Flash PHP. Podstawy

Autor: Steve Webster
Tłumaczenie: Marek Korbecki
ISBN: 83-7197-658-5
Tytuł oryginału: Foundation PHP for Flash
Przykłady na ftp: 995 kB
Liczba stron: 392

Możliwości PHP i Flasha wspaniale się uzupełniają. Oczywiście, Flash sam w sobie jest wspaniałym narzędziem, ale tworzenie zmieniających się, w pełni interaktywnych witryn Flasha wymaga zastosowania dodatkowych technik. PHP jest bezpłatnym i łatwym sposobem osiągnięcia tego celu. Lista jego funkcji jest długa, więc może on w znaczący sposób zwiększyć dynamikę witryny Flasha.

Niniejsza książka ma zaznajomić Czytelnika ze wspaniałymi możliwościami, jakie daje wykorzystywanie skryptów wykonywanych po stronie serwera podczas tworzenia witryn Flasha. Napisaliśmy ją, aby w odpowiednim tempie przeprowadzić go przez pierwsze próby tworzenia skryptów PHP.

Książka niniejsza skierowana jest do projektantów witryn internetowych, a w szczególności tych, którzy w swej pracy korzystają z Flasha. Naszym celem jest również zapoznanie Czytelników z językiem PHP w przyjazny sposób, szczególną uwagę koncentrując na praktyczne zastosowanie przedstawionych wiadomości. Mając na uwadze sposób omawiania przykładów w tej książce założyliśmy, że każdy Czytelnik dysponuje podstawową wiedzą na temat Flasha, a zastosowanie PHP ma pozwolić mu na zwiększenie możliwości tworzonych witryn. Pomimo tego w niniejszej książce uwzględnimy pełne wyjaśnienia odnoszące się do tego programu. Wyjaśnienia te pojawiać się będą w całej książce.

W każdym rozdziale omawiać będziemy:

- podstawy określonego aspektu tworzenia skryptów PHP;
- przykład zastosowania danego skryptu w aplikacji Flasha. Podane przykłady są rozbudowane, mogą być stosowane bezpośrednio lub adaptowane do innych potrzeb.

Naszym celem nie było podawanie ogromnych ilości teorii i następnie pozostawienie Czytelnika samemu sobie. Dążyliśmy do praktycznego pokazania, w jaki sposób PHP wspomaga projektantów wykorzystujących Flasha.

W niniejszej książce dokładnie przedstawimy założenia PHP i, co ważniejsze, sposoby jego wykorzystywania podczas tworzenia coraz to bardziej złożonych i interesujących aplikacji sieciowych. W każdym rozdziale przedstawimy przykład prostego kodu, szczególnie zwracając uwagę na te elementy, które mogą okazać się przydatne podczas projektowania witryn. Zaprezentujemy zestaw 12 interesujących, atrakcyjnych aplikacji — od filmów rejestracji i logowania aż po pełne forum stworzone we Flashu.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 12.
Przykład 3. — forum

W niniejszym rozdziale pokażemy, jak utworzyć:

- główny plan złożonej aplikacji;
- pełny interfejs Flasha dla forum dyskusyjnego, umożliwiającego użytkownikom odczytywanie i publikowanie swoich wypowiedzi, odpowiadanie na istniejące wątki, a nawet rejestrowanie się;
- skrypty obsługujące wypowiedzi, wątki i proces rejestracji, przechowujące informacje w bazie danych MySQL i, w razie potrzeby, ich odczytywanie.

Podczas tworzenia tych wszystkich, wspaniałych aplikacji Czytelnik był prawdopodobnie zbyt zajęty, aby zauważyć, że nasza długa podróż poprzez PHP dobiega końca. Wspaniałym uwieńczeniem nauki PHP będzie utworzenie najbardziej użytecznej aplikacji — systemu obsługi tablicy ogłoszeniowej! Tak, oto nadszedł czas na obiecane **forum Flasha**.

Będzie to bardzo istotny moment procesu zaznajamiania się z różnymi technikami. Wykorzystamy tu bowiem kilka zarówno średnich, jak i zaawansowanych technik ActionScript po stronie Flasha. Będziemy jednak uwzględniać dokładne wyjaśnienia dotyczące poszczególnych technik i przyczyn ich zastosowania.

Zanim jednak zabierzemy się do pracy, trzeba omówić kilka koncepcji, które uwzględnimy tworząc omawianą aplikację.

- **Wypowiedź**

  Wypowiedź jest pojedynczą wiadomością umieszczaną na tablicy.

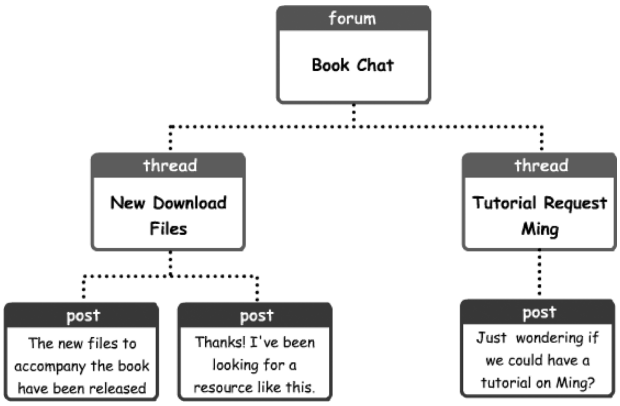
- **Wątek**

  Wątek jest zbiorem wypowiedzi dotyczących jednego tematu. Na przykład, jeśli pojawi się pytanie na temat średniej długości życia małp, ta wypowiedź oraz dotyczące jej odpowiedzi utworzą jeden wątek.

  Wykorzystanie koncepcji wątków umożliwia grupowanie wypowiedzi na jeden temat i przedstawianie ich jako całej dyskusji.

- **Forum**

  Forum jest zbiorem wątków. Niektóre tablice mogą zawierać wiele forów, z których każde dotyczy jednego tematu, grupującego wiele wątków. Tablica dyskusyjna, którą zbudujemy w tym rozdziale, będzie obsługiwała tylko jedno forum, a nasz przykład pozwoli na omówienie sposobu tworzenia obydwóch typów tablic dyskusyjnych. Dwa w cenie jednego — jaśniej powiedzieć już się nie da, nieprawdaż?



Po tych wstępnych rozważaniach przyjrzyjmy się podstawowym etapom budowy aplikacji. Warto zwrócić uwagę, że wszystkie elementy są omawiane bez rozważania sposobów implementacji, a zatem na tym etapie nie będziemy się zastanawiać, skąd będą pobierane dane, w jaki sposób i dokąd. Jest to bardzo dobra metoda projektowania aplikacji, gdyż pozwala na wykorzystanie innego ogólnego projektu i zaimplementowanie go za pomocą innych technik.

Poniżej przedstawiliśmy listę funkcji niezbędnych do prawidłowego działania aplikacji.

- odczytywanie listy wątków na forum;
- wyświetlanie zawartości forum;
- wyświetlanie listy wypowiedzi po wybraniu wątku przez użytkownika;
- wyświetlanie wątków;
- udostępnienie użytkownikowi przycisku umożliwiającego powrót do widoku forum.

Ponadto trzeba zapewnić możliwość rejestrowania nowych użytkowników, a także zakładania nowych wątków i odpowiadania na już istniejące.

Plan główny

Czas ponownie wziąć do rąk kartkę papieru i pióro i zastanowić się nad wyglądem interfejsu użytkownika. Trzeba wziąć pod uwagę wszystkie etapy pracy aplikacji (według powyższego opisu) i wszystkim poświęcić należytą uwagę.

Z analizy listy etapów, sporządzonej w poprzednim punkcie, wynika, że interfejs użytkownika musi się składać z pięciu głównych sekcji:

- widok forum;
- widok wątku;
- nowa wypowiedź;
- odpowiedź dla wypowiedzi;
- rejestracja.

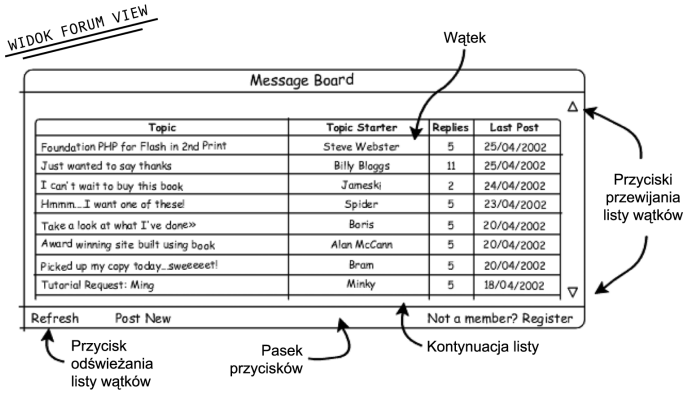
Omówimy teraz te sekcje po kolei.

Widok forum

Widok *Forum View* będzie prezentował listę wątków na danym forum. Oprócz wyświetlania wątków trzeba dać użytkownikowi możliwość odświeżania listy. W ten sposób uzyska on możliwość sprawdzania, czy pojawiły się nowe wątki. Należy także umożliwić użytkownikowi tworzenie nowych wątków, a także zapewnić wygodny dostęp do sekcji rejestracyjnej.

Oczywiście, łatwo się zorientować, co będzie tu potrzebne, nieprawdaż? Potrzebny będzie **pasek przycisków**! Zajmie on niewielką część u dołu interfejsu i będzie zawierał przyciski pozwalające sterować wszystkimi funkcjami aplikacji.

Spójrzmy na uproszczony schemat...



Zawartość okna odpowiada dokładnie naszym wymaganiom.

Znaczną część tej sekcji wypełnia lista wątków otwartych na forum. Być może wkrótce nasze forum stanie się na tyle popularne, że wątków na liście będzie więcej, niż można wyświetlić w jednym oknie. Trzeba zatem uzupełnić je przyciskami przewijania, umożliwiającymi użytkownikowi przesuwanie listy. Należy to zaliczyć do zadań Flasha, a zatem wrócimy do tego później.

Oczywiście, sposób prezentacji informacji zależy w zupełności od twórców aplikacji. Sądzimy jednakże, że najlepszym rozwiązaniem będzie wykorzystanie konwencjonalnego układu forum. Po co wymyślać koło na nowo, skoro można od razu zacząć je toczyć? Użyjemy więc takiego układu kolumn, jaki spotyka się na większości forum: temat dyskusji, założyciel wątku, liczba wypowiedzi oraz data wysłania ostatniej wiadomości.

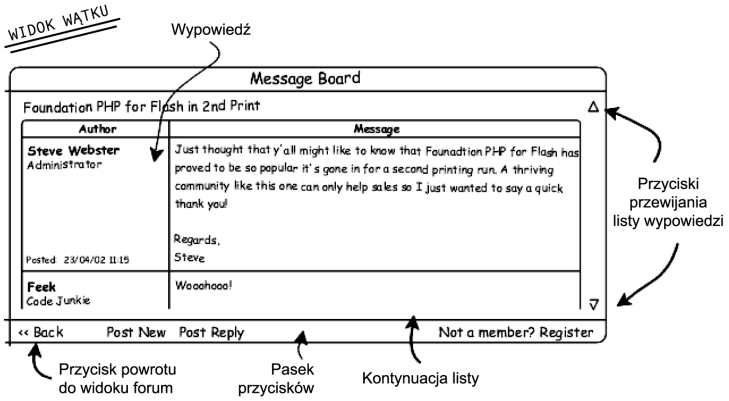
Przyjęcie takiej konwencji ułatwi wskazanie informacji, jakie trzeba będzie przechowywać w bazie danych. Na naszej tablicy zastosujemy jeszcze jedno, standardowe rozwiązanie: ostatnia przesłana wypowiedź będzie wyświetlana jako pierwsza. Działanie tej funkcji całkowicie zależy od sposobu przechowywania informacji w bazie danych — możemy zastosować dowolny układ — ale na pierwszym etapie lepiej będzie trzymać się konwencji. Żeby łamać zasady, trzeba je najpierw poznać!

Warto zwrócić uwagę przynajmniej na jeden element: pasek przycisków, z którego jesteśmy bardzo dumni. Będzie on zawierał różne przyciski, w zależności od uruchomionej sekcji aplikacji. Na przykład, jeśli użytkownik otworzy sekcję *Post Reply* (Wysyłanie odpowiedzi), udostępnianie mu przycisku widoku forum nie miałoby sensu, gdyż aby wysłać odpowiedź, należy najpierw wybrać wątek. Jest to logiczne, ale będzie stanowiło o większej złożoności gotowej aplikacji.

Widok wątku

Omówiliśmy *Widok forum*. Kolejnym widokiem jest *Thread View* — widok wątku. Jego zadaniem będzie wyświetlanie listy wypowiedzi należących do wątku wybranego przez użytkownika. Także w tym przypadku na pasku przycisków pojawią się przyciski udostępniające funkcje konieczne dla tej sekcji.

Uważamy, że widok wątku powinien mieć podobną budowę do widoku forum. Całkowita zmiana założeń projektu mogłaby wprowadzić zamęt. Oto wstępny szkic:



W pewnym sensie treść tej tabeli odpowiada widokowi poprzedniej tabeli. Uwzględniono tu jednak pojedyncze wypowiedzi dotyczące wybranego wątku. Lewa strona tabeli podaje informacje o autorach i datach wysłania poszczególnych wypowiedzi. Po prawej zaś stronie wyświetlana jest pełna treść każdej z nich.

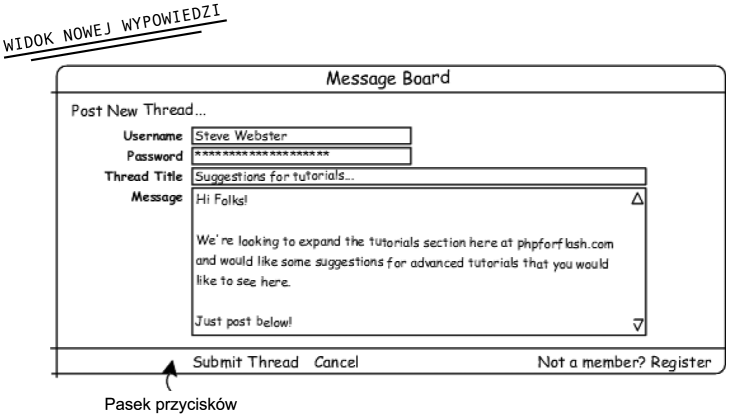
Warto nadmienić, że w omawianym przykładzie przyjęto konwencję wyświetlania wątków w porządku odwrotnie chronologicznym, ogólnie akceptowany jest również rzeczywisty porządek chronologiczny (zgodny z czasem pojawiania się kolejnych wypowiedzi). Wyświetlanie wypowiedzi w porządku zgodnym z kolejnością ich dołączania umożliwia użytkownikom prześledzenie przebiegu dyskusji.

Przycisk *Refresh* (Odśwież) został zamieniony na przycisk *Back* (Wstecz), który będzie tu służył do przywracania widoku forum. Umieścimy tu także jeden przycisk dodatkowy — *Post Reply* (Odpowiedz). Jeśli użytkownik naciśnie ten przycisk, będzie mógł udzielić odpowiedzi na bieżący wątek. Zwróćmy uwagę, że przycisk *Post New* jest dostępny przez cały czas, dzięki czemu użytkownik nie będzie musiał wracać do widoku forum, by otworzyć nowy wątek. To się nazywa ergonomia!

Otwieranie nowych wątków

Skoro już wiemy, jak naszą tablicę ogłoszeń będą widzieć jej użytkownicy, musimy zaprojektować interfejs, który pozwoli im na zakładanie nowych wątków.

W tym celu utworzymy we Flashu prosty formularz, wyposażony w użyteczne przyciski na pasku. Nie ma tu niczego szczególnego, co należałoby objaśniać w jakiś specjalny sposób. A zatem, przyjrzyjmy się naszemu projektowi.



Uwzględniliśmy tu wszystkie elementy formularza, potrzebne użytkownikowi do założenia nowego wątku na tablicy ogłoszeń. Formularz ten zawiera najważniejsze elementy wszystkich trzech głównych tabel.

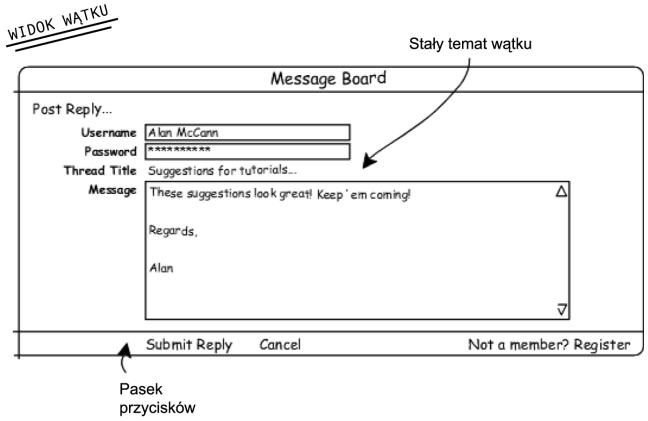
Przede wszystkim, umieścimy tu pola **nazwy użytkownika** i **hasła**, których zadaniem będzie potwierdzanie zarejestrowania użytkownika. Informacje te będą porównywane z do danych zapisanych w pliku, co umożliwi potwierdzenie uprawnień danej osoby do umieszczania wypowiedzi na tablicy.

Kolejne pola służą do wpisywania tematu wątku oraz treści wypowiedzi. Jest oczywiste, że oprócz zakładania wątku należy również utworzyć rozpoczynającą go wypowiedź.

Na powyższym szkicu widzimy również przycisk *Submit Thread* (Prześlij wątek) uruchamiający proces zamieszczania wątku, przycisk *Cancel* (Anuluj) na wypadek, gdyby użytkownik zmienił zdanie oraz przycisk *Register* (Rejestruj), którym może posłużyć się użytkownik dotychczas nie zarejestrowany.

Widok odpowiedzi

Interfejs sekcji *Post Reply* jest niemal dokładnie taki sam, jak widok sekcji *Post New*. Wynika to z konieczności dostarczenia w obydwóch przypadkach tych samych, podstawowych informacji.



Jedyną zauważalną różnicą jest to, że temat wątku nie jest wyświetlany w edytowalnym polu tekstowym. Jeśli użytkownik zdecyduje się odpowiedzieć na wątek, powinien się go trzymać! Jeżeli natomiast zechce otworzyć nowy, należy umożliwić mu przejście do odpowiedniego okna. Tytuł wątku trzeba oznaczyć w taki sposób, by każdy zapominalski użytkownik pamiętał, na jaki temat się wypowiada!

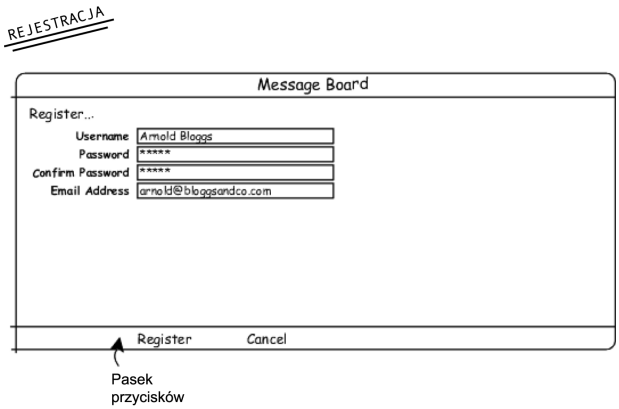
Warto także zwrócić uwagę na zamianę przycisku *Submit Thread* na przycisk *Submit Reply*, z czym musi się wiązać również zastosowanie odpowiedniej akcji, umożliwiającej wykonanie żądanej operacji.

Rejestracja

Sekcja *Register* naszej aplikacji będzie sekcją umożliwiającą użytkownikom rejestrowanie się na tablicy ogłoszeń. Absolutne minimum informacji, jakie musi obsługiwać, to:

- nazwa użytkownika;
- hasło;
- adres e-mail.

Można by posunąć się dalej i dodawać kolejne elementy, ale tymczasem pozostańmy przy tych podstawowych. A zatem, spójrzmy na poniższy szkic:



Jak widać, ta sekcja naszej aplikacji wygląda dość skromnie. Można by uzupełnić ją wieloma innymi elementami. Spośród nich szczególnie godnym polecenia pomysłem jest regulamin, który każdy nowy użytkownik musiałby zaakceptować, aby uzyskać dostęp do forum!

Projektowanie układu tabel

Można powiedzieć, że budowa interfejsu została gruntownie przemyślana. Na razie nie musimy martwić się tym, czy całość naszej koncepcji będzie funkcjonowała poprawnie i możemy przystąpić do kodowania. Trzeba określić rodzaje informacji, jakie będą przechowywane. Aby wykorzystać swoje umiejętności w posługiwaniu się bazą danych, w celu przechowywania danych wykorzystamy MySQL.

A więc, jakie informacje powinniśmy przechowywać? Oto ich lista:

Użytkownicy:

- nazwa użytkownika;
- hasło;
- tytuł;
- adres e-mail.

Wypowiedzi:

- autor wypowiedzi;
- treść wypowiedzi;
- data utworzenia wypowiedzi.

Wątki:

- temat wątku;
- użytkownik otwierający wątek;

- liczba odpowiedzi na pierwszą wypowiedź;
- data utworzenia ostatniej wypowiedzi w wątku.

Tym razem będziemy przechowywać znaczną ilość informacji, a zatem warto je rozdzielić i zachować w logicznym układzie tabel (*Users*, *Posts*, *Threads*). Podążając tym śladem, należy utworzyć następujące tabele:

Tabela: forumUsers

Nazwa kolumny	Typ danych	Opis
`userID`	Integer	Klucz główny tabeli. Możemy ją wykorzystać do odrębnego oznaczenia każdego użytkownika.
`username`	String	Nazwa użytkownika.
`password`	String	Hasło użytkownika.
`email`	String	Adres e-mail użytkownika.
`title`	String	Tytuł użytkownika, na przykład `Administrator`.

Tabela: forumThreads

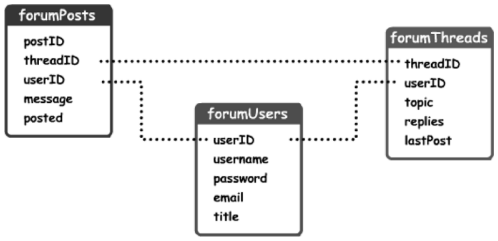
Nazwa kolumny	Typ danych	Opis
`threadID`	Integer	Klucz główny tabeli. Wykorzystamy ją do oznaczania poszczególnych wątków.
`userID`	Integer	Identyfikator użytkownika, który utworzył wątek.
`topic`	String	Temat wątku.
`replies`	Integer	Liczba odpowiedzi na początkową wypowiedź.
`lastPost`	Integer	Data opublikowania ostatniej wypowiedzi, zapisana w postaci uniksowego znacznika czasu.

Tabela: forumPosts

Nazwa kolumny	Typ danych	Opis
`postID`	Integer	Klucza główny tabeli. Wykorzystamy ją do oznaczania poszczególnych wypowiedzi.
`threadID`	Integer	Identyfikator wątku, do którego należy wypowiedź.
`userID`	Integer	Identyfikator użytkownika, który utworzył wypowiedź.
`message`	String	Zasadnicza treść wypowiedzi.
`posted`	Integer	Data utworzenia wypowiedzi, zapisana w postaci uniksowego znacznika czasu.

Przyglądając się tym tabelom można zauważyć, że są one między sobą w pewien sposób powiązane. Na przykład, w tabeli `forumPosts` wykorzystujemy klucz główny tabeli `forumThreads` (`threadID`) do identyfikacji wątku, do którego należy dana wypowiedź. W tej samej tabeli `userID` jest identyfikatorem autora wypowiedzi.

Wspomniane powiązania (relacje) można przedstawić za pomocą poniższego diagramu.



Film Flasha: kilka przemyśleń

Skoro wszystko już zaplanowaliśmy, czas przystąpić do tworzenia filmu Flasha, w którym odbywać się będzie całe przedstawienie!

Film Flasha, który teraz opracujemy na potrzeby forum dyskusyjnego, będzie nieco bardziej skomplikowany, niż tworzone dotychczas. Wynika to z natury całego przedsięwzięcia. To zaś oznacza, że każdy etap pracy opatrzymy szczegółowymi objaśnieniami, dzięki którym Czytelnik nie powinien napotkać większych trudności.

Nie należy poddawać się emocjom, a zatem musimy zachować dyscyplinę i przystąpić do przemyślenia sposobu wizualnej reprezentacji listy wątków na forum i listy wypowiedzi w wybranym wątku. Można to rozwiązać na wiele sposobów, ale najelegantszą metodą będzie zastosowanie funkcji `attachMovie`, która pojawiła się po raz pierwszy we Flashu 5.

Działanie tej funkcji będzie polegało na tworzeniu klonu klipu filmowego dla każdego wyświetlanego wątku lub wypowiedzi. Klon ten będzie przechowywany w bibliotece.

Składnia `attachMovie` wygląda następująco:

```
someMovieClip.attachMovie(idName, newname, depth);
```

Funkcja ta będzie dołączała klon klipu filmowego z biblioteki, opatrzonego identyfikatorem `idName`, do klipu `someMovieClip`. Nowy klon `idName` będzie nosił nazwę `newname`, zaś zadaniem argumentu `depth` jest wyznaczanie poziomu klipu.

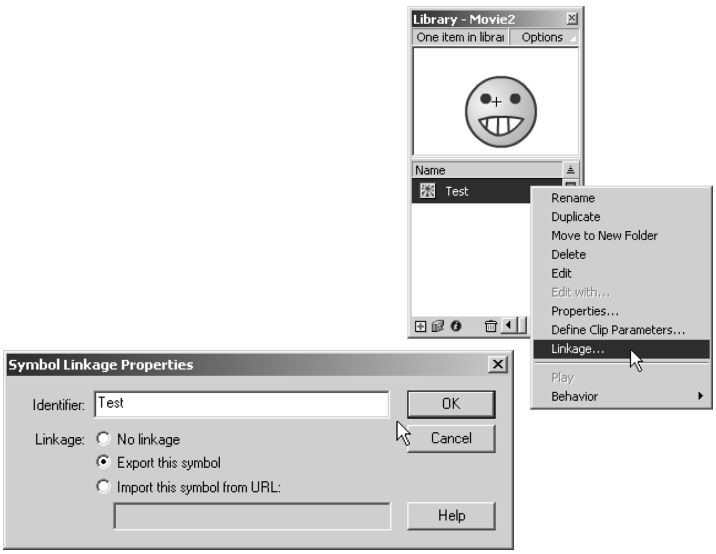
Właściwość `depth` jest bardzo użytecznym, wartym uwagi narzędziem Flasha. W razie utworzenia kilku klonów klipu filmowego na tym samym poziomie starszy z nich będzie zastępowany przez nowszy — klony nie mogą zajmować tego samego poziomu! Zapamiętanie tej zasady pomoże Czytelnikowi uniknąć problemów w przyszłości!

Gdy spotkaliśmy się z funkcją `attachMovie` po raz pierwszy, sądziliśmy, że jako argument `idName` wystarczy wskazać nazwę klipu filmowego (którą definiuje się w oknie dialogowym właściwości symbolu). Jakże się myliliśmy! Prawda jest taka, że trzeba tu

wejść w nowy, ekscytujący obszar Flasha, zwany **przyłączaniem symboli**. Jeśli Czytelnik pogubił się w powyższych przemyśleniach, niech pozwoli sobie wytłumaczyć pewne istotne sprawy.

Zazwyczaj jeśli klip filmowy znajduje się w bibliotece, ale nie jest wykorzystywany w pliku Flasha, wówczas nie zostaje również włączony do pliku SWF, powstającego podczas publikacji filmu. Jednakże, Flash 5 pozwala na wskazanie klipów, które należy wyeksportować *bez względu* na to, czy zostały wykorzystane w filmie, czy też nie. To właśnie tę technikę nazywa się *przyłączaniem symboli*.

Aby przyłączyć dany klip do filmu, należy wskazać jego **identyfikator**:



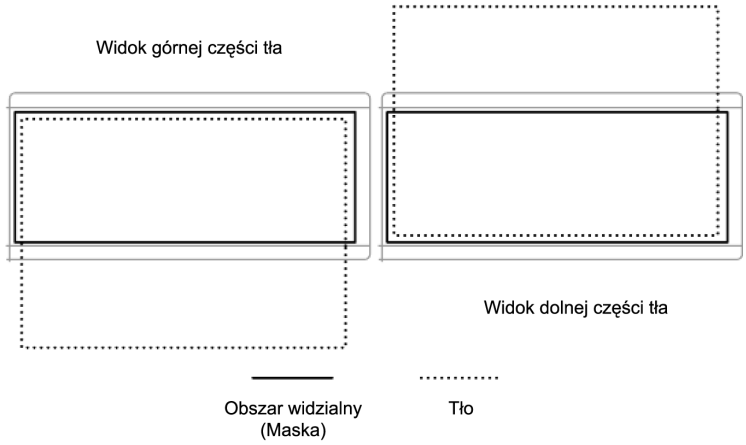
Poprzez ten identyfikator można odwoływać się, zapisując funkcję `attachMovie`. Czy teraz już Czytelnik rozumie powyższe rozważania?

Nie będziemy zajmować się teraz mechanizmem tworzenia list wątków i wypowiedzi — wrócimy do tego w odpowiedniej sekcji. Trzeba jednak mieć na uwadze fakt, że będziemy z tej techniki korzystać w naszym filmie, a zatem warto przygotować się do tego przed przystąpieniem do rzeczywistego działania.

Interesujące jest, że tworzenie list wątków i wypowiedzi w ten sposób otwiera kolejne zagadnienie — jeśli liczba wątków uniemożliwi jednorazowe ich wyświetlenie, to czy ta lista nie przykryje naszego starannie zaprojektowanego okna?

Odpowiedź brzmi tak, że mogłaby... ale tylko jeśli jej na to pozwolimy. W tej sytuacji wykorzystamy pusty klip filmowy (który nazwiemy *tłem* – *canvas*), na który nałożymy (za pomocą funkcji `attachMovie`) klip wątku bądź wypowiedzi, w zależności od rodzaju widoku. Następnie wystarczy jedynie zamaskować tło w taki sposób, by widoczna była tylko żądana część.

Spójrzmy na poniższy rysunek, przedstawiający tę ideę:



Z powyższego wynika, że jeśli wysokość tła jest większa od wysokości obszaru widzialnego (czyli maski), tło nie jest wyświetlane w całości. Aby przesunąć widok w dół w celu obejrzenia dolnej części tła, należy zmniejszyć wartość właściwości `_y` klipu filmowego, aż dolna krawędź tła znajdzie się w obszarze wyświetlania.

Jest to dość prosty proces, wymagający nie więcej niż dwóch linii kodu ActionScript.

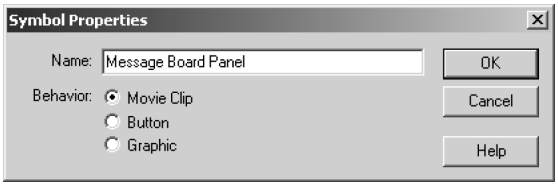
Budowa forum we Flashu

Jeśli już zakończyliśmy etap drapania się po głowie, możemy swoje palce skierować ku klawiaturze. Tak, nadszedł czas na opracowanie głównego pliku Flasha. Oczywiście, zawsze istnieje możliwość napotkania różnych problemów — ale przynajmniej możemy teraz przystąpić do pracy z większą pewnością!

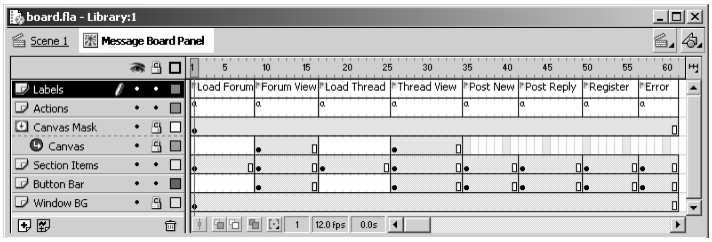
Załóżmy, że Czytelnik już dokładnie wie o co chodzi. A zatem:

1. W aplikacji tej wykorzystamy dobrze nam znany detektor `onClipEvent`, w związku z czym cały interfejs użytkownika znajdzie się w klipie filmowym.

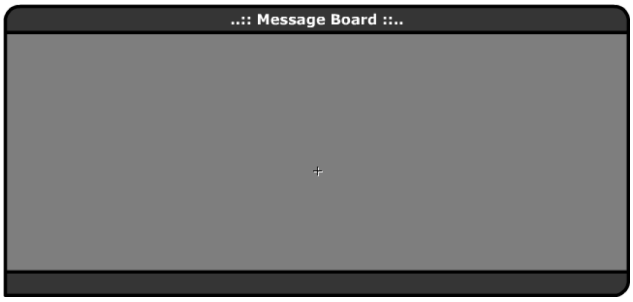
   Utwórz więc klip filmowy, nadaj mu odpowiednią nazwę, a następnie kliknij przycisk *OK*.



2. Teraz utwórz strukturę warstw i ujęć klipu filmowego. Choć jest ona dość rozbudowana, trzeba zachować możliwość identyfikowania poszczególnych, zaprojektowanych wcześniej, sekcji za pomocą etykiet ujęć.



3. Jak zwykle, na warstwie *Window BG* umieść tło aplikacji.



4. Na warstwie *Section Items* umieść naszą ulubioną animację zegara, dającą użytkownikowi sygnał o tym, że trwa ładowanie danych.

5. Ostatnim elementem sekcji *Load Forum* jest kod ActionScript na warstwie *Actions*.

   Kod ten będzie wywoływał skrypt PHP wczytujący listę wątków na forum. Ponadto zdefiniujemy funkcję, którą będziemy wykorzystywać w całym filmie.

```
// Tworzymy losową liczbę, dopisywaną do URL
randNum = Math.random()*1000000000;

// Wywołujemy skrypt PHP odczytujący informacje o forum
loadVariables ("viewforum.php?" add randNum, this);

// Wstrzymujemy odtwarzanie klipu filmowego do czasu załadowania danych
stop ();
```

   To będzie wszystko do momentu wczytania forum. Mówiąc najprościej, za pomocą powyższego kodu tworzymy losową liczbę, dopisywaną do adresu URL skryptu PHP. W ten sposób przeglądarka nie będzie wyświetlała kopii pliku zapisanej w pamięci podręcznej. Powody takiego postępowania opisano w rozdziale 1.

Losową liczbę należy dopisać na końcu części `url` w wywołaniu funkcji `loadVariables` odwołującej się do pliku *viewforum.php* (o którym pomówimy za chwilę). Następnie trzeba zatrzymać odtwarzanie klipu filmowego do chwili ukończenia wczytywania informacji. Do ponownego uruchomienia odtwarzania wykorzystamy detektor `onClipEvent`, umieszczony na końcu sekcji.

6. Kończąc opracowywanie tego ujęcia, zdefiniuj funkcję pobierającą pojedynczy argument (`threadID`) i wywołującą skrypt *viewthread.php*, przekazując jej identyfikator wątku.

```
//*************** [ FUNCTION HEADER ] ******************
// * viewThread()
// * Ładuje wątek z określonym identyfikatorem threadID
// ***********************************************************
function viewThread (threadID) {
    // Tworzymy losową liczbę, dopisywaną do URL
    randNum = Math.random()*1000000000;

    // Ładujemy wątek
    loadVariables ("viewthread.php?threadID=" add threadID add "&" add randNum,
    ➥ this);

    // Czekamy na załadowanie danych
    gotoAndStop ("Load Thread");
}
```

Należy zwrócić uwagę na ponowne zastosowanie sztuczki z zastosowaniem liczby losowej, co zapobiega wykorzystywaniu bufora przeglądarki — jest to jedna z tych rzeczy, dzięki którym życie programisty staje się prostsze!

7. Następnie należy spowodować wznowienie odtwarzania klipu i zatrzymanie w ujęciu *Load Thread*. Zatem jeśli klip filmowy podejmie pracę (po wczytaniu wszystkich danych), użytkownik przechodzi do widoku wątku *Thread View*.

8. Przyjrzyj się ujęciu *Forum View*. Tutaj po raz pierwszy umieszczamy pasek przycisków na warstwie *Button Bar*.

Kod ActionScript dla poszczególnych przycisków wygląda następująco:

Refresh

```
on (release) {
    gotoAndPlay("Load Forum");
}
```

Post New

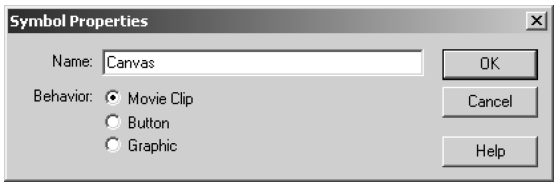
```
on (release) {
    gotoAndPlay("Post New");
}
```

Register

```
on (release) {
    gotoAndPlay("Register");
}
```

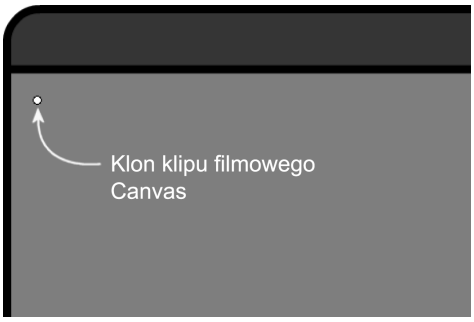
To całkiem proste — definicje w nawiasach dają pewność, że przejście nastąpi do właściwego ujęcia listwy czasowej.

9. Teraz trzeba utworzyć tło, na której będziemy umieszczać dołączane klipy filmowe. Jak mówiliśmy wcześniej, tło jest po prostu pustym klipem filmowym. Zatem utwórz je, naciskając przyciski *Ctrl+F8*. Nadaj nowemu klipowi nazwę *Canvas* i naciśnij przycisk *OK*.

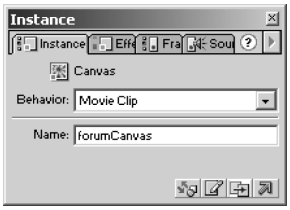


Po utworzeniu klipu wróć do listwy czasowej klipu filmowego *Message Board Panel*.

10. Upewnij się, że w ujęciu *Forum View* jest wybrana warstwa *Canvas*, a następnie przeciągnij nowo powstały klip *Canvas* z biblioteki do sceny. Klip ten jest pusty, zatem po przeciągnięciu będzie miał postać małego, białego kółka. Wybierz to kółko i umieść je w okolicach lewego górnego narożnika głównej sekcji klipu filmowego.



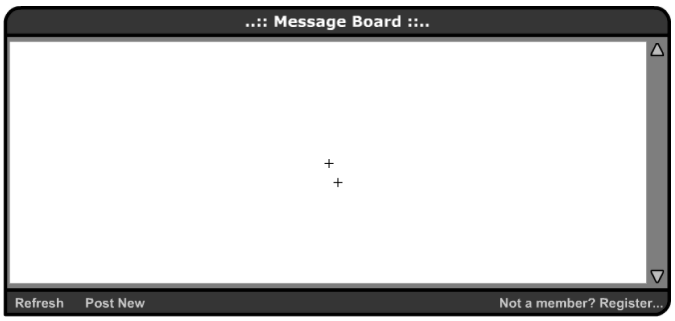
Upewnij się, że klon klipu *Canvas* jest wciąż wybrany, i nadaj mu nazwę, poprzez którą będzie można odwoływać się do niego z ActionScript.



Zaproponowaliśmy nazwę *forumCanvas*, gdyż potrzebne będą dwa klony klipu *Canvas*: jeden dla ujęcia widoku forum *Forum View*, drugi zaś dla ujęcia widoku wątku *Thread View*.

11. Skopiuj i wklej ten klon na warstwę *Canvas* w ujęciu *Thread View* i zmień nazwę kopii na *forumCanvas*.

12. Następnie należy utworzyć maskę na warstwie *Canvas Mask*. Pozwoli ona (mówiliśmy o tym wcześniej) na ukrycie tej części klipu *Canvas*, której nie chcemy wyświetlać. Upewnij się, że jest wybrana warstwa *Canvas Mask*, wybierz kolor kontrastujący (wybór ten jest całkowicie dowolny) z kolorem warstwy *Window BG*, po czym narysuj duży prostokąt, niemal w całości pokrywający główny obszar aplikacji.



13. Po wykonaniu czynności wymienionych w poprzednim punkcie, wybierz warstwę jako maskę. W tym celu kliknij prawym przyciskiem myszy nazwę warstwy (lub w razie korzystania z komputera Macintosh kliknij przytrzymując wciśnięty klawisz *Ctrl*) i wybierz opcję *Mask* (Maska).

Maska zniknie wraz z tłem, a ich warstwy zostaną zablokowane. Jeśli zajdzie potrzeba uaktywnienia warstw, użyj ich ikon.

14. Przed wykonaniem kolejnych czynności trzeba skonstruować klip filmowy, który będzie dołączany do *forumCanvas*. Posłuży on do wyświetlania listy wątków. Aby wykonać to zadanie, konieczne jest utworzenie trzech klipów filmowych o dokładnie takiej samej szerokości, z punktem środkowym umieszczonym w lewym górnym narożniku.

Nagłówek Forum

15. Najpierw wykonamy klip, który będzie stanowił nagłówek forum, czyli *Forum Header*. Jego jedyną funkcją jest wyświetlanie nazw kolumn w widoku *Forum View*, ale będzie również stanowił wizualne zwieńczenie listy wątków. Utwórz więc klip podobny do pokazanego poniżej i nadaj mu nazwę *Forum Header*.

Zwróć uwagę, że środkowy punkt klipu znajduje się w jego lewym górnym narożniku. Wiąże się to z metodą, którą zastosujemy podczas tworzenia widoku *Foum View*. A zatem upewnij się, że punkt ten znajduje się we właściwym miejscu.

Stopka widoku forum

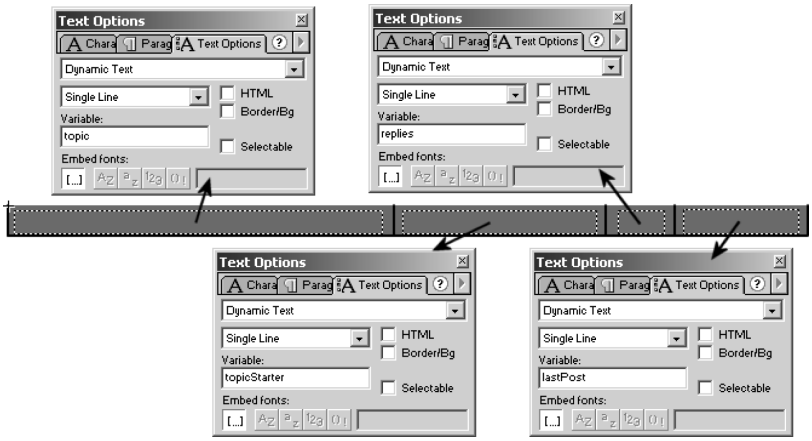
16. Klip ten będzie pełnił rolę atrakcyjnej wizualnie, dolnej krawędzi listy wątków. W ten sposób lista ta nie będzie ucięta na ostatniej pozycji. Utwórz zatem klip filmowy, taki jak zaprezentowany poniżej, i nadaj mu nazwę *Forum Footer*.

Forum Thread

17. Jest to główny klip, który wielokrotnie będziemy dołączać do tła, prezentując wszystkie wątki na forum. Będzie nam potrzebny niewidoczny przycisk, który utworzyliśmy w poprzednim przykładzie. A zatem można go stamtąd pobrać.

    Zacznij od utworzenia tła i linii reprezentujących kolumny. Najlepiej będzie skopiować całość z klipu *Forum Header*, dzięki czemu będą zachowane stałe szerokości kolumn.

18. Następnie utwórz pola tekstowe, których zadaniem będzie wyświetlanie informacji o wątkach:



19. Na koniec dodaj niewidoczny przycisk, przykrywając nim cały obszar wątku. Pozwoli to użytkownikom na otwieranie wybranych wątków poprzez ich kliknięcie.

20. Trzeba też przypisać przyciskowi następujący kod:

```
on (release) {
    _parent._parent.threadID = threadID;
    _parent._parent.topic = topic;
    _parent._parent.viewThread(threadID);
}
```

Sekcja `_parent._parent` jest konieczna ze względu na zagnieżdżenie tego klipu wewnątrz klipu filmowego *Canvas*, który z kolei zagnieździmy wewnątrz klipu *Message Board Panel*, do listwy czasowej którego chcemy się odwoływać.

A więc za pomocą powyższego kodu utworzymy dwie zmienne na listwie czasowej klipu filmowego *Message Board Panel*, po czym wywołamy funkcję `viewThread()`, którą utworzyliśmy wcześniej — całkiem pomysłowo, nieprawdaż?

Po utworzeniu wszystkich potrzebnych klipów filmowych trzeba zmienić ich właściwość *Symbol Linkage*, dzięki czemu zostaną one wyeksportowane wraz z klipem w trakcie publikacji.

21. Kliknij prawym przyciskiem myszy kolejno każdy z klipów filmowych: *Forum Header*, *Forum Footer* oraz *Forum Thread* i wybierz opcję *Linkage* z menu kontekstowego. Zaznacz przełącznik *Export this symbol* (Wyeksportuj ten symbol), a następnie wpisz identyfikatory:

    - Forum Header;
    - Forum Footer;
    - Forum Thread.

22. Można teraz powrócić na warstwę *Section Items* klipu *Forum View* i uzupełnić ją przyciskami przewijania. Do przycisków tych należy przypisać następujący kod:

    Przewijanie do góry

```
on (release) {
    if (forumCanvas._y < _140) {
        forumCanvas._y += 20;
    }
}
```

    Przewijanie w dół

```
on (release) {
    if (forumCanvas._y + forumCanvas._height > 110) {
        forumCanvas._y -= 20;
    }
}
```

    Pozwoli to na przewijanie tła i przeglądanie wątków niemieszczących się w oknie.

23. Dochodzimy teraz do najistotniejszego fragmentu kodu ActionScript na warstwie *Actions*, w ujęciu *Forum View*. To właściwie ten kod będzie tworzył widok *Forum View*, na podstawie zmiennych przekazywanych przez skrypt PHP.

Najpierw ukryj tło, aby użytkownik nie widział budowanej listy wątków:

```
// Ukrywamy tło forum
forumCanvas._visible = false;
```

24. Następnie, za pomocą `attachMovie`, dołącz klon klipu *Forum Header* do klipu *forumCanvas*.

```
// Dołączamy nagłówek klipu do tła i ustalamy tytuł
forumCanvas.attachMovie("Forum Header", "header", 255);
```

25. Punkty środkowe klonów dołączanych za pomocą funkcji `attachMovie` są zawsze ustawiane w pozycji (`0,0`) klipu filmowego. Należy sprawdzić wysokość tła, gdyż umieszczając kolejny klip będziemy musieli znać wartość `_y`. Jeśli odejmiemy jeden od wysokości nagłówka forum, będziemy mieli pewność, że po dołączeniu następnego klonu nie będą pojawiały się przerwy.

```
// Ustawiamy zmienną śledzącą punkt umieszczenia następnego
// klipu filmowego na tle
nextY = forumCanvas.header._height -1;
```

26. Następnie odczytujemy za pomocą pętli kolejne wątki zwracane przez skrypt PHP, dołączając klon klipu filmowego *Forum Thread* i odpowiednio ustawiając pozycję (`x,y`).

```
// Dla każdego wątku zwróconego przez skrypt PHP...
for (count=0; count<threadCount; count++) {
    // Dołączamy klip filmowy wątku do tła
    forumCanvas.attachMovie("Forum Thread", "thread" add count, count);

    // Ustalamy współrzędne X i Y dla klipu filmowego wątku
    forumCanvas["thread" add count]._x = 0;
    forumCanvas["thread" add count]._y = nextY;
```

27. Następnie pobieramy wszystkie elementy wątku i kopiujemy je do utworzonego przed chwilą klonu `Forum Thread`. W ten sposób wypełnimy pola tekstowe i ustawimy kilka niewidocznych zmiennych, na przykład `threadID`.

```
// Ustalamy szczegóły wątku
forumCanvas["thread" add count].threadID = this["thread" add count add "ID"];
forumCanvas["thread" add count].topic = this["thread" add count add "Topic"];
forumCanvas["thread" add count].topicStarter = this["thread" add count add
➥ "topicStarter"];
forumCanvas["thread" add count].replies = this["thread" add count add "Replies"];
forumCanvas["thread" add count].lastPost = this["thread" add count add "LastPost"];
```

28. Na koniec należy uaktualnić zmienną `nextY`, dodając do jej wartości wysokość nowo dołączonego klonu klipu filmowego i wyznaczając w ten sposób pozycję kolejnego klonu.

```
// Kolejny klip na tle umieszczamy tuż poniżej obecnego
    nextY += forumCanvas["thread" add count]._height -1);
}
```

29. Po przetworzeniu wszystkich wątków należy umieścić klon klipu *Forum Footer* na samym dole listy wątków.

```
// Dołączamy do tła stopkę wraz z jej pozycją
forumCanvas.attachMovie("Forum Footer", "footer", count);
forumCanvas.footer._x = nextX;
forumCanvas.footer._y = nextY;
```

30. Teraz można wyświetlić tło i zatrzymać w tym miejscu klip filmowy.

```
// Wyświetlamy tło wątku
forumCanvas._visible = true;

// Wstrzymujemy klip filmowy
stop ();
```

31. Kolejne ujęcie, *Load Thread*, jest niemal takie samo, jak ujęcie *Load Forum*. Należy jedynie uwzględnić niewielką różnicę w zapisie kodu ActionScript na warstwie *Actions*:

```
// Wstrzymujemy klip filmowy
stop();
```

Ponieważ funkcja ładująca wątek została zapisana wcześniej, teraz wystarczy zatrzymać wykonywanie klipu.

32. Ujęcie widoku wątku *Thread View* jest bardzo podobne do ujęcia widoku *Forum View*, a klip filmowy *threadCanvas* znajduje się już na warstwie *Canvas*.

Przed wpisaniem kodu ActionScript trzeba dodać jeden przycisk. Z wcześniej opracowanych szkiców wynika, że w ujęciu tym należy uwzględnić przycisk odpowiedzi *Post Reply*. Do ujęcia należy zatem przypisać taki oto kod ActionScript:

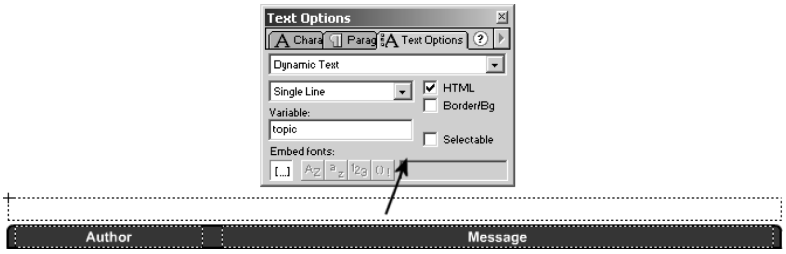
```
on (release) {
    gotoAndPlay ("Post Reply");
}
```

Przycisk ten powoduje jedynie przejście do odpowiedniej sekcji klipu filmowego.

33. Następnie utwórz klipy filmowe, które będą dołączane do *threadCanvas* w celu uformowania listy wypowiedzi. Także i w tym przypadku potrzebne będą trzy klipy — wszystkie o tej samej szerokości i z punktem środkowym położonym w lewym górnym narożniku.

Nagłówek wątku

34. Pierwszym na liście klipem filmowym jest nagłówek wątku, *Thread Header*. Różni się on nieco w stosunku do klipu *Forum Header*, gdyż obejmuje jedynie dwie kolumny, a widoczne na nim pole tekstowe posłuży do wyświetlania tematu wątku. Również w tym przypadku klip będzie pełnił rolę ozdobnej, górnej krawędzi. Utwórz więc klip filmowy, podobny do pokazanego poniżej, i nadaj mu nazwę *Thread Header*.



Także tutaj należy zwrócić uwagę na umieszczenie punktu środkowego klipu w jego lewym górnym narożniku. Konieczność ta wynika z metody zastosowanej do budowy widoku *Thread View*. Należy więc upewnić się, że punkt środkowy został umiejscowiony w odpowiednim miejscu.

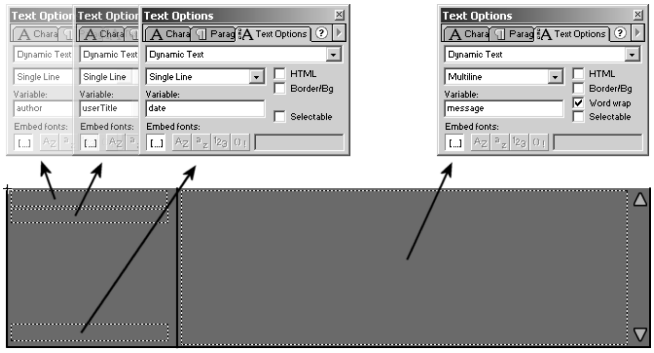
Stopka wątku

35. Także ten klip będzie pełnił funkcję estetyczną. Jego jedynym zadaniem będzie utworzenie ozdobnej, dolnej krawędzi, wyświetlanej tuż poniżej ostatniej wypowiedzi w wątku. Utwórz klip podobny do zaprezentowanego poniżej i nadaj mu nazwę *Thread Footer*.

Widok wypowiedzi

36. Jest to klip, na który będzie oddziaływała funkcja `attachMovie`, rozciągając go ponad całą powierzchnią tła. W ten sposób będą wyświetlane wszystkie wypowiedzi wybranego wątku. Najpierw utwórz tło oraz linie wyznaczające kolumny. Zalecamy w tym celu skopiowanie klipu filmowego *Thread Header*, co zapewni kolumnom jednakową szerokość.

37. Następnie umieść pola tekstowe, które będą wyświetlały informacje o wypowiedzi.



38. Na koniec dodaj przyciski przewijania, pozwalające użytkownikowi przewijać treść wiadomości.

39. Do przycisków przypisz następujący kod ActionScript:

    Przewijanie do góry

    ```
    on (release) {
        message.scroll++;
    }
    ```

    Przewijanie w dół

    ```
    on (release) {
        message.scroll--;
    }
    ```

    Po utworzeniu wszystkich klipów filmowych trzeba dokonać ustawień ich właściwości *Symbol Linkage*, co zapewni ich wyeksportowanie wraz z klipem nadrzędnym.

    Kliknij prawym przyciskiem myszy kolejno każdy z klipów: *Thread Header*, *Thread Footer* oraz *Forum Post* i wybierz opcję *Linkage*. Zaznacz także przełącznik *Export this symbol* i wprowadź następujące identyfikatory:

    - Thread Header;
    - Thread Footer;
    - Forum Post.

40. Teraz trzeba dopisać kod ActionScript na warstwie *Actions*. Jest on podobny do kodu wpisanego w poprzednim przypadku i został bogato opatrzony komentarzami. W razie ewentualnych wątpliwości wróć do sekcji *Forum View*.

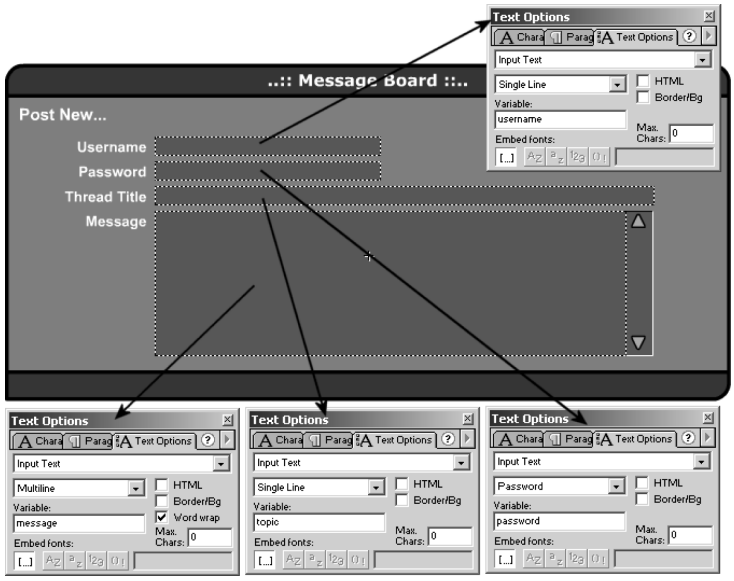
    ```
    // Ukrywamy tło wątku
    threadCanvas._visible = false;

    // Dołaczamy klip nagłówka do tła i ustalamy tytuł
    threadCanvas.attachMovie("Thread Header", "header", 255);
    threadCanvas.header.topic = topic;

    // Ustawiamy zmienną śledzącą punkt umieszczenia kolejnego
    // klipu filmowego na tle
    nextY = threadCanvas.header._height -1;

    // Dla każdej wypowiedzi w wątku...
    for (count=0; count < postCount; count++) {
        // Dołączamy klip filmowy wypowiedzi do tła
        threadCanvas.attachMovie("Thread Post", "post" add count, count);

        // Ustalamy współrzędne X i Y klipu wypowiedzi
        threadCanvas["Post" add count]._x = 0;
       threadCanvas["Post" add Count]._y = nextY;

        // Ustalamy szczegóły wypowiedzi
        threadCanvas["post" add Count].author = this["post" add count add "Author"];
        threadCanvas["post" add Count].userTitle = this["post" add count add
        ➥ "UserTitle"];
        threadCanvas["post" add Count].date = "Posted: " add this["post" add count add
        ➥ "Date"];
        threadCnavas["post" add Count].message = this["post" add count add "Message"];
    ```

```
    // Kolejny klip na tle umieszczamy tuż poniżej bieżącego
    nextY += threadCanvas["post" add count]._height -1;
}

// Dodajemy do tła stopkę i ustalamy pozycję
threadCanvas.attachMovie("Thread Footer", "footer", count);
threadCanvas.footer._x = 0;
threadCanvas.footer._Y = nextY;

// Wyświetlamy tło wątku
threadCanvas._visible = true;

// Wstrzymujemy klip filmowy
stop ();
```

Ostatni szlif forum we Flashu

Musimy jeszcze wykonać zaledwie kilka elementów! Pierwszym z nich jest sekcja tworzenia nowych wypowiedzi, *Post New*. Będzie to miejsce powstawania nowych wątków — a przynajmniej miejsce, w którym użyjemy PHP do tworzenia nowych wątków. Sekcja ta będzie stanowić coś więcej, niż prosty formularz Flasha, gdyż zostanie wyposażona w jeden lub dwa dodatkowe przyciski.

1. Utwórz pola tekstowe na warstwie *Section Items*. Poniższy rysunek może posłużyć jako wzorzec.



2. Dodaj przyciski przewijania pola treści wypowiedzi i dołącz do nich kod ActionScript:

   Przewijanie w górę

```
on (release) {
    message.scroll++;
}
```

Przewijanie w dół

```
on (release) {
    message.scroll--;
}
```

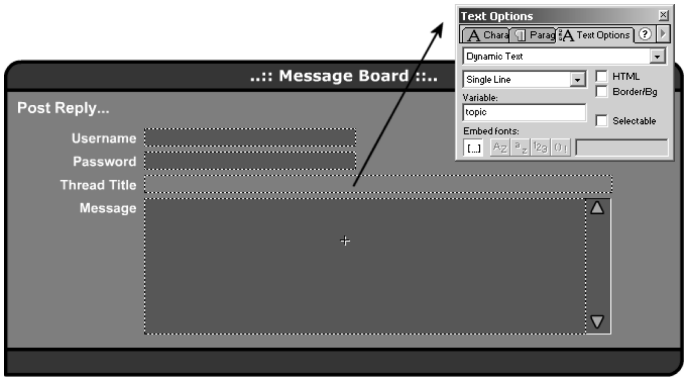
3. Utwórz także przyciski na warstwie *Button Bar*.

4. Nie zapomnij o zapisaniu ich kodu.

```
Submit Thread
on (release) {
    loadVariables("postnew.php", this, "POST");
    gotoAndPlay ("Load Forum");
}

Cancel
on (release) {
    gotoAndPlay ("Load Forum");
}
```

Skrypt dla tego ujęcia, na warstwie *Actions*, zawiera jedynie akcję `stop`.

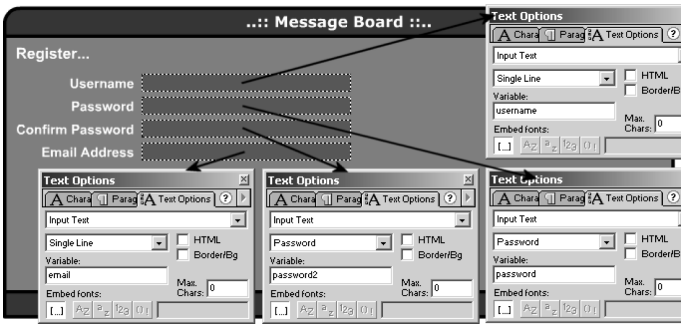
5. Ujęcie *Post Reply* jest niemal takie samo, jak w poprzednim przypadku. Różnica polega na tym, że pole tekstowe *Thread Title* nie jest edytowalne, a zamiast przycisku *Submit Thread* pojawia się przycisk *Submit Reply*.



Kod ActionScript dla przycisków wygląda następująco:

```
Submit Reply
on (release) {
    loadVariables("postreply.php", this, "POST");
    gotoAndPlay ("Load Thread");
}

Cancel
on (release) {
    gotoAndPlay ("Load Thread");
}
```

6. Po przejściu do ujęcia *Register* znów należy utworzyć prosty formularz, zaprezentowany na poniższym rysunku:



7. Zmianie ulega także zestaw przycisków na warstwie *Button Bar*: należy tu umieścić jedynie dwa przyciski, *Register* oraz *Cancel*.

Oto kod dla przycisku *Register*:

```
on (release) {
    if (username == "" || password == "" || password2 == "" || email == ""){
    errorMsg = "Passwords do not match";
    gotoAndPlay("Error");
} else {

        if (password != password2) {
            errorMsg = "Passwords do not match";
            gotoAndPlay("Error");
        } else {
            loadVariables("register.php", this, "POST");
            gotoAndPlay("Load Forum");
        }
    }
}
```

Zadaniem tego kodu jest kontrola wypełnienia wymaganych elementów formularza. Jeśli nie zostaną one wypełnione, kod wysyła komunikat o błędzie i powoduje przejście do ujęcia *Error*.

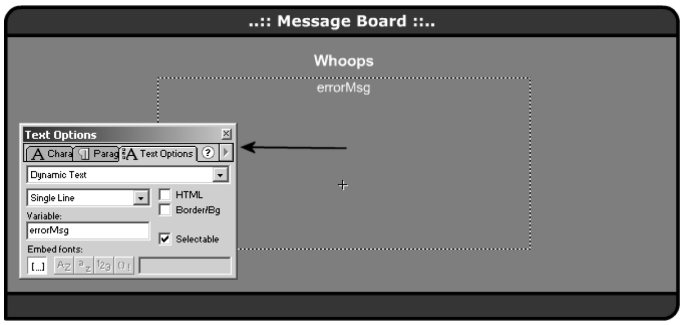
Jeżeli wymagane pola zostaną wypełnione, następuje sprawdzenie, czy obydwa hasła są takie same. Jest to mechanizm bezpieczeństwa, który pozwala upewnić się, że użytkownik nie dokonał omyłkowego wpisu. Ponieważ popełnienie takiego samego błędu dwukrotnie jest mało prawdopodobne, rozwiązanie to stanowi doskonały sposób wyłapywania błędów.

Jeśli jednak wszystko przebiegnie poprawnie, kod wywołuje skrypt *register.php* i powraca do ujęcia *Load Forum*.

Przycisk *Cancel* służy do bezpośredniego przechodzenia do widoku forum.

```
on (release) {
    gotoAndPlay("Load Forum");
}
```

8. Także w tym przypadku na warstwie *Actions* należy zapisać jedynie akcję `stop`.
9. Ostatnim ujęciem, które należy opracować, jest ujęcie *Error*. Zawierać ono będzie jedno pole tekstowe oraz jeden przycisk.



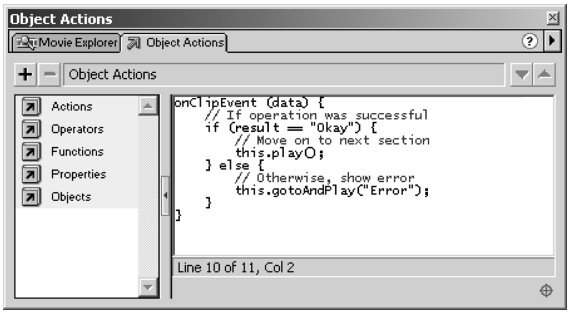
Jedynym istotnym elementem tego ujęcia jest kod ActionScript przypisany do przycisku *Back* i zapisany na warstwie *Button Bar*. Powoduje on bezpośrednie przejście do widoku forum — do tego miejsca, w którym pojawiły się problemy!

```
on (release) {
    gotoAndPlay("Load Forum");
}
```

Jeszcze ostatnia uwaga! Kod ActionScript na warstwie *Actions* po raz kolejny zawiera wyłącznie akcję `stop`.

Teraz trzeba przeciągnąć kopię klipu filmowego *Message Board Panel* z biblioteki do sceny głównej i przypisać do niego poniższy kod ActionScript, co zapewni obsługę wprowadzanych danych:

```
onClipEvent (data) {
    // Jeśli operacja przebiegła pomyślnie
    if (result == "Okay") {
        // Przechodzimy do następnej sekcji
        this.play();
    } else {
        // W przeciwnym razie wyświetlamy informację o błędzie
        this.gotoAndPlay("Error");
    }
}
```



Skrypty PHP

W tej sekcji utworzymy skrypty PHP, które będą działały za kulisami interfejsu stworzonego za pomocą Flasha.

Tym razem trzeba przygotować siedem skryptów, ale przed tym zadaniem powinniśmy przypomnieć sobie o dobrych zwyczajach i poświęcić chwilę czasu na omówienie zamierzonych celów.

Oto lista skryptów, które przygotujemy:

common.php

Skrypt ten będzie zawierał informacje konfiguracyjne oraz wspólne funkcje, wykorzystywane przez pozostałe skrypty.

setup.php

Przed rozpoczęciem opracowywania aplikacji trzeba się upewnić, że odpowiednia baza danych istnieje (a jeśli tak nie jest, to ją utworzyć), po czym zdefiniować wymagane przez aplikację tabele. Takie będzie zadanie tego skryptu.

viewforum.php

Ten skrypt posłuży przede wszystkim do odczytywania listy wątków na forum, która będzie wyświetlana we Flashu.

viewthread.php

Skrypt podobny do skryptu *viewforum.php*. Posłuży on do odczytywania wypowiedzi w wybranym wątku.

postnew.php

Jeśli użytkownik zechce utworzyć nowy wątek, potrzebny będzie skrypt przetwarzający dane. Zadaniem tego skryptu będzie obsługa zamieszczania nowych wątków.

postreply.php

Ten skrypt jest podobny do poprzedniego. Będzie on obsługiwał odpowiedzi na istniejące wątki.

register.php

Trzeba także zadbać o umożliwienie użytkownikom rejestracji, koniecznej dla uzyskania dostępu do funkcji umieszczania wypowiedzi na forum. Skrypt ten umożliwi przeprowadzanie takiej właśnie rejestracji.

Warto zwrócić uwagę na fakt, że wszystkie te skrypty są funkcjonalnie wyspecjalizowane. Można by utworzyć dwa lub trzy skrypty, które pełniłyby te same zadania. Jednakże tworzenie małych, wyspecjalizowanych skryptów zapewnia im większą elastyczność i zrozumiałość. Oznacza to również uniknięcie wykonywania na serwerze zbędnych funkcji.

A więc, czas na PHP!

Skrypt common.php

Podobnie jak w przypadku poprzednich, wieloskryptowych aplikacji, tak i tutaj wszelkie szczegóły dotyczące bazy danych oraz funkcje ogólnego zastosowania będziemy przechowywać w pojedynczym pliku. Plik ten będzie dołączany do pozostałych skryptów za pomocą funkcji `include`.

1. Pierwsza część tego skryptu będzie dokładnie taka sama, jak w poprzednich przypadkach. Ograniczymy się zatem jedynie do przedstawienia jej listingu — jednakże skrypt został opatrzony komentarzami, a pełne objaśnienia można znaleźć, w razie potrzeby, w rozdziale 10.

```
<?
// common.php
// Case Study 3: Forum - Foundation PHP for Flash

// Szczegóły bazy danych
$dbHost = "localhost";
$dbUser = "uzytkownik";
$dbPass = "haslo";
$dbName = "phpforflash";


// Funkcje wspólne

/*****************************************************************
**  Funkcja: dbconnect()                                        **
**     Opis: Wyświetla połączenie z bazą danych na serwerze i  **
**           operacje wybierania bazy danych                   **
*****************************************************************/
function dbConnect() {
    // Dostęp do zmiennych globalnych
    global $dbHost;
    global $dbUser;
    global $dbPass;
    global $dbName;

    // Otwarcie połączenia z serwerem bazy danych
    $link = @mysql_connect($dbHost, $dbUser, $dbPass);

    // Jeśli otwarcie połączenia nie powiodło się...
    if (!$link) {
        // Informujemy Flasha o błędzie i opuszczamy skrypt
        fail("Couldn't connect to database server");
    }

    // Wybieramy bazę danych. Jeśli się to nie powiedzie...
    if (!@mysql_select_db($dbName)) {
        // Informujemy Flasha o błędzie i opuszczamy skrypt
        fail("Couldn't find database $dbName");
    }

    return $link;
}
```

```
/************************************************************
**   Funkcja: fail()                                       **
** Parametry: $errorMsg - Informacja o błędzie użytkownika  **
**     Opis: Raportuje informacje o błędzie, zwraca        **
**           z powrotem do filmu Flasha i opuszcza skrypt    **
************************************************************/
function fail($errorMsg) {
    // Informacja o błędzie kodu URL
    $errorMsg = urlencode($errorMsg);

    // Wyświetlamy informację o błędzie i opuszczamy skrypt
    print "&result=Fail&errormsg=$errorMsg";
    exit;
}
```

2. Uwzględniając potrzeby naszego forum do skryptu dodaliśmy jedną funkcję — jej zadaniem będzie weryfikacja nazwy użytkownika i hasła oraz porównywanie ich z danymi zapisanymi w bazie danych. Zapisz zatem tę funkcję w skrypcie.

```
function auth($username, $password) {
```

   Z powyższego wynika, że funkcja ta nosi nazwę `auth`, a przekazywać jej będziemy argumenty `username` i `password`, które należy poddać weryfikacji.

3. Teraz trzeba przeprowadzić szyfrowanie wpisanego hasła. Jest to potrzebny zabieg, gdyż hasła będą przechowywane w bazie danych w postaci zaszyfrowanej. Zatem aby porównać hasło podane przez użytkownika z hasłem zapisanym w bazie, należy je zaszyfrować.

```
// Szyfrujemy hasło
$crypt = md5($password);
```

   Funkcja `md5` wykorzystuje algorytm, zwany **mieszaniem md5**, którego działaniu można poddawać ciągi znaków. W ten sposób można otrzymać unikatową, zaszyfrowaną wersję oryginalnego ciągu znaków.

4. Po zaszyfrowaniu hasła należy użyć zapytania sprawdzającego, czy którykolwiek element tabeli `forumUsers` pasuje do podanych informacji.

```
$query = "SELECT userID FROM forumUsers WHERE username = '$username' AND
➥ password = '$crypt'";
```

5. Następnie trzeba wykonać zapytanie. Należy także poddać sprawdzeniu wartość zwracaną przez funkcję `mysql_query()`, która dostarczy informacji o odnalezieniu pasującego do siebie zestawu nazwy użytkownika i hasła. Jeśli taki zestaw zostanie odnaleziony, ze zwróconych wyników zostanie wydzielony identyfikator `userID`. Jeżeli jednak obecność takiego zestawu nie zostanie stwierdzona, `$userID` przyjmie wartość `-1`, co będzie świadczyło o braku autoryzacji użytkownika.

```
// Wykonujemy zapytanie
$result = mysql_query($query);

// Jeśli odpowiednik został znaleziony...
if (mysql_num_rows($result) == 1) {
    // Uzyskujemy identyfikator użytkownika z wyników
    $user = mysql_fetch_array($result);
    $userID = $user['userID'];
```

```
    } else {
        // W przeciwnym razie ustawiamy nazwę użytkownika na -1
        $userID = -1;
    }
```

6. Na koniec funkcja zwraca wartość zmiennej `$userID` i przekazuje ją funkcji wywołującej.

```
    // Zwracamy identyfikator użytkownika
    return $userID;
}

?>
```

7. Ponadto zastosujemy tu funkcję umożliwiającą weryfikowanie poprawności adresu e-mail. Wykorzystamy w tym celu skomplikowane **wyrażenie regularne**, które, na szczęście, omawialiśmy już w rozdziale 5.

```
function checkEmail($email)
{
    // Definiujemy wyrażenie regularne
    $regexp = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*
    ➥ (\.[a-z]{2,3})$";

    if (eregi($regexp, $email))  {
          return true;
    }
    else
    {
      return false;
    }
}
```

Mówiąc najprościej, funkcja ta zwraca wartość `true`, jeśli podany adres e-mail będzie poprawnie sformułowany. W przeciwnym razie jest zwracana wartość `false`.

Skrypt setup.php

Następnym opracowywanym skryptem będzie skrypt przygotowawczy, służący do tworzenia struktury bazy danych i tabel dla naszej aplikacji. Także i w tym przypadku treść skryptu będzie bardzo podobna do tego, który wykorzystywaliśmy w poprzednim przykładzie. Jedyną różnicą jest to, że teraz należy przygotować trzy tabele zamiast jednej.

Jeśli poniższy kod będzie wymagał dodatkowych wyjaśnień, znajdują się one w sekcji opisującej skrypt przygotowawczy w rozdziale 10.

```
<?
// setup.php
// Case Study 3 - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');

// Otwieramy połączenie z serwerem bazy danych
link = @mysql_connect($dbHost, $dbUser, $dbPass);
```

```
// Jesli otwarcie połączenia nie powiodło się...
if (!$link) {
    // Informujemy użytkownika o błędzie i opuszczamy skrypt
    print "Couldn't connect to database server";
    exit;
}

// Tworzymy bazę danych
print "Attempting to create database $dbName <br>\n";
if(!@mysql_create_db($dbName)) {
    // Informujemy użytkownika o błędzie
    print "# Couldn't create database <br>\n";
} else {
    // Informujemy użytkownika o powodzeniu
    print "# Database created successfully <br>\n";
}

// Wybieramy bazę danych
print "Attempting to select database $dbName <br>\n";
if(!@mysql_select_db($dbName)) {
    // Informujemy użytkownika o błęzie i opuszczamy skrypt
    print "# Couldn't select database <br>\n";
    exit;
} else {
    // Informujemy użytkownika o powodzeniu
    print "# Database selected successfully <br>\n";
}


print "Attempting to create tables<br>\n";

// Tworzymy tabelę użytkowników
$query = "CREATE TABLE forumUsers (
          userID INTEGER AUTO_INCREMENT PRIMARY KEY,
          username VARCHAR(20),
          password VARCHAR(40),
          title VARCHAR(30),
          email VARCHAR(255)";

$result = @mysql_query($query);

if (!$result) {
    // Informujemy użytkownika o wystąpieniu błędu
    print "# Error creating forumUsers table<br>\n";
    print mysql_error();
} else {
    // Informujemy użytkownika o powodzeniu
    print "# forumUsers table created<br>\n";
}

// Tworzymy tabelę wątków
$query = "CREATE TABLE forumThreads (
          threadID INTEGER AUTO_INCREMENT PRIMARY KEY,
          userID INTEGER,
          topic VARCHAR(100),
          replies INTEGER DEFAULT 0,
          lastPost INTEGER)";
```

```
$result = @mysql_query($query);

if (!$result) {
    // Informujemy użytkownika o błędzie
    print "# Error creating forumThreads table<br>\n";
    print mysql_error();
} else {
    // Informujemy użytkownika o powodzeniu
    print "# forumThreads table created<br>\n";
}

// Tworzymy tabelę wypowiedzi
$query = "CREATE TABLE forumPosts (
          postID INTEGER AUTO_INCREMENT PRIMARY KEY,
          threadID INTEGER,
          userID INTEGER,
          message MEDIUMTEXT,
          posted INTEGER)";

$result = @mysql_query($query);

if (!$result) {
    // Informujemy użytkownika o błędzie
    print "# Error creating forumPosts table<br>\n";
    print mysql_error();
} else {
    // Informujemy użytkownika o powodzeniu
    print "# forumPosts table created<br>\n";
}

print "End of setup";

?>
```

Skrypt viewforum.php

Teraz opracujemy bardziej zagmatwane skrypty PHP. Skrypt *viewforum.php* posłuży do odczytywania wszystkich wątków na forum wyświetlanym we Flashu.

Teraz kiedy Czytelnicy stali się już znawcami PHP, powinni umieć wskazać fragmenty, z którymi już mieli okazję się zetknąć. Z tego też powodu nie będziemy szeroko komentować zagadnień, które już omawialiśmy. Ograniczymy się do skrótowych objaśnień dotyczących przeznaczenia poszczególnych fragmentów kodu.

1. Jak zwykle, rozpocznij zapisywanie skryptu od określenia załadowania pliku konfiguracyjnego, nawiązania połączenia z serwerem bazy danych oraz wybrania bazy dla aplikacji.

```
<?
// viewforum.php
// Case Study 3: Forum - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');
```

```
// Otwieramy połączenie z bazą danych
$link = dbConnect();
```

2. Następnie zbuduj zapytanie odczytujące wszystkie wątki na forum. Zwróć uwagę, że stosujemy tu klauzulę `ORDER BY`, która ma zapewnić wyświetlanie wątków w kolejności odwrotnie chronologicznej.

```
// Budujemy zapytanie odczytujące forum
$query = "SELECT * FROM forumThreads ORDER BY lastPost DESC";
```

3. Kolejną operacją będzie wykonanie zapytania i, w razie niepowodzenia, wyświetlenie komunikatu o błędzie.

```
// Wykonujemy zapytanie
$result = mysql_query($query);

// Jeśli wykonanie zapytania nie powiodło się...
if (!$result) {
    // Informujemy użytkownika o błędzie i opuszczamy skrypt
    fail("Couldn't list threads from database");
}
```

Jeśli operacja przebiegnie poprawnie, odczytana będzie liczba wątków na forum. Posłuży do tego funkcja `mysql_num_rows`. Warto tu sobie przypomnieć treść rozdziału 9., gdzie mówiliśmy, że funkcja ta zwraca liczbę elementów w zbiorze wyników, co poprzedza wykonanie polecenia `SELECT`.

```
// Sprawdzamy liczbę wątków na forum
$threadCount = mysql_num_rows($result);
```

4. Następnie liczba wątków dopisywana jest do pierwszej zmiennej, zwracanej do Flasha. Zwróć uwagę, że dodajemy teraz tylko tę wartość — przed otrzymaniem końcowego wyniku dodamy do zmiennej dalsze wartości!

```
// Tworzymy zmienną przechowującą wynik
$output = "threadCount=$threadCount";
```

5. Następnie należy uruchomić pętlę `for`, przetwarzającą kolejne wątki zwracane przez polecenie `SELECT`.

```
// Dla każdego zwróconego wątku...
for ($count = 0; $count < $threadCount; $count++)
{
```

6. Pętla ta odczytuje, za pomocą funkcji `mysql_fetch_array`, kolejne wątki ze zbioru wyników MySQL zawartych w tablicy.

Tablicę tę wykorzystamy do utworzenia kilku zmiennych o opisowych nazwach. Przeprowadzimy tu usuwanie ukośników ze wszystkich elementów, które tego wymagają. Dodatkowo, za pomocą funkcji `strftime`, dokonamy konwersji uniksowych znaczników czasu reprezentujących daty i godziny utworzenia poszczególnych wątków.

```
    // Otrzymujemy z bazy danych szczegóły wypowiedzi
    $thread = mysql_fetch_array($result);
    $threadID = $thread['threadID'];
    $userID = $thread['userID'];
    $topic = stripslashes($thread['topic']);
    $replies = $thread['replies'];
    $lastPost = strftime("%d/%m/%y %H:%M", $thread['lastPost']);
```

7. Kolejny fragment może wyglądać nieco dziwnie, ale jest to jedynie wykonanie kolejnego zapytania. Zapytanie to będzie odczytywało nazwę użytkownika, który utworzył bieżący wątek. Należy w tym celu wykonać odrębne zapytanie, gdyż w tabeli `forumThreads` przechowywany jest tylko identyfikator `userID`. Wartość ta posłuży do wybrania odpowiedniego użytkownika z tabeli `forumUsers`.

```
// Budujemy i uruchamiamy zapytanie odczytujące nazwę użytkownika,
// który utworzył dany wątek
$query = "SELECT username FROM forumUsers WHERE userID = $userID";
$result2 = @mysql_query($query);

// Otrzymujemy z wyników informacje o użytkowniku...
$user = @mysql_fetch_array($result2);
$username = $user['username'];
```

8. Następnie trzeba dopisać szczegóły każdego wątku do zmiennej `$output`, przygotowując je w ten sposób do przesłania do filmu Flasha. Warto teraz przypomnieć sobie rozważania prowadzone w chwili, gdy powstawał kod ActionScript dla ujęcia Forum *View* filmu Flasha. Dyskutowaliśmy wtedy na temat formatu, jaki należałoby nadać wynikom zwracanym przez skrypt, aby umożliwić efektywną obsługę informacje we Flashu. W poniższym fragmencie kodu uwzględniliśmy ten właśnie format!

```
    // Dopisujemy szczegóły wątku do wyniku
    $output .= "&thread" . $count . "ID=" . $threadID;
    $output .= "&thread" . $count . "Topic=" . urlencode($topic);
    $output .= "&thread" . $count . "TopicStarter=" . urlencode($username);
    $output .= "&thread" . $count . "Replies=" . $replies;
    $output .= "&thread" . $count . "LastPost=" . $lastPost;
}
```

9. Skrypt kończy się przesłaniem zachowanych wyników z powrotem do Flasha. Ponadto dodamy zmienną, przesyłającą do Flasha informację o powodzeniu operacji, po czym nastąpi zamknięcie połączenia z serwerem MySQL.

```
// Wyświetlamy wszystkie wątki w jednym ciągu
echo $output;

// Informaujemy Flasha o powodzeniu
print "&result=Okay";

// Zamykamy połączenie z serwerem bazy danych
mysql_close($link);

?>
```

Skrypt viewthread.php

Teraz przedstawimy sposób tworzenia skryptu, który będzie służył do odczytywania wszystkich wypowiedzi wybranego wątku. Jego wywołanie nastąpi, jeśli użytkownik kliknie jeden z wątków wyświetlonych w widoku *Forum View*.

Także i w tym przypadku większa część kodu powinna być Czytelnikowi znana, a zatem będzie on omówiony dość ogólnie. Szerszych objaśnień można jednak poszukać w poprzednich rozdziałach niniejszej książki!

1. Tak jak dotychczas, pisanie skryptu rozpocznij od określenia załadowania pliku konfiguracyjnego, otwarcia połączenia z serwerem i wybrania bazy danych dla aplikacji. Zagadnienie to powinno być już dobrze znane wszystkim Czytelnikom.

```
<?
// viewthread.php
// Case Study 3: Forum - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');

// Otwieramy połączenie z bazą danych
$link = dbConnect();
```

2. Kolejną operacją jest zbudowanie zapytania odczytującego wszystkie wypowiedzi w wybranym wątku (identyfikowanym przez zmienną `$threadID`, przekazywanej z filmu Flasha) i zwracającego je do Flasha.

```
// Budujemy zapytanie odczytujące wątek
$query = "SELECT * FROM forumPosts WHERE threadID = $threadID ORDER BY posted ASC";
```

3. Następnie trzeba wykonać zapytanie i w razie niepowodzenia wygenerować komunikat o błędzie.

```
// Wykonujemy zapytanie
$result = @mysql_query($query);

// Jeśli wykonanie zapytania nie powiodło się...
if (!$result) {
    // Informujemy Flasha o błędzie i opuszczamy skrypt
    fail("Couldn't fetch posts from database");
}
```

4. Jeżeli wszystko się powiedzie, funkcja `mysql_num_rows` odczyta liczbę wypowiedzi w wybranym wątku.

```
// Sprawdzamy liczbę wypowiedzi w tym wątku
$postCount = @mysql_num_rows($result);
```

5. Następnie liczba wypowiedzi jest zapisywana w pierwszej zmiennej odsyłanej do Flasha.

```
// Przygotowujemy zmienną przechowującą wynik
$output = "&postCount=$postCount";
```

6. Dalej pojawia pętla `for`, przetwarzająca kolejne wypowiedzi zwracane przez polecenie `SELECT`.

```
// Dla każdej zwróconej wypowiedzi...
for ($count = 0; $count < $postCount; $count++) {
```

7. Pierwszą czynnością, jaką należy wykonać w pętli, jest zastosowanie funkcji `mysql_fetch_array` do odczytania kolejnej wypowiedzi ze zbioru wyników MySQL i zwrócenie jej w postaci tablicy.

```
    // Uzyskujemy z bazy danych szczegóły wypowiedzi
    $post = mysql_fetch_array($result);
    $userID = $post['userID'];
    $message = stripslashes($post['message']);
    $posted = strftime("%d/%m/%y %H:%M", $post['posted']);
```

Jeszcze raz tworzymy odpowiednie zmienne. Operacja ta obejmuje usunięcie ukośników ze wszystkich wymagających tego elementów oraz konwersję uniksowych znaczników czasowych za pomocą funkcji `strftime`.

8. Kolejny fragment można sobie przypomnieć z poprzedniego skryptu. Służy on do odczytywania szczegółów dotyczących użytkownika, który przesłał wypowiedź. W tym celu wykorzystano zmienną `$userID`, przechowywaną wraz z bieżącą wypowiedzią w `forumPosts`.

```
// Budujemy i uruchamiamy zapytanie odczytujące nazwę
// i tytuł użytkownika, który utworzył wypowiedź
$query = "SELECT username, title FROM forumUsers WHERE userID = $userID";
$result2 = @mysql_query($query);

// Otrzymujemy z wyniku informację o użytkowniku
$user = @mysql_fetch_array($result2);
$username = $user['username'];
$userTitle = $user['title'];
```

9. Teraz należy dodać uzyskane szczegóły do zmiennej `$output`, przygotowując ją w ten sposób do odesłania do filmu Flasha.

```
// Dopisujemy szczegóły wypowiedzi do wyniku
$output .= "&post" . $count . "Author=" . urlencode($username);
$output .= "&post" . $count . "Date=" . urlencode($posted);
$output .= "&post" . $count . "UserTitle=" . urlencode($userTitle);
$output .= "&post" . $count . "Message=" . urlencode($message);
}
```

10. Na koniec zachowane wyniki są odsyłane do Flasha. Ponadto dodajemy zmienną, przesyłającą do filmu Flasha informację o powodzeniu operacji. Ostatnim zabiegiem jest zamknięcie połączenia z serwerem MySQL.

```
// Wyświetlamy wszystkie wypowiedzi w jednym ciągu
echo $output;

// Informujemy Flasha o powodzeniu
print "&result=Okay";

// Zamykamy połączenie z serwerem bazy danych
mysql_close($link);

?>
```

Skrypt postnew.php

Po opracowaniu skryptów pracujących po wizualnej stronie działania aplikacji należy utworzyć skrypty, które umożliwią nam dopisywanie i manipulacje danymi.

Pierwszym z nich jest *postnew.php*. Jego zadaniem jest współpraca z sekcją *Post New* filmu Flasha w celu umożliwienia tworzenia i umieszczania nowych wątków na forum.

1. Zapewne wszyscy Czytelnicy rozpoznają pierwszy fragment kodu! A zatem, czyńmy swoją powinność...

```
<?
// postnew.php
// Case Study 3: Forum - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');

// Otwieramy połączenie z bazą danych
$link = dbConnect();
```

Warto się zastanowić, jak często trzeba by wpisywać kod połączenia z bazą danych, gdyby nie utworzony wcześniej plik *common.php*!

2. Po otwarciu połączenia z bazą danych można zastosować funkcję `auth`, zapisaną podczas tworzenia pliku *common.php*, w celu porównania informacji przekazanych przez film Flasha z danymi użytkownika i w ten sposób dokonania autoryzacji.

```
// Autoryzacja dostępu użytkownika do bazy danych
$userID = auth($username, $password);
```

Przypomnijmy, że jeśli wspomniana funkcja znajdzie zgodność podanych informacji z zawartymi w bazie danymi, zwraca identyfikator `userID`. W przeciwnym razie funkcja zwraca wartość `-1`...

3. Sprawdzając tę wartość można stwierdzić, czy weryfikacja użytkownika przebiegła pomyślnie. Jeżeli podane przez niego informacje nie zostaną dopasowane do zawartości bazy danych, do Flasha trafi komunikat o błędzie i nastąpi opuszczenie skryptu.

```
// Jeśli autoryzacja nie powiodła się...
if ($userID == -1) {
    // Informujemy Flasha o błędzie i opuszczamy skrypt
    fail("Invalid username and/or password");
}
```

4. Następnie należy odczytać czas bieżący, zapisany w postaci uniksowego znacznika czasowego, stosowanego w zapytaniach badających datę i godzinę utworzenia nowego wątku i wypowiedzi.

```
// Odczytujemy czas bieżący
$posted = time();
```

5. Teraz utwórz pierwsze z dwóch, koniecznych w tym skrypcie, zapytań. Posłuży ono do tworzenia nowych wątków.

```
// Budujemy i uruchamiamy zapytanie wstawiające nowy wątek
$query = "INSERT INTO forumThreads (userID, topic, lastPost) VALUES ($userID,
➥ '$topic', $posted)";
```

Następnie należy uruchomić zapytanie. Jeśli operacja nie powiedzie się, wątek nie zostanie utworzony. W takiej sytuacji do Flasha trafi komunikat o błędzie i nastąpi wyjście ze skryptu.

```
if(!mysql_query($query)) {
    fail("Error inserting thread");
}
```

6. Kolejnym elementem jest funkcja związana z MySQL, której dotychczas nie omawialiśmy, przede wszystkim dlatego, że dotąd nie znaleźliśmy dla niej odpowiedniego zastosowania. Funkcja `mysql_insert_id` jest dość precyzyjnym narzędziem. Zwraca ona ostatnią liczbę typu integer, wygenerowaną dla kolumny wskazanej jako `AUTO_INCREMENT`, podczas połączenia z bieżącą bazą danych.

```
// Odczytujemy identyfikator threadID nowego wątku
$threadID = mysql_insert_id();
```

   W omawianym przypadku kolumną tabeli `forumThreads` oznaczoną atrybutem `AUTO_INCREMENT` jest kolumna `threadID`. Funkcja `mysql_insert_id` jest uruchamiana po pomyślnym wprowadzeniu nowego wątku (zgodnie z powyższym opisem), co pozwoli na użycie `$threadID` w momencie dodawania nowej wypowiedzi do nowego wątku (czym zajmiemy za chwilę).

7. Czas na zbudowanie zapytania dopisującego nową wypowiedź do tabeli `forumPosts`. Do powiązania tej nowej wypowiedzi z nowo utworzonym wątkiem wykorzystujemy wartość `$threadID` uzyskaną dzięki funkcji `mysql_insert_id` w poprzedniej sekcji.

```
// Budujemy i uruchamiamy zapytanie wstawiające nową wypowiedź
$query = "INSERT INTO forumPosts (threadID, userID, message, posted)
➥ VALUES($threadID, $userID, '$message', $posted)";
```

8. Dalej należy wykonać zapytanie. Jeśli operacja ta nie powiedzie się, będzie to równoznaczne z niemożnością dodania nowej wypowiedzi. Wtedy należy przesłać do Flasha informację o niepowodzeniu i wyjść ze skryptu.

```
if(!mysql_query($query)) {
    fail("Error inserting post");
}
```

9. Skrypt należy zakończyć przesłaniem do Flasha informacji o powodzeniu operacji i zamknięciem połączenia z bazą danych MySQL.

```
// Informujemy Flasha o powodzeniu
print "&result=Okay";

// Zamykamy połączenie z serwerem bazy danych
mysql_close($link);

?>
```

No cóż, opracowaliśmy już pięć skryptów, a dwa są jeszcze wciąż przed nami. Niektóre skrypty prawdopodobnie nie sprawiają Czytelnikom już żadnych trudności.

Skrypt postreply.php

Ten skrypt będzie obsługiwał wszystkie żądania dopisania odpowiedzi do istniejącego wątku. Jego działanie będzie bardzo podobne do *postnew.php*, a to dlatego, że obydwa służą do umieszczania wypowiedzi na forum! Główna różnica polega na tym, że zadaniem tego skryptu jest uaktualnianie wiersza w tabeli `forumThread`, a nie tworzenie nowego. Jest logiczne, że aby odpowiedzieć na wątek, musi on już istnieć!

1. Zacznij od starych, dobrych znajomych elementów skryptu.

```
<?
// postreply.php
// Case Study 3: Forum - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');

// Otwieramy połączenie z bazą danych
$link = dbConnect();

// Autoryzacja użytkownika
$userID = auth($username, $password);

// Jeśli autoryzacja nie powiodła się...
if ($userID == -1) {
    // Informujemy Flasha i opuszczamy skrypt
    fail("Invalid username and/or password");
}
```

   Po otwarciu połączenia z bazą danych następuje weryfikacja informacji użytkownika dostarczonych z Flasha poprzez porównanie ich z danymi zarejestrowanego użytkownika. Służy do tego funkcja `auth`. Jeśli weryfikacja przebiegnie niepomyślnie, przesyłamy do Flasha odpowiednią informację, a następnie opuszczamy skrypt!

2. Następnie odczytujemy bieżący czas w postaci uniksowego znacznika czasu.

```
// Odczytujemy czas bieżący
$posted = time();
```

3. Następnie tworzymy zapytanie, które posłuży do dopisania nowej wypowiedzi do tabeli `forumPosts` lub, w razie konieczności, do wygenerowania komunikatu o błędzie. Jest to dokładnie ten sam kod, co w poprzednim skrypcie, z tą tylko różnicą, że tym razem zmienna `$threadID` jest dostarczana przez film Flasha.

```
// Budujemy i uruchamiamy zapytanie wstawiające nową wypowiedź
$query = "INSERT INTO forumPosts (threadID, userID, message, posted)
VALUES($threadID, $userID, '$message', $posted)";
if(!mysql_query($query)) {
    fail("Error inserting thread");
}
```

4. Następnie należy utworzyć i uruchomić zapytanie uaktualniające dane w tabeli `forumThreads` dotyczące wątku wskazanego przez `$threadID`. Mówiąc najprościej, polega to na dodaniu wartości 1 do liczby odpowiedzi na wątek i uaktualnieniu znacznika czasowego `lastPost`.

```
// Budujemy i uruchamiamy zapytanie uaktualniające liczbę wypowiedzi w wątku
$query = "UPDATE forumThreads SET replies = replies + 1, lastPost = $posted WHERE
➥ threadID = $threadID";
if(!mysql_query($query)) {
    fail("Error inserting thread");
}
```

5. Na zakończenie skryptu przesyłamy do Flasha informację o powodzeniu operacji i zamykamy połączenie z bazą danych MySQL.

```
// Informujemy Flasha o powodzeniu
print "&result=Okay";

// Zamykamy połączenie z serwerem bazy danych
mysql_close($link);

?>
```

Został nam jeszcze tylko jeden skrypt...

Skrypt register.php

```
<?
// register.php
// Case Study 3: Forum - Foundation PHP for Flash

// Dołączamy plik konfiguracyjny
include('common.php');

// Otwieramy połączenie z bazą danych
$link = dbConnect();
```

1. Następny fragment skryptu służy do ustanawiania tytułu, który będzie nadawany nowym, rejestrującym się użytkownikom. Tytuł użytkownika pojawiać się będzie poniżej jego nazwy, w widoku wątku *Thread View* i, ogólnie rzecz biorąc, jest on przeznaczony do identyfikacji statusu użytkownika. My ograniczyliśmy się do jednego tytułu dla wszystkich użytkowników, oprócz utworzonego wcześniej konta administratorskiego. Istnieje jednak zawsze możliwość zmiany zasad wedle życzenia projektanta aplikacji!

   ```
   // Ustalamy tytuł dla nowych użytkowników
   $title = "Code Junkie";
   ```

   Wróć do miejsca, w którym zapisywaliśmy funkcję `auth` w skrypcie *common.php*. Mówiliśmy tam o tym, że hasła użytkowników będą przechowywane w bazie danych w postaci zaszyfrowanej. Zastosujemy tu znowu funkcję mieszania `md5`, gdyż jest ona prosta w użyciu i zapewnia skuteczne i jednoznaczne szyfrowanie.

   Dlatego też najpierw musimy zaszyfrować hasło dostarczone skryptowi przez film Flasha.

   ```
   // Szyfrujemy hasło
   $crypt = md5($password);
   ```

   Następnie za pomocą funkcji `checkMail` ze skryptu *common.php* sprawdzamy poprawność podanego adresu e-mail.

   ```
   // Jeśli adres e-mail jest niepoprawny...
   if (!checkEmail($email)) {
       // Informujemy Flasha o błędzie i opuszczamy skrypt
       fail("Invalid email address");
   }
   ```

2. Następnie trzeba się upewnić, że podana nazwa użytkownika nie figuruje w tabeli `forumUsers`. Jeśli taka sama nazwa zostanie tam odnaleziona, należy przesłać do Flasha raport o błędzie i wyjść ze skryptu!

```
// Budujemy zapytanie wyszukujące duplikaty adresów e-mail i nazw użytkowników
$query = "SELECT * FROM forumUsers WHERE username='$username'";

if(!mysql_query($query)) {
    fail("Couldn't search database for duplicates");
}

// Jeśli odpowiednik został znaleziony...
if (mysql_num_rows($query) != 0) {
    // Informujemy Flasha o błędzie i opuszczamy skrypt!
    fail("Username $username already registered");
}
```

3. Kolejnym elementem jest zapytanie wprowadzające nowego użytkownika do tabeli `forumUsers`. Zwróć uwagę na zastosowanie `$crypt` zamiast `$password`, co pozwala na zachowanie zaszyfrowanej wersji oryginalnego hasła, a ponieważ w dalszym ciągu jest to ciąg znaków, należy ująć go w apostrofy!

```
// Budujemy zapytanie dodające nowego użytkownika
$query = "INSERT INTO forumUsers (username, password, title, email) VALUES
➥ ('$username', '$crypt', '$title', '$email')";
if(!mysql_query($query)) {
    fail("Username $username already exists");
}
```

4. Na koniec należy wysłać do Flasha raport o powodzeniu operacji i zamknąć połączenie z serwerem bazy danych.

```
// Informujemy Flasha o powodzeniu
print "&result=Okay";

// Zamykamy połączenie z serwerem bazy danych
mysql_close($link);

?>
```

To wszystko — niezbędne skrypty są już gotowe i teraz należy przesłać je na serwer, po czym uruchomić.