adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

Jackrabbit FileVault Validation

Konrad Windszus, Netcentric

About Me

- Passionate about Open Source
  - Apache Sling PMC member
  - Apache Jackrabbit PMC member
  - ACS AEM Commons Committer
- Working at Netcentric

About Jackrabbit FileVault

- Default packaging format for JCR
  - Enhanced serialization compared to JCR 2.0
- Became Open Source in 2013
  - Apache Maven Plugin contributed in 2017
- Lacks contributors

Recent Changes in FileVault

- Performance improvements in Maven Plugin
- Migration to Git
- Documentation improvements

- *and Validation capabilities*

adaptTo()

FileVault Validation - Why

- Packages are complex
  - Uncovered nodes have unexpected default behaviour
  - Serialization Format *JCR (Enhanced) Document View XML* is hard to read
  - Package types introduced further restrictions
  - Installation order sometimes important (ancestor nodes first)

Example 1: ACS AEM Commons

- Invalid filter.xml
  - `<include pattern="/etc/acs-commons/qr-code/jcr:content/clientlib-author" mode="merge"/>`
- Relying on ancestor node's type
- https://github.com/Adobe-Consulting-Services/acs-aem-commons/issues/2044

Example 2: Can you spot the issue?

Repository before import

```
+ /content [sling:Folder]
  + /test  [nt:unstructured]
    - myProperty = “foo”
```

filter.xml

```
<filter root=“/content” mode=“merge” />
```

Package’s jcr_root

```
+ /content/test
  - .content.xml
```

```
<jcr:root xmlns …
  jcr:primaryNode=“nt:unstructured”
    myProperty2=“bar” />
```

Repository after import

```
+ /content [nt:folder]
  + /test
    - myProperty = ”foo”
```

? https://issues.apache.org/jira/browse/JCRVLT-255

adaptTo()

Example 3: Can you spot the issue?

Repository before import

```
+ /content [nt:unstructured]
  + oldChild [nt:unstructured]
    - myProperty: “foo”
```

filter.xml

```
<filter root=“/content” />
```

Package’s jcr_root

```
+ /content
  - .content.xml
```

```
<jcr:root xmlns …
  <newChild jcr:primaryNode=“nt:unstructured”
    myProperty2=“bar” />
  <oldChild>
</jcr:root>
```

Repository after import

```
+ /content [nt:unstructured]
  + /oldChild [nt:unstructured]
    - myProperty: “foo”
  + /newChild [nt:unstructured]
    - myProperty2 = ”bar”
```

?

https://issues.apache.org/jira/browse/JCRVLT-251

adaptTo()

Default Validators 1/2

- jackrabbit-properties
  - Validates properties.xml
- jackrabbit-dependencies
  - Overlapping filter rules check
- jackrabbit-docviewparser
  - Validates Enhanced Docview XML
- jackrabbit-mergelimitations & jackrabbit-emptyelements
  - Catches issues from example 2 & 3

adaptTo()

Default Validators 2/2

- jackrabbit-filter
  - Checks for uncovered nodes
  - Validates filter.xml against schema
- jackrabbit-nodetypes
  - Checks for compliance with primary/mixin node types
  - Enforces a jcr:primaryType property on every node
  - CND for AEM namespaces & nodetypes available at https://git.io/JU8H0
- jackrabbit-packagetype
  - Checks for compliance with package type rules

adaptTo()

FileVault Package Types

- Content
  - Only mutable parts of the repo (/conf, /content, /…)
  - No nested OSGi bundles/configuration and packages
- Application
  - Only immutable parts of the repo (/apps/, …)
  - No nested OSGi bundles/configuration and packages
- Container
  - Only nested OSGi bundles/configurations and packages
- Mixed
  - Legacy

FileVault Package Types: Why?

- Makes your code cleaner
- AEMaaCS relies on the separation of mutable/immutable packages
- Eases conversion to Sling Feature Model
  - (https://github.com/apache/sling-org-apache-sling-feature-cpconverter)
- Eases deployment

adaptTo()

False Positives?

- Report at https://issues.apache.org/jira/projects/JCRVLT
- Disable
  - Per Message Type (via option `severity*`)
  - Per Validator (via option `isDisabled`)
  - Last resort: Skip validation (via Maven parameter `skipValidation`)

adaptTo()

AEM Content Classification Validator

- Content classificiation determines allowed usage of AEM's resource types
  - API compliance for non Java code
- Important for rolling updates with AEMaaCS
- *netcentric-aem-classification* checks usage of resource types against a *predefined* map
  - Maps for AEM 6.5.5 and AEMaaCS available
  - Maven Plugin for generating a map from a JCR repo

AEM Content Classification Validator

- Checks against usage of deprecated resource types
  - Deprecations from property *cq:deprecated*
  - Deprecations from release notes

False Positives - Content Classification

- Raise a ticket with Adobe Support
  - Known FPs available at https://git.io/JU8HT
  - Raise an issue at https://git.io/JU8Hm
- Ignore FP resource type via configuration *whitelistedResourcePathPatterns*

adaptTo()

Differences to OakPal

- Faster
  - No repository started during validation
  - Incremental build support in Eclipse (m2e)
- (Hopefully) Less configuration
- Probably more False Positives (still)
  - Try it out!
  - Report bugs once you find them!

The Future of FileVault and Validators

- Validator for AEMaaCS limitations
- Fixes for import modes != replace
- Validator for overlapping filter rules

- Maybe your validator?!!
  - https://jackrabbit.apache.org/jcr/mailing-lists.html

adaptTo()

References

- https://jackrabbit.apache.org/filevault/validation.html
- https://jackrabbit.apache.org/filevault-package-maven-plugin/validators.html
- https://github.com/Netcentric/aem-nodetypes
- https://github.com/Netcentric/aem-classification