SeasideXUL

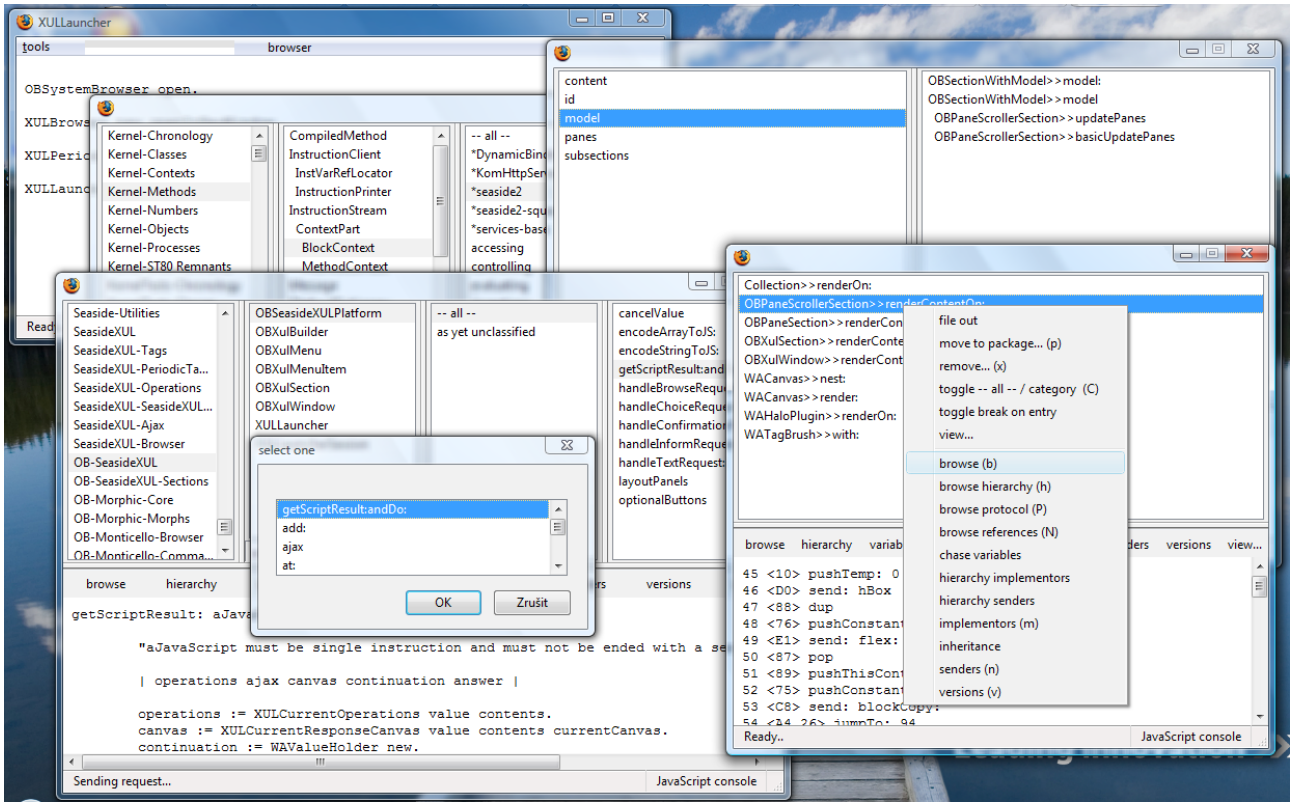
Project mame	SeasideXUL
Authors & Affiliations	Pavel Krivanek - Nidea s.r.o., The Czech Republic
Homepage	http://code.google.com/p/seasidexul/
Smalltalk dialects	Squeak
License	Open-source (MIT)
Keywords	XUL, Seaside, GUI, Ajax, OmniBrowser

Abstract

SeasideXUL is a web framework that enables to create thin-client desktop applications with standard look & feel based on remote XUL in Smalltalk. It is an additional layer on top of Seaside.

Why is SeasideXUL unique

- it brings native look & feel to Smalltalk web applications
- it allows to use remote XUL in the way that is not possible in any other current framework regardless of programming language
- it is the first Seaside application that uses fully Ajax-based communication with components calls and forms submitting
- whole fully capable Smalltalk development environment can be used on remote or local headless images with OmniBrowser
- rich set of example codes



XUL

XUL is an XML user interface markup language developed by the Mozilla project. It operates in Mozilla cross-platform applications such as Firefox and Flock. As its main benefit, XUL provides a simple and portable definition of common widgets. The set of this widgets is wider than in case of HTML and it is closer to the world of common desktop applications so it includes trees, toolbars, group boxes, color pickers, spacers and other widgets that cannot be easily created in HTML. Moreover XUL relies on multiple existing web standards and technologies, including CSS, JavaScript or DOM and it can be combined with next technologies like HTML and SVG. The Gecko layout engine provides the only complete implementation of XUL today.

XUL offers only the GUI rendering. The main application logic must be handled by different layer. It is often the native code connected via special interface (XPCOM). That is suitable mainly for the local standalone applications. Very popular way exerted in Firefox extensions is to write whole application logic in JavaScript.

The project SeasideXUL uses the third way - remotely generated XUL transported via HTTP similarly to HTML pages. This alternative is suitable for the intranet applications but may have several unpleasant limitations.

Goals

The main goal of the SeasideXUL project is to allow to generate the XUL code from Seaside and bring the useful solution for the everlasting question of native Smalltalk user interface with standardlook & feel. Seaside is a web framework based on components and continuations that allows to create web applications in the way that is very similar to creation of common applications with native GUI. So it is natural to use Seaside directly for this purpose and SeasideXUL tries to help with that.

Seaside doesn't use any template system. It assembles simple universal components. The HTML code is generated directly from Smalltalk using a canvas and block closures. So the first task was to create a special canvas class that will generate XUL instead of HTML code. The code that generates user interface then looks like this:

```
xul groupBox flex: 1; with: [
   xul caption label: 'orientation'.
   xul vBox with: [
      xul description value: 'some text'.
      xul checkbox label: 'Left'.
      xul separator flex: 1.
      xml button flex: 1; label: 'Submit'. ]].
```

Seaside is designed for the HTML generation where every new request regenerates whole page and the main part of its know-how attends to hiding this fact to a programmer. This approach is very unfitting for XUL and to regenerate whole user interface every time the user makes some fundamental change is a nonsense. That's why SeasideXUL must use communication via XMLHttpRequests. They are commonly used in Seaside too (using Scriptaculous) but it cannot be used for some basic operations like components calls.

So SeasideXUL had to bring the next layer above Seaside that mediates all communication via Ajax (except the initial request) including components calls, forms submitting and so on. That was not done in any Seaside application before.

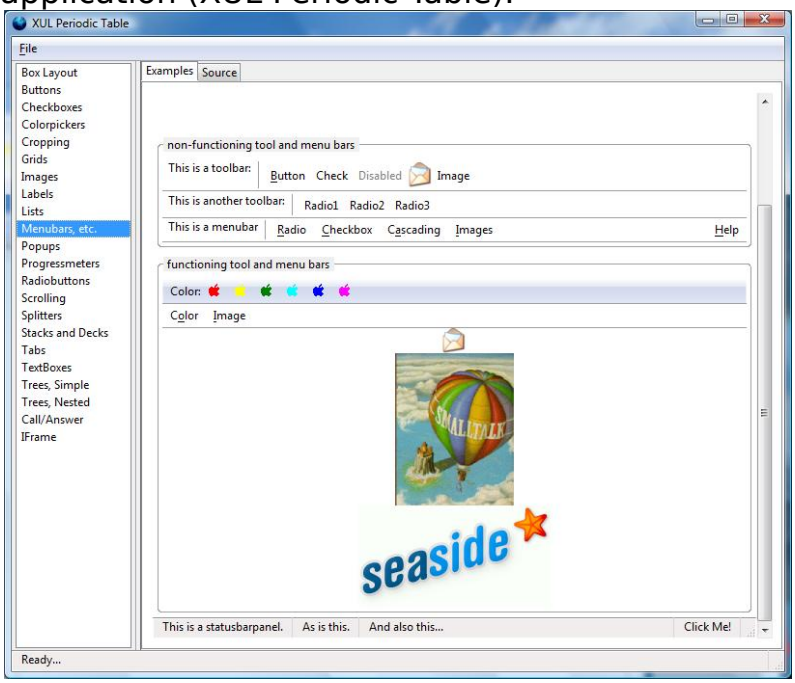
```
xul button
    label: 'login';
    onCommand: (xul ajax callback: [
        | user |
        user := self call: LoginComponent new.
        self call: (UserInfoComponent new user: user) ]).
```

For components calls SeasideXUL uses special marks in the code that limit the component - it's done using simple decorations. Much harder task is to handle control flow because components must stop the computation and wait for the call result. But when the called component returns the control back, the calling component must respond to different client request. This problem is solved using several collaborating continuations, exceptions and special canvas properties.

Forms have similar usage like in case of standard Seaside HTML forms including bindings to accessors but their physical representation and processing is very different and it is all done via special Ajax requests.

```
formId := WAExternalID new.
xul form: formId with: [
        xul textBox on: #surname of: self.
        xul colorPicker type: 'button'; on: #color of: self.
        xul button
                label: 'Submit';
                onCommand: (xul formSubmiter
                        formId: formId;
                callback: [ self refresh. ]) ].
```

All main properties and rendering possibilities are demonstrated in the application named SeasideXUL Periodic Table that is based on standard XUL demonstration application (XUL Periodic Table).



Remote XUL limitations

There are three ways how to run remote XUL applications:

1) open the URL in Firefox. It is the easiest way but it has several security restrictions that limit practical usage including lack of rich text editors.
2) to create the application settings (set of several files and directories) and run it using Mozilla program named XULRunner
3) to create the application settings and run it in Firefox 3 with the -app argument.

The second and third way were not designed for remote XUL applications and they have similar security restrictions like the first one. Mozilla recommends to create a Firefox extension callaborating with remote server for remote XUL applications. However SeasideXUL uses unique workarounds that enables to use unrestricted remote XUL applications even without Firefox extensions. The need of application settings files and specific software on the client side limits the usage for general web applications so the intranet applications and infromation systems are the main deployment areas.

OmniBrowser and SeasideXUL

SeasideXUL includes subproject named OBSeasideXUL. It is the next OmniBrowser platform that uses XUL for OmniBrowser rendering. OmniBrowser is the general framework for generation of Smalltalk development tools. So we can use various Browsers, Monticello, debugger and other tools for remote or local images including headless images and images that have no other user interface like the Squeak KernelImage.

Smalltalk dialects

SeasideXUL is developed on Squeak but it uses well portable code (like Seaside) so the set of supported Smalltalk dialects will include all platforms capable to run Seaside like VisualWorks, Gemstone/S or GNU Smalltalk soon. Especially Gemstone and the Squeak KernelImage may profit from OmniBrowser support and offer whole IDE although they have no own GUI.