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

jvierine@tml.hut.fi
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

jvierine@tml.hut.fi
• 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)

jvierine@tml.hut.fi
Available at http://www.xsmiles.org
Supported standards

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

jvierine@tml.hut.fi
X-Smiles Architecture

GUI Layer
XMLGUI DesktopGUI PhoneGUI DigitvGUI MLFC Specific controls

MLFCs
GUIMLFC SMILMLFC XHTMLMLFC SVGMLFC XFORMSXMLFC EVENTSMMLFC X3DMLFC

Browser Core
MLFCManager GUIManager Configuration Doc. History

XML Processing
Retrieval Parser Transformer

jvierine@tml.hut.fi
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

jvierine@tml.hut.fi
SMIL

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

jvierine@tml.hut.fi
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

<table>
<thead>
<tr>
<th>Tag</th>
<th>Visual</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadGUI</td>
<td></td>
<td>Load a GUI class</td>
</tr>
<tr>
<td>window</td>
<td></td>
<td>Specify window properties</td>
</tr>
<tr>
<td>contentArea</td>
<td>x</td>
<td>Area for browser content</td>
</tr>
<tr>
<td>mlfcControls</td>
<td>x</td>
<td>MLFC specific controls</td>
</tr>
<tr>
<td>currentURI</td>
<td></td>
<td>URI of the current page</td>
</tr>
<tr>
<td>statusArea</td>
<td>x</td>
<td>Status text</td>
</tr>
</tbody>
</table>

`jvierine@tml.hut.fi`
ECMA functions

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

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statusTextChanged</td>
<td>Browser status text changed</td>
</tr>
<tr>
<td>browserWorking</td>
<td>Document begin processed</td>
</tr>
<tr>
<td>browserResting</td>
<td>Processing finished</td>
</tr>
<tr>
<td>currentURIChanged</td>
<td>The document has changed</td>
</tr>
</tbody>
</table>

jvierine@tml.hut.fi
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
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

jvierine@tml.hut.fi
Considerations

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