

Dynamic Markup Language based User Interfaces for a Browser

Juha Vierinen
Petri Vuorimaa

jvierine@tml.hut.fi

Overview of presentation

- Introduction
- Background
- Design
- Demo / Screenshot
- Conclusions

Introduction

Problems of GUI development for multiple target devices:

- Many competing GUI libraries (HAVi, Sun)
- Lack of decent development environment
- GUI development is iterative \Rightarrow lot of recompiling \Rightarrow unhappy developer

- Interpreted declarative language for GUI description is more natural
- Markup languages and scripts can be used for GUI description
- Custom tags can be used for data flow between core software modules and GUI

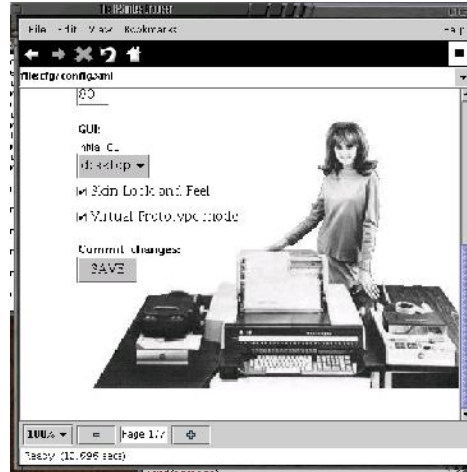
Background

- Web services can be used in many cases as an alternative to traditional applications
- User interface adaptation is time consuming
- Scripting languages are popular for GUI development

- Toolkits, such as Tcl/tk and Macromedia Director have proven useful especially for prototyping
- Mozilla's GUI development language (XUL)
- Browser user interfaces on different devices
- Browser as a toolkit for other applications

X-Smiles Project

- Cross-platform browser
- Purpose is to study different web standards and different client devices
- Single codebase for document rendering, multiple GUIs
- Rendering of mixed documents (different languages)

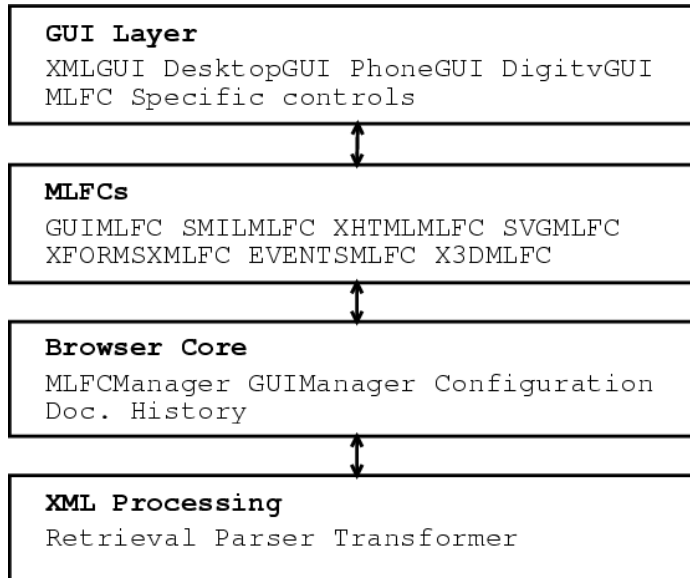


- Available at <http://www.xsmiles.org>

Supported standards

- XForms
- XMLEvents
- ECMAScript
- XHTML
- SMIL
- XFrames
- SVG

X-Smiles Architecture



XForms

- Abstract user interaction definitions
- Styling with e.g., CSS
- Functionality separated from presentation

XMLEvents

- Specification for integrating event listeners and handlers with Document Object Model (DOM) event interfaces
- Interoperable way of associating behaviors with document-level markup

SMIL

- Language for authoring interactive audiovisual presentations
- Easy-to-learn HTML-like language

Design

- Mixing of different markup languages to get needed features
- Additional markup tags for describing browser components (e.g., content area)
- Scripting and XForms were used for the needed user interface controls and behaviour

GUI Elements

Tag	Visual	Description
loadGUI		Load a GUI class
window		Specify window properties
contentArea	x	Area for browser content
mlfcControls	x	MLFC specific controls
currentURI		URI of the current page
statusArea	x	Status text

ECMA functions

Method	Description
<code>back()</code>	Go back
<code>forward()</code>	Go forward
<code>home()</code>	Go home
<code>reload()</code>	Reload page
<code>stop()</code>	Stop loading
<code>reloadGUI()</code>	Reload the GUI
<code>openLocation(uri)</code>	Open a URI
<code>changeStylesheet(title)</code>	Change the stylesheet
<code>getCurrentStylesheet()</code>	Give current stylesheet title
<code>getStylesheetTitles()</code>	List available stylesheets
<code>newWindow()</code>	New browser window
<code>closeWindow()</code>	Close current window
<code>save()</code>	Save current document

XML Events

Event	Description
statusTextChanged	Browser status text changed
browserWorking	Document begin processed
browserResting	Processing finished
currentURICHanged	The document has changed

Implementation

- Implementation using the X-Smiles MLFC (Markup Language Functional Language) framework
- Special GUI, which basically uses browser to render the GUI file
- Exposing internal method calls to scripts



jvierine@tml.hut.fi



juvierine@tml.hut.fi

Conclusions

Upsides:

- Declarative interpreted languages fit well for GUI development
- Easier development for different devices
- More powerful expression for visual oriented UIs

- Hides complexity of underlying components

Downsides:

- One more layer of abstraction
- Overhead of one document begin rendered for GUI
- More difficult to maintain, as files are not compiled, thus interfaces not verified

Considerations

- Scripting security \Rightarrow sandboxing?
- Mixing XML visual content description languages uncharted territory