STUDIA INFORMATICA 2003
Volume 24 Number 2B (54)

Andrzej KALIŚ, Jacek GRUBER
Politechnika Wrocławska, Wydziałowy Zakład Informatyki

PROGRAMOWANIE WWW Z WYKORZYSTANIEM DELPHI

Streszczenie. Najważniejszą cechą programistycznego środowiska *Delphi* jest jego produktywność, czyli prostota i szybkość tworzenia programów dla platformy *Windows* oraz *Linux*. W pracy podjęto próbę przedstawienia możliwości *Delphi* wspomagających tworzenie aplikacji *WWW*.

Słowa kluczowe: Delphi, protokoły sieciowe, WebBroker, WebSnap, usługi sieciowe, IntraWeb.

WWW PROGRAMMING USING DELPHI

Summary. The most important feature of the Delphi environment is it's productivity, which we can understand as easy and rapid application development for Windows and Linux. In this paper there was an attempt of showing the possibilities of Delphi in WWW application development.

Keywords: Delphi, protocols, WebBroker, WebSnap, Web Services, IntraWeb.

1. Wprowadzenie

Firma *Borland* udostępniła programistom i projektantom już siedem wersji środowiska programistycznego *Delphi* z coraz większymi możliwościami funkcjonalnymi i nowocześniejszymi technologiami. Jednak dopiero wersja *Delphi 3* z 1997 roku posiadała elementy interesujących nas tutaj mechanizmów do programowania *Internetu* oraz technologie: *COM*, *ActiveX* czy wielowarstwowe aplikacje bazodanowe. Przełomowy dla rozwoju *Delphi* był rok 1998, w którym ujawniła się nowa strategia firmy *INPRISE* (*Borland* zmienił nazwę) zorientowana na programowanie systemów dla przedsiębiorstw z wykorzystaniem technologii *CORBA*, *MIDAS*, *MTS* czy *DCOM* [1]. Późniejsze wersje pakietu koncentrowały wysiłek

programisty na tym, co miał wytworzyć, a nie na sposobie pisania oprogramowania. Dużym przełomem technologicznym były rozwiązania zastosowane w *Delphi 6*, całkowicie zgodne z *Kylix* (*Delphi* dla *Linuxa*). Zwiększono rolę *XML* w programowaniu serwisów sieciowych, (*SOAP*, *WSDL*, *UDDI*). Znacznie uproszczono zasady tworzenia aplikacji *WWW* (*WebBroker*, *WebSnap*) umożliwiając nadzorowanie aplikacji w środowisku *Delphi* (*Web App Debugger*) oraz udostępniając wielu kreatorów (*wizards*). Do palety komponentów włączono szeroko znane pakiety firmy *Nevrona* pod nazwą *Internet Direct* (w skrócie *Indy*). Ostatnia, siódma wersja *Delphi* zawiera szereg unowocześnień upraszczających programowanie sieci *WWW*, w tym obsługę serwera *Apache 2*, nową przeglądarkę rejestrów *UDDI* do importowania dokumentów *WSDL*, a przede wszystkim dość rewolucyjną technologię *IntraWeb*.

2. Internetowe zastosowania Delphi

Delphi udostępnia szeroki wachlarz mechanizmów, wspomagających tworzenie aplikacji internetowych – mechanizmy te można podzielić na następujące kategorie:

- obsługę protokołów sieciowych (*TCP/IP*, *FTP*, *POP3*, *SMTP* i inne),
- wspomaganie tworzenia rozszerzeń serwerów *WWW*,
- wspomaganie usług sieciowych (*SOAP*, *WSDL*, *UDDI*).

Łatwość wykorzystania *Delphi* wynika z jego komponentowej budowy. Prostota programowania polega na dokładaniu do programu potrzebnego zestawu komponentów, ustawianiu ich właściwości (w *Object Inspectorze*) oraz na dodaniu zwykle niewielkiej ilości własnego kodu napisanego w *Object Pascalu*. Programista skupia swoją uwagę na tym, co aplikacja ma robić, nie interesując się wieloma szczegółami technicznymi, przykładowo związanymi z implementacją protokołów sieciowych albo rozbiorem syntaktycznym stron *HTML*, bądź *XML*. Obsługa baz danych oraz podtrzymywanie architektury klient-serwer to kolejne mocne strony *Delphi*, ponieważ architektura klient-serwer (*DataSnap*) jest niejako integralną częścią aplikacji sieciowych oraz *Internetu* jako takiego.

3. Obsługa protokołów sieciowych

Internet Direct (*Indy*) jest zestawem komponentów [2], które dostępne są zarówno w *Delphi 6*, jak i w *Kylix*, ponieważ po instalacji w obu środowiskach są one widoczne w palecie komponentów. Zestaw był także dostępny w *Delphi 5*, ale na osobnej płycie *CD-ROM* (tzw. *Companion Disc*). Pakiet ten rozprowadzany jest również na zasadach *Open*

Source i można pobrać najnowsze jego wersje z witryny firmy *Nevrona* (www.nevrona.com). *Indy* składa się z komponentów umożliwiających wsparcie implementacji protokołów po stronie aplikacji klienckiej oraz serwera. Jedną z najważniejszych cech *Indy* jest zgodność z pozostałymi narzędziami programistycznymi, takimi jak wspomniany już *Kylix, oraz Delphi 4* i *5*, a także *C++ Builder* wersji *4* i *5*. *Indy* obsługuje wielobajtowy zestaw znaków (*MBCS*) włączając w to języki azjatyckie. Implementacja protokołów przesłania nam większość szczegółów koniecznych do ich prawidłowej realizacji, zatem programista nie musi ich znać. Wskazana jest jednak ich ogólna znajomość i warto zapoznać się z oficjalnymi dokumentami *RFC* (*Request for Comments*) publikowanymi przez organizację *IETF* (*Internet Engineering Task Force*) – patrz http://www.ietf.org lub http://www. w3c.org.

3.1. Komponenty aplikacji klienta

tIdTCPClient – komponent ten jest wykorzystywany do uzyskania podstawowej komunikacji w oparciu o protokół *TCP* (*RFC 793*). Jego podstawowymi właściwościami są: *Port*, poprzez który komunikujemy się z serwerem, oraz *Host*, któremu nadajemy wartość równą *URL* serwera. Jest to bazowy komponent do implementacji innych protokołów wykorzystujących *TCP*, takich jak **tIdSMTP** oraz **tIdFTP**.

tIdFTP – jest rozwiniętym klientem *FTP* (*File Transfer Protocol*, *RFC 959*) obsługującym transfery aktywne i pasywne. Są tu dostępne wszystkie polecenia *FTP*.

tIdHTTP – komponent implementuje protokół *HTTP* w zgodzie z wersją *1.0* (*RFC 1945*) oraz 1.1 (*RFC 2616*, *RFC 2660*). Obsługuje *proxy* i autoryzację klienta.

tIdPOP3 – jest to klient *POP3* obsługujący operacje *Post Office Protocol* (*RFC 1939*).

tIdSMTP – to klient *SMTP* (*Simple Mail Transfer Protocol*, *RFC 821*, *1869*, *2554*).

3.2. Komponenty aplikacji serwera i komponenty pomocnicze

Serwerem nazywamy aplikację, która wykonuje żądania innych aplikacji nazywanych klientami. Do najważniejszych z nich należą:

tIdTCPServer – służy do budowy serwerów *TCP*. Jest podstawą do implementacji innych protokołów wykorzystujących *TCP* (np. **tIdHTTPServer** czy **tIdNNTPServer**).

tIdHTTPServer – jest używany do zbudowania serwera HTTP i innych serwerów pracujących w oparciu o protokół *HTTP*.

tIdIRCServer – komponent dostarcza podstawowych elementów do zbudowania serwera *IRC* (*Internet Relay Chat*) znaną jako „*chat*".

Nie mniej ważne od wyliczonych wyżej komponentów zawiera zakładka *Indy Misc* z palety komponentów. Wykorzystuje się je do wsparcia procesu tworzenia aplikacji klientów

i serwerów (między innymi komponenty kodujące, szyfrujące czy zarządzające wątkami). Ważnym spośród nich jest **tIdMessage**, który reprezentuje internetowy format wiadomości (*RFC 822*, *1036*) ze wszystkimi związanymi częściami i nagłówkami (*RFC 2045* do *2049*). Obiekty tej klasy są także wykorzystywane przez **tIdSMTP**, **tIdPOP3** oraz **tIdNNTP**.

4. Tworzenie rozszerzeń serwerów

Sieć *WWW* była do niedawna zorientowana jedynie na wymianę gotowych dokumentów. Z czasem możliwości serwerów *WWW* zostały rozszerzone. W takich rozwiązaniach klient (zazwyczaj przeglądarka) wysyła do serwera żądanie. Serwer przekazuje je do aplikacji rozszerzeń, która je obsługuje i udostępnia wynik przetwarzania w postaci dynamicznie wykreowanej strony *HTML* odsyłanej do klienta. Współpraca serwera ze skryptami odbywa się poprzez ściśle określony interfejs. W środowisku *Delphi* możemy wygenerować następujące rozszerzenia: *CGI* (*Common Gateway Interface*), *ISAPI* (*Internet Server API*), *NSAPI* (*Netscape server API*), *Apache DSO*, *Apache 2 DSO* oraz *ASP*. Technologie *WebBroker* oraz WebSnap firmy Borland pozwalają ukryć różnice implementacyjne między tymi interfejsami.

4.1. Technologia WebBroker

Znajomość podstawowych klas środowiska *Delphi* umożliwia zrozumienie zasad tworzenia aplikacji *WWW*. Klasy odzwierciedlają podstawową architekturę aplikacji.

Klasa tWebDispatcher.

Nazwa „dyspozytor" trafnie oddaje przeznaczenie tej klasy. Do niej są kierowane wszystkie komunikaty żądań *HTTP*. Przekierowuje ona komunikaty do odpowiednich metod obsługi zdarzeń (**tWebActionItem**) i po przetworzeniu odbiera zawartość, którą zwraca jako odpowiedź do klienta. Po wybraniu typu realizowanego rozszerzenia *Delphi* tworzy formularz *WebModule* (moduł *WWW*), w którym zawiera się już klasa dyspozytora. Jednakże podstawowymi elementami modułu są jego akcje, z których każda określa reakcje na pojedyncze żądania klienta. Pojedyncze akcje reprezentowane są przez klasę **tWebActionItem** i mogą być definiowane w edytorze akcji modułu *WWW*. Właściwość *PathInfo* określa związek danej akcji z jednym z członów specyfikacji *URL*, zaś w *MethodType* definiujemy metodę żądań, czyli *mtGET*, *mtHEAD*, *mtPOST* czy *mtPUT*.

Klasa tWebModule.

Formularz *WWW* oparty na **tWebModule** pozwala programiście, poprzez jego inspektora, dotrzeć do listy akcji oraz do poziomu zdarzeń. Na formularzu możemy umieścić dowolne niewidoczne komponenty, w szczególności komponenty *Indy*, np. **tIdSMTP**, by wysłać list.

Klasa tWebActionItem.

Obiekt jest wykorzystywany do sformułowania odpowiedzi na zapytanie zawarte w wiadomości *HTTP* (kojarzy ścieżkę w *URL* i typ metody z właściwościami *PathInfo* oraz *MethodType*). Istotną możliwością jest „podpięcie" do akcji jej producenta odpowiedzi (*Producer*). Jeżeli tego nie zrobimy, wywoływana jest metoda obsługi zdarzenia *OnAction*. Polecenia i odpowiedzi są kierowane do producenta lub obsługi *OnAction* w postaci obiektów **tWebRequest** i **tWebResponse**. Pierwszy jest podstawową klasą dla wszystkich obiektów reprezentujących żądania klienta, a w drugim określamy odpowiedź. Obiekty pozwalają programiście rozwiązywać problem bez wnikania w realizację protokołu *HTTP* lub *API* serwera.

Klasą podstawową dla obiektów producentów jest **tCustomContentProducer**. Ich zadaniem jest przygotowanie zawartości odpowiedzi przesyłanej do klienta (w postaci *HTML*, *XML*, *WML* lub *MIME*). Najczęściej jest to dynamicznie utworzona zawartość strony *HTML*. Zasada pracy producentów zawartości stron (**tPageProducer**) opiera się na poszukiwaniu w szablonie strony *HTML* niewidocznych znaczników *<#tag>*. Po napotkaniu znacznika wyzwalane jest zdarzenie *OnHTMLTag*, w którym realizujemy zastąpienie niewidocznych znaczników konkretną treścią. Jeżeli chcemy włączyć do stron *HTML* raporty tabelaryczne zawierające rekordy pochodzące ze zbiorów danych bądź stanowiących wynik zapytania *SQL*, to posługujemy się komponentami **tDataSetTableProducer** lub **tQueryTableProducer**.

4.2. Technologia WebSnap

W *Delphi 6* wprowadzono technologię *WebSnap*, która jest wynikiem integracji technologii *WebBroker* oraz *InternetExpress* (odpowiadające im zakładki *Internet* i *InternetExpress* pozostawiono niezmienione). Dzięki temu przenoszenie istniejących aplikacji na „platformę" *WebSnap* nie jest skomplikowane. Więcej informacji o różnicach między technologią *WebBroker* a *WebSnap* można znaleźć w [3]. Najważniejsze w tej technologii jest to, że pozwala ona stworzyć warstwę prezentacyjną bez żadnych kontrolek *ActiveX* i czynności rekonfiguracyjnych po stronie klienta.

Zasadniczymi komponentami technologii są tzw. "adaptery" (**tAdapter, tPagedAdapter tDataSetAdapter** i inne), które umożliwiają wykorzystanie technik skryptowych (*JavaScript*, *VBScript*) w tworzonych aplikacjach. Dla przykładu, **tDataSetAdapter** wykorzystuje język skryptowy do stworzenia strony prezentującej zawartość bazy danych. Programista powinien jednak znać podstawy tych języków skryptowych, przynajmniej *JavaScript* (*ECMA-262*, http://www.ecma.ch), aby zrozumieć zasady wypełniania treścią stron *WWW* (choć „standardową", aczkolwiek profesjonalną aplikację, tworzymy tu bez jednej linijki własnego kodu). Druga część komponentów *WebSnap* składa się z tzw. „dyspozytorów" (**tPageDispatcher, tAdapterDispatcher**) organizujących przydzielanie żądań HTTP i odpowiedzi na nie

w zależności od zadanej „logiki biznesowej". Trzecia grupa komponentów to tzw. „spisy wartości" (**tStringsValuesList**, **tDataSetValuesList** czy **tWebUserList**), które pozwalają bardzo prosto włączać do stron *WWW* linie tekstu, dane albo listy wyliczane. Ostatnia grupa komponentów, tzw. „producentów" (**tXSLPageProducer**, **tAdapterPageProducer** i inne), potrafi generować dokumenty *HTML* (*XML*) na podstawie różnych stron *HTML* (*XML*). Wymienione komponenty są kładzione na jeden wspólny moduł, **tWebSnapDataModule** bądź **tWebAppPageModule**.

Do istotnych nowości w technologii *WebSnap* należy możliwość śledzenia aplikacji bez uruchamiania jej na serwerze, a także podgląd stron bez uruchamiania przeglądarki.

5. Usługi sieciowe

Pojawienie się różnorodnych przenośnych urządzeń spowodowało, że wiele komercyjnych aplikacji *WWW* obsługuje nie tylko protokół *HTTP*, ale także *WAP* (*Wireless Access Protocol*). Urządzenia te potrafią też zwracać odpowiedź nie tylko w postaci strony *HTML*, ale i w standardzie *WML* (*Wireless Markup Language*). Próby integracji obu tych podejść spowodowały powstanie nowej technologii *WWW*, opierającej się na wykorzystaniu protokołu *SOAP* (*Simple Object Access Protocol*) oraz języka *XML* (*Extensible Markup Language*). Aplikacje wymieniają między sobą informacje w formacie *XML* za pośrednictwem protokołu komunikacyjnego *HTTP* albo *HTTPS*. Jednakże podstawą „usługowo-zorientowanego *WWW*" są usługi sieciowe (*web services*), czyli zbiór obiektów, których metody mogą być wywołane poprzez *Internet*. Informacje o dostępnych metodach zawarte są w dokumentach *WSDL* (*Web Service Description Language*), a dla poszukiwań serwisów wykorzystuje się rejestry *UDDI* (*Universal Description Discovery and Integration*). Sprawia to, że możliwa staje się współpraca między różnymi platformami sprzętowymi i systemowymi.

W *Delphi* usługom sieciowym dedykowana jest zakładka *WebServices* z palety komponentów bądź kreatora do tworzenia *SOAP Server Application*. Pozwala to implementować serwery usług sieciowych dla „upublicznionej" usługi sieciowej oraz aplikacje klienckie korzystające z tych usług. Kreator umieszcza w module sieciowym trzy podstawowe elementy: **tHTTPSoapDispatcher** (dyspozytor komunikatów *SOAP*), **tHTTPSoapPascalInvoker** (interpretuje komunikaty i dokonuje wywołania związanego z nim interfejsu wywoływalnego), oraz **tWSDLHTMLPublish**, którego zadaniem jest utworzenie dokumentu *WSDL* zawierającego informację o metodach udostępnianych przez usługę. Jedyną rzeczą, którą programista powinien znać, jest definiowanie i implementowanie interfejsu wywoływalnego, gdy tworzymy serwer usługi, a także sposób wykorzystania usług, gdy je zaimportujemy ze zdalnego

dokumentu *WSDL*. Sam import jest banalny, bo wykonuje go za nas kreator *WSDL Importer*. Dokładniejsze omówienie usług sieciowych w środowisku *Delphi* można znaleźć w [3, 4].

6. IntraWeb

IntraWeb jest zestawem komponentów pozwalających tworzyć aplikacje *WWW* po raz pierwszy udostępnionym w *Delphi 7*. Firma *AtoZed Software*, wykonawca tych komponentów, udostępniała je również dla *Delphi 5* i *6*, *Kylix 2* i *3* oraz *C++ Builder 5* i *6*. Niezwykłość tej nowej technologii nie polega na tym, że za jej pomocą możemy tworzyć samodzielne aplikacje *WWW*, które generują strony z *HTML* (czy *JavaScript*). Również nie na tym, że możemy tworzyć różne rozszerzenia serwera (takie jak *ISAPI DLL* czy *Apache*), czy wspierać usługi sieciowe, oraz nie na tym, że możemy utworzyć samodzielną aplikację, która jest pełnowartościowym serwerem nasłuchującym żądań *HTTP*.

Jej niezwykłość to umożliwienie tworzenia aplikacji *IntraWeb* („weblikacji") w sposób nie różniący się zbytnio od tworzenia typowej aplikacji *Delphi*. Zatem nawet niedoświadczony w tworzeniu aplikacji *WWW* programista może wytworzyć funkcjonującą aplikację nie znając szczegółów współpracy stron *HTML* z protokołem *HTTP* i nie ucząc się języka *JavaScript* do tworzenia interfejsu użytkownika. Więcej informacji o *IntraWeb* można zaczerpnąć na stronie firmy http://www.atozedsoftware.com/intraweb.

7. Podsumowanie

Mamy nadzieję, że artykuł daje wystarczające podstawy merytoryczne tym osobom, które rozważają możliwość zastosowanie Delphi w swoich pracach (nie tylko w zakresie programowania sieciowego). Jednak poleca się wyniki pracy [5], w której można znaleźć szczegółowe porównanie możliwości Delphi z takimi narzędziami, jak: IBM WebSphere Studio, Microsoft Visual Studio .NET oraz BEA WebLogic Workshop.

LITERATURA

1. Pacheco X., Teixeira S.: DELPHI 6 Vademecum profesjonalisty. Helion, Gliwice 2002.
2. Jensen C., Anderson L.: Building Kylix Applications. Osborne/McGraw-Hill, 2001.
3. Delphi 6 for Windows Developer's Guide. Borland Software Corporation, 2001.

4. Polistchuck D.: Mabaging Sessions with Delphi 6 Web Services. http://community.borland.com/article/0,1410,27575,00.html.
5. Web Services Develoment Tools Compared. A Borland White Paper, September 2002.

Recenzent: Dr inż. Wojciech Mikanik

Wpłynęło do Redakcji 28 kwietnia 2003 r.

Abstract

In this paper there was shown the use of Delphi integrated development environment (IDE) in designing, developing, testing, debugging, and deploying Internet applications, allowing rapid prototyping and a shorter development time. Delphi lets you bring its full power to the Web in the following ways: by encapsulating the HTTP in easily accessible objects, by providing and application framework around the application programming interfaces of the most popular and powerful Web servers, by providing a Rapid Application Development approach to building Web server extensions. In this work there was described a wide range of tools for writing Web server applications, including: the Internet Direct, set of open source socket components that consists of clients, servers, and support components, the Web Broker architecture, with which you can create cross-platform server applications; WebSnap, with which you can design Web pages in a GUI environment; support for working with XML documents; and an architecture for using SOAP-based Web Services. The Intra-Web technology was also mentioned which is a revolutionary new way to create your web-based applications. IntraWeb allows you to create your applications („weblication") in a true RAD manner by dragging and dropping components on a "web form" and defining events and setting properties the same way that you would in a normal Delphi application. The components that implement many of these features are not available in all editions of Delphi.

Adresy

Andrzej KALIŚ: Politechnika Wrocławska, Wydziałowy Zakład Informatyki, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Polska, A.Kalis@ci.pwr.wroc.pl .

Jacek GRUBER: Politechnika Wrocławska, Wydziałowy Zakład Informatyki, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Polska, J.Gruber@ci.pwr.wroc.pl .