TCL-TK. Programowanie

Autor: Dariusz Chrobak
ISBN: 83-7197-969-X
Format: B5, stron: 272

Język programowania Tcl (Tool Command Language) należy do licznej rodziny interpretowanych języków skryptowych. Interpreter Tcl może zostać uruchomiony pod kontrolą wielu systemów operacyjnych, między innymi Linuksa i Windows. Aplikacja Tcl jest niemal w pełni niezależna od wyboru platformy systemowej. Tcl pozwala między innymi na uruchamiania wielu procesów (programów), korzystanie z przekierowań wejścia-wyjścia, tworzenie potoków poleceń i tworzenie gniazd sieciowych do komunikacji TCP/IP. Rozszerzeniem Tcl o trudnym do przecenienia znaczeniu jest pakiet narzędziowy Tk. Dzięki poleceniom Tk tworzenie i manipulowanie okienkami składającymi się na graficzny interfejs użytkownika (GUI) jest niezwykle proste.

Jeśli potrzebny jest Ci wygodny język skryptowy, w którym można szybko tworzyć niezależne od systemu operacyjnego aplikacje o rozbudowanym interfejsie użytkownika, Tcl/Tk może okazać się wymarzonym narzędziem do tego celu.

Książka „Tcl/Tk. Programowanie” to wyczerpujący i kompletny podręcznik tego języka programowania, wzbogacony wieloma przydatnymi przykładami.

W książce omówiono między innymi:

- Podstawy Tcl, środowisko pracy programisty
- Polecenia, zmienne, wykonywanie obliczeń, instrukcje sterujące, procedury i inne elementy składni języka
- Struktury danych: łańcuchy i wyrażenia regularne, listy i tablice
- Korzystanie z plików, potoków i gniazd sieciowych
- Pakiety i biblioteki, dostęp do baz danych
- Tworzenie interfejsu użytkownika z wykorzystaniem Tk
- Obsługę zdarzeń
- Kontrolki dostępne w Tk, tworzenie menu aplikacji

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

Spis treści

Rozdział 6.

Wprowadzenie do Tk

Tk (**T**ool **k**it) jest pakietem narzędziowym rozszerzającym standard Tcl o dodatkowe polecenia służące do tworzenia **graficznego interfejsu użytkownika** (GUI). Elementy GUI to kontrolki Tk. W dalszej części książki kontrolki będą często nazywane okienkami.

Pakiet narzędziowy Tk, oprócz poleceń tworzących kontrolki, dostarcza również polecenia służące do rozmieszczania kontrolek **kontenerów** (mogą nimi być inne kontrolki). Właściwie są to zbiory poleceń nazywane najczęściej **zarządcami geometrii** lub **zarządcami rozkładu**.

Istnieją również, a może przede wszystkim i takie polecenia, dzięki którym aplikacja Tk jest w stanie odpowiadać na różne zdarzenia generowane przez jej otoczenie. Dzięki temu kliknięcie przyciskiem myszki, gdy kursor znajduje się nad obszarem ekranu, zajmowanym przez określoną kontrolkę, może spowodować oczekiwaną (zaprogramowaną) przez użytkownika reakcję aplikacji.

Inną zaletą pakietu Tk jest wysoki stopień niezależności kodu aplikacji od wyboru platformy systemowej. Czy będzie to popularny system Windows, czy nieco bardziej wymagający X Window System dla platformy Linux, kontrolki Tk będą funkcjonowały niemal tak samo. Występujące drobne różnice nie powinny mieć większego wpływu na jakość tworzonego GUI.

Bieżący rozdział wprowadza niezbędne pojęcia związane z programowaniem przy użyciu pakietu narzędziowego Tk, pokazując przy tym współdziałanie oraz zależności pomiędzy poszczególnymi elementami graficznego interfejsu użytkownika.

Aplikacja Tk

Aplikacja Tk to wykonywany przez powłokę *wish* skrypt Tcl wzbogacony poleceniami pakietu narzędziowego Tk. Na skrypt Tcl-Tk składają się dwie zasadnicze części:

- część inicjująca graficzny interfejs użytkownika — GUI;
- procedury obsługi zdarzeń.

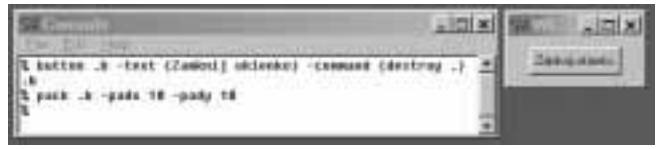
Po wykonaniu poleceń inicjujących GUI, aplikacja Tk oczekuje na zdarzenia generowane przez jej otoczenie. Na zajście zdarzenia aplikacja może odpowiedzieć wykonaniem odpowiedniej procedury obsługi.

Uruchomienie powłoki *wish* spowoduje pojawienie się **głównego okna aplikacji**. W systemie Windows dodatkowo pojawi się **konsola** (rysunek 1.3), okienko umożliwiające porozumiewanie się z programem powłoki. Główne okno aplikacji to pewien wyróżniony obszar ekranu, wokół którego zarządca okien utworzy „trójwymiarowe" krawędzie oraz doda pasek tytułowy wraz z kilkoma przyciskami, umożliwiającymi zmianę rozmiarów czy też zamknięcie okna. To ostanie działanie jest równoznaczne z zakończeniem działania aplikacji. Wygląd głównego okna aplikacji jest uzależniony od rodzaju platformy systemowej, na której uruchomiono powłokę *wish*.

Po uruchomieniu powłoki poleceniem `wish` (w systemie Linux) lub `wish83.exe` (w systemie Windows) można wydać kilka poleceń, tworząc prosty GUI (rysunek 6.1).

```
% button .b -text {Zamknij okienko} -command {destroy .}
=> .b
% pack .b -padx 10 -pady 10
=>
```

Rysunek 6.1. *Konsola i przycisk w głównym oknie aplikacji*



Powyższy przykład pokazuje **hierarchiczną strukturę** kontrolek aplikacji Tk. W głównym oknie aplikacji pojawiło się **okienko wewnętrzne** (przycisk), którego kliknięcie spowoduje zakończenie działania programu. Proszę zwrócić uwagę na nazwę przycisku: `.b`. Podstawową cechą okienek wewnętrznych jest to, że nie mogą być wyświetlane poza obszarem okienka, w którym się pojawiły. Można to łatwo sprawdzić, zmieniając myszką rozmiar głównego okna aplikacji.

Kolejnym rodzajem okienek Tk są tak zwane **okienka wysokiego poziomu**. Jedno z nich, utworzone poleceniem

```
% toplevel .a
=> a
```

zostało pokazane na rysunku 6.2. Okienka wysokiego poziomu tym różnią się od okienek wewnętrznych, że wyświetlane są niezależnie od głównego okna aplikacji. Podobnie też są traktowane przez zarządcę okien platformy systemowej. Zniszczenie głównego okna aplikacji spowoduje zniszczenie okienek wysokiego poziomu, ale nie odwrotnie.

Aplikację Tk można uruchomić jednym z poniższych poleceń. Dla systemu Windows będzie to polecenie:

```
wish83.exe ?nazwaPliku? ?opcje? ?arg arg arg ...?
```

Dla systemu Linux polecenie to będzie miało następującą formę:

```
wish ?nazwaPliku? ?opcje? ?arg arg arg ...?
```

Rysunek 6.2. *Główne okno aplikacji wraz z okienkiem wysokiego poziomu*

Argument *nazwaPliku* jest nazwą skryptu Tcl-Tk. Spośród kilku dostępnych opcji wspomnę o jednej:

`-geometry` *geometria*

Opcja ta pozwala określić początkowe położenie i rozmiar głównego okna aplikacji. Argument *geometria* powinien mieć następujący format:

szerokośćxwysokość±x±y

Wszystkie wielkości podawane są w jednostkach ekranu (piksele, punkty, milimetry, centymetry, cale). Znak *+x* (*-x*) oznacza odległość lewej (prawej) krawędzi okienka od lewej (prawej) krawędzi ekranu. Podobnie *+y* (*-y*) oznacza odległość górnej (dolnej) krawędzi okienka od górnej (dolnej) krawędzi ekranu.

Argumenty *arg arg ...* stanowią listę wartości przekazywanych do aplikacji. Powłoka *wish*, rozpoczynając wykonywanie skryptu, tworzy pięć zmiennych:

argc

Jest to liczba całkowita określająca ilość argumentów przekazywanych do skryptu. W przypadku ich braku wartością *argc* jest liczba 0.

argv

Jest to łańcuch znaków zawierający, oddzielone spacjami, argumenty skryptu. Jeżeli ich nie podano, łańcuch *argv* jest pusty.

argv0

Jest to łańcuch zawierający nazwę uruchomionego skryptu. W przeciwnym razie łańcuch *argv0* jest pusty.

geometry

Ten zapis oznacza, że zmiennej przypisywana jest wartość opcji `-geometry`.

tcl_interactive

Ta zmienna przyjmuje wartość 1, jeżeli interpreter został uruchomiony w trybie interaktywnym, bez specyfikacji nazwy skryptu. W przeciwnym razie zmienna przyjmuje wartość 0.

Kontrolki

Kontrolki Tk to okienka posiadające charakterystyczne właściwości, w tym charakterystyczne cechy graficzne. Kontrolki podzielone są na **klasy**. Nazwa klasy rozpoczyna się od dużej litery, np. **Button** (przycisk) i jest zwracana przez polecenie

```
wm class kontrolka
```

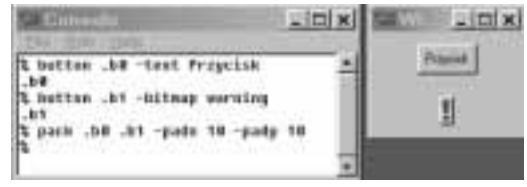
gdzie *kontrolka* jest nazwą kontrolki. Na przykład po tworzeniu pola wejściowego poleceniem

```
% entry .e
=> .e
```

kolejne polecenie zwróci nazwę klasy:

```
% wm class .e
=> Entry
```

Kontrolki należące do tej samej klasy charakteryzują się tym samym zestawem opcji konfiguracyjnych, dzięki którym możliwe jest nadanie im indywidualnego wyglądu i sposobu działania. Na przykład każda kontrolka wspomnianej już klasy **Button** może wyświetlać jeden z elementów: tekst, bitmapę lub obrazek (rysunek 6.3) i może w różny sposób reagować, powiedzmy na kliknięcie przyciskiem myszki.



Rysunek 6.3. *Indywidualne cechy wyglądu kontrolek klasy Button*

Lista standardowych klas kontrolek dostarczanych w pakiecie narzędziowym Tk jest wystarczająco obszerna, by móc tworzyć nawet skomplikowane GUI. W dalszych rozdziałach książki opisane zostaną następujące klasy kontrolek:

- **Button** (przycisk)
- **Frame** (ramka)
- **Canvas** (płótno)
- **Checkbutton** (pole wyboru)
- **Entry** (pole wejściowe)
- **Image** (element graficzny)
- **Label** (etykieta)
- **Listbox** (lista wyboru)
- **Menu** (pasek menu)
- **Menubutton** (przycisk menu)

- `Message` (wiadomość)
- `Radiobutton` (pole opcji)
- `Scale` (skala)
- `Scrollbar` (pasek przewijania)
- `Text` (pole tekstowe)
- `Toplevel` (okno wysokiego poziomu)

Jak już wspomniano, okienka aplikacji Tk zorganizowane są w sposób hierarchiczny. Na szczycie znajduje się główne okno aplikacji. Jest ono rodzicem (kontenerem) dla wszystkich pozostałych kontrolek (nawet okienek wysokiego poziomu), które z kolei również mogą być rodzicami innych okienek. Ta struktura odbija się w sposobie identyfikacji kontrolek. Można mówić o **ścieżkach nazw** lub **drzewiastej strukturze** kontrolek aplikacji Tk. Zapis `.a.b.c` oznacza, iż okienko `c` jest dzieckiem okienka `b`, którego rodzicem jest okienko `a`, będące bezpośrednim potomkiem głównego okna aplikacji `.`.

Ścieżki nazw, chociaż mogą nasuwać porównania z techniką programowania obiektowego (C++, Java), nie mają z nią nic wspólnego. Określają kontener dla kontrolek usytuowanych w hierarchii o jeden szczebel niżej.

Tworzenie i usuwanie kontrolek

Polecenie o nazwie takiej samej jak nazwa klasy, ale rozpoczynające się od małej litery jest nazwą polecenia tworzącego kontrolkę. Ogólna postać takiego polecenia jest następująca:

```
klasa kontrolka ?opcja wart ...?
```

Argument *klasa* odpowiada klasie kontrolki Tk. Argument *kontrolka* jest ścieżką nazwy kontrolki, a argumenty *opcja* i *wart* są odpowiednio **przełącznikiem opcji konfiguracyjnej** i jej wartością. Przełącznik opcji (dalej opcja) jest konstrukcją składającą się z myślnika i nazwy opcji (np.: `-background`). Polecenie zwraca nazwę kontrolki.

Interesujące jest to, że ścieżka nazwy utworzonej kontrolki sama staje się nazwą polecenia. Wynika stąd, że kontrolki posiadają własne zestawy poleceń, których postać jest następująca:

```
kontrolka arg0 ?arg ...?
```

Argument *arg0*, nazywany modyfikatorem, jest obowiązkowy, ponieważ definiuje operację wykonywaną przez lub na kontrolce *kontrolka*. Analogicznie do konstrukcji obiektowych można powiedzieć, że modyfikator określa metodę obiektu reprezentującego kontrolkę. Polecenie

```
kontrolka cget opcja
```

zwraca aktualną wartość opcji *opcja* kontrolki *kontrolka*, a polecenie

```
kontrolka configure ?opcja? ?wart opcja wart ?
```

umożliwia zmianę wartości wybranej opcji, czy też zwraca aktualną wartość jednej lub wszystkich opcji.

Wypróbujmy:

```
% button .b -text {Kliknij mnie}
=> .b
% .b cget -text
=> Kliknij mnie
```

Usunięcie jednej lub kilku kontrolek jest równie proste, jak ich tworzenie. Wystarczy wykonać polecenie `destroy` z argumentem będącym listą ścieżek nazw kontrolek przeznaczonych do usunięcia:

```
destroy kontrolka ?kontrolka ...?
```

Możemy teraz usunąć przycisk z ostatniego przykładu:

```
% destroy .b
=>
% .b cget -text
=> invalid command name ".b"
```

Rzeczywiście, przycisk został zniszczony, o czym świadczy reakcja powłoki *wish* na próbę pobrania wartości jednej z opcji konfiguracyjnych (`-text`).

Zarządcy rozkładów

O ostatecznych rozmiarach i położeniach kontrolek w obszarze GUI decydują zarządcy rozkładu (rysunek 6.4). Każdy z nich w charakterystyczny dla siebie sposób rozmieszcza grupę kontrolek w ich kontenerze. Zarządca `place` jest najprostszy i wymaga jawnego określenia takich parametrów, jak: szerokość, wysokość, a także współrzędne lewego górnego rogu kontrolki. Kolejny zarządca, `pack`, stara się upakować kontrolki, minimalizując ich rozmiary. Jego działanie jest konsekwencją narzucenia pewnych warunków, na przykład takiego, który nakazuje umieszczenie kontrolki przy lewej krawędzi dostępnego obszaru. Ostatnim standardowym zarządcą geometrii jest `grid`, grupujący kontrolki w komórkach siatki.

Rysunek 6.4. *Efekt rozmieszczenia trzech przycisków przez zarządców: place, pack i grid*

Wymiana informacji

Aplikacja Tk wymienia dane z użytkownikiem, innymi aplikacjami, a także z zarządcą okien platformy systemowej. Prawie zawsze zamierzona praca aplikacji wymaga współdziałania pomiędzy poszczególnymi jej elementami — kontrolkami.

Użytkownik generuje **zdarzenia**, korzystając z urządzeń peryferyjnych: myszki lub klawiatury. **Źródłem zdarzenia** staje się ta z kontrolek aplikacji, której bezpośrednio dotyczy operacja wykonana myszką lub klawiaturą (**typ zdarzenia**). Z każdym typem zdarzenia oraz jego źródłem można poleceniem `bind` powiązać procedurę obsługi wykonywaną w odpowiedzi na zajście zdarzenia. Na tym właśnie polega tak zwane zdarzeniowe oprogramowanie GUI. Użytkownik generuje zdarzenia, po czym aplikacja odpowiada wykonaniem dedykowanej procedury obsługi.

O ile do zajścia zdarzenia związanego z myszką wystarcza wprowadzenie jej kursora nad obszar kontrolki (zdarzenie typu `<Enter>`), o tyle reakcja na zdarzenia klawiatury uzależniona jest od tego, która z kontrolek aplikacji posiada **skupienie** (**focus**). W danej chwili, w odpowiedzi na zdarzenia klawiatury, mogą być wykonywane skrypty obsługi powiązane tylko z jedną kontrolką, tą właśnie która aktualnie posiada skupienie. Sposób zmiany skupienia uzależniony jest od klasy kontrolki. Dla okienek z klas **`Entry`** lub **`Text`** wystarczy kliknięcie nad ich obszarem lewym przyciskiem myszki, a dla kontrolek z klas **`Button`** czy **`Checkbutton`** konieczne jest użycie kombinacji klawiszy *Tab* lub *Shift+Tab*. Inne z kontrolek, na przykład należące do klasy **`Frame`**, mogą przyjmować skupienie tylko w wyniku wykonania polecenia `focus`.

Czasem zachodzi potrzeba ograniczenia reakcji na zachodzące zdarzenia tylko do jednego drzewa kontrolek. Mechanizm ten, realizowany przez polecenie `grab`, jest najczęściej wykorzystywany do budowy okienek dialogowych.

Inny sposób interakcji użytkownika z aplikacją polega na wykorzystaniu opcji `-command` kontrolek należących do takich klas, jak: **`Button`**, **`Checkbutton`** i **`Radiobutton`**. Opcja `-command` umożliwia, bez konieczności korzystania z polecenia `bind`, zdefiniowanie skryptu wykonywanego po kliknięciu lewego przycisku myszki nad obszarem wybranego okienka.

Wymieniać informacje mogą również dwie działające niezależne od siebie aplikacje. Polecenie `send` wysyła skrypt Tcl do innej aplikacji, gdzie jest wykonywany. Następnie wynik zostaje zwrócony do aplikacji wysyłającej. W ten sposób pewne zadania informatyczne można przydzielić do wykonania kilku programom.

Specyficznym oddziaływaniem pomiędzy aplikacją Tk, a jej otoczeniem jest wykorzystanie zarządcy okien platformy systemowej do zmiany geometrii głównego okienka aplikacji oraz okienek wysokiego poziomu. Korzystając ze zbioru poleceń `wm`, można dodatkowo sprecyzować szereg innych parametrów kontrolek wysokiego poziomu.

Jak wspomniano wyżej, współpraca pomiędzy samymi okienkami również nie należy do wyjątków. Kontrolki klas **`Canvas`**, **`Entry`**, **`Listbox`**, **`Text`** można łączyć z paskami przewijania (**`Scrollbar`**). W tym przypadku położenie suwaka na pasku przewijania jest odwzorowywane na widzialny obszar powiązanego z nim okienka.