Glare UI - Flashing user-interfaces with Smalltalk
glare.crimson.ch

Philipp Bunge Tudor Gîrba Lukas Renggli

Software Composition Group, University of Bern, Switzerland
scg.iam.unibe.ch

Keywords. Adobe Flex, user interface, thin client, browsers

Background and description

Rich internet applications (RIA) provide functionality and user-experience known from traditional desktop applications, but run in a web browser and do not require an installation. Most of todays RIA development approaches force the developer to implement too much logic on the client-side, causing severe portability, maintenance and security issues.

Adobe Flash provides the base technology to build cross-platform applications. Adobe Flex is a Flash framework to compose sophisticated user-interfaces using ready-made widgets. Flex applications run in any Flash enabled web browser. Adobe AIR adds the possibility to run flash applications independent of a web browser and integrate into the host-system like a native application.

Glare UI is a Smalltalk framework to build Adobe Flex applications in Smalltalk. View and business logic are implemented and executed on the server. Client and server communicate using the efficient binary protocol AMF through a bi-directional connection. View state is modeled completely on the server. At any point the client can request the list of changes made to the user-interface or re-request the entire user interface. The server subscribes to particular client events and automatically executes the registered Smalltalk code when it is notified of the event. The event object is serialized by the client and contains all relevant information about the event. This allows one to make changes to the business model and update the view.

The flashing counter example

To demonstrate the *Glare UI* framework we present the implementation of a counter, an application displaying a number and two buttons that increase respectively de-

crease the number. The Smalltalk code implementing this functionality is shown in Figure 1.

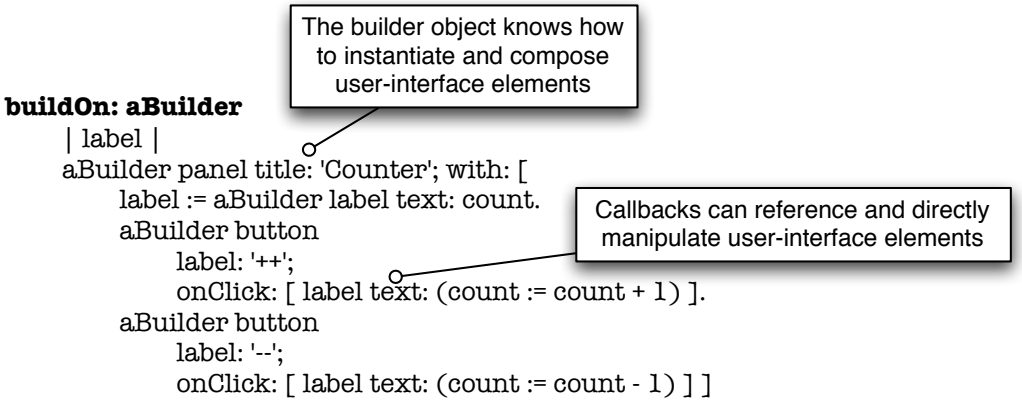

Figure 1: The counter example.

We define a method `buildOn:` that is automatically called with the first request to the application. This method takes a builder object as argument.

In our example, we use the builder to instantiate and to configure a panel that nests a label and two buttons. In this example we want to change the label later on, so we store the object into a temporary variable. Using the `onClick:` event handler, we define actions for each of the two buttons. From within the callback we modify our model and directly manipulate the component tree by changing the contents of the label. The resulting application in the web browser and as a standalone AIR application can be seen in Figure 2.

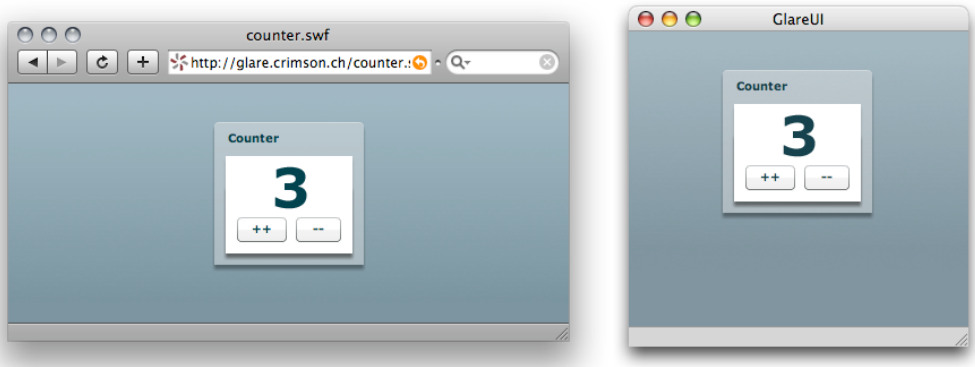

Figure 2: The counter embedded in Safari and as a standalone AIR application.

A more complex scenario could also use the builder object to create, change or remove components. The framework figures out automatically what parts of the user-interface need to be updated, and it updates the view accordingly.

Implementation and Applications

Glare UI makes use of the Seaside [DLR07] request handler and thus has no need for a particular web server. The framework can be integrated with existing Seaside web applications. Glare UI is developed in VisualWorks, but will be ported to an open Smalltalk dialect soon.

One target for Glare UI is to open the door for a new infrastructure for building interactive browsers. We intend to build an engine for scripting browsers that have an interaction flow that is dedicated to specific data, much like we can now use Mondrian to script visualizations [MGL06].

Availability: glare.crimson.ch

The VisualWorks code can be downloaded from the following coordinates:

- Environment: db.iam.unibe.ch:5432_scgStore
- User: storeguest
- Password: storeguest
- Bundles: Glare, GlareUI

The code for the Flex Client can be found at: glare.crimson.ch/browser/flex.

Glare is an Open Source software distributed under the MIT license, that grants unrestricted copy, redistribution, usage and embedding in both free and proprietary software.

References

[DLR07] Stéphane Ducasse, Adrian Lienhard, and Lukas Renggli. Seaside: A flexible environment for building dynamic web applications. *IEEE Software*, 24(5):56–63, 2007.

[MGL06] Michael Meyer, Tudor Gîrba, and Mircea Lungu. Mondrian: An agile visualization framework. In *ACM Symposium on Software Visualization (SoftVis'06)*, pages 135–144, New York, NY, USA, 2006. ACM Press.