothing
00037
00038
         Doc Author:
00039
             Willem van der Schans, Trelent AI
00040
00041
             self.classObj = None
00042
00043
             logger()
00044
00045
             00046
00047
             self.__ShowGui(self.__CreateFrame(), "Data Tool")
00048
             print("\n\n----\n\n")
00049
00050
00051
         def __ShowGui(self, layout, text):
00052
00053
00054
              _ShowGui function is the main function that displays the GUI.
00055
         It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three
00056
             1) The type of element to be displayed (e.g., "Text", "InputText", etc.)
00057
             2) A dictionary containing any additional parameters for that element (e.g., size, default value,
00058
             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
00059
00060
         Args:
00061
             self: Represent the instance of the class
00062
             layout: Pass the layout of the window to be created
00063
             text: Set the title of the window
00064
00065
         Returns:
00066
             A window object
00067
00068
         Doc Author:
00069
             Willem van der Schans, Trelent AI
00070
00071
             # Todo Gitlab Update
00072
             versionChecker()
00073
00074
             window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00075
                                finalize=True,
00076
                                icon=ImageLoader("taskbar_icon.ico"))
00077
00078
             while True:
00079
                 event, values = window.read()
00080
00081
                 if event == "Construction Monitor":
00082
                     print(
00083
                         f"\n{datetime.datetime.todav().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
```

```
----Initiating Construction Monitor API Call-----")
00084
                    ConstructionMonitorMain(ConstructionMonitorInit())
00085
00086
                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
                 -Closing Construction Monitor API Call-----\n")
00087
                 elif event == "Utah Real Estate":
00088
                   print(
                       f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00089
                 -Initiating Utah Real Estate API Call-----")
00090
                    UtahRealEstateMain(UtahRealEstateInit())
00091
                    print(
                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
                 -Closing Utah Real Estate API Call-----\n")
                elif event == "Realtor.Com":
00093
00094
                    print(
                       f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
                 -Initiating Realtor.com API Call----")
                  realtorCom()
00097
                    print(
                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00098
                 --Closing Realtor.com API Call-----\n")
                elif event == "CFPB Mortgage":
00100
                   print(
00101
                       f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
                --Initiating ffiec.cfpb API Call----")
00102
                    CFBP()
00103
                    print(
                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00104
                ---Closing ffiec.cfpb API Call------\n")
elif event == "Authorization Utility":
00105
00106
                   print(
                       f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00107
               ---Initiating Authorization Utility----")
00108
                    AuthUtil()
                    print(
00109
                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00110
             ----Closing Authorization Utility-----\n")
elif event == "Open Data Folder":
00111
                   00112
00113
            -----Data Folder Opened-----")
00114
                    try:
                        os.system(f"start
00115
     {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
00116
                   except:
00117
00118
                            os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                        except Exception as e:
00119
                           print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00120
     Initializer.py | Error = {e} | Documents folder not found")
00121
                            PopupWrapped(
00122
                                text="Documents folder not found. Please create a Windows recognized documents
     folder",
00123
                                windowType="errorLarge")
00124
00125
                 elif event in ('Exit', None):
00126
                    try:
00127
00128
                     except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00129
    Initializer.py | Error = {e} | Error on program exit, for logging purposes only.")
00130
00131
                 elif event == sg.WIN_CLOSED or event == "Quit":
00132
00133
00134
             window.close()
00135
00136
         def ___CreateFrame(self):
00137
00138
00139
         The __CreateFrame function is a helper function that creates the layout for the main window.
00140
         It returns a list of lists, which is then passed to sq.Window() as its layout parameter.
00141
00142
         Args:
00143
             self: Represent the instance of the class
00144
00145
         Returns:
             A list of lists, which is then passed to the sq
00146
00147
00148
         Doc Author:
00149
             Willem van der Schans, Trelent AI
```

4.19 CFBP/Core.py 147

```
00150
00151
            sg.theme('Default1')
00152
            line0 = [sg.HSeparator()]
00153
00154
00155
            line1 = [sg.Image(ImageLoader("logo.png")),
00156
                    sg.Push(),
00157
                    sq.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
00158
                    sg.Push(),
00159
                    sg.Push()]
00160
00161
            line3 = [sg.HSeparator()]
00162
00163
            line4 = [sg.Push(),
00164
                    sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
00165
                    sg.Push()]
00166
00167
            line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                     sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00168
00169
            00170
00171
00172
                     sq.Push()]]
00173
00174
            line8 = [sq.HSeparator()]
00175
00176
            line9 = [sq.Push(),
00177
                    sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
00178
                    sq.Push()]
00179
            00180
00181
00182
            line11 = [sg.HSeparator()]
00183
00184
            layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]
00185
00186
00187
            return lavout
```

4.19 CFBP/Core.py

```
00001 import datetime
00002 import threading
00003 import time
00004
00005 import pandas as pd
00006 import requests
00007
00008 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00009 from API_Calls.Functions.DataFunc.Settings import settings
00010 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00011 from API_Calls.Functions.Gui.BatchGui import confirmDialog
00012 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00013
00014
00015 class CFBP:
00016
00017
          def __init__(self, state_arg=None, year_arg=None):
00018
00019
          The \_init\_ function is called when the class is instantiated.
00020
          Its job is to initialize the object with some default values, and do any other setup that might be
     necessary.
00021
          The __init__ function can take arguments, but it doesn't have to.
00022
00023
          Args:
00024
              self: Represent the instance of the class
00025
              state_arg: Set the state_arg attribute of the class
00026
              year_arg: Set the year of data to be retrieved
00027
00028
          Returns:
00029
              A popupwrapped object
00030
00031
          Doc Author:
00032
              Willem van der Schans, Trelent AI
00033
00034
              self.state_arg = state_arg
00035
              self.year_arg = year_arg
```

```
00036
                                      self.uiString = None
00037
                                      self.link = None
00038
00039
                                      eventReturn = confirmDialog()
                                      if eventReturn == "Continue":
00040
00041
                                                 startTime = datetime.datetime.now().replace(microsecond=0)
00042
                                                 self.__showUi()
00043
                                                            f'' \{ datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ API \ Link = (API \ Link) + (A
00044
                {self.link}")
00045
                                                F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
00046
                                                print(
00047
                                                            f'' \{ date time.date time.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3] \} \ | \ Data \ retrieved \ with the strong property of the str
                in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0)
                startTime).total_seconds()))}")
00048
00049
                                                self.uiString = (
00050
                                                           f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg} data retrieved \n
               Data Saved at {F.getPath()}")
00051
00052
                                                PopupWrapped(text=self.uiString, windowType="noticeLarge")
00053
                                      else:
00054
                                              print(
00055
                                                          f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
                Request")
00056
00057
00058
                          def __showUi(self):
00059
00060
                           The _showUi function is a function that creates a progress bar window.
00061
00062
                           The __showUi function takes class variables and returns a windowobj.
00063
00064
00065
                          Args:
00066
                                     self: Represent the instance of the class
00067
00068
                           Returns:
00069
                                     The uiobj variable
00070
00071
                           Doc Author:
00072
                                      Willem van der Schans, Trelent AI
00073
00074
                                      uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)
00075
00076
                                      threadGui = threading.Thread(target=self.__dataGetter,
00077
                                                                                                                     daemon=False)
00078
                                      threadGui.start()
00079
00080
                                      while threadGui.is_alive():
00081
                                                uiObj.textUpdate()
00082
                                                uiObj.windowPush()
00083
                                      else:
00084
                                                 uiObj.stopWindow()
00085
00086
                          def __dataGetter(self):
00087
00088
                           The __dataGetter function is a private function that gets the data from the CFPB API.
00089
                           It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.
00090
00091
00092
                                      self: Represent the instance of the class
00093
00094
                          Returns:
00095
                                    A response object
00096
00097
                           Doc Author:
                           Willem van der Schans, Trelent AI
00098
00099
00100
                                      arg_dict_bu = locals()
00101
00102
                                      link = settings.settingCFBPLink
00103
00104
                                      if self.state arg is None:
                                               self.state_arg = "UT"
00105
00106
                                      else:
00107
00108
00109
                                      if self.year_arg is None:
                                               self.year_arg = str(datetime.date.today().year - 1)
00110
00111
```

```
00112
00113
00114
              passFlag = False
00115
00116
              while not passFlag:
00117
00118
                  self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"
00119
00120
                  response = requests.get(self.link)
00121
                  if response.status_code == 400:
00122
00123
                      self.year_arg = int(self.year_arg) - 1
00124
00126
                      passFlag = True
00128
              RESTError (response)
00129
              raise SystemExit(0)
```

4.20 ConstructionMonitor/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
{\tt 00015~from~API\_Calls.Functions.DataFunc.BatchProcessing~import~BatchCalculator}
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.DataFunc.Settings import settings
00018 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00019 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00020 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00021 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00022 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00023
00024
00025 class ConstructionMonitorInit:
00026
00027
          def __init__(self):
00028
00029
00030
          The __init__ function is called when the class is instantiated.
00031
          It sets up the variables that will be used by other functions in this class.
00032
00033
00034
00035
             self: Represent the instance of the class
00036
00037
          Returns:
00038
              None
00039
00040
          Willem van der Schans, Trelent AI
00041
00042
00043
              self.size = None
00044
              self.SourceInclude = None
00045
              self.dateStart = None
00046
              self.dateEnd = None
00047
              self.rest_domain = None
00048
              self.auth_key = None
00049
              self.ui_flag = None
00050
              self.append_file = None
00051
00052
              passFlag = False
00053
00054
              while not passFlag:
                  if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00056
                          "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
```

```
00057
                      Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00058
                          "Security").joinpath("auth.json")):
00059
00060
                          f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00061
                              "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00062
                          key = f.readline()
00063
                          f.close()
00064
                          f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00065
                              "Security").joinpath("auth.json"), "rb")
00066
                          authDict = json.load(f)
00067
                          fernet = Fernet(key)
00068
                          self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
00069
                          passFlag = True
00070
                      except Exception as e:
00071
                          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")
00072
                          AuthUtil()
00073
                  else:
00074
                      AuthUtil()
00075
00076
              self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00077
00078
          def ShowGui(self, layout, text):
00079
00080
00081
          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.
00082
          The text parameter specifies what title should appear at the top of the window.
00083
00084
00085
00086
              self: Refer to the current instance of a class
00087
              layout: Determine what the gui will look like
00088
              text: Set the title of the window
00089
00090
          Returns:
00091
             A dictionary of values
00092
00093
          Doc Author:
          Willem van der Schans, Trelent AI
00094
00095
00096
              window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00097
                                 finalize=True,
                                 icon=ImageLoader("taskbar_icon.ico"))
00098
00099
00100
              while True:
                  event, values = window.read()
00101
00102
                  if event == "Submit":
00103
00104
                      try:
00105
                          self.__SetValues(values)
00106
00107
                      except Exception as e:
00108
                          print(e)
00109
                          RESTError (993)
00110
                          raise SystemExit(933)
00111
                  elif event == sg.WIN_CLOSED or event == "Quit":
00112
00113
00114
              window.close()
00115
00116
          @staticmethod
00117
          def ___CreateFrame():
00118
00119
          The __CreateFrame function creates the GUI layout for the application.
00120
00121
              The function returns a list of lists that contains all the elements to be displayed in the GUI
00122
             This is done by creating each line as a list and then appending it to another list which will
     contain all lines.
00123
00124
          Args:
00125
00126
          Returns:
00127
              The layout for the gui
00128
00129
          Doc Author:
          Willem van der Schans, Trelent AI
00130
00131
00132
              sq.theme('Default1')
00133
00134
              line00 = [sq.HSeparator()]
```

```
00135
00136
             line0 = [sg.Image(ImageLoader("logo.png")),
00137
                      sq.Push(),
00138
                      sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"),
     justification="center"),
00139
                      sa.Push().
00140
                      sg.Push()]
00141
00142
             line1 = [sg.HSeparator()]
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"),
00145
     key="-Cal-",
00146
                               size=(20, 1)),
00147
                      sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]
00148
00149
             line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                      sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
00150
00151
                               size=(20, 1)),
                      sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00152
     target="-EndCal-")]
00153
00154
             line5 = [sq.HSeparator()]
00155
00156
             line6 = [sq.Push()]
                      sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00157
00158
                      sq.Push()1
00159
00160
             line7 = [sq.HSeparator()]
00161
             00162
00163
00164
                               size=(20, 1)),
                      sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00165
                                    target="-AppendingFile-")]
00166
00167
00168
             line9 = [sg.HSeparator()]
00169
00170
             line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00171
00172
             layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]
00173
00174
             return layout
00175
00176
         def __SetValues(self, values):
00177
00178
00179
         The __SetValues function is used to set the values of the variables that are used in the __GetData
00180
         The __SetValues function takes a dictionary as an argument, and then sets each variable based on what
      is passed into
00181
         the dictionary. The keys for this dictionary are defined by the user when they create their own
     instance of this class.
00182
00183
00184
             self: Represent the instance of the class
00185
             values: Pass in the values from the ui
00186
00187
         Returns:
00188
            A dictionary of values
00189
00190
         Doc Author:
00191
             Willem van der Schans, Trelent AI
00192
00193
             self.size = 1000
00194
              if values["-Cal-"] != "":
00195
00196
                 self.dateStart = values["-Cal-"]
00197
             else:
00198
                 self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00199
00200
             if values["-EndCal-"] != "":
00201
                 self.dateEnd = values["-EndCal-"]
00202
             else:
00203
                 self.dateEnd = date.today().strftime("%Y-%m-%d")
00204
00205
             self.rest domain = settings.settingCMRestDomain
00206
00207
             self.SourceInclude = None
00208
00209
             if values["-append_file-"] != "":
```

```
00210
                                 self.append_file = str(values["-append_file-"])
00211
                          else:
00212
                                 self.append_file = None
00213
00214
                         self.ui_flag = True
00215
00216
00217 class ConstructionMonitorMain:
00218
00219
                  def __init__(self, siteClass):
00220
00221
00222
                  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.
00223
00224
00225
00226
                  Args:
00227
                         self: Represent the instance of the class
00228
                          siteClass: Identify the site that is being used
00229
00230
                  Returns:
00231
                         Nothing
00232
00233
                  Doc Author:
                  Willem van der Schans, Trelent AI
00234
00235
                          self.__siteClass = siteClass
00236
                          self.__restDomain = None
00237
                          self.__headerDict = None
00238
00239
                          self.__columnSelection = None
                         self.__appendFile = None
00240
00241
00242
                          self.__parameterDict = {}
00243
                          self.__search_id = None
00244
                          self.\_\_record\_val = 0
00245
                          self. batches = 0
00246
00247
                          self.__ui_flag = None
00248
                         self.dataframe = None
00249
00250
00251
                                 self.mainFunc()
00252
00253
                          except SystemError as e:
00254
                                 if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
00255
                                        print (
00256
                                                f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in
         ConstructionMonitorMain class")
00257
00258
                          except AttributeError as e:
                                 # This allows for user cancellation of the program using the quit button
if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
00259
00260
00261
                                        RESTError (1101)
00262
                                         print(f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ Error \ \{e\}") 
00263
                                        pass
00264
                                 elif e is not None:
00265
                                       print(
                                                \verb|f"ConstructionMonitor/Core.py | Error = \{e\} | Authentication Error | Please update | Authentication | Core.py | Please update | Core.py | Core
00266
         keys in AuthUtil")
00267
                                        RESTError (401)
00268
                                        print(e)
00269
00270
                                 else:
00271
00272
                          except Exception as e:
00273
                                 print(e)
00274
                                 RESTError (1001)
00275
                                 raise SystemExit(1001)
00276
00277
                  def mainFunc(self):
00278
                  The mainFunc function is the main function of this module. It will be called by the GUI or CLI to
00279
           execute
00280
                  the code in this module. The mainFunc function will first create a parameter dictionary using the
             ParameterCreator
00281
                  method, then it will get a count of all records that match its parameters using the __getCountUI
          method, and then
00282
                  it will calculate how many batches are needed to retrieve all records with those parameters using
           BatchCalculator.
00283
                 After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI
           mode). Then it shows
```

```
00284
         a progress bar for each
00285
00286
00287
            self: Refer to the current object
00288
00289
00290
            The dataframe
00291
00292
         Doc Author:
00293
             Willem van der Schans, Trelent AI
00294
00295
             self.__ParameterCreator()
00296
00297
             print (
00298
                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict =
     {self.__parameterDict}")
00299
             print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
00300
     {self.__restDomain}")
00301
00302
             self.__getCountUI()
00303
00304
             self. batches = BatchCalculator(self. record val, self. parameterDict)
00305
00306
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
00307
     {self.__batches} | Rows {self.__record_val}")
00308
00309
             if self.__batches != 0:
                 startTime = datetime.datetime.now().replace(microsecond=0)
00310
00311
                 eventReturn = BatchInputGui(self.__batches, self.__record_val)
                 if eventReturn == "Continue":
00312
                    print(
00313
                        00314
     {self.__batches} batches sent to server")
00315
                    BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00316
                                                     ParameterDict=self.__parameterDict,
00317
                                                     HeaderDict=self.__headerDict,
00318
                                                     ColumnSelection=self.__columnSelection,
00319
                                                     BatchesNum=self.__batches,
00320
                                                     Type="construction_monitor")
00321
                    BatchGuiObject.BatchGuiShow()
00322
                     self.dataframe = BatchGuiObject.dataframe
                    print(
00323
00324
                         retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in
      {\tt \{time.strftime('\%H:\%H:\%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - 1000000000)} \\
      startTime).total_seconds()))}")
00325
                    FileSaver("cm", self.dataframe, self.__appendFile)
00326
                 else:
00327
00328
                        {self.__batches} batches canceled by user")
00329
00330
                 RESTError (994)
00331
                 raise SystemExit (994)
00332
         def __ParameterCreator(self):
00333
00334
00335
         The __ParameterCreator function is used to create the parameter dictionary that will be passed into
     the
00336
              _Request function. The function takes <mark>in</mark> a siteClass object <mark>and</mark> extracts all of its attributes,
     except for
00337
             those that start with '__' or are callable. It then creates a dictionary from these attributes and
     stores it as
00338
             self.__parameterDict.
00339
00340
         Args:
00341
            self: Make the function a method of the class
00342
00343
         Returns:
00344
             A dictionary of parameters and a list of non parameter variables
00345
00346
         Doc Author:
00347
             Willem van der Schans, Trelent AI
00348
00349
               _Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                             not key.startswith('__') and not callable(key)}
00350
00351
00352
             self. restDomain = Source dict["rest domain"]
             __Source_dict.pop("rest_domain")
00353
```

```
00354
              self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00355
              __Source_dict.pop("auth_key")
00356
              self.__columnSelection = __Source_dict["SourceInclude"]
              __Source_dict.pop("SourceInclude")
00357
00358
              self.__ui_flag = __Source_dict["ui_flag"]
00359
              __Source_dict.pop("ui_flag")
00360
              self.__appendFile = __Source_dict["append_file"]
00361
              __Source_dict.pop("append_file")
00362
              temp_dict = copy.copy(__Source_dict)
00363
00364
              for key, value in temp_dict.items():
00365
                  if value is None:
00366
                      __Source_dict.pop(key)
00367
00368
00369
              self.__parameterDict = copy.copy(__Source_dict)
00370
00371
00372
          def __getCount(self):
00373
          The __getCount function is used to get the total number of records that are returned from a query.
00374
00375
          This function is called by the __init__ function and sets the self.__record_val variable with this
      value.
00376
00377
          Args:
00378
              self: Represent the instance of the class
00379
00380
          Returns:
00381
              The total number of records in the database
00382
00383
          Doc Author:
          Willem van der Schans, Trelent AI
00384
00385
00386
                _count_resp = None
00387
00388
              try:
00389
00390
                  __temp_param_dict = copy.copy(self.__parameterDict)
00391
00392
                  __count_resp = requests.post(url=self.__restDomain,
00393
                                                headers=self.__headerDict,
00394
                                                json=__temp_param_dict)
00395
00396
              except requests.exceptions.Timeout as e:
00397
                  print(e)
00398
                  RESTError (790)
00399
                  raise SystemExit(790)
00400
              except requests.exceptions.TooManyRedirects as e:
00401
                  print(e)
00402
                  RESTError (791)
00403
                  raise SystemExit(791)
00404
              except requests.exceptions.MissingSchema as e:
00405
                  print(e)
00406
                  RESTError (1101)
00407
              except requests.exceptions.RequestException as e:
00408
                  print(e)
00409
                  RESTError (405)
00410
                  raise SystemExit(405)
00411
00412
              __count_resp = __count_resp.json()
00413
00414
              self.__record_val = __count_resp["hits"]["total"]["value"]
00415
              del __count_resp, __temp_param_dict
00416
00417
00418
          def __getCountUI(self):
00419
00420
00421
               _getCountUI function is a wrapper for the __getCount function.
00422
          It allows the user to run __getCount in a separate thread, so that they can continue working while it
     runs.
00423
          The function will display a progress bar and update with text as it progresses through its tasks.
00424
00425
          Args:
00426
              self: Access the class variables and methods
00427
00428
          Returns:
00429
              The count of the number of records in the database
00430
00431
          Doc Author:
00432
              Willem van der Schans, Trelent AI
```

4.21 Realtor/Core.py 155

```
00433
00434
              if self.__ui_flag:
00435
                  uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00436
00437
                  threadGui = threading.Thread(target=self.__getCount,
00438
                  threadGui.start()
00439
00440
00441
                  while threadGui.is_alive():
00442
                      uiObj.textUpdate()
00443
                      uiObj.windowPush()
00444
                      uiObj.stopWindow()
              else:
00448
                  self.__getCount()
```

4.21 Realtor/Core.py

```
00001 import datetime
00002 import threading
00003 import time
00005 import pandas as pd
00006 import requests
00007 from bs4 import *
00008
00009 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00010 from API_Calls.Functions.DataFunc.Settings import settings
00011 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00012 from API Calls.Functions.Gui.BatchGui import confirmDialog
00013 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00014
00015
00016 class realtorCom:
00017
00018
          def __init__(self):
00019
00020
          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
00021
     anything else in your class.
00022
00023
              self: Represent the instance of the class
00024
00025
00026
          Returns:
00027
              A new object
00028
00029
          Doc Author:
          Willem van der Schans, Trelent AI
00030
00031
00032
              self.__page_html = None
00033
              self.__update_date = None
00034
              self.__last_date = None
              self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00035
00036
              self.__linkDict = {}
00037
              self.dfState = None
00038
              self.dfCounty = None
              self.dfZip = None
00039
00040
              self.uiString = "Files Saved to \n"
00041
              eventReturn = confirmDialog()
00042
00043
00044
                  page_html = requests.get(settings.settingRealtorLink).text
00045
                  self.__page_html = BeautifulSoup(page_html, "html.parser")
                  startTime = datetime.datetime.now().replace(microsecond=0)
00046
00047
                  self.__linkGetter()
00048
                      f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Link Dictionary =
00049
      {self.__idDict}")
00050
                  self.__showUi()
00051
                  PopupWrapped(text=self.uiString, windowType="noticeLarge")
00052
                  print(
                      f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
      in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
00054
             else:
```

```
00055
                  print(
                      f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
00056
      Request")
00057
00058
          def __showUi(self):
00059
00060
00061
00062
          The __showUi function is a helper function that creates and displays the progress window.
00063
          It also starts the dataUpdater thread, which will update the progress bar as it runs.
00064
00065
00066
          Args:
00067
              self: Represent the instance of the class
00068
00069
          Returns:
00070
              A popupwrapped object
00071
00072
          Doc Author:
          Willem van der Schans, Trelent AI
00073
00074
00075
              uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)
00076
00077
              threadGui = threading.Thread(target=self.__dataUpdater,
00078
                                           daemon=False)
00079
              threadGui.start()
00080
00081
              while threadGui.is alive():
00082
                  uiObj.textUpdate()
00083
                  uiObj.windowPush()
00084
              else:
00085
                  uiObj.stopWindow()
00086
00087
          def __linkGetter(self):
00088
00089
00090
               _linkGetter function <mark>is</mark> a private function that takes the idDict dictionary <mark>and</mark> adds
00091
          a link to each entry \underline{i}n the dictionary. The link \underline{i}s used to access historical data \underline{f}or each
00092
          scope symbol.
00093
00094
00095
              self: Refer to the object itself
00096
00097
          Returns:
00098
              A dictionary of all the links to the history pages
00099
          Doc Author:
00100
00101
              Willem van der Schans, Trelent AI
00102
00103
              for key, value in self.__idDict.items():
                  for row in self.__page_html.find_all("div", {"class": "monthly"}):
00104
00105
00106
                          for nestedRow in row.find_all("a"):
                              if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
00107
00108
                                  self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
00109
                      except Exception as e:
00110
                          Realtor/Core.py | Error = {e} | Error while getting document links for realtor.com")
00111
                          RESTError (801)
00112
                          raise SystemExit(801)
00113
00114
          def __dataUpdater(self):
00115
00116
00117
          The __dataUpdater function is a private function that updates the dataframes for each of the three
00118
              types of realtor data. It takes class variables and return the path to the saved file. The
      function first creates an empty
00119
              dictionary called tempdf, then iterates through each key in self. __idDict (which contains all
      three ids).
00120
              For each key, it reads in a csv file from the link associated with that id and saves it to tempdf
      as a pandas
00121
              DataFrame object. Then, depending on which type of realtor data we are dealing with
      (State/County/Zip), we save
00122
00123
00124
         Aras:
00125
             self: Access the attributes and methods of the class
00126
00127
          Returns:
00128
              The path of the saved file
00129
```

```
00130
          Doc Author:
          Willem van der Schans, Trelent AI
00131
00132
00133
              for key, value in self.__idDict.items():
00134
                  tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00135
00136
                  if kev == "State":
00137
                     self.dfState = tempdf
00138
                  elif key == "County":
                     self.dfCounty = tempdf
00139
                  elif key == "Zip":
00140
                     self.dfZip = tempdf
00141
00142
                  FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00144
                  self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

4.22 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 sq
00011 import requests
00012 from cryptography.fernet import Fernet
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.DataFunc.Settings import settings
00018 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00019 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00020 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00021 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00022 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00023
00024
00025 class UtahRealEstateInit:
00026
00027
          def __init__(self):
00028
00029
00030
          The __init__ function is called when the class is instantiated.
00031
          It sets up the initial state of the object.
00032
00033
00034
00035
             self: Represent the instance of the class
00036
00037
          Returns:
            The __createframe function
00038
00039
00040
          Doc Author:
         Willem van der Schans, Trelent AI
00041
00042
00043
              self.StandardStatus = None
00044
              self.ListedOrModified = None
00045
              self.dateStart = None
00046
              self.dateEnd = None
00047
              self.select = None
              self.file_name = None
00048
00049
              self.append_file = None
00050
00051
              self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00052
00053
          def __ShowGui(self, layout, text):
00054
00055
          The __ShowGui function is a helper function that creates the GUI window and displays it to the user.
00057
          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
```

```
00059
                  method then uses these parameters to create an instance of sg.Window with all its attributes set
          accordingly.
00060
00061
00062
                         self: Refer to the current class instance
00063
                         layout: Pass the layout of the window to be created
00064
                         text: Set the title of the window
00065
00066
                        A dictionary of values
00067
00068
00069
                  Doc Author:
                 Willem van der Schans, Trelent AI
00070
00071
00072
                         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00073
                                                           finalize=True,
00074
                                                           icon=ImageLoader("taskbar_icon.ico"))
00075
00076
                         while True:
00077
                               event, values = window.read()
00078
00079
                                if event == "Submit":
08000
                                       try:
00081
                                              self.__SetValues(values)
00082
                                              break
00083
                                       except Exception as e:
00084
                                              print(e)
00085
                                              RESTError (993)
00086
                                              raise SystemExit(993)
                                elif event == sg.WIN_CLOSED or event == "Quit":
00087
00088
00089
00090
                         window.close()
00091
00092
                  @staticmethod
00093
                  def __CreateFrame():
00094
00095
                          _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
                 Doc Author:
00104
                 Willem van der Schans, Trelent AI
00105
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
                                        sq.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
                         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00122
00123
                                        sq.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date",
          "Close Date"],
00124
                                                              key="-type-", size=(31, 1))]
00125
                        line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00126
00127
                                        sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
          key="-DateStart-",
00128
                                                        disabled=False, size=(20, 1)),
                                        sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00129
          target="-DateStart-")]
00130
00131
                         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                                         \verb|sg.Input(default\_text=(date.today().strftime("%Y-%m-%d"))|, | key="-DateEnd-", | left = l
00132
          disabled=False,
```

```
00133
                               size=(20, 1)),
                      sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-',
00134
     target="-DateEnd-")]
00135
00136
             line7 = [sg.HSeparator()]
00137
00138
             line8 = [sg.Push(),
                      sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00139
00140
                      sg.Push()]
00141
00142
             line9 = [sg.HSeparator()]
00143
             00144
00145
00146
                                size=(20, 1)),
00147
                       sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00148
                                     target="-AppendingFile-")]
00149
00150
             line11 = [sq.HSeparator()]
00151
00152
             line12 = [sq.Push(), sq.Submit(focus=True), sq.Quit(), sq.Push()]
00153
00154
             layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
00155
                       line12]
00156
00157
             return layout
00158
00159
         def SetValues(self, values):
00160
00161
         The \_SetValues function is used to set the values of the variables that are used in the
00162
              _GetData function. The values are passed from a dictionary called 'values' which is created
00163
00164
            by parsing through an XML file using ElementTree. This function also sets default values for
00165
            some of these variables if they were not specified in the XML file.
00166
00167
00168
             self: Represent the instance of the class
00169
             values: Pass the values from the qui to this function
00170
00171
         Returns:
00172
             A dictionary with the following keys:
00173
00174
         Doc Author:
00175
             Willem van der Schans, Trelent AI
00176
00177
             self.StandardStatus = values["-status-"]
00178
00179
             self.ListedOrModified = values["-type-"]
00180
00181
             if values["-DateStart-"] != "":
00182
                 self.dateStart = values["-DateStart-"]
00183
             else:
00184
                 self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00185
00186
             if values["-DateEnd-"] != "":
00187
                 self.dateEnd = values["-DateEnd-"]
00188
00189
                 self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00190
00191
             self.select = None
00192
00193
              if values["-append_file-"] != "":
00194
                 self.append_file = str(values["-append_file-"])
00195
             else:
00196
                 self.append_file = None
00197
00198
00199 class UtahRealEstateMain:
00200
00201
         def __init__(self, siteClass):
00202
00203
00204
         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.
00205
00206
00207
         Aras:
00208
             self: Represent the instance of the class
00209
             siteClass: Determine which site to pull data from
00210
00211
         Returns:
00212
             Nothing
```

```
00213
00214
          Doc Author:
          Willem van der Schans, Trelent AI
00215
00216
00217
              self.dataframe = None
00218
              self.\_batches = 0
00219
              self.__siteClass = siteClass
00220
              self.__headerDict = None
00221
              self.__parameterString = ""
00222
              self.__appendFile = None
00223
              self.__dateStart = None
00224
              self.__dateEnd = None
00225
              self.__restDomain = settings.settingURERestDomain
              self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00226
00227
                  "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00228
              self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00229
                  "Security").joinpath("auth.json")
00230
              self.key = None
00231
              self.__record_val = None
00232
00233
              try:
00234
                 self.mainFunc()
00235
              except KeyError as e:
00236
                  # This allows for user cancellation of the program using the guit button
00237
                  if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00238
                      RESTError (1101)
00239
                      print(e)
00240
00241
                  else:
00242
00243
              except Exception as e:
00244
                  print(e)
00245
                  RESTError (1001)
                  raise SystemExit(1001)
00246
00247
          def mainFunc(self):
00248
00249
00250
          The mainFunc function is the main function of this module. It will be called by the GUI when a user
00251
      clicks on
00252
          the " Run" button in the GUI. The mainFunc function should contain all of your code for
      running your program, and it
00253
          should return a dataframe that contains all the data you want to display in your final report.
00254
00255
00256
              self: Reference the object itself
00257
00258
          Returns:
00259
             A dataframe
00260
          Doc Author:
00261
          Willem van der Schans, Trelent AI
00262
00263
00264
              passFlag = False
00265
00266
              while not passFlag:
00267
                  if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00268
00269
                          f = open(self.keyPath, "rb")
00270
                          key = f.readline()
00271
                          f.close()
00272
                          f = open(self.filePath, "rb")
00273
                          authDict = json.load(f)
00274
                          fernet = Fernet(key)
00275
                          authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00276
                          self.__headerDict = {authDict["ure"]["parameter"]: authkey}
00277
                          passFlag = True
00278
                      except Exception as e:
00279
                         print(
00280
                              f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
     UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
00281
                          AuthUtil()
00282
                  else:
00283
                     AuthUtil()
00284
00285
              self.__ParameterCreator()
00286
00287
                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String =
00288
      {self.__parameterString}")
00289
              print (
```

```
00290
                                         {self. restDomain}")
00291
00292
                               self.__getCountUI()
00293
                               if self.__record_val is None:
00294
00295
                                       self.__record_val = 0
00296
00297
                               self.__batches = BatchCalculator(self.__record_val, None)
00298
00299
                                       f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
00300
             {self.__batches} | Rows {self.__record_val}")
00301
00302
                               if self. batches != 0:
00303
                                        startTime = datetime.datetime.now().replace(microsecond=0)
00304
                                        eventReturn = BatchInputGui(self.__batches, self.__record_val)
00305
                                         if eventReturn == "Continue":
00306
                                                print(
                                                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
00307
             {self.__batches} batches sent to server")
00308
                                                 BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                                                                                               ParameterDict=self.__parameterString,
00309
                                                                                                                               HeaderDict=self.__headerDict,
00310
00311
                                                                                                                               BatchesNum=self.__batches,
                                                                                                                               Type="utah_real_estate")
00312
00313
                                                 BatchGuiObject.BatchGuiShow()
00314
                                                 self.dataframe = BatchGuiObject.dataframe
00315
                                                  print(
                                                          f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ Dataframe = (1.5) | \ Dataframe = 
00316
             retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in
              startTime).total_seconds()))}")
                                                FileSaver("ure", self.dataframe, self._appendFile)
00317
00318
                                        else:
00319
                                                print(
00320
                                                          f"\{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]\} \ | \ Request for the property of the pr
             {self.__batches} batches canceled by user")
00321
                               else:
                                       RESTError (994)
00322
00323
                                        raise SystemExit (994)
00324
00325
                                   _ParameterCreator(self):
00326
00327
                      The __ParameterCreator function is used to create the filter string for the ReST API call.
00328
                      The function takes in a siteClass object and extracts all of its parameters into a dictionary.
00329
                      It then creates an appropriate filter string based on those parameters.
00330
00331
00332
                              self: Bind the object to the class
00333
00334
00335
                             A string to be used as the parameter in the api call
00336
00337
                      Doc Author:
                      Willem van der Schans, Trelent AI
00338
00339
00340
                               filter string = ""
00341
                               __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if not key.startswith('__') and not callable(key)}
00342
00343
00344
00345
                               self.__appendFile = __Source_dict["append_file"]
                               __Source_dict.pop("append_file")
00346
00347
00348
                               temp_dict = copy.copy(__Source_dict)
00349
                               for key, value in temp_dict.items():
00350
                                        if value is None:
00351
                                                 __Source_dict.pop(key)
00352
                                        else:
00353
00354
                               if __Source_dict["ListedOrModified"] == "Listing Date":
00355
00356
                                         filter string =
             f"$filter=ListingContractDate%20qt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEn
                               elif __Source_dict["ListedOrModified"] == "Modification Date":
00357
00358
                                       filter_string =
             f"$filter=ModificationTimestamp$20gt$20{__Source_dict['dateStart']}T:00:00:00Z$20and$20ModificationTimestamp$20le$20{__Source_dict["ListedOrModified"] == "Close Date":
00359
00360
                                        filter_string =
             f"$filter=CloseDate%20qt%20{ Source dict['dateStart']}%20and%20CloseDate%201e%20{ Source dict['dateEnd']}
```

```
00361
              filter_string = filter_string +
00362
      f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['StandardStatus']}'"
00363
00364
              self.__parameterString = filter_string
00365
00366
          def __getCount(self):
00367
00368
          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
00369
00370
          the $count parameter in OData to return only an integer value representing how many records would be
      returned
00371
          by the query.
00372
00373
          Args:
00374
              self: Represent the instance of the class
00375
00376
          Returns:
00377
             The number of records in the data set
00378
00379
          Doc Author:
          Willem van der Schans, Trelent AI
00380
00381
00382
               _count_resp = None
00383
00384
              try:
00385
                   _count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00386
                                               headers=self.__headerDict)
00387
00388
              except requests.exceptions.Timeout as e:
00389
                  print(e)
00390
                  RESTError (790)
                  raise SystemExit(790)
00391
              {\tt except} \ {\tt requests.exceptions.TooManyRedirects} \ {\tt as} \ {\tt e:}
00392
00393
                 print(e)
00394
                  RESTError (791)
00395
                  raise SystemExit(791)
00396
              except requests.exceptions.MissingSchema as e:
00397
                  print(e)
00398
                  RESTError (1101)
00399
              except requests.exceptions.RequestException as e:
                  print(e)
00400
                  RESTError (405)
00401
00402
                  raise SystemExit(405)
00403
00404
              self.__record_val = int(__count_resp.json()["@odata.count"])
00405
00406
          def __getCountUI(self):
00407
00408
00409
          The __getCountUI function is a wrapper for the __getCount function.
00410
          It creates a progress window and updates it while the __getCount function runs.
00411
          The purpose of this is to keep the GUI responsive while running long processes.
00412
00413
00414
             self: Represent the instance of the class
00415
00416
          Returns:
00417
             A popupwrapped object
00418
00419
          Doc Author:
00420
              Willem van der Schans, Trelent AI
00421
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()
```