STUDIA INFORMATICA 2003
Volume 24 Number 4 (56)

Jarosław FRANCIK
Politechnika Śląska, Instytut Informatyki

ON THE USE OF DYNAMIC HTML IN STANDALONE APPLICATIONS[1]

Summary. Web applications have emerged in recent years as a mainstream in software development. In the same time single-station applications appeared as a separate, totally incompatible world. The paper presents a simple solution for development of standalone applications with a "weblication" look-and-feel.

Keywords: application programming, standalone application, serwer-side scripting, dynamic HTML, databases

O WYKORZYSTANIU DYNAMICZNYCH STRON WWW W APLIKACJACH NIESIECIOWYCH

Streszczenie. Aplikacje internetowe stały się w ostatnich latach dominującym kierunkiem w rozwoju oprogramowania. W tym samym czasie aplikacje jednostanowiskowe pozostały w oddzielnym, zupełnie niekompatybilnym świecie. Artykuł przedstawia proste rozwiązanie tworzenia jednostanowiskowych aplikacji spełniających standardy wyznaczone przez „weblikacje".

Słowa kluczowe: tworzenie aplikacji, aplikacje jednostanowiskowe, techniki po stronie serwera, dynamiczny HTML, bazy danych

1. Introduction

The history of World Wide Web started with HTML [1] pages which were completely static. Soon after the technology was introduced to the business, behavior characteristic for regular applications has emerged to be an important demand. This initiated the dynamic aspect

[1] The work supported by the Fund of Statutory Research of the Institute of Informatics, Silesian University of Technology.

of the web. Introducing server-side technologies appeared to be a real breakthrough [eg. 2, 3]. This enabled full integration of the Internet and database technologies, and consequently started dynamic development of web applications in e-business solutions, including even enterprise management. Eventually, so called weblications became a *de facto* industrial standard in software development.

There are at least two major factors of the world wide success of the weblications. One of them is obviously their unlimited availability. The other factor is not connected with Internet itself, however in practice is limited to the Internet applications: that's the user interface, with its hypertext link based navigation, strong graphics/multimedia support, dynamic contents and easy database connectivity.

While weblications became more and more the mainstream in the software industry, standalone (single-station) applications seem to remain out of the beaten track. Of the two factors of success mentioned above, the world-wide availability is something they in fact don't need by definition. However, the high standard of the web user interface is a feature in which most standalone applications seem to be rather resistant, exploiting rather an old schema of document-centric applications with standardized dialog boxes and controls.

Introducing an HTML-based interface to standalone applications is a strong demand, and in fact it is already achieved. Browsing web pages locally makes no problem, even if they contain dynamic elements based on client-side scripting. Some applications include web-like views in their interfaces; a nice example could be a new version of Microsoft Outlook Express along with many Windows XP system applets, in which hypertext links replace traditional pushbuttons and menu options. Control items, text and graphics freely mixed in the same area make their new look-and-feel. All this seems to lead to a paradigm shift towards web user interface (WUI) that should be soon ubiquitous.

From the point of view of a software developer, an ability to display HTML pages with client-side dynamic elements is highly insufficient. weblications would never evolve without server-side scripting that enables creating pages dynamically, on demand, depending on the current context. This context is usually defined by previous user actions as well as system database contents. This functionality is normally supported by web servers, like IIS or Appache. However installing such a server just for needs of a standalone application would be like breaking a butterfly on the wheel. The problem is that there is no obvious method to do it otherwise[1]. In this paper a relatively simple solution is proposed. Its only technical requirement is a standard web browser control, available in Windows system.

[1] Some walk-around of the problem could also be Java applets or ActiveX controls.

2. Standalone HTML Application Requirements

An HTML application as understood in this paper is an application that displays HTML pages as its user interface. The goal is not just creating a couple of web pages however; what is the clue is making the application and the web pages work together to obtain results impossible to obtain otherwise. The application is the executive part of the system, very much like the server-side scripting machinery in a traditional weblication. It bears functionality typical in local applications, like file, document, database or input-output operations.

Among the requirements for this type of application, the most important are:

1) weblication look-and-feel, ie. navigation based on hypertext links, control items, text and extended graphics/multimedia contents sharing the same place;
2) ability of the application to prepare web pages on demand, dynamically;
3) flexible database connectivity;
4) ability to use web forms;
5) ability of the application to collect data posted or got from the page;
6) ability of the web page to initialize actions executed by the application (executive scripts);
7) ability of the application to initialize actions executed by the web page; an ability to invoke web page scripts should be enough.

3. Simple Solution for Standalone Server-Side Scripting (6S)

6S is a simple architecture framework that meets all the requirements enumerated in the previous section. The only external component is a typical ActiveX Web Browser control, available in Microsoft Windows system. The solution is so simple that may be easily implemented in any language, only applying directions given below. However a simple C++, MFC based library is available at [4].

A notice should be done that the name of 6S framework may be a little confusing: it obviously is not a server-side solution, as it is intended to be used locally only. Thus, the contradiction in its name is just a pun.

3.1. The Application Outline in 6S

The only requirement is that an application should present an ActiveX Web Browser control. This control typically implements *IWebBrowser2* interface [5] and is available in all Windows system since Internet Explorer 4.0 (ProgId: *Shell.Explorer.2*). MFC applications may use a standard *CHtmlView* object. Visual Basic applications may use *Microsoft Internet Controls / WebBrowser* component.

Applying Web Browser control makes it possible to display output in form of HTML pages (requirement no. 1).

3.2. URL Interception and Dynamic Pages

The Web Browser control, through its *DWebBrowserEvents2* sink interface, raises an event *BeforeNavigate* (this event is directly available in both MFC and VB applications). It is raised just before browser navigates to another page. It may be used by the application to intercept the URL. After checking, what the URL is, the application may prepare the page or even create it (requirement no. 2). Of course, many applications will produce web pages depending on some database contents (requirement no. 3). Some useful library tools in C++ that help to modify HTML files so that to include actual data are available at [4].

Application of XML/XSL files [6] makes preparing pages containing data extracted from a database even easier. After the URL is detected this is only an XML file that is generated on the base of the database contents. The whole visual definition of the page is contained in an XSL [7] file and does not need to be modified.

3.3. Using Web Forms

Web forms (requirement no. 4) can be easily placed on any HTML page. Collecting data posted or got from a form or any other page (requirement no. 5) is as easy as URL interception presented in the previous subsection. The application should detect the URL that is a target of a given form. Data sent with the GET method are then available as a part of this URL after the "?" character. Data posted with the POST method are available directly as one of the *BeforeNavigate* event parameters.

3.4. Invoking Executive Scripts

Executive Scripts (requirement no. 6) are functions invoked in the application, on demand of the web page. To support such scripting once more URL interception mechanism should be used. After a specified URL string is detected, the application may perform any demanded action. Parameters may be sent with GET or POST method as described above, or even coded in any other (non-standard) form in the URL string – as it is delivered to the application verbatim. A good idea is to start all the *hrefs* intended to invoke scripts e.g. with the letters "Invoke:"

3.5. Invoking Web Page Scripts

Invoking web page scripts from the application code (requirement no. 7) involves acquisition an *IDispatch* interface that is created dynamically by the web page, and which supports (contains) all the scripting functions available in the HTML code. It involves several stages that should be executed as follows:

- get the *IWebBrowser2::document* attribute;
- query for the *IHtmlDocument* interface of the *document* object;
- get the *script* attribute of the *document* object;
- the *script* object supports an *IDispatch* interface that may be directly used to call a script function (it implements all the script functionality contained in the current page).

In a similar way other interfaces may be acquired to directly manipulate the page elements from the level of the application code.

3.6. Sample Code

A short sample routine in VB shows how a *href="Invoke:Test()"* link in the web page may trigger an application to collect additional data and call back a script routine in the page, transferring these additional data to it (fig. 1).

```
Private Sub WebBrowser1_BeforeNavigate2(ByVal pDisp As Object,
           URL As Variant, Flags As Variant, TargetFrameName As Variant,
             PostData As Variant, Headers As Variant, Cancel As Boolean)
If Left(URL, 7) = "invoke:" Then     ' invoke any executive script?
  If Mid(URL, 8, 5) = "Test(" Then ' invoke the Test script?
    Cancel = True                   ' cancel navigation to this URL
    Dim doc As HTMLDocument
    Set doc = WebBrowser1.Document ' get the HTMLDocument
    Dim script
    Set sc = doc.script             ' get the script
    a = 5                           ' collect application-dependent data
    sc.test (a)                     ' call the script function
  End If
  ' Check for the other scripts to invoke...
```

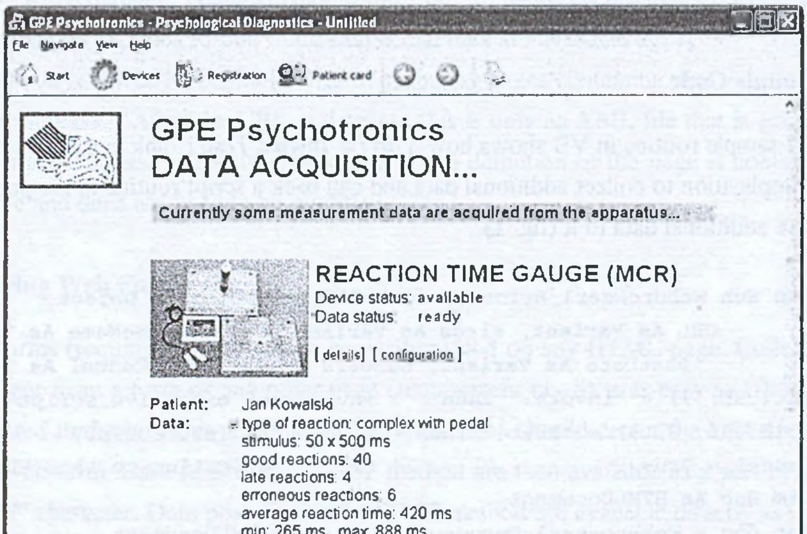
Fig. 1. Sample routine in a 6S application
Rys. 1. Przykładowa procedura w aplikacji wykonanej w ramach 6S

The same code written in C++ would be a bit longer (*QueryInterface* and *IDispatch* methods would be called directly), but not much more complicated. Some useful examples of the code may be found at [4].

4. Psychotronics: a Sample Application

Psychotronics [8] is a standalone application intended to control a set of electronic equipment for psychological diagnosis, for example reaction time gauges. Its primary requirement is acquisition, presentation and storing measurement data from the apparatus. The application is intended to be used in a psychologist's surgery, just near the apparatus it controls. Thus, it is a typical example of a standalone application, for which distant access would make no sense.

The application uses a serial port to acquire data and a database to store them. The user interface applies a series of HTML and XML/XSL pages, most of them generated dynamically. A screenshot with a view generated on the basis of the data acquired from one of the gauges is shown in fig. 2. The Psychotronics project has been implemented in C++.



GPE Psychotronics
DATA ACQUISITION...

Currently some measurement data are acquired from the apparatus...

REACTION TIME GAUGE (MCR)
Device status: available
Data status: ready
[details] [configuration]

Patient: Jan Kowalski
Data: type of reaction: complex with pedal
stimulus: 50 x 500 ms
good reactions: 40
late reactions: 4
erroneous reactions: 6
average reaction time: 420 ms
min: 265 ms max: 888 ms
[more details]

Fig. 2. Psychotronics: a sample application built with 6S framework (originally in Polish, translated into English)

Rys. 2. Psychotronics: przykładowa aplikacja zbudowana na podstawie architektury 6S (pierwotnie w języku polskim, przetłumaczona na angielski)

5. Conclusion

Just several years ago a strong demand was to provide newly born Internet applications with the functionality that was obvious in standalone applications. When Internet browsing is the most natural way of utilization computers, current demands are apparently inverse: functionality and look-and-feel of the weblications have to become available for standalone applications. The presented 6S application framework is an easy-to-use conception to fulfill this demand.

The 6S scripting is not compatible with any server-side scripting standard. However converting the 6S applications into full weblications, if necessary, should be easier than in case of traditional standalone applications, even if rewriting all the scripts would be involved. Some method, maybe useful, however yet unexplored, could be converting 6S applications into CGI or ISAPI modules.

REFERENCES

1. HyperText Markup Language (HTML). World Wide Web Consortium: http://www.w3.org/MarkUp.
2. PHP: Hypertext Preprocessor. The PHP Group: http://www.php.net.
3. ASP.NET Web: The Official Microsoft ASP.NET Site: Home Page. Microsoft Corp.: http://www.asp.net.
4. 6S: The Simple Solution for Standalone Server-Side Scripting. http://www-zo.iinf.polsl.gliwice.pl/~jfrancik/6s.
5. Microsoft Developers Network MSDN.
6. The Extensible Markup Language (XML). World Wide Web Consortium: http://www.w3.org/XML.
7. The Extensible Stylesheet Language (XML). World Wide Web Consortium: http://www.w3.org/Style/XSL.
8. M. W. Korchut, Aparatura do diagnostyki psychologicznej (Apparatus for psychological diagnostics), Psychotronics Co: and EIEWIN Co.: http://www.eiewin.com.pl.

Recenzent: Dr inż. Maciej Bargielski

Wpłynęło do Redakcji 13 maja 2003 r.

Omówienie

Jeszcze kilka lat temu poważnym wyzwaniem dla twórców będących wówczas nowością aplikacji internetowych było udostępnienie im możliwości charakterystycznych dla aplikacji jednostanowiskowych. W ostatnich latach, kiedy „weblikacje" stały się dominującym kierunkiem w rozwoju oprogramowania, a przeglądanie Internetu stało się nie tylko najpopularniejszym ale i najbardziej naturalnym sposobem wykorzystania komputera, sytuacja się odwróciła: coraz trudniej jest zapewnić aplikacjom jednostanowiskowym utrzymanie wysokich standardów narzuconych przez nowoczesne rozwiązania internetowe – i to przede wszystkim w zakresie interfejsu użytkownika. W szczególności daje się odczuć brak powszechnie dostępnej technologii dynamicznego generowania stron HTML, którą to możliwość w zakresie technologii internetowych zapewniają skrypty po stronie serwera. Przedstawione w artykule rozwiązanie 6S (Simple Solution for Standalone Serwer-Side Scripting) jest próbą pokonania tego ograniczenia za pomocą bardzo prostych środków technicznych. Architektura 6S jest na tyle prosta, że nie wymaga w zasadzie żadnego wsparcia ze strony specjalizowanej biblioteki mimo iż taka biblioteka (dla języka C++) jest dostępna [4]. Na rys. 1 przedstawiono też typowy fragment programu korzystającego z mechanizmów 6S, tym razem w języku Visual Basic. Poza dynamicznym generowaniem stron architektura 6S pozwala na pełne wykorzystanie formularzy internetowych, inicjowanie wywołania podprogramów aplikacyjnych przez stronę WWW i odwrotnie – wywoływanie skryptów na stronie z inicjatywy aplikacji. Skuteczność stworzonej platformy została zademonstrowana na przykładzie aplikacji Psychotronics (rys. 2), obsługującej aparaturę do diagnostyki psychologicznej.

Adres

Jarosław FRANCIK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Poland, jfrancik@ps.edu.pl.