Sparksee

6.0.2

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

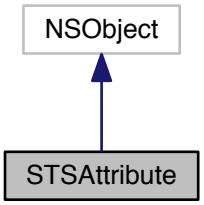
Here are the classes, structs, unions and interfaces with brief descriptions:

3 Class Documentation

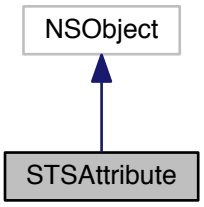
3.1 STSAttribute Class Reference

Attribute data class.

Inheritance diagram for STSAttribute:



Collaboration diagram for STSAttribute:



Instance Methods

- (int) - getId
  Gets the Sparksee attribute identifier.
- (int) - getTypeId
  Gets the Sparksee type identifier.
- (NSString ∗) - getName
  Gets the unique attribute name.
- (enum STSDataType) - getDataType
  Gets the data type.
- (long long) - getSize
  Gets the number of different values.
- (long long) - getCount
  Gets the number of non-NULL values.
- (enum STSAttributeKind) - getKind
  Gets the attribute kind.
- (BOOL) - isSessionAttribute
  Check if it's a session attribute or a persistent one.
- (BOOL) - isArrayAttribute
  Check if it's an array attribute.
- (int) - getArraySize
  Gets the number of elements in the array.

Class Methods

- (int) + getInvalidAttribute

  Invalid attribute identifier constant.

3.1.1 Detailed Description

Attribute data class.

It contains information about an attribute.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.1.2 Method Documentation

3.1.2.1 - (int) getArraySize

Gets the number of elements in the array.

Returns

The size of the array

3.1.2.2 - (long long) getCount

Gets the number of non-NULL values.

Returns

The number of non-NULL values.

3.1.2.3 - (enum STSDataType) getDataType

Gets the data type.

Returns

The DataType.

3.1.2.4 - (int) getId

Gets the Sparksee attribute identifier.

Returns

The Sparksee attribute identifier.

3.1.2.5 - (enum STSAttributeKind) getKind

Gets the attribute kind.

Returns

The AttributeKind.

3.1.2.6 - (NSString∗) getName

Gets the unique attribute name.

Returns

The unique attribute name.

3.1.2.7 - (long long) getSize

Gets the number of different values.

Returns

The number of different values.

3.1.2.8 - (int) getTypeId

Gets the Sparksee type identifier.

Returns

The Sparksee type identifier.

3.1.2.9 - (BOOL) isArrayAttribute

Check if it's an array attribute.

Returns

True if it's an array attribute, or false otherwise.

3.1.2.10 - (BOOL) isSessionAttribute

Check if it's a session attribute or a persistent one.

Returns

True if it's a session attribute, or false otherwise.

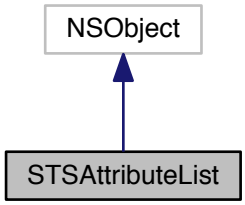
The documentation for this class was generated from the following file:

- Sparksee.h

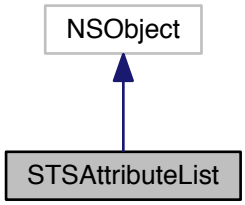
3.2 STSAttributeList Class Reference

Sparksee attribute identifier list.

Inheritance diagram for STSAttributeList:



Collaboration diagram for STSAttributeList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - add:

  Adds a Sparksee attribute identifier at the end of the list.
- (void) - clear

  Clears the list.
- (id) - initWithArray:

  Creates a new AttributeList instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new AttributeList instance from the given NSEnumerator.
- (STSAttributeListIterator ∗) - iterator

  Gets a new AttributeListIterator.

3.2.1 Detailed Description

Sparksee attribute identifier list.

It stores a Sparksee attribute identifier list.

Use AttributeListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.2.2 Method Documentation

3.2.2.1 - (void) add: (int) *attr*

Adds a Sparksee attribute identifier at the end of the list.

Parameters

attr	[in] Sparksee attribute identifier.

3.2.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.2.2.3 - (id) init

Constructor.

This creates an empty list.

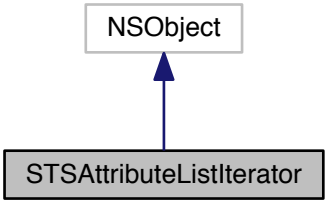
The documentation for this class was generated from the following file:

- Sparksee.h

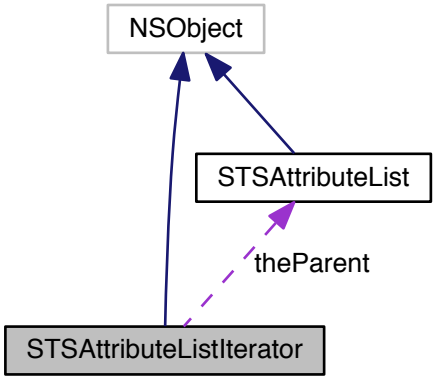
3.3 STSAttributeListIterator Class Reference

AttributeList iterator class.

Inheritance diagram for STSAttributeListIterator:



Collaboration diagram for STSAttributeListIterator:



Instance Methods

- (int) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.3.1 Detailed Description

AttributeList iterator class.

Iterator to traverse all the Sparksee attribute identifier into a AttributeList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.3.2 Method Documentation

3.3.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.3.2.2 - (int) next

Moves to the next element.

Returns

The next element.

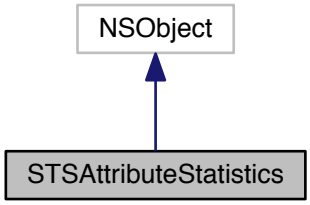
The documentation for this class was generated from the following file:

- Sparksee.h

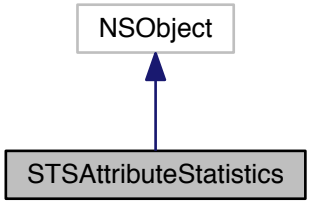
3.4 STSAttributeStatistics Class Reference

Attribute statistics class.

Inheritance diagram for STSAttributeStatistics:



Collaboration diagram for STSAttributeStatistics:



Instance Methods

- (long long) - getTotal

  Gets the number of objects with a non-NULL Value (BASIC statistic).
- (long long) - getNull

  Gets the number of objects NULL a Value (BASIC statistics).
- (long long) - getDistinct

  Gets the number of distinct values (BASIC statistics).
- (STSValue ∗) - getMin

  Gets the minimum existing value (BASIC statistics).
- (STSValue ∗) - getMax

  Gets the maximum existing value (BASIC statistics).
- (int) - getMaxLengthString

  Gets the maximum length.
- (int) - getMinLengthString

  Gets the minimum length.
- (double) - getAvgLengthString

  Gets the average length.
- (STSValue ∗) - getMode

  Gets the mode.
- (long long) - getModeCount

  Gets the number of objects with a Value equal to the mode.
- (double) - getMean

  Gets the mean or average.
- (double) - getVariance

  Gets the variance.
- (double) - getMedian

  Gets the median.

3.4.1 Detailed Description

Attribute statistics class.

It contains statistic data about an attribute.

Some fields are valid just for numerical attributes and others just for string attributes. Also, some statistics are considered BASIC because computing them do not require to traverse all the different values of the attribute. For each getter method the documentation tells if the statistic is BASIC or not. See the Graph class method get↩ AttributeStatistics or check out the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.4.2 Method Documentation

3.4.2.1 - (double) getAvgLengthString

Gets the average length.

If the attribute is not an string attribute, it just returns 0.

Returns

The average length.

3.4.2.2 - (long long) getDistinct

Gets the number of distinct values (BASIC statistics).

Returns

The number of distinct values.

3.4.2.3 - (STSValue*) getMax

Gets the maximum existing value (BASIC statistics).

Returns

The maximum existing value.

3.4.2.4 - (int) getMaxLengthString

Gets the maximum length.

If the attribute is not an string attribute, it just returns 0.

Returns

The maximum length.

3.4.2.5 - (double) getMean

Gets the mean or average.

Mean or average: Sum of all Values divided by the number of observations.

It is computed just for numerical attributes.

Returns

The mean.

3.4.2.6 - (double) getMedian

Gets the median.

Median: Middle value that separates the higher half from the lower.

If a $<$ b $<$ c, then the median of the list {a, b, c} is b, and if a $<$ b $<$ c $<$ d, then the median of the list {a, b, c, d} is the mean of b and c, i.e. it is (b + c)/2

It is computed just for numerical attributes.

Returns

The median.

3.4.2.7 - (STSValue*) getMin

Gets the minimum existing value (BASIC statistics).

Returns

The minimum existing value.

3.4.2.8 - (int) getMinLengthString

Gets the minimum length.

If the attribute is not an string attribute, it just returns 0.

Returns

The minimum length.

3.4.2.9 - (STSValue*) getMode

Gets the mode.

Mode: Most frequent Value.

Returns

The mode.

3.4.2.10 - (long long) getModeCount

Gets the number of objects with a Value equal to the mode.

Returns

The number of objects with a Value equal to the mode.

3.4.2.11 - (long long) getNull

Gets the number of objects NULL a Value (BASIC statistics).

Returns

The number of objects NULL a Value.

3.4.2.12 - (long long) getTotal

Gets the number of objects with a non-NULL Value (BASIC statistic).

Returns

The number of objects with a non-NULL Value.

3.4.2.13 - (double) getVariance

Gets the variance.

It is computed just for numerical attributes.

Returns

The variance.

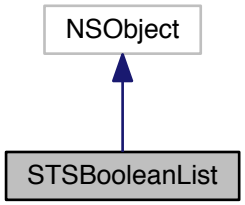
The documentation for this class was generated from the following file:

- Sparksee.h

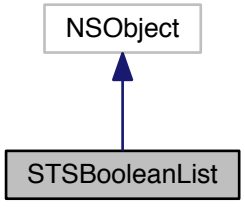
3.5 STSBooleanList Class Reference

Boolean list.

Inheritance diagram for STSBooleanList:



Collaboration diagram for STSBooleanList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - add:

  Adds a Boolean at the end of the list.
- (void) - clear

  Clears the list.
- (id) - initWithArray:

  Creates a new BooleanList instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new BooleanList instance from the given NSEnumerator.
- (STSBooleanListIterator ∗) - iterator

  Gets a new BooleanListIterator.

3.5.1 Detailed Description

Boolean list.

It stores a Boolean list.

Use BooleanListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.5.2 Method Documentation

3.5.2.1 - (void) add: (BOOL) *value*

Adds a Boolean at the end of the list.

Parameters

value	[in] Boolean.

3.5.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.5.2.3 - (id) init

Constructor.

This creates an empty list.

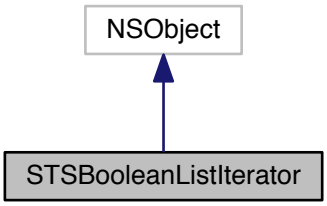
The documentation for this class was generated from the following file:

- Sparksee.h

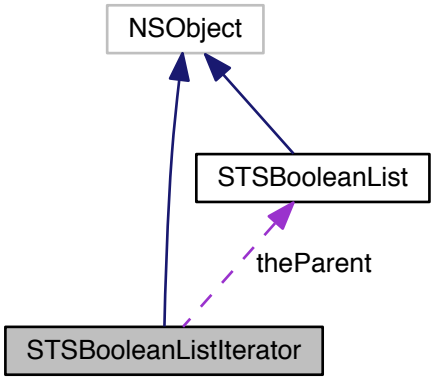
3.6 STSBooleanListIterator Class Reference

BooleanList iterator class.

Inheritance diagram for STSBooleanListIterator:



Collaboration diagram for STSBooleanListIterator:



Instance Methods

- (BOOL) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.6.1 Detailed Description

BooleanList iterator class.

Iterator to traverse all the strings into a BooleanList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.6.2 Method Documentation

3.6.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.6.2.2 - (BOOL) next

Moves to the next element.

Returns

The next element.

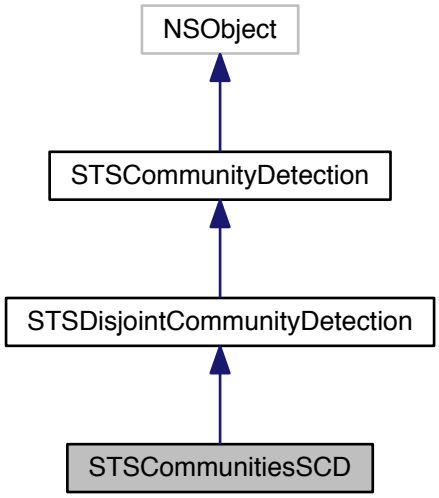
The documentation for this class was generated from the following file:

- Sparksee.h

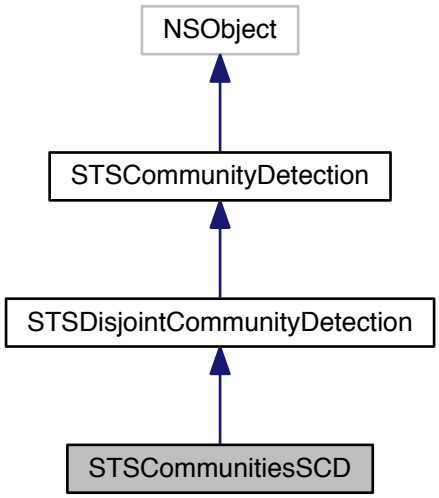
3.7 STSCommunitiesSCD Class Reference

CommunitiesSCD class.

Inheritance diagram for STSCommunitiesSCD:



Collaboration diagram for STSCommunitiesSCD:



Instance Methods

- (id) - initWithSession:

  Creates a new instance of CommunitiesSCD.
- (void) - setLookAhead:

  Sets the size of the lookahead iterations to look (5 by default).
- (void) - run

  Executes the algorithm.
- (void) - addEdgeType:

  Allows connectivity through edges of the given type.
- (void) - addAllEdgeTypes

  Allows connectivity through all edge types of the graph.
- (STSDisjointCommunities *) - getCommunities

  Returns the results generated by the execution of the algorithm.
- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.
- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - includeNodes:

  Set additional nodes that can be used.
- (void) - includeEdges:

  Set additional edges that can be used.
- (void) - close

  Closes the CommunityDetection instance.
- (BOOL) - isClosed

  Check if the CommunityDetection instance is closed.

3.7.1 Detailed Description

CommunitiesSCD class.

Implementation of the community detection algorithm "Scalable Community Detection" based on the paper "High quality, scalable and parallel community detection for large real graphs" by Arnau Prat-Perez, David Dominguez-Sal, Josep-Lluis Larriba-Pey - WWW 2014.

The purpose of this algorithm is to find disjoint communities in an undirected graph or in a directed graph which will be considered as an undirected one.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the DisjointCommunities class using the getCommunities method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.7.2 Method Documentation

3.7.2.1 - (void) addAllEdgeTypes

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

3.7.2.2 - (void) addEdgeType: (int) *type*

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters

type	[in] Edge type.

3.7.2.3 - (void) addNodeType: (int) *type*

Allows connectivity through nodes of the given type.

Parameters

type	null

3.7.2.4 - (void) close

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

3.7.2.5 - (void) excludeEdges: (STSObjects *) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.7.2.6 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.7.2.7 - (STSDisjointCommunities∗) getCommunities

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class DisjointCommunities which contain information related to the disjoint communities found.

3.7.2.8 - (void) includeEdges: (STSObjects ∗) *edges*

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.7.2.9 - (void) includeNodes: (STSObjects ∗) *nodes*

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.7.2.10 - (id) initWithSession: (STSSession ∗) *session*

Creates a new instance of CommunitiesSCD.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the communities.

Parameters

session	[in] Session to get the graph from and calculate the communities

3.7.2.11 - (void) setLookAhead: (int) *lookahead*

Sets the size of the lookahead iterations to look (5 by default).

Parameters

lookahead	[in] Number of iterations. It must be positive or zero.

3.7.2.12 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the disjoint communities found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

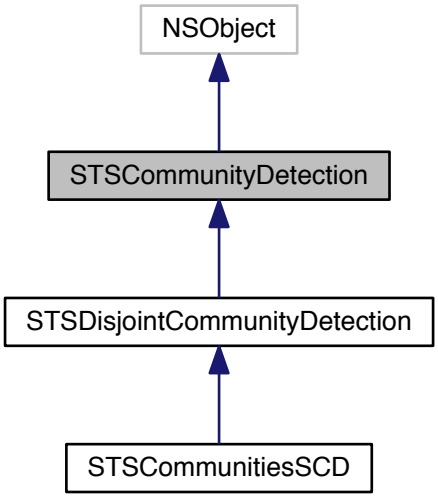
The documentation for this class was generated from the following file:

- Sparksee.h

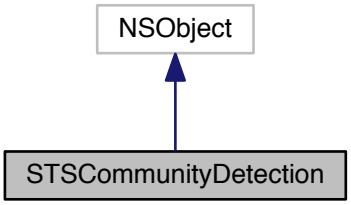
3.8 STSCommunityDetection Class Reference

CommunityDetection class.

Inheritance diagram for STSCommunityDetection:



Collaboration diagram for STSCommunityDetection:



Instance Methods

- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - includeNodes:

  Set additional nodes that can be used.

- (void) - includeEdges:

  Set additional edges that can be used.
- (void) - run

  Runs the algorithm in order to find the connected components.
- (void) - close

  Closes the CommunityDetection instance.
- (BOOL) - isClosed

  Check if the CommunityDetection instance is closed.

3.8.1 Detailed Description

CommunityDetection class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.8.2 Method Documentation

3.8.2.1 - (void) addNodeType: (int) *type*

Allows connectivity through nodes of the given type.

Parameters

type	null

3.8.2.2 - (void) close

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

3.8.2.3 - (void) excludeEdges: (STSObjects *) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.8.2.4 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.8.2.5 - (void) includeEdges: (STSObjects *) *edges*

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.8.2.6 - (void) includeNodes: (STSObjects *) *nodes*

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.8.2.7 - (void) run

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in STSDisjointCommunityDetection, and STSCommunitiesSCD.

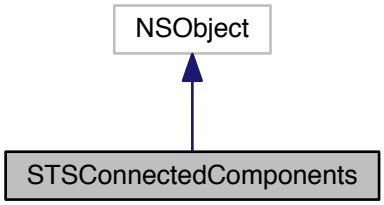
The documentation for this class was generated from the following file:

- Sparksee.h

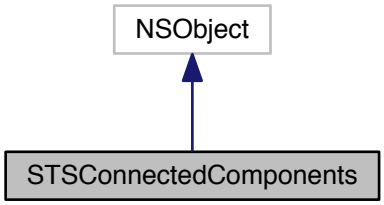
3.9 STSConnectedComponents Class Reference

ConnectedComponents class.

Inheritance diagram for STSConnectedComponents:



Collaboration diagram for STSConnectedComponents:



Instance Methods

- (id) - initWithS:materializedattribute:

  Creates a new instance of ConnectedComponents.
- (long long) - getConnectedComponent:

  Returns the connected component where the given node belongs to.
- (long long) - getCount

  Returns the number of connected components found in the graph.
- (STSObjects ∗) - getNodes:

  Returns the collection of nodes contained in the given connected component.
- (long long) - getSize:

  Returns the number of nodes contained in the given connected component.
- (void) - close

  Closes the ConnectedComponents instance.
- (BOOL) - isClosed

  Check if the ConnectedComponents instance is closed.

3.9.1 Detailed Description

ConnectedComponents class.

This class contains the results processed on a Connectivity algorithm.

These results contain information related to the connected components found. We must consider that each connected component has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different connected components found.

When executing any implementation of the Connectivity, it is possible to indicate whether the results of the execution must be stored persistently using the class Connectivity setMaterializedAttribute method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.9.2 Method Documentation

3.9.2.1 - (void) close

Closes the ConnectedComponents instance.

It must be called to ensure the integrity of all data.

3.9.2.2 - (long long) getConnectedComponent: (long long) *idNode*

Returns the connected component where the given node belongs to.

Parameters

idNode	[in] The node identifier for which the connected component identifier where it belongs will be returned.

Returns

The connected component identifier where the given node identifier belongs to.

3.9.2.3 - (long long) getCount

Returns the number of connected components found in the graph.

Returns

The number of connected components found in the graph.

3.9.2.4 - (STSObjects*) getNodes: (long long) *idConnectedComponent*

Returns the collection of nodes contained in the given connected component.

Parameters

idConnectedComponent	The connected component for which the collection of nodes contained in it will be returned.

Returns

The collection of node identifiers contained in the given connected component.

3.9.2.5 - (long long) getSize: (long long) *idConnectedComponent*

Returns the number of nodes contained in the given connected component.

Parameters

idConnectedComponent	The connected component for which the number of nodes contained in it will be returned.

Returns

The number of nodes contained in the given connected component.

3.9.2.6 - (id) initWithS: (STSSession $*$) *s* materializedattribute:(NSString $*$) *materializedattribute*

Creates a new instance of ConnectedComponents.

This constructor method can only be called when a previous execution of any implementation of the Connectivity class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any Connectivity execution see the documentation of the Connectivity::SetMaterializedAttribute method.

Parameters

s	[in] Session to get the graph Graph on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
materializedattribute	[in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the connected components found in the graph.

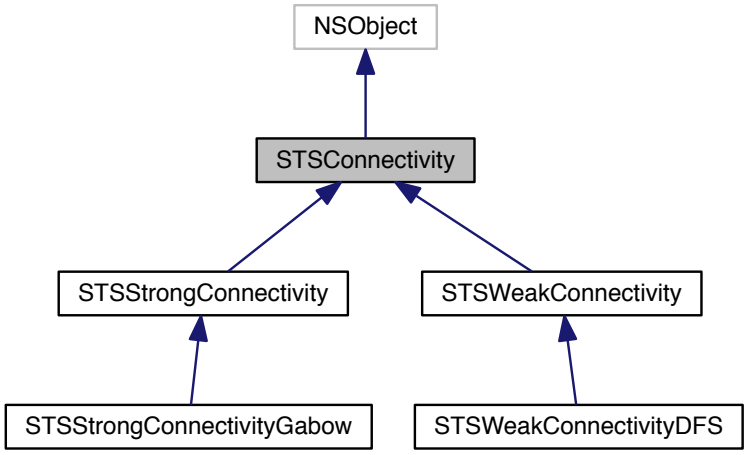
The documentation for this class was generated from the following file:

- Sparksee.h

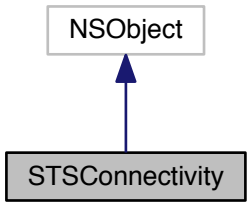
3.10 STSConnectivity Class Reference

Connectivity class.

Inheritance diagram for STSConnectivity:



Collaboration diagram for STSConnectivity:



Instance Methods

- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (STSConnectedComponents ∗) - getConnectedComponents

  Returns the results generated by the execution of the algorithm.
- (void) - run

Runs the algorithm in order to find the connected components.

- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

- (void) - close

  Closes the Connectivity instance.

- (BOOL) - isClosed

  Check if the Connectivity instance is closed.

3.10.1 Detailed Description

Connectivity class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.10.2 Method Documentation

3.10.2.1 - (void) addNodeType: (int) *t*

Allows connectivity through nodes of the given type.

Parameters

t	null

3.10.2.2 - (void) close

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

3.10.2.3 - (void) excludeEdges: (STSObjects *) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.10.2.4 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.10.2.5 - (STSConnectedComponents∗) getConnectedComponents

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

3.10.2.6 - (void) run

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in STSWeakConnectivityDFS, and STSStrongConnectivityGabow.

3.10.2.7 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

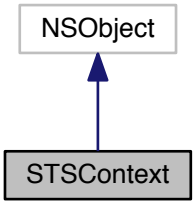
The documentation for this class was generated from the following file:

- Sparksee.h

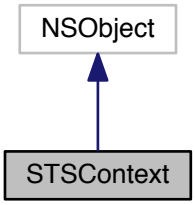
3.11 STSContext Class Reference

Context class.

Inheritance diagram for STSContext:



Collaboration diagram for STSContext:



Instance Methods

- (void) - addEdgeType:d:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (STSObjects ∗) - compute

  Gets the resulting collection of nodes.

- (void) - setMaximumHops:include:

  Sets the maximum hops restriction.
- (id) - initWithSession:node:

  Creates a new instance.
- (void) - close

  Closes the Context instance.
- (BOOL) - isClosed

  Check if the Context instance is closed.

Class Methods

- (STSObjects *) + computeWithArguments:node:nodeTypes:edgeTypes:dir:maxhops:include:

  Helper method to easily compute a context from a node.

3.11.1 Detailed Description

Context class.

It provides a very similar functionality than the Traversal classes. The main difference is Context returns a resulting collection whereas Traversal provides an iterator behaviour.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.11.2 Method Documentation

3.11.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *d*

Allows for traversing all edge types of the graph.

Parameters

d	[in] Edge direction.

3.11.2.2 - (void) addEdgeType: (int) *t* d:(enum STSEdgesDirection) *d*

Allows for traversing edges of the given type.

Parameters

t	[in] Edge type.
d	[in] Edge direction.

3.11.2.3 - (void) addNodeType: (int) *t*

Allows for traversing nodes of the given type.

Parameters

t	null

3.11.2.4 - (void) close

Closes the Context instance.

It must be called to ensure the integrity of all data.

3.11.2.5 - (STSObjects∗) compute

Gets the resulting collection of nodes.

Returns

The resulting collection of nodes.

3.11.2.6 + (STSObjects∗) computeWithArguments: (STSSession ∗) *session* node:(long long) *node* nodeTypes:(STSTypeList ∗) *nodeTypes* edgeTypes:(STSTypeList ∗) *edgeTypes* dir:(enum STSEdgesDirection) *dir* maxhops:(int) *maxhops* include:(BOOL) *include*

Helper method to easily compute a context from a node.

Parameters

session	[in] Session to get the graph from and perform operation.
node	[in] Node to start traversal from.
nodeTypes	[in] Allowed node type list. NULL means all node types are allowed.
edgeTypes	[in] Allowed edge type list. NULL means all edge types are allowed.
dir	[in] Allowed direction for the allowed edge types.
maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
include	[in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

Returns

Returns an Objects with the computed context of a node.

3.11.2.7 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.11.2.8 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.11.2.9 - (id) initWithSession: (STSSession *) *session* node:(long long) *node*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform operation.
node	[in] Node to start traversal from.

3.11.2.10 - (void) setMaximumHops: (int) *maxhops* include:(BOOL) *include*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

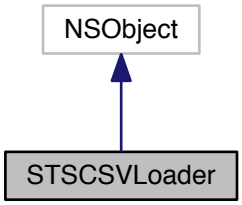
maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.
include	[in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

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

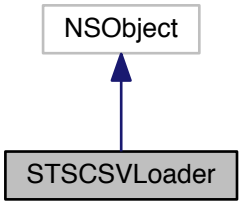
- Sparksee.h

3.12 STSCSVLoader Class Reference

Inheritance diagram for STSCSVLoader:



Collaboration diagram for STSCSVLoader:



Class Methods

- (void) + loadNodes:fileName:nodeType:separator:header:columns:attrNames:dataTypes:attrKinds:

  Loads nodes from a CSV file.
- (void) + loadEdges:fileName:edgeType:tailNodeType:headNodeType:tail:head:separator:directed:header:onMissingTail:onMissingHead:columns:attrNames:dataTypes:attrKinds:

  Loads edges from a CSV file.

3.12.1 Method Documentation

3.12.1.1 + (void) loadEdges: (STSGraph ∗) *graph* fileName:(NSString ∗) *fileName* edgeType:(NSString ∗) *edgeType* tailNodeType:(NSString ∗) *tailNodeType* headNodeType:(NSString ∗) *headNodeType* tail:(int) *tail* head:(int) *head* separator:(NSString ∗) *separator* directed:(BOOL) *directed* header:(BOOL) *header* onMissingTail:(enum STSMissingEndpoint) *onMissingTail* onMissingHead:(enum STSMissingEndpoint) *onMissingHead* columns:(NSArray ∗) *columns* attrNames:(NSArray ∗) *attrNames* dataTypes:(NSArray ∗) *dataTypes* attrKinds:(NSArray ∗) *attrKinds*

Loads edges from a CSV file.

nodeType[in] The type of the edge to load. If it does not exist, it creates it

Parameters

graph	[in] The graph to load the edges into
fileName	[in] The name of the file
edgeType	null
tailNodeType	[in] The type of the tail nodes. If it does not exist and onMissingTail is set to "Create", it creates it. Otherwise, an exception is thrown
headNodeType	[in] The type of the head nodes. It it does not exist and onMissingHead is set to "Create", it creates it. Otherwise, an exception is thrown
tail	[in] The tail column index. Default: 0
head	[in] The head column index. Default: 1
separator	[in] The column separator. Default: ","
directed	[in] True if this edge is directed or not. False otherwise. Default: True
header	[in] True if the CSV contains a header. False otherwise. Default: True
onMissingTail	[in] The policy to follow when a tail is missing. Default: "IsError"
onMissingHead	[in] The policy to follow when a head is missing. Default: "IsError"
columns	[in] The list of columns to load. tail and head columns must be in this list if this list is specified. If the list is empty, all columns are loaded. Default: empty
attrNames	[in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty
dataTypes	[in] The dataTypes of the attributes if this do not already exist. If this list is empty, the method tries to infer the data type from the first non-header row. Default: empty
attrKinds	[in] The attributeKinds of the attributes if these do not already exist. If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty

3.12.1.2 + (void) loadNodes: (STSGraph ∗) *graph* fileName:(NSString ∗) *fileName* nodeType:(NSString ∗) *nodeType* separator:(NSString ∗) *separator* header:(BOOL) *header* columns:(NSArray ∗) *columns* attrNames:(NSArray ∗) *attrNames* dataTypes:(NSArray ∗) *dataTypes* attrKinds:(NSArray ∗) *attrKinds*

Loads nodes from a CSV file.

Parameters

graph	[in] The graph to load the nodes into
fileName	[in] The name of the file
nodeType	[in] The type of the node to load. If it does not exist, it creates it
separator	[in] The separator of the CSV file. Default: ","
header	[in] True if the CSV contains a header. False otherwise. Default: True
columns	[in] The list of columns to load. tail and head columns mus be in this list if this list is specified. Default: all columns
attrNames	[in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty
dataTypes	[in] The dataTypes of the attributes if this do not already exist. Default: empty If this list is empty, the method tries to infer the data type from the first non-header row
attrKinds	[in] The attributeKinds of the attributes if these do not already exist. Default: empty If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty

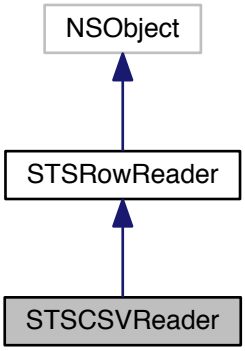
The documentation for this class was generated from the following file:

- Sparksee.h

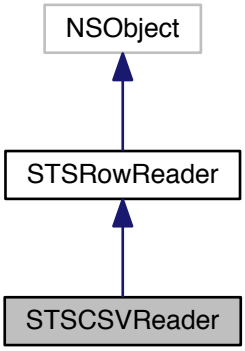
3.13 STSCSVReader Class Reference

CSVReader interface.

Inheritance diagram for STSCSVReader:



Collaboration diagram for STSCSVReader:



Instance Methods

- (id) - init

  Constructs CSVReader.
- (void) - setSeparator:

  Sets the character used to separate fields in the file.
- (void) - setQuotes:

  Sets the character used to quote fields.

- (void) - setMultilines:
  Allows the use of fields with more than one line.
- (void) - setSingleLine
  Only allows single line fields.
- (void) - setStartLine:
  Sets the number of lines to be skiped from the beginning.
- (void) - setNumLines:
  Used to limit the number of lines that will be read.
- (void) - setLocale:
  Sets the locale that will be used to read the file.
- (void) - open:
  Opens the source file path.
- (BOOL) - reset
  Moves the reader to the beginning.
- (BOOL) - read:
  Reads the next row as a string array.
- (int) - getRow
  The row number for the current row.
- (void) - close
  Closes the reader.

3.13.1 Detailed Description

CSVReader interface.

A very simple CSV reader.

It works as any other RowReader, but open must be called once before the first read operation.

Using the format RFC 4180.

Except: leading and trailing spaces, adjacent to CSV separator character, are trimmed.

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (").

Fields with multiple lines can be allowed (and the maximum lines specified), but the default is a single line.

The locale string can be used to set the language, country and the file encoding. The format must be "[language↩_territory][.codeset]". But only the file encoding is being used in the current version.

The languages supported are: "en_US", "es_ES" and "ca_ES".

The file encodings supported are: "utf8" and "iso88591".

For example:

To don't change the default locales, use an empty string: "".

To read a file in utf8 with the default language settings use ".utf8".

To read a file in iso88591 with English language use: "en_US.iso88591".

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.13.2 Method Documentation

3.13.2.1 - (void) close

Closes the reader.

Exceptions

System.IO.IOException	If the close fails.

Implements STSRowReader.

3.13.2.2 - (int) getRow

The row number for the current row.

Returns

The current row number; 0 if there is no current row.

Exceptions

System.IO.IOException	If it fails.

Implements STSRowReader.

3.13.2.3 - (void) open: (NSString ∗) *filePath*

Opens the source file path.

File can be optionally compressed in GZIP format.

Parameters

filePath	[in] CSV file path.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.13.2.4 - (BOOL) read: (STSStringList ∗) *row*

Reads the next row as a string array.

Parameters

row	[out] A string list with each comma-separated element as a separate entry.

Returns

Returns true if a row had been read or false otherwise.

Exceptions

System.IO.IOException	If bad things happen during the read.

Implements STSRowReader.

3.13.2.5 - (BOOL) reset

Moves the reader to the beginning.

Restarts the reader.

Returns

true if the reader can be restarted, false otherwise.

Exceptions

System.IO.IOException	If bad things happen during the restart.

Implements STSRowReader.

3.13.2.6 - (void) setLocale: (NSString ∗) *localeStr*

Sets the locale that will be used to read the file.

Parameters

localeStr	[in] The locale string for the file encoding.

3.13.2.7 - (void) setMultilines: (int) *numExtralines*

Allows the use of fields with more than one line.

Parameters

numExtralines	[in] Maximum number of extra lines for each column (0==unlimited, N==N+1 total rows).

3.13.2.8 - (void) setNumLines: (int) *numLines*

Used to limit the number of lines that will be read.

Parameters

numLines	[in] The maximum number of lines to read (0 == unlimited)

3.13.2.9 - (void) setQuotes: (NSString ∗) *quotes*

Sets the character used to quote fields.

Parameters

quotes	[in] Quote character.

Exceptions

System.ApplicationException	null

3.13.2.10 - (void) setSeparator: (NSString ∗) *sep*

Sets the character used to separate fields in the file.

Parameters

sep	[in] Separator character.

Exceptions

System.ApplicationException	null

3.13.2.11 - (void) setStartLine: (int) *startLine*

Sets the number of lines to be skiped from the beginning.

Parameters

startLine	[in] The line number to skip for start reading

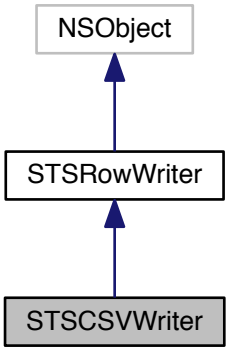
The documentation for this class was generated from the following file:

- Sparksee.h

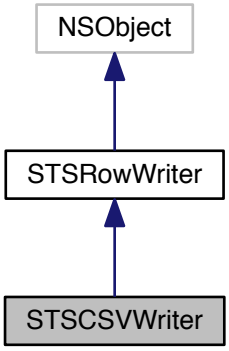
3.14 STSCSVWriter Class Reference

CSVWriter interface.

Inheritance diagram for STSCSVWriter:



Collaboration diagram for STSCSVWriter:



Instance Methods

- (id) - init

  Creates a new instance.
- (void) - setSeparator:

  Sets the character used to separate fields in the file.
- (void) - setQuotes:

  Sets the character used to quote fields.
- (void) - setAutoQuotes:

  Sets on/off the automatic quote mode.
- (void) - setForcedQuotes:

  Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

- (void) - setLocale:

  Sets the locale that will be used to write the file.
- (void) - open:

  Opens the output file path.
- (void) - write:

  Writes the next row.
- (void) - close

  Closes the writer.

3.14.1 Detailed Description

CSVWriter interface.

A very simple CSV writer implementing RowWriter.

It works as any other RowWriter, but open must be called once before the first write operation.

It uses the format RFC 4180: http://tools.ietf.org/html/rfc4180

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (") and autoquote is enabled.

See the CSVReader locale documentation or the SPARKSEE User Manual.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.14.2 Method Documentation

3.14.2.1 - (void) close

Closes the writer.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the close fails.

Implements STSRowWriter.

3.14.2.2 - (void) open: (NSString ∗) *f*

Opens the output file path.

Parameters

f	[in] Output file path.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.14.2.3 - (void) setAutoQuotes: (BOOL) *autoquotes*

Sets on/off the automatic quote mode.

If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced quotes are provided, there will not be any quote.

Parameters

autoquotes	[in] If TRUE it enables the automatic quote mode, if FALSE it disables it.

3.14.2.4 - (void) setForcedQuotes: (STSBooleanList ∗) *forcequotes*

Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

Parameters

forcequotes	[in] A booleanList with the position for each column that must be quoted set to true.

3.14.2.5 - (void) setLocale: (NSString ∗) *localeStr*

Sets the locale that will be used to write the file.

Parameters

localeStr	[in] The locale string for the file encoding.

3.14.2.6 - (void) setQuotes: (NSString ∗) *quotes*

Sets the character used to quote fields.

Parameters

quotes	[in] Quote character.

Exceptions

System.ApplicationException	null

3.14.2.7 - (void) setSeparator: (NSString ∗) *sep*

Sets the character used to separate fields in the file.

Parameters

sep	[in] Separator character.

Exceptions

System.ApplicationException	null

3.14.2.8 - (void) write: (STSStringList ∗) *row*

Writes the next row.

Parameters

row	[in] Row of data.

Exceptions

System.ApplicationException	null
System.IO.IOException	If bad things happen during the write.

Implements STSRowWriter.

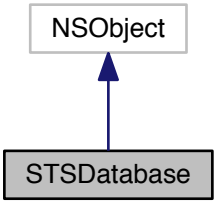
The documentation for this class was generated from the following file:

- Sparksee.h

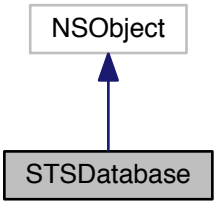
3.15 STSDatabase Class Reference

Database class.

Inheritance diagram for STSDatabase:



Collaboration diagram for STSDatabase:



Instance Methods

- (NSString ∗) - getAlias

  Gets the alias of the Database.
- (NSString ∗) - getPath

  Gets the path of the Database.
- (STSSession ∗) - createSession

  Creates a new Session.
- (void) - enableRollback

  Enables the rollback mechanism.
- (void) - disableRollback

  Disables the rollback mechanism.
- (void) - getStatistics:

  Gets Database statistics.
- (int) - getCacheMaxSize

  Gets the cache maximum size (in MB).
- (void) - setCacheMaxSize:

  Sets the cache maximum size (in MB).
- (void) - fixCurrentCacheMaxSize

  Sets the cache maximum size to the current cache size in use.
- (void) - dumpSchema:

  Dump the database schema to the given file using the Sparksee scripting format.
- (void) - redoPrecommitted:

  Redo a pending precommitted transaction recovered.
- (void) - close

  Closes the Database instance.
- (BOOL) - isClosed

  Check if the Database instance is closed.

3.15.1 Detailed Description

Database class.

All the data of the Database is stored into a persistent file which just can be created or open through a Sparksee instance.

Also, all the manipulation of a Database must be done by means of a Session which can be initiated from a Database instance.

Multiple Databases do not share the memory, that is there is no negotiation among them. In those cases, memory must be prefixed for each Database. To do that, use the SPARKSEEConfig.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.15.2 Method Documentation

3.15.2.1 - (void) close

Closes the Database instance.

It must be called to ensure the integrity of all data.

3.15.2.2 - (void) dumpSchema: (NSString ∗) *filePath*

Dump the database schema to the given file using the Sparksee scripting format.

Parameters

filePath	null

3.15.2.3 - (NSString∗) getAlias

Gets the alias of the Database.

Returns

The alias of the Database.

3.15.2.4 - (int) getCacheMaxSize

Gets the cache maximum size (in MB).

Returns

Returns the current cache max size.

3.15.2.5 - (NSString*) getPath

Gets the path of the Database.

Returns

The path of the Database.

3.15.2.6 - (void) getStatistics: (STSDatabaseStatistics *) *stats*

Gets Database statistics.

Parameters

stats	[out] The DatabaseStatistics instance.

3.15.2.7 - (void) redoPrecommitted: (long long) *txId*

Redo a pending precommitted transaction recovered.

YOU SHOULD NOT USE THIS METHOD. This method only exists for a specific service.

Parameters

txId	null

3.15.2.8 - (void) setCacheMaxSize: (int) *megaBytes*

Sets the cache maximum size (in MB).

0 means unlimited which is all the physical memory of the computer minus a small margin.

Parameters

megaBytes	[in] The new cache max size.

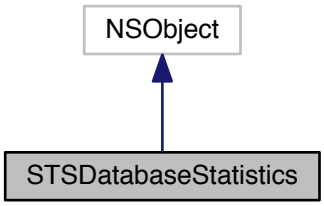
The documentation for this class was generated from the following file:

- Sparksee.h

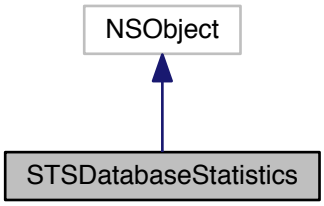
3.16 STSDatabaseStatistics Class Reference

Database statistics.

Inheritance diagram for STSDatabaseStatistics:



Collaboration diagram for STSDatabaseStatistics:



Instance Methods

- (long long) - getRead
  Gets total read data in KBytes.
- (long long) - getWrite
  Gets total written data in KBytes.
- (long long) - getData
  Gets database size in KBytes.
- (long long) - getCache
  Gets cache size in KBytes.
- (long long) - getTemp
  Gets temporary storage file size in KBytes.
- (long long) - getSessions
  Gets the number of sessions.

3.16.1 Detailed Description

Database statistics.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.16.2 Method Documentation

3.16.2.1 - (long long) getCache

Gets cache size in KBytes.

Returns

Cache size in KBytes.

3.16.2.2 - (long long) getData

Gets database size in KBytes.

Returns

Database size in KBytes.

3.16.2.3 - (long long) getRead

Gets total read data in KBytes.

Returns

Total read data in KBytes.

3.16.2.4 - (long long) getSessions

Gets the number of sessions.

Returns

The number of sessions.

3.16.2.5 - (long long) getTemp

Gets temporary storage file size in KBytes.

Returns

Temporary storage file size in KBytes.

3.16.2.6 - (long long) getWrite

Gets total written data in KBytes.

Returns

Total read written in KBytes.

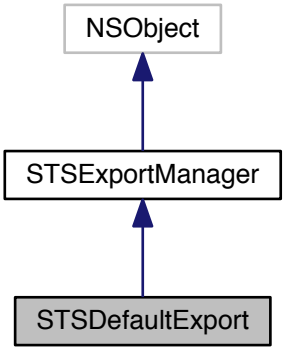
The documentation for this class was generated from the following file:

- Sparksee.h

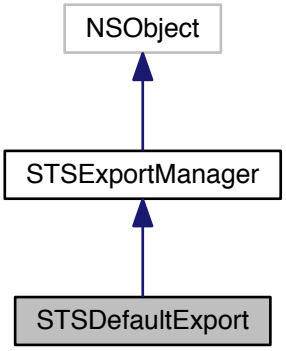
3.17 STSDefaultExport Class Reference

Default implementation for ExportManager class.

Inheritance diagram for STSDefaultExport:



Collaboration diagram for STSDefaultExport:



Instance Methods

- (id) - init

  Creates a new instance.
- (void) - prepare:

  Default implementation of the ExportManager class method Prepare.
- (void) - close

  Default implementation of the ExportManager class method Release.

- (BOOL) - getGraph:

  Default implementation of the ExportManager class method GetGraph.
- (BOOL) - getNodeType:nodeExport:

  Default implementation of the ExportManager class method GetNodeType.
- (BOOL) - getEdgeType:edgeExport:

  Default implementation of the ExportManager class method GetEdgeType.
- (BOOL) - getNode:nodeExport:

  Default implementation of the ExportManager class method GetNode.
- (BOOL) - getEdge:edgeExport:

  Default implementation of the ExportManager class method GetEdge.
- (BOOL) - enableType:

  Default implementation of the ExportManager class method EnableType.

3.17.1 Detailed Description

Default implementation for ExportManager class.

It uses the default values from GraphExport, NodeExport and EdgeExport to export all node and edge types.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.17.2 Method Documentation

3.17.2.1 - (BOOL) enableType: (int) *type*

Default implementation of the ExportManager class method EnableType.

This enables all node and edge types to be exported.

Parameters

type	[in] The type to enable.

Returns

TRUE.

Implements STSExportManager.

3.17.2.2 - (BOOL) getEdge: (long long) *edge* edgeExport:(STSEdgeExport *) *edgeExport*

Default implementation of the ExportManager class method GetEdge.

This sets the default EdgeExport values and sets the OID as the label. Also, it exports the edge as directed just if the edge is directed.

Parameters

edge	[in] An edge.
edgeExport	[out] The EdgeExport that will store the information.

Returns

TRUE.

Implements STSExportManager.

3.17.2.3 - (BOOL) getEdgeType: (int) *type* edgeExport:(STSEdgeExport ∗) *edgeExport*

Default implementation of the ExportManager class method GetEdgeType.

This sets de default EdgeExport values.

Parameters

type	[in] An edge type.
edgeExport	[out] The EdgeExport that will store the information.

Returns

TRUE.

Implements STSExportManager.

3.17.2.4 - (BOOL) getGraph: (STSGraphExport ∗) *graphExport*

Default implementation of the ExportManager class method GetGraph.

This sets the default GraphExport values and "Graph" as the label.

Parameters

graphExport	[out] The GraphExport that will store the information.

Returns

TRUE.

Implements STSExportManager.

3.17.2.5 - (BOOL) getNode: (long long) *node* nodeExport:(STSNodeExport ∗) *nodeExport*

Default implementation of the ExportManager class method GetNode.

This sets the default NodeExport values and sets the OID as the label.

Parameters

node	[in] A node.
nodeExport	[out] The NodeExport that will store the information.

Returns

TRUE.

Implements STSExportManager.

3.17.2.6 - (BOOL) getNodeType: (int) *type* nodeExport:(STSNodeExport ∗) *nodeExport*

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

Parameters

type	[in] A node type.
nodeExport	[out] The NodeExport that will store the information.

Returns

TRUE.

Implements STSExportManager.

3.17.2.7 - (void) prepare: (STSGraph ∗) *graph*

Default implementation of the ExportManager class method Prepare.

Parameters

graph	null

Implements STSExportManager.

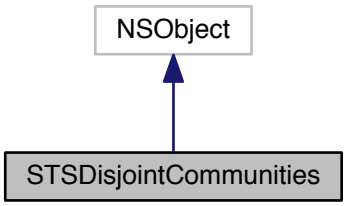
The documentation for this class was generated from the following file:

- Sparksee.h

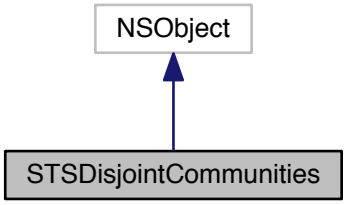
3.18 STSDisjointCommunities Class Reference

DisjointCommunities class.

Inheritance diagram for STSDisjointCommunities:



Collaboration diagram for STSDisjointCommunities:



Instance Methods

- (id) - initWithSession:materializedattribute:

  Creates a new instance of DisjointCommunities.
- (long long) - getCommunity:

  Returns the disjoint community where the given node belongs to.
- (long long) - getCount

  Returns the number of communities found in the graph.
- (STSObjects ∗) - getNodes:

  Returns the collection of nodes contained in the given community.
- (long long) - getSize:

  Returns the number of nodes contained in the given community.
- (void) - close

  Closes the DisjointCommunities instance.
- (BOOL) - isClosed

  Check if the DisjointCommunities instance is closed.

3.18.1 Detailed Description

DisjointCommunities class.

This class contains the results processed on a DisjointCommunityDetection algorithm.

These results contain information related to the communities found. We must consider that each community has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different communities found.

When executing any implementation of the DisjointCommunityDetection, it is possible to indicate whether the results of the execution must be stored persistently using the class DisjointCommunityDetection setMaterializedAttribute method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.18.2 Method Documentation

3.18.2.1 - (void) close

Closes the DisjointCommunities instance.

It must be called to ensure the integrity of all data.

3.18.2.2 - (long long) getCommunity: (long long) *idNode*

Returns the disjoint community where the given node belongs to.

Parameters

idNode	[in] The node identifier for which the disjoint community identifier where it belongs will be returned.

Returns

The disjoint community identifier where the given node identifier belongs to.

3.18.2.3 - (long long) getCount

Returns the number of communities found in the graph.

Returns

The number of communities found in the graph.

3.18.2.4 - (STSObjects*) getNodes: (long long) *idCommunity*

Returns the collection of nodes contained in the given community.

Parameters

idCommunity	The community for which the collection of nodes contained in it will be returned.

Returns

The collection of node identifiers contained in the given community.

3.18.2.5 - (long long) getSize: (long long) *idCommunity*

Returns the number of nodes contained in the given community.

Parameters

idCommunity	The community for which the number of nodes contained in it will be returned.

Returns

The number of nodes contained in the given community.

3.18.2.6 - (id) initWithSession: (STSSession ∗) *session* materializedattribute:(NSString ∗) *materializedattribute*

Creates a new instance of DisjointCommunities.

This constructor method can only be called when a previous execution of any implementation of the Disjoint↩ CommunityDetection class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any DisjointCommunityDetection execution see the documentation of the DisjointCommunityDetection::SetMaterializedAttribute method.

Parameters

session	[in] Session to get the graph Graph on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
materializedattribute	[in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the communities found in the graph.

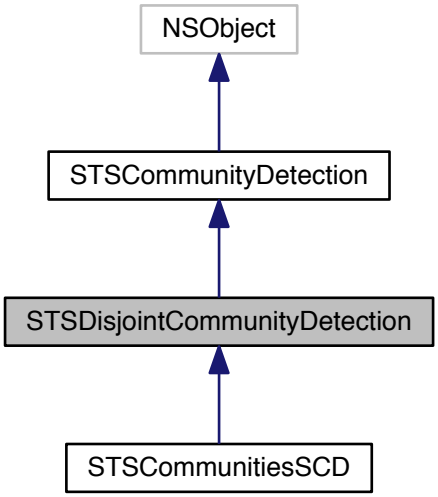
The documentation for this class was generated from the following file:

- Sparksee.h

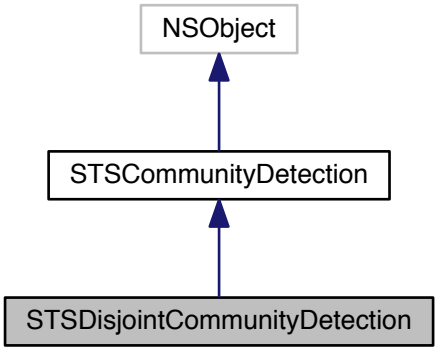
3.19 STSDisjointCommunityDetection Class Reference

DisjointCommunityDetection class.

Inheritance diagram for STSDisjointCommunityDetection:



Collaboration diagram for STSDisjointCommunityDetection:



Instance Methods

- (void) - addEdgeType:

  Allows connectivity through edges of the given type.
- (void) - addAllEdgeTypes

  Allows connectivity through all edge types of the graph.
- (STSDisjointCommunities ∗) - getCommunities

  Returns the results generated by the execution of the algorithm.

- (void) - run

  Runs the algorithm in order to find the communities.
- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.
- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - includeNodes:

  Set additional nodes that can be used.
- (void) - includeEdges:

  Set additional edges that can be used.
- (void) - close

  Closes the CommunityDetection instance.
- (BOOL) - isClosed

  Check if the CommunityDetection instance is closed.

3.19.1 Detailed Description

DisjointCommunityDetection class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components, finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.19.2 Method Documentation

3.19.2.1 - (void) addAllEdgeTypes

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

3.19.2.2 - (void) addEdgeType: (int) *type*

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

Parameters

type	[in] Edge type.

3.19.2.3 - (void) addNodeType: (int) *type*

Allows connectivity through nodes of the given type.

Parameters

type	null

3.19.2.4 - (void) close

Closes the CommunityDetection instance.

It must be called to ensure the integrity of all data.

3.19.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.19.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.19.2.7 - (STSDisjointCommunities∗) getCommunities

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class DisjointCommunities which contain information related to the disjoint communities found.

3.19.2.8 - (void) includeEdges: (STSObjects *) edges

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.19.2.9 - (void) includeNodes: (STSObjects *) nodes

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.19.2.10 - (void) run

Runs the algorithm in order to find the communities.

This method can be called only once.

Implements STSCommunityDetection.

Implemented in STSCommunitiesSCD.

3.19.2.11 - (void) setMaterializedAttribute: (NSString *) attributeName

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the disjoint communities found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class DisjointCommunities indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the disjoint communities found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

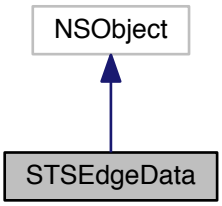
The documentation for this class was generated from the following file:

- Sparksee.h

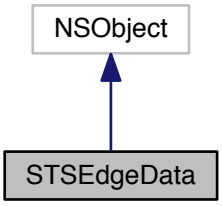
3.20 STSEdgeData Class Reference

Edge data class.

Inheritance diagram for STSEdgeData:



Collaboration diagram for STSEdgeData:



Instance Methods

- (long long) - getEdge

  Gets the edge identifier.
- (long long) - getTail

  Gets the tail of the edge.
- (long long) - getHead

  Gets the head of the edge.

3.20.1 Detailed Description

Edge data class.

It stores the tail and the head of an edge instance.

In case of undirected eges, the tail and the head are just the two ends of the edge.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.20.2 Method Documentation

3.20.2.1 - (long long) getEdge

Gets the edge identifier.

Returns

The Sparksee edge identifier.

3.20.2.2 - (long long) getHead

Gets the head of the edge.

Returns

The Sparksee edge identifier of the head of the edge.

3.20.2.3 - (long long) getTail

Gets the tail of the edge.

Returns

The Sparksee edge identifier of the tail of the edge.

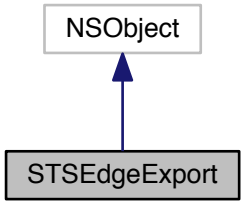
The documentation for this class was generated from the following file:

- Sparksee.h

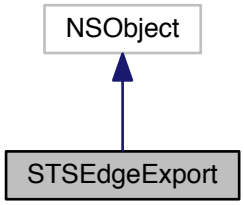
3.21 STSEdgeExport Class Reference

Stores edge exporting values.

Inheritance diagram for STSEdgeExport:



Collaboration diagram for STSEdgeExport:



Instance Methods

- (id) - init
  Creates a new instance.
- (void) - setDefaults
  Sets to default values.
- (NSString ∗) - getLabel
  Gets the edge label.
- (void) - setLabel:
  Sets the edge label.
- (BOOL) - asDirected
  Gets if the edge should be managed as directed.
- (void) - setAsDirected:
  Sets if the edge should be managed as directed.
- (int) - getColorRGB
  Gets the edge color.

- (void) - setColorRGB:
  Sets the edge color.
- (int) - getLabelColorRGB
  Gets the edge label color.
- (void) - setLabelColorRGB:
  Sets the edge label color.
- (int) - getWidth
  Gets the edge width.
- (void) - setWidth:
  Sets the edge width.
- (int) - getFontSize
  Gets the edge label font size.
- (void) - setFontSize:
  Sets the edge label font size.
- (void) - getColorRed:green:blue:alpha:
  Get the edge color separated in RGBA.
- (void) - setColorRed:green:blue:alpha:
  Set the edge color with separated RGBA components.
- (void) - getLabelColorRed:green:blue:alpha:
  Get the edge label color separated in RGBA.
- (void) - setLabelColorRed:green:blue:alpha:
  Set the edge label color with separated RGBA components.

3.21.1 Detailed Description

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

As directed: TRUE.

Color: 13882323 (OxD3D3D3, Light gray).

Label color: 0 (Ox000000, Black).

Width: 5px.

Font size: 10.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.21.2 Method Documentation

3.21.2.1 - (BOOL) asDirected

Gets if the edge should be managed as directed.

TRUE is the default value. If TRUE, use as directed, otherwise use as undirected.

Returns

The edge direction.

3.21.2.2 - (void) getColorRed: (double *) *red* green:(double *) *green* blue:(double *) *blue* alpha:(double *) *alpha*

Get the edge color separated in RGBA.

Parameters

red	[out] The red color component ([0..1]).
green	[out] The green color component ([0..1]).
blue	[out] The blue color component ([0..1]).
alpha	[out] The alpha component ([0..1]).

3.21.2.3 - (int) getColorRGB

Gets the edge color.

Returns

The edge color.

3.21.2.4 - (int) getFontSize

Gets the edge label font size.

Returns

The edge label font size.

3.21.2.5 - (NSString∗) getLabel

Gets the edge label.

Returns

The edge label.

3.21.2.6 - (void) getLabelColorRed: (double ∗) *red* green:(double ∗) *green* blue:(double ∗) *blue* alpha:(double ∗) *alpha*

Get the edge label color separated in RGBA.

Parameters

red	[out] The red color component ([0..1]).
green	[out] The green color component ([0..1]).
blue	[out] The blue color component ([0..1]).
alpha	[out] The alpha component ([0..1]).

3.21.2.7 - (int) getLabelColorRGB

Gets the edge label color.

Returns

The edge label color.

3.21.2.8 - (int) getWidth

Gets the edge width.

Returns

The edge width.

3.21.2.9 - (void) setAsDirected: (BOOL) *directed*

Sets if the edge should be managed as directed.

Parameters

directed	[in] If TRUE, use as directed, otherwise use as undirected.

3.21.2.10 - (void) setColorRed: (double) *red* green:(double) *green* blue:(double) *blue* alpha:(double) *alpha*

Set the edge color with separated RGBA components.

Parameters

red	[in] The red color component ([0..1]).
green	[in] The green color component ([0..1]).
blue	[in] The blue color component ([0..1]).
alpha	[in] The alpha component ([0..1]).

3.21.2.11 - (void) setColorRGB: (int) *color*

Sets the edge color.

Parameters

color	[in] The edge color.

3.21.2.12 - (void) setFontSize: (int) *size*

Sets the edge label font size.

Parameters

size	[in] The edge label font size.

3.21.2.13 - (void) setLabel: (NSString ∗) *label*

Sets the edge label.

Parameters

label	[in] The edge label.

3.21.2.14 - (void) setLabelColorRed: (double) *red* green:(double) *green* blue:(double) *blue* alpha:(double) *alpha*

Set the edge label color with separated RGBA components.

Parameters

red	[in] The red color component ([0..1]).
green	[in] The green color component ([0..1]).
blue	[in] The blue color component ([0..1]).
alpha	[in] The alpha component ([0..1]).

3.21.2.15 - (void) setLabelColorRGB: (int) *color*

Sets the edge label color.

Parameters

color	[in] The edge label color.

3.21.2.16 - (void) setWidth: (int) *width*

Sets the edge width.

Parameters

width	[in] The edge width.

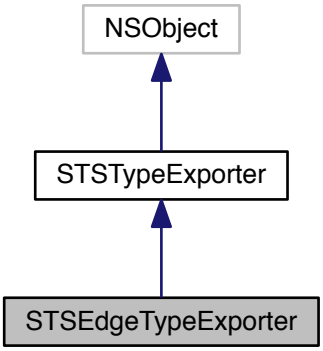
The documentation for this class was generated from the following file:

- Sparksee.h

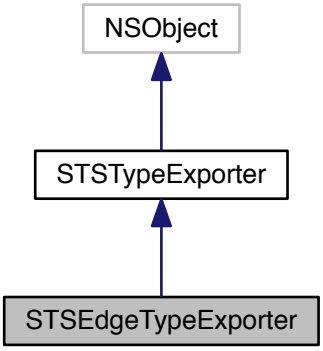
3.22 STSEdgeTypeExporter Class Reference

EdgeTypeExporter class.

Inheritance diagram for STSEdgeTypeExporter:



Collaboration diagram for STSEdgeTypeExporter:



Instance Methods

- (id) - init

  Creates a new instance.
- (id) - initWithRowWriter:graph:type:attrs:hPos:tPos:hAttr:tAttr:

  Creates a new instance.
- (void) - run

  See the TypeExporter class Run method.
- (void) - setHeadAttribute:

  Sets the attribute that will be used to get the value to be dumped for the head of the edge.
- (void) - setHeadPosition:

  Sets the position (index column) of the head attribute in the exported data.

- (void) - setTailAttribute:

  Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
- (void) - setTailPosition:

  Sets the position (index column) of the tail attribute in the exported data.
- (void) - registerListener:

  Registers a new listener.
- (void) - setRowWriter:

  Sets the output data destination.
- (void) - setGraph:

  Sets the graph that will be exported.
- (void) - setType:

  Sets the type to be exported.
- (void) - setAttributes:

  Sets the list of Attributes.
- (void) - setFrequency:

  Sets the frequency of listener notification.
- (void) - setHeader:

  Sets the presence of a header row.

3.22.1 Detailed Description

EdgeTypeExporter class.

Specific TypeExporter implementation for edge types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.22.2 Method Documentation

3.22.2.1 - (id) initWithRowWriter: (STSRowWriter ∗) *rowWriter* graph:(STSGraph ∗) *graph* type:(int) *type* attrs:(STSAttributeList ∗) *attrs* hPos:(int) *hPos* tPos:(int) *tPos* hAttr:(int) *hAttr* tAttr:(int) *tAttr*

Creates a new instance.

Parameters

rowWriter	[in] Output RowWriter.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be exported.
hPos	[in] The position (index column) for the head value.
tPos	[in] The position (index column) for the tail value.
hAttr	[in] The attribute identifier to get the value to be dumped for the head.
tAttr	[in] The attribute identifier to get the value to be dumped for the tail.

3.22.2.2 - (void) registerListener: (STSTypeExporterListener ∗) *tel*

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

3.22.2.3 - (void) run

See the TypeExporter class Run method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeExporter.

3.22.2.4 - (void) setAttributes: (STSAttributeList ∗) *attrs*

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

3.22.2.5 - (void) setFrequency: (int) *freq*

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.22.2.6 - (void) setGraph: (STSGraph ∗) *graph*

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

3.22.2.7 - (void) setHeadAttribute: (int) *attr*

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

Parameters

attr	[in] Head Attribute

3.22.2.8 - (void) setHeader: (BOOL) *header*

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

3.22.2.9 - (void) setHeadPosition: (int) *pos*

Sets the position (index column) of the head attribute in the exported data.

Parameters

pos	[in] Head position

3.22.2.10 - (void) setRowWriter: (STSRowWriter ∗) *rw*

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

3.22.2.11 - (void) setTailAttribute: (int) *attr*

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

Parameters

attr	[in] Tail Attribute

3.22.2.12 - (void) setTailPosition: (int) *pos*

Sets the position (index column) of the tail attribute in the exported data.

Parameters

pos	[in] Tail position

3.22.2.13 - (void) setType: (int) *type*

Sets the type to be exported.

Parameters

type	[in] Type identifier.

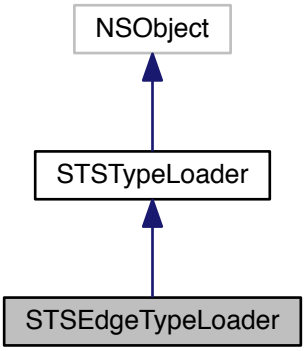
The documentation for this class was generated from the following file:

- Sparksee.h

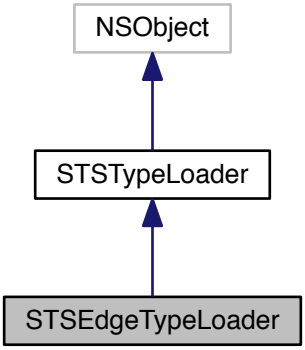
3.23 STSEdgeTypeLoader Class Reference

EdgeTypeLoader class.

Inheritance diagram for STSEdgeTypeLoader:



Collaboration diagram for STSEdgeTypeLoader:



Instance Methods

- (id) - init

  Creates a new instance.
- (void) - run

  See the TypeLoader class Run method.
- (void) - runTwoPhases

  See the TypeLoader class RunTwoPhases method.
- (void) - runNPhases:

  See the TypeLoader class RunNPhases method.
- (void) - setHeadAttribute:

  Sets the attribute that will be used to find the head of the edge.
- (void) - setHeadPosition:

  Sets the position of the head attribute in the source data.
- (void) - setTailAttribute:

  Sets the attribute that will be used to find the tail of the edge.
- (void) - setTailPosition:

  Sets the position of the tail attribute in the source data.
- (void) - setTailMEP:

  Sets the flag to create missing tail nodes when loading nodes or edges.
- (void) - setHeadMEP:

  Sets the flag to create missing head nodes when loading nodes or edges.
- (void) - setLogError:

  Sets a log error file.
- (void) - setLogOff

  Truns off all the error reporting.
- (void) - registerListener:

  Registers a new listener.
- (void) - setRowReader:

  Sets the input data source.
- (void) - setGraph:

  Sets the graph where the data will be loaded.
- (void) - setLocale:

  Sets the locale that will be used to read the data.
- (void) - setType:

  Sets the type to be loaded.
- (void) - setAttributes:

  Sets the list of Attributes.
- (void) - setAttributePositions:

  Sets the list of attribute positions.
- (void) - setTimestampFormat:

  Sets a specific timestamp format.
- (void) - setFrequency:

  Sets the frequency of listener notification.

3.23.1 Detailed Description

EdgeTypeLoader class.

Specific TypeLoader implementation for edge types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.23.2 Method Documentation

3.23.2.1 - (void) registerListener: (STSTypeLoaderListener *) *tel*

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

3.23.2.2 - (void) run

See the TypeLoader class Run method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.23.2.3 - (void) runNPhases: (int) *partitions*

See the TypeLoader class RunNPhases method.

Parameters

partitions	null

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.23.2.4 - (void) runTwoPhases

See the TypeLoader class RunTwoPhases method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.23.2.5 - (void) setAttributePositions: (STSInt32List ∗) *attrsPos*

Sets the list of attribute positions.

Parameters

attrsPos	[in] Attribute positions (column index >=0).

3.23.2.6 - (void) setAttributes: (STSAttributeList ∗) *attrs*

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

3.23.2.7 - (void) setFrequency: (int) *freq*

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.23.2.8 - (void) setGraph: (STSGraph ∗) *graph*

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

3.23.2.9 - (void) setHeadAttribute: (int) *attr*

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters

attr	[in] Head Attribute

3.23.2.10 - (void) setHeadMEP: (enum STSMissingEndpoint) *mep*

Sets the flag to create missing head nodes when loading nodes or edges.

flagTrue if the nodes need to be created. False otherwise

Parameters

mep	null

3.23.2.11 - (void) setHeadPosition: (int) *pos*

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters

pos	[in] Head position

3.23.2.12 - (void) setLocale: (NSString ∗) *localeStr*

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

localeStr	[in] The locale string for the read data. See CSVReader.

3.23.2.13 - (void) setLogError: (NSString ∗) *path*

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.23.2.14 - (void) setLogOff

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

3.23.2.15 - (void) setRowReader: (STSRowReader ∗) *rr*

Sets the input data source.

Parameters

rr	[in] Input RowReader.

3.23.2.16 - (void) setTailAttribute: (int) *attr*

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

Parameters

attr	[in] Tail Attribute

3.23.2.17 - (void) setTailMEP: (enum STSMissingEndpoint) *mep*

Sets the flag to create missing tail nodes when loading nodes or edges.

flagTrue if the nodes need to be created. False otherwise

Parameters

mep	null

3.23.2.18 - (void) setTailPosition: (int) *pos*

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters

pos	[in] Tail position

3.23.2.19 - (void) setTimestampFormat: (NSString ∗) *timestampFormat*

Sets a specific timestamp format.

Parameters

timestampFormat	[in] A string with the timestamp format definition.

3.23.2.20 - (void) setType: (int) *type*

Sets the type to be loaded.

Parameters

type	[in] Type identifier.

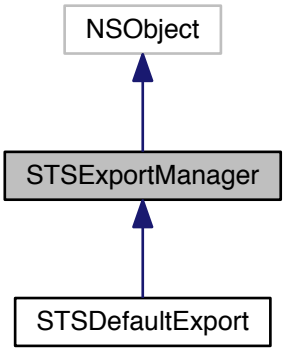
The documentation for this class was generated from the following file:

- Sparksee.h

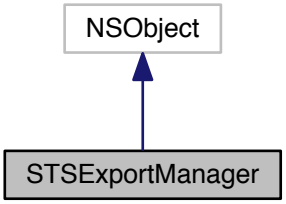
3.24 STSExportManager Class Reference

Defines how to export a graph to an external format.

Inheritance diagram for STSExportManager:



Collaboration diagram for STSExportManager:



Instance Methods

- (void) - prepare:

  Prepares the graph for the export process.
- (void) - close

  Ends the export process.
- (BOOL) - getGraph:

  Gets the graph export definition.
- (BOOL) - getNodeType:nodeExport:

  Gets the default node export definition for the given node type.
- (BOOL) - getEdgeType:edgeExport:

  Gets the default node export definition for the given edge type.
- (BOOL) - getNode:nodeExport:

  Gets the node export definition for the given node.
- (BOOL) - getEdge:edgeExport:

  Gets the edge export definition for the given edge.
- (BOOL) - enableType:

  Gets whether a node or edge type must be exported or not.

3.24.1 Detailed Description

Defines how to export a graph to an external format.

This is an interface which must be implemented by the user. While the export proces, a call for each node or edge type and node or edge object is done to get how to export that element.

It is possible to export a Graph to a diferent fortmats. Nowadays, available formats are defined in the ExportType enum.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.24.2 Method Documentation

3.24.2.1 - (void) close

Ends the export process.

It is called once after the export process.

Implemented in STSDefaultExport.

3.24.2.2 - (BOOL) enableType: (int) *type*

Gets whether a node or edge type must be exported or not.

Parameters

type	Node or edge type identifier.

Returns

If TRUE all objects of the given type will be exported, otherwise they will not be exported.

Implemented in STSDefaultExport.

3.24.2.3 - (BOOL) getEdge: (long long) *edge* edgeExport:(STSEdgeExport *) *edgeExport*

Gets the edge export definition for the given edge.

Parameters

edge	Edge identifier.
edgeExport	[out] The EdgeExport which defines how to export given edge.

Returns

TRUE if the given EdgeExport has been updated, otherwise FALSE will be returned and the default Edge↩ Export for the type the edge belongs to will be used.

Implemented in STSDefaultExport.

3.24.2.4 - (BOOL) getEdgeType: (int) *type* edgeExport:(STSEdgeExport *) *edgeExport*

Gets the default node export definition for the given edge type.

GetEdge has a higher priority than this function. That is, only if GetEdge returns FALSE, the EdgeExport of this function will be used.

Parameters

type	[in] Edge type identifier.
edgeExport	[out] The EdgeExport which defines how to export the edges of the given type.

Returns

TRUE.

Implemented in STSDefaultExport.

3.24.2.5 - (BOOL) getGraph: (STSGraphExport *) *graphExport*

Gets the graph export definition.

Parameters

graphExport	[out] The GraphExport which defines how to export the graph.

Returns

TRUE.

Implemented in STSDefaultExport.

3.24.2.6 - (BOOL) getNode: (long long) *node* nodeExport:(STSNodeExport ∗) *nodeExport*

Gets the node export definition for the given node.

Parameters

node	Node identifier.
nodeExport	[out] The NodeExport which defines how to export given node.

Returns

TRUE if the given NodeExport has been updated, otherwise FALSE will be returned and the default Node↩ Export for the type the node belongs to will be used.

Implemented in STSDefaultExport.

3.24.2.7 - (BOOL) getNodeType: (int) *type* nodeExport:(STSNodeExport ∗) *nodeExport*

Gets the default node export definition for the given node type.

GetNode has a higher priority than this function. That is, only if GetNode returns FALSE, the NodeExport of this function will be used.

Parameters

type	[in] Node type identifier.
nodeExport	[out] The NodeExport which defines how to export the nodes of the given type.

Returns

TRUE.

Implemented in STSDefaultExport.

3.24.2.8 - (void) prepare: (STSGraph ∗) *graph*

Prepares the graph for the export process.

It is called once before the export process.

Parameters

graph	Graph to be exported.

Implemented in STSDefaultExport.

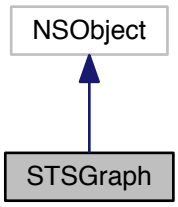
The documentation for this class was generated from the following file:

- Sparksee.h

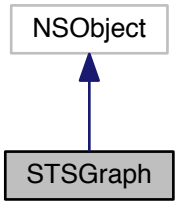
3.25 STSGraph Class Reference

Graph class.

Inheritance diagram for STSGraph:



Collaboration diagram for STSGraph:



Instance Methods

- (int) - createNodeType:

  Creates a new node type.
- (long long) - createNode:

  Creates a new node instance.
- (int) - createEdgeType:directed:neighbors:

  Creates a new edge type.
- (int) - createRestrictedEdgeType:tail:head:neighbors:

Creates a new restricted edge type.

- (void) - indexNeighbors:neighbors:

  Creates or destroys the neighbors index of an edge type.
- (long long) - createEdge:tail:head:

  Creates a new edge instance.
- (long long) - createEdgeWithAttributes:tailAttr:tailV:headAttr:headV:

  Creates a new edge instance.
- (long long) - countNodes

  Gets the number of nodes.
- (long long) - countEdges

  Gets the number of edges.
- (STSEdgeData *) - getEdgeData:

  Gets information about an edge.
- (long long) - getEdgePeer:node:

  Gets the other end for the given edge.
- (void) - drop:

  Drops the given OID.
- (void) - dropWithObjects:

  Drops all the OIDs from the given collection.
- (int) - getObjectType:

  Gets the Sparksee node or edge type identifier for the given OID.
- (int) - createAttribute:name:dt:kind:

  Creates a new attribute.
- (int) - createAttributeWithDefault:name:dt:kind:defaultValue:

  Creates a new attribute with a default value.
- (int) - createSessionAttribute:dt:kind:

  Creates a new Session attribute.
- (int) - createSessionAttributeWithDefault:dt:kind:defaultValue:

  Creates a new Session attribute with a default value.
- (int) - createArrayAttribute:name:dt:size:

  Creates a new array attribute.
- (int) - createSessionArrayAttribute:dt:size:

  Creates a new Session array attribute.
- (void) - setAttributeDefaultValue:value:

  Sets a default value for an attribute.
- (void) - indexAttribute:kind:

  Updates the index of the given attribute.
- (void) - getAttributeInValue:attr:value:

  Gets the Value for the given attribute and OID.
- (STSValue *) - getAttributeWithOid:attr:

  Gets the Value for the given attribute and OID.
- (STSTextStream *) - getAttributeText:attr:

  Gets the read-only TextStream for the given text attribute and OID.
- (void) - setAttributeText:attr:tstream:

  Sets the writable TextStream for the given text attribute and OID.
- (STSValueArray *) - getArrayAttribute:attr:

  Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.
- (STSValueArray *) - setArrayAttribute:attr:value:

  Sets all the elements of the ValueArray for the given array attribute and OID and returns it.
- (void) - setArrayAttributeVoid:attr:value:

  Sets all the elements of the ValueArray for the given array attribute and OID.

- (void) - setAttribute:attr:value:

  Sets the Value for the given attribute and OID.
- (STSAttributeStatistics *) - getAttributeStatistics:basic:

  Gets statistics from the given attribute.
- (long long) - getAttributeIntervalCount:lower:includeLower:higher:includeHigher:

  Gets how many objects have a value into the given range for the given attribute.
- (int) - findType:

  Gets the Sparksee type identifier for the given type name.
- (STSType *) - getType:

  Gets information about the given type.
- (void) - removeType:

  Removes the given type.
- (void) - renameTypeWithName:newName:

  Renames a type.
- (void) - renameType:newName:

  Renames a type.
- (int) - findAttribute:name:

  Gets the Sparksee attribute identifier for the given type identifier and attribute name.
- (STSAttribute *) - getAttribute:

  Gets information about the given attribute.
- (void) - removeAttribute:

  Removes the given attribute.
- (void) - renameAttribute:newName:

  Renames an attribute.
- (long long) - findObject:value:

  Finds one object having the given Value for the given attribute.
- (long long) - findOrCreateObject:value:

  Finds one object having the given Value for the attribute or it creates one does not exist any.
- (STSObjects *) - selectWithType:

  Selects all OIDs belonging to the given type.
- (STSObjects *) - selectWithAttrValue:cond:value:

  Selects all OIDs satisfying the given condition for the given attribute.
- (STSObjects *) - selectWithAttrValues:cond:lower:higher:

  Selects all OIDs satisfying the given condition for the given attribute.
- (STSObjects *) - selectWithAttrValueRestriction:cond:value:restriction:

  Selects all OIDs satisfying the given condition for the given attribute.
- (STSObjects *) - selectWithAttrValuesRestriction:cond:lower:higher:restriction:

  Selects all OIDs satisfying the given condition for the given attribute.
- (STSKeyValues *) - topkWithAttr:order:k:

  Gets a KeyValues iterator as a result of the TopK operation.
- (STSKeyValues *) - topkWithAttrValue:operation:lower:order:k:

  Gets a KeyValues iterator as a result of the TopK operation.
- (STSKeyValues *) - topkWithAttrValues:operation:lower:higher:order:k:

  Gets a KeyValues iterator as a result of the TopK operation.
- (STSKeyValues *) - topK:order:k:restriction:

  Gets a KeyValues iterator as a result of the TopK operation.
- (STSObjects *) - explode:etype:dir:

  Selects all edges from or to the given node OID and for the given edge type.
- (STSObjects *) - explodeWithObjects:etype:dir:

  Selects all edges from or to each of the node OID in the given collection and for the given edge type.
- (long long) - degree:etype:dir:

Gets the number of edges from or to the given node OID and for the given edge type.

- (STSObjects *) - neighbors:etype:dir:

  Selects all neighbor nodes from or to the given node OID and for the given edge type.
- (STSObjects *) - neighborsWithObjects:etype:dir:

  Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.
- (STSObjects *) - edges:tail:head:

  Gets all the edges of the given type between two given nodes (tail and head).
- (long long) - findEdge:tail:head:

  Gets any of the edges of the given type between two given nodes (tail and head).
- (long long) - findOrCreateEdge:tail:head:

  Gets any of the edges of the specified type between two given nodes (tail and head).
- (STSObjects *) - tails:

  Gets all the tails from the given edges collection.
- (STSObjects *) - heads:

  Gets all the heads from the given edges collection.
- (void) - tailsAndHeads:tails:heads:

  Gets all the tails and heads from the given edges collection.
- (STSTypeList *) - findNodeTypes

  Gets all existing Sparksee node type identifiers.
- (STSTypeList *) - findEdgeTypes

  Gets all existing Sparksee edge type identifiers.
- (STSTypeList *) - findTypes

  Gets all existing Sparksee node and edge type identifiers.
- (STSAttributeList *) - findAttributes:

  Gets all existing Sparksee attribute identifiers for the given type identifier.
- (STSAttributeList *) - getAttributes:

  Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.
- (STSValues *) - getValues:

  Gets the Value collection for the given attribute.
- (void) - dumpData:

  Dumps logical data to a file.
- (void) - dumpStorage:

  Dumps internal storage data to a file.
- (void) - exportGraph:type:em:

  Exports the Graph.
- (void) - backup:

  Dumps all the data to a backup file.
- (void) - encryptedBackup:keyInHex:ivInHex:

  Dumps all the data to a backup file.

3.25.1 Detailed Description

Graph class.

Each Database has a Graph associated, which is the persistent graph which contains all data stored into the graph database and is retrieved from a Session.

Check out the 'API' and the 'SPARKSEE graph database' sections in the SPARKSEE User Manual for more details on the use of this class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.25.2 Method Documentation

3.25.2.1 - (void) backup: (NSString ∗) *file*

Dumps all the data to a backup file.

See the Sparksee class Restore methods.

Parameters

file	[in] Output backup file path.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.25.2.2 - (long long) countEdges

Gets the number of edges.

Returns

The number of edges.

3.25.2.3 - (long long) countNodes

Gets the number of nodes.

Returns

The number of nodes.

3.25.2.4 - (int) createArrayAttribute: (int) *type* name:(NSString ∗) *name* dt:(enum STSDataType) *dt* size:(int) *size*

Creates a new array attribute.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Base Data type for the new array attribute elements.
size	[in] The array size.

Returns

Unique Sparksee attribute identifier.

3.25.2.5 - (int) createAttribute: (int) *type* name:(NSString ∗) *name* dt:(enum STSDataType) *dt* kind:(enum STSAttributeKind) *kind*

Creates a new attribute.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.

Returns

Unique Sparksee attribute identifier.

3.25.2.6 - (int) createAttributeWithDefault: (int) *type* name:(NSString ∗) *name* dt:(enum STSDataType) *dt* kind:(enum STSAttributeKind) *kind* defaultValue:(STSValue ∗) *defaultValue*

Creates a new attribute with a default value.

Parameters

type	[in] Sparksee node or edge type identifier.
name	[in] Unique name for the new attribute.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.
defaultValue	[in] The default value to use in each new node/edge.

Returns

Unique Sparksee attribute identifier.

3.25.2.7 - (long long) createEdge: (int) *type* tail:(long long) *tail* head:(long long) *head*

Creates a new edge instance.

Parameters

type	[in] Sparksee type identifier.
tail	[in] Source Sparksee OID.
head	[in] Target Sparksee OID.

Returns

Unique OID of the new edge instance.

3.25.2.8 - (int) createEdgeType: (NSString ∗) *name* directed:(BOOL) *directed* neighbors:(BOOL) *neighbors*

Creates a new edge type.

Parameters

name	[in] Unique name for the new edge type.
directed	[in] If TRUE, this creates a directed edge type, otherwise this creates a undirected edge type.
neighbors	[in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns

Unique Sparksee type identifier.

3.25.2.9 - (long long) createEdgeWithAttributes: (int) *type* tailAttr:(int) *tailAttr* tailV:(STSValue ∗) *tailV* headAttr:(int) *headAttr* headV:(STSValue ∗) *headV*

Creates a new edge instance.

The tail of the edge will be any node having the given tailV Value for the given tailAttr attribute identifier, and the head of the edge will be any node having the given headV Value for the given headAttr attribute identifier.

Parameters

type	[in] Sparksee type identifier.
tailAttr	[in] Sparksee attribute identifier.
tailV	[in] Value.
headAttr	[in] Sparksee attribute identifier.
headV	[in] Value.

Returns

Unique OID of the new edge instance.

3.25.2.10 - (long long) createNode: (int) *type*

Creates a new node instance.

Parameters

type	[in] Sparksee type identifier.

Returns

Unique OID of the new node instance.

3.25.2.11 - (int) createNodeType: (NSString ∗) *name*

Creates a new node type.

Parameters

name	[in] Unique name for the new node type.

Returns

Unique Sparksee type identifier.

3.25.2.12 - (int) createRestrictedEdgeType: (NSString ∗) *name* tail:(int) *tail* head:(int) *head* neighbors:(BOOL) *neighbors*

Creates a new restricted edge type.

Parameters

name	[in] Unique name for the new edge type.
tail	[in] Tail Sparksee node type identifier.
head	[in] Head Sparksee node type identifier.
neighbors	[in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns

Unique Sparksee type identifier.

3.25.2.13 - (int) createSessionArrayAttribute: (int) *type* dt:(enum STSDataType) *dt* size:(int) *size*

Creates a new Session array attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Base Data type for the new array attribute elements.
size	[in] The array size.

Returns

Unique Sparksee attribute identifier.

3.25.2.14 - (int) createSessionAttribute: (int) *type* dt:(enum STSDataType) *dt* kind:(enum STSAttributeKind) *kind*

Creates a new Session attribute.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.

Returns

Unique Sparksee attribute identifier.

3.25.2.15 - (int) createSessionAttributeWithDefault: (int) *type* dt:(enum STSDataType) *dt* kind:(enum STSAttributeKind) *kind* defaultValue:(STSValue *) *defaultValue*

Creates a new Session attribute with a default value.

Session attributes are exclusive for the Session (just its Session can use the attribute) and are automatically removed when the Session is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters

type	[in] Sparksee node or edge type identifier.
dt	[in] Data type for the new attribute.
kind	[in] Attribute kind.
defaultValue	[in] The default value to use in each new node/edge.

Returns

Unique Sparksee attribute identifier.

3.25.2.16 - (long long) degree: (long long) *oid* etype:(int) *etype* dir:(enum STSEdgesDirection) *dir*

Gets the number of edges from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

The number of edges.

3.25.2.17 - (void) drop: (long long) *oid*

Drops the given OID.

It also removes its egdges as well as its attribute values.

Parameters

oid	[in] Sparksee OID to be removed.

3.25.2.18 - (void) dropWithObjects: (STSObjects ∗) *objs*

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs the same operation for each object in the given set.

Parameters

objs	[in] Objects collection with the OIDs to be removed.

3.25.2.19 - (void) dumpData: (NSString ∗) *file*

Dumps logical data to a file.

Parameters

file	[in] Output file path.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.25.2.20 - (void) dumpStorage: (NSString ∗) *file*

Dumps internal storage data to a file.

Parameters

file	[in] Output file path.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.25.2.21 - (STSObjects∗) edges: (int) *etype* tail:(long long) *tail* head:(long long) *head*

Gets all the edges of the given type between two given nodes (tail and head).

Parameters

etype	[in] Sparksee edge type identifier.

Parameters

tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Objects instance.

3.25.2.22 - (void) encryptedBackup: (NSString ∗) *file* keyInHex:(NSString ∗) *keyInHex* ivInHex:(NSString ∗) *ivInHex*

Dumps all the data to a backup file.

See the Sparksee class RestoreEncryptedBackup methods.

Parameters

file	[in] Output backup file path.
keyInHex	[In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes).
ivInHex	[In] The AES Initialization Vector as an hexadecimal string (16 bytes).

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.25.2.23 - (STSObjects∗) explode: (long long) *oid* etype:(int) *etype* dir:(enum STSEdgesDirection) *dir*

Selects all edges from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

3.25.2.24 - (STSObjects∗) explodeWithObjects: (STSObjects ∗) *objs* etype:(int) *etype* dir:(enum STSEdgesDirection) *dir*

Selects all edges from or to each of the node OID in the given collection and for the given edge type.

Parameters

objs	[in] Sparksee node OID collection.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

3.25.2.25 - (void) exportGraph: (NSString ∗) *file* type:(enum STSExportType) *type* em:(STSExportManager ∗) *em*

Exports the Graph.

Parameters

file	[in] Output file.
type	[in] Export type.
em	[in] Defines how to do the export for each graph object.

Exceptions

System.IO.IOException	null

3.25.2.26 - (int) findAttribute: (int) *type* name:(NSString ∗) *name*

Gets the Sparksee attribute identifier for the given type identifier and attribute name.

Parameters

type	[in] Sparksee type identifier.
name	[in] Unique attribute name.

Returns

The Sparksee attribute identifier for the given type and attribute name or InvalidAttribute if there is no attribute with the given name for the given type.

3.25.2.27 - (STSAttributeList∗) findAttributes: (int) *type*

Gets all existing Sparksee attribute identifiers for the given type identifier.

Parameters

type	[in] Sparksee type identifier.

Returns

Sparksee attribute identifier list.

3.25.2.28 - (long long) findEdge: (int) *etype* tail:(long long) *tail* head:(long long) *head*

Gets any of the edges of the given type between two given nodes (tail and head).

If there are more than one, then any of them will be returned. And in case there are no edge between the given tail and head, the Objects InvalidOID will be returned.

Parameters

etype	[in] Sparksee edge type identifier.
tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Any of the edges or the Objects InvalidOID.

3.25.2.29 - (STSTypeList*) findEdgeTypes

Gets all existing Sparksee edge type identifiers.

Returns

Sparksee edge type identifier list.

3.25.2.30 - (STSTypeList*) findNodeTypes

Gets all existing Sparksee node type identifiers.

Returns

Sparksee node type identifier list.

3.25.2.31 - (long long) findObject: (int) *attr* value:(STSValue *) *value*

Finds one object having the given Value for the given attribute.

If there are more than one, then any of them will be returned. And in case there are no object having this Value, the Objects InvalidOID will be returned.

Parameters

attr	[in] Sparksee attribute identifier.
value	[in] Value.

Returns

Sparksee OID or the Objects InvalidOID.

3.25.2.32 - (long long) findOrCreateEdge: (int) *etype* tail:(long long) *tail* head:(long long) *head*

Gets any of the edges of the specified type between two given nodes (tail and head).

If it can not find any edge of this type between them it tries to create a new one.

Parameters

etype	[in] Sparksee edge type identifier.
tail	[in] Tail node identifier.
head	[in] Head node identifier.

Returns

Any of the edges or the Objects InvalidOID.

3.25.2.33 - (long long) findOrCreateObject: (int) *attr* value:(STSValue ∗) *value*

Finds one object having the given Value for the attribute or it creates one does not exist any.

If the attribute is a node or edge attribute and at least one node/edge with that value is found, it returns one of them. But if it does not exist, then: If it's a node attribute it will create it and set the attribute. If it's an edge attribute it will return the InvalidOID.

Using this method with a global attribute will return the InvalidOID.

Parameters

attr	[in] Sparksee attribute identifier.
value	[in] Value.

Returns

Sparksee OID or the Objects InvalidOID.

3.25.2.34 - (int) findType: (NSString ∗) *name*

Gets the Sparksee type identifier for the given type name.

Parameters

name	[in] Unique type name.

Returns

The Sparksee type identifier for the given type name or the Type InvalidType if there is no type with the given name.

3.25.2.35 - (STSTypeList∗) findTypes

Gets all existing Sparksee node and edge type identifiers.

Returns

Sparksee node and edge type identifier list.

3.25.2.36 - (STSValueArray∗) getArrayAttribute: (long long) *oid* attr:(int) *attr*

Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.

Returns

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

3.25.2.37 - (STSAttribute∗) getAttribute: (int) *attr*

Gets information about the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

Returns

The Attribute for the given Sparksee attribute identifier.

3.25.2.38 - (long long) getAttributeIntervalCount: (int) *attr* lower:(STSValue ∗) *lower* includeLower:(BOOL) *includeLower* higher:(STSValue ∗) *higher* includeHigher:(BOOL) *includeHigher*

Gets how many objects have a value into the given range for the given attribute.

This only works for the attributes with the AttributeKind Indexed or Unique.

Given values must belong to the same DataType than the attribute.

Parameters

attr	[in] Sparksee attribute identifier.
lower	[in] Lower bound Value of the range.
includeLower	[in] If TRUE, include lower bound Value of the range.
higher	[in] Higher bound Value of the range.
includeHigher	[in] If TRUE, include higher bound Value of the range.

Returns

Number of objects having a value into the given range.

3.25.2.39 - (void) getAttributeInValue: (long long) *oid* attr:(int) *attr* value:(STSValue ∗) *value*

Gets the Value for the given attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
value	[in|out] Value for the given attribute and for the given OID.

3.25.2.40 - (STSAttributeList*) getAttributes: (long long) *oid*

Gets all Sparksee attribute identifiers with a non-NULL value for the given Sparksee OID.

Parameters

oid	[in] Sparksee OID.

Returns

Sparksee attribute identifier list.

3.25.2.41 - (STSAttributeStatistics*) getAttributeStatistics: (int) *attr* basic:(BOOL) *basic*

Gets statistics from the given attribute.

The statistics can only be obtained from Indexed or Unique attributes.

Parameters

attr	[in] Sparksee attribute identifier.
basic	[in] If FALSE all statistics are computed, if TRUE just those statistics marked as basic will be computed (see description of the AttributeStatistics class). Of course, computing just basic statistics will be faster than computing all of them.

Returns

An AttributeStatistics instace.

3.25.2.42 - (STSTextStream*) getAttributeText: (long long) *oid* attr:(int) *attr*

Gets the read-only TextStream for the given text attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.

Returns

A TextStream. This returns a TextStream to read.

3.25.2.43 - (STSValue*) getAttributeWithOid: (long long) *oid* attr:(int) *attr*

Gets the Value for the given attribute and OID.

The other version of this call, where the Value is an output parameter instead of the return, is better because it allows the user to reuse an existing Value instance, whereas this call always creates a new Value instance to be returned.

It never returns NULL. Thus, in case the OID has a NULL value for the attribute it returns a NULL Value instance.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.

Returns

A new Value instance having the attribute value for the given OID.

3.25.2.44 - (STSEdgeData*) getEdgeData: (long long) *edge*

Gets information about an edge.

Parameters

edge	[in] Sparksee edge identifier.

Returns

An EdgeData instance.

3.25.2.45 - (long long) getEdgePeer: (long long) *edge* node:(long long) *node*

Gets the other end for the given edge.

Parameters

edge	[in] Sparksee edge identifier.
node	[in] Sparksee node identifier. It must be one of the ends of the edge.

Returns

The other end of the edge.

3.25.2.46 - (int) getObjectType: (long long) *oid*

Gets the Sparksee node or edge type identifier for the given OID.

Parameters

oid	[in] Sparksee OID.

Returns

Sparksee node or edge type identifier.

3.25.2.47 - (STSType∗) getType: (int) *type*

Gets information about the given type.

Parameters

type	[in] Sparksee type identifier.

Returns

The Type for the given Sparksee type identifier.

3.25.2.48 - (STSValues∗) getValues: (int) *attr*

Gets the Value collection for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

Returns

Returns a Values object.

3.25.2.49 - (STSObjects∗) heads: (STSObjects ∗) *edges*

Gets all the heads from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.

Returns

The heads collection.

3.25.2.50 - (void) indexAttribute: (int) *attr* kind:(enum STSAttributeKind) *kind*

Updates the index of the given attribute.

This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one is Indexed or Unique.

Parameters

attr	[in] Sparksee attribute identifier.
kind	[in] Attribute kind.

3.25.2.51 - (void) indexNeighbors: (int) *edgeType* neighbors:(BOOL) *neighbors*

Creates or destroys the neighbors index of an edge type.

Parameters

edgeType	[in] Sparksee Edge type identifier.
neighbors	[in] If TRUE, it indexes the neighbor nodes, otherwise it removes the index.

3.25.2.52 - (STSObjects*) neighbors: (long long) *oid* etype:(int) *etype* dir:(enum STSEdgesDirection) *dir*

Selects all neighbor nodes from or to the given node OID and for the given edge type.

Parameters

oid	[in] Sparksee node OID.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

3.25.2.53 - (STSObjects*) neighborsWithObjects: (STSObjects *) *objs* etype:(int) *etype* dir:(enum STSEdgesDirection) *dir*

Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

Parameters

objs	[in] Sparksee node OID collection.
etype	[in] Sparksee edge type identifier.
dir	[in] Direction.

Returns

Objects instance.

3.25.2.54 - (void) removeAttribute: (int) *attr*

Removes the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.

3.25.2.55 - (void) removeType: (int) *type*

Removes the given type.

This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

Parameters

type	[in] Sparksee type identifier.

3.25.2.56 - (void) renameAttribute: (int) *attr* newName:(NSString ∗) *newName*

Renames an attribute.

The new name must be available.

Parameters

attr	[in] Sparksee attribute identifier.
newName	[in] The new name for the attribute.

3.25.2.57 - (void) renameType: (int) *type* newName:(NSString ∗) *newName*

Renames a type.

The new name must be available.

Parameters

type	[in] The type to be renamed.
newName	[in] The new name for the type.

3.25.2.58 - (void) renameTypeWithName: (NSString ∗) *oldName* newName:(NSString ∗) *newName*

Renames a type.

The new name must be available.

Parameters

oldName	[in] The current name of the type to be renamed.
newName	[in] The new name for the type.

3.25.2.59 - (STSObjects∗) selectWithAttrValue: (int) *attr* cond:(enum STSCondition) *cond* value:(STSValue ∗) *value*

Selects all OIDs satisfying the given condition for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied.
value	[in] Value to be satisfied.

Returns

Objects instance.

3.25.2.60 - (STSObjects∗) selectWithAttrValueRestriction: (int) *attr* cond:(enum STSCondition) *cond* value:(STSValue ∗) *value* restriction:(STSObjects ∗) *restriction*

Selects all OIDs satisfying the given condition for the given attribute.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied.
value	[in] Value to be satisfied.
restriction	[in] Objects to limit the select in this set of objects.

Returns

Objects instance.

3.25.2.61 - (STSObjects∗) selectWithAttrValues: (int) *attr* cond:(enum STSCondition) *cond* lower:(STSValue ∗) *lower* higher:(STSValue ∗) *higher*

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two Value arguments.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied. It must be the Between Condition.
lower	[in] Lower-bound Value to be satisfied.
higher	[in] Higher-bound Value to be satisfied.

Returns

Objects instance.

3.25.2.62 - (STSObjects∗) selectWithAttrValuesRestriction: (int) *attr* cond:(enum STSCondition) *cond* lower:(STSValue ∗) *lower* higher:(STSValue ∗) *higher* restriction:(STSObjects ∗) *restriction*

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two Value arguments.

Parameters

attr	[in] Sparksee attribute identifier.
cond	[in] Condition to be satisfied. It must be the Between Condition.
lower	[in] Lower-bound Value to be satisfied.
higher	[in] Higher-bound Value to be satisfied.
restriction	[in] Objects to limit the select in this set of objects.

Returns

Objects instance.

3.25.2.63 - (STSObjects∗) selectWithType: (int) *type*

Selects all OIDs belonging to the given type.

Parameters

type	[in] Sparksee type identifier.

Returns

Objects instance.

3.25.2.64 - (STSValueArray∗) setArrayAttribute: (long long) *oid* attr:(int) *attr* value:(STSValue ∗) *value*

Sets all the elements of the ValueArray for the given array attribute and OID and returns it.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray. And you can get it again with the GetArrayAttribute operation.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.
value	[in] Value for all the array elements of the given attribute and OID.

Returns

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

3.25.2.65 - (void) setArrayAttributeVoid: (long long) *oid* attr:(int) *attr* value:(STSValue ∗) *value*

Sets all the elements of the ValueArray for the given array attribute and OID.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray which can be obtained using the GetArrayAttribute operation.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee array attribute identifier.
value	[in] Value for all the array elements of the given attribute and OID.

3.25.2.66 - (void) setAttribute: (long long) *oid* attr:(int) *attr* value:(STSValue ∗) *value*

Sets the Value for the given attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
value	[in] Value for the given attribute and for the given OID.

3.25.2.67 - (void) setAttributeDefaultValue: (int) *attr* value:(STSValue ∗) *value*

Sets a default value for an attribute.

The default value will be applied to all the new nodes or edges.

The given value must have the same DataType as the attribute or be a NULL value to remove the current default value.

Parameters

attr	[in] The attribute.
value	[in] The default value to use for this attribute.

3.25.2.68 - (void) setAttributeText: (long long) *oid* attr:(int) *attr* tstream:(STSTextStream ∗) *tstream*

Sets the writable TextStream for the given text attribute and OID.

Parameters

oid	[in] Sparksee OID.
attr	[in] Sparksee attribute identifier.
tstream	[in] New Text value. This corresponds to a TextStream to write.

3.25.2.69 - (STSObjects*) tails: (STSObjects *) *edges*

Gets all the tails from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.

Returns

The tails collection.

3.25.2.70 - (void) tailsAndHeads: (STSObjects *) *edges* tails:(STSObjects *) *tails* heads:(STSObjects *) *heads*

Gets all the tails and heads from the given edges collection.

Parameters

edges	[in] Sparksee edge identifier collection.
tails	[in|out] If not NULL, all the tails will be stored here.
heads	[in|out] If not NULL, all the heads will be stored here.

3.25.2.71 - (STSKeyValues*) topK: (int) *attribute* order:(enum STSOrder) *order* k:(int) *k* restriction:(STSObjects *) *restriction*

Gets a KeyValues iterator as a result of the TopK operation.

restrictonObjects to limit the topk to this set of objects

Parameters

attribute	The attribute to perform the TopK on
order	The ordering semantincs of the top-k
k	The size of the TopK
restriction	null

Returns

A KeyValues instance

3.25.2.72 - (STSKeyValues*) topkWithAttr: (int) *attribute* order:(enum STSOrder) *order* k:(int) *k*

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

3.25.2.73 - (STSKeyValues*) topkWithAttrValue: (int) *attribute* operation:(enum STSCondition) *operation* lower:(STSValue *) *lower* order:(enum STSOrder) *order* k:(int) *k*

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

3.25.2.74 - (STSKeyValues*) topkWithAttrValues: (int) *attribute* operation:(enum STSCondition) *operation* lower:(STSValue *) *lower* higher:(STSValue *) *higher* order:(enum STSOrder) *order* k:(int) *k*

Gets a KeyValues iterator as a result of the TopK operation.

Parameters

attribute	The attribute to perform the TopK on
operation	The operation to perform in the top k
lower	The lower bound Value to be satisfied
higher	The upper bound Value to be satisfied
order	The ordering semantincs of the top-k
k	The size of the TopK

Returns

A KeyValues instance

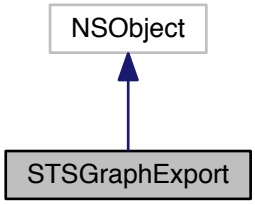
The documentation for this class was generated from the following file:

- Sparksee.h

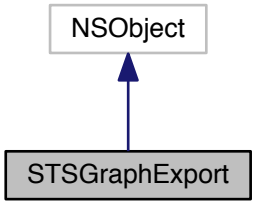
3.26 STSGraphExport Class Reference

Stores the graph exporting values.

Inheritance diagram for STSGraphExport:



Collaboration diagram for STSGraphExport:



Instance Methods

- (id) - init

  Creates a new GraphExport instance.
- (void) - setDefaults

  Sets to default values.
- (NSString ∗) - getLabel

  Gets the graph label.
- (void) - setLabel:

  Sets the graph label.

3.26.1 Detailed Description

Stores the graph exporting values.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.26.2 Method Documentation

3.26.2.1 - (NSString∗) getLabel

Gets the graph label.

Returns

The graph label.

3.26.2.2 - (void) setLabel: (NSString ∗) *label*

Sets the graph label.

Parameters

label	[in] The graph label.

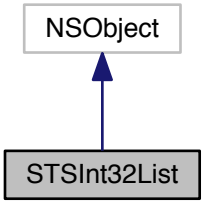
The documentation for this class was generated from the following file:

- Sparksee.h

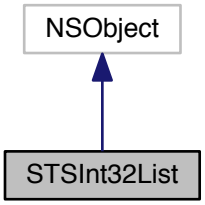
3.27 STSInt32List Class Reference

Sparksee 32-bit signed integer list.

Inheritance diagram for STSInt32List:



Collaboration diagram for STSInt32List:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - add:

  Adds an 32-bit signed integer at the end of the list.
- (void) - clear

  Clears the list.
- (id) - initWithArray:

  Creates a new Int32List instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new Int32List instance from the given NSEnumerator.
- (STSInt32ListIterator ∗) - iterator

  Gets a new Int32ListIterator.

3.27.1 Detailed Description

Sparksee 32-bit signed integer list.

It stores a 32-bit signed integer list.

Use Int32ListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.27.2 Method Documentation

3.27.2.1 - (void) add: (int) *value*

Adds an 32-bit signed integer at the end of the list.

Parameters

value	[in] The integer.

3.27.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.27.2.3 - (id) init

Constructor.

This creates an empty list.

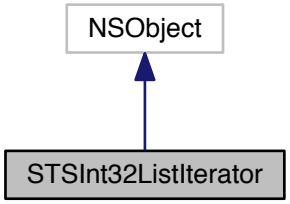
The documentation for this class was generated from the following file:

- Sparksee.h

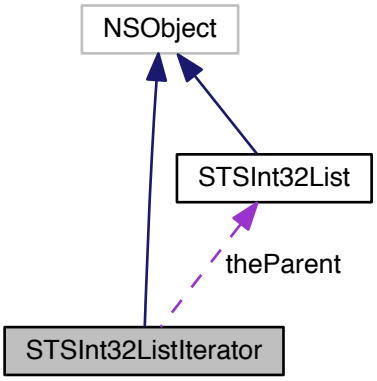
3.28 STSInt32ListIterator Class Reference

Int32List iterator class.

Inheritance diagram for STSInt32ListIterator:



Collaboration diagram for STSInt32ListIterator:



Instance Methods

- (int) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.28.1 Detailed Description

Int32List iterator class.

Iterator to traverse all the integer into a Int32List instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.28.2 Method Documentation

3.28.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.28.2.2 - (int) next

Moves to the next element.

Returns

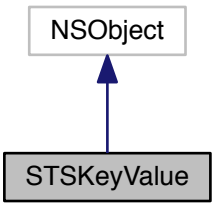
The next element.

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

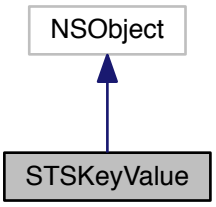
- Sparksee.h

3.29 STSKeyValue Class Reference

Inheritance diagram for STSKeyValue:



Collaboration diagram for STSKeyValue:



Instance Methods

- (NSString ∗) - stringValue

  Returns the receiver's value as a human-readable string.
- (BOOL) - isEqual:

  Check if both Value instances are equal.
- (NSUInteger) - hash

  Get the hash value of this Value.

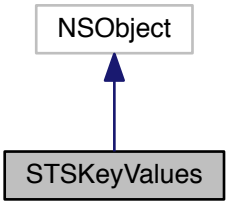
The documentation for this class was generated from the following file:

- Sparksee.h

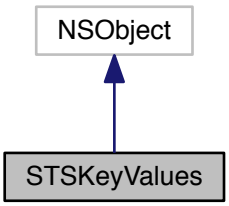
3.30 STSKeyValues Class Reference

Value set class.

Inheritance diagram for STSKeyValues:



Collaboration diagram for STSKeyValues:



Instance Methods

- (BOOL) - hasNext

  Checks if the KeyValues has more KeyValue pairs.
- (STSKeyValue ∗) - next

  Gets the next KeyValue pair.
- (void) - nextKeyValue:

  Gets the next KeyValue pair.
- (void) - close

  Closes the KeyValues instance.
- (BOOL) - isClosed

  Check if the KeyValues instance is closed.

3.30.1 Detailed Description

Value set class.

This is a set of Value instances, that is there is no duplicated elements.

Use a ValuesIterator to traverse all the elements into the set.

When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.30.2 Method Documentation

3.30.2.1 - (void) close

Closes the KeyValues instance.

It must be called to ensure the integrity of all data.

3.30.2.2 - (BOOL) hasNext

Checks if the KeyValues has more KeyValue pairs.

Returns

Returns ture if there are more KeyValue pairs

3.30.2.3 - (STSKeyValue*) next

Gets the next KeyValue pair.

Returns

Returns the next KeyValue pair

3.30.2.4 - (void) nextKeyValue: (STSKeyValue *) *kv*

Gets the next KeyValue pair.

Parameters

kv	Returns the next KeyValue pair

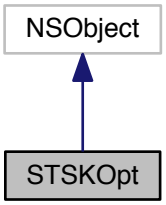
The documentation for this class was generated from the following file:

- Sparksee.h

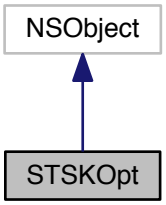
3.31 STSKOpt Class Reference

KOpt class.

Inheritance diagram for STSKOpt:



Collaboration diagram for STSKOpt:



Instance Methods

- (id) - initWithSession:

  Creates a new instance.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - setEdgeWeightAttributeType:

  Sets the attribute to use as edge weight.
- (double) - getCurrentCost

  Returns tour cost.

- (STSOidList *) - getCurrentTour

  Returns tour as a list of nodes.
- (void) - setCurrentTour:

  Sets current tour as a list of nodes.
- (void) - setMaxIterations:

  Sets maximum number of iterations.
- (void) - setTimeLimit:

  Limits execution time.
- (void) - runTwoOpt

  Runs 2-Opt local search.
- (void) - runThreeOpt

  Runs 3-Opt local search.
- (void) - close

  Closes the KOpt instance.
- (BOOL) - isClosed

  Check if the KOpt instance is closed.

3.31.1 Detailed Description

KOpt class.

Implements the 2-Opt and 3-Opt algorithms

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.31.2 Method Documentation

3.31.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.31.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Exceptions

System.ApplicationException	null

3.31.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

sparksee::gdb::Error

Parameters

type	[in] Node type.

Exceptions

System.ApplicationException	null

3.31.2.4 - (void) close

Closes the KOpt instance.

It must be called to ensure the integrity of all data.

3.31.2.5 - (id) initWithSession: (STSSession ∗) *session*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform algorithm.

3.31.2.6 - (void) setCurrentTour: (STSOidList ∗) *tour*

Sets current tour as a list of nodes.

Parameters

tour	[in] Initial tour that needs to be improved.

3.31.2.7 - (void) setEdgeWeightAttributeType: (int) *attr*

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double.

sparksee::gdb::Error

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

Exceptions

System.ApplicationException	null

3.31.2.8 - (void) setMaxIterations: (long long) *maxIterations*

Sets maximum number of iterations.

By default the algorithm will run until no improvement can be made in the current tour.

Parameters

maxIterations	[in] Maximum number of iterations

3.31.2.9 - (void) setTimeLimit: (long long) *maxTime*

Limits execution time.

Parameters

maxTime	[in] Time limit in miliseconds

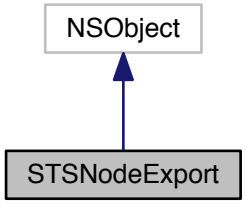
The documentation for this class was generated from the following file:

- Sparksee.h

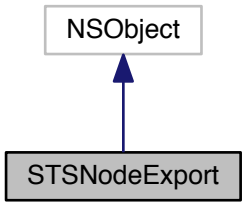
3.32 STSNodeExport Class Reference

Stores the node exporting values.

Inheritance diagram for STSNodeExport:



Collaboration diagram for STSNodeExport:



Instance Methods

- (id) - init

  Creates a new instance.
- (void) - setDefaults

  Sets to default values.
- (NSString ∗) - getLabel

  Gets the node label.
- (void) - setLabel:

  Sets the node label.
- (enum STSNodeShape) - getShape

  Gets the node shape.
- (void) - setShape:

  Sets the node shape.
- (int) - getColorRGB

  Gets the node color.
- (void) - setColorRGB:

  Sets the node color.
- (int) - getLabelColorRGB

  Gets the node label color.
- (void) - setLabelColorRGB:

  Sets the node label color.
- (int) - getHeight

  Gets the node height.
- (void) - setHeight:

  Sets the node height.
- (int) - getWidth

  Gets the node width.
- (void) - setWidth:

  Gets the node width.
- (BOOL) - isFit

  Gets whether the node size is fitted to the label or not.
- (void) - setFit:

  Sets the node fit property.
- (int) - getFontSize

Gets the node label font size.

- (void) - setFontSize:

  Sets the node label font size.

- (void) - getColorRed:green:blue:alpha:

  Get the node color separated in RGBA.

- (void) - setColorRed:green:blue:alpha:

  Set the node color with separated RGBA components.

- (void) - getLabelColorRed:green:blue:alpha:

  Get the node label color separated in RGBA.

- (void) - setLabelColorRed:green:blue:alpha:

  Set the node label color with separated RGBA components.

3.32.1 Detailed Description

Stores the node exporting values.

When 'fit' is set to TRUE, then 'height' and 'width' will be ignored.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

Shape: Box.

Color: 10863606 (0xa5c3f6).

Label color: 0 (Ox000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.32.2 Method Documentation

3.32.2.1 - (void) getColorRed: (double *) *red* green:(double *) *green* blue:(double *) *blue* alpha:(double *) *alpha*

Get the node color separated in RGBA.

Parameters

red	[out] The red color component ([0..1]).
green	[out] The green color component ([0..1]).
blue	[out] The blue color component ([0..1]).
alpha	[out] The alpha component ([0..1]).

3.32.2.2 - (int) getColorRGB

Gets the node color.

Returns

The node color.

3.32.2.3 - (int) getFontSize

Gets the node label font size.

Returns

The node label font size.

3.32.2.4 - (int) getHeight

Gets the node height.

Returns

The node height in pixels.

3.32.2.5 - (NSString∗) getLabel

Gets the node label.

Returns

The node label.

3.32.2.6 - (void) getLabelColorRed: (double ∗) *red* green:(double ∗) *green* blue:(double ∗) *blue* alpha:(double ∗) *alpha*

Get the node label color separated in RGBA.

Parameters

red	[out] The red color component ([0..1]).
green	[out] The green color component ([0..1]).
blue	[out] The blue color component ([0..1]).
alpha	[out] The alpha component ([0..1]).

3.32.2.7 - (int) getLabelColorRGB

Gets the node label color.

Returns

The node label color.

3.32.2.8 - (enum STSNodeShape) getShape

Gets the node shape.

Returns

The node shape.

3.32.2.9 - (int) getWidth

Gets the node width.

Returns

The node width in pixels.

3.32.2.10 - (BOOL) isFit

Gets whether the node size is fitted to the label or not.

Returns

If TRUE, then the node size is fitted to the label, otherwise the size is fixed with the values of 'height' and 'width'.

3.32.2.11 - (void) setColorRed: (double) *red* green:(double) *green* blue:(double) *blue* alpha:(double) *alpha*

Set the node color with separated RGBA components.

Parameters

red	[in] The red color component ([0..1]).
green	[in] The green color component ([0..1]).
blue	[in] The blue color component ([0..1]).
alpha	[in] The alpha component ([0..1]).

3.32.2.12 - (void) setColorRGB: (int) *color*

Sets the node color.

Parameters

color	The node color.

3.32.2.13 - (void) setFit: (BOOL) *fit*

Sets the node fit property.

Parameters

fit	[in] If TRUE, then the node size is fitted to the label ('height' and 'width' will be ignored), otherwise the size is fixed with the values of 'height' and 'width'.

3.32.2.14 - (void) setFontSize: (int) *size*

Sets the node label font size.

Parameters

size	[in] The node label font size.

3.32.2.15 - (void) setHeight: (int) *height*

Sets the node height.

Parameters

height	[in] The node height in pixels.

3.32.2.16 - (void) setLabel: (NSString ∗) *label*

Sets the node label.

Parameters

label	[in] The node label.

3.32.2.17 - (void) setLabelColorRed: (double) *red* green:(double) *green* blue:(double) *blue* alpha:(double) *alpha*

Set the node label color with separated RGBA components.

Parameters

red	[in] The red color component ([0..1]).
green	[in] The green color component ([0..1]).
blue	[in] The blue color component ([0..1]).
alpha	[in] The alpha component ([0..1]).

3.32.2.18 - (void) setLabelColorRGB: (int) *color*

Sets the node label color.

Parameters

color	[in] The node label color.

3.32.2.19 - (void) setShape: (enum STSNodeShape) *shape*

Sets the node shape.

Parameters

shape	[in] The node shape.

3.32.2.20 - (void) setWidth: (int) *width*

Gets the node width.

Parameters

width	The node width in pixels.

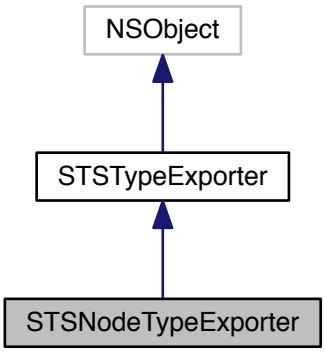
The documentation for this class was generated from the following file:

- Sparksee.h

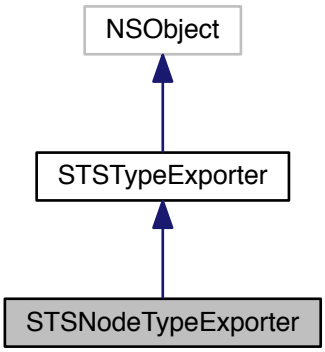
3.33 STSNodeTypeExporter Class Reference

NodeTypeExporter class.

Inheritance diagram for STSNodeTypeExporter:



Collaboration diagram for STSNodeTypeExporter:



Instance Methods

- (id) - init
  Creates a new instance.
- (id) - initWithRowWriter:graph:type:attrs:
  Creates a new instance.
- (void) - run
  See the TypeExporter class Run method.
- (void) - registerListener:
  Registers a new listener.
- (void) - setRowWriter:
  Sets the output data destination.
- (void) - setGraph:
  Sets the graph that will be exported.
- (void) - setType:
  Sets the type to be exported.
- (void) - setAttributes:
  Sets the list of Attributes.
- (void) - setFrequency:
  Sets the frequency of listener notification.
- (void) - setHeader:
  Sets the presence of a header row.

3.33.1 Detailed Description

NodeTypeExporter class.

Specific TypeExporter implementation for node types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.33.2 Method Documentation

3.33.2.1 - (id) initWithRowWriter: (STSRowWriter ∗) *rowWriter* graph:(STSGraph ∗) *graph* type:(int) *type* attrs:(STSAttributeList ∗) *attrs*

Creates a new instance.

Parameters

rowWriter	[in] Output RowWriter.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be exported.

3.33.2.2 - (void) registerListener: (STSTypeExporterListener ∗) *tel*

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

3.33.2.3 - (void) run

See the TypeExporter class Run method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeExporter.

3.33.2.4 - (void) setAttributes: (STSAttributeList ∗) *attrs*

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

3.33.2.5 - (void) setFrequency: (int) *freq*

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.33.2.6 - (void) setGraph: (STSGraph ∗) *graph*

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

3.33.2.7 - (void) setHeader: (BOOL) *header*

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

3.33.2.8 - (void) setRowWriter: (STSRowWriter ∗) *rw*

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

3.33.2.9 - (void) setType: (int) *type*

Sets the type to be exported.

Parameters

type	[in] Type identifier.

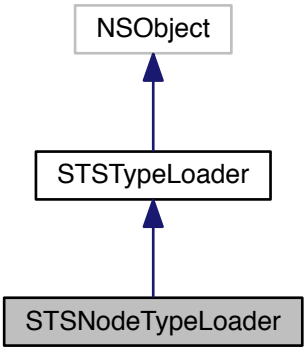
The documentation for this class was generated from the following file:

- Sparksee.h

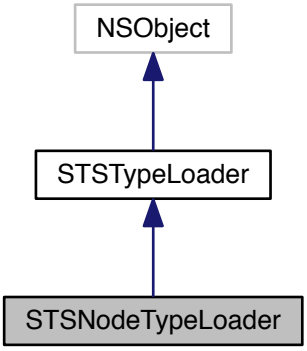
3.34 STSNodeTypeLoader Class Reference

NodeTypeLoader class.

Inheritance diagram for STSNodeTypeLoader:



Collaboration diagram for STSNodeTypeLoader:



Instance Methods

- (id) - init

  Creates a new instance.
- (id) - initWithRowReader:graph:type:attrs:attrsPos:

  Creates a new instance.
- (void) - run

  See the TypeLoader class Run method.
- (void) - runTwoPhases

  See the TypeLoader class RunTwoPhases method.
- (void) - runNPhases:

  See the TypeLoader class RunNPhases method.

- (void) - setLogError:

  Sets a log error file.
- (void) - setLogOff

  Truns off all the error reporting.
- (void) - registerListener:

  Registers a new listener.
- (void) - setRowReader:

  Sets the input data source.
- (void) - setGraph:

  Sets the graph where the data will be loaded.
- (void) - setLocale:

  Sets the locale that will be used to read the data.
- (void) - setType:

  Sets the type to be loaded.
- (void) - setAttributes:

  Sets the list of Attributes.
- (void) - setAttributePositions:

  Sets the list of attribute positions.
- (void) - setTimestampFormat:

  Sets a specific timestamp format.
- (void) - setFrequency:

  Sets the frequency of listener notification.

3.34.1 Detailed Description

NodeTypeLoader class.

Specific TypeLoader implementation for node types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.34.2 Method Documentation

3.34.2.1 - (id) initWithRowReader: (STSRowReader ∗) *rowReader* graph:(STSGraph ∗) *graph* type:(int) *type* attrs:(STSAttributeList ∗) *attrs* attrsPos:(STSInt32List ∗) *attrsPos*

Creates a new instance.

Parameters

rowReader	[in] Input RowReader.
graph	[in] Graph.
type	[in] Type identifier.
attrs	[in] Attribute identifiers to be loaded.
attrsPos	[in] Attribute positions (column index >=0).

3.34.2.2 - (void) registerListener: (STSTypeLoaderListener ∗) *tel*

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

3.34.2.3 - (void) run

See the TypeLoader class Run method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.34.2.4 - (void) runNPhases: (int) *partitions*

See the TypeLoader class RunNPhases method.

Parameters

partitions	null

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.34.2.5 - (void) runTwoPhases

See the TypeLoader class RunTwoPhases method.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implements STSTypeLoader.

3.34.2.6 - (void) setAttributePositions: (STSInt32List ∗) *attrsPos*

Sets the list of attribute positions.

Parameters

attrsPos	[in] Attribute positions (column index >=0).

3.34.2.7 - (void) setAttributes: (STSAttributeList *) attrs

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

3.34.2.8 - (void) setFrequency: (int) freq

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.34.2.9 - (void) setGraph: (STSGraph *) graph

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

3.34.2.10 - (void) setLocale: (NSString *) localeStr

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

localeStr	[in] The locale string for the read data. See CSVReader.

3.34.2.11 - (void) setLogError: (NSString *) path

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.34.2.12 - (void) setLogOff

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

3.34.2.13 - (void) setRowReader: (STSRowReader ∗) *rr*

Sets the input data source.

Parameters

rr	[in] Input RowReader.

3.34.2.14 - (void) setTimestampFormat: (NSString ∗) *timestampFormat*

Sets a specific timestamp format.

Parameters

timestampFormat	[in] A string with the timestamp format definition.

3.34.2.15 - (void) setType: (int) *type*

Sets the type to be loaded.

Parameters

type	[in] Type identifier.

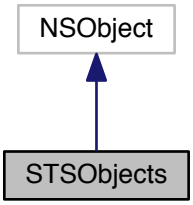
The documentation for this class was generated from the following file:

- Sparksee.h

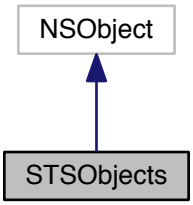
3.35 STSObjects Class Reference

Object identifier set class.

Inheritance diagram for STSObjects:



Collaboration diagram for STSObjects:



Instance Methods

- (STSObjects ∗) - clone

  Creates a new Objects instance as a copy of the given one.
- (long long) - count

  Gets the number of elements into the collection.
- (BOOL) - add:

  Adds an element into the collection.
- (BOOL) - exists:

  Gets if the given element exists into the collection.
- (long long) - any

  Gets an element from the collection.
- (BOOL) - remove:

  Removes an element from the collection.
- (void) - clear

  Clears the collection removing all its elements.
- (long long) - union:

  Performs the union operation.
- (long long) - intersection:

  Performs the intersection operation.

- (long long) - difference:

  Performs the difference operation.
- (BOOL) - equals:

  Checks if the given Objects contains the same information.
- (BOOL) - contains:

  Check if this objects contains the other one.
- (long long) - cloneWithObjects:

  Performs the copy operation.
- (STSObjects *) - sample:samples:

  Creates a new Objects instance which is a sample of the calling one.
- (STSObjectsIterator *) - iterator

  Gets an ObjectsIterator.
- (STSObjectsIterator *) - iteratorFromIndex:

  Gets an ObjectsIterator skipping index elements.
- (STSObjectsIterator *) - iteratorFromElement:

  Gets an ObjectsIterator starting from the given element.
- (void) - close

  Closes the Objects instance.
- (BOOL) - isClosed

  Check if the Objects instance is closed.
- (BOOL) - isEqual:

  Check if both Objects instances are equal.
- (NSUInteger) - hash

  Get the hash value of this Objects.

Class Methods

- (long long) + getInvalidOID

  Invalid OID constant.
- (STSObjects *) + combineUnion:objs2:

  Creates a new Objects instance which is the union of the two given.
- (STSObjects *) + combineIntersection:objs2:

  Creates a new Objects instance which is the intersection of the two given.
- (STSObjects *) + combineDifference:objs2:

  Creates a new Objects instance which is the difference of the two given.

3.35.1 Detailed Description

Object identifier set class.

It stores a collection of Sparksee object identifiers as a set. As a set, there is no order and no duplicated elements.

This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.

This class is not thread-safe.

ObjectsIterator must be used to traverse all the elements into the set.

When the Objects instance is closed, it closes all existing and non-closed ObjectsIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.35.2 Method Documentation

3.35.2.1 - (BOOL) add: (long long) *e*

Adds an element into the collection.

Parameters

e	[in] Element to be added.

Returns

TRUE if the element is added, FALSE if the element was already into the collection.

3.35.2.2 - (long long) any

Gets an element from the collection.

Returns

Any element from the collection.

Exceptions

System.ApplicationException	null
System.ApplicationException	whether the collection is empty.

3.35.2.3 - (STSObjects∗) clone

Creates a new Objects instance as a copy of the given one.

Returns

The new Objects instance.

3.35.2.4 - (long long) cloneWithObjects: (STSObjects ∗) *objs*

Performs the copy operation.

This updates the Objects calling instance and copies the given Objects instance.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

3.35.2.5 - (void) close

Closes the Objects instance.

It must be called before closing the Session to ensure the integrity of all data.

3.35.2.6 + (STSObjects*) combineDifference: (STSObjects *) *objs1* objs2:(STSObjects *) *objs2*

Creates a new Objects instance which is the difference of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

3.35.2.7 + (STSObjects*) combineIntersection: (STSObjects *) *objs1* objs2:(STSObjects *) *objs2*

Creates a new Objects instance which is the intersection of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

3.35.2.8 + (STSObjects*) combineUnion: (STSObjects *) *objs1* objs2:(STSObjects *) *objs2*

Creates a new Objects instance which is the union of the two given.

Two given Objects belong to the same Session.

Parameters

objs1	[in] Objects instance.
objs2	[in] Objects instance.

Returns

New Objects instance.

3.35.2.9 - (BOOL) contains: (STSObjects ∗) *objs*

Check if this objects contains the other one.

Parameters

objs	Objects collection.

Returns

True if it contains the given object.

3.35.2.10 - (long long) count

Gets the number of elements into the collection.

Returns

The number of elements into the collection.

3.35.2.11 - (long long) difference: (STSObjects ∗) *objs*

Performs the difference operation.

This updates the Objects calling instance removing those existing elements at the given Objects instance.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

3.35.2.12 - (BOOL) equals: (STSObjects ∗) *objs*

Checks if the given Objects contains the same information.

Parameters

objs	[in] Objects instance.

Returns

True if the objects are equal or false otherwise.

3.35.2.13 - (BOOL) exists: (long long) *e*

Gets if the given element exists into the collection.

Parameters

e	[in] Element.

Returns

TRUE if the element exists into the collection, FALSE otherwise.

3.35.2.14 - (long long) intersection: (STSObjects ∗) *objs*

Performs the intersection operation.

Updates the Objects calling instance setting those existing elements at both two collections and removing all others.

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

3.35.2.15 - (STSObjectsIterator∗) iterator

Gets an ObjectsIterator.

Returns

ObjectsIterator instance.

3.35.2.16 - (STSObjectsIterator∗) iteratorFromElement: (long long) *e*

Gets an ObjectsIterator starting from the given element.

Objects collection has no order, so this method is implementation-dependent. e[in] The first element to traverse in the resulting

Parameters

e	[in] The first element to traverse in the resulting ObjectsIterator instance.

Returns

ObjectsIterator instance.

3.35.2.17 - (STSObjectsIterator*) iteratorFromIndex: (long long) *index*

Gets an ObjectsIterator skipping index elements.

Objects collection has no order, so this method is implementation-dependent.

Parameters

index	[in] The number of elements to skip from the beginning. It must be in the range [0..Size).

Returns

ObjectsIterator instance.

3.35.2.18 - (BOOL) remove: (long long) *e*

Removes an element from the collection.

Parameters

e	[in] Element to be removed.

Returns

TRUE if the element is removed, FALSE if the element was not into the collection.

3.35.2.19 - (STSObjects*) sample: (STSObjects *) *exclude* samples:(long long) *samples*

Creates a new Objects instance which is a sample of the calling one.

Parameters

exclude	[in] If not NULL, elements into this collection will be excluded from the resulting one.
samples	[in] Number of elements into the resulting collection.

Returns

Sample collection.

3.35.2.20 - (long long) union: (STSObjects *) *objs*

Performs the union operation.

This adds all existing elements of the given Objects instance to the Objects calling instance

Parameters

objs	[in] Objects instance.

Returns

Number of elements into the collection once the operation has been executed.

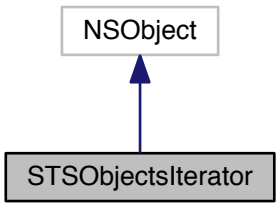
The documentation for this class was generated from the following file:

- Sparksee.h

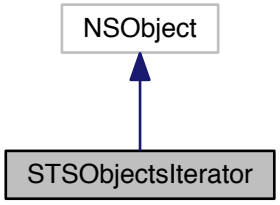
3.36 STSObjectsIterator Class Reference

ObjectsIterator class.

Inheritance diagram for STSObjectsIterator:



Collaboration diagram for STSObjectsIterator:



Instance Methods

- (BOOL) - hasNext

  Gets if there are more elements to traverse.
- (long long) - next

  Gets the next element to traverse.
- (void) - close

  Closes the ObjectsIterator instance.
- (BOOL) - isClosed

  Check if the ObjectsIterator instance is closed.

3.36.1 Detailed Description

ObjectsIterator class.

Iterator to traverse all the object identifiers from an Objects instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.36.2 Method Documentation

3.36.2.1 - (void) close

Closes the ObjectsIterator instance.

It must be called before closing the parent Objects.

3.36.2.2 - (BOOL) hasNext

Gets if there are more elements to traverse.

Returns

TRUE if there are more elements to traverse, FALSE otherwise.

3.36.2.3 - (long long) next

Gets the next element to traverse.

Returns

The next element.

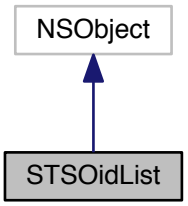
The documentation for this class was generated from the following file:

- Sparksee.h

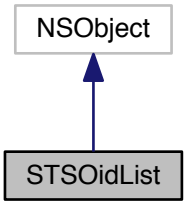
3.37 STSOidList Class Reference

Sparksee object identifier list.

Inheritance diagram for STSOidList:



Collaboration diagram for STSOidList:



Instance Methods

- (int) - count
  Number of elements in the list.
- (id) - init
  Constructor.
- (id) - initWithNumInvalidOIDs:
  Constructor.
- (void) - add:
  Adds a Sparksee object identifier at the end of the list.
- (void) - set:oid:
  Sets a Sparksee object identifier at the specified position of the list.
- (void) - clear
  Clears the list.
- (id) - initWithArray:
  Creates a new OIDList instance from the given array.
- (id) - initWithNSEnumerator:
  Creates a new OIDList instance from the given NSEnumerator.
- (STSOidListIterator *) - iterator
  Gets a new OIDListIterator.

3.37.1 Detailed Description

Sparksee object identifier list.

It stores a Sparksee object identifier list.

Use OIDListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.37.2 Method Documentation

3.37.2.1 - (void) add: (long long) *attr*

Adds a Sparksee object identifier at the end of the list.

Parameters

attr	[in] Sparksee object identifier.

3.37.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.37.2.3 - (id) init

Constructor.

This creates an empty list.

3.37.2.4 - (id) initWithNumInvalidOIDs: (int) *numInvalidOIDs*

Constructor.

This creates a list with N invalid oids.

Parameters

numInvalidOIDs	[in] The number of invalid oids added to the list.

3.37.2.5 - (void) set: (int) *pos* oid:(long long) *oid*

Sets a Sparksee object identifier at the specified position of the list.

Parameters

pos	[in] List position [0..Count()-1].
oid	[in] Sparksee object identifier.

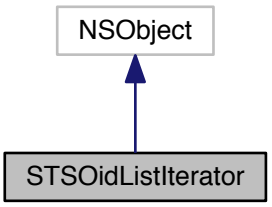
The documentation for this class was generated from the following file:

- Sparksee.h

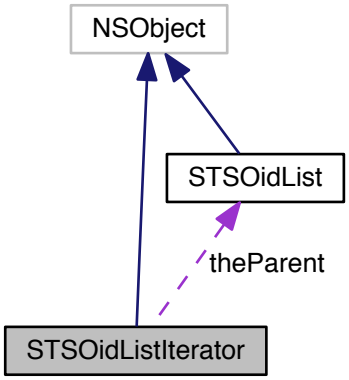
3.38 STSOidListIterator Class Reference

OIDList iterator class.

Inheritance diagram for STSOidListIterator:



Collaboration diagram for STSOidListIterator:



Instance Methods

- (long long) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.38.1 Detailed Description

OIDList iterator class.

Iterator to traverse all the Sparksee object identifier into a OIDList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.38.2 Method Documentation

3.38.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.38.2.2 - (long long) next

Moves to the next element.

Returns

The next element.

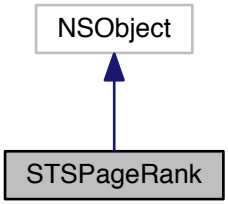
The documentation for this class was generated from the following file:

- Sparksee.h

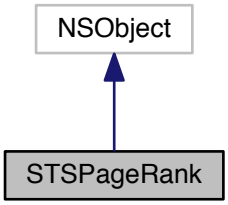
3.39 STSPageRank Class Reference

PageRank class.

Inheritance diagram for STSPageRank:



Collaboration diagram for STSPageRank:



Instance Methods

- (id) - initWithSession:

  Builds the PageRank.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - setNumIterations:

  Sets the number of iterations to run the PageRank for.
- (void) - setTolerance:

  Sets the tolerance threashold to continue computing the PageRank after each iteration.

- (void) - setDamping:

  Sets the damping value for the PageRank.
- (void) - setInitialPageRankValue:

  Sets the initial PageRank value.
- (void) - setOutputAttributeType:

  Sets the output attribute type.
- (void) - setEdgeWeightAttributeType:

  Sets the attribute to use as edge weight.
- (void) - setDefaultWeight:

  Sets the default weight for those cases when a given edge does not have a weight attribute set.
- (void) - setStartingNode:

  Sets the starting node of the page rank to compute the Personalized PageRank variant.
- (void) - run

  Runs the algorithm.
- (void) - close

  Closes the PageRank instance.
- (BOOL) - isClosed

  Check if the PageRank instance is closed.

3.39.1 Detailed Description

PageRank class.

Implements the PageRank algorithm

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.39.2 Method Documentation

3.39.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters

dir	[in] Edge direction.

3.39.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Exceptions

System.ApplicationException	null

3.39.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	null

Exceptions

System.ApplicationException	null

3.39.2.4 - (void) close

Closes the PageRank instance.

It must be called to ensure the integrity of all data.

3.39.2.5 - (id) initWithSession: (STSSession ∗) *session*

Builds the PageRank.

Parameters

session	[in] The session to use

3.39.2.6 - (void) run

Runs the algorithm.

sparksee::gdb::Error

Exceptions

System.ApplicationException	null

3.39.2.7 - (void) setDamping: (double) *damping*

Sets the damping value for the PageRank.

Parameters

damping	[in] The damping value. Default: 0.85

3.39.2.8 - (void) setDefaultWeight: (double) *weight*

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters

weight	[in] The default weight

3.39.2.9 - (void) setEdgeWeightAttributeType: (int) *attr*

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

sparksee::gdb::Error

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

Exceptions

System.ApplicationException	null

3.39.2.10 - (void) setInitialPageRankValue: (double) *startValue*

Sets the initial PageRank value.

If a starting node is set, this initial value is only set for the starting node and the rest of nodes are set to 0.0

Parameters

startValue	[in] The initial value to set. Default: 0.0

3.39.2.11 - (void) setNumIterations: (int) *numIterations*

Sets the number of iterations to run the PageRank for.

Parameters

numIterations	[in] The number of interations to set. Default: 20

3.39.2.12 - (void) setOutputAttributeType: (int) *attr*

Sets the output attribute type.

If the PageRank will run on more than one node type, then the output attribute must be of type GLOBAL_TYPE or NODES_TYPE. Otherwise, it must be a valid attribute for the used node type.

Parameters

attr	[in] The attribute to store the result. Default: InvalidAttribute

Exceptions

System.ApplicationException	null

3.39.2.13 - (void) setStartingNode: (long long) *startNode*

Sets the starting node of the page rank to compute the Personalized PageRank variant.

sparksee::gdb::Error

Parameters

startNode	null

Exceptions

System.ApplicationException	null

3.39.2.14 - (void) setTolerance: (double) *tolerance*

Sets the tolerance threashold to continue computing the PageRank after each iteration.

If all the changes to any PPR value after an iteration are below that tolerance threshold, the algorithm finishes.

Parameters

tolerance	[in] The tolerance to use normalized between 0 and 1. Default: 0.000001

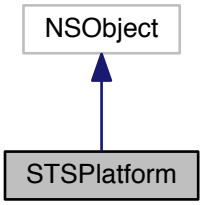
The documentation for this class was generated from the following file:

- Sparksee.h

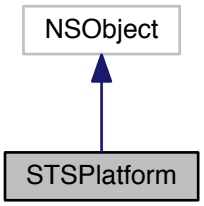
3.40 STSPlatform Class Reference

Platform class.

Inheritance diagram for STSPlatform:



Collaboration diagram for STSPlatform:



Class Methods

- (void) + getStatistics:

  Gets platform data and statistics.

3.40.1 Detailed Description

Platform class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.40.2 Method Documentation

3.40.2.1 + (void) getStatistics: (STSPlatformStatistics *) *stats*

Gets platform data and statistics.

Parameters

stats	[in|out] This updates the given PlatformStatistics.

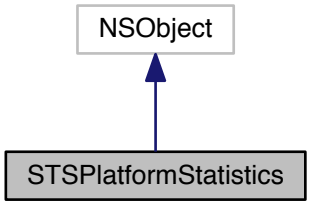
The documentation for this class was generated from the following file:

- Sparksee.h

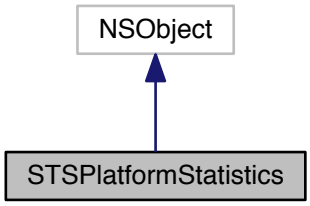
3.41 STSPlatformStatistics Class Reference

Platform data and statistics.

Inheritance diagram for STSPlatformStatistics:



Collaboration diagram for STSPlatformStatistics:



Instance Methods

- (id) - init

  Creates a new instance setting all values to 0.
- (int) - getNumCPUs

  Gets the number of CPUs.

- (long long) - getRealTime

  Gets time in microseconds (since epoch).
- (long long) - getUserTime

  Gets CPU user time.
- (long long) - getSystemTime

  Gets CPU system time.
- (long long) - getTotalMem

  Gets physical memory size in Bytes.
- (long long) - getAvailableMem

  Gets avialable (free) memory size in Bytes.

3.41.1 Detailed Description

Platform data and statistics.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.41.2 Method Documentation

3.41.2.1 - (long long) getAvailableMem

Gets avialable (free) memory size in Bytes.

Returns

Avialable (free) memory size in Bytes.

3.41.2.2 - (int) getNumCPUs

Gets the number of CPUs.

Returns

The number of CPUs.

3.41.2.3 - (long long) getRealTime

Gets time in microseconds (since epoch).

Returns

Time in microseconds (since epoch).

3.41.2.4 - (long long) getSystemTime

Gets CPU system time.

Returns

CPU system time.

3.41.2.5 - (long long) getTotalMem

Gets physical memory size in Bytes.

Returns

Physical memory size in Bytes.

3.41.2.6 - (long long) getUserTime

Gets CPU user time.

Returns

CPU user time.

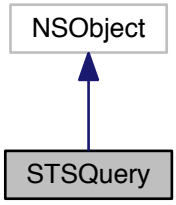
The documentation for this class was generated from the following file:

- Sparksee.h

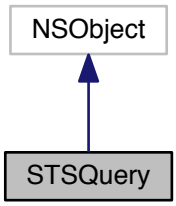
3.42 STSQuery Class Reference

Query class.

Inheritance diagram for STSQuery:



Collaboration diagram for STSQuery:



Instance Methods

- (STSResultSet ∗) - execute:reiterable:

  Executes the given statement.
- (STSQueryStream ∗) - setStream:handler:

  Sets a query stream handler.
- (void) - setDynamic:value:

  Sets the value for a dynamic paramater.
- (void) - close

  Closes the Query instance.
- (BOOL) - isClosed

  Check if the Query instance is closed.

3.42.1 Detailed Description

Query class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.42.2 Method Documentation

3.42.2.1 - (STSResultSet∗) execute: (NSString ∗) *stmt* reiterable:(BOOL) *reiterable*

Executes the given statement.

Parameters

stmt	[in] Query statement.
reiterable	[in] Whether we want the resultset to be reiterable or not

Returns

A ResultSet instance with the contents of the result of the query.

3.42.2.2 - (void) setDynamic: (NSString ∗) *name* value:(STSValue ∗) *value*

Sets the value for a dynamic paramater.

Parameters

name	[in] Parameter name
value	[in] Parameter value

3.42.2.3 - (STSQueryStream∗) setStream: (NSString ∗) *stream* handler:(STSQueryStream ∗) *handler*

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.

Parameters

stream	[in] The stream name
handler	[in] Query stream handler

Returns

The previous handler, or NULL if it does not exists

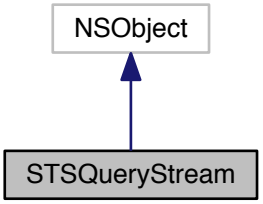
The documentation for this class was generated from the following file:

- Sparksee.h

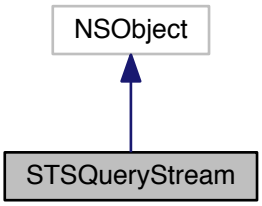
3.43 STSQueryStream Class Reference

Query stream interface.

Inheritance diagram for STSQueryStream:



Collaboration diagram for STSQueryStream:



Instance Methods

- (BOOL) - prepare:

  Prepares the stream before it is started.
- (BOOL) - start:

  Starts the stream.
- (BOOL) - fetch:

  Gets the next row and moves the iterator forward.

3.43.1 Detailed Description

Query stream interface.

A QueryStream is the interface between the application and the STREAM operator. When the operator starts inside a Query, the method is prepared with query-defined arguments. Then, if there are input operations, the STREAM operator builds the ResultSets and starts the iteration. Finally, the operator fetches rows until no more are available.

Application exceptions must be cached by the subclass that implements the interface.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.43.2 Method Documentation

3.43.2.1 - (BOOL) fetch: (STSValueList ∗) *list*

Gets the next row and moves the iterator forward.

The end of sequence is denoted by returning TRUE with an empty row. A valid row must contain as many values (even NULL) as expected by the query.

Parameters

list	[out] Storage for the new rows

Returns

TRUE if there is a row or end of sequence, FALSE on error

3.43.2.2 - (BOOL) prepare: (STSValueList *) *list*

Prepares the stream before it is started.

Parameters

list	[in] Optional list of arguments

Returns

FALSE on error

3.43.2.3 - (BOOL) start: (STSResultSetList *) *list*

Starts the stream.

Parameters

list	[in] Optional list of input ResultSets

Returns

FALSE on error

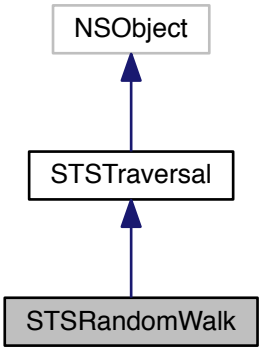
The documentation for this class was generated from the following file:

- Sparksee.h

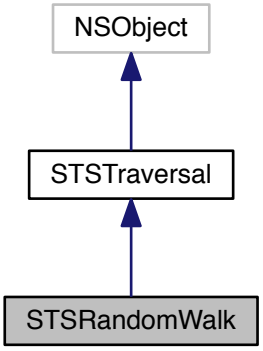
3.44 STSRandomWalk Class Reference

RandomWalk class.

Inheritance diagram for STSRandomWalk:



Collaboration diagram for STSRandomWalk:



Instance Methods

- (id) - initWithSession:node:

  Builds the RandomWalk.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.

- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (long long) - next

  Gets the next object of the traversal.
- (BOOL) - hasNext

  Gets if there are more objects to be traversed.
- (int) - getCurrentDepth

  Returns the depth of the current node.
- (void) - setEdgeWeightAttributeType:

  Sets the attribute to use as edge weight.
- (void) - setDefaultWeight:

  Sets the default weight for those cases when a given edge does not have a weight attribute set.
- (void) - reset:

  Sets the starting node of the RandomWalk.
- (void) - setReturnParameter:

  Sets the return parameter of the RandomWalk.
- (void) - setInOutParameter:

  Sets the In-Out parameter of the RandomWalk.
- (void) - setSeed:

  Sets the seed of the random walk.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - close

  Closes the Traversal instance.
- (BOOL) - isClosed

  Check if the Traversal instance is closed.

3.44.1 Detailed Description

RandomWalk class.

Implements the RandomWalk algorithm

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.44.2 Method Documentation

3.44.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

Implements STSTraversal.

3.44.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Implements STSTraversal.

3.44.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

Implements STSTraversal.

3.44.2.4 - (void) close

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

3.44.2.5 - (void) excludeEdges: (STSObjects *) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

Implements STSTraversal.

3.44.2.6 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

Implements STSTraversal.

3.44.2.7 - (int) getCurrentDepth

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

Implements STSTraversal.

3.44.2.8 - (BOOL) hasNext

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

Implements STSTraversal.

3.44.2.9 - (id) initWithSession: (STSSession *) *session* node:(long long) *node*

Builds the RandomWalk.

Parameters

session	[in] The session to use
node	[in] The starting node of the traversal

3.44.2.10 - (long long) next

Gets the next object of the traversal.

Returns

A node or edge identifier.

Implements STSTraversal.

3.44.2.11 - (void) reset: (long long) *startNode*

Sets the starting node of the RandomWalk.

This method resets the RandomWalk.

sparksee::gdb::Error

Parameters

startNode	null

3.44.2.12 - (void) setDefaultWeight: (double) *weight*

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

Parameters

weight	[in] The default weight

3.44.2.13 - (void) setEdgeWeightAttributeType: (int) *attr*

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

Parameters

attr	[in] The attribute type to use as a weight. Default: InvalidAttribute

3.44.2.14 - (void) setInOutParameter: (double) *val*

Sets the In-Out parameter of the RandomWalk.

Parameters

val	The In-Out parameter to set. Default: 1.0

3.44.2.15 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

3.44.2.16 - (void) setReturnParameter: (double) *val*

Sets the return parameter of the RandomWalk.

Parameters

val	The return parameter to set. Default: 1.0

3.44.2.17 - (void) setSeed: (int) *seed*

Sets the seed of the random walk.

Parameters

seed	The seed to generate the random numbers that drive the random walk

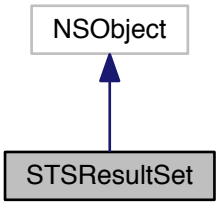
The documentation for this class was generated from the following file:

- Sparksee.h

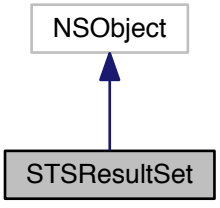
3.45 STSResultSet Class Reference

ResultSet class.

Inheritance diagram for STSResultSet:



Collaboration diagram for STSResultSet:



Instance Methods

- (int) - getNumColumns

  Gets the number of columns.
- (NSString ∗) - getColumnName:

  Gets the name for the given column.
- (int) - getColumnIndex:

  Gets the column index for the given column name.
- (enum STSDataType) - getColumnDataType:

  Gets the datatype for the given column.
- (BOOL) - next

  Fetches the next row.
- (void) - getColumnWithValue:value:

  Gets the value for the given column.
- (STSValue ∗) - getColumn:

  Gets the value for the given column.
- (void) - rewind

  Positions the cursor before the first row.
- (NSString ∗) - getJSON:

  Returns rows in JSON format.
- (void) - close

  Closes the ResultSet instance.
- (BOOL) - isClosed

  Check if the ResultSet instance is closed.

3.45.1 Detailed Description

ResultSet class.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.45.2 Method Documentation

3.45.2.1 - (STSValue∗) getColumn: (int) *index*

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

Parameters

index	[in] Column index.

Returns

The Value of the given column.

3.45.2.2 - (enum STSDataType) getColumnDataType: (int) *index*

Gets the datatype for the given column.

Parameters

index	[in] Column index.

Returns

DataType for the given column.

3.45.2.3 - (int) getColumnIndex: (NSString ∗) *name*

Gets the column index for the given column name.

Parameters

name	[in] Column name.

Returns

Column index.

3.45.2.4 - (NSString∗) getColumnName: (int) *index*

Gets the name for the given column.

Parameters

index	[in] Column index.

Returns

Column name.

3.45.2.5 - (void) getColumnWithValue: (int) *index* value:(STSValue ∗) *value*

Gets the value for the given column.

QueryExceptionIf a database access error occurs.

Parameters

index	[in] Column index.
value	[in|out] Value.

3.45.2.6 - (NSString*) getJSON: (int) *rows*

Returns rows in JSON format.

Rows are returned from the current position.

Parameters

rows	[in] Maximum number of rows

Returns

JSON representation of the next <rows> rows in the resultset

3.45.2.7 - (int) getNumColumns

Gets the number of columns.

Columns are in the range [0...COLUMNS).

Returns

The number of columns.

3.45.2.8 - (BOOL) next

Fetches the next row.

A ResultSet cursor is initially positioned before the first row; the first call to the method "Next" makes the first row the current row; the second call makes the second row the current row, and so on.

QueryExceptionIf a database access error occurs.

Returns

TRUE if the next row has been successfully fetched, FALSE otherwise.

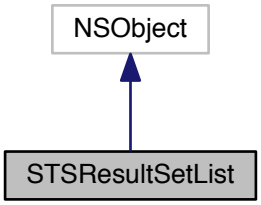
The documentation for this class was generated from the following file:

- Sparksee.h

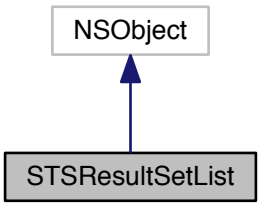
3.46 STSResultSetList Class Reference

ResultSet list.

Inheritance diagram for STSResultSetList:



Collaboration diagram for STSResultSetList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - clear

  Clears the list.
- (STSResultSet ∗) - get:

  Returns the ResultSet at the specified position in the list.
- (STSResultSetListIterator ∗) - iterator

  Gets a new ResultSetListIterator.

3.46.1 Detailed Description

ResultSet list.

It stores a ResultSet list.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.46.2 Method Documentation

3.46.2.1 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.46.2.2 - (STSResultSet*) get: (int) *index*

Returns the ResultSet at the specified position in the list.

Parameters

index	[in] Index of the element to return, starting at 0.

3.46.2.3 - (id) init

Constructor.

This creates an empty list.

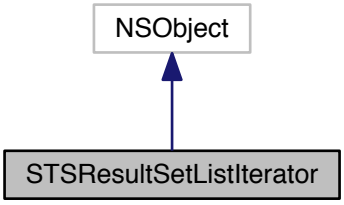
The documentation for this class was generated from the following file:

- Sparksee.h

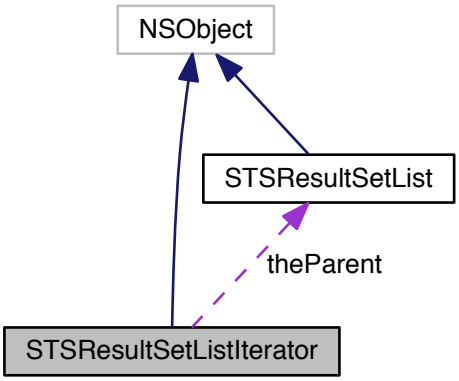
3.47 STSResultSetListIterator Class Reference

ResultSetList iterator class.

Inheritance diagram for STSResultSetListIterator:



Collaboration diagram for STSResultSetListIterator:



Instance Methods

- (STSResultSet ∗) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.47.1 Detailed Description

ResultSetList iterator class.

Iterator to traverse all the values into a ResultSetList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.47.2 Method Documentation

3.47.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.47.2.2 - (STSResultSet*) next

Moves to the next element.

Returns

The next element.

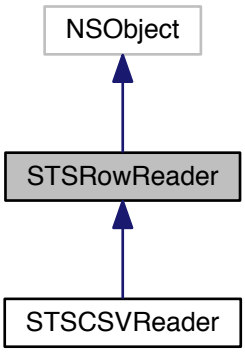
The documentation for this class was generated from the following file:

- Sparksee.h

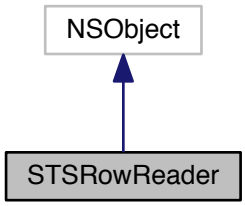
3.48 STSRowReader Class Reference

RowReader interface.

Inheritance diagram for STSRowReader:



Collaboration diagram for STSRowReader:



Instance Methods

- (BOOL) - reset

  Moves the reader to the beginning.
- (BOOL) - read:

  Reads the next row as a string array.
- (int) - getRow

  The row number for the current row.
- (void) - close

  Closes the reader.

3.48.1 Detailed Description

RowReader interface.

Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once close is called no more read operations can be executed.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.48.2 Method Documentation

3.48.2.1 - (void) close

Closes the reader.

Exceptions

System.IO.IOException	If the close fails.

Implemented in STSCSVReader.

3.48.2.2 - (int) getRow

The row number for the current row.

Returns

The current row number; 0 if there is no current row.

Exceptions

System.IO.IOException	If it fails.

Implemented in STSCSVReader.

3.48.2.3 - (BOOL) read: (STSStringList *) *row*

Reads the next row as a string array.

Parameters

row	[out] A string list with each comma-separated element as a separate entry.

Returns

Returns true if a row had been read or false otherwise.

Exceptions

System.IO.IOException	If bad things happen during the read.

Implemented in STSCSVReader.

3.48.2.4 - (BOOL) reset

Moves the reader to the beginning.

Restarts the reader.

Returns

true if the reader can be restarted, false otherwise.

Exceptions

System.IO.IOException	If bad things happen during the restart.

Implemented in STSCSVReader.

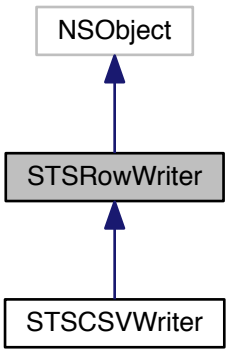
The documentation for this class was generated from the following file:

- Sparksee.h

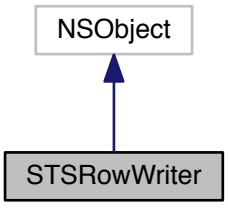
3.49 STSRowWriter Class Reference

RowWriter interface.

Inheritance diagram for STSRowWriter:



Collaboration diagram for STSRowWriter:



Instance Methods

- (void) - write:

  Writes the next row.
- (void) - close

  Closes the writer.

3.49.1 Detailed Description

RowWriter interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more write operations can be executed.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.49.2 Method Documentation

3.49.2.1 - (void) close

Closes the writer.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the close fails.

Implemented in STSCSVWriter.

3.49.2.2 - (void) write: (STSStringList ∗) *row*

Writes the next row.

Parameters

row	[in] Row of data.

Exceptions

System.ApplicationException	null
System.IO.IOException	If bad things happen during the write.

Implemented in STSCSVWriter.

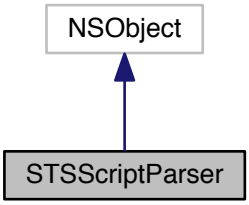
The documentation for this class was generated from the following file:

- Sparksee.h

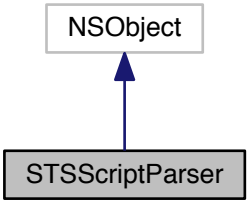
3.50 STSScriptParser Class Reference

ScriptParser.

Inheritance diagram for STSScriptParser:



Collaboration diagram for STSScriptParser:



Instance Methods

- (id) - init

  Constructor.
- (void) - setOutputLog:

  Sets the output log.
- (void) - setErrorLog:

  Sets the error log.
- (BOOL) - parse:execute:localeStr:

  Parses the given input file.

Class Methods

- (void) + generateSchemaScript:db:

  Writes an script with the schema definition for the given database.

3.50.1 Detailed Description

ScriptParser.

The ScriptParser can create schemas and load data from a set of commands in a sparksee script.

A SPARKSEE script contains an ordered list of commands. ScriptParser will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined SPARKSEE schema.

Check out the 'Scripting' chapter in the SPARKSEE User Manual for a comprehensive explanation on the grammar of the SPARKSEE commands and how they work.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.50.2 Method Documentation

3.50.2.1 + (void) generateSchemaScript: (NSString ∗) *path* db:(STSDatabase ∗) *db*

Writes an script with the schema definition for the given database.

Parameters

path	[in] Filename of the script to be writen.
db	[in] Database.

Exceptions

System.IO.IOException	If bad things happen opening or writing the file.

3.50.2.2 - (BOOL) parse: (NSString ∗) *path* execute:(BOOL) *execute* localeStr:(NSString ∗) *localeStr*

Parses the given input file.

Parameters

path	[in] Input file path.
execute	[in] If TRUE the script is executed, if FALSE it is just parsed.
localeStr	[in] The locale string for reading the input file. See CSVReader.

Returns

TRUE if ok, FALSE in case of error.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.50.2.3 - (void) setErrorLog: (NSString ∗) *path*

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters

path	[in] Path of the error log.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.50.2.4 - (void) setOutputLog: (NSString ∗) *path*

Sets the output log.

If not set, output log corresponds to standard output.

Parameters

path	[in] Path of the output log.

Exceptions

System.IO.IOException	If bad things happen opening the file.

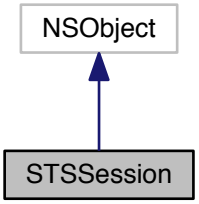
The documentation for this class was generated from the following file:

- Sparksee.h

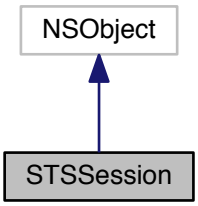
3.51 STSSession Class Reference

Session class.

Inheritance diagram for STSSession:



Collaboration diagram for STSSession:



Instance Methods

- (STSGraph *) - getGraph
  Gets the Graph instance.
- (STSObjects *) - createObjects
  Creates a new Objects instance.
- (void) - begin
  Begins a transaction.
- (void) - beginUpdate
  Begins an update transaction.
- (void) - commit
  Commits a transaction.
- (void) - rollback
  Rollbacks a transaction.
- (STSQuery *) - createQuery:
  Creates a new Query.
- (long long) - preCommit
  PreCommits a transaction.
- (long long) - getInMemoryPoolCapacity
  Gets the capacity of the in-memory pool.
- (void) - close
  Closes the Session instance.
- (BOOL) - isClosed
  Check if the Session instance is closed.

3.51.1 Detailed Description

Session class.

A Session is a stateful period of activity of a user with the Database.

All the manipulation of a Database must be enclosed into a Session. A Session can be initiated from a Database instance and allows for getting a Graph instance which represents the persistent graph (the graph database).

Also, temporary data is associated to the Session, thus when a Session is closed, all the temporary data associated to the Session is removed too. Objects or Values instances or even session attributes are an example of temporary data.

Moreover, a Session is exclusive for a thread, thus if it is shared among threads results may be fatal or unexpected.

Check out the 'Processing' and 'Transactions' sections in the SPARKSEE User Manual for details about how Sessions work and the use of transactions.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.51.2 Method Documentation

3.51.2.1 - (void) close

Closes the Session instance.

It must be called to ensure the integrity of all data.

3.51.2.2 - (STSObjects*) createObjects

Creates a new Objects instance.

Returns

The new Objects instance.

3.51.2.3 - (STSQuery*) createQuery: (enum STSQueryLanguage) *lang*

Creates a new Query.

Parameters

lang	The query language to create the query for

3.51.2.4 - (STSGraph*) getGraph

Gets the Graph instance.

Returns

The Graph instance.

3.51.2.5 - (long long) getInMemoryPoolCapacity

Gets the capacity of the in-memory pool.

Returns

Returns the capacity of the in-memory pool

3.51.2.6 - (long long) preCommit

PreCommits a transaction.

YOU SHOULD NOT USE THIS METHOD. This method exists for a specific service and it is used to force that the transaction is written in the recovery log even though it had not been committed yet (and may never be).

Returns

The transaction id. This has to be used in RedoPrecommited in case of a recovery process.

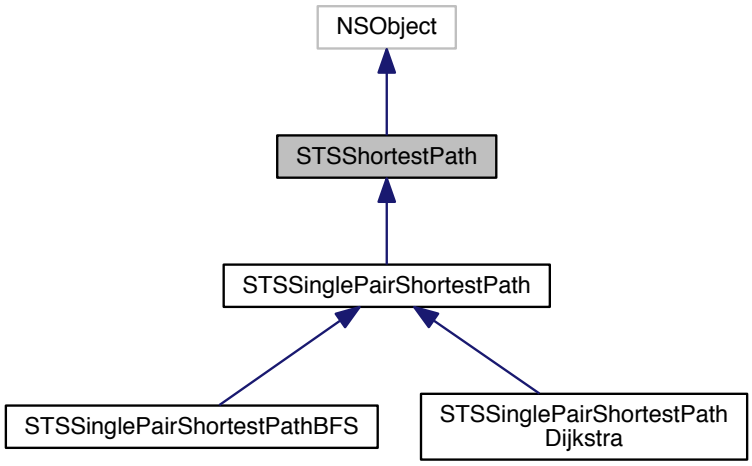
The documentation for this class was generated from the following file:

- Sparksee.h

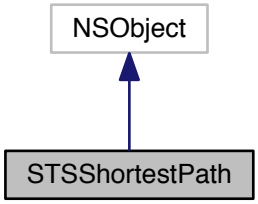
3.52 STSShortestPath Class Reference

ShortestPath class.

Inheritance diagram for STSShortestPath:



Collaboration diagram for STSShortestPath:



Instance Methods

- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - run

  Runs the algorithm.
- (void) - close

  Closes the ShortestPath instance.
- (BOOL) - isClosed

  Check if the ShortestPath instance is closed.

3.52.1 Detailed Description

ShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph.

The user must set which node and edge types can be used for the traversal.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.52.2 Method Documentation

3.52.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.52.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.52.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	null

3.52.2.4 - (void) close

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

3.52.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.52.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.52.2.7 - (void) run

Runs the algorithm.

This method can only be called once.

Implemented in STSSinglePairShortestPathDijkstra, and STSSinglePairShortestPathBFS.

3.52.2.8 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

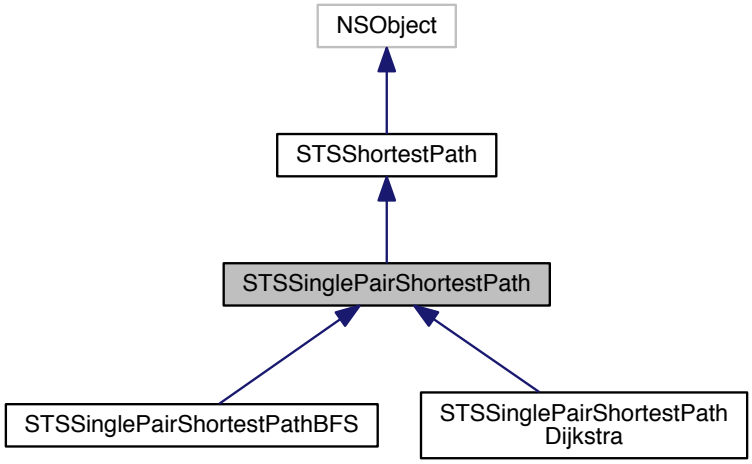
The documentation for this class was generated from the following file:

- Sparksee.h

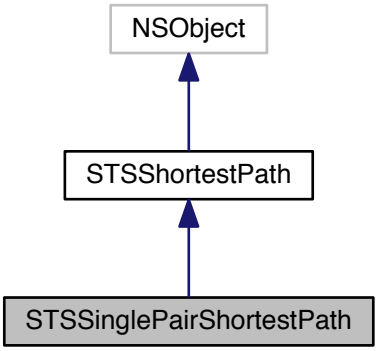
3.53 STSSinglePairShortestPath Class Reference

SinglePairShortestPath class.

Inheritance diagram for STSSinglePairShortestPath:



Collaboration diagram for STSSinglePairShortestPath:



Instance Methods

- (STSOidList *) - getPathAsNodes

  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- (STSOidList *) - getPathAsEdges

  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- (double) - getCost

  Gets the cost of the shortest path.
- (BOOL) - exists

  Returns TRUE If a path exists or FALSE otherwise.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - run

  Runs the algorithm.
- (void) - close

  Closes the ShortestPath instance.
- (BOOL) - isClosed

  Check if the ShortestPath instance is closed.

3.53.1 Detailed Description

SinglePairShortestPath class.

Classes implementing this abstract class solve the shortest path problem in a graph from a given source node and to a given destination node.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.53.2 Method Documentation

3.53.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.53.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.53.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	null

3.53.2.4 - (void) close

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

3.53.2.5 - (void) excludeEdges: (STSObjects *) edges

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.53.2.6 - (void) excludeNodes: (STSObjects *) nodes

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.53.2.7 - (double) getCost

Gets the cost of the shortest path.

The cost for unweighted algorithms is the number of hops of the shortest path. For weighted algoritms, the cost is the sum of the costs of the edges of the shortest path.

Returns

The cost of the shortest path.

Implemented in STSSinglePairShortestPathDijkstra, and STSSinglePairShortestPathBFS.

3.53.2.8 - (STSOidList*) getPathAsEdges

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

Implemented in STSSinglePairShortestPathDijkstra, and STSSinglePairShortestPathBFS.

3.53.2.9 - (STSOidList*) getPathAsNodes

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

Implemented in STSSinglePairShortestPathDijkstra, and STSSinglePairShortestPathBFS.

3.53.2.10 - (void) run

Runs the algorithm.

This method can only be called once.

Implemented in STSSinglePairShortestPathDijkstra, and STSSinglePairShortestPathBFS.

3.53.2.11 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

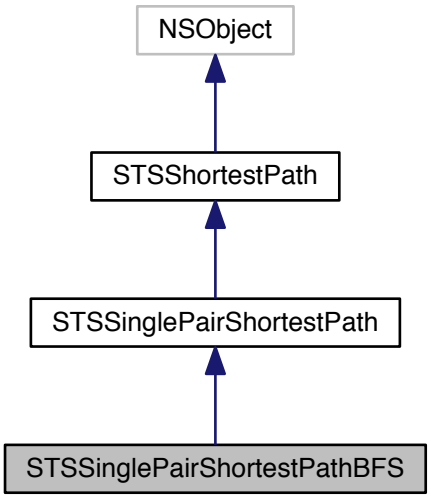
The documentation for this class was generated from the following file:

- Sparksee.h

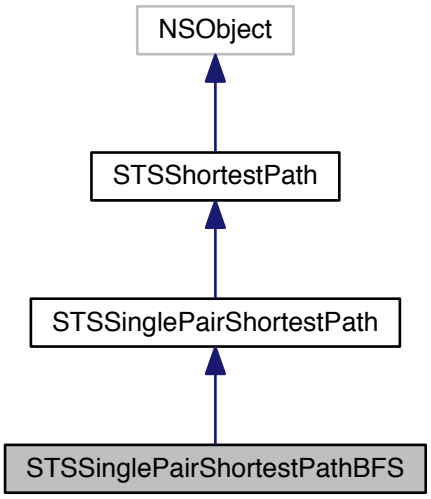
3.54 STSSinglePairShortestPathBFS Class Reference

SinglePairShortestPathBFS class.

Inheritance diagram for STSSinglePairShortestPathBFS:



Collaboration diagram for STSSinglePairShortestPathBFS:



Instance Methods

- (void) - run

  Executes the algorithm.
- (STSOidList ∗) - getPathAsNodes

  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- (STSOidList ∗) - getPathAsEdges

  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- (double) - getCost

  Gets the cost of the shortest path.
- (id) - initWithSession:src:dst:

  Creates a new instance.
- (void) - checkOnlyExistence

  Set that only the path existence must be calculated and not the path itself.
- (BOOL) - exists

  Returns TRUE If a path exists or FALSE otherwise.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

Set which nodes can't be used.

- (void) - excludeEdges:

  Set which edges can't be used.

- (void) - close

  Closes the ShortestPath instance.

- (BOOL) - isClosed

  Check if the ShortestPath instance is closed.

3.54.1 Detailed Description

SinglePairShortestPathBFS class.

It solves the single-pair shortest path problem using a BFS-based implementation.

It is a unweighted algorithm, that is it assumes all edges have the same cost.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.54.2 Method Documentation

3.54.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.54.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.54.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	null

3.54.2.4 - (void) checkOnlyExistence

Set that only the path existence must be calculated and not the path itself.

That method should improve the performance of the algorithm, but a call to GetPathAsNodes or GetPathAsEdges will generate an exception even if the path exists.

3.54.2.5 - (void) close

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

3.54.2.6 - (void) excludeEdges: (STSObjects *) edges

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.54.2.7 - (void) excludeNodes: (STSObjects *) nodes

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.54.2.8 - (double) getCost

Gets the cost of the shortest path.

The cost is the number of hops of the shortest path.

Returns

The cost of the shortest path.

Implements STSSinglePairShortestPath.

3.54.2.9 - (STSOidList*) getPathAsEdges

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

Implements STSSinglePairShortestPath.

3.54.2.10 - (STSOidList*) getPathAsNodes

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

Implements STSSinglePairShortestPath.

3.54.2.11 - (id) initWithSession: (STSSession *) *session* src:(long long) *src* dst:(long long) *dst*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
src	[in] Source node.
dst	[dst] Destination node.

3.54.2.12 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

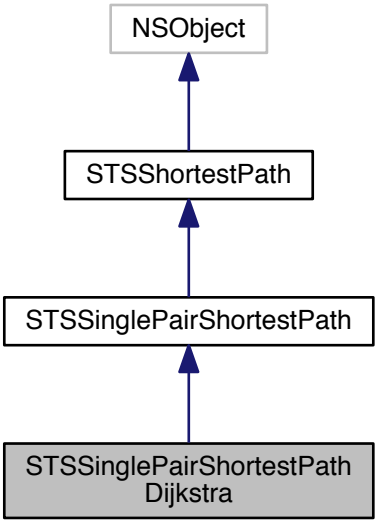
The documentation for this class was generated from the following file:

- Sparksee.h

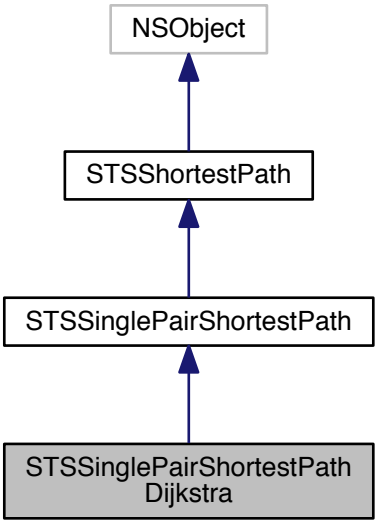
3.55 STSSinglePairShortestPathDijkstra Class Reference

SinglePairShortestPathDijkstra class.

Inheritance diagram for STSSinglePairShortestPathDijkstra:



Collaboration diagram for STSSinglePairShortestPathDijkstra:



Instance Methods

- (void) - run

Executes the algorithm.

- (STSOidList *) - getPathAsNodes

  Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- (STSOidList *) - getPathAsEdges

  Gets the shortest path between the source node and the destination node as an ordered set of edges.
- (double) - getCost

  Gets the cost of the shortest path.
- (id) - initWithSession:src:dst:

  Creates a new instance.
- (void) - addWeightedEdgeType:dir:attr:

  Allows for traversing edges of the given type using the given attribute as the weight.
- (void) - setUnweightedEdgeCost:

  Sets the weight assigned to the unweighted edges.
- (void) - setDynamicEdgeCostCallback:

  Set a class callback to dynamically calculate the cost of the edges.
- (BOOL) - exists

  Returns TRUE If a path exists or FALSE otherwise.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - close

  Closes the ShortestPath instance.
- (BOOL) - isClosed

  Check if the ShortestPath instance is closed.

3.55.1 Detailed Description

SinglePairShortestPathDijkstra class.

It solves the single-pair shortest path problem using a Dijkstra-based implementation.

It is a weighted algorithm, so it takes into account the cost of the edges to compute a minimum-cost shortest path. That is, the user may set for each edge type which attribute should be used to retrieve the cost of the edge. If no attribute is given for an edge type, this will assume the edge has a fixed cost (the default is 1). Only numerical attribute can be set as weight attributes (that is Long, Integer or Double attributes are allowed).

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.55.2 Method Documentation

3.55.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.55.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.55.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	null

3.55.2.4 - (void) addWeightedEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir* attr:(int) *attr*

Allows for traversing edges of the given type using the given attribute as the weight.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.
attr	[in] Attribute to be used as the weight. It must be a global attribute or an attribute of the given edge type.

3.55.2.5 - (void) close

Closes the ShortestPath instance.

It must be called to ensure the integrity of all data.

3.55.2.6 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.55.2.7 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.55.2.8 - (double) getCost

Gets the cost of the shortest path.

The cost is the sum of the weights of the edges in the shortest path.

Returns

The cost of the shortest path.

Implements STSSinglePairShortestPath.

3.55.2.9 - (STSOidList*) getPathAsEdges

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns

Ordered set of edge identifiers.

Implements STSSinglePairShortestPath.

3.55.2.10 - (STSOidList*) getPathAsNodes

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns

Ordered set of node identifiers.

Implements STSSinglePairShortestPath.

3.55.2.11 - (id) initWithSession: (STSSession *) *session* src:(long long) *src* dst:(long long) *dst*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
src	[in] Source node.
dst	[dst] Destination node.

3.55.2.12 - (void) setDynamicEdgeCostCallback: (STSSinglePairShortestPathDijkstraDynamicCost *) *dynCostCalculator*

Set a class callback to dynamically calculate the cost of the edges.

The callback can be set to NULL (the default) to use the normal attribute based cost weights. The given class must be kept alive by the user for as long as the algorithm is running.

Parameters

dynCostCalculator	[in] Class callback to calculate the edge costs

3.55.2.13 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

3.55.2.14 - (void) setUnweightedEdgeCost: (double) *weight*

Sets the weight assigned to the unweighted edges.

All the edges from the types added without an explicit weight attribute will get this weight. The default weight for this edges is 1.

Parameters

weight	[in] The weight value for unweighted edges.

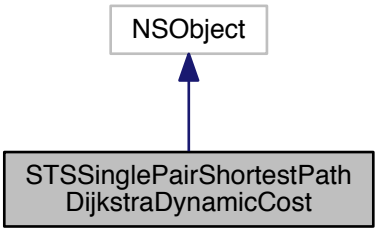
The documentation for this class was generated from the following file:

- Sparksee.h

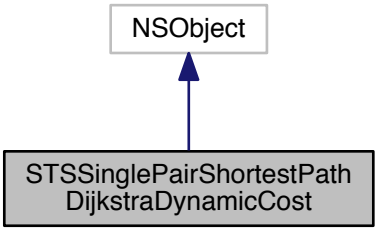
3.56 STSSinglePairShortestPathDijkstraDynamicCost Class Reference

Defines how to calculate an edge weight.

Inheritance diagram for STSSinglePairShortestPathDijkstraDynamicCost:



Collaboration diagram for STSSinglePairShortestPathDijkstraDynamicCost:



Instance Methods

- (double) - calculateEdgeCost:sourceCost:sourceLevel:targetNode:edge:edgeWeightAttr:

  Dynamic calculation of an edge weight.

3.56.1 Detailed Description

Defines how to calculate an edge weight.

This is an interface which must be implemented by the user. While the algorithm is running, one or more calls for each edge that can be in the shortest path will be issued to get the weight of the edge that could change based on the predecessor node and the current minimum path from the source.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.56.2 Method Documentation

3.56.2.1 - (double) calculateEdgeCost: (long long) *sourceNode* sourceCost:(double) *sourceCost* sourceLevel:(int) *sourceLevel* targetNode:(long long) *targetNode* edge:(long long) *edge* edgeWeightAttr:(int) *edgeWeightAttr*

Dynamic calculation of an edge weight.

This method will be called several times during the algorithm execution. It can be called for the same edge multiple times because the path to the source node (and as a consequence the source weight and levels) might change while the algorithm is running. This method will have the normal default implementation that only uses the specified edge attribute to get the weight or the unweighted edge cost if the edge doesn't have the specified cost attribute. But it can be overriden in a derived user class to implement a dinamic edge weight, that may need the sourceCost and sourceLevels to calculate the weight.

Parameters

sourceNode	[in] The current node
sourceCost	[in] Current total cost up to the source node
sourceLevel	[in] Current path size up to the source node
targetNode	[in] The next node
edge	[in] The edge to the next node that this methos must weight
edgeWeightAttr	[in] The attribute that stores the edge weight or InvalidAttribute

Returns

Returns the current weight of the edge (>=0) or <0 if the edge can not be used.

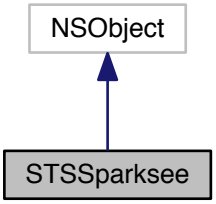
The documentation for this class was generated from the following file:

- Sparksee.h

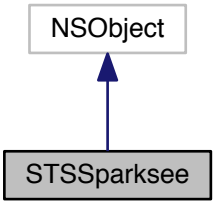
3.57 STSSparksee Class Reference

Sparksee class.

Inheritance diagram for STSSparksee:



Collaboration diagram for STSSparksee:



Instance Methods

- (id) - initWithConfig:

  Creates a new instance.
- (STSDatabase *) - create:alias:

  Creates a new Database instance.
- (STSDatabase *) - createWithConfig:alias:config:

  Creates a new Database instance.
- (STSDatabase *) - open:readOnly:

  Opens an existing Database instance.
- (STSDatabase *) - openWithConfig:readOnly:config:

  Opens an existing Database instance.
- (STSDatabase *) - restore:backupFile:

  Restores a Database from a backup file.
- (STSDatabase *) - restoreWithConfig:backupFile:config:

  Restores a Database from a backup file.
- (STSDatabase *) - restoreEncryptedBackup:backupFile:keyInHex:ivInHex:

  Restores a Database from an encrypted backup file.
- (STSDatabase *) - restoreEncryptedBackupWithConfig:backupFile:config:keyInHex:ivInHex:

  Restores a Database from an encrypted backup file.
- (BOOL) - verifyChecksums:

  Verifies the Checksum integrity of the given image file and it's recovery log file.
- (BOOL) - addChecksums:

  Converts a database WITHOUT checksums into a database WITH checksums.
- (BOOL) - removeChecksums:

  Converts a database WITH checksums into a database WITHOUT checksums.
- (BOOL) - encrypt:

  Converts a database WITHOUT encryption into an encrypted database.
- (BOOL) - decrypt:

  Converts a database WITH encryption into a database WITHOUT encryption.
- (void) - resizeInMemoryPool:

  Resizes the in memory pool allocator.
- (long long) - getInMemoryPoolCapacity

  Gets the capacity of the in-memory pool (in Megabytes)
- (void) - close

  Closes the Sparksee instance.
- (BOOL) - isClosed

  Check if the Sparksee instance is closed.

Class Methods

- (NSString ∗) + getVersion

  Sparksee version.

3.57.1 Detailed Description

Sparksee class.

All Sparksee programs must have one single Sparksee instance to manage one or more Database instances.

This class allows for the creation of new Databases or open an existing one.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.57.2 Method Documentation

3.57.2.1 - (BOOL) addChecksums: (NSString ∗) *path*

Converts a database WITHOUT checksums into a database WITH checksums.

Any error found is logged. The database (and it's recovery log if present) does NOT have checksums. sparksee↩
::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the checksum could be added or false otherwise.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

3.57.2.2 - (STSDatabase∗) create: (NSString ∗) *path* alias:(NSString ∗) *alias*

Creates a new Database instance.

Parameters

path	[in] Database storage file.
alias	[in] Database alias name.

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.57.2.3 - (STSDatabase∗) createWithConfig: (NSString ∗) *path* alias:(NSString ∗) *alias* config:(STSSparkseeConfig ∗) *config*

Creates a new Database instance.

Parameters

path	[in] Database storage file.
alias	[in] Database alias name.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created.

3.57.2.4 - (BOOL) decrypt: (NSString ∗) *path*

Converts a database WITH encryption into a database WITHOUT encryption.

Any error found is logged. The database (and it's recovery log if present) is encrypted. The current encryption had been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the database could be unencrypted. The errors are returned with an exception.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

3.57.2.5 - (BOOL) encrypt: (NSString ∗) *path*

Converts a database WITHOUT encryption into an encrypted database.

Any error found is logged. The database (and it's recovery log if present) does is NOT encrypted. The encryption had already been configured using SparkseeConfig. sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the database could be encrypted. The errors are returned with an exception.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

3.57.2.6 - (long long) getInMemoryPoolCapacity

Gets the capacity of the in-memory pool (in Megabytes)

Returns

Returns the capacity of the in memory pool

3.57.2.7 - (id) initWithConfig: (STSSparkseeConfig ∗) *config*

Creates a new instance.

Parameters

config	[in/out] Sparksee configuration (may be updated).

3.57.2.8 - (STSDatabase∗) open: (NSString ∗) *path* readOnly:(BOOL) *readOnly*

Opens an existing Database instance.

Parameters

path	[in] Database storage file.
readOnly	[in] If TRUE, open Database in read-only mode.

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file does not exist.

3.57.2.9 - (STSDatabase∗) openWithConfig: (NSString ∗) *path* readOnly:(BOOL) *readOnly* config:(STSSparkseeConfig ∗) *config*

Opens an existing Database instance.

The provided configuration will be used for all the database settings but not for the license, which must already be properly setup in this class.

Parameters

path	[in] Database storage file.
readOnly	[in] If TRUE, open Database in read-only mode.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file does not exist.

3.57.2.10 - (BOOL) removeChecksums: (NSString ∗) *path*

Converts a database WITH checksums into a database WITHOUT checksums.

Any error found is logged. The database (and it's recovery log if present) have checksums. sparksee::gdb::File↩ NotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the database image file

Returns

Returns true if the checksum could be removed. The errors are returned with an exception.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

3.57.2.11 - (void) resizeInMemoryPool: (long long) *newCapacity*

Resizes the in memory pool allocator.

Parameters

newCapacity	The new capacity of the in memory pool allocator

Returns

Returns true if was successful

Exceptions

System.SystemException	null

3.57.2.12 - (STSDatabase∗) restore: (NSString ∗) *path* backupFile:(NSString ∗) *backupFile*

Restores a Database from a backup file.

See the Graph class Backup method.

Parameters

path	[in] Database storage file.
backupFile	[in] The Backup file to be restored.

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created, or the exported data file does not exists.

3.57.2.13 - (STSDatabase∗) restoreEncryptedBackup: (NSString ∗) *path* backupFile:(NSString ∗) *backupFile* keyInHex:(NSString ∗) *keyInHex* ivInHex:(NSString ∗) *ivInHex*

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters

path	[in] Database storage file.
backupFile	[in] The Backup file to be restored.
keyInHex	[In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
ivInHex	[In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created, or the exported data file does not exists.

3.57.2.14 - (STSDatabase∗) restoreEncryptedBackupWithConfig: (NSString ∗) *path* backupFile:(NSString ∗) *backupFile* config:(STSSparkseeConfig ∗) *config* keyInHex:(NSString ∗) *keyInHex* ivInHex:(NSString ∗) *ivInHex*

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

Parameters

path	[in] Database storage file.
backupFile	[in] The Backup file to be restored.
config	[in] Use a specific configuration instead of the one in this Sparksee
keyInHex	[In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes).
ivInHex	[In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes).

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created, or the exported data file does not exists.

3.57.2.15 - (STSDatabase∗) restoreWithConfig: (NSString ∗) *path* backupFile:(NSString ∗) *backupFile* config:(STSSparkseeConfig ∗) *config*

Restores a Database from a backup file.

See the Graph class Backup method.

Parameters

path	[in] Database storage file.
backupFile	[in] The Backup file to be restored.
config	[in] Use a specific configuration instead of the one in this Sparksee

Returns

A Database instance.

Exceptions

System.ApplicationException	null
System.IO.IOException	If the given file cannot be created, or the exported data file does not exists.

3.57.2.16 - (BOOL) verifyChecksums: (NSString ∗) *path*

Verifies the Checksum integrity of the given image file and it's recovery log file.

The main database file checksums are always checked.

When the recovery log file exists, the recovery log file is also checked for checksum and format errors.

When the checksums of the main file or the recovery file are incorrect, the funcion returns false, in any other error it throws an exception.

Any error found is logged as a WARNING.

sparksee::gdb::FileNotFoundExceptionsparksee::gdb::Error

Parameters

path	The path to the image file

Returns

Returns true if the checksum verification succeeded. Returns false if an invalid checksum was detected.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

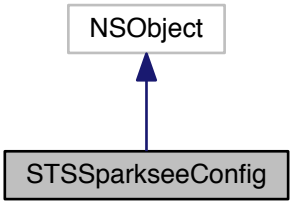
The documentation for this class was generated from the following file:

- Sparksee.h

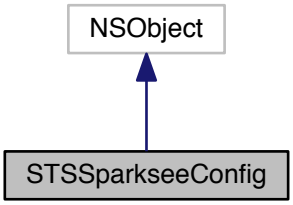
3.58 STSSparkseeConfig Class Reference

Sparksee configuration class.

Inheritance diagram for STSSparkseeConfig:



Collaboration diagram for STSSparkseeConfig:



Instance Methods

- (id) - init

  Creates a new instance.
- (id) - initWithPath:

  Creates a new instance with a specific config file.
- (id) - initWithPath:clientId:licenseId:

  Creates a new instance with a specific config file and IDs.
- (int) - getExtentSize

  Gets the size of a extent.
- (void) - setExtentSize:

  Sets the size of the extents in KB.
- (int) - getExtentPages

  Gets the number of pages per extent.
- (void) - setExtentPages:

  Sets the number of pages per extent.

- (int) - getPoolFrameSize

  Gets the size of a pool frame in number of extents.
- (void) - setPoolFrameSize:

  Sets the size of a pool frame in number of extents.
- (int) - getPoolPersistentMinSize

  Gets the minimum size for the persistent pool in number of frames.
- (void) - setPoolPersistentMinSize:

  Sets the minimum size for the persistent pool in number of frames.
- (int) - getPoolPersistentMaxSize

  Gets the maximum size for the persistent pool in number of frames.
- (void) - setPoolPersistentMaxSize:

  Sets the maximum size for the persistent pool in number of frames.
- (int) - getPoolTemporaryMinSize

  Gets the minimum size for the temporary pool in number of frames.
- (void) - setPoolTemporaryMinSize:

  Sets the minimum size for the temporary pool in number of frames.
- (int) - getPoolTemporaryMaxSize

  Gets the maximum size for the temporary pool in number of frames.
- (void) - setPoolTemporaryMaxSize:

  Sets the maximum size for the temporary pool in number of frames.
- (int) - getPoolPartitions

  Gets the number of partitions in each PartitionedPool.
- (void) - setPoolPartitions:

  Sets the number of pools in each PartitionedPool.
- (int) - getCacheMaxSize

  Gets the maximum size for the cache (all pools) in MB.
- (void) - setCacheMaxSize:

  Sets the maximum size for the cache (all pools) in MB.
- (NSString ∗) - getLicense

  Gets the license key.
- (int) - setLicense:

  Sets the license key.
- (NSString ∗) - getLogFile

  Gets the log file.
- (void) - setLogFile:

  Sets the log file.
- (enum STSLogLevel) - getLogLevel

  Gets the log level.
- (void) - setLogLevel:

  Sets the log level.
- (BOOL) - getCacheStatisticsEnabled

  Gets whether cache statistics are enabled or disabled.
- (void) - setCacheStatisticsEnabled:

  Enables or disables cache statistics.
- (NSString ∗) - getCacheStatisticsFile

  Gets the cache statistics log file.
- (void) - setCacheStatisticsFile:

  Sets the cache statistics log file.
- (long long) - getCacheStatisticsSnapshotTime

  Gets the cache statistics snapshot time in microseconds.
- (void) - setCacheStatisticsSnapshotTime:

Sets the cache statistics snapshot time.

- (BOOL) - getRollbackEnabled

  Gets whether the rollback is enabled or disabled.
- (void) - setRollbackEnabled:

  Enables or disables the rollback.
- (BOOL) - getRecoveryEnabled

  Gets whether the recovery is enabled or disabled.
- (void) - setRecoveryEnabled:

  Enables or disables the recovery.
- (NSString ∗) - getRecoveryLogFile

  Gets the recovery log file.
- (void) - setRecoveryLogFile:

  Sets the recovery log file.
- (int) - getRecoveryCacheMaxSize

  Gets the maximum size for the recovery log cache in extents.
- (void) - setRecoveryCacheMaxSize:

  Sets the maximum size for the recovery log cache in extents.
- (long long) - getRecoveryCheckpointTime

  Gets the delay time (in microseconds) between automatic checkpoints.
- (void) - setRecoveryCheckpointTime:

  Sets the delay time (in microseconds) between automatic checkpoints.
- (BOOL) - getChecksumEnabled

  Gets whether the storage checksum usage is enabled or disabled.
- (void) - setChecksumEnabled:

  Enables or disables the storage checksum usage.
- (BOOL) - getHighAvailabilityEnabled

  Gets whether high availability mode is enabled or disabled.
- (void) - setHighAvailabilityEnabled:

  Enables or disables high availability mode.
- (NSString ∗) - getHighAvailabilityIP

  Gets the IP address and port of the instance.
- (void) - setHighAvailabilityIP:

  Sets the IP address and port of the instance.
- (NSString ∗) - getHighAvailabilityCoordinators

  Gets the coordinators address and port list.
- (void) - setHighAvailabilityCoordinators:

  Sets the coordinators address and port list.
- (long long) - getHighAvailabilitySynchronization

  Gets the synchronization polling time.
- (void) - setHighAvailabilitySynchronization:

  Sets the synchronization polling time.
- (long long) - getHighAvailabilityMasterHistory

  Gets the master's history log.
- (void) - setHighAvailabilityMasterHistory:

  Sets the master's history log.
- (BOOL) - getCallStackDump

  Gets whether the signals will be captured to dump the call stack or not.
- (void) - setCallStackDump:

  Sets whether the signals will be captured to dump the call stack or not.
- (NSString ∗) - getClientId

  Gets the client identifier.

- (void) - setClientId:

  Set the client identifier.
- (NSString ∗) - getLicenseId

  Gets the license identifier.
- (void) - setLicenseId:

  Set the license identifier.
- (int) - getLicensePreDownloadDays

  Get the number of days before expiration when a new license will be downloaded.
- (void) - setLicensePreDownloadDays:

  Start trying to automatically download a new license at the specified number of days before expiration.
- (int) - downloadLicense

  Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).
- (NSString ∗) - getDownloadStatus

  Gets a message with the license download result.
- (BOOL) - downloadExpected

  Check if a new license will be automatically downloaded with the current settings.
- (NSString ∗) - getLicenseRequest

  Get information useful to manually request a license.
- (NSString ∗) - getTmpFolder

  Gets the temporary folder used for temporary staging.
- (void) - setTmpFolder:

  Sets the temporary folder used for temporary staging.
- (BOOL) - getTmpEnabled

  Gets whether using temporary storage for computations is enabled.
- (void) - setTmpEnabled:

  Sets whether to use temporary storage for computations or not.
- (long long) - getInMemAllocSize

  Gets the in-memory allocator size.
- (void) - setInMemAllocSize:

  Sets the in-memory allocator size.
- (BOOL) - save

  Save the current configuration in the specified config file.
- (BOOL) - saveAll

  Save all the current configuration settings in the specified config file.
- (BOOL) - getEncryptionEnabled

  Gets whether the storage encryption is enabled or disabled.
- (BOOL) - setAESEncryptionEnabled:ivInHex:

  Enables storage encryption using AES with the given key and iv.
- (NSString ∗) - getAESKey

  Get the AES encryption key in a hexadecimal encoded string.
- (NSString ∗) - getAESIV

  Get the AES initialization vector in a hexadecimal encoded string.
- (void) - setEncryptionDisabled

  Disables storage encryption.
- (void) - setSparkseeConfigFile:

  Sets the config file path.
- (NSString ∗) - getSparkseeConfigFile

  Gets the config file path.

3.58.1 Detailed Description

Sparksee configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overrided with values from a properties file (see SparkseeProperties class). Also, this settings can be overrided calling a specific setter.

For each field, it is shown its default value and the property to override this value:

Extent size: 4KB ('sparksee.storage.extentsize' at SparkseeProperties).

Pages per extent: 1 page ('sparksee.storage.extentpages' at SparkseeProperties).

Checksums enabled: true ('sparksee.storage.checksum' at SparkseeProperties).

Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at SparkseeProperties).

Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at SparkseeProperties).

Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at SparkseeProperties).

Minimum size for the temporary pool: 16 frames ('sparksee.io.pool.temporal.minsize' at SparkseeProperties).

Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at SparkseeProperties).

Number of pools in the pool cluster: 0 pools ('sparksee.io.pool.clustersize' at SparkseeProperties). 0 or 1 means the clustering is disabled.

Maximum size for the cache (all pools): 0 MB ('sparksee.io.cache.maxsize' at SparkseeProperties).

License code: "" ('sparksee.license' at SparkseeProperties). No license code means evaluation license.

Log level: Info ('sparksee.log.level' at SparkseeProperties).

Log file: "sparksee.log" ('sparksee.log.file' at SparkseeProperties).

Cache statistics: false (disabled) ('sparksee.cache.statistics' at SparkseeProperties).

Cache statistics log file: "statistics.log" ('sparksee.cache.statisticsFile' at SparkseeProperties).

Cache statistics snapshot time: 1000 msecs [TimeUnit] ('sparksee.cache.statisticsSnapshotTime' at SparkseeProperties).

Recovery enabled: false ('sparksee.io.recovery' at SparkseeProperties).

Recovery log file: "" ('sparksee.io.recovery.logfile' at SparkseeProperties).

Recovery cache max size: 1MB ('sparksee.io.recovery.cachesize' at SparkseeProperties).

Recovery checkpoint time: 60 seconds [TimeUnit] ('sparksee.io.recovery.checkpointTime' at SparkseeProperties).

High-availability: false (disabled) ('sparksee.ha' at SparkseeProperties).

High-availability coordinators: "" ('sparksee.ha.coordinators' at SparkseeProperties).

High-availability IP: "" ('sparksee.ha.ip' at SparkseeProperties).

High-availability sync polling: 0 (disabled) [TimeUnit] ('sparksee.ha.sync' at SparkseeProperties).

High-availability master history: 1D (1 day) [TimeUnit] ('sparksee.ha.master.history' at SparkseeProperties).

Use of TimeUnit:

Those variables using TimeUnit allow for:

<X>[D|H|M|S|s|m|u]

where <X> is a number followed by an optional character which represents the unit: D for days, H for hours, M for minutes, S or s for seconds, m for milliseconds and u for microseconds. If no unit character is given, seconds are assumed.

Capture abort signals to dump the call stack ('sparksee.callstackdump' at SparkseeProperties) is enabled by default on most platforms.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.58.2 Method Documentation

3.58.2.1 - (int) downloadLicense

Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).

Returns

Returns -1 if a valid license key couldn't be downloaded, 0 if the same license was downloaded or 1 if a different license is downloaded.

3.58.2.2 - (NSString∗) getAESIV

Get the AES initialization vector in a hexadecimal encoded string.

Returns

The AES initialization vector as an hexadecimal string.

3.58.2.3 - (NSString∗) getAESKey

Get the AES encryption key in a hexadecimal encoded string.

Returns

The AES encryption Key as an hexadecimal string.

3.58.2.4 - (int) getCacheMaxSize

Gets the maximum size for the cache (all pools) in MB.

Returns

The maximum size for the cache (all pools) in MB.

3.58.2.5 - (BOOL) getCacheStatisticsEnabled

Gets whether cache statistics are enabled or disabled.

Returns

TRUE if cache statistics are enabled, FALSE otherwise.

3.58.2.6 - (NSString*) getCacheStatisticsFile

Gets the cache statistics log file.

Useless if cache statistics are disabled.

Returns

The cache statistics log file.

3.58.2.7 - (long long) getCacheStatisticsSnapshotTime

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

Returns

The cache statistics snapshot time in microseconds.

3.58.2.8 - (BOOL) getCallStackDump

Gets whether the signals will be captured to dump the call stack or not.

Returns

TRUE if the signals must be captured, FALSE otherwise.

3.58.2.9 - (BOOL) getChecksumEnabled

Gets whether the storage checksum usage is enabled or disabled.

Returns

TRUE if the checksum is enabled, FALSE otherwise.

3.58.2.10 - (NSString*) getClientId

Gets the client identifier.

Returns

The client identifier.

3.58.2.11 - (NSString*) getDownloadStatus

Gets a message with the license download result.

It can be used when the DownloadLicense call failed.

Returns

The log download result.

3.58.2.12 - (BOOL) getEncryptionEnabled

Gets whether the storage encryption is enabled or disabled.

Returns

TRUE if the encryption is enabled, FALSE otherwise.

3.58.2.13 - (int) getExtentPages

Gets the number of pages per extent.

Returns

The number of pages per extent.

3.58.2.14 - (int) getExtentSize

Gets the size of a extent.

Returns

The size of a extent in KB.

3.58.2.15 - (NSString∗) getHighAvailabilityCoordinators

Gets the coordinators address and port list.

Returns

The coordinators address and port list.

3.58.2.16 - (BOOL) getHighAvailabilityEnabled

Gets whether high availability mode is enabled or disabled.

Returns

TRUE if high availability mode is enabled, FALSE otherwise.

3.58.2.17 - (NSString∗) getHighAvailabilityIP

Gets the IP address and port of the instance.

Returns

The IP address and port of the instance.

3.58.2.18 - (long long) getHighAvailabilityMasterHistory

Gets the master's history log.

Returns

The master's history log.

3.58.2.19 - (long long) getHighAvailabilitySynchronization

Gets the synchronization polling time.

Returns

The Synchronization polling time.

3.58.2.20 - (long long) getInMemAllocSize

Gets the in-memory allocator size.

Returns

Returns the in-memory allocator size

3.58.2.21 - (NSString*) getLicense

Gets the license key.

Returns

The license key.

3.58.2.22 - (NSString*) getLicenseId

Gets the license identifier.

Returns

The license identifier.

3.58.2.23 - (int) getLicensePreDownloadDays

Get the number of days before expiration when a new license will be downloaded.

Returns

Returns the number of days or -1 if it will never be automatically downloaded.

3.58.2.24 - (NSString*) getLogFile

Gets the log file.

Returns

The log file.

3.58.2.25 - (enum STSLogLevel) getLogLevel

Gets the log level.

Returns

The LogLevel.

3.58.2.26 - (int) getPoolFrameSize

Gets the size of a pool frame in number of extents.

Returns

The size of a pool frame in number of extents.

3.58.2.27 - (int) getPoolPartitions

Gets the number of partitions in each PartitionedPool.

Returns

The number of partitions in each PartitionedPool.

3.58.2.28 - (int) getPoolPersistentMaxSize

Gets the maximum size for the persistent pool in number of frames.

Returns

The maximum size for the persistent pool in number of frames.

3.58.2.29 - (int) getPoolPersistentMinSize

Gets the minimum size for the persistent pool in number of frames.

Returns

The minimum size for the persistent pool in number of frames.

3.58.2.30 - (int) getPoolTemporaryMaxSize

Gets the maximum size for the temporary pool in number of frames.

Returns

The maximum size for the temporary pool in number of frames.

3.58.2.31 - (int) getPoolTemporaryMinSize

Gets the minimum size for the temporary pool in number of frames.

Returns

The minimum size for the temporary pool in number of frames.

3.58.2.32 - (int) getRecoveryCacheMaxSize

Gets the maximum size for the recovery log cache in extents.

Returns

The maximum size for the recovery log cache in extents.

3.58.2.33 - (long long) getRecoveryCheckpointTime

Gets the delay time (in microseconds) between automatic checkpoints.

Returns

The delay time (in microseconds) between automatic checkpoints.

3.58.2.34 - (BOOL) getRecoveryEnabled

Gets whether the recovery is enabled or disabled.

Returns

TRUE if the recovery is enabled, FALSE otherwise.

3.58.2.35 - (NSString*) getRecoveryLogFile

Gets the recovery log file.

Returns

The recovery log file.

3.58.2.36 - (BOOL) getRollbackEnabled

Gets whether the rollback is enabled or disabled.

Returns

TRUE if the rollback is enabled, FALSE otherwise.

3.58.2.37 - (NSString∗) getSparkseeConfigFile

Gets the config file path.

Returns

Returns the configured path or an empty string

3.58.2.38 - (BOOL) getTmpEnabled

Gets whether using temporary storage for computations is enabled.

Returns

True if enabled

3.58.2.39 - (NSString∗) getTmpFolder

Gets the temporary folder used for temporary staging.

Returns

The temporary folder path

3.58.2.40 - (id) init

Creates a new instance.

Values are set with default values.

3.58.2.41 - (id) initWithPath: (NSString ∗) *path*

Creates a new instance with a specific config file.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.

3.58.2.42 - (id) initWithPath: (NSString ∗) *path* clientId:(NSString ∗) *clientId* licenseId:(NSString ∗) *licenseId*

Creates a new instance with a specific config file and IDs.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the config file already exists, the client and license ids should have the same values as the given arguments. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.
clientId	[in] The client identifier.
licenseId	[in] The license identifier.

3.58.2.43 - (BOOL) save

Save the current configuration in the specified config file.

It will try to save the current configuration only if a config file was specified. This method will be automatically used when a Sparksee class downloads a new License.

Returns

Returns true if the config file is successfully written or false otherwise.

3.58.2.44 - (BOOL) saveAll

Save all the current configuration settings in the specified config file.

It will try to save the current configuration only if a config file was specified. The saved config file WILL CONATAIN all the modified settings including the secret ENCRYPTION KEYS. You should usually just use the Save method instead.

Returns

Returns true if the config file is successfully written or false otherwise.

3.58.2.45 - (BOOL) setAESEncryptionEnabled: (NSString ∗) *keyInHex* ivInHex:(NSString ∗) *ivInHex*

Enables storage encryption using AES with the given key and iv.

The key and initialization vector will not be saved in the config file.

Parameters

keyInHex	[In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes).
ivInHex	[In] The AES Initialization Vector as an hexadecimal string (16 bytes).

Returns

Returns true if the encryption had been enabled with the given arguments or false otherwise.

3.58.2.46 - (void) setCacheMaxSize: (int) *megaBytes*

Sets the maximum size for the cache (all pools) in MB.

Parameters

megaBytes	[in] The maximum size for the cache (all pools) in MB. It must be non-negative.

3.58.2.47 - (void) setCacheStatisticsEnabled: (BOOL) *status*

Enables or disables cache statistics.

Parameters

status	[in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

3.58.2.48 - (void) setCacheStatisticsFile: (NSString ∗) *filePath*

Sets the cache statistics log file.

Useless if cache statistics are disabled.

Parameters

filePath	[in] The cache statistics log file.

3.58.2.49 - (void) setCacheStatisticsSnapshotTime: (long long) *microSeconds*

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters

microSeconds	[in] The cache statistics snapshot time in microseconds.

3.58.2.50 - (void) setCallStackDump: (BOOL) *status*

Sets whether the signals will be captured to dump the call stack or not.

Parameters

status	[in] If TRUE signals must be captured.

3.58.2.51 - (void) setChecksumEnabled: (BOOL) *status*

Enables or disables the storage checksum usage.

Parameters

status	[in] If TRUE this enables the checksum, if FALSE then disables it.

3.58.2.52 - (void) setClientId: (NSString ∗) *clientId*

Set the client identifier.

If you don't have a client identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

Parameters

client↩*Id*	[in] The client identifier.

3.58.2.53 - (void) setExtentPages: (int) *pages*

Sets the number of pages per extent.

Parameters

pages	[in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

3.58.2.54 - (void) setExtentSize: (int) *kBytes*

Sets the size of the extents in KB.

Parameters

kBytes	[in] The size of an extent in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

3.58.2.55 - (void) setHighAvailabilityCoordinators: (NSString ∗) *ip*

Sets the coordinators address and port list.

Parameters

ip	[in] The coordinators address and port list.

3.58.2.56 - (void) setHighAvailabilityEnabled: (BOOL) *status*

Enables or disables high availability mode.

Parameters

status	[in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.

3.58.2.57 - (void) setHighAvailabilityIP: (NSString ∗) *ip*

Sets the IP address and port of the instance.

Parameters

ip	[in] The IP address and port of the instance.

3.58.2.58 - (void) setHighAvailabilityMasterHistory: (long long) *filePath*

Sets the master's history log.

Parameters

filePath	[in] The master's history log.

3.58.2.59 - (void) setHighAvailabilitySynchronization: (long long) *microSeconds*

Sets the synchronization polling time.

Parameters

microSeconds	[in] The synchronization polling time.

3.58.2.60 - (void) setInMemAllocSize: (long long) *size*

Sets the in-memory allocator size.

Parameters

size	The size to set the in-memory allocator to

3.58.2.61 - (int) setLicense: (NSString ∗) *key*

Sets the license key.

Parameters

key	[in] The license key. Returns -1 if the new key can't be used, 0 if it's the same license or 1 if the new license is applied.

3.58.2.62 - (void) setLicenseId: (NSString ∗) *licenseId*

Set the license identifier.

If you don't have a license identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

Parameters

license↩*Id*	[in] The license identifier.

3.58.2.63 - (void) setLicensePreDownloadDays: (int) *days*

Start trying to automatically download a new license at the specified number of days before expiration.

Parameters

days	[in] Number of days before expiration or -1 if the license should never be downloaded.

3.58.2.64 - (void) setLogFile: (NSString ∗) *filePath*

Sets the log file.

Parameters

filePath	[in] The log file.

3.58.2.65 - (void) setLogLevel: (enum STSLogLevel) *level*

Sets the log level.

Parameters

level	[in] The LogLevel.

3.58.2.66 - (void) setPoolFrameSize: (int) *extents*

Sets the size of a pool frame in number of extents.

Parameters

extents	[in] The size of a pool frame in number of extents. It must be non-negative.

3.58.2.67 - (void) setPoolPartitions: (int) *pools*

Sets the number of pools in each PartitionedPool.

Parameters

pools	[in] The number of pools in each PartitionedPool. It must be non-negative.

3.58.2.68 - (void) setPoolPersistentMaxSize: (int) *frames*

Sets the maximum size for the persistent pool in number of frames.

Parameters

frames	[in] The maximum size for the persistent pool in number of frames. It must be non-negative.

3.58.2.69 - (void) setPoolPersistentMinSize: (int) *frames*

Sets the minimum size for the persistent pool in number of frames.

Parameters

frames	[in] The minimum size for the persistent pool in number of frames. It must be non-negative.

3.58.2.70 - (void) setPoolTemporaryMaxSize: (int) *frames*

Sets the maximum size for the temporary pool in number of frames.

Parameters

frames	[in] The maximum size for the temporary pool in number of frames. It must be non-negative.

3.58.2.71 - (void) setPoolTemporaryMinSize: (int) *frames*

Sets the minimum size for the temporary pool in number of frames.

Parameters

frames	[in] The minimum size for the temporary pool in number of frames. It must be non-negative.

3.58.2.72 - (void) setRecoveryCacheMaxSize: (int) *extents*

Sets the maximum size for the recovery log cache in extents.

Parameters

extents	[in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

3.58.2.73 - (void) setRecoveryCheckpointTime: (long long) *microSeconds*

Sets the delay time (in microseconds) between automatic checkpoints.

Parameters

microSeconds	[in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each commited transaction.

3.58.2.74 - (void) setRecoveryEnabled: (BOOL) *status*

Enables or disables the recovery.

Parameters

status	[in] If TRUE this enables the recovery, if FALSE then disables it.

3.58.2.75 - (void) setRecoveryLogFile: (NSString ∗) *filePath*

Sets the recovery log file.

Parameters

filePath	[in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log)

3.58.2.76 - (void) setRollbackEnabled: (BOOL) *status*

Enables or disables the rollback.

Parameters

status	[in] If TRUE this enables the rollback, if FALSE then disables it.

3.58.2.77 - (void) setSparkseeConfigFile: (NSString ∗) *path*

Sets the config file path.

The file is not loaded in this operation, see the constructor options if you need to load the file. The config file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

Parameters

path	[in] File path to the config file.

3.58.2.78 - (void) setTmpEnabled: (BOOL) *enabled*

Sets whether to use temporary storage for computations or not.

Parameters

enabled	True to use temporary storage for computation

3.58.2.79 - (void) setTmpFolder: (NSString ∗) *tmpFolder*

Sets the temporary folder used for temporary staging.

Parameters

tmpFolder	The temporary folder to set

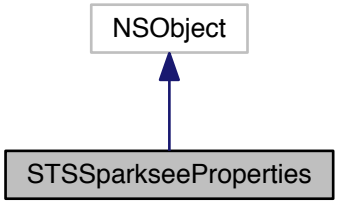
The documentation for this class was generated from the following file:

- Sparksee.h

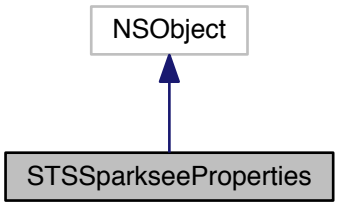
3.59 STSSparkseeProperties Class Reference

Sparksee properties file.

Inheritance diagram for STSSparkseeProperties:



Collaboration diagram for STSSparkseeProperties:



Class Methods

- (void) + load:
  Loads properties from the given file path.
- (NSString ∗) + get:def:
  Gets a property.
- (int) + getInteger:def:
  Gets a property as an integer.
- (BOOL) + getBoolean:def:
  Gets a property as a boolean.
- (long long) + getTimeUnit:def:
  Gets a property as a time unit.
- (void) + clear
  Remove all the properties.
- (void) + setHI:
  YOU SHOULD NOT USE THIS METHOD.

3.59.1 Detailed Description

Sparksee properties file.

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: key=value

By default, this loads properties from the file './sparksee.cfg'. The user may choose to load a different file by calling the method Load().

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

3.59.2 Method Documentation

3.59.2.1 + (NSString∗) get: (NSString ∗) *key* def:(NSString ∗) *def*

Gets a property.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The value of the property, or def if the key is not found.

3.59.2.2 + (BOOL) getBoolean: (NSString ∗) *key* def:(BOOL) *def*

Gets a property as a boolean.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The property value, or def if the key is not found or in case of error.

3.59.2.3 + (int) getInteger: (NSString ∗) *key* def:(int) *def*

Gets a property as an integer.

Parameters

key	[in] The name of the property to lookup.
def	[in] Default value to be returned in case there is no property with the name key.

Returns

The property value, or def if the key is not found or in case of error.

3.59.2.4 + (long long) getTimeUnit: (NSString ∗) *key* def:(long long) *def*

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Valid format for the string representation: Blanks at the begining or at the end are ignored. No blanks are allowed between the time duration and the unit time.

Allowed time units: 'D' for days, 'H' for hours, 'M' for minutes, 'S' o 's' for seconds, 'm' for milliseconds and 'u' for microseconds.

There is a special case: If no time unit is given, seconds is the default. So, '10' means 10 seconds.

Parameters

key	[in] The name of the property to lookup.
def	[in] The default value (in microseconds) to be returned in case there is no property with the name key.

Returns

The time duration in microseconds, or def if the key is not found or in case of error.

3.59.2.5 + (void) load: (NSString ∗) *path*

Loads properties from the given file path.

Parameters

path	[in] File path to load properties from.

3.59.2.6 + (void) setHl: (NSString ∗) *value*

YOU SHOULD NOT USE THIS METHOD.

This method exists only for a specific service.

Parameters

value	null

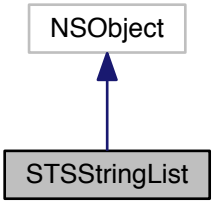
The documentation for this class was generated from the following file:

- Sparksee.h

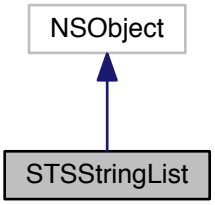
3.60 STSStringList Class Reference

String list.

Inheritance diagram for STSStringList:



Collaboration diagram for STSStringList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - add:

  Adds a String at the end of the list.
- (void) - clear

  Clears the list.
- (id) - initWithArray:

  Creates a new StringList instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new StringList instance from the given NSEnumerator.
- (STSStringListIterator ∗) - iterator

  Gets a new StringListIterator.

3.60.1 Detailed Description

String list.

It stores a String (unicode) list.

Use StringListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.60.2 Method Documentation

3.60.2.1 - (void) add: (NSString ∗) *str*

Adds a String at the end of the list.

Parameters

str	[in] String.

3.60.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.60.2.3 - (id) init

Constructor.

This creates an empty list.

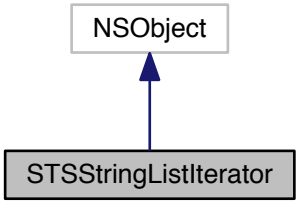
The documentation for this class was generated from the following file:

- Sparksee.h

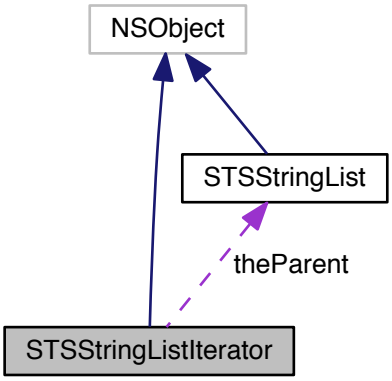
3.61 STSStringListIterator Class Reference

StringList iterator class.

Inheritance diagram for STSStringListIterator:



Collaboration diagram for STSStringListIterator:



Instance Methods

- (NSString ∗) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.61.1 Detailed Description

StringList iterator class.

Iterator to traverse all the strings into a StringList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.61.2 Method Documentation

3.61.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.61.2.2 - (NSString*) next

Moves to the next element.

Returns

The next element.

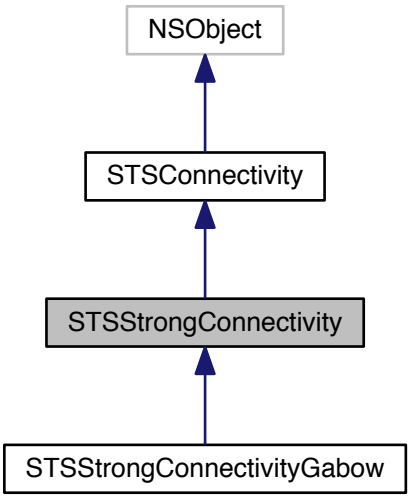
The documentation for this class was generated from the following file:

- Sparksee.h

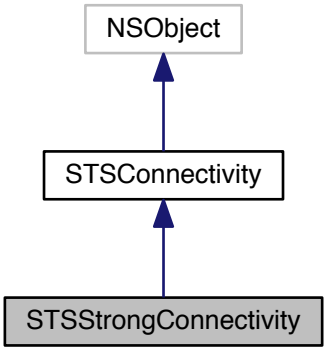
3.62 STSStrongConnectivity Class Reference

StrongConnectivity class.

Inheritance diagram for STSStrongConnectivity:



Collaboration diagram for STSStrongConnectivity:



Instance Methods

- (void) - addEdgeType:dir:

  Allows connectivity through edges of the given type.
- (void) - addAllEdgeTypes:

  Allows connectivity through all edge types of the graph.
- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (STSConnectedComponents ∗) - getConnectedComponents

  Returns the results generated by the execution of the algorithm.
- (void) - run

  Runs the algorithm in order to find the connected components.
- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- (void) - close

  Closes the Connectivity instance.
- (BOOL) - isClosed

  Check if the Connectivity instance is closed.

3.62.1 Detailed Description

StrongConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.62.2 Method Documentation

3.62.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows connectivity through all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.62.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows connectivity through edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.62.2.3 - (void) addNodeType: (int) *t*

Allows connectivity through nodes of the given type.

Parameters

t	null

3.62.2.4 - (void) close

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

3.62.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.62.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.62.2.7 - (STSConnectedComponents∗) getConnectedComponents

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

3.62.2.8 - (void) run

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in STSWeakConnectivityDFS, and STSStrongConnectivityGabow.

3.62.2.9 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

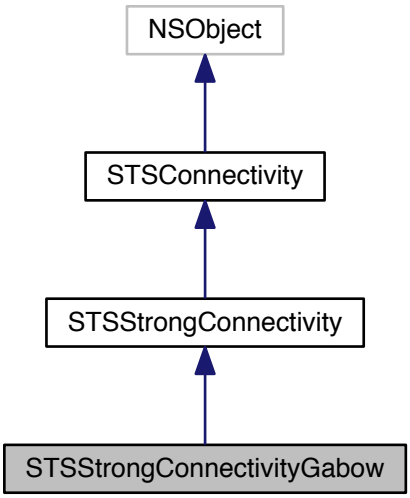
The documentation for this class was generated from the following file:

- Sparksee.h

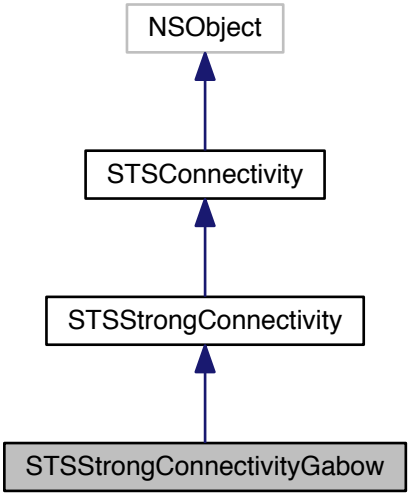
3.63 STSStrongConnectivityGabow Class Reference

This class can be used to solve the problem of finding strongly connected components in a directed graph.

Inheritance diagram for STSStrongConnectivityGabow:



Collaboration diagram for STSStrongConnectivityGabow:



Instance Methods

- (id) - initWithSession:

  Creates a new instance of StrongConnectivityGabow.
- (void) - run

  Executes the algorithm.
- (void) - addEdgeType:dir:

  Allows connectivity through edges of the given type.
- (void) - addAllEdgeTypes:

  Allows connectivity through all edge types of the graph.
- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (STSConnectedComponents ∗) - getConnectedComponents

  Returns the results generated by the execution of the algorithm.
- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- (void) - close

  Closes the Connectivity instance.
- (BOOL) - isClosed

  Check if the Connectivity instance is closed.

3.63.1 Detailed Description

This class can be used to solve the problem of finding strongly connected components in a directed graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type. This implementation is based on the Gabow algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the GetConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.63.2 Method Documentation

3.63.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows connectivity through all edge types of the graph.

Parameters

dir	[in] Edge direction.

3.63.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows connectivity through edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

3.63.2.3 - (void) addNodeType: (int) *t*

Allows connectivity through nodes of the given type.

Parameters

t	null

3.63.2.4 - (void) close

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

3.63.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.63.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.63.2.7 - (STSConnectedComponents∗) getConnectedComponents

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

3.63.2.8 - (id) initWithSession: (STSSession ∗) *session*

Creates a new instance of StrongConnectivityGabow.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the strong connected components.

Parameters

session	[in] Session to get the graph from and calculate the connectivity

3.63.2.9 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

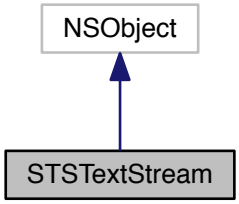
The documentation for this class was generated from the following file:

- Sparksee.h

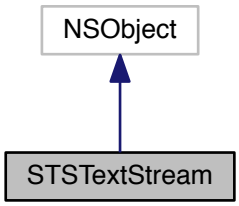
3.64 STSTextStream Class Reference

TextStream class.

Inheritance diagram for STSTextStream:



Collaboration diagram for STSTextStream:



Instance Methods

- (id) - initWithAppend:

  Creates a new instance.
- (BOOL) - isNull

  Returns TRUE if the stream is not available.
- (void) - writeString:

  Write data to the stream.
- (NSString ∗) - readString:

  Read data from the stream.
- (void) - close

  Closes the TextStream instance.
- (BOOL) - isClosed

  Check if the TextStream instance is closed.

3.64.1 Detailed Description

TextStream class.

It allows for reading and writting Text attribute values.

It is very important to close the stream once no more reading or writting operations will be performed to ensure data is successfully stored.

Whereas string attributes are set and got using the Value class, text attributes are operated using a stream pattern.

Use of TextStream for writing: (i) Create a TextStream instance and (ii) set the stream for a text attribute of a node or edge instance with the graph SetAttributeText method. Once the set attribute text has been done, (iii) perform as many write operations as you need to the TextStream instance. Lastly, (iv) exeucte Close to flush and close the stream.

Use of TextStream for reading: (i) Get the stream of a text attribute of a node or edge instance with the Get↩ AttributeText graph method. Once you have the TextStream instance, (ii) you can execute Read operations to read from the stream. (iii) The end of the stream is reached when Read returns 0. Finally, (iv) execute Close to close stream resources.

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.64.2 Method Documentation

3.64.2.1 - (void) close

Closes the TextStream instance.

It must be called to ensure the integrity of all data.

3.64.2.2 - (id) initWithAppend: (BOOL) *append*

Creates a new instance.

A TextStream only can be created by the user to write data.

Parameters

append	[in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the begining of the stream.

3.64.2.3 - (BOOL) isNull

Returns TRUE if the stream is not available.

It returns for reading or writing data.

Returns

FALSE if the stream is ready

3.64.2.4 - (NSString*) readString: (int) *length*

Read data from the stream.

Reads a certain amount of characters (<= length). If the result string is empty, there is no more data to be read from the stream.

Parameters

length	[in] Maximum length to be read. It must be > 0.

Returns

Returns the string readed.

3.64.2.5 - (void) writeString: (NSString *) *inStr*

Write data to the stream.

Parameters

inStr	[in] The source string.

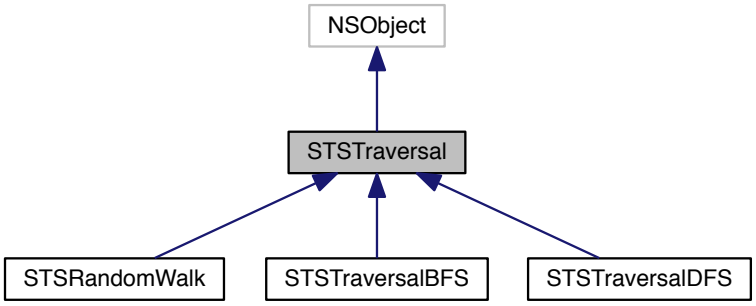
The documentation for this class was generated from the following file:

- Sparksee.h

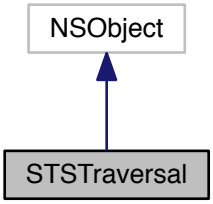
3.65 STSTraversal Class Reference

Traversal class.

Inheritance diagram for STSTraversal:



Collaboration diagram for STSTraversal:



Instance Methods

- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (long long) - next

  Gets the next object of the traversal.
- (BOOL) - hasNext

  Gets if there are more objects to be traversed.
- (int) - getCurrentDepth

  Returns the depth of the current node.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - close

  Closes the Traversal instance.
- (BOOL) - isClosed

  Check if the Traversal instance is closed.

3.65.1 Detailed Description

Traversal class.

Any class implementing this abstract class can be used to traverse a graph.

Once the instance has been created and the allowed node and edge types has been set, it can be used as an iterator, retrieving the next object identifier of the traversal until there are no more.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.65.2 Method Documentation

3.65.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.65.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.65.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

Implemented in STSRandomWalk.

3.65.2.4 - (void) close

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

3.65.2.5 - (void) excludeEdges: (STSObjects *) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.65.2.6 - (void) excludeNodes: (STSObjects *) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.65.2.7 - (int) getCurrentDepth

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

Implemented in STSRandomWalk, STSTraversalBFS, and STSTraversalDFS.

3.65.2.8 - (BOOL) hasNext

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

Implemented in STSRandomWalk, STSTraversalBFS, and STSTraversalDFS.

3.65.2.9 - (long long) next

Gets the next object of the traversal.

Returns

A node or edge identifier.

Implemented in STSRandomWalk, STSTraversalBFS, and STSTraversalDFS.

3.65.2.10 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

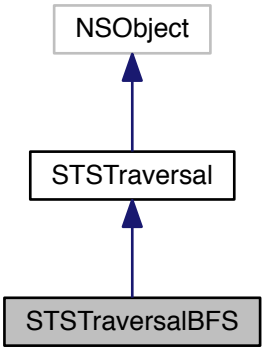
The documentation for this class was generated from the following file:

- Sparksee.h

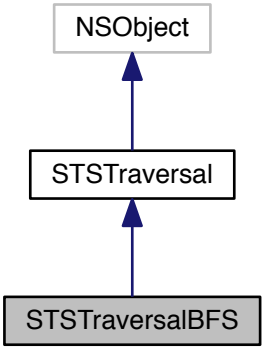
3.66 STSTraversalBFS Class Reference

Breadth-First Search implementation of Traversal.

Inheritance diagram for STSTraversalBFS:



Collaboration diagram for STSTraversalBFS:



Instance Methods

- (long long) - next

  Gets the next object of the traversal.
- (BOOL) - hasNext

  Gets if there are more objects to be traversed.
- (int) - getCurrentDepth

  Returns the depth of the current node.
- (id) - initWithSession:node:

  Creates a new instance.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - close

  Closes the Traversal instance.
- (BOOL) - isClosed

  Check if the Traversal instance is closed.

3.66.1 Detailed Description

Breadth-First Search implementation of Traversal.

Starting from a source node, it visits all its neighbors at distance 1, then all its neighbors at distance 2, and so on.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.66.2 Method Documentation

3.66.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.66.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.66.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

Implemented in STSRandomWalk.

3.66.2.4 - (void) close

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

3.66.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.66.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.66.2.7 - (int) getCurrentDepth

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

Implements STSTraversal.

3.66.2.8 - (BOOL) hasNext

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

Implements STSTraversal.

3.66.2.9 - (id) initWithSession: (STSSession *) *session* node:(long long) *node*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
node	[in] Node to start traversal from.

3.66.2.10 - (long long) next

Gets the next object of the traversal.

Returns

A node or edge identifier.

Implements STSTraversal.

3.66.2.11 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

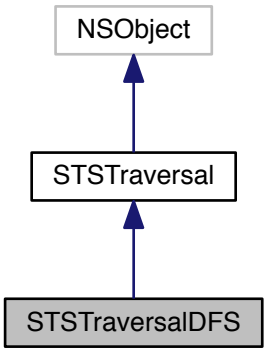
The documentation for this class was generated from the following file:

- Sparksee.h

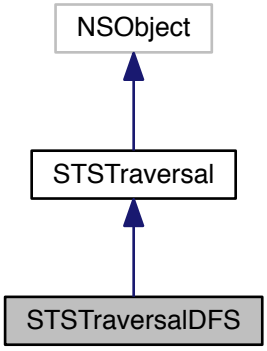
3.67 STSTraversalDFS Class Reference

Depth-First Search (DFS) implementation of Traversal.

Inheritance diagram for STSTraversalDFS:



Collaboration diagram for STSTraversalDFS:



Instance Methods

- (long long) - next

  Gets the next object of the traversal.
- (BOOL) - hasNext

  Gets if there are more objects to be traversed.
- (int) - getCurrentDepth

  Returns the depth of the current node.
- (id) - initWithSession:node:

  Creates a new instance.
- (void) - addEdgeType:dir:

  Allows for traversing edges of the given type.
- (void) - addAllEdgeTypes:

  Allows for traversing all edge types of the graph.
- (void) - addNodeType:

  Allows for traversing nodes of the given type.
- (void) - addAllNodeTypes

  Allows for traversing all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (void) - setMaximumHops:

  Sets the maximum hops restriction.
- (void) - close

  Closes the Traversal instance.
- (BOOL) - isClosed

  Check if the Traversal instance is closed.

3.67.1 Detailed Description

Depth-First Search (DFS) implementation of Traversal.

Starting from a source or root node, it visits as far as possible along each branch before backtracking.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.67.2 Method Documentation

3.67.2.1 - (void) addAllEdgeTypes: (enum STSEdgesDirection) *dir*

Allows for traversing all edge types of the graph.

Parameters

dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.67.2.2 - (void) addEdgeType: (int) *type* dir:(enum STSEdgesDirection) *dir*

Allows for traversing edges of the given type.

Parameters

type	[in] Edge type.
dir	[in] Edge direction.

Implemented in STSRandomWalk.

3.67.2.3 - (void) addNodeType: (int) *type*

Allows for traversing nodes of the given type.

Parameters

type	The node type to add

Implemented in STSRandomWalk.

3.67.2.4 - (void) close

Closes the Traversal instance.

It must be called to ensure the integrity of all data.

3.67.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.67.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

Implemented in STSRandomWalk.

3.67.2.7 - (int) getCurrentDepth

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to Next().

Returns

The depth of the current node.

Implements STSTraversal.

3.67.2.8 - (BOOL) hasNext

Gets if there are more objects to be traversed.

Returns

TRUE if there are more objects, FALSE otherwise.

Implements STSTraversal.

3.67.2.9 - (id) initWithSession: (STSSession *) *session* node:(long long) *node*

Creates a new instance.

Parameters

session	[in] Session to get the graph from and perform traversal.
node	[in] Node to start traversal from.

3.67.2.10 - (long long) next

Gets the next object of the traversal.

Returns

A node or edge identifier.

Implements STSTraversal.

3.67.2.11 - (void) setMaximumHops: (int) *maxhops*

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters

maxhops	[in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

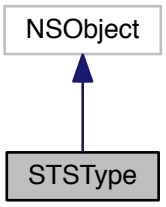
The documentation for this class was generated from the following file:

- Sparksee.h

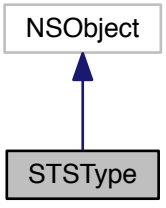
3.68 STSType Class Reference

Type data class.

Inheritance diagram for STSType:



Collaboration diagram for STSType:



Instance Methods

- (int) - getId

  Gets the Sparksee type identifier.
- (enum STSObjectType) - getObjectType

  Gets the object type.
- (NSString ∗) - getName

  Gets the unique type name.
- (long long) - getNumObjects

  Gets the number of objects belonging to the type.
- (BOOL) - getIsDirected

  Gets if this is a directed edge type.
- (BOOL) - getIsRestricted

  Gets if this is a restricted edge type.
- (BOOL) - getAreNeighborsIndexed

  Gets if this is an edge type with neighbors index.
- (int) - getRestrictedFrom

  Gets the tail or source type identifier for restricted edge types.
- (int) - getRestrictedTo

  Gets the head or target type identifier for restricted edge types.

Class Methods

- (int) + getInvalidType

  Invalid type identifier.
- (int) + getGlobalType

  Global type identifier.
- (int) + getNodesType

  Identifier for all nodeType attributes.
- (int) + getEdgesType

  Identifier for all edgeType attributes.

3.68.1 Detailed Description

Type data class.

It contains information about a node or edge type.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.68.2 Method Documentation

3.68.2.1 - (BOOL) getAreNeighborsIndexed

Gets if this is an edge type with neighbors index.

Returns

TRUE for edges types with neighbors index, FALSE otherwise.

3.68.2.2 - (int) getId

Gets the Sparksee type identifier.

Returns

The Sparksee type identifier.

3.68.2.3 - (BOOL) getIsDirected

Gets if this is a directed edge type.

Returns

TRUE for directed edge types, FALSE otherwise.

3.68.2.4 - (BOOL) getIsRestricted

Gets if this is a restricted edge type.

Returns

TRUE for restricted edge types, FALSE otherwise.

3.68.2.5 - (NSString∗) getName

Gets the unique type name.

Returns

The unique type name.

3.68.2.6 - (long long) getNumObjects

Gets the number of objects belonging to the type.

Returns

The number of objects belonging to the type.

3.68.2.7 - (enum STSObjectType) getObjectType

Gets the object type.

Returns

The object type.

3.68.2.8 - (int) getRestrictedFrom

Gets the tail or source type identifier for restricted edge types.

Returns

For restricted edge types, the tail or source type identifier, the Type InvalidType otherwise.

3.68.2.9 - (int) getRestrictedTo

Gets the head or target type identifier for restricted edge types.

Returns

For restricted edge types, the head or target type identifier, the Type InvalidType otherwise.

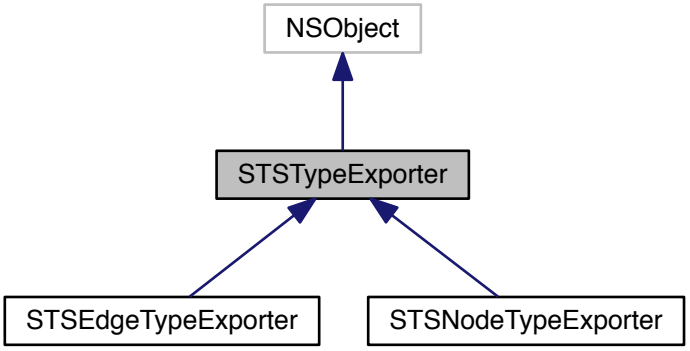
The documentation for this class was generated from the following file:

- Sparksee.h

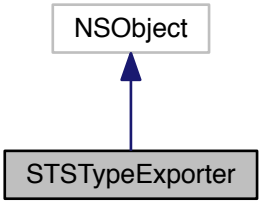
3.69 STSTypeExporter Class Reference

Base TypeExporter class.

Inheritance diagram for STSTypeExporter:



Collaboration diagram for STSTypeExporter:



Instance Methods

- (void) - registerListener:

  Registers a new listener.
- (void) - run

  Runs export process.
- (void) - setRowWriter:

  Sets the output data destination.
- (void) - setGraph:

  Sets the graph that will be exported.
- (void) - setType:

  Sets the type to be exported.
- (void) - setAttributes:

  Sets the list of Attributes.
- (void) - setFrequency:

  Sets the frequency of listener notification.
- (void) - setHeader:

  Sets the presence of a header row.

3.69.1 Detailed Description

Base TypeExporter class.

Base class to export a node or edge type from a graph using a RowWriter.

TypeExporterListener can be registered to receive information about the progress of the export process by means of TypeExporterEvent. The default frequency of notification to listeners is 100000.

By default no header row is created.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.69.2 Method Documentation

3.69.2.1 - (void) registerListener: (STSTypeExporterListener *) *tel*

Registers a new listener.

Parameters

tel	[in] TypeExporterListener to be registered.

3.69.2.2 - (void) run

Runs export process.

Exceptions

System.ApplicationException	null
System.IO.IOException	If bad things happen writing to the RowWriter.

Implemented in STSNodeTypeExporter, and STSEdgeTypeExporter.

3.69.2.3 - (void) setAttributes: (STSAttributeList *) attrs

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be exported

3.69.2.4 - (void) setFrequency: (int) freq

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.69.2.5 - (void) setGraph: (STSGraph *) graph

Sets the graph that will be exported.

Parameters

graph	[in] Graph.

3.69.2.6 - (void) setHeader: (BOOL) header

Sets the presence of a header row.

Parameters

header	[in] If TRUE, a header row is dumped with the name of the attributes.

3.69.2.7 - (void) setRowWriter: (STSRowWriter *) rw

Sets the output data destination.

Parameters

rw	[in] Input RowWriter.

3.69.2.8 - (void) setType: (int) *type*

Sets the type to be exported.

Parameters

type	[in] Type identifier.

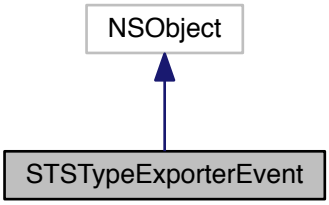
The documentation for this class was generated from the following file:

- Sparksee.h

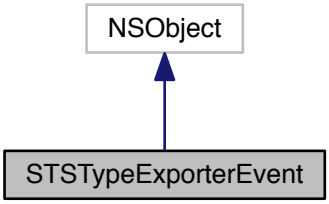
3.70 STSTypeExporterEvent Class Reference

Provides information about the progress of an TypeExproter instance.

Inheritance diagram for STSTypeExporterEvent:



Collaboration diagram for STSTypeExporterEvent:



Instance Methods

- (int) - getTypeId

  Gets the type identifier.
- (long long) - getCount

  Gets the current number of objects exported.
- (long long) - getTotal

  Gets the total number of objects exported.
- (BOOL) - isLast

  Gets if this is the last event or not.

3.70.1 Detailed Description

Provides information about the progress of an TypeExproter instance.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.70.2 Method Documentation

3.70.2.1 - (long long) getCount

Gets the current number of objects exported.

Returns

The current number of objects exported.

3.70.2.2 - (long long) getTotal

Gets the total number of objects exported.

Returns

The total number of objects exported.

3.70.2.3 - (int) getTypeId

Gets the type identifier.

Returns

The type identifier.

3.70.2.4 - (BOOL) isLast

Gets if this is the last event or not.

Returns

TRUE if this is the last event, FALSE otherwise.

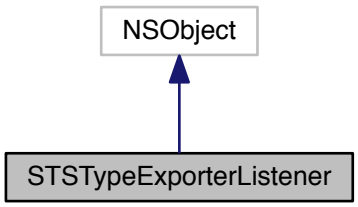
The documentation for this class was generated from the following file:

- Sparksee.h

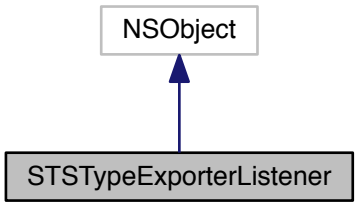
3.71 STSTypeExporterListener Class Reference

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Inheritance diagram for STSTypeExporterListener:



Collaboration diagram for STSTypeExporterListener:



Instance Methods

- (void) - notifyEvent:

  Method to be notified from a TypeExporter.

3.71.1 Detailed Description

Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.71.2 Method Documentation

3.71.2.1 - (void) notifyEvent: (STSTypeExporterEvent *) *tee*

Method to be notified from a TypeExporter.

Parameters

tee	[in] Notified event.

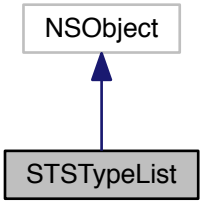
The documentation for this class was generated from the following file:

- Sparksee.h

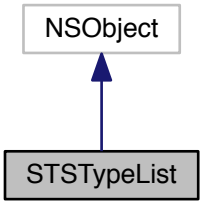
3.72 STSTypeList Class Reference

Sparksee type identifier list.

Inheritance diagram for STSTypeList:



Collaboration diagram for STSTypeList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - add:

  Adds a Sparksee type identifier at the end of the list.
- (void) - clear

  Clears the list.
- (id) - initWithArray:

  Creates a new TypeList instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new TypeList instance from the given NSEnumerator.
- (STSTypeListIterator ∗) - iterator

  Gets a new TypeListIterator.

3.72.1 Detailed Description

Sparksee type identifier list.

It stores a Sparksee node or edge type identifier list.

Use TypeListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.72.2 Method Documentation

3.72.2.1 - (void) add: (int) *type*

Adds a Sparksee type identifier at the end of the list.

Parameters

type	[in] Sparksee type identifier.

3.72.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.72.2.3 - (id) init

Constructor.

This creates an empty list.

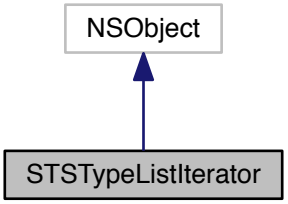
The documentation for this class was generated from the following file:

- Sparksee.h

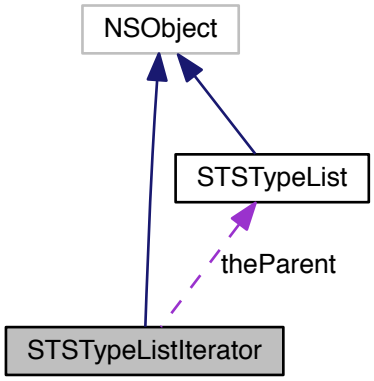
3.73 STSTypeListIterator Class Reference

TypeList iterator class.

Inheritance diagram for STSTypeListIterator:



Collaboration diagram for STSTypeListIterator:



Instance Methods

- (int) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.73.1 Detailed Description

TypeList iterator class.

Iterator to traverse all the Sparksee node or edge type identifiers into a TypeList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.73.2 Method Documentation

3.73.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.73.2.2 - (int) next

Moves to the next element.

Returns

The next element.

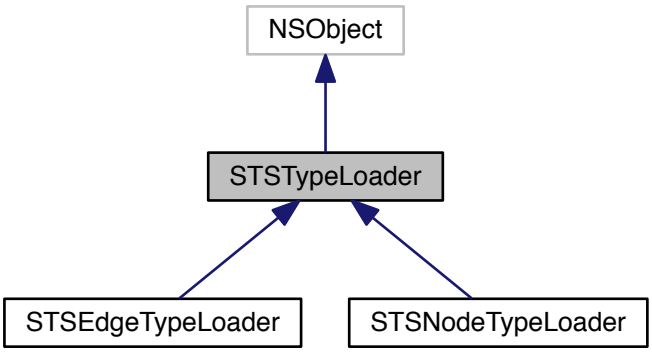
The documentation for this class was generated from the following file:

- Sparksee.h

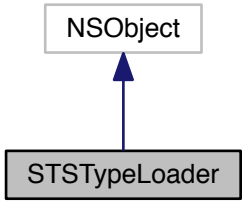
3.74 STSTypeLoader Class Reference

Base TypeLoader class.

Inheritance diagram for STSTypeLoader:



Collaboration diagram for STSTypeLoader:



Instance Methods

- (void) - setLogError:
  Sets a log error file.
- (void) - setLogOff
  Truns off all the error reporting.
- (void) - registerListener:
  Registers a new listener.
- (void) - run
  Run the loader.
- (void) - runTwoPhases
  Run the loader for two phases loading.
- (void) - runNPhases:
  Run the loader for N phases loading.
- (void) - setRowReader:
  Sets the input data source.
- (void) - setGraph:
  Sets the graph where the data will be loaded.
- (void) - setLocale:
  Sets the locale that will be used to read the data.
- (void) - setType:
  Sets the type to be loaded.
- (void) - setAttributes:
  Sets the list of Attributes.
- (void) - setAttributePositions:
  Sets the list of attribute positions.
- (void) - setTimestampFormat:
  Sets a specific timestamp format.
- (void) - setFrequency:
  Sets the frequency of listener notification.

3.74.1 Detailed Description

Base TypeLoader class.

Base class to load a node or edge type from a graph using a RowReader.

TypeLoaderListener can be registered to receive information about the progress of the load process by means of TypeLoaderEvent. The default frequency of notification to listeners is 100000.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.74.2 Method Documentation

3.74.2.1 - (void) registerListener: (STSTypeLoaderListener *) *tel*

Registers a new listener.

Parameters

tel	TypeLoaderListener to be registered.

3.74.2.2 - (void) run

Run the loader.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implemented in STSNodeTypeLoader, and STSEdgeTypeLoader.

3.74.2.3 - (void) runNPhases: (int) *partitions*

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

Parameters

partitions	[in] Number of horizontal partitions to perform the load.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implemented in STSNodeTypeLoader, and STSEdgeTypeLoader.

3.74.2.4 - (void) runTwoPhases

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

Exceptions

System.ApplicationException	null
System.IO.IOException	null

Implemented in STSNodeTypeLoader, and STSEdgeTypeLoader.

3.74.2.5 - (void) setAttributePositions: (STSInt32List *) attrsPos

Sets the list of attribute positions.

Parameters

attrsPos	[in] Attribute positions (column index >=0).

3.74.2.6 - (void) setAttributes: (STSAttributeList *) attrs

Sets the list of Attributes.

Parameters

attrs	[in] Attribute identifiers to be loaded

3.74.2.7 - (void) setFrequency: (int) freq

Sets the frequency of listener notification.

Parameters

freq	[in] Frequency in number of rows managed to notify progress to all listeners

3.74.2.8 - (void) setGraph: (STSGraph *) graph

Sets the graph where the data will be loaded.

Parameters

graph	[in] Graph.

3.74.2.9 - (void) setLocale: (NSString *) localeStr

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters

localeStr	[in] The locale string for the read data. See CSVReader.

3.74.2.10 - (void) setLogError: (NSString *) path

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Parameters

path	[in] The path to the error log file.

Exceptions

System.IO.IOException	If bad things happen opening the file.

3.74.2.11 - (void) setLogOff

Truns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

3.74.2.12 - (void) setRowReader: (STSRowReader ∗) *rr*

Sets the input data source.

Parameters

rr	[in] Input RowReader.

3.74.2.13 - (void) setTimestampFormat: (NSString ∗) *timestampFormat*

Sets a specific timestamp format.

Parameters

timestampFormat	[in] A string with the timestamp format definition.

3.74.2.14 - (void) setType: (int) *type*

Sets the type to be loaded.

Parameters

type	[in] Type identifier.

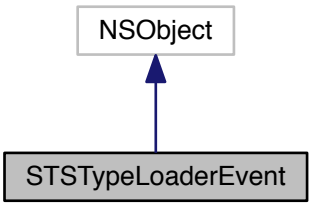
The documentation for this class was generated from the following file:

- Sparksee.h

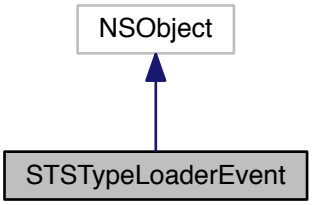
3.75 STSTypeLoaderEvent Class Reference

Provides information about the progress of a TypeLoader instance.

Inheritance diagram for STSTypeLoaderEvent:



Collaboration diagram for STSTypeLoaderEvent:



Instance Methods

- (int) - getTypeId
  Gets the type identifier.
- (long long) - getCount
  Gets the current number of objects created.
- (int) - getPhase
  Gets the current phase.
- (int) - getTotalPhases
  Gets the total number of phases.
- (int) - getPartition
  Gets the current partition.
- (int) - getTotalPartitions
  Gets the total number of partitions.
- (int) - getTotalPartitionSteps
  Gets the total number of steps in the current partition.
- (BOOL) - isLast
  Gets if this is the last event or not.

3.75.1 Detailed Description

Provides information about the progress of a TypeLoader instance.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.75.2 Method Documentation

3.75.2.1 - (long long) getCount

Gets the current number of objects created.

Returns

The current number of objects created.

3.75.2.2 - (int) getPartition

Gets the current partition.

Returns

The current partition.

3.75.2.3 - (int) getPhase

Gets the current phase.

Returns

The current phase.

3.75.2.4 - (int) getTotalPartitions

Gets the total number of partitions.

Returns

The total number of partitions.

3.75.2.5 - (int) getTotalPartitionSteps

Gets the total number of steps in the current partition.

Returns

The total number steps in the current partition.

3.75.2.6 - (int) getTotalPhases

Gets the total number of phases.

Returns

The total number of phases.

3.75.2.7 - (int) getTypeId

Gets the type identifier.

Returns

The type identifier.

3.75.2.8 - (BOOL) isLast

Gets if this is the last event or not.

Returns

TRUE if this is the last event, FALSE otherwise.

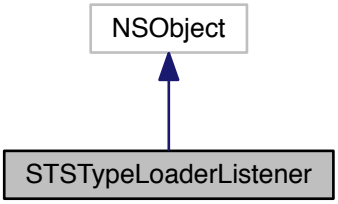
The documentation for this class was generated from the following file:

- Sparksee.h

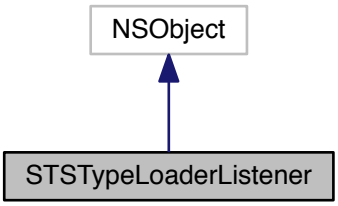
3.76 STSTypeLoaderListener Class Reference

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.

Inheritance diagram for STSTypeLoaderListener:



Collaboration diagram for STSTypeLoaderListener:



Instance Methods

- (void) - notifyEvent:
  Method to receive events from a Loader.

3.76.1 Detailed Description

Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.76.2 Method Documentation

3.76.2.1 - (void) notifyEvent: (STSTypeLoaderEvent *) *ev*

Method to receive events from a Loader.

Parameters

ev	Loader.LoaderEvent with information from a running Loader.

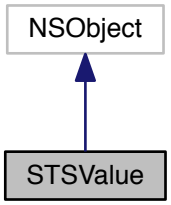
The documentation for this class was generated from the following file:

- Sparksee.h

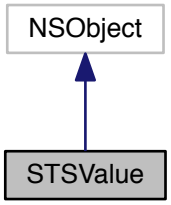
3.77 STSValue Class Reference

Value class.

Inheritance diagram for STSValue:



Collaboration diagram for STSValue:



Instance Methods

- (id) - init

  Creates a new instance.
- (id) - initWithValue:

  Copy constructor.
- (BOOL) - isNull

  Gets if this is a NULL Value.
- (void) - setNullVoid

  Sets the Value to NULL.
- (enum STSDataType) - getDataType

  Gets the DataType.
- (BOOL) - getBoolean

  Gets Boolean Value.
- (int) - getInteger

  Gets Integer Value.
- (long long) - getLong

  Gets Long Value.
- (double) - getDouble

  Gets Double Value.

- (long long) - getTimestamp

  Gets Timestamp Value.
- (NSString ∗) - getString

  Gets String Value.
- (long long) - getOid

  Gets OID Value.
- (void) - setBooleanVoid:

  Sets the Value.
- (void) - setIntegerVoid:

  Sets the Value.
- (void) - setLongVoid:

  Sets the Value.
- (void) - setDoubleVoid:

  Sets the Value.
- (void) - setTimestampVoid:

  Sets the Value.
- (void) - setTimestampVoidWithYear:month:day:hour:minutes:seconds:millisecs:

  Sets the Value.
- (void) - setStringVoid:

  Sets the Value.
- (void) - setOidVoid:

  Sets the OID Value.
- (void) - setVoid:

  Sets the Value.
- (int) - compare:

  Compares with the given Value.
- (BOOL) - equals:

  Compares with the given Value.
- (NSString ∗) - stringValue

  Returns the receiver's value as a human-readable string.
- (BOOL) - isEqual:

  Check if both Value instances are equal.
- (NSUInteger) - hash

  Get the hash value of this Value.
- (STSValue ∗) - setNull

  Sets the value to NULL.
- (STSValue ∗) - setBoolean:

  Sets the value to the given boolean.
- (STSValue ∗) - setInteger:

  Sets the value to the given integer number.
- (STSValue ∗) - setLong:

  Sets the value to the given long long number.
- (STSValue ∗) - setDouble:

  Sets the value to the given double number.
- (STSValue ∗) - setTimestamp:

  Sets the value to the given timestamp.
- (STSValue ∗) - setTimestampWithYear:month:day:hour:minutes:seconds:milliseconds:

  Sets the value to the given timestamp.
- (STSValue ∗) - setString:

  Sets the value to the given string.
- (STSValue ∗) - setOid:

Sets the value to the given OID.

- (STSValue ∗) - set:

  Sets the value to the given Value.

- (NSDate ∗) - getTimestampAsNSDate

  Gets Timestamp Value as a NSDate.

- (STSValue ∗) - setTimestampWithNSDate:

  Sets the value to the given NSDate timestamp.

Class Methods

- (int) + getMaxLengthString

  Maximum number of characters allowed for a String.

3.77.1 Detailed Description

Value class.

It is a container which stores a value and its data type (domain). A Value can be NULL.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.77.2 Method Documentation

3.77.2.1 - (int) compare: (STSValue ∗) *value*

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

Parameters

value	Given value to compare to.

Returns

0 if this Value is equal to the given one; a value less than 0 if this Value is less than the given one; and a value greater than 0 if this Value is greater than the given one.

3.77.2.2 - (BOOL) equals: (STSValue ∗) *value*

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

Parameters

value	Given value to compare to.

Returns

TRUE if this Value is equal to the given one; FALSE otherwise.

3.77.2.3 - (BOOL) getBoolean

Gets Boolean Value.

This must be a non-NULL Boolean Value.

Returns

The Boolean Value.

3.77.2.4 - (enum STSDataType) getDataType

Gets the DataType.

Value cannot be NULL.

Returns

The DataType.

3.77.2.5 - (double) getDouble

Gets Double Value.

This must be a non-NULL Double Value.

Returns

The Double Value.

3.77.2.6 - (int) getInteger

Gets Integer Value.

This must be a non-NULL Integer Value.

Returns

The Integer Value.

3.77.2.7 - (long long) getLong

Gets Long Value.

This must be a non-NULL Long Value.

Returns

The Long Value.

3.77.2.8 - (long long) getOid

Gets OID Value.

This must be an non-NULL OID Value.

Returns

The OID Value.

3.77.2.9 - (NSString∗) getString

Gets String Value.

This must be a non-NULL String Value.

Returns

The String Value.

3.77.2.10 - (long long) getTimestamp

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

Returns

The Timestamp Value.

3.77.2.11 - (NSDate ∗) getTimestampAsNSDate

Gets Timestamp Value as a NSDate.

This must be a non-NULL Timestamp Value.

Returns

The Timestamp Value.

3.77.2.12 - (id) init

Creates a new instance.

It creates a NULL Value.

3.77.2.13 - (id) initWithValue: (STSValue ∗) *value*

Copy constructor.

Parameters

value	[in] Value to be copied.

3.77.2.14 - (BOOL) isNull

Gets if this is a NULL Value.

Returns

TRUE if this is a NULL Value, FALSE otherwise.

3.77.2.15 - (STSValue∗) set: (STSValue ∗) *value*

Sets the value to the given Value.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.16 - (STSValue∗) setBoolean: (BOOL) *value*

Sets the value to the given boolean.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.17 - (void) setBooleanVoid: (BOOL) *value*

Sets the Value.

Parameters

value	[in] New Boolean value.

3.77.2.18 - (STSValue∗) setDouble: (double) *value*

Sets the value to the given double number.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.19 - (void) setDoubleVoid: (double) *value*

Sets the Value.

Parameters

value	[in] New Double value.

3.77.2.20 - (STSValue*) setInteger: (int) *value*

Sets the value to the given integer number.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.21 - (void) setIntegerVoid: (int) *value*

Sets the Value.

Parameters

value	[in] New Integer value.

3.77.2.22 - (STSValue*) setLong: (long long) *value*

Sets the value to the given long long number.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.23 - (void) setLongVoid: (long long) *value*

Sets the Value.

Parameters

value	[in] New Long value.

3.77.2.24 - (STSValue*) setNull

Sets the value to NULL.

Returns

Returns this Value.

3.77.2.25 - (STSValue*) setOid: (long long) *value*

Sets the value to the given OID.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.26 - (void) setOidVoid: (long long) *value*

Sets the OID Value.

Parameters

value	[in] New OID value.

3.77.2.27 - (STSValue*) setString: (NSString *) *value*

Sets the value to the given string.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.28 - (void) setStringVoid: (NSString ∗) *value*

Sets the Value.

Parameters

value	[in] New String value.

3.77.2.29 - (STSValue∗) setTimestamp: (long long) *value*

Sets the value to the given timestamp.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.30 - (void) setTimestampVoid: (long long) *value*

Sets the Value.

Parameters

value	[in] New Timestamp value.

3.77.2.31 - (void) setTimestampVoidWithYear: (int) *year* month:(int) *month* day:(int) *day* hour:(int) *hour* minutes:(int) *minutes* seconds:(int) *seconds* millisecs:(int) *millisecs*

Sets the Value.

Parameters

year	[in] The year (>=1970).
month	[in] The month ([1..12]).
day	[in] The of the month ([1..31]).
hour	[in] The hour ([0..23]).
minutes	[in] The minutes ([0..59]).
seconds	[in] The seconds ([0..59]).
millisecs	[in] The milliseconds ([0..999]).

3.77.2.32 - (STSValue∗) setTimestampWithNSDate: (NSDate ∗) *date*

Sets the value to the given NSDate timestamp.

Parameters

value	[in] The new value.

Returns

Returns this Value.

3.77.2.33 - (STSValue*) setTimestampWithYear: (int) *year* month:(int) *month* day:(int) *day* hour:(int) *hour* minutes:(int) *minutes* seconds:(int) *seconds* milliseconds:(int) *millisecs*

Sets the value to the given timestamp.

Parameters

year	[in] The year (>=1970).
month	[in] The month ([1..12]).
day	[in] The of the month ([1..31]).
hour	[in] The hour ([0..23]).
minutes	[in] The minutes ([0..59]).
seconds	[in] The seconds ([0..59]).
millisecs	[in] The milliseconds ([0..999]).

Returns

Returns this Value.

3.77.2.34 - (void) setVoid: (STSValue *) *value*

Sets the Value.

Parameters

value	[in] New value.

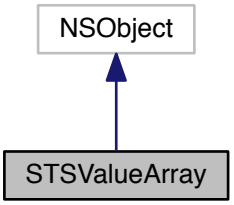
The documentation for this class was generated from the following file:

- Sparksee.h

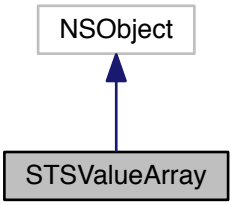
3.78 STSValueArray Class Reference

ValueArray class.

Inheritance diagram for STSValueArray:



Collaboration diagram for STSValueArray:



Instance Methods

- (void) - get:value:

  Get a Value from the array.
- (void) - setAt:value:

  Set a Value to a specific array position.
- (void) - set:

  Set a Value to the whole array.
- (void) - makeNull

  Sets the attribute array to Null.
- (int) - size

  Returns the array size.
- (void) - setDouble:

  Set all the values of this double array.
- (void) - setDoubleRange:values:

  Set a subset of the values of this double array.
- (void) - setInteger:

  Set all the values of this int array.
- (void) - setIntegerRange:values:

  Set a subset of the values of this int array.

- (void) - setLong:

  Set all the values of this long array.
- (void) - setLongRange:values:

  Set a subset of the values of this long array.
- (void) - setBoolean:

  Set all the values of this bool array.
- (void) - setBooleanRange:values:

  Set a subset of the values of this bool array.
- (void) - setTimestamp:

  Set all the values of this timestamp array.
- (void) - setTimestampRange:values:

  Set a subset of the values of this timestamp array.
- (void) - setOid:

  Set all the values of this oid array.
- (void) - setOidRange:values:

  Set a subset of the values of this oid array.
- (NSArray ∗) - getDouble

  Get all the Values from this double array.
- (NSArray ∗) - getDoubleRange:size:

  Get a subset of the Values from this double array.
- (NSArray ∗) - getInteger

  Get all the values from this int array

Exceptions

AppError	*If the ValueArray is not available*
UnsupportedOperationError	*If array DataType is not int.*

- (NSArray ∗) - getIntegerRange:size:

  Get a subset of the values from this int array.
- (NSArray ∗) - getLong

  Get all the values from this long array

Exceptions

AppError	*If the ValueArray is not available*
UnsupportedOperationError	*If array DataType is not long.*

- (NSArray ∗) - getBoolean

  Get all the values from this bool array

Exceptions

AppError	*If the ValueArray is not available*
UnsupportedOperationError	*If array DataType is not bool.*

- (NSArray ∗) - getTimestamp

  Get all the values from this timestamp array

Exceptions

AppError	*If the ValueArray is not available*
UnsupportedOperationError	*If array DataType is not timestamp.*

- (NSArray ∗) - getOid

  Get all the values from this oid array

Exceptions

AppError	*If the ValueArray is not available*
UnsupportedOperationError	*If array DataType is not oid.*

- (void) - close

  Closes the ValueArray instance.
- (BOOL) - isClosed

  Check if the ValueArray instance is closed.

3.78.1 Detailed Description

ValueArray class.

It allows for getting and setting ValueArray attribute values.

It is very important to close the ValueArray once no more get or set operations will be performed.

Creation of a new ValueArray: (i) Set all the ValueArray elements using Graph::SetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Use of an existing ValueArray: (i) Get a ValueArray instance using Graph::GetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.78.2 Method Documentation

3.78.2.1 - (void) close

Closes the ValueArray instance.

It must be called to ensure the integrity of all data.

3.78.2.2 - (void) get: (int) *index* value:(STSValue ∗) *value*

Get a Value from the array.

AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the element to get [0..N-1]
value	[out] Value to get the array element

Exceptions

System.ArgumentException	If the index is out of range or the Value not valid
System.ApplicationException	null

3.78.2.3 - (NSArray∗) getDouble

Get all the Values from this double array.

Returns

Returns all the array values

3.78.2.4 - (NSArray∗) getDoubleRange: (int) *index* size:(int) *size*

Get a subset of the Values from this double array.

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

Returns

Returns the array values in the selected range

3.78.2.5 - (NSArray∗) getIntegerRange: (int) *index* size:(int) *size*

Get a subset of the values from this int array.

Parameters

index	[in] Position of the first element to get [0..N-1]
size	[in] Number of elements to get [1..N]

Exceptions

WrongArgumentError	If the index or size is out of range
UnsupportedOperationError	If array DataType is not int
AppError	If the ValueArray is not available

3.78.2.6 - (void) makeNull

Sets the attribute array to Null.

The ValueArray can not be used after this call.

Exceptions

System.ApplicationException	null

3.78.2.7 - (void) set: (STSValue ∗) *value*

Set a Value to the whole array.

AppErrorIf the ValueArray is not available

Parameters

value	[in] Value to set in all the array elements

Exceptions

System.ArgumentException	If the Value is not valid
System.ApplicationException	null

3.78.2.8 - (void) setAt: (int) *index* value:(STSValue ∗) *value*

Set a Value to a specific array position.

AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the element to set [0..N-1]
value	[in] Value to set in the array element

Exceptions

System.ArgumentException	If the index is out of range or the Value not valid
System.ApplicationException	null

3.78.2.9 - (void) setBoolean: (NSArray ∗) *values*

Set all the values of this bool array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not bool

3.78.2.10 - (void) setBooleanRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this bool array.

UnsupportedOperationErrorIf array DataType is not bool AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

3.78.2.11 - (void) setDouble: (NSArray ∗) *values*

Set all the values of this double array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not Double

3.78.2.12 - (void) setDoubleRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this double array.

UnsupportedOperationErrorIf array DataType is not Double AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

3.78.2.13 - (void) setInteger: (NSArray ∗) *values*

Set all the values of this int array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not int

3.78.2.14 - (void) setIntegerRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this int array.

UnsupportedOperationErrorIf array DataType is not int AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

3.78.2.15 - (void) setLong: (NSArray ∗) *values*

Set all the values of this long array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not long

3.78.2.16 - (void) setLongRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this long array.

UnsupportedOperationErrorIf array DataType is not long AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

3.78.2.17 - (void) setOid: (NSArray ∗) *values*

Set all the values of this oid array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not oid

3.78.2.18 - (void) setOidRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this oid array.

UnsupportedOperationErrorIf array DataType is not oid AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

3.78.2.19 - (void) setTimestamp: (NSArray ∗) *values*

Set all the values of this timestamp array.

AppErrorIf the ValueArray is not available

Parameters

values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ApplicationException	If array DataType is not timestamp

3.78.2.20 - (void) setTimestampRange: (int) *index* values:(NSArray ∗) *values*

Set a subset of the values of this timestamp array.

UnsupportedOperationErrorIf array DataType is not timestamp AppErrorIf the ValueArray is not available

Parameters

index	[in] Position of the first element to set [0..N-1]
values	[in] All the values to set

Exceptions

System.ApplicationException	null
System.ArgumentException	If the index or size is out of range
System.ApplicationException	null

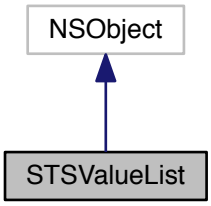
The documentation for this class was generated from the following file:

- Sparksee.h

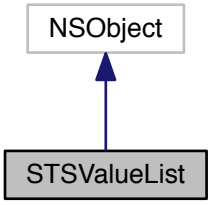
3.79 STSValueList Class Reference

Value list.

Inheritance diagram for STSValueList:



Collaboration diagram for STSValueList:



Instance Methods

- (int) - count

  Number of elements in the list.
- (id) - init

  Constructor.
- (void) - clear

  Clears the list.
- (void) - add:

  Adds a value to the end of the list.
- (STSValue ∗) - get:

  Returns the Value at the specified position in the list.
- (id) - initWithArray:

  Creates a new ValueList instance from the given array.
- (id) - initWithNSEnumerator:

  Creates a new ValueList instance from the given NSEnumerator.
- (STSValueListIterator ∗) - iterator

  Gets a new ValueListIterator.

3.79.1 Detailed Description

Value list.

It stores a Value list.

Use ValueListIterator to access all elements into this collection.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.79.2 Method Documentation

3.79.2.1 - (void) add: (STSValue *) value

Adds a value to the end of the list.

Parameters

value	[in] The value to add

3.79.2.2 - (int) count

Number of elements in the list.

Returns

Number of elements in the list.

3.79.2.3 - (STSValue*) get: (int) index

Returns the Value at the specified position in the list.

Parameters

index	[in] Index of the element to return, starting at 0.

3.79.2.4 - (id) init

Constructor.

This creates an empty list.

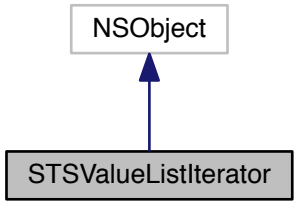
The documentation for this class was generated from the following file:

- Sparksee.h

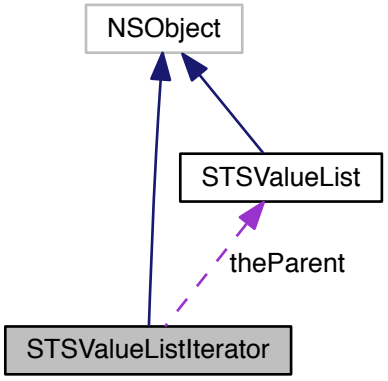
3.80 STSValueListIterator Class Reference

ValueList iterator class.

Inheritance diagram for STSValueListIterator:



Collaboration diagram for STSValueListIterator:



Instance Methods

- (STSValue ∗) - next

  Moves to the next element.
- (BOOL) - hasNext

  Gets if there are more elements.

3.80.1 Detailed Description

ValueList iterator class.

Iterator to traverse all the values into a ValueList instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.80.2 Method Documentation

3.80.2.1 - (BOOL) hasNext

Gets if there are more elements.

Returns

TRUE if there are more elements, FALSE otherwise.

3.80.2.2 - (STSValue∗) next

Moves to the next element.

Returns

The next element.

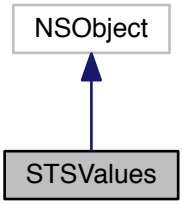
The documentation for this class was generated from the following file:

- Sparksee.h

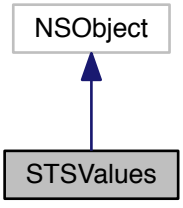
3.81 STSValues Class Reference

Value set class.

Inheritance diagram for STSValues:



Collaboration diagram for STSValues:



Instance Methods

- (long long) - count

  Gets the number of elements into the collection.
- (STSValuesIterator ∗) - iterator:

  Gets a ValuesIterator.
- (void) - close

  Closes the Values instance.
- (BOOL) - isClosed

  Check if the Values instance is closed.

3.81.1 Detailed Description

Value set class.

This is a set of Value instances, that is there is no duplicated elements.

Use a ValuesIterator to traverse all the elements into the set.

When the Values instance is closed, it closes all existing and non-closed ValuesIterator instances too.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.81.2 Method Documentation

3.81.2.1 - (void) close

Closes the Values instance.

It must be called to ensure the integrity of all data.

3.81.2.2 - (long long) count

Gets the number of elements into the collection.

Returns

The number of elements into the collection.

3.81.2.3 - (STSValuesIterator∗) iterator: (enum STSOrder) *order*

Gets a ValuesIterator.

Parameters

order	[in] Ascending or descending order.

Returns

ValuesIterator instance.

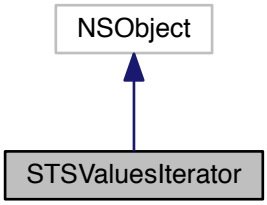
The documentation for this class was generated from the following file:

- Sparksee.h

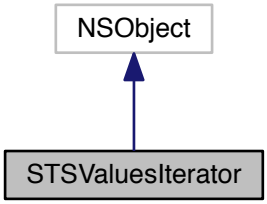
3.82 STSValuesIterator Class Reference

Values iterator class.

Inheritance diagram for STSValuesIterator:



Collaboration diagram for STSValuesIterator:



Instance Methods

- (BOOL) - hasNext

  Gets if there are more elements to traverse.
- (STSValue ∗) - next

  Gets the next element to traverse.
- (void) - close

  Closes the ValuesIterator instance.
- (BOOL) - isClosed

  Check if the ValuesIterator instance is closed.

3.82.1 Detailed Description

Values iterator class.

It allows for traversing all the elements into a Values instance.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.82.2 Method Documentation

3.82.2.1 - (void) close

Closes the ValuesIterator instance.

It must be called to ensure the integrity of all data.

3.82.2.2 - (BOOL) hasNext

Gets if there are more elements to traverse.

Returns

TRUE if there are more elements to traverse, FALSE otherwise.

3.82.2.3 - (STSValue*) next

Gets the next element to traverse.

Returns

The next element.

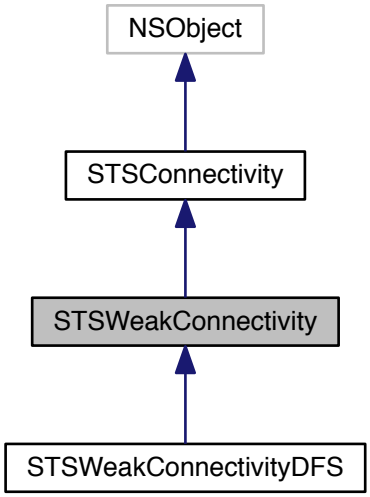
The documentation for this class was generated from the following file:

- Sparksee.h

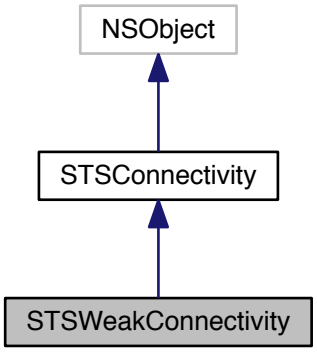
3.83 STSWeakConnectivity Class Reference

WeakConnectivity class.

Inheritance diagram for STSWeakConnectivity:



Collaboration diagram for STSWeakConnectivity:



Instance Methods

- (void) - addEdgeType:

  Allows connectivity through edges of the given type.

- (void) - addAllEdgeTypes

  Allows connectivity through all edge types of the graph.
- (void) - addNodeType:

  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes

  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:

  Set which nodes can't be used.
- (void) - excludeEdges:

  Set which edges can't be used.
- (STSConnectedComponents *) - getConnectedComponents

  Returns the results generated by the execution of the algorithm.
- (void) - run

  Runs the algorithm in order to find the connected components.
- (void) - setMaterializedAttribute:

  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- (void) - close

  Closes the Connectivity instance.
- (BOOL) - isClosed

  Check if the Connectivity instance is closed.

3.83.1 Detailed Description

WeakConnectivity class.

Any class implementing this abstract class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.83.2 Method Documentation

3.83.2.1 - (void) addAllEdgeTypes

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

3.83.2.2 - (void) addEdgeType: (int) *type*

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

Parameters

type	[in] Edge type.

3.83.2.3 - (void) addNodeType: (int) *t*

Allows connectivity through nodes of the given type.

Parameters

t	null

3.83.2.4 - (void) close

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

3.83.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.83.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.83.2.7 - (STSConnectedComponents∗) getConnectedComponents

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

3.83.2.8 - (void) run

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in STSWeakConnectivityDFS, and STSStrongConnectivityGabow.

3.83.2.9 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

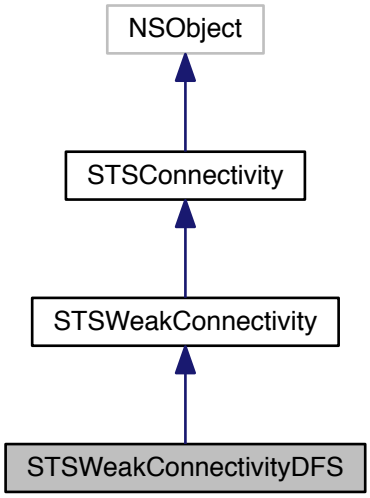
The documentation for this class was generated from the following file:

- Sparksee.h

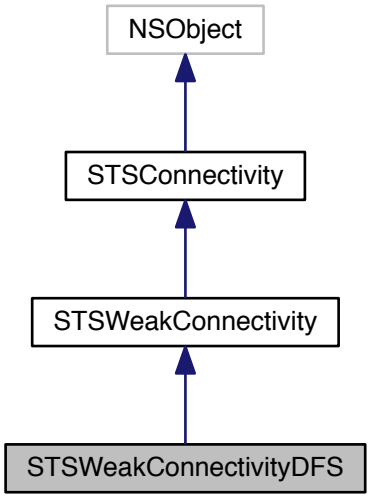
3.84 STSWeakConnectivityDFS Class Reference

WeakConnectivityDFS class.

Inheritance diagram for STSWeakConnectivityDFS:



Collaboration diagram for STSWeakConnectivityDFS:



Instance Methods

- (id) - initWithSession:

  Creates a new instance of WeakConnectivityDFS.

- (void) - run
  Executes the algorithm.
- (void) - addEdgeType:
  Allows connectivity through edges of the given type.
- (void) - addAllEdgeTypes
  Allows connectivity through all edge types of the graph.
- (void) - addNodeType:
  Allows connectivity through nodes of the given type.
- (void) - addAllNodeTypes
  Allows connectivity through all node types of the graph.
- (void) - excludeNodes:
  Set which nodes can't be used.
- (void) - excludeEdges:
  Set which edges can't be used.
- (STSConnectedComponents *) - getConnectedComponents
  Returns the results generated by the execution of the algorithm.
- (void) - setMaterializedAttribute:
  Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.
- (void) - close
  Closes the Connectivity instance.
- (BOOL) - isClosed
  Check if the Connectivity instance is closed.

3.84.1 Detailed Description

WeakConnectivityDFS class.

This class can be used to solve the problem of finding weakly connected components in an undirected graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u. This implementation is based on the Depth-First Search (DFS) algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the ConnectedComponents class using the getConnectedComponents method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author

Sparsity Technologies http://www.sparsity-technologies.com

3.84.2 Method Documentation

3.84.2.1 - (void) addAllEdgeTypes

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in Any direction.

3.84.2.2 - (void) addEdgeType: (int) *type*

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

Parameters

type	[in] Edge type.

3.84.2.3 - (void) addNodeType: (int) *t*

Allows connectivity through nodes of the given type.

Parameters

t	null

3.84.2.4 - (void) close

Closes the Connectivity instance.

It must be called to ensure the integrity of all data.

3.84.2.5 - (void) excludeEdges: (STSObjects ∗) *edges*

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters

edges	[in] A set of edge identifiers that must be kept intact until the destruction of the class.

3.84.2.6 - (void) excludeNodes: (STSObjects ∗) *nodes*

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters

nodes	[in] A set of node identifiers that must be kept intact until the destruction of the class.

3.84.2.7 - (STSConnectedComponents∗) getConnectedComponents

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns

Returns an instance of the class ConnectedComponents which contain information related to the connected components found.

3.84.2.8 - (id) initWithSession: (STSSession ∗) *session*

Creates a new instance of WeakConnectivityDFS.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the weak connected components.

Parameters

session	[in] Session to get the graph from and calculate the connectivity

3.84.2.9 - (void) setMaterializedAttribute: (NSString ∗) *attributeName*

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class ConnectedComponents indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters

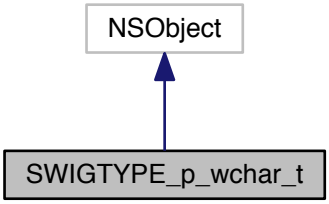
attributeName	[in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

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

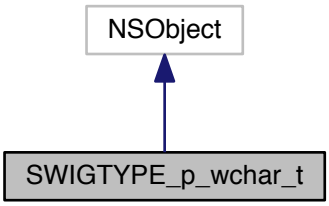
- Sparksee.h

3.85 SWIGTYPE_p_wchar_t Class Reference

Inheritance diagram for SWIGTYPE_p_wchar_t:



Collaboration diagram for SWIGTYPE_p_wchar_t:



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

- Sparksee.h

Index