ACTF Meeting (July 31st, 2008)

Kentarou Fukuda, IBM
Michael Squillace, IBM

ACTF Status Update

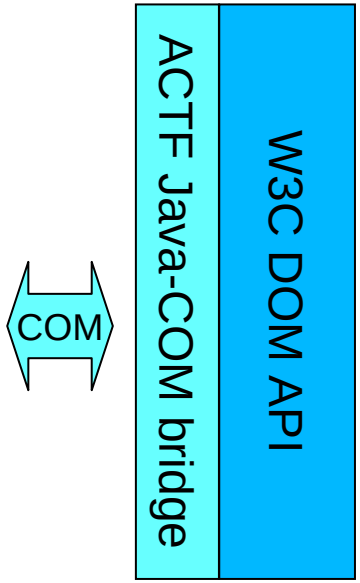
- Tools
  - Released
    - AccProbe 0.4.1
  - Planned
    - AccProbe 0.4.2 in early September
    - Javaco 0.4.0 in early August
    - Webelo 0.2.0 in mid August
    - aDesigner 0.0.2 in early September
    - aiBrowser 0.0.2 in early September
- ACTF Release
  - 3Q, 2008: Build 0.1 release
    - Validation SDK
    - Visualization SDK
    - Alternative Interface SDK

  (Plug-in structure/APIs will be changed in August)

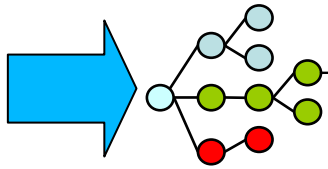
How To Use Live-DOM of Internet Explorer

ACTF IE Browser Component (org.eclipse.actf.model.ui.editors.ie)

IE Editor Component
(Embed IE as an Eclipse Editor)



Runtime DOM in IE



IE-based voice browser

Visualization tool

Inspection/Validation tool

- Provides access to Live-DOM via COM bridge (W3C DOM)
  - Also provides access to Flash Object Model (W3C DOM)
- Provides DOM based on the source HTML (W3C DOM (HTML))
- Provides screen shot of the Web page in BMP format.

How to obtain DOMs from IE Browser Component?

- via org.eclipse.actf.model.ui.IModelService API
  - getDocument(): parses original HTML source and returns DOM
  - getLiveDocument(): returns Live-DOM of IE
  - In addition to this, we can save HTML into file
    - saveOriginalDocument(FileName): saves original HTML source
    - saveDocumentAsHTMLFile(FileName): saves current outerHTML of IE as HTML file
    - HTML files can be parsed by ACTF HTML parser (org.eclipse.actf.model.dom.html.HTMLParser)

Difference between parsed DOM and Live-DOM

- HTML Parser DOM (org.eclipse.actf.model.dom.html)
  - org.w3c.dom.html Interface is implemented
  - can clone nodes

- Live-DOM (org.eclipse.actf.model.dom.dombycom)
  - org.w3c.dom.Node/Element are implemented
  - org.w3c.dom.html Interface is partially implemented
  - can’t clone node yet (planned to implement in future release)
  - can access to currentStyle information of the Live-DOM Element (IElementEx)

```
if(element instanceof IElementEx) {
  IStyle style = ((IElementEx)element).getStyle();
  backgroundColor = (String) style.get(“backgroundColor”);
  ....
}
```

(see CurrentStylesImpl (org.eclipse.actf.model.ui.editors.ie.impl) for more details)

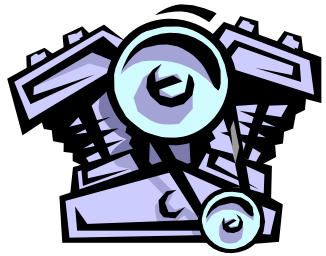
  - two types of attribute value
    - getAttribute(): returns only ‘specified’ attribute value
    - getCurrentAttribute(): returns ‘actual’ attribute value (automatically assigned value by IE)

Web Accessibility Checker implementation of aDesigner

HTML Blind Visualization and Checker

get Model (DOM)

HTML Visualizer



Visualization Engine

IHtml Check Target

HTML Checkers

HTML Visualizer invokes registered checkers automatically

IProblem Item

Visualization result

Check result (IProblemItem)

Report View
(org.eclipse.actf.visualization.ui.report)

ExtensionPoint for checker registration:
org.eclipse.actf.visualization.eval.checkers

class: implementation of IHtmlChecker
infoprovider: implementation of ICheckerInfoProvider

HTML Checker Implementation

- org.eclipse.actf.examples.adesigner.eval.html.Checker
  - Receives IHtmlCheckTarget that contains target HTML information as HtmlEvalUtil
    - getTarget(): returns target DOM selected in preference page
    - getOriginalDom(): returns DOM based on original HTML source
    - getIeDom(): returns Live-DOM snapshot of IE

    - HtmlEvalUtil also provides useful data for accessibility check
      - Array of image elements, anchor elements, table elements, etc.
      - Array of 'href' values
      - etc.

    (Please see org.eclipse.actf.visualization.eval.html.HtmlEvalUtil for more details)

  - Check accessibility by using DOM API and data from HtmlEvalUtil

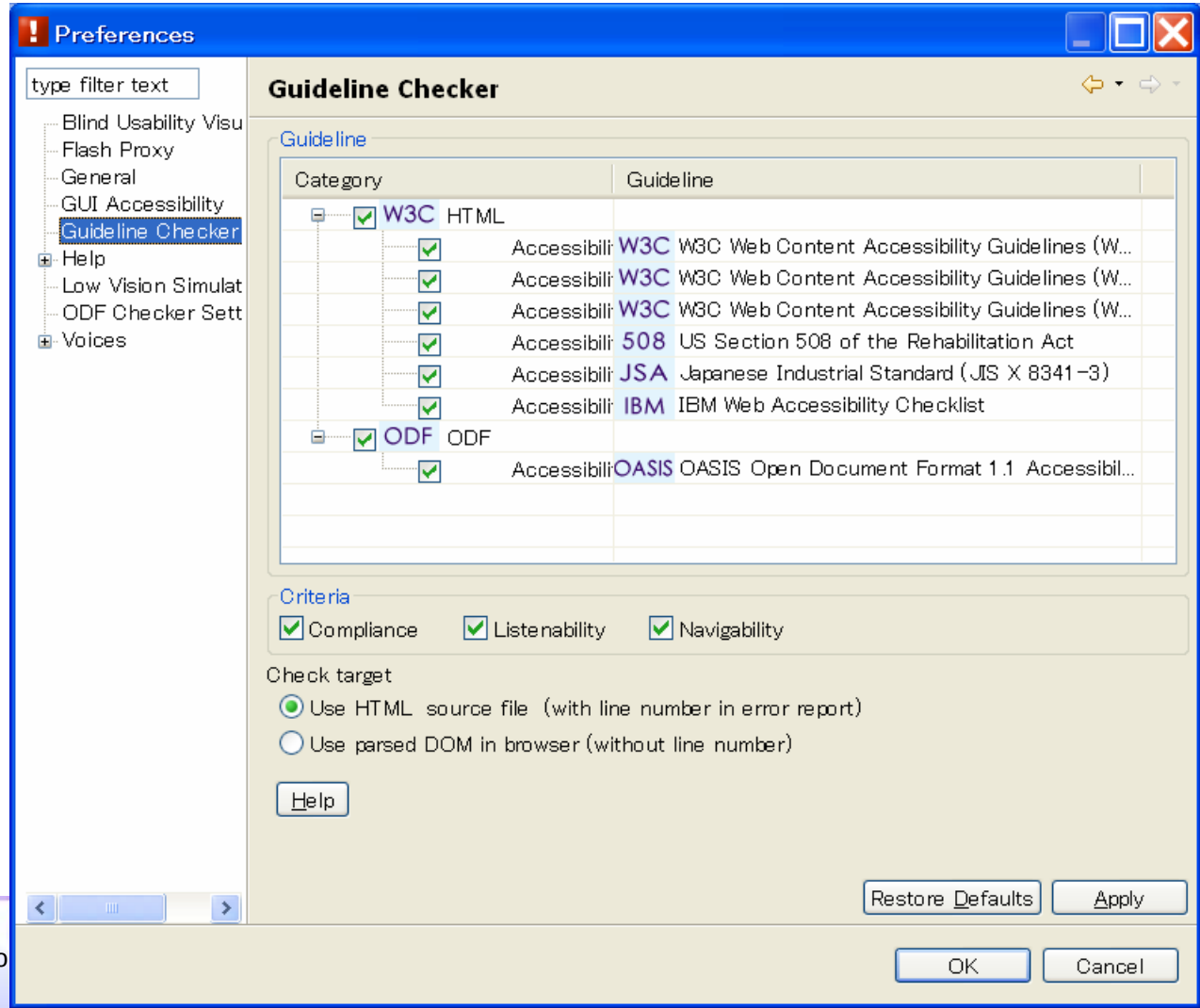
  - Create and return IProblemItem for each accessibility issue

    IProblemItem problem = new ProblemItemImpl("CheckItemID");

(Please see org.eclipse.actf.examples.adesigner.eval.html.internal.CheckEngine)

Guideline Checker Preferences Page

- Users can select
  - Guideline
    - WCAG1.0
    - Section 508
    - JIS
    - etc.
  - Evaluation Criteria
    - Compliance
    - Listenability
    - Navigability
  - Target DOM
    - Source
    - Live-DOM



Configuration Files for Customization (Used in ICheckerInfoProvider)

- Guideline XML
  - Define guideline items
    - Guideline name
    - URL of help page
    - Etc.

- Make validation rule/logic reusable and reduce code/rule clone
- Provide easy customization

- Check Item XML
  - Define mapping between validation items and guideline items
  - Define corresponding criteria for each check item
    - Compliance
    - Listenability
    - Navigability
    - (User original)

- Properties File
  - Define error descriptions in multiple languages

Evaluation Rule Set and Rating Customization

- XML configuration file
  - enable/disable check item
  - modify corresponding guideline items
  - modify corresponding ratings and scores
  - create new guidelines/rating metrics

guideline.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
+ <!--  -->
- <guideline name="WCAG" order="1">
    <category>Accessibility</category>
    <description>W3C Web Content Accessibility Guidelines (WCAG)
      1.0</description>
  - <levels>
    - <level id="P1">
        <category>Accessibility</category>
        <description>W3C Web Content Accessibility Guidelines (WCAG)
          1.0 (P1)</description>
      </level>
    + <level id="P2">
    + <level id="P3">
    </levels>
  + <mimetypes>
  - <items>
    - <gItem id="1.1" level="P1">
        <helpUrl>http://www.w3.org/TR/WAI-WEBCONTENT/wai-
          pageauth.html#tech-text-equivalent</helpUrl>
      </gItem>
    - <gItem id="2.1" level="P1">
        <helpUrl>http://www.w3.org/TR/WAI-WEBCONTENT/wai-
          pageauth.html#tech-color-convey</helpUrl>
      </gItem>
```

checkitem.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
+ <!--  -->
- <checker-config>
  - <checkitem id="C_0.0" type="error">
    - <guideline>
        <gItem name="WCAG" id="1.1" />
        <gItem name="Section508" id="a" />
        <gItem name="JIS" id="5.4(e)" />
        <gItem name="IBMGuideline" id="6" />
      </guideline>
    - <metrics>
        <mItem name="Compliance" score="2" />
        <mItem name="Listenability" score="2" />
      </metrics>
    - <description>
        <desc>Provide ALT attribute for each APPLET</desc>
      </description>
    </checkitem>
  - <checkitem id="C_0.1" type="info">
    - <guideline>
        <gItem name="WCAG" id="1.1" />
        <gItem name="Section508" id="a" />
        <gItem name="JIS" id="5.4(e)" />
        <gItem name="IBMGuideline" id="6" />
      </guideline>
```

Rating Customization Example: Usability Evaluation

- Navigability
  - Reaching time
  - Existence of headings or skip-links for the main content
  - Ratio of accessible links in the page
  - Structure of FORM elements
  - Structure of TABLE elements

- Listenability
  - Appropriateness of ALT attributes
  - Redundant text
  - Space-separated characters

ICheckerInfoProvider Examples

org.eclipse.actf.examples.adesigner.eval.odf.ODFCheckerInfoProvider.java

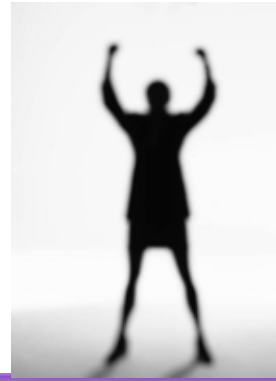
```
public class OdfCheckerInfoProvider implements ICheckerInfoProvider {
    private static final String BUNDLE_NAME =
        "org.eclipse.actf.examples.adesigner.eval.odf.resources.description"; //$NON-NLS-1$

    private static final ResourceBundle RESOURCE_BUNDLE = ResourceBundle.getBundle(BUNDLE_NAME);

    public InputStream[] getCheckItemInputStreams() {
        InputStream is = OdfChecker.class.getResourceAsStream("resources/ODFcheckitem.xml");
        return new InputStream[] { is };
    }

    public InputStream[] getGuidelineInputStreams() {
        InputStream is = OdfChecker.class.getResourceAsStream("resources/ODFGuide.xml");
        return new InputStream[] { is };
    }

    public ResourceBundle getDescriptionRB() {
        return RESOURCE_BUNDLE;
    }

}
```

see also: org.eclipse.actf.examples.adesigner.eval.html.HtmlCheckerInfoProvider.java

Summary

1. Develop IHtmlChecker implementation
2. Write Guideline XML if needed
3. Write Check Item XML and prepare properties file for Error messages
4. Implement ICheckerInfoProvider and refer your XML/properties file
5. Register your checker as an extension

   Extension Point:
   org.eclipse.actf.visualization.eval.checkers



Q and A

Backup

How to obtain ModelService

- from org.eclipse.actf.model.ui.util.ModelServiceUtils
  - getActiveModelService(): returns active ModelService
  - launch(URL)/launch(URL, EditorID): launch (or activate) and return IEditorPart.
    Editors of ACTF implement IModelServiceHolder, then you can obtain ModelService by ((IModelServiceHolder)editor).getModelServise();

- from org.eclipse.actf.mediator.MediatorEvent
  - Implement IACTFReportGenerator or IACTFReportViewer to your views
    (or Implement IMediatorEventListener and register it to Mediator)
  - MediatorEvent contains IModelServiceHolder.
    - Example of MediatorEvent
      - Change of Active ModelService
      - Change of Input of ModelService
      - Change of ACTF Visualization View
      - Change of ACTF Visualization results

- Example
  - PartControlSimpleVisualizer.java in SimpleVisualizer project
  - LowVisionView in org.eclipse.actf.visualization.lowvision project

How to invoke Checkers manually

- Example: BlindVisualizerHtml, BlindVisualizerOdfByHtml

```
IChecker[] checkers = CheckerExtension.getCheckers();

for (IChecker checker : checkers) {
  if (checker instanceof IHtmlChecker) {
    checkResults.addAll(
            ((IHtmlChecker) checker).checkHtml(checkTarget)
     );
  } else if (checker.isTargetFormat(datasource.getCurrentMIMEType())
            && checker.isEnabled()) {
    checkResults.addAll(
            tmpResults.addAll(checker.check(checkTarget))
    );
  }
}
```

How to Try “Simple Visualizer”

1. Create workspace based on aDesigner build instruction
   - http://www.eclipse.org/actf/downloads/tools/aDesigner/build.php
2. Check out 2 plugins from Eclipse CVS
   - Repository path: /cvsroot/technology
   - path of plugins
     - org.eclipse.actf/org.eclipse.actf.examples/fetures/org.eclipse.actf.examples.simplevisualizer-feature
     - org.eclipse.actf/org.eclipse.actf.examples/plugins/org.eclipse.actf.examples.simplevisualizer
   - (see http://www.eclipse.org/actf/contributors.php for more details)
3. Launch Simple Visualizer
   - Visit org.eclipse.actf.examples.simplevisualizer
   - Open simplevisualizer.product
   - Select “Launch an Eclipse application”