Podręcznik PHP - wprowadznie

O podręczniku

Podręcznik ten nie jest "jeszcze jednym kursem PHP". To zbiorowe opracowanie, które nie tylko uczy, jak programować w języku PHP, lecz również pokazuje, jak robić to dobrze. Staramy się demonstrować najnowsze możliwości projektu, stawiając na aktualność. Podręcznik może być czytany przez wszystkich: zarówno przez osoby, które jeszcze nigdy nie miały do czynienia z programowaniem, jak i tych znających już jakieś języki.

PHP, choć na pierwszy rzut oka prosty do nauczenia, jest w rzeczywistości bardzo rozbudowanym projektem i zwyczajne wypisanie listy konstrukcji oraz funkcji nie wystarczy, aby powiedzieć o sobie "jestem doświadczonym programistą". Doświadczenie przychodzi z czasem, jednak w tym podręczniku staramy się to maksymalnie ułatwić, koncentrując się także na praktycznej stronie programowania. Pokazujemy, czego należy unikać, na co uważać, a w "Studiach przypadku" wyjaśniamy, jak zdobytą wiedzę wykorzystać do napisania skryptu realizującego konkretne zadanie.

Od czytelników tego podręcznika wymagamy pewnej znajomości języka HTML i orientowania się w tworzeniu stron WWW. Związane jest to z omawianym tu materiałem ukierunkowanym w głównej mierze na pisanie dynamicznych witryn internetowych. Jeżeli czujesz, że brakuje Ci wiedzy, zacznij swoją przygodę od poznania języka HTML. Znajomość programowania i algorytmów nie jest konieczna, aczkolwiek pozwoli na szybsze przyswajanie materiału.

Po przeczytaniu podręcznika będziesz wiedział:

- Co to jest PHP i jak działa?
- Jak postawić na własnym komputerze prosty serwer WWW.
- Jak pisać aplikacje w języku PHP
- Jak zarządzać bazami danych
- W jaki sposób napisać profesjonalny i łatwy w rozbudowie dynamiczny serwis WWW
- Jak poprawić bezpieczeństwo swych aplikacji i ustrzec się przed włamywaczami?

Podręcznik poprawiać i uzupełniać może każdy. Jeśli tylko masz czas i nie widzisz żadnych przeszkód w liberalnej licencji GNU FDL, pomóż w jego rozwoju: dopisz rozdział, zaktualizuj zawartość, popraw błędy (dalsze informacje). Każda pomoc jest ważna, gdyż tworzą to ludzie dla ludzi. Pamiętaj jednak, aby nie używać żadnych materiałów objętych prawem autorskim, a wszelkie większe edycje konsultować w dyskusji z innymi.

Czym jest PHP?

PHP to skryptowy język programowania - programy w nim napisane nie są kompilowane do postaci kodu maszynowego zrozumiałego dla procesora, lecz wykonywane przez specjalną aplikację zwaną interpreterem PHP. PHP został pierwotnie stworzony do wspomagania tworzenia dynamicznych stron WWW, lecz obecnie można w nim pisać także zwyczajne aplikacje dla systemu operacyjnego.

PHP jest projektem open-source. Każdy może pobrać za darmo jego kopię, zainstalować i używać bez żadnych ograniczeń. Do kodu źródłowego zapewniony jest pełny dostęp - jeżeli programista ma odpowiednie zdolności, może go modyfikować do woli oraz nadsyłać własne propozycje zmian do osób nadzorujących projekt. Dzięki takiej wolności PHP rozwija się bardzo dynamicznie, a w Internecie można znaleźć setki modyfikacji oraz dodatkowych modułów.

Mimo otwartości kodu, nad rozwojem oficjalnej wersji projektu czuwa firma *Zend Company* z Izraela założona przez twórców PHP. Zapewnia ona dodatkowe narzędzia i opiekę prawną, a także wyznacza kierunki rozwoju projektu.

Bardzo istotną cechą PHP jest skalowalność. Może go używać z powodzeniem zarówno początkujący programista, dzięki uproszczeniom składni, jak i ekspert, który znajdzie tutaj wszystkie zaawansowane narzędzia niezbędne do napisania rozbudowanej aplikacji. PHP znaleźć można wszędzie: od prywatnych stron domowych przez gry

internetowe aż do potężnych witryn korporacji lub portali. Opanowanie języka jest proste także dlatego, iż Internet pełen jest przewodników oraz artykułów pokazujących, jak w praktyce wykorzystywać wiele z jego możliwości.

Kiedy zaczynaliśmy pisać podręcznik, pod sam koniec 2005 roku, najnowszą dostępną wersją PHP była 5.1.1. Obecnie (koniec 2006 roku) dostępna jest wersja 5.2.0. Jej możliwości są zbliżone do tych oferowanych przez konkurencyjne języki skryptowe dla stron WWW: ASP, JSP. Staramy się, aby podręcznik opisywał możliwie najnowszą wersję, jednak z powodu dużej ilości materiału niekiedy mogą pojawiać się fragmenty odnoszące się do starszych wydań. W podręczniku **nie** opisujemy PHP 4.x, praktycznie nierozwijanego, ale wciąż popularnego na serwerach oraz wśród starszych programistów.

Jak PHP współpracuje ze stroną WWW?

PHP jest językiem server-side, tj. pracuje po stronie serwera WWW. Przeciwieństwem są języki client-side pracujące po stronie przeglądarki użytkownika (np. JavaScript). Aby wykorzystywać go na własnej stronie, musisz upewnić się, że twój serwer ma zainstalowaną jego obsługę. Zanim przejdziemy dalej, należy zrozumieć zasadę, na jakiej PHP generuje dynamiczne strony WWW.

Kiedy wpisujemy adres w przeglądarce internetowej, żądanie wyświetlenia strony kierowane jest do serwera HTTP zwanego także serwerem WWW. Jeśli stwierdzi na podstawie rozszerzenia pliku, że dany dokument zawiera kod PHP, serwer kieruje do jego interpretera żądanie przetworzenia podanego pliku. Interpreter wyszukuje w jego treści tzw. wstawki PHP wplecione w statyczny kod HTML i zastępuje je wynikiem ich wykonywania. Utworzony kod HTML jest zwracany serwerowi, a ten wysyła go z powrotem do internauty.

W tym procesie kod PHP nigdy nie opuszcza serwera. Internauta zawsze otrzyma wyłącznie utworzony przez PHP kod HTML. Oto przykład. Jeśli mamy plik PHP o następującej treści:

```
<?php
echo 'Podaj hasło';
?>
```

To internauta zobaczy jedynie dokument o takiej treści:

```
Podaj hasło
```

Cały PHP zniknie, a na jego miejscu pojawi się utworzony przez niego kod HTML.

Dzięki pracy po stronie serwera, PHP idealnie nadaje się do tworzenia złożonych aplikacji zarządzających dużymi ilościami danych: forami dyskusyjnymi, systemami zarządzania treścią, sklepami internetowymi. Generują one odpowiedni kod HTML dla przeglądarek internautów, a w momencie, kiedy oni go przeglądają, PHP już zakończył nad nim swą pracę. Jest to bardzo istotne, ponieważ wszelkie dalsze reakcje na poczynania użytkownika należy albo pozostawić przeglądarce, albo obsłużyć je za pomocą języka JavaScript.

W PHP stworzono m.in. aplikację MediaWiki, za pomocą której podręcznik ten jest oficjalnie dostępny w ramach projektu WikiBooks. PHP zarządza tutaj pobieraniem treści żądanej strony z bazy, sformatowaniem jej, a w przypadku kliknięcia na opcję "Edytuj" - dodaniem nowej wersji tekstu. JavaScript pracujący po stronie przeglądarki użytkownika ułatwia edycję tekstu, obsługując przyciski automatycznie wstawiające niektóre rodzaje formatowania oraz znaki narodowe. Jest to dobry przykład współpracowania tych dwóch języków w naprawdę dynamicznej aplikacji internetowej.

Historia projektu

Pierwsza wersja PHP, rozpowszechniana pod nazwą PHP/FI (*Personal Home Page/Forms Interpreter*), została

stworzona przez Rasmusa Lerdorfa w roku 1994 jako zestaw skryptów Perla służący do monitorowania internautów odwiedzających jego witrynę. Gdy ruch stał się zbyt duży, przepisał je w języku C, dodając przy tym nowe opcje. Niedługo później ludzie zaczęli prosić go o możliwość użycia tych narzędzi na swoich stronach, zatem 8 czerwca 1995 roku autor udostępnił publicznie kod źródłowy (PHP Tools 1.0). Już kilka miesięcy później projekt przekształcił się w zalążek znanego obecnie języka programowania, gdy został połączony z innym narzędziem Rasmusa Lerdorfa - *Form Interpreter*, które dało drugi człon nazwy. W 1997 roku pojawiło się PHP/FI 2.0, posiadające wtedy kilka tysięcy aktywnych użytkowników na całym świecie oraz obsługujące 50 tys. domen. Co ciekawe, wersja ta spędziła większość "życia" na beta testach. Oficjalne wydanie było tylko jedno i ukazało się w listopadzie 1997 roku.

W 1997 roku projektem zainteresowali się dwaj izraelscy programiści: Zeev Suraski i Andi Gutmans. Odkryli oni, że PHP/FI ma zbyt małe możliwości na potrzeby aplikacji eCommerce, którą tworzyli na uniwersytecie. Zdecydowali wtedy, że przepiszą kod PHP całkowicie od nowa, korzystając z pomocy już istniejącej społeczności PHP. W czerwcu 1998 roku ogłosili PHP 3.0 jako następcę PHP/FI, którego dalszy rozwój został wtedy zatrzymany. Był to wielki krok naprzód. PHP 3.0 posiadało całkowicie nową architekturę, która znacznie zwiększała wydajność. Pojawiły się w niej zalążki programowania obiektowego, ale najważniejszą cechą aplikacji była jej modułowość. Użytkownicy mogli rozszerzać teraz funkcjonalność języka poprzez dodawanie nowych modułów.

Krótko po wydaniu PHP 3, w zimie 1998 Zeev Suraski oraz Andi Gutmans jeszcze raz zabrali się za przepisywanie kodu źródłowego PHP, korzystając z doświadczeń nabytych przy pracach nad poprzednią wersją. Za główne cele obrali poprawienie modułowości oraz wydajności złożonych aplikacji. Choć dotychczasowa wersja potrafiła sobie z nimi poradzić, nie była jednak stworzona do tego celu i przegrywała przez to z innymi rozwiązaniami.

W połowie roku 1999 ukazał się oficjalnie Zend Engine, nowy silnik języka skryptowego, wokół którego niedługo później zaczęto budować PHP 4. Jego nazwa to kompromisowe połączenie imion twórców projektu. Nowa, oparta o niego wersja PHP, ukazała się w maju 2000 roku. Tak jak poprzednio, był to potężny krok naprzód. Programiści mieli do dyspozycji teraz wiele nowych narzędzi, konstrukcji językowych oraz bezpieczniejszy system wejścia/wyjścia. Od strony administracyjnej pojawiło się oficjalne wsparcie dla wielu nowych serwerów. Przez cztery lata od chwili wydania ukazały się trzy kolejne edycje tej wersji oznaczone numerami: 4.1, 4.2 oraz 4.3. W każdej z nich odczuwalne było zwiększenie bezpieczeństwa, szybkości działania oraz możliwości. W 2004 roku obsługiwały one łącznie 20% wszystkich domen sieciowych. Również obecnie, dwa i pół roku po premierze PHP 5, "czwórka" jest bardzo chętnie wykorzystywana przez administratorów ze względu na dużą stabilność.

W 2002 roku Zeev Suraski oraz Andi Gutmans ponownie rozpoczęli znaczącą modernizację silnika PHP mającą na celu dodanie do tego języka modelu obiektowego z prawdziwego zdarzenia. W lutym 2003 ukazała się pierwsza wersja alpha nowej wersji PHP oznaczonej numerem 5.0.0. Stabilna wersja ukazała się prawie półtora roku później, w lipcu 2004 roku. Nowości sprawiły, że PHP może konkurować teraz z innymi rozwiązaniami server-side, jak równy z równym. Pojawił się całkowicie nowy model programowania obiektowego, przez co niestety została utracona część kompatybilności z poprzednimi wersjami PHP w niektórych skryptach. Jest to spowodowane zmianą sposobu reprezentacji obiektów. Przebudowano także wiele modułów, w tym do obsługi XML'a i komunikacji z bazą danych, czyniąc je bardziej przyjaznymi dla programistów.

W połowie roku 2005 zaczęły pojawiać się oficjalne sygnały, że rozpoczęto wstępne prace nad PHP 6. Obecnie publicznie dostępne są codzienne snapshoty rozwojowego repozytorium kodu źródłowego, które można ściągnąć i przetestować. Głównym celem jest dalsze dążenie do ujednolicenia projektu, wprowadzenia dalszych możliwości wymaganych przez złożone projekty (m.in. pełne wsparcie unicode czy system cache'owania kodu). Usuwane są też kolejne archaiczne rozwiązania pochodzące jeszcze z czasów PHP/FI oraz PHP3, co w przypadku najstarszych skryptów ponownie spowoduje problemy z kompatybilnością.

Możliwości

Budowa PHP

PHP ma budowę modułową. Jądro projektu zapewnia obsługę wszystkich elementów języka oraz dostęp do podstawowego zestawu funkcji. Aby dodać więcej możliwości, instaluje się dodatkowe moduły. Część z nich ma charakter oficjalny i jest dołączona do każdej dystrybucji, pozostałe zgrupowane są w tzw. repozytorium PECL i należy je samodzielnie pobrać.

PHP, dzięki otwartości kodu źródłowego, pracuje bez problemu na niemal każdym istniejącym obecnie 32-bitowym systemie operacyjnym (z Unixem oraz Windowsem na czele) oraz pozwala na łatwą integrację z większością

serwerów WWW, np. Apache, IIS, OmniHTTPD.

Co PHP może zrobić?

PHP 5.1 oferuje swoim programistom wiele możliwości:

1. Komunikacja z wieloma popularnymi bazami danych poprzez jednolity interfejs
2. Obsługa wielu popularnych protokołów sieciowych, m.in. SSL, IMAP, SMTP, IRC.
3. Profesjonalne wsparcie standardu XML
4. Tworzenie obrazków w wielu popularnych formatach graficznych
5. Wiele funkcji obróbki tekstu
6. Wyrażenia regularne
7. Bardzo elastyczne tablice o mieszanych kluczach
8. Wsparcie dla usług sieciowych (SOAP, XML-RPC)
9. Zaawansowany model programowania obiektowego
10. Możliwość zintegrowania z platformą .NET
11. Obsługa obiektów COM w Windows
12. Funkcje kryptograficzne
13. Funkcje konwersji kalendarza
14. Funkcje konwersji kodowań
15. Algorytmy kompresji: GZip oraz BZip2.

PHP jest nie tylko językiem internetowym. Można go także używać do pisania aplikacji na konsolę systemu operacyjnego, a nawet programów okienkowych z graficznym interfejsem użytkownika. Dlatego stanowi atrakcyjną alternatywę dla języków takich, jak Perl. Instalator napisany w PHP, uruchamiany z konsoli systemowej, posiada m.in. repozytorium PEAR.

Wady PHP

Początkowe lata rozwoju miały jednak w sobie coś z pospolitego ruszenia. Z tamtego okresu do dziś przetrwało trochę niekonsekwencji oraz nieścisłości, które sprawiają pewne kłopoty programistom. Twórcy powoli je usuwają, ale potrwa to jeszcze wiele lat.

1. Ujednolicenie nazewnictwa - większość starszych modułów korzysta z różnych standardów nazywania poszczególnych funkcji, co utrudnia ich zapamiętywanie.
2. Kilka niepotrzebnych rozwiązań utrudniających życie programistom: *magic quotes* oraz *register globals*

Konkurencja ma jeszcze jedną przewagę nad PHP. Interpretery języków takich, jak ASP, kompilują skrypty do bardziej czytelnej dla nich postaci, oszczędzając wiele czasu. PHP za każdym uruchomieniem wykonuje całą pracę od zera i choć jego interpreter jest niezwykle wydajny, z powodu nadkładania sobie pracy szybko traci zapas mocy. Możliwość kompilacji skryptów dostępna jest wyłącznie jako rozszerzenie.

Twórcy PHP świadomi są tych ograniczeń i starają się nie ignorować programistów. Problemem jest zachowanie wstecznej kompatybilności z olbrzymią bazą już stworzonego kodu PHP, który musi na nowych wersjach działać, najlepiej bez przeróbek. Zmiany są wprowadzane stopniowo i dlatego każdy, kto chciałby związać swoją karierę programisty WWW z językiem PHP, powinien śledzić aktualności pojawiające się na stronach www.zend.com oraz www.php.net.

Popularność PHP

PHP jest mimo swoich wad niezwykle popularny. Wykonane w nim są dosłownie wszystkie witryny: od stron domowych przez gry internetowe aż do komercyjnych portali oraz witryn wielkich korporacji. Polska społeczność PHP jest bardzo aktywna i napisała do tego języka wiele poradników co w połączeniu z łatwym dostępem do książek, daje naprawdę atrakcyjny produkt, jeśli chodzi o wsparcie. Nietrudno jest także zlokalizować dobry hosting, który za niewielką opłatą udostępni na serwerze konto z PHP oraz bazą danych.

Jak się uczyć?

Nauka PHP to długi i ciągły proces. Nie należy się spodziewać, że podręcznik ten nauczy kogokolwiek wszystkiego, ponieważ nie jest to jego założenie. Materiały pomocnicze pełnią funkcję wprowadzenia, zainteresowania i pokazania tematu, a cała reszta zależy już od konkretnego człowieka. Pamiętaj, że bez praktyki dużo nie osiągniesz. Dyskutuj z innymi programistami, analizuj cudze skrypty, podpatruj rozwiązania lub też wymyślaj własne. Eksperymenty nie niosą ze sobą żadnych poważnych skutków ubocznych i powinny być praktykowane przez każdego.

Nie przejmuj się, że twoje pierwsze projekty będą miały w sobie nieco z chaotyczności. To normalny proces, niemniej nie można na tym poprzestać. Za drugim razem zadaj sobie pytanie: co poprzednio mogłem zrobić lepiej? Czego mnie ten projekt nauczył? W ten sposób stopniowo dojdziesz do wprawy.

Duże znaczenie ma również rozumienie samych komputerów i algorytmów. PHP jest normalnym językiem programowania i pewnych spraw nie da się ominąć. Teoria bardzo dobrze uzupełnia praktykę, dlatego dokształcaj się także w tym względzie. Pozwoli Ci to zrozumieć, dlaczego dany problem rozwiązywany jest tak, a nie inaczej lub dlaczego w ogóle się nim zajmujemy.

Cała wiedza o projekcie PHP zgromadzona jest w obszernej dokumentacji zawierającej m.in. opis wszystkich modułów. Należy nauczyć się sprawnego poruszania po niej i orientowania się, gdzie co leży. Temu zagadnieniu poświęcony został jeden z rozdziałów podręcznika. Do dokumentacji warto zaglądać regularnie, gdyż jest ona zawsze zgodna z najnowszą dostępną wersją PHP. Stamtąd najszybciej się dowiesz, czy nie pojawiły się nowe funkcje, parametry, możliwości wykorzystania istniejących elementów. Jeżeli zamierzasz wiązać się na dłużej ze środowiskiem PHP, adres www.php.net powinien nawet znaleźć się wśród najczęściej przeglądanych przez Ciebie witryn. Stanie w miejscu, kiedy projekt ten posuwa się naprzód, jest kiepską alternatywą i bez orientowania się w zachodzących zmianach szybko odkryjesz, że twoje skrypty z tajemniczych powodów nie chcą pracować na nowych wersjach, co przyprawia o złość klienta.

[edytuj] Teoria czy praktyka?

Wielu początkujących programistów zadaje sobie pytanie - czy skupić się na teorii, czy też raczej na praktyce? Odpowiedź jest prosta: nie można się skupiać wyłącznie na jednym zagadnieniu. Ważna jest bowiem wiedza jak działa ten język programowania, ale nie można nie wiedzieć jak go wykorzystać.

Przed przystąpieniem do pisania skryptu powinniśmy rozrysować sobie jego projekt na papierze oraz ustalić pewne konwencje, jakich będziemy się trzymali w kodzie. Istotne jest, aby planowane przedsięwzięcie umieć zrealizować w praktyce. Choć projekty często tworzymy, aby nauczyć się czegoś nowego, poprzeczka nie powinna być windowana zbyt wysoko, gdyż wtedy nigdy go nie ukończymy. O wiele lepsze jest stopniowe poznawanie coraz bardziej złożonych elementów.

W książce tej duży nacisk kładziony jest na ponowne wykorzystywanie raz napisanego kodu. Uważamy, że nie ma sensu raz po raz na nowo odkrywać koła i tworzyć wszystkiego od nowa. Dobry programista po pewnym czasie utworzy sobie zbiór bibliotek z najczęściej używanymi funkcjami, które będzie przenosić między kolejnymi projektami, oszczędzając sobie zbędnej pracy. Praktyki te także powinny być w naszym planie uwzględnione. Inne sprawy, na które powinniśmy zwrócić uwagę, to:

- Nazewnictwo - czy stosujemy nazwy polskie, czy angielskie. W jakim stylu je zapisujemy? *nazwa_nazwa* czy *nazwaNazwa*?
- Jednolity styl kodowania
- Jakie biblioteki zewnętrzne wykorzystamy? W jakim stopniu?
- Ogólna budowa całej aplikacji:
    - Jaką drogę pokonują dane podczas tworzenia strony?
    - W jaki sposób dane są przetwarzane?
    - Jak są dane wyświetlane?
    - Jak połączone są ze sobą poszczególne elementy aplikacji?
    - Czy przewidujemy w przyszłości dalszą rozbudowę projektu? Jeśli tak, jakie kroki podejmiemy, aby ją maksymalnie ułatwić?

Projekty aplikacji można sporządzać również na komputerze. Pomocne będą tu wszelkie edytory tekstu, generatory diagramów itd. Po głębszym poznaniu języka PHP można zaznajomić się również z diagramami UML wykorzystywanymi w zawodowych zespołach programistycznych.

Fora dyskusyjne

Niestety, nie każdy problem będziemy w stanie rozwiązać samodzielnie. Wtedy z pomocą przyjdą nam fora dyskusyjne, których dużo jest w polskim Internecie. Pamiętaj, że często **aby uzyskać odpowiedź, musisz zastosować się do reguł forum**. Wielu administratorów nie lubi użytkowników, którzy umieszczają tematy nie tam, gdzie trzeba oraz nie stosują się do widocznych wszędzie informacji. Zanim zadasz jakieś pytanie, poświęć przynajmniej kilka minut na zapoznanie się z opisami poszczególnych forów, wstawkami zatytułowanymi "Zanim napiszesz posta" itd. Ich przeczytanie to chwilka, ale zapoznanie się z nimi może oszczędzić ci przykrości przy kontakcie z moderatorem. Zamieszczamy ten krótki poradnik, gdyż mnóstwo początkujących programistów nie zwraca na te niby oczywiste rzeczy uwagi, co skutkuje długim oczekiwaniem na pomoc lub konfliktem z moderatorem na samym wstępie.

Zanim napiszesz posta, upewnij się, że:

- Umieszczasz go na odpowiednim forum. Niektóre serwisy posiadają podział na forum dla spraw podstawowych i zaawansowanych. Zdecydowanie zalecamy pisanie na tym pierwszym. Wciskanie się na siłę na forum dla zaawansowanych programistów z tematami "Jak połączyć się z bazą XXX" niemal na pewno zostaną źle odebrane przez internautów.
- O niektóre sprawy w ogóle nie powinno się pytać, np. "czy można zrobić...", "czy jest jakiś skrypt", "gdzie mogę znaleźć XXX". Prawie na pewno dostaniesz link do wyszukiwarki Google ze złośliwą adnotacją. I to w zasadzie jest prawda. Z pomocą wyszukiwarki często znajdziesz odpowiedź znacznie szybciej, niż gdybyś miał oczekiwać na odpowiedź na forum. Wielu programistów traktuje nawet forum - miejsce rozwiązania problemu jako ostateczność, kiedy inne źródła zawiodły.
- Problem nie jest wyjaśniony w jakimś artykule w serwisie. Wiele serwisów posiada bazę artykułów i porad, które wyjaśniają niektóre sprawy. Warto się z nimi zapoznać przed przystąpieniem do pisania.

Aby szybko uzyskać odpowiedź, upewnij się, że:

- Podałeś wersje oprogramowania
- Podałeś fragment kodu źródłowego powodujący błąd
- W załączonym kodzie nie ma jakichś ważnych danych osobowych, haseł itd.
- Wyjaśniłeś dokładnie, co jest z nim nie tak. Nie używaj nigdy pojedynczego zwrotu *nie działa*, gdyż jest on wieloznaczny i prawie zawsze zostaniesz zapytany o więcej szczegółów
- Twój post jest poprawnie napisany pod względem ortograficznym. Zła interpunkcja lub robienie błędów w podstawowych wyrazach jest oznaką ignorancji, a tłumaczenia, że to nie jest lekcja polskiego, mogą budzić niesmak. Jeżeli nie jesteś pewny, pisz posty w edytorze tekstu ze sprawdzaniem pisowni.

Oczywiście istotna jest także sprawa tytułu tematu. Najlepsze są te streszczające istotę problemu, np. "Problem z połączeniem z bazą danych". Nazwa "Mam problem, pomocy!" nie jest już dobra, gdyż nie mówi, co jest w środku. W efekcie temat może zostać niezauważony przez osobę znającą rozwiązanie. Zwróć też uwagę, czy forum nie wymaga dodania jakiegoś krótkiego prefiksu, np. "[php]" w celu łatwiejszego skatalogowania tematu.

Na koniec pamiętaj - nawet jeżeli zostaniesz przez moderatora publicznie upomniany, nie jest to jeszcze powód do rozpaczy. Zadaniem moderatora jest utrzymanie porządku i nie oznacza to, iż żywi on do Ciebie przez to jakąś osobistą urazę. Po prostu przeproś i w przyszłości staraj się unikać podobnego błędu.