

Jakub Vrána

1001 TIPŮ A TRIKŮ PRO

PHP



CD obsahuje

veškeré zdrojové kódy z knihy,
instalaci Apache, MySQL
a další užitečné nástroje

Nejlepší postupy a řešení pro vaše webové aplikace

- ▲ Využití AJAXu, návrhových vzorů, export a import dat
- ▲ Zabezpečení dat, ladění výkonu aplikace
- ▲ Práce se soubory, XML dokumenty, knihovnamy a frameworky
- ▲ Regulární výrazy, zpracování formulářů a e-mailů

Jakub Vrána

1001 tipů a triků pro PHP

Computer Press
Brno
2012

1001 tipů a triků pro PHP

Jakub Vrána

Obálka: Martin Sodomka

Odpovědný redaktor: Martin Herodek

Technický redaktor: Jiří Matoušek

Objednávky knih:

<http://knihy.cpress.cz>

www.albatrosmedia.cz

eshop@albatrosmedia.cz

bezplatná linka 800 555 513

ISBN 978-80-251-2940-1

Vydalo nakladatelství Computer Press v Brně roku 2012 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 16097.

© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

Dotisk 1. vydání.

 **ALBATROS** MEDIA a.s.

Stručný obsah

Úvod	33
Přehledný kód	35
Běžové prostředí	47
Instalace a konfigurace	53
Syntaxe jazyka	75
Jazykové konstrukce	89
Proměnné	103
Pole	121
Objektové programování	131
Návrhové vzory	153
Regulární výrazy	159
Webové aplikace	167
Zpracování formulářů	189
Práce se soubory	209
Databáze	227
Ukládání dat	271
Vícejazyčné aplikace	289
E-mail	303
AJAX	317
Práce s obrázky	321
XML dokumenty	325
Export a import dat	335
Knihovny a frameworky	347
Bezpečnost aplikace	355
Zabezpečení dat	377
Výkonnost	395
Hotové aplikace	417
Vývojové prostředí	423
Vývoj PHP	437
Poslední tip	443

Obsah

Úvod	33
Přehledný kód	35
1 Jak je to s velikostí písmen	35
2 Jak je to s velikostí písmen u zkratek	35
3 Jak je to s velikostí písmen v URL	35
4 Diakritika v identifikátorech	36
5 Diakritika v komentářích	36
6 Rozmístění tříd do souborů	36
7 Pojmenování rozhraní	36
8 Co to je značka Byte-Order-Mark	36
9 Čeština nebo angličtina?	36
10 Jazyk textů a komentářů	37
11 Pojmenování funkcí	37
12 Název aplikace v identifikátorech	37
13 Velikost písmen v SQL dotazech	38
14 Jak vypadá Spaghetti code	38
15 Jak na odsazování kódu	38
16 Znak pro odsazování kódu	39
17 Odsazování konstrukce switch	39
18 Odsazování kódu v kombinaci s HTML	39
19 Zarovnávání operátorů	40
20 Odřádkování před operátorem	40
21 Jak na konce řádků	41
22 Konec řádku na konci souboru	41
23 Ukončení posledního prvku pole	41
24 Pořadí parametrů funkcí	41
25 Pojmenování parametrů funkcí	42
26 Závorky kolem ternárního operátoru	42
27 Použití apostrofy nebo uvozovky?	42
28 Řídící konstrukce s bloky kódu	43
29 Volání funkce před její definicí	43
30 V jakém pořadí definovat metody	44
31 Funkce vypisující data	44
32 Postradatelné proměnné a funkce	44

	Běhové prostředí	47
33	Použití PHP pro webové aplikace	47
34	Použití PHP z příkazové řádky	47
35	Syntaktická kontrola skriptu	47
36	Jiná konfigurace při spuštění z příkazové řádky	47
37	Spuštění PHP kódu bez vytváření souboru	48
38	Interaktivní vyhodnocování kódu	48
39	Vstup a výstup z příkazového řádku	48
40	Chyba skriptu spuštěného z příkazového řádku	48
41	Použití chybového výstupu	49
42	Načtení vstupu z webové služby	49
43	Zápis na výstup z webové aplikace	49
44	Načtení konfigurace PHP	50
45	Detekce běhového prostředí	50
46	Definice konfiguračního souboru	50
47	Pravidelné spuštění úloh	51
48	Pravidelné spuštění úloh z webu	51
49	Aktuální adresář při spuštění skriptu	51
50	Co je to PHP-GTK	51
51	Jak na PHP pro Javu	52
52	Jak na PHP pro .NET	52
	Instalace a konfigurace	53
53	Jak zjistit verzi PHP	53
54	Aktuální verze PHP	53
55	Nastavení konfigurace PHP	53
56	Konfigurace pro adresář	53
57	Centrální konfigurace pro adresář	54
58	Vynucení konfigurace	54
59	Použití konstant v konfiguraci PHP	54
60	Použití jednotek v konfiguraci PHP	54
61	Pravdivostní hodnoty konfiguračních direktiv	55
62	Otevírání vzdálených souborů	55
63	Vkládání vzdálených souborů	55
64	HTML nebo XHTML?	56
65	Výběr kódování	56
66	Nastavení kódování	56
67	Předefinování řetězcových funkcí	56

68	Jak na hlášení chyb	57
69	Práce s neinicializovanými proměnnými	57
70	Proměnná s chybovou hláškou	58
71	Využití obsluhy chyby pro získání chybové hlášky	58
72	HTML kód v chybách	58
73	Chyby v SQL dotazech	59
74	Instalace PHP na vlastním počítači	59
75	Instalace PHP na IIS	60
76	Instalace PHP a aplikací na IIS	60
77	Jaká je licence PHP	60
78	Jaká je licence MySQL	60
79	Nahrávání extenzí	61
80	Název souboru s extenzí	61
81	Vývojové a produkční prostředí	61
82	Odkazy z chybových hlášek	62
83	Konfigurace aplikace specifická pro server	62
84	Povolení ladícího režimu	62
85	Nevhodné povolení ladícího režimu	63
86	Registrace proměnných zvenku	63
87	Pořadí proměnných zvenku	64
88	Automatické ošetřování vstupu od uživatele	64
89	Vypnutí automatického ošetřování vstupu od uživatele	64
90	Automatické ošetření načítaných dat	65
91	Automatické zahájení session	65
92	Platnost session proměnných	65
93	Mazání starých session dat	66
94	Přidání session identifikátoru do URL	66
95	Odmítnutí session identifikátoru z URL	66
96	Konstanta se session identifikátorem	66
97	Vytváření session identifikátoru	67
98	Oddělovač parametrů URL	67
99	Nastavení limitů PHP	67
100	Nastavení omezení paměti	68
101	Co je to bezpečný režim	68
102	Omezení adresáře skriptů	69
103	Umístění dočasných souborů	69
104	Jak skrýt přítomnost PHP	69
105	Vložení souboru do všech skriptů	69

106	Konfigurace e-mailu	70
107	Jak nastavit časové pásmo	70
108	Konfigurace MySQL	70
109	Co je to MySQLnd	70
110	Zastaralé direktivy	71
111	Jaký použít oddělovač adresářů	71
112	Oddělovač cest	71
113	Vypsání XML značky	71
114	Použití konstant pro vlastní konfiguraci	72
115	Konstanty bez rozlišení velikosti písmen	72
116	Definice absolutní cesty	72
117	Definice absolutního URL	73
118	Kde je uložena cesta k sobě	73
119	Instalace vlastní aplikace	73
	Syntaxe jazyka	75
120	Možnosti uzavírání PHP kódu	75
121	Vynechání koncové značky	75
122	Konec řádku za koncovou značkou	75
123	Ukončení příkazu	75
124	Ignorování zbytku souboru	75
125	Ukončení příkazu pro ignorování zbytku souboru	76
126	Druhy komentářů	76
127	Ukončení jednořádkového komentáře	76
128	Dokumentační komentáře	77
129	Značky dokumentačních komentářů	77
130	Jak na „Nutno dodělat“	77
131	Zpracování dokumentačních komentářů	78
132	Jak na anotace	78
133	Co je operátor identity	78
134	Jaký použít operátor nerovnosti	78
135	Pořadí porovnávání	79
136	Co je to ternární operátor	79
137	Zkrácený ternární operátor	79
138	Jak používat operátor plus	79
139	Pořadí násobení a dělení	80
140	Bitový posun	80
141	Obvyklý bitový posun	80
142	Bitová negace	80

143	Definice konstant	81
144	Použití konstant	81
145	Nedefinované konstanty	81
146	Konstanty obsahující pole	81
147	Jak uložit návratovou hodnotu funkce	82
148	Interní funkce s proměnným počtem parametrů	82
149	Předefinování funkce	83
150	Definice výjimky	83
151	Rozšíření výjimky	83
152	Probublání výjimky	84
153	Eskalace výjimky	84
154	Příklad chyb na výjimky	85
155	Typy výjimek v PHP	85
156	Zobrazení typu výjimky	85
157	Obsluha nezachycené výjimky	86
158	Skrytí chyb	86
159	Zobrazení všech chyb	87
160	Úklid po chybě	87
	Jazykové konstrukce	89
161	Univerzálnost cyklu for	89
162	Došli jsme na konec cyklu?	89
163	Počet průchodů cyklem	89
164	Jak použít operátor čárka	90
165	Alternativní syntaxe řídicích konstrukcí	90
166	Řetězení podmíněného příkazu	90
167	Vyvolání chyby v podmíněném příkazu	90
168	Pořadí bloků podmíněného příkazu	91
169	Zkrácené vyhodnocování podmínek	91
170	Jaké zvolit pořadí podmínek	92
171	Opakované vyhodnocování podmínky	92
172	Jaké použít logické operátory	92
173	Priorita logických operátorů	92
174	Priorita operátoru zřetězení	93
175	Vyskočení z nadřazeného cyklu	93
176	Co jsou speciální funkce	93
177	Používat include nebo require?	94
178	Kdy používat include_once?	94
179	Kontrola opakovaného vložení zevnitř knihovny	94

180	Ukončení vkládání souboru	95
181	Testování úspěšnosti vložení souboru	95
182	Využití návratové hodnoty vložených souborů	95
183	Globální proměnné ve vkládaných souborech	96
184	Ukončení funkce bez návratové hodnoty	96
185	Inicializace statických proměnných	96
186	Hromadné přiřazení proměnných	97
187	Jakou použít koncovku u vkládaných souborů	97
188	Vkládání souboru z aktuálního adresáře	98
189	Vložení souboru z adresáře zpracovávaného skriptu	98
190	Použití jmenného prostoru	98
191	Co je to aserce	99
192	Funkce s libovolným počtem parametrů	99
193	Co je to anonymní funkce	100
194	Jak používat uzávěry	100
195	Vytvoření funkce z řetězce	101
196	Vyhodnocení kódu v řetězci	101
197	Omezené vyhodnocení kódu v řetězci	101

Proměnné **103**

198	Rozsah viditelnosti proměnných	103
199	Přístup ke globálním proměnným z funkce	103
200	Reference a globální proměnné	104
201	Proměnné proměnné	104
202	Proměnné s nepovolenými znaky	104
203	Proměnné s rozšířenými znaky	105
204	Je proměnná nastavena?	105
205	Jsou proměnné nastavené?	105
206	Zrušení proměnné	106
207	Přiřazení proměnných hodnotou	106
208	Přiřazení proměnných referencí	106
209	Předání parametrů referencí	107
210	Předání parametrů referencí při volání	107
211	Přiřazení objektů referencí	107
212	Volitelný parametr předávaný referencí	108
213	Jak použít přiřazení jako výraz	108
214	Zachování typu proměnné	108
215	Zachování významu proměnné	109
216	Zjištění typu proměnné	109

217	Ověření typu řetězec	109
218	Nastavení typu proměnné	109
219	Převod na pravdivostní hodnotu	110
220	Jaké používá PHP velikosti čísel	110
221	Jak pracovat s přesnými čísly	110
222	Bitový posun	110
223	Zápis hexadecimálních čísel	111
224	Zápis čísel v osmičkové soustavě	111
225	Zápis čísel ve vědecké notaci	111
226	Převod uživatelského vstupu na číslo	111
227	Kontrola číselného řetězce	111
228	Používání čísel v kódu	112
229	Nastavení jazyka	112
230	Řetězec jako posloupnost bajtů	112
231	Přístup k jednomu znaku v řetězci	113
232	Binární bezpečnost řetězcových funkcí	113
233	Speciální znaky v řetězci	113
234	Znaková sada a kódování	113
235	Jak na převod kódování	114
236	Délka řetězce v kódování UTF-8	114
237	Samotné zpětné lomítko v řetězci	114
238	Jak na víceřádkové řetězce	114
239	Co je to heredoc	115
240	Řetězec bez interpretace speciálních znaků	115
241	Přístup k prvkům pole uvnitř řetězce	116
242	Přístup ke složitějším strukturám z řetězce	116
243	Přístup k libovolným strukturám z řetězce	116
244	Je řetězec neprázdný?	116
245	Porovnávání řetězců	117
246	Porovnání řetězců bez rozlišení velikosti písmen	117
247	Porovnání číselných řetězců	118
248	Porovnání prvních několika znaků řetězce	118
249	Končí text tímto řetězcem?	118
250	Záměna více řetězců najednou	118
251	Záměna více bajtů najednou	119
252	Ošetřování parametrů funkcí	119
253	Ošetření chyby při deserializaci proměnných	119

	Pole	121
254	Sjednocení polí	121
255	Spojení polí	121
256	Přidání prvku na konec pole	121
257	Přidání prvku na začátek pole	121
258	Vložení prvků doprostřed pole	122
259	Jaký je typ klíčů pole	122
260	Načítání unikátních identifikátorů	123
261	Procházení polí	123
262	Procházení pole referencí	123
263	Získání náhodného prvku pole	124
264	Jak realizovat víceúrovňová pole	124
265	Výchozí parametry funkce v poli	124
266	Třídění polí	125
267	Stabilita třídění	125
268	Třídění českých polí	125
269	Zřetězení prvků pole	126
270	Obalení prvků pole řetězcem	126
271	Výchozí hodnota nenastaveného prvku pole	126
272	Vyhledání hodnoty v poli	127
273	Získání prvního prvku pole	127
274	Získání libovolného prvku pole	127
275	Průchod více poli současně	128
276	Pole pevně dané velikosti	128
277	Jak implementovat frontu a zásobník	128
278	Rychlejší fronta a zásobník	129
279	Jak detekovat pole	129
	Objektové programování	131
280	Vytvoření objektu bez parametrů konstruktoru	131
281	Jak na konstruktor v PHP 4	131
282	Jak na konstruktor od PHP 5	131
283	Zakázání vytvoření objektu	131
284	Co je to vícenásobný konstruktor	132
285	Volání destrukturu při chybě	133
286	Přístup k objektu vrácenému funkcí	133
287	Co je to standardní třída	133
288	Přetypování na objekt	133
289	Jak se odkázat na aktuální objekt	134

290	Zjištění názvu třídy	134
291	Inicializace vlastností	134
292	Veřejné vlastnosti	135
293	Neveřejné vlastnosti v PHP 4	135
294	Deklarace více vlastností	135
295	Jak definovat interní veřejné metody	135
296	Převod na řetězec	136
297	Explicitní převod na řetězec	136
298	Práce s neinicializovanými vlastnostmi	137
299	Vyvolání chyby při zápisu do neinicializované vlastnosti	137
300	Vlastnosti pouze pro čtení	137
301	Jak na serializaci objektů	138
302	Jak na deserializaci objektů	139
303	Jak exportovat objekty	139
304	Jak klonovat objekty	139
305	Co je mělká kopie objektu	140
306	Co je hluboká kopie objektu	140
307	Automatická hluboká kopie objektu	141
308	Předávání metod k zavolání	141
309	Předávání statických metod k zavolání	141
310	Automatické nahrávání tříd	142
311	Jmenné prostory při automatickém nahrávání tříd	142
312	Více funkcí pro nahrávání tříd	142
313	Deserializace objektů bez definovaných tříd	142
314	Viditelnost protected	143
315	Viditelnost členů jiných objektů stejné třídy	143
316	Viditelnost pro přátele	143
317	Využití abstraktních tříd	144
318	Určení typu parametru	145
319	Co je to rozhraní	145
320	Implementace více rozhraní	146
321	Změna parametrů zděděných metod	146
322	Procházení objektů	147
323	Přístup k indexu objektu	148
324	Jak zjistit počet prvků objektu	148
325	Objekt jako pole	149
326	Jak zavést silnou kontrolu typů	149
327	Kontrola skalárních typů v parametrech	149

328	Co je to reflexe	150
329	Využití reflexe pro testování	150
330	Objekty jako klíče pole	150

Návrhové vzory **153**

331	Singleton	153
332	Důkladný Singleton	153
333	Registry	154
334	Registry s odloženou inicializací	154
335	Factory	155
336	Observer	156
337	Model-View-Controller	156
338	Active record	157

Regulární výrazy **159**

339	Knihovny regulárních výrazů	159
340	Oddělovač regulárních výrazů	159
341	Zpětné lomítko v regulárním výrazu	159
342	Nadbytečná zpětná lomítka	160
343	Speciální znaky v regulárním výrazu	160
344	Libovolný počet opakování	160
345	Konec řetězce v regulárním výrazu	161
346	Co jsou to třídy znaků	161
347	Nalezení výčtu znaků v řetězci	161
348	Kontrola alfanumerických znaků	161
349	Jak vyhledat všechna písmena	162
350	Ošetření textu v regulárním výrazu	162
351	Nalezené číslo řádku	162
352	Nalezení více regulárních výrazů	162
353	Neuložení nalezeného výskytu	163
354	Pojmenování nalezených výskytů	163
355	Zvýraznění odkazů v textu	163
356	Zkrácení odkazů v textu	164
357	Jak zjistit počet provedených záměn	165
358	Co jsou hladové kvantifikátory	165
359	Aserce v regulárních výrazech	165

	Webové aplikace	167
360	Struktura adresy URL	167
361	Předání parametrů v URL	167
362	Speciální znaky v názvech parametrů	168
363	Odstranění parametru z URL	168
364	Sestavení parametrů URL	168
365	Odkaz na první stránku	169
366	Odkazy na části stránky	169
367	Absolutní a relativní odkazy	169
368	Použití relativních odkazů	170
369	Jak používat stavové hlavičky	170
370	Alternativní způsob nastavení stavových hlaviček	171
371	Okamžik odeslání hlaviček	171
372	Poslání stavového kódu v případě chyby	171
373	Způsoby přesměrování	172
374	Jak adresovat výchozí stránku aplikace	172
375	Přesměrování na domácí stránku aplikace	172
376	Jaká je struktura souboru pro roboty	173
377	Skrytí stránek před vyhledávači	173
378	Co znamená bezstavovost webových aplikací	173
379	Uložení informací o uživateli do cookies	174
380	Nastavení a přečtení cookie	174
381	Práce s odeslanou cookie	174
382	Odstranění cookie	174
383	Použití cookies pro předvyplnění formuláře	175
384	Nastavení doby platnosti cookies	175
385	Prodloužení doby platnosti cookie	175
386	Velikost cookies	176
387	Cookies pro více adresářů	176
388	Cookies pro více domén	177
389	Cookies třetích stran	177
390	Co jsou session proměnné	177
391	Nastavení a přečtení session proměnné	177
392	Trvalé nastavení filtru uživatele	178
393	Automatické vložení konce stránky	178
394	Umístění vkládaných souborů do jiného adresáře	179
395	Umístění vkládaných souborů do podadresáře	179
396	Jak vytvořit pěknou URL	179

397	Přepis URL	180
398	Přepis pěkných URL	180
399	Přesměrování starých URL	180
400	Automatické vytvoření pěkného URL	181
401	Pěkná URL s identifikátorem	182
402	Přepis všech URL	182
403	Změna systému URL	183
404	Jak na přesměrování domén	183
405	Doména s www nebo bez?	183
406	Pěkná URL pro chudé	184
407	Pěkná URL přes chybovou stránku	184
408	Jak nastavit chybovou stránku	184
409	Velikost písmen v URL	185
410	Návrat na předchozí stránku	185
411	Detekce zastaralého odkazu	186
412	Automatické zastavení skriptu	186
413	Externí vyhledávání na stránkách	186
414	Vyhledávání v placených stránkách	187
415	Jak správně realizovat demoverzi	187
416	Zjištění aktuální verze	188
	Zpracování formulářů	189
417	Rozdíl mezi metodami GET a POST	189
418	Přístup k datům od uživatele	189
419	Přístup k datům od uživatele z funkcí	190
420	Přesměrování po odeslání formuláře metodou POST	190
421	Zobrazení informace o provedené operaci	191
422	Uložení informace o provedené operaci	191
423	Předání informace o provedené operaci	192
424	Kód přesměrování po odeslání formuláře	192
425	Formát hlavičky Location	193
426	Hlavička Location se zabezpečenými stránkami	193
427	Hlavička Location s číslem portu	193
428	Přesměrování na část dokumentu	193
429	Typ dat z formuláře	194
430	Prázdná hodnota z formuláře	194
431	Odeslání pole	195
432	Zpracování odeslaného pole	195
433	Indexace prvků v odeslaném poli	195

434	Složitější index v odeslaném poli	196
435	Odeslání formuláře na sebe sama	196
436	Metoda GET přepíše parametry v URL	196
437	Detekce odeslání formuláře	197
438	Jak nastavit název odesílacího tlačítka	197
439	Jak realizovat více odesílacích tlačítek	198
440	Odesílací tlačítko pro každý řádek	198
441	Jak realizovat košík s produkty	199
442	Ukládání košíku do session proměnné	199
443	Ukládání košíku do cookie	199
444	Ukládání košíku do databáze	200
445	Přidání položky do košíku JavaScriptem	200
446	Jak zjistit celkovou cenu košíku	201
447	Převod košíku do databáze	202
448	Opakované odeslání formuláře	202
449	Náhled komentáře	203
450	Smazání odeslaného komentáře	204
451	Logování IP adresy	205
452	Logování identifikátoru prohlížeče	205
453	Zobrazení IP adresy	205
454	Jak vytvořit výběrový seznam	206
455	Jak vytvořit výběrový seznam se skupinami	207
456	Zpracování formuláře po automatickém odhlášení	207
457	Jaká je maximální délka řetězců	208
458	Vypnutí doplňování hodnot	208
	Práce se soubory	209
459	Získání celého obsahu souboru	209
460	Jak určit délku načtených dat	209
461	Vypsání obsahu souboru	209
462	Jak detekovat konce řádků	209
463	Jak zjistit čitelnost souboru	210
464	Cache informací o souboru	210
465	Zapsání proměnné do souboru	210
466	Co je atomicita operací	210
467	Jak na zamykání souborů	211
468	Atomická práce se soubory	211
469	Datum poslední modifikace aktuálního souboru	211
470	Vytvoření dočasného souboru	211

471	Práce s daty v paměti	212
472	Procházení adresářů	212
473	Rekurzivní procházení adresářů	212
474	Vestavené rekurzivní procházení adresářů	213
475	Průchod souborů ve více adresářích	213
476	Poslání dat metodou POST	213
477	Získání souboru přes proxy server	214
478	Přihlašovací údaje ke vzdálenému souboru	214
479	Získání vrácených hlaviček	214
480	Stažení souboru extenzí CURL	214
481	Přenos cookies při stahování souboru	215
482	Ruční zpracování cookies	215
483	Stažení souboru na pozadí	216
484	Ruční získání HTTP odpovědi	216
485	Ruční stažení souboru	217
486	Zpracování HTML stránky	217
487	Zjištění kódování HTML stránky	218
488	Zjištění jazyka HTML dokumentu	219
489	Zjištění kódování českého textu	219
490	Zapsání binárních dat	220
491	Načtení binárních dat	220
492	Práce s komprimovanými soubory	220
493	Nahrávání souborů	221
494	Práva k nahraným souborům	221
495	Nahrání více souborů	221
496	Přesunutí souboru od uživatele	222
497	Kontrola nahraného souboru	222
498	Původní název nahraného souboru	222
499	Načtení souboru od uživatele	222
500	Vlastník nahraného souboru	223
501	Výchozí práva vytvářených souborů	223
502	Velikost nahrávaného souboru	223
503	Zjištění koncovky souboru	223
504	Ukládání souborů do databáze	224
505	Uložení souboru od uživatele do databáze	224
506	Načítání souborů z databáze	224
507	Ukládání souborů do PostgreSQL	225
508	Stažení části souboru	225

	Databáze	227
509	Nezávislost na databázovém systému	227
510	Výběr databázového systému	227
511	Jednotná práce s různými databázovými systémy	227
512	Co umožňuje extenze PDO	228
513	Ošetření chyb v PDO	228
514	Co umožňuje knihovna Dibi	229
515	Co umožňuje knihovna NotORM	229
516	Přenášení nevyužitých dat z databáze	230
517	Co je to ORM	230
518	Jak používat operátor OR v MySQL	231
519	Hledání v seznamu hodnot	231
520	Zapisování identifikátorů	231
521	Ošetření identifikátorů	232
522	Co umožňuje databáze information_schema	232
523	Vestavěná databáze	232
524	Vlastní funkce v SQLite	233
525	Pevná struktura databáze	233
526	Pojmenování databázových tabulek	233
527	Využití typu int unsigned	233
528	Uložení desetinných čísel	234
529	Využití typu char	234
530	Maximální délka řetězce	234
531	Zadání neplatných dat	234
532	Zobrazení varování	234
533	Využití typu enum	235
534	Jaký je typ dat pocházejících z databáze	235
535	Formát kalendářního data	235
536	Nastavení časového pásma	236
537	Přehledný zápis SQL dotazů	236
538	Názvy sloupců při spojení více tabulek	237
539	Seznam sloupců v dotazu INSERT	237
540	Pevný název tabulky	237
541	Co dělá hvězdička v dotazu SELECT	238
542	Způsoby spojení tabulek	238
543	Význam hodnoty NULL	238
544	K čemu slouží indexy	238
545	Co jsou unikátní indexy	239

546	Využití unikátních klíčů	239
547	Co je to primární klíč	240
548	Co jsou automatické identifikátory	240
549	Jak na víceloupcové indexy	240
550	Existuje záznam?	241
551	Zjištění počtu řádků	241
552	Zjištění celkového počtu řádků	242
553	Jak realizovat stránkování	242
554	Omezení počtu stránek	243
555	Trvalé odkazy na stránkování	243
556	Počet zobrazovaných záznamů	244
557	Existuje další stránka?	244
558	Získání počtu měněných záznamů	245
559	Mezery v SQL příkazu	245
560	Postupné přenášení dat z databáze	246
561	Komunikace s databází během postupného přenášení dat	246
562	Hromadné vložení více záznamů	246
563	Položení více dotazů najednou	247
564	Zpracování vícenásobného výsledku	247
565	Asynchronní spuštění dotazu	248
566	Přístup k datům vráceným z databáze	248
567	Uložení vráceného řádku do proměnných	249
568	Automatické připojení k databázovému serveru	249
569	Určení kódování přenášených dat	249
570	Výchozí kódování přenášených dat	250
571	Ošetření řetězců	250
572	Ohraničování řetězců	250
573	Ošetření hodnot	250
574	Ošetření proměnných v PDO	251
575	Vázání proměnných	251
576	Vázání proměnných v MySQLi	252
577	Vázání proměnných v PDO	252
578	Zástupný znak ve vázání proměnných	253
579	Emulace vázání proměnných	253
580	Kdy nelze použít vázání proměnných	254
581	Uvolnění výsledku dotazu	254
582	Zavření připojení k databázovému serveru	254
583	Práce s více databázemi	255

584	Více připojení k databázovému serveru	255
585	Více připojení ke stejnému serveru	255
586	Co je to replikace MySQL	256
587	Jak vyřešit zpoždění replikace	256
588	Funkce pro vytvoření seznamu hodnot	257
589	Pořadí záznamu	258
590	Konstantní počet dotazů	258
591	Fronta požadavků	259
592	Co je perzistentní připojení	259
593	Omezení počtu perzistentních připojení	259
594	Úklid perzistentního připojení	260
595	Jak na perzistentní připojení v MySQLi	260
596	Jak na obnovení připojení	260
597	Jaká nabízí MySQL úložišťe	260
598	Jak realizovat transakce	261
599	Získání dat pro úpravu	262
600	Chyby v transakcích	262
601	Zamykání tabulek	264
602	Využití cizích klíčů	264
603	Rekurzivní mazání odkazovaných záznamů	264
604	Vypnutí referenční integrity	265
605	Prohledávání databáze	265
606	Jak vytvořit fulltextový index	266
607	Operátory fulltextové vyhledávání	266
608	Fulltextový index s tabulkami InnoDB	266
609	Vyhledávání ve více tabulkách	267
610	Fulltextové vyhledávání Sphinx Search	267
611	Sphinx Search pro češtinu	268
612	Nalezení textu pomocí Sphinx	268
613	Zvýraznění nalezeného textu Sphinx	269
	Ukládání dat	271
614	Určení pořadí záznamů	271
615	Vlastní pořadí záznamů	271
616	Ukládání odpovědí ankety	271
617	Ukládání hlasů v anketě	272
618	Změna hlasu v anketě	272
619	Vložení složitějšího objektu do textu	273
620	Záměna značky v textu na objekt	273

621	Složení článku z elementárních objektů	274
622	Zjištění počtu přístupů po dnech	274
623	Import dat do databáze	275
624	Aktualizace importovaných dat	275
625	Přepsání importovaných dat	276
626	Smazání starých dat při importu	276
627	Uložení PSČ	277
628	Zobrazení PSČ	277
629	Kontrola rodného čísla	277
630	Kontrola IČ	278
631	Odkaz na počáteční písmena	278
632	Výpis záznamů od daného písmene	279
633	Zvýraznění aktivních počátečních písmen	280
634	Zobrazení otevírací doby	280
635	Je otevřeno?	281
636	Zobrazení otevíracích hodin	281
637	Zvýraznění dnešní otevírací doby	282
638	Jak ukládat parametry produktů	282
639	Zobrazení parametrů	282
640	Uložení pravdivostních hodnot	283
641	Práce s datem	283
642	Jak realizovat sezónní ceníky	284
643	Zjištění aktuální sezónní ceny	284
644	Zjištění ceny za více období	284
645	Zobrazení kalendáře	284
646	Získání části data	285
647	Posun v čase	285
648	Uložení nastavení do databáze	286
649	Co je to tag cloud	286
650	Jak ukládat hierarchická data	287
651	Logaritmičké odsazování diskusí	287
	Vícejazyčné aplikace	289
652	Internacionalizace a lokalizace	289
653	Formátování výstupů	289
654	Formátování čísel	289
655	Formátování čísla s pevnou mezerou	290
656	Formátování data	290
657	Převod data z mezinárodního formátu	290

658	Zpracování formátovaného data	291
659	Jak zobrazit české názvy měsíců	291
660	Jak zobrazit české názvy dní v týdnu	292
661	Zobrazení českého formátu data	292
662	Zobrazení relativního českého data	293
663	Jak zvolit identifikátor překladů	294
664	Překlady obsahující proměnnou	294
665	Vyhledání textů k překladu	294
666	Označení textů k překladu	295
667	Překlad textů ve zdrojovém kódu	296
668	Jednotné a množné číslo	296
669	K čemu slouží knihovna Gettext	297
670	Načítání překladů z databáze	298
671	Doplňování překladů do databáze	298
672	Jak překládat texty s vazbou na data	299
673	Odlišná data pro každý jazyk	299
674	Stejná data pro všechny jazyky	299
675	Práce s nepřeloženými texty	300
676	Jak předat adresu jazykových verzí	300
677	Jak předat kód jazyka	301
678	Detekce jazyka uživatele	301
679	Zobrazení detekované jazykové verze	302
680	Přepnutí jazykové verze	302
	E-mailý	303
681	Skrytí e-mailové adresy	303
682	Kontrola e-mailové adresy	303
683	Odesílání e-mailů	304
684	Oddělovač hlaviček	304
685	Čeština v těle zprávy	305
686	Čeština v hlavičkách zprávy	305
687	E-mail v kódování UTF-8	305
688	Jakou strukturu má patička zprávy	306
689	E-mailý ve formátu HTML	306
690	Převod HTML na text	307
691	Připojení souboru k e-mailu	307
692	Odeslání sestavené zprávy	308
693	Obrázky v e-mailu	308
694	Vložení obrázku do e-mailu	308

695	Odpovědní formulář	309
696	Kontakt na uživatele bez vyzrazení jeho e-mailu	310
697	Potvrzení e-mailové adresy	310
698	Upozornění na nové komentáře	311
699	Jedno upozornění na nové komentáře	311
700	Co zadat do adresy odesílatele	312
701	Notifikace o platbě	312
702	Stahování notifikací o platbě	312
703	Stažení seznamu plateb	313
704	Podepisování e-mailů	313
705	Podpis zprávy pomocí PGP	314
706	Podpis zprávy pomocí PKCS #7	314
707	Šifrování e-mailů	315
	AJAX	317
708	Využití AJAXu	317
709	Jak zjistit stav aplikace	317
710	Informování uživatele o provádění požadavku	317
711	Pořadí zpracování požadavků	318
712	Minimalizace počtu požadavků	318
713	Typ požadavků na server	318
714	Alternativní formát odesílaných dat	319
715	Formát odpovědi ze serveru	319
716	Identifikace požadavků AJAX	319
	Práce s obrázky	321
717	Přidání rozměrů obrázků	321
718	Doplnění rozměrů obrázků do HTML kódu	321
719	Co jsou to gravatary	322
720	Jaké volit formáty obrázků	322
721	Načtení obrázku v libovolném formátu	322
722	Jak vytvářet grafy	323
723	Co jsou to QR kódy	323
724	Vytvoření QR kódu	324
	XML dokumenty	325
725	Co obsahuje hlavička XML dokumentů	325
726	Načtení RSS exportu	325
727	Pravidelné načítání RSS exportu	325

728	Informování o RSS exportu	325
729	RSS export vyhledávání	326
730	Zrušení RSS exportu	326
731	Počet položek v RSS exportu	326
732	Co je to PubSubHubbub	327
733	Data načtená pomocí SimpleXML	327
734	Vyhledávání v SimpleXML	328
735	Test HTML atributu class	328
736	Získání textového obsahu značky	328
737	Jmenné prostory XML	329
738	Zjištění pořadí ve výsledcích vyhledávání	329
739	Entity při načtení XML dokumentů	330
740	Kontrola XML podle DTD	330
741	Kódování při zpracování HTML dokumentů	331
742	Vnucení kódování při zpracování HTML dokumentů	331
743	Zpracování velkých XML dokumentů	331
744	Převod XML dokumentu	332
745	Vytváření XML dokumentů	333
746	Úprava XML dokumentů	333
	Export a import dat	335
747	Co je to JSON	335
748	Načtení dat ve formátu JSON	335
749	Ošetření proměnné pro JavaScript	335
750	Využití protokolu data	336
751	Export do CSV	336
752	Jak nastavit oddělovač v CSV	336
753	Co nabízí formát Excelu	337
754	Export do Excelu	337
755	Export do šablony Excelu	337
756	Co nabízí formát PDF	337
757	Export do PDF	338
758	Převod HTML do PDF	338
759	Import kurzů měn	338
760	Jak vytvořit mapu serveru	339
761	Priorita v mapě serveru	340
762	Vytvoření mapy serveru	340
763	Mapa serveru z databáze	340
764	Odkaz na mapu serveru	341

765	Velikost mapy serveru	341
766	Export zboží	342
767	Formát exportu zboží	342
768	Vygenerování exportu zboží	342
769	Export geografických dat	343
770	Co umožňují webové služby	343
771	Položení XML-RPC požadavku	344
772	Co je WSDL dokument	344
773	Připojení k SOAP serveru	344
774	Vytvoření SOAP serveru	345
775	Ladění SOAP požadavků	345

Knihovny a frameworky 347

776	Inteligentní objekty	347
777	Jak na ladění aplikací	347
778	Jak vytvořit inteligentní formuláře	348
779	Zadávání formátovaného textu	349
780	Kopírování formátovaného textu do Texy	349
781	Vizuální editory formátovaného textu	349
782	Vyčištění HTML kódu	350
783	Opravení HTML kódu	350
784	Jak použít šablony Smarty	350
785	Dědičnost šablon	351
786	Co je Nette Latte	351
787	Použití Nette Latte s atributy	352
788	Zvýraznění zdrojového kódu	352
789	Zvýraznění sebe sama	352
790	Geolokace z IP adresy	353
791	Zkrácení adresy	353
792	Zjištění cíle zkráceného odkazu	353

Bezpečnost aplikace 355

793	Validace dat	355
794	Sanitizace dat	355
795	Cross-site scripting	355
796	Vytvoření HTML entit	356
797	Množina ošetřovaných dat	356
798	Nedůvěryhodné proměnné serveru	356
799	Odstranění značek	357

800	Využití šablon	357
801	Kontextově citlivé ošetřování dat	357
802	Oddělení HTML a PHP kódu	357
803	Určení kódování stránky	358
804	Kontrola kódování dat	358
805	Okamžik ošetřování dat	358
806	Jak na filtrování dat	359
807	Nastavení výchozího filtru	359
808	Vypnutí výchozího filtru	359
809	K čemu slouží blacklist a whitelist	360
810	Vkládání souborů	360
811	Inicializace proměnných	360
812	Nepodmíněná inicializace proměnných	361
813	Cookies neviditelné z prohlížeče	362
814	Jak vytvořit bezpečné cookies	362
815	Co je to Cross-Site Request Forgery	362
816	Obrana proti CSRF	363
817	Obrana proti CSRF pomocí hlavičky Referer	363
818	Obrana proti CSRF využitím parametru URL	364
819	Rozsah obrany proti CSRF	364
820	Obrana proti CSRF při e-mailových operacích	364
821	Co je to ClickJacking	365
822	Co je to Session Hijacking	365
823	Skrytí adresy stránky při navštívení odkazu	366
824	Jak zjistit totožnost uživatele	366
825	Co je to Session Fixation	367
826	Podvržení hlaviček	367
827	Pseudonáhodná a náhodná čísla	367
828	Inicializace náhodných čísel	368
829	Používání pseudonáhodných čísel	368
830	Skutečně náhodné číslo	368
831	Ukládání souborů od uživatele	369
832	Ověření názvu souboru od uživatele	369
833	Kontrola souborů od uživatele	369
834	Posílání souborů od uživatele	370
835	Omezení počtu operací podle IP adresy	370
836	Uložení informace o proxy serveru	371
837	Co jsou to reverzní proxy servery	371

838	Pozor na veřejné proxy servery	371
839	Jak odhalit Tor	371
840	Rozlišení robota a člověka	372
841	Zmatení robota	372
842	Detekce člověka pomocí JavaScriptu	373
843	Nespoléhat se na CAPTCHA	373
844	Zavření prohlížeče	373
845	Uživatel pro práci s databází	374
846	Více uživatelů na jednom serveru	374
847	Oddělení více uživatelů mimo PHP	374
848	Zakázání spuštění externích programů	374
849	Co je to penetrační testování	375

Zabezpečení dat

377

850	Co je to autentizace a autorizace	377
851	Způsoby autentizace	377
852	Způsoby získání hesla od uživatele	377
853	Základní HTTP autentizace	377
854	Pokročilá HTTP autentizace	378
855	Zobrazení přihlašovacího formuláře	378
856	Volitelné přihlášení	379
857	Umístění přihlašovacího formuláře	379
858	E-mail jako přihlašovací jméno	379
859	Co je to Security by Obscurity	380
860	K čemu slouží hašovací funkce	380
861	Ukládání hesel	380
862	Jak na omezení hesla	381
863	Jak kontrolovat složitost hesla	381
864	Bezpečnost hašovacích funkcí	382
865	Bezpečnější ukládání hesel	382
866	Hláška při zadání neplatného hesla	383
867	Opakované zadání neplatného hesla	383
868	Poslání zapomenutého hesla	384
869	Změna hesla	385
870	Předvyplnění hesla	385
871	Vypnutí doplnění přihlašovacího jména a hesla	386
872	Zadání části hesla	386
873	Přihlášení na více doménách	386
874	Přihlášení na více doménách s využitím JavaScriptu	387

875	Ukládání čísel kreditních karet	387
876	Vytvoření klíče pro asymetrickou kryptografii	388
877	Kdy použít protokol HTTPS	389
878	Jak na důvěryhodné certifikáty	389
879	Protokol výzva – odpověď	389
880	Proč použít role	391
881	Okamžik kontroly oprávnění	391
882	Reakce na chybějící oprávnění	392
883	Uložení session dat	392
884	Skrytí zdrojových kódů	394
885	Uložení přihlašovacích údajů k databázi	394
	Výkonnost	395
886	Jak použít HTTP hlavičky	395
887	K čemu slouží hlavička Expires	395
888	K čemu slouží hlavička Last-Modified	395
889	Jak zajistit vždy aktuální data	396
890	K čemu slouží hlavička If-Modified-Since	396
891	Čas poslední modifikace v databázi	396
892	Poslední modifikace souvisejících objektů	397
893	K čemu slouží hlavička ETag	397
894	K čemu slouží hlavička If-None-Match	397
895	Privátní cache	398
896	Cache při používání session proměnných	398
897	Pořadí předávaných parametrů	398
898	Reverzní proxy servery	398
899	HTTP hlavičky v požadavku	398
900	Bufferování výstupu	400
901	Bufferování celé stránky	400
902	Vysypání výstupu	401
903	Komprese výstupu	401
904	Komprese statických souborů	401
905	Jak zjistit aktuální datum a čas	401
906	K čemu slouží akcelerátory	402
907	Vyhodnocení podmínky cyklu	402
908	Automatické nahrávání tříd	402
909	Minimalizace kódu	403
910	Vlastní minimalizace kódu	403
911	Jak vytvořit archiv PHP skriptů	404

912	Načtení dat z PHP archivu	404
913	Vytvoření PHP archivu	404
914	Vytvoření PHP archivu s webovou aplikací	404
915	Převod PHP kódu do C++	405
916	Měření rychlosti	405
917	Měření rychlosti webové aplikace	405
918	Co je to profilování	406
919	Jak zjistit množství zabrané paměti	406
920	Chytřejší uvolňování paměti	406
921	Informování uživatele o průběhu operace	406
922	Pořadí zpracování šablon	406
923	Kopírování při zápisu	407
924	Kopírování s ternárním operátorem	407
925	Kopírování a reference	408
926	Uvozovky nebo apostrofy	408
927	Mikrooptimalizace	408
928	Prohledávání pole	409
929	Vyhledání více hodnot v poli	409
930	Ukládání do souborové cache	409
931	Jak funguje sdílená paměť	410
932	Vzdálená sdílená paměť	410
933	Ukládání session dat do Memcache	411
934	Zamykání session souborů	411
935	Ukládání session souborů do adresářů	411
936	Ruční mazání zastaralých session souborů	411
937	Někdy se dá obejít bez session proměnných	412
938	Jak realizovat frontu požadavků	412
939	Kešování SQL dotazů	412
940	Využití transakcí	413
941	Rychlý commit, pomalý rollback	413
942	Synchronizace disku při dokončení transakce	413
943	Zobrazení vysokého čísla stránky	414
944	Paralelní zpracování	414
945	Spouštění programů na pozadí	415
	Hotové aplikace	417
946	MediaWiki	417
947	WordPress	417
948	phpBB	418

949	phpMyAdmin	419
950	Adminer	419
951	Adminer Editor	420
952	Rozšíření pro Adminer	421
953	Synchronizace databáze	421
	Vývojové prostředí	423
954	Proč používat systém pro správu verzí	423
955	Co je to Subversion	423
956	Co je to Git	424
957	Správa požadavků	424
958	Propojení správy požadavků s verzováním	424
959	Propojení Subversion se správou požadavků	424
960	Použití verzovacího systému pro nasazení aplikace	425
961	Co je to Phing	425
962	Balíčky operačního systému	426
963	Prostředí pro vývoj svobodného softwaru	426
964	Testování aplikací	426
965	Testování pomocí PHPUnit	427
966	Testování pomocí PHPT	427
967	Vlastní spuštění testů	427
968	Co je to Selenium IDE	428
969	Co je to Selenium RC	428
970	Pokrytí kódu	429
971	Ladicí výpis proměnných	430
972	Formátovaný ladicí výpis proměnných	430
973	Ladicí informace v hlavičkách	431
974	Ladění pomocí FirePHP	431
975	Ladění aplikací	431
976	Připojení k FTP	432
977	K čemu slouží editor a vývojové prostředí	432
978	Co nabízí editor SciTE	432
979	Jak vypadá ikona PHP skriptů	433
980	Jak nastavit vlastní písmeno disku pro skripty	433
981	Asociace PHP skriptů	434
982	Asociace PHP skriptů pro Firefox	434
983	Jak na zálohování	434

	Vývoj PHP	437
984	Přímé odkazy do PHP manuálu	437
985	Poznámky dokumentace	437
986	Zrcadla stránek PHP	437
987	Vývojová verze dokumentace	438
988	Jak na hlášení chyb	438
989	Proč používat diskusní fóra	438
990	Kde hledat zdrojové kódy PHP	439
991	Jaká je architektura PHP	439
992	Jak realizovat více vláken	440
993	Tvorba extenzí	440
994	Křížová reference zdrojových kódů	440
995	K čemu slouží testy PHP	440
996	Zdrojové kódy dokumentace	441
997	Sestavení dokumentace	441
998	Pseudotypy používané v dokumentaci	441
999	Překlad oficiální dokumentace	442
1000	České weblogy o PHP	442
	Poslední tip	443
1001	Používejte vámi vytvářené aplikace	443
	Rejstřík	445

Úvod

Kromě toho, že se živím programováním, tak se také podílím na tvorbě oficiální dokumentace PHP a učím základy vytváření webových stránek na vysoké škole. Vedle toho píšu *blog.php.vrana.cz* a vedu několik školení o tvorbě webových aplikací se zaměřením na PHP, MySQL a JavaScript. Ze všech těchto vstupů čerpám inspiraci, a proto doufám, že všechny tipy a triky v této knize budou pro čtenáře nějakým způsobem zajímavé. S vatovými triky následujícího typu se tedy v této knize rozhodně nesetkáte:

1. Věděli jste, že oblouk nakreslíte funkcí `imageArc`?
2. Věděli jste, že obdélník nakreslíte funkcí `imageRectangle`?
3. Věděli jste, že čáru nakreslíte funkcí `imageLine`?

Místo toho najdete třeba i tipy týkající se výkonnosti webových aplikací, verzovacích systémů nebo dokonce vývoje PHP. I to je totiž pro programátora v PHP důležité.

Knihy u čtenáře předpokládá znalost syntaxe PHP a schopnost dohledávat si informace v dokumentaci. Pokud tedy nějaký tip např. pojednává o zajímavém využití nějaké funkce, tak se nevysvětluje význam jednotlivých parametrů, protože ten je popsán v dokumentaci.

Knihy byla napsána v době PHP 5.3, v době zrušení vývojové verze PHP 6, která měla zavést podporu pro Unicode. Zabývá se tedy verzemi PHP 5.3 a staršími.

Kompletní text knihy lze prohledávat pomocí hledání na webu *php.vrana.cz*.

Komu je kniha určena

Tipy jsou rozděleny do tří kategorií pro začátečníky, pokročilé a znalce. Rozdělení ale není ani tak podle náročnosti, jako spíš podle jejich potřebnosti.



začátečník

Tipy pro začátečníky by měl znát i začínající programátor (i když někdy mohou být složitější).



pokročilý

Tipy pro pokročilé by měla znát většina programátorů PHP.



znalec

Bez znalosti tipů pro znalce se dá často obejít, i když samozřejmě mohou poskytnout nějakou zkratku nebo usnadnění práce.

Odlišení písma

V knize se používá tučné písmo pro názvy produktů, kurziva pro odborné termíny, neproporcionální písmo pro ukázky kódu. V ukázkách kódu jsou tučně klíčová slova a vestavěné funkce, kurzivou proměnné.

Poděkování

Chtěl bych poděkovat Davidu Grudlovi, který je autorem několika knihoven použitých v této knize a také spoluautorem kódu u několika tipů. Také mu děkuji za řadu cenných připomínek.

Doprovodné CD

Doprovodný disk obsahuje kromě zdrojových kódů také řadu odkazů na užitečné stránky a také několik užitečných nástrojů, jež vám programování v jazyce PHP výrazně usnadní nebo alespoň zpříjemní.

CD stačí vložit do počítače a rozhraní se spustí automaticky. Pokud nemáte automatické spouštění disků povoleno, vyhledejte na CD kořenový adresář a otevřete soubor `spustit_CD.html`.

Jestliže rozhraní CD otevřete v prohlížeči Internet Explorer, Opera nebo Google Chrome, budete z CD moci instalovat doprovodný software okamžitě. V případě jiných prohlížečů se zobrazí výzva k uložení instalačního souboru na pevný disk. V tomto případě doporučujeme spustit instalaci přímo z CD. Obsah CD najdete ve složce obsah.

Přehledný kód

1 Jak je to s velikostí písmen



V důsledku toho, že PHP převzalo řadu funkcí a obrátů z jiných programovacích jazyků, nemá jednotné pravidlo pro používání velkých a malých písmen. Alespoň některá pravidla jsou ale společná:

- Proměnné a funkce je zvykem začínat malým písmenem.
- Názvy tříd obvykle začínají velkým písmenem.
- Konstanty se píšou celé velkými písmeny a slova se oddělují podtržítkem.

Nejednotnost tkví hlavně v oddělování slov. Většina vestavěných funkcí PHP používá pro oddělení slov podtržítko (např. `array_keys`), pokud to ovšem nejsou zkratky (jako např. `strpos`). Některé funkce se toho ale nedrží a slova nijak neoddělují (např. `imageCreate`).

Objektový kód obvykle slova odděluje zvětšením písmene (např. `PDO::errorCode`), i když ani to neplatí vždy (`mysqli::select_db`).

U funkcí a tříd PHP velikost písmen nerozlišuje, u proměnných ale ano. V této knize budeme pro oddělování slov používat zvětšení písmene, z důvodu lepší čitelnosti i u vestavěných funkcí PHP.

2 Jak je to s velikostí písmen u zkratk



Pokud je součástí identifikátoru (např. proměnné nebo funkce) nějaká zkratka, je vhodné ji zapisovat malými písmeny stejně jako ostatní slova. Jinak bude identifikátor hůře čitelný, obzvlášť v situaci, kdy by zkratek bylo více za sebou.

```
<?php
// dobře čitelné
createXmlHttpRequest($xmlHttp);

// špatně čitelné, proměnné by navíc měly začínat malým písmenem
createXMLHttpRequest($XMLHTTP);
?>
```

3 Jak je to s velikostí písmen v URL



Adresu URL je zvykem zapisovat malými písmeny a slova oddělovat spojovníkem, stejný způsob pojmenování tedy volíme u skriptů přímo zobrazovaných na webu.

U souborů vkládaných do jiných skriptů odvíjíme název souboru od toho, co v souboru definujeme. Pokud to je např. třída, tak soubor můžeme pojmenovat stejně jako třídu, včetně velikosti písmen.

4 Diakritika v identifikátorech



PHP dovoluje v názvech proměnných a v dalších identifikátorech používat i znaky z horní poloviny ASCII tabulky. Pokud se tedy rozhodneme zadávat identifikátory česky, můžeme je psát i s diakritikou. V praxi se toho ale téměř nevyužívá, protože tento způsob zápisu nemusí podporovat všechny editory.

5 Diakritika v komentářích



Pokud se rozhodneme komentáře k programu psát česky, měli bychom zásadně používat diakritiku (háčky a čárky). Souvislejší texty bez diakritiky se totiž velmi špatně čtou. Zdrojové kódy se v poslední době obvykle píšou v kódování UTF-8.

6 Rozmístění tříd do souborů



Každá třída se obvykle umísťuje do samostatného souboru pojmenovaného podle jejího názvu. Toto pravidlo se občas porušuje u definic výjimek, které se buď definují hromadně v jednom souboru, nebo spolu s třídou, která je vyvolává.

7 Pojmenování rozhraní



Název rozhraní (interface) v některých projektech začíná znakem I, zdaleka to ale není pravidlem. Stejně jako třídy se rozhraní obvykle umísťují do souborů pojmenovaných podle jejich názvu.

8 Co to je značka Byte-Order-Mark



Soubory ve znakové sadě Unicode mohou začínat značkou *Byte-Order-Mark* (BOM). To je posloupnost bajtů na začátku souboru, která obsahuje informaci o použitém kódování. Platí to i pro kódování UTF-8, které lze používat pro vytváření PHP skriptů. Tuto značku ale PHP nijak neinterpretuje, takže ji není vhodné používat. Jinak může dojít k tomu, že např. funkce header skončí chybou, protože už skript odeslal nějaký výstup (neviditelnou značku BOM) nebo že skript odešle těchto značek více (pokud do sebe vkládáme více skriptů, které by měly značku uvedenou).

9 Čeština nebo angličtina?



Při programování bychom si měli vybrat nejen programovací jazyk (což je nevyhnutelné), ale i ten lidský, který budeme používat pro identifikátory a texty v programu. U identifikátorů, jako je název proměnných nebo funkcí, je obvykle přirozené používat angličtinu, protože i návazný kód používá angličtinu.

```
<?php
$xmlReader = new XmlReader; // působí přirozeně
$cteckaXml = new XmlReader; // působí krkolmně
?>
```

V některých případech se navíc používají ustálené obraty jako např. název metody `getConnection`. České varianty, ať už v podobě `getPripojeni` nebo třeba `dejPripojeni`, působí nepřirozeně.

V knize tedy budeme používat anglické identifikátory.



Poznámka: Stejně pravidlo lze použít i pro pojmenování souborů.

10 Jazyk textů a komentářů



začátečník

U vypisovaných textů je situace daná tím, zda vytváříme jednojazyčnou nebo vícejazyčnou aplikaci. U jednojazyčné je přirozené psát texty rovnou ve výsledném jazyce, u vícejazyčné potom v angličtině (pokud je jedním z výsledných jazyků).

U komentářů záleží na tom, kdo s nimi bude pracovat. Pokud všichni autoři i čtenáři komentářů ovládají dostatečně dobře angličtinu, tak je vhodné psát komentáře právě v ní, aby byl jazyk v souladu s identifikátory. Pokud taková situace nenastane, tak je vhodné psát komentáře jazykem, který všichni zúčastnění ovládají nejlépe.

Vypisované texty a komentáře v knize budou tedy česky.

11 Pojmenování funkcí



začátečník

Název funkcí a metod je vhodné začínat slovesem určujícím, co přesně vlastně funkce nebo metoda dělá. Např. `getConnection` je lepší název než samotné `connection`, protože je z názvu jasné, že jde o získání připojení, a ne např. o jeho nastavení. Slovesem je vhodné název funkce vždy začínat, `getConnection` je lepší než `connectionGet`. U neobjektového kódu je možné názvu funkce předřadit modul, kterého se týká (např. `imageCreate`), u metod objektů to není potřeba.

12 Název aplikace v identifikátorech



pokročilý

Budoucí verze PHP mohou používat nová klíčová slova nebo jiné identifikátory, které by mohly kolidovat s identifikátory použitými naší aplikací. V zájmu dopředné kompatibility bychom tedy v hlavním jmenném prostoru aplikace neměli používat příliš obecné identifikátory. Např. PHP 5.1 zavedlo třídu `Date` (ta byla po sporech nakonec přejmenována na `DateTime`). Stejně tak mohou nové identifikátory zavádět používané knihovny.

Předřazovat všem identifikátorům název aplikace ale také není zrovna nejšťastnější, protože to identifikátory prodlužuje a zpřehledňuje.

```
<?php
// v budoucnu může kolidovat
class Database {
}

// dlouhý a nepřehledný identifikátor
class MyAppDatabase {
}
```

?>

Čisté řešení tohoto problému nabízí jmenné prostory, které jsou k dispozici od PHP 5.3.

```
<?php
namespace MyApp;
class Database {
}
new Database; // použití zevnitř jmenného prostoru
new \MyApp\Database; // použití kdekoliv
?>
```

13 Velikost písmen v SQL dotazech



začátečník

V SQL dotazech je zvykem klíčová slova uvádět velkými písmeny a vlastní identifikátory (především názvy tabulek a sloupců) malými písmeny. Slova se obvykle oddělují podtržítkem. Někdo se kloní k oddělování slov zvětšením písmene stejně jako v PHP kódu, s tím ale mohou být za určitých okolností problémy (např. MySQL v závislosti na konfiguraci může názvy tabulek převádět na malá písmena).

Pokud se u identifikátorů rozhodneme používat češtinu, je zvykem ji psát bez diakritiky, i když by si většina databázových serverů poradila i s ní. V komentářích tabulek a sloupců je naopak diakritiku vhodné používat.

14 Jak vypadá Spaghetti code



pokročilý

O špagetovém kódu mluvíme tehdy, když jsou jednotlivé části aplikace promíchané tak, že se v nich dá jen těžko vyznat. U webových aplikací je obzvláště jednoduché takový kód vytvořit, protože se v nich používá řada jazyků zapisovaných v prostém textu:

- **JavaScript** je vhodné vyčlenit do samostatného souboru, který z HTML odkážeme značkou `<script src="">`. Přímou v HTML kódu je vhodné uvádět pouze inicializaci proměnných JavaScriptu, které jsou specifické pro daného uživatele.
- **CSS** je také vhodné dát do samostatného souboru a odkázat ho značkou `<link rel="stylesheet" href="">`. Provázání s HTML kódem zajišťují názvy značek, tříd a identifikátorů.
- **HTML** je vhodné nemíchat s řídicím PHP kódem, ale vyčlenit ho do samostatné šablony. Ta může využívat nějaký šablonovací systém nebo může být zapsaná také v PHP, vždy ale platí, že už slouží jen k vypsání dat.
- **SQL** je užitečné centralizovat do tzv. *modelu* (což je část aplikace zajišťující především získávání a ukládání dat), případně se bez něj díky ORM můžeme dokonce obejít.

Výhodou rozdělení aplikace do jednotlivých částí je jejich menší složitost a snazší udržitelnost. Umožňuje to také větší specializaci vývojářů, kdy nehrozí, že HTML kodér poškodí práci PHP programátora.



Poznámka: Některé ukázky kódu v knize se této zásady nebudou kvůli jednoduchosti držet.

15 Jak na odsazování kódu



Bílé znaky jako mezery a konce řádků se sice v PHP až na výjimky ignorují, i tak je ale vhodné kvůli čitelnosti dodržovat pevná pravidla jejich zápisu. Každý příkaz se obvykle píše na jeden řádek – dlouhé příkazy lze rozdělit, ale nikdy bychom neměli na jeden řádek dávat příkazů více. Bloky kódu je zvykem odsazovat, otevírací závorka se umísťuje obvykle na konec předchozího řádku, u funkcí a tříd ji lze dát i na samostatný řádek.



Poznámka: V této knize se budeme zásad správného formátování kódu držet, otevírací složená závorka bude vždy na konci předchozího řádku.

16 Znak pro odsazování kódu



O znacích používaných pro odsazování kódu se vedlo mnoho bezvýsledných debat. Někdo používá dvě mezery, někdo čtyři, někdo trvá na osmi mezerách. Osobně používám tabulátor z těchto důvodů:

- Každý programátorský editor dovoluje nastavit jeho zobrazovanou šířku. Ostatní editory ho obvykle zobrazují jako osm mezer.
- Odsazení je otázkou jednoho stisknutí klávesy v libovolných podmínkách.
- V uloženém souboru zabírá tabulátor jen jeden bajt.

17 Odsazování konstrukce switch



Formátování konstrukce `switch` je ošemetné, protože vytváří dvě úrovně. Někdo doporučuje zarovnávat `case` na stejnou úroveň jako `switch`, někdo o úroveň dál. Pokud je ve větších `case` jen jeden příkaz, tak se mi zdá nejpřehlednější ho uvést rovnou na řádek `case`:

```
<?php
function numbers($number) {
    switch ($number) {
        case 0: return "nula";
        case 1: return "jedna";
        case 2: return "dva";
        default: return null;
    }
}
?>
```

18 Odsazování kódu v kombinaci s HTML



Pokud kombinujeme PHP kód s HTML, tak může být správné odsazování oříškem. Můžeme odsazovat buď pouze PHP kód, nebo pouze HTML, nejpřehlednější je ale asi odsazování obojího.

```
<!-- Odsazování pouze PHP -->
<?php foreach ($data as $row) { ?>
    <tr>
        <td><?php echo htmlspecialchars($row["name"]); ?></td>
```



```

        </tr>
<?php } ?>

<!-- Odsazování pouze HTML -->
<?php foreach ($data as $row) { ?>
<tr>
    <td><?php echo htmlspecialchars($row["name"]); ?></td>
</tr>
<?php } ?>

<!-- Kombinace odsazování -->
<?php foreach ($data as $row) { ?>
    <tr>
        <td><?php echo htmlspecialchars($row["name"]); ?></td>
    </tr>
<?php } ?>

```

Situace se ovšem zkomplikuje, když chceme v bloku PHP kódu vypsat neuzavřenou značku, kterou uzavřeme později. Tam kombinace odsazování příliš dobře nefunguje.

19 Zarovnávání operátorů



začátečník

S formátováním kódu není vhodné to přehánět. Někteří programátoři se vyžívají třeba v tom, že při více přiřazeních pod sebou zarovnávají operátor přiřazení:

```

<?php
$int  = 0;
$s    = "";
$array = array();
?>

```

S takovýmto odsazováním je spousta práce, která čitelnost žádným zásadním způsobem nezlepší. Problém nastane hlavně v případě, kdy chceme přiřadit další proměnnou s delším názvem – pak musíme zarovnání všude opravit.

Zcela tragicky takovéto zarovnání dopadne v případě, kdy se na zdrojový kód podíváme s proporcionálním písmem, které někteří programátoři používají (protože je čitelnější a vejde se ho na obrazovku víc).

```

<?php
$int  = 0;
$s    = "";
$array = array();
?>

```

Obrázek 1. Zobrazení kódu proporcionálním písmem v editoru SciTE

20 Odřádkování před operátorem



začátečník

Pokud by jeden příkaz vytvořil příliš dlouhý řádek, lze ho rozdělit na více řádků. Odřádkování je vhodné udělat před operátorem a pokračování příkazu odsadit:

```

<?php
$ma i l

```

```
->setFrom("alice@example.com")
->addTo($email)
->setSubject("Pokusná zpráva")
;
?>
```

21 Jak na konce řádků



Na Windows se řádky ukončují znaky CR+LF (v PHP řetězci se zapisuje jako `\r\n`), na Linuxu pouze znakem LF (`\n`). PHP koncům řádek nepřikládá zvláštní význam, takže je mu jedno, co použijeme. Programátorské editory si poradí s oběma druhy konců řádků, primitivní textové editory (např. **Notepad** na Windows) vyžadují nativní konce řádků.

Kvůli menší velikosti se obvykle používá znak LF, každopádně je vhodné v celém souboru používat stejný znak.

22 Konec řádku na konci souboru



Poslední řádek souboru je vhodné ukončit koncem řádku (klávesa `Enter`). Usnadňuje to přidávání dalších příkazů a zpřehledňuje to rozdíly mezi verzemi, které něco doplňují na konec souboru.

23 Ukončení posledního prvku pole



Za posledním prvkem v definici pole je možné v PHP uvést čárku stejně jako za ostatními prvky. Velmi užitečné je to hlavně u definice složitějších polí, kde jednotlivé prvky napíšeme na samostatné řádky. Zjednodušuje se tím budoucí přidávání nebo prohazování prvků.

```
<?php
$array = array(
    "první",
    "druhý",
    "třetí",
);
?>
```

24 Pořadí parametrů funkcí



Při vytváření funkce s více než jedním parametrem bychom se měli zamyslet nad tím, v jakém pořadí budeme parametry předávat. U funkcí, které s něčím manipulují, je zvykem v prvním parametru předávat právě tuto hodnotu. Pokud funkce manipulují s objektem, tak je obvykle lepší funkci změnit na metodu tohoto objektu. Další parametry se obvykle uvádí od těch přesnějších po ty vágnější (nejdříve např. předáme ID a až potom název). Volitelné parametry uvádíme pochopitelně až na konci, jinak to v PHP ani nejde.

Funkce by každopádně neměly mít parametrů příliš mnoho, ukázkou z tohoto pohledu nevhodně navržené funkce je `setCookie`, která má 7 parametrů.

25 Pojmenování parametrů funkcí



PHP na rozdíl od některých jiných programovacích jazyků nedovoluje při volání funkce pojmenovat předávané parametry. Všechny parametry je potřeba předat v pořadí, které funkce definuje. Pokud má funkce více podobných parametrů za sebou, je vhodné si hodnotu parametru zapsat do proměnné s popisným názvem.

```
<?php
// nepřehledné
$sqlite->singleQuery($query, true, false);

// přehlednější
$firstRowOnly = true;
$decodeBinary = false;
$sqlite->singleQuery($query, $firstRowOnly, $decodeBinary);

// třetí možnost
$sqlite->singleQuery($query,
    $firstRowOnly = true,
    $decodeBinary = false
);

// využití komentáře
$sqlite->singleQuery($query,
    true, // firstRowOnly
    false // decodeBinary
);

// dá se využít i toho, že se řetězec převede na pravdivostní hodnotu
$sqlite->singleQuery($query, "firstRowOnly", false);
?>
```

Dobrý editor zdrojového kódu nám samozřejmě význam parametru odhalí, ale při prohlížení kódu nemusí být editor k dispozici (ukázky kódu v článcích, prohlížení kódu z repozitáře, atd.).

26 Závorky kolem ternárního operátoru



Kolem ternárního operátoru (neboli podmíněného výrazu) je vhodné dělat závorky, což zlepší jeho čitelnost:

```
<?php
$a = ($m ? f() : ($n ? g() : $o));
?>
```

Ještě přehlednější je v takovýchto případech použít řídicí konstrukci `if`.

27 Použití apostrofy nebo uvozovky?



PHP dovoluje pro zápis řetězců používat uvozovky nebo apostrofy. Uvnitř uvozovek lze používat proměnné a speciální znaky, uvnitř apostrofů ne. Při výběru zápisu bychom

se měli řídit hlavně přehledností. Pokud vypisovaný kód obsahuje uvozovky, tak bývá přehlednější ho uzavřít do apostrofů, jinak budeme v této knize používat uvozovky.

```
<?php
echo '<option value="">(vyberte)</option>';
echo "<div>$html</div>\n";
?>
```

28 Řídicí konstrukce s bloky kódu



Pokud za řídicími konstrukcemi jako `if`, `while` nebo `foreach` uvedeme jenom jeden příkaz, nemusíme ho uzavírat do bloku kódu (složených závorek). I tak je vhodné to ale udělat, protože se tím zlepší přehlednost kódu a usnadní se jeho další modifikace (přidání příkazu).

```
<?php
// přehledný a snadno rozšiřitelný kód
if ($valid) {
    f();
}

// funkce se zavolá vždy, protože se vykoná pouze prázdný příkaz (;)
if ($valid); // chyba
    f();
?>
```

29 Volání funkce před její definicí



PHP dovoluje použít funkci v kódu dříve, než je definovaná. Definice funkce totiž vzniká ve fázi kompilace skriptu, ale zavolá se až ve fázi spuštění. Lepší je ale obvykle funkci nejprve definovat a pak teprve zavolat. Pokud bychom se totiž v budoucnu rozhodli funkci přesunout do vloženého souboru, tak přestane fungovat (vkládané soubory se totiž kompilují až při spuštění hlavního skriptu).

```
<?php
// bude fungovat
function defineFirst() {
}
defineFirst();

// bude fungovat
runFirst();
function runFirst() {
}

// bude fungovat
include "defineFirst.inc.php";
defineFirst();

// nebude fungovat
runFirst();
include "runFirst.inc.php";
?>
```

Do souboru s definicí funkcí je lepší nedávat žádný globální kód. Když toto pravidlo, např. u jednoduchého skriptu spouštěného z příkazové řádky, porušíme a v jednom souboru nadefinujeme funkce i globální kód, tak je zvykem nejprve uvádět definice konstant a funkcí a pak teprve spouštěný kód.

30 V jakém pořadí definovat metody



Na pořadí definic metod stejně jako u funkcí nezáleží, kvůli čitelnosti je ale vhodné dodržovat určitá pravidla. Pořadí definic ve třídě může být takovéto:

- Nejprve vlastnosti v pořadí `public`, `protected`, `private`.
- Konstruktor, destruktorka a další magické metody.
- Další metody seřazené opět podle viditelnosti a podle logických celků.

31 Funkce vypisující data



Ve většině programovacích jazyků je lepší se vyhnout vytváření funkcí, které přímo vypisují nějaká data. Co kdybychom pak výstup chtěli použít někde jinde, třeba ho umístit do e-mailu? Místo toho proto funkce obvykle vrací řetězec, který vypíše až volající.

V PHP se velmi pohodlně kombinuje textový výstup s programovým kódem a jakýkoliv výstup lze snadno zachytit. Proto lze ospravedlnit i vytváření funkcí, které něco přímo vypisují. Pokud bychom náhodou s výstupem chtěli udělat něco jiného, můžeme použít *output buffering*:

```
<?php
ob_start();
phpInfo();
$phpInfo = ob_get_clean();
?>
```

32 Postradatelné proměnné a funkce



Pokud výsledek nějakého výpočtu používáme na více místech, je vhodné si ho přiřadit do proměnné. Pokud stejnou posloupnost příkazů používáme víckrát, je vhodné je uzavřít do funkce a volat tuto funkci. Když více funkcí pracuje se stejným objektem, je vhodné je uzavřít do třídy. Tyto zásady vedou k vytváření přehledného, výkonného a snadno udržitelného kódu.

Co když ale výsledek výpočtu používáme jen na jednom místě, co když funkci voláme jen jednou? Vyplatí se proměnnou nebo funkci vytvářet v takovém případě? Z pohledu běhu programu to je zbytečné, výkonnost to zcela nepatrně zhorší, ale program to obvykle zpřehlední – kus kódu tak vlastně dostane své jméno. Porovnejme následující tři ukázky kódu:

```
<?php
// nejefektivnější, ale nepřehledný kód
echo simplexml_load_file(mysql_result(mysql_query("
```

```
SELECT filename
FROM xml
WHERE id = $id
"), 0))->title;

// pojmenování jednotlivých výpočtů
$result = mysql_query("SELECT filename FROM xml WHERE id = $id");
$filename = mysql_result($result, 0);
$xml = simplexml_load_file($filename);
echo $xml->title;

// pojmenování celé operace
function getTitleFromXml($id) {
    $result = mysql_query("SELECT filename FROM xml WHERE id = $id");
    $filename = mysql_result($result, 0);
    $xml = simplexml_load_file($filename);
    return $xml->title;
}
echo getTitleFromXml($id);
?>
```

Postradatelné proměnné a funkce jsou vlastně způsob dokumentování kódu. Pokud si tedy nějaký blok kódu zaslouží komentář, může ho zpřehlednit také vyčlenění do funkce nebo pojmenování vypočtených hodnot.