

C# Concurrency

АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ И МНОГОПОТОЧНОСТЬ

Нир ДОВОИЦКИ



Санкт-Петербург • Москва • Минск

2025

Краткое содержание

Часть I

Основы асинхронного программирования и многопоточности

Глава 1. Асинхронное программирование и многопоточность	22
Глава 2. Компилятор переписывает ваш код	34
Глава 3. Ключевые слова <code>async</code> и <code>await</code>	45
Глава 4. Основы многопоточности	65
Глава 5. <code>async/await</code> и многопоточность	88
Глава 6. Когда использовать <code>async/await</code>	101
Глава 7. Классические ловушки многопоточности и как их избежать	115

Часть II

Продвинутое использование `async/await` и многопоточности

Глава 8. Обработка последовательности элементов в фоновом режиме	140
Глава 9. Отмена фоновых заданий	160
Глава 10. Ожидаем собственные события	174
Глава 11. Выбор потока для выполнения асинхронного кода	188
Глава 12. <code>async/await</code> и исключения	211
Глава 13. Потокобезопасные коллекции	219
Глава 14. Асинхронная генерация коллекций / <code>await foreach</code> и <code>IAsyncEnumerable</code>	255

Оглавление

Предисловие	12
Благодарности	14
О книге	16
Кому адресована эта книга	16
Структура издания	16
О примерах программного кода	17
От издательства	18
Об авторе	19
Иллюстрация на обложке	20
Вступительное слово от сообщества DotNetRu	21

Часть I

Основы асинхронного программирования и многопоточности

Глава 1. Асинхронное программирование и многопоточность	24
1.1. Что такое многопоточность	25
1.2. Многоядерные процессоры	28
1.3. Асинхронное программирование	31
1.4. Совместное использование многопоточности и асинхронного программирования	33
1.5. Эффективность программного обеспечения и облачные вычисления	34
Итоги главы	35
Глава 2. Компилятор переписывает ваш код	36
2.1. Лямбда-функции	37
2.2. Инструкция yield return	39
Итоги главы	44

Глава 3. Ключевые слова <code>async</code> и <code>await</code>	45
3.1. Сложность асинхронного кода	46
3.2. Знакомство с <code>Task</code> и <code>Task<T></code>	48
3.2.1. Мы уже приехали?	50
3.2.2. Разбуди меня, когда приедем	51
3.2.3. Синхронный вариант	52
3.2.4. После завершения задачи	53
3.3. Как работают <code>async/await</code>	55
3.4. Асинхронные методы <code>void</code>	60
3.5. <code>ValueTask</code> и <code>ValueTask<T></code>	62
3.6. А что насчет многопоточности?	63
Итоги главы	63
Глава 4. Основы многопоточности	65
4.1. Способы запуска кода в другом потоке	66
4.1.1. <code>Thread.Start</code>	66
4.1.2. Пул потоков	70
4.1.3. <code>Task.Run</code>	72
4.2. Доступ к одним и тем же переменным из нескольких потоков	75
4.2.1. Отказ от общих данных	77
4.2.2. Неизменяемые общие данные	77
4.2.3. Использование блокировок и мьютексов	78
4.2.4. Взаимоблокировки	80
4.3. Замечания о приложениях с графическим пользовательским интерфейсом	81
4.4. Ожидание другого потока	82
4.5. Другие методы синхронизации	83
4.6. Параметры потока	84
4.6.1. Режим выполнения потока в фоне	85
4.6.2. Язык и локаль	85
4.6.3. СОМ-апартаменты	85
4.6.4. Текущий пользователь	86
4.6.5. Приоритет потока	86
Итоги главы	86
Глава 5. <code>async/await</code> и многопоточность	88
5.1. Асинхронное программирование и многопоточность	89
5.2. Где выполняется код после вызова <code>await</code>	92
5.3. Блокировки и <code>async/await</code>	95
5.4. Потоки пользовательского интерфейса	98
Итоги главы	100

Глава 6. Когда использовать async/await	101
6.1. Преимущества асинхронности на серверах.....	102
6.2. Преимущества асинхронности в нативных клиентских приложениях.....	108
6.3. Недостатки async/await.....	109
6.3.1. Асинхронное программирование заразно.....	109
6.3.2. Асинхронное программирование имеет больше пограничных случаев.....	111
6.3.3. Многопоточность имеет еще больше пограничных случаев.....	112
6.3.4. async/await — это дорого.....	112
6.4. Когда использовать async/await	113
Итоги главы.....	114
Глава 7. Классические ловушки многопоточности и как их избежать	115
7.1. Частичное обновление	116
7.2. Изменение порядка доступа к памяти при компиляции	119
7.3. Взаимоблокировки.....	123
7.4. Состояние гонки.....	130
7.5. Синхронизация.....	133
7.6. Голодание	135
Итоги главы.....	138

Часть II

Продвинутое использование async/await и многопоточности

Глава 8. Обработка последовательности элементов в фоновом режиме	140
8.1. Параллельная обработка элементов.....	141
8.1.1. Параллельная обработка элементов с помощью класса Thread.....	142
8.1.2. Параллельная обработка элементов с использованием пула потоков.....	144
8.1.3. Асинхронная параллельная обработка элементов.....	146
8.1.4. Класс Parallel.....	148
8.2. Последовательная обработка элементов в фоновом режиме.....	152
8.2.1. Последовательная обработка элементов в фоновом режиме с помощью класса Thread	152
8.2.2. Паттерн очереди заданий и BlockingCollection	154
8.2.3. Обработка важных элементов с помощью постоянных очередей.....	157
Итоги главы.....	158

Глава 9. Отмена фоновых заданий	160
9.1. Введение в CancellationToken	160
9.2. Отмена с использованием исключения	168
9.3. Получение обратного вызова, когда вызывающий код отменяет операцию	169
9.4. Реализация тайм-аутов	170
9.5. Комбинирование методов отмены	171
9.6. Специальные токены отмены	172
Итоги главы	172
Глава 10. Ожидаем собственные события	174
10.1. Знакомство с TaskCompletionSource	175
10.2. Выбор потока для запуска продолжения	179
10.3. Пример: ожидание инициализации	180
10.4. Пример: адаптация старых API	182
10.5. Асинхронные операции в старом стиле (BeginXXX, EndXXX)	183
10.6. Пример: асинхронные структуры данных	184
Итоги главы	187
Глава 11. Выбор потока для выполнения асинхронного кода	188
11.1. Поведение await в многопоточной среде	189
11.1.1. await в потоках пользовательского интерфейса	190
11.1.2. await в потоках, отличных от потока пользовательского интерфейса	191
11.2. Контексты синхронизации	193
11.3. Переключение потоков — ConfigureAwait(false)	197
11.4. Дополнительные параметры ConfigureAwait	204
11.5. Task.Yield: возможность выполнить другой код	205
11.6. Планировщики задач	207
Итоги главы	209
Глава 12. async/await и исключения	211
12.1. Исключения и асинхронный код	211
12.2. await и AggregateException	215
12.3. Потеря исключений	216
12.4. Исключения и методы async void	217
Итоги главы	217

Глава 13. Потокобезопасные коллекции	219
13.1. Проблемы использования обычных коллекций	220
13.2. Потокобезопасные коллекции	225
13.2.1. ConcurrentDictionary<TKey,TValue>	225
13.2.2. BlockingCollection<T>	229
13.2.3. Асинхронные альтернативы BlockingCollection	232
13.2.4. ConcurrentQueue<T> и ConcurrentStack<T>	233
13.2.5. ConcurrentBag<T>	234
13.2.6. Когда следует использовать потокобезопасные коллекции	235
13.2.7. Когда не следует использовать потокобезопасные коллекции	235
13.3. Неизменяемые коллекции	235
13.3.1. Как работают неизменяемые коллекции	236
13.3.2. Как использовать неизменяемые коллекции	242
13.3.3. ImmutableInterlocked	244
13.3.4. ImmutableDictionary<TKey,TValue>	245
13.3.5. ImmutableHashSet<T> и ImmutableSortedSet<T>	246
13.3.6. ImmutableList<T>	247
13.3.7. ImmutableQueue<T> и ImmutableStack<T>	248
13.3.8. ImmutableArray<T>	248
13.3.9. Когда использовать неизменяемые коллекции	250
13.4. Замороженные коллекции	250
13.4.1. Когда использовать замороженные коллекции	252
Итоги главы	252
 Глава 14. Асинхронная генерация коллекций / await foreach и IEnumerable	255
14.1. Итерации по асинхронной коллекции	256
14.2. Создание асинхронной коллекции	258
14.3. Отмена итерации по асинхронной коллекции	261
14.4. Другие варианты	264
14.5. IEnumerable<T> и LINQ	264
14.6. Пример: итерации по данным, получаемым асинхронно	265
14.7. Пример: асинхронная очередь, подобная BlockingCollection<T>	266
Итоги главы	269