LOTUS DOMINO

Tworzenie Aplikacji W Środowisku Lotus Domino

Michał Kalewski
mkalewski@cs.put.poznan.pl

Instytut Informatyki Politechniki Poznańskiej

Domino – tworzenie aplikacji

Plan Prezentacji

(1) Wstęp
(2) Korzystanie z programu Domino Designer
(3) Programowanie aplikacji w środowisku Domino
(4) Język @formuł
(5) LotusScript
(6) Inne możliwości programowania
(7) Literatura

Domino – tworzenie aplikacji

Wstęp

Projektowanie aplikacji dla środowiska Domino odbywa się przy użyciu specjalnego programu nazwanego Domino Designer. Aplikacja ta umożliwia tworzenie programów, które następnie są umieszczane na serwerze Domino.

Dostęp do opublikowanych programów jest możliwy za pomocą klienta — Lotus Notes lub przy użyciu dowolnej przeglądarki internetowej (jeżeli twórca programu udostępni go użytkownikom internetowym).

Programista aplikacji Domino może dowolnie manipulować prawami dostępu do niej i do większości jej elementów. Odbywa się to z wykorzystaniem mechanizmu list kontroli dostępu ACL (ang. *Access Control List*), w których można określać prawa dostępu dla użytkowników i grup użytkowników.

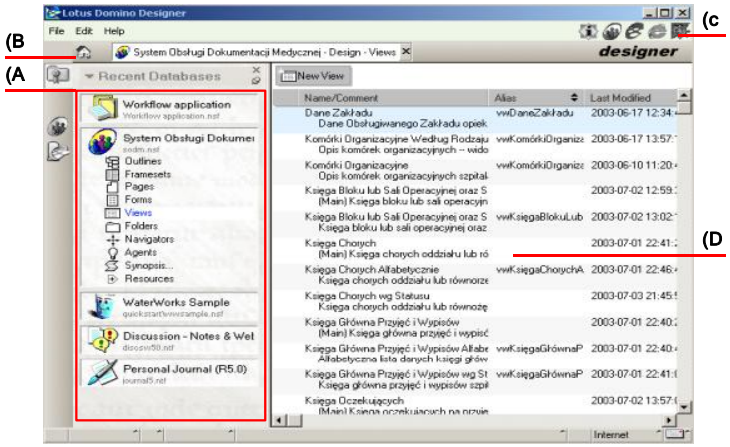
Domino – tworzenie aplikacji

Wstęp

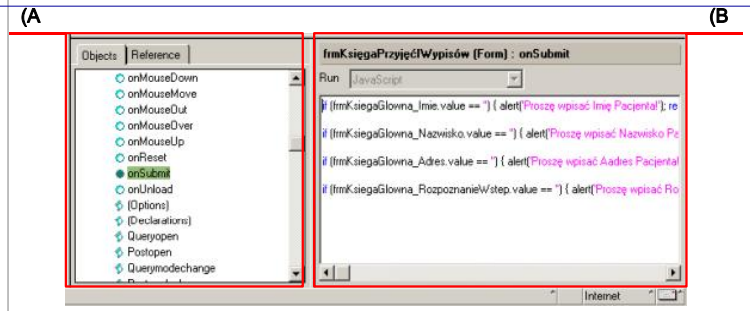
Programista tworzący oprogramowanie dla środowiska Domino przy użyciu Designer-a ma do dyspozycji szereg **elementów projektowych** (ang. *Domino design elements*), z których korzysta budując aplikację.

Każda aplikacja Domino zawiera swoją bazę danych dokumentów oraz składa się z widoków i form. Widoki i formy można umieszczać wewnątrz ramek (na wzór ramek standardu HTML — ang. *frames*). Dodatkowo formy zawierają dowolne elementy zwane polami, wśród nich dostępne są przyciski (ang. *button*), listy wyboru (ang. *dialog list*), pola wyboru (ang. *check box*) oraz inne elementy znane z graficznych interfejsów użytkownika.

W odniesieniu do wszystkich elementów aplikacji Domino można używać języka **JavaScript** oraz standardu **DOM** (ang. *Document Object Model*). Środowisko Domino posiada także własne języki programowania: **język formuł** oraz **LotusScript**.



(A) panel projektowy (ang. *design pane*), **(B)** pasek otwartych elementów (ang. *window tab*), **(C)** przyciski podglądu aplikacji (ang. *preview buttons*), **(D)** obszar roboczy (ang. *work area*).



Panel programowy (ang. *programmer's pane*) wyświetlany w obszarze roboczym wybranego elementu aplikacji;
(A) lista informacyjna (ang. *info list*), składająca się z: widoku obiektów (ang. *objects view*) oraz listy referencyjnej (ang. *reference view*); **(B)** obszar kodowania (ang. *script area*).

Obszar kodowania pozwala na wprowadzanie kodu w wybranym języku: formuł, JavaScript lub LotusScript. Możliwe jest także sprawdzanie poprawności składni kodu oraz poprawianie błędów edycyjnych.

Widok obiektów zawiera listę wszystkich obiektów i zdarzeń jakie dotyczą otwartego elementu projektowego;
wybranie jakiejś pozycji z tej listy powoduje, że w obszarze kodowania pojawia się kod dotyczący tego (wskazanego) obiektu lub zdarzenia — elementy listy, którym przypisano jakiś kod są oznaczone ciemniejszym kolorem.

Zawartość listy referencyjnej zależy od wybranego języka programowania:
-- jeżeli wybranym językiem jest język formuł, to lista referencyjna zawiera komendy, funkcje i pola jakie można stosować w formułach;
-- jeśli programowanie odbywa się przy użyciu języka JavaScript, to okno to zawiera informacje dotyczące standardu DOM;
-- w przypadku języka LotusScript lista referencyjna może zawierać (do wyboru) m.in. funkcje języka, klasy dotyczące obsługi środowiska Domino lub klasy OLE.

Do tworzenia aplikacji w środowisku Domino wykorzystuje się tzw. elementy projektowe (ang. *Domino design elements*); są one podstawą każdej aplikacji i mogą być uzupełniane kodami wspieranych przez środowisko Domino języków programowania.

Baza danych aplikacji
Baza danych przechowuje samą aplikację Domino i jest zbiorem wszystkich wprowadzonych przy użyciu tej aplikacji informacji; jest ona umieszczona w jednym pliku dyskowym. Aplikacja środowiska Domino używa tylko jednej (własnej) bazy danych; istnieje jednak możliwość dostępu do baz danych innych aplikacji oraz innych serwerów Domino. Dane wszystkich aplikacji Domino są zorganizowane w logiczne struktury nazywane **dokumentami**.

Formy (ang. *forms*)
Forma jest strukturą służącą do wprowadzania oraz prezentowania informacji do i z bazy danych. Forma może zawierać dowolne dostępne elementy graficznego interfejsu użytkownika (np. listy wyboru, pola tekstowe lub elementy graficzne) oraz przyciski akcji. Baza danych aplikacji zawiera dokumenty, które są utworzone za pomocą form. Istnieje także możliwość utworzenia tzw. **pod-form** (ang. *subforms*), które mogą być współdzielone przez wiele form.

Widoki (ang. *views*)
Widok jest listą dokumentów, które są zapisane w bazie danych. Możliwe jest określenie kryteriów, na podstawie których wybierane są dokumenty do wyświetlenia w widoku — niepodanie żadnego kryterium powoduje wyświetlenie wszystkich dokumentów zawartych w bazie danych aplikacji. Dokumenty na wyświetlanej liście widoku mogą być grupowane i/lub sortowane w zależności od ich zawartości.

Pola (ang. *fields*)
Pola umieszczone na formach służą do wprowadzania i przechowywania danych; pola formy określają jakie dane może zawierać pojedynczy dokument. Dane wprowadzane do pola lub w nim wyświetlane mogą być przetwarzane przez formuły lub funkcje. Istnieje także możliwość tworzenia **współdzielonych pól** (ang. *shared fields*), czyli pól, które są używane przez różne formy aplikacji. Wśród dostępnych pól są również **tablice**, które pomagają prezentować dane, ale także umożliwiają dowolne rozmieszczenie innych pól na formie.

Przyciski akcji (ang. *action buttons*)
Przyciski akcji służą do uruchamiania formuł oraz funkcji, mogą być używane do tworzenia menu aplikacji i automatyzacji czynności związanych z edycją dokumentów (np. zapisywanie zmian).

Struktury (ang. *outline designer*)
Struktura pozwala na tworzenie połączeń pomiędzy wszystkimi elementami aplikacji, zarządzanie tymi połączeniami oraz wyświetlanie (w ramach aplikacji) graficznej prezentacji tych połączeń wraz z odsyłaczami (ang. *links*) do prezentowanych części aplikacji.

Grupy ramek (ang. *frameset*)
W ramach Domino Designer R5 możliwe jest tworzenie grupy ramek, które zawierają różne elementy aplikacji; rozwiązanie to jest wzorowane na funkcji o tej samej nazwie standardu HTML.

Strony (ang. *pages*)
Strony są nowymi elementami Domino Designer i są przeznaczone do prezentacji statycznych informacji, tzn. informacji, która została wprowadzona na etapie tworzenia aplikacji — nie można prezentować na stronach informacji pochodzących np. z bazy danych aplikacji oraz umieszczać na nich pól do wprowadzania danych.

Do oprogramowania aplikacji tworzonej w środowisku Domino można użyć języka formuł, języków JavaScript i LotusScript, a także języka Java oraz C i C++ — wykorzystując dostępne API Środowisko Domino pozwala także na używanie standardu CORBA (ang. *Common Object Request Broker Architecture*) oraz technologii OLE (*ang. Object Linking and Embeding*).

Jednym z najważniejszych rozwiązań stosowanych w środowisku Domino do automatyzacji zadań jest system **agentów** (ang. *agents*). Agentem nazywa się samodzielny program, który realizuje określone zadania w bazie danych, np. dokonuje segregacji dokumentów, zmienia wartości pól w dokumentach, wysyła wiadomości poczty elektronicznej, kasuje określone dokumenty, lub też realizuje bardziej złożone czynności jak współpraca z zewnętrzną aplikacją.

Język @formuł to wyrażenia posiadające pewne cechy języków programowania; m.in. możliwe jest przy ich pomocy tworzenie zmiennych i przypisywanie im wartości oraz używanie operatorów logicznych.

Formuły składają się w głównej mierze z funkcji nazywanych **`@funkcjami`** — wzorowanymi na języku makr w produktach firmy Lotus, np. Lotus 1-2-3.

Predefiniowane **`@funkcje`** pozwalają na łatwe manipulowanie dokumentami Domino i ich zawartością oferując szeroki zakres gotowych rozwiązań — np. dotyczących wartości pól dokumentu lub odpowiedzi na wybrany dokument.

```
@SetField("Status" ; "Open");
```

Przykład. Język formuł: ustawienie wartości pola "Status" na wartość "Open".

Każda formuła zawiera jedną lub więcej komend, które są wykonywane według kolejności ich występowania.

W zależności od obiektu, któremu przypisano formułę i innych kryteriów, formuła może być wykonana wielokrotnie: osobno dla każdego dokumentu bazy danych.

Możliwe jest także selektywne wybranie dokumentów z bazy danych, dla których ma zostać wykonana formuła np. z użyciem specjalnej komendy SELECT.

Język formuł nie zawiera żadnych komend pozwalających na realizację pętli i przekazywanie sterowania w inne miejsce formuły, z wyjątkiem zwrotu wartości (kończącej jednocześnie wykonywanie) formuły.

Zmienne

```
variableName := value;
```

Przykłady:

```
date := @Created;

nMonths := "1" : "2" : "3" : "4" : "5" : "6„
: "7" : "8" : "9" : "10" : "11" : "12";

@Prompt([OK]; "Value of MyNumber";
                                @Text(TotalValue));


FIELD Subject := "No Subject"
```


Podstawowe Operatory

:=	Assignment
:	List concatenation
=	Equal
<> != =! ><	Not Equal
!	Logical NOT
&	Logical AND
|	Logical OR

Język @Formuł

Formuły mogą być także stosowane do obsługi agentów; programy agentów napisane w języku formuł operują (przy każdym uruchomieniu) na wszystkich dokumentach bazy danych lub na dokumentach, które wybrano poprzez funkcję selekcji spośród wszystkich dokumentów bazy.

```
@if (@Attachments > 0 ;
     @Return(" ");
     @MailSend("User" ; " " ; " " ; Subject ;
               "Please handle this" + @NewLine ;
               "Body" ; " ")
    );
```

Przykład. Język formuł: agent wysyłający każdy dokument z bazy danych, który nie posiada załącznika, do wskazanego użytkownika.

Język @Formuł

W ramach języka formuł można stosować tzw. **`@komendy`** - **`@commands`** - które pozwalają na wykonywanie operacji aplikacji Lotus Notes; m.in. wykonywanie wszystkich poleceń menu oraz obsługę otwartych dokumentów.

Komendy mają następującą składnie:

```
@command([command-name]; arg1; arg2; ... argn)
```


Język @Formuł

```
@If(@Command([FileOpenDatabase]; "NEWSUBJ"); "";
                                              @Return(""));
@Command([Compose]; ""; "Main Topic");
@Command([EditGotoField]; "Subject");
@Command([EditInsertText]; "New subject");
@Command([EditGotoField]; "Body")
```

Przykład. Język formuł: edycja dokumenty bazy danych „NEWSUBJ".

```
@Command([EditMakeDocLink]);
@Command([Compose]; „frmKsięgaPracowniDiagn");
@Command([EditGotoField]; "frmKsiegaPracDiag_Link");
@Command([EditPaste]);
@Command([EditTop])
```

Przykład. Język formuł:utworzenie połączenia pomiędzy dokumentami.

LotusScript

Język @formuł znakomicie nadaje się do realizacji prostych funkcji i nieskomplikowanych zadań, takich jak np. translacja wprowadzanych danych lub walidacja poprawności zawartości pól dokumentu;

LotusScript natomiast w przeciwieństwie do języka @formuł umożliwia programowanie pętli, funkcji wielokrotnego wyboru (ang. *case*) i wielu innych konstrukcji nie dostępnych w formułach.

LotusScript jest w pełni obiektowym językiem skryptowym pozwalającym realizować programy w produktach firmy Lotus (może być także używany m.in. w programach Lotus 1-2-3 97, LSCube lub Word Pro 96) i udostępnia predefiniowany zbiór klas obsługi produktów, w ramach których jest używany.

Domino – tworzenie aplikacji

LotusScript

Do obsługi bazy danych oraz aplikacji Domino w języku LotusScript używa się specjalnego modelu dostępu do danych o nazwie DOM (ang. *Domino Object Model*), którego hierarchia jest wzorowana na internetowym standardzie DOM (ang. *Document Object Model*).

DOM w wersji Domino to zbiór klas (dostępnych dla języków Java, LotusScript, Visual Basic oraz standardu CORBA i poprzez odpowiednie interfejsy także dla języka C++) dotyczących obsługi wszystkich elementów aplikacji.

Domino – tworzenie aplikacji

LotusScript

Caołść klas podzielono na dwie grupy:

- tzw. **grupę interfejsu użytkownika** (ang. *front-end*) — którą stanowią klasy pozwalające na manipulowanie aktualnym interfejsem użytkownika aplikacji; są one przeważnie używane do obsługi zdarzeń systemu i pozwalają na dostęp do elementów, na których aktualnie pracuje użytkownika (np. do elementów aktualnie otwartego dokumentu);
- tzw. **grupę interfejsu bazy danych** (ang. *back-end*) — która zawiera klasy używane do obsługi danych Domino; klasy tej grupy nie zawierają obsługi zdarzeń lub elementów obsługi interfejsu użytkownika, ich przeznaczeniem jest dowolne manipulowanie zawartością baz danych i ich obsługą (poprzez np. sterownie listami kontroli dostępu).

Domino – tworzenie aplikacji

LotusScript

W ramach jednego programu można dowolnie stosować obiekty obu grup;

zależności pomiędzy klasami DOM są ujęte w hierarchii obiektów (ang. *object hierarchy*).

Domino – tworzenie aplikacji

LotusScript

```
Dim db       As New NotesDatabase("Server" , "db.nsf");
Dim dc       As NotesDocumentCollection
Dim doc      As NotesDocument
Dim item     As NotesItem
Dim subject  As String
Set dc = db.AllDocuments
Set doc = dc.GetFirstDocument()
While Not(doc Is Nothing)
   Set item = doc.GetFirstItem("Subject")
   subject = item.text
   Set doc = dc.GetNextDocument(doc)
Wend
```

Przykład. LotusScript: skrypt pobierający wartość pola "Subject" ze wszystkich dokumentów bazy danych "db.nsf" – ilustracja zależności pomiędzy obiektami DOM.

Każdy obiekt środowiska Domino posiada listę skojarzonych z nim zdarzeń jakie mogą wystąpić w systemie; przykładowo przycisk akcji posiada m.in. zdarzenie naciśnięcia (ang. *click*), które jest aktywowane, gdy użytkownik użyje przycisku; innym przykładem zdarzenia może być otwarcie przez użytkownika dokumentu — zdarzenie otwarcia (ang. *open*).

Każde zdarzenie środowiska może zostać dowolnie oprogramowane przy użyciu LotusScript, JavaScript (w przypadku zdarzeń obsługiwanych także przez przeglądarki internetowe) lub języka @formuł. System tych zdarzeń nazwano **modelem zdarzeń** (ang. *event model*). Lista zdarzeń wybranego elementu projektowego znajduje się w panelu programowym w liście informacyjnej; po wybraniu żądanego zdarzenia z listy, kod akcji jaka ma zostać wykonana po jego zaistnieniu należy umieścić w obszarze kodowania.


```
Sub Postopen(Source As Notesuidocument)
   Dim session As New NotesSession
   If source.EditMode Then
      Call source.FieldSetText("Creator",
                               sesion.CommonUserName)
   End If
End Sub
```

Przykład. LotusScript: obsługa zdarzenia otwarcia dokumentu (ang. *postopen*); jeżeli otwierany dokument jest nowym dokumentem, to w pole o nazwie "Creator" wpisywana jest nazwa użytkownika.

Module

{ Dim | Private | Public } *varName* **[As** *dataType* **]**

```
Dim string1$
Private string2$
Public string3$
Dim string4 As String
Private string5 As String
Public string6 As String
```

Procedure

{ Dim | Static } *varName* **[As** *dataType* **]**

```
Dim string 7$
Static string8$
Dim string9 As String
Static string10 As String
```

User-defined data type

varName **As** *dataType*

string11 As String

Class

[Private | Public] *varName* **As** *dataType*

```
string12 As String
Private string13 As String
Public string14 As String
```

[Public | Private] [Static] Function *functionName* **[(** *parameters* **)] [As** *dataType* **]**

Data type	Value range	Size
Integer Signed short integer	-32,768 to 32,767	2 bytes
Long Signed long integer	-2,147,483,648 to 2,147,483,647	4 bytes
Single Single-precision floating-point	-3.402823E+38 to 3.402823E+38	4 bytes
Double Double-precision floating-point	-1.7976931348623158+308 to 1.7976931348623158+308	8 bytes
Currency Fixed-point integer scaled to 4 decimal places	-922,337,203,685,477.5807 to 922,337,203,685,477.5807	8 bytes
String	Limited by available memory	2 bytes/ character

LotusScript

LotusScript zawiera również funkcje synchronizujące, pozwalające na obsługę sekcji krytycznej dostępnej np. dla agentów aplikacji Domino.

Funkcje synchronizujące implementują system blokad.

- **CreateLock** – creates a lock
- **DestroyLock** – destroys a lock
- **CodeLock** – allows an agent to enter a critical section, blocking other cooperating agents from entering
- **CodeUnlock** – unblocks a section, allowing the next cooperating agent to enter the section
- **CodeLockCheck** – checks the existence of certain lock
- **Sleep** – goes to sleep for a specified amount of time

Inne Możliwości Programowania

Część dostępnych zdarzeń w modelu zdarzeń Domino może być oprogramowana przy użyciu języka JavaScript.

Pewne elementy DOM (ang. *Domino Object Model*) mogą także być dostępne dla skryptów JavaScript; odbywa się to poprzez system mapowania elementów Domino DOM na standard DOM (ang. *Document Object Model*) — wówczas elementy projektowe Domino są dostępne w postaci kodu HTML (lub jako aplety w nim osadzone).

Zastosowanie tego języka jest możliwe w przypadku gdy aplikacja Domino jest uruchamiana przy użyciu przeglądarki internetowej oraz kiedy użytkownika wykorzystuje program Notes.

```
if (frmKsiegaOdmow_Imie.value == '')
  { alert('Proszę podać Imię Pacjenta!'); return false; }
```


Inne Możliwości Programowania

Środowisko umożliwia także programowanie aplikacji w języku Java; język ten można wykorzystać do implementacji agentów, apletów, serwletów oraz autonomicznych aplikacji, które współdziałają ze środowiskiem Domino.

Aby korzystać z języka Java do obsługi środowiska Domino niezbędny jest odpowiedni zestaw bibliotek, który jest dostępny w Internecie [Internet: http://www-10.lotus.com/ldd/down.nsf].

Zdalny dostęp do środowiska Domino dla aplikacji i apletów Java jest możliwy dzięki użyciu standardu CORBA Standard ten zapewnia warstwę pośredniczącą pomiędzy klientami, którzy zdalnie wywołują metody obiektów środowiska. Do komunikacji używany jest protokół IIOP (ang. *Internet Inter-ORB Protocol*), który korzysta z internetowych protokołów TCP/IP.

Inne Możliwości Programowania

Dodatkowo producent udostępnia pakiet dynamicznie łączonych bibliotek (ang. *dynamic-link library*) o nazwie LSX (ang. *Lotus Software eXtension*), które umożliwiają tworzenie aplikacji C++ dla środowiska Domino.

Klasy dostępne w ramach LSX są identyczne jak klasy LotusScript i pozwalają również na zdalny dostęp do serwerów środowiska.

Bibliografia

- *LotusDomino. Administering the Domino System vol. 1,2*, Lotus Development Corporation, 1999 r.
- *Lotus Notes Step by Step*, Lotus Development Corporation, 1999 r.
- Collins F., Morrison D., Nielsen S., Serpola S., Strobl S., *Lotus Domino Release 5.0: A developer's Handbook*, International Technical Support Organization, 1999 r.
- Internet: www.redbooks.ibm.com
- Internet: www.notes.net
- Internet: www.pl.lotus.com