Paper Review - OpenLazslo

Vladimir Smatanik

vladimir.smatanik@gmail.com

Richard Dérer

richard.derer@gmail.com

Jacek Marek

pl.jacekmarek@gmail.com

Alexandru Dimitriu

alexdimitriu2005@gmail.com

Department of Information and Computing Sciences,
Utrecht University,

January 30, 2011

Contents

1 Stream-based Compilation **3**
1.1 Rationale 3
1.2 Workaround 3
1.3 Technical Aspects 4

2 Laszlo Foundation Classes **4**
2.1 Overview 4
2.2 Architectural Solution 5

3 Architectural Description Review **6**

1 Stream-based Compilation

1.1 Rationale

At the moment, OpenLaszlo only supports file-based compilation. To increase its usability by allowing stream-based compilation, small changes to the architecture need to be done.

This would enable and simplify these uses:

- Cocoon integration
- Zope integration
- Netkernel integration
- Tag library

1.2 Workaround

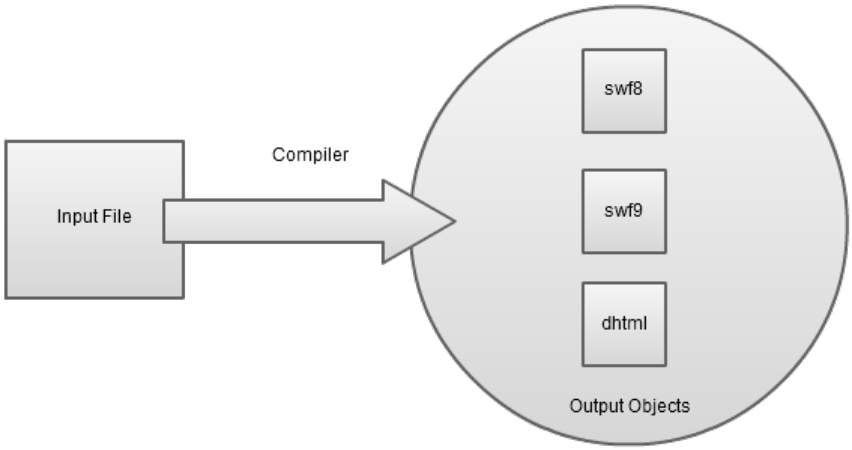

Figure 1: File-Based Compilation

There exists a workaround for providing stream-based compilation, which, however, decreases maintainability: copy the stream to a temporary file and use the regular compiler API to process the file. However, even if the compilation speed remains almost the same, it may be difficult to clean up the file.

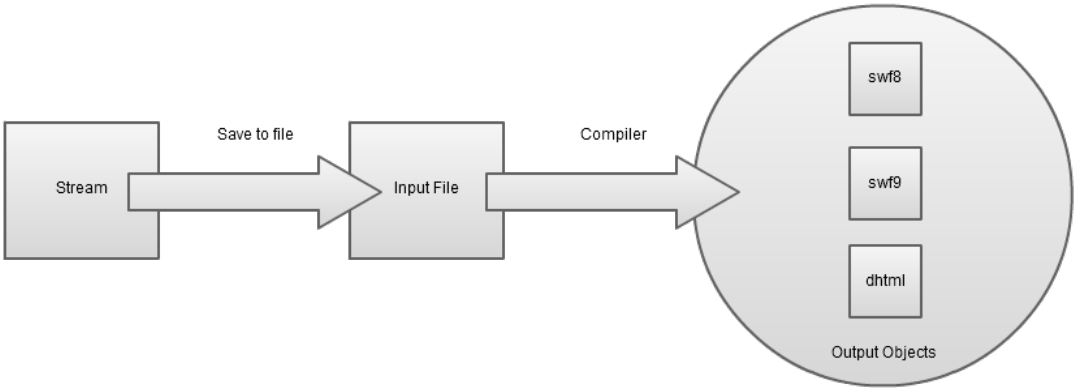

Figure 2: Workaround to support Stream-Based Compilation

1.3 Technical Aspects

To be able to support stream-based compilation, without relying on a workaround that has its drawbacks, some changes are required in the architecture, more exactly in the actual implementation of the compiler package, *org.openlaszlo.compiler*. Some of the required changes are:

- FileResolver class
  - Remove the assertion that protocol is a file - *final String FILE_PROTOCOL = "file"*;
  - Use URI method for resolution
- Parser class
  - Use last URI component instead of *File.getParent()* in *getUserPathname* method
  - Redesign the *read(File file)* method
- Many classes - replace *File* class with *org.xml.sax.InputSource*

In addition to these, changes at the Cache level have to be done to support streams as well.

2 Laszlo Foundation Classes

2.1 Overview

LFC stands for Laszlo Foundation Classes, which is a set of binary files that are included in every OpenLaszlo application. Variety of compilation options are provided to the user so that these can be built manually for a user-specific environment (such as for DHTML, SWF9 or SWF10 runtime). This set of libraries, as those may be called, will then be included in every OpenLaszlo application that is being developed.

The current system architecture does not allow to avoid using OpenLaszlo LFC binary files. This obviously provides the user with a set of stable and always available functionalities that were designed to be small and robust, however, this is not always the case. According to article by Kunal Kumar [2], a simple Hello World application might require 72 kilobytes that are sent through the network. This is acceptable for variety of systems, but might be a significant drawback in some environments, such as on mobile devices.

Currently a great majority of OpenLaszlo applications use Flash SWF files. DHTML added in version 4.x gives an alternative, but since it is not a W3C standard, SWF files are still preferred by most users. Therefore, a solution that might sound reasonable would be to reuse a subset of Flash and DHTML functionalities that are currently available on mobile devices with Flash/DHTML support and to eliminate or at least reduce the overhead of network traffic for such applications.

2.2 Architectural Solution

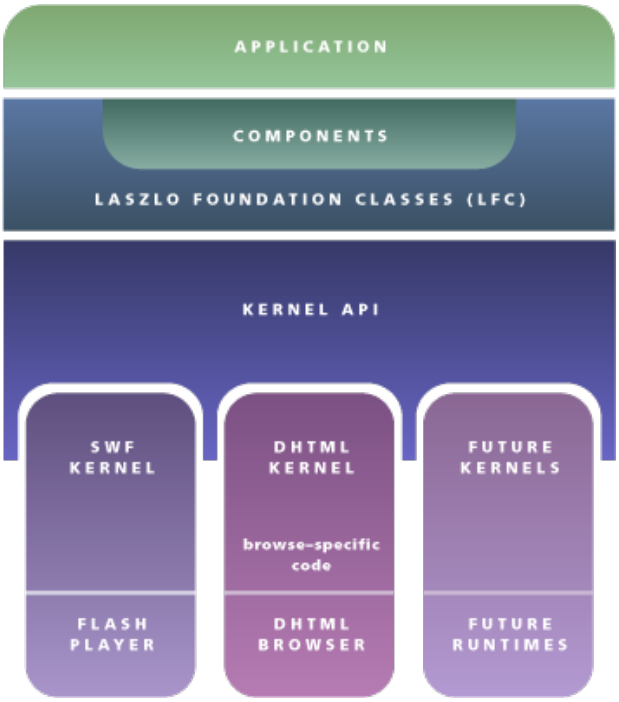

Figure 3: Layer hierarchy of OpenLaszlo system

Laszlo Systems decided to introduce Laszlo Foundation Classes as a middle communication layer between the user defined application and low level kernel runtimes (e.g., Flash Player

and DHTML browser), as can be seen in Figure 3. Removing or even minimizing the size of LFC layer would change the architecture drastically. It would lead to direct dependence between adjacent layers (application and kernel layers) and since application layer is the one created by the user, it must be well separated from the kernel on which it operates.

3 Architectural Description Review

Architectural Description document provided by group A gave us general overview on how OpenLaszlo works (according to 4+1 model viewpoints), a short description of quality aspects and detailed analysis of one trade-off in particular - the use of two connectivity strategies - proxied and non-proxied retrieval. In comparison section, more details about Laszlo Presentation server version 2.02 and Expanz were provided.

However, none of these parts gave us enough clues, neither while searching for the requirements that could be changed, nor while trying to find more information about their influences on the system. The first requirement change is the support of stream-based compilation. In the document, no information about file-based compilation, which is actually supported by OpenLaszlo, were found. In viewpoints of 4+1 model, only interface compiler and its functionality were described. The second requirement change is the possibility to avoid the use of LFC binary files, in order to make applications smaller. In the document reviewed, both LFC and file sizes were only mentioned in the changes of detailed version.

References

[1] Laszlo Systems Inc. *System Administrator's Guide to Deploying OpenLaszlo Applications*. 2007. URL: `http://webtop.laszlosystems.com/ol-webtop/docs/deployers/deployers.architecture.html`.

[2] Kunal Kumar. *Test Drive: OpenLaszlo*. 2011. URL: `http://www.topcoder.com/tc?module=Static&d1=features&d2=030807`.