WING+ WEB-APPLICATION DEVELOPMENT FRAMEWORK

Ahmad Golchin[1] and Ahmad Baraani[2]

[1]Department of Computer Engineering, Sheikhbahaie University, Isfahan, Iran
ahmadgolchin@gmail.com
[2]Engineering Faculty, University of Isfahan, Isfahan, Iran
ahmadb@eng.ui.ac.ir

ABSTRACT

Nowadays web development has been expanded greatly, not only for ordinary reasons, but also for enterprise and other important purposes. So it's obvious that there would be some challenges and problems if you want to develop important applications such as an automation system on the web. Some of the problems that I've encountered myself as a software developer group manager are source code safety, high performance rate at runtime, optimum resource usage and the ability to customize some essential parts of application that are not allowed or their source code is not available. In this document a new web programming framework called Wing+ will be introduced to you, that is designed to resolve these issues and so many other problems that are out of the scope of this paper.

KEYWORDS

C++ Web Framework, Source Code Safety, Modular System, Optimum Resource Usage, High Performance

1. INTRODUCTION

There are some issues that seem to still not have been resolved in a remarkable way. The basic motivation for designing and introducing the Wing+ framework is a result of encountering these problems. I am going to clarify these problems for you to have a better understanding of what this document is going to concentrate on.

1.1 Motivation

- Source Code Safety: Do you install your web application on your own server or do you buy it from a hosting provider? If you're planning to implement a web-based automation system for another company, where do you setup their system (in most cases on the company's computers)? If you use your own server, can you ensure its source code safety versus the hackers?

- Sometimes you need a very high performance rate and the least runtime to do some tasks online. Unfortunately most of the web programming languages and frameworks are not fast enough and optimum to do some of these heavy tasks as you will see later. I'll give an example from my own experience when I decided to convert my Windows based OCR (Optical Character Recognition) software to a web-based one, I figured out that I need a very expensive set of servers called cluster, because the language I used to write my web program was so slow in comparison to the one I wrote the Windows based software.

- In my opinion it is very hard to manipulate some essential parts of a web application such as a Session Manager or the Server Interface, when you need to fully customize them. So instead we should try and look at other possibilities and ways for improvement.

- It seems that common web programming languages do almost everything themselves, so when you need to do something more exclusive and professional, it's better to look for specialized methods instead of using predefined and automated tools, so you can optimize you performance.

Measures have been taken in this regard such as CSP [1] and Wt [2] frameworks. However they have tried to resolve the preceding problems, it seems that they did not do good at all of the mentioned issues, as it will be discussed in parts 2 and 5.4 of this paper. This document introduces a new web application development framework and even maybe a new programming language that tries to resolve the mentioned issues, with more concentration on source code safety and performance. Hackers can steal your source code easily by intruding into your server. For instance this could be achieved by uploading a script on the server and manipulating the remote share settings, and so it's possible to gain file system access, or for someone to steal your session while you are working on your server through the host panel software, then the intruder can gain access to your files and so many other possibilities [3]. Wing+ is designed for creating web applications which require source code security, high performance rate at runtime, simple customisability and efficient resource usage to lower the costs of hardware which is needed. There are two essential ideas that helped me to overcome the introduced problems. Choosing an appropriate programming language that provides the basics of high performance and source code security, and designing a highly flexible and customizable API that covers the last two problems were the keys which helped turn Wing+ into what it is today.

1.2 Wing+ Main Features

Some of essential features that Wing+ has are the following.

- Introduced new fully customizable API for HTTP applications, makes it possible to design all parts of your application modularly. For an instance you may define Session Manager as a separated module and attach it into your application without manipulating or even knowing anything about implementation of the default Session Manager that Wing+ framework is using.
  Another sample is that you're allowed to implement server interface and application boot-up driver yourself if you don't want to use Wing+ embedded server. You can make your Wing+ applications to run under other HTTP servers such as Apache or IIS.

- It's possible to run an application based on the source codes you've put them on to the server or without them by removing them after the first compilation. Compilation will occur automatically when you run the application by the boot-up driver module.

- The pipeline filtering that you can do easily with just implementing related part of Wing+ API makes the security sanity checkups in a very convenient way.

- There are two approaches to designing web pages. First, you can implement a response handler class and create your pages like you did for Windows based forms. The second approach that is more familiar to web developers is to use Wing+ blend markup language that is a mixture of HTML, C++ codes and Wing+ components. The second approach splits the designer's and programmer's jobs and makes development processes more structured but the first one is more flexible. Both of these designing methods are available in Wing+ and there's no limitation.

- Wing+ provides a very simple approach for the programmer to create data entities in different scopes such request's, application's and session's and an easy way for designer to bind these entities to the components.

- Component module implementation has made it possible to create your own tags for the blend markup design method and widgets for the class based design method.

- All parts of the application are in the form of modules and you can dynamically add modules to your program while it is running, and so the application will be in full control.

This article consists of 6 essential sections that try to illustrate how these benefits are made available in Wing+. Section 2 is an introduction to related works and their features, section 3 and 4 will describe the architecture of Wing+ and its basic concepts, and what's found in the 5th section is an assessment of Wing+ and other frameworks proving that Wing+ resolves the preceding problems in a remarkable approach.

2. RELATED WORKS

What's found in this section is an introduction to CSP [1] and Wt [2] frameworks. They follow the mentioned motivations but here are some essential differences such as selecting the appropriate API and flexible structure which leads to poor customizability.

2.1 CSP (C++ Server Pages)

C++ Server Pages (CSP) is a Web Engine for advanced Web Application Development that uses blended Markup Language / C++ scripts (such as HTML/C++, XML/C++, WML/C++ etc.) Similar to ASP and JSP, it provides a great easiness in creating web pages with dynamic content, as well as complex business applications. However, instead of Java, Javascript or VBscript, it uses C++ [1]. The main reason that made CSP to what is it today was high performance needed in some Web Applications. It just supports only page based development approach and low customizability so that the only customizable levels in CSP are Session Management and Application. In CSP, programmer is not allowed to design and append his own created components in order to simplify development process for the future works and so it's one of the main weaknesses found in CSP until now.

2.2 Wt (Pronounced Witty)

Wt (pronounced as *witty*) is a C++ library for developing web applications. The API is **widget-centric** and uses well-tested patterns of desktop GUI development tailored to the web. To the developer, it offers abstraction of web-specific implementation details, including client-server protocols, event handling, graphics support, graceful degradation (or progressive enhancement), and URL handling. Unlike many page-based frameworks, Wt was designed for creating stateful applications that are at the same time highly interactive (leveraging techniques such as WebSockets and Ajax to their fullest) and accessible (supporting plain HTML browsers), using automatic **graceful degradation or progressive enhancement**. Things that are natural and simple with Wt would require an impractical amount of effort otherwise: switching widgets using animations, while being perfectly indexed by search robots with clean URLs, or having a persistent chat widget open throughout, that even works in legacy browsers like Microsoft Internet Explorer 6. The library comes with an application server that acts as a stand-alone Http(s)/WebSockets server or integrates through FastCGI with other web servers.It does not support the page based development that is familiar to web developers and is suggested to Windows developers who want to migrate their application simply to a Web-based one.

3. WING+ ARCHITECTURE

Wing+ is based on modules. In order to create an application the programmer should implement some modules which may include pages, pipeline filters, and business logic entities and so on. This can be achieved by coping the binary files of modules into the application's "lib" or "bin" folder or by placing their source code inside the "src" folder. If you put source codes inside the "src" directory of the application you are deploying, they will be compiled automatically by the application itself according to the host architecture, so there is no need to worry about its portability. Wing+ needed to be based on an appropriate programming language which is fast, efficient, reliable and familiar to the programmers' community. I selected C++ for many reasons which will be described in the following section.

3.1. Wing+ Is Based On C++

The reasons which qualified C++ as the best choice for Wing+ implementation are the following.

- Incredibly high processing efficiency. Benchmarks have shown a range of 80 to 250 times higher processing speed than ASP [1]. C++ is designed for creating high performance applications and it's more popular among other object oriented programming languages [4].

- The use of pure C++ allows the use of tons of libraries that are currently available. It is important to notice that the libraries written in C++ are tens or hundreds of times more than in any other language [1].

- It is widely accepted that the most skilled programmers in the IT market are the C++ ones. However, CGI, ISAPI and other frameworks where C++ applies, do not provide the web developer with facilities for efficient application development. As a result, until now, web development could not take advantage of the best programmers [1].

- The way in which a C++ program is get complied is a"1toN" method and therefore there is no way to reproduce the source code from the assembly compiled instructions [14]. This makes you sure of your important data you keep them inside the source code, such as encryption keys or the way you create them and the secret formulas you want to keep hidden from anyone specially hackers. In some web programming languages, the pure source code must be available on the server like PHP. Unfortunately in other cases that you upload compiled codes, it is possible to simply decompile the binary code to the same source code as you've written, because their compilation method is one to one or the target is an intermediate language. For example by using JDecompiler software that is available for free on the web you can extract the exact java source code from the compiled byte-code, or via Salamander decompiler it is possible to convert executable

(.EXE and .DLL) files from intermediate language (IL, CIL, MSIL) binary formats to the high level code that applies to Microsoft .Net languages (such as C#.NET) [5].

- C++ supports all of the 5 programming models such as object oriented [4] so it should be very convenient.
- Unlike other Microsoft .NET languages, C++ is cross-platform and as a result you will see that wing+ can be used in both Linux and Win32 platforms.

3.2. Wing+ Module Categories

There are two kinds of modules available inside the Wing+ API called **core modules** and the **application modules**. Core modules are the ones which can be shared for many instances of applications and they have to control the basic tasks of a HTTP application such as request dispatching, handling errors, writing logs, managing user sessions and so on. These modules are predefined and should be implemented again only if the programmer had a better idea of it or needed another kind of implementation. The second type of modules that have been mentioned above are those that are needed to generate the appropriate response for the client, such as the response handler, pipeline filter, HTML components and scoped entities. By reading the application descriptor XML file, the server will load the defined modules with their initialization parameters that you assigned them in the descriptor, and then the application starts to do its job. Descriptor file provides static load of modules, but it’s not the only choice that you have. In many cases you may need to attach some modules to your application dynamically while it’s running. This will be achieved through the loaded core modules interfaces and calling their functions.

3.3. Core Modules

There are six types of core modules within the Wing+ API. However each one has its own job to do but they have a close collaboration with each other so that the application starts up normally and get’s to work. The below chart depicts digest process of a Wing+ application’s startup as a result of core modules cooperation.

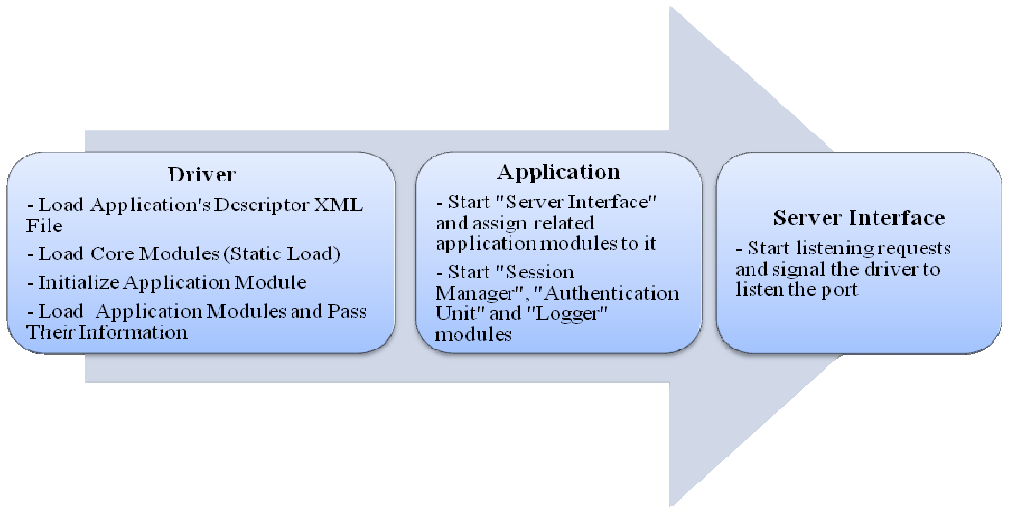
Driver
- Load Application's Descriptor XML File
- Load Core Modules (Static Load)
- Initialize Application Module
- Load Application Modules and Pass Their Information
Application
- Start "Server Interface" and assign related application modules to it
- Start "Session Manager", "Authentication Unit" and "Logger" modules
Server Interface
- Start listening requests and signal the driver to listen the port

Figure 1. Core modules functions order on startup

3.3.1 Driver

The Driver is the start point of an application and can be either a stand-alone container server, embedded server or a module written for another server application such as Apache or IIS. The Implementation of HTTP request and response objects also must be done in this module. The descriptor XML file of each Wing+ application will tell the driver of other modules that should be loaded. Therefore every module that you want to implement yourself can be attached to the application by a simple definition done inside the descriptor. Another reason for attending this part to the API as an implementable module was to make any kind of request serving available. If you do not trust the Wing+ embedded server driver, you can change your driver to the Apache server module and benefit from its great thread-pooling and other special facilities. This depends on the Wing+ driver package you install on your host computer. However the embedded server that is designed for Wing+ is an experimental one, but the load impact test results that compares the Apache Tomcat web server with Wing+ embedded server shows that it is very reliable. The test specifications are:

Testing Tool: Jakarta JMeter 2.4 (Apache foundation product)
Number of parallel users: 250
Number of WPP/JSP pages: 10
Number of requests for each page: 180
Total requests: 450,000

Table 1. Wing+ embedded server versus Apache Tomcat server

	No. of Generated Responses	**Median Response Time (ms)**	**Average Response Time (ms)**	**Minimum Response Time (ms)**	**Maximum Response Time (ms)**	**Std. Dev.**	**Elapsed Time (s)**	**Memory Usage**
Wing+ server	450,000 *– 0.0% error*	1	9	0	240	36.5	116	468 KB
Java - server	450,000 *– 51% error*	1	12	0	3049	105.5	101	668.2 MB

Response time and throughput for both of them is somehow the same, but memory usage of Wing+ embedded-server is significantly more efficient than the Apache Tomcat Server.

3.3.2 WP-Application

This part enables the programmer to share data among all of modules in a very simple approach. A pointer of this module will be sent for all other modules in order to make global data (Application Scope) available all over the program. Dynamic module loading can be done through this module too. Programmer is allowed to implement this module again in order to change the way of data sharing throughout the application.

3.3.3 Server Interface

The process of response generation and all of its related issues occurs in this part. Handling error pages, static and dynamic resource rendering, pipe line filtering and resource protection are samples of server interface tasks which should be implemented correctly. Some of the application modules such as response handlers that are related to the mentioned tasks should be registered by this module.

3.3.4 Session Manager

Sessions are very important entities in a multi-agent client/server system. There are many security and performance considerations that result into several differences in session manager implementation. In order to support any kind of management, this module was appended to the API. Therefore if you need to generate and maintain you application's sessions in a different way, you should implement one yourself and identify it by the descriptor XML file inside your application. With so doing, Wing+ will use the session manager you've implemented yourself instead of the default one.

The default session manager available in Wing+ is an adaptation of Luke Morphey's article published by SANS institute [6] and is designed for single server hosts (not for clusters).

3.3.5 Authentication Unit

Resource protection and membership is another important demand that a multi-agent system needs to consider. The authentication unit has to store data about the protected resources, roles and user profiles. Afterwards it should send a signal for the server interface on each request that indicates whether this resource is available for the current user or not. The implementation of this part may be varying depending on the many storage methods which are available for this specific type of information. For example you can store these data on a database or a simple XML file.

3.3.6 Logger

This module can be called from everywhere inside the application in order to write events down. You can implement this part to classify different messages and store them on a simple file or on the database.

3.4 Application Modules

The modules which were previously discussed were the essential parts of a web application, and at least one instance of their implementation must be available so that the program can starts up normally. Unlike the core modules, existence of all the application modules is optional, but they help the program to respond to clients in any way the developer chooses. Every application module is a base class with some virtual methods to override them. All of them have a common method named "init" that will receive a pointer of the WP-Application core module at the program's startup and should store it somewhere if it needs to access the global shared data later.

3.4.1 Response Handler

Maybe the most important application module is the response handler. There's another virtual function in this module that must be override named "doResponse". This will be called on every time a request arrives from the server interface module and should write an appropriate HTML response according to the client's request via the Request and Response objects given to it. The

Request and Response objects are actually those defined by the driver module. All dynamic resources in a Wing+ application are response handlers.

3.4.2 Filter

If some security measures or other kinds of checkups are required before the request reaches the static or dynamic resources, a filter must be defined in order to handle the checkup. By overriding the "doFilter" function of this module it is possible to check the request and make a decision to let it continue its way through the pipeline or to ban it. It is possible to chain the filters together by assigning them a priority number. By doing this, the client's request will reach the filter with the least priority index and afterwards if the current filter authorizes the request; it will reach the next filter and so on until being banned or reaching the specified resource.

3.4.3 Scoped Entity

There are three types of scoped entities called request, session and application entities. These names specify the storage scope needed. These are the business logic entities that every software engineer uses. If you are designing your application based on a layered architecture, the scoped entities play the role of the classes in the Business Logic Layer which interacts with the User Interface and Data Access. This is where the dynamic module loading facility of Wing+ may prove to be very useful.

3.4.4 HTML Component

If you use response handlers or .WPP files (that will be discussed later) to generate the response, you may need some widgets or HTML tags to make the procedure of designing more convenient. In order to do so, you can design the components you need and identify them to the Wing+ linker.

3.4.5 Session Listener

Implementing a session listener module, enables you to be informed when a session is created and assigned to a user, when a session is going to be destroyed or when an attribute changes in a session.

4. WING+ PAGES

The main concept which brought me to think about Wing+ pages (.WPP) were the events that occurred inside the Java [7] .The relationship between Wing+ pages and response handlers is exactly the same as the relationship that is between Java Servlets and .JSP files. Each .WPP file is converted into a response handler module at the program's startup and then all of them will be compiled (using GNU compilers, g++ for Linux and MinGW for Win32) and linked in a single module named default application module. After the first execution, you can remove all the .WPP files from your host in order to keep them safe from hackers or source code thefts. This is exactly what you do with other module source codes, since .WPP files are modules as well. The .WPP compiler is an embedded part of Wing+ main library and if you want to implement the driver or application module yourself, there is no need to write a compiler for this reason. WPP syntax is a mixture of HTML, C++ codes, Wing+ Components and directives. The directives do some pre-compile tasks such as making components and entities available throughout the WPP page, binding variables and methods defined inside a particular entity to the components and so on.

5. WING+ AND RELATED WORKS ASSESSMENT

There are two important factors that illustrate how appropriate Wing+ is for professional tasks. In order to prove Wing+ potentials, some benchmark algorithms are chosen and executed via Wing+ and some of other well-known programming languages that will help you to judge for yourself. Specifications of the computer in which the tests were executed on are as followed: A quad-core 2.4Ghz Intel® Q6600® machine with 4GB of RAM and 250GB SATA II disk drive and Ubuntu Linux operating system.

5.1. Runtime Evaluation

The list below illustrates the execution runtime of each algorithm per each programming language in the form of seconds. Please note that the best result is bolded in each row. As you can see the runtime of Wing+ is significantly better than PHP and Ruby and slightly faster than Java (with –server option) and C#.

Table 2. Runtime results for Wing+ and other Web development platforms

	Wing+	**C# Mono**	**PHP**	**Java 6 –server**	**Ruby 1.9**
regex-dna	**6.77**	83.58	34.65	23.61	28.42
k-nucleotide[7]	**12.35**	100.2	460.39	32.22	613.36
binary-trees	**12.84**	46.38	993.99	13.31	268.23
Mandlebrot[8]	**23.9**	65.62	1892.58	28.36	5166.19
reverse complement	**1.16**	3.09	6.11	3.01	8.71
Pidgits[9]	**2.74**	5.31	5.93	5.47	18.96
fannkuch-redux[10]	**61.85**	106.41	4319.47	67.72	5313.47
n-body[11]	**21.05**	34.81	906.43	23.03	1695.78
spectral norm[12]	**9.88**	15.79	556.32	15.98	746.56

5.2. Memory Usage Evaluation

Note that the numbers are given in kilobytes.

Table 3. Memory usage results for Wing+ and other Web development platforms

	Wing+	**C# Mono**	**PHP**	**Java 6 -server**	**Ruby 1.9**
regex-dna	**176636**	363944	219236	555968	264436
k-nucleotide	**132656**	534828	247132	410760	157472
binary-trees	**148412**	281060	1248044	497544	196788
Mandlebrot	32040	38632	2960	65908	**2480**
reverse complement	245376	**165104**	443632	293392	173740
Pidgits	**1668**	5608	3892	17188	10400

fannkuch-redux	**944**	3968	2528	13084	2480
n-body	**820**	4940	2560	13028	2748
spectral norm	**888**	4320	14376	13572	12172

5.3. Deduction

The following chart indicates the average runtime and memory usage for each programming language that take part in the two previous tests. There are two columns for every item, in which the first column represents the runtime and the second one shows the memory usage in scale of megabytes.

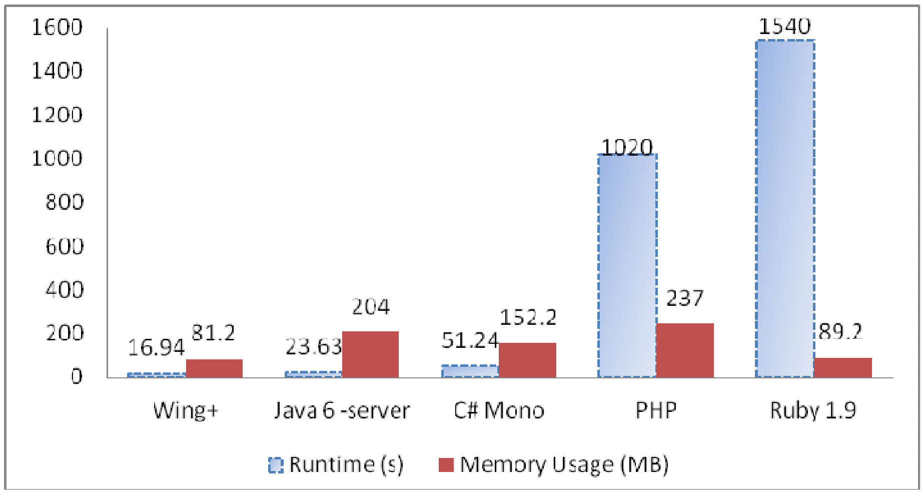

Figure 2. Average runtime and memory

5.4. Related and Similar Works

Wing+ is not the only web development framework that is based on C++ programming language. There are two other frameworks named CSP [1] and Wt [2] that would be described in this section. However there are some basic differences between Wing+ and them, but the performance and memory usage is somehow the same.

Table 4. Wing+, CSP and Wt features

	Wing+	**CSP**	**Wt (Witty)**
Blend Markup & Script Based Design	YES .wpp files	YES .csp files	NO
Class (Widget) Based Design	YES	NO	YES
User Defined Components	YES	NO	YES
Pipeline Filtering	YES	NO	NO
Operating System	Win32 – Linux	Win32	Win32 - Linux
Custom& User Defined Server	YES	NO	NO
Custom Session Manager	YES	YES	NO
Custom Application Manager	YES	YES	NO
Custom Authentication Manager	YES	NO	NO
Automatic Scoped Entity	YES	NO	NO

Management			
Rich HTML Components	NO	NO	YES
Dynamic Module Load	YES	NO	NO
Fully Source Code Independent Run	YES	NO	YES

However Wing+ is newer than the afore mentioned frameworks and so its components and predefined tools will probably need some expanding, but it has the basic ones and enables programmers to design everything that they will need. As I mentioned in the previous sections, everything in Wing+ is defined as an implementable module in order to be fully customizable, so there is no need to worry about the predefined facilities. For the time being Wing+ is in the experimental level of development. In the next releases the number and quality of Wing+ predefined components will be increased in order to satisfy its users.

6. CONCLUSIONS

As it's mentioned before, there are some issues and problems that a Web developer team would encounter if they tend to implement a Web-based system that do heavy tasks in point of the server side processes or that is an enterprise software in which its source code may be valuable for many other people. In most of Web programming languages it's so easy to decompile binary codes into the desired source code because of the nature of those languages [14]. Therefore high performance rate at runtime and source code safety will become subjects to think about. On the other hand if tasks associated with an application consume system resources a lot, so the overhead of the Web-Application itself must be the least to decrease the resource usage and consequently to lower down the hardware costs. So the other problem is efficient resource usage. Another subject is how much you are able to customize the essential parts of your application such as Session Manager or Server and so on. Wing+, the introduced new framework is designed to resolve this type of subjects. It is based on the power of C++ programming language and a flexible API that is fully customizable. The use of C++ helps developers to ensure of their source code safety because of its compilation method [14] and take advantage of the other facilities that brought with C++ such as a great runtime performance and numerous libraries compatible with C++. Some benchmark algorithms have been tested and executed on Wing+ and some other Web development programming languages to prove the advantage of Wing+ over the other Web development tools in point of runtime performance and efficient resource usage. Another benefit that comes with Wing+ is its modular architecture nature that enables users to customize their applications easily in every level they need without knowing anything about the default implementation and source code of the Wing+.

REFERENCES

[1] Micronova company's official web site, www.micronova.com/CSP.html

[2] Witty official web site: www.webtoolkit.eu/

[3] Rohas, Nagpal, (Jun 2010) "Source Code Theft &the Law", Club Hack Magazine

[4] Deitel&Deitel, (2005) C++ how to program, 5th edition

[5] Salamander decompiler web site, www.remotesoft.com/salamander

[6] Luke, Murphey, (2005) “Secure Session Management: Preventing Security Voids in Web Applications”, SANS Institute

[7] Simon, Brown & Sam, Dalton, (2005) “The Expert’s Voice in Java – Pro JSP 2”, 4th edition, Apress publications, Chapter 3

[8] Vicente, Arnau& Ignacio, Marin, "A Fast Algorithm for the Exhaustive Analysis of 12-Nucleotide-Long DNASequences. Applications to Human Genomics."

[9] Eric, W. Weisstein, "Mandelbrot Set." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/

[10] Eric, W. Weisstein, "Pi Digits." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/PiDigits.html

[11] Robert, Sedgewlck, (2008) "Permutation Generation Methods", pp150-151

[12] Diacu, F., (1996) "The Mathematical Intelligencer", the solution of the n-body Problem, pp66 - 70

[13] Eric, W. Weisstein, (2002) "Hundred-Dollar, Hundred-Digit Challenge Problems", challenge #3 Spectral Norm

[14] Short, Christopher, (2011) “Source Code Revelation Vulnerabilities” 6th edition, SANS Institute

Authors

Ahmad Golchin

He was born in 1990 and began computer programming in 1999 using Microsoft QBasic, and now he knows how to program in many other languages such as Assembly, C, C++, Java, Pascal, ASP, PHP, Ruby, VB.net, and C#.net. Database Administration, Digital Image Processing, Neural Networks methods and Internet Computing issues are some of his best proficiencies. Now he is studying computer engineering final year in Sheikhbahaee university of Isfahan.

Ahmad Baraani

Ahmad Baraani-Dastjerdi is an associate professor of computer engineering at the Faculty of Engineering of the University of Isfahan (UI). He got his BS in Statistics and Computing in 1977. He got his MS & PhD degrees in Computer Science from George Washington University in 1979 & University of Wollongong in 1996, respectively. He is Head of the Research Department of the Communication systems and Information Security (CSIS). He co-authored three books in Persian and received an award of "the Best e-Commerce Iranian Journal Paper" (2005). Currently, he is teaching PhD and MS courses of Advance Topics in Database, Data Protection, Advance Databases, and Machining Learning. His research interests lie in Databases, Data security, Information Systems, e-Learning, e-Commerce, Security in e-Commerce, and Security in e-Learning.