

内存上升监视器 (MemoryAscending) 使用手册

模块名称: AT.Toolkit.MemoryAscending

功能定位: 实时监控 .NET 应用内存增长趋势, 自动检测疑似内存泄漏, 定位泄漏类型及持有链

运行环境: ATT 框架

目录

- 安装部署
- 界面总览
- 启动监控
- 查看监控数据
- 泄漏模拟测试
- 泄漏自动诊断
- 屏蔽类型管理
- 配置参数说明
- 常见问题排查

1. 安装部署

第 1 步: 下载



第 2 步：安装



第 3 步：重启 ATT 应用



2. 界面总览

操作

内存监控配置

快照间隔(秒): 10 | 连续增长阈值: 10 | 排名前N: 10

显示泄漏测试面板 停止监控

▶ 正在泄漏: 非托管内存 | byte[]: 0块(0MB) | String: 0条 | 事件: 0个 | 非托管: 3块(6MB)

托管byte[] | 每次+2MB | 托管String | 每次+500条

事件订阅 | 每次+50个 | 非托管内存 泄漏中...

全部停止并释放

排名	所属模块	类型名	总大小(MB)	实例数	增量大小(M...)	增量数	连续增长
1	mscorlib.dll	System.Int32[]	4.52	21,589	+1.62	+25	1
2	mscorlib.dll	System.Collections.Generic.Dictionary<System.UInt64, Microsoft.Diagnostics.Runtime...	1.86	24,338	+0.95	+12,404	1
3	mscorlib.dll	System.Collections.Generic.Dictionary<System.String, AT.Toolkit.MemoryAscending.M...	0.92	4	+0.82	+3	1
4	mscorlib.dll	System.Collections.Generic.Dictionary<System.UInt64, Microsoft.Diagnostics.Runtime...	0.69	3	+0.59	+2	1
5	mscorlib.dll	System.ValueTuple<System.UInt64, System.Int32>[]	0.10	3	+0.05	+1	1
6	mscorlib.dll	System.Collections.Generic.Dictionary<System.UInt64, Microsoft.Diagnostics.Runtime...	0.02	2	+0.01	+1	1
7	mscorlib.dll	System.Collections.Hashtable+bucket[]	6.88	22,197	+0.01	+13	1
8	DevExpress.Utils.v2...	DevExpress.XtraEditors.TextEditController.LineCharInfo[]	0.02	176	0.00	+9	1
9	DevExpress.XtraBar...	DevExpress.XtraBars.Docking.Helpers.DockZoneReference	0.02	388	0.00	+71	1
10	DevExpress.Utils.v2...	DevExpress.Utils.Drawing.GraphicsCache	0.12	1,265	0.00	+31	1

快照#2 16:30:09 耗时:1649ms ✓ 扫描:1,466,359个对象
私有字节: 1028.8 MB (增量: +87.3 MB) | GC堆: 175.5 MB (增量: +10.2 MB) | 非托管: 853.3 MB (增量: +77.1 MB)
信息 工作集: 1023.0 MB | GC回收: Gen0=760 Gen1=165 Gen2=20

3. 启动监控

第 1 步: 设置监控参数

参数	默认值	说明
快照间隔(秒)	10	每隔多少秒做一次完整堆分析, 范围 10~36000
连续增长阈值	10	某类型连续增长多少次后判定为「疑似泄漏」, 范围 2~20
排名前N	10	显示内存占用/增长 Top 多少个类型, 范围 1~100

第 2 步: 点击「▶ 启动监控」

- 按钮变为 **「■ 停止监控」**
- 配置区自动锁定（不可修改参数）
- 系统开始周期性快照

第 3 步：等待数据积累

- **第 1 次快照：** 建立基线，无增量数据
- **第 2 次快照起：** 开始计算增量，趋势图出现曲线
- **第 N 次快照：** 连续增长计数器累积，达到阈值后自动触发泄漏诊断

第 4 步：停止监控

再次点击 **「■ 停止监控」** 即可停止。

4. 查看监控数据

4.1 趋势图

曲线	颜色	含义
专用字节	● 蓝色面积	进程独占的全部物理内存 (托管 + 非托管 + 运行时)
GC 堆	● 黄色面积	.NET GC 管理的托管堆大小
非托管	● 红色虚线	估算值 = 专用字节 - GC堆

操作提示： 鼠标悬停在图表上可看到各时间点的精确数值 (GB)。

4.2 类型排名表格

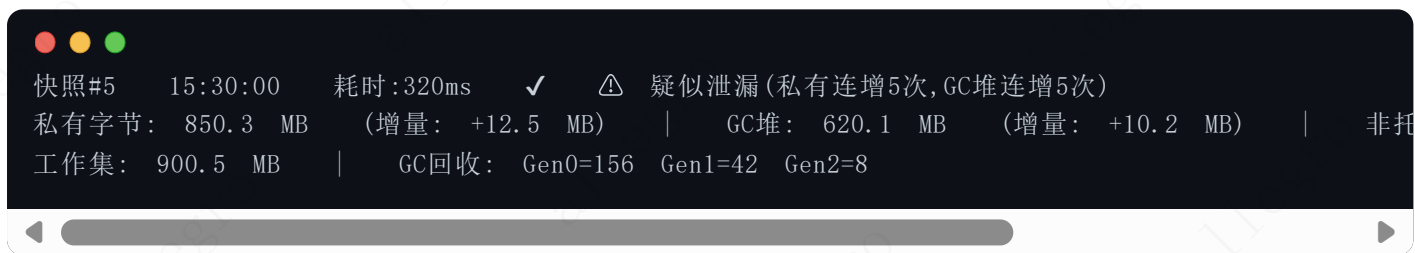
默认按 **「连续增长次数」** 降序排列，显示以下列：

列名	说明
排名	当前排序位次
所属模块	类型所在的 DLL 名称

列名	说明
类型名	.NET 类型全名, 如 <code>System.Byte[]</code>
总大小(MB)	该类型所有实例的总内存占用
实例数	堆中该类型的对象数量
增量大小(MB)	相比上次快照, 大小变化 (黄色=增长, 绿色=减少)
增量数	相比上次快照, 实例数量变化
连续增长	连续多少次快照该类型都在增长 (达到阈值时红色高亮)

4.3 信息摘要

底部摘要栏显示:



5. 泄漏模拟测试

用于验证监控工具的检测能力, **仅在调试阶段使用。**

第 1 步: 点击「显示泄漏测试面板」

展开面板后显示 4 个泄漏场景按钮 + 1 个清除按钮。

第 2 步: 选择泄漏场景

按钮	场景	每次泄漏量	预期检测结果
托管byte[]	List<byte[]> 不释放	+2MB	类型排名出现 System.Byte[] , 总大小持续增长
托管String	List<string> 不释放	+500条	System.String 实例数持续增长
事件订阅	event += handler 不注销	+50个	EventLeakSubscriber 类型持续增长
非托管内存	Marshal.AllocHGlobal 不释放	+2MB	私有字节增长但 GC堆不变, 非托管估算增长

- **泄漏间隔** = 快照间隔的一半 (确保每次快照都能检测到新增)
- 点击已运行的按钮可**停止**该场景
- 同时只能运行一种场景

第 3 步：观察检测结果

启动泄漏后，配合「启动监控」观察：

1. **趋势图** — 曲线是否持续上升
2. **排名表格** — 对应类型是否出现在榜首，连续增长是否递增
3. **信息摘要** — 是否触发「⚠ 疑似泄漏」警告

第 4 步：清除泄漏

点击「**×** 全部停止并释放」：

- 停止所有自动泄漏
- 释放全部已分配的内存
- 触发 GC 强制回收
- 后续快照可观察内存是否回落

6. 泄漏自动诊断

触发条件

当某个类型的 **连续增长次数** \geq **连续增长阈值** 时，系统自动生成诊断报告。

报告内容

诊断报告会自动弹出到通知窗口，包含：

```

=== 自动诊断报告 ===

▼ 疑似泄漏 (连续增长 $\geq$ 10次)

△ System.Collections.Hashtable+ValueCollection
  连续增长: 10次 | 总大小: 0.0MB (+0.0MB) | 实例: 75个 (+1)
  所属模块: mscorlib.dll
  持有链:
    (Root追踪超时)
  排查建议:
  该类型来自模块[mscorlib.dll]，检查该模块中是否有集合/缓存/事件未释放

△ System.Collections.Generic.Dictionary<System.String,
AT.Toolkit.MemoryAscending.Model.TypeMemoryInfo> + ValueCollection
  连续增长: 10次 | 总大小: 0.0MB (+0.0MB) | 实例: 6个 (+1)
  所属模块: mscorlib.dll
  持有链:
    [反向追踪] 分析增量 +1 个最新实例
    [持有链] System.Linq.Enumerable+<TakeIterator>d__25<AT.T....source(4处)
    → System.Linq.OrderedEnumerable<AT.Toolkit.Memory....source(4处) →
    System.Linq.Enumerable+WhereEnumerableIterator<....source(4处) →
    System.Collections.Generic.Dictionary<System.String,
    AT.Toolkit.MemoryAscending.Model.TypeMemoryInfo> + ValueCollection
    (发现7种持有者，均为系统类型，无自定义模块)
  排查建议: 检查日志集合是否有上限、是否循环拼接字符串(应用StringBuilder)
  
```

连续增长判定规则

每次快照对比上次，如果某类型同时满足以下**与条件**则视为"增长"：

- `DeltaCount > 0` (实例数净增)
- `DeltaSize > 0` (总大小净增)

连续 N 次都增长 → 连续增长计数器 +N。一旦某次**不增长**，计数器**归零**。

7. 屏蔽类型管理

操作方式

在类型排名表格中 **右键点击** 某一行：

菜单项	说明
× 屏蔽此类型	将该类型从排名和泄漏检测中排除
✓ 取消屏蔽	恢复显示和检测
📄 管理已屏蔽类型	查看所有已屏蔽类型，点击可取消

	System.String
	System.Collections.Generic.Dictionary<System.UInt64, Microsoft.Diagnostics.Runtime
	System.Int32[]
	System.ValueTuple<System.UInt64, Sys
2...	DevExpress.XtraEditors.ViewInfo.TextEditViewInfo
	DevExpress.Utils.FrozenAppearance
	DevExpress.XtraEditors.TextEditController.LineCharInfo[]
....	DevExpress.Charts.Native.RefinedPoint
...	DevExpress.XtraGrid.Views.Grid.ViewInfo.GridCellInfo

× 屏蔽此类型: System.Int32[]

📄 管理已屏蔽类型 (1个)

右键可以屏蔽

使用场景

- 某些框架类型（如 DevExpress 控件）正常增长但不是泄漏 → 屏蔽
- 减少排名噪音，聚焦真正的泄漏目标

持久化

屏蔽配置自动保存到工具配置文件，重启后保留。

8. 配置参数说明

参数	类型	默认值	范围	说明
IntervalSeconds	int	10	10~36000	快照间隔(秒)
TopN	int	10	1~100	排名显示前 N 个类型
LeakThreshold	int	10	2~20	连续增长多少次判定为疑似泄漏
SnapshotTimeoutMs	int	30000	-	单次堆遍历超时(ms), 自动按间隔的 80% 计算
BlockedTypes	List	空	-	已屏蔽的类型名列表

9. 常见问题排查

Q1: 快照失败, 提示超时

原因: 进程内存过大 (>2GB), CLRMD 堆遍历时间超过超时限制。

解决: 增大 **快照间隔** 至 30~60 秒 (超时自动按间隔的 80% 计算)。

Q2: 趋势图曲线锯齿状波动, 不是持续上升

原因: 正常现象。GC 回收导致内存周期性下降, 不是泄漏。

判断: 观察连续增长计数器是否超过阈值, 低阈值波动属正常。

Q3: 非托管内存增长但排名表格看不到对应类型

原因: 非托管内存不受 GC 管理, CLRMD 无法追踪。

解决: 观察趋势图中「红色虚线(非托管估算)」是否持续上升, 配合代码审查

`Marshal.AllocHGlobal`、`HImage`、`FileStream` 等未 Dispose 的对象。

Q4: 屏蔽了类型后, 诊断报告还是会提示

原因: 屏蔽类型会从排名表格和 `SuspectedLeakTypes` 中排除, 不影响进程级连续增长判定。

说明: 如果私有字节或 GC堆整体连续增长超过阈值, 仍会触发诊断。
