

Real-Time Twitter Data Stream Analysis

SE-2XB3-G24

Generated by Doxygen 1.8.11

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	Package se2xb3	9
5.2	Package se2xb3.bootstrap	9
5.3	Package se2xb3.config	9
5.4	Package se2xb3.control	9
5.5	Package se2xb3.data	10
5.6	Package se2xb3.data.algorithms	10
5.7	Package se2xb3.data.algorithms.sort	10
5.8	Package se2xb3.data.algorithms.sort.textbook	10
5.9	Package se2xb3.data.models	11
5.10	Package se2xb3.data.processing	11
5.11	Package se2xb3.injection	11
5.12	Package se2xb3.io	11
5.13	Package se2xb3.io.files	11
5.14	Package se2xb3.io.source	12
5.15	Package se2xb3.io.web	12
5.16	Package se2xb3.io.web.handlers	12
5.17	Package se2xb3.tests	12

6	Class Documentation	13
6.1	se2xb3.config.AppConfig Interface Reference	13
6.1.1	Detailed Description	13
6.1.2	Member Function Documentation	14
6.1.2.1	hostname()	14
6.1.2.2	trendRefreshRate()	14
6.2	se2xb3.control.AppController Class Reference	14
6.2.1	Detailed Description	15
6.2.2	Constructor & Destructor Documentation	16
6.2.2.1	AppController()	16
6.2.3	Member Function Documentation	16
6.2.3.1	getDataController()	16
6.2.3.2	getExecutorService()	16
6.2.3.3	getInstance()	16
6.2.3.4	getIOController()	17
6.2.3.5	shutdown()	17
6.2.3.6	shutdownApp()	17
6.2.3.7	start()	17
6.2.4	Member Data Documentation	18
6.2.4.1	app	18
6.2.4.2	dataController	18
6.2.4.3	executorService	18
6.2.4.4	ioController	18
6.3	se2xb3.io.AsyncMessageLooper< M > Class Template Reference	18
6.3.1	Detailed Description	19
6.3.2	Constructor & Destructor Documentation	20
6.3.2.1	AsyncMessageLooper(IMessageReceiver< M > receiver, ExecutorService executorService)	20
6.3.3	Member Function Documentation	20
6.3.3.1	enqueue(M msg)	20
6.3.3.2	run()	20

6.3.3.3	shutdown()	21
6.3.4	Member Data Documentation	21
6.3.4.1	blockingQueue	21
6.3.4.2	doShutdown	21
6.3.4.3	messageReceiver	21
6.4	se2xb3.data.models.BoundingBox Class Reference	22
6.4.1	Member Data Documentation	22
6.4.1.1	attributes	22
6.4.1.2	coordinates	22
6.4.1.3	type	22
6.5	se2xb3.config.Constants Interface Reference	23
6.5.1	Detailed Description	24
6.5.2	Member Data Documentation	24
6.5.2.1	DATA_SOURCE_STREAM_RATE	24
6.5.2.2	DEFAULT_REST_RESP_COUNT	24
6.5.2.3	INPUT_FILE	25
6.5.2.4	INPUT_FILE_NAME	25
6.5.2.5	JAVADOC_INDEX_FILE	25
6.5.2.6	JAVADOC_ROOT_PATH	25
6.5.2.7	LIVE_STREAM	25
6.5.2.8	MAX_REST_RESP_COUNT	25
6.5.2.9	REST_REQUEST_NO_MATCH	25
6.5.2.10	REST_RESOURCE_TRENDS_ALL	25
6.5.2.11	REST_RESOURCE_TRENDS_HASHTAGS	25
6.5.2.12	REST_RESOURCE_TRENDS_USERS	25
6.5.2.13	REST_TREND_RESOURCE_PATH_REGEX	26
6.5.2.14	TESTING	26
6.5.2.15	TWEET_REGEX	26
6.5.2.16	WEB_APP_ROOT_FILE	26
6.5.2.17	WEB_APP_ROOT_PATH	26

6.6	se2xb3.control.Controller Class Reference	26
6.6.1	Detailed Description	27
6.6.2	Constructor & Destructor Documentation	27
6.6.2.1	Controller()	27
6.6.3	Member Function Documentation	28
6.6.3.1	print(String s)	28
6.6.3.2	println(String s)	28
6.6.3.3	shutdown()	28
6.7	se2xb3.io.web.handlers.CORSHandler Class Reference	28
6.7.1	Detailed Description	29
6.7.2	Member Function Documentation	30
6.7.2.1	handle(Context ctx)	30
6.7.3	Member Data Documentation	30
6.7.3.1	_CLASS_NAME	30
6.7.3.2	log	30
6.7.3.3	requestCount	30
6.8	se2xb3.data.DataController Class Reference	31
6.8.1	Detailed Description	32
6.8.2	Constructor & Destructor Documentation	33
6.8.2.1	DataController(AppController appController)	33
6.8.3	Member Function Documentation	33
6.8.3.1	getTrendingHashtags(int count)	33
6.8.3.2	getTrendingUsers(int count)	33
6.8.3.3	getTrendingWords(int count)	34
6.8.3.4	getTrends()	34
6.8.3.5	getTrends(int count)	34
6.8.3.6	getTweetList()	35
6.8.3.7	onMessageReceivedEvent(String msg)	35
6.8.3.8	resetData()	35
6.8.3.9	shutdown()	35

6.8.4	Member Data Documentation	36
6.8.4.1	app	36
6.8.4.2	dataHandler	36
6.8.4.3	tweetList	36
6.9	se2xb3.data.DataHandler Class Reference	37
6.9.1	Detailed Description	39
6.9.2	Constructor & Destructor Documentation	39
6.9.2.1	DataHandler(DataController dataController)	39
6.9.3	Member Function Documentation	39
6.9.3.1	getTrendingHashtags(int count)	39
6.9.3.2	getTrendingUsers(int count)	40
6.9.3.3	getTrendingWords(int count)	40
6.9.3.4	getTrends()	41
6.9.3.5	getTrends(int count)	41
6.9.3.6	print(String s)	41
6.9.3.7	println(String s)	42
6.9.3.8	processTweet(String tweetStr)	42
6.9.3.9	resetData()	42
6.9.4	Member Data Documentation	42
6.9.4.1	byValue	42
6.9.4.2	dataController	42
6.9.4.3	graphStrategy	42
6.9.4.4	objectMapper	42
6.9.4.5	searchStrategy	42
6.9.4.6	sortStrategy	42
6.9.4.7	tweetProcessor	42
6.9.4.8	tweetsList	42
6.10	se2xb3.io.source.DataSource Class Reference	43
6.10.1	Detailed Description	45
6.10.2	Constructor & Destructor Documentation	45

6.10.2.1	DataSource(IOController controller)	45
6.10.3	Member Function Documentation	45
6.10.3.1	getExecutorService()	45
6.10.3.2	getQueue()	46
6.10.3.3	getUrl()	46
6.10.3.4	passMessageToController(String msg)	46
6.10.3.5	run()	46
6.10.3.6	setQueue(IMessageQueue< String > queue)	47
6.10.3.7	setUrl(String url)	47
6.10.3.8	shutdown()	47
6.10.3.9	startSource()	47
6.10.4	Member Data Documentation	48
6.10.4.1	controller	48
6.10.4.2	msgHandler	48
6.10.4.3	queue	48
6.10.4.4	url	48
6.11	se2xb3.control.EventBus Class Reference	48
6.11.1	Detailed Description	49
6.11.2	Constructor & Destructor Documentation	49
6.11.2.1	EventBus()	49
6.11.3	Member Function Documentation	49
6.11.3.1	postNewMessageEvent(String msg)	49
6.11.4	Member Data Documentation	49
6.11.4.1	bus	49
6.12	se2xb3.io.source.FileDataSource Class Reference	50
6.12.1	Detailed Description	51
6.12.2	Constructor & Destructor Documentation	52
6.12.2.1	FileDataSource(IOController controller)	52
6.12.3	Member Function Documentation	52
6.12.3.1	print(String s)	52

6.12.3.2	println(String s)	52
6.12.3.3	readData()	52
6.12.3.4	startSource()	53
6.12.4	Member Data Documentation	53
6.12.4.1	_CLASS_NAME	53
6.12.4.2	log	53
6.13	se2xb3.tests.FileDataSourceTest Class Reference	54
6.13.1	Detailed Description	54
6.13.2	Member Function Documentation	55
6.13.2.1	setUp()	55
6.13.2.2	startSource()	55
6.13.2.3	waitForever()	55
6.13.3	Member Data Documentation	55
6.13.3.1	lock	55
6.13.3.2	waitForTest	55
6.14	se2xb3.io.files.FileFinder Class Reference	55
6.14.1	Detailed Description	56
6.14.2	Constructor & Destructor Documentation	56
6.14.2.1	FileFinder()	56
6.14.3	Member Function Documentation	56
6.14.3.1	findFileByName(String name, String extensions)	56
6.14.3.2	findFileByName(String name, String rootDir, String extensions)	56
6.14.3.3	findFileByName(String name, String rootDir, String extensions, boolean recursive)	57
6.14.3.4	findFileByName(String name, String rootDir, String[] extensions, boolean recursive)	57
6.15	se2xb3.data.algorithms.GraphStrategy Class Reference	57
6.15.1	Detailed Description	58
6.15.2	Constructor & Destructor Documentation	58
6.15.2.1	GraphStrategy()	58
6.15.3	Member Function Documentation	58
6.15.3.1	setStrategy()	58

6.16	se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator Class Reference	59
6.16.1	Constructor & Destructor Documentation	60
6.16.1.1	Heaplterator()	60
6.16.2	Member Function Documentation	60
6.16.2.1	hasNext()	60
6.16.2.2	next()	60
6.16.2.3	remove()	60
6.16.3	Member Data Documentation	60
6.16.3.1	copy	60
6.17	se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.Heaplterator Class Reference	61
6.17.1	Constructor & Destructor Documentation	62
6.17.1.1	Heaplterator()	62
6.17.2	Member Function Documentation	62
6.17.2.1	hasNext()	62
6.17.2.2	next()	62
6.17.2.3	remove()	62
6.17.3	Member Data Documentation	62
6.17.3.1	copy	62
6.18	se2xb3.io.source.IDataSource Interface Reference	63
6.18.1	Detailed Description	63
6.19	se2xb3.io.IMessageQueue< M > Interface Template Reference	64
6.19.1	Detailed Description	65
6.19.2	Member Function Documentation	65
6.19.2.1	enqueue(M msg)	65
6.19.2.2	shutdown()	65
6.20	se2xb3.io.IMessageReceiver< M > Interface Template Reference	66
6.20.1	Detailed Description	67
6.20.2	Member Function Documentation	67
6.20.2.1	receiveMessage(M msg)	67
6.21	se2xb3.io.IOController Class Reference	68

6.21.1 Detailed Description	69
6.21.2 Constructor & Destructor Documentation	69
6.21.2.1 IOController(AppController appController)	69
6.21.3 Member Function Documentation	70
6.21.3.1 getExecutorService()	70
6.21.3.2 getTrendingHashtags(int count)	70
6.21.3.3 getTrendingUsers(int count)	70
6.21.3.4 getTrendingWords(int count)	71
6.21.3.5 getTrends()	71
6.21.3.6 getTrends(int count)	71
6.21.3.7 init()	72
6.21.3.8 onMessageReceived(String msg)	72
6.21.3.9 shutdown()	72
6.21.4 Member Data Documentation	72
6.21.4.1 app	72
6.21.4.2 dataSource	72
6.21.4.3 webServer	72
6.22 se2xb3.data.processing.IProcessor< T > Interface Template Reference	73
6.22.1 Detailed Description	74
6.22.2 Member Function Documentation	74
6.22.2.1 process(T t)	74
6.23 se2xb3.data.algorithms.sort.ISort Interface Reference	75
6.23.1 Detailed Description	75
6.23.2 Member Function Documentation	76
6.23.2.1 sort(Comparable[] a)	76
6.24 se2xb3.io.web.JsonParser Class Reference	76
6.24.1 Detailed Description	77
6.24.2 Constructor & Destructor Documentation	77
6.24.2.1 JsonParser()	77
6.24.3 Member Function Documentation	77

6.24.3.1	<code>mapToTrendList(Map< String, Integer > trendMap)</code>	77
6.24.3.2	<code>toContainerObject(List< Map< String, Integer >> trends)</code>	78
6.24.3.3	<code>toContainerObject(Map< String, Integer > trends)</code>	78
6.24.3.4	<code>toJson(List< Map< String, Integer >> list)</code>	79
6.24.3.5	<code>toJson(Map< String, Integer > map)</code>	80
6.24.4	Member Data Documentation	80
6.24.4.1	<code>mapper</code>	80
6.25	<code>se2xb3.data.algorithms.sort.textbook.MaxPQ< Key > Class Template Reference</code>	81
6.25.1	Detailed Description	83
6.25.2	Constructor & Destructor Documentation	84
6.25.2.1	<code>MaxPQ(int initCapacity)</code>	84
6.25.2.2	<code>MaxPQ()</code>	84
6.25.2.3	<code>MaxPQ(int initCapacity, Comparator< Key > comparator)</code>	84
6.25.2.4	<code>MaxPQ(Comparator< Key > comparator)</code>	84
6.25.2.5	<code>MaxPQ(Key[] keys)</code>	85
6.25.3	Member Function Documentation	85
6.25.3.1	<code>delMax()</code>	85
6.25.3.2	<code>exch(int i, int j)</code>	86
6.25.3.3	<code>insert(Key x)</code>	86
6.25.3.4	<code>isEmpty()</code>	86
6.25.3.5	<code>isMaxHeap()</code>	86
6.25.3.6	<code>isMaxHeap(int k)</code>	87
6.25.3.7	<code>iterator()</code>	87
6.25.3.8	<code>less(int i, int j)</code>	87
6.25.3.9	<code>max()</code>	87
6.25.3.10	<code>resize(int capacity)</code>	88
6.25.3.11	<code>sink(int k)</code>	88
6.25.3.12	<code>size()</code>	88
6.25.3.13	<code>swim(int k)</code>	88
6.25.4	Member Data Documentation	88

6.25.4.1	comparator	88
6.25.4.2	n	88
6.25.4.3	pq	88
6.26	se2xb3.data.algorithms.sort.textbook.Merge Class Reference	89
6.26.1	Detailed Description	90
6.26.2	Constructor & Destructor Documentation	90
6.26.2.1	Merge()	90
6.26.3	Member Function Documentation	90
6.26.3.1	indexSort(Comparable[] a)	90
6.26.3.2	isSorted(Comparable[] a)	91
6.26.3.3	isSorted(Comparable[] a, int lo, int hi)	91
6.26.3.4	less(Comparable v, Comparable w)	91
6.26.3.5	merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi)	91
6.26.3.6	merge(Comparable[] a, int[] index, int[] aux, int lo, int mid, int hi)	92
6.26.3.7	show(Comparable[] a)	92
6.26.3.8	sort(Comparable[] a, Comparable[] aux, int lo, int hi)	92
6.26.3.9	sort(Comparable[] a)	92
6.26.3.10	sort(Comparable[] a, int[] index, int[] aux, int lo, int hi)	92
6.27	se2xb3.data.algorithms.sort.textbook.MergeBU Class Reference	93
6.27.1	Detailed Description	94
6.27.2	Constructor & Destructor Documentation	94
6.27.2.1	MergeBU()	94
6.27.3	Member Function Documentation	94
6.27.3.1	isSorted(Comparable[] a)	94
6.27.3.2	less(Comparable v, Comparable w)	94
6.27.3.3	merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi)	94
6.27.3.4	show(Comparable[] a)	95
6.27.3.5	sort(Comparable[] a)	95
6.28	se2xb3.data.algorithms.sort.MergeSortStrategy Class Reference	96
6.28.1	Detailed Description	97

6.28.2	Constructor & Destructor Documentation	97
6.28.2.1	MergeSortStrategy()	97
6.28.3	Member Function Documentation	97
6.28.3.1	sort(Comparable[] a)	97
6.29	se2xb3.data.algorithms.sort.textbook.MergeX Class Reference	98
6.29.1	Detailed Description	99
6.29.2	Constructor & Destructor Documentation	99
6.29.2.1	MergeX()	99
6.29.3	Member Function Documentation	99
6.29.3.1	exch(Object[] a, int i, int j)	99
6.29.3.2	insertionSort(Comparable[] a, int lo, int hi)	99
6.29.3.3	insertionSort(Object[] a, int lo, int hi, Comparator comparator)	99
6.29.3.4	isSorted(Comparable[] a)	100
6.29.3.5	isSorted(Comparable[] a, int lo, int hi)	100
6.29.3.6	isSorted(Object[] a, Comparator comparator)	100
6.29.3.7	isSorted(Object[] a, int lo, int hi, Comparator comparator)	100
6.29.3.8	less(Comparable a, Comparable b)	100
6.29.3.9	less(Object a, Object b, Comparator comparator)	100
6.29.3.10	merge(Comparable[] src, Comparable[] dst, int lo, int mid, int hi)	101
6.29.3.11	merge(Object[] src, Object[] dst, int lo, int mid, int hi, Comparator comparator)	101
6.29.3.12	show(Object[] a)	101
6.29.3.13	sort(Comparable[] src, Comparable[] dst, int lo, int hi)	101
6.29.3.14	sort(Comparable[] a)	102
6.29.3.15	sort(Object[] a, Comparator comparator)	102
6.29.3.16	sort(Object[] src, Object[] dst, int lo, int hi, Comparator comparator)	102
6.29.4	Member Data Documentation	102
6.29.4.1	CUTOFF	102
6.30	se2xb3.io.MessageHandler Class Reference	103
6.30.1	Detailed Description	104
6.30.2	Constructor & Destructor Documentation	105

6.30.2.1	MessageHandler(dataSource dataSource)	105
6.30.3	Member Function Documentation	105
6.30.3.1	getDataSource()	105
6.30.3.2	receiveMessage(String msg)	105
6.30.3.3	setDataSource(dataSource dataSource)	106
6.30.3.4	shutdown()	106
6.30.4	Member Data Documentation	106
6.30.4.1	dataSource	106
6.31	se2xb3.data.algorithms.sort.textbook.MinPQ< Key > Class Template Reference	107
6.31.1	Detailed Description	109
6.31.2	Constructor & Destructor Documentation	110
6.31.2.1	MinPQ(int initCapacity)	110
6.31.2.2	MinPQ()	110
6.31.2.3	MinPQ(int initCapacity, Comparator< Key > comparator)	110
6.31.2.4	MinPQ(Comparator< Key > comparator)	110
6.31.2.5	MinPQ(Key[] keys)	111
6.31.3	Member Function Documentation	111
6.31.3.1	delMin()	111
6.31.3.2	exch(int i, int j)	112
6.31.3.3	greater(int i, int j)	112
6.31.3.4	insert(Key x)	112
6.31.3.5	isEmpty()	112
6.31.3.6	isMinHeap()	113
6.31.3.7	isMinHeap(int k)	113
6.31.3.8	iterator()	113
6.31.3.9	min()	113
6.31.3.10	resize(int capacity)	114
6.31.3.11	sink(int k)	114
6.31.3.12	size()	114
6.31.3.13	swim(int k)	114

6.31.4	Member Data Documentation	114
6.31.4.1	comparator	114
6.31.4.2	n	114
6.31.4.3	pq	114
6.32	se2xb3.data.models.Place Class Reference	115
6.32.1	Detailed Description	116
6.32.2	Constructor & Destructor Documentation	116
6.32.2.1	Place()	116
6.32.3	Member Data Documentation	116
6.32.3.1	bounding_box	116
6.32.3.2	country	116
6.32.3.3	country_code	116
6.32.3.4	full_name	116
6.32.3.5	id	116
6.32.3.6	name	116
6.32.3.7	place_type	116
6.32.3.8	type	116
6.32.3.9	url	116
6.33	se2xb3.data.algorithms.SearchStrategy Class Reference	117
6.33.1	Detailed Description	117
6.33.2	Constructor & Destructor Documentation	117
6.33.2.1	SearchStrategy()	117
6.33.3	Member Function Documentation	117
6.33.3.1	setStrategy()	117
6.34	se2xb3.data.algorithms.SortStrategy Class Reference	118
6.34.1	Detailed Description	118
6.34.2	Constructor & Destructor Documentation	119
6.34.2.1	SortStrategy()	119
6.34.3	Member Function Documentation	119
6.34.3.1	setStrategy()	119

6.34.3.2	sort(Comparable[] a)	119
6.34.3.3	sortTweetList(List< Tweet > tweetList)	119
6.34.4	Member Data Documentation	119
6.34.4.1	mergeSort	119
6.34.4.2	sortStrategy	119
6.35	se2xb3.tests.TestDataSource Class Reference	120
6.35.1	Detailed Description	121
6.35.2	Constructor & Destructor Documentation	121
6.35.2.1	TestDataSource()	121
6.35.3	Member Function Documentation	121
6.35.3.1	endTest()	121
6.35.3.2	main(String[] args)	121
6.35.3.3	waitForever()	122
6.35.4	Member Data Documentation	122
6.35.4.1	blockingQueue	122
6.35.4.2	isDone	122
6.35.4.3	lock	122
6.35.4.4	testing	122
6.35.4.5	waitForTest	122
6.36	se2xb3.io.web.Trend Class Reference	122
6.36.1	Detailed Description	123
6.36.2	Constructor & Destructor Documentation	123
6.36.2.1	Trend(String i, int c)	123
6.36.3	Member Data Documentation	123
6.36.3.1	count	123
6.36.3.2	item	123
6.37	se2xb3.io.web.handlers.TrendHandler Class Reference	124
6.37.1	Detailed Description	126
6.37.2	Constructor & Destructor Documentation	126
6.37.2.1	TrendHandler(IOController ioController)	126

6.37.3	Member Function Documentation	126
6.37.3.1	getTrendingHashtags(String count)	126
6.37.3.2	getTrendingUsers(String count)	126
6.37.3.3	getTrendingWords(String count)	127
6.37.3.4	getTrends()	127
6.37.3.5	getTrends(String count)	127
6.37.3.6	handle(Context ctx)	128
6.37.3.7	isNumber(String s)	129
6.37.3.8	parseInt(String s)	129
6.37.4	Member Data Documentation	129
6.37.4.1	_CLASS_NAME	129
6.37.4.2	ioController	129
6.37.4.3	log	129
6.37.4.4	requestCount	129
6.38	se2xb3.data.models.Tweet Class Reference	130
6.38.1	Detailed Description	130
6.38.2	Member Data Documentation	131
6.38.2.1	id_str	131
6.38.2.2	text	131
6.38.2.3	user	131
6.38.2.4	user_name	131
6.39	se2xb3.data.models.TweetOld Class Reference	132
6.39.1	Constructor & Destructor Documentation	133
6.39.1.1	TweetOld()	133
6.39.1.2	TweetOld(JsonNode n)	134
6.39.2	Member Function Documentation	134
6.39.2.1	checkNull(String key)	134
6.39.2.2	checkNullLong(String key)	134
6.39.2.3	getText()	135
6.39.3	Member Data Documentation	135

6.39.3.1	coordinates	135
6.39.3.2	created_at	135
6.39.3.3	favorite_count	135
6.39.3.4	geo	135
6.39.3.5	id	135
6.39.3.6	id_str	135
6.39.3.7	IF_NULL_INT	135
6.39.3.8	IF_NULL_LONG	135
6.39.3.9	IF_NULL_STR	135
6.39.3.10	in_reply_to_screen_name	135
6.39.3.11	in_reply_to_status_id	136
6.39.3.12	in_reply_to_status_id_str	136
6.39.3.13	in_reply_to_user_id	136
6.39.3.14	in_reply_to_user_id_str	136
6.39.3.15	lang	136
6.39.3.16	node	136
6.39.3.17	place	136
6.39.3.18	quoted_status	136
6.39.3.19	quoted_status_id	136
6.39.3.20	retweet_count	136
6.39.3.21	source	136
6.39.3.22	text	136
6.39.3.23	timestamp_ms	136
6.39.3.24	user	136
6.40	se2xb3.data.processing.TweetProcessor Class Reference	137
6.40.1	Detailed Description	139
6.40.2	Constructor & Destructor Documentation	139
6.40.2.1	TweetProcessor()	139
6.40.3	Member Function Documentation	139
6.40.3.1	deserializeTweet(String tweetStr)	139

6.40.3.2	getAllMentions()	140
6.40.3.3	getHashtags()	140
6.40.3.4	getUsers()	140
6.40.3.5	graphWord(String word, Tweet tweet)	140
6.40.3.6	jsonStringToTree(String jsonStr)	141
6.40.3.7	jsonToTweet(String s)	141
6.40.3.8	process(String tweetStr)	141
6.40.3.9	processTweetText(Tweet tweet)	142
6.40.3.10	resetData()	142
6.40.4	Member Data Documentation	142
6.40.4.1	graph	142
6.40.4.2	objectMapper	142
6.41	se2xb3.injection.TwitterAppInjector Class Reference	143
6.41.1	Detailed Description	144
6.41.2	Constructor & Destructor Documentation	144
6.41.2.1	TwitterAppInjector()	144
6.41.3	Member Function Documentation	144
6.41.3.1	configure()	144
6.42	se2xb3.bootstrap.TwitterProcessorApp Class Reference	145
6.42.1	Detailed Description	145
6.42.2	Constructor & Destructor Documentation	146
6.42.2.1	TwitterProcessorApp()	146
6.42.3	Member Function Documentation	146
6.42.3.1	main(String[] args)	146
6.42.4	Member Data Documentation	147
6.42.4.1	app	147
6.43	se2xb3.data.models.User Class Reference	147
6.43.1	Detailed Description	149
6.43.2	Constructor & Destructor Documentation	149
6.43.2.1	User()	149

6.43.2.2	User(JsonNode n)	149
6.43.3	Member Data Documentation	149
6.43.3.1	contributors_enabled	149
6.43.3.2	created_at	149
6.43.3.3	default_profile	149
6.43.3.4	default_profile_image	149
6.43.3.5	description	149
6.43.3.6	favourites_count	150
6.43.3.7	follow_request_sent	150
6.43.3.8	followers_count	150
6.43.3.9	following	150
6.43.3.10	friends_count	150
6.43.3.11	geo_enabled	150
6.43.3.12	id	150
6.43.3.13	id_str	150
6.43.3.14	is_translator	150
6.43.3.15	lang	150
6.43.3.16	listed_count	150
6.43.3.17	location	150
6.43.3.18	name	150
6.43.3.19	notifications	150
6.43.3.20	profile_background_color	150
6.43.3.21	profile_background_image_url	150
6.43.3.22	profile_background_image_url_https	150
6.43.3.23	profile_background_tile	150
6.43.3.24	profile_banner_url	150
6.43.3.25	profile_image_url	150
6.43.3.26	profile_image_url_https	150
6.43.3.27	profile_link_color	150
6.43.3.28	profile_sidebar_border_color	150

6.43.3.29 profile_sidebar_fill_color	151
6.43.3.30 profile_text_color	151
6.43.3.31 profile_use_background_image	151
6.43.3.32 screen_name	151
6.43.3.33 statuses_count	151
6.43.3.34 time_zone	151
6.43.3.35 url	151
6.43.3.36 utc_offset	151
6.43.3.37 verified	151
6.44 se2xb3.io.web.WebServer Class Reference	152
6.44.1 Detailed Description	153
6.44.2 Constructor & Destructor Documentation	154
6.44.2.1 WebServer()	154
6.44.2.2 WebServer(IOController io)	154
6.44.3 Member Function Documentation	154
6.44.3.1 run()	154
6.44.3.2 shutdown()	155
6.44.4 Member Data Documentation	155
6.44.4.1 _CLASS_NAME	155
6.44.4.2 ioController	155
6.44.4.3 log	155
6.44.4.4 requestCount	155
6.45 se2xb3.data.algorithms.WordGraph Class Reference	156
6.45.1 Detailed Description	157
6.45.2 Constructor & Destructor Documentation	157
6.45.2.1 WordGraph()	157
6.45.3 Member Function Documentation	157
6.45.3.1 addWord(String word, Tweet tweet)	157
6.45.3.2 eq(String a)	158
6.45.3.3 eq(String a, String b)	158

6.45.3.4	getAllMentions()	158
6.45.3.5	getHashtags()	158
6.45.3.6	getTrends()	159
6.45.3.7	getUsers()	159
6.45.3.8	isBlacklisted(String s)	159
6.45.3.9	max()	159
6.45.3.10	resetData()	160
6.45.3.11	size()	160
6.45.4	Member Data Documentation	160
6.45.4.1	sortedMap	160
6.45.4.2	w	160
6.45.4.3	words	160
6.46	se2xb3.data.algorithms.WordNode Class Reference	160
6.46.1	Detailed Description	162
6.46.2	Constructor & Destructor Documentation	162
6.46.2.1	WordNode(String wordId, Tweet tweet)	162
6.46.3	Member Function Documentation	162
6.46.3.1	add(Tweet tweet)	162
6.46.3.2	compareTo(WordNode other)	163
6.46.3.3	size()	163
6.46.3.4	toString()	163
6.46.4	Member Data Documentation	163
6.46.4.1	id	163
6.46.4.2	list	163

7 File Documentation	165
7.1 AppConfig.java File Reference	165
7.2 AppController.java File Reference	165
7.3 AsyncMessageLooper.java File Reference	165
7.4 Constants.java File Reference	166
7.5 Controller.java File Reference	166
7.6 CORSHandler.java File Reference	166
7.7 DataController.java File Reference	166
7.8 DataHandler.java File Reference	167
7.9 DataSource.java File Reference	167
7.10 EventBus.java File Reference	167
7.11 FileDataSource.java File Reference	167
7.12 FileDataSourceTest.java File Reference	168
7.13 FileFinder.java File Reference	168
7.14 GraphStrategy.java File Reference	168
7.15 IDataSource.java File Reference	168
7.16 IMessageQueue.java File Reference	169
7.17 IMessageReceiver.java File Reference	169
7.18 IOController.java File Reference	169
7.19 IProcessor.java File Reference	169
7.20 ISort.java File Reference	170
7.21 JsonParser.java File Reference	170
7.22 MaxPQ.java File Reference	170
7.23 Merge.java File Reference	170
7.24 MergeBU.java File Reference	171
7.25 MergeSortStrategy.java File Reference	171
7.26 MergeX.java File Reference	171
7.27 MessageHandler.java File Reference	171
7.28 MinPQ.java File Reference	172
7.29 Place.java File Reference	172
7.30 SearchStrategy.java File Reference	172
7.31 SortStrategy.java File Reference	172
7.32 TestDataSource.java File Reference	173
7.33 TrendHandler.java File Reference	173
7.34 Tweet.java File Reference	173
7.35 TweetProcessor.java File Reference	173
7.36 TwitterAppInjector.java File Reference	174
7.37 TwitterProcessorApp.java File Reference	174
7.38 User.java File Reference	174
7.39 WebServer.java File Reference	174
7.40 WordGraph.java File Reference	175
7.41 WordNode.java File Reference	175

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

se2xb3	9
se2xb3.bootstrap	9
se2xb3.config	9
se2xb3.control	9
se2xb3.data	10
se2xb3.data.algorithms	10
se2xb3.data.algorithms.sort	10
se2xb3.data.algorithms.sort.textbook	10
se2xb3.data.models	11
se2xb3.data.processing	11
se2xb3.injection	11
se2xb3.io	11
se2xb3.io.files	11
se2xb3.io.source	12
se2xb3.io.web	12
se2xb3.io.web.handlers	12
se2xb3.tests	12

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

se2xb3.config.AppConfig	13
se2xb3.data.models.BoundingBox	22
Comparable	
se2xb3.data.algorithms.WordNode	160
se2xb3.config.Constants	23
se2xb3.data.DataHandler	37
se2xb3.io.source.DataSource	43
se2xb3.io.source.FileDataSource	50
se2xb3.io.source.FileDataSource	50
se2xb3.io.web.handlers.TrendHandler	124
se2xb3.io.web.WebServer	152
se2xb3.control.Controller	26
se2xb3.control.AppController	14
se2xb3.data.DataController	31
se2xb3.io.IOController	68
se2xb3.control.EventBus	48
se2xb3.tests.FileDataSourceTest	54
se2xb3.io.files.FileFinder	55
se2xb3.data.algorithms.GraphStrategy	57
se2xb3.io.source.IDataSource	63
se2xb3.io.IMessageQueue< M >	64
se2xb3.io.AsyncMessageLooper< M >	18
se2xb3.io.IMessageQueue< String >	64
se2xb3.io.IMessageReceiver< M >	66
se2xb3.io.IMessageReceiver< String >	66
se2xb3.io.MessageHandler	103
se2xb3.data.processing.IProcessor< T >	73
se2xb3.data.processing.IProcessor< String >	73
se2xb3.data.processing.TweetProcessor	137
se2xb3.data.algorithms.sort.ISort	75
se2xb3.data.algorithms.sort.MergeSortStrategy	96
Iterable	
se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >	81

se2xb3.data.algorithms.sort.textbook.MinPQ< Key >	107
se2xb3.io.web.JsonParser	76
se2xb3.data.algorithms.sort.textbook.Merge	89
se2xb3.data.algorithms.sort.textbook.MergeBU	93
se2xb3.data.algorithms.sort.textbook.MergeX	98
se2xb3.data.models.Place	115
Runnable	
se2xb3.io.AsyncMessageLooper< M >	18
se2xb3.io.source.DataSource	43
se2xb3.io.web.WebServer	152
se2xb3.data.algorithms.SearchStrategy	117
se2xb3.data.algorithms.SortStrategy	118
se2xb3.tests.TestDataSource	120
se2xb3.io.web.Trend	122
se2xb3.data.models.Tweet	130
se2xb3.data.models.TweetOld	132
se2xb3.bootstrap.TwitterProcessorApp	145
se2xb3.data.models.User	147
se2xb3.data.algorithms.WordGraph	156
AbstractModule	
se2xb3.injection.TwitterAppInjector	143
Handler	
se2xb3.io.web.handlers.CORSHandler	28
se2xb3.io.web.handlers.TrendHandler	124
Iterator	
se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator	61
se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator	59

Chapter 3

Class Index

3.1 Class List

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

se2xb3.config.AppConfig	13
se2xb3.control.AppController	14
se2xb3.io.AsyncMessageLooper< M >	18
se2xb3.data.models.BoundingBox	22
se2xb3.config.Constants	23
se2xb3.control.Controller	26
se2xb3.io.web.handlers.CORSHandler	28
se2xb3.data.DataController	31
se2xb3.data.DataHandler	37
se2xb3.io.source.DataSource	43
se2xb3.control.EventBus	48
se2xb3.io.source.FileDataSource	50
se2xb3.tests.FileDataSourceTest	54
se2xb3.io.files.FileFinder	55
se2xb3.data.algorithms.GraphStrategy	57
se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator	59
se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.Heaplterator	61
se2xb3.io.source.IDataSource	63
se2xb3.io.IMessageQueue< M >	64
se2xb3.io.IMessageReceiver< M >	66
se2xb3.io.IOController	68
se2xb3.data.processing.IProcessor< T >	73
se2xb3.data.algorithms.sort.ISort	75
se2xb3.io.web.JsonParser	76
se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >	81
se2xb3.data.algorithms.sort.textbook.Merge	89
se2xb3.data.algorithms.sort.textbook.MergeBU	93
se2xb3.data.algorithms.sort.MergeSortStrategy	96
se2xb3.data.algorithms.sort.textbook.MergeX	98
se2xb3.io.MessageHandler	103
se2xb3.data.algorithms.sort.textbook.MinPQ< Key >	107
se2xb3.data.models.Place	115
se2xb3.data.algorithms.SearchStrategy	117
se2xb3.data.algorithms.SortStrategy	118
se2xb3.tests.TestDataSource	120

se2xb3.io.web.Trend	122
se2xb3.io.web.handlers.TrendHandler	124
se2xb3.data.models.Tweet	130
se2xb3.data.models.TweetOld	132
se2xb3.data.processing.TweetProcessor	137
se2xb3.injection.TwitterAppInjector	143
se2xb3.bootstrap.TwitterProcessorApp	145
se2xb3.data.models.User	147
se2xb3.io.web.WebServer	152
se2xb3.data.algorithms.WordGraph	156
se2xb3.data.algorithms.WordNode	160

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

AppConfig.java	165
AppController.java	165
AsyncMessageLooper.java	165
Constants.java	166
Controller.java	166
CORSHandler.java	166
DataController.java	166
DataHandler.java	167
DataSource.java	167
EventBus.java	167
FileDataSource.java	167
FileDataSourceTest.java	168
FileFinder.java	168
GraphStrategy.java	168
IDataSource.java	168
IMessageQueue.java	169
IMessageReceiver.java	169
IOController.java	169
IProcessor.java	169
ISort.java	170
JsonParser.java	170
MaxPQ.java	170
Merge.java	170
MergeBU.java	171
MergeSortStrategy.java	171
MergeX.java	171
MessageHandler.java	171
MinPQ.java	172
Place.java	172
SearchStrategy.java	172
SortStrategy.java	172
TestDataSource.java	173
TrendHandler.java	173
Tweet.java	173
TweetProcessor.java	173

TwitterAppInjector.java	174
TwitterProcessorApp.java	174
User.java	174
WebServer.java	174
WordGraph.java	175
WordNode.java	175

Chapter 5

Namespace Documentation

5.1 Package se2xb3

Packages

- package [bootstrap](#)
- package [config](#)
- package [control](#)
- package [data](#)
- package [injection](#)
- package [io](#)
- package [tests](#)

5.2 Package se2xb3.bootstrap

Classes

- class [TwitterProcessorApp](#)

5.3 Package se2xb3.config

Classes

- interface [AppConfig](#)
- interface [Constants](#)

5.4 Package se2xb3.control

Classes

- class [AppController](#)
- class [Controller](#)
- class [EventBus](#)

5.5 Package se2xb3.data

Packages

- package [algorithms](#)
- package [models](#)
- package [processing](#)

Classes

- class [DataController](#)
- class [DataHandler](#)

5.6 Package se2xb3.data.algorithms

Packages

- package [sort](#)

Classes

- class [GraphStrategy](#)
- class [SearchStrategy](#)
- class [SortStrategy](#)
- class [WordGraph](#)
- class [WordNode](#)

5.7 Package se2xb3.data.algorithms.sort

Packages

- package [textbook](#)

Classes

- interface [ISort](#)
- class [MergeSortStrategy](#)

5.8 Package se2xb3.data.algorithms.sort.textbook

Classes

- class [MaxPQ](#)
- class [Merge](#)
- class [MergeBU](#)
- class [MergeX](#)
- class [MinPQ](#)

5.9 Package se2xb3.data.models

Classes

- class [BoundingBox](#)
- class [Place](#)
- class [Tweet](#)
- class [TweetOld](#)
- class [User](#)

5.10 Package se2xb3.data.processing

Classes

- interface [IProcessor](#)
- class [TweetProcessor](#)

5.11 Package se2xb3.injection

Classes

- class [TwitterAppInjector](#)

5.12 Package se2xb3.io

Packages

- package [files](#)
- package [source](#)
- package [web](#)

Classes

- class [AsyncMessageLooper](#)
- interface [IMessageQueue](#)
- interface [IMessageReceiver](#)
- class [IOController](#)
- class [MessageHandler](#)

5.13 Package se2xb3.io.files

Classes

- class [FileFinder](#)

5.14 Package se2xb3.io.source

Classes

- class [DataSource](#)
- class [FileDataSource](#)
- interface [IDataSource](#)

5.15 Package se2xb3.io.web

Packages

- package [handlers](#)

Classes

- class [JsonParser](#)
- class [Trend](#)
- class [WebServer](#)

5.16 Package se2xb3.io.web.handlers

Classes

- class [CORSHandler](#)
- class [TrendHandler](#)

5.17 Package se2xb3.tests

Classes

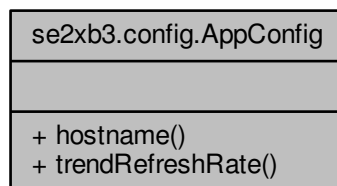
- class [FileDataSourceTest](#)
- class [TestDataSource](#)

Chapter 6

Class Documentation

6.1 se2xb3.config.AppConfig Interface Reference

Collaboration diagram for se2xb3.config.AppConfig:



Public Member Functions

- String `hostname ()`
- int `trendRefreshRate ()`

6.1.1 Detailed Description

Author

Dawson

Version

1.0

Since

4/3/2017

6.1.2 Member Function Documentation

6.1.2.1 String se2xb3.config.AppConfig.hostname ()

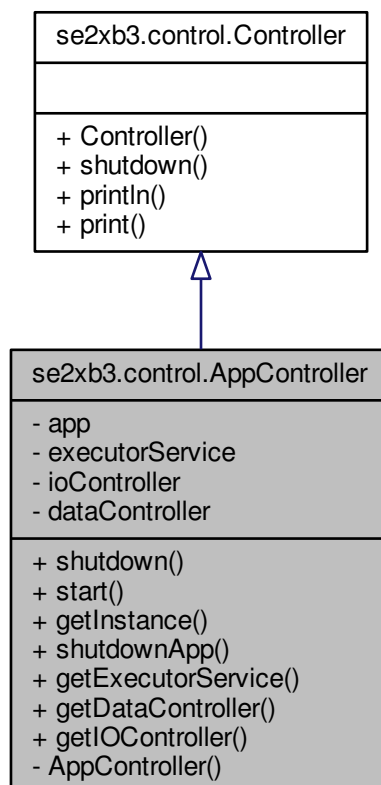
6.1.2.2 int se2xb3.config.AppConfig.trendRefreshRate ()

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

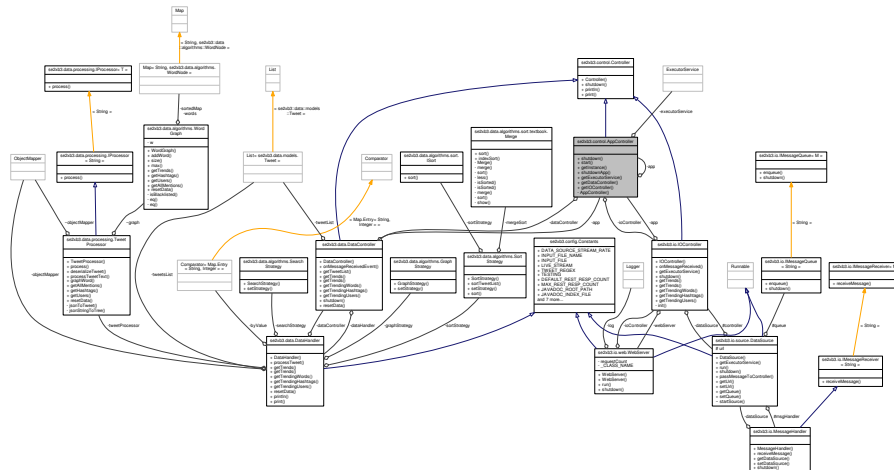
- [AppConfig.java](#)

6.2 se2xb3.control.AppController Class Reference

Inheritance diagram for se2xb3.control.AppController:



Collaboration diagram for se2xb3.control.AppController:



Public Member Functions

- void [shutdown](#) ()

Static Public Member Functions

- static [AppController](#) [start](#) ()
- static [AppController](#) [getInstance](#) ()
- static void [shutdownApp](#) ()
- static [ExecutorService](#) [getExecutorService](#) ()
- static [DataController](#) [getDataController](#) ()
- static [IOController](#) [getIOController](#) ()

Private Member Functions

- [AppController](#) ()

Static Private Attributes

- static [AppController](#) [app](#) = [getInstance](#)()
- static volatile [ExecutorService](#) [executorService](#)
- static [IOController](#) [ioController](#)
- static [DataController](#) [dataController](#)

6.2.1 Detailed Description

Main app controller.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `se2xb3.control.AppController.AppController ()` [private]

Private constructor for singleton instance.

```

27         {
28             //app = this;
29             executorService = Executors.newFixedThreadPool(5);
30             ioController = new IOController(this);
31             dataController = new DataController(this);
32     }
```

6.2.3 Member Function Documentation

6.2.3.1 `static DataController se2xb3.control.AppController.getDataController ()` [static]

Get instance the data controller instance.

Returns

a reference to the IOController

```

98         {
99             return getInstance().dataController;
100     }
```

6.2.3.2 `static ExecutorService se2xb3.control.AppController.getExecutorService ()` [static]

Return the executorService instance

Returns

executorService the executor service

```

89         {
90             return executorService;
91     }
```

6.2.3.3 `static AppController se2xb3.control.AppController.getInstance ()` [static]

Singleton getter.

Returns

a reference to the app controller

```

49         {
50             if (app == null) app = new AppController();
51             return app;
52     }
```

6.2.3.4 static IOController se2xb3.control.AppController.getIOController () [static]

Get instance the data controller instance.

Returns

a reference to the IOController

```

107                                     {
108         return ioController;
109     }

```

6.2.3.5 void se2xb3.control.AppController.shutdown ()

Shutdown thread pool.

```

79                                     {
80         dataController.shutdown();
81         ioController.shutdown();
82     }

```

6.2.3.6 static void se2xb3.control.AppController.shutdownApp () [static]

Shutdown thread pool.

```

58                                     {
59         app.shutdown();
60         try {
61             System.out.println("attempt to shutdown executor");
62             TimeUnit.SECONDS.sleep(2);
63             executorService.shutdown();
64             executorService.awaitTermination(10, TimeUnit.SECONDS);
65         } catch (InterruptedException e) {
66             System.err.println("tasks interrupted");
67         } finally {
68             if (!executorService.isTerminated()) {
69                 System.err.println("cancel non-finished tasks");
70             }
71             executorService.shutdownNow();
72             System.out.println("shutdown finished");
73         }
74     }

```

6.2.3.7 static AppController se2xb3.control.AppController.start () [static]

Start app.

Returns

a reference to the app controller

```

39                                     {
40 //         if(app == null) app = new AppController();
41         return getInstance();
42     }

```

6.2.4 Member Data Documentation

6.2.4.1 **AppController** `se2xb3.control.AppController.app = getInstance()` `[static]`, `[private]`

6.2.4.2 **DataController** `se2xb3.control.AppController.dataController` `[static]`, `[private]`

6.2.4.3 **volatile ExecutorService** `se2xb3.control.AppController.executorService` `[static]`, `[private]`

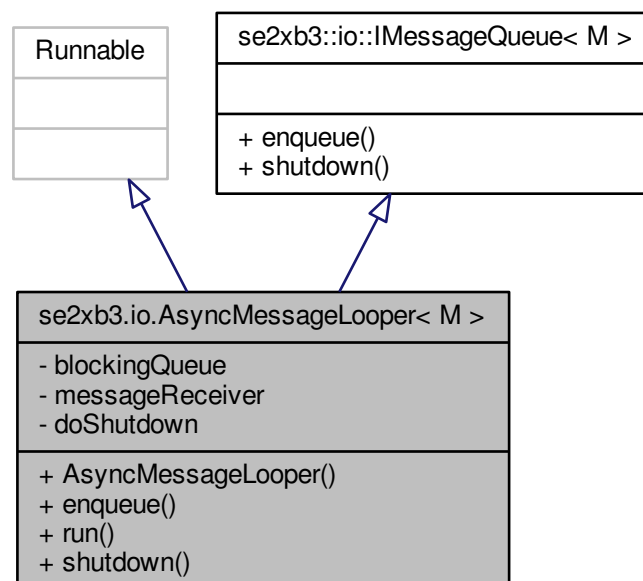
6.2.4.4 **IOController** `se2xb3.control.AppController.ioController` `[static]`, `[private]`

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

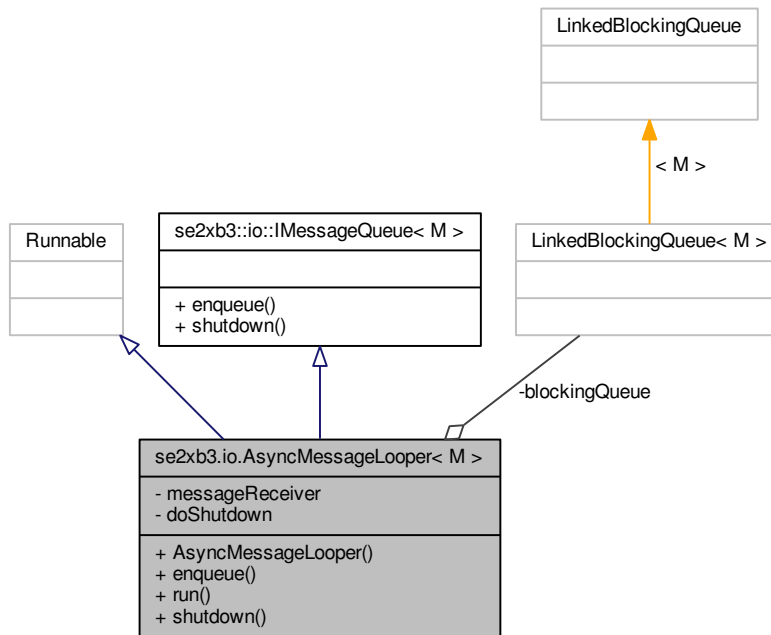
- [AppController.java](#)

6.3 `se2xb3.io.AsyncMessageLooper< M >` Class Template Reference

Inheritance diagram for `se2xb3.io.AsyncMessageLooper< M >`:



Collaboration diagram for se2xb3.io.AsyncMessageLooper< M >:



Public Member Functions

- [AsyncMessageLooper](#) ([IMessageReceiver< M >](#) receiver, `ExecutorService` executorService)
- void [enqueue](#) (M msg)
- void [run](#) ()
- void [shutdown](#) ()

Private Attributes

- volatile `LinkedBlockingQueue< M >` [blockingQueue](#) = new `LinkedBlockingQueue<M>()`
- [IMessageReceiver< M >](#) [messageReceiver](#)
- boolean [doShutdown](#) = false

6.3.1 Detailed Description

A class that asynchronously decouples incoming messages from the rest of the system by running the buffer in separate thread. It continuously tries to take messages from a blocking queue.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `se2xb3.io.AsyncMessageLooper< M >.AsyncMessageLooper (IMessageReceiver< M > receiver, ExecutorService executorService)`

Constructor that takes a message receiver

Parameters

<i>receiver</i>	a message receiver
<i>executorService</i>	an executor

```

28         {
29             messageReceiver = receiver;
30             executorService.execute(this);
31         }

```

6.3.3 Member Function Documentation

6.3.3.1 `void se2xb3.io.AsyncMessageLooper< M >.enqueue (M msg)`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, throwing an `IllegalStateException` if no space is currently available.

Parameters

<i>msg</i>	a message
------------	-----------

Implements `se2xb3.io.IMessageQueue< M >`.

```

40         {
41             blockingQueue.add(msg);
42         }

```

6.3.3.2 `void se2xb3.io.AsyncMessageLooper< M >.run ()`

Start message loop thread.

```

47         {
48             while (!doShutdown) {
49
50                 M msg;
51                 try {
52                     // Loop forever, taking messages from the queue when available.
53                     msg = blockingQueue.take();
54
55                     //if(msg instanceof String) System.out.println(msg);
56                     messageReceiver.receiveMessage(msg);
57                 } catch (InterruptedException e) {
58                     e.printStackTrace();
59                 }
60
61             }
62         }

```

6.3.3.3 `void se2xb3.io.AsyncMessageLooper< M >.shutdown ()`

Shutdown ththread by breaking out of the message loop.

Implements [se2xb3.io.IMessageQueue< M >](#).

```
68         {
69             doShutdown = true;
70             blockingQueue.add( (M) new Object() );
71         }
72     }
```

6.3.4 Member Data Documentation

6.3.4.1 `volatile LinkedBlockingQueue<M> se2xb3.io.AsyncMessageLooper< M >.blockingQueue = new LinkedBlockingQueue<M>() [private]`

6.3.4.2 `boolean se2xb3.io.AsyncMessageLooper< M >.doShutdown = false [private]`

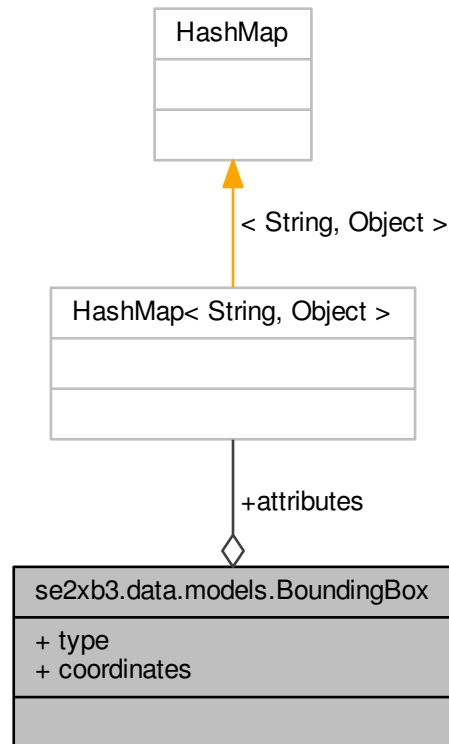
6.3.4.3 `IMessageReceiver<M> se2xb3.io.AsyncMessageLooper< M >.messageReceiver [private]`

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

- [AsyncMessageLooper.java](#)

6.4 se2xb3.data.models.BoundingBox Class Reference

Collaboration diagram for se2xb3.data.models.BoundingBox:



Public Attributes

- String [type](#)
- int[][] [coordinates](#)
- HashMap<String, Object> [attributes](#)

6.4.1 Member Data Documentation

6.4.1.1 HashMap<String, Object> se2xb3.data.models.BoundingBox.attributes

6.4.1.2 int [][] se2xb3.data.models.BoundingBox.coordinates

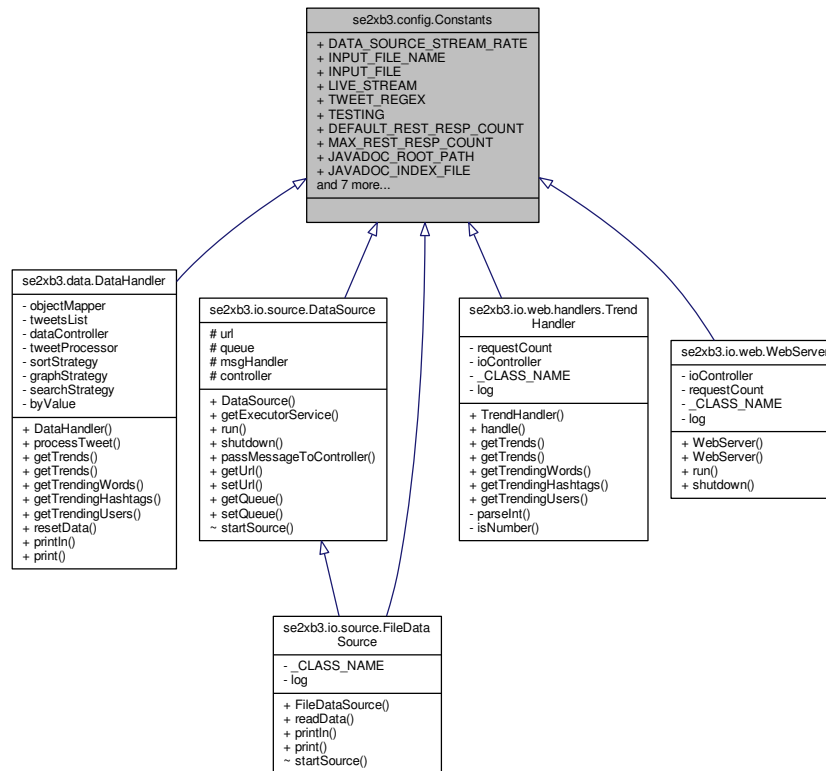
6.4.1.3 String se2xb3.data.models.BoundingBox.type

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

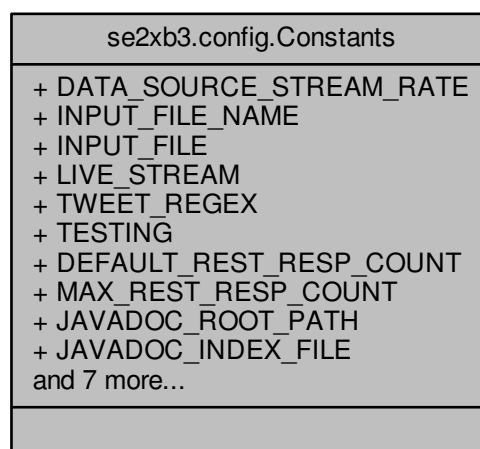
- [Place.java](#)

6.5 se2xb3.config.Constants Interface Reference

Inheritance diagram for se2xb3.config.Constants:



Collaboration diagram for se2xb3.config.Constants:



Public Attributes

- int `DATA_SOURCE_STREAM_RATE` = 500
- String `INPUT_FILE_NAME` = "tweets-simple-min-100k.txt"
- String `INPUT_FILE` = "data/tweets-simple-min-100k.txt"
- boolean `LIVE_STREAM` = false
- String `TWEET_REGEX` = "([@|#]*\\w)+"
- boolean `TESTING` = true
- int `DEFAULT_REST_RESP_COUNT` = 10
- int `MAX_REST_RESP_COUNT` = 500
- String `JAVADOC_ROOT_PATH` = "docs"
- String `JAVADOC_INDEX_FILE` = `JAVADOC_ROOT_PATH` + "/" + "index.html"
- String `WEB_APP_ROOT_PATH` = ""
- String `WEB_APP_ROOT_FILE` = "index.html"
- String `REST_TREND_RESOURCE_PATH_REGEX` = "trends/:type?/:count?"
- String `REST_RESOURCE_TRENDS_ALL` = "all"
- String `REST_RESOURCE_TRENDS_HASHTAGS` = "hashtags"
- String `REST_RESOURCE_TRENDS_USERS` = "users"
- String `REST_REQUEST_NO_MATCH` = "NO REST RESOURCE MATCH"

6.5.1 Detailed Description

System constants

Author

Dawson Myers

Version

1.0

Since

3/10/2017

6.5.2 Member Data Documentation

6.5.2.1 `int se2xb3.config.Constants.DATA_SOURCE_STREAM_RATE = 500`

The rate at which the simulated data source inserts messages into system. The default value is { `DATA_SOURCE_STREAM_RATE`}.

6.5.2.2 `int se2xb3.config.Constants.DEFAULT_REST_RESP_COUNT = 10`

Default count of items to respond to a REST request. The default value is { `DEFAULT_REST_RESP_COUNT`}.

6.5.2.3 String se2xb3.config.Constants.INPUT_FILE = "data/tweets-simple-min-100k.txt"

The default input file name is set to { INPUT_FILE}.

6.5.2.4 String se2xb3.config.Constants.INPUT_FILE_NAME = "tweets-simple-min-100k.txt"

The default input file name is set to { INPUT_FILE_NAME}.

6.5.2.5 String se2xb3.config.Constants.JAVADOC_INDEX_FILE = JAVADOC_ROOT_PATH + "/" + "index.html"

Javadoc index.html path. The default value is { JAVADOC_INDEX_FILE}.

6.5.2.6 String se2xb3.config.Constants.JAVADOC_ROOT_PATH = "docs"

Javadoc root url path (i.e. the root path for DawsonMyers.ca/docs is "docs"). The default value is { JAVADOC_ROOT_PATH}.

6.5.2.7 boolean se2xb3.config.Constants.LIVE_STREAM = false

True if live streaming is enabled. The default value is { LIVE_STREAM}.

6.5.2.8 int se2xb3.config.Constants.MAX_REST_RESP_COUNT = 500

Maximum count of items that a REST request can ask for (i.e. /trends/users/500). The default value is { MAX_REST_RESP_COUNT}. This is to stop people from requesting something like this: trends/1000000000000.

6.5.2.9 String se2xb3.config.Constants.REST_REQUEST_NO_MATCH = "NO REST RESOURCE MATCH"

REST response message when the request doesn't match any resources. The default value is { REST_REQUEST_NO_MATCH}.

6.5.2.10 String se2xb3.config.Constants.REST_RESOURCE_TRENDS_ALL = "all"

All word mentions REST resource name. The default value is { REST_RESOURCE_TRENDS_ALL}.

6.5.2.11 String se2xb3.config.Constants.REST_RESOURCE_TRENDS_HASHTAGS = "hashtags"

Hashtag mentions REST resource name. The default value is { REST_RESOURCE_TRENDS_HASHTAGS}.

6.5.2.12 String se2xb3.config.Constants.REST_RESOURCE_TRENDS_USERS = "users"

User mentions REST resource name. The default value is { REST_RESOURCE_TRENDS_USERS}.

6.5.2.13 String `se2xb3.config.Constants.REST_TREND_RESOURCE_PATH_REGEX` = "trends/:type?/:count?"

The regex for the trend REST resource. The default value is { `REST_TREND_RESOURCE_PATH_REGEX`}.

6.5.2.14 boolean `se2xb3.config.Constants.TESTING` = true

6.5.2.15 String `se2xb3.config.Constants.TWEET_REGEX` = "([@|#]*\\w)+"

The regular expression to parse words, hashtags (words starting with '#'), and users (words starting with '@'). The default value is { `TWEET_REGEX`}.

6.5.2.16 String `se2xb3.config.Constants.WEB_APP_ROOT_FILE` = "index.html"

Web app index.html path. This is the file that bootstraps the web app. The default value is { `WEB_APP_ROOT_FILE`}.

6.5.2.17 String `se2xb3.config.Constants.WEB_APP_ROOT_PATH` = ""

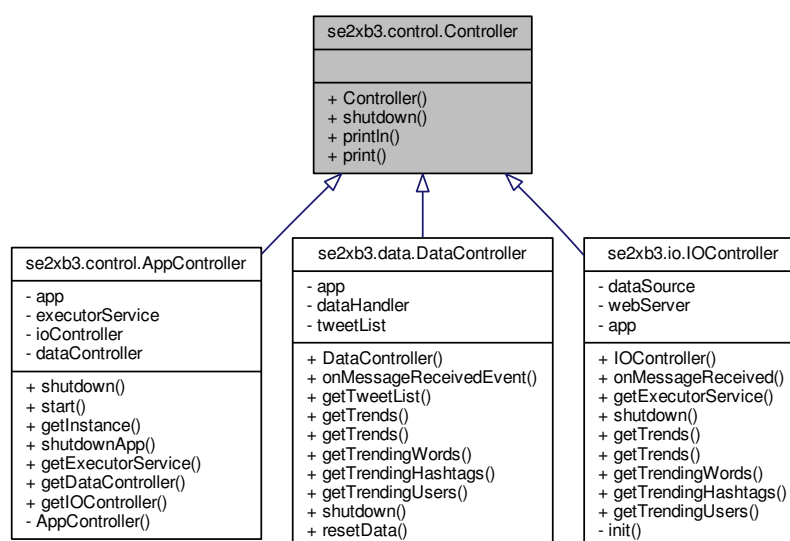
Web app root url path (i.e. the root path for DawsonMyers.ca/ is "" or "/"). This is the url that bootstraps the web app.

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

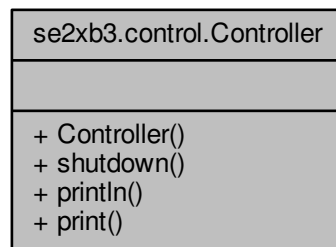
- [Constants.java](#)

6.6 se2xb3.control.Controller Class Reference

Inheritance diagram for `se2xb3.control.Controller`:



Collaboration diagram for se2xb3.control.Controller:



Public Member Functions

- [Controller](#) ()
- abstract void [shutdown](#) ()

Static Public Member Functions

- static void [println](#) (String s)
- static void [print](#) (String s)

6.6.1 Detailed Description

Abstract base class for controllers

Author

Dawson Myers

Version

1.0

Since

3/12/2017

6.6.2 Constructor & Destructor Documentation

6.6.2.1 se2xb3.control.Controller.Controller ()

11 {}

6.6.3 Member Function Documentation

6.6.3.1 static void se2xb3.control.Controller.print (String s) [static]

```
23         {  
24     System.out.print(s);  
25     }
```

6.6.3.2 static void se2xb3.control.Controller.println (String s) [static]

```
19         {  
20     System.out.println(s);  
21     }
```

6.6.3.3 abstract void se2xb3.control.Controller.shutdown () [abstract]

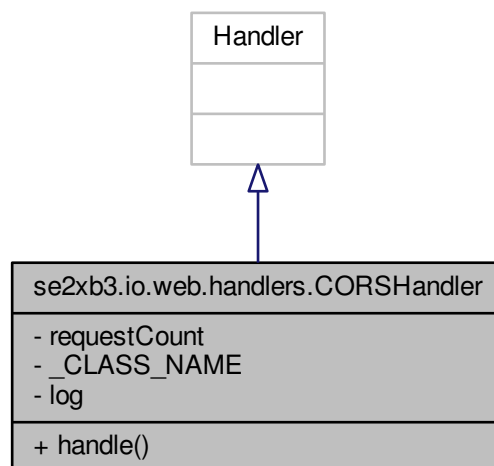
Shutdown blocking queue.

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

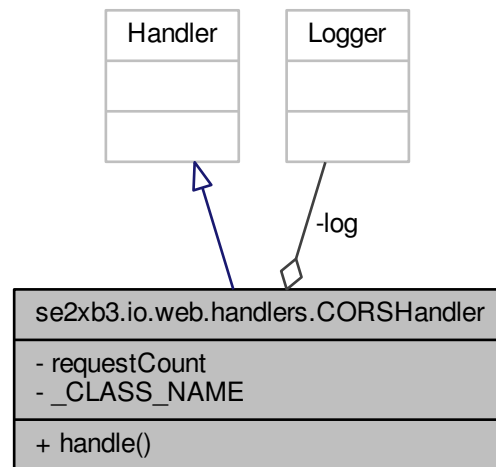
- [Controller.java](#)

6.7 se2xb3.io.web.handlers.CORSHandler Class Reference

Inheritance diagram for se2xb3.io.web.handlers.CORSHandler:



Collaboration diagram for se2xb3.io.web.handlers.CORSHandler:



Public Member Functions

- void [handle](#) (Context ctx) throws Exception

Private Attributes

- long [requestCount](#) = 0

Static Private Attributes

- static final String [_CLASS_NAME](#) = CORSHandler.class.getSimpleName()
- static final Logger [log](#) = LoggerFactory.getLogger([_CLASS_NAME](#))

6.7.1 Detailed Description

Add CORS headers for cross-site REST requests.

Author

Dawson Myers

Version

1.0

Since

3/27/2017

6.7.2 Member Function Documentation

6.7.2.1 void se2xb3.io.web.handlers.CORSHandler.handle (Context ctx) throws Exception

Handle requests by adding the CORS headers to each request that is processed. Then, continue to the next handler in the chain without sending the response.

Parameters

<i>ctx</i>	the response context
------------	----------------------

Exceptions

<i>Exception</i>	an exception
------------------	--------------

```

28         {
29             MutableHeaders headers = ctx.getResponse().getHeaders();
30             headers.set("Access-Control-Allow-Origin", "*");
31             headers.set("Access-Control-Allow-Headers", "x-requested-with, origin, content-type, " +
32                 "accept");
33             headers.set("Access-Control-Allow-Methods", "GET,PUT,POST,DELETE");
34             requestCount++;
35             if (requestCount % 100 == 0) {
36                 log.info("Request count = " + requestCount);
37             }
38             // pass to next handler
39             ctx.next();
40         }

```

6.7.3 Member Data Documentation

6.7.3.1 final String se2xb3.io.web.handlers.CORSHandler._CLASS_NAME = CORSHandler.class.getSimpleName()
[static], [private]

6.7.3.2 final Logger se2xb3.io.web.handlers.CORSHandler.log = LoggerFactory.getLogger(_CLASS_NAME) [static],
[private]

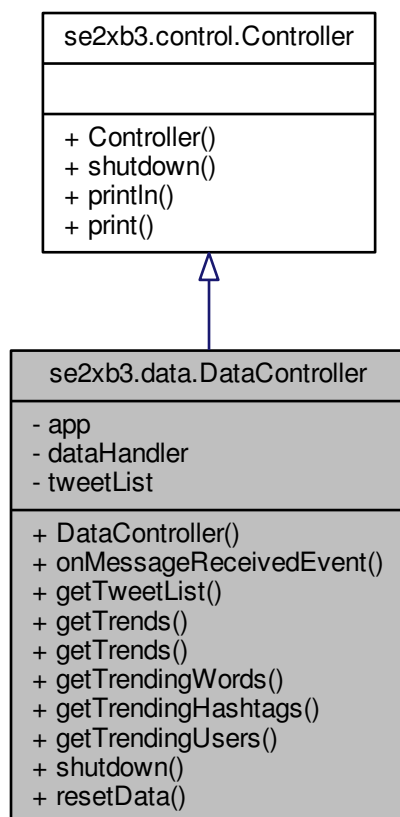
6.7.3.3 long se2xb3.io.web.handlers.CORSHandler.requestCount = 0 [private]

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

- [CORSHandler.java](#)

6.8 se2xb3.data.DataController Class Reference

Inheritance diagram for se2xb3.data.DataController:



Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.8.2 Constructor & Destructor Documentation**6.8.2.1** `se2xb3.data.DataController.DataController (AppController appController)`

Constructor that takes an instance of the AppController.

Parameters

<i>appController</i>	the app controller instance
----------------------	-----------------------------

```

27                                     {
28         app = appController;
29         dataHandler = new DataHandler(this);
30     }
```

6.8.3 Member Function Documentation**6.8.3.1** `Map<String, Integer> se2xb3.data.DataController.getTrendingHashtags (int count)`

Get trending hashtags.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map of the top trending items

```

82                                     {
83         return dataHandler.getTrendingHashtags(count);
84     }
```

6.8.3.2 `Map<String, Integer> se2xb3.data.DataController.getTrendingUsers (int count)`

Get trending users.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map of the top trending items

```

92                                     {
93         return dataHandler.getTrendingUsers(count);
94     }
```

6.8.3.3 Map<String, Integer> se2xb3.data.DataController.getTrendingWords (int *count*)

Get trending words.

Parameters

<i>count</i>	the number of top trending items to get
--------------	---

Returns

a map of the top trending items

```

72                                     {
73         return dataHandler.getTrendingWords(count);
74     }
```

6.8.3.4 List<Map<String, Integer> > se2xb3.data.DataController.getTrends ()

Get list of trends

Returns

a list of maps of the top trending items

```

53                                     {
54         return dataHandler.getTrends();
55     }
```

6.8.3.5 List<Map<String, Integer> > se2xb3.data.DataController.getTrends (int *count*)

Get list of trends

Parameters

<i>count</i>	the number of top trending items to get
--------------	---

Returns

a list of maps of the top trending items

```

62                                     {
63         return dataHandler.getTrends(count);
64     }

```

6.8.3.6 List<Tweet> se2xb3.data.DataController.getTweetList ()

Get list of trends

Returns

a list of maps of the top trending items

```

45                                     {
46         return tweetList;
47     }

```

6.8.3.7 void se2xb3.data.DataController.onMessageReceivedEvent (String msg)

Called whenever a message is received from the IO subsystem.

Parameters

<i>msg</i>	a message string
------------	------------------

```

36                                     {
37     //         System.out.println(msg);
38         dataHandler.processTweet(msg);
39     }

```

6.8.3.8 void se2xb3.data.DataController.resetData ()

```

104                                     {
105         dataHandler.resetData();
106     }

```

6.8.3.9 void se2xb3.data.DataController.shutdown ()

Shutdown.

```

100                                     {
101
102     }

```

6.8.4 Member Data Documentation

6.8.4.1 **AppController** `se2xb3.data.DataController.app` `[private]`

6.8.4.2 **DataHandler** `se2xb3.data.DataController.dataHandler` `[private]`

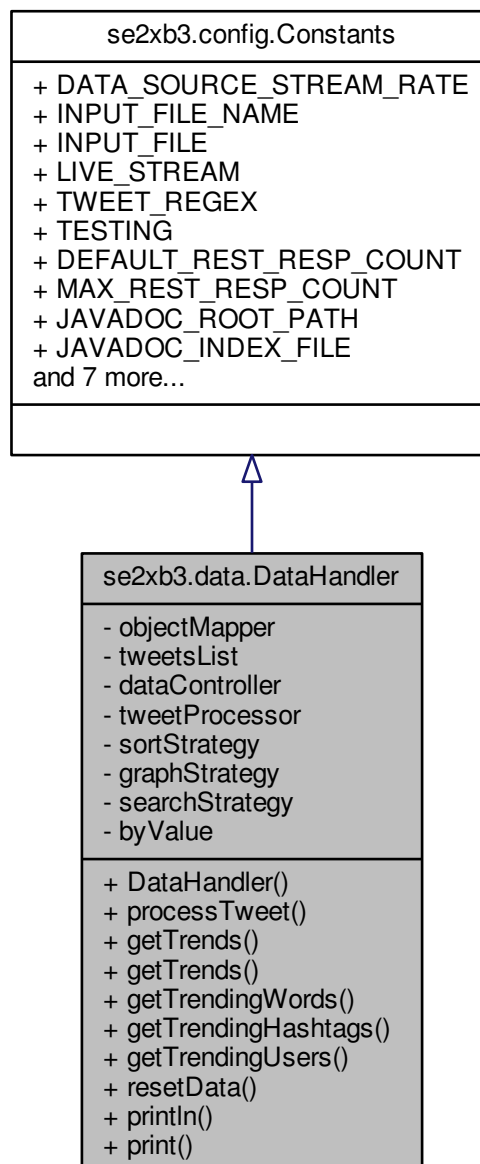
6.8.4.3 **List<Tweet>** `se2xb3.data.DataController.tweetList = new ArrayList<>()` `[private]`

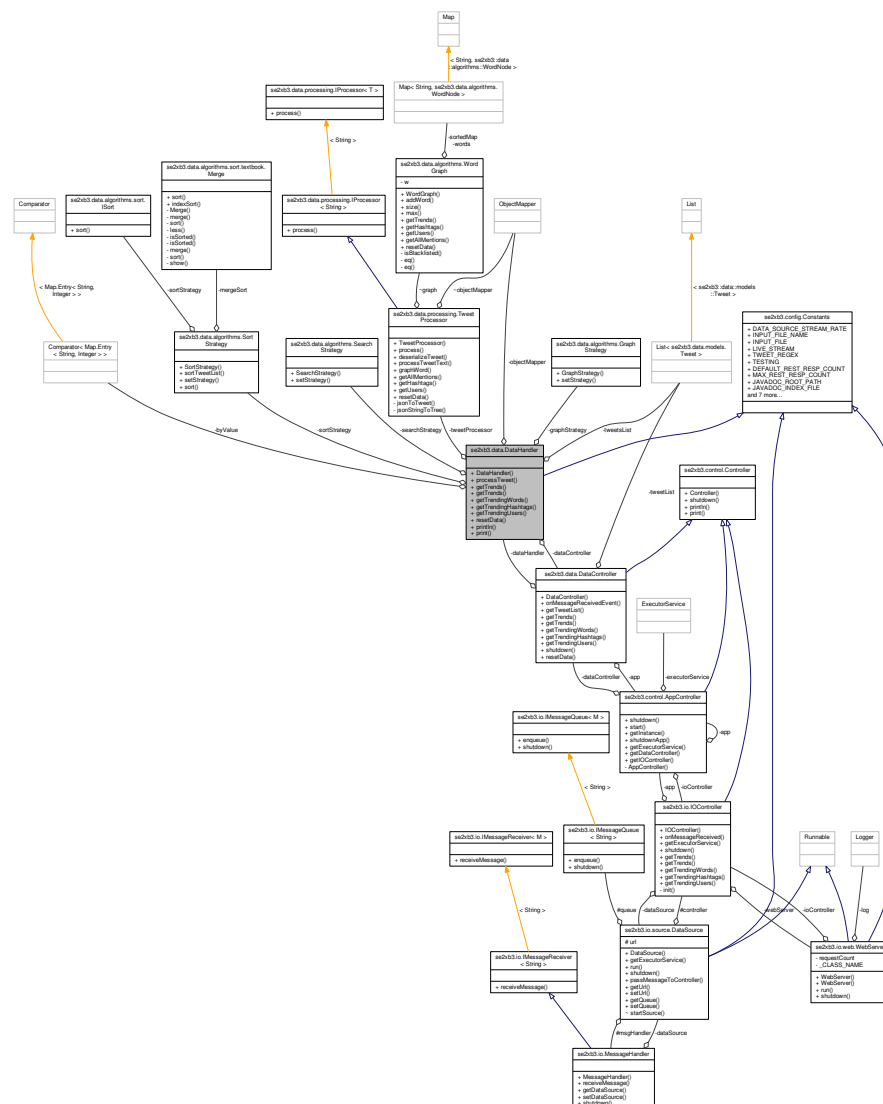
The documentation for this class was generated from the following file:

- [DataController.java](#)

6.9 se2xb3.data.DataHandler Class Reference

Inheritance diagram for se2xb3.data.DataHandler:





Public Member Functions

- **DataHandler** (**DataController** **dataController**)
- void **processTweet** (String tweetStr)
- List< Map< String, Integer > > **getTrends** ()
- List< Map< String, Integer > > **getTrends** (int count)
- Map< String, Integer > **getTrendingWords** (int count)
- Map< String, Integer > **getTrendingHashtags** (int count)
- Map< String, Integer > **getTrendingUsers** (int count)
- void **resetData** ()

Static Public Member Functions

- static void **println** (String s)
- static void **print** (String s)

Private Attributes

- ObjectMapper `objectMapper` = new ObjectMapper()
- List< `Tweet` > `tweetsList` = new ArrayList<>()
- `DataController` `dataController`
- `TweetProcessor` `tweetProcessor` = new `TweetProcessor`()
- `SortStrategy` `sortStrategy` = new `SortStrategy`()
- `GraphStrategy` `graphStrategy` = new `GraphStrategy`()
- `SearchStrategy` `searchStrategy` = new `SearchStrategy`()
- Comparator< Map.Entry< String, Integer > > `byValue`

Additional Inherited Members

6.9.1 Detailed Description

A class to process tweets and use various algorithms to sort, search, and graph the data.

Author

Dawson Myers

Version

1.0

Since

2/23/2017

6.9.2 Constructor & Destructor Documentation

6.9.2.1 se2xb3.data.DataHandler.DataHandler (`DataController` *dataController*)

Constructor

Parameters

<i>dataController</i>	a reference to the data controller
-----------------------	------------------------------------

```

44                                     {
45         this.dataController = dataController;
46     }
```

6.9.3 Member Function Documentation

6.9.3.1 Map<String, Integer> se2xb3.data.DataHandler.getTrendingHashtags (int *count*)

Get trending hashtags.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map containing the top trending items

```

116                                     {
117
118     WordNode[] hashtagMentions = tweetProcessor.getHashtags();
119     sortStrategy.sort(hashtagMentions);
120     Map<String, Integer> m = new LinkedHashMap<>();
121     Arrays.stream(hashtagMentions)
122         .limit(count) // only grab the top n number of items
123         .collect(Collectors.toMap(a -> ((WordNode) a).id, a -> ((WordNode) a).size()))
124         .entrySet().stream().sorted(byValue.reversed()).forEachOrdered(a -> m.put(a.getKey
125             (), a.getValue()));
126     return m;
127 }
```

6.9.3.2 Map<String, Integer> se2xb3.data.DataHandler.getTrendingUsers (int count)

Get trending users.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map containing the top trending items

```

135                                     {
136     WordNode[] userMentions = tweetProcessor.getUsers();
137     sortStrategy.sort(userMentions);
138     Map<String, Integer> m = new LinkedHashMap<>();
139     Arrays.stream(userMentions)
140         .limit(count)
141         .collect(Collectors.toMap(a -> ((WordNode) a).id, a -> ((WordNode) a).size()))
142         .entrySet().stream().sorted(byValue.reversed()).forEachOrdered(a -> m.put(a.getKey
143             (), a.getValue()));
144     return m;
145 }
```

6.9.3.3 Map<String, Integer> se2xb3.data.DataHandler.getTrendingWords (int count)

Get trending words.

Parameters

<i>count</i>	number of top trending items to get
--------------	-------------------------------------

Returns

a map containing the top trending items

```

95                                     {
96     WordNode[] allMentions = tweetProcessor.getAllMentions();
97
98     sortStrategy.sort(allMentions);
99     Map<String, Integer> m = new LinkedHashMap<>();
100
101     Arrays.stream(allMentions)
102         .filter(a -> !a.id.contains("http"))
103         .limit(count)
104         .collect(Collectors.toMap(a -> ((WordNode) a).id, a -> ((WordNode) a).size()))
105         .entrySet().stream().sorted(byValue.reversed()).forEachOrdered(a -> m.put(a.getKey
106             (), a.getValue()));
107     return m;
108 }
```

6.9.3.4 List<Map<String, Integer>> se2xb3.data.DataHandler.getTrends ()

Get Get trending words.

Returns

list containing the maps of the most popular word, hashtags, and user mentions.

```

69                                     {
70     return getTrends(DEFAULT_REST_RESP_COUNT);
71 }
```

6.9.3.5 List<Map<String, Integer>> se2xb3.data.DataHandler.getTrends (int count)

Get trending words.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

list containing the maps of the most popular word, hashtags, and user mentions.

```

79                                     {
80     List<Map<String, Integer>> trends = new ArrayList<>();
81
82     trends.add(getTrendingWords(count));
83     trends.add(getTrendingHashtags(count));
84     trends.add(getTrendingUsers(count));
85
86     return trends;
87 }
```

6.9.3.6 static void se2xb3.data.DataHandler.print (String s) [static]

```

153                                     {
154     System.out.print(s);
155 }
```

6.9.3.7 static void se2xb3.data.DataHandler.println (String s) [static]

```

149
150     System.out.println(s);
151 }

```

6.9.3.8 void se2xb3.data.DataHandler.processTweet (String tweetStr)

Process a tweet string by deserializing it into a new tweet object. Then store it in a list and send it for further processing and analysis.

Parameters

<i>tweetStr</i>	a string containing a JSON encoded tweet
-----------------	--

```

55
56     Tweet tweet = tweetProcessor.deserializeTweet(tweetStr);
57
58     // the simulator never cleared this list as it loaded tweets into the system, causing the
59     // memory to run out and cause the program to crash
60     tweetsList.add(tweet);
61     tweetProcessor.processTweetText(tweet);
62 }

```

6.9.3.9 void se2xb3.data.DataHandler.resetData ()

```

157
158     tweetProcessor.resetData();
159     tweetsList.clear();
160 }

```

6.9.4 Member Data Documentation

6.9.4.1 Comparator<Map.Entry<String, Integer> > se2xb3.data.DataHandler.byValue [private]

Initial value:

```

=
    (entry1, entry2) -> entry1.getValue().compareTo(entry2.getValue())

```

6.9.4.2 DataController se2xb3.data.DataHandler.dataController [private]

6.9.4.3 GraphStrategy se2xb3.data.DataHandler.graphStrategy = new GraphStrategy() [private]

6.9.4.4 ObjectMapper se2xb3.data.DataHandler.objectMapper = new ObjectMapper() [private]

6.9.4.5 SearchStrategy se2xb3.data.DataHandler.searchStrategy = new SearchStrategy() [private]

6.9.4.6 SortStrategy se2xb3.data.DataHandler.sortStrategy = new SortStrategy() [private]

6.9.4.7 TweetProcessor se2xb3.data.DataHandler.tweetProcessor = new TweetProcessor() [private]

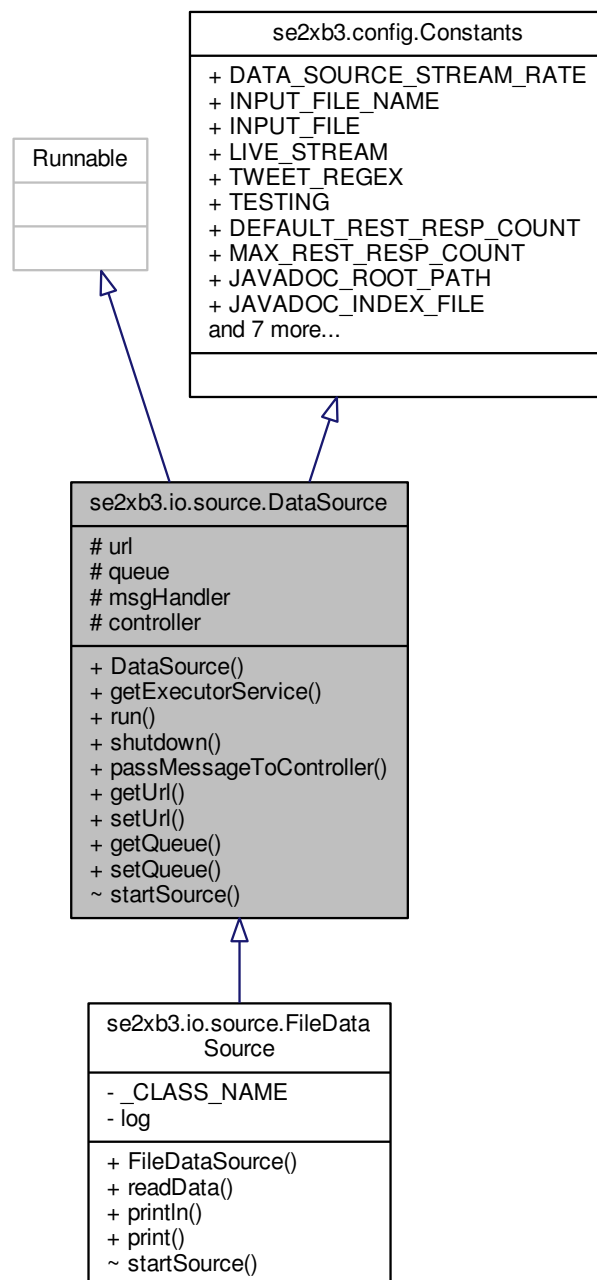
6.9.4.8 List<Tweet> se2xb3.data.DataHandler.tweetsList = new ArrayList<>() [private]

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

- [DataHandler.java](#)

6.10 se2xb3.io.source.DataSource Class Reference

Inheritance diagram for se2xb3.io.source.DataSource:



Additional Inherited Members

6.10.1 Detailed Description

Abstract base class for data sources.

Author

Dawson Myers

Version

1.0

Since

3/10/2017

6.10.2 Constructor & Destructor Documentation

6.10.2.1 se2xb3.io.source.DataSource.DataSource (*IOController controller*)

Constructor that takes an [IOController](#).

Parameters

<i>controller</i>	the controller instances
-------------------	--------------------------

```

38                                     {
39 //         super("DataSource Thread");
40         this.controller = controller;
41 //         start();
42         // start data source asynchronously.
43         getExecutorService().execute(this);
44     }
```

6.10.3 Member Function Documentation

6.10.3.1 ExecutorService se2xb3.io.source.DataSource.getExecutorService ()

Return the executorService instance

Returns

executorService

```

52                                     {
53         return controller.getExecutorService();
54     }
```

6.10.3.2 IMessageQueue<String> se2xb3.io.source.DataSource.getQueue ()

Get the message queue.

Returns

the queue

```

121                                     {
122         return queue;
123     }
```

6.10.3.3 String se2xb3.io.source.DataSource.getUrl ()

Get the url string.

Returns

a url string

```

101                                     {
102         return url;
103     }
```

6.10.3.4 synchronized void se2xb3.io.source.DataSource.passMessageToController (String msg)

Receive new message from message handler and send it to the controller to be processed.

Parameters

<i>msg</i>	a new message string
------------	----------------------

```

92                                     {
93         controller.onMessageReceived(msg);
94     }
```

6.10.3.5 void se2xb3.io.source.DataSource.run ()

Method to run when the thread is started.

```

59                                     {
60         // The handler has to be given a ref to this DataSource instance so
61         // that it can pass messages back to the controller using the
62         // passMessageToController() hook in this class. The messages are
63         // passed as they are received from the message buffer.
64         msgHandler = new MessageHandler(this);
65         // msgHandler = new MessageHandler(controller);
66
67         // The queue is given the handler instance so that messages can be
68         // taken from the queue and given to the handler. The queue runs in a
69         // separate thread.
```



```

70         queue = new AsyncMessageLooper<String>(msgHandler,
71         getExecutorService());
72         // Now that all the components involved in passing data through the
73         // IO subsystem have been initialized, the data source implementation
74         // can be started and begin feeding messages into it as they are
75         // received.
76         startSource();
77     }

```

6.10.3.6 void se2xb3.io.source.DataSource.setQueue (IMessageQueue< String > queue)

Set the message queue.

Parameters

<i>queue</i>	a message queue
--------------	-----------------

```

130                                     {
131 //     public void setQueue(AsyncMessageBuffer<String> queue) {
132         this.queue = queue;
133     }

```

6.10.3.7 void se2xb3.io.source.DataSource.setUrl (String url)

Set the url string. This could be any relevant url that is used in conjunction with a data source.

Parameters

<i>url</i>	a url for the data source
------------	---------------------------

```

112                                     {
113         this.url = url;
114     }

```

6.10.3.8 void se2xb3.io.source.DataSource.shutdown ()

Shutdown blocking queue.

```

82                                     {
83         queue.shutdown();
84     }

```

6.10.3.9 abstract void se2xb3.io.source.DataSource.startSource () [abstract], [package]

This method is called when the thread is started. It begins the process of reading data from a data source.

6.10.4 Member Data Documentation

6.10.4.1 **IOController** `se2xb3.io.source.DataSource.controller` [protected]

6.10.4.2 **MessageHandler** `se2xb3.io.source.DataSource.msgHandler` [protected]

6.10.4.3 **IMessageQueue<String>** `se2xb3.io.source.DataSource.queue` [protected]

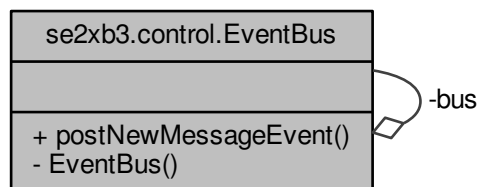
6.10.4.4 **String** `se2xb3.io.source.DataSource.url` = **Constants.INPUT_FILE** [protected]

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

- [DataSource.java](#)

6.11 se2xb3.control.EventBus Class Reference

Collaboration diagram for se2xb3.control.EventBus:



Static Public Member Functions

- static void [postNewMessageEvent](#) (String msg)

Private Member Functions

- [EventBus](#) ()

Static Private Attributes

- static [EventBus](#) `bus` = null

6.11.1 Detailed Description

A class to containing observable events using an eventbus.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.11.2 Constructor & Destructor Documentation

6.11.2.1 se2xb3.control.EventBus.EventBus () [private]

Constructor for singleton instance of [EventBus](#).

```

21         {
22 //         eventBus = new MBassador();
23     }
```

6.11.3 Member Function Documentation

6.11.3.1 static void se2xb3.control.EventBus.postNewMessageEvent (String msg) [static]

Singleton getter. Post new message to be processed.

Parameters

<i>msg</i>	a message string
------------	------------------

```

38         {
39     AppController.getDataController().onMessageReceivedEvent(msg);
40     }
```

6.11.4 Member Data Documentation

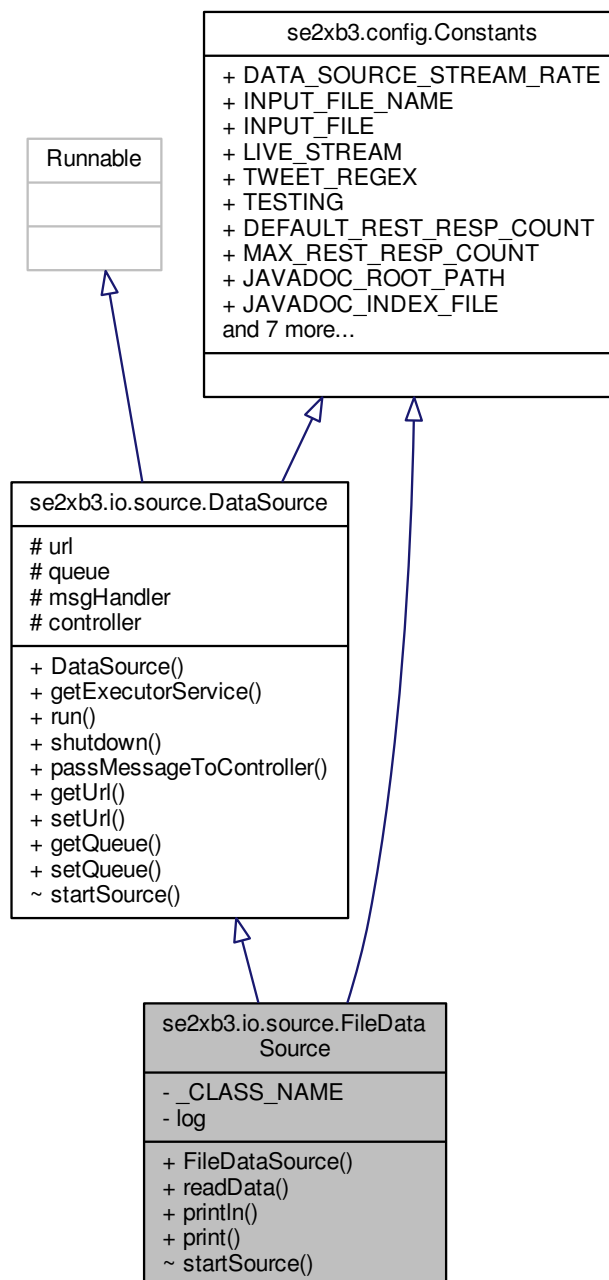
6.11.4.1 EventBus se2xb3.control.EventBus.bus = null [static], [private]

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

- [EventBus.java](#)

6.12 se2xb3.io.source.FileDataSource Class Reference

Inheritance diagram for se2xb3.io.source.FileDataSource:



The diagram illustrates a complex system architecture with multiple modeling contexts and their interactions. Key components include:

- ModelingContext**: A base class or interface defining methods like `getEntityTypes()`, `getRelationshipTypes()`, `getAttributes()`, `getOperations()`, `getConstraints()`, `getViews()`, `getQueries()`, `getReports()`, `getDashboards()`, `getCharts()`, `getTables()`, `getColumns()`, `getRows()`, `getCells()`, `getValues()`, `getFormulas()`, `getScripts()`, `getStyles()`, `getThemes()`, `getFonts()`, `getColors()`, `getImages()`, `getVideos()`, `getAudio()`, `getAnimations()`, `getTransitions()`, `getEffects()`, `getFilters()`, `getSorts()`, `getGroups()`, `getLayers()`, `getZIndexes()`, `getOpacitys()`, `getStrokeWidths()`, `getStrokeDashes()`, `getStrokeColors()`, `getFillColors()`, `getGradients()`, `getPatterns()`, `getBackgrounds()`, `getBorders()`, `getCorners()`, `getRounds()`, `getSquares()`, `getCircles()`, `getEllipses()`, `getRectangles()`, `getPolygons()`, `getLines()`, `getCurves()`, `getPaths()`, `getShapes()`, `getIcons()`, `getLogos()`, `getEmblems()`, `getSeals()`, `getCoatsOfArms()`, `getFlags()`, `getBanners()`, `getPosters()`, `getFlyers()`, `getBrochures()`, `getManuals()`, `getHelpFiles()`, `getFAQs()`, `getGlossaries()`, `getIndexes()`, `getTableOfContents()`, `getFootnotes()`, `getReferences()`, `getSources()`, `getTargets()`, `getDestinations()`, `getOrigins()`, `getLocations()`, `getAddresses()`, `getCoordinates()`, `getDistances()`, `getDirections()`, `getAngles()`, `getSpeeds()`, `getWeights()`, `getVolumes()`, `getAreas()`, `getPerimeters()`, `getCircumferences()`, `getRadii()`, `getDiameters()`, `getCenters()`, `getMidpoints()`, `getEndpoints()`, `getStartPoints()`, `getEndPoints()`, `getInitialStates()`, `getFinalStates()`, `getIntermediateStates()`, `getTransientStates()`, `getStableStates()`, `getUnstableStates()`, `getReachableStates()`, `getUnreachableStates()`, `getPossibleStates()`, `getImpossibleStates()`, `getAllowedStates()`, `getForbiddenStates()`, `getRequiredStates()`, `getOptionalStates()`, `getMandatoryStates()`, `getProhibitedStates()`, `getRecommendedStates()`, `getPreferredStates()`, `getDisallowedStates()`, `getEncouragedStates()`, `getDiscouragedStates()`, `getApprovedStates()`, `getDisapprovedStates()`, `getAcceptedStates()`, `getRejectedStates()`, `getAgreedStates()`, `getDisagreedStates()`, `getConsentedStates()`, `getDisconsentedStates()`, `getAssentedStates()`, `getDisassentedStates()`, `getPartookStates()`, `getDidnotPartookStates()`, `getObed States()`, `getDisobeyedStates()`, `getObeyedStates()`, `getDisobeyedStates()`, `getCompliedStates()`, `getDiscompliedStates()`, `getConformedStates)`

- `FileDataSource` (IOController controller)
- `void readData ()`

- static void **println** (String s)
- static void **print** (String s)

- void startSource ()

- static final String **_CLASS_NAME** = FileDataSource.class.getSimpleName()
- static final Logger **log** = getLogger(**_CLASS_NAME**)

6.12.1 Detailed Description

Generated by Doxygen

Author

Dawson Myers

Version

1.0

Since

3/10/2017

6.12.2 Constructor & Destructor Documentation**6.12.2.1** `se2xb3.io.source.FileDataSource.FileDataSource (IOController controller)`

Constructor that takes an [IOController](#) instance.

Parameters

<i>controller</i>	an instance of the IOController
-------------------	---

```

29                                     {
30         super(controller);
31         setUrl(INPUT\_FILE);
32     }
```

6.12.3 Member Function Documentation**6.12.3.1** `static void se2xb3.io.source.FileDataSource.print (String s) [static]`

```

107                                     {
108         System.out.print(s);
109     }
```

6.12.3.2 `static void se2xb3.io.source.FileDataSource.println (String s) [static]`

```

103                                     {
104         System.out.println(s);
105     }
```

6.12.3.3 `void se2xb3.io.source.FileDataSource.readData ()`

Begin reading tweets from a file. Each line of text is one JSON encoded tweet. The strings are added to the queue as they are read.

```

48         {
49             //log.info("Beginning to read from file");
50             File file = new File(INPUT_FILE);
51
52             String filePath = "";
53
54             // check is INPUT_FILE exists, if not, try to find it using INPUT_FILE_NAME
55             if (!file.exists()) {
56                 log.info("INPUT_FILE " + INPUT_FILE + " not found. Beginning recursive search for
it");
57                 filePath = FileFinder.findFileByName(INPUT_FILE_NAME, "txt");
58             }
59
60             // if length is greater than 1, a file was found
61             if (filePath.length() > 0) {
62                 log.info("Input file " + INPUT_FILE_NAME + " found at path " + filePath);
63                 url = filePath;
64             }
65
66             while (true) {
67                 try (BufferedReader br = new BufferedReader(new FileReader(url))) {
68                     String line = "";
69                     log.info("Beginning to read from file");
70                     int count = 0;
71
72                     // read all lines in file and insert them into the queue
73                     while ((line = br.readLine()) != null) {
74                         //count++;
75                         // simulate stream
76                         try {
77                             Thread.sleep(DATA_SOURCE_STREAM_RATE);
78                         } catch (InterruptedException e) {
79                             e.printStackTrace();
80                         }
81                         // queue.enqueue("Msg "+count++);
82                         queue.enqueue(line);
83                     }
84
85                     // reset data and read it back in again
86                     AppController.getDataController().resetData();
87
88                     // println("Finished reading file");
89                     log.info("Finished reading file. Read " + count + " lines of data");
90                     //if(TestDataSource.testing) TestDataSource.endTest();
91
92                     } catch (FileNotFoundException e) {
93                         e.printStackTrace();
94                     } catch (IOException e) {
95                         e.printStackTrace();
96                     }
97             }
98
99     }

```

6.12.3.4 void se2xb3.io.source.FileDataSource.startSource () [package]

This method is called when the thread is started (a hook). It begins the process of reading data from a data source.

```

39         {
40             readData();
41         }

```

6.12.4 Member Data Documentation

6.12.4.1 final String se2xb3.io.source.FileDataSource._CLASS_NAME = FileDataSource.class.getSimpleName() [static], [private]

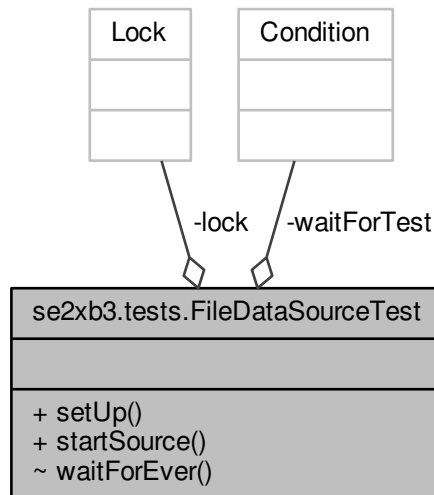
6.12.4.2 final Logger se2xb3.io.source.FileDataSource.log = getLogger(_CLASS_NAME) [static], [private]

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

- [FileDataSource.java](#)

6.13 se2xb3.tests.FileDataSourceTest Class Reference

Collaboration diagram for se2xb3.tests.FileDataSourceTest:



Public Member Functions

- void [setUp](#) () throws Exception
- synchronized void [startSource](#) () throws Exception

Package Functions

- synchronized void [waitForever](#) ()

Private Attributes

- final Lock [lock](#) = new ReentrantLock()
- final Condition [waitForTest](#) = lock.newCondition()

6.13.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.13.2 Member Function Documentation

6.13.2.1 void se2xb3.tests.FileDataSourceTest.setUp () throws Exception

```

23                                     {
24
25     }
```

6.13.2.2 synchronized void se2xb3.tests.FileDataSourceTest.startSource () throws Exception

```

31                                     {
32     IOController controller = new IOController(AppController.getInstance());
33     FileDataSource source = new FileDataSource(controller);
34     Thread.currentThread().wait();
35
36     waitForEver();
37 }
```

6.13.2.3 synchronized void se2xb3.tests.FileDataSourceTest.waitForEver () [package]

```

40                                     {
41     try {
42         lock.lock();
43         while (true)
44             waitForTest.await();
45     } catch (InterruptedException e) {
46         e.printStackTrace();
47     } finally {
48         lock.unlock();
49     }
50 }
```

6.13.3 Member Data Documentation

6.13.3.1 final Lock se2xb3.tests.FileDataSourceTest.lock = new ReentrantLock() [private]

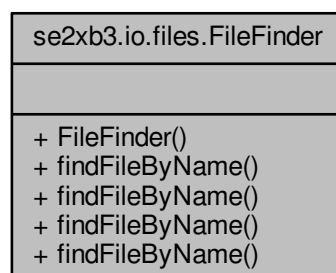
6.13.3.2 final Condition se2xb3.tests.FileDataSourceTest.waitForTest = lock.newCondition() [private]

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

- [FileDataSourceTest.java](#)

6.14 se2xb3.io.files.FileFinder Class Reference

Collaboration diagram for se2xb3.io.files.FileFinder:



Public Member Functions

- [FileFinder](#) ()

Static Public Member Functions

- static String [findFileByName](#) (String name, String extensions)
- static String [findFileByName](#) (String name, String rootDir, String extensions)
- static String [findFileByName](#) (String name, String rootDir, String extensions, boolean recursive)
- static String [findFileByName](#) (String name, String rootDir, String[] extensions, boolean recursive)

6.14.1 Detailed Description

A class to recursively find files.

Author

Dawson Myers

Version

1.0

Since

3/29/2017

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `se2xb3.io.files.FileFinder.FileFinder ()`

```
18 {}
```

6.14.3 Member Function Documentation

6.14.3.1 `static String se2xb3.io.files.FileFinder.findFileByName (String name, String extensions)` `[static]`

```
20
21         return findFileByName(name, ".", new String[]{extensions}, true);
22     }
```

6.14.3.2 `static String se2xb3.io.files.FileFinder.findFileByName (String name, String rootDir, String extensions)` `[static]`

```
23
24         return findFileByName(name, rootDir, new String[]{extensions}, true);
25     }
```

6.14.3.3 static String se2xb3.io.files.FileFinder.findFileByName (String name, String rootDir, String extensions, boolean recursive) [static]

```

26
27         return findFileByName(name, rootDir, new String[]{extensions}, recursive);
28     }
    {

```

6.14.3.4 static String se2xb3.io.files.FileFinder.findFileByName (String name, String rootDir, String[] extensions, boolean recursive) [static]

```

29
30     {
31         File root = new File(rootDir);
32
33         //String[] extensions = { "xml", "java", "dat" };
34
35         Collection files = FileUtils.listFiles(root, extensions, recursive);
36
37         String filePath = "";
38
39         for (Iterator iterator = files.iterator(); iterator.hasNext();) {
40             File file = (File) iterator.next();
41             if (file.getName() != null && file.getName().equals(name)) {
42                 filePath = file.getAbsolutePath();
43             }
44         }
45
46         if (filePath.length() == 0) {
47             try {
48                 throw new FileNotFoundException("Could not find file " + name + "in root dir " +
49                                                     rootDir);
50             } catch (FileNotFoundException e) {
51                 e.printStackTrace();
52             }
53         }
54
55         return filePath;
56     }

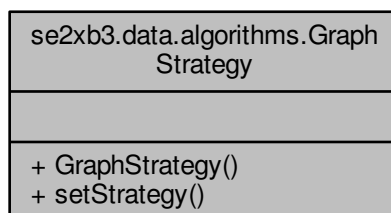
```

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

- [FileFinder.java](#)

6.15 se2xb3.data.algorithms.GraphStrategy Class Reference

Collaboration diagram for se2xb3.data.algorithms.GraphStrategy:



Public Member Functions

- [GraphStrategy](#) ()
- void [setStrategy](#) ()

6.15.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/12/2017

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `se2xb3.data.algorithms.GraphStrategy.GraphStrategy ()`

```
10 {}
```

6.15.3 Member Function Documentation

6.15.3.1 `void se2xb3.data.algorithms.GraphStrategy.setStrategy ()`

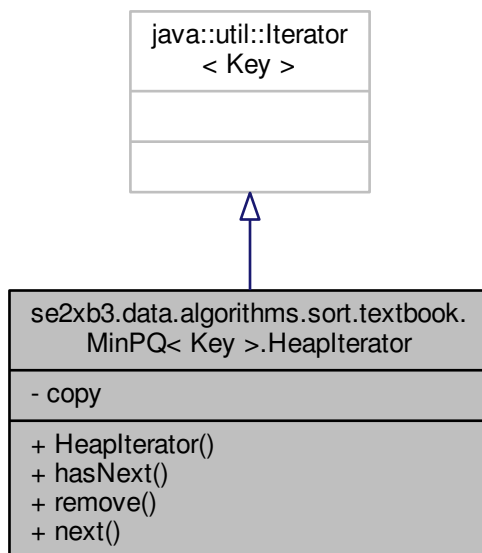
```
12                                     {
13
14 }
```

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

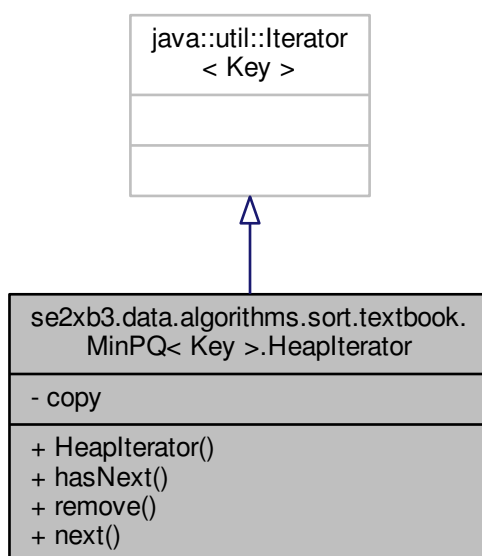
- [GraphStrategy.java](#)

6.16 se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator Class Reference

Inheritance diagram for se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator:



Collaboration diagram for se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator:



Public Member Functions

- [HeapIterator](#) ()
- boolean [hasNext](#) ()
- void [remove](#) ()
- Key [next](#) ()

Private Attributes

- [MinPQ](#)< Key > [copy](#)

6.16.1 Constructor & Destructor Documentation

6.16.1.1 `se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator.HeapIterator ()`

```

253         {
254             if (comparator == null) copy = new MinPQ<Key>(size());
255             else copy = new MinPQ<Key>(size(),
comparator);
256             for (int i = 1; i <= n; i++)
257                 copy.insert(pq[i]);
258         }

```

6.16.2 Member Function Documentation

6.16.2.1 `boolean se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator.hasNext ()`

```

260 { return !copy.isEmpty(); }

```

6.16.2.2 `Key se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator.next ()`

```

263         {
264             if (!hasNext()) throw new NoSuchElementException();
265             return copy.delMin();
266         }

```

6.16.2.3 `void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator.remove ()`

```

261 { throw new UnsupportedOperationException(); }

```

6.16.3 Member Data Documentation

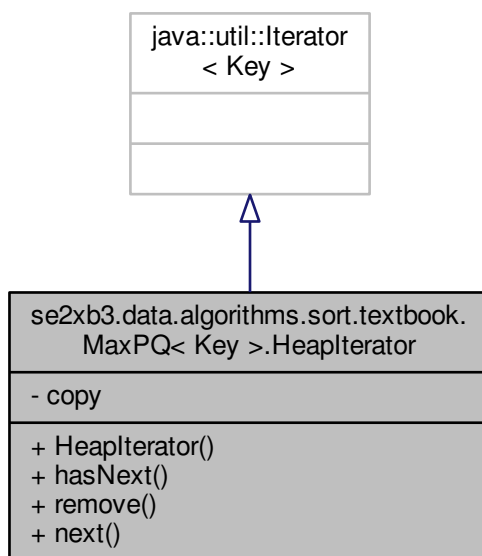
6.16.3.1 `MinPQ<Key> se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator.copy [private]`

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

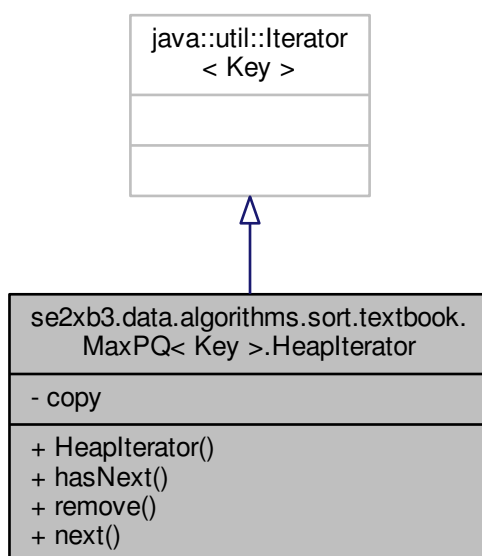
- [MinPQ.java](#)

6.17 se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.Heaplterator Class Reference

Inheritance diagram for se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.Heaplterator:



Collaboration diagram for se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.Heaplterator:



Public Member Functions

- [HeapIterator](#) ()
- boolean [hasNext](#) ()
- void [remove](#) ()
- Key [next](#) ()

Private Attributes

- [MaxPQ](#)< Key > [copy](#)

6.17.1 Constructor & Destructor Documentation

6.17.1.1 `se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator.HeapIterator ()`

```

264         {
265             if (comparator == null) copy = new MaxPQ<Key>(size());
266             else copy = new MaxPQ<Key>(size(),
comparator);
267             for (int i = 1; i <= n; i++)
268                 copy.insert(pq[i]);
269         }

```

6.17.2 Member Function Documentation

6.17.2.1 `boolean se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator.hasNext ()`

```

271 { return !copy.isEmpty(); }

```

6.17.2.2 `Key se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator.next ()`

```

274         {
275             if (!hasNext()) throw new NoSuchElementException();
276             return copy.delMax();
277         }

```

6.17.2.3 `void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator.remove ()`

```

272 { throw new UnsupportedOperationException(); }

```

6.17.3 Member Data Documentation

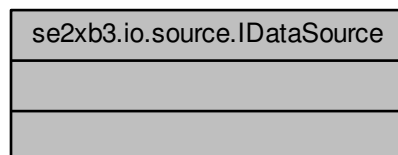
6.17.3.1 `MaxPQ<Key> se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator.copy` [private]

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

- [MaxPQ.java](#)

6.18 se2xb3.io.source.IDataSource Interface Reference

Collaboration diagram for se2xb3.io.source.IDataSource:



6.18.1 Detailed Description

Author

Dawson

Version

1.0

Since

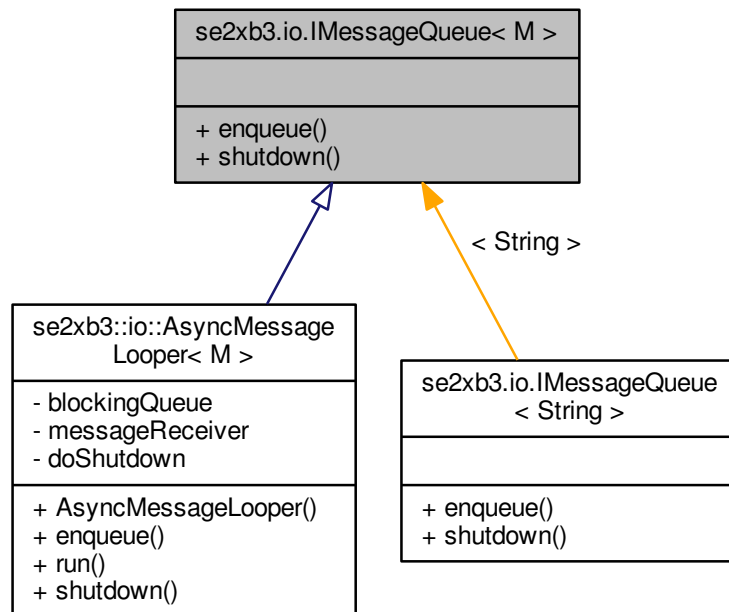
3/11/2017

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

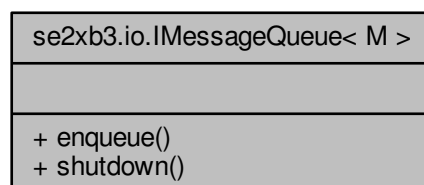
- [IDataSource.java](#)

6.19 se2xb3.io.IMessageQueue< M > Interface Template Reference

Inheritance diagram for se2xb3.io.IMessageQueue< M >:



Collaboration diagram for se2xb3.io.IMessageQueue< M >:



Public Member Functions

- void `enqueue` (M msg)
- void `shutdown` ()

6.19.1 Detailed Description

An interface for a message queue.

Author

Dawson

Version

1.0

Since

3/11/2017

6.19.2 Member Function Documentation

6.19.2.1 void se2xb3.io.IMessageQueue< M >.enqueue (M msg)

Inserts the specified element into a queue.

Parameters

<i>msg</i>	a message
------------	-----------

Implemented in [se2xb3.io.AsyncMessageLooper< M >](#).

6.19.2.2 void se2xb3.io.IMessageQueue< M >.shutdown ()

Shutdown.

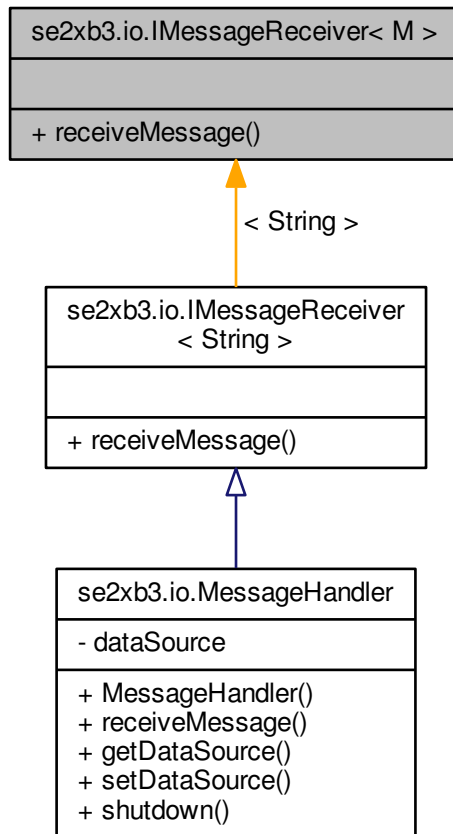
Implemented in [se2xb3.io.AsyncMessageLooper< M >](#).

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

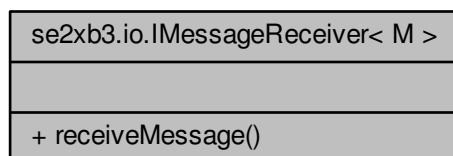
- [IMessageQueue.java](#)

6.20 se2xb3.io.IMessageReceiver< M > Interface Template Reference

Inheritance diagram for se2xb3.io.IMessageReceiver< M >:



Collaboration diagram for se2xb3.io.IMessageReceiver< M >:



Public Member Functions

- void `receiveMessage` (M msg)

6.20.1 Detailed Description

Author

Dawson

Version

1.0

Since

3/11/2017

6.20.2 Member Function Documentation

6.20.2.1 void se2xb3.io.IMessageReceiver< M >.receiveMessage (M *msg*)

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

- [IMessageReceiver.java](#)

Public Member Functions

- [IOController](#) ([AppController](#) appController)
- void [onMessageReceived](#) (String msg)
- ExecutorService [getExecutorService](#) ()
- void [shutdown](#) ()
- List< Map< String, Integer > > [getTrends](#) ()
- List< Map< String, Integer > > [getTrends](#) (int count)
- Map< String, Integer > [getTrendingWords](#) (int count)
- Map< String, Integer > [getTrendingHashtags](#) (int count)
- Map< String, Integer > [getTrendingUsers](#) (int count)

Private Member Functions

- void [init](#) ()

Private Attributes

- [DataSource](#) [dataSource](#)
- [WebServer](#) [webServer](#)

Static Private Attributes

- static [AppController](#) [app](#)

Additional Inherited Members

6.21.1 Detailed Description

Controls all subsystems that involve input/output operations with external resources.

The first subsystem is a DataSource which is an abstract class that can be implemented for any external data source (e.g. Twitter/Facebook/Instagram streams, IoT message brokers, databases, etc.). FileDataSource is a concrete implementation of DataSource and is used in the current system setup. It simulates a stream of tweets by reading prerecorded tweets from a file and then inserting them one at a time into the system to be processed.

The second subsystem is a web server. It provides a REST endpoint that provides data trends from the processed tweets. It also serves an AngularJs web app.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.21.2 Constructor & Destructor Documentation

6.21.2.1 se2xb3.io.IOController.IOController ([AppController](#) appController)

A constructor that takes an instance of the AppConstroller.

Parameters

<i>appController</i>	the controller instance
----------------------	-------------------------

```

45                                     {
46         app = appController;
47         init();
48     }

```

6.21.3 Member Function Documentation**6.21.3.1 ExecutorService se2xb3.io.IOController.getExecutorService ()**

Return the executorService instance

Returns

executorService

```

76                                     {
77         return AppController.getExecutorService();
78     }

```

6.21.3.2 Map<String, Integer> se2xb3.io.IOController.getTrendingHashtags (int count)

Get trending hashtags.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map containing the top trending items

```

125                                     {
126         count = abs(count);
127         return AppController.getDataController().getTrendingHashtags(count);
128     }

```

6.21.3.3 Map<String, Integer> se2xb3.io.IOController.getTrendingUsers (int count)

Get trending users.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map containing the top trending items

```

136                                     {
137         count =    abs(count);
138         return AppController.getDataController().getTrendingUsers(count);
139     }

```

6.21.3.4 Map<String, Integer> se2xb3.io.IOController.getTrendingWords (int count)

Get trending words.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a map containing the top trending items

```

114                                     {
115         count =    abs(count);
116         return AppController.getDataController().getTrendingWords(count);
117     }

```

6.21.3.5 List<Map<String, Integer> > se2xb3.io.IOController.getTrends ()

Get trends from data controller

Returns

list of trends

```

93                                     {
94         return AppController.getDataController().getTrends();
95     }

```

6.21.3.6 List<Map<String, Integer> > se2xb3.io.IOController.getTrends (int count)

Get trends from data controller

Parameters

<i>count</i>	the number of trends to get
--------------	-----------------------------

Returns

list of trends

```

103                                     {
104         count = abs(count);
105         return ApplicationController.getDataController().getTrends(count);
106     }

```

6.21.3.7 void se2xb3.io.IOController.init () [private]

Initialize the controller by creating a new data source instance.

```

53                                     {
54         dataSource = new FileDataSource(this);
55         webServer = new WebServer(this);
56         getExecutorService().execute(webServer);
57     }

```

6.21.3.8 void se2xb3.io.IOController.onMessageReceived (String msg)

Post the message to the event bus so that it can be delivered for processing.

Parameters

<i>msg</i>	a message string
------------	------------------

```

66                                     {
67 //         EventBus.postNewMessageEvent(msg);
68         ApplicationController.getDataController().onMessageReceivedEvent(msg);
69     }

```

6.21.3.9 void se2xb3.io.IOController.shutdown ()

Shutdown blocking queue thread.

```

84                                     {
85         dataSource.shutdown();
86     }

```

6.21.4 Member Data Documentation

6.21.4.1 ApplicationController se2xb3.io.IOController.app [static], [private]

6.21.4.2 DataSource se2xb3.io.IOController.dataSource [private]

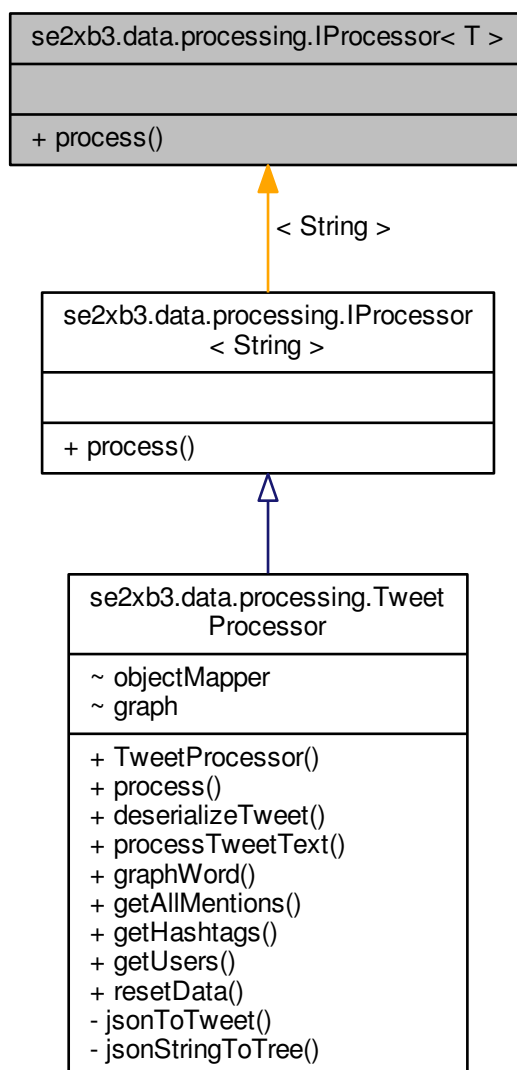
6.21.4.3 WebServer se2xb3.io.IOController.webServer [private]

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

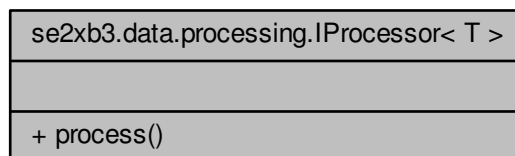
- [IOController.java](#)

6.22 se2xb3.data.processing.IProcessor< T > Interface Template Reference

Inheritance diagram for se2xb3.data.processing.IProcessor< T >:



Collaboration diagram for `se2xb3.data.processing.IProcessor< T >`:



Public Member Functions

- Object [process](#) (T t)

6.22.1 Detailed Description

Classes that implement this interface provide a strategy for processing an instance of a generic parametrized type.

Author

Dawson

Version

1.0

Since

3/11/2017

6.22.2 Member Function Documentation

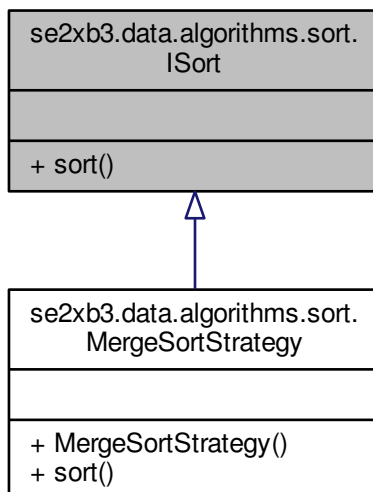
6.22.2.1 Object `se2xb3.data.processing.IProcessor< T >.process (T t)`

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

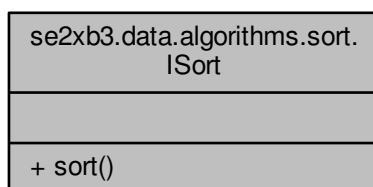
- [IProcessor.java](#)

6.23 se2xb3.data.algorithms.sort.ISort Interface Reference

Inheritance diagram for se2xb3.data.algorithms.sort.ISort:



Collaboration diagram for se2xb3.data.algorithms.sort.ISort:



Public Member Functions

- void `sort` (Comparable[] a)

6.23.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/26/2017

6.23.2 Member Function Documentation

6.23.2.1 void `se2xb3.data.algorithms.sort.ISort.sort (Comparable[] a)`

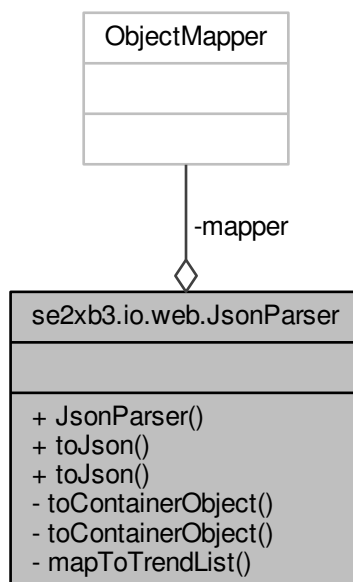
Implemented in [se2xb3.data.algorithms.sort.MergeSortStrategy](#).

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

- [ISort.java](#)

6.24 se2xb3.io.web.JsonParser Class Reference

Collaboration diagram for `se2xb3.io.web.JsonParser`:



Public Member Functions

- [JsonParser \(\)](#)

Static Public Member Functions

- static String [toJson](#) (List< Map< String, Integer >> list)
- static String [toJson](#) (Map< String, Integer > map)

Static Private Member Functions

- static List< List< [Trend](#) > > [toContainerObject](#) (List< Map< String, Integer >> trends)
- static List< [Trend](#) > [toContainerObject](#) (Map< String, Integer > trends)
- static List< [Trend](#) > [mapToTrendList](#) (Map< String, Integer > trendMap)

Static Private Attributes

- static ObjectMapper [mapper](#) = new ObjectMapper()

6.24.1 Detailed Description

A class to handle JSON encoding and decoding for the web server.

Author

Dawson Myers

Version

1.0

Since

3/26/2017

6.24.2 Constructor & Destructor Documentation

6.24.2.1 se2xb3.io.web.JsonParser.JsonParser ()

```
21 {}
```

6.24.3 Member Function Documentation

6.24.3.1 static List<Trend> se2xb3.io.web.JsonParser.mapToTrendList (Map< String, Integer > trendMap) [static], [private]

Convert a

```
Map<String, Integer>
```

to a list of trends

```
List<Trend>
```

.

Parameters

<i>trendMap</i>	trend data to be converted into trend pojos
-----------------	---

Returns

map of trend pojos

```

86                                     {
87         List<Trend> trendList = new ArrayList<>();
88
89         trendMap
90             .entrySet()
91             .stream()
92             .forEachOrdered(a -> trendList.add(new Trend(a.getKey(), a.getValue())));
93
94         return trendList;
95     }

```

6.24.3.2 `static List<List<Trend>> se2xb3.io.web.JsonParser.toContainerObject (List< Map< String, Integer >> trends)` [static], [private]

Convert trends to a list of list of trends. This is done to give the trends proper format when converting to JSON.

Parameters

<i>trends</i>	trend data to be converted into trend pojos
---------------	---

```

60                                     {
61         List<List<Trend>> t = new ArrayList<>();
62
63         for (int i = 0; i < trends.size(); i++) {
64             t.add(mapToTrendList(trends.get(i)));
65         }
66
67         return t;
68     }

```

6.24.3.3 `static List<Trend> se2xb3.io.web.JsonParser.toContainerObject (Map< String, Integer > trends)` [static], [private]

Convert trends to a list of list of trends. This is done to give the trends proper format when converting to JSON.

Parameters

<i>trends</i>	trend data to be converted into trend pojos
---------------	---

```

76                                     {
77         return mapToTrendList(trends);
78     }

```


6.24.3.4 `static String se2xb3.io.web.JsonParser.toJson (List< Map< String, Integer >> list)` `[static]`

Convert trends to a JSON encoded string.

Parameters

<i>list</i>	an object to be serialized into a JSON string
-------------	---

Returns

a JSON encoded string

```

30                                     {
31         try {
32             return mapper.writeValueAsString(toContainerObject(list));
33         } catch (JsonProcessingException e) {
34             e.printStackTrace();
35         }
36         return "none";
37     }

```

6.24.3.5 static String se2xb3.io.web.JsonParser.toJson (Map< String, Integer > map) [static]

Convert trends to a JSON encoded string.

Parameters

<i>map</i>	an object to be serialized into a JSON string
------------	---

Returns

a JSON encoded string

```

45                                     {
46         try {
47             return mapper.writeValueAsString(toContainerObject(map));
48         } catch (JsonProcessingException e) {
49             e.printStackTrace();
50         }
51         return "none";
52     }

```

6.24.4 Member Data Documentation

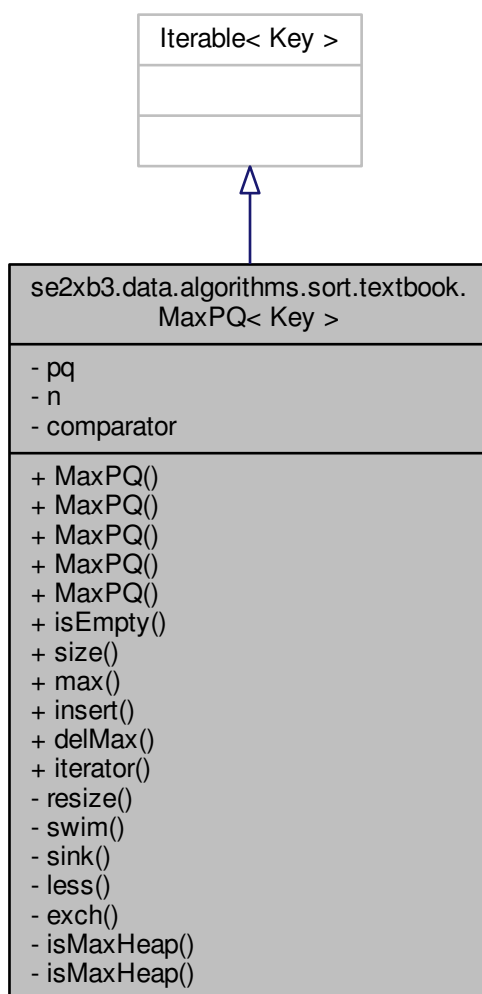
6.24.4.1 ObjectMapper se2xb3.io.web.JsonParser.mapper = new ObjectMapper() [static], [private]

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

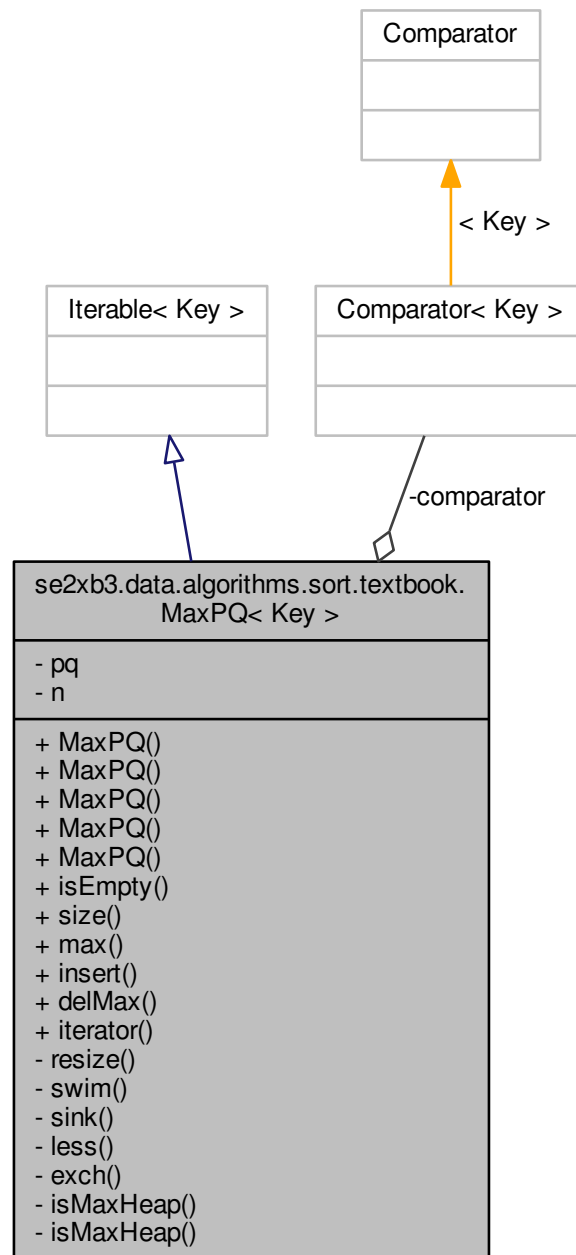
- [JsonParser.java](#)

6.25 se2xb3.data.algorithms.sort.textbook.MaxPQ< Key > Class Template Reference

Inheritance diagram for se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >:



Collaboration diagram for `se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >`:



Classes

- class [HeapIterator](#)

Public Member Functions

- [MaxPQ](#) (int initCapacity)

- `MaxPQ` ()
- `MaxPQ` (int initCapacity, Comparator< Key > `comparator`)
- `MaxPQ` (Comparator< Key > `comparator`)
- `MaxPQ` (Key[] keys)
- boolean `isEmpty` ()
- int `size` ()
- Key `max` ()
- void `insert` (Key x)
- Key `delMax` ()
- Iterator< Key > `iterator` ()

Private Member Functions

- void `resize` (int capacity)
- void `swim` (int k)
- void `sink` (int k)
- boolean `less` (int i, int j)
- void `exch` (int i, int j)
- boolean `isMaxHeap` ()
- boolean `isMaxHeap` (int k)

Private Attributes

- Key[] `pq`
- int `n`
- Comparator< Key > `comparator`

6.25.1 Detailed Description

The

`MaxPQ`

class represents a priority queue of generic keys. It supports the usual *insert* and *delete-the-maximum* operations, along with methods for peeking at the maximum key, testing if the priority queue is empty, and iterating through the keys.

This implementation uses a binary heap. The *insert* and *delete-the-maximum* operations take logarithmic amortized time. The *max*, *size*, and *is-empty* operations take constant time. Construction takes time proportional to the specified capacity or the number of items used to initialize the data structure.

For additional documentation, see [Section 2.4](#) of *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne.

Author

Robert Sedgewick
Kevin Wayne

Parameters

<code><Key></code>	the generic type of key on this priority queue
--------------------------	--

6.25.2 Constructor & Destructor Documentation

6.25.2.1 `se2xb3.data.algorithms.sort.textbook.MaxPQ<Key>.MaxPQ (int initCapacity)`

Initializes an empty priority queue with the given initial capacity.

Parameters

<i>initCapacity</i>	the initial capacity of this priority queue
---------------------	---

```

60         {
61         pq = (Key[]) new Object[initCapacity + 1];
62         n = 0;
63     }
```

6.25.2.2 `se2xb3.data.algorithms.sort.textbook.MaxPQ<Key>.MaxPQ ()`

Initializes an empty priority queue.

```

68         {
69         this(1);
70     }
```

6.25.2.3 `se2xb3.data.algorithms.sort.textbook.MaxPQ<Key>.MaxPQ (int initCapacity, Comparator<Key> comparator)`

Initializes an empty priority queue with the given initial capacity, using the given comparator.

Parameters

<i>initCapacity</i>	the initial capacity of this priority queue
<i>comparator</i>	the order in which to compare the keys

```

79         {
80         this.comparator = comparator;
81         pq = (Key[]) new Object[initCapacity + 1];
82         n = 0;
83     }
```

6.25.2.4 `se2xb3.data.algorithms.sort.textbook.MaxPQ<Key>.MaxPQ (Comparator<Key> comparator)`

Initializes an empty priority queue using the given comparator.

Parameters

<i>comparator</i>	the order in which to compare the keys
-------------------	--

```

90                                     {
91         this(1, comparator);
92     }

```

6.25.2.5 se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.MaxPQ (Key[] keys)

Initializes a priority queue from the array of keys. Takes time proportional to the number of keys, using sink-based heap construction.

Parameters

<i>keys</i>	the array of keys
-------------	-------------------

```

100                                     {
101         n = keys.length;
102         pq = (Key[]) new Object[keys.length + 1];
103         for (int i = 0; i < n; i++)
104             pq[i+1] = keys[i];
105         for (int k = n/2; k >= 1; k--)
106             sink(k);
107         assert isMaxHeap();
108     }

```

6.25.3 Member Function Documentation

6.25.3.1 Key se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.delMax ()

Removes and returns a largest key on this priority queue.

Returns

a largest key on this priority queue

Exceptions

<i>NoSuchElementException</i>	if this priority queue is empty
-------------------------------	---------------------------------

```

175                                     {
176         if (isEmpty()) throw new NoSuchElementException("Priority queue underflow");
177         Key max = pq[1];
178         exch(1, n--);
179         sink(1);
180         pq[n+1] = null; // to avoid loitering and help with garbage collection
181         if ((n > 0) && (n == (pq.length - 1) / 4)) resize(pq.length / 2);
182         assert isMaxHeap();
183         return max;
184     }

```

6.25.3.2 void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.exch (int *i*, int *j*) [private]

```

220                                     {
221     Key swap = pq[i];
222     pq[i] = pq[j];
223     pq[j] = swap;
224 }
```

6.25.3.3 void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.insert (Key *x*)

Adds a new key to this priority queue.

Parameters

<i>x</i>	the new key to add to this priority queue
----------	---

```

158                                     {
159
160     // double size of array if necessary
161     if (n >= pq.length - 1) resize(2 * pq.length);
162
163     // add x, and percolate it up to maintain heap invariant
164     pq[++n] = x;
165     swim(n);
166     assert isMaxHeap();
167 }
```

6.25.3.4 boolean se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.isEmpty ()

Returns true if this priority queue is empty.

Returns

`true`
if this priority queue is empty;

`false`

otherwise

```

118                                     {
119     return n == 0;
120 }
```

6.25.3.5 boolean se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.isMaxHeap () [private]

```

227                                     {
228     return isMaxHeap(1);
229 }
```


6.25.3.6 `boolean se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.isMaxHeap (int k)` [private]

```

232         {
233         if (k > n) return true;
234         int left = 2*k;
235         int right = 2*k + 1;
236         if (left <= n && less(k, left)) return false;
237         if (right <= n && less(k, right)) return false;
238         return isMaxHeap(left) && isMaxHeap(right);
239     }

```

6.25.3.7 `Iterator<Key> se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.iterator ()`

Returns an iterator that iterates over the keys on this priority queue in descending order. The iterator doesn't implement

```
remove()
```

since it's optional.

Returns

an iterator that iterates over the keys in descending order

```

253         {
254         return new HeapIterator();
255     }

```

6.25.3.8 `boolean se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.less (int i, int j)` [private]

```

211         {
212         if (comparator == null) {
213             return ((Comparable<Key>) pq[i]).compareTo(pq[j]) < 0;
214         }
215         else {
216             return comparator.compare(pq[i], pq[j]) < 0;
217         }
218     }

```

6.25.3.9 `Key se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.max ()`

Returns a largest key on this priority queue.

Returns

a largest key on this priority queue

Exceptions

<i>NoSuchElementException</i>	if this priority queue is empty
-------------------------------	---------------------------------

```

137         {
138             if (isEmpty()) throw new NoSuchElementException("Priority queue underflow");
139             return pq[1];
140         }

```

6.25.3.10 void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.resize (int capacity) [private]

```

143         {
144             assert capacity > n;
145             Key[] temp = (Key[]) new Object[capacity];
146             for (int i = 1; i <= n; i++) {
147                 temp[i] = pq[i];
148             }
149             pq = temp;
150         }

```

6.25.3.11 void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.sink (int k) [private]

```

198         {
199             while (2*k <= n) {
200                 int j = 2*k;
201                 if (j < n && less(j, j+1)) j++;
202                 if (!less(k, j)) break;
203                 exch(k, j);
204                 k = j;
205             }
206         }

```

6.25.3.12 int se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.size ()

Returns the number of keys on this priority queue.

Returns

the number of keys on this priority queue

```

127         {
128             return n;
129         }

```

6.25.3.13 void se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.swim (int k) [private]

```

191         {
192             while (k > 1 && less(k/2, k)) {
193                 exch(k, k/2);
194                 k = k/2;
195             }
196         }

```

6.25.4 Member Data Documentation

6.25.4.1 Comparator<Key> se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.comparator [private]

6.25.4.2 int se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.n [private]

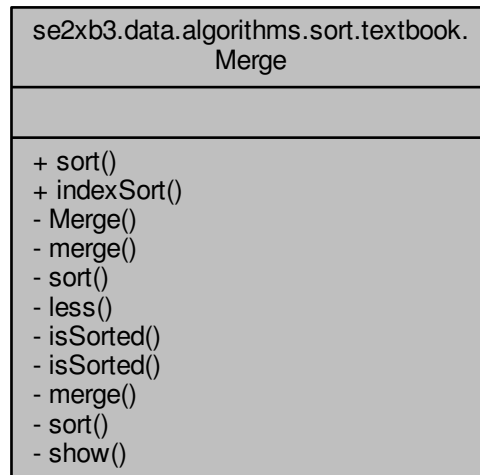
6.25.4.3 Key [] se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.pq [private]

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

- [MaxPQ.java](#)

6.26 se2xb3.data.algorithms.sort.textbook.Merge Class Reference

Collaboration diagram for se2xb3.data.algorithms.sort.textbook.Merge:



Static Public Member Functions

- static void `sort` (Comparable[] a)
- static int[] `indexSort` (Comparable[] a)

Private Member Functions

- `Merge` ()

Static Private Member Functions

- static void `merge` (Comparable[] a, Comparable[] aux, int lo, int mid, int hi)
- static void `sort` (Comparable[] a, Comparable[] aux, int lo, int hi)
- static boolean `less` (Comparable v, Comparable w)
- static boolean `isSorted` (Comparable[] a)
- static boolean `isSorted` (Comparable[] a, int lo, int hi)
- static void `merge` (Comparable[] a, int[] index, int[] aux, int lo, int mid, int hi)
- static void `sort` (Comparable[] a, int[] index, int[] aux, int lo, int hi)
- static void `show` (Comparable[] a)

6.26.1 Detailed Description

The

[Merge](#)

class provides static methods for sorting an array using mergesort.

For additional documentation, see [Section 2.2](#) of *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne. For an optimized version, see [MergeX](#).

Author

Robert Sedgewick
Kevin Wayne

6.26.2 Constructor & Destructor Documentation

6.26.2.1 `se2xb3.data.algorithms.sort.textbook.Merge.Merge () [private]`

```
40 { }
```

6.26.3 Member Function Documentation

6.26.3.1 `static int [] se2xb3.data.algorithms.sort.textbook.Merge.indexSort (Comparable[] a) [static]`

Returns a permutation that gives the elements in the array in ascending order.

Parameters

<i>a</i>	the array
----------	-----------

Returns

a permutation

```
p[]
```

such that

```
a[p[0]]
```

```
,
```

```
a[p[1]]
```

```
, ...,
```

```
a[p[N-1]]
```

are in ascending order

```

136                                     {
137         int n = a.length;
138         int[] index = new int[n];
139         for (int i = 0; i < n; i++)
140             index[i] = i;
141
142         int[] aux = new int[n];
143         sort(a, index, aux, 0, n-1);
144         return index;
145     }

```

6.26.3.2 static boolean se2xb3.data.algorithms.sort.textbook.Merge.isSorted (Comparable[] a) [static], [private]

```

98                                     {
99         return isSorted(a, 0, a.length - 1);
100     }

```

6.26.3.3 static boolean se2xb3.data.algorithms.sort.textbook.Merge.isSorted (Comparable[] a, int lo, int hi) [static], [private]

```

102                                     {
103         for (int i = lo + 1; i <= hi; i++)
104             if (!less(a[i], a[i-1])) return false;
105         return true;
106     }

```

6.26.3.4 static boolean se2xb3.data.algorithms.sort.textbook.Merge.less (Comparable v, Comparable w) [static], [private]

```

91                                     {
92         return v.compareTo(w) < 0;
93     }

```

6.26.3.5 static void se2xb3.data.algorithms.sort.textbook.Merge.merge (Comparable[] a, Comparable[] aux, int lo, int mid, int hi) [static], [private]

```

43                                     {
44         // precondition: a[lo .. mid] and a[mid+1 .. hi] are sorted subarrays
45         assert isSorted(a, lo, mid);
46         assert isSorted(a, mid+1, hi);
47
48         // copy to aux[]
49         for (int k = lo; k <= hi; k++) {
50             aux[k] = a[k];
51         }
52
53         // merge back to a[]
54         int i = lo, j = mid+1;
55         for (int k = lo; k <= hi; k++) {
56             if (i > mid) a[k] = aux[j++];
57             else if (j > hi) a[k] = aux[i++];
58             else if (less(aux[j], aux[i])) a[k] = aux[j++];
59             else a[k] = aux[i++];
60         }
61
62         // postcondition: a[lo .. hi] is sorted
63         assert isSorted(a, lo, hi);
64     }

```

6.26.3.6 `static void se2xb3.data.algorithms.sort.textbook.Merge.merge (Comparable[] a, int[] index, int[] aux, int lo, int mid, int hi)` [static],[private]

```

113                                     {
114
115         // copy to aux[]
116         for (int k = lo; k <= hi; k++) {
117             aux[k] = index[k];
118         }
119
120         // merge back to a[]
121         int i = lo, j = mid+1;
122         for (int k = lo; k <= hi; k++) {
123             if (i > mid)                index[k] = aux[j++];
124             else if (j > hi)            index[k] = aux[i++];
125             else if (less(a[aux[j]], a[aux[i]])) index[k] = aux[j++];
126             else                        index[k] = aux[i++];
127         }
128     }

```

6.26.3.7 `static void se2xb3.data.algorithms.sort.textbook.Merge.show (Comparable[] a)` [static],[private]

```

157                                     {
158         for (int i = 0; i < a.length; i++) {
159             //      StdOut.println(a[i]);
160         }
161     }

```

6.26.3.8 `static void se2xb3.data.algorithms.sort.textbook.Merge.sort (Comparable[] a, Comparable[] aux, int lo, int hi)` [static],[private]

```

67                                     {
68         if (hi <= lo) return;
69         int mid = lo + (hi - lo) / 2;
70         sort(a, aux, lo, mid);
71         sort(a, aux, mid + 1, hi);
72         merge(a, aux, lo, mid, hi);
73     }

```

6.26.3.9 `static void se2xb3.data.algorithms.sort.textbook.Merge.sort (Comparable[] a)` [static]

Rearranges the array in ascending order, using the natural order.

Parameters

<i>a</i>	the array to be sorted
----------	------------------------

```

79                                     {
80         Comparable[] aux = new Comparable[a.length];
81         sort(a, aux, 0, a.length-1);
82         assert isSorted(a);
83     }

```

6.26.3.10 `static void se2xb3.data.algorithms.sort.textbook.Merge.sort (Comparable[] a, int[] index, int[] aux, int lo, int hi)` [static],[private]

```

148                                     {

```

```

149         if (hi <= lo) return;
150         int mid = lo + (hi - lo) / 2;
151         sort(a, index, aux, lo, mid);
152         sort(a, index, aux, mid + 1, hi);
153         merge(a, index, aux, lo, mid, hi);
154     }

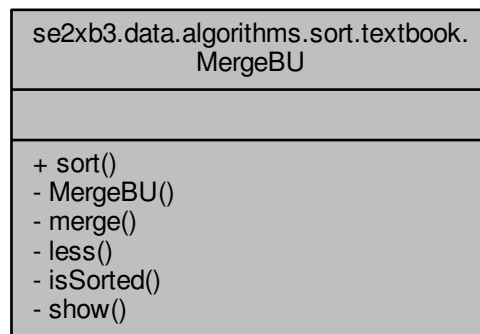
```

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

- [Merge.java](#)

6.27 se2xb3.data.algorithms.sort.textbook.MergeBU Class Reference

Collaboration diagram for se2xb3.data.algorithms.sort.textbook.MergeBU:



Static Public Member Functions

- static void [sort](#) (Comparable[] a)

Private Member Functions

- [MergeBU](#) ()

Static Private Member Functions

- static void [merge](#) (Comparable[] a, Comparable[] aux, int lo, int mid, int hi)
- static boolean [less](#) (Comparable v, Comparable w)
- static boolean [isSorted](#) (Comparable[] a)
- static void [show](#) (Comparable[] a)

6.27.1 Detailed Description

The

`MergeBU`

class provides static methods for sorting an array using bottom-up mergesort.

For additional documentation, see [Section 2.1](#) of *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne.

Author

Robert Sedgewick
Kevin Wayne

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `se2xb3.data.algorithms.sort.textbook.MergeBU.MergeBU ()` [private]

```
40 { }
```

6.27.3 Member Function Documentation

6.27.3.1 `static boolean se2xb3.data.algorithms.sort.textbook.MergeBU.isSorted (Comparable[] a)` [static], [private]

```
91                                     {
92     for (int i = 1; i < a.length; i++)
93         if (less(a[i], a[i-1])) return false;
94     return true;
95 }
```

6.27.3.2 `static boolean se2xb3.data.algorithms.sort.textbook.MergeBU.less (Comparable v, Comparable w)` [static], [private]

```
83                                     {
84     return v.compareTo(w) < 0;
85 }
```

6.27.3.3 `static void se2xb3.data.algorithms.sort.textbook.MergeBU.merge (Comparable[] a, Comparable[] aux, int lo, int mid, int hi)` [static], [private]

```
43                                     {
44
45     // copy to aux[]
46     for (int k = lo; k <= hi; k++) {
47         aux[k] = a[k];
48     }
49
50     // merge back to a[]
51     int i = lo, j = mid+1;
52     for (int k = lo; k <= hi; k++) {
53         if (i > mid) a[k] = aux[j++]; // this copying is unnecessary
54         else if (j > hi) a[k] = aux[i++];
55         else if (less(aux[j], aux[i])) a[k] = aux[j++];
56         else a[k] = aux[i++];
57     }
58
59 }
```


6.27.3.4 `static void se2xb3.data.algorithms.sort.textbook.MergeBU.show (Comparable[] a)` `[static],[private]`

```
98
99     for (int i = 0; i < a.length; i++) {
100 //         StdOut.println(a[i]);
101     }
102 }
```

6.27.3.5 `static void se2xb3.data.algorithms.sort.textbook.MergeBU.sort (Comparable[] a)` `[static]`

Rearranges the array in ascending order, using the natural order.

Parameters

<i>a</i>	the array to be sorted
----------	------------------------

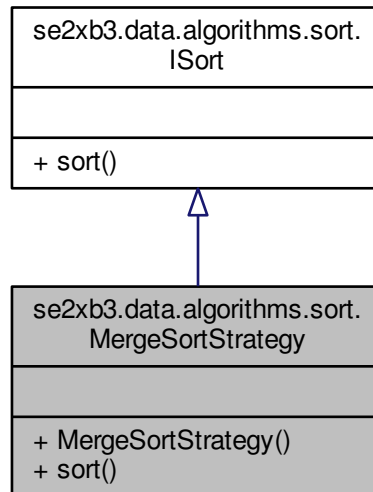
```
65
66     int n = a.length;
67     Comparable[] aux = new Comparable[n];
68     for (int len = 1; len < n; len *= 2) {
69         for (int lo = 0; lo < n-len; lo += len+len) {
70             int mid = lo+len-1;
71             int hi = Math.min(lo+len+len-1, n-1);
72             merge(a, aux, lo, mid, hi);
73         }
74     }
75     assert isSorted(a);
76 }
```

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

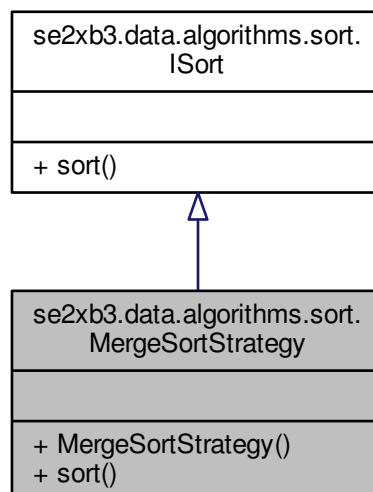
- [MergeBU.java](#)

6.28 se2xb3.data.algorithms.sort.MergeSortStrategy Class Reference

Inheritance diagram for se2xb3.data.algorithms.sort.MergeSortStrategy:



Collaboration diagram for se2xb3.data.algorithms.sort.MergeSortStrategy:



Public Member Functions

- [MergeSortStrategy](#) ()
- void [sort](#) (Comparable[] a)

6.28.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/26/2017

6.28.2 Constructor & Destructor Documentation

6.28.2.1 se2xb3.data.algorithms.sort.MergeSortStrategy.MergeSortStrategy ()

```
12 {}
```

6.28.3 Member Function Documentation

6.28.3.1 void se2xb3.data.algorithms.sort.MergeSortStrategy.sort (Comparable[] a)

Implements [se2xb3.data.algorithms.sort.ISort](#).

```
15                                     {
16     Merge.sort(a);
17 }
```

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

- [MergeSortStrategy.java](#)

6.29 se2xb3.data.algorithms.sort.textbook.MergeX Class Reference

Collaboration diagram for se2xb3.data.algorithms.sort.textbook.MergeX:

se2xb3.data.algorithms.sort.textbook. MergeX
- CUTOFF
+ sort() + sort() - MergeX() - merge() - sort() - insertionSort() - exch() - less() - less() - merge() - sort() - insertionSort() - isSorted() - isSorted() - isSorted() - isSorted() - show()

Static Public Member Functions

- static void [sort](#) (Comparable[] a)
- static void [sort](#) (Object[] a, Comparator comparator)

Private Member Functions

- [MergeX](#) ()

Static Private Member Functions

- static void [merge](#) (Comparable[] src, Comparable[] dst, int lo, int mid, int hi)
- static void [sort](#) (Comparable[] src, Comparable[] dst, int lo, int hi)
- static void [insertionSort](#) (Comparable[] a, int lo, int hi)
- static void [exch](#) (Object[] a, int i, int j)
- static boolean [less](#) (Comparable a, Comparable b)
- static boolean [less](#) (Object a, Object b, Comparator comparator)
- static void [merge](#) (Object[] src, Object[] dst, int lo, int mid, int hi, Comparator comparator)
- static void [sort](#) (Object[] src, Object[] dst, int lo, int hi, Comparator comparator)
- static void [insertionSort](#) (Object[] a, int lo, int hi, Comparator comparator)
- static boolean [isSorted](#) (Comparable[] a)
- static boolean [isSorted](#) (Comparable[] a, int lo, int hi)
- static boolean [isSorted](#) (Object[] a, Comparator comparator)
- static boolean [isSorted](#) (Object[] a, int lo, int hi, Comparator comparator)
- static void [show](#) (Object[] a)

Static Private Attributes

- static final int `CUTOFF` = 7

6.29.1 Detailed Description

The

`MergeX`

class provides static methods for sorting an array using an optimized version of mergesort.

For additional documentation, see [Section 2.2](#) of *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne.

Author

Robert Sedgewick
Kevin Wayne

6.29.2 Constructor & Destructor Documentation

6.29.2.1 `se2xb3.data.algorithms.sort.textbook.MergeX.MergeX()` `[private]`

```
43 { }
```

6.29.3 Member Function Documentation

6.29.3.1 `static void se2xb3.data.algorithms.sort.textbook.MergeX.exch(Object[] a, int i, int j)` `[static], [private]`

```
110                                     {
111     Object swap = a[i];
112     a[i] = a[j];
113     a[j] = swap;
114 }
```

6.29.3.2 `static void se2xb3.data.algorithms.sort.textbook.MergeX.insertionSort(Comparable[] a, int lo, int hi)` `[static], [private]`

```
98                                     {
99     for (int i = lo; i <= hi; i++)
100         for (int j = i; j > lo && less(a[j], a[j-1]); j--)
101             exch(a, j, j-1);
102 }
```

6.29.3.3 `static void se2xb3.data.algorithms.sort.textbook.MergeX.insertionSort(Object[] a, int lo, int hi, Comparator comparator)` `[static], [private]`

```
182                                     {
183     for (int i = lo; i <= hi; i++)
184         for (int j = i; j > lo && less(a[j], a[j-1], comparator); j--)
185             exch(a, j, j-1);
186 }
```

6.29.3.4 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.isSorted (Comparable[] *a*) [static], [private]

```

192                                     {
193     return isSorted(a, 0, a.length - 1);
194 }

```

6.29.3.5 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.isSorted (Comparable[] *a*, int *lo*, int *hi*) [static], [private]

```

196                                     {
197     for (int i = lo + 1; i <= hi; i++)
198         if (less(a[i], a[i-1])) return false;
199     return true;
200 }

```

6.29.3.6 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.isSorted (Object[] *a*, Comparator *comparator*) [static], [private]

```

202                                     {
203     return isSorted(a, 0, a.length - 1, comparator);
204 }

```

6.29.3.7 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.isSorted (Object[] *a*, int *lo*, int *hi*, Comparator *comparator*) [static], [private]

```

206                                     {
207     for (int i = lo + 1; i <= hi; i++)
208         if (less(a[i], a[i-1], comparator)) return false;
209     return true;
210 }

```

6.29.3.8 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.less (Comparable *a*, Comparable *b*) [static], [private]

```

117                                     {
118     return a.compareTo(b) < 0;
119 }

```

6.29.3.9 static boolean se2xb3.data.algorithms.sort.textbook.MergeX.less (Object *a*, Object *b*, Comparator *comparator*) [static], [private]

```

122                                     {
123     return comparator.compare(a, b) < 0;
124 }

```

6.29.3.10 `static void se2xb3.data.algorithms.sort.textbook.MergeX.merge (Comparable[] src, Comparable[] dst, int lo, int mid, int hi)` [static],[private]

```

45                                     {
46
47         // precondition: src[lo .. mid] and src[mid+1 .. hi] are sorted subarrays
48         assert isSorted(src, lo, mid);
49         assert isSorted(src, mid+1, hi);
50
51         int i = lo, j = mid+1;
52         for (int k = lo; k <= hi; k++) {
53             if (i > mid)          dst[k] = src[j++];
54             else if (j > hi)      dst[k] = src[i++];
55             else if (less(src[j], src[i])) dst[k] = src[j++]; // to ensure stability
56             else                 dst[k] = src[i++];
57         }
58
59         // postcondition: dst[lo .. hi] is sorted subarray
60         assert isSorted(dst, lo, hi);
61     }

```

6.29.3.11 `static void se2xb3.data.algorithms.sort.textbook.MergeX.merge (Object[] src, Object[] dst, int lo, int mid, int hi, Comparator comparator)` [static],[private]

```

143                                     {
144
145         // precondition: src[lo .. mid] and src[mid+1 .. hi] are sorted subarrays
146         assert isSorted(src, lo, mid, comparator);
147         assert isSorted(src, mid+1, hi, comparator);
148
149         int i = lo, j = mid+1;
150         for (int k = lo; k <= hi; k++) {
151             if (i > mid)          dst[k] = src[j++];
152             else if (j > hi)      dst[k] = src[i++];
153             else if (less(src[j], src[i], comparator)) dst[k] = src[j++];
154             else                 dst[k] = src[i++];
155         }
156
157         // postcondition: dst[lo .. hi] is sorted subarray
158         assert isSorted(dst, lo, hi, comparator);
159     }

```

6.29.3.12 `static void se2xb3.data.algorithms.sort.textbook.MergeX.show (Object[] a)` [static],[private]

```

213                                     {
214         for (int i = 0; i < a.length; i++) {
215             //StdOut.println(a[i]);
216         }
217     }

```

6.29.3.13 `static void se2xb3.data.algorithms.sort.textbook.MergeX.sort (Comparable[] src, Comparable[] dst, int lo, int hi)` [static],[private]

```

63                                     {
64         // if (hi <= lo) return;
65         if (hi <= lo + CUTOFF) {
66             insertionSort(dst, lo, hi);
67             return;
68         }
69         int mid = lo + (hi - lo) / 2;
70         sort(dst, src, lo, mid);
71         sort(dst, src, mid+1, hi);
72
73         // if (!less(src[mid+1], src[mid])) {
74         //     for (int i = lo; i <= hi; i++) dst[i] = src[i];
75         //     return;
76         // }
77
78         // using System.arraycopy() is a bit faster than the above loop
79         if (!less(src[mid+1], src[mid])) {
80             System.arraycopy(src, lo, dst, lo, hi - lo + 1);
81             return;
82         }
83
84         merge(src, dst, lo, mid, hi);
85     }

```

6.29.3.14 static void se2xb3.data.algorithms.sort.textbook.MergeX.sort (Comparable[] *a*) [static]

Rearranges the array in ascending order, using the natural order.

Parameters

<i>a</i>	the array to be sorted
----------	------------------------

```

91                                     {
92     Comparable[] aux = a.clone();
93     sort(aux, a, 0, a.length-1);
94     assert isSorted(a);
95 }
```

6.29.3.15 static void se2xb3.data.algorithms.sort.textbook.MergeX.sort (Object[] *a*, Comparator *comparator*) [static]

Rearranges the array in ascending order, using the provided order.

Parameters

<i>a</i>	the array to be sorted
<i>comparator</i>	the comparator that defines the total order

```

137                                     {
138     Object[] aux = a.clone();
139     sort(aux, a, 0, a.length-1, comparator);
140     assert isSorted(a, comparator);
141 }
```

6.29.3.16 static void se2xb3.data.algorithms.sort.textbook.MergeX.sort (Object[] *src*, Object[] *dst*, int *lo*, int *hi*, Comparator *comparator*) [static],[private]

```

162                                     {
163     // if (hi <= lo) return;
164     if (hi <= lo + CUTOFF) {
165         insertionSort(dst, lo, hi, comparator);
166         return;
167     }
168     int mid = lo + (hi - lo) / 2;
169     sort(dst, src, lo, mid, comparator);
170     sort(dst, src, mid+1, hi, comparator);
171
172     // using System.arraycopy() is a bit faster than the above loop
173     if (!less(src[mid+1], src[mid], comparator)) {
174         System.arraycopy(src, lo, dst, lo, hi - lo + 1);
175         return;
176     }
177
178     merge(src, dst, lo, mid, hi, comparator);
179 }
```

6.29.4 Member Data Documentation

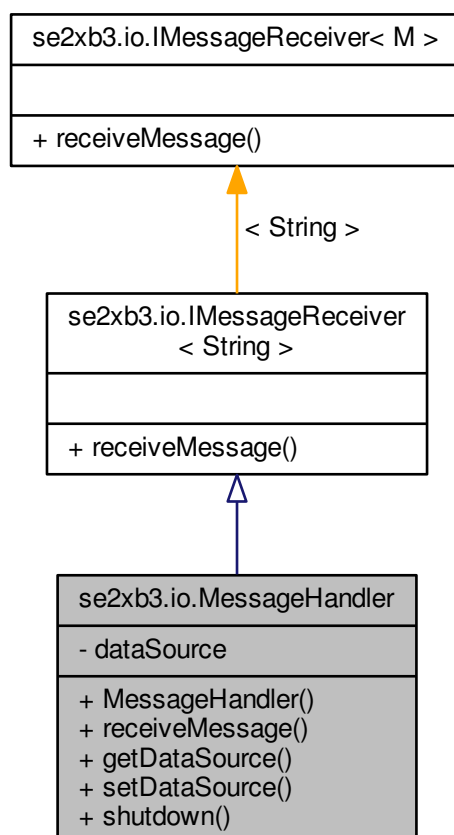
6.29.4.1 final int se2xb3.data.algorithms.sort.textbook.MergeX.CUTOFF = 7 [static],[private]

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

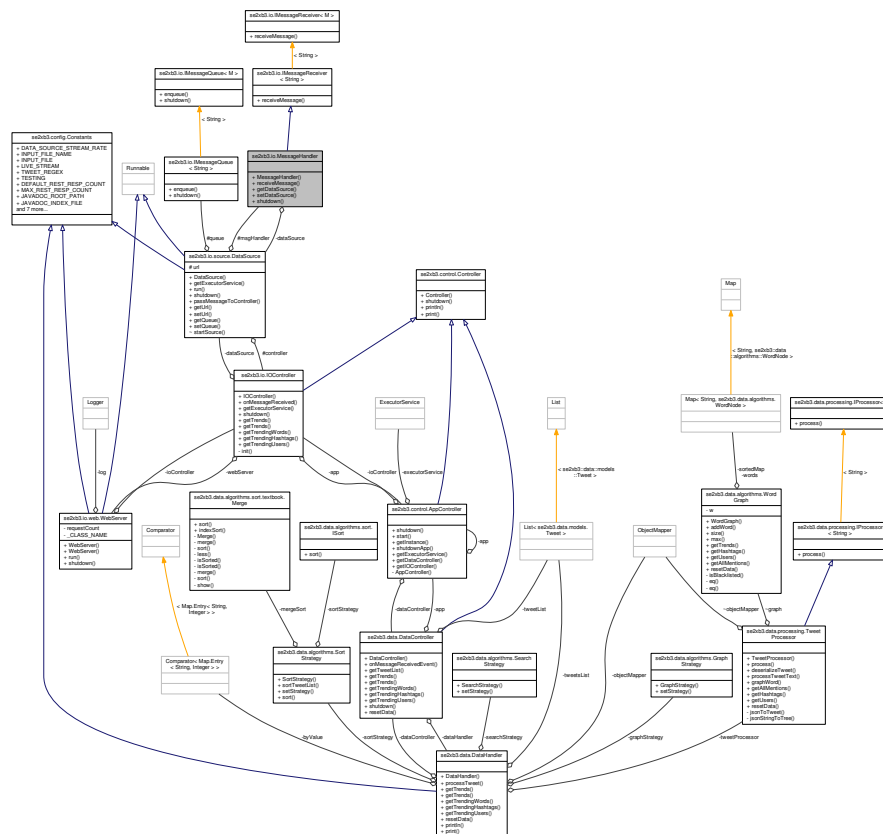
- [MergeX.java](#)

6.30 se2xb3.io.MessageHandler Class Reference

Inheritance diagram for se2xb3.io.MessageHandler:



Collaboration diagram for `se2xb3.io.MessageHandler`:



Public Member Functions

- [MessageHandler](#) ([DataSource](#) dataSource)
- synchronized void [receiveMessage](#) (String msg)
- [DataSource](#) [getDataSource](#) ()
- void [setDataSource](#) ([DataSource](#) dataSource)
- void [shutdown](#) ()

Private Attributes

- [DataSource](#) [dataSource](#)

6.30.1 Detailed Description

A class that handles messages from a data source. When a data source receives a message it inserts it into an implementation of [IMessageQueue](#) and then waits for more messages to arrive. Messages are then taken from the queue as fast as they can be processed and passed to the message handler by calling [receiveMessage\(\)](#) on it. The message is then passed to the [IOController](#) instance where it is then sent to be processed by the data controller.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.30.2 Constructor & Destructor Documentation**6.30.2.1 se2xb3.io.MessageHandler.MessageHandler (DataSource *dataSource*)**

Constructor that takes a DataSource instance. This reference is required so that messages from the data source can be passed back to the controller using the onMessageReceived() hook.

Parameters

<i>dataSource</i>	a data source implementation
-------------------	------------------------------

```
31                                     {
32         this.dataSource = dataSource;
33     }
```

6.30.3 Member Function Documentation**6.30.3.1 DataSource se2xb3.io.MessageHandler.getDataSource ()**

Get DataSource instance.

Returns

a data source implementation

```
51                                     {
52         return dataSource;
53     }
```

6.30.3.2 synchronized void se2xb3.io.MessageHandler.receiveMessage (String *msg*)

Receive new message from message queue and send it to be processed.

Parameters

<i>msg</i>	a string message
------------	------------------

```
42                                     {
43         dataSource.passMessageToController(msg);
44     }
```

6.30.3.3 void se2xb3.io.MessageHandler.setDataSource (DataSource *dataSource*)

Set DataSource instance.

Parameters

<i>dataSource</i>	a data source implementation
-------------------	------------------------------

```
60                                     {
61         this.dataSource = dataSource;
62     }
```

6.30.3.4 void se2xb3.io.MessageHandler.shutdown ()

Shutdown.

```
67                                     {
68     }
```

6.30.4 Member Data Documentation

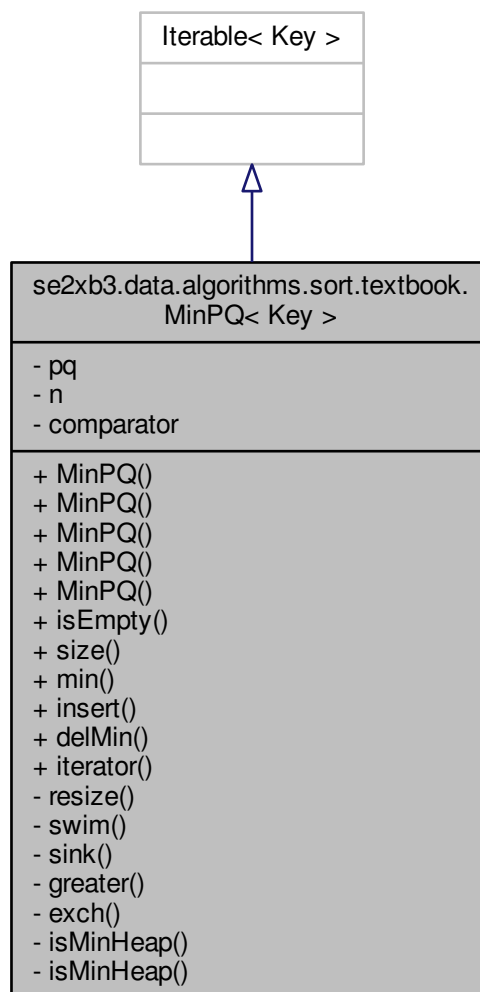
6.30.4.1 DataSource se2xb3.io.MessageHandler.dataSource [private]

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

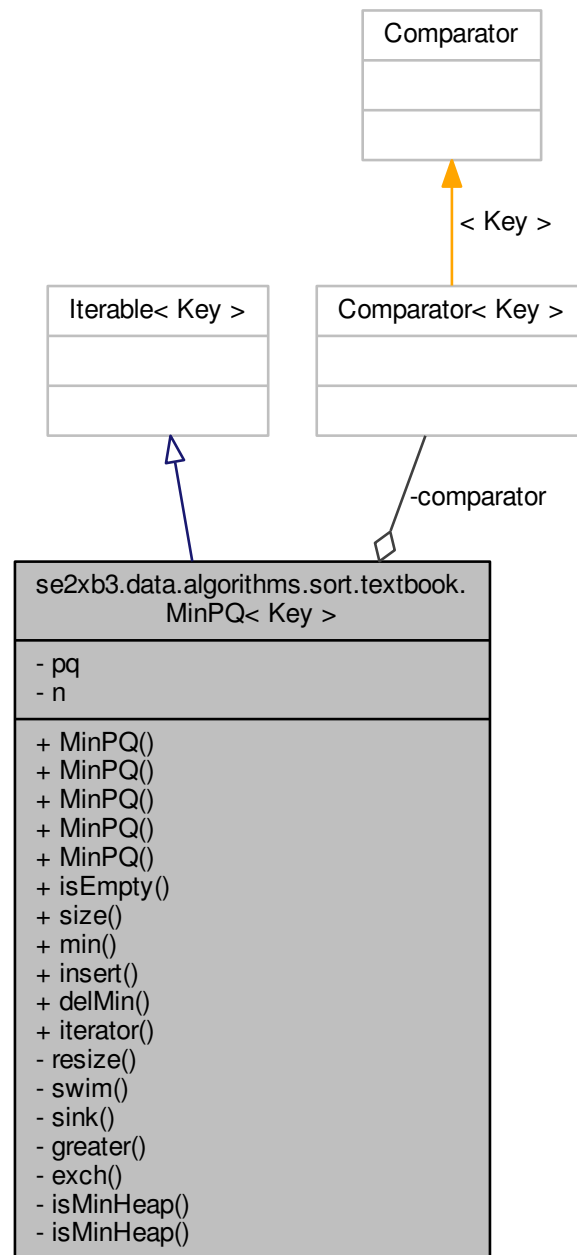
- [MessageHandler.java](#)

6.31 se2xb3.data.algorithms.sort.textbook.MinPQ< Key > Class Template Reference

Inheritance diagram for se2xb3.data.algorithms.sort.textbook.MinPQ< Key >:



Collaboration diagram for `se2xb3.data.algorithms.sort.textbook.MinPQ< Key >`:



Classes

- class [HeapIterator](#)

Public Member Functions

- [MinPQ](#) (int initCapacity)

- [MinPQ](#) ()
- [MinPQ](#) (int initCapacity, Comparator< Key > [comparator](#))
- [MinPQ](#) (Comparator< Key > [comparator](#))
- [MinPQ](#) (Key[] keys)
- boolean [isEmpty](#) ()
- int [size](#) ()
- Key [min](#) ()
- void [insert](#) (Key x)
- Key [delMin](#) ()
- Iterator< Key > [iterator](#) ()

Private Member Functions

- void [resize](#) (int capacity)
- void [swim](#) (int k)
- void [sink](#) (int k)
- boolean [greater](#) (int i, int j)
- void [exch](#) (int i, int j)
- boolean [isMinHeap](#) ()
- boolean [isMinHeap](#) (int k)

Private Attributes

- Key[] [pq](#)
- int [n](#)
- Comparator< Key > [comparator](#)

6.31.1 Detailed Description

The

[MinPQ](#)

class represents a priority queue of generic keys. It supports the usual *insert* and *delete-the-minimum* operations, along with methods for peeking at the minimum key, testing if the priority queue is empty, and iterating through the keys.

This implementation uses a binary heap. The *insert* and *delete-the-minimum* operations take logarithmic amortized time. The *min*, *size*, and *is-empty* operations take constant time. Construction takes time proportional to the specified capacity or the number of items used to initialize the data structure.

For additional documentation, see [Section 2.4](#) of *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne.

Author

Robert Sedgewick
Kevin Wayne

Parameters

<code><Key></code>	the generic type of key on this priority queue
--------------------------	--

6.31.2 Constructor & Destructor Documentation

6.31.2.1 `se2xb3.data.algorithms.sort.textbook.MinPQ<Key>.MinPQ (int initCapacity)`

Initializes an empty priority queue with the given initial capacity.

Parameters

<i>initCapacity</i>	the initial capacity of this priority queue
---------------------	---

```

58         {
59             pq = (Key[]) new Object[initCapacity + 1];
60             n = 0;
61         }

```

6.31.2.2 `se2xb3.data.algorithms.sort.textbook.MinPQ<Key>.MinPQ ()`

Initializes an empty priority queue.

```

66         {
67             this(1);
68         }

```

6.31.2.3 `se2xb3.data.algorithms.sort.textbook.MinPQ<Key>.MinPQ (int initCapacity, Comparator<Key> comparator)`

Initializes an empty priority queue with the given initial capacity, using the given comparator.

Parameters

<i>initCapacity</i>	the initial capacity of this priority queue
<i>comparator</i>	the order to use when comparing keys

```

77         {
78             this.comparator = comparator;
79             pq = (Key[]) new Object[initCapacity + 1];
80             n = 0;
81         }

```

6.31.2.4 `se2xb3.data.algorithms.sort.textbook.MinPQ<Key>.MinPQ (Comparator<Key> comparator)`

Initializes an empty priority queue using the given comparator.

Parameters

<i>comparator</i>	the order to use when comparing keys
-------------------	--------------------------------------

```

88                                     {
89         this(1, comparator);
90     }

```

6.31.2.5 se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.MinPQ (Key[] keys)

Initializes a priority queue from the array of keys.

Takes time proportional to the number of keys, using sink-based heap construction.

Parameters

<i>keys</i>	the array of keys
-------------	-------------------

```

99                                     {
100         n = keys.length;
101         pq = (Key[]) new Object[keys.length + 1];
102         for (int i = 0; i < n; i++)
103             pq[i+1] = keys[i];
104         for (int k = n/2; k >= 1; k--)
105             sink(k);
106         assert isMinHeap();
107     }

```

6.31.3 Member Function Documentation

6.31.3.1 Key se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.delMin ()

Removes and returns a smallest key on this priority queue.

Returns

a smallest key on this priority queue

Exceptions

<i>NoSuchElementException</i>	if this priority queue is empty
-------------------------------	---------------------------------

```

170                                     {
171         if (isEmpty()) throw new NoSuchElementException("Priority queue underflow");
172         exch(1, n);
173         Key min = pq[n--];
174         sink(1);
175         pq[n+1] = null; // avoid loitering and help with garbage collection
176         if ((n > 0) && (n == (pq.length - 1) / 4)) resize(pq.length / 2);
177         assert isMinHeap();
178         return min;
179     }

```

6.31.3.2 void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.exch (int *i*, int *j*) [private]

```

215                                     {
216     Key swap = pq[i];
217     pq[i] = pq[j];
218     pq[j] = swap;
219 }

```

6.31.3.3 boolean se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.greater (int *i*, int *j*) [private]

```

206                                     {
207     if (comparator == null) {
208         return ((Comparable<Key>) pq[i]).compareTo(pq[j]) > 0;
209     }
210     else {
211         return comparator.compare(pq[i], pq[j]) > 0;
212     }
213 }

```

6.31.3.4 void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.insert (Key *x*)

Adds a new key to this priority queue.

Parameters

<i>x</i>	the key to add to this priority queue
----------	---------------------------------------

```

154                                     {
155     // double size of array if necessary
156     if (n == pq.length - 1) resize(2 * pq.length);
157
158     // add x, and percolate it up to maintain heap invariant
159     pq[++n] = x;
160     swim(n);
161     assert isMinHeap();
162 }

```

6.31.3.5 boolean se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.isEmpty ()

Returns true if this priority queue is empty.

Returns

```

true
if this priority queue is empty;

false

otherwise

```

```

115                                     {
116     return n == 0;
117 }

```

6.31.3.6 boolean se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.isMinHeap () [private]

```

222         {
223         return isMinHeap(1);
224     }

```

6.31.3.7 boolean se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.isMinHeap (int k) [private]

```

227         {
228         if (k > n) return true;
229         int left = 2*k;
230         int right = 2*k + 1;
231         if (left <= n && greater(k, left)) return false;
232         if (right <= n && greater(k, right)) return false;
233         return isMinHeap(left) && isMinHeap(right);
234     }

```

6.31.3.8 Iterator<Key> se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.iterator ()

Returns an iterator that iterates over the keys on this priority queue in ascending order.

The iterator doesn't implement

```
remove()
```

since it's optional.

Returns

an iterator that iterates over the keys in ascending order

```
245 { return new HeapIterator(); }
```

6.31.3.9 Key se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.min ()

Returns a smallest key on this priority queue.

Returns

a smallest key on this priority queue

Exceptions

<i>NoSuchElementException</i>	if this priority queue is empty
-------------------------------	---------------------------------

```

134         {
135         if (isEmpty()) throw new NoSuchElementException("Priority queue underflow");
136         return pq[1];
137     }

```

6.31.3.10 void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.resize (int *capacity*) [private]

```

140         {
141             assert capacity > n;
142             Key[] temp = (Key[]) new Object[capacity];
143             for (int i = 1; i <= n; i++) {
144                 temp[i] = pq[i];
145             }
146             pq = temp;
147         }

```

6.31.3.11 void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.sink (int *k*) [private]

```

193         {
194             while (2*k <= n) {
195                 int j = 2*k;
196                 if (j < n && greater(j, j+1)) j++;
197                 if (!greater(k, j)) break;
198                 exch(k, j);
199                 k = j;
200             }
201         }

```

6.31.3.12 int se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.size ()

Returns the number of keys on this priority queue.

Returns

the number of keys on this priority queue

```

124         {
125             return n;
126         }

```

6.31.3.13 void se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.swim (int *k*) [private]

```

186         {
187             while (k > 1 && greater(k/2, k)) {
188                 exch(k, k/2);
189                 k = k/2;
190             }
191         }

```

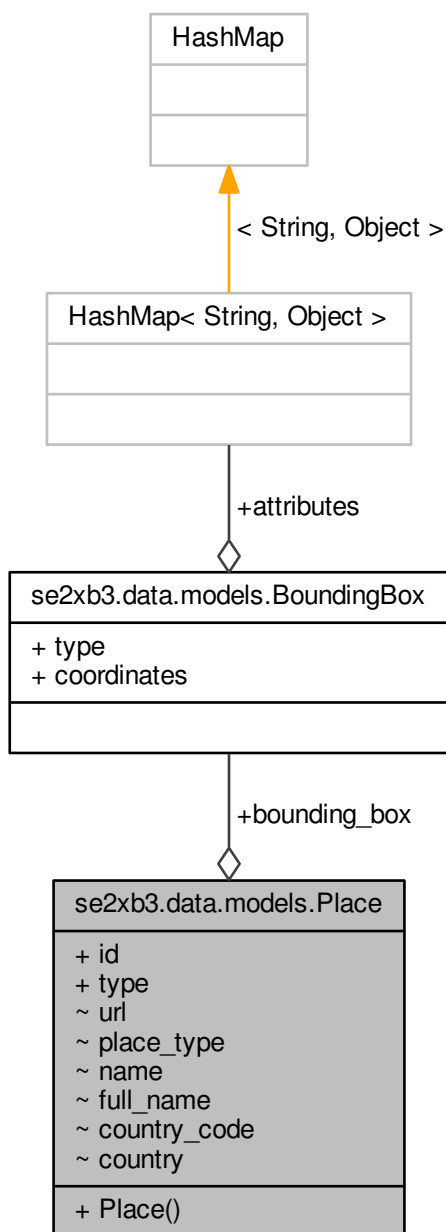
6.31.4 Member Data Documentation**6.31.4.1 Comparator<Key> se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.comparator [private]****6.31.4.2 int se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.n [private]****6.31.4.3 Key[] se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.pq [private]**

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

- [MinPQ.java](#)

6.32 se2xb3.data.models.Place Class Reference

Collaboration diagram for se2xb3.data.models.Place:



Public Member Functions

- [Place](#) ()

Public Attributes

- long [id](#)
- String [type](#)
- [BoundingBox](#) [bounding_box](#)

Package Attributes

- String [url](#)
- String [place_type](#)
- String [name](#)
- String [full_name](#)
- String [country_code](#)
- String [country](#)

6.32.1 Detailed Description

Author

Dawson

Version

1.0

Since

2/23/2017

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `se2xb3.data.models.Place.Place ()`

```
14 {}
```

6.32.3 Member Data Documentation

6.32.3.1 `BoundingBox se2xb3.data.models.Place.bounding_box`

6.32.3.2 `String se2xb3.data.models.Place.country` [package]

6.32.3.3 `String se2xb3.data.models.Place.country_code` [package]

6.32.3.4 `String se2xb3.data.models.Place.full_name` [package]

6.32.3.5 `long se2xb3.data.models.Place.id`

6.32.3.6 `String se2xb3.data.models.Place.name` [package]

6.32.3.7 `String se2xb3.data.models.Place.place_type` [package]

6.32.3.8 `String se2xb3.data.models.Place.type`

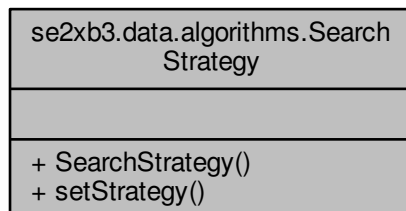
6.32.3.9 `String se2xb3.data.models.Place.url` [package]

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

- [Place.java](#)

6.33 se2xb3.data.algorithms.SearchStrategy Class Reference

Collaboration diagram for se2xb3.data.algorithms.SearchStrategy:



Public Member Functions

- [SearchStrategy](#) ()
- void [setStrategy](#) ()

6.33.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/12/2017

6.33.2 Constructor & Destructor Documentation

6.33.2.1 se2xb3.data.algorithms.SearchStrategy.SearchStrategy ()

```
10 {}
```

6.33.3 Member Function Documentation

6.33.3.1 void se2xb3.data.algorithms.SearchStrategy.setStrategy ()

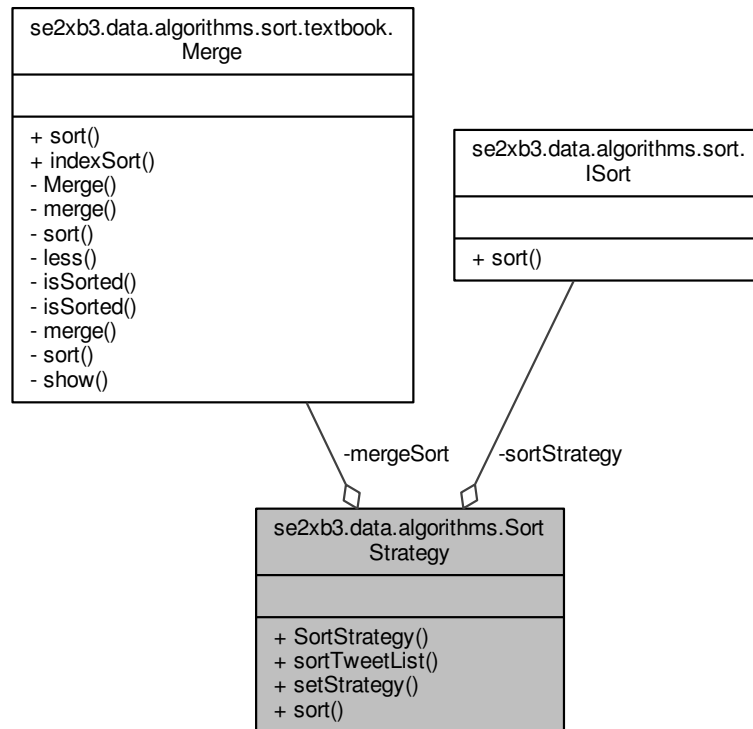
```
11                                     {
12
13     }
```

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

- [SearchStrategy.java](#)

6.34 se2xb3.data.algorithms.SortStrategy Class Reference

Collaboration diagram for se2xb3.data.algorithms.SortStrategy:



Public Member Functions

- [SortStrategy](#) ()
- List< [Tweet](#) > [sortTweetList](#) (List< [Tweet](#) > tweetList)
- void [setStrategy](#) ()
- void [sort](#) (Comparable[] a)

Private Attributes

- [Merge](#) mergeSort
- [ISort](#) sortStrategy

6.34.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/12/2017

6.34.2 Constructor & Destructor Documentation

6.34.2.1 se2xb3.data.algorithms.SortStrategy.SortStrategy ()

```
20 {setStrategy();}
```

6.34.3 Member Function Documentation

6.34.3.1 void se2xb3.data.algorithms.SortStrategy.setStrategy ()

```
26                                     {
27     sortStrategy = new MergeSortStrategy();
28 }
```

6.34.3.2 void se2xb3.data.algorithms.SortStrategy.sort (Comparable[] a)

```
31                                     {
32     sortStrategy.sort(a);
33 }
```

6.34.3.3 List<Tweet> se2xb3.data.algorithms.SortStrategy.sortTweetList (List< Tweet > tweetList)

```
22                                     {
23
24     return null;
25 }
```

6.34.4 Member Data Documentation

6.34.4.1 Merge se2xb3.data.algorithms.SortStrategy.mergeSort [private]

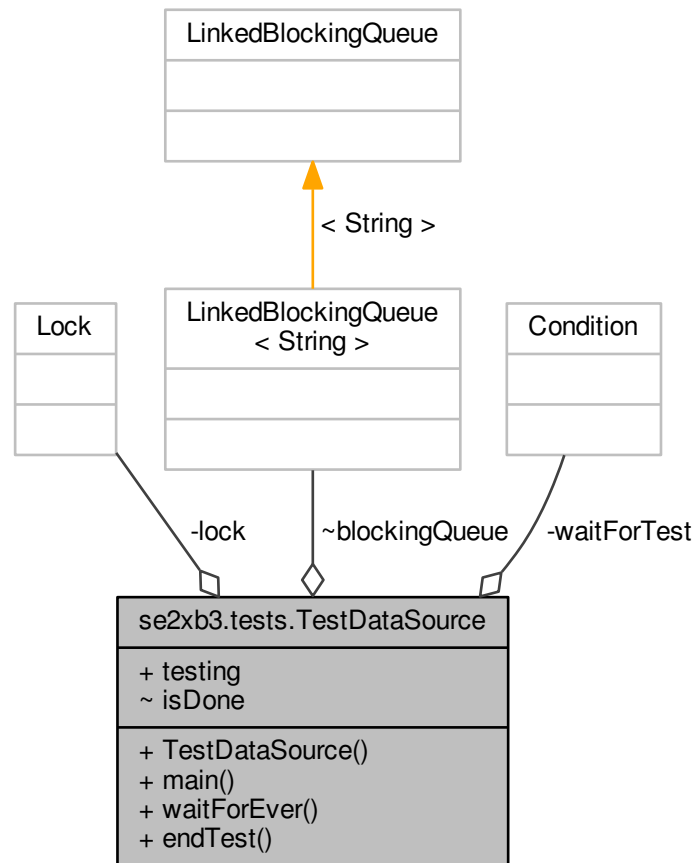
6.34.4.2 ISort se2xb3.data.algorithms.SortStrategy.sortStrategy [private]

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

- [SortStrategy.java](#)

6.35 se2xb3.tests.TestDataSource Class Reference

Collaboration diagram for se2xb3.tests.TestDataSource:



Public Member Functions

- [TestDataSource](#) ()

Static Public Member Functions

- static void [main](#) (String[] args)
- static void [waitForEver](#) ()
- static void [endTest](#) ()

Static Public Attributes

- static volatile boolean [testing](#) = true

Static Package Attributes

- static volatile boolean `isDone` = false
- static volatile `LinkedBlockingQueue< String >` `blockingQueue`

Static Private Attributes

- static final `Lock` `lock` = new `ReentrantLock()`
- static final `Condition` `waitForTest` = `lock.newCondition()`

6.35.1 Detailed Description

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.35.2 Constructor & Destructor Documentation

6.35.2.1 `se2xb3.tests.TestDataSource.TestDataSource ()`

```
20 {}
```

6.35.3 Member Function Documentation

6.35.3.1 `static void se2xb3.tests.TestDataSource.endTest ()` `[static]`

```
44                                     {
45 //         isDone = true;
46 //         waitForTest.signal();
47
48         blockingQueue.add("Done");
49     }
```

6.35.3.2 `static void se2xb3.tests.TestDataSource.main (String[] args)` `[static]`

```
23                                     {
24         AppController app = AppController.getInstance();
25         waitForEver();
26     }
```

6.35.3.3 static void se2xb3.tests.TestDataSource.waitForEver () [static]

```

32         {
33     try {
34         blockingQueue.take(); // wait here
35         AppController.shutdownApp();
36     } catch (InterruptedException e) {
37         e.printStackTrace();
38     } finally {
39     }
40 }
41 }
42 }
```

6.35.4 Member Data Documentation

6.35.4.1 volatile LinkedBlockingQueue<String> se2xb3.tests.TestDataSource.blockingQueue [static], [package]

Initial value:

```

= new
    LinkedBlockingQueue<> ()
```

6.35.4.2 volatile boolean se2xb3.tests.TestDataSource.isDone = false [static], [package]

6.35.4.3 final Lock se2xb3.tests.TestDataSource.lock = new ReentrantLock() [static], [private]

6.35.4.4 volatile boolean se2xb3.tests.TestDataSource.testing = true [static]

6.35.4.5 final Condition se2xb3.tests.TestDataSource.waitForTest = lock.newCondition() [static], [private]

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

- [TestDataSource.java](#)

6.36 se2xb3.io.web.Trend Class Reference

Collaboration diagram for se2xb3.io.web.Trend:

se2xb3.io.web.Trend
+ item + count
+ Trend()

Public Member Functions

- [Trend](#) (String i, int c)

Public Attributes

- String [item](#)
- int [count](#)

6.36.1 Detailed Description

A tend pojo class.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 se2xb3.io.web.Trend.Trend (String *i*, int *c*)

```
105                                     {
106     item = i;
107     count = c;
108 }
```

6.36.3 Member Data Documentation

6.36.3.1 int se2xb3.io.web.Trend.count

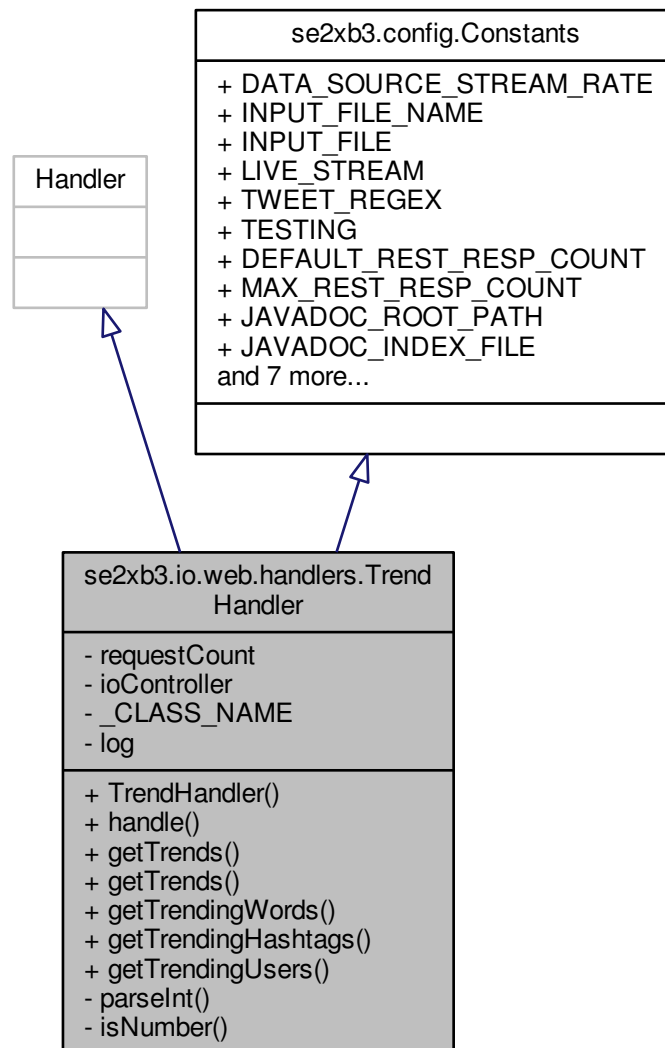
6.36.3.2 String se2xb3.io.web.Trend.item

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

- [JsonParser.java](#)

6.37 se2xb3.io.web.handlers.TrendHandler Class Reference

Inheritance diagram for se2xb3.io.web.handlers.TrendHandler:



[illegible]

- **TrendHandler** (IOController ioController)
- void **handle** (Context ctx) throws Exception
- String **getTrends** ()
- String **getTrends** (String count)
- String **getTrendingWords** (String count)
- String **getTrendingHashtags** (String count)
- String **getTrendingUsers** (String count)

- int **parseInt** (String s)
- boolean **isNumber** (String s)

- long requestCount = 0
- IOController ioController

- static final String `_CLASS_NAME` = TrendHandler.class.getSimpleName()
- static final Logger `log` = LoggerFactory.getLogger(`_CLASS_NAME`)

Additional Inherited Members

6.37.1 Detailed Description

Handle all REST request received by the web server with a prefix of `trends/`.

Author

Dawson Myers

Version

1.0

Since

3/27/2017

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `se2xb3.io.web.handlers.TrendHandler.TrendHandler (IOController ioController)`

```
26 {this.ioController = ioController;}
```

6.37.3 Member Function Documentation

6.37.3.1 `String se2xb3.io.web.handlers.TrendHandler.getTrendingHashtags (String count)`

Get trending hashtags.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a JSON encoded string containing the top trending items

```
157                                     {
158         if(count != null && count != ""){
159             return toJson(ioController.getTrendingHashtags(Integer.parseInt(
count)));
160         }
161         return toJson(ioController.getTrendingHashtags(
DEFAULT_REST_RESP_COUNT));
162     }
```

6.37.3.2 `String se2xb3.io.web.handlers.TrendHandler.getTrendingUsers (String count)`

Get trending users.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a JSON encoded string containing the top trending items

```
169                                     {
170         if(count != null && count != ""){
171             return toJson(ioController.getTrendingUsers(Integer.parseInt(count)
172         ));
173         }
174         return toJson(ioController.getTrendingUsers(
175         DEFAULT_REST_RESP_COUNT));
176     }
```

6.37.3.3 String se2xb3.io.web.handlers.TrendHandler.getTrendingWords (String count)

Get trending words.

Parameters

<i>count</i>	number of top trending items to get.
--------------	--------------------------------------

Returns

a JSON encoded string containing the top trending items

```
145                                     {
146         if(count != null && count != ""){
147             return toJson(ioController.getTrendingWords(Integer.parseInt(count)
148         ));
149         }
150         return toJson(ioController.getTrendingWords(
151         DEFAULT_REST_RESP_COUNT));
152     }
```

6.37.3.4 String se2xb3.io.web.handlers.TrendHandler.getTrends ()

Get trends from data controller

Returns

JSON encoded string containing the trends

```
123                                     {
124         return toJson(ioController.getTrends());
125     }
```

6.37.3.5 String se2xb3.io.web.handlers.TrendHandler.getTrends (String count)

Get trends from data controller.

Parameters

<i>count</i>	the number of items to get
--------------	----------------------------

Returns

JSON encoded string containing the trends

```

133         {
134             if(count != null && count != ""){
135                 return toJson(ioController.getTrends(Integer.parseInt(count)));
136             }
137             return toJson(ioController.getTrends());
138         }

```

6.37.3.6 void se2xb3.io.web.handlers.TrendHandler.handle (Context ctx) throws Exception

This method handles all REST request matching the path pattern `trends/:type?/:count?`. The optional query parameter for the `trends` REST resource are `:type?` and `:count?` (a prefix of ':' indicates a parameter and a suffix of '?' means it's optional). Request can be made in the following ways:

- **trends/{count}**

Type parameter is empty. The response will include the top **count** overall word, hashtag, and user mentions as a JSON array containing 3 arrays (one for each trend). If the the request is just `trends/` without the **count** parameter empty, then the a default value of 10 is used.

- **trends/{type}/{count}**

The response will include the top **count** (an integer value) trends of the type **type**. Where **type** is either overall word, hashtag, or user mentions as a JSON array. If the the request is just `trends/{type}/` without the **count** parameter empty, then the a default value of 10 is used.

```

55         {
56             String type = ctx.getPathTokens().get("type");
57             String count = ctx.getPathTokens().get("count");
58
59             // print out the request count every 100 request
60             requestCount++;
61             if (requestCount % 100 == 0) {
62                 log.info("Trend request count = " + requestCount);
63             }
64
65             if(count != null && count.length() > 3) count = count.substring(0, 2);
66             // trends/ or count is too long
67             if (type == null || type.length() == 0){
68                 // if(type.length() > MAX_REST_RESP_COUNT)
69                 //     ctx.render(toJson(ioController.getTrends(MAX_REST_RESP_COUNT)));
70                 //     else
71                 ctx.render(toJson(ioController.getTrends(
72                     DEFAULT_REST_RESP_COUNT)));
73                 return;
74                 // path = trends/#
75             } else if (isNumber(type)) {
76                 ctx.render(toJson(ioController.getTrends(Integer.parseInt(type))));
77                 return;
78             } else if (type.toLowerCase().equals(REST_RESOURCE_TRENDS_ALL)) {
79                 // }else if (type.toLowerCase().equals("all")) {
80                 if (isNumber(count)) {
81                     ctx.render(toJson(ioController.getTrendingWords(Integer.
82                         parseInt(count))));
83                     return;
84                 }
85                 ctx.render(toJson(ioController.getTrendingWords(
86                     DEFAULT_REST_RESP_COUNT)));
87                 return;
88                 // trends/hashtags/#
89             } else if (type.toLowerCase().equals(REST_RESOURCE_TRENDS_HASHTAGS)) {
90                 if (isNumber(count)) {

```

```

88         ctx.render(toJson(ioController.getTrendingHashtags(Integer.
    parseInt(count))));
89         return;
90     }
91     ctx.render(toJson(ioController.getTrendingHashtags(
    DEFAULT_REST_RESP_COUNT));
92     return;
93     // trends/users/#
94 } else if (type.toLowerCase().equals(REST_RESOURCE_TRENDS_USERS)) {
95     if (isNumber(count)) {
96         ctx.render(toJson(ioController.getTrendingUsers(Integer.
    parseInt(count))));
97         return;
98     }
99     ctx.render(toJson(ioController.getTrendingUsers(
    DEFAULT_REST_RESP_COUNT));
100    return;
101 }
102 ctx.render(REST_REQUEST_NO_MATCH);
103 //     ctx.render("no match");
104 }

```

6.37.3.7 boolean se2xb3.io.web.handlers.TrendHandler.isNumber (String s) [private]

Check if a string contains all numeric characters

Parameters

s	a string
---	----------

Returns

true if string is a number

```

115         {
116         return s != null && s.length() != 0 && s.chars().allMatch( Character::isDigit );
117     }

```

6.37.3.8 int se2xb3.io.web.handlers.TrendHandler.parseInt (String s) [private]

```

106         {
107         return abs(Integer.parseInt(s));
108     }

```

6.37.4 Member Data Documentation

6.37.4.1 final String se2xb3.io.web.handlers.TrendHandler._CLASS_NAME = TrendHandler.class.getSimpleName()
[static], [private]

6.37.4.2 IOController se2xb3.io.web.handlers.TrendHandler.ioController [private]

6.37.4.3 final Logger se2xb3.io.web.handlers.TrendHandler.log = LoggerFactory.getLogger(_CLASS_NAME) [static],
[private]

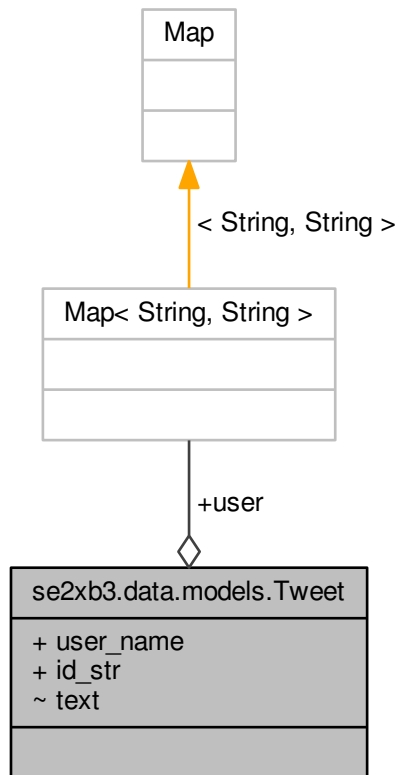
6.37.4.4 long se2xb3.io.web.handlers.TrendHandler.requestCount = 0 [private]

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

- [TrendHandler.java](#)

6.38 se2xb3.data.models.Tweet Class Reference

Collaboration diagram for se2xb3.data.models.Tweet:



Public Attributes

- `Map< String, String >` [user](#)
- `String` [user_name](#)
- `String` [id_str](#)

Package Attributes

- `String` [text](#)

6.38.1 Detailed Description

Author

Dawson

Version

1.0

Since

2/23/2017

6.38.2 Member Data Documentation

6.38.2.1 String se2xb3.data.models.Tweet.id_str

6.38.2.2 String se2xb3.data.models.Tweet.text [package]

6.38.2.3 Map<String, String> se2xb3.data.models.Tweet.user

6.38.2.4 String se2xb3.data.models.Tweet.user_name

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

- [Tweet.java](#)

Public Attributes

- String [created_at](#)
- long [id](#)
- String [id_str](#)
- String [source](#)
- long [in_reply_to_status_id](#)
- String [in_reply_to_status_id_str](#)
- [User](#) [user](#)
- String [geo](#)
- [Place](#) [place](#)
- String [timestamp_ms](#)
- long [quoted_status_id](#)
- [Tweet](#) [quoted_status](#)
- int [retweet_count](#)

Package Attributes

- String [text](#)
- long [in_reply_to_user_id](#)
- String [in_reply_to_user_id_str](#)
- String [in_reply_to_screen_name](#)
- String [coordinates](#)
- String [lang](#)
- int [favorite_count](#)

Static Package Attributes

- static final String [IF_NULL_STR](#) = "none"
- static final int [IF_NULL_INT](#) = -1
- static final long [IF_NULL_LONG](#) = -1

Private Member Functions

- String [checkNull](#) (String key)
- Long [checkNullLong](#) (String key)

Private Attributes

- [JsonNode](#) [node](#)

6.39.1 Constructor & Destructor Documentation

6.39.1.1 se2xb3.data.models.TweetOld.TweetOld ()

```
57 {}
```

6.39.1.2 se2xb3.data.models.TweetOld.TweetOld (JsonNode n)

```

59         {
60             node = n;
61             created_at = n.findValue("created_at").asText(IF_NULL_STR);
62             id = n.findValue("id").asLong(IF_NULL_LONG);
63
64             id_str = n.findValue("id_str").asText(IF_NULL_STR);
65             text = n.findValue("text").asText(IF_NULL_STR);
66             in_reply_to_status_id_str = n.findValue("in_reply_to_status_id_str").
asText(IF_NULL_STR);
67             in_reply_to_user_id_str = n.findValue("in_reply_to_user_id_str").asText(
IF_NULL_STR);
68             in_reply_to_screen_name = n.findValue("in_reply_to_screen_name").asText(
IF_NULL_STR);
69             timestamp_ms = checkNull("timestamp_ms");
70             // timestamp_ms = n.findValue("timestamp_ms").asText(IF_NULL_STR);
71             lang = n.findValue("lang").asText(IF_NULL_STR);
72
73             in_reply_to_status_id = checkNullLong("in_reply_to_status_id");
74             // in_reply_to_status_id = n.findValue("in_reply_to_status_id").asLong(IF_NULL_INT);
75             in_reply_to_user_id = checkNullLong("in_reply_to_user_id");
76             //n.findValue("in_reply_to_user_id").asLong(IF_NULL_INT);
77
78             quoted_status_id = checkNullLong("quoted_status_id");
79             //n.findValue("quoted_status_id").asLong(IF_NULL_LONG);
80
81             retweet_count = n.findValue("retweet_count").asInt(
IF_NULL_INT);
82             favorite_count = n.findValue("favorite_count").asInt(
IF_NULL_INT);
83
84             user = new User(n.findValue("user"));
85             if (n.findValue("quoted_status") != null) {
86                 // quoted_status = new Tweet(n.findValue("quoted_status"));
87             }
88
89             if (n.findValue("place") != null) {
90                 //place = new Place(n.findValue("quoted_status"));
91             }
92         }

```

6.39.2 Member Function Documentation

6.39.2.1 String se2xb3.data.models.TweetOld.checkNull (String key) [private]

Check if the value of a field exists. Returns the value of the field if it is not null, "none" is returned otherwise.

Parameters

key	
-----	--

Returns

the value of the field if not null, "none" otherwise.

```

101         {
102             if(node.findValue(key) != null){
103                 return node.findValue(key).asText(IF_NULL_STR);
104             }
105             return IF_NULL_STR;
106         }

```

6.39.2.2 Long se2xb3.data.models.TweetOld.checkNullLong (String key) [private]

Check if the value of a field exists. Returns the value of the field if it is not null, -1L is returned otherwise.

Parameters

<i>key</i>	
------------	--

Returns

the value of the field if not null, "none" otherwise.

```

113         {
114             if (node.findValue(key) != null) {
115                 return node.findValue(key).asLong(IF_NULL_LONG);
116             }
117             return IF_NULL_LONG;
118         }

```

6.39.2.3 String se2xb3.data.models.TweetOld.getText ()

Get the value of the tweet text.

Returns

tweet text.

```

124         {
125             return text;
126         }

```

6.39.3 Member Data Documentation

6.39.3.1 String se2xb3.data.models.TweetOld.coordinates [package]

6.39.3.2 String se2xb3.data.models.TweetOld.created_at

6.39.3.3 int se2xb3.data.models.TweetOld.favorite_count [package]

6.39.3.4 String se2xb3.data.models.TweetOld.geo

6.39.3.5 long se2xb3.data.models.TweetOld.id

6.39.3.6 String se2xb3.data.models.TweetOld.id_str

6.39.3.7 final int se2xb3.data.models.TweetOld.IF_NULL_INT = -1 [static], [package]

6.39.3.8 final long se2xb3.data.models.TweetOld.IF_NULL_LONG = -1 [static], [package]

6.39.3.9 final String se2xb3.data.models.TweetOld.IF_NULL_STR = "none" [static], [package]

6.39.3.10 String se2xb3.data.models.TweetOld.in_reply_to_screen_name [package]

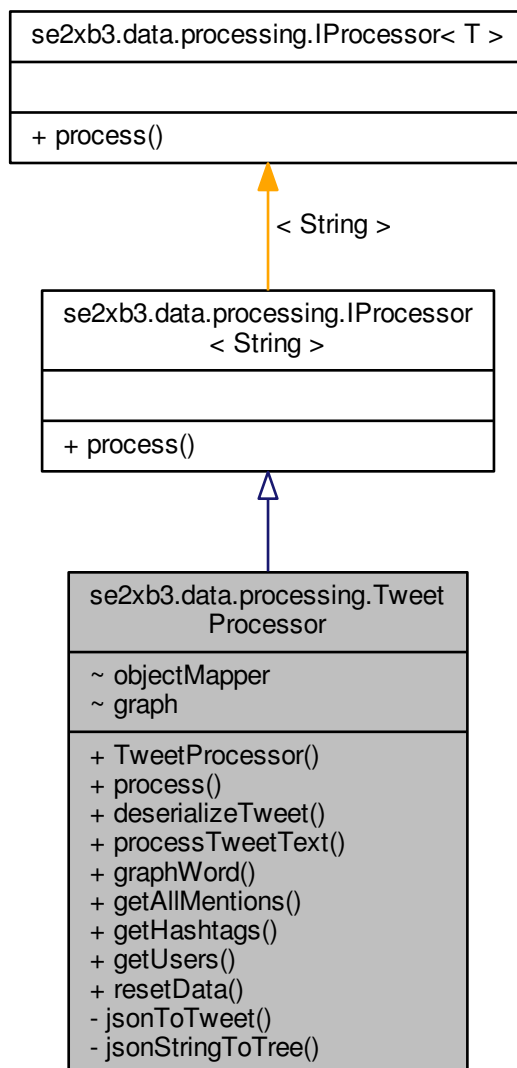
- 6.39.3.11 `long se2xb3.data.models.TweetOld.in_reply_to_status_id`
- 6.39.3.12 `String se2xb3.data.models.TweetOld.in_reply_to_status_id_str`
- 6.39.3.13 `long se2xb3.data.models.TweetOld.in_reply_to_user_id` [package]
- 6.39.3.14 `String se2xb3.data.models.TweetOld.in_reply_to_user_id_str` [package]
- 6.39.3.15 `String se2xb3.data.models.TweetOld.lang` [package]
- 6.39.3.16 `JsonNode se2xb3.data.models.TweetOld.node` [private]
- 6.39.3.17 `Place se2xb3.data.models.TweetOld.place`
- 6.39.3.18 `Tweet se2xb3.data.models.TweetOld.quoted_status`
- 6.39.3.19 `long se2xb3.data.models.TweetOld.quoted_status_id`
- 6.39.3.20 `int se2xb3.data.models.TweetOld.retweet_count`
- 6.39.3.21 `String se2xb3.data.models.TweetOld.source`
- 6.39.3.22 `String se2xb3.data.models.TweetOld.text` [package]
- 6.39.3.23 `String se2xb3.data.models.TweetOld.timestamp_ms`
- 6.39.3.24 `User se2xb3.data.models.TweetOld.user`

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

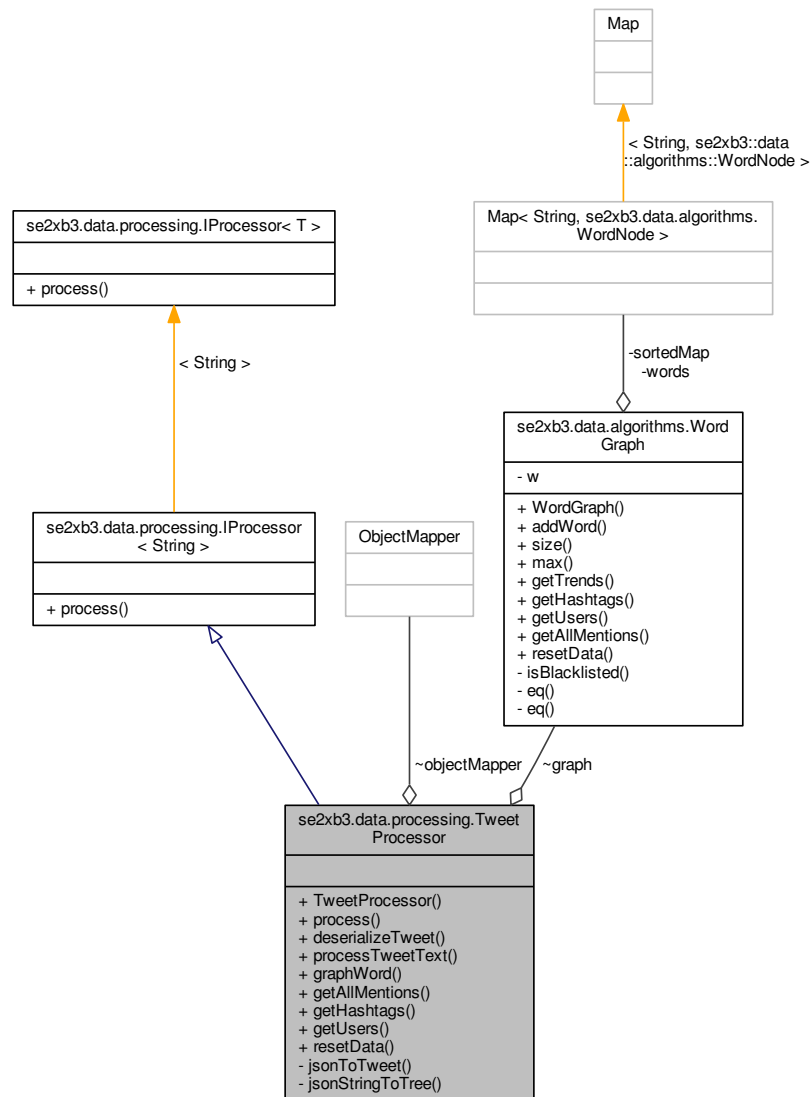
- [Tweet.java](#)

6.40 se2xb3.data.processing.TweetProcessor Class Reference

Inheritance diagram for se2xb3.data.processing.TweetProcessor:



Collaboration diagram for `se2xb3.data.processing.TweetProcessor`:



Public Member Functions

- [TweetProcessor](#) ()
- Object [process](#) (String tweetStr)
- [Tweet deserializeTweet](#) (String tweetStr)
- void [processTweetText](#) ([Tweet](#) tweet)
- void [graphWord](#) (String word, [Tweet](#) tweet)
- [WordNode\[\]](#) [getAllMentions](#) ()
- [WordNode\[\]](#) [getHashtags](#) ()
- [WordNode\[\]](#) [getUsers](#) ()
- void [resetData](#) ()

Package Attributes

- ObjectMapper [objectMapper](#) = new ObjectMapper()
- WordGraph [graph](#) = new WordGraph()

Private Member Functions

- [Tweet jsonToTweet](#) (String s)
- JsonNode [jsonStringToTree](#) (String jsonStr)

6.40.1 Detailed Description

A class to process JSON encoded tweets.

Author

Dawson Myers

Version

1.0

Since

3/11/2017

6.40.2 Constructor & Destructor Documentation

6.40.2.1 se2xb3.data.processing.TweetProcessor.TweetProcessor ()

```
26 {}
```

6.40.3 Member Function Documentation

6.40.3.1 Tweet se2xb3.data.processing.TweetProcessor.deserializeTweet (String tweetStr)

Deserialize tweet stored in a JSON encoded string. Then, return a new instance of a Tweet containing the tweet data.

Parameters

<i>tweetStr</i>	a JSON encoded string containing a tweet
-----------------	--

Returns

a Tweet object containing the data from the JSON string

```

52
53 //         JsonNode n = jsonStringToTree(tweetStr); {
54         return jsonToTweet(tweetStr);
55 //         return new Tweet(n);
56     }

```

6.40.3.2 WordNode [] se2xb3.data.processing.TweetProcessor.getAllMentions ()

Get array of all mentions as an array of word nodes

Returns

an array of all of the word nodes

```

121
122         return graph.getAllMentions(); {
123     }

```

6.40.3.3 WordNode [] se2xb3.data.processing.TweetProcessor.getHashtags ()

Get array of hashtag mentions as an array of word nodes

Returns

an array of all words that start with a #

```

130
131         return graph.getHashtags(); {
132     }

```

6.40.3.4 WordNode [] se2xb3.data.processing.TweetProcessor.getUsers ()

Get array of user mentions as an array of word nodes

Returns

all words that start with a @

```

139
140         return graph.getUsers(); {
141     }

```

6.40.3.5 void se2xb3.data.processing.TweetProcessor.graphWord (String word, Tweet tweet)

This is building a graph of words and tweets. One tweet is connected to many word nodes.

Parameters

<i>word</i>	a word to add to the graph
<i>tweet</i>	the tweet that uses the word

```

112                                     {
113         graph.addWord(word, tweet);
114     }

```

6.40.3.6 JsonNode se2xb3.data.processing.TweetProcessor.jsonStringToTree (String *jsonStr*) [private]

Convert JSON string into a tree containing the parsed data. The tree structure provides a means to easily search through the tree and extract the data we want.

Parameters

<i>jsonStr</i>	a JSON encoded string containing a tweet
----------------	--

Returns

a tree of JsonNode objects

```

74                                     {
75         JsonNode node = null;
76         try {
77             node = objectMapper.readValue(jsonStr, JsonNode.class);
78         } catch (IOException e) {
79             e.printStackTrace();
80         }
81         return node;
82     }

```

6.40.3.7 Tweet se2xb3.data.processing.TweetProcessor.jsonToTweet (String *s*) [private]

```

58                                     {
59         Tweet tweet = null;
60         try {
61             tweet = objectMapper.readValue(s, Tweet.class);
62         } catch (IOException e) {
63             e.printStackTrace();
64         }
65         return tweet;
66     }

```

6.40.3.8 Object se2xb3.data.processing.TweetProcessor.process (String *tweetStr*)

Process a string containing a JSON encoded string. The string is deserialized and the data is stored in a new instance of a Tweet.

Parameters

<i>tweetStr</i>	a JSON encoded string containing a tweet
-----------------	--

Returns

a Tweet object containing the data from the JSON string

```

36                                     {

```

```

37         JsonNode n      = jsonStringToTree(tweetStr);
38         //         Tweet tweet = new Tweet(n);
39         //         return tweet;
40         return jsonToTweet(tweetStr);
41     }

```

6.40.3.9 void se2xb3.data.processing.TweetProcessor.processTweetText (Tweet tweet)

Process the text of each tweet as they are received. The words are extracted from the string using regular expressions and then inserted into the word graph.

Parameters

<i>tweet</i>	a tweet
--------------	---------

```

90         {
91             // extract all words from the tweet's text using a regular expression
92             Matcher matcher = Pattern.compile(TWEET_REGEX).matcher(tweet.text);
93
94             String word = "";
95
96             while (matcher.find()) {
97                 word = matcher.group(0);
98                 // ignore words shorter than 3 characters
99                 if (word.length() < 3) continue;
100
101                 graphWord(word, tweet);
102             }
103         }
104     }

```

6.40.3.10 void se2xb3.data.processing.TweetProcessor.resetData ()

Reset all data.

```

146         {
147             graph.resetData(); // = new WordGraph();
148         }

```

6.40.4 Member Data Documentation

6.40.4.1 WordGraph se2xb3.data.processing.TweetProcessor.graph = new WordGraph() [package]

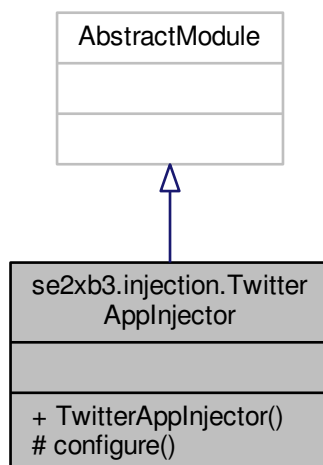
6.40.4.2 ObjectMapper se2xb3.data.processing.TweetProcessor.objectMapper = new ObjectMapper() [package]

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

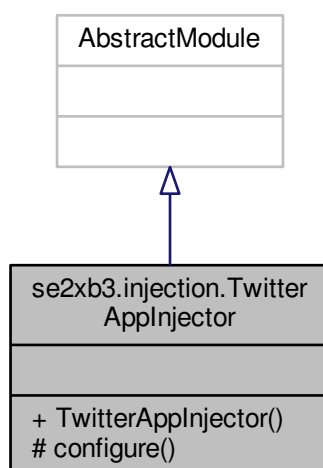
- [TweetProcessor.java](#)

6.41 se2xb3.injection.TwitterAppInjector Class Reference

Inheritance diagram for se2xb3.injection.TwitterAppInjector:



Collaboration diagram for se2xb3.injection.TwitterAppInjector:



Public Member Functions

- [TwitterAppInjector \(\)](#)

Protected Member Functions

- void [configure](#) ()

6.41.1 Detailed Description

App injector module.

Author

Dawson Myers

Version

1.0

Since

3/26/2017

6.41.2 Constructor & Destructor Documentation

6.41.2.1 `se2xb3.injection.TwitterAppInjector.TwitterAppInjector ()`

```
14 {}
```

6.41.3 Member Function Documentation

6.41.3.1 `void se2xb3.injection.TwitterAppInjector.configure ()` `[protected]`

Configures a Binder via the exposed methods.

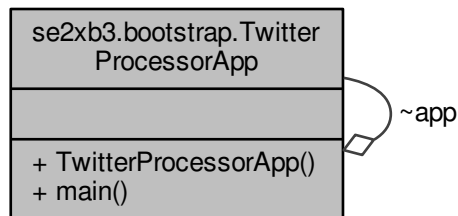
```
20                                     {
21
22     }
```

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

- [TwitterAppInjector.java](#)

6.42 se2xb3.bootstrap.TwitterProcessorApp Class Reference

Collaboration diagram for se2xb3.bootstrap.TwitterProcessorApp:



Public Member Functions

- [TwitterProcessorApp](#) ()

Static Public Member Functions

- static void [main](#) (String[] args)

Static Package Attributes

- static [TwitterProcessorApp](#) app

6.42.1 Detailed Description

Bootstrap the app

Author

Dawson Myers

Version

1.0

Since

3/26/2017

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `se2xb3.bootstrap.TwitterProcessorApp.TwitterProcessorApp ()`

```
15 {}
```

6.42.3 Member Function Documentation

6.42.3.1 `static void se2xb3.bootstrap.TwitterProcessorApp.main (String[] args)` `[static]`

Bootstrap the app

Parameters

<i>args</i>	input args
-------------	------------

```
23
24
25
26
27
28
    {
        ApplicationController app = ApplicationController.getInstance();
        TestDataSource.waitForEver();
        //TestDataSource.main(new String[]{"a"});
    }
```

6.42.4 Member Data Documentation

6.42.4.1 TwitterProcessorApp se2xb3.bootstrap.TwitterProcessorApp.app [static],[package]

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

- [TwitterProcessorApp.java](#)

6.43 se2xb3.data.models.User Class Reference

Collaboration diagram for se2xb3.data.models.User:



Public Member Functions

- [User](#) ()
- [User](#) (JsonNode n)

Public Attributes

- long [id](#)
- String [id_str](#)
- boolean [verified](#)
- int [followers_count](#)
- String [created_at](#)
- boolean [geo_enabled](#)
- String [profile_background_color](#)

Package Attributes

- String [name](#)
- String [screen_name](#)
- String [location](#)
- String [url](#)
- String [description](#)
- int [friends_count](#)
- int [listed_count](#)
- int [favourites_count](#)
- int [statuses_count](#)
- int [utc_offset](#)
- String [time_zone](#)
- String [lang](#)
- boolean [contributors_enabled](#)
- boolean [is_translator](#)
- boolean [profile_background_tile](#)
- boolean [profile_use_background_image](#)
- boolean [default_profile](#)
- boolean [default_profile_image](#)
- String [profile_background_image_url](#)
- String [profile_background_image_url_https](#)
- String [profile_link_color](#)
- String [profile_sidebar_border_color](#)
- String [profile_sidebar_fill_color](#)
- String [profile_text_color](#)
- String [profile_image_url](#)
- String [profile_image_url_https](#)
- String [profile_banner_url](#)
- String [following](#)
- String [follow_request_sent](#)
- String [notifications](#)

6.43.1 Detailed Description

Author

Dawson

Version

1.0

Since

2/23/2017

6.43.2 Constructor & Destructor Documentation

6.43.2.1 se2xb3.data.models.User.User ()

```
25 {}
```

6.43.2.2 se2xb3.data.models.User.User (JsonNode n)

```
27         {
28
29         id = n.findValue("id").asLong();
30         followers_count = n.findValue("followers_count").asInt();
31         favourites_count = n.findValue("favourites_count").asInt();
32         statuses_count = n.findValue("statuses_count").asInt();
33         utc_offset = n.findValue("utc_offset").asInt();
34         friends_count = n.findValue("friends_count").asInt();
35         listed_count = n.findValue("listed_count").asInt();
36
37         id_str = n.findValue("id_str").asText();
38         name = n.findValue("name").asText();
39         screen_name = n.findValue("screen_name").asText();
40         location = n.findValue("location").asText();
41         description = n.findValue("description").asText();
42         created_at = n.findValue("created_at").asText();
43         time_zone = n.findValue("time_zone").asText();
44         lang = n.findValue("lang").asText();
45
46     }
```

6.43.3 Member Data Documentation

6.43.3.1 boolean se2xb3.data.models.User.contributors_enabled [package]

6.43.3.2 String se2xb3.data.models.User.created_at

6.43.3.3 boolean se2xb3.data.models.User.default_profile [package]

6.43.3.4 boolean se2xb3.data.models.User.default_profile_image [package]

6.43.3.5 String se2xb3.data.models.User.description [package]

- 6.43.3.6 `int se2xb3.data.models.User.favourites_count` [package]
- 6.43.3.7 `String se2xb3.data.models.User.follow_request_sent` [package]
- 6.43.3.8 `int se2xb3.data.models.User.followers_count`
- 6.43.3.9 `String se2xb3.data.models.User.following` [package]
- 6.43.3.10 `int se2xb3.data.models.User.friends_count` [package]
- 6.43.3.11 `boolean se2xb3.data.models.User.geo_enabled`
- 6.43.3.12 `long se2xb3.data.models.User.id`
- 6.43.3.13 `String se2xb3.data.models.User.id_str`
- 6.43.3.14 `boolean se2xb3.data.models.User.is_translator` [package]
- 6.43.3.15 `String se2xb3.data.models.User.lang` [package]
- 6.43.3.16 `int se2xb3.data.models.User.listed_count` [package]
- 6.43.3.17 `String se2xb3.data.models.User.location` [package]
- 6.43.3.18 `String se2xb3.data.models.User.name` [package]
- 6.43.3.19 `String se2xb3.data.models.User.notifications` [package]
- 6.43.3.20 `String se2xb3.data.models.User.profile_background_color`
- 6.43.3.21 `String se2xb3.data.models.User.profile_background_image_url` [package]
- 6.43.3.22 `String se2xb3.data.models.User.profile_background_image_url_https` [package]
- 6.43.3.23 `boolean se2xb3.data.models.User.profile_background_tile` [package]
- 6.43.3.24 `String se2xb3.data.models.User.profile_banner_url` [package]
- 6.43.3.25 `String se2xb3.data.models.User.profile_image_url` [package]
- 6.43.3.26 `String se2xb3.data.models.User.profile_image_url_https` [package]
- 6.43.3.27 `String se2xb3.data.models.User.profile_link_color` [package]
- 6.43.3.28 `String se2xb3.data.models.User.profile_sidebar_border_color` [package]

6.43.3.29 String se2xb3.data.models.User.profile_sidebar_fill_color [package]

6.43.3.30 String se2xb3.data.models.User.profile_text_color [package]

6.43.3.31 boolean se2xb3.data.models.User.profile_use_background_image [package]

6.43.3.32 String se2xb3.data.models.User.screen_name [package]

6.43.3.33 int se2xb3.data.models.User.statuses_count [package]

6.43.3.34 String se2xb3.data.models.User.time_zone [package]

6.43.3.35 String se2xb3.data.models.User.url [package]

6.43.3.36 int se2xb3.data.models.User.utc_offset [package]

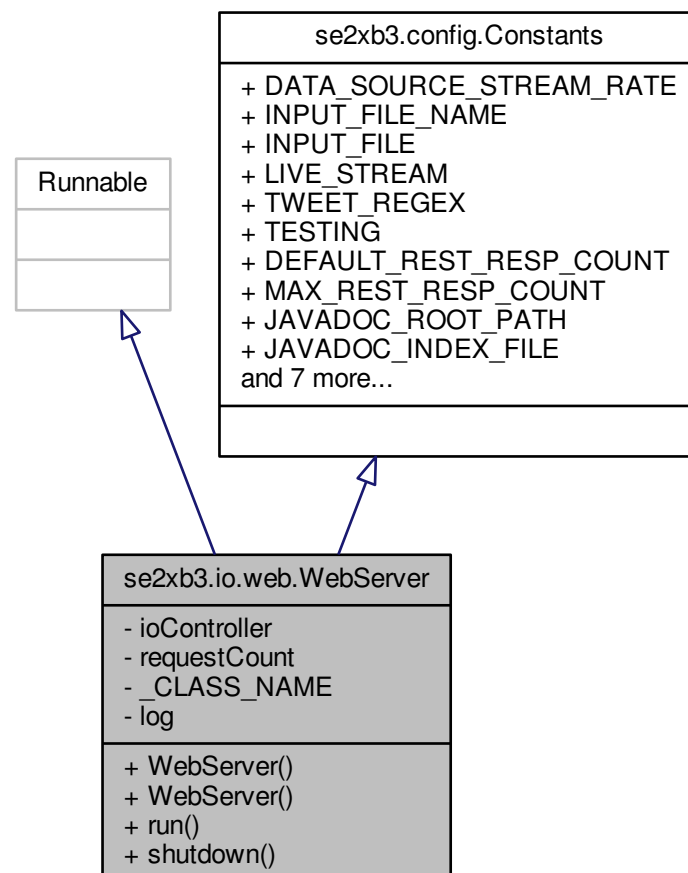
6.43.3.37 boolean se2xb3.data.models.User.verified

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

- [User.java](#)

6.44 se2xb3.io.web.WebServer Class Reference

Inheritance diagram for se2xb3.io.web.WebServer:



[illegible]

- `WebServer ()`
- `WebServer (IOController io)`
- `void run ()`
- `void shutdown ()`

- `IOController ioController`
- `long requestCount = 0`

- static final String `_CLASS_NAME` = `WebServer.class.getSimpleName()`
- static final Logger `log` = `LoggerFactory.getLogger(_CLASS_NAME)`

6.44.1 Detailed Description

Author

Version

Since

Generated by Doxygen

6.44.2 Constructor & Destructor Documentation

6.44.2.1 `se2xb3.io.web.WebServer.WebServer ()`

```
26 {}
```

6.44.2.2 `se2xb3.io.web.WebServer.WebServer (IOController io)`

```
27 {ioController = io;}
```

6.44.3 Member Function Documentation

6.44.3.1 `void se2xb3.io.web.WebServer.run ()`

This method is called in another thread via `executer` service. It starts the server, configures the base directory by finding the folder containing the marker file (`.ratpack`) and sets it to the root for the web server. It then configures the handler chain with the following:

```
1. .files(f -> f.dir(""))
```

Set the root directory to be served statically

```
2. .all(new CORSHandler())
```

Configuring all requests to first go through the `CORSHandler` which adds the following headers to all requests:

- `headers.set("Access-Control-Allow-Origin", "*");`
- `headers.set("Access-Control-Allow-Headers", "x-requested-with, origin, content-type, accept");`
- `headers.set("Access-Control-Allow-Methods", "GET,PUT,POST,DELETE");`

This is required so that cross-origin REST requests can be made. Otherwise the client browser would block the request for security reasons.

```
3. .prefix("", nested -> nested.fileSystem("", Chain::files))
```

Configure all files and folders (recursively) to be served at the web server root.

```
4. .path("", ctx -> ctx.render(ctx.file("index.html")))
```

This path handler handles requests with an empty path (i.e. `example.com`) and serves the `index.html` file which bootstraps the AngularJs web app.

```
5. .path("trends/:type?:count?", new TrendHandler(ioController))
```

All REST request matching the path patten `trends/:type?:count?` are delegated to the `TrendHandler`. The optional query parameter for the `trends` REST resource are `:type?` and `:count?` (a prefix of `:` indicates a parameter and a suffix of `?` means it's optional). Request can be made in the following ways:

- `trends/{count}`

Type parameter is empty. The response will include the top **count** overall word, hashtag, and user mentions as a JSON array containing 3 arrays (one for each trend). If the the request is just `trends/` without the **count** parameter empty, then the a default value of 10 is used.

- **trends/{type}/{count}**

The response will include the top **count** (an integer value) trends of the type **type**. Where **type** is either overall word, hashtag, or user mentions as a JSON array. If the the request is just *trends/{type}/* without the **count** parameter empty, then the a default value of 10 is used.

```

100         {
101             try {
102                 RatpackServer
103                     .start(server -> {
104                         server.serverConfig(serverConfigBuilder -> serverConfigBuilder
105                             .baseDir(BaseDir.find())
106                             .env()
107                             .sysProps())
108                         .handlers(chain -> chain
109                             // statically serves all files in the dir containing the .ratpack file
110                             .files(f -> f.dir(""))
111                             // apply the CORS handler to all request to allo cross-site REST requests
112                             .all(new CORSHandler())
113                             .all(RequestLogger.ncsa())
114                             .all(req -> {
115                                 requestCount++;
116                                 if (requestCount % 100 == 0) {
117                                     log.info("Request count = " + requestCount);
118                                 }
119                             })
120                             // statically serves all files in the dir containing the .ratpack file
121                             .prefix("", nested -> nested.fileSystem("", Chain::files))
122                             .path("", ctx -> ctx.render(ctx.file(
123                                 WEB_APP_ROOT_FILE)))
124                             .path("", ctx -> ctx.render(ctx.file("index.html")))
125                             .path(JAVADOC_ROOT_PATH, ctx -> ctx.render(ctx.file(
126                                 JAVADOC_INDEX_FILE)))
127                             .path("docs", ctx -> ctx.render(ctx.file("docs/index.html")))
128                             .files(f -> f.dir("dist").indexFiles("index.html"))
129                             // delegate all trend/* REST requests to the trend handler
130                             // the :type?:count? path tokens are optional
131                             // default path /trends will get a response with the top ten of the
132                             // all, hashtag, and user mention trends
133                             .path(REST_TREND_RESOURCE_PATH_REGEX, new
134                                 TrendHandler(ioController));
135                             .path("trends/:type?:count?", new TrendHandler(ioController));
136                         });
137                     } catch (Exception e) {
138                         e.printStackTrace();
139                     }

```

6.44.3.2 void se2xb3.io.web.WebServer.shutdown ()

Shutdown the web server.

```

145         {
146
147     }

```

6.44.4 Member Data Documentation

6.44.4.1 final String se2xb3.io.web.WebServer._CLASS_NAME = WebServer.class.getSimpleName() [static], [private]

6.44.4.2 IOController se2xb3.io.web.WebServer.ioController [private]

6.44.4.3 final Logger se2xb3.io.web.WebServer.log = LoggerFactory.getLogger(_CLASS_NAME) [static], [private]

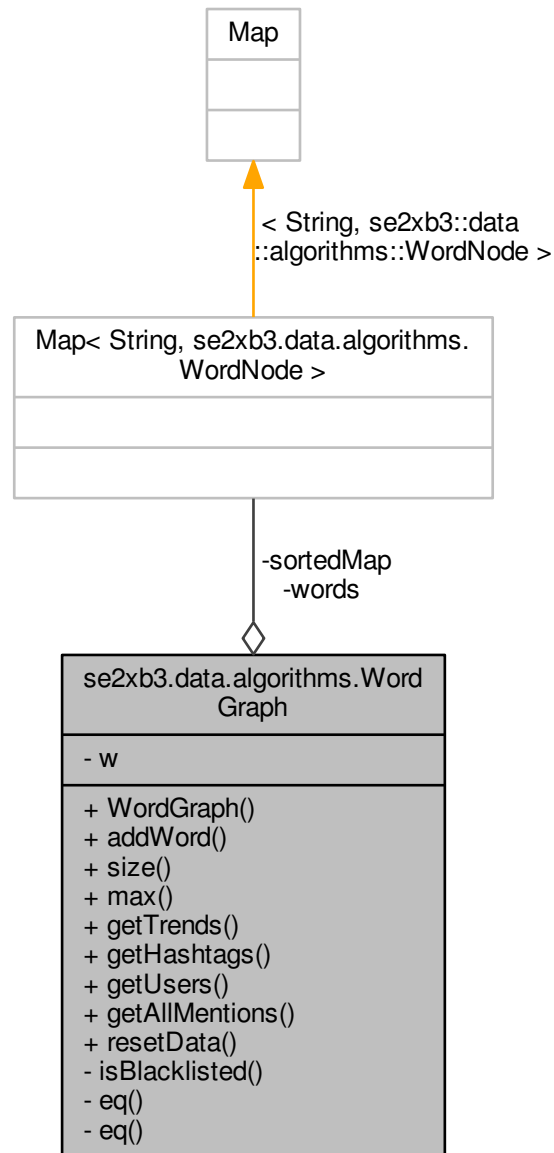
6.44.4.4 long se2xb3.io.web.WebServer.requestCount = 0 [private]

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

- [WebServer.java](#)

6.45 se2xb3.data.algorithms.WordGraph Class Reference

Collaboration diagram for se2xb3.data.algorithms.WordGraph:



Public Member Functions

- [WordGraph](#) ()
- void [addWord](#) (String word, [Tweet](#) tweet)
- int [size](#) ()
- [WordNode](#) [max](#) ()
- void [getTrends](#) ()

- [WordNode\[\]](#) [getHashtags](#) ()
- [WordNode\[\]](#) [getUsers](#) ()
- [WordNode\[\]](#) [getAllMentions](#) ()
- void [resetData](#) ()

Private Member Functions

- boolean [isBlacklisted](#) (String s)
- boolean [eq](#) (String a)
- boolean [eq](#) (String a, String b)

Private Attributes

- Map< String, [WordNode](#) > [words](#) = new ConcurrentHashMap<>()
- Map< String, [WordNode](#) > [sortedMap](#)
- String [w](#) = ""

6.45.1 Detailed Description

A class that generates a graph of words and tweets. Each tweet is connected to all of the words that is contained in the tweet text.

Author

Dawson

Version

1.0

Since

3/10/2017

6.45.2 Constructor & Destructor Documentation

6.45.2.1 se2xb3.data.algorithms.WordGraph.WordGraph ()

```
22 {}
```

6.45.3 Member Function Documentation

6.45.3.1 void se2xb3.data.algorithms.WordGraph.addWord (String word, Tweet tweet)

```
24                                     {
25     String lowercase = word.toLowerCase();
26     if(!words.containsKey(lowercase)) words.put(lowercase, new WordNode(word,tweet));
27     else words.get(lowercase).add(tweet);
28     //     if(!words.containsKey(word)) words.put(word, new WordNode(word,tweet));
29     //     else words.get(word).add(tweet);
30 }
```

6.45.3.2 boolean se2xb3.data.algorithms.WordGraph.eq (String a) [private]

```
139 {return eq(w, a);}
```

6.45.3.3 boolean se2xb3.data.algorithms.WordGraph.eq (String a, String b) [private]

```
140
141         {
142     return a.toLowerCase().equals(b.toLowerCase());
143 }
```

6.45.3.4 WordNode [] se2xb3.data.algorithms.WordGraph.getAllMentions ()

```
107
108 //         {
109 //     return words
110 //         .entrySet()
111 //         .stream()
112 //         .toArray(WordNode[]::new);
113
114 // convert to list and filter out irrelevant words
115 List<WordNode> list = words.values()
116     .stream()
117     .filter(w -> {
118         String s = w.id;
119         if(s.startsWith("#") || s.startsWith("@")) return true;
120         if(s.length() < 5) return false;
121         if(isBlacklisted(s)) return false;
122         return true;
123     })
124     .collect(Collectors.toList());
125
126 WordNode[] a = new WordNode[list.size()];
127 for (int i = 0; i < list.size(); i++) {
128     a[i] = list.get(i);
129 }
130 return a;
131 }
```

6.45.3.5 WordNode [] se2xb3.data.algorithms.WordGraph.getHashtags ()

Get array of hashtag word nodes

Returns

all words that start with a #

```
63
64         {
65 //     return words
66 //         .entrySet()
67 //         .stream()
68 //         .filter((a) -> !a.getKey().startsWith("#"))
69 //         .toArray(WordNode[]::new);
70 List<WordNode> list = words.values()
71     .stream()
72     .filter((a) -> a.id.startsWith("#"))
73     .collect(Collectors.toList());
74
75 WordNode[] a = new WordNode[list.size()];
76 for (int i = 0; i < list.size(); i++) {
77     a[i] = list.get(i);
78 }
79 return a;
80
81 }
```


6.45.3.6 void se2xb3.data.algorithms.WordGraph.getTrends ()

```

53         {
54
55     }

```

6.45.3.7 WordNode [] se2xb3.data.algorithms.WordGraph.getUsers ()

Get array of user mentions as an array of word nodes

Returns

all words that start with a #

```

87         {
88
89     //         return words
90     //         .entrySet()
91     //         .stream()
92     //         .filter((a) -> !a.getKey().startsWith("@"))
93     //         .toArray(WordNode[]::new);
94
95     List<WordNode> list = words.values()
96                             .stream()
97                             .filter((a) -> a.id.startsWith("@"))
98                             .collect(Collectors.toList());
99
100     WordNode[] a = new WordNode[list.size()];
101     for (int i = 0; i < list.size(); i++) {
102         a[i] = list.get(i);
103     }
104     return a;
105 }

```

6.45.3.8 boolean se2xb3.data.algorithms.WordGraph.isBlacklisted (String s) [private]

```

133         {
134     w = s;
135     return eq("their") || eq("every") || eq("first") || eq("there") ||
eq("those") || eq("them")
136         || eq("they") || eq("These");// || eq("")
137 }

```

6.45.3.9 WordNode se2xb3.data.algorithms.WordGraph.max ()

```

36         {
37
38     sortedMap =
39     words.entrySet().stream()
40     .sorted((a,b)-> a.getValue().compareTo(b.getValue()))
41     .collect(Collectors.toMap(Map.Entry::getKey,Map.Entry::getValue));
42
43     WordNode[] wn = words.values().stream()
44     .sorted((a,b)-> b.compareTo(a)).toArray(WordNode[]::new);
45
46     WordNode[] wn1 = words.values().stream().toArray(WordNode[]::new);
47
48     //WordNode[] wn = (WordNode[]) words.values().toArray();// Merge.sort();
49
50     return sortedMap.get(1);
51 }

```

6.45.3.10 void se2xb3.data.algorithms.WordGraph.resetData ()

```

144         {
145             words.clear();
146         }

```

6.45.3.11 int se2xb3.data.algorithms.WordGraph.size ()

```

32         {
33             return words.size();
34         }

```

6.45.4 Member Data Documentation

6.45.4.1 Map<String, WordNode> se2xb3.data.algorithms.WordGraph.sortedMap [private]

6.45.4.2 String se2xb3.data.algorithms.WordGraph.w = "" [private]

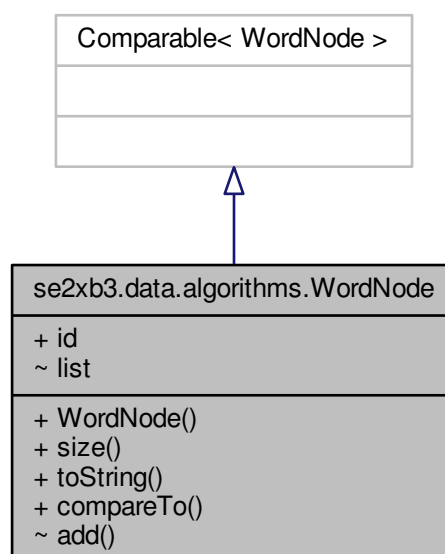
6.45.4.3 Map<String, WordNode> se2xb3.data.algorithms.WordGraph.words = new ConcurrentHashMap<>()
[private]

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

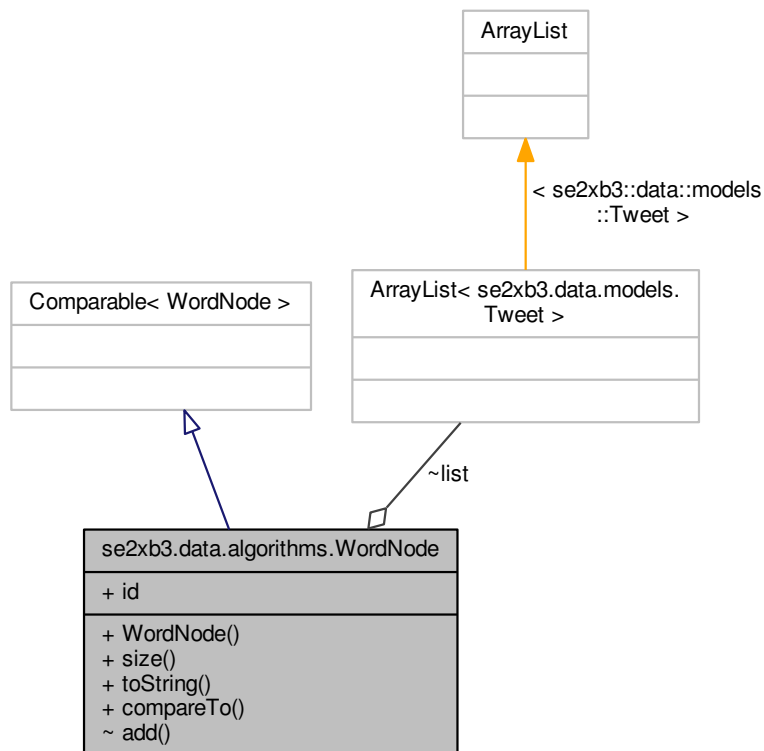
- [WordGraph.java](#)

6.46 se2xb3.data.algorithms.WordNode Class Reference

Inheritance diagram for se2xb3.data.algorithms.WordNode:



Collaboration diagram for se2xb3.data.algorithms.WordNode:



Public Member Functions

- [WordNode](#) (String wordId, [Tweet](#) tweet)
- int [size](#) ()
- String [toString](#) ()
- int [compareTo](#) ([WordNode](#) other)

Public Attributes

- String [id](#)

Package Functions

- void [add](#) ([Tweet](#) tweet)

Package Attributes

- ArrayList< [Tweet](#) > [list](#) = new ArrayList<>()

6.46.1 Detailed Description

A class for word nodes that are used to build the word graph.

Author

Dawson

Version

1.0

Since

3/10/2017

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `se2xb3.data.algorithms.WordNode.WordNode (String wordId, Tweet tweet)`

[WordNode](#) constructor

Parameters

<i>wordId</i>	the word
<i>tweet</i>	the tweet that uses it

```

24                                     {
25         id = wordId;
26         list.add(tweet);
27     }
```

6.46.3 Member Function Documentation

6.46.3.1 `void se2xb3.data.algorithms.WordNode.add (Tweet tweet)` [package]

Adds a reference to a tweet that uses this word.

Parameters

<i>tweet</i>	a tweet that uses the word
--------------	----------------------------

```

34                                     {
35         list.add(tweet);
36     }
```

6.46.3.2 int se2xb3.data.algorithms.WordNode.compareTo (WordNode other)

Compare another word node to this word node. The comparison is configured to sort nodes in descending order according to their size.

Parameters

<i>other</i>	a node to compare to
--------------	----------------------

Returns

-1 if this node is larger, 0 if the same, and 1 if smaller

```

64         {
65
66         // descending order
67         return this.size() > other.size() ? -1 : this.size() == other.size() ? 0 : 1;
68         // ascending order
69         // return this.size() > other.size() ? 1: this.size() == other.size() ? 0: -1;
70     }

```

6.46.3.3 int se2xb3.data.algorithms.WordNode.size ()

Gets the number of tweets that uses this word.

Returns

the number of references to tweets that use this word

```

43         {
44         return list.size();
45     }

```

6.46.3.4 String se2xb3.data.algorithms.WordNode.toString ()

The string representation of this node.

Returns

a string containing the id and size of this node

```

52         {
53         return id + " : " + size();
54     }

```

6.46.4 Member Data Documentation

6.46.4.1 String se2xb3.data.algorithms.WordNode.id

6.46.4.2 ArrayList<Tweet> se2xb3.data.algorithms.WordNode.list = new ArrayList<>() [package]

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

- [WordNode.java](#)

Chapter 7

File Documentation

7.1 AppConfig.java File Reference

Classes

- interface [se2xb3.config.AppConfig](#)

Packages

- package [se2xb3.config](#)

7.2 AppController.java File Reference

Classes

- class [se2xb3.control.AppController](#)

Packages

- package [se2xb3.control](#)

7.3 AsyncMessageLooper.java File Reference

Classes

- class [se2xb3.io.AsyncMessageLooper< M >](#)

Packages

- package [se2xb3.io](#)

7.4 Constants.java File Reference

Classes

- interface [se2xb3.config.Constants](#)

Packages

- package [se2xb3.config](#)

7.5 Controller.java File Reference

Classes

- class [se2xb3.control.Controller](#)

Packages

- package [se2xb3.control](#)

7.6 CORSHandler.java File Reference

Classes

- class [se2xb3.io.web.handlers.CORSHandler](#)

Packages

- package [se2xb3.io.web.handlers](#)

7.7 DataController.java File Reference

Classes

- class [se2xb3.data.DataController](#)

Packages

- package [se2xb3.data](#)

7.8 DataHandler.java File Reference

Classes

- class [se2xb3.data.DataHandler](#)

Packages

- package [se2xb3.data](#)

7.9 DataSource.java File Reference

Classes

- class [se2xb3.io.source.DataSource](#)

Packages

- package [se2xb3.io.source](#)

7.10 EventBus.java File Reference

Classes

- class [se2xb3.control.EventBus](#)

Packages

- package [se2xb3.control](#)

7.11 FileDataSource.java File Reference

Classes

- class [se2xb3.io.source.FileDataSource](#)

Packages

- package [se2xb3.io.source](#)

7.12 FileDataSourceTest.java File Reference

Classes

- class [se2xb3.tests.FileDataSourceTest](#)

Packages

- package [se2xb3.tests](#)

7.13 FileFinder.java File Reference

Classes

- class [se2xb3.io.files.FileFinder](#)

Packages

- package [se2xb3.io.files](#)

7.14 GraphStrategy.java File Reference

Classes

- class [se2xb3.data.algorithms.GraphStrategy](#)

Packages

- package [se2xb3.data.algorithms](#)

7.15 IDataSource.java File Reference

Classes

- interface [se2xb3.io.source.IDataSource](#)

Packages

- package [se2xb3.io.source](#)

7.16 IMessageQueue.java File Reference

Classes

- interface [se2xb3.io.IMessageQueue< M >](#)

Packages

- package [se2xb3.io](#)

7.17 IMessageReceiver.java File Reference

Classes

- interface [se2xb3.io.IMessageReceiver< M >](#)

Packages

- package [se2xb3.io](#)

7.18 IOController.java File Reference

Classes

- class [se2xb3.io.IOController](#)

Packages

- package [se2xb3.io](#)

7.19 IProcessor.java File Reference

Classes

- interface [se2xb3.data.processing.IProcessor< T >](#)

Packages

- package [se2xb3.data.processing](#)

7.20 ISort.java File Reference

Classes

- interface [se2xb3.data.algorithms.sort.ISort](#)

Packages

- package [se2xb3.data.algorithms.sort](#)

7.21 JsonParser.java File Reference

Classes

- class [se2xb3.io.web.JsonParser](#)
- class [se2xb3.io.web.Trend](#)

Packages

- package [se2xb3.io.web](#)

7.22 MaxPQ.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >](#)
- class [se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator](#)

Packages

- package [se2xb3.data.algorithms.sort.textbook](#)

7.23 Merge.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.textbook.Merge](#)

Packages

- package [se2xb3.data.algorithms.sort.textbook](#)

7.24 MergeBU.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.textbook.MergeBU](#)

Packages

- package [se2xb3.data.algorithms.sort.textbook](#)

7.25 MergeSortStrategy.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.MergeSortStrategy](#)

Packages

- package [se2xb3.data.algorithms.sort](#)

7.26 MergeX.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.textbook.MergeX](#)

Packages

- package [se2xb3.data.algorithms.sort.textbook](#)

7.27 MessageHandler.java File Reference

Classes

- class [se2xb3.io.MessageHandler](#)

Packages

- package [se2xb3.io](#)

7.28 MinPQ.java File Reference

Classes

- class [se2xb3.data.algorithms.sort.textbook.MinPQ< Key >](#)
- class [se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.Heaplterator](#)

Packages

- package [se2xb3.data.algorithms.sort.textbook](#)

7.29 Place.java File Reference

Classes

- class [se2xb3.data.models.Place](#)
- class [se2xb3.data.models.BoundingBox](#)

Packages

- package [se2xb3.data.models](#)

7.30 SearchStrategy.java File Reference

Classes

- class [se2xb3.data.algorithms.SearchStrategy](#)

Packages

- package [se2xb3.data.algorithms](#)

7.31 SortStrategy.java File Reference

Classes

- class [se2xb3.data.algorithms.SortStrategy](#)

Packages

- package [se2xb3.data.algorithms](#)

7.32 TestDataSource.java File Reference

Classes

- class [se2xb3.tests.TestDataSource](#)

Packages

- package [se2xb3.tests](#)

7.33 TrendHandler.java File Reference

Classes

- class [se2xb3.io.web.handlers.TrendHandler](#)

Packages

- package [se2xb3.io.web.handlers](#)

7.34 Tweet.java File Reference

Classes

- class [se2xb3.data.models.Tweet](#)
- class [se2xb3.data.models.TweetOld](#)

Packages

- package [se2xb3.data.models](#)

7.35 TweetProcessor.java File Reference

Classes

- class [se2xb3.data.processing.TweetProcessor](#)

Packages

- package [se2xb3.data.processing](#)

7.36 TwitterAppInjector.java File Reference

Classes

- class [se2xb3.injection.TwitterAppInjector](#)

Packages

- package [se2xb3.injection](#)

7.37 TwitterProcessorApp.java File Reference

Classes

- class [se2xb3.bootstrap.TwitterProcessorApp](#)

Packages

- package [se2xb3.bootstrap](#)

7.38 User.java File Reference

Classes

- class [se2xb3.data.models.User](#)

Packages

- package [se2xb3.data.models](#)

7.39 WebServer.java File Reference

Classes

- class [se2xb3.io.web.WebServer](#)

Packages

- package [se2xb3.io.web](#)

7.40 WordGraph.java File Reference

Classes

- class [se2xb3.data.algorithms.WordGraph](#)

Packages

- package [se2xb3.data.algorithms](#)

7.41 WordNode.java File Reference

Classes

- class [se2xb3.data.algorithms.WordNode](#)

Packages

- package [se2xb3.data.algorithms](#)

Index

- `_CLASS_NAME`
 - `se2xb3::io::source::FileDataSource`, [53](#)
 - `se2xb3::io::web::WebServer`, [155](#)
 - `se2xb3::io::web::handlers::CORSHandler`, [30](#)
 - `se2xb3::io::web::handlers::TrendHandler`, [129](#)
- `add`
 - `se2xb3::data::algorithms::WordNode`, [162](#)
- `addWord`
 - `se2xb3::data::algorithms::WordGraph`, [157](#)
- `app`
 - `se2xb3::bootstrap::TwitterProcessorApp`, [147](#)
 - `se2xb3::control::AppController`, [18](#)
 - `se2xb3::data::DataController`, [36](#)
 - `se2xb3::io::IOController`, [72](#)
- `AppConfig.java`, [165](#)
- `AppController`
 - `se2xb3::control::AppController`, [16](#)
- `AppController.java`, [165](#)
- `AsyncMessageLooper`
 - `se2xb3::io::AsyncMessageLooper`, [20](#)
- `AsyncMessageLooper.java`, [165](#)
- `attributes`
 - `se2xb3::data::models::BoundingBox`, [22](#)
- `blockingQueue`
 - `se2xb3::io::AsyncMessageLooper`, [21](#)
 - `se2xb3::tests::TestDataSource`, [122](#)
- `bounding_box`
 - `se2xb3::data::models::Place`, [116](#)
- `bus`
 - `se2xb3::control::EventBus`, [49](#)
- `byValue`
 - `se2xb3::data::DataHandler`, [42](#)
- `CORSHandler.java`, [166](#)
- `CUTOFF`
 - `se2xb3::data::algorithms::sort::textbook::MergeX`, [102](#)
- `checkNull`
 - `se2xb3::data::models::TweetOld`, [134](#)
- `checkNullLong`
 - `se2xb3::data::models::TweetOld`, [134](#)
- `comparator`
 - `se2xb3::data::algorithms::sort::textbook::MaxPQ`, [88](#)
 - `se2xb3::data::algorithms::sort::textbook::MinPQ`, [114](#)
- `compareTo`
 - `se2xb3::data::algorithms::WordNode`, [162](#)
- `configure`
 - `se2xb3::injection::TwitterApplInjector`, [144](#)
- `Constants.java`, [166](#)
- `contributors_enabled`
 - `se2xb3::data::models::User`, [149](#)
- `Controller`
 - `se2xb3::control::Controller`, [27](#)
- `controller`
 - `se2xb3::io::source::DataSource`, [48](#)
- `Controller.java`, [166](#)
- `coordinates`
 - `se2xb3::data::models::BoundingBox`, [22](#)
 - `se2xb3::data::models::TweetOld`, [135](#)
- `copy`
 - `se2xb3::data::algorithms::sort::textbook::MaxPQ`↔
↔`HeapIterator`, [62](#)
 - `se2xb3::data::algorithms::sort::textbook::MinPQ`↔
↔`HeapIterator`, [60](#)
- `count`
 - `se2xb3::io::web::Trend`, [123](#)
- `country`
 - `se2xb3::data::models::Place`, [116](#)
- `country_code`
 - `se2xb3::data::models::Place`, [116](#)
- `created_at`
 - `se2xb3::data::models::TweetOld`, [135](#)
 - `se2xb3::data::models::User`, [149](#)
- `DATA_SOURCE_STREAM_RATE`
 - `se2xb3::config::Constants`, [24](#)
- `DEFAULT_REST_RESP_COUNT`
 - `se2xb3::config::Constants`, [24](#)
- `DataController`
 - `se2xb3::data::DataController`, [33](#)
- `dataController`
 - `se2xb3::control::AppController`, [18](#)
 - `se2xb3::data::DataHandler`, [42](#)
- `DataController.java`, [166](#)
- `DataHandler`
 - `se2xb3::data::DataHandler`, [39](#)
- `dataHandler`
 - `se2xb3::data::DataController`, [36](#)
- `DataHandler.java`, [167](#)
- `DataSource`
 - `se2xb3::io::source::DataSource`, [45](#)
- `dataSource`
 - `se2xb3::io::IOController`, [72](#)
 - `se2xb3::io::MessageHandler`, [106](#)
- `DataSource.java`, [167](#)
- `default_profile`

- se2xb3::data::models::User, 149
- default_profile_image
 - se2xb3::data::models::User, 149
- delMax
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 85
- delMin
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 111
- description
 - se2xb3::data::models::User, 149
- deserializeTweet
 - se2xb3::data::processing::TweetProcessor, 139
- doShutdown
 - se2xb3::io::AsyncMessageLooper, 21
- endTest
 - se2xb3::tests::TestDataSource, 121
- enqueue
 - se2xb3::io::AsyncMessageLooper, 20
 - se2xb3::io::IMessageQueue, 65
- eq
 - se2xb3::data::algorithms::WordGraph, 157, 158
- EventBus
 - se2xb3::control::EventBus, 49
- EventBus.java, 167
- exch
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 85
 - se2xb3::data::algorithms::sort::textbook::MergeX, 99
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 111
- executorService
 - se2xb3::control::AppController, 18
- favorite_count
 - se2xb3::data::models::TweetOld, 135
- favourites_count
 - se2xb3::data::models::User, 149
- FileDataSource
 - se2xb3::io::source::FileDataSource, 52
- FileDataSource.java, 167
- FileDataSourceTest.java, 168
- FileFinder
 - se2xb3::io::files::FileFinder, 56
- FileFinder.java, 168
- findFileByName
 - se2xb3::io::files::FileFinder, 56, 57
- follow_request_sent
 - se2xb3::data::models::User, 150
- followers_count
 - se2xb3::data::models::User, 150
- following
 - se2xb3::data::models::User, 150
- friends_count
 - se2xb3::data::models::User, 150
- full_name
 - se2xb3::data::models::Place, 116
- geo
 - se2xb3::data::models::TweetOld, 135
- geo_enabled
 - se2xb3::data::models::User, 150
- getAllMentions
 - se2xb3::data::algorithms::WordGraph, 158
 - se2xb3::data::processing::TweetProcessor, 140
- getDataController
 - se2xb3::control::AppController, 16
- getDataSource
 - se2xb3::io::MessageHandler, 105
- getExecutorService
 - se2xb3::control::AppController, 16
 - se2xb3::io::IOController, 70
 - se2xb3::io::source::DataSource, 45
- getHashtags
 - se2xb3::data::algorithms::WordGraph, 158
 - se2xb3::data::processing::TweetProcessor, 140
- getIOController
 - se2xb3::control::AppController, 16
- getInstance
 - se2xb3::control::AppController, 16
- getQueue
 - se2xb3::io::source::DataSource, 45
- getText
 - se2xb3::data::models::TweetOld, 135
- getTrendingHashtags
 - se2xb3::data::DataController, 33
 - se2xb3::data::DataHandler, 39
 - se2xb3::io::IOController, 70
 - se2xb3::io::web::handlers::TrendHandler, 126
- getTrendingUsers
 - se2xb3::data::DataController, 33
 - se2xb3::data::DataHandler, 40
 - se2xb3::io::IOController, 70
 - se2xb3::io::web::handlers::TrendHandler, 126
- getTrendingWords
 - se2xb3::data::DataController, 34
 - se2xb3::data::DataHandler, 40
 - se2xb3::io::IOController, 71
 - se2xb3::io::web::handlers::TrendHandler, 127
- getTrends
 - se2xb3::data::DataController, 34
 - se2xb3::data::DataHandler, 41
 - se2xb3::data::algorithms::WordGraph, 158
 - se2xb3::io::IOController, 71
 - se2xb3::io::web::handlers::TrendHandler, 127
- getTweetList
 - se2xb3::data::DataController, 35
- getUrl
 - se2xb3::io::source::DataSource, 46
- getUsers
 - se2xb3::data::algorithms::WordGraph, 159
 - se2xb3::data::processing::TweetProcessor, 140
- graph
 - se2xb3::data::processing::TweetProcessor, 142
- GraphStrategy
 - se2xb3::data::algorithms::GraphStrategy, 58

- graphStrategy
 - se2xb3::data::DataHandler, 42
- GraphStrategy.java, 168
- graphWord
 - se2xb3::data::processing::TweetProcessor, 140
- greater
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 112
- handle
 - se2xb3::io::web::handlers::CORSHandler, 30
 - se2xb3::io::web::handlers::TrendHandler, 128
- hasNext
 - se2xb3::data::algorithms::sort::textbook::MaxPQ↔::HeapIterator, 62
 - se2xb3::data::algorithms::sort::textbook::MinPQ↔::HeapIterator, 60
- HeapIterator
 - se2xb3::data::algorithms::sort::textbook::MaxPQ↔::HeapIterator, 62
 - se2xb3::data::algorithms::sort::textbook::MinPQ↔::HeapIterator, 60
- hostname
 - se2xb3::config::AppConfig, 14
- IDataSource.java, 168
- IF_NULL_INT
 - se2xb3::data::models::TweetOld, 135
- IF_NULL_LONG
 - se2xb3::data::models::TweetOld, 135
- IF_NULL_STR
 - se2xb3::data::models::TweetOld, 135
- IMessageQueue.java, 169
- IMessageReceiver.java, 169
- INPUT_FILE_NAME
 - se2xb3::config::Constants, 25
- INPUT_FILE
 - se2xb3::config::Constants, 24
- IOController
 - se2xb3::io::IOController, 69
- IOController.java, 169
- IProcessor.java, 169
- ISort.java, 170
- id
 - se2xb3::data::algorithms::WordNode, 163
 - se2xb3::data::models::Place, 116
 - se2xb3::data::models::TweetOld, 135
 - se2xb3::data::models::User, 150
- id_str
 - se2xb3::data::models::Tweet, 131
 - se2xb3::data::models::TweetOld, 135
 - se2xb3::data::models::User, 150
- in_reply_to_screen_name
 - se2xb3::data::models::TweetOld, 135
- in_reply_to_status_id
 - se2xb3::data::models::TweetOld, 135
- in_reply_to_status_id_str
 - se2xb3::data::models::TweetOld, 136
- in_reply_to_user_id
 - se2xb3::data::models::TweetOld, 136
- in_reply_to_user_id_str
 - se2xb3::data::models::TweetOld, 136
- indexSort
 - se2xb3::data::algorithms::sort::textbook::Merge, 90
- init
 - se2xb3::io::IOController, 72
- insert
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 86
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 112
- insertionSort
 - se2xb3::data::algorithms::sort::textbook::MergeX, 99
- ioController
 - se2xb3::control::AppController, 18
 - se2xb3::io::web::WebServer, 155
 - se2xb3::io::web::handlers::TrendHandler, 129
- is_translator
 - se2xb3::data::models::User, 150
- isBlacklisted
 - se2xb3::data::algorithms::WordGraph, 159
- isDone
 - se2xb3::tests::TestDataSource, 122
- isEmpty
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 86
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 112
- isMaxHeap
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 86
- isMinHeap
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 112, 113
- isNumber
 - se2xb3::io::web::handlers::TrendHandler, 129
- isSorted
 - se2xb3::data::algorithms::sort::textbook::Merge, 91
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 94
 - se2xb3::data::algorithms::sort::textbook::MergeX, 99, 100
- item
 - se2xb3::io::web::Trend, 123
- iterator
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 87
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 113
- JAVADOC_INDEX_FILE
 - se2xb3::config::Constants, 25
- JAVADOC_ROOT_PATH
 - se2xb3::config::Constants, 25
- JsonParser
 - se2xb3::io::web::JsonParser, 77
- JsonParser.java, 170

- jsonStringToTree
 - se2xb3::data::processing::TweetProcessor, 141
- jsonToTweet
 - se2xb3::data::processing::TweetProcessor, 141
- LIVE_STREAM
 - se2xb3::config::Constants, 25
- lang
 - se2xb3::data::models::TweetOld, 136
 - se2xb3::data::models::User, 150
- less
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 87
 - se2xb3::data::algorithms::sort::textbook::Merge, 91
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 94
 - se2xb3::data::algorithms::sort::textbook::MergeX, 100
- list
 - se2xb3::data::algorithms::WordNode, 163
- listed_count
 - se2xb3::data::models::User, 150
- location
 - se2xb3::data::models::User, 150
- lock
 - se2xb3::tests::FileDataSourceTest, 55
 - se2xb3::tests::TestDataSource, 122
- log
 - se2xb3::io::source::FileDataSource, 53
 - se2xb3::io::web::WebServer, 155
 - se2xb3::io::web::handlers::CORSHandler, 30
 - se2xb3::io::web::handlers::TrendHandler, 129
- MAX_REST_RESP_COUNT
 - se2xb3::config::Constants, 25
- main
 - se2xb3::bootstrap::TwitterProcessorApp, 146
 - se2xb3::tests::TestDataSource, 121
- mapToTrendList
 - se2xb3::io::web::JsonParser, 77
- mapper
 - se2xb3::io::web::JsonParser, 80
- max
 - se2xb3::data::algorithms::WordGraph, 159
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 87
- MaxPQ.java, 170
- MaxPQ
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 84, 85
- Merge
 - se2xb3::data::algorithms::sort::textbook::Merge, 90
- merge
 - se2xb3::data::algorithms::sort::textbook::Merge, 91
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 94
 - se2xb3::data::algorithms::sort::textbook::MergeX, 100, 101
- Merge.java, 170
- MergeBU.java, 171
- MergeBU
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 94
- mergeSort
 - se2xb3::data::algorithms::SortStrategy, 119
- MergeSortStrategy
 - se2xb3::data::algorithms::sort::MergeSortStrategy, 97
- MergeSortStrategy.java, 171
- MergeX.java, 171
- MergeX
 - se2xb3::data::algorithms::sort::textbook::MergeX, 99
- MessageHandler
 - se2xb3::io::MessageHandler, 105
- MessageHandler.java, 171
- messageReceiver
 - se2xb3::io::AsyncMessageLooper, 21
- min
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 113
- MinPQ.java, 172
- MinPQ
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 110, 111
- msgHandler
 - se2xb3::io::source::DataSource, 48
- n
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 114
- name
 - se2xb3::data::models::Place, 116
 - se2xb3::data::models::User, 150
- next
 - se2xb3::data::algorithms::sort::textbook::MaxPQ↔::HeapIterator, 62
 - se2xb3::data::algorithms::sort::textbook::MinPQ↔::HeapIterator, 60
- node
 - se2xb3::data::models::TweetOld, 136
- notifications
 - se2xb3::data::models::User, 150
- objectMapper
 - se2xb3::data::DataHandler, 42
 - se2xb3::data::processing::TweetProcessor, 142
- onMessageReceived
 - se2xb3::io::IOController, 72
- onMessageReceivedEvent
 - se2xb3::data::DataController, 35
- parseInt
 - se2xb3::io::web::handlers::TrendHandler, 129
- passMessageToController
 - se2xb3::io::source::DataSource, 46

- Place
 - se2xb3::data::models::Place, 116
- place
 - se2xb3::data::models::TweetOld, 136
- Place.java, 172
- place_type
 - se2xb3::data::models::Place, 116
- postNewMessageEvent
 - se2xb3::control::EventBus, 49
- pq
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 114
- print
 - se2xb3::control::Controller, 28
 - se2xb3::data::DataHandler, 41
 - se2xb3::io::source::FileDataSource, 52
- println
 - se2xb3::control::Controller, 28
 - se2xb3::data::DataHandler, 41
 - se2xb3::io::source::FileDataSource, 52
- process
 - se2xb3::data::processing::IProcessor, 74
 - se2xb3::data::processing::TweetProcessor, 141
- processTweet
 - se2xb3::data::DataHandler, 42
- processTweetText
 - se2xb3::data::processing::TweetProcessor, 142
- profile_background_color
 - se2xb3::data::models::User, 150
- profile_background_image_url
 - se2xb3::data::models::User, 150
- profile_background_image_url_https
 - se2xb3::data::models::User, 150
- profile_background_tile
 - se2xb3::data::models::User, 150
- profile_banner_url
 - se2xb3::data::models::User, 150
- profile_image_url
 - se2xb3::data::models::User, 150
- profile_image_url_https
 - se2xb3::data::models::User, 150
- profile_link_color
 - se2xb3::data::models::User, 150
- profile_sidebar_border_color
 - se2xb3::data::models::User, 150
- profile_sidebar_fill_color
 - se2xb3::data::models::User, 150
- profile_text_color
 - se2xb3::data::models::User, 151
- profile_use_background_image
 - se2xb3::data::models::User, 151
- queue
 - se2xb3::io::source::DataSource, 48
- quoted_status
 - se2xb3::data::models::TweetOld, 136
- quoted_status_id
 - se2xb3::data::models::TweetOld, 136
- REST_REQUEST_NO_MATCH
 - se2xb3::config::Constants, 25
- REST_RESOURCE_TRENDS_ALL
 - se2xb3::config::Constants, 25
- REST_RESOURCE_TRENDS_HASHTAGS
 - se2xb3::config::Constants, 25
- REST_RESOURCE_TRENDS_USERS
 - se2xb3::config::Constants, 25
- REST_TREND_RESOURCE_PATH_REGEX
 - se2xb3::config::Constants, 25
- readData
 - se2xb3::io::source::FileDataSource, 52
- receiveMessage
 - se2xb3::io::IMessageReceiver, 67
 - se2xb3::io::MessageHandler, 105
- remove
 - se2xb3::data::algorithms::sort::textbook::MaxPQ↔
::HeapIterator, 62
 - se2xb3::data::algorithms::sort::textbook::MinPQ↔
::HeapIterator, 60
- requestCount
 - se2xb3::io::web::WebServer, 155
 - se2xb3::io::web::handlers::CORSHandler, 30
 - se2xb3::io::web::handlers::TrendHandler, 129
- resetData
 - se2xb3::data::DataController, 35
 - se2xb3::data::DataHandler, 42
 - se2xb3::data::algorithms::WordGraph, 159
 - se2xb3::data::processing::TweetProcessor, 142
- resize
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 113
- retweet_count
 - se2xb3::data::models::TweetOld, 136
- run
 - se2xb3::io::AsyncMessageLooper, 20
 - se2xb3::io::source::DataSource, 46
 - se2xb3::io::web::WebServer, 154
- screen_name
 - se2xb3::data::models::User, 151
- se2xb3, 9
- se2xb3.bootstrap, 9
- se2xb3.bootstrap.TwitterProcessorApp, 145
- se2xb3.config, 9
- se2xb3.config.AppConfig, 13
- se2xb3.config.Constants, 23
- se2xb3.control, 9
- se2xb3.control.AppController, 14
- se2xb3.control.Controller, 26
- se2xb3.control.EventBus, 48
- se2xb3.data, 10
- se2xb3.data.algorithms, 10
- se2xb3.data.algorithms.GraphStrategy, 57
- se2xb3.data.algorithms.SearchStrategy, 117

- se2xb3.data.algorithms.sort, [10](#)
- se2xb3.data.algorithms.sort.ISort, [75](#)
- se2xb3.data.algorithms.sort.MergeSortStrategy, [96](#)
- se2xb3.data.algorithms.sort.textbook, [10](#)
- se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >, [81](#)
- se2xb3.data.algorithms.sort.textbook.MaxPQ< Key >.HeapIterator, [61](#)
- se2xb3.data.algorithms.sort.textbook.Merge, [89](#)
- se2xb3.data.algorithms.sort.textbook.MergeBU, [93](#)
- se2xb3.data.algorithms.sort.textbook.MergeX, [98](#)
- se2xb3.data.algorithms.sort.textbook.MinPQ< Key >, [107](#)
- se2xb3.data.algorithms.sort.textbook.MinPQ< Key >.HeapIterator, [59](#)
- se2xb3.data.algorithms.SortStrategy, [118](#)
- se2xb3.data.algorithms.WordGraph, [156](#)
- se2xb3.data.algorithms.WordNode, [160](#)
- se2xb3.data.DataController, [31](#)
- se2xb3.data.DataHandler, [37](#)
- se2xb3.data.models, [11](#)
- se2xb3.data.models.BoundingBox, [22](#)
- se2xb3.data.models.Place, [115](#)
- se2xb3.data.models.Tweet, [130](#)
- se2xb3.data.models.TweetOld, [132](#)
- se2xb3.data.models.User, [147](#)
- se2xb3.data.processing, [11](#)
- se2xb3.data.processing.IProcessor< T >, [73](#)
- se2xb3.data.processing.TweetProcessor, [137](#)
- se2xb3.injection, [11](#)
- se2xb3.injection.TwitterAppInjector, [143](#)
- se2xb3.io, [11](#)
- se2xb3.io.AsyncMessageLooper< M >, [18](#)
- se2xb3.io.files, [11](#)
- se2xb3.io.files.FileFinder, [55](#)
- se2xb3.io.IMessageQueue< M >, [64](#)
- se2xb3.io.IMessageReceiver< M >, [66](#)
- se2xb3.io.IOController, [68](#)
- se2xb3.io.MessageHandler, [103](#)
- se2xb3.io.source, [12](#)
- se2xb3.io.source.DataSource, [43](#)
- se2xb3.io.source.FileDataSource, [50](#)
- se2xb3.io.source.IDataSource, [63](#)
- se2xb3.io.web, [12](#)
- se2xb3.io.web.handlers, [12](#)
- se2xb3.io.web.handlers.CORSHandler, [28](#)
- se2xb3.io.web.handlers.TrendHandler, [124](#)
- se2xb3.io.web.JsonParser, [76](#)
- se2xb3.io.web.Trend, [122](#)
- se2xb3.io.web.WebServer, [152](#)
- se2xb3.tests, [12](#)
- se2xb3.tests.FileDataSourceTest, [54](#)
- se2xb3.tests.TestDataSource, [120](#)
- se2xb3::bootstrap::TwitterProcessorApp
 - app, [147](#)
 - main, [146](#)
 - TwitterProcessorApp, [146](#)
- se2xb3::config::AppConfig
 - hostname, [14](#)
 - trendRefreshRate, [14](#)
- se2xb3::config::Constants
 - DATA_SOURCE_STREAM_RATE, [24](#)
 - DEFAULT_REST_RESP_COUNT, [24](#)
 - INPUT_FILE_NAME, [25](#)
 - INPUT_FILE, [24](#)
 - JAVADOC_INDEX_FILE, [25](#)
 - JAVADOC_ROOT_PATH, [25](#)
 - LIVE_STREAM, [25](#)
 - MAX_REST_RESP_COUNT, [25](#)
 - REST_REQUEST_NO_MATCH, [25](#)
 - REST_RESOURCE_TRENDS_ALL, [25](#)
 - REST_RESOURCE_TRENDS_HASHTAGS, [25](#)
 - REST_RESOURCE_TRENDS_USERS, [25](#)
 - REST_TREND_RESOURCE_PATH_REGEX, [25](#)
 - TESTING, [26](#)
 - TWEET_REGEX, [26](#)
 - WEB_APP_ROOT_FILE, [26](#)
 - WEB_APP_ROOT_PATH, [26](#)
- se2xb3::control::AppController
 - app, [18](#)
 - AppController, [16](#)
 - dataController, [18](#)
 - executorService, [18](#)
 - getDataController, [16](#)
 - getExecutorService, [16](#)
 - getIOController, [16](#)
 - getInstance, [16](#)
 - ioController, [18](#)
 - shutdown, [17](#)
 - shutdownApp, [17](#)
 - start, [17](#)
- se2xb3::control::Controller
 - Controller, [27](#)
 - print, [28](#)
 - println, [28](#)
 - shutdown, [28](#)
- se2xb3::control::EventBus
 - bus, [49](#)
 - EventBus, [49](#)
 - postNewMessageEvent, [49](#)
- se2xb3::data::DataController
 - app, [36](#)
 - DataController, [33](#)
 - dataHandler, [36](#)
 - getTrendingHashtags, [33](#)
 - getTrendingUsers, [33](#)
 - getTrendingWords, [34](#)
 - getTrends, [34](#)
 - getTweetList, [35](#)
 - onMessageReceivedEvent, [35](#)
 - resetData, [35](#)
 - shutdown, [35](#)
 - tweetList, [36](#)
- se2xb3::data::DataHandler
 - byValue, [42](#)
 - dataController, [42](#)

- DataHandler, 39
- getTrendingHashtags, 39
- getTrendingUsers, 40
- getTrendingWords, 40
- getTrends, 41
- graphStrategy, 42
- objectMapper, 42
- print, 41
- println, 41
- processTweet, 42
- resetData, 42
- searchStrategy, 42
- sortStrategy, 42
- tweetProcessor, 42
- tweetsList, 42
- se2xb3::data::algorithms::GraphStrategy
 - GraphStrategy, 58
 - setStrategy, 58
- se2xb3::data::algorithms::SearchStrategy
 - SearchStrategy, 117
 - setStrategy, 117
- se2xb3::data::algorithms::SortStrategy
 - mergeSort, 119
 - setStrategy, 119
 - sort, 119
 - SortStrategy, 119
 - sortStrategy, 119
 - sortTweetList, 119
- se2xb3::data::algorithms::WordGraph
 - addWord, 157
 - eq, 157, 158
 - getAllMentions, 158
 - getHashtags, 158
 - getTrends, 158
 - getUsers, 159
 - isBlacklisted, 159
 - max, 159
 - resetData, 159
 - size, 160
 - sortedMap, 160
 - w, 160
 - WordGraph, 157
 - words, 160
- se2xb3::data::algorithms::WordNode
 - add, 162
 - compareTo, 162
 - id, 163
 - list, 163
 - size, 163
 - toString, 163
 - WordNode, 162
- se2xb3::data::algorithms::sort::ISort
 - sort, 76
- se2xb3::data::algorithms::sort::MergeSortStrategy
 - MergeSortStrategy, 97
 - sort, 97
- se2xb3::data::algorithms::sort::textbook::MaxPQ::↔
 - HeapIterator
 - copy, 62
 - hasNext, 62
 - HeapIterator, 62
 - next, 62
 - remove, 62
- se2xb3::data::algorithms::sort::textbook::MaxPQ
 - comparator, 88
 - delMax, 85
 - exch, 85
 - insert, 86
 - isEmpty, 86
 - isMaxHeap, 86
 - iterator, 87
 - less, 87
 - max, 87
 - MaxPQ, 84, 85
 - n, 88
 - pq, 88
 - resize, 88
 - sink, 88
 - size, 88
 - swim, 88
- se2xb3::data::algorithms::sort::textbook::Merge
 - indexSort, 90
 - isSorted, 91
 - less, 91
 - Merge, 90
 - merge, 91
 - show, 92
 - sort, 92
- se2xb3::data::algorithms::sort::textbook::MergeBU
 - isSorted, 94
 - less, 94
 - merge, 94
 - MergeBU, 94
 - show, 94
 - sort, 95
- se2xb3::data::algorithms::sort::textbook::MergeX
 - CUTOFF, 102
 - exch, 99
 - insertionSort, 99
 - isSorted, 99, 100
 - less, 100
 - merge, 100, 101
 - MergeX, 99
 - show, 101
 - sort, 101, 102
- se2xb3::data::algorithms::sort::textbook::MinPQ::↔
 - HeapIterator
 - copy, 60
 - hasNext, 60
 - HeapIterator, 60
 - next, 60
 - remove, 60
- se2xb3::data::algorithms::sort::textbook::MinPQ
 - comparator, 114
 - delMin, 111
 - exch, 111

- greater, 112
- insert, 112
- isEmpty, 112
- isMinHeap, 112, 113
- iterator, 113
- min, 113
- MinPQ, 110, 111
- n, 114
- pq, 114
- resize, 113
- sink, 114
- size, 114
- swim, 114
- se2xb3::data::models::BoundingBox
 - attributes, 22
 - coordinates, 22
 - type, 22
- se2xb3::data::models::Place
 - bounding_box, 116
 - country, 116
 - country_code, 116
 - full_name, 116
 - id, 116
 - name, 116
 - Place, 116
 - place_type, 116
 - type, 116
 - url, 116
- se2xb3::data::models::Tweet
 - id_str, 131
 - text, 131
 - user, 131
 - user_name, 131
- se2xb3::data::models::TweetOld
 - checkNull, 134
 - checkNullLong, 134
 - coordinates, 135
 - created_at, 135
 - favorite_count, 135
 - geo, 135
 - getText, 135
 - IF_NULL_INT, 135
 - IF_NULL_LONG, 135
 - IF_NULL_STR, 135
 - id, 135
 - id_str, 135
 - in_reply_to_screen_name, 135
 - in_reply_to_status_id, 135
 - in_reply_to_status_id_str, 136
 - in_reply_to_user_id, 136
 - in_reply_to_user_id_str, 136
 - lang, 136
 - node, 136
 - place, 136
 - quoted_status, 136
 - quoted_status_id, 136
 - retweet_count, 136
 - source, 136
 - text, 136
 - timestamp_ms, 136
 - TweetOld, 133
 - user, 136
- se2xb3::data::models::User
 - contributors_enabled, 149
 - created_at, 149
 - default_profile, 149
 - default_profile_image, 149
 - description, 149
 - favourites_count, 149
 - follow_request_sent, 150
 - followers_count, 150
 - following, 150
 - friends_count, 150
 - geo_enabled, 150
 - id, 150
 - id_str, 150
 - is_translator, 150
 - lang, 150
 - listed_count, 150
 - location, 150
 - name, 150
 - notifications, 150
 - profile_background_color, 150
 - profile_background_image_url, 150
 - profile_background_image_url_https, 150
 - profile_background_tile, 150
 - profile_banner_url, 150
 - profile_image_url, 150
 - profile_image_url_https, 150
 - profile_link_color, 150
 - profile_sidebar_border_color, 150
 - profile_sidebar_fill_color, 150
 - profile_text_color, 151
 - profile_use_background_image, 151
 - screen_name, 151
 - statuses_count, 151
 - time_zone, 151
 - url, 151
 - User, 149
 - utc_offset, 151
 - verified, 151
- se2xb3::data::processing::IProcessor
 - process, 74
- se2xb3::data::processing::TweetProcessor
 - deserializeTweet, 139
 - getAllMentions, 140
 - getHashtags, 140
 - getUsers, 140
 - graph, 142
 - graphWord, 140
 - jsonStringToTree, 141
 - jsonToTweet, 141
 - objectMapper, 142
 - process, 141
 - processTweetText, 142
 - resetData, 142

- TweetProcessor, 139
- se2xb3::injection::TwitterAppInjector
 - configure, 144
 - TwitterAppInjector, 144
- se2xb3::io::AsyncMessageLooper
 - AsyncMessageLooper, 20
 - blockingQueue, 21
 - doShutdown, 21
 - enqueue, 20
 - messageReceiver, 21
 - run, 20
 - shutdown, 20
- se2xb3::io::IMessageQueue
 - enqueue, 65
 - shutdown, 65
- se2xb3::io::IMessageReceiver
 - receiveMessage, 67
- se2xb3::io::IOController
 - app, 72
 - dataSource, 72
 - getExecutorService, 70
 - getTrendingHashtags, 70
 - getTrendingUsers, 70
 - getTrendingWords, 71
 - getTrends, 71
 - IOController, 69
 - init, 72
 - onMessageReceived, 72
 - shutdown, 72
 - webServer, 72
- se2xb3::io::MessageHandler
 - dataSource, 106
 - getDataSource, 105
 - MessageHandler, 105
 - receiveMessage, 105
 - setDataSource, 105
 - shutdown, 106
- se2xb3::io::files::FileFinder
 - FileFinder, 56
 - findFileByName, 56, 57
- se2xb3::io::source::DataSource
 - controller, 48
 - DataSource, 45
 - getExecutorService, 45
 - getQueue, 45
 - getUrl, 46
 - msgHandler, 48
 - passMessageToController, 46
 - queue, 48
 - run, 46
 - setQueue, 47
 - setUrl, 47
 - shutdown, 47
 - startSource, 47
 - url, 48
- se2xb3::io::source::FileDataSource
 - _CLASS_NAME, 53
 - FileDataSource, 52
 - log, 53
 - print, 52
 - println, 52
 - readData, 52
 - startSource, 53
- se2xb3::io::web::JsonParser
 - JsonParser, 77
 - mapToTrendList, 77
 - mapper, 80
 - toContainerObject, 78
 - toJson, 78, 80
- se2xb3::io::web::Trend
 - count, 123
 - item, 123
 - Trend, 123
- se2xb3::io::web::WebServer
 - _CLASS_NAME, 155
 - ioController, 155
 - log, 155
 - requestCount, 155
 - run, 154
 - shutdown, 155
 - WebServer, 154
- se2xb3::io::web::handlers::CORSHandler
 - _CLASS_NAME, 30
 - handle, 30
 - log, 30
 - requestCount, 30
- se2xb3::io::web::handlers::TrendHandler
 - _CLASS_NAME, 129
 - getTrendingHashtags, 126
 - getTrendingUsers, 126
 - getTrendingWords, 127
 - getTrends, 127
 - handle, 128
 - ioController, 129
 - isNumber, 129
 - log, 129
 - parseInt, 129
 - requestCount, 129
 - TrendHandler, 126
- se2xb3::tests::FileDataSourceTest
 - lock, 55
 - setUp, 55
 - startSource, 55
 - waitForever, 55
 - waitForTest, 55
- se2xb3::tests::TestDataSource
 - blockingQueue, 122
 - endTest, 121
 - isDone, 122
 - lock, 122
 - main, 121
 - TestDataSource, 121
 - testing, 122
 - waitForever, 121
 - waitForTest, 122
- SearchStrategy

- se2xb3::data::algorithms::SearchStrategy, 117
- searchStrategy
 - se2xb3::data::DataHandler, 42
- SearchStrategy.java, 172
- setDataSource
 - se2xb3::io::MessageHandler, 105
- setQueue
 - se2xb3::io::source::DataSource, 47
- setStrategy
 - se2xb3::data::algorithms::GraphStrategy, 58
 - se2xb3::data::algorithms::SearchStrategy, 117
 - se2xb3::data::algorithms::SortStrategy, 119
- setUp
 - se2xb3::tests::FileDataSourceTest, 55
- setUrl
 - se2xb3::io::source::DataSource, 47
- show
 - se2xb3::data::algorithms::sort::textbook::Merge, 92
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 94
 - se2xb3::data::algorithms::sort::textbook::MergeX, 101
- shutdown
 - se2xb3::control::AppController, 17
 - se2xb3::control::Controller, 28
 - se2xb3::data::DataController, 35
 - se2xb3::io::AsyncMessageLooper, 20
 - se2xb3::io::IMessageQueue, 65
 - se2xb3::io::IOController, 72
 - se2xb3::io::MessageHandler, 106
 - se2xb3::io::source::DataSource, 47
 - se2xb3::io::web::WebServer, 155
- shutdownApp
 - se2xb3::control::AppController, 17
- sink
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 114
- size
 - se2xb3::data::algorithms::WordGraph, 160
 - se2xb3::data::algorithms::WordNode, 163
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 114
- sort
 - se2xb3::data::algorithms::SortStrategy, 119
 - se2xb3::data::algorithms::sort::ISort, 76
 - se2xb3::data::algorithms::sort::MergeSortStrategy, 97
 - se2xb3::data::algorithms::sort::textbook::Merge, 92
 - se2xb3::data::algorithms::sort::textbook::Merge↔BU, 95
 - se2xb3::data::algorithms::sort::textbook::MergeX, 101, 102
- SortStrategy
 - se2xb3::data::algorithms::SortStrategy, 119
- sortStrategy
 - se2xb3::data::DataHandler, 42
 - se2xb3::data::algorithms::SortStrategy, 119
- SortStrategy.java, 172
- sortTweetList
 - se2xb3::data::algorithms::SortStrategy, 119
- sortedMap
 - se2xb3::data::algorithms::WordGraph, 160
- source
 - se2xb3::data::models::TweetOld, 136
- start
 - se2xb3::control::AppController, 17
- startSource
 - se2xb3::io::source::DataSource, 47
 - se2xb3::io::source::FileDataSource, 53
 - se2xb3::tests::FileDataSourceTest, 55
- statuses_count
 - se2xb3::data::models::User, 151
- swim
 - se2xb3::data::algorithms::sort::textbook::MaxPQ, 88
 - se2xb3::data::algorithms::sort::textbook::MinPQ, 114
- TESTING
 - se2xb3::config::Constants, 26
- TWEET_REGEX
 - se2xb3::config::Constants, 26
- TestDataSource
 - se2xb3::tests::TestDataSource, 121
- TestDataSource.java, 173
- testing
 - se2xb3::tests::TestDataSource, 122
- text
 - se2xb3::data::models::Tweet, 131
 - se2xb3::data::models::TweetOld, 136
- time_zone
 - se2xb3::data::models::User, 151
- timestamp_ms
 - se2xb3::data::models::TweetOld, 136
- toContainerObject
 - se2xb3::io::web::JsonParser, 78
- toJson
 - se2xb3::io::web::JsonParser, 78, 80
- toString
 - se2xb3::data::algorithms::WordNode, 163
- Trend
 - se2xb3::io::web::Trend, 123
- TrendHandler
 - se2xb3::io::web::handlers::TrendHandler, 126
- TrendHandler.java, 173
- trendRefreshRate
 - se2xb3::config::AppConfig, 14
- Tweet.java, 173
- tweetList
 - se2xb3::data::DataController, 36
- TweetOld
 - se2xb3::data::models::TweetOld, 133
- TweetProcessor

- se2xb3::data::processing::TweetProcessor, [139](#)
- tweetProcessor
 - se2xb3::data::DataHandler, [42](#)
- TweetProcessor.java, [173](#)
- tweetsList
 - se2xb3::data::DataHandler, [42](#)
- TwitterAppInjector
 - se2xb3::injection::TwitterAppInjector, [144](#)
- TwitterAppInjector.java, [174](#)
- TwitterProcessorApp
 - se2xb3::bootstrap::TwitterProcessorApp, [146](#)
- TwitterProcessorApp.java, [174](#)
- type
 - se2xb3::data::models::BoundingBox, [22](#)
 - se2xb3::data::models::Place, [116](#)
- url
 - se2xb3::data::models::Place, [116](#)
 - se2xb3::data::models::User, [151](#)
 - se2xb3::io::source::DataSource, [48](#)
- User
 - se2xb3::data::models::User, [149](#)
- user
 - se2xb3::data::models::Tweet, [131](#)
 - se2xb3::data::models::TweetOld, [136](#)
- User.java, [174](#)
- user_name
 - se2xb3::data::models::Tweet, [131](#)
- utc_offset
 - se2xb3::data::models::User, [151](#)
- verified
 - se2xb3::data::models::User, [151](#)
- w
 - se2xb3::data::algorithms::WordGraph, [160](#)
- WEB_APP_ROOT_FILE
 - se2xb3::config::Constants, [26](#)
- WEB_APP_ROOT_PATH
 - se2xb3::config::Constants, [26](#)
- waitForever
 - se2xb3::tests::FileDataSourceTest, [55](#)
 - se2xb3::tests::TestDataSource, [121](#)
- waitForTest
 - se2xb3::tests::FileDataSourceTest, [55](#)
 - se2xb3::tests::TestDataSource, [122](#)
- WebServer
 - se2xb3::io::web::WebServer, [154](#)
- webServer
 - se2xb3::io::IOController, [72](#)
- WebServer.java, [174](#)
- WordGraph
 - se2xb3::data::algorithms::WordGraph, [157](#)
- WordGraph.java, [175](#)
- WordNode
 - se2xb3::data::algorithms::WordNode, [162](#)
- WordNode.java, [175](#)
- words
 - se2xb3::data::algorithms::WordGraph, [160](#)