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 USC107) 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.

Overview

If you think about it, an application is made up of two things: the code itself, and the data that stores the state of the application during execution. When a .NET application runs, four sections of memory (heaps) are created to be used for storage:

- **the Code Heap** stores the actual native code instructions after they have been Just in Time compiled **(JITed)**
- the **Small Object Heap (SOH)** stores allocated objects that are less than 85 K in size
- the **Large Object Heap (LOH)** stores allocated objects *greater* than 85 K (although there are some exceptions, which are covered in Chapter 2)
- finally, there's the **Process Heap**, but that's another story.

Everything on a heap has an address, and these addresses are used to track program execution and application state changes.

Applications are usually written to encapsulate code into methods and classes, so .NET has to keep track of chains of method calls as well as the data state held within each of those method calls. When a method is called, it has its own cocooned environment where any data variables it creates exist only for the lifetime of the call. A method can also get data from global/static objects, and from the parameters passed to it.

In addition, when one method calls another, the local state of the calling method (variables, parameters) has to be remembered while the method to be called executes.

Once the called method finishes, the original state of the caller needs to be restored so that it can continue executing.

To keep track of everything (and there is often quite a lot of “everything”) .NET maintains a stack data structure, which it uses to track the state of an execution thread and all the method calls made.