Badger

Project: Badger

This file has been automatically generated. Please do not edit it

API Reference

Badger.Data.Caliburnutility

Class Badger.Data.CaliburnUtility

Source: CaliburnUtility.cs

Methods

```
void ShowPopupWindow(PropertyChangedBase viewModel, string windowHeader, bool isDialog = true)
```

- **Summary**

  Show a pop-up window, can be a dialog, which once showed up does not allow interaction with the background window. It also can be an independent window, which does allow interaction with any other window of the application.

- **Parameters**

  - *viewModel*: ViewModel to be shown in the pop-up window
  - *windowHeader*: Title of the window.
  - *isDialog*: Whether its a dialog or a window.

`string SelectFolder(string initialDirectory)`

- **Summary**

  Show an emergent dialog to allow users to select directory paths.

- **Parameters**

  - *initialDirectory*: Where everything starts.

- **Return Value**

  Show an emergent dialog to allow users to select directory paths.

Badger.Data.Logfileutils

Class Badger.Data.LogFileUtils

Source: LogFileUtils.cs

Methods

`Series GetVariableData(Log.EpisodesData episode, Report trackParameters, int variableIndex)`

- **Summary**

  Gets the data of a variable from an episode using the parameters of the target track

- **Parameters**

  - *episode*: The episode
  - *trackParameters*: The track parameters
  - *variableIndex*: Index of the variable

- **Return Value**

  Gets the data of a variable from an episode using the parameters of the target track

`double GetEpisodeAverage(Log.EpisodesData episode, int variableIndex, Report trackParameters)`

- **Summary**

  Gets the average value of a variable in an episode using the track parameters

- **Parameters**

  - *episode*: The episode
  - *variableIndex*: Index of the variable
  - *trackParameters*: The track parameters

- **Return Value**

  Gets the average value of a variable in an episode using the track parameters

`SeriesGroup GetAveragedData(List episodes, Report trackParameters, int variableIndex)`

- **Summary**

  Gets the averaged data of the given variable from a list of episodes using the track parameters

- **Parameters**

  - *episodes*: The episode list
  - *trackParameters*: The track parameters
  - *variableIndex*: Index of the variable

- **Return Value**

  Gets the averaged data of the given variable from a list of episodes using the track parameters

`Track LoadTrackData(LoggedExperimentalUnitViewModel expUnit, List reports)`

- **Summary**

  Creates a Track object from a logged experimental unit and a list of reports

- **Parameters**

- *expUnit*: The logged experimental unit
- *reports*: The list of reports we want

- **Return Value**

  Creates a Track object from a logged experimental unit and a list of reports

Badger.Data.Trackgroup

Class Badger.Data.TrackGroup

Source: TrackGroup.cs

Methods

`void Consolidate(string inGroupSelectionFunction, string inGroupSelectionVariable, string inGroupSelectionReportType

, BindableCollection groupBy)`

- **Summary**

  When grouping tracks by a fork, this function must be called to select a track inside the group. We call this "consolidating" the track group.

- **Parameters**
  - *inGroupSelectionFunction*: The function used to compare tracks inside the group
  - *inGroupSelectionVariable*: The variable used to evaluate tracks
  - *groupBy*: The list of forks used to group tracks

Badger.Files

Class Badger.Files

Source: Files.cs

Methods

`int SaveExperimentBatchFile(BindableCollection experiments,

string batchFilename, LogFunction log, ProgressUpdateFunction progressUpdateFunction)`

- **Summary**

  Save an experiment batch: the list of (possibly forked) experiments is saved a as set of experiments without forks and a .exp-batch file in the root directory referencing them all and the forks/values each one took.

- **Parameters**
  - *experiments*:
  - *batchFilename*:
  - *log*:

- **Return Value**

  Save an experiment batch: the list of (possibly forked) experiments is saved a as set of experiments without forks and a .exp-batch file in the root directory referencing them all and the forks/values each one took.

```
SaveFileDialog SaveFileDialog(string description, string filter, string suggestedFileName= null)
```

- **Summary**

  Show a dialog used to save a file with an specific extension.

- **Parameters**
  - *description*: Short description of the file type
  - *filter*: Extension
  - *suggestedFileName*: Name suggested for output. Null by default

- **Return Value**

  Show a dialog used to save a file with an specific extension.

`string SelectOutputDirectoryDialog(string initialDirectory= null)`

- **Summary**

  Shows a dialog used to select a directory where files are to be saved. If something goes wrong, null is returned

- **Parameters**

  - *initialDirectory*: The directory from which to being browsing

- **Return Value**

  Shows a dialog used to select a directory where files are to be saved. If something goes wrong, null is returned

`bool OpenFileDialog(ref string fileName, string description, string extension)`

- **Summary**

  Open a file that fulfills the requirements passed as parameters.

- **Parameters**

  - *fileName*: The name of the file
  - *description*: Description of the file type
  - *extension*: The extension of the file type

- **Return Value**

  Open a file that fulfills the requirements passed as parameters.

Badger.Viewmodels.Confignodeviewmodel

Class Badger.ViewModels.ConfigNodeViewModel

Source: ConfigNodeViewModel.cs

Methods

`void ForkThisNode(ConfigNodeViewModel originNode)`

- **Summary**

  Forks this node

- **Parameters**

  - *originNode*: The origin node.

`void LinkThisNode(ConfigNodeViewModel originNode)`

- **Summary**

  Take the right-clicked node as the origin node to link with all the posible linkable nodes (i.e. nodes of the same class). Linkable nodes CanBeLinked property value are set to true.

- **Parameters**

  - *originNode*: The origin node of the linking process

`void CancelLinking(ConfigNodeViewModel originNode)`

- **Summary**

  Cancel a linking process between two nodes.

`void Link(ConfigNodeViewModel targetNode)`

- **Summary**

  Actually perform the linking with the node.

- **Parameters**

  - *targetNode*:

`void UnlinkNode()`

- **Summary**

  Unlink the node removing it from its origin linked nodes list and restore it to its original node class.

Badger.Viewmodels.Editorwindowviewmodel

Class Badger.ViewModels.EditorWindowViewModel

Source: EditorWindowViewModel.cs

Methods

`void NewExperiment()`

- **Summary**

  Creates a new experiment and adds it to the Editor tab

`void SaveSelectedExperimentOrProject()`

- **Summary**

  Saves the selected experiment or project

`void LoadExperimentalUnit(string experimentalUnitConfigFile)`

- **Summary**

  Loads an experimental unit on the Editor tab. This method can be used from any child window to load experimental units (i.e, from the ReportViewer)

- **Parameters**

  - *experimentalUnitConfigFile*: The experimental unit file.

`void LoadExperimentOrProject()`

- **Summary**

  Shows a dialog window where the user can select an experiment or project for loading

`void ClearExperiments()`

- **Summary**

  Clears the experiments tab

`void Close(ExperimentViewModel e)`

- **Summary**

  Close a tab (experiment view) and removes it from experiments list.

- **Parameters**

  - *e*: The experiment to be removed

`void RunExperimentalUnitLocallyWithCurrentParameters(ExperimentViewModel experiment)`

- **Summary**

  Runs locally the experiment with its currently selected parameters

- **Parameters**

  - *experiment*: The experiment to be run

`void ShowWires(ExperimentViewModel experiment)`

- **Summary**

  Shows the wires defined in the current experiment

- **Parameters**

  - *experiment*: The selected experiment

`void RunExperimentsInEditor()`

- **Summary**

  Runs all the experiments open in the editor. Saves a batch file read by the experiment monitor, and also a project to be able to modify the experiment and rerun it later

Badger.Viewmodels.Experimentviewmodel

Class Badger.ViewModels.ExperimentViewModel

Source: ExperimentViewModel.cs

Methods

```
public ExperimentViewModel(string appDefinitionFileName, string configFilename)
```

- **Summary**

  This constructor builds the whole tree of ConfigNodes either - with default values ("New") or - with a configuration file ("Load")

- **Parameters**

  - *appDefinitionFileName*:
  - *configFilename*:

```
void ShowWires()
```

- **Summary**

  Shows a new window with the wires used in the experiment

```
ConfigNodeViewModel DepthFirstSearch(string nodeName, string alias = "")
```

- **Summary**

  Implementation of depth first search algorithm for experiment tree.

- **Parameters**

  - *targetNode*:

```
ConfigNodeViewModel DepthFirstSearch(ConfigNodeViewModel targetNode)
```

- **Summary**

  Implementation of depth first search algorithm for experiment tree.

- **Parameters**

  - *targetNode*:

```
void CheckLinkableNodes(ConfigNodeViewModel originNode, bool link = true)
```

- **Summary**

- **Parameters**

  - *originNode*:
  - *link*:

Badger.Viewmodels.Functionlogviewmodel

Class Badger.ViewModels.FunctionLogViewModel

Source: FunctionLogViewModel.cs

Methods

```
void NextFunction()
```

- **Summary**

  Shows the first sample of the next function in the log

```
void NextSample()
```

- **Summary**

  Shows the next sample of the current function

```
void PreviousFunction()
```

- **Summary**

Shows the first sample of the previous the function in the log

```
void PreviousSample()
```

- **Summary**

  Shows the previous the sample of the current function

```
void ExportAll()
```

- **Summary**

  Exports all the function samples, each one in a different "png" file

Badger.Viewmodels.Herdagentselectionviewmodel

Class Badger.ViewModels.HerdAgentSelectionViewModel

Source: HerdAgentSelectionViewModel.cs

Methods

```
void AddInOrder(BindableCollection intList, int newInt)
```

- **Summary**

  Adds in order to a list of ints if input int is not on the list

- **Parameters**

  - *intList*: in-out list where the new integer is added
  - *newInt*: new integer to be added to the list

```
void AddInOrder(BindableCollection stringList, string newString)
```

- **Summary**

  Adds in order to a list of strings if input string is not on the list, assuming all strings are formatted using version numbers or ip addresses (xxx.xxx.xxx.xxx)

- **Parameters**

  - *stringList*: in-out string list
  - *newString*: new string to be added

Badger.Viewmodels.Herdagentviewmodel

Class Badger.ViewModels.HerdAgentViewModel

Source: HerdAgentViewModel.cs

Methods

```
bool IsLocalIpAddress(string host)
```

- **Summary**

  Determines whether an IP address is local

- **Parameters**

  - *host*: The IP address

Badger.Viewmodels.Linkednodeviewmodel

Class Badger.ViewModels.LinkedNodeViewModel

Source: LinkedNodeViewModel.cs

Methods

`public LinkedNodeViewModel(ExperimentViewModel parentExperiment, ConfigNodeViewModel originNode,

ConfigNodeViewModel targetNode)`

- **Summary**

  Constructor used from the experiment editor

- **Parameters**
  - *parentExperiment*:
  - *originNode*:
  - *targetNode*:

`public LinkedNodeViewModel(ExperimentViewModel parentExperiment, ConfigNodeViewModel parentNode,

XmlNode classDefinition, XmlNode configNode = null)`

- **Summary**

  Constructor called when loading an experiment config file

- **Parameters**
  - *parentExperiment*:
  - *parentNode*:
  - *classDefinition*: Class of the node in app definitions
  - *parentXPath*:
  - *configNode*: Node in simion.exp file with the configuration for a node class

`void CreateLinkedNode()`

- **Summary**

Badger.Viewmodels.Loggedexperimentalunitviewmodel

Class Badger.ViewModels.LoggedExperimentalUnitViewModel

Source: LoggedExperimentalUnitViewModel.cs

Methods

`public LoggedExperimentalUnitViewModel(string filename)`

- **Summary**

  Fake constructor for testing purposes

- **Parameters**
  - *filename*: path to the experimental unit

`public LoggedExperimentalUnitViewModel(ExperimentalUnit model)`

- **Summary**

  Main constructor

- **Parameters**
  - *configNode*:
  - *baseDirectory*:
  - *updateFunction*:

`int GetVariableIndex(string variableName)`

- **Summary**

  Gets the index of a variable

- **Parameters**
  - *variableName*: Name of the variable

- **Return Value**

  Gets the index of a variable

Badger.Viewmodels.Loggedexperimentviewmodel

Class Badger.ViewModels.LoggedExperimentViewModel

Source: LoggedExperimentViewModel.cs

Methods

```
public LoggedExperimentViewModel(Experiment experiment)
```

- **Summary**

  Class constructor.

- **Parameters**

  - *experiment*: The experiment with all the data used in the view model

```
void AddVariable(string variableName)
```

- **Summary**

  Call after reading the log file descriptor of each experimetal unit

- **Parameters**

  - *variableName*:

Badger.Viewmodels.Loggedforkviewmodel

Class Badger.ViewModels.LoggedForkViewModel

Source: LoggedForkViewModel.cs

Methods

```
void GroupByThisFork()
```

- **Summary**

  Method is called from the context menu informs the parent window that results should be grouped by this fork.

Badger.Viewmodels.Loggedvariableviewmodel

Class Badger.ViewModels.LoggedVariableViewModel

Source: LoggedVariableViewModel.cs

Methods

```
void SetNotifying(bool notifying)
```

- **Summary**

  Sets the notifying property. Needs to be used after loading a view model from a file

- **Parameters**

  - *notifying*: the value to be set

Badger.Viewmodels.Logqueryresultviewmodel

Class Badger.ViewModels.LogQueryResultViewModel

Source: LogQueryResultViewModel.cs

Methods

```
void ImportNonSerializable(string inputBaseFolder)
```

- **Summary**

  Imports non serializable data

- **Parameters**
  - *inputBaseFolder*: The input base folder.

`void SetNotifying(bool notifying)`

- **Summary**

  Sets the notifying property. Needs to be set after loading serialized data

- **Parameters**
  - *notifying*: The value set

Badger.Viewmodels.Logqueryviewmodel

Class Badger.ViewModels.LogQueryViewModel

Source: LogQueryViewModel.cs

Methods

`void AddGroupByFork(string forkName)`

- **Summary**

  Adds the fork to the list of group-by forks

- **Parameters**
  - *forkName*: Name of the fork.

`bool IsForkUsedToGroup(string forkName)`

- **Summary**

  Returns whether a fork is used to group tracks or not

- **Parameters**
  - *forkName*: Name of the fork

`void ResetGroupBy()`

- **Summary**

  Resets the forks used to group tracks

`string GetVariableProcessFunc(string variable)`

- **Summary**

  Gets the process function used for the variable

- **Parameters**
  - *variable*: The variable.

- **Return Value**

  Gets the process function used for the variable

`bool IsVariableSelected(string variable, string reportType)`

- **Summary**

  Returns whether the specified variable-reportType is selected

- **Parameters**
  - *variable*: The variable
  - *reportType*: Type of the report

`void AddLogVariables(List variables)`

- **Summary**

  Adds the variables to the list of variables in the log. Called when loading a logged experimental unit

- **Parameters**

- *variables*: The variables

`bool LogVariableExists(string variable)`

- **Summary**

  Returns whether the variable exists in th elog or not

- **Parameters**
  - *variable*: The variable.

`void Validate()`

- **Summary**

  This function validates the current query and sets CanGenerateReports accordingly

`void Execute(BindableCollection experiments

,LoadOptions.PerExperimentalUnitFunction OnExpUnitProcessed, out List resultTracks, out List reports)`

- **Summary**

  Executes the specified query.

- **Parameters**
  - *experiments*: The list of logged experiments on which the query will be processed
  - *OnExpUnitProcessed*: Callback function called when an exp unit is processed. Used to update the progress
  - *resultTracks*: Output list of track groups
  - *reports*: Output list of reports

Badger.Viewmodels.Mainwindowviewmodel

Class Badger.ViewModels.MainWindowViewModel

Source: MainWindowViewModel.cs

Methods

`public MainWindowViewModel()`

- **Summary**

  Class constructor.

`void ShowReportWindow()`

- **Summary**

  Show the report viewer tab

`void ShowExperimentMonitor()`

- **Summary**

  Shows the experiment monitor tab

`void ShowEditorWindow()`

- **Summary**

  Shows the editor tab

`void LogToFile(string logMessage)`

- **Summary**

  Logs a message with the current time-date. If the Badger log file doesn't exist, it creates it. Uses a lock to avoid multi-thread issues

- **Parameters**
  - *logMessage*: The log message.

Badger.Viewmodels.Monitoredexperimentalunitviewmodel

Class Badger.ViewModels.MonitoredExperimentalUnitViewModel

Source: MonitoredExperimentalUnitViewModel.cs

Methods

```
public MonitoredExperimentalUnitViewModel(ExperimentalUnit expUnit, PlotViewModel plot)
```

- **Summary**

  Constructor

- **Parameters**

  - *expUnit*: The model: an instance of ExperimentalUnit
  - *plot*: The plot used to monitor data

```
void AddEvaluationValue(double xNorm, double y)
```

- **Summary**

  Adds a (xNorm,y) value to the series of evaluations.

- **Parameters**

  - *xNorm*: The normalized value in x (0 is the beginning and 1 the end of the experiment)
  - *y*: The average reward obtained in this evaluation

Badger.Viewmodels.Monitoredjobviewmodel

Class Badger.ViewModels.MonitoredJobViewModel

Source: MonitoredJobViewModel.cs

Methods

```
void OnMessageReceived(string experimentId, string messageId, string messageContent)
```

- **Summary**

  Callback method that is called from the job dispatcher when a message is received

- **Parameters**

  - *experimentId*: The experiment identifier
  - *messageId*: The message identifier
  - *messageContent*: Content of the message

```
void OnStateChanged(string experimentId, Monitoring.State state)
```

- **Summary**

  Called method executed when the state of an experimental unit changes

- **Parameters**

  - *experimentId*: The experiment identifier
  - *state*: The state

```
void OnAllStatesChanged(Monitoring.State state)
```

- **Summary**

  Callback method called when the state of all the experimental unit in a job changes

- **Parameters**

  - *state*: The state.

```
void OnExperimentalUnitLaunched(ExperimentalUnit expUnit)
```

- **Summary**

  Callback method executed when an experimental unit is launched

- **Parameters**

- *expUnit*: The expererimental unit

Badger.Viewmodels.Monitorwindowviewmodel

Class Badger.ViewModels.MonitorWindowViewModel

Source: MonitorWindowViewModel.cs

Methods

`public MonitorWindowViewModel()`

- **Summary**

  Constructor.

- **Parameters**

  - *MainWindowViewModel.Instance.LogToFile*:
  - *batchFileName*:

`void CleanExperimentBatch()`

- **Summary**

  Deletes all the log files in the batch

`void RunExperimentBatch()`

- **Summary**

  Runs the selected experiment in the experiment editor.

`void ShowReports()`

- **Summary**

  Shows a Report window with the data of the currently finished experiment(s) already load and ready to make reports.

`void SelectExperimentBatchFile()`

- **Summary**

  Shows a dialog to select which experiment batch file to load and loads it

`bool LoadExperimentBatch(string batchFileName)`

- **Summary**

  Initializes the experiments to be monitored through a batch file that contains all required data for the task.

- **Parameters**

  - *batchFileName*: The batch file with experiment data

`void DispatchOnMessageReceived(Job job, string experimentId, string messageId, string messageContent)`

- **Summary**

  Callback method on message received. It dispatches it to the job view model of the job passes as argument

- **Parameters**

  - *job*: The job
  - *experimentId*: The identifier of the experimental unit
  - *messageId*: The message identifier
  - *messageContent*: Content of the message

`void DispatchOnStateChanged(Job job, string experimentId, Monitoring.State state)`

- **Summary**

  Callback method executed when the state of an experimental unit changes. This method dispatches it to the correct job view model handling that job

- **Parameters**

  - *job*: The job

- *experimentId*: The experiment identifier
- *state*: The state

`void DispatchOnAllStatesChanged(Job job, Monitoring.State state)`

- **Summary**

  Callback method called when all the states of a job change. It dispatches the change to the correct job view model handling that job

- **Parameters**
  - *job*: The job
  - *state*: The state

`void DispatchOnExperimentalUnitLaunched(Job job, ExperimentalUnit expUnit)`

- **Summary**

  Callback method on experimental unit launched that dispatches the event to the correct job view model

- **Parameters**
  - *job*: The job
  - *expUnit*: The exp unit

`void OnJobAssigned(Job job)`

- **Summary**

  Callback method used to inform the monitor window view model when that a job has been assigned

- **Parameters**
  - *job*: The job.

`void OnJobFinished(Job job)`

- **Summary**

  Callback method executed when a job is finished

- **Parameters**
  - *job*: The job.

`void SetRunning(bool running)`

- **Summary**

  Sets the state as running

`void RunExperimentsAsync(List freeHerdAgents)`

- **Summary**

  Async method that runs all the experiments using the free herd agents

- **Parameters**
  - *freeHerdAgents*: The free herd agents.

`bool ExistsRequiredFile(string file)`

- **Summary**

  Checks whether a required file to run an experiment exits or not.

- **Parameters**
  - *file*: The file to check

- **Return Value**

  Checks whether a required file to run an experiment exits or not.

`double CalculateGlobalProgress()`

- **Summary**

  Calculate the global progress of experiments in queue.

- **Return Value**

  Calculate the global progress of experiments in queue.

`void StopExperiments()`

- **Summary**

  Stops all experiments in progress.

Badger.Viewmodels.Plotpropertiesviewmodel

Class Badger.ViewModels.PlotPropertiesViewModel

Source: PlotPropertiesViewModel.cs

Methods

`void HighlightSeries(int seriesId)`

- **Summary**

  Highlight a series

- **Parameters**

  - *seriesId*:

`void DimLineSeriesColor(PlotLineSeriesPropertiesViewModel lineSeriesProperties)`

- **Summary**

  Apply some opacity to the original color of the LineSeries.

- **Parameters**

  - *lineSeriesProperties*:

`void ResetLineSeriesOpacity(PlotLineSeriesPropertiesViewModel lineSeriesProperties)`

- **Summary**

  Restore the original color of the LineSeries.

- **Parameters**

  - *lineSeriesProperties*:

Badger.Viewmodels.Plotviewmodel

Class Badger.ViewModels.PlotViewModel

Source: PlotViewModel.cs

Methods

`void ResetAxes()`

- **Summary**

  Resets the axes of the plot to the default range [0,1]

`void InitPlot(string title, string xAxisName, string yAxisName)`

- **Summary**

  Initializes the plot

- **Parameters**

  - *title*: The title
  - *xAxisName*: Name of the x axis
  - *yAxisName*: Name of the y axis

`int AddLineSeries(string title, string description ="", bool isVisible = true)`

- **Summary**

  Adds a line series to the plot

- **Parameters**

  - *title*: The title of the series
  - *description*: The description of the series
  - *isVisible*: Initial visibility given to the series

- **Return Value**

  Adds a line series to the plot

`void AddLineSeriesValue(int seriesIndex, double xValue, double yValue)`

- **Summary**

  Adds a vale to a given line series

- **Parameters**

  - *seriesIndex*: Index of the series
  - *xValue*: The x value
  - *yValue*: The y value

`void ClearLineSeries()`

- **Summary**

  Clears the line series.

`void HighlightLineSeries(int seriesId)`

- **Summary**

  Identify which LineSeries is hovered and make a call to the dimLineSeriesColor method passing the correct LineSeriesProperties object as parameter. In order to highlight a LineSeries what we actually do is to dim, that is, apply certain opacity, to all the other LineSeries.

- **Parameters**

  - *seriesId*: Id of the LineSeries to be highlighted

`void ResetLineSeriesColors()`

- **Summary**

  Reset all LineSeries color to its original, removing the opacity in case that some was applied before by the highlightLineSeries method.

Badger.Viewmodels.Shepherdviewmodel

Class Badger.ViewModels.ShepherdViewModel

Source: ShepherdViewModel.cs

Methods

`int GetAvailableHerdAgents(ref List outList)`

- **Summary**

  Gets the available herd agents.

- **Parameters**

  - *outList*: The out list where the herd agents are added.

- **Return Value**

  Gets the available herd agents.

`void SelectHerdAgents()`

- **Summary**

  Shows a pop-up window where the user to select/deselect herd agents

`void ConfigureJobDispatcher()`

- **Summary**

  Shows a pop-up window where the user can configure the job dispatcher