# Очистка неуправляемой памяти .NET

## Аннотация

В данной статье были изучены различные подходы для очистки неуправляемой памяти и в последствии повышении производительности приложения на платформе .NET. Основная цель данной статьи является обзор и сравнительный анализ существующих инструментов и методов для работы с неуправляемой памятью. Для достижения цели был поставлен ряд задач, таких как: сбор информации о существующий решениях и методах, выделение критериев по которым будет проводиться сравнение, определение недостатков и преимуществ существующих решений и формирование перечня требований для существующих методов . Выявлены преимущества использования различных подходов, такие как метод Dispose(), Finalize и другие.  
В ходе практического анализа были выявлены решения, которые показали лучшие результаты по выбранным критериям, а именно метод Disponse() .

Ключевые слова: .NET; неуправляемая память; производительность.

## Введение

При создании приложения, отличающиеся высокой производительностью, следует закладывать эту свойство при планировании и проектировании наравне с другими возможностями приложения. Для управления памятью в приложениях .NET существуют стандартные механизмы для очистки неуправляемых ресурсов. Предметом исследования являются механизмы очистки неуправляемых ресурсов. Объект исследования - неуправляемые ресурсы. Цель данной работы заключается обзоре и сравнительном анализе инструментов управления неуправляемыми ресурсами в .NET [4].  
Для достижения цели потребуется решить ряд задач:

* Собрать информацию о существующих решениях.
* Выделить критерии, по которым будет проводиться сравнение.
* Определить недостатки и преимущества существующих решений.

## Сравнение аналогов

### Принцип отбора аналогов

Шаблон ликвидации объекта, именуемый также шаблоном удаления, налагает определенные правила на время жизни объекта. Шаблон удаления используется только для объектов, осуществляющих доступ к неуправляемым ресурсам [1].  
Существует три варианта шаблона удаления:

* **Метод Dispose() и Disponse(boolean)**
* **Метод Dispose() и Finalize**
* **Finalize**

## Аналоги

### Метод Dispose() и Disponse(boolean)

Реализация метода Dispose() необходима для освобождения неуправляемых ресурсов, которые использует приложение. Dispose(bool disposing) в свою очередь является защищенным и вызывается только методом Dispose(). Boolean флаг задает в каком режиме будет работать этот метод. Флаг true указывает методу, что нужно очистить все ресурсы: управляемые и неуправляемые. Флаг false указывает на очистку только неуправляемых ресурсов [3].

### Метод Dispose() и Finalize

Логика очистки реализуется перегруженной версией метода Dispose(bool disposing). При вызове перегруженногоFinilize в качестве параметра disposing передается значение false, чтобы избежать очистки управляемых ресурсов, так как мы не можем быть уверенными в их состоянии, что они до сих пор находятся в памяти. И в этом случае остается полагаться на Finilize этих ресурсов. В обоих случаях освобождаются неуправляемые ресурсы [5].  
### Finalize  
Метод Finalize уже определен в базовом для всех типов классе Object, однако данный метод нельзя так просто переопределить. И фактическая его реализация происходит через создание деструктора.  
На уровне памяти это выглядит следующим образом : сборщик мусора при размещении объекта в куче определяет, поддерживает ли данный объект метод Finalize. И если объект имеет метод Finalize, то указатель на него сохраняется в специальной таблице, которая называется очередь финализации. Когда наступает момент сборки мусора, сборщик видит, что данный объект должен быть уничтожен, и если он имеет метод Finalize, то он копируется в еще одну таблицу и окончательно уничтожается лишь при следующем проходе сборщика мусора [2]. ## Критерии сравнения аналогов  
Изучив вышеперечисленные аналоги, сформирована таблица 1, которая содержит сравнение описанных выше решений по выбранным критериям:  
### Простота реализации  
Сложность алгоритма шаблоны удаления.  
### Наличие автоматизации  
Очистка неуправляемых ресурсов происходит во время работы приложения, либо разработчик реализует на уровне программного кода. ### Влияние на производительность  
Насколько представленный шаблон влияет на работу приложения по очистке памяти.  
## Таблица сравнения по критериям  
Таблица 1. Сравнение по критериям

|  |  |  |  |
| --- | --- | --- | --- |
| Шаблоны | Простота реализации | Наличие автоматизации | Влияние на производительность |
| Dispose | Простая реализация | Нет | Крайне высокая |
| Dispose и Finalize | Средняя сложность | Да, присутствует реализация на уровне программного кода | Высокая |
| Finalize | Простая реализация | Да | Средняя, но из-за стандартов языка негативно отражается на производительности |

## Выводы по итогам сравнения

Лучшим решением на основании обзора и анализа аналогов оказался Dispose, так как он имеет очень высокое влияние на производительность, что важно в приложении. Хорошие результаты показал и комбинированный шаблон, однако Finalize не лучший вариант без связки с Dispose .  
## Выводы  
По результатам сравнительного анализа существующих решений, которые реализуют освобождение неуправляемой памяти, лучшим решением оказался шаблон ликвидации на основе метода Dispose() и Dispose(boolean). Этот шаблон, как и комбинированный показали себя способными обеспечить высокую производительность с минимальными ошибками. Finalize показал себя как хорошее подспорье начинающим программистам, но не пригодным для управления неуправляемыми ресурсами. В планах на дальнейшие исследования, реализация различных методов Dispose, которые будут удовлетворять запросы разработчиков в производительных приложениях.

## Источники

1. Реализация метода Dispose/Saisang Cai - Текст: электронный//Microsoft Docs: Интернет-портал -URL: <https://docs.microsoft.com/ru-ru/dotnet/standard/garbage-collection/implementing-dispose>
2. Финализируемые объекты/Алексей Крамаренко -Текст: электронный//METANIT.COM:Интернет-портал -URL:<https://metanit.com/sharp/tutorial/8.2.php>
3. Полное руководство по С# 2.0: учебное пособие/Шилдт Герберт - Эком, 2007. - 902 c.
4. Высокопроизводительный код на платформе .NET: учебное пособие/Уотсон Бен - Питер, 2019. - 118 c.
5. Dispose pattern/Сергей Тепляков - Текст: электронный//habr.com: Интернет-портал -URL: <https://habr.com/ru/post/129283/>