

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2010

Python. Projekty do wykorzystania

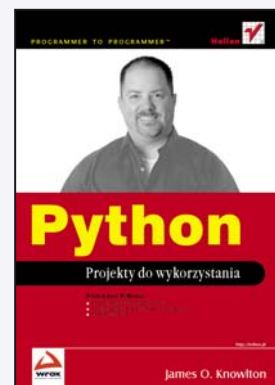
Autor: James O. Knowlton

Tłumaczenie: Marek Pętlicki

ISBN: 978-83-246-2200-9

Tytuł oryginału: [Python: Create - Modify - Reuse](#)

Format: 172×245, stron: 264



Poznaj moc Pythona!

- Jak przygotować środowisko pracy?
- Jak wykorzystać usługi systemu operacyjnego?
- Jak testować kod?

Historia języka Python sięga początku lat 90. Od tego czasu zdobył on sobie ogromną popularność i jest stosowany w wielu rozwiązaniach. Jego wydajność została doceniona również przez firmę Google podczas tworzenia platformy Google App Engine. Python w przeciwieństwie do wielu innych języków nie wymusza jednego sposobu programowania. Używając go, możesz programować obiektowo, strukturalnie i funkcjonalnie. Jeżeli do tego dodać automatyczny system zarządzania pamięcią oraz dynamicznie sprawdzane typy, rozwiązanie to nabiera kuszących rumieńców.

Dzięki tej książce dowiesz się, jak przygotować swoje środowisko pracy i rozpocząć przygodę z językiem Python. Autor zagwarantuje Ci, że będzie to kształcąca przygoda. Na konkretnych, praktycznych projektach pokaże Ci, jak wykorzystać potencjał drzemący w tym języku. Wśród przykładów znajdziesz opis takich zagadnień, jak wykonywanie migawek katalogów i plików, projektowanie katalogu filmów i systemu testującego wydajność WWW czy też tworzenie systemu gromadzenia opinii. Najważniejsze jest jednak to, że każde z tych rozwiązań możesz bez trudu zaadaptować do potrzeb Twojego projektu. Książka ta stanowi świetną lekturę zarówno dla zaawansowanych programistów Pythona, jak i tych, którzy chcą dopiero rozpocząć z nim pracę!

- Instalacja środowiska
- Składnia języka
- Operatory, wyrażenia, instrukcje
- Sterowanie przebiegiem programu
- Wykorzystanie modułów
- Praca z systemem plików
- Połączenie z bazą danych
- Uruchomienie serwera WWW w Pythonie
- Konfiguracja mod_python dla serwera Apache
- Interakcja z systemem operacyjnym
- Testowanie

Ułatwaj sobie pracę – korzystaj z bogactwa praktycznych projektów!

Spis treści

O autorze	11
Podziękowania	13
Wstęp	15
Rozdział 1. Wstęp do Pythona	21
Pierwsze kroki	21
Pobranie i instalacja Pythona	21
Interpreter Pythona	22
Środowisko edycyjno-wykonawcze	22
Struktura składniowa	23
Słowa kluczowe	24
Wiersze i wcięcia	24
Typy danych i identyfikatory	24
Operatory	26
Wyrażenia i instrukcje	27
Wyrażenia	27
Instrukcje	28
Iteracje i podejmowanie decyzji	28
Iteracje	29
Podejmowanie decyzji	30
Funkcje	31
Definicja funkcji	31
Moduły	32
Importowanie modułów	32
W jaki sposób Python wyszukuje ładowane moduły	33
Klasy	34
Podsumowanie	35

Część I Projekty

37

Rozdział 2. Program do wykonywania migawek katalogów i plików 39

Używanie programu	40
Tworzenie migawki	40
Wyświetlanie nazw plików migawek	41
Porównywanie migawek	41
Pomoc	43
Koniec pracy	43
Podsumowanie funkcji programu	43
Projekt	44
Elementy aplikacji	45
Program główny	46
Moduły	46
Analiza kodu	47
Snapshot.py	47
snapshothelper.py	54
Testowanie	60
Możliwości rozbudowy programu	61
Podsumowanie	61

Rozdział 3. System katalogu filmów DVD 63

Wykorzystanie programu	64
Instalacja bazy danych MySQL	64
Dodawanie filmu do bazy	65
Przeszukiwanie bazy filmów	66
Modyfikacja rekordu w bazie	69
Usunięcie rekordu	73
Eksport rekordów do pliku CSV	74
Projekt	74
Elementy aplikacji	75
Moduły	75
Omówienie kodu	77
dvd.py	78
add_dvd.py	80
lookup_dvds.py	82
modify_dvd.py	86
delete_dvd.py	91
csvreport_dvd.py	94
Testowanie	96
Potencjalna rozbudowa programu	96
Podsumowanie	96

Rozdział 4. Program testujący wydajność WWW 97

Używanie programu	98
Uruchamianie serwera WWW w Pythonie	98
Uruchomienie klienta testującego wydajność	99
Testowanie połączeń klienckich ze zdalnymi adresami WWW	100
Testowanie wydajności wbudowanego serwera WWW	101
Wyświetlenie dziennika	102

Projekt	103
Elementy aplikacji	104
Moduły	104
Analiza kodu	105
webserver.py	106
webperf.py	108
webclient.py	110
Pliki pomocnicze	116
Testowanie	116
Możliwe modyfikacje programu	117
Podsumowanie	117

Rozdział 5. System gromadzenia opinii użytkowników 119

Wykorzystanie programu	119
Wymagania wstępne	120
Uruchomienie programu	127
Projekt	129
Elementy aplikacji	129
Moduły	130
Analiza kodu	130
form.html	130
form.py	132
Testowanie	136
Modyfikowanie programu	137
Podsumowanie	137

Rozdział 6. System zarządzania testami 139

Używanie programu	139
Uruchamianie testów	140
Wyświetlanie listy wywołań testów	143
Wyświetlanie wyników testów	143
Zapis raportu w formacie HTML	144
Sprawdzanie zawartości pliku HTML	145
Wyświetlenie ekranu pomocy	145
Projekt	146
Moduły	146
Analiza kodu	148
test_manager.py	148
Program główny	149
test_run.py	152
test_list.py	159
test_results.py	160
test_html.py	162
Testowanie	165
Możliwości modyfikacji programu	166
Podsumowanie	166
Kilka informacji na temat modułu minidom	166

Rozdział 7. System weryfikacji wersji oprogramowania 167

Używanie programu	168
Konfiguracja zdalnych komputerów	168
Uruchamianie programu: składnia wiersza poleceń	168
Projekt	171
Moduły	171
Analiza kodu	173
version_checker.py	173
check_versions.py	176
csv_report.py	181
Testowanie	182
Możliwe modyfikacje programu	182
Kwestie bezpieczeństwa	182
Podsumowanie	183

Rozdział 8. System zarządzania treścią 185

Ogólne informacje o Plone	185
Czym jest Plone?	185
Instalacja i konfiguracja Plone	186
Pobieranie pakietu instalacyjnego Plone	186
Rozpakowanie pakietu instalacyjnego	187
Uruchomienie instalatora Plone	189
Uruchomienie Plone	189
Hasło konta admin serwera Plone	189
Zalogowanie na konto admin	190
Konfiguracja poczty e-mail	190
Dodanie konta użytkownika	192
Zalogowanie na koncie nowo utworzonego użytkownika	193
Projekt	194
Nawigacja	196
Zarządzanie treścią	196
Tworzenie strony	196
Tworzenie kolekcji	198
Uprawnienia użytkowników	202
Podsumowanie	202

Część II Zagadnienia zaawansowane 203**Rozdział 9. Interakcja z systemem operacyjnym 205**

Podstawowe usługi systemu operacyjnego	206
Moduł os	206
Moduł time — formatowanie i przekształcanie czasu systemowego	207
Moduł optparse — obsługa parametrów wywołania	209
Moduł platform — informacje o platformie systemowej	209
Moduł getpass — generowanie i weryfikacja haseł	210
Możliwości, jakie daje moduł	210
Wykorzystanie usług systemu Windows	211
Moduł winreg — dostęp do Rejestru Windows	211
Moduł winsound	211

Moduł win32serviceutil — zarządzanie usługami systemu Windows	213
Moduł win32net — wykorzystanie funkcji sieciowych systemu Windows	214
Inne możliwości	216
Wykorzystanie usług systemów Unix i Linux	216
Moduł termios — uniksowy interfejs TTY	216
Moduł resource — zarządzanie zasobami systemów Unix	217
Moduł syslog — zapis i odczyt uniksowego dziennika systemowego syslog	219
Moduł commands — wywoływanie poleceń i przechwytywanie wyników	221
Inne możliwości	222
Podsumowanie	223
Rozdział 10. Usuwanie błędów i testowanie	225
Debugger Pythona	225
Uruchamianie debugera	225
Graficzny debugger wbudowany w IDLE	227
Środowiska testowe Pythona	229
Dlaczego warto testować	229
Testy jednostkowe	230
Podsumowanie	239
Uwagi na zakończenie	239
Dodatek A Co dalej — zasoby, które mogą być pomocne	241
Dodatek B Instalacja dodatkowego oprogramowania	243
Skorowidz	253

5

System gromadzenia opinii użytkowników

Wszystkie aplikacje analizowane przez nas do tej pory wykorzystywały tekstowy interfejs użytkownika, obsługiwany przez wprowadzanie znaków w konsoli tekstowej. W tym rozdziale poznamy sposób wykorzystania modułu `mod_python` uzupełniający popularny serwer WWW Apache o obsługę języka Python, co pozwoli nam na obsługę logiki aplikacji WWW właśnie z użyciem kodu w Pythonie.

Aplikacja gromadząca opinie użytkowników będzie realizowała następujące funkcje:

- wykorzysta `mod_python` (moduł rozszerzeń serwera Apache) do obróbki danych wprowadzonych przez użytkownika do formularza HTML;
- za pomocą modułu `smtplib` połączy się z serwerem SMTP i wyśle wiadomość e-mail na predefiniowany adres e-mail webmastera;
- każdy komentarz wpisany przez użytkownika do formularza HTML zapisze dodatkowo w pliku dziennika w formacie CSV, zapisanym na serwerze WWW. Ten dziennik może być przeglądany i obrabiany za pomocą dowolnej aplikacji typu arkusz kalkulacyjny.

Wykorzystanie programu

Program posiada w zasadzie dwa interfejsy użytkownika: stronę WWW, na której użytkownik wprowadza komentarze, które są następnie wysyłane e-mailem, oraz plik dziennika, który może być przeglądany za pomocą dowolnej aplikacji arkusza kalkulacyjnego (zrzuty ekranu prezentowane w niniejszym rozdziale zostały wykonane za pomocą aplikacji Calc wchodzącej w skład pakietu OpenOffice.org).

Wymagania wstępne

Zanim uruchomimy naszą aplikację, potrzebujemy działającego serwera WWW Apache z zainstalowanym i skonfigurowanym modulem `mod_python`. Moduł `mod_python` to rozszerzenie serwera Apache uzupełniające jego możliwości o kilka dodatkowych mechanizmów (*handlerów*). W naszej aplikacji wykorzystamy *handler* o nazwie Publisher.

Czym jest handler?

W przypadku serwera Apache *handlerem* nazywamy instrukcję obsługiwaną przez serwer WWW, realizującą określone działania w reakcji na napotkanie pliku określonego typu. Można na przykład wyobrazić sobie handler uruchamiający program Adobe Reader w przypadku próby otwarcia pliku w formacie PDF.

Serwer Apache ma wbudowanych kilka *domyślnych* handlerów, a kilka dodatkowych jest dostępnych *opcjonalnie* — do aktywacji dla administratora, jeśli stwierdzi taką potrzebę (ich konfigurację można znaleźć w pliku `<katalog instalacyjne serwera>\conf\httpd.conf`). Moduł `mod_python` uzupełnia tę kolekcję handlerów o kilka specyficznych dla plików o rozszerzeniu `*.py`.

Instalacja serwera Apache

Apache to najpopularniejszy serwer WWW ze wszystkich tego typu aplikacji spotykanych w internecie. Jest dostępny dla wszystkich znaczących systemów operacyjnych, w tym oczywiście dla Windowsa i Linuksów. Serwer Apache to dość skomplikowane oprogramowanie i posiada bardzo rozbudowane możliwości konfiguracyjne, jednak sama procedura instalacji jest dość prosta i zrozumiała.

Przykłady prezentowane w tym rozdziale są oparte na systemie Windows (również omówienie procedury instalacyjnej), ale instalacja serwera Apache w systemach Unix i Linux to zadanie równie proste. Przedstawiony opis może z powodzeniem posłużyć jako podstawowy przewodnik po procedurze instalacji.

Należy pamiętać, że aplikacja Apache Server for Windows jest wspierana dla systemów Windows 2000, Windows 2003 i Windows XP (z zainstalowaną poprawką Service Pack 2), ale nie będzie działać w systemach Windows 95, ME czy 98.

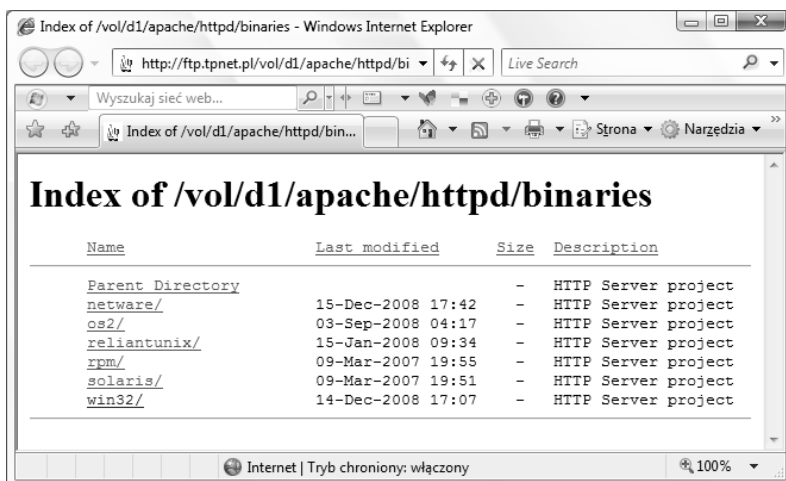
Pobieranie instalatora serwera Apache

Pakiety instalacyjne serwera Apache (bieżącą wersją jest 2.2) można znaleźć na stronie <http://httpd.apache.org/download.cgi>. W przypadku instalowania serwera pod Windowsem warto zapoznać się z poradami umieszczonymi pod adresem <http://www.hightechimpact.com/Apache/httpd/binaries/win32/README.html>.

Pliki do pobrania nie znajdują się jednak na wymienionej wyżej stronie WWW, a na serwerach lustrzanych, na przykład <http://ftp.tpnet.pl/vol/d1/apache/httpd/binaries/>. Pod tym adresem zobaczymy listing katalogów zawierających pliki instalacyjne przeznaczone dla różnych systemów operacyjnych. Przykład takiego listingu przedstawia rysunek 5.1.

Rysunek 5.1.

Zawartość serwera lustrzanego z plikami instalacyjnymi serwera Apache



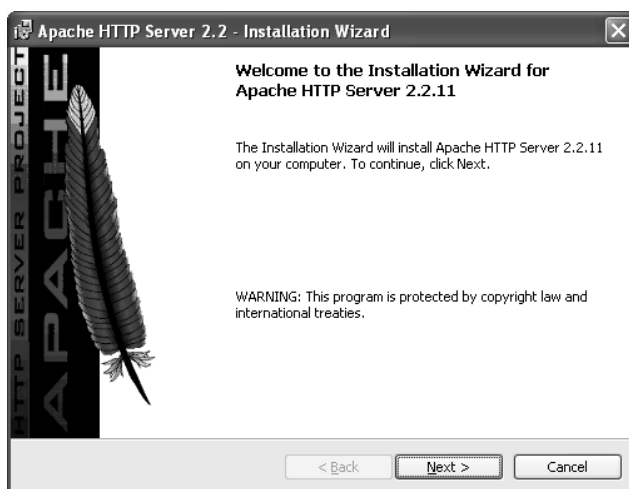
Klikamy katalog pod nazwą *win32*, pojawi się jego zawartość w postaci listingu plików dostępnych do pobrania. W chwili pisania tej książki aktualną wersją była 2.2.11, założmy zatem, że pobieramy plik pod nazwą *apache_2.2.11-win32-x86-no_ssl.msi*. To jest instalator serwera Apache w wersji 2.2.11 bez obsługi protokołu SSL (nie będzie nam potrzebny do realizacji projektu opisywanego w tym rozdziale).

Należy pamiętać, że opisane tu instrukcje dotyczą serwera Apache w wersji dla systemu operacyjnego Windows jedynie w zakresie niezbędnym do realizacji opisywanego projektu. Apache to bardzo rozbudowana aplikacja o ogromnych możliwościach konfiguracyjnych, a pełna instrukcja procedury konfiguracyjnej tego serwera znacząco wybiega poza tematykę niniejszej książki.

Po pobraniu pliku instalatora należy dwukrotnie kliknąć jego ikonę w Eksploratorze Windows, co rozpocznie procedurę instalacyjną. Na pierwszym ekranie zobaczymy komunikat powitalny instalatora, co zostało przedstawione na rysunku 5.2.

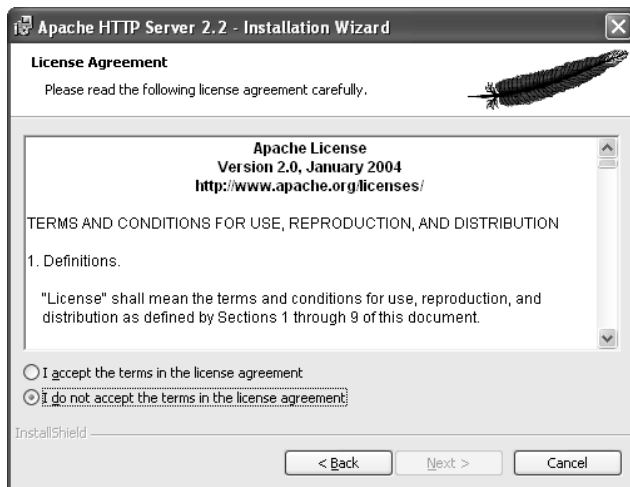
Rysunek 5.2.

Ekran powitalny instalatora serwera Apache



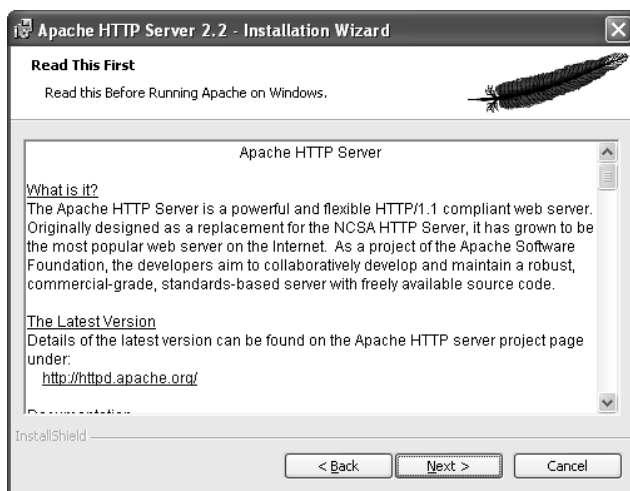
Po kliknięciu przycisku *Next* pojawi się strona z umową licencyjną, która została przedstawiona na rysunku 5.3.

Rysunek 5.3.
*Umowa licencyjna
serwera Apache*



Aby kontynuować instalację, należy zaznaczyć opcję *I accept...* i kliknąć przycisk *Next*. Następnym ekranem instalatora są informacje o aplikacji (*Read This First*), przedstawione na rysunku 5.4.

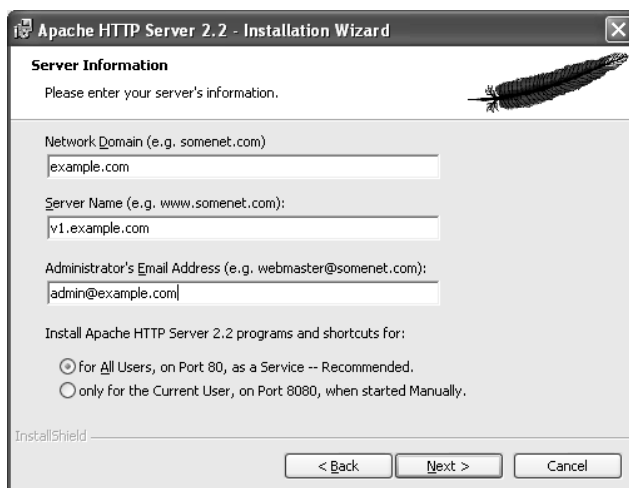
Rysunek 5.4.
*Informacje
o aplikacji Apache*



Po kliknięciu przycisku *Next* pojawi się ekran konfiguracyjny o zawartości przedstawionej na rysunku 5.5.

Jeśli serwer Apache jest konfigurowany przede wszystkim z myślą o przetestowaniu aplikacji omawianej w tym rozdziale, w polu *Network Domain* można wpisać dowolną nazwę domeny (na przykład *example.com*). Standardowo pola na tej stronie kreatora powinny zostać wypełnione automatycznie na podstawie informacji uzyskanych od systemu operacyjnego, dlatego można pozostawić je bez modyfikacji. Po kliknięciu przycisku *Next* pojawi się strona z wyborem typu instalacji, która została przedstawiona na rysunku 5.6.

Rysunek 5.5.
*Konfiguracja
instalowanego
serwera Apache*



Rysunek 5.6.
*Wybór typu instalacji
serwera Apache*



Typ instalacji pozostawiamy na wartości domyślnej (*Typical*) i klikamy *Next*. Następną stroną kreatora instalacji służy do zdefiniowania docelowego katalogu instalacji, co zostało przedstawione na rysunku 5.7.

Po kliknięciu przycisku *Next* pojawi się strona potwierdzenia instalacji, na której należy kliknąć przycisk *Install*. Po kilku minutach zostanie wyświetlone okno z informacją o zakończeniu instalacji, przedstawione na rysunku 5.8.

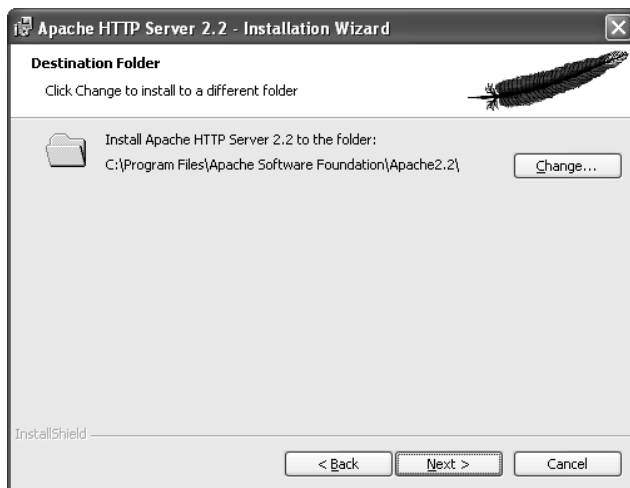
Po kliknięciu przycisku *Finish* okno instalatora zostanie zamknięte.

Instalacja modułu `mod_python`

Po zainstalowaniu serwera Apache potrzebujemy uzupełnić go o obsługę języka Python, w tym celu zainstalujemy moduł `mod_python`.

Rysunek 5.7.

Wybór docelowego katalogu instalacji serwera Apache

**Rysunek 5.8.**

Informacja o pomyślnym zakończeniu instalacji serwera Apache

**Czym jest mod_python?**

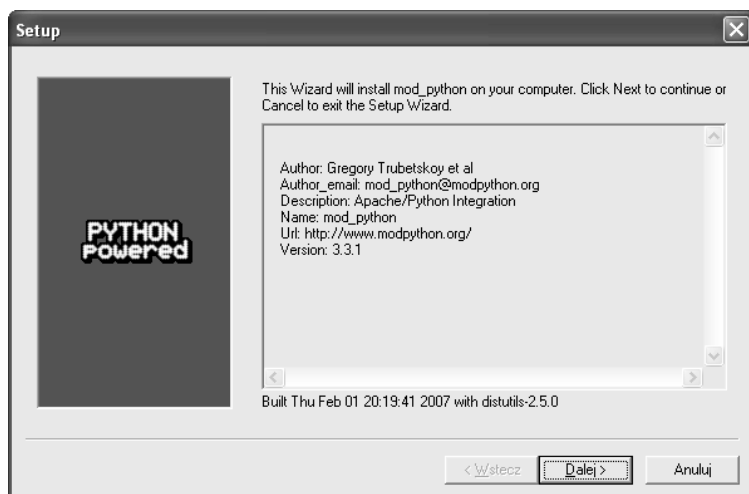
Moduł `mod_python` to rozszerzenie serwera Apache polegające na osadzeniu interpretera języka Python w samym serwerze Apache. Dzięki temu uzyskujemy możliwość integracji Pythona z mechanizmem serwowania treści przez serwer WWW. Z użyciem modułu `mod_python` mamy możliwość tworzenia aplikacji WWW działających wydajniej niż w przypadku zastosowania mechanizmu Common Gateway Interface (CGI). Dodatkowo dzięki `mod_python` aplikacje WWW mają dostęp do zaawansowanych mechanizmów serwera Apache, jak również do innych niezbędnych technologii, jak bazy danych czy obróbka formatu XML. A to wszystko za pomocą naszego ulubionego Pythona.

Pobieranie instalatora mod_python

Instalator modułu `mod_python` można znaleźć na stronie <http://httpd.apache.org/modules/python-download.cgi> (w okresie powstawania tej książki aktualna wersja nosiła numer 3.3.1). Na powyższej stronie wystarczy kliknąć odnośnik *Win32 Binaries*.

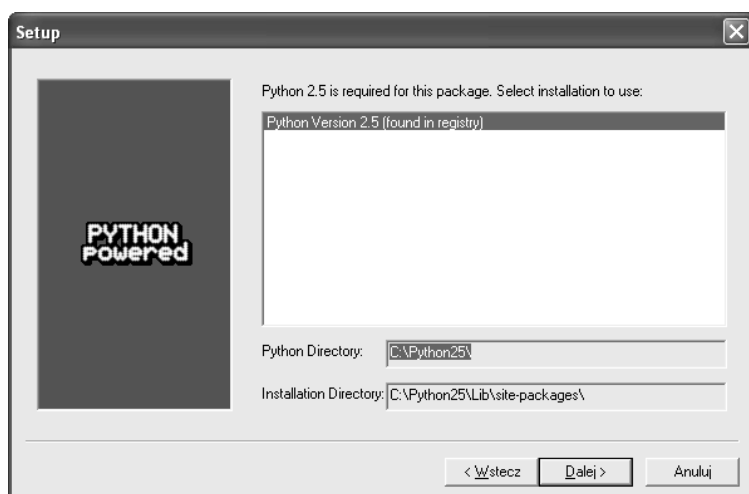
Po pobraniu pliku należy dwukrotnie kliknąć jego nazwę, co uruchomi instalator. Rysunek 5.9 przedstawia pierwszy ekran instalatora.

Rysunek 5.9.
Ekran powitalny
instalatora modułu
`mod_python`



Po kliknięciu przycisku *Dalej* zostanie wyświetlony ekran przedstawiony na rysunku 5.10, na którym należy wskazać lokalizację instalacji Pythona.

Rysunek 5.10.
Wybór katalogu
z instalacją
środowiska języka
Python



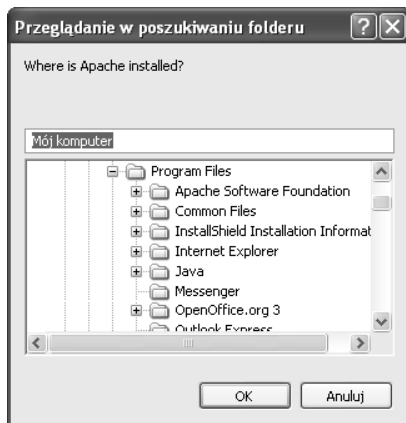
W przypadku standardowej instalacji Pythona na tym ekranie będzie zaznaczona właściwa lokalizacja, wystarczy zatem kliknąć przycisk *Dalej*.

Następny ekran będzie ekranem potwierdzenia, gdzie również należy kliknąć *Dalej*, po czym rozpocznie się instalacja.

W trakcie instalacji pojawi się dodatkowa prośba o wskazanie lokalizacji serwera Apache, co przedstawia rysunek 5.11.

Rysunek 5.11.

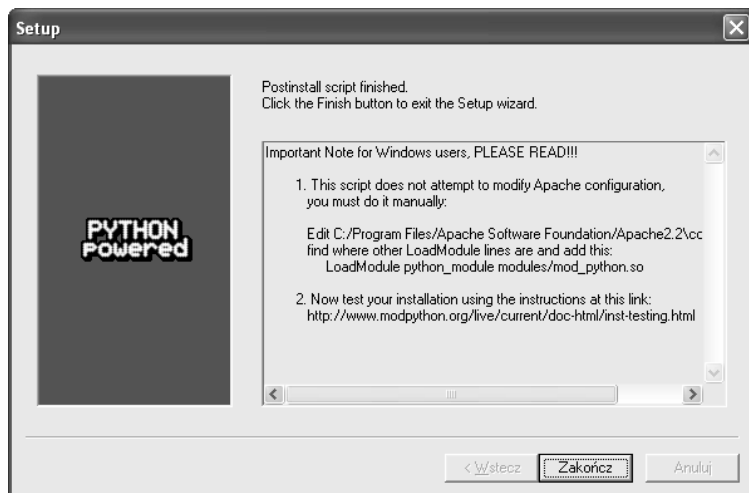
Wskazanie katalogu z instalacją serwera Apache



Należy wskazać lokalizację serwera Apache i kliknąć *OK*. Rysunek 5.12 przedstawia ostatni ekran kreatora instalacji modułu `mod_python`.

Rysunek 5.12.

Ważne informacje konfiguracyjne dotyczące modułu `mod_python`



To ostatnie okno należy pozostawić na chwilę otwarte, ponieważ zawiera istotne informacje konfiguracyjne. Wpis konfiguracyjny serwera Apache zawarty w *punkcie 1.* na tym ekranie należy skopiować do schowka, po czym można kliknąć przycisk *Zakończ*.

Konfiguracja użycia modułu `mod_python`

Konfiguracja użycia modułu `mod_python` w serwerze Apache

Aktywacja wykorzystania modułu `mod_python` w serwerze Apache składa się z dwóch etapów:

1. W pliku *<katalog serwera Apache>\conf\httpd.conf* należy odszukać sekcję, w której ładowane są moduły. Można ją rozpoznać po sekwencji wierszy rozpoczynających się słowem kluczowym `LoadModule`.

Do tej sekcji należy wkleić tekst skopiowany z ekranu instalatora:

```
LoadModule python_module modules/mod_python.so
```

2. W tym samym pliku należy dodać następujące wpisy (w dowolnym miejscu pliku):

```
AddHandler mod_python .py
PythonHandler mod_python.publisher
PythonDebug On
```

Skopiowanie plików aplikacji we właściwe miejsce

Konfiguracja naszej aplikacji składa się z następujących działań:

1. Utworzenie katalogu *<katalog serwera Apache>\htdocs\test*.
2. Utworzenie katalogu *c:\logs*.
3. Skopiowanie plików *form.py* oraz *form.html* do katalogu *htdocs\test*.
4. Skopiowanie pliku *feedbacklog.csv* do katalogu *c:\logs*.

Uruchomienie programu

Choć „pod maską” program wykorzystuje dość skomplikowane mechanizmy, jednak jego interfejs użytkownika jest bardzo prosty. Użytkownicy wpisują swoje komentarze w formularzu wyświetlanym w przeglądarce WWW, a administrator ma dostęp do pliku CSV, w którym zapisywane są wszystkie wpisy.

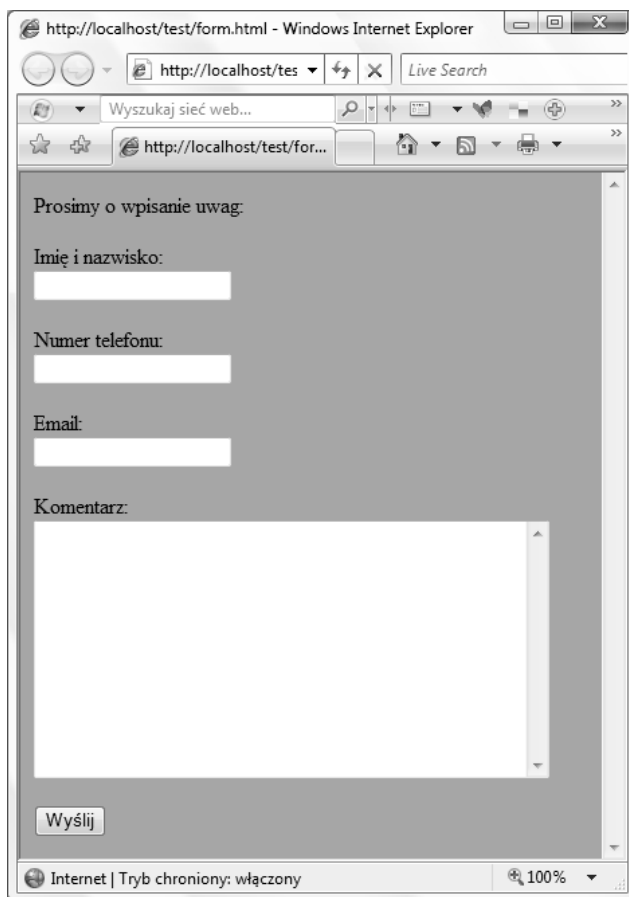
Wprowadzanie komentarzy na stronie WWW

Jeśli serwer Apache jest uruchomiony na tym samym komputerze, z którego będzie testowana aplikacja, należy w przeglądarce WWW wpisać adres *http://localhost/test/form.html*. Formularz służący do gromadzenia opinii użytkowników został przedstawiony na rysunku 5.13.

Aplikacja jest oparta na założeniu, że na maszynie, na której działa serwer Apache, jest również uruchomiony serwer SMTP. Jeśli jest inaczej, przy próbie wysyłki formularza pojawi się komunikat o błędzie. Nie przejmujemy się tym w tej chwili, niedługo zmodyfikujemy skrypt tak, żeby wykorzystywał rzeczywiste serwery SMTP.

Rysunek 5.13.

Formularz do
gromadzenia opinii
użytkowników



Po poprawnej wysyłce formularza zostanie wyświetlony następujący ekran wyniku:

Szanowny(a) Jan Kowalski,
Dziękujemy za uwagi, postaramy się skontaktować w najbliższym czasie.

Obsługa problemów

W przypadku pominięcia dowolnego z pól aplikacja wyświetli następujący komunikat:

Brak wymaganego parametru, proszę wrócić do formularza i poprawić błąd

Przeglądanie pliku dziennika

Na serwerze sprawdzamy zawartość katalogu `c:\logs`.

Znajdujący się tam plik `feedbacklog.csv` można otworzyć w dowolnym programie arkusza kalkulacyjnego. Rysunek 5.14 przedstawia przykładową zawartość tego pliku (została ona nieco sformatowana, aby poprawić czytelność).

	A	B	C	D	E	F
1	Dziennik informacji zwrotnych użytkowników WWW					
2	Data	Nazwisko	Telefon	Email	Komentarz	
3	11.01.2009	Jim Knowlton	(111)111-111	jim@example.com	To jest komentarz. Staram się wpisać kilka pełnych zdań, żeby sprawdzić wynik działania aplikacji.	
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

Rysunek 5.14. Plik *feedbacklog.csv* otwarty w arkuszu kalkulacyjnym

Projekt

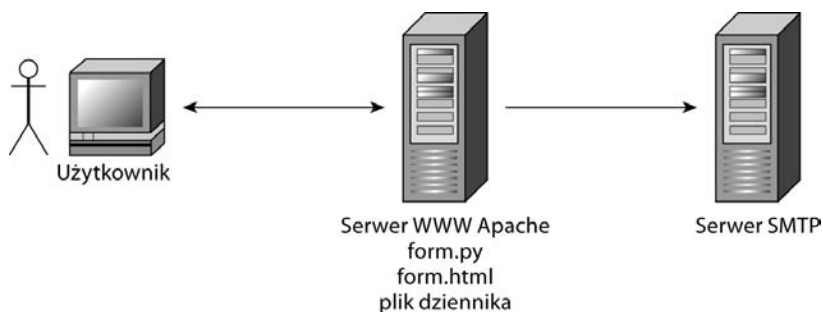
Aplikacja posiada dwa interfejsy użytkownika: do wprowadzania danych służy interfejs WWW, a do przeglądania danych przez administratorów służy plik CSV.

Elementy aplikacji

Podstawowa architektura programu jest dość prosta i została przedstawiona na rysunku 5.15.

Rysunek 5.15.

Diagram działania aplikacji



Przepływ logiki aplikacji jest następujący:

- Użytkownik łączy się z serwerem WWW, wywołując dokument *form.html* (znajdujący się w katalogu *test*). Ten dokument prezentuje formularz HTML w oknie przeglądarki WWW.

- Po kliknięciu przycisku *Wyślij* dane wypełnionego formularza są przekazywane do skryptu *form.py*.
- Skrypt *form.py* tworzy wiadomość e-mail i wysyła ją do webmastera, wykorzystując do tego serwer SMTP i parametry połączenia zdefiniowane w kodzie skryptu.
- Dodatkowo skrypt *form.py* zapisuje plik CSV, który można otworzyć w programie arkusza kalkulacyjnego.

Moduły

Program składa się tylko z jednego modułu — *form.py*.

form.py

Plik *form.py* definiuje moduł wywoływany z pliku *form.html*. Zawiera dwie funkcje, których zestawienie przedstawia tabela 5.1.

Tabela 5.1. Funkcje zdefiniowane w module *form.py*

Funkcja	Typ zwracanej wartości	Opis
<code>email(req, name, phone, email, comment)</code>	string	Pobiera dane wprowadzone w formularzu i na ich podstawie buduje wiadomość e-mail, a następnie wysyła do przeglądarki WWW informację o pomyślnej wysyłce e-maila.
<code>WriteCSVLog(name, phone, email, comment)</code>	none	Zapisuje w pliku CSV dane użytkownika oraz treść komentarza wraz z aktualną datą.

Analiza kodu

Aplikacja analizowana przez nas w niniejszym rozdziale może wydać się prosta, ponieważ mamy do czynienia tylko z jednym plikiem kodu Pythona — ale mamy tu także do czynienia z dużą liczbą rozproszonych elementów.

W celu zaoszczędzenia miejsca pominąłem kod nagłówka modułu, ale zalecam jego stosowanie w każdym tworzonym pliku. Współpracownicy z pewnością to docenią.

form.html

Nie jest to plik w Pythonie, ale to bardzo istotny plik naszego projektu, zatem przeanalizujemy również i jego zawartość:

```

<HTML>
<BODY LANG="pl-PL" BGCOLOR="#ccffff">
<P>Prosimy o wpisanie uwag:
</P>
<FORM ACTION="form.py/email" METHOD="POST">
<P>Imię i nazwisko:<BR><INPUT TYPE=TEXT NAME="name"><BR><BR><BR>
</P>
<P>Numer telefonu:<BR><INPUT TYPE=TEXT NAME="phone"><BR><BR><BR>
</P>
<P>Email:<BR><INPUT TYPE=TEXT NAME="email"></P>
<P><BR>Komentarz:</P>
<P>
<TEXTAREA NAME="comment" ROWS=10 COLS=45 STYLE="width: 4in; height: 2in"></TEXTAREA>
<BR><BR><BR>
</P>
<P><INPUT TYPE=SUBMIT VALUE="Wyślij">
</P>
</FORM>
</BODY>
</HTML>

```

Znacznik `<BODY>` definiuje treść dokumentu HTML i pozwala zdefiniować kolor tła:

```
<BODY LANG="pl-PL" BGCOLOR="#ccffff">
```

Następnie zdefiniowany jest tekst zachęcający do wprowadzania uwag:

```
<P>Prosimy o wpisanie uwag:
</P>
```

Kolejny wiersz to inicjalizacja formularza oraz wskazanie handlera, czyli funkcji w Pythonie, która obsługuje dane formularza:

```
<FORM ACTION="form.py/email" METHOD="POST">
```

W następnej części pliku definiowane są pola dla nazwiska, numeru telefonu, adresu e-mail i komentarza:

```

<P>Imię i nazwisko:<BR><INPUT TYPE=TEXT NAME="name"><BR><BR><BR>
</P>
<P>Numer telefonu:<BR><INPUT TYPE=TEXT NAME="phone"><BR><BR><BR>
</P>
<P>Email:<BR><INPUT TYPE=TEXT NAME="email"></P>
<P><BR>Komentarz:</P>
<P>
<TEXTAREA NAME="comment" ROWS=10 COLS=45 STYLE="width: 4in; height: 2in"></TEXTAREA>
<BR><BR><BR>
</P>

```

Następnie znajduje się kod HTML definiujący przycisk *Wyślij*:

```
<P><INPUT TYPE=SUBMIT VALUE="Wyślij">
</P>
```

Plik kończy się znacznikami zamykającymi wykorzystywane wcześniej elementy struktury dokumentu:

```
</FORM>
</BODY>
</HTML>
```

form.py

Plik *form.py* definiuje logikę programu:

```
import smtplib, csv, datetime, sys

WEBMASTER = "jknowlton525@gmail.com"
SMTP_SERVER = "localhost"

def writeCSVLog(name, phone, email, comment):
    python_exec = sys.executable
    if python_exec.find("exe") != -1:
        dir_root = "c:\\logs\\"
    else:
        dir_root = "//usr//local//logs//"
    today = datetime.datetime.now().strftime("%m/%d/%Y")
    row = [today, name, phone, email, comment]
    try:
        writer = csv.writer(open(dir_root + "feedbacklog.csv", "a"))
        writer.writerow(row)
    except:
        print "Wystąpił problem zapisu w pliku dziennika!"
        sys.exit

def email(req, name, phone, email, comment):

    # weryfikacja kompletności parametrów
    if not (name and phone and email and comment):
        return "Brak wymaganego parametru, \
        proszę wrócić do formularza i poprawić błąd"

    # treść e-maila
    msg = ""
    From: %s
    Subject: informacje od użytkownika
    To: %s

    Oto mój komentarz:

    %s

    Dziękuję,

    %s
    %s

    """ % (email, WEBMASTER, comment, name, phone)
```

```

# wysyłka e-maila
try:
    conn = smtplib.SMTP(SMTP_SERVER)
    conn.sendmail(email, [WEBMASTER], msg)
    conn.quit()
except:
    print "Wystąpił problem z wysyłką wiadomości e-mail!"
    sys.exit

# potwierdzenie dla użytkownika
s = ""
<html>
<BODY BGCOLOR="#ccffff" DIR="LTR">
Szanowny(a) %s,<br>

Dziękujemy za uwagi, postaramy się
skontaktować w najbliższym czasie.
</BODY>
</html>"" % name

writeCSVLog(name, phone, email, comment)
return s

```

W pierwszej kolejności importowane są niezbędne moduły:

```
import smtplib, csv, datetime
```

Następnie inicjalizowane są zmienne definiujące adres e-mail webmastera oraz adres serwera SMTP:

```
WEBMASTER = "jknowlton525@gmail.com"
SMTP_SERVER = "localhost"
```

Dla uproszczenia przyjęliśmy, że serwer SMTP znajduje się na tej samej fizycznej maszynie co serwer WWW wykonujący kod. W przeciwnym razie wystarczy zmienić wartość zmiennej SMTP_SERVER na nazwę hosta lub adres IP serwera SMTP.

W kolejności logiki aplikacji warto przeanalizować funkcję `email()`, która jest zdefiniowana w dalszej części pliku.

email(req, name, phone, email, comment)

Funkcja `email()` jest główną funkcją programu. Przyjmuje parametry wprost ze strony WWW (formularza HTML) i wysyła je w odpowiednio sformatowanej wiadomości e-mail, a na końcu wywołuje funkcję zapisującą dane w pliku dziennika. Oto cała zawartość tej funkcji:

```

def email(req, name, phone, email, comment):

    # weryfikacja kompletności parametrów
    if not (name and phone and email and comment):
        return "Brak wymaganego parametru, \
        proszę wrócić do formularza i poprawić błąd"

```

```

    # treść e-maila
    msg = """\
From: %s
Subject: informacje od użytkownika
To: %s

Oto mój komentarz:

%s

Dziękuję.

%s
%s

"" % (email, WEBMASTER, comment, name, phone)

    # wysyłka e-maila
    try:
        conn = smtplib.SMTP(SMTP_SERVER)
        conn.sendmail(email, [WEBMASTER], msg)
        conn.quit()
    except:
        print "Wystąpił problem z wysyłką wiadomości e-mail!"
        sys.exit

    # potwierdzenie dla użytkownika
    s = """\
<html>
<BODY BGCOLOR="#ccffff" DIR="LTR">
Szanowny(a) %s,<br>

Dziękujemy za uwagi, postaramy się
skontaktować w najbliższym czasie.
</BODY>
</html>"" % name

    writeCSVLog(name, phone, email, comment)
    return s

```

Parametry funkcji odpowiadają polom formularza zdefiniowanego w pliku HTML:

```
def email(req, name, phone, email, comment):
```

W pierwszej kolejności funkcja przeprowadza kontrolę błędów, która w naszym przypadku polega na weryfikacji kompletności danych przesłanych w formularzu:

```

    # weryfikacja kompletności parametrów
    if not (name and phone and email and comment):
        return "Brak wymaganego parametru, \
        proszę wrócić do formularza i poprawić błąd"

```

Kolejny fragment kodu konstruuje wiadomość e-mail, wykorzystując informacje wprowadzone przez użytkownika w formularzu HTML:

```

    # treść e-maila
    msg = """\
From: %s
Subject: informacje od użytkownika
To: %s

Oto mój komentarz:

%s

Dziękuję.

%s
%s

"" % (email, WEBMASTER, comment, name, phone)

```

Kolejnym etapem jest wysyłka przygotowanego e-maila:

```

# wysyłka e-maila
try:
    conn = smtplib.SMTP(SMTP_SERVER)
    conn.sendmail(email, [WEBMASTER], msg)
    conn.quit()
except:
    print "Wystąpił problem z wysyłką wiadomości e-mail!"
    sys.exit

```

Następnie program wysyła do przeglądarki WWW komunikat o tym, że wiadomość e-mail została wysłana prawidłowo:

```

# potwierdzenie dla użytkownika
s = """\
<html>
<BODY BGCOLOR="#ccffff" DIR="LTR">
Szanowny(a) %s,<br>

Dziękujemy za uwagi, postaramy się
skontaktować w najbliższym czasie.
</BODY>
</html>"" % name

```

Na końcu wywoływana jest funkcja `writeCSVLog()`, która zapisuje odpowiednie informacje w pliku dziennika:

```

writeCSVLog(name, phone, email, comment)
return s

```

writeCSVLog(name, phone, email, comment)

Funkcja `writeCSVLog()`, jak sugeruje nazwa, zapisuje w pliku dziennika odpowiednie informacje o wysłanym komentarzu użytkownika:

```
def writeCSVLog(name, phone, email, comment):
    python_exec = sys.executable
    if python_exec.find("exe") != -1:
        dir_root = "c:\\logs\\"
    else:
        dir_root = "//usr//local//logs/"
    today = datetime.datetime.now().strftime("%m/%d/%Y")
    row = [today, name, phone, email, comment]
    try:
        writer = csv.writer(open(dir_root + "feedbacklog.csv", "a"))
        writer.writerow(row)
```

Parametry wywołania funkcji odpowiadają informacjom, które mają być zapisane w dzienniku:

```
def writeCSVLog(name, phone, email, comment):
```

Następnie kod odczytuje bieżącą datę i formatuje ją w sposób czytelny dla człowieka, data ta będzie użyta we wpisie w dzienniku:

```
today = datetime.datetime.now().strftime("%m/%d/%Y")
```

Następny wiersz definiuje listę z bieżącą datą i przekazanymi parametrami:

```
row = [today, name, phone, email, comment]
```

Na końcu otwierany jest plik dziennika i następuje zapis danych:

```
try:
    writer = csv.writer(open(dir_root + "feedbacklog.csv", "a"))
    writer.writerow(row)
except:
    print "Wystąpił problem zapisu w pliku dziennika!"
```

Testowanie

Istnieje wiele sposobów testowania aplikacji WWW. Oto kilka z użytecznych pomysłów:

- Testowanie danych poszczególnych pól. Należy pamiętać, że każde pole formularza jest przekazywane w parametrze wywołania funkcji w Pythonie, zatem należy się upewnić, że aplikacja zachowa się prawidłowo po wprowadzeniu przez użytkownika nieprawidłowych danych (jak cyfry w miejsce nazwiska).
- Wprowadzenie w formularzu bardzo dużych porcji danych i sprawdzenie, czy program będzie działał prawidłowo.
- Weryfikacja samego interfejsu użytkownika (strony WWW), jak maksymalizacja, zmniejszenie rozmiaru okna czy poprawność wyświetlania i działania w różnych przeglądarkach WWW.

Modyfikowanie programu

Istnieje kilka potencjalnych dróg rozbudowy naszego projektu, oto niektóre z nich:

- Można zaimplementować interfejs użytkownika dla „administratora”, za pośrednictwem którego będzie mógł pobierać z serwera plik CSV dziennika.
- Informacje dziennika można zapisywać w formacie XML lub w bazie danych, w przypadku której przeszukiwanie dużych zbiorów danych jest znacznie łatwiejsze.
- Można skonstruować mechanizm analizujący pliki dziennika i wysyłający po kilku dniach odpowiednie wiadomości zwrotne.

Podsumowanie

W tym rozdziale zbudowaliśmy formularz WWW służący do gromadzenia opinii użytkowników i przesyłający te komentarze w formie wiadomości e-mail. Nauczyliśmy się, w jaki sposób z poziomu Pythona wykorzystywać serwer pocztowy (SMTP) oraz wykorzystywać moduł `csv` do zapisu danych dziennika w formacie CSV. Oprócz tego Czytelnik miał okazję nabyć następujących umiejętności:

- instalacji modułu `mod_python` serwera Apache;
- konfiguracji modułu `mod_python` pod kątem posiadanej wersji interpretera Pythona;
- tworzenia formularza WWW i uruchamiania skryptów w Pythonie w reakcji na działania użytkownika na stronie WWW;
- wysyłania wiadomości e-mail z poziomu programu w Pythonie.