adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

28th – 30th SEPTEMBER 2020

OakPAL in the Cloud

Mark Adamcin, Bounteous

About the speaker

Mark Adamcin

Principal Architect @ Bounteous

Maintainer: OakPAL
Maintainer: Jenkins CRX Content Package Deployer Plugin

About OakPAL

adaptTo()

OakPAL is an OSS project

- Apache 2.0
- Integrated in Cognifide's gradle-aem-plugin
- Used for validation in ACS AEM Commons

adaptTo()

OakPAL is used @Adobe

- Winner of 2019 AEM Rockstar award at Adobe Summit NA
- Used @ Adobe for package validation in Cloud Manager Pipelines
  - Identified ~11000 Sonar violations in customer packages last month
  - 8 documented sonar checks

What does OakPAL do?

What does OakPAL do?

OakPAL scans code packages and reports violations defined by user-provided checklists

```
[INFO] --- oakpal-maven-plugin:2.2.2:verify (default) @ demo.ui.content ---
[INFO] OakPAL Check Reports
[INFO]   respect-existing-editable-template
[ERROR]    +- <MAJOR> expected path missing: /conf/classic-app/settings/wcm/templ
[INFO]   respect-existing-ace-on-us
[ERROR]    +- <MAJOR> expected: principal:classic-app-us-authors ace:type=allow;p
[ERROR] ** Violations were reported at or above severity: MAJOR **
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  16.584 s
[INFO] Finished at: 2020-09-29T05:49:19-07:00
[INFO] ------------------------------------------------------------------------
```

What does OakPAL do?

A checklist defines two things:

1. The initial state of a JCR Repository
2. Acceptance criteria as configured “checks”

adaptTo()

OakPAL Approach to Validation

- Use a concrete JcrPackageManager to install one or more packages into a concrete Oak repository
- Propagate ProgressTracker events to extensible ProgressChecks with access to a fully-functional Jackrabbit JCR Session
- Report violations as MINOR, MAJOR, or SEVERE

OakPAL Approach to Validation

- Convenient, declarative syntax to define initial repository state
- Publish your own checklists and validation plans as Maven artifacts
- Write ProgressChecks in Java, or in a JSR-223 scripting language

A Short OakPAL Demo

oakpal vs. vault-validation

OakPAL vs. vault-validation

- Vault-validation performs incremental, static validation of a package and its contents
  - Validation is performed without the overhead of a running repository
  - Will lead to faster and contextually relevant validation feedback in IDEs with incremental maven execution
  - Installation effects can only be inferred by validators, not directly observed

OakPAL vs. vault-validation

- OakPAL performs installation of assembled zip files into a fresh JCR repository and observes changes
  - OakPAL is better for validating the actual changes that a package installation will make to a repository, like detecting unintended deletion of nodes
  - OakPAL won’t provide incremental feedback as quickly as vault-validation, and it won’t be able to provide direct source code references for violations

OakPAL vs. vault-validation

- Use `filevault-package:validate-*` goals to check correctness and enforce project conventions for package artifact source code
- Use `oakpal-maven-plugin:scan` to enforce evolving content policies established by Sling, AEM, or your own organization

OakPAL and AEM Cloud Service

OakPAL and AEM Cloud Service

- AEMaaCS adds new content package restrictions and conventions
  - Embedded Packages instead of Subpackages
  - First-class support for Repoinit
  - Container, Mutable, and Immutable content types

OakPAL and AEM Cloud Service

- To meet this challenge, OakPAL now supports:
  - Defining initial repository state using repoinit scripts
  - Scanning Embedded Packages for violations, just like Subpackages
  - Applying embedded repoinit OSGi configurations

Q & A