

Building a Web-based interface for configuring the Nightly build system at LHC Computing Grid Project

Morten Dam Jørgensen

September 9, 2008

1 Motivation

In relation to the construction of the Large Hadron Collider [LHC], a massive effort has been made to accommodate the need for information processing and systems control. The LHC Computing Grid Project [LCG] is one of these efforts. As the name imply the main focus is to deliver a scalable infrastructure enabling data analysis and storage. One of the activities at LCG is the Applications Area [LCG-AA] responsible for development of various tools for Simulation, Analysis and Data management.

In order to ensure reliable execution on the various platforms supported by the LCG-AA applications, a nightly build system has been implemented headed by the [SPI] workgroup. Until recently the build process has been managed by manually editing a XML document containing the information regarding which projects to be build, where, in what order and when. Because this method requires clear-text editing of the different parameters, user-errors by misconfiguration can occur, to avoid this a certain level of input validation is needed.

The following text describes a system implemented to simplify the editing and validation of the build process configuration.

2 Introduction

The nightly build system [NBS] used by the LCG-AA and LHCb (and indirectly by all other groups) is developed by the SPI group, and is a python machinery employing CMT and QMTest for the build and test of the software, which is then deployed on a variety of architectures, like Scientific Linux, OS X and Windows XP, both in 32 and 64bit versions.

To configure how this process takes place, a XML file defining the versions and which order the projects are being build is used. The content itself is not impossible to edit by hand, but the error rate is sensitive to both syntax errors and wrong naming of project versions.

To redeem these problems as well as to provide the configuration to a wider circle of users, it was decided to try and make a web-based tool to handle the configuration and

validation of this file.

A few goals for the interface were put into place, the most important where the focus on validating the new input against the repositories containing the project versions as well as insuring a level of security around the application to prevent misuse.

2.1 Goals

Various requirements were defined doing the project,

- Configure NBS over the web
- Standard CERN provided Linux, i.e. slc4
- Validation of critical input
- Cross-browser / platform
- Extendable without too much trouble.
- Single-Sign-On authentication
- Well documented (both for users and developers)

All of them have been for filled, along with further features and niceties, as described in the following sections.

3 The Application structure

The website (mis-)uses a Model-Viewer-Controller [MVC] pattern, separating the different logic and interfaces into discreet areas. The misuse comes from the fact that smaller MVC areas combines to form a loose cloud of interacting areas, rather than having one big structure employing just one pattern. To understand this look at figure 1, showing how everything is loosely coupled, but in principle sharing three states: XSL-Templates/Stylesheets for building the interfaces, python modules for logic, and javascript serving as a "meta layer" between the interface and the back-end logic.

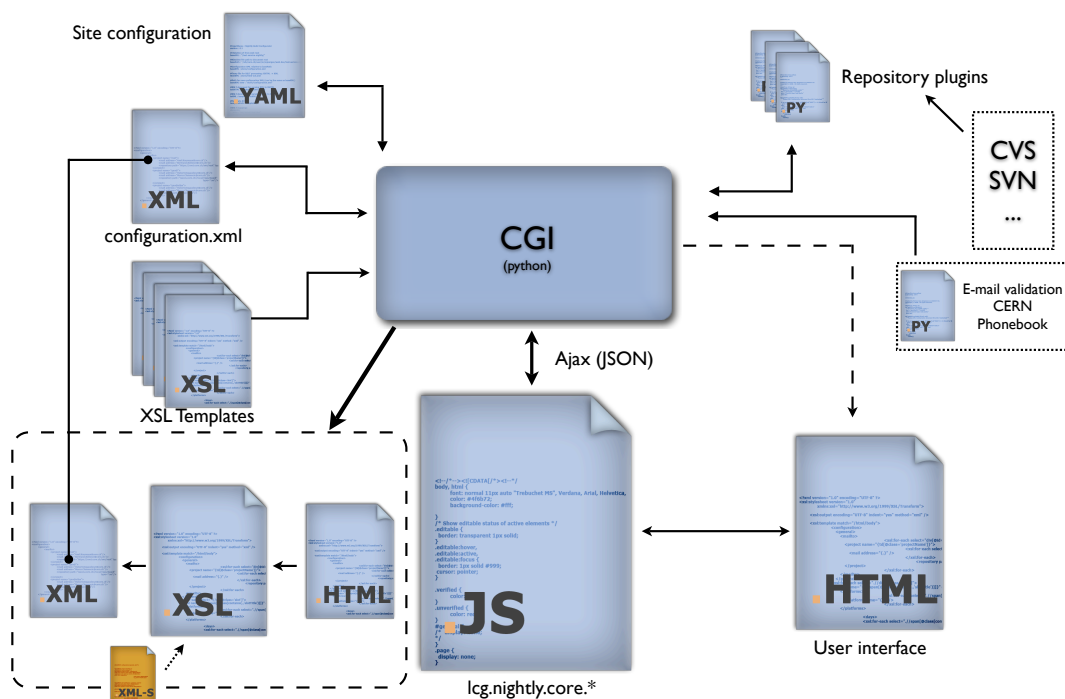


Figure 1: Flow structure

1. Get configuration.xml
2. Parse XML with XSL-T to XHTML
3. Return XHTML to browser
4. When Website DOM is ready ask server for configuration, event-handlers etc.
 - (a) user interaction

- (b) validate CVS/SVN tags
 - (c) validate users, content etc.
5. Submit to server
 6. Parse XHTML to configuration.xml with XSL-T

3.1 The Back-end

The back-end is programmed in Python, and runs as CGI (Common Gateway Interface) scripts on an Apache server. The back-end serves both as the generator of the page, by parsing the various data through XSL-T documents, but also for the validation part, validating project tags against CVS, SVN and other types of repositories, as well as validating user email addresses against the CERN phonebook, and so on.

- Site management forwarded to CGI/Python
- Apache 1.3, Python 2.3, IT-dep standard webserver (self-contained app)
- New elements represented as XSL-T for easy extension
- Shibboleth Single Sign-On Authentication
- Central configuration file for both front- and back-end (YAML).
- Plugin-structure for future repository types
- E-mail validation against CERN accounts
- Logging of changes (commit configuration to CVS)
- Configuration is validated as proper XML before saving (parses XSL-T)

4 The Front-end

The front-end consisting of a webpage and a large amount of javascript, serves as the interface and event-handler. Every operation performed on the configuration is done through events that are interpreted by javascript which then communicates with the back-end structure for validation or saving/reading the configuration.

- jQuery Framework: Cross-browser JS, adds abstraction to browser specific javascripting
- Movable, deletable and editable elements position and edit intuitively
- Interactive feedback on events makes it possible to use trial-and-error without destroying unsaved modifications at other places in the config.

- Ajax Behind the scene server calls Website acts like a regular application
- Novel handling of validation state Regions with open input fields means unvalidated data, and hence no submitting. If all data is locked down it means that it is verified.
- Namespaced Javascript for future integration (lcg.nightly.core.*)
- Fine-grained user control (Project-level, Slot-level and Site-level) LHCb specific. Project contacts can be given partial access to their own builds.

5 Conclusion

Every goal in the above have been for filled, and the project is now in production in the PH/SFT and LHCb group. The hope is that the product can serve as a platform for further build systems in other CERN groups (ATLAS, CMS etc).