Gardner API Utility Documentation

Willem van der Schans
Version 1.0.2
4/17/2023 3:22:00 PM

Table of Contents

README: Gardner Policy Institute API Utility

Author: Willem van der Schans

Commissioner: Gardner Policy Institute

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

Notes

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

VERSION INFO

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

Note: All dependencies are included in the Windows installer

Authentication Requirements

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

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

Changelog

Initial release
Version: 1.0.0

Date: 2023-04-08
Core Functionality

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

User InterFace

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

File Functionality

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

Logging and Error Handling

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

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

Security

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

GUI

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

Other

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

Version: 0.9.5
Date: 2023-04-05

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

Version: 0.9.0
Date: 2023-03-16

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

Version: 0.8.0
Date: 2023-03-12

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

Version: 0.7.0
Date: 2023-03-02

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

Version: 0.6.0
Date: 2023-02-25

```
Enhanced GUI Utility Improvements.
    - Descriptive processing pop-ups
    - Warning popups
```

```
    - MultiThreading
    - Completion Time Estimation
```

Version: 0.5.0
Date: 2023-02-15

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

Version: 0.4.0
Date: 2022-12-07

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

Version: 0.3.0
Date: 2022-11-15

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

Version: 0.2.0
Date: 2022-11-03

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

Version: 0.1.0
Date: 2022-10-25

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

Version: 0.0.0
Date: 2022-10-15

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

License

Class Index

Class List

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

Class Documentation

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**

Private Member Functions

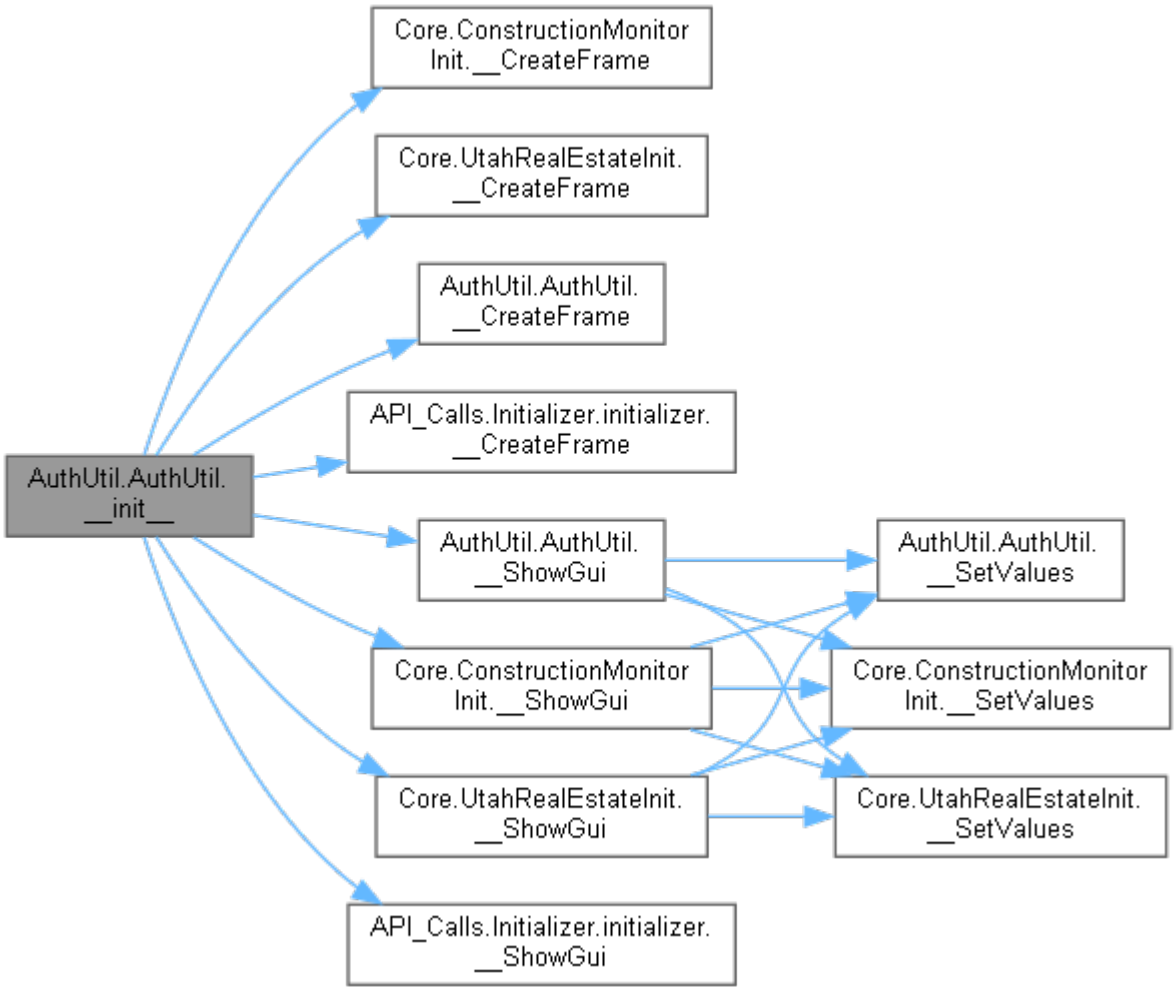
- def __SetValues (self, values)
- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Constructor & Destructor Documentation

def AuthUtil.AuthUtil.__init__ (*self*)

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

Args:
self: Represent the instance of the class

Returns:
None

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

Here is the call graph for this function:

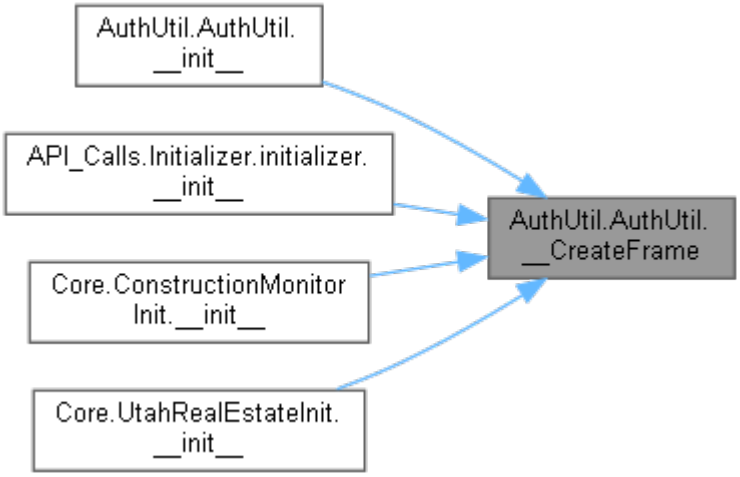


Member Function Documentation

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

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

Args:
self: Access the class attributes and methods

Returns:
A list of lists

Doc Author:
Trelent
```

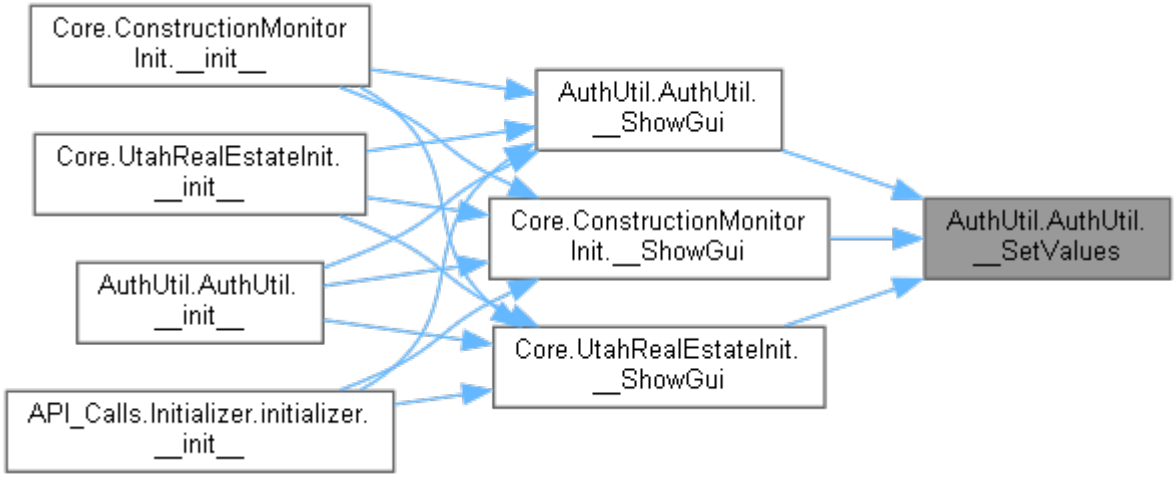
Here is the caller graph for this function:



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

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

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

Returns:
A dictionary of the values entered by the user

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

Here is the caller graph for this function:

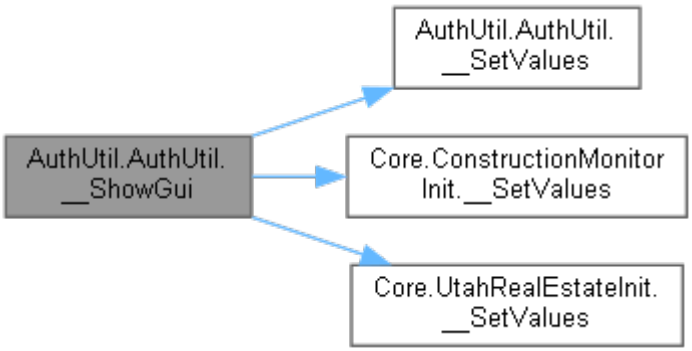


def AuthUtil.AuthUtil.__ShowGui (*self*, *layout*, *text*) `[private]`

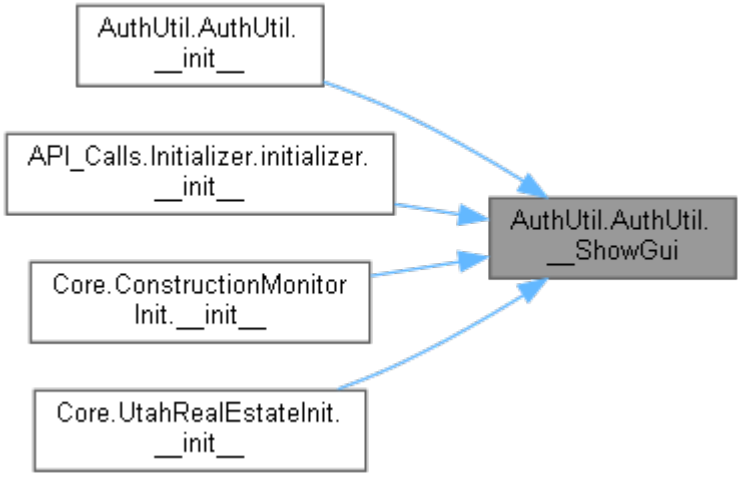
```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the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

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

```
Returns:
A window object
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

- AuthUtil.py

BatchProcessing.BatchProcessorConstructionMonitor Class Reference

Public Member Functions

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

Public Attributes

- **dataframe**
- **valueObject**

Private Attributes

- **__numBatches**
- **__parameterDict**
- **__restDomain**
- **__headerDict**
- **__columnSelection**
- **__maxRequests**
- **__requestCount**
- **__requestCalls**
- **__dateTracker**

Constructor & Destructor Documentation

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

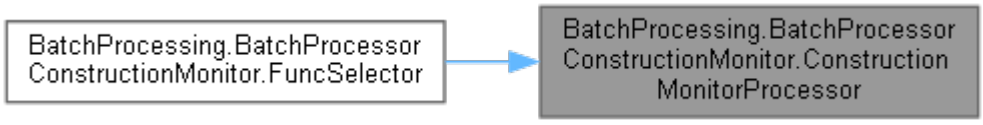
```
The __init__ function is the constructor for a class. It is called when an object of
that class
is created, and it sets up the attributes of that object. In this case, we are setting
up our
object to have a dataframe attribute (which will be used to store all of our data),
as well as
attributes for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

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

Returns:
An object of the class

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

Member Function Documentation

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

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

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

Returns:
A dataframe

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

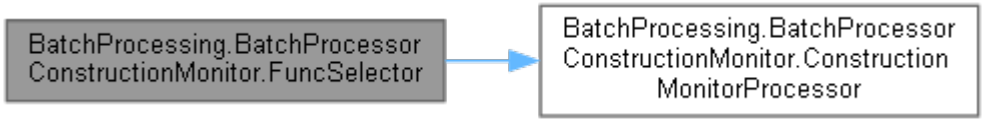
Here is the caller graph for this function:



def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (*self*)

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

Args:
self: Represent the instance of the class

Returns:
The result of the constructionmonitorprocessor function

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

Here is the call graph for this function:



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

- BatchProcessing.py

BatchProcessing.BatchProcessorUtahRealEstate Class Reference

Public Member Functions

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

Public Attributes

- **dataframe**
- **valueObject**

Private Attributes

- **__numBatches**
- **__parameterString**
- **__restDomain**
- **__headerDict**

Constructor & Destructor Documentation

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

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

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

Returns:
The instance of the class

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

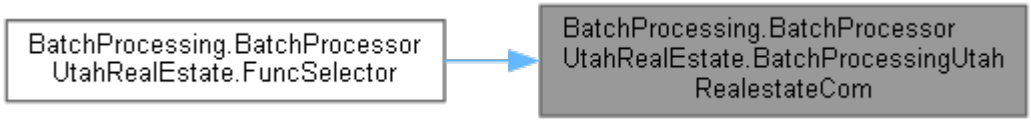
Member Function Documentation

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

```
The BatchProcessingUtahRealestateCom function is a function that takes in the
valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary
information for this
```

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

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

Returns:
A dataframe of the scraped data

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

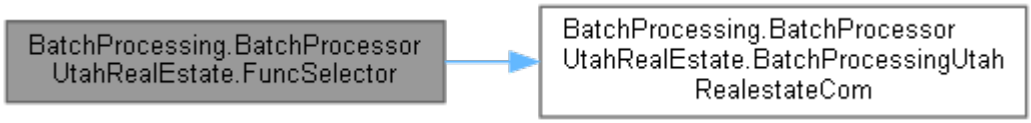
Here is the caller graph for this function:



def BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector (*self*)

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

Args:
self: Represent the instance of the class

Returns:
The function that is selected by the user

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

Here is the call graph for this function:



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

- BatchProcessing.py

BatchProgressGUI.BatchProgressGUI Class Reference

Public Member Functions

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

Public Attributes

- **dataframe**

Private Attributes

- **__parameterDict**
- **__restDomain**
- **__headerDict**
- **__columnSelection**
- **__type**
- **__layout**
- **__batches**
- **__window**
- **__batch_counter**

Constructor & Destructor Documentation

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

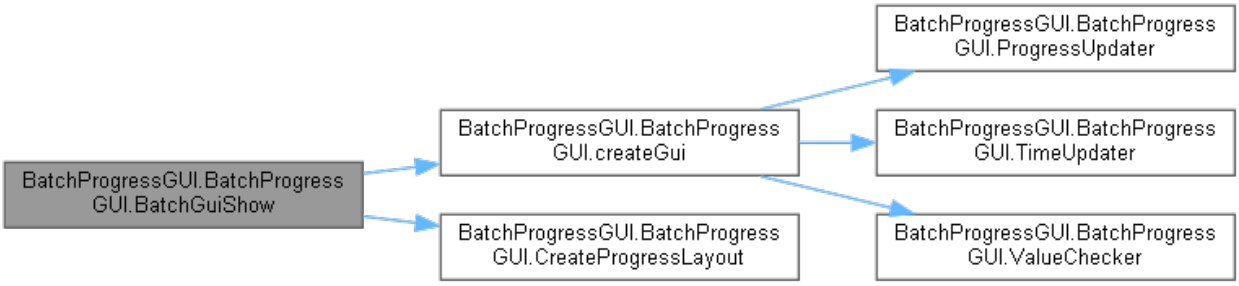
```
The __init__ function is the first function that gets called when an object of this
class is created.
It initializes all the variables and sets up a layout for the GUI. It also creates a
window to display
the dataframe in.

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

Returns:
Nothing

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

Member Function Documentation

def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (*self*)

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

Args:
self: Represent the instance of the class

Returns:
The __type of the batchgui class

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

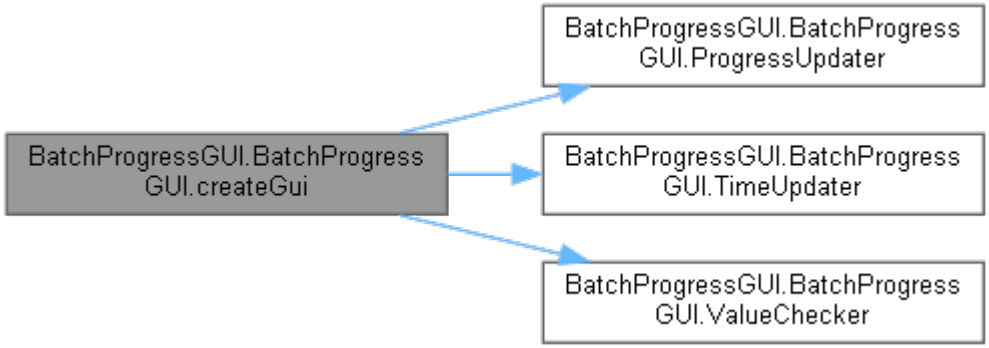
Here is the call graph for this function:



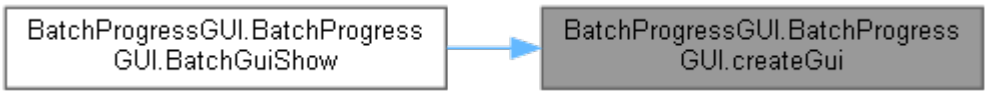
def BatchProgressGUI.BatchProgressGUI.createGui (*self*, *Sourcetype*)

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

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

Returns:
The dataframe

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

Here is the call graph for this function:



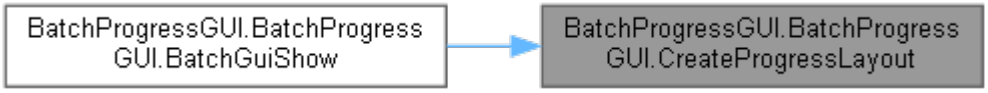
Here is the caller graph for this function:



def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (*self*)

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

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

Args:
self: Access the class variables and methods

Returns:
A list of lists

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

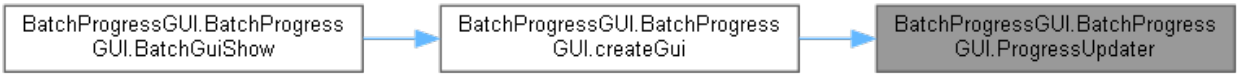
Here is the caller graph for this function:



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

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

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

Returns:
The value of the batch counter

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

Here is the caller graph for this function:

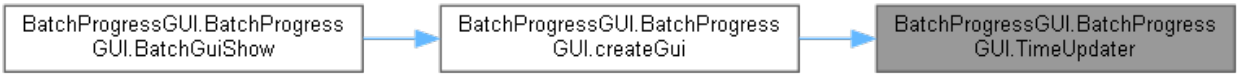


def BatchProgressGUI.BatchProgressGUI.TimeUpdater (*self*, *start_time*)

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

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

```
start_time: Get the time when the function is called

Returns:
A string that is updated every 0

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

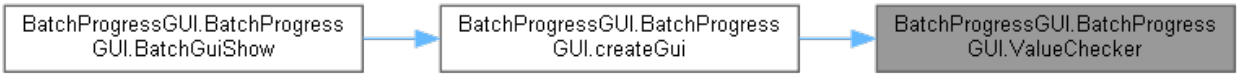
Here is the caller graph for this function:



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

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

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

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

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

Here is the caller graph for this function:



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

- BatchProgressGUI.py

Core.Cencus Class Reference

Public Member Functions

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

Public Attributes

- **state_arg**
- **year_arg**
- **uiString**
- **link**

Private Member Functions

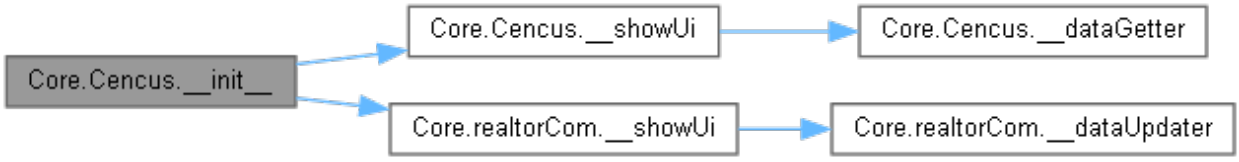
- def __showUi (self)
- def __dataGetter (self)

Constructor & Destructor Documentation

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

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

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

Returns:
A popupwrapped object

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

Here is the call graph for this function:



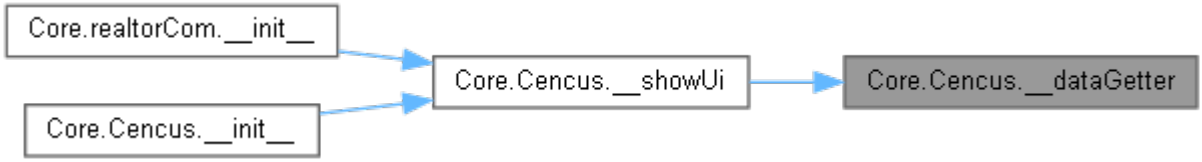
Member Function Documentation

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

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

Args:
self: Represent the instance of the class

Returns:
A response object
```

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

Here is the caller graph for this function:

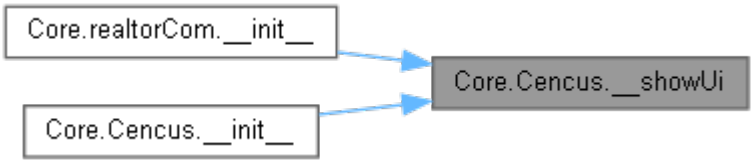


def Core.Cencus.__showUi (*self*) `[private]`

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


Args:
self: Represent the instance of the class

Returns:
The uiobj variable

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

Here is the call graph for this function:

Here is the caller graph for this function:



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

- CFBP/Core.py

Core.ConstructionMonitorInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- **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

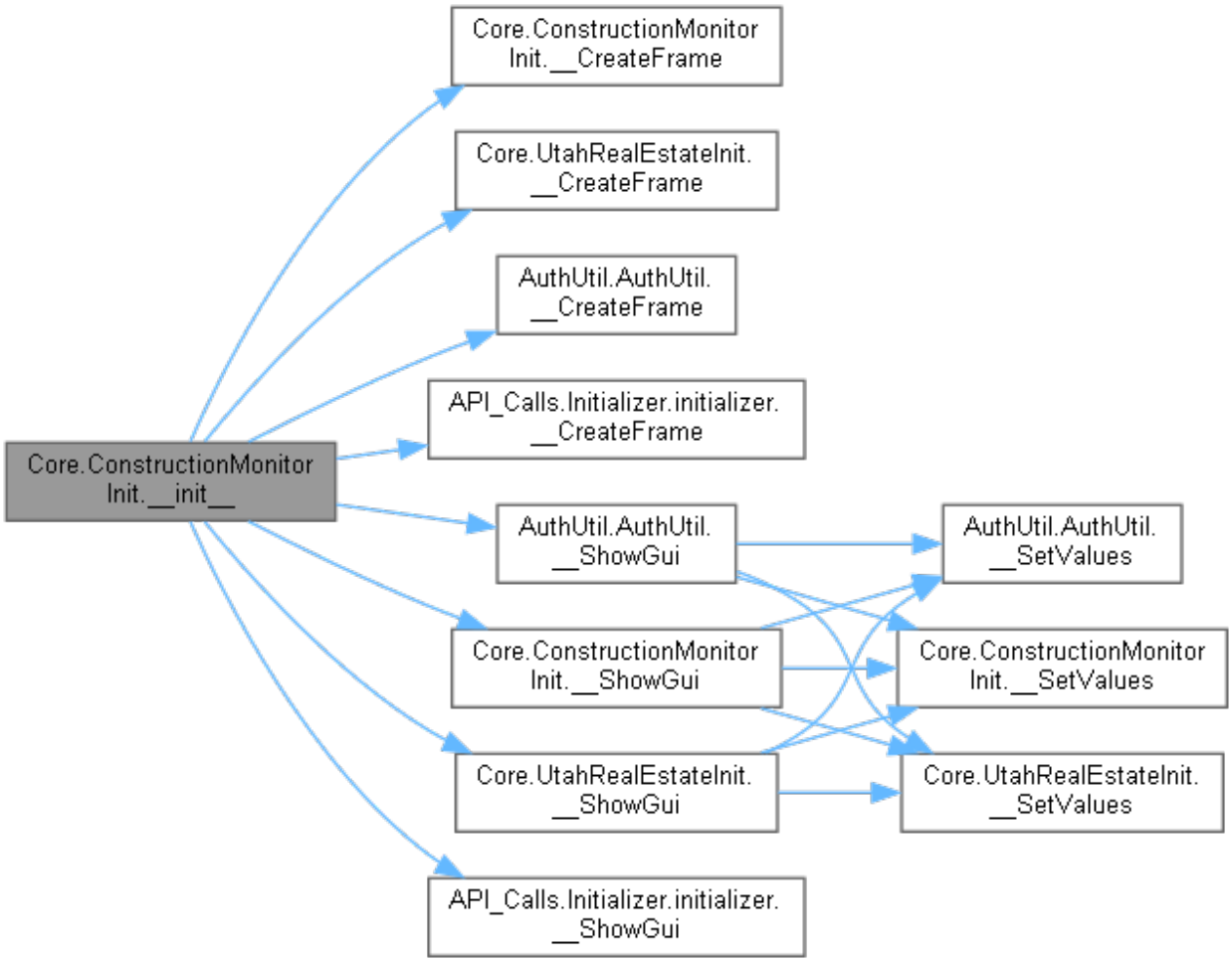
- def __CreateFrame ()

Constructor & Destructor Documentation

def Core.ConstructionMonitorInit.__init__ (*self*)

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


Args:
self: Represent the instance of the class

Returns:
None

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

Here is the call graph for this function:

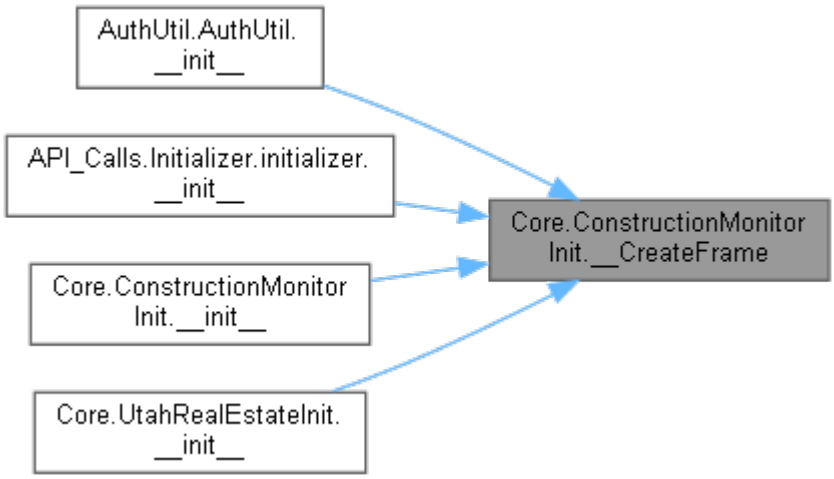


Member Function Documentation

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

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

Args:

Returns:
The layout for the gui

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

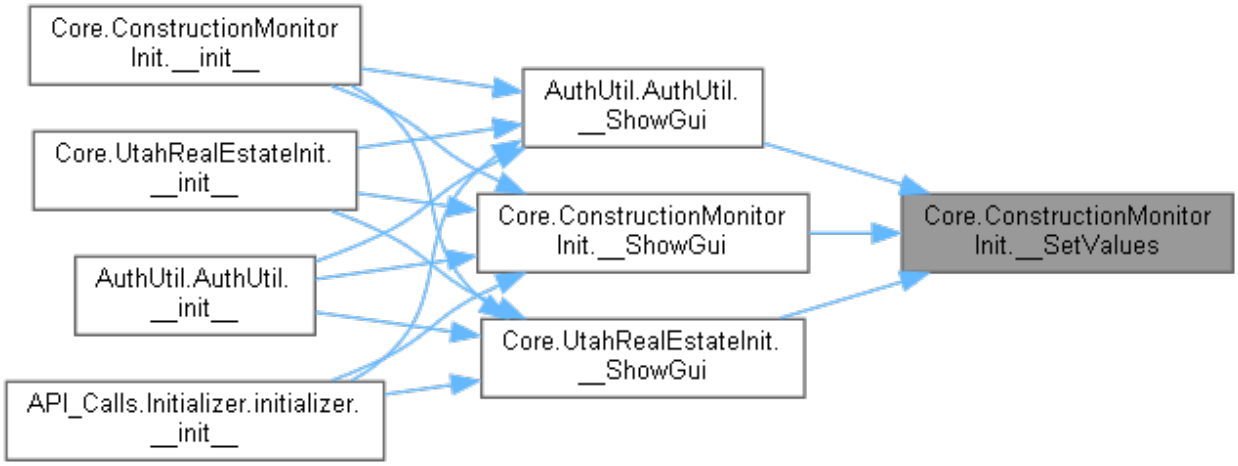
Here is the caller graph for this function:



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

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

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

Returns:
A dictionary of values

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

Here is the caller graph for this function:

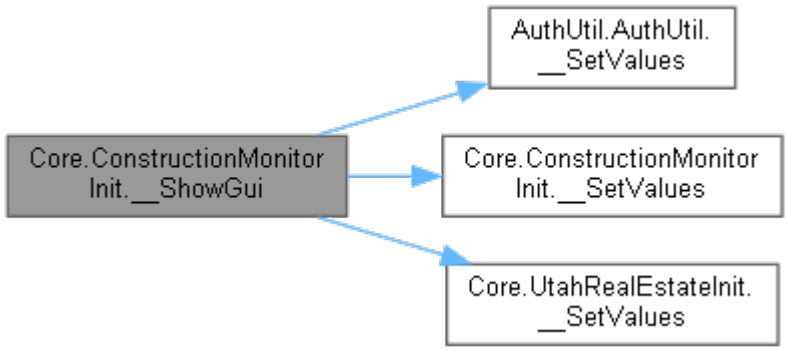


def Core.ConstructionMonitorInit.__ShowGui (*self*, *layout*, *text*) `[private]`

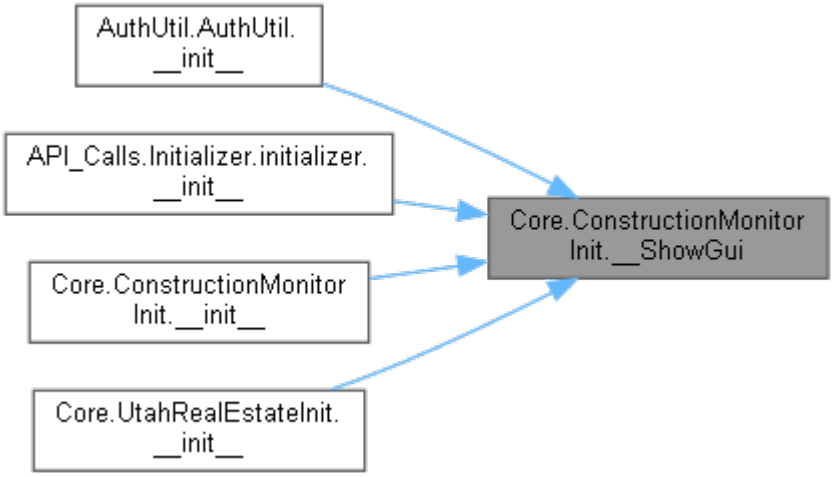
```
The __ShowGui function is the main function that creates and displays the GUI.
It takes in a layout, which is a list of lists containing all the elements to be displayed
on screen.
The text parameter specifies what title should appear at the top of the window.

Args:
self: Refer to the current instance of a class
layout: Determine what the gui will look like
```

```
text: Set the title of the window

Returns:
A dictionary of values

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

- ConstructionMonitor/Core.py

Core.ConstructionMonitorMain Class Reference

Public Member Functions

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

Public Attributes

- **dataframe**

Private Member Functions

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

Private Attributes

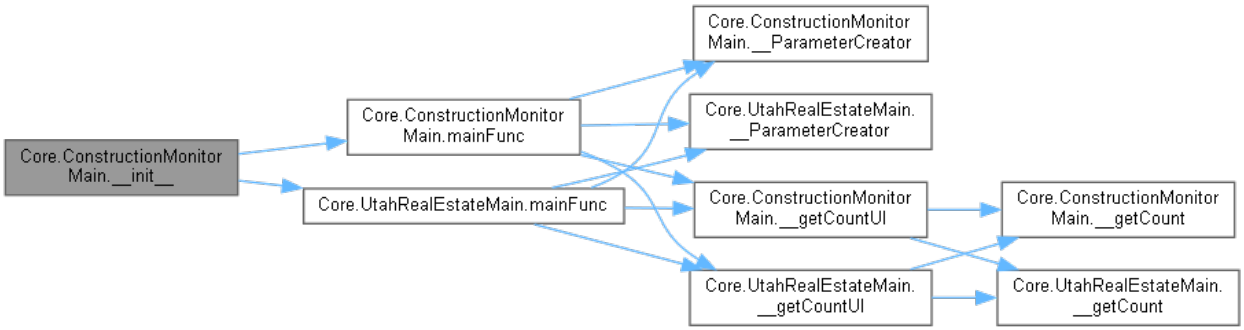
- **__siteClass**
- **__restDomain**
- **__headerDict**
- **__columnSelection**
- **__appendFile**
- **__parameterDict**
- **__search_id**
- **__record_val**
- **__batches**
- **__ui_flag**

Constructor & Destructor Documentation

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

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


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

Returns:
Nothing

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

Here is the call graph for this function:

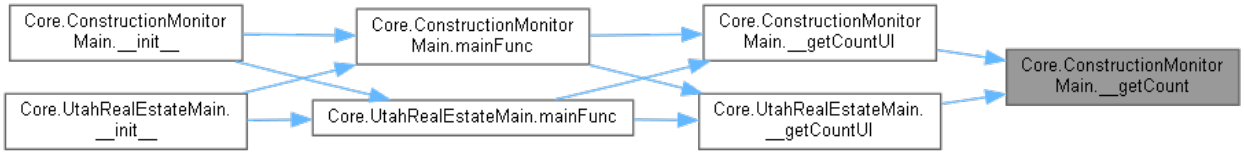


Member Function Documentation

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

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

Args:
self: Represent the instance of the class

Returns:
The total number of records in the database

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

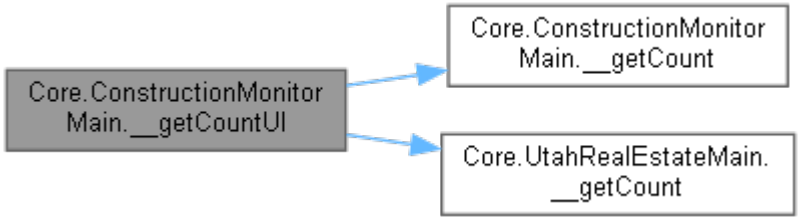
Here is the caller graph for this function:



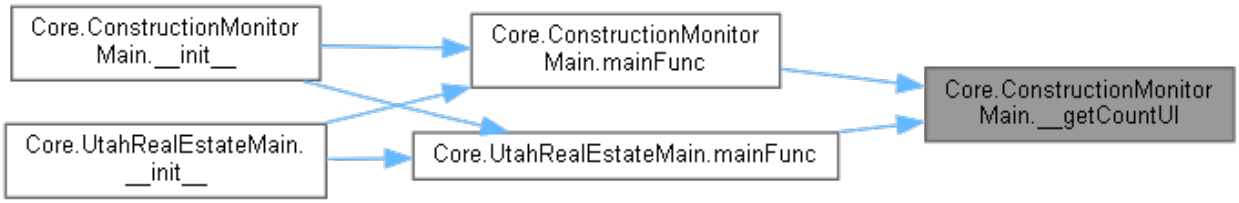
def Core.ConstructionMonitorMain.__getCountUI (*self*) `[private]`

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

Args:
self: Access the class variables and methods

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

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

Here is the call graph for this function:



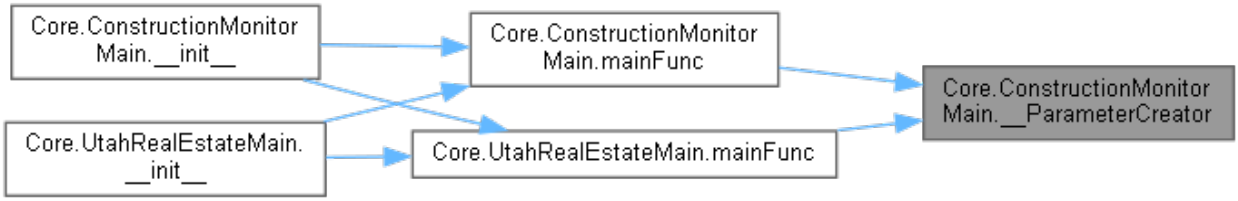
Here is the caller graph for this function:



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

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

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

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

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

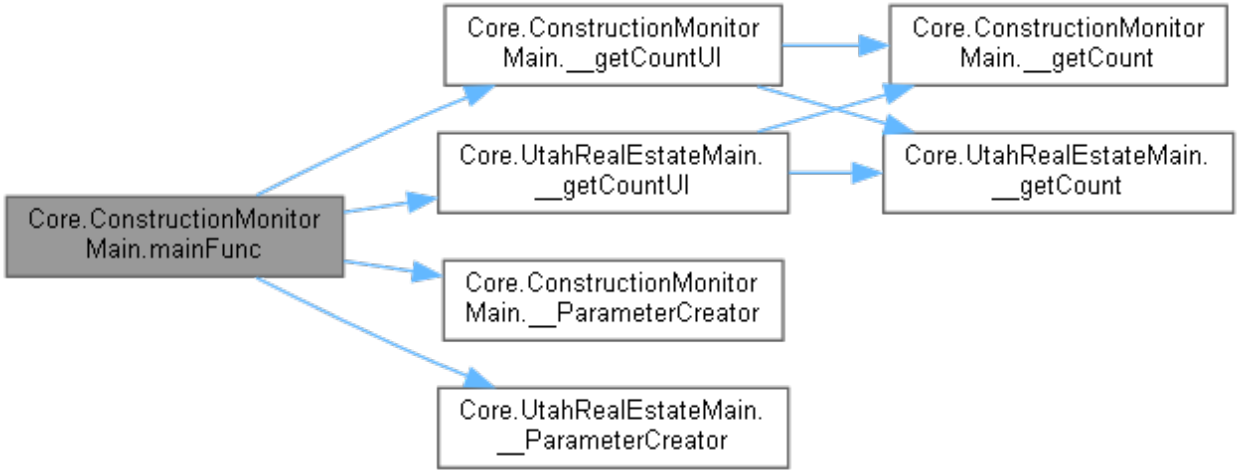
Here is the caller graph for this function:



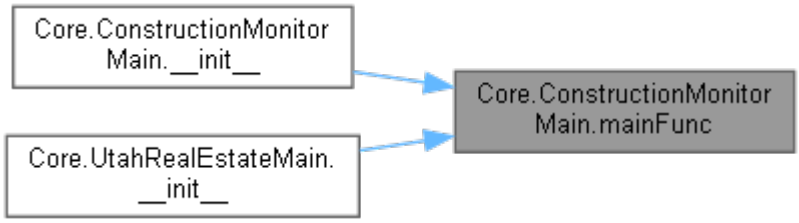
def Core.ConstructionMonitorMain.mainFunc (*self*)

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

Args:
self: Refer to the current object

Returns:
The dataframe

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

- ConstructionMonitor/Core.py

DataTransfer.DataTransfer Class Reference

Public Member Functions

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

Private Attributes

- **__value**

Constructor & Destructor Documentation

def DataTransfer.DataTransfer.__init__ (*self*)

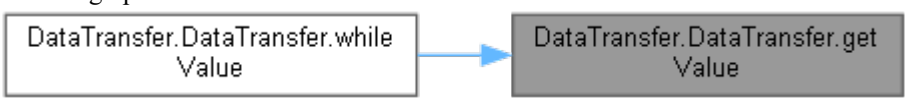
```
The __init__ function is called when the class is instantiated.
It sets the initial value of self.__value to 0.

Args:
self: Represent the instance of the class

Returns:
Nothing

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

Member Function Documentation

def DataTransfer.DataTransfer.getValue (*self*)

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

Args:
self: Represent the instance of the class

Returns:
The value of the instance variable

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

Here is the caller graph for this function:



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

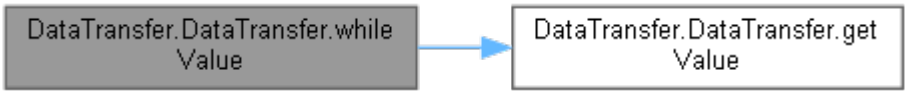
```
The setValue function sets the value of the object.


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

```
value: Set the value of the instance variable __value

Returns:
The value that was passed to it

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

def DataTransfer.DataTransfer.whileValue (*self*)

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

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

Returns:
The value of the input

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

Here is the call graph for this function:



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

- DataTransfer.py

FileSaver.FileSaver Class Reference

Public Member Functions

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

Public Attributes

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

Constructor & Destructor Documentation

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

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

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

Returns:
Nothing

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

Member Function Documentation

def FileSaver.FileSaver.getPath (*self*)

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

Args:
self: Represent the instance of the class

Returns:
The path to the file

Doc Author:
```

```
Willem van der Schans, Trelent AI
```

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

- FileSaver.py

API_Calls.Initializer.initializer Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- **classObj**

Private Member Functions

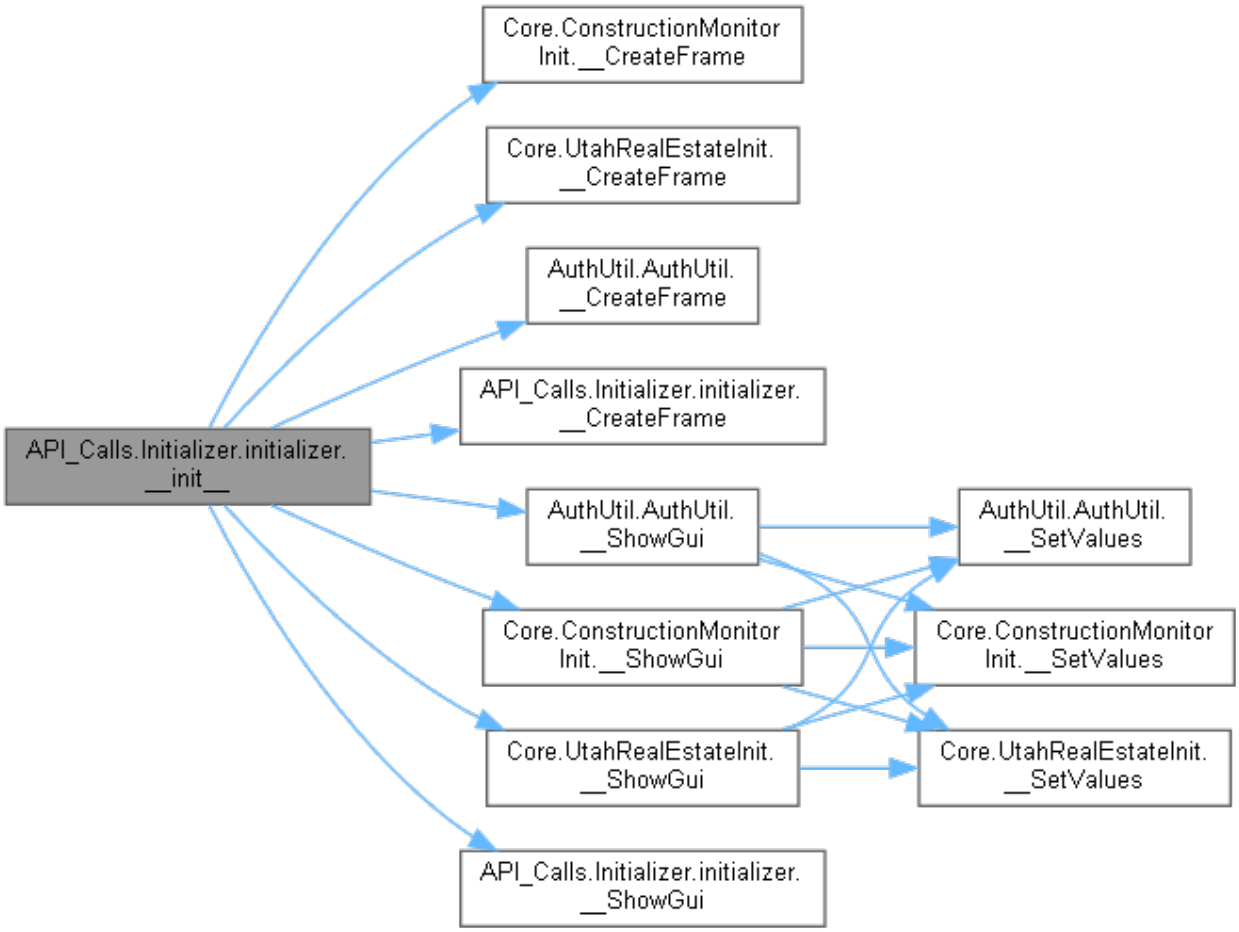
- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

Constructor & Destructor Documentation

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

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


Args:
self: Represent the instance of the class

Returns:
Nothing

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

Here is the call graph for this function:

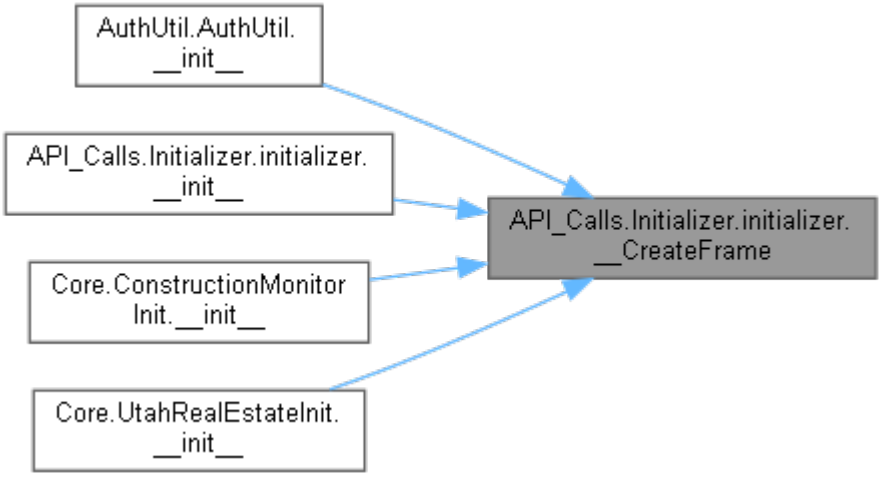


Member Function Documentation

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

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

Args:
self: Represent the instance of the class

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

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

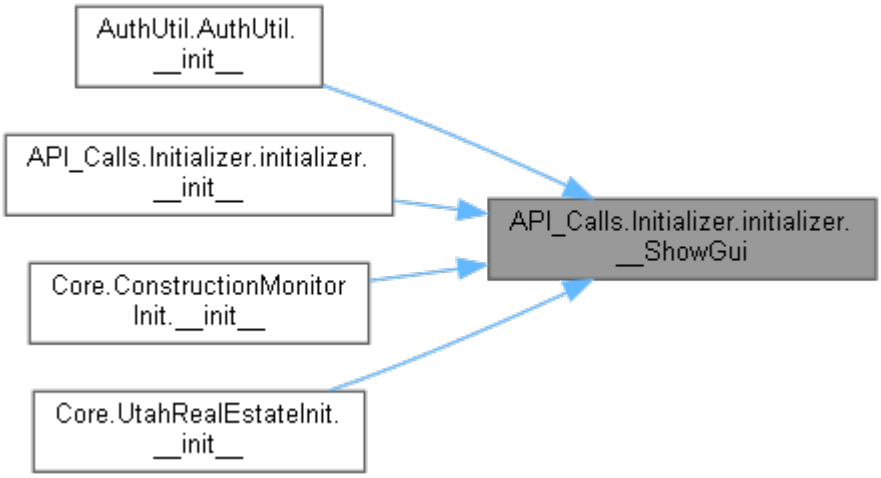
Here is the caller graph for this function:



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

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

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

Returns:
A window object

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

Here is the caller graph for this function:



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

- Initializer.py

PopupWrapped.PopupWrapped Class Reference

Public Member Functions

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

Private Member Functions

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

Private Attributes

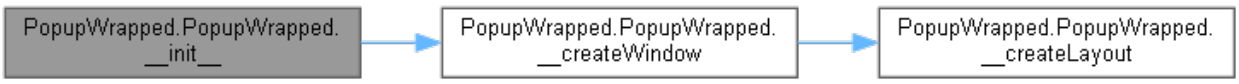
- **__text**
- **__type**
- **__error**
- **__layout**
- **__windowObj**
- **__thread**
- **__counter**

Constructor & Destructor Documentation

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

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

Here is the call graph for this function:



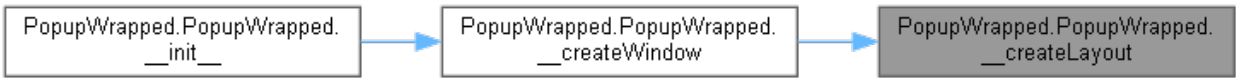
Member Function Documentation

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

```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates
a layout based on that information.
Args:
self: Refer to the current instance of a class
Returns:
```

```
A list of lists
Doc Author:
Willem van der Schans, Trelent AI
```

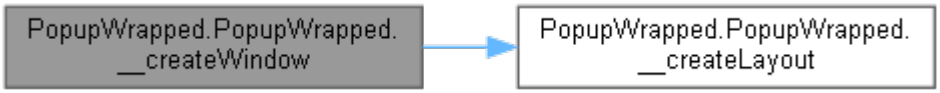
Here is the caller graph for this function:



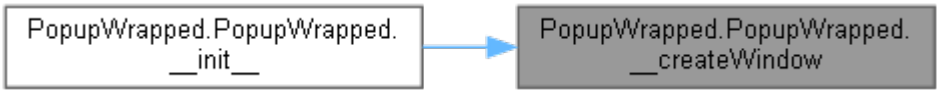
def PopupWrapped.PopupWrapped.__createWindow (*self*) `[private]`

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

Here is the call graph for this function:



Here is the caller graph for this function:



def PopupWrapped.PopupWrapped.stopWindow (*self*)

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

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

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

def PopupWrapped.PopupWrapped.windowPush (*self*)

```
The windowPush function is used to update the values of a window object.
The function takes in an event and values from the window object, then checks if the
event starts with 'update'.
If it does, it will take everything after 'update' as a key for updating that specific
value.
It will then update that value using its key and refresh the window.
Args:
self: Reference the object that is calling the function
Returns:
A tuple containing the event and values
Doc Author:
Willem van der Schans, Trelent AI
```

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

- PopupWrapped.py

Core.realtorCom Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- **dfState**
- **dfCounty**
- **dfZip**
- **uiString**

Private Member Functions

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

Private Attributes

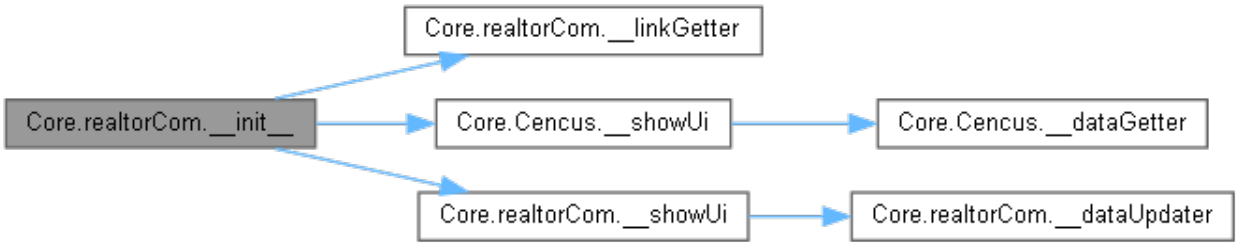
- **__page_html**
- **__update_date**
- **__last_date**
- **__idDict**
- **__linkDict**

Constructor & Destructor Documentation

def Core.realtorCom.__init__ (*self*)

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

Args:
self: Represent the instance of the class

Returns:
A new object

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

Here is the call graph for this function:

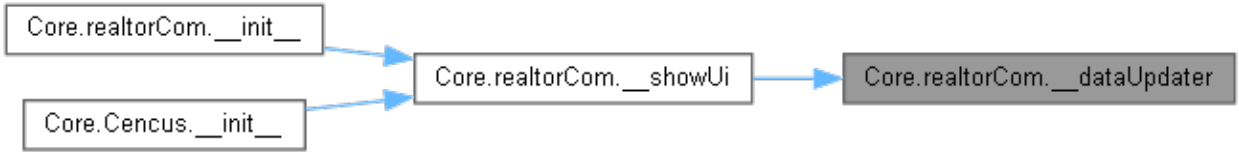


Member Function Documentation

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

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


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

Returns:
The path of the saved file

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

Here is the caller graph for this function:



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

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

Args:
self: Refer to the object itself

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

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

Here is the caller graph for this function:

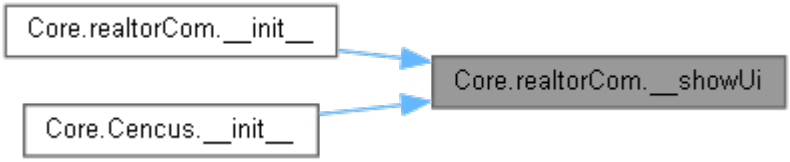
def Core.realtorCom.__showUi (*self*) [private]

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


Args:
self: Represent the instance of the class

Returns:
A popupwrapped object
```

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

Here is the call graph for this function:

Here is the caller graph for this function:



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

- Realtor/Core.py

Core.UtahRealEstateInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- **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

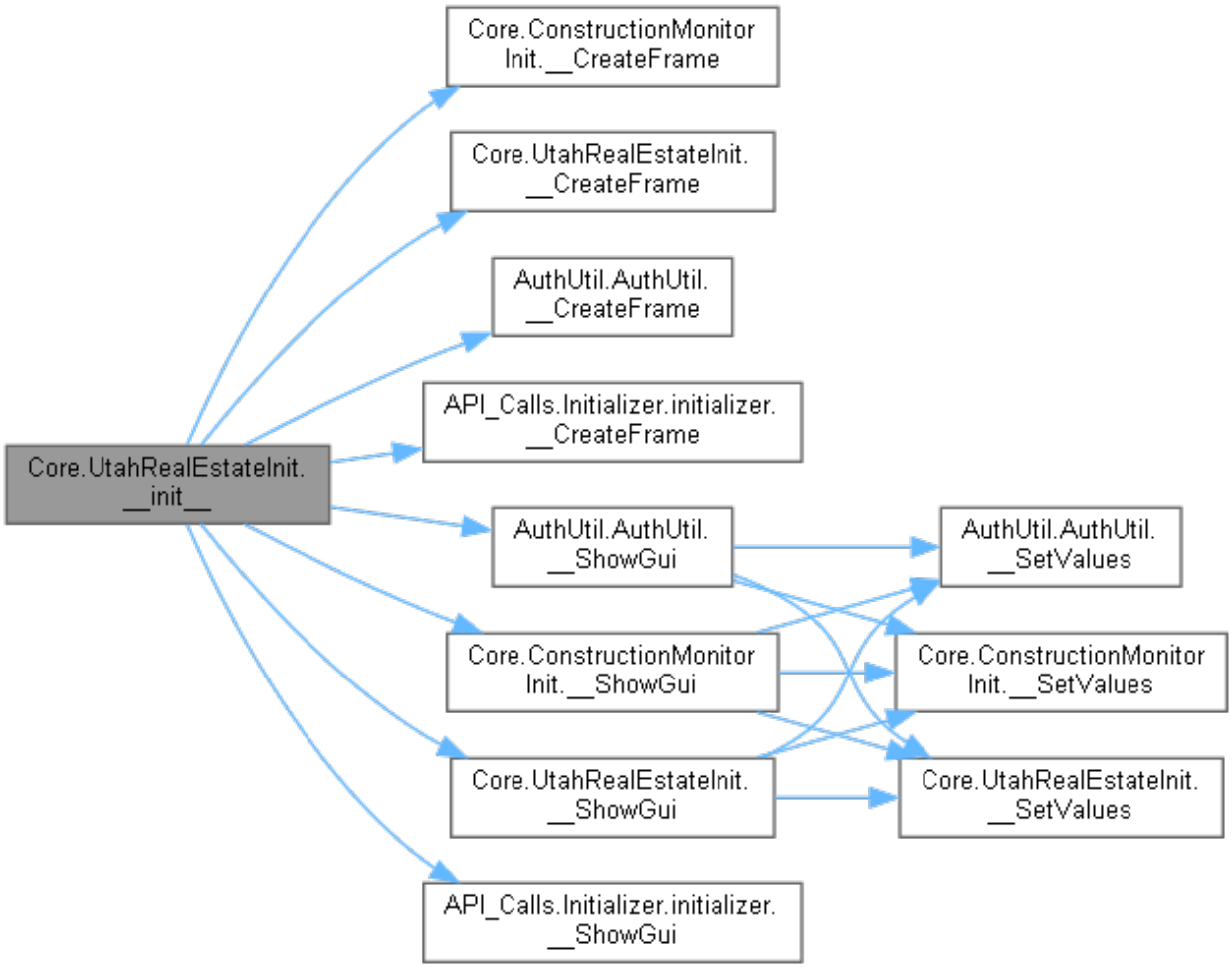
- def __CreateFrame ()

Constructor & Destructor Documentation

def Core.UtahRealEstateInit.__init__ (*self*)

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


Args:
self: Represent the instance of the class

Returns:
The __createframe function

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

Here is the call graph for this function:

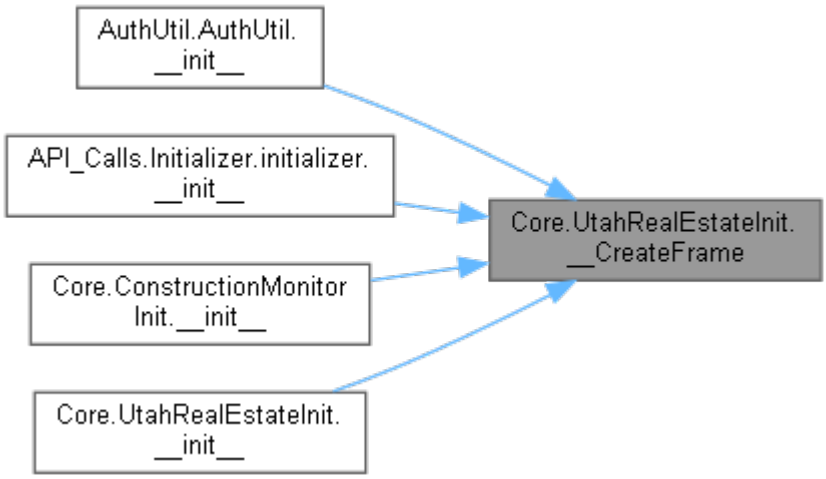


Member Function Documentation

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

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

Args:

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

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

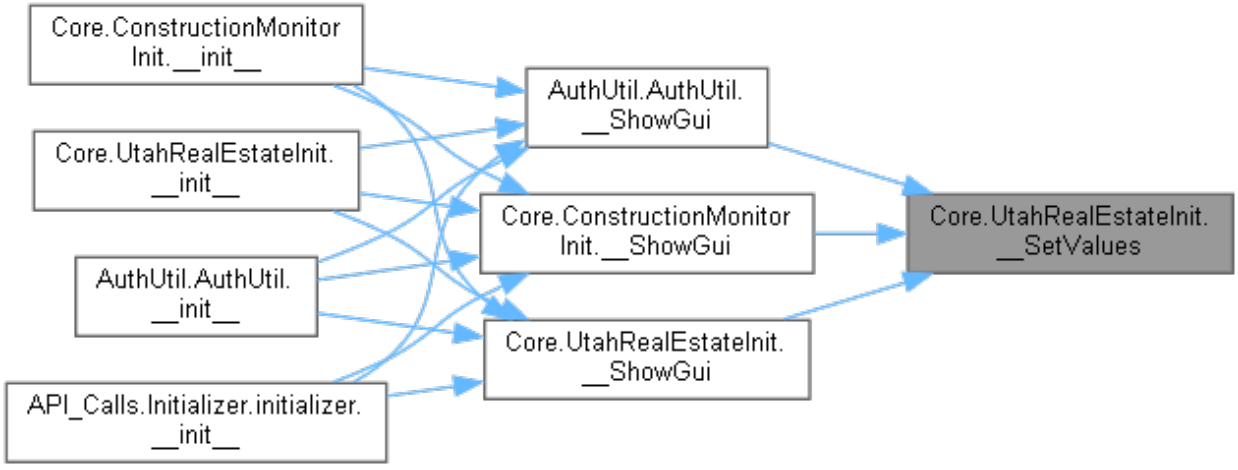
Here is the caller graph for this function:



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

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

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

Returns:
A dictionary with the following keys:

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

Here is the caller graph for this function:

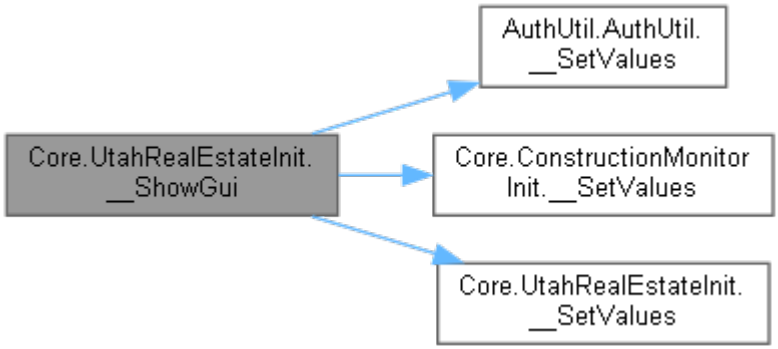


def Core.UtahRealEstateInit.__ShowGui (*self*, *layout*, *text*) `[private]`

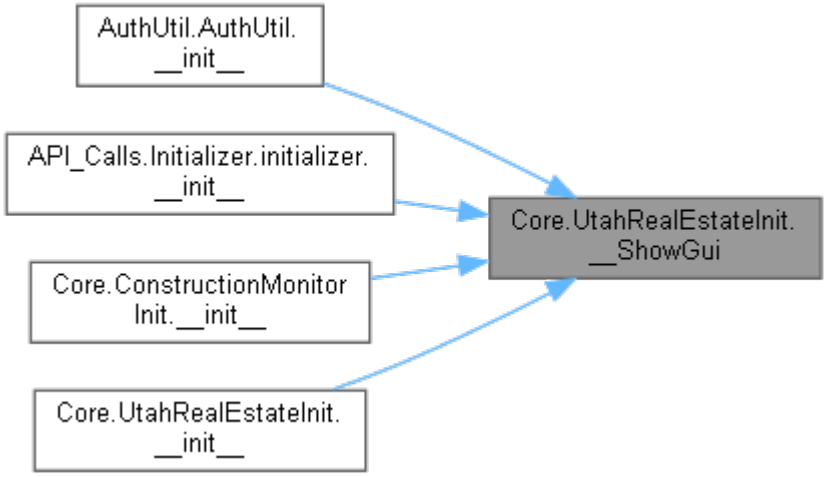
```
The __ShowGui function is a helper function that creates the GUI window and displays
it to the user.
It takes in two parameters: layout, which is a list of lists containing all the elements
for each row;
and text, which is a string containing what will be displayed as the title of the window.
The __ShowGui
```

```
method then uses these parameters to create an instance of sg.Window with all its
attributes set accordingly.

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

Returns:
A dictionary of values

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

- UtahRealEstate/Core.py

Core.UtahRealEstateMain Class Reference

Public Member Functions

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

Public Attributes

- **dataframe**
- **keyPath**
- **filePath**
- **key**

Private Member Functions

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

Private Attributes

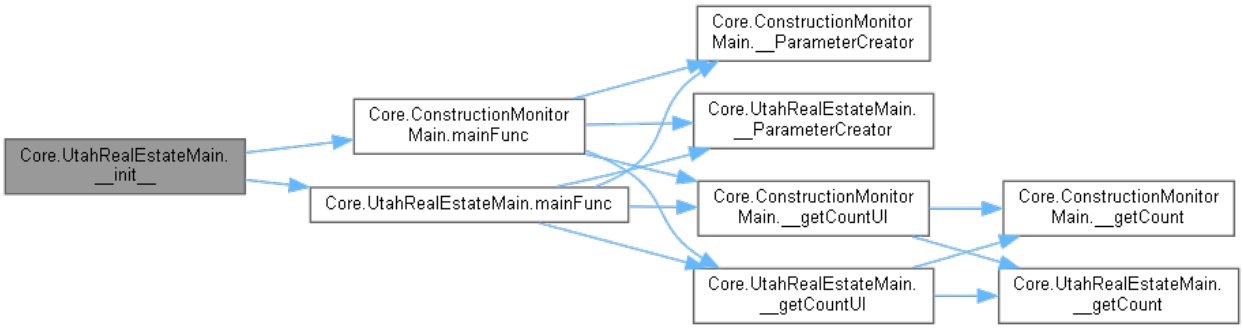
- **__batches**
- **__siteClass**
- **__headerDict**
- **__parameterString**
- **__appendFile**
- **__dateStart**
- **__dateEnd**
- **__restDomain**
- **__record_val**

Constructor & Destructor Documentation

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

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

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

Returns:
Nothing

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

Here is the call graph for this function:

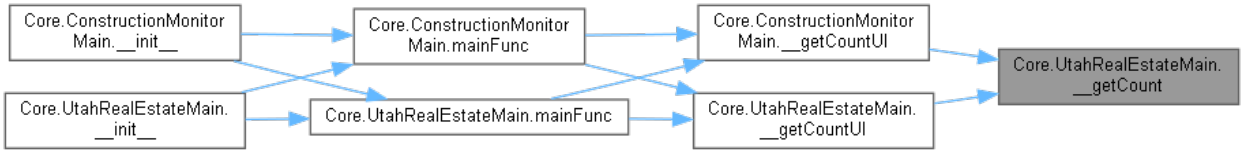


Member Function Documentation

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

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

Args:
self: Represent the instance of the class

Returns:
The number of records in the data set

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

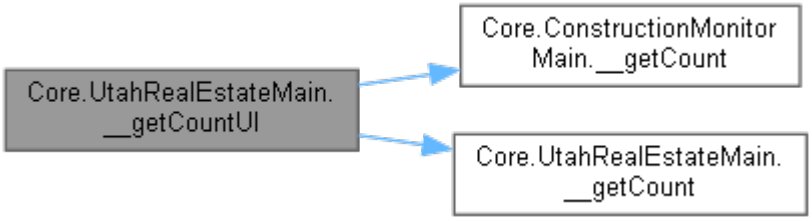
Here is the caller graph for this function:



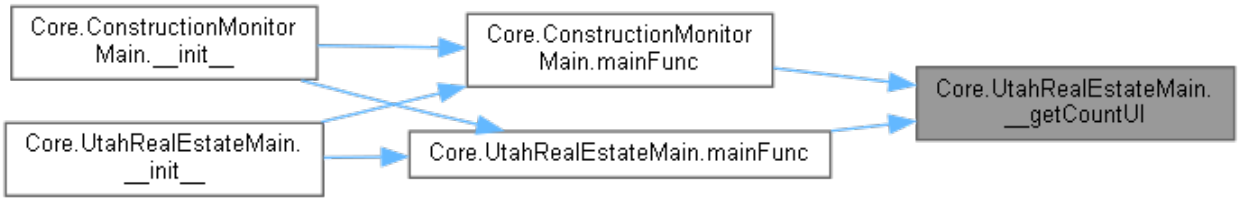
def Core.UtahRealEstateMain.__getCountUI (*self*) `[private]`

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

Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

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

Here is the call graph for this function:



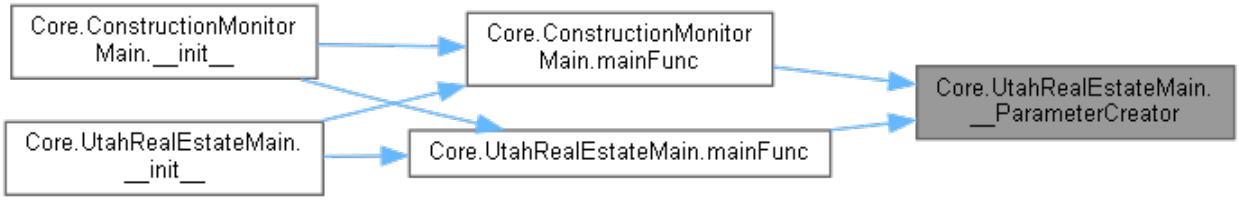
Here is the caller graph for this function:



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

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

Args:
self: Bind the object to the class

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

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

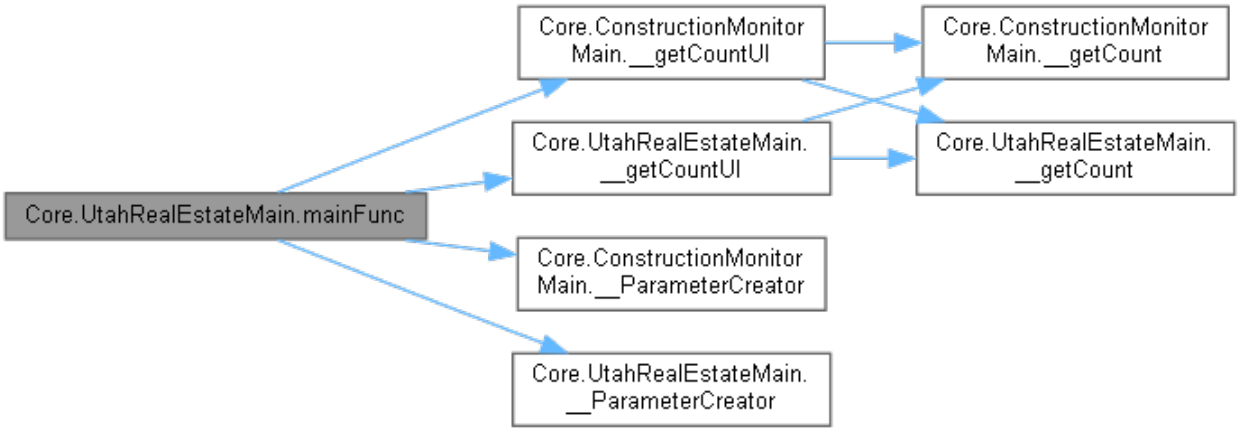
Here is the caller graph for this function:



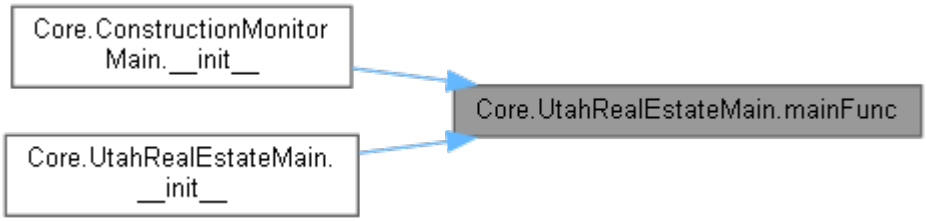
def Core.UtahRealEstateMain.mainFunc (*self*)

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

Args:
self: Reference the object itself

Returns:
A dataframe

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

- UtahRealEstate/Core.py

Index