

**Username:** Pralay Patoria **Book:** Under the Hood of .NET Memory Management. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC 107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

---

## Introduction

A true memory leak is rare; memory hogs are much more common. A memory hog will cause the framework to hold on to more memory than you would anticipate, and looks suspiciously like a memory leak. Regardless of the cause, the symptoms and the pain are the same. Your application may take a performance hit as the system starts paging. You may notice that your application seems to freeze periodically for garbage collection. In the worst-case scenario, your application may crash with

### `OutOfMemoryExceptions`.

Naturally, different applications share common areas of contention that are often problematic, and we're going to take a look at some of the more commonly occurring pitfalls in this chapter. I should make it clear that we are going to skim through a broad range of technologies to give you a sense of where their common pitfalls are, as well as dip into a few unique gotchas which you'll only find in one place. The purpose of this is not to give you an A-Z guide, but rather to give you an appreciation of the .NET memory management landscape, and perhaps give you a few nudges towards areas to watch out for. With that in mind, let's dive right in.