

Spis treści |

O autorze	11
O recenzentach	12
Wstęp	13

CZĘŚĆ 1. Prezentacja czystego kodu

ROZDZIAŁ 1

Co to takiego czysty kod i dlaczego powinno Ci na nim zależeć?	19
O czym będzie ta książka?	20
Wyjaśnienie, czym jest czysty kod	21
Znaczenie czystego kodu dla zespołów	22
Znaczenie czystego kodu we własnych projektach	24
Podsumowanie	25

ROZDZIAŁ 2

Kto może decydować o tym, jakie są „dobre praktyki”?	26
Kto w ogóle decyduje o tych sprawach?	27
Najlepsze rozwiązania — skąd tak naprawdę się biorą?	28
Zasady wzorców projektowych	28
Świadomość sytuacji	33
Zachowaj konsekwencję, a szybciej uzyskasz rezultaty	35
O narzędziach do analizy kodu	36
O testowaniu i jego wielu formach	37
Podsumowanie	39

ROZDZIAŁ 3

Programuj, nie rób akrobacji	41
Czym jest kod?	41
Trochę historii	42
Przeznaczenie kodu	42

Pisz zrozumiale, a nie sprytnie	43
Uwagi na temat możliwości utrzymania kodu	45
Używanie operatorów binarnych i zapisu ósemkowego, szesnastkowego i dwójkowego	46
Nadawanie wartości zmiennym i stosowanie instrukcji goto	46
Przesadne komentowanie	47
Korzystanie z operatorów trójargumentowych	47
Stosowanie skrótów	48
Wprowadzanie w kodzie mikrooptymalizacji	48
Programowanie na nowo metod z biblioteki SPL	48
Podsumowanie	49

ROZDZIAŁ 4

Tu chodzi o coś więcej niż sam kod	51
PHP jako ekosystem	51
Wybór właściwych bibliotek	53
Parę słów o wersjonowaniu semantycznym	55
Czym jest wersjonowanie semantyczne?	56
Jak sobie radzić z wersjonowaniem semantycznym	57
Stabilność kontra trendy	58
Podsumowanie	60

ROZDZIAŁ 5

Optymalizacja czasu pracy i rozdzielenie odpowiedzialności	61
Konwencje nazewnnicze i organizacyjne	61
Pliki klas i interfejsów	62
Pliki wykonywalne	62
Elementy zawartości i zasoby sieci WWW	62
Nazewnictwo klas, interfejsów i metod	63
Nazewnictwo folderów	63
Rozdzielenie odpowiedzialności	64
Rozsyłanie zdarzeń	65
Objaśnienie polimorfizmu — interfejsy i klasy abstrakcyjne	67
Interfejsy	67
Klasy abstrakcyjne	68
Podsumowanie	69

ROZDZIAŁ 6

PHP ewoluuje — deprecjacje i rewolucje	71
Nowe wersje PHP w porównaniu ze starymi	71
Ścisła kontrola typów	72
Raportowanie błędów	72
Atrybuty	73
Przełom w wersji 8	74
Konstrukcja match	74
Argumenty nazwane	75
Klasy i właściwości tylko do odczytu	75
Migrowanie zasobów do odpowiednich klas	76
Ochrona wrażliwych argumentów przed wyciekami	78
Podsumowanie	79

CZĘŚĆ 2. Utrzymywanie jakości kodu

ROZDZIAŁ 7

Narzędzia jakości kodu	83
Wymagania techniczne	83
Sprawdzanie składni i stylu kodu	84
Linter wbudowany w PHP	84
PHP CS Fixer: szperacz kodu	85
Statyczna analiza kodu	90
phpcpd — wykrywacz kopiowania i wklejania kodu	90
PHPMD: wykrywacz bałaganu w PHP	92
PHPStan — analizator statyczny dla języka PHP	98
Psalm — maszyna lintująca do analizy statycznej dla PHP	104
Rozszerzenie środowisk IDE	108
PHP Inspections (EA Extended)	110
Intelephense	113
Podsumowanie	115
Materiały dodatkowe	116

ROZDZIAŁ 8

Wskaźniki jakości kodu	117
Wymagania techniczne	117
Prezentacja wskaźników jakości kodu	118
Aspekty jakości oprogramowania	118
Wskaźniki jakości kodu	119

Zbieranie wskaźników w PHP	123
phploc	124
PHP Depend	126
PhpMetrics	128
Zalety i wady korzystania ze wskaźników	135
Zalety	135
Wady	136
Podsumowanie	137
Materiały dodatkowe	137

ROZDZIAŁ 9

Organizacja narzędzi jakości PHP	138
Wymagania techniczne	138
Instalowanie narzędzi jakości kodu przy użyciu menedżera Composer	138
Instalowanie narzędzi jakości kodu z wykorzystaniem sekcji require-dev	139
Instalacja globalna	141
Skrypty menedżera Composer	142
Instalowanie narzędzi jakości kodu jako plików phar	143
Utrzymywanie plików phar w porządku	144
Zarządzanie plikami phar przy użyciu programu Phive	145
Dodawanie menedżera Phive do projektu	147
Podsumowanie	147

ROZDZIAŁ 10

Testowanie automatyczne	149
Wymagania techniczne	150
Dlaczego potrzebujesz testów automatycznych?	150
Łatwiejsza refaktoryzacja dzięki testom	152
Typy testów automatycznych	153
Testy jednostkowe	153
Testy integracyjne	157
Testy E2E	159
Piramida testowa w praktyce	161
Uwagi o pokryciu kodu	162
Zapoznanie z pokryciem kodu	162
Jak generować raporty o pokryciu kodu	163
Korzystanie z adnotacji @covers	167
Co testować?	167
Podsumowanie	169
Materiały dodatkowe	169

ROZDZIAŁ 11

Ciągła integracja	171
Wymagania techniczne	171
Dlaczego ciągła integracja jest potrzebna?	172
Koszty błędu	172
Jak zapobiegać błędom	173
Prezentacja ciągłej integracji	174
Potok budowy	175
Etap 1. — budowanie projektu	176
Etap 2. — analiza kodu	178
Etap 3. — testy	179
Etap 4. — wdrożenie	180
Integrowanie potoku z przepływem pracy	182
Budowanie potoku w narzędziu GitHub Actions	182
GitHub Actions w skrócie	183
Etap 1. — budowanie projektu	184
Etap 2. — analiza kodu	187
Etap 3. — testy	189
Etap 4. — wdrożenie	190
Integrowanie potoku z własnym przepływem pracy	191
Twój lokalny potok — haki narzędzia Git	194
Konfigurowanie haków narzędzia Git	195
Haki narzędzia Git w praktyce	197
Zaawansowane użycie	199
Dygresja — wprowadzanie ciągłej integracji do istniejącego oprogramowania	200
Krok po kroku	201
Spojrzenie na ciągłe dostarczanie	202
Podsumowanie	203
Materiały dodatkowe	204

ROZDZIAŁ 12

Praca w zespole	205
Wymagania techniczne	205
Standardy pisania kodu	206
Podążanie za istniejącymi standardami	206
Zasady pisania kodu	209
Przykładowe zasady pisania kodu	210
Ustalanie zasad	217
Przeglądy kodu	218
Dlaczego należy dokonywać przeglądów kodu	219
Co powinny obejmować przeglądy kodu?	219

Najlepsze rozwiązania dotyczące przeglądów kodu	220
Zapewnienie przeprowadzania przeglądów kodu	222
Definicja ukończenia	223
Wnioski na temat przeglądów kodu	224
Wzorce projektowe	224
Zapoznanie ze wzorcami projektowymi	224
Wzorce projektowe często spotykane w języku PHP	225
Antywzorce	236
Podsumowanie	238
Materiały dodatkowe	239

ROZDZIAŁ 13

Tworzenie efektywnej dokumentacji	240
Wymagania techniczne	240
Dlaczego dokumentacja ma znaczenie?	240
Dlaczego dokumentacja jest ważna?	241
Dokumentacja dla programistów	242
Tworzenie dokumentacji	243
Dokumenty tekstowe	243
Diagramy	244
Generatory dokumentacji	246
Dokumentacja w kodzie źródłowym	251
Adnotacje nie są kodem	251
Nieczytelny kod	251
Nieaktualne komentarze	252
Bezużyteczne komentarze	252
Błędne lub nieprzydatne sekcje DocBlock	253
Komentarze TODO	253
Kiedy komentowanie jest przydatne?	253
Testy jako dokumentacja	254
Podsumowanie	255
Materiały dodatkowe	255