Filtering by reference

This page describes the filters that allow you to filter by the references to objects.

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.

Kept in memory exclusively by

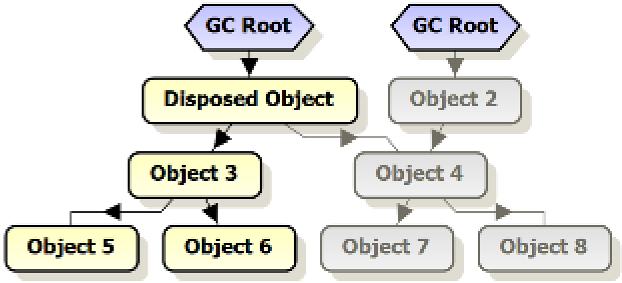
Disposed objects

This filter shows disposed objects and objects where all chains of references between the object and a GC root go through a disposed object.

> ⚠ The disposed object itself is included in the objects shown. Objects which are disposed but are on the finalizer queue are not shown.

To enable this filter, select **Kept in memory exclusively by**, then click **Add new item**, then select **Disposed objects**.

Example: The yellow objects are the objects shown by this filter.



Disposed objects should not normally be kept in memory, so this filter can help identify a memory leak.

When you have identified an object that is kept in memory only by a disposed object, display it on the object retention graph, and follow the chains of references to identify the objects keeping the selected object in memory.

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

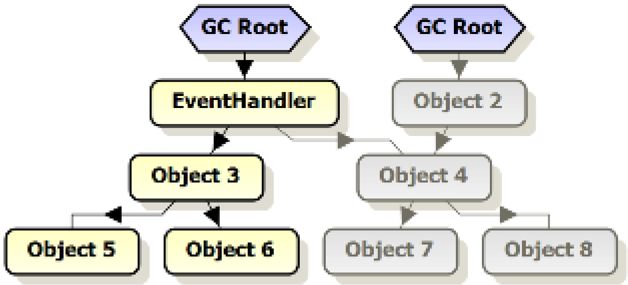
Event handlers

Show only objects where all chains of references between the object and a GC root go through an event handler.

> ⚠ The event handler is included in the objects shown. Objects which are held in memory only by event handlers but are on the finalizer queue are not shown.

To enable this filter, select **Kept in memory exclusively by**, then click **Add new item**, then select **Event handlers**.

Example: The yellow objects are the objects shown by this filter.



Event handlers should not normally be the only reason an object is kept in memory, so this filter can help identify a memory leak.

When you have identified an object that is kept in memory only by an event handler, display it on the object retention graph, and follow the chain of references from the object to the event handler; it is likely that the cause of the problem is an object close to the event handler.

GC root of type GC root type

Show objects with GC roots of the specified types.

The GC root object is included in the objects shown.

To enable this filter, select **Kept in memory exclusively by**, then click **Add new item**, then select **GC root of type...**.

When you select more than one type of GC root, objects are shown if they have a GC root which is any of the selected types.

This filter is especially useful because, if you filter objects that are kept in memory exclusively by a GC root of type finalizer queue, this will show objects on which Dispose() should be called.

To look for specific types of GC roots, combine this filter with the **Objects which are GC roots** filter.

Undisposed objects

Show undisposed objects where no chain of references between the object and its GC roots go through a disposed object.

Use this if you want an estimate of how much memory could be freed by disposing the objects.

To enable this filter, select **Kept in memory exclusively by**, then click **Add new item**, then select **Undisposed objects**.

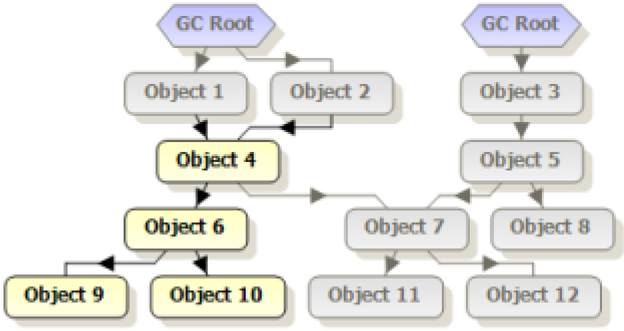
Add class / interface...

Show instances of the specified class or interface (including derived types) and objects where instances of the specified class or interface exist in all chains of references between the object and a GC root.

Instances of the specified class are included in the objects shown.

To enable this filter, select **Kept in memory exclusively by**, then click **Add class / interface**, then select the required class or interface.

Example: The yellow objects are the objects shown by the filter (where *Object 4* is the only object that is an instance of the selected class or that implements the selected interface).



This filter shows objects where instances of the specified class or interface (including derived types) exist in *all* chains of references between the object and a GC root. To show objects where instances of the specified class exist in *at least one* chain of references (but not necessarily in *all* chains of references) between the object and a GC root, use the **Referenced by instances of class/interface** filter. Objects which match your criteria but are on the finalizer queue are *not* shown.

If you select more than one class / interface, objects which are kept in memory by all of the listed classes / interfaces are shown.

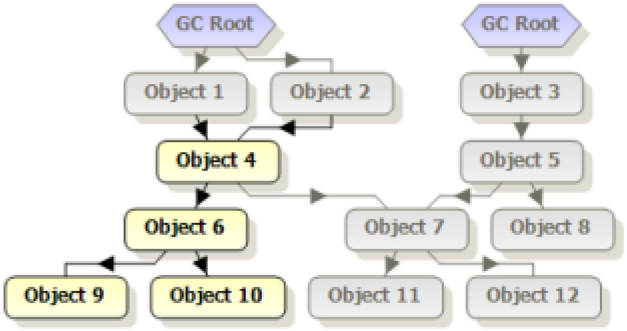
Add namespace...

Show objects from the specified namespace and objects where instances of classes from the specified namespace exist in all chains of references between the object and a GC root.

To enable this filter, select **Kept in memory exclusively by**, then click **Add namespace**, then select the required namespace.

Example: objects kept in memory exclusively by a class from the namespace that *Object 4* is an instance of:

(Note that *Object 4* - which is an instance of the a class in the specified namespace - is included in the objects shown)



If you select more than one namespace, objects which are kept in memory by all of the listed namespaces are shown.

⚠ Objects which match your criteria but are on the finalizer queue are *not* shown.

When you select multiple criteria for this filter, objects are only shown if they are satisfy all of the selected criteria.

Referenced by

The **Referenced by** filters behave identically to the **Kept in memory exclusively by** filters, except that objects are shown if at least one of the paths of reference to the GC route goes through the specified type or GC Root. The reference may be direct or indirect.

(With the Kept in memory exclusively by filters, objects are shown if all of the paths of reference to the GC route go through the specified type or GC Root.)

When you select multiple criteria for this filter, objects are only shown if they are satisfy all of the selected criteria.

Never referenced by

The Never referenced by filters have the opposite behavior to the Objects referenced by filters. Objects are shown if none of the paths of reference (including indirect references) to the GC root goes through the specified type of GC Root.

⚠ When viewing the Class List, the number of classes shown when the Never referenced by filter is enabled, plus the number of classes shown when the Referenced by filter is enabled, might add up to more than the total number of classes in the application. This is because some classes might have instances that are referenced by the selected type, while other instances of the same class are never referenced by the selected type.

The **Add class/interface** filter can be especially useful. Some examples of when you might want to use this filter:

- You know that all the data in your application should be referenced by a single main class. Apply the **Never referenced by an instance of class /interface** filter to remove objects referenced by the main class from the results.
- Your application performs caching (for example, web applications that cache the query results). Cached objects are deliberately kept in memory for a period of time. Apply the **Never referenced by an instance of class/interface** filter to exclude objects referenced by a cache class (for example, *System.Web.Caching.Cache*, for ASP.NET applications).

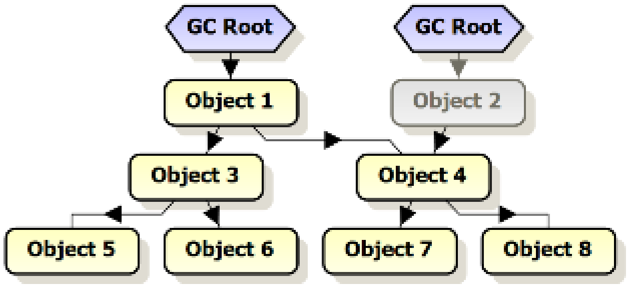
Relationships between objects

Filters on the **Filter by Reference** tab enable you to narrow down your search for memory problems by concentrating on certain types of relationships between objects.

Referenced by

Objects may be in memory because another object references them; the object is on at least one of the chains of references between the selected object and a GC root.

Example: objects referenced by *Object 1*

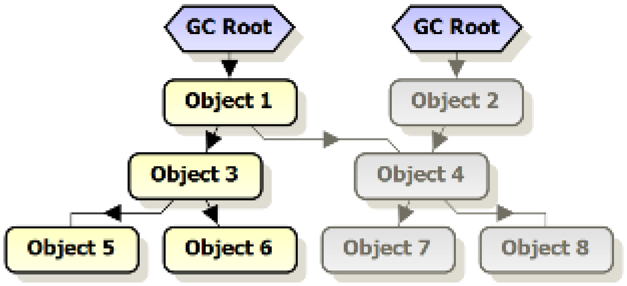


In this example all objects except *Object 2* are referenced by *Object 1*, either directly or indirectly. Note that the filter selection includes the specified object, *Object 1*.

Kept in memory exclusively by

Some filters show objects where the selected object is in all chains of references between the objects and a GC root - that is, objects are kept in memory *only* by the selected object.

Example: objects kept in memory by *Object 1* (note that the filter includes the selected object, *Object* 1)



In this example, only four objects are kept in memory only by *Object 1*. *Object 1* is not in all the chains of references between the remaining objects and their GC roots; for example, *Object 4* has another GC root which references it via *Object 2*. Note that the filter selection includes the specified object, *Object 1*.