CREATIVE MATH. & INF.
17 (2008), 115 - 126

Impact of modern Web technologies on e-learning platforms

CEZAR TOADER

ABSTRACT. This paper presents a consistent overview of the evolution of Web 2.0 relevant technologies in order to sustain the connection with e-learning 2.0 concepts. Relevant changes in Web application presentation layer are discussed, and the actual trends in Web programming are taken into account. Web browsers drawbacks and HTML transformation are emphasized, rich clients advantages are presented, and programming frameworks are discussed. Also, several Web technologies for Web learning, used at North University of Baia Mare, are presented.

1. INTRODUCTION

Web applications offer a complex functionality to a large number of users. Regarding the asynchronous distance learning process, and the quality of presentations in front of learners, Web applications and Web systems appear suitable for sustain a complex learning process, distributed in time and space, oriented to a large number of learners. The performance and reliability of Web-based learning systems have become very important. The quality of presentations and the use of modern technologies concern the Web-based learning applications developers. This paper presents the complex landscape of Web technologies, actual trends and the connection with e-learning 2.0.

2. WEB 2.0

Most Web 2.0 applications are built upon a conservative list of mature open technologies. This short list of technologies is often referred as the *Web 2.0 stack*. Candidate technologies need to be stable and mature, supported by a very large majority of browsers on different environments, and also to be mastered by a large class of Web developers.

2.1. **XSLT and XPath.**

The most ancient of the technologies discussed here is XSLT (Extensible Stylesheet Language Transformations) and its sister technology XPath (XML Path Language). Although both technologies were specified in November 1999, they are still waiting to be admitted to the Web 2.0 stack. XSLT and XPath were very hot topics among Web developers in 1999 and 2000, but their slow support by Web browsers discouraged developers from using them. Unfortunately, now that they are correctly supported by the main browsers, developers have been slow to come back to XSLT.

Received: 26.10.2007. In revised form: 16.11.2007.
2000 *Mathematics Subject Classification.* 68N99.
Key words and phrases. *Web 2.0, e-Learning 2.0, XSLT, SVG, XForms, OpenLaszlo, XUL, XAML, AJAX.*.

XSLT and XPath were developed by the W3C XSL Working Group [14]. They are parts of a series of recommendations collectively known as XSL that also includes XSL-FO (XSL Formatting Objects), a recommendation that describes *an XML vocabulary for specifying formatting semantics*. XSL-FO defines how documents must be formatted and is often used to produce PDF or print layouts.

Note that the acronym XSL (Extensible Stylesheet Language) is also used by the W3C to designate XSL-FO (XSL Formatting Objects). Many people improperly use XSL to designate an old version of XSLT supported by IE 5.x and even to designate XSLT.

2.2. **SVG.** SVG stands for Scalable Vector Graphics, and its first version was published as a W3C recommendation in September 2001 [9]. It was superseded in January 2003 by SVG 1.1 and the W3C is now working on SVG 1.2.

SVG is an advanced language for two-dimensional drawings with support of animation. It's difficult to present SVG without mentioning that it can be seen as a competitor to both PDF and Flash. SVG 1.0 was a monolithic document, which, like many other W3C specifications, has been split into the following pieces: SVG Tiny, SVG Full, SVG Print, SVG XML Binding Language (sXBL).

What makes SVG so interesting? Consider these elements:

- SVG is an open standard;
- SVG is based on XML and can be produced by XML tools such as XSLT;
- SVG is a text-based format. The text in SVG drawings can be copied and pasted and indexed;
- As a vector graphic format, SVG can be scaled, and this leaves room for many applications;
- SVG has two levels of built-in support for animations: simple animations can be described declaratively in XML, and more complex animations can be written in JavaScript through a well-specified DOM interface that understands the semantics of SVG drawings.

SVG is natively supported in recent versions of Mozilla / Firefox, Safari, and Konqueror. Entries on Microsoft blogs indicate that it should be supported by a later version of Internet Explorer. The situation is not desperate for users of Internet Explorer and older versions of other browsers, since an SVG plug-in is available on the Adobe Web site.

Should we use browser-side SVG now? The answer is probably no, except maybe for mobile applications. Between native implementations still immature, implementing different subsets of the recommendation, and an Adobe SVG plug-in no longer in development, the path is narrow.

2.3. **XForms.** XForms became a W3C recommendation in October 2003. XForms defines itself as *an XML application that represents the next generation of forms for the Web*. However, this definition does not adequately express what can be done with this specification. XForms can be seen as a language to define user interfaces at large and can compete with the unofficial collaboration of Web browsers manufacturers' technologies [13].

No native implementation of XForms currently exists in any major browser, although Mozilla is working on such an implementation (which is already available as an extension). The native support of XForms in browsers is tempered by the reluctance of Microsoft for this standard, as well as by the WHAT WG (Web Hypertext Application Technology Working Group), an organization that defines itself as *a loose unofficial collaboration of Web browser manufacturers and interested parties*. The WHAT WG includes people from Mozilla, Opera, and Safari and has published Web Forms 2.0 (www.whatwg.org / specs / web-forms / current-work), a specification more closely to HTML forms and far less ambitious than XForms.

To deploy XForms applications today, you need to rely either on a plug-in (several of them have been developed but none of them work on other browsers than Internet Explorer), on the XForms Firefox extension, on a specific browser such as X-Smiles, or a client / server implementation.

A large number of pure JavaScript XForms implementations are currently under development. The amount of work to implement XForms shouldn't be underestimated. Important pieces of the XML architecture on which XForms relies are missing in major browsers. They will most probably represent good alternatives in the future, but their current versions are still not mature enough to be considered today as viable alternatives.

Client / server implementations seem the most promising in the short term. A first generation of XForms Client server implementations were rapidly developed after the specification was published. The idea was to transform the XForms documents into HTML and do all the processing server side. A second generation of XForms client server implementations that takes full advantage of Web 2.0 technologies started to appear in 2005. These implementations make extensive use of Ajax.

2.4. **HTML.** Among the Web 2.0 technologies, the one that has undergone more changes than any of the other technologies, and which is most likely to change again is HTML.

HTML was *de-facto* frozen by the arrival of XML in 1998, and today's Web is still based on the set of elements defined in 1999 by HTML 4.01 [5]. This wouldn't be a problem if, in the meantime, there hadn't been an expectation and increasing pressure to get HTML moving. Back in 1998, the HTML Working Group chairs expressed their wish to define a new HTML version with new features even if that broke upward compatibility: "There is no requirement for strict upwards compatibility, although the migration path will be carefully considered. New features and richer authoring environments will provide compelling reasons for upgrading to the next generation of HTML." (http: / / www.w3.org / MarkUp / future /).

With the success of XML, the HTML Working Group had to postpone this goal to undertake the more urgent task of turning HTML into an XML vocabulary with the exact same set of elements as HTML 4.01 and to split the result, XHTML, into modules so that XHTML subsets can be used in mobile phones and that new modules can be added. Now that this task is considered accomplished, the Working Group has resumed working on the next generation of XHTML, called XHTML 2.0.

In the meantime, Microsoft seems to have lost any interest for HTML. Internet Explorer 6.0 was published in 2001 and Microsoft has stopped any development activity on its browser since it resumed working on Internet Explorer 7.0 in 2005.

By contrast, the other Web browsers developers have regained energy, encouraged by the good results of Firefox. Impatient with the slow progress of XHTML 2.0 and often disagreeing with the options taken by the W3C Working Group, they have created an informal consortium, the WHATWG. They propose an alternative evolution path for HTML.

The situation right now consists of two different visions for the next HTML: on the one side is the W3C, which seems to have lost the support of browser makers, and on the other side an informal consortium of browser makers that represents less than 20 percent of cumulative market share.

Since none of these actors seems able to impose their vision, Microsoft appears to be in a situation to arbitrate the debate between one of these two visions; alternatively, we may continue to see HTML stagnate.

2.5. **The W3C Proposals.** XForms is the W3C proposal to get rid of HTML forms limitations. XForms was designed to be usable embedded in other XML vocabularies such as XHTML 1.1, but also as an XHTML 2.0 module and within XHTML 2.0. XForms is the replacement of the XHTML 1.1 Forms module. In other words, this means that you cannot use HTML forms any longer with XHTML 2.0 but must use XForms instead. As XForms is significantly more powerful but also significantly more complex than HTML forms, this has become one of the major objections to XHTML 2.0.

The most spectacular changes between XHTML 1.1 and XHTML 2.0 come from attributes. In XHTML 2.0 any element with a *src* attribute behaves like an object and any element with a *href* attribute is considered as a link.

The option taken by XHTML 2.0 is to add a very limited number of new elements, such as *section* and *summary* and a new type of list, *nl* for navigation lists. HTML Working Group have chosen to use a new attribute, *role*. The XHTML 2.0 *role* attribute uses *Qnames*, short for *Qualified Name*. These names use XML namespace prefixes, and are composed like Java class names.

2.6. **The WHATWG Proposals.** The WHATWG (http://www.whatwg.org) is actively working on two specifications:

Web Forms 2.0, as an extension to HTML 4.01 forms,
(http://www.whatwg.org/specs/web-forms/current-work);

Web Applications 1.0, known as HTML 5, as their proposal for next HTML,
(http://www.whatwg.org/specs/web-apps/current-work).

Both documents try to leverage the experience gained from the current browser implementations and maximize upward compatibility. Their basis can be considered to be common current practices rather than current specifications. A striking example of this position is that HTML 5 defines its own parsing rules, which are neither SGML nor XML but look like the detailed specifications of how current browsers parse HTML documents.

Whereas XForms is completely changing the processing model of interactive Web applications, Web Forms 2.0 is an update of HTML 4.01 forms.

2.7. **XHTML 2.0 vs. HTML 5.** Except for the difference between Web Forms 2.0 and XForms, which are radically different, the differences between XHTML 2.0 and HTML 5 show very different visions of how new features should be added to future versions.

HTML 5 may be seen as simpler with its new elements ready to use. However, new requests for new features will keep coming up. These requests will need to be filtered out to decide which of them should result in creating new elements and which ones should be rejected. This will inevitably lead to inflation in the number of HTML 5 elements and to a problem for Web authors of those feature requests that have been rejected.

With XHTML 2.0, by contrast, if you need a new feature, you create a new *role* value in your own namespace. And if you think that this feature is generic enough, you try to persuade the HTML Working Group to add this value to the set of predefined values.

3. Comparing several rich clients

First, we must see that a large number of applications are installed locally but take advantage of Web-based data stores. Some examples:

- Many anti-virus programs retrieve updated definition files of the latest viruses and also often update the application itself.
- A growing number of news aggregators rely on multiple RSS feeds to provide up-to-date information of interest to the user.
- Accountancy packages often have the capability to import bank statements and account details, usually through a Web service hosted by the bank in question.
- Internet messaging (IM) clients are used to chat, participate in video conferences, and exchange files. Many also have a sophisticated UI that enables contact management and related tasks.

However, the rich client needs to overcome the traditional difficulties of data storage and deployment; namely, it must be able to access an up-to-date version of any data it needs and it must be able to access and install any necessary upgrades, service packs and bug fixes that are produced by an application's creators.

This need has led to a range of client types, from those that are still browser-based but use a design framework to get over the problem of cross-browser development, to those that are standalone applications that can utilize data from the Web as well as auto upgrade. In the middle there are applications that require local files but take advantage of the browser's user interface and rendering capabilities.

Below, three open frameworks for creating rich clients will be quickly examined:

- OpenLaszlo, which enables developers to create sophisticated browser-based user interfaces [8].
- XUL, another browser-based framework that can completely transform the client application [15].

- XAML, a language comprising declarative markup and compiled code that can create desktop applications that can also take advantage of online resources [12].

All three of these frameworks make heavy use of XML and they exhibit varying degrees of separation between the traditional browser-based application and a modern rich client.

3.1. **OpenLaszlo.** OpenLaszlo is designed to aid development of rich cross-browser applications using a combination of declarative XML and JavaScript. The XML and script is read by a Java servlet and transformed into a standard Web page. The client-side features are provided by either Flash or DHTML, depending on how the application is configured [8].

Although OpenLaszlo runs in a Web browser it still can be considered a rich client because of the responsiveness of the user interface and the variety of controls used to capture and display information.

OpenLaszlo is an extremely powerful framework but it does have one drawback: it can only produce a rich user interface *within* the browser window; it cannot configure elements such as the browser's menu bars or affect the general styling and colors.

3.2. **XUL.** XUL, which stands for *XML user interface language*, has the capabilities to configure the browser's menu bars and affect the general styling and color scheme and is targeted at the Mozilla browsers [15].

Technically, XUL requires a browser based on the Gecko engine. This includes all Firefox versions and Netscape from version 6 upwards. In fact, the Firefox browser itself is built using XUL so instead of using XUL to create a rich client, it might be better to say using XUL to customize Firefox.

XUL files must be downloaded and intentionally installed by the user. These files can range from single pages that are designed and act similarly to the Laszlo LZX files, to packages that add new functionality and completely transform the browser. Many current add-ins for Firefox are actually XUL packages.

3.3. **XAML.** The final product we'll look at is Microsoft XAML which stands for *eXtensible Application Markup Language*. XAML is part of the new generation version of Microsoft Windows and can be used to create desktop-based or browser-based applications [12].

XAML is an integral part of the Microsoft new technologies ready to use in Windows Vista operating system. These technologies expose the following three foundations: presentation, communication, and workflow. These are combined with the current .NET 2.0 components to give the .NET Framework 3.0.

To experiment with XAML you'll need to install the relevant libraries, available at msdn2.microsoft.com / en-us / library / ms747122.aspx .

3.4. **Browsers vs. Rich Clients.** The separation of content from presentation and access to data across HTTP, through Web services and other routes, has given developers the best of two worlds – a rich and responsive client interface coupled with the ability to retrieve and edit data from disparate sources.

There are several rich clients, varying from those that are still Web pages at heart to those that are definitely desktop applications but with the power to use data distributed across many machines. Ajax is the primary way of building this functionality into applications.

One issue for companies always was data sharing. As prices for desktop computers dropped, companies went from having only one machine to having one on each desk. Centralized data storage was needed alongside individual machines with large processing power. This gave rise to the idea of a file and print server, a system still in use today.

A second issue was application deployment. To install an application individually on numerous machines was a time consuming task, and upgrades and bug fixes were a drain on manpower and resources, as opposed to the remote installs and upgrades possible today.

The Internet and the World Wide Web (some years later) led to a new paradigm. The browser was a new phenomenon. A Web browser runs on a client that has substantial processing power in its own right but is itself a fairly low user of this potential. Often most of the processing is done on the server. The browser simply renders files, usually stored on a remote machine in the form of HTML, and is capable of presenting data and images to the user as well as accepting input and returning it to the originating server for processing and storage.

The World Wide Web, conceived by Tim Berners-Lee runs on the Internet using the HTTP protocol for transmission and relying on other technologies, such as HTML, for data markup.

Despite its popularity, the Web browser as originally designed has several disadvantages:

- Although there are a growing number of standards governing nearly all aspects of data markup, browsers differ in their compliance and interpretation of these rules. This means that producing an application that looks and behaves identically in all browsers is virtually impossible.

- This situation is further worsened by the fact that of each browser different versions are still in use. This means applications must be tested against dozens of different front ends.

- Because of the Web's inherent lack of security and the fact that there are a number of malicious sites in existence, browsers have strict rules about what an application can do. Client-side storage of data and access to the local machine's other features and programs are two examples of possible restrictions. This means that a lot of the power of the client is not available for use, even if the user of the application desires it.

- Because the browser is designed to be suitable for all sites, it often has features that are unsuitable for a specific application as well as missing those useful on a particular site.

One solution to these drawbacks is the use of rich clients that can take advantage of local processing power. They are also not hampered by security restrictions designed to protect users against malicious software, as opposed to applications intentionally installed on a machine specifically to solve a business or personal need.

4. THE CONTENT OF E-LEARNING 2.0

A new concept, *e-learning 2.0*, is build nowadays and there are many attempts to define it. e-learning 2.0 starts with the trend towards:

- Content made of small, dynamic, interactive, updatable pieces;
- Content delivered closer to place of work;
- Content delivered in pieces over time.

However, the *e-learning 2.0* is closely connected to *Web 2.0*, when it comes to distance, asynchronous learning, testing, and keeping records.

The content of e-learning activities is vast and varied, according to the needs of online learners. Each content type and learning style brings with it specific challenges to custom course development.

The e-learning audience can be an adult learner involved in professional training or a young learner. The developers need to create interfaces and activities that fit the needs of the learners.

The following are some examples of the e-learning methods and learning activities:

- The *tutorial* method delivers content mostly as on-screen text. It assumes a self-motivated, learner.
- The tutorials are usually supplemented with pictures, videos or Flash movies.
- The *case study* delivers content mostly as narrative. This activity provides a real life situation similar to the one the learner might encounter.
- *Exploratory learning* features open-ended lessons, which require learners to ask questions and conduct investigations.
- *Software Simulations* enable the end users to learn a new software application through an artificial, asynchronous process.
- *Virtual machine simulations* allow learners to practice complex processes or to operate delicate instruments before they are required to follow the same processes in real life.
- The *branching scenarios* increase interactivity. It involves a narrative in which the learner chooses an action and directs the course of the narrative. The branching story prepares the learner for a similar situation in real life.

Every activity above requires support from interactive, responsive Web 2.0 applications.

5. DEVELOPING WEB-BASED LEARNING SYSTEM AT NUBM

At North University of Baia Mare, NUBM, we are developing a Web-based Learning System for Distance Education Department. Our work has two directions at the moment: content management systems based on Web programming [11] and secure network infrastructure [10], see Figure 1.

In other Romanian universities there are also intensive activities on developing different platforms serving e-learning process. The learner can be in front of a desktop computer or can be outside the building, on the move, using a PDA [4].

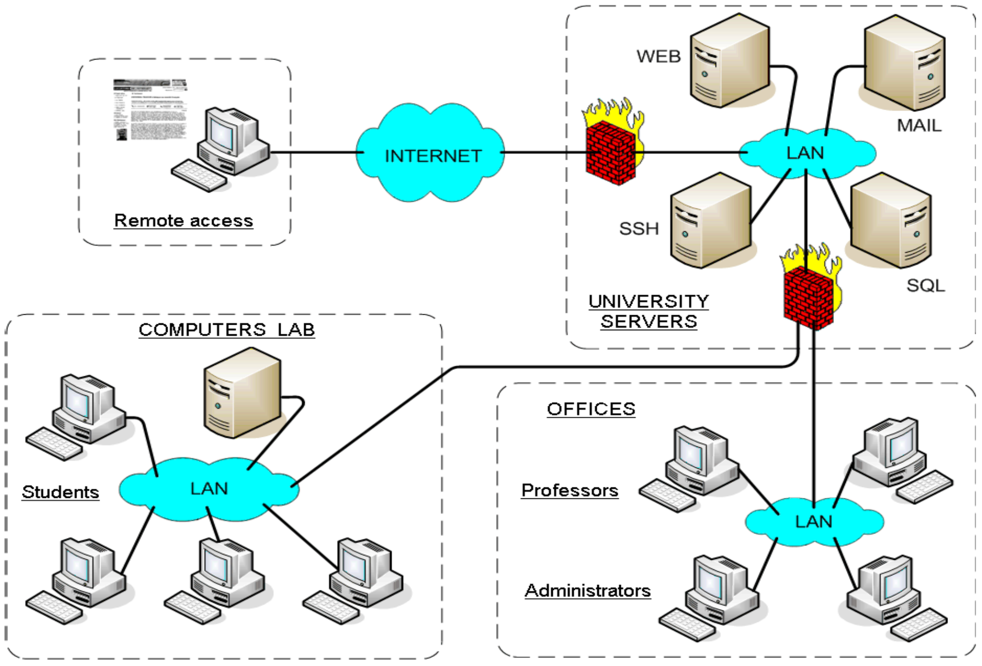


Figure 1. Simple infrastructure for e-learning platform at NUBM

The learning platform can build a complex, consistent view. For tutorials about technologies, phenomena, technical processes learners need a new user experience. They need pages with interactivity, dynamic elements, visual effects, and support for different multimedia. The developing process of a LMS (Learning Management System) at NUBM is a complex project and involves educators, students and engineers.

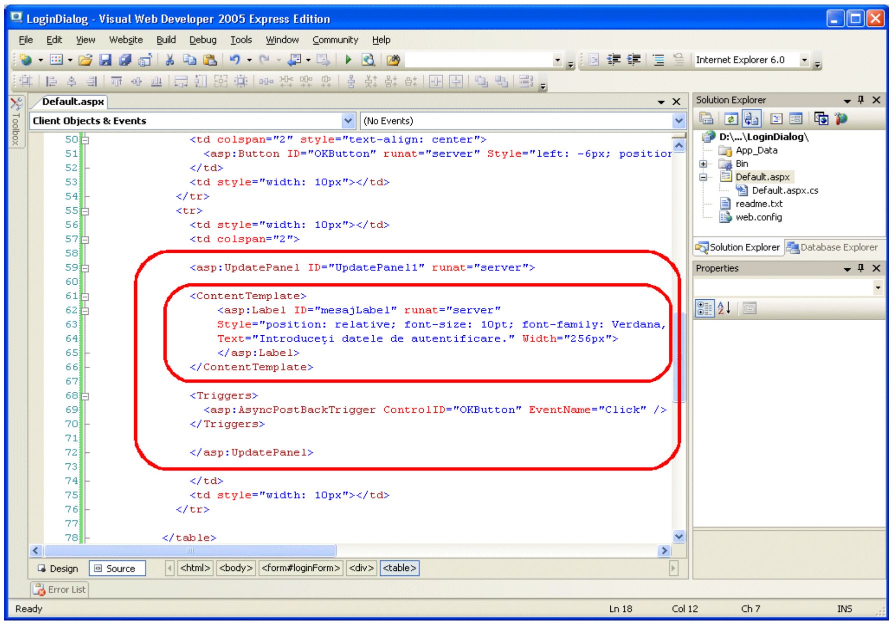
There are numerous developers which bring consistent arguments to develop rich clients. A very attractive solution is OpenLaszlo because on this platform the programmer can build SWF files or DHTML files, having the same aspect and behavior [8].

On the other hand, there are many developers which intent to use Flash plugin only when its qualities are necessary and the Web page should look more dynamic. They consider more useful to develop Web pages based on XML, DHTML and JavaScript. This approach has advantages when it comes to bookmark pages, for example. But major advantages concerning the Web page interactivity appear when the AJAX technology is used.

6. APPLYING AJAX TECHNOLOGY

The term *AJAX* means *Asynchronous JavaScript and XML* and, practically, it is an open-technologies suite, designated to improve the level of interactivity between the user and the Web interface. This term was proposed by Jesse James Garret in 2005, but the component technologies of AJAX existed before 2005.

At North University of Baia Mare the developers are accommodated with ASP.NET AJAX and they are using *Visual Web Developer 2005* or *Visual Web Studio 2005*. The framework offers the necessary controls to build a modern Web application. One of the main controls is *UpdatePanel* which marks the area of partial updates, see Figure 2.

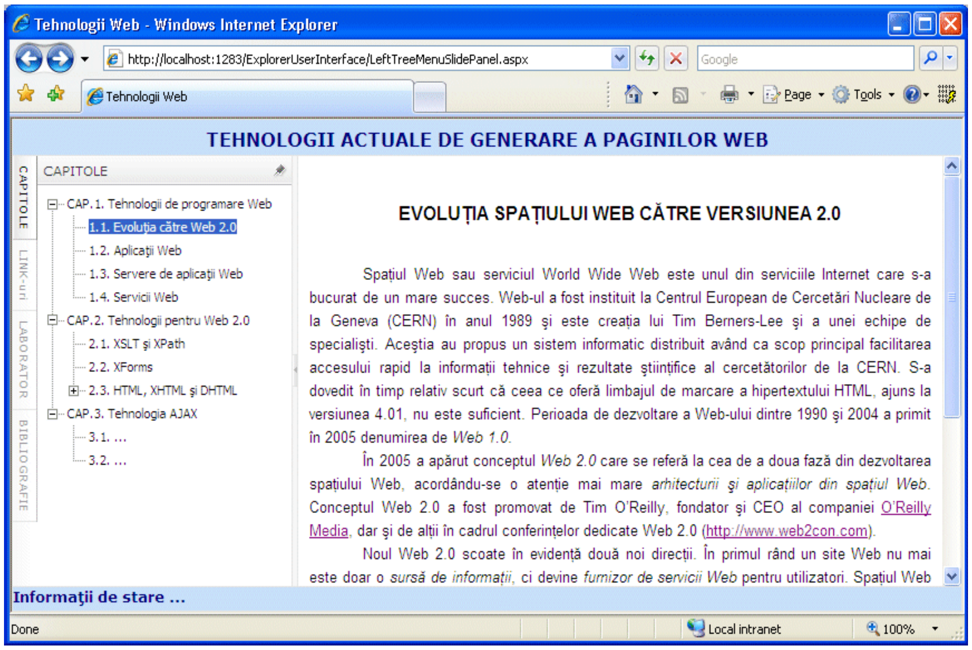


```
                <td colspan="2" style="text-align: center">
                  <asp:Button ID="OKButton" runat="server" Style="left: -6px; positior
                </td>
                <td style="width: 10px"></td>
              </tr>
              <tr>
                <td style="width: 10px"></td>
                <td colspan="2">

                <asp:UpdatePanel ID="UpdatePanel1" runat="server">

                <ContentTemplate>
                    <asp:Label ID="mesajLabel" runat="server"
                    Style="position: relative; font-size: 10pt; font-family: Verdana,
                    Text="Introduceţi datele de autentificare." Width="256px">
                    </asp:Label>
                </ContentTemplate>

                <Triggers>
                  <asp:AsyncPostBackTrigger ControlID="OKButton" EventName="Click" />
                </Triggers>

                </asp:UpdatePanel>

                </td>
                <td style="width: 10px"></td>
              </tr>

            </table>
```

Figure 2. Defining the partial update zone in a Web page using *UpdatePanel* control

A typical Web page in front of the learner should look like in the Figure 3. There is a main panel with the relevant information using the most of the page, but there are also other important frames: the title zone at the top, the status and messages zone at the bottom, and a navigation panel in the left side.

The navigation panel shows the structure of the course as a tree which can expand or collapse. The items of the course are the leaf of the tree. Clicking on an item triggers the main panel refresh and change of shown topic, see Figure 3.


Tehnologii Web - Windows Internet Explorer
http://localhost:1283/ExplorerUserInterface/LeftTreeMenuSlidePanel.aspx
Google
Tehnologii Web
Page
Tools
CAPITOLE
LINK-uri
LABORATOR
BIBLIOGRAFIE
Informaţii de stare ...
Done
Local intranet
100%

TEHNOLOGII ACTUALE DE GENERARE A PAGINILOR WEB

CAPITOLE

- CAP. 1. Tehnologii de programare Web
  - 1.1. Evoluţia către Web 2.0
  - 1.2. Aplicaţii Web
  - 1.3. Servere de aplicaţii Web
  - 1.4. Servicii Web
- CAP. 2. Tehnologii pentru Web 2.0
  - 2.1. XSLT şi XPath
  - 2.2. XForms
  - 2.3. HTML, XHTML şi DHTML
- CAP. 3. Tehnologia AJAX
  - 3.1. ...
  - 3.2. ...

EVOLUŢIA SPAŢIULUI WEB CĂTRE VERSIUNEA 2.0

Spaţiul Web sau serviciul World Wide Web este unul din serviciile Internet care s-a bucurat de un mare succes. Web-ul a fost instituit la Centrul European de Cercetări Nucleare de la Geneva (CERN) în anul 1989 şi este creaţia lui Tim Berners-Lee şi a unei echipe de specialişti. Aceştia au propus un sistem informatic distribuit având ca scop principal facilitarea accesului rapid la informaţii tehnice şi rezultate ştiinţifice al cercetătorilor de la CERN. S-a dovedit în timp relativ scurt că ceea ce oferă limbajul de marcare a hipertextului HTML, ajuns la versiunea 4.01, nu este suficient. Perioada de dezvoltare a Web-ului dintre 1990 şi 2004 a primit în 2005 denumirea de *Web 1.0*.

În 2005 a apărut conceptul *Web 2.0* care se referă la cea de a doua fază din dezvoltarea spaţiului Web, acordându-se o atenţie mai mare *arhitecturii şi aplicaţiilor din spaţiul Web*. Conceptul Web 2.0 a fost promovat de Tim O'Reilly, fondator şi CEO al companiei O'Reilly Media, dar şi de alţii în cadrul conferinţelor dedicate Web 2.0 (http://www.web2con.com).

Noul Web 2.0 scoate în evidenţă două noi direcţii. În primul rând un site Web nu mai este doar o *sursă de informaţii*, ci devine *furnizor de servicii Web* pentru utilizatori. Spaţiul Web

Figure 3. The Web page for e-learning at NUBM

The tree shown in the navigation panel is read from a XML file. A very important quality of this navigation panel is that it is collapsible and resizable.

7. CONCLUSIONS

As a first conclusion, three technologies should be ready for Web 2.0:

- *XSLT* appears to be a good solution for keeping your JavaScript slim and focused on treatments, while leaving all the low-level formatting to XSLT, together with higher-level tasks such as sorting and filtering content.
- *SVG* is a very powerful technology. It is an XML-based open standard that can be generated by XML tools and animated either declaratively or in JavaScript. Unfortunately, its implementation is still immature.
- Although *XForms* is a relatively young technology, it can be used by today's browsers through client / server implementations. With this architecture, XForms is a declarative alternative that can be used to deploy Web 2.0 applications using JavaScript and Ajax on the client.

On the other hand, rich clients appear as a viable alternative to usual browsers. Rich clients can take advantage of local processing power. They are also not hampered by security restrictions designed to protect users against malicious software.

All three rich client frameworks above have their place on the rich client podium. Each has its own strengths and weaknesses.

Nowadays, it is possible to obtain interesting and attractive visual effects in a Web page using AJAX technology. Thus, the web page behavior is almost similar with the behavior of a desktop application and this leads to a new and interesting user experience.

AJAX is a group of technologies for *remote scripting* and its purpose is to help creating Web interfaces with increased interactivity, dynamics and execution speed, comparing with the old technologies known as *Web 1.0*. AJAX technology features [6]:

- Support for data presentation (XHTML and CSS);
- Support for interactivity and dynamic rendering (based on Document Object Model);
- Support for data exchange and manipulation (XML, JSON, XSLT);
- Support for asynchronous data transfer (using XMLHttpRequest object);
- Support for data processing (JavaScript / ECMAScript).

Nowadays, programmers develop Web 2.0 applications to build modern web-learning systems. AJAX technology is used to create interfaces with increased interactivity, dynamics and execution speed.

REFERENCES

[1] Berglund, A., (ed.), *XML Path Language XPath 2.0*, 2007, www.w3.org/TR/xpath20/
[2] Buraga, S., *XML Technologies*, Polirom Publishing House, 2006
[3] Buraga, S., *An Extensible Framework for Building Interactive Courses on Web*, 5th International Symposium on Economic Informatics - IE 2001, Bucharest, 2001
[4] Huţanu., C., Vlaicu, A., Turcu, C., Risteiu, M., *CmL-plus: A new vision for the first m-learning platform in the Romanian DE System*, 6th WSEAS Int. Conf. on Applied Computer Science, China, 2007, pp. 327
[5] HTML – W3C, *HTML 4.01 Specifications*, http://www.w3.org/TR/html401/
[6] Garret, J.J., AJAX: *A New Approach to Web Applications*, AdaptivePath, 2005, http://adaptivepath.com/publications/essays/archives/000385.php
[7] Kay, M. (ed.), *XSL Transformation (XSLT), Version 2.0*, 2007, www.w3.org/TR/xslt20/
[8] OpenLaszlo 4.0, 2007 – www.openlaszlo.org
[9] SVG – W3C, Scalable Vector Graphics (SVG), http://www.w3.org/Graphics/SVG/
[10] Toader, C., Petrovan, A., Costea, C., *Business Models and Secure Web learning on PKI*, The 7th International Multidisciplinary Conference, North University of Baia Mare, 2007, pp. 701
[11] Toader, C., *Web Applications Programming*, Risoprint Publishing House, 2005
[12] XAML – MSDN, http: // msdn2.microsoft.com/en-us/library/ ms747122.aspx
[13] XForms – W3C, *The Forms Working Group*, http://www.w3.org/MarkUp/Forms/
[14] XSLT – W3C, *Extensible Stylesheet Language (XSL) Version 1.1*, December 2006, http://www.w3.org/TR/2006/REC-xsl11-20061205/
[15] XUL – Mozilla, *XML User Interface Language (XUL)*, 2007, www.mozilla.org/projects/xul

NORTH UNIVERSITY OF BAIA MARE
DEPARTMENT OF COMPUTER ENGINEERING AND ELECTRONICS
DR. VICTOR BABEŞ
430083 BAIA MARE, ROMANIA
HTTP://WWW.UBM.RO
E-mail address: cezar.toader@gmail.com