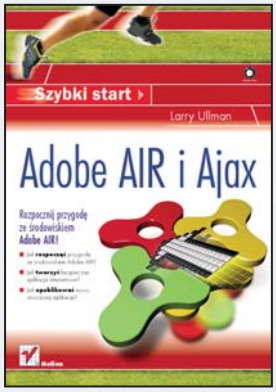
Adobe Air i Ajax. Szybki start



Autor: Larry Ullman
Tłumaczenie: Krzysztof Rychlicki-Kicior
ISBN: 978-83-246-2192-7
Tytuł oryginału: Adobe AIR (Adobe Integrated Runtime) with Ajax: Visual QuickPro Guide
Format: 170×230, stron: 400

Rozpocznij przygodę ze środowiskiem Adobe AIR!

- Jak rozpocząć przygodę ze środowiskiem Adobe AIR?
- Jak tworzyć bezpieczne aplikacje internetowe?
- Jak opublikować nowo stworzoną aplikację?

Adobe AIR to środowisko wykonawcze dostępne na wielu platformach. Pozwala ono opracowywać nowoczesne i atrakcyjne aplikacje internetowe. Do ich tworzenia możesz wykorzystać takie technologie, jak HTML/AJAX, Adobe Flex lub Adobe Flash. Pomimo krótkiej obecności na rynku – premiera miała miejsce w lutym 2008 roku – środowisko to już zdobyło sobie rzeszę fanów. O jego jakości świadczy także liczba aplikacji, które powstały w oparciu o Adobe AIR.

Książka, którą trzymasz w ręku, to świetny podręcznik, cierpliwie wprowadzający w tajniki Adobe AIR. Liczne przykłady i opisy „krok po kroku" sprawiają, że AIR błyskawicznie i bezproblemowo odkrywa swe kolejne tajemnice. Dzięki lekturze dowiesz się, jak zainstalować środowisko deweloperskie oraz rozpocząć przygodę z Adobe AIR. W kolejnych rozdziałach zaznajomisz się ze sposobami tworzenia okien, menu oraz importu i eksportu danych. Ponadto nauczysz się wykonywać operacje na plikach i katalogach oraz sprawdzisz, jak połączyć się z bazą danych z poziomu Twojej aplikacji. „Adobe Air i Ajax. Szybki start" to wyczerpujący przewodnik, który sprawi, że zaczniesz swobodnie wykorzystywać możliwości środowiska Adobe AIR.

- Instalacja bibliotek uruchomieniowych oraz środowiska deweloperskiego
- Przegląd narzędzi programistycznych dla AIR
- Debugowanie aplikacji
- Tworzenie okien, menu oraz ich obsługa
- Importowanie oraz eksportowanie danych
- Operacje na plikach i katalogach
- Łączenie z bazą danych oraz przetwarzanie danych w niej zapisanych
- Wykorzystanie operacji sieciowych
- Zapewnienie bezpieczeństwa aplikacjom korzystającym z AIR
- Przygotowanie i publikacja aplikacji

Sprawdź i wykorzystaj możliwości Adobe AIR!

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

Spis treści

Tworzenie aplikacji 2

Gdy przystępujesz do tworzenia aplikacji Adobe AIR, masz do wyboru różne technologie i narzędzia programistyczne. Do technologii należą Ajax (HTML i JavaScript), Adobe Flex i Adobe Flash. W niniejszej książce będę omawiał wyłącznie aplikacje stworzone z wykorzystaniem Ajaksa. W przypadku narzędzi programistycznych nie ma prawie żadnych ograniczeń. W tym rozdziale zaprezentuję proces tworzenia aplikacji przy użyciu edytora tekstowego i AIR SDK (z ang. *Software Development Kit* — zestaw narzędzi programistycznych). W następnym rozdziale pokażę Ci, jak wykorzystywać bardziej rozbudowane narzędzia, takie jak Dreamweaver CS3 czy Aptana Studio Integrated Development Environments (IDE).

Tworzenie aplikacji AIR należy rozpocząć od utworzenia folderu projektu. Następnie dodasz do niego dwa pliki: HTML i XML. Na końcu przetestujesz i skompilujesz aplikację przy użyciu narzędzi z pakietu AIR SDK. Najpierw jednak zajmiemy się instalacją i konfiguracją tego pakietu — jest to, podobnie jak w przypadku samej biblioteki AIR, proces jednorazowy.

Instalacja SDK

Adobe AIR SDK to pakiet niezależny od biblioteki AIR, którą wykorzystujemy do uruchamiania aplikacji. W skład SDK wchodzą:

- dwa narzędzia uruchamiane z wiersza poleceń, umożliwiające testowanie i pakowanie aplikacji;
- frameworki (biblioteki udostępniające użyteczne funkcje);
- przykłady (na przykład ikony aplikacji);
- szablon pliku XML.

Do prawidłowego działania SDK niezbędny jest Java Runtime Environment (JRE) lub Java Development Kit (JDK). Obydwa zestawy aplikacji możesz pobrać za darmo ze strony *http://java.sun.com* (jako fragment pakietu Java Standard Edition, Java SE). Po zakończeniu instalacji (w przypadku systemu Mac OS X istnieje duże prawdopodobieństwo, że JRE jest już zainstalowane; w systemie Windows z reguły trzeba je dopiero zainstalować) możesz wykonać poniższe kroki.

Aby zainstalować SDK:

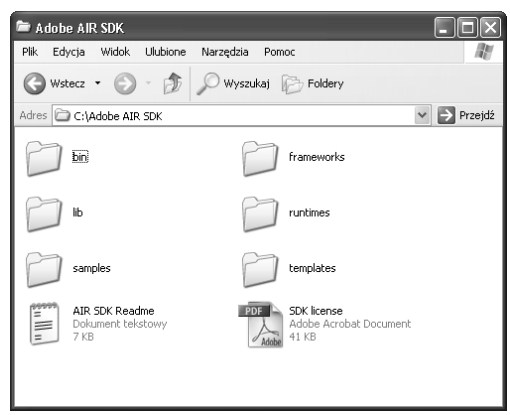
1. Pobierz SDK dla Twojego systemu operacyjnego ze strony Adobe (rysunek 2.1) — *http://www.adobe.com/products/air/tools/sdk/*. W momencie pisania tej książki SDK jest dostępny zarówno dla systemu Windows, jak i Mac OS X.

Rysunek 2.1. *Pobierz SDK odpowiedni dla Twojego systemu*

2. Otwórz i rozpakuj pobrany plik (rysunek 2.2). Użytkownicy systemu Windows muszą rozpakować archiwum ZIP, podczas gdy posiadacze systemu Mac OS X muszą zamontować plik *.dmg* (obraz dysku).
3. Skopiuj zawartość pobranego pliku do innego katalogu na Twoim komputerze. Możesz wybrać dowolny katalog — na przykład *Pulpit* lub katalog wewnątrz Twojego katalogu domowego. Niezależnie od wybranej ścieżki należy ją zapamiętać, by można było wykonać następne kroki.
4. Zaktualizuj ścieżkę systemową, tak aby wskazywała na podkatalog *bin* Twojego SDK. Ten krok został szerzej omówiony w dwóch następnych podrozdziałach.

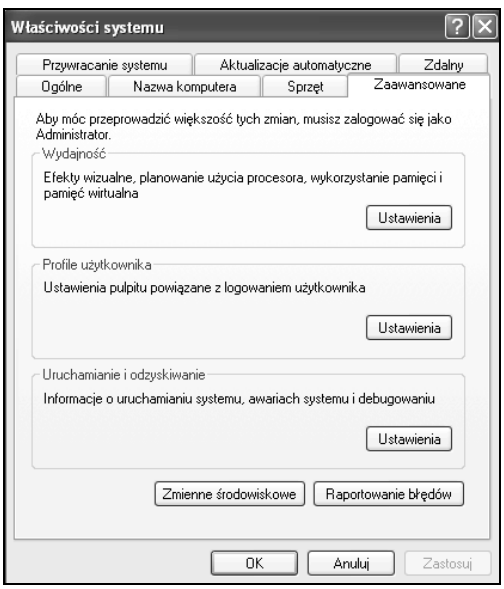
Wskazówki

- JRE i JDK są wymagane tylko w przypadku instalacji AIR SDK. Końcowi użytkownicy Twojej aplikacji będą musieli zainstalować tylko bibliotekę AIR (zob. rozdział 1.).
- Poza samym SDK możesz także pobrać stosowną dokumentację, przykładowe aplikacje i kody źródłowe ze strony Adobe.



Rysunek 2.2. *Zawartość katalogu Adobe AIR SDK*

Rysunek 2.3. *Uzyskiwanie dostępu do okna Właściwości systemu*



Rysunek 2.4. *Ścieżkę systemową można edytować w oknie Zmienne środowiskowe wywoływanym z zakładki Zaawansowane*

Aktualizowanie ścieżki w systemie Windows

Dwa narzędzia, które otrzymujemy razem z SDK — AIR Development Tool (*adt*) i AIR Debug Launcher (*adl*) — są uruchamiane z poziomu konsoli. Oznacza to, że do ich uruchomienia w systemie Windows musisz skorzystać z wiersza poleceń, a nie graficznego interfejsu. Składnia konsoli nie jest skomplikowana, niemniej występuje pewien kruczek — oba programy (*adt* i *adl*) muszą być „rozpoznawane" przez Twój komputer. W tym celu musisz dodać podkatalog *bin* z katalogu zawierającego SDK (rysunek 2.2) do ścieżki systemowej.

Ścieżka stanowi zwykle listę katalogów, w których system ma szukać wywoływanych programów. Na co dzień nie korzystasz ze ścieżki, ponieważ nie masz zbyt często do czynienia z wierszem poleceń. Wystarczy zrealizować poniższe kroki, a systemowa ścieżka nie będzie miała przed Tobą tajemnic.

Aby zmodyfikować ścieżkę systemową:

1. Zamknij wszystkie otwarte okna wiersza poleceń.

   Zmiana ścieżki, którą zamierzasz wykonać, odniesie zamierzony efekt w oknach, które zostaną otwarte po wykonaniu zmiany. Aby uniknąć niepotrzebnego zamieszania, zamknij wszystkie okna konsoli, zanim przystąpisz do właściwej zmiany ścieżki.

2. Uruchom okno *Właściwości systemu*, klikając prawym przyciskiem ikonę *Mój komputer* i wybierając opcję *Właściwości* (rysunek 2.3).

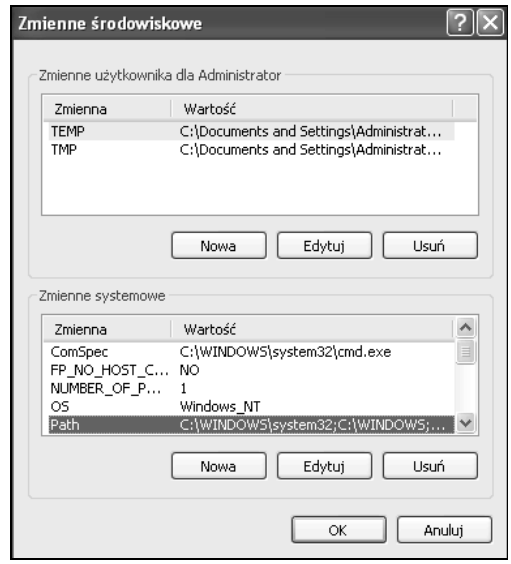
3. W oknie *Właściwości systemu* wybierz zakładkę *Zaawansowane* (rysunek 2.4).

4. Kliknij przycisk *Zmienne środowiskowe*. Jest on widoczny na dole rysunku 2.4.

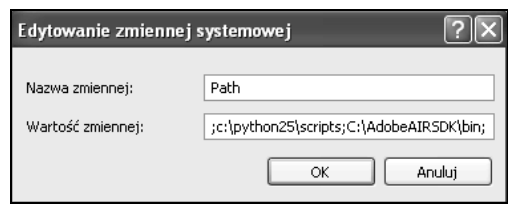
5. W oknie *Zmienne środowiskowe* kliknij element *Path* na liście *Zmienne systemowe*, aby go zaznaczyć (rysunek 2.5).
6. Kliknij przycisk *Edytuj*, aby wyświetlić okno *Edytowanie zmiennej środowiskowej*.
7. Na końcu pola tekstowego *Wartość zmiennej* umieść średnik, a następnie pełną ścieżkę do podkatalogu *bin* (rysunek 2.6).

   Musisz postępować bardzo ostrożnie; w żadnym przypadku nie usuwaj dotychczasowej zawartości tej zmiennej!

   Aby upewnić się, że wprowadzasz poprawną ścieżkę, uruchom Eksploratora Windows (rysunek 2.7) w folderze *SDK* i skopiuj adres. Upewnij się, że ścieżka, którą wklejasz, jest zakończona podkatalogiem *bin*, ponieważ jest to tak naprawdę najważniejszy element całej ścieżki.
8. Kliknij *OK* we wszystkich trzech oknach dialogowych, aby je zamknąć.



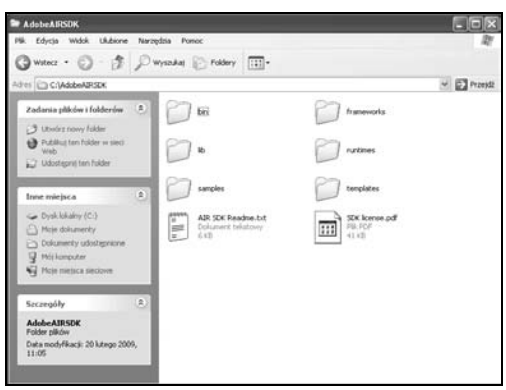
Rysunek 2.5. *Lista zmiennych systemowych znajduje się w dolnej części okna Zmienne środowiskowe*



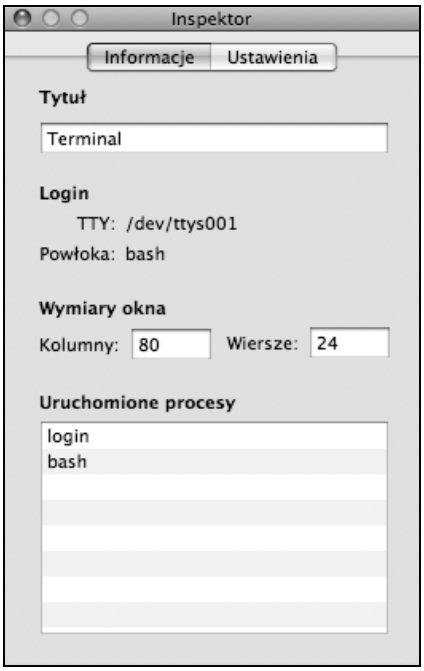
Rysunek 2.6. *Zmienna Path zawiera listę wszystkich katalogów, które są przeszukiwane przez system pod kątem programów wykonywalnych. Wszystkie katalogi są rozdzielane średnikiem*

Wskazówka

- Teoretycznie nie musisz modyfikować ścieżki, aby korzystać z narzędzi w wierszu poleceń. Jeśli jednak tego nie zrobisz, w celu uruchomienia tych programów trzeba będzie wprowadzić polecenie takie jak `C:\"Documents ↪and Settings"\"Larry Ullman"\Desktop\ ↪SDK\bin\adt` zamiast po prostu `adt`. Zmiana ścieżki jest niezwykle przydatnym skrótem.



Rysunek 2.7. *Folder SDK, znajdujący się w głównym katalogu dysku C, wraz ze ścieżką widoczną w oknie Eksploratora*



Rysunek 2.8. *Jeżeli używasz systemów uniksowych (w tym także Mac OS X), musisz wiedzieć, z jakiej powłoki korzystasz, aby poprawnie zmienić ścieżkę. Informacje o wykorzystywanej powłoce znajdziesz w programie Inspektor*

Aktualizowanie ścieżki w systemie Mac OS X

Dwa narzędzia, które otrzymujemy razem z SDK — AIR Development Tool (*adt*) i AIR Debug Launcher (*adl*) — są uruchamiane z poziomu powłoki. Oznacza to, że w celu ich uruchomienia w systemie Mac OS X musisz skorzystać z aplikacji Terminal, a nie z graficznego interfejsu. Składnia poleceń wykorzystywanych w powłoce nie jest skomplikowana, niemniej występuje pewien kruczek — oba programy (*adt* i *adl*) muszą być „rozpoznawane" przez Twój komputer. W tym celu musisz dodać podkatalog *bin* z katalogu zawierającego SDK do ścieżki systemowej.

Ścieżka stanowi zwykle listę katalogów, w których system ma szukać wywoływanych programów. Na co dzień nie korzystasz ze ścieżki, ponieważ nie masz zbyt często do czynienia z powłoką. Wystarczy zrealizować poniższe kroki, a systemowa ścieżka nie będzie miała przed Tobą tajemnic.

Aby zmodyfikować ścieżkę systemową:

1. Zamknij wszystkie otwarte okna Terminala.

   Zmiana ścieżki, którą zamierzasz wykonać, odniesie zamierzony efekt w oknach, które zostaną otwarte po wykonaniu zmiany.

2. Sprawdź wykorzystywaną przez Ciebie powłokę, korzystając z opcji *Powłoka/Pokaż Inspektora*, aby wyświetlić okno *Inspektor* (rysunek 2.8).

Sposób zmiany ścieżki zależy od wykorzystywanej przez Ciebie powłoki (jeśli zainteresowały Cię powłoki same w sobie, przeszukaj internet pod kątem wyrażenia *powłoka uniksowa* lub *unix shell*). Pozycja *Powłoka* w oknie *Inspektor* określa wykorzystywaną powłokę. Do najczęściej spotykanych powłok należą (nazwa programu znajduje się w nawiasach):

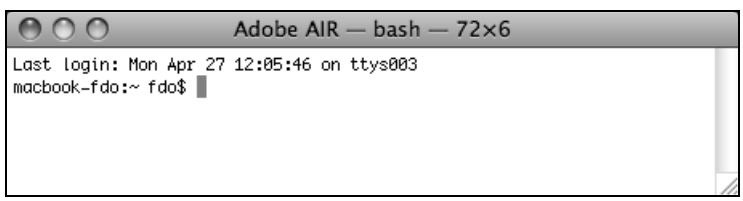
- Bourne (*sh*),
- Bourne Again Shell (*bash* — to nie ja wymyśliłem tę nazwę),
- C shell (*csh*),
- T shell lub T C shell (*tcsh*),
- Korn shell (*ksh*).

Najnowsze wersje system Mac OS X wykorzystują domyślnie powłokę bash (jak na rysunku 2.8). W poniższych instrukcjach również będę korzystał z tej powłoki. Jeśli w oknie *Inspektor* widnieje nazwa innej powłoki, musisz przeszukać internet pod kątem zmiany ścieżki w tej konkretnej powłoce.

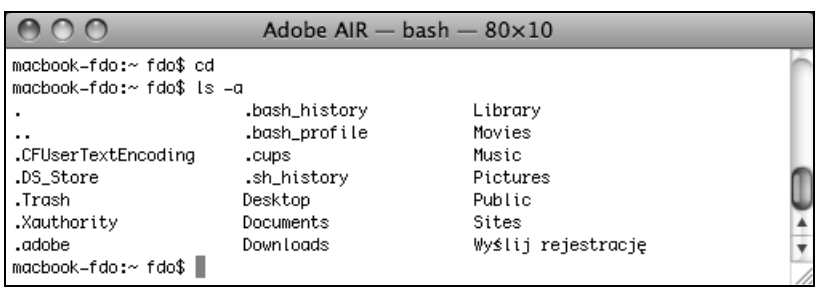
3. Otwórz okno Terminala (*Powłoka/Nowe okno* lub *Command+N*), jeśli do tej pory tego nie zrobiłeś (rysunek 2.9).
4. Przejdź do swojego katalogu domowego, wprowadzając polecenie `cd` i potwierdzając klawiszem *Return*.

   Możliwe, że ten krok nie będzie konieczny, ponieważ po otwarciu Terminala z reguły znajdujesz się w swoim katalogu domowym, jednak ostrożności nigdy za wiele. Polecenie `cd` jest wykorzystywane do zmiany aktualnego katalogu (z ang. *change directory* — zmień katalog). Wywołanie go bez żadnych dodatkowych argumentów (jak na przykład nazwa katalogu) spowoduje zmianę aktualnego katalogu na domowy.
5. Wypisz wszystkie pliki w katalogu przy użyciu polecenia `ls -a` (jak zawsze polecenie należy potwierdzić klawiszem *Return* — rysunek 2.10).

   Polecenie `ls` wyświetla zawartość aktualnego katalogu; opcja `-a` określa, że wyświetlone powinny być wszystkie pliki, także te ukryte.



Rysunek 2.9. *Okno programu Terminal*



Rysunek 2.10. *Lista wszystkich plików znajdujących się w katalogu domowym*

6. Jeśli nie widzisz na liście pliku *.bash_profile* (rysunek 2.10), utwórz go, korzystając z polecenia `touch .bash_profile`. Pliki, które rozpoczynają się kropką, są plikami specjalnymi, ukrytymi. Plik *.bash_profile* określa sposób zachowania powłoki bash. Jeśli plik nie istnieje, polecenie `touch` utworzy go.

7. Otwórz plik *.bash_profile* w dowolnym edytorze tekstu (rysunek 2.11).

   Ja korzystam z popularnego (i doskonałego) edytora BBEdit. Aby otworzyć plik, wystarczy wywołać w powłoce polecenie `bbedit .bash_profile`. Możesz także skorzystać z darmowego edytora TextWrangler firmy BareBones (*www.barebones.com*) lub z jednego z wielu popularnych edytorów działających w powłoce: vi, vim, emacs, pico itd.

8. W pliku *.bash_profile* dodaj poniższy wiersz:

   ```
   export PATH="$PATH:/ścieżka/do/SDK/AIR/bin/"
   ```

   Polecenie `export PATH` zmienia ścieżkę w powłoce bash. Jej nowa wartość będzie się składać z dotychczasowej ścieżki (`$PATH`), a także pełnej ścieżki do podkatalogu *bin* (musisz podać faktyczną ścieżkę w miejsce fragmentu `/path/to/`; zapoznaj się z pierwszą wskazówką do tego podrozdziału). Każdy katalog w ścieżce jest oddzielany dwukropkiem.

   Jeśli Twój plik *.bash_profile* zawiera już wiersz `export PATH`, dodaj na jego końcu dwukropek i pełną ścieżkę do aktualnego katalogu.

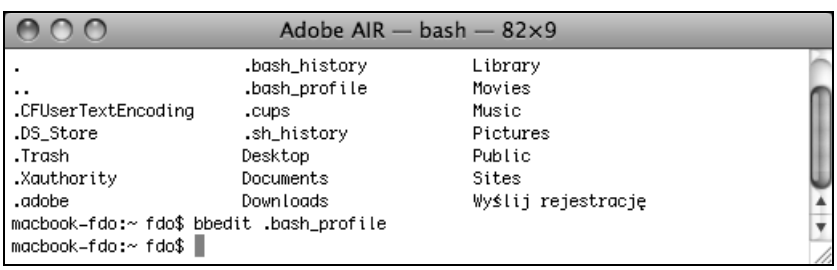
9. Zapisz i zamknij plik.

10. Zamknij okno Terminala.

Zmiana ścieżki nastąpi przy ponownym otwarciu Terminala.

Wskazówki

- W przypadku większości programów w systemie Mac OS X możesz wstawiać pełną ścieżkę katalogu do pliku, przeciągając ten katalog do pliku. Jeśli przeciągniesz podkatalog *bin* (widoczny w programie Finder) do pliku *.bash_profile* w programie BBEdit, pełna ścieżka do podkatalogu *bin* zostanie wstawiona do pliku *.bash_profile* w miejscu, w którym zwolnisz przycisk myszy.
- Teoretycznie nie musisz modyfikować ścieżki, aby korzystać z narzędzi w *Terminalu*. Jeśli jednak tego nie zrobisz, w celu uruchomienia tych programów będziesz musiał wprowadzić polecenie takie jak `/Users/larryullman/➥Desktop/AIR/SDK/bin/adt` zamiast po prostu `adt`. Zmiana ścieżki jest niezwykle przydatnym skrótem.



Rysunek 2.11. *Polecenie bbedit .bash_profile uruchamia plik .bash_profile w edytorze BBEdit*

Tworzenie struktury projektu

Chociaż tworzenie ściśle określonej struktury katalogów dla aplikacji AIR nie jest konieczne, moim zdaniem to bardzo dobry programistyczny zwyczaj. Programiści aplikacji webowych starają się organizować swoje pliki i zasoby; podobnie powinni postępować inni programiści. Nie każdy projekt musi zawierać taką samą strukturę — być może będziesz stosować inne konwencje nazewnictwa — jednak pewne podstawy, opisane poniżej, są warte naśladowania.

W poniższych krokach omówię praktyki i zwyczaje, które można zastosować w dowolnym projekcie. W przykładzie, który zrealizujemy w tym rozdziale, nie trzeba będzie korzystać z plików CSS ani JavaScript, jednak warto wiedzieć, jak z nimi postępować.

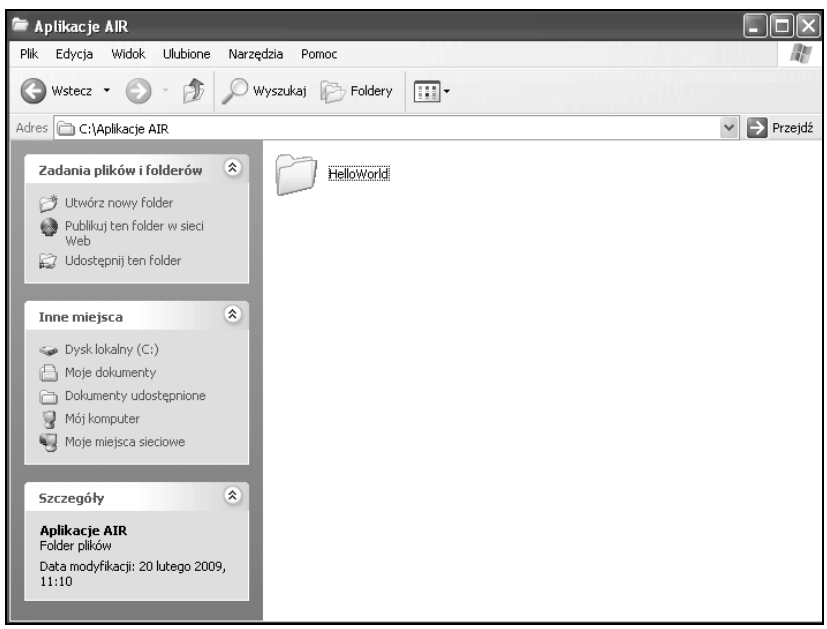
Aby utworzyć strukturę aplikacji:

1. Utwórz nowy folder w dowolnej lokalizacji na Twoim komputerze. Będziesz w nim przechowywać swoje aplikacje AIR.

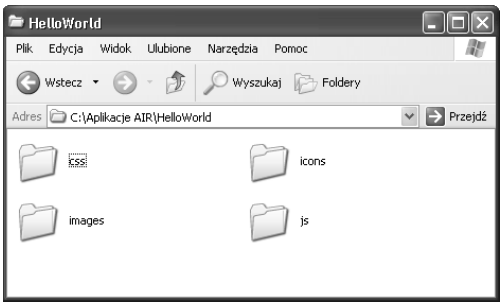
   W systemie Windows kliknij prawym przyciskiem, a następnie wybierz opcję *Nowy/Folder*, a w systemie Mac OS X wybierz opcję *Plik/Nowy katalog* lub skorzystaj z kombinacji klawiszy *Command+Shift+N*. Możesz stworzyć katalog o nazwie Aplikacje AIR wewnątrz Twojego katalogu domowego lub na Pulpicie. Ważne, aby wszystkie Twoje aplikacje były przechowywane w jednym miejscu, dzięki czemu łatwiej będzie tworzyć kolejne aplikacje.

2. Wewnątrz katalogu utworzonego w kroku 1. możesz tworzyć katalogi, w których będą znajdować się osobne aplikacje. Pamiętaj, aby wszystkie te foldery znajdowały się wewnątrz jednego folderu nadrzędnego (na przykład *Aplikacje AIR*).

3. Folder aplikacji powinien mieć tę samą nazwę, co aplikacja (rysunek 2.12). Na początek, aby tradycji stało się zadość, stworzymy aplikację *Hello, World!*. Utworzę więc folder *HelloWorld*, w którym znajdą się wszystkie pliki tej aplikacji.



Rysunek 2.12. *Folder HelloWorld będzie zawierał pliki pierwszej przykładowej aplikacji*



Rysunek 2.13. *Struktura katalogów prostej aplikacji*

4. Wewnątrz katalogu *HelloWorld* utwórz katalog dla kaskadowych arkuszy stylów.

   Z oczywistych względów folder ten powinien nosić nazwę *css*. Znajdą się w nim wszystkie pliki CSS wykorzystywane przez aplikację. Pamiętaj, że podstawę każdej aplikacji AIR stanowią pliki HTML. Z tego względu struktura katalogów Twoich aplikacji może przypominać strukturę strony internetowej.

5. Wewnątrz katalogu aplikacji utwórz katalog dla skryptów JavaScript.

   Folder ten otrzyma nazwę *js*. Będą w nim przechowywane wszystkie skrypty i pliki tworzone w języku JavaScript.

6. Wewnątrz katalogu aplikacji utwórz katalog do przechowywania obrazków.

   Nie powinno dziwić, że nazwiemy go *images*. Moglibyśmy skorzystać także z nazwy *assets* lub *imgs*. Tak naprawdę nie ma to żadnego znaczenia; istotne jest tylko, że właśnie do tego katalogu będą trafiać wszystkie obrazki.

7. Wewnątrz katalogu aplikacji utwórz katalog dla ikon (rysunek 2.13).

   Po chwili zastanowienia postanowiłem utworzyć także katalog *icons*. W rozdziale 16. zajmiemy się tworzeniem specjalnych ikon na potrzeby naszych aplikacji. Pliki nie należą do obrazków wykorzystywanych w aplikacji, dlatego postanowiłem umieścić je w osobnym katalogu.

Wskazówka

- W swojej aplikacji możesz umieścić także inne katalogi, na przykład *audio* (do przechowywania dźwięków wykorzystywanych w aplikacji), *docs* (dla dokumentacji) lub *resources* (dla innych zasobów). Oczywiście nazwy te nie są obowiązkowe, stanowią tylko sugestię.

Tworzenie pliku HTML

Pierwszym plikiem, który utworzę w tej aplikacji, jest plik HTML (nazywany też stroną główną). W trakcie tworzenia aplikacji AIR z wykorzystaniem języków HTML i JavaScript ten dokument będzie stanowił ich podstawę.

Jedną z największych zalet biblioteki AIR jest możliwość wykorzystania wiedzy na temat tworzenia aplikacji webowych do tworzenia aplikacji uruchamianych na komputerze użytkownika — desktopowych. Oznacza to, że do tworzenia interfejsu aplikacji możesz wykorzystywać te same mechanizmy, co w przypadku stron internetowych. Możesz także uruchamiać strony internetowe w przeglądarce internetowej — aplikacja będzie wyglądać w niej tak samo, jak uruchamiana samodzielnie.

W pierwszym przykładzie zademonstruję klasyczną stronę typu *Hello, World!*. Powstała aplikacja nie będzie realizować żadnej praktycznej funkcjonalności, pozwoli Ci jednak zapoznać się z procesem tworzenia, pakowania i uruchamiania aplikacji, co przyda się w następnych rozdziałach, aż do końca książki.

Aby utworzyć plik HTML:

1. Utwórz nowy dokument HTML w dowolnym edytorze tekstowym (skrypt 2.1) i wprowadź w nim następującą treść:

```
<html>
<head>
    <title>Witaj, świecie!</title>
</head>
```

   Nie tworzymy prawdziwej strony internetowej, dlatego nie musimy umieszczać wielu dodatkowych znaczników, takich jak `DOCTYPE`, `META` itd.

2. Dodaj ciało (treść) dokumentu HTML:

```
<body>
    <h1>Witaj, świecie!</h1>
</body>
```

   Jak już wspomnieliśmy, nie jest to najbardziej rozbudowana aplikacja, jaką przyjdzie nam napisać. Dzięki temu działaniu dowiesz się, jak łatwo można tworzyć własne aplikacje desktopowe.

3. Zakończ dokument HTML:

```
</html>
```

4. Zapisz plik pod nazwą *HelloWorld.html* w katalogu aplikacji.

   Plik ten powinien być umieszczony w głównym katalogu aplikacji, a nie w którymś z podkatalogów.

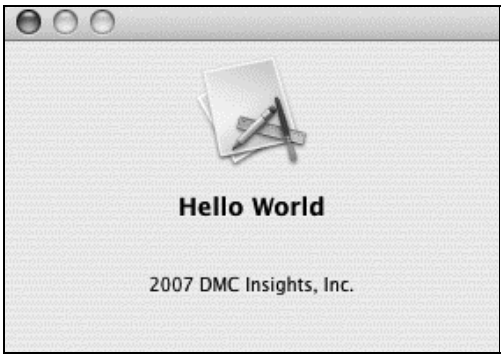
Wskazówki

- Silnik renderujący język HTML — mechanizm, który interpretuje ten język i tworzy efekt graficzny — wykorzystywany przez AIR nosi nazwę WebKit (*www.webkit.org*). Jest on wykorzystywany również przez przeglądarkę Safari, dlatego właśnie ta przeglądarka interpretuje dokumenty HTML w sposób najbardziej zbliżony do biblioteki AIR. Z Safari 3 mogą korzystać zarówno użytkownicy systemów Mac, jak i Windows.
- Aplikacje AIR mogą być tworzone z wykorzystaniem Ajaksa, Flasha lub Fleksa. Aplikacje AIR tworzone w Ajaksie mają rozszerzenie *.html*. W przypadku pozostałych dwóch formatów mamy do czynienia z plikami SWF (format Shockwave).

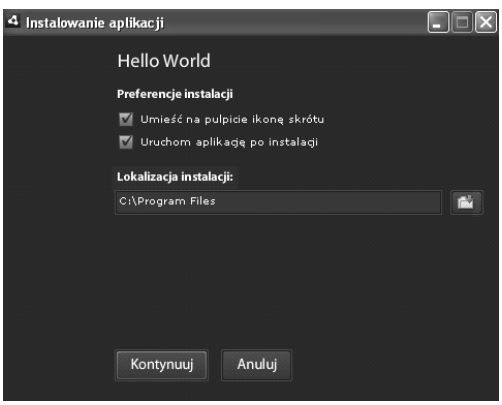
Skrypt 2.1. *Plik HTML, który stanowi podstawę naszej aplikacji AIR*

Skrypt

```
<html>
<head>
    <title>Witaj, świecie!</title>
</head>
<body>
    <h1>Witaj, świecie!</h1>
</body>
</html>
```



Rysunek 2.14. *Okno About aplikacji, w którym widnieje nazwa programu, nota copyright i informacja o autorze*



Rysunek 2.15. *Niektóre z informacji zawartych w pliku deskryptora aplikacji są wykorzystywane w trakcie instalacji*

Tworzenie pliku XML

Oprócz pliku HTML Twoja aplikacja AIR musi zawierać także plik XML, który jest określany jako plik deskryptora aplikacji. W pliku tym umieszcza się wszelkie metadane (informacje związane z programem) dla aplikacji. W ich skład wchodzą między innymi:

- nazwa,
- wersja,
- autor,
- opis,
- nota copyright,
- ikony,
- domyślny folder instalacji,
- wygląd okna i jego zachowanie,
- i inne.

Wiele spośród tych informacji jest widocznych w menu *About* (rysunek 2.14), a także w trakcie procesu instalacji (rysunek 2.15).

Jeśli nigdy wcześniej nie tworzyłeś dokumentów XML, nie musisz się martwić: dokumenty takie nie różnią się znacznie od plików HTML. Wyjaśnię Ci dokładnie wszystkie szczegóły, które powinieneś wiedzieć. Skupię się przy tym na *wymaganych* elementach XML. W kolejnych rozdziałach (zwłaszcza w rozdziale 16.) poznasz inne dodatkowe ustawienia, które podaje się w tym pliku.

Aby utworzyć plik XML:

1. Utwórz dokument XML w edytorze tekstowym (skrypt 2.2) i rozpocznij go następującą instrukcją:

```
<?xml version="1.0" encoding= "utf-8" ?>
```

   Pliki XML są zwykłymi dokumentami tekstowymi, które można tworzyć w niemal dowolnym edytorze tekstowym. Na początku każdego pliku umieszcza się deklarację XML (jest to powyższy wiersz). Określa ona wykorzystywaną wersję XML (1.0 to jedna z najczęściej stosowanych) i kodowanie (zapoznaj się z pierwszą wskazówką na końcu podrozdziału).

2. Dodaj znacznik `application`.

```
<application>
</application>
```

   Wszystkie pliki XML muszą określać jeden znacznik bazowy (zauważ, że pliki HTML zawierają znacznik bazowy `html`). W przypadku deskryptora aplikacji AIR jest to znacznik `application`. Cała pozostała treść danych XML musi znaleźć się pomiędzy znacznikami otwierającym i zamykającym `application`.

3. Dodaj atrybut `xmlns` do znacznika otwierającego `application`.

```
<application xmlns="http://ns.adobe.com/
➥air/application/1.0">
```

   Atrybut `xmlns` określa *przestrzeń nazw XML*. Przestrzenie nazw są skomplikowaną konstrukcją programistyczną, której nie musisz na szczęście znać. Musisz tylko wiedzieć, że wartość tego atrybutu określa najstarszą wersję biblioteki AIR, z którą aplikacja może współpracować. W tym przypadku odwołujemy się do pierwszej oficjalnej wersji biblioteki. Programy napisane dla tej wersji powinny uruchamiać się na dowolnej wersji biblioteki AIR opublikowanej od momentu wydania wersji 1.0.

Skrypt 2.2. *Plik deskryptora aplikacji w formacie XML, wymagany przez każdą aplikację AIR*

Skrypt

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/
➥air/application/1.0">
    <id>com.dmci.air.HelloWorld</id>
    <filename>Hello World</filename>
    <version>1.0</version>
    <initialWindow>
        <content>HelloWorld.html</content>
        <visible>true</visible>
    </initialWindow>
</application>
```

4. Pomiędzy znacznikami otwierającym i zamykającym `application` dodaj element `id`.

```
<id>com.dmci.air.HelloWorld</id>
```

Wartość znacznika ID określa unikalne odniesienie (nazwa) AIR dla programu. Zalecana postać to `com.firma.aplikacja`. Powinna w ten sposób powstać unikatowa wartość, niemniej musi ona też mieć jakieś znaczenie. Aplikacja AIR firmy Adobe przyjmie `id` o wartości `com.adobe.air.` ↪`nazwa`. W przypadku aplikacji stworzonej przez moją firmę (DMC Insights, Inc.) korzystam z nazwy `com.dmci.air.` ↪`HelloWorld`. Powinieneś dostosować format według własnego uznania.

W nazwach możesz korzystać z liter A-Z, cyfr 0-9, znaków kropki i myślnika. Nie stosuj spacji. Maksymalna długość elementu `id` to 255 znaków.

5. Pomiędzy znacznikami otwierającym i zamykającym `application` dodaj znacznik `filename`.

```
<filename>Hello World</filename>
```

Znacznik ten określa nazwę aplikacji, jaką będą widzieć jej użytkownicy. Nazwa ta jest widoczna w menu *About* (rysunek 2.14), w skrótach (rysunek 2.16), w menu *Start* (tylko w systemie Windows) itd. Z drugiej strony istnieje przecież znacznik `id`, który stanowi nazwę wykorzystywaną tylko w mechanizmach aplikacji — końcowi użytkownicy prawdopodobnie nigdy nie będą mieli z nią do czynienia.

6. Pomiędzy znacznikami otwierającym i zamykającym `application` dodaj element `version`.

```
<version>1.0</version>
```

Ten element określa wersję aplikacji. Chociaż może to być wartość dowolna, w praktyce powinna ona mieć ściśle określoną budowę. Wersje beta aplikacji otrzymują z reguły liczby mniejsze od 1. Kolejne ważne dla aplikacji aktualizacje otrzymują jako wersję kolejne liczby całkowite (od 1 do 2, od 2 do 3 itd.), natomiast w przypadku niewielkich poprawek do „dużego" numeru wersji dodaje się część dziesiętną (niewielka aktualizacja wersji 1.1 zwiększy numer wersji do 1.2). Najważniejsze, aby kolejne wersje aplikacji miały większe numery, dzięki czemu użytkownik wie, że zmiana wersji oznacza aktualizację.

Jak widać na skrypcie 2.2, każdy element (lub para znaczników) jest umieszczony pomiędzy znacznikami `application` otwierającym a zamykającym. Nie ma znaczenia, w jakiej kolejności elementy są podawane.

7. Pomiędzy znacznikami otwierającym i zamykającym `application` dodaj element `initialWindow`.

```
<initialWindow>
</initialWindow>
```

Element `initialWindow` od tego momentu będzie zawierał wartości określające treść i wygląd głównego okna aplikacji.

Rysunek 2.16. *Skrót do zainstalowanej aplikacji, w którym jest wykorzystywany element filename z pliku XML*

8. Pomiędzy znacznikami otwierającym i zamykającym `initialWindow` dodaj znacznik `content`.

```
<content>HelloWorld.html</content>
```

Wartość elementu content określa dokładną nazwę bazowego pliku HTML (skrypt 2.1). Najlepiej jest przechowywać zarówno pliki HTML, jak i XML w tym samym folderze, dzięki czemu w tym miejscu wystarczy podać samą nazwę pliku HTML (bez dodatkowych oznaczeń ścieżki). Gdybyś zdecydował się wybrać dwa różne katalogi, musiałbyś w pliku XML skorzystać z relatywnej ścieżki do pliku HTML (na przykład *../HelloWorld.html* lub *content/HelloWorld.html*).

9. Pomiędzy znacznikami otwierającym i zamykającym `initialWindow` dodaj znacznik `visible`.

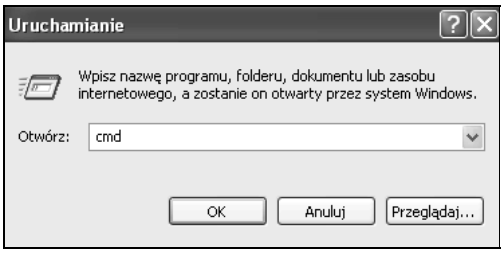
```
<visible>true</visible>
```

W tym rozdziale chcę się skupić na wymaganych elementach deskryptora aplikacji. Element `visible` (stanowiący fragment znacznika `initialWindow`) nie jest co prawda wymagany, ale z niewiadomych przyczyn jego domyślna wartość to `false`. Oznacza to, że aplikacja po napisaniu, przetestowaniu i wdrożeniu będzie działać, ale nie będzie widoczna! Zakładam, że chcesz, aby użytkownicy mogli oglądać rezultaty Twojej pracy — powinieneś więc dodać ten wiersz.

10. Zapisz plik pod nazwą *application.xml* w tym samym katalogu, co plik *HelloWorld.html*. Możesz nadać też inną nazwę (z rozszerzeniem *.xml*), jednak przyjęło się stosować właśnie nazwę *application.xml*. Mógłbyś także skorzystać z nazw *HelloWorld.xml* lub *HelloWorld-app.xml* (dzięki temu widać, że to właśnie ten plik jest deskryptorem aplikacji HelloWorld).

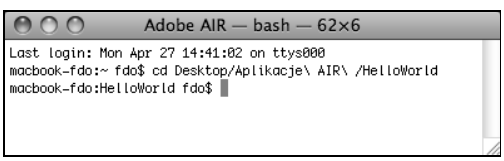
Wskazówki

- Kodowanie określa, jaki rodzaj znaków będzie obsługiwany w danym pliku. UTF-8 jest jednym z najpopularniejszych standardów kodowania; nie powinien on sprawiać problemów w plikach XML.
- Jedną z przydatnych funkcji środowisk programistycznych obsługujących bibliotekę AIR (na przykład Dreamweaver z rozszerzeniem AIR lub Aptana) jest pomoc w tworzeniu plików XML. W rozdziale 3. piszę więcej na ten temat.
- Jeśli utworzyłeś dwie aplikacje AIR o tym samym `id`, zostaną one potraktowane przez bibliotekę uruchomieniową jako ten sam program, przez co nie będziesz mógł korzystać z obu aplikacji na jednym komputerze. Dwie aplikacje mogą mieć te same wartości znaczników `filename`, jednak taka sytuacja byłaby myląca dla końcowych użytkowników.



Rysunek 2.17. *Skorzystaj z narzędzia Uruchom, aby otworzyć konsolę systemu Windows*

Rysunek 2.18. *Okno konsoli systemu Windows (na Twoim komputerze prawdopodobnie ujrzysz biały tekst na czarnym tle)*



Rysunek 2.19. *Przejście w strukturze katalogów do katalogu aplikacji HelloWorld w oknie Terminala systemu Mac OS X*

Testowanie aplikacji

Po utworzeniu całej aplikacji (w tym przypadku jednego pliku HTML i jednego XML) musisz ją przetestować i skompilować (kompilacja jest ostatnim krokiem, w którego wyniku powstaje plik wykonywalny). Do testowania aplikacji wykorzystuje się narzędzie uruchamiane z linii poleceń — AIR Debug Launcher (*adl*), które wchodzi w skład SDK. Składnia polecenia jest prosta:

```
adl ApplicationXMLFile.xml
```

Możliwość testowania aplikacji w trakcie ich tworzenia jest niezwykle ważna. Możesz również skompilować całą aplikację, zainstalować ją i zobaczyć, jak (czy w ogóle) działa. Po wykonaniu poniższych kroków zaoszczędzisz sporo czasu — będziesz musiał kompilować aplikację tylko raz, już po zakończeniu tworzenia i testowania aplikacji.

Aby przetestować aplikację AIR:

1. Uruchom powłokę w swoim systemie.

   W systemie Windows wybierz opcję *Uruchom* z menu *Start* i wprowadź polecenie `cmd` w pole tekstowe (rysunek 2.17). Rysunek 2.18 przedstawia uruchomiony wiersz polecenia.

   Użytkownicy systemu Mac OS X muszą jedynie uruchomić aplikację Terminal (*Programy/Narzędzia*). Jeśli okno powłoki nie zostanie otwarte automatycznie, wybierz opcję *Powłoka/Nowe okno* lub wciśnij kombinację klawiszy *Command+N*.

2. Przejdź do katalogu projektu — wprowadź polecenie `cd ścieżka_do_projektu_HelloWorld` i wciśnij klawisz *Enter/Return* (rysunek 2.19).

   Musisz zmienić argument polecenia, aby dopasować go do lokalizacji projektu. Możesz też wpisać polecenie `cd` i spację, a następnie przeciągnąć folder *HelloWorld* do okna konsoli. Ścieżka do katalogu zostanie automatycznie wpisana w tym oknie.

3. Wprowadź poniższe polecenie i wciśnij klawisz *Enter/Return* (rysunek 2.20).

```
adl application.xml
```

Aplikacja powinna uruchomić się w osobnym oknie (rysunek 2.21). Plik XML zawiera odniesienie do głównego dokumentu — pliku HTML — dlatego tak wywołane polecenie wystarcza do przetestowania aplikacji.

Jeśli zobaczysz informację o niemożności znalezienia Javy (rysunek 2.22), oznacza to, że JRE nie zostało jeszcze zainstalowane w systemie. Jeżeli system nie rozpoznaje polecenia `adl`, musisz jeszcze raz zmodyfikować ścieżkę systemową, ponieważ wcześniej popełniłeś jakiś błąd w trakcie wykonywania tego kroku (zapoznaj się z instrukcjami poświęconymi modyfikowaniu ścieżki, zawartymi w tym rozdziale).

4. Zamknij narzędzie *adl*, aby zamknąć aplikację i powrócić do powłoki.

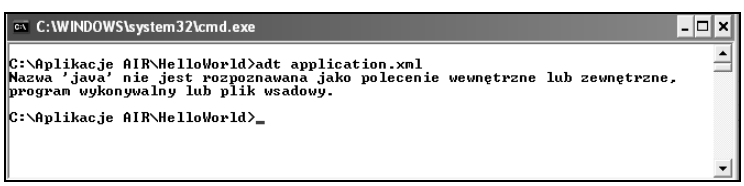
Testowanie w przeglądarce

Aplikacje AIR opisywane i tworzone w niniejszej książce wykorzystują technologię Ajax (czyli HTML i JavaScript), dlatego można testować je także w przeglądarce internetowej. AIR wykorzystuje ten sam silnik renderujący, co przeglądarka Safari firmy Apple, dlatego to właśnie w tej aplikacji uzyskuje się najlepsze (najbardziej zbliżone do oryginału) rezultaty. Przeglądarka ta jest dostępna w wersji 3 zarówno w systemie Windows, jak i Mac OS X. Dobre efekty powinno także przynieść testowanie aplikacji w Firefoksie — zwłaszcza że przeglądarka ta udostępnia znakomite narzędzia do debugowania kodu JavaScript.

Teoretycznie można testować aplikacje także w Internet Explorerze. Odradzam jednak takie rozwiązanie z dwóch przyczyn. Po pierwsze, JavaScript w tej przeglądarce nie zawsze jest wykonywany tak samo, jak w aplikacjach AIR (to typowy problem wszystkich aplikacji Ajaksowych). Po drugie, Internet Explorer zawiera wiele dziwnych mechanizmów, które komplikują tworzenie i testowanie aplikacji webowych ponad miarę (przynajmniej moim zdaniem).

Rysunek 2.20. *Wywołanie programu AIR Debug Launcher w systemie Windows*

Rysunek 2.21. *Efekt działania aplikacji w systemie Windows*



Rysunek 2.22. *Jeśli aplikacja adl nie może znaleźć Java Runtime Environment (JRE), prawdopodobnie zobaczysz taką informację o błędzie*

Tworzenie certyfikatu

Po zakończeniu testowania aplikacji AIR możesz wykonać kompilację. Skompilowana aplikacja teoretycznie mogłaby być uruchamiana przez końcowych użytkowników. Teoretycznie, ponieważ w praktyce AIR wymaga, aby każda aplikacja miała *cyfrowy podpis certyfikujący*. Dzięki temu łatwiej jest zweryfikować autentyczność aplikacji — użytkownik ma pewność, że aplikacja nie uszkodzi jego komputera.

Istnieją dwa rodzaje certyfikatów, które możesz wykorzystywać. Możesz zakupić certyfikaty w centrum autentyfikującym, potwierdzającym autentyczność certyfikatów wystawionych dla firm. Do takich centrów należą Thawte lub VeriSign. Certyfikaty wystawiane przez te firmy wymagają najwyższego poziomu bezpieczeństwa, ponieważ firmy te sprawdzają dane osoby/instytucji, która chce uzyskać certyfikat (dzięki temu cracker nie uzyska certyfikatu potwierdzającego jego autentyczność jako na przykład banku). Certyfikaty można też generować samemu. W ten sposób będziesz w stanie tworzyć instalatory AIR dla Twoich aplikacji, jednak końcowi użytkownicy aplikacji nie będą mieli pewności, że Twoja aplikacja działa bezpiecznie. Generowanie certyfikatów w ten sposób jest bezpłatne, ale użytkownicy muszą mieć do Ciebie zaufanie. Jeśli jednak chcesz tylko przetestować aplikację, tworzenie własnego certyfikatu na pewno ma sens. W następnych krokach zademonstruję metodę tworzenia takiego certyfikatu przy użyciu narzędzia *adt* (AIR Development Tool).

Aby utworzyć certyfikat:

1. Uruchom powłokę systemową.

   Sposoby uruchamiania powłoki (wiersza poleceń) omówiłem zarówno dla systemu Windows, jak i Mac OS X w poprzednim podrozdziale.

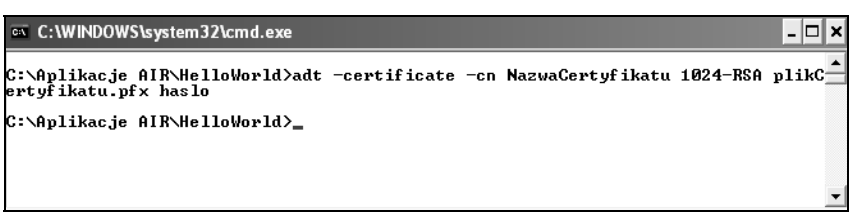
2. Przejdź do katalogu, w którym znajduje się Twoja aplikacja, wprowadzając polecenie `cd ścieżka_do_aplikacji_AIR` i wciskając klawisz *Enter/Return*.

   Certyfikat najlepiej jest utworzyć w tym samym katalogu, co aplikację (zakładam, że posiadasz aplikację utworzoną wcześniej w tym rozdziale). Musisz zmienić wyżej podane polecenie, aby dopasować je do położenia Twojej aplikacji na dysku.

   Jeśli chcesz utworzyć certyfikat gdzieś indziej (na przykład na Pulpicie), nic nie stoi na przeszkodzie, ale musisz skorzystać z polecenia `cd`, aby upewnić się, czy na pewno znajdujesz się w katalogu, w którym chciałeś się znaleźć.

3. Wprowadź następujące polecenie i wciśnij klawisz *Enter/Return* (rysunek 2.23).

```
adt -certificate -cn NazwaCertyfikatu
↪1024-RSA plikCertyfikatu.pfx haslo
```



Rysunek 2.23. *Tworzenie nowego (automatycznie podpisanego) certyfikatu o nazwie plikCertyfikatu.pfx*

Za pomocą tej instrukcji utworzysz automatycznie podpisany certyfikat (korzystamy tylko z niezbędnych opcji; kursywą zaznaczyłem wartości, które prawdopodobnie będziesz chciał zmienić). Wartość *`NazwaCertyfikatu`* powinna być zmieniona na nazwę, którą z reguły nadajesz certyfikatom. Może to być nazwa Twojej firmy lub inna wartość, która będzie wskazywać, że jesteś twórcą aplikacji.

Wartość *`plikCertyfikatu.pfx`* określa nazwę generowanego pliku. Również w tym miejscu warto wykorzystać sensowną nazwę (na przykład `CertyfikatAplikacji`); nie zapomnij jednak o rozszerzeniu *.pfx*. Argument *`haslo`* określa hasło, które będzie skojarzone z certyfikatem (nie może zawierać polskich znaków). Trzeba będzie wprowadzić je w momencie kompilowania aplikacji (w jednym z następnych kroków).

Oznaczenie `1024-RSA` wskazuje, z jakiego rodzaju klucza korzystamy w przypadku tego certyfikatu (w ten sposób określamy siłę klucza, a więc jedną z głównych kwestii bezpieczeństwa). Można także skorzystać z wartości `2048-RSA`.

4. Sprawdź zawartość katalogu, aby upewnić się, że certyfikat został utworzony poprawnie.

Wskazówki

- Pełną listę opcji dostępnych dla generowania certyfikatów znajdziesz po wywołaniu polecenia `adt --help`. Możesz też przejrzeć oficjalną dokumentację biblioteki AIR, aby poznać więcej szczegółów na ten temat.
- Każdy certyfikat wygenerowany przez narzędzie *adt* będzie unikalny, nawet jeśli wykonasz te same czynności w celu jego wygenerowania. Przy tworzeniu nowszych wersji swojej aplikacji pamiętaj o podpisaniu jej z wykorzystaniem tego samego certyfikatu, który został użyty w oryginalnej wersji aplikacji.
- Każdy certyfikat generowany samodzielnie będzie aktualny przez pięć lat od momentu powstania. Oznacza to, że każda aplikacja utworzona przy użyciu takiego certyfikatu musi być aktualizowana przynajmniej raz na pięć lat. Po tym czasie trzeba opublikować nową wersję aplikacji, zawierającą nowy certyfikat.

Kompilowanie aplikacji

Po utworzeniu i przetestowaniu aplikacji możesz wykonać kompilację (nazywaną też *pakowaniem*) aplikacji. W wyniku tego procesu otrzymasz plik *.air*, który możesz dostarczać użytkownikom i instalować.

Składnia polecenia `adt` pozwalająca na skompilowanie aplikacji ma postać:

```
adt -package -storetype pkcs12 -keystore
↪plikCertyfikatu.pfx NazwaAplikacji.air
↪DeskryptorAplikacji.xml BazowyPlikHTML.html
```

Argument `-package` wskazuje, że chcemy utworzyć aplikację spakowaną. Parametry `-storetype pkcs12 -keystore` *`plikCertyfikatu.pfx`* określają certyfikat, który będzie wykorzystywany w aplikacji (korzystamy z pliku utworzonego w poprzednim podrozdziale). Następny argument definiuje nazwę pliku *.air*, który zostanie utworzony. Na końcu podajemy nazwę pliku XML, pliku bazowego HTML i wszystkich innych plików, które muszą być razem spakowane. Każdy zasób, plik lub folder, wykorzystywany przez aplikację, musi być podany w wywołaniu polecenia `adt`.

Aby w procesie kompilacji dołączyć także katalogi (to konieczne, jeśli Twoja aplikacja będzie zawierać pliki CSS, skrypty JavaScript, obrazki itd.), musiałbyś skorzystać z polecenia:

```
adt -package -storetype pkcs12 -keystore
↪plikCertyfikatu.pfx NazwaAplikacji.air
↪DeskryptorAplikacji.xml BazowyPlikHTML.html
↪css icons images js …
```

Wykorzystajmy zdobytą wiedzę, aby skompilować aplikację HelloWorld.

Aby skompilować aplikację:

1. Uruchom powłokę systemową.

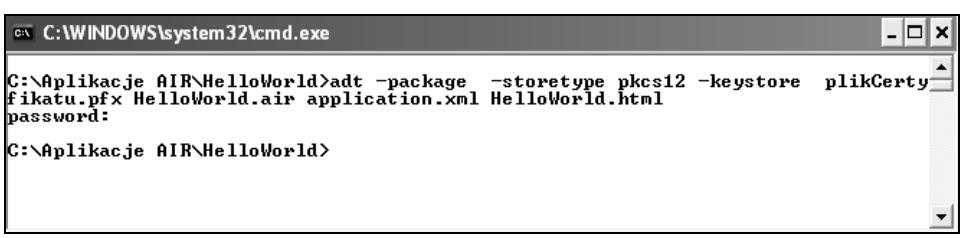
   Sposoby uruchamiania powłoki (wiersza poleceń) omówiłem zarówno dla systemu Windows, jak i Mac OS X w poprzednim podrozdziale.

2. Przejdź do katalogu, w którym znajduje się Twoja aplikacja, wprowadzając polecenie `cd` *`ścieżka_do_aplikacji_AIR`* i wciskając klawisz *Enter/Return*.

   Ponownie musisz dostosować polecenie, tak aby mogło ono być wykonane poprawnie na Twoim komputerze.

3. Wprowadź następujące polecenie i wciśnij klawisz *Enter/Return* (rysunek 2.24).

   ```
   adt -package  -storetype pkcs12
   ↪-keystore  /ścieżka/do/certyfikatu/
   ↪plikCertyfikatu.pfx HelloWorld.air
   ↪application.xml HelloWorld.html
   ```



Rysunek 2.24. *Tworzenie pliku .air przy użyciu programu adt (wraz ze spakowaniem wymienionych elementów)*

Musisz zmienić argument */ścieżka_do_certyfikatu/plikCertyfikatu.pfx* na relatywną (względem katalogu aplikacji) ścieżkę dostępu do Twojego certyfikatu. Kiedy wykonywałem to polecenie (rysunek 2.24), znajdowałem się w folderze *HelloWorld*, wewnątrz katalogu *Aplikacje AIR*. W katalogu *Aplikacje AIR* znajduje się też plik *plikCertyfikatu.pfx*, dlatego odwołując się do tego pliku, podałem ścieżkę *../plikCertyfikatu.pfx*. Zapis ten oznacza, że w hierarchii katalogów należy przejść jeden katalog wyżej i tam odszukać plik *plikCertyfikatu.pfx*. Jedynym problemem związanym z wywoływaniem powyższego polecenia jest konieczność wprowadzenia go w jednym wierszu (nie możesz wciskać klawisza *Enter/Return* w trakcie podawania tego polecenia).

Po wpisaniu całego polecenia możesz wcisnąć klawisz *Enter/Return*. Zostaniesz zapytany o hasło certyfikatu.

4. Sprawdź, czy udało się poprawnie skompilować aplikację — wystarczy potwierdzić obecność pliku *HelloWorld.air* w folderze aplikacji (rysunek 2.25).

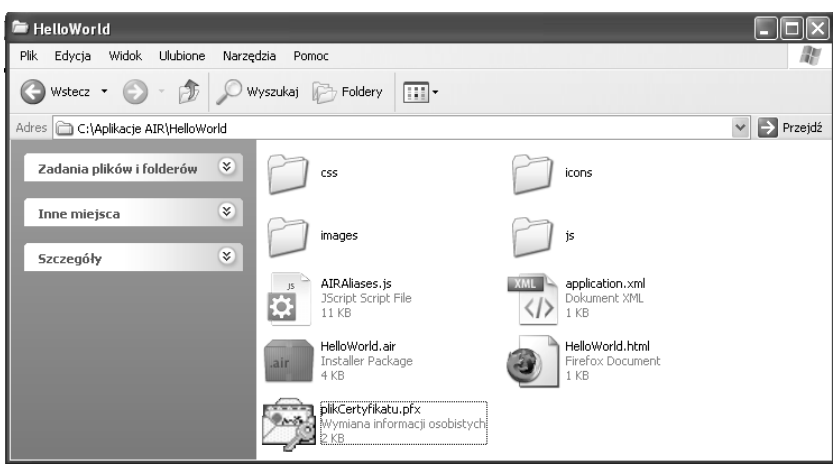
5. Zainstaluj i uruchom aplikację *HelloWorld.air*, korzystając z instrukcji podanych w rozdziale 1.

Wskazówka

- Narzędzie *adt* mógłbyś wywołać również w następujący sposób:

```
adt -package -storetype pkcs12 -keystore
➥plikCertyfikatu.pfx NazwaAplikacji.air
➥DeskryptorAplikacji.xml .
```

  Kropka na końcu polecenia reprezentuje wszystkie pliki i katalogi w bieżącym katalogu, dzięki czemu wszystkie pliki trafią do instalatora aplikacji. Jest to dość wygodne rozwiązanie, ale niezbyt eleganckie. Wywołanie z dokładnym określeniem plików wchodzących w skład instalacji to z pewnością najlepsze rozwiązanie.



Rysunek 2.25. *Nowo utworzony plik HelloWorld.air może być rozpowszechniany i wykorzystywany do instalacji naszej aplikacji*