Online Web Laboratories implemented as Web Services

Milena Stanković, Ivan Petković, Milan Rajković
Faculty of Electronic Engineering, Niš
Beogradska 14, 18000 Niš

1 INTRODUCTION

During the last decade, World Wide Web (WWW) has experienced a rapid growth and has become the most wide spread distributed model. The simplicity of WWW is one of the reasons for its incredible popularity. Web clients and servers communicate using HTTP protocol, exchanging mainly MIME content. The model is completely "loosely coupled" which means that clients and servers can be developed and maintained independently, without almost any centralized activity. Moreover, everything about the Web is completely standardized, which makes it ideal for creating applications whose distributed parts work on different machines and operating systems.

The expansion of the WWW resulted in new approach on building the Web sites: they evolved from a simple unstructured set of pages to the complex information systems. Many organizations and companies have seen the benefit from having a Web site, as their online presence increased the number of their customers.

One problem more is that standard Web applications usually have limited functionality. The advantage of having a lite client can sometimes be the disadvantage: interfaces that users have are far from the ones they would get in standard desktop applications.

This paper shows two possible solutions of this problem. The first one is increasing the functionality of a standard browser-based application by introducing a new user interface. For this purpose, a new framework has been developed, allowing programmers to create reusable and consistent code for browser based applications with extended functionality.

The second solution is more general, and it involves using Web Services. This is a new technology, based on exchanging XML messages, which allows application to application development on the Web. Therefore, Web applications don’t have to be browser based. Clients can be created using any programming language with an interface typical for desktop applications. The main purpose of Web Services is integrating different information systems in a completely standardized and platform independent way.

Both the approaches have been used for creating an Online Web laboratory, developed at the Faculty of Electronic Engineering in Niš.1 This is a Web based, distributed application, built mainly for educational and researching purposes in the fields of automatic controls, logic design, image processing, programming languages and Internet technologies. The central part of the laboratory is a Web Service application which exposes the essential functionality. It can be accessed either through a Web portal or directly from client applications built on technologies that support Web Services.

1 INTRODUCTION

During the last decade, World Wide Web (WWW) has experienced a rapid growth and has become the most wide spread distributed model. The simplicity of WWW is one of the reasons for its incredible popularity. Web clients and servers communicate using HTTP protocol, exchanging mainly MIME content. The model is completely "loosely coupled" which means that clients and servers can be developed and maintained independently, without almost any centralized activity. Moreover, everything about the Web is completely standardized, which makes it ideal for creating applications whose distributed parts work on different machines and operating systems.

The expansion of the WWW resulted in new approach on building the Web sites: they evolved from a simple unstructured set of pages to the complex information systems. Many organizations and companies have seen the benefit from having a Web site, as their online presence increased the number of their customers.

One problem more is that standard Web applications usually have limited functionality. The advantage of having a lite client can sometimes be the disadvantage: interfaces that users have are far from the ones they would get in standard desktop applications.

This paper shows two possible solutions of this problem. The first one is increasing the functionality of a standard browser-based application by introducing a new user interface. For this purpose, a new framework has been developed, allowing programmers to create reusable and consistent code for browser based applications with extended functionality.

The second solution is more general, and it involves using Web Services. This is a new technology, based on exchanging XML messages, which allows application to application development on the Web. Therefore, Web applications don’t have to be browser based. Clients can be created using any programming language with an interface typical for desktop applications. The main purpose of Web Services is integrating different information systems in a completely standardized and platform independent way.

Both the approaches have been used for creating an Online Web laboratory, developed at the Faculty of Electronic Engineering in Niš.1 This is a Web based, distributed application, built mainly for educational and researching purposes in the fields of automatic controls, logic design, image processing, programming languages and Internet technologies. The central part of the laboratory is a Web Service application which exposes the essential functionality. It can be accessed either through a Web portal or directly from client applications built on technologies that support Web Services.

In section 2 of this paper, the framework for supporting complex Web applications is presented. Section 3 explains Web Services technology. Section 4 describes in more details a part of the Online Laboratory intended for digital image processing and section 5 is a conclusion.

2 FRAMEWORK

A new framework has been developed to support Virtual Web Laboratory Portal. It is based on the method which successfully eliminates redundancy problem [1], and it supports all DOM compatible browsers [2]. Its main purpose is to provide intuitive users interface and the robust platform to support various Web services. Visually, it is represented as a virtual desktop, with icons, menus, background and windows. User will be able to open multiple documents, and each document will be placed in the separate window.

Potential reusability of this framework is high, since it can be used as a base for almost every type of the Web application. Moreover, the possibility for working with multiple Web documents extends the functionality of the standard Web applications. At the high level model design, MVC (Model View Controller) design pattern is implemented. This provides users interface independent from the Web application’s behavior and data model.

During the design phase, many design patterns are used in order to develop a consistent, easy to maintain and robust user interface. For example, implementation is abstracted, providing a possibility to develop the same application in HTML or some other technology, depending on the target platform and the application type. Also, the framework is browser-independent.

1 This work has been performed within the project "Virtual Web Laboratories for Permanent Engineering Education" (IT.1.2.0180A), supported by Ministry of Science, Technologies and Development of Serbia.
The framework has been developed on the client side. It is consisted of the user interface and event handling logic (behavioral aspect of the framework). Using some of the methods to retrieve the user’s preferences (either by reading the browser’s cookies, or by reading the preferences previously stored in the database), user can drastically change look-and-feel of the application. That means changing the application "skin" (background image, color profile, window appearance...), but also adding some new icons on his customized virtual desktop. Even the whole session (position of the opened windows, and its contents) can be saved and retrieved next time user visits the page.

The framework does not depend on the server side technology used. Also, an additional Web methodology can be used for server side components and overall design of the Web application structure. A client side component Virtual Desktop have been developed using HTML, JavaScript and DOM (Document Object Model) technologies.

2.1 Architecture

We can look at the architecture of the Web application which is deployed on multiple tiers:

![Web Application architecture](image)

- **Client side**: Here is implemented Virtual Desktop, which includes complete users interface and windows manager. These windows are generated as HTML object and their main purpose is to store Web pages, they act like containers. Even handling is also implemented in the Virtual Desktop, which provides response to the users actions or other events.

- **Web server**: on this tier can be located server-side functionality of the Web application. User requests are processed and returned as an HTML output to the user, or processed and forwarder to the Application tier. Server side scripts (PHP, ASP or JSP) are located on this tier. Various Web methodologies can be applied when developing this tier, like Fusebox [3]. For the purposes of the Web portal, PHP is used.

- **Application tier**: Web services are located on this tier. They respond to the input from the other tiers, or more generally applications.

- **Database tier**: the last tier is database tier. We used MySQL database for the portal.

2.2 User interface and Virtual Desktop

Virtual Desktop provides independent look-and-feel of the users’ interface. Moreover, it can easily be customized to the user preferences or changed on the Web site scale.

![Virtual Desktop Use Case](image)

Virtual Desktop consists of the User interface and the Window manager (Multiple Pages Management). User interface is a set of icons, menus and windows with coded behavior. For example, VWLAB Web portal has on the Virtual Desktop a main menu, which can be used to navigate the Portal, and a set of icons which has a function of shortcuts to some URLs.

Opening the document results in creating a new window with specified document as its contents. Loading of a Web page is seamless, because it is loaded in a buffer first, and then transferred to the corresponding window. User can move, minimize or resize the windows. Support for the multiple document results that the user can use several services at the same time. Viewing the Internet pages (not only on the local site) in the window is also possible. A good example of taking advantage of this option is a student who uses the local web service for completing the laboratory exercise, but at same time searches the local or Internet pages for the more detailed explanation of some concepts involved in the exercise.
3 XML WEB SERVICES

The biggest advantage of the Web model over the traditional distributed models (DCOM, CORBA, RMI etc.) is that it is loosely coupled which makes it ideal for integrating information systems developed independently and on different technologies. But, the Web has one major drawback: all Web applications are browser based, and application to application development over the Web is not possible. This is because the MIME content which comes from the server is “human readable”, and it is not structured in a way that applications other than Web browsers could understand it. Web Services are introduced in order to solve this problem.

The basic idea behind Web Services is to adapt the Web programming model for using in applications that are not browser-based. The goal is to provide a platform for building distributed applications using software running on different operating systems and devices, written using different programming languages and tools from multiple vendors, all potentially developed and deployed independently. [1]

One of the formal definitions of Web Services is given by Webservices.org: “Web Services are encapsulated, loosely coupled contracted functions offered via standard protocols”, where:

- "Encapsulated" means the implementation of the function is never seen from the outside.
- “Loosely coupled” means changing the implementation of one function does not require change of the invoking function.
- “Contracted” means there are publicly available descriptions of the function’s behavior, how to bind to the function as well as its input and output parameters. [5]

The most important step that has been made is switching from HTML to XML for representing the data being send and received. XML is a structured language, and applications can understand it, which is why Web Service applications don’t have to be browser based.

3.1 The Web Services Architecture

There are a couple of issues which have to be addressed when dealing with any distributed architecture. The first one is the protocol which is used for exchanging data between clients and servers. The second is the way that server describes its interface so that clients can create correct calls. Finally, clients have to be able to locate servers, therefore some kind of central registry is necessary. The Web Services architecture covers this issue in a very simple and completely standardized way.

SOAP

Protocol for communication is SOAP (Simple Object Access Protocol). It is a lightweight protocol, completely standardized by W3C organization and the current version is 1.2. SOAP is intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. [6]

Three main characteristics of SOAP protocol are [7]:

1) SOAP is simple and extensible. Simplicity is crucial for Web Services, because they have to provide communication between systems built on totally different technologies. On the other hand, SOAP is also extensible, which means that adding new features is possible, especially those dealing with security, message routing etc.

2) SOAP can use any underlying protocol for communication. Although it is usually related to HTTP, it can be used with TCP, SMTP or any other protocol.

3) SOAP is completely independent of any particular programming model. Although many developers immediately equate SOAP to making RPC calls, it provides much broader approach, allowing any number of message exchange patterns.

WSDL

WSDL (Web Service Definition Language) is an XML based language for describing Web Services. It provides documentation for a distributed service, which is the basic assumption for the automated application to application communication.

Whereas SOAP defines communication between a requester and a provider, WSDL describes services offered by the provider and can be used as a recipe for creating proper SOAP message for accessing these services. A WSDL document has a role similar to an IDL file in CORBA technology, or the Remote Interface in JAVA RMI technology. [5]

UDDI

Universal Description, Discovery, and Integration (UDDI) is a standard designed to provide a searchable directory as repository of Web Services. Thus, it represents the service broker that enables service requesters to find a suitable service provider. [4]

Service providers can register with any of four hosts that currently exist: IBM, HP, SAP and Microsoft. Registration data is periodically exchanged between the hosts, so they all have, more or less, the same information.

UDDI has two main parts: registration and discovery. For accessing the data, service providers and potential users use SAOP API functions or some of the user interfaces offered by UDDI operators or independent vendors.

Information which UDDI offers can be classified into:

- White pages, which provide all information about a provider, such as a company name, address, contact, Web site address etc.
- Yellow pages which can be searched by the type of a service, location, products etc.
- Green pages which contain technical information about the service.

Considering all aspects of the Web Services architecture mentioned above, it can be represented as shown in the Figure 3 [5]

Using Web Services with HTTP as an underlying protocol

Although SOAP can be used with any underlying protocol, HTTP is the most commonly used. The architecture of a distributed application built in this way is fairly simple and similar to a traditional Web application. This architecture is shown in the Figure 4.

4 WEB SERVICES FOR DIGITAL IMAGE PROCESSING

In this chapter we will present one specific part of the Online Web Laboratory concerning the digital image processing. We will show the architecture of the Web Service application and also the way these services are accessed from the Web portal using PHP server side scripting.

4.1 DIP Web Services architecture

DIP (Digital Image Processing) Web Service application is based on the classes for digital image processing called DIP++, developed at the Faculty of Electronic Engineering in Nis. These classes are written in Microsoft Visual C++ 6.0.

The most important part of the DIP Web Service application is an application server which exposes Web Services. On the other hand, this server should call actual functions for digital image processing which are members of DIP++ classes. Therefore, the simplest solution would be to create the application server in the same programming language as DIP++ classes, in this case Microsoft Visual C++ 6.0. Unfortunately, this language doesn’t support Web Services, which makes this solution impossible.

Of course, it is always possible to rewrite DIP ++ classes in some other programming language that supports Web Services. But, these classes are very complex and they have been developed over a couple of years, so rewriting and retesting would require a significant effort.

The way out is in introducing a new layer in the application architecture. This layer is a DLL file created in MS Visual C++ 6.0, which exposes simple functions for digital image processing that can be called from any programming language. In this way we can create the application server using the technology we want, and call functions in the DLL for image processing. On the other hand, the functions in DLL file are responsible for instancing DIP++ classes, calling their methods, and finally destroying the instances. The drawback of this solution is that we hide the complete DIP class hierarchy, because the application server can invoke only the simple functions that DLL file exposes. In this way we loose all the advantages of object-oriented programming model. But the main goal is still accomplished: we can create a Web Service application in any programming language, and we can reuse DIP classes without changing their implementation. Figure 5 shows the architecture explained above.

The application server is created in Delphi 7. We created it as an ISAPI for Microsoft IIS. Here is an example of the Web Service that implements the median filter:

```pascal
function MedianFilter(inBuffer: TByteDynArray; windowWidth, windowHeight: Integer; token: String): TByteDynArray; stdcall;
```
4.2 Invoking DIP Web Services Using the Web Portal

Developers have a variety of choices when it comes to creating a client for a Web Service application. Although many modern programming languages can be used for implementing the client, sometimes it is necessary to show the results in a Web browser. If this is the case, Web Services have to be invoked from a Web Server application and the easiest way to achieve this is by using a server side scripting language.

When creating the Online Web laboratory, we used PHP for calling the Web Services. The most popular implementation of SOAP for PHP is the one created by Deitrich Ayala in the form of NuSOAP classes. They allow creating both SOAP clients and SOAP servers, which means that PHP can be used not only for invoking, but also for creating Web Services. This classes can also parse a WSDL file, which can simplify the process of serializing the data being send and received.

Considering the fact that an image is serialized as a base64Binary data on the server side, it is important to use functions for base64 encoding and decoding on the client side as well. For this purpose, we used PHP functions `base64_encode` and `base64_decode`.

Figure 6 shows the result of digital image processing obtained using the Web portal.

**Figure 5: DIP Web Services architecture**

Parameter `inBuffer` represents the image which is sent to the Web Service. Type `TByteDynArray` is a Delphi type which maps directly to base64 binary blocks of data. As a result, the image is represented as a base64Binary data in a WSDL file and a SOAP message.

If the array representing the image was not base64 encoded, every element would be mapped separately in a SOAP message. This would slow the processes of serializing and transmitting which would result in low performances of the application.

**Figure 6: Digital image processing using the Web portal**
5 CONCLUSION

The implementation of the Online Web Laboratories shows that Web technologies can be used for creating robust and easy to maintain distributed applications with fully functional user interfaces. Using the technology of Web Services combined with the new framework for Web applications resulted in a modern, multi-tier application which can be executed on any platform. Users can access the application either through a Web portal, or using their own client application, implemented in any technology that supports Web Services. Therefore, the Online Web Laboratories can be easily integrated with other applications or information systems.

The Web Laboratory covers several areas of researching. Here, we presented in more details the implementation of the Web Services for digital image processing. Some specific problems related to reusing classes written in C++ were also addressed here.

6 REFERENCES


Abstract: During the last decade, the Web model has experienced a rapid growth due to its simplicity and loose coupling. But the main drawback of the Web is that clients have simple interfaces and limited functionality. This paper introduces two possible solutions of this problem used for implementing the Online Web Laboratories at the Faculty of Electronic Engineering in Nis. The first approach is using a new framework for creating robust and fully functional Web applications, and it is based mostly on client and server side scripting. The second involves exposing the core functionality in the form of Web Services, which provides the possibility of creating a client using any technology that supports Web Services. These two approaches combined result in a modern, multi-tier application which can be executed on any platform. It can also be easily integrated with other applications or information systems.