

Bill Wagner

Bardziej efektywny C#

50 sposobów ulepszenia języka C#

APN Promise
Warszawa 2018

Spis treści

Wprowadzenie	vii
Podziękowania	xi
O autorze	xii
Rozdział 1: Praca z typami danych	1
Punkt 1: Korzystanie z właściwości zamiast dostępnych pól danych	1
Punkt 2: Preferowanie niejawnych właściwości dla zmiennych danych	9
Punkt 3: Preferowanie niezmienności typów wartościowych	13
Punkt 4: Rozróżnianie pomiędzy typami wartościowymi a referencyjnymi	20
Punkt 5: Zagwarantowanie, że 0 jest poprawnym stanem dla typów wartościowych	26
Punkt 6: Upewnienie się, że właściwości zachowują się jak dane	31
Punkt 7: Ograniczanie zakresu typu poprzez użycie krotek	37
Punkt 8: Definiowanie lokalnych funkcji na typach anonimowych	43
Punkt 9: Zrozumienie zależności pomiędzy różnymi koncepcjami równości	49
Punkt 10: Zrozumienie pułapek metody GetHashCode()	59
Rozdział 2: Projektowanie API	67
Punkt 11: Unikanie operatorów konwersji w tworzonych API	67
Punkt 12: Stosowanie parametrów opcjonalnych w celu minimalizacji przeciążeń metod	72
Punkt 13: Ograniczanie widoczności naszych typów	76
Punkt 14: Preferowanie definiowania i implementowania interfejsów poprzez dziedziczenie	80
Punkt 15: Rozróżnienie pomiędzy metodami interfejsów a metodami wirtualnymi	89
Punkt 16: Implementowanie wzorca zdarzeń dla powiadomień	94
Punkt 17: Unikanie zwracania referencji do obiektów klas wewnętrznych	101

Punkt 18: Preferowanie przesłoneń dla obsługi zdarzeń	105
Punkt 19: Unikanie przeciążania metod zdefiniowanych w klasach bazowych	108
Punkt 20: Zdarzenia zwiększają powiązania między obiektami w czasie wykonywania	114
Punkt 21: Deklarowanie tylko niewirtualnych zdarzeń	116
Punkt 22: Tworzenie grup metod, które są jasne, minimalne i kompletne	123
Punkt 23: Metody częściowe dla konstruktorów, mutatorów i modułów obsługi zdarzeń	130
Punkt 24: Unikanie interfejsu ICloneable ze względu na ograniczenia możliwości projektowych	136
Punkt 25: Ograniczanie parametrów tablicowych do tablic params ...	140
Punkt 26: Natychmiastowe raportowanie błędów w iteratorach i metodach asynchronicznych	145
Rozdział 3: Programowanie asynchroniczne oparte na zadaniach	151
Punkt 27: Używanie metod asynchronicznych do pracy asynchronicznej	151
Punkt 28: Nigdy nie tworzymy metod asynchronicznych bez zwracanej wartości	156
Punkt 29: Unikanie łączenia metod synchronicznych i asynchronicznych	162
Punkt 30: Używanie metod asynchronicznych w celu uniknięcia alokowania wątków i przetaczania kontekstów ...	167
Punkt 31: Unikanie niepotrzebnego przekierowywania kontekstu ...	169
Punkt 32: Komponowanie pracy asynchronicznej przy użyciu obiektów Task	174
Punkt 33: Implementowanie protokołu anulowania zadania	180
Punkt 34: Buforowanie uogólnionych typów zwracanych asynchronicznie	188
Rozdział 4: Przetwarzanie równoległe	193
Punkt 35: Jak PLINQ implementuje algorytmy równoległe	193
Punkt 36: Pamiętanie o wyjątkach przy konstruowaniu algorytmów równoległych	206
Punkt 37: Używanie puli wątków zamiast tworzenia nowych wątków .	213
Punkt 38: Komunikacja międzywątkowa przy użyciu BackgroundWorker	219

Punkt 39: Wywołania międzywątkowe w środowiskach XAML	223
Punkt 40: Wykorzystanie lock() jako pierwszego wyboru dla synchronizacji.	233
Punkt 41: Używanie możliwie najmniejszego zasięgu dla uchwytów blokad	241
Punkt 42: Unikanie wywoływania nieznanego kodu w zablokowanych sekcjach	245
Rozdział 5: Programowanie dynamiczne	249
Punkt 43: Zalety i wady typowania dynamicznego	249
Punkt 44: Używanie typowania dynamicznego w celu użycia typu czasu wykonania w ogólnych parametrach typu. .259	
Punkt 45: Używanie DynamicObject lub IDynamicMetaObjectProvider dla typów dynamicznych	263
Punkt 46: Używanie Expression API.	274
Punkt 47: Minimalizowanie obecności obiektów dynamicznych w publicznych API	282
Rozdział 6: Uczestnictwo w globalnej społeczności C#	289
Punkt 48: Szukaj najlepszej odpowiedzi, a nie najbardziej popularnej . .289	
Punkt 49: Uczestniczenie w specyfikacjach i kodowaniu	291
Punkt 50: Automatyzowanie najlepszych praktyk przy użyciu analizatorów.	293
Indeks.	295