Foreword

This document is about nanogear an under-development C++ Web Framework. Ideas in this document came after developing a web application in C++ called Kexi Web Forms.

Introducing Nanogear

It's a matter of fact. There are no modern, open source, easy to use C++ web development frameworks around (with the notable exception of Wt[1] which is not a framework but rather a library, but it has different scopes than the ones presented in this document).

Architectural details

The framework should be designed as a set of loosely-coupled, independent and highly reusable components, so that it can be used outside the Web-side context. It should allow the creation of resource-oriented applications out-of-the-box with ease.

The framework should contain the following essential modules:

- **REST API:** RESTful applications are a simple and smart way to provide web services. The framework should provide a module representing all REST concepts (Resource, Representation, Connectors, etc).
- **REST HTTP Implementation:** Implementation of the REST API above using HTTP as transport protocol
- **Server:** The server which can be used to deploy REST-based applications.

These are some useful tools which can be implemented after the first release:

- **CLI Shell:** A command-line driven interface to initialize a project and create all the needed boilerplate just like the other major web application frameworks

The following modules can be implemented later:

- **DB Library:** A rather simple database abstraction library using Table Gateway Pattern and Row Gateway Pattern (investigate Active Record Pattern[2]).
- **Template Engine**: An easy-to-use template engine with a not-so-complex language supporting conditionals, looping and STL containers traversing.
- **Wt (Web toolkit) adapter:** A set of adapter classes to ease the development of Wt-based applications.
- **Easy deployment:** Solutions to ease the deployment on etherogeneous environments.
- **Integration with Ajax toolkits:** Provide integration with most used Ajax toolkits.

1 http://www.webtoolkit.eu/wt
2 http://en.wikipedia.org/wiki/Active_record_pattern

Requirements:

The framework **MUST**:

- Provide complete, exhaustive API documentation
- Provide a reference manual and examples
- Come with a comprehensive unit testing suite (either using boost's facilities, CppUnit or Google's testing framework)
- Look like an extension of the STL (no Java style camelCase function names, etc)
- Provide programs licensed under the GPL 3 and libraries licensed under LGPL 3

The framework **SHOULD**:

- Depend on Boost: Boost is a comprehensive collection of peer-reviewed library. It contains all the stuff needed to develop the framework, it has a pretty liberal and GPL-Compatible license and it has strict requirements over code style and correctness thus ensuring maximum portability and compiler compatibility. It acts just like an extension of the STL with no extraneous style or coding conventions.

The framework **SHOULD NOT**:

- Depend on Qt: despite being an excellent framework for creating desktop, graphical, applications it is not suitable for this purpose. It uses custom, STL-incompatible, data types everywhere, it's not transparently integrated with the STL and the Qt «» STL conversion functions add some unneeded overhead, thread safety is not guaranteed and it doesn't make use of exceptions. It makes massive use of on-the-fly machine-generated code (an example of this is the Signals/Slot mechanism and the helper code generated by moc) which is an unneeded complication.

The framework **MUST NOT:**

- Provide components licensed under BSD-style licenses, these licenses kill the open source philosophy. The LGPL ensures that the application can be used in commercial environments for free and ensures that every contribution is kept open and gets in the upstream project.
- Be dual licensed: while I'm not against dual-licensing, I personally prefer to release the code for free even for commercial environment. A source of incomes could be selling services or getting investments from other companies.

Rationale

The goal is to bring new and modern web technologies and development paradigms to the C++ world thus eliminating the myth that C++ is not suitable for web development.

A solid framework would help C++ to gain power also in this area and allows programmers to not throw away their tools and practices.

About the author

Lorenzo Villani is a 18-years old CS student living in Florence, Italy.

He began using computers since the age of 5 with an Olivetti 386SX computer with Windows 3.0 installed.

After some time he bought a 2nd-hand computer equipped with an old Pentium 133 MHz processor, 32mb of ram and 4gb of hard disk drive and Windows 95 installed.

Nothing exceptional happened until he got his aunt's computer, a Pentium 2-MMX 450Mhz processor with 64mb of ram, an 8GB Hard drive and an ATI Rage video card equipped with 8mb of Vram. After using Windows 98 SE for some time he decided to try Linux. After buying “Linux Magazine” he put the RedHat Linux 7.2 CDs into the drive. After fighting with Anaconda (the Red Hat Linux system installer) he managed to get it running with KDE.

After some time he switched to SuSE Linux 8.1 where he discovered the C and, some time later, the C++ programming languages.

After that he tried several Linux distributions (including Debian, Gentoo and Slackware) and finally found in Fedora and RHEL (and CentOS) his best friends and went back learning C++.

He's now a Fedora and KDE developer.[3]

Document History

- Friday, 5 September 2008: Fourth revision
- Tuesday, 26 August 2008: Third revision
- Friday, 22 August 2008: Second revision
- Saturday, 17 August 2008: First revision

3 Full LinkedIn profile: http://www.linkedin.com/in/lorenzovillani