Working with basic filters

This page contains a description of the basic filters in ANTS Memory Profiler.

For a summary of how to use them, see Using filters to find objects. For a description of which filters are most useful, see Suggestions for using filters.

Comparing snapshots

Show objects in your current snapshot, based on the comparison with your baseline snapshot.

From the current snapshot show only new objects

Show only objects created between the baseline snapshot and the current snapshot.

Use this filter to find objects that did not exist in the baseline snapshot. This is useful if you want to understand why memory usage has increased between the snapshots.

From the current snapshot show only surviving objects

Show only objects that remain in memory in both the baseline snapshot and the current snapshot.

Use this filter to find objects that exist in both snapshots. This is useful if you want to look for objects which might be leaking in the current snapshot, because they should have been disposed between the two snapshots.

From the current snapshot show only survivors in growing classes

Show only objects that match all of the following conditions:

- were present in the baseline snapshot
- are present in the current snapshot
- are instances of a class which has more instances in the current snapshot than in the baseline snapshot.

When this filter is applied, in the **Class List**, the Instance Diff column assumes that all instances in the baseline snapshot are potential survivors, and then calculates the difference between the number of survivors in the current snapshot and the number of potential survivors from the baseline snapshot. An instance diff of 0 means that all objects survived; an instance diff of -10 means that 10 objects that were in the baseline snapshot are not in the current snapshot.

Because only growing classes are included in this filter, the instance diff will never be a positive number.

This filter is particularly useful, because classes with an instance diff of 0 are good candidates for finding memory leaks (no instances of the class were disposed between the two snapshots).

From the current snapshot show only zombie objects

Show only objects that are still in memory in the current snapshot, even though there were indications in the baseline snapshot that they would not survive.

This includes objects with the following characteristics:

- Objects on which `Dispose()` has been called in the baseline snapshot, but which are still in memory in the current snapshot because garbage collection was prevented for some reason
- Objects on the finalizer queue in the baseline snapshot (this includes objects still on the finalizer queue in the current snapshot and objects which are no longer on the finalizer queue in the current snapshot)

When this filter is applied, in the **Class List**, the Instance Diff column compares the number of objects on the finalizer queue in the baseline snapshot to the number of objects on the finalizer queue in the current snapshot. An instance diff of -1 means that one object that was on the finalizer queue in the baseline snapshot is no longer on the finalizer queue.

This filter is useful because these objects should not normally still be in memory in the current snapshot.

Show Only...

Classes with source

Show only objects with source code.

Note that you should not normally use this filter when you start looking for the cause of a memory problem (see Understanding ANTS Memory Profiler).

Objects on the large object / Gen 0 / Gen 1 / Gen 2 heap

Show only objects in the selected area of memory.

Use this filter to understand how objects are allocated in memory. Looking at objects on the Gen 2 heap may be especially helpful, because these are the longest-surviving small objects.

You can also use this filter to investigate objects on the large object heap, which is useful if you have large object heap fragmentation.

Note that you will not normally see any objects on the Gen 0 heap during profiling, because taking a snapshot forces a garbage collection. For more information, see 'Understanding the small object heaps' in the .NET Memory Primer.

> ⚠ This filter is not available when you are profiling a .NET 1.1 application.