pro!vision

.adaptTo() 2011

15.09.2011

Apache Wicket and Sling

About Me

Martin Wehner

Senior Developer at pro!vision GmbH

pro!vision GmbH

Specialized Solution Provider for CMS and Web Projects
International Day CQ Projects since 2003
Customers in Automotive and Telecommunication
Adobe Premium Partner

Located in Berlin & Frankfurt am Main

Overview

- Goals
- Apache Wicket overview
- How the integration work
- Sample applications (DEMO)
- Problems & Solutions
- Future Developments

Wicket Support in Apache Sling

- Goals
  - Enable complex, dynamic form driven applications to run in Sling
    - e.g. mobile carrier self service site
  - Be as transparent as possible for the application
  - Access underlying Sling/JCR stack from Wicket
- Not a Goal
  - RESTful Wicket
  - Substituting Sling/JSP

Apache Wicket Overview

- Open Source, component based
  - Good opportunities for reuse
- Pure Java, no XML
  - Feels like Swing
- Strong separation of logic and markup
- Clean object-oriented architecture
  - All aspects are pluggable
  - Forces you to think clean too
- Goodies like transparent AJAX and back-button support

Integration Classes Overview

- *WicketApplicationManager*
  - Gathers references to available Applications
  - Handles servlet registration (resource type, selectors etc.) and configuration
- *SlingWicketServlet/Filter*
  - Handles requests
  - Sling specific URL handling
  - Injects *BundleClassResolver*
- *BundleClassResolver*
  - OSGi-aware class and resource loading
- *SlingMultipartServletWebRequest*
  - Adapts Sling upload handling for Wicket

URL Handling

- Suffix component of the URL is extracted in *SlingWicketFilter* and handled by Wicket
  - Default URL:

http://localhost:8080/wicket/linkomatic.html/**?wicket:interface=:19::::**

  - Bookmarkable Page URL:

http://localhost:8080/wicket/linkomatic.html/**?wicket:bookmarkablePage=:org.apache.sling.samples.wicket.linkomatic.Page3**

  - Mounted Page URL:

http://localhost:8080/wicket/niceurl.html/**path/to/page2**

Demo

ClassLoader Issues

- Problem:
  - Wicket *DefaultClassResolver* uses thread context class loader:

    Thread.*currentThread().getContextClassLoader()*

- Solution:
  - Custom *BundleClassResolver* that uses bundle class loader:

    bundleContext.getBundle().loadClass()

ClassLoader Issues (Cont.)

- Problem:
  - Wicket uses *ClassResolver* before it allows the application to set it..
- Solution:
  - Circumvent settings access protection by patching the flag at runtime via reflection

Multipart Content Handling (Upload)

- Problem:
  - Wicket and Sling both abstract the upload file handling, so the frameworks fight over request parameters. Sling wins..
- Solution:
  - Custom *SlingMultipartServletWebRequest* that delegates upload handling to Sling *RequestParameters*
  - Unfortunately no pratical upload size restriction possible yet

URL Mapping & Response Rewriting

- Problem:
  - Sling adds mapping support to ServletResponse.encodeUrl(), breaking Wicket path handling
  - CQ5 intercepts the response for post-processing by the Rewriter infrastructure, breaking Wicket default parameter encoding
- Solution:
  - *WicketSlingHttpServletResponseWrapper* that strips all wrappers down to the bare ServletResponse when necessary

Limitations

- Sling and Wicket are frameworks with different concepts and approaches
  - Don't expect a nice 1:1 mapping
  - Wicket is not as hierarchy oriented as Sling/JCR
- Both assume to have full control over request & response
  - Expect some fighting

Limitations (Cont.)

- No way to match Wicket Pages to resource nodes/CQ pages
  - Each resource has to have its own application
    - But you might want to map a whole resource subtree to wicket pages in bigger applications
  - Each Wicket application has its own session
    - So you have to fall back to the servlet session to share data

Mixing Framework Components

- Wicket applications can access the whole Sling stack
- Wicket components can include Sling components
- Not possible to embed Wicket applications & components in Sling components
  - (At least not non-trivial ones)
  - Wicket needs to be able to set HTTP headers and handle contributors to the HTML <head>

Mixing Framework Components (Cont.)

- Possible Solution:
  - Intercept Wicket output and HTTP header
  - Strip everything but the output of the component
  - Handle redirects and includes appropriately (e.g. client side redirect)
- Current Workarounds:
  - Include application in IFrame
  - Include application via AJAX, intercept links and buttons and handle them via XHR
  - Duplicate page frame in Wicket

Future

- Submission to the Apache Sling project
- Annotation
- Configurable Sling upload size restrictions
- Port to Wicket 1.5
- More flexible modes of integration

Q & A

Contact Information

pro!vision GmbH

Karim Khan
kkhan@pro-vision.de
Tel. +49 (69) 8700328-10

Office Berlin
pro!vision GmbH
Wilmersdorfer Str. 50/51
10627 Berlin
Germany
Tel. +49 (30) 818828-50

Office Frankfurt
pro!vision GmbH
Löwengasse 27 A
60385 Frankfurt am Main
Germany
Tel. +49 (69) 8700328-0